A Framework for Traffic Collision Prediction Using Historical Accident Information and Real-Time Sensor Data: A Case Study for the City of Ottawa

by

Enrique A. Reverón Yanes

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University
Ottawa, Ontario
July 2019

Abstract

According to recent studies, beyond being a major worldwide problem with huge economic impact, traffic collisions are poised to become as well one of the most important leading causes of death. Proactive traffic enforcement and intervention should be based on a thorough analysis on the collision data available to identify leading causes of accidents, the most prone locations as well as to predict the conditions for collision occurrence. This thesis presents a novel framework for collision prediction that takes into consideration historical and real-time factors, such as weather, geospatial information and social event data that can be obtained with existing sensor technology. A prototype is proposed, implemented and evaluated for the city of Ottawa, Canada, to predict: (1) accident frequency (collision vs no-collisions) and (2) accident severity (in terms of fatal, injury and property damage only accidents). The best performance was achieved in both cases using gradient boosted trees.

Acknowledgements

I would like to express my gratitude to my family in Canada, Perla and Leonardo, for their full support and encouragement during these two years that I have spent studying in Canada. It has been a very difficult journey for us, I will be grateful forever for your support and love.

I would like to thank to my family outside Canada; my father Enrique, my sisters Mariela and Miriam, my aunt Maria Julieta, my cousins Maria Julieta and Jose Manuel, my step-daughter Ivana; for their motivation and support.

This achievement is dedicated specially to my mother, Aida; I have a special debt with her, I would have liked it so much that you celebrated this with me in person.

Finally, I would like to thank Dr. Ana-Maria Cretu for her guidance, motivation and encouragement through my study and research.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Appendices

List of Acronyms

AADT: Annual Average Daily Traffic

AdaBoost: Adaptive Boosting

ANFIS: Adaptive Neuro Fuzzy Inference System

AUC: Area Under ROC Curve

AV: Avenue

BBC: Baseball Club

CART: Classification and Regression Trees

CFS: Correlation Feature Selection

CPU: Central Processing Unit

CRISP-DM: Cross-Industry Standard Process for Data Mining

CUDA: Compute Unified Device Architecture

DBSCAN : Density-Based Spatial Clustering of Applications with Noise

DR: Drive

FC: Football Club

FN: False Negatives

FP: False Positives

FPR: False Positive Rate

GB: Gigabyte

GBT: Gradient Boosted Trees

GDP: Gross Domestic Product

GIS: Geographic Information System

GPU: Graphic Processing Unit

HTML: Hypertext Markup Language

ID: Identifier

IDE: Integrated Development Environment

MCC: Matthews Correlation Coefficient

MCMC: Markov Chain Monte Carlo

MLP: Multi-layer Perceptron

MVCR: Motor Vehicle Collision Report Manual

NNs: Neural Networks

NPV: Negative Predictive Value

PDO: Property Damage Only

PPV: Positive Predictive Value – Precision

RAM: Random Access Memory

RF: Random Forest

ROC: Receiver Operating Characteristics Curve

RRI: Road Risk Index

SMOTE: Synthetic Minority Over-sampling Technique

ST: Street

SVM: Support Vector Machines

TN: True Negatives

TP: True Positives

VRAM: Video RAM

1. Introduction

1.1. Research Problem Justification

Traffic collisions represent a major problem at a worldwide scale. According to a recent report of the World Health Organization, each year nearly 1.35 million people die on the world's roads, on average one person every 24 seconds, and cause up to 50 million non-fatal injuries [1]. The same document identifies road accidents as the eight-leading cause of death for all age groups, and astonishingly "*currently the leading cause of death for children and young adults aged 5–29 years, signaling a need for a shift in the current child and adolescent health agenda which, to date, has largely neglected road safety*" [1]. Statistics show also that the vulnerable road users (pedestrians, cyclists and motorcyclists) are involved in more than half of all road traffic deaths [1].

Beyond injury and life loss, the impact of these collisions is economically important: road accidents cost $518 billion globally, costing individual countries up to 3% of their annual GDP [2]. Moreover, it is expected that unless action is taken, road traffic injuries are to become the fifth leading cause of death by 2030 [2].

In Canada, in the last five years (2013-2017), statistics suggest an average of 1800 people die on related collisions annually [3], from which Ontario accounts for 500 during the same period [4]. As well, in Canada, in 2010, transport incidents were the third cause (17%) of injury deaths (after falls and suicides) and the second cause of overall injury costs (17%), with a total estimated cost of $4.3 billion nationwide [5]. In Ontario, transport incidents are the third cause of injury-related deaths, with a total estimated cost to the province of $1.2 billion [5]. The total estimated annual cost of the road crashes for the Canadian society is about $37 billion (2.2% of Canadian GDP) [6].

This justifies the interest of the research community in developing techniques to predict conditioning factors for accident occurrence. As part of the current challenges of smart cities, such techniques are meant to support and enable proactive traffic intervention schemes (such as real-time traffic light and electronic road sign

control, police personnel and radar equipment deployment and preparedness for accident interventions, just to mention a few) that can help to create safer roads.

In this work, we propose a novel collision prediction framework based on a series of historical and real-time factors, including weather, geospatial information, social events and date/time, to create a machine learning model that can predict collisions. The novelty of the research work presented lies in the consideration of a broader spectrum of accidents factors, including vehicle data, roads, events, time/date, weather and driver, and thus proposing a more complete and comprehensive solution with respect to similar work in the literature. A proof-of-concept of the proposed framework was implemented and validated with the support and collaboration of the Transportation Data Collection & Analytics Section of the City of Ottawa, as well as the Business Performance Section of the Ottawa Police Service.

1.2. Thesis Objectives

The fundamental challenge of this research was to generate a general framework for traffic accident collision prediction using machine learning algorithms and GIS visualizations, as well as demonstrate a proof-of-concept of such a framework using public available datasets for the city of Ottawa, Canada. As a result, the following objectives were established as part of this research effort:

1) Assemble and generate a comprehensive dataset of traffic collisions for the city of Ottawa, Canada including a broad series of historical and real-time factors (i.e. weather, geospatial information, social events and date/time).
2) Design a framework for traffic collisions prediction based on historical and real-time factors.
3) Implement a proof-of-concept of the framework using a selection of machine learning algorithms to classify:
    a. accident frequency (i.e. collision/no collisions) and
    b. accident severity (i.e. fatal injury, non-fatal injury and property damage only).

4) Test and validate the framework for the generated dataset; validate if social events (i.e. sport events and statutory days) have an important impact on accident collision rates.
5) Generate a series of map visualizations of results based on a Geographic Information System (GIS) that can be used for proactive traffic intervention schemes.

1.3. Thesis Organization

The thesis is structured as follows: Chapter 2 presents an overview of the background to the problem domain and the relevant literature. Chapter 3 provides a review of the methodology used. Chapter 4 discusses the details of the proposed framework for traffic accident collision prediction using machine learning algorithms. A prototype for the City of Ottawa is presented in Chapters 5 to 7 as follows: the business/data understanding and preparation are described in Chapter 5, the modeling, model comparison and evaluation in Chapter 6 and the experimental results in Chapter 7. Finally, the conclusion and future work are discussed in Chapter 8.

2. Literature Review

Due to its importance, the topic of traffic accident modeling and prediction has gained the interest of many researchers. Different studies are focusing on identifying the most important accident causes and creating methods for accident modeling and prediction. This chapter presents an overview of relevant work in the literature on the most important accident factors (section 2.1), the accident trends (section 2.2) and the different methods for accident modeling and prediction (section 2.3). Finally, section 2.4 provides a summary of the literature review.

2.1. Accident Factors

With the goal of the thesis in mind, one important aspect is to identify a series of accidents factors that can be used as a basis for the development of the accident prediction framework. This section is summarizing relevant literature on this topic.

The *2017 Saskatchewan Traffic Accident Facts* [7] identifies 4 major categories of traffic accident causes as being: (1) human condition, (2) human action, (3) vehicle condition and (4) driving environment factors. Among these categories, vehicle condition seems to have the least impact (between 1.7% and 2%) regardless of the collision type, as shown in Figure 2-1. According to this study, human action has the biggest impact over the fatal (38%, and equal to human condition) and injury collisions (46%). Considering all collision types, the environment condition is the most important factor (41.2%).

The main causes of traffic accidents within each main category are shown in Figure 2-2, while the environmental factors and their relative contribution are detailed in Figure 2-3. As illustrated in the latter, most accidents occur on dry surfaces (42%), in daylight (52%) and in clear weather (46%).

An analysis of collisions according to the type of the roads, namely urban streets, provincial highways, rural roads and other roads, is presented in Figure 2-4.

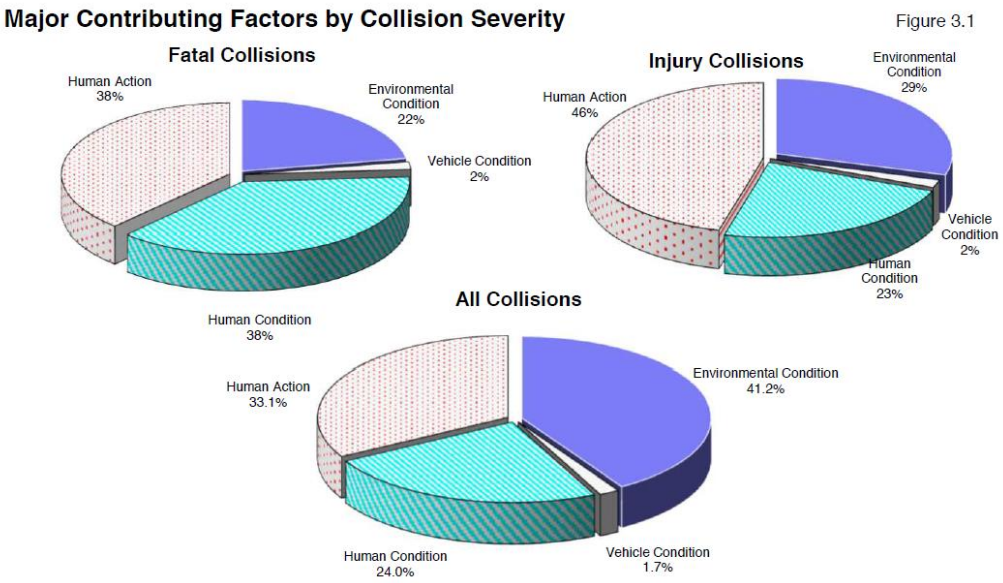


Figure 2-1 Distribution of major accident categories over fatal, injury and all collisions (from [7])

Human Condition
Driver Inattention
Driver Distraction
Driver Inexperience/Confusion
Driving While Impaired
Other Human Conditions
Had Been Drinking
Fell Asleep
Extreme Fatigue
Lost Consciousness/Other Illness
Physical/Medical Disability
Drugs (Prescription or Illegal)
Defective Eyesight/Hearing

Human Action
Fail to Yield
Following Too Closely
Taking Evasive Action
Driving Too Fast for Conditions
Other Human Action
Passing or Improper Lane Usage
Turning Improperly
Traffic Control Device Disregarded
Backing Unsafely
Careless Driving/Stunting
Exceeding Speed Limit
Pedestrian Action Contributed
Driving Wrong Way in One-way Traffic
Fail to Signal

Vehicle Condition
Other Vehicle Condition/Defect
View from Vehicle Obstructed
Defective Brakes
Defective Tires/Tire Blowout
Jackknife/Trailer Swing
Vehicle Overloaded/Improperly Loaded
Load Shifted/Spilled
Defective Suspension/Wheel Failure
Defective Steering
Defective Engine/Power Train/Wiring
Defective Lights
Lights Not On
Defective Exhaust System

Environmental Condition
Animal Action (Wild)
Road Condition (Surface or Structure)
Uninvolved Vehicle
Weather Conditions
View Obstructed/Limited
Other Environmental Condition
Obstruction/Debris on Road
Snow Drift
Sun Glare
Animal Action (Domestic)
Excessive Loose Gravel
Uninvolved Pedestrian
Construction Zone
Soft or Defective Shoulder
Traffic Control Device Not Working
Lane Marking Inadequate

Figure 2-2 Risk factors within each main category of traffic accident causes (from [7])

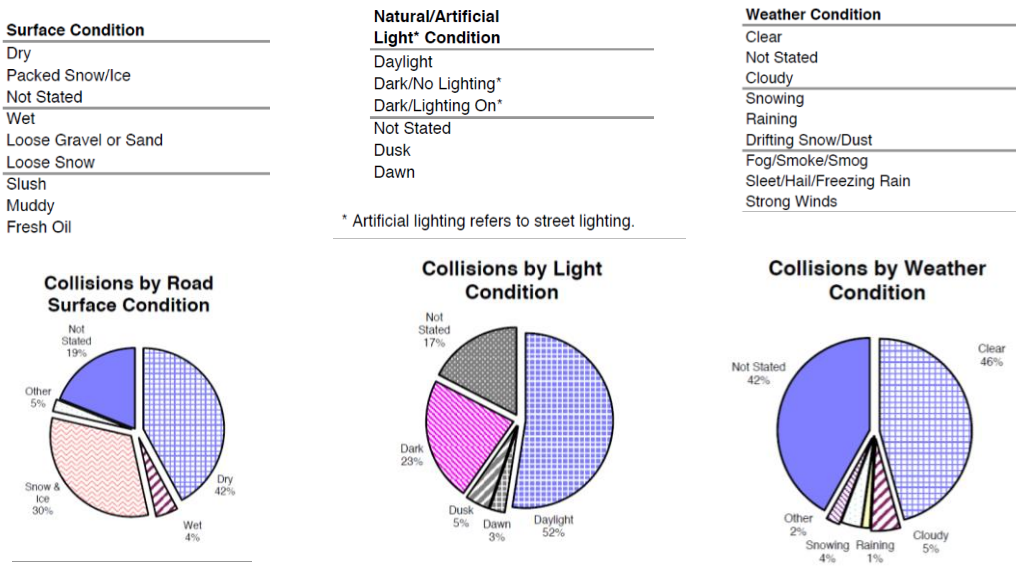


Figure 2-3 Environmental factors for traffic accidents (from [7])

One can notice that the environment conditions seem to have by far the most important impact (over 58% of the total) over provincial highways, rural and other roads. On urban streets, the main factor seems to be human action. If these statistics prove relevant in a general case, an intelligent system should take them into account in order to improve its predictions.

The impact of weather on the accident occurrence is analysed in various research articles, in particular the effects of winter and rainy weather. The authors of [8] focus their study during the winter season using retrospective accident and car probe data. They validate the importance of weather, but conclude that it cannot serve alone as predictor of accidents. Afrin [9] concludes that, during winter months, the most important risks factors are: temperature, snowfall, visibility obstruction, winter traffic, months and timing of precipitation. Yu *et al.* [10] investigated crash injury severity on a mountainous freeway and found that the four most important factors are the steep grade indicator, the speed standard deviation, the temperature and the snow season indicator (i.e. snow/dry). The authors of [11] studied the risk factors on three different weather conditions over a freeway in California, including clear weather, rainy weather, and reduced visibility weather using collected data from weather and vehicle loop detection stations. They found that the speed

difference between upstream and downstream detector stations was an important risk factor in each crash prediction model, particularly in the case of reduced visibility and rainy weather. In clear weather, the speed variance played an important role, while in rainy weather, the rainfall intensity was identified to be an important risk factor; finally, in reduced visibility conditions, the visibility factor contributes significantly.

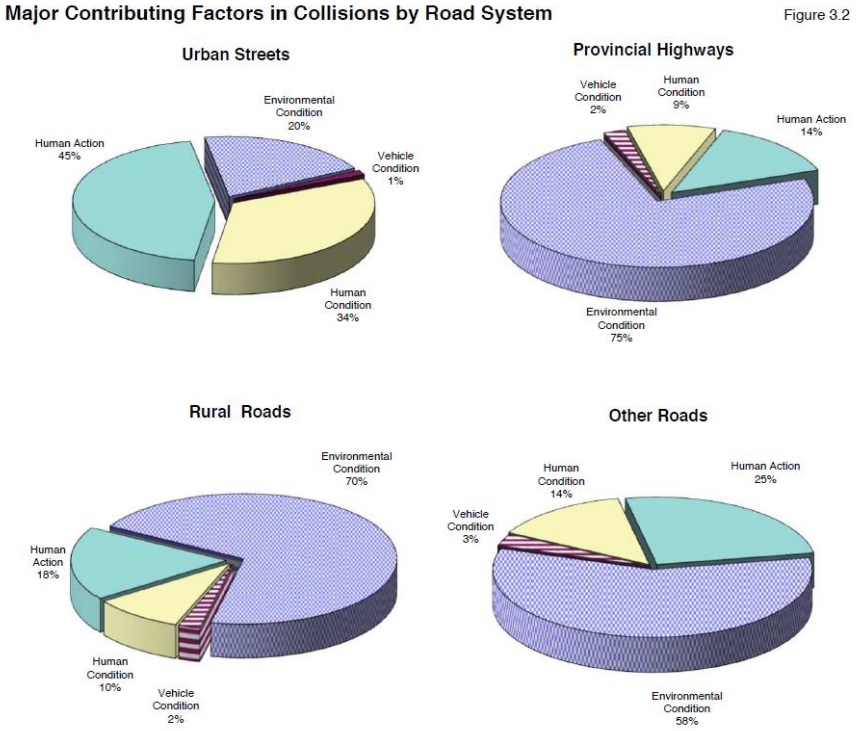


Figure 2-4 Distribution of major accident categories over the road system (from [7])

Figure 2-5 shows another source likely to improve collisions predictions, the driver's gender and age.

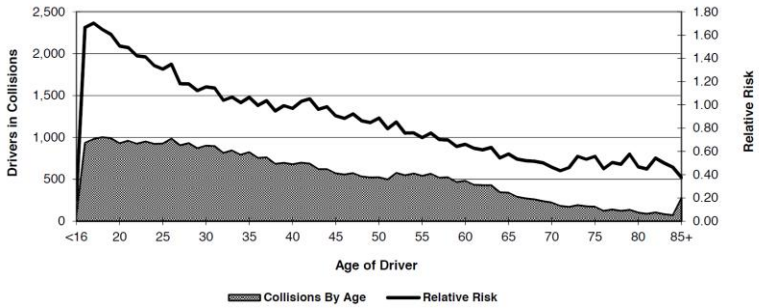


Figure 2-5 Drivers in collisions and relative risk by driver age (extracted from [7])

The analysis of causes of traffic in [12] results as well in 4 major categories: driver factors affecting the traffic safety, road factors, environmental factors and vehicle factors, sensibly similar to [7]. However, it gives more importance to the road factors uncoupling them from the environment conditions (i.e. line of sight to ensure, intersections, surface water, traffic signs, traffic marking) and gives less importance to the human actions (i.e. fail to yield, exceeding speed limit, fail to signal, etc.), as it can be noticed in Figure 2-6.

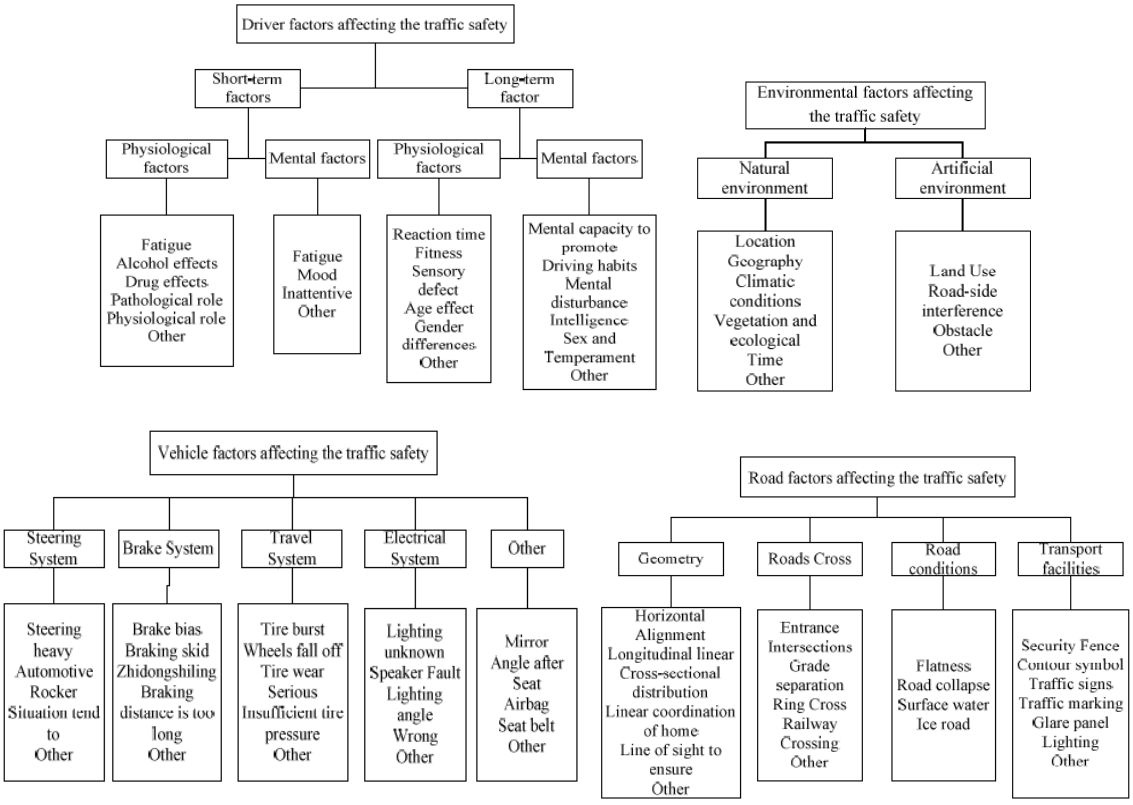


Figure 2-6 Traffic accident causes (from [12])

A diagram showing the dependencies of various accident causes and their relationships is created by [13] and shown in Figure 2-7. Similar studies to [7] would be required to quantify the impact of these causes.

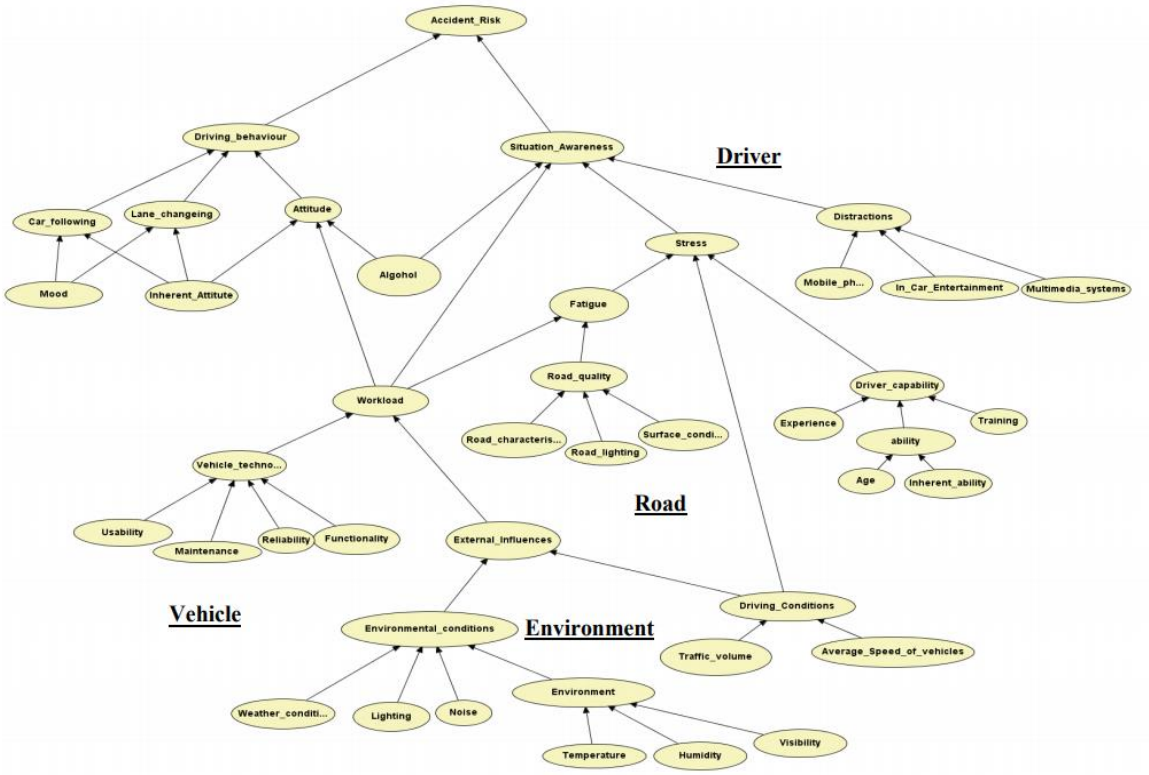


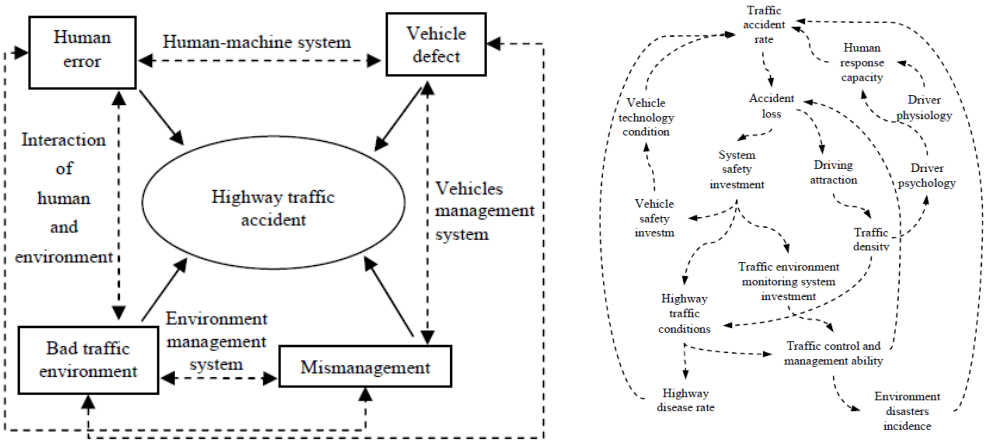
Figure 2-7 Accident causes dependency diagram (from [13])

Figure 2-8 shows similar main risks factors identified by [14] (human, vehicle and equipment; and environment), while also considering three different crash phases: (1) pre-crash: mainly focusing on crash prevention, (2) crash: focusing on the injury prevention during the crash and; (3) post-crash: focusing on life sustaining. This study therefore suggests a complete new approach that includes the idea of accident timing phases.

		FACTORS		
PHASE		HUMAN	VEHICLES AND EQUIPMENT	ENVIRONMENT
Pre-crash	Crash prevention	Information Attitudes Impairment Police enforcement	Roadworthiness Lighting Braking Handling Speed management	Road design and road layout Speed limits Pedestrian facilities
Crash	Injury prevention during the crash	Use of restraints Impairment	Occupant restraints Other safety devices Crash-protective design	Crash-protective roadside objects
Post-crash	Life sustaining	First-aid skill Access to medics	Ease of access Fire risk	Rescue facilities Congestion

Figure 2-8 Accident risk factors per crash phase (from [14])

Finally, a system approach could be used to incorporate the various causes into a model, similar to the highway traffic safety analysis model is proposed in [15]. The authors study the causes of highway traffic accidents and design a model using system dynamics. The system models accidents as an interaction of 4 major categories of factors (Figure 2-9a). The possible interaction is studied as a cause-effect relationship, as shown in Figure 2-9b. Within each block of the system, a model is proposed using system dynamics based on accumulated variables, decision variables and the structure comprising them.

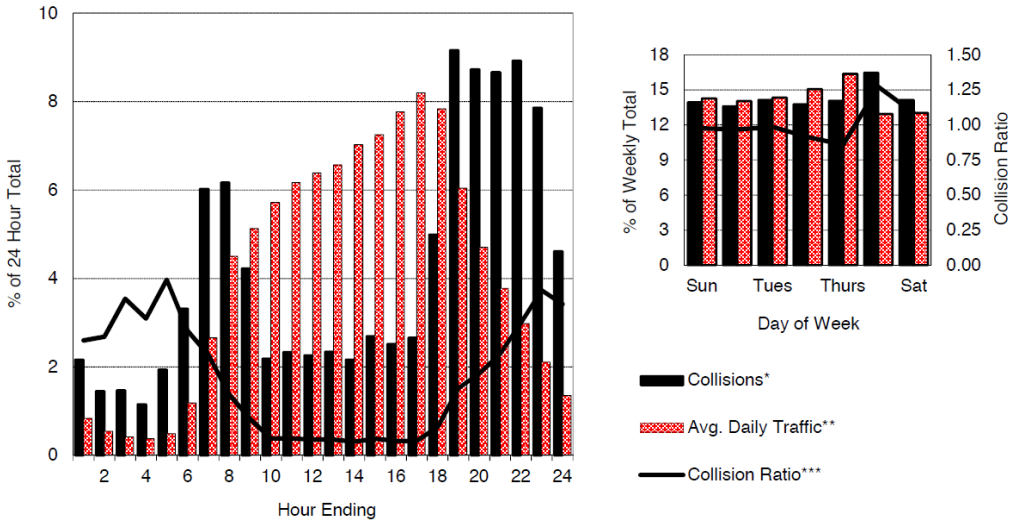


(a) **(b)**

Figure 2-9 (a) Highway traffic accident analysis model and (b) modeled cause-effect relationships within (from [15])

2.2. Accident Trends

Several reports have analyzed the frequency of accidents over the days of the week, over the time of day and over the months. A few Canadian studies [16], [17] identify Friday as the day of the week the most collision-prone. As shown in Figure 2-10b, another study over the provincial highways in Saskatchewan identified similar trends [7]. Almost 17% of all police-reported collisions in Canada seem to happen on Fridays [18], [19], [20]. Friday is theoretically followed by Wednesday, Thursday, Tuesday then Monday. Saturday is the second lowest and Sunday the lowest in terms of accident occurrence [20].



(a) (b)

Figure 2-10 Saskatchewan provincial highways collisions by (a) time of the day and (b) week day (from [7])

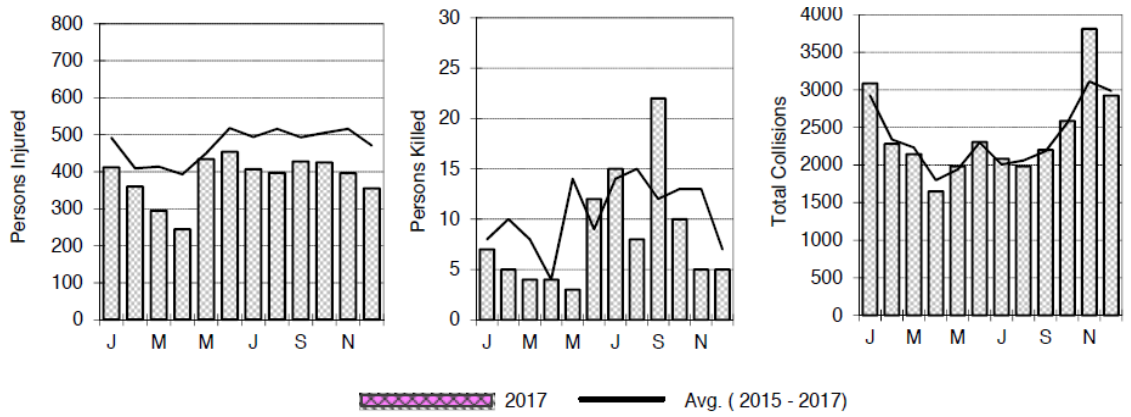


Figure 2-11 Collisions and victims by month of occurrence (from [7])

A study of the accident distributions over days identified the evening rush hour (starting at 3 p.m. and ending at 6 p.m.) as the time when the most collisions occur [18]. A similar study over highways identifies trend periods during the morning (starting at 7 a.m. and ending at 9 a.m.) and afternoon rush hour (starting at 6:00 p.m. and ending at 12 a.m.), illustrated in Figure 2-10a [7]. In terms of fatal accidents, the majority seem to occur between 6 p.m. and 9 p.m., while the second-deadliest time of the day was the interval between 3 p.m. to 6 p.m. [21]. During the week, most fatal accidents occurred between 5:00 and 5:59 p.m [22].

Finally, a certain trend can be identified over the months [17], [23]. In general, the fall and winter months from October to January and in June, a higher number of accidents seem to occur, as it can be seen in Figure 2-11. In terms of fatal accidents, the summer months seems to be the deadliest, with June hovering on the top [24]. The statistics of an insurance company based on their claim reports over 10 years showed that the biggest amount of collisions happened from December to February, supporting partially the previous findings [19]. In terms of days of the week, the analysis results from the same insurance company over 2016 are consistent with the previous studies [20].

A periodical re-evaluation of these trends and their customization for specific regions can be included in an intelligent system to improve the predictions of collisions by either considering them as input variables or by weighing stronger prone days, times and months [25]. This is an aspect that will be studied and incorporated in this thesis work.

2.3. Methods for Accident Modeling and Prediction

Once the important factors and trends for accident prediction have been identified, it is important to identify in the literature the different methods used for the modeling and prediction of traffic collisions. Because it represents an important problem that affects the whole society, it has been tackled by an important number of researchers. In fact, a variety of techniques covered under the umbrella terms “computational intelligence” and “data mining” were employed in the literature for this purpose, including: neural networks, support vector machines, fuzzy and hybrid fuzzy techniques, regression analysis, decision trees, Bayesian networks, association rules, clustering techniques, case-base reasoning and ontologies. While some solutions are most often used than others, in particular neural networks, regression, fuzzy techniques and decision trees, none of these is a method of clear choice within the research community. These techniques and their use for accident modeling and prediction are briefly described in the following sub-sections.

2.3.1. Neural Networks

One of the most frequently employed solutions is *neural networks*. The strength of neural networks stems from their intrinsic non-linearity, their computational simplicity and the ability to learn, and therefore to predict. A neural network has the capacity to learn high-dimensional, redundant mappings from sets of measured data without an a priori mathematical model. Although they may take a relatively long time to learn, the recall phase is done in real-time. An estimation of the output can be provided instantaneously for input values that were not part of the training set.

Various forms have been employed in the literature for traffic accidents: backpropagation networks in [26], [27], probabilistic neural networks in [28], [29], and wavelet neural networks in [30]. A form of neural network, support vector machines (SVM) has also been used in the context of traffic accidents in several publications [31], [32], [33], [34]. Hybrid neural network solutions, such as adaptive neuro-fuzzy techniques (e.g. ANFIS), are proposed by [35], [36], [25] and Grey-neural network hybrids by [37], [38].

A *backpropagation neural network* [27] is trained over data representing 102 signalized intersections and 3441 crash records from 1999-2004. The network has 34 inputs representing the road width, the percentage of motorcycles, the complexity of signal timing scenario, the overall traffic volume, the percentage of left turning vehicles, the signal cycle time, the directional traffic volume, the existence of warning signs, the median type between fast and slow traffic lanes, the obstacles within traffic lanes, and the percentage of right turning vehicles and predicts the number of accidents. It was found that the neural network model provides more reliable predictions (i.e. 13.53% misjudgment rate) than negative binomial regression model. Another backpropagation neural network [26] classifies the accident type as single vehicle accident, rear-end accident, front collision accident, side collision accident, or scratch accident using as inputs the traffic volume at intersection, the location and type of intersection, the grade of intersecting roads and the traffic control mode.

A solution for traffic accident loss prediction based on *wavelet neural networks* and that takes into account the number of accidents, the number of deaths, and the number of injuries is proposed in [30].

Lv *et al.* [31] predict traffic collisions using *Support Vector Machines* based on three variables: the 5-minute standard deviation of time headway, the 2-minute standard deviation of speed and the 1-minute standard deviation of traffic volume, identifying 76.7% (recall) of the hazardous traffic conditions.

Li *et al.* [25] use a hybrid neural network model to predict the road risk index (RRI). The raw data is split into 3 clusters using *fuzzy C-means clustering* and, for each cluster, a separate neural network is developed to predict a different crash rating: fatal, injury and property damage only (PDO). For clarification: a *fatal accident* is a collision that results in at least one death, with the death occurring either at the scene of the collision or within 30 days from the date of the collision [7]. An *injury accident* is a collision that results in at least one non-fatal injury (any bodily harm) from all persons that were involved in the collision [7]. Finally, a *property damage only* accident is a collision that only involves property damage to the vehicles and/or the property due to the collision, when no apparent (or stated) injuries or deaths have occurred [7]. Each of the separate networks has as inputs the road roughness, the speed limit, the segment length, the number of lanes, the annual average daily traffic (ADDT) per lane, the width per lane, the curve and the grade and three outputs corresponding to the three crash ratings (i.e. fatal, injury, PDO). To compute the output of the hybrid network, the Euclidean distance is calculated between the input and the cluster centers, and the neural network corresponding to the minimum-distance cluster is applied to predict the accident type and the road risk index. The model is also augmented with a dynamic layer to include influences of weather, time of day and day of week by employing multiplicative correction factors. The hybrid network solution demonstrated a better performance than ordinary least squares and quantile regression.

Polat and Durduran [35] propose a *K-Means attribute weighting for a neural network*, in order to increase the performance of the classification algorithm and to transform the linearly non-separable traffic accidents dataset to a linearly separable

set. Their dataset contains 358 observations including 179 without accident and 179 with accident of a highway in Turkey. The considered input variables are: the day, temperature, humidity, weather conditions, and month of the occurred traffic accidents and the output is accident/no accident. The performance of their neural network is 74.15%.

Zhu [37] predicts traffic fatalities by training a *Grey backpropagation network* on the China traffic fatality data from 1990 to 2006. Grey relation entropy analysis is also used for accident prediction in [28], [39].

2.3.2. Fuzzy Techniques

Another often used method is fuzzy techniques and their variants, including fuzzy logic [40], [41], fuzzy regression [42], fuzzy clustering [43], [25], evolutionary fuzzy rules [44], fuzzy rules and genetic programming [45], fuzzy granular decision trees [46] and hybrid neuro-fuzzy approaches (ANFIS) [36], [25], [35].

In [41], a *fuzzy logic inference system* predicts the number of accidents per kilometer at each artery road based on: (1) the annual average daily traffic (AADT): low, medium and high, (2) the driving difficulty: easy, medium, less difficult and difficult, (3) the lane width: narrow, medium and wide and (4) the velocity: low and high. The output variable considers five categories: few, less, medium, more and many. The total error achieved is 8%.

An *Adaptive Neuro-Fuzzy Inference System* (ANFIS) is employed in [36] to predict road accidents based on a set of candidate variables such as: mean horizontal curvature, shoulder width, road width, land use, access points, longitudinal grade, and horizontal curve density. From these, various variable selection solutions explored, including the Akaike Information Criterion, the Bayesian Information Criterion (BIC), and Mallows' Cp, showed that the best subset includes the variables: land use, road width, access points, and shoulder width.

A *Fuzzy Granular Decision Tree*, a generalization of the classical decision tree where training data are fuzzified in the form of membership functions, is proposed in [46] to generate the road accident rules applying the discrete and continuous data

stored in accident databases. Fuzzy granular entropy is calculated for each object in the data set. According to the calculated entropy, generality and redundancy criteria, the fuzzy granular decision tree is constructed. The solution takes into consideration the environmental factors and spatial characteristics of accident locations, namely the weather (clear, raining, fog), the surface (dry, not dry), the road lighting (day-light or dusky/dark), the collision time, the road radius, the road slope, the distance from intersection (very near, near, far), and the distance from population to categorize the roads into two classes: safe and unsafe. The final model accuracy is 83.1%.

2.3.3. Decision Trees

Another popular solution for traffic data analysis and for the identification of accident causes is decision trees [47], [46], [27], [48], [32], [49], [50], [51], [52], [53].

Zhang and Fan [47] obtain the probability distribution of the factors that cause various kinds of traffic accidents based on 1-year data from the Saskatchewan Highway Authority by employing a decision tree. The considered factors include: attention, drink, physical (condition), inexperience, rule (break rules), mistakes, speed, vehicle, road, weather, and sight. Experiments are conducted through three different aspects with respect to age, season and gender. In particular, with respect to age, major contributing factors for junior group were 'drink', 'rule' and 'mistake', for adult group: 'rule', 'drink' and 'attention' and for the senior group 'rule', 'physical' and 'mistake'. In terms of seasons, major causes for winter accidents were 'drink' and 'rule', while for non-winter condition: 'drink', 'mistake' and 'rule'. Finally, according to gender, for the male group: 'drink', 'mistake' and 'rule' and the female group: 'rule', 'weather' and 'drink'. The obtained performance over the three aspects varies between 75.9% and 82%.

The authors of [50] classify accidents as fatal, injury and non-injury using *CART decision trees, TreeNet, Random Forests* and *ensemble methods*. A series of 32 attributes are used over a dataset of 4-years in Addis Ababa, Ethiopia including: collision type, subcity, age and occupation of victims, vehicle type, health condition of victims, immediate cause of accident, category of victims, accident hour, driving

licence level, driving experience, day of accident, light condition, vehicle plate category, road separation, age of driver, specific week of a month, service year of vehicle, vehicle ownership, educational level of driver, type of road junction, gender of driver, road orientation, pedestrian movement during the accident, technical status of vehicle, road condition, weather condition, and road surface type. The ensemble model that includes CART decision trees, TreeNet and random forests achieves the best performance in terms of accuracy: 87.6% over fatal, 78.4% injury and 100% over non-injury accidents.

A CART-based decision tree for predicting injury level (fatal, injury and no-injury) using 22 variables is proposed in [51]. The variables employed include: the month, the hour, the day of week, weather condition, light condition, road surface, road obstruction, accident location, control type, divided highway, speed limit, gender, age, qualification, restraint system, sobriety condition, driver/vehicle/pedestrian action, collision type and contributing circumstance and their description and type. The model obtains a performance of 96.4% for injury and 88.5% for non-injury, but is not able to predict fatal accidents (0%).

Saha *et al.* [52] investigate *boosted regression trees* for finding the association between the variables and crash predictions over five years of crash data on two urban and suburban facility types, two-lane undivided arterials and four-lane divided arterials.

A *gradient boosted tree* was used by [53] to classify accident/non-accident samples, considering a series features such as: weather (i.e. temperature, wind speed, snow depth, etc.), time (i.e. hour of the day, day of the week, month of the year, etc.), static information (i.e. speed limit, road curvature, average traffic volumes, proximity to nearest intersection, road sinuosity, etc.), human factors (i.e. population density, proximity to billboard, etc.) and road network graph derived features (i.e. centrality and flow). The author also proposes a method to generate non-collision samples. The model was able to predict 90% of accidents (recall) with a precision of 30% and accuracy of 68.6%.

Other researchers make use decision trees and random forests for the variable selection process (as detailed in section 2.3.8 below).

2.3.4. Regression

The last among the most often-used techniques is regression, in its various forms: joint probability regression [54], generalized linear regression [55], logistic regression [56], logit [57], ordered probit, ordered logit, multinomial logit [58], Moran's I local indicator of spatial association [59], [60], multivariate Poisson-lognormal [61], just to mention a few.

Joint probability regression is used in [54] to model the frequency and severity of crashes. The considered variables are road density, junction density, vehicle speed, time, land-use pattern.

The authors of [55] propose a *generalized linear regression model* to produce an estimate of the collision frequency for a location based on the site-specific characteristics using traffic data from British Columbia. Two major sources of data are used: (1) highway segments described by land use (urban/rural), road class (arterial, expressway, freeway), median (divided/un-divided), and the number of lanes (2 or 4) and (2) highway intersections described as signal/stop-controlled and 4-leg or 3-leg, respectively. Five-year collision history data (2001-2005) and 5-year traffic volumes for each highway segment are used to create distinct models per highway segment and per intersection for each type of collision, including fatal and injury collisions (combined in "severe") and property damage only collisions. For example, a model is proposed for a two-lane highway segment for severe collisions and another one for two-lane highway segment for property damage only collisions. The estimation of model parameters is done using generalized linear regression.

A *logistic regression model* for accident severity in two categories (major and minor accidents) is proposed in [56] based on accident date (working day, holiday), weather conditions, pavement type, road cross-section, accident location, road alignment, road type, traffic control and lighting conditions.

Environmental factors critical to intersection crash occurrence are identified via *negative binomial regression* and artificial neural networks over 102 signalized intersections and 3441 crashes in [27]. With these factors, a decision tree is applied

to categorize intersections' safety level (safe, fairly safe, fairly unsafe, fairly dangerous, dangerous, extremely dangerous).

The time of day of crashes involving large trucks is analyzed using a logit regression model in [57].

Zhan *et al.* [61] employ a *multivariate Poisson-lognormal* regression to model property damage, possible injury and evident injury accidents over the Washington state highway crash dataset based on 8 variables including: the highway segment length, the average annual daily travel per lane, the maximum grade difference in the segment, the number of horizontal curves per mile in the segment, the percentage of trucks in the traffic, the low precipitation indicator (≤12 in. per year), the heavy snow indicator (≥18 in. per year) and the local road indicator.

2.3.5. Bayesian Networks

Bayesian networks and classifiers have also been used for traffic analysis and prediction [62], [48], [63], [64], [11]. A traffic accident causality analysis using a-10 node Bayesian network is produced in [62]. The considered factors are the following: (1) environment (terrain, weather and traffic control), (2) road (road type, pavement type, road alignment, type of intersection and road section, road cross-section, road condition); and (3) traffic accident (cause of accident, accident form, accident type) and (4) accident casualties (number of deaths, number of serious injuries, number of light injuries, and number of property damage accidents).

Paikari *et al.* [63] use *clustering* and *Bayesian Belief Networks* to predict car accidents per street, using historical and real-time data. The historical data includes the collision data (i.e. time and location, alcohol involvement, traffic and road conditions, etc.) and location related data (i.e. road geometry, traffic signs; and proximity to schools, hospitals, bars, etc.). The real-time data includes the weather conditions and road conditions (i.e. temperature, snow, rain, icy surface, etc.); and traffic updates (i.e. traffic flow, accidents, link/lane blockage, over capacity percentage, etc.).

Gregoriades [13] capitalizes on an integration of a microscopic road network simulation with *Bayesian Belief Networks* for improved prediction of road accident risk. The proposed Bayesian network classifies accidents as fatal, major or minor.

A *Markov Chain Monte Carlo (MCMC)*-Bayesian network is used by Qin *et al.* [64] to analyze the crash occurrence by crash type (multi-vehicle same direction, multi-vehicle intersecting direction, single vehicle, multi-vehicle opposite direction) using time of day factors.

2.3.6. Association Rules

Association rules play also a role in traffic data analysis [65], [12]. A solution for mining a 3-year data set of traffic accidents based on association rules is proposed in [65]. The data set contains 20 binary variables (referring to vehicles involved, the causes of the accident and human losses) and 3 nominal values (time of accident, scene and feature). Association rules are also exploited in [12] for cause analysis in road traffic accidents.

2.3.7. Other Techniques for Traffic Accident Modeling and Prediction

A few other solutions proposed for traffic accident modeling and prediction are clustering techniques such as K-nearest neighbors [66], [48], C-means clustering [66], DBSCAN [67], case-base reasoning [68], [69] and ontologies [67].

The work of [66] uses clustering techniques on data representing mean and standard deviation of traffic volume, speed and time headway in order to predict accidents (property loss, slight injury, fatal, severe injury) in a simulated real-time highway traffic model. They show that, for their dataset, the *K-nearest neighbour clustering* outperforms C-means clustering, achieving 80% accuracy.

The study of [48] compares three methods, namely *J48 Decision Tree, Naïve Bayes, and K-Nearest Neighbors* to classify accident severity (e.g. fatal, severe injury, slight injury and property loss). Input variables are the subcity (where the accident occurred), the particular area (school, market), the road separation, the road

orientation, the type of road junction, the road surface type, the road surface conditions, the weather conditions, and the light conditions. The three methods obtained very similar results (within 0.8% difference; roughly 80% performance in terms of accuracy).

Jagannathan *et al.* [68] use a system capitalizing on *case-based reasoning* to predict the outcome of current traffic flow conditions based on historical flow data cases that led to accidents and to differentiate between accident and no-accident conditions. Four groups of attributes are used in the decision-making process including: (1) nominal attributes: time of day, time-interval/distance, (2) single point attributes: average number of vehicles (flow) over all lanes, average velocity of vehicles over all lanes, average headway between vehicles over all lanes, average occupancy measured by the sensor, standard deviation in average number of vehicles per lane (flow) between lanes, standard deviation in average velocity of vehicles between lanes, standard deviation in average headway of vehicles between all lanes, and standard deviation in occupancy of vehicles between all lanes; (3) temporal attributes: variation in average speed over all lanes and variation in average flow over all lanes; and (4) spatial attributes, namely: variation in average velocity over all lanes and variation in average flow over all lanes. The similarity is computed using the K-nearest neighbour algorithm and is based on combinations of attribute similarities in a similarity measure (i.e. the similarity measure is calculated as the average of the time-day category similarity, of the similarity based on spatio-temporal single point attributes, and of the similarity based on variation of flow over distance or time interval).

A case-based reasoning solution combined with GIS data for Metro LRT (Edmonton) traffic collision analysis is discussed in [69].

An improved version of density-based spatial clustering of applications with noise (DBSCAN) that takes into consideration both the number of accidents and their severity level is proposed in [67]. The authors also propose an ontology-based traffic accident risk mapping framework, in which the ontology represents the domain knowledge related to the traffic accidents and supports the data retrieval based on users' requirements.

2.3.8. Variable/Attribute Selection

Due to the fact that a large number of variables characterize traffic accidents, appropriate variables will have to be selected in order to identify the most salient ones, and therefore enable the learning and prediction of expert systems for collision detection and prediction.

Various solutions have been employed in the literature in this context, including: Akaike Information Criterion, Bayesian Information Criterion, Mallows' Cp [36], decision trees, namely CART decision tree [32] and random forests [70], and the Correlation-based Feature Selector (CFS) method that proposes a heuristic measure of merit of feature subset [33].

Hosseinpour *et al.* [36] use three criteria, namely Akaike Information Criterion, Bayesian Information Criterion, and Mallows' Cp to select the best subset of variables, e.g. the one that obtains the smallest values for these criteria. In particular, the values are computed over a pool of 7 variables and their various combinations.

A Correlation-based Feature Selector (CFS) method is applied to evaluate candidate factors possibly related to zonal crash frequency in handling high-dimension spatial data in [33].

A simplified CFS is also the *maximal information coefficient*, which captures a wide range of associations not limited to specific function types (e.g., linear, exponential, or periodic) or even to all functional relations [71].

Various decision trees have also been employed in order to select appropriate variables. In [32], a CART decision tree is used to select the most significant contributing variables prior to SVM training for real-time crash risk evaluation. The variable importance is calculated based on the number of times a variable appeared and its relative position in the tree.

Unlike the classification and regression tree models, random forest models can provide unbiased error estimates and do not require a cross-validation dataset [72]. In [70], during the tree growing procedure, the out-of-bag approach is employed (e.g. one-third of the data are left out from the training trees, data further used to achieve

unbiased estimate of variable importance as trees are added to the forest). The importance of a variable is therefore estimated through the random forest algorithm by monitoring how much the prediction error increases when out-of-bag data for that variable is permuted while all others were left unchanged.

2.4. Summary of Literature

The analysis of causes of road collisions found in the literature identifies mainly 4 major categories: (1) human condition/driver (i.e. driver inattention, distraction, inexperience, age, gender, drinking, drugs), (2) human action (i.e. fail to yield, traffic control disregarded, following too closely), (3) vehicle condition (defective brakes, lights or steering, seat belts) and (4) environmental and road factors (i.e. road surface condition, snow drift, sun glare, construction zone, lane marking inadequate, traffic control device not working). It is important to state, that while all these factors are known to be causes of accidents, not all of them are available for prediction in a real use scenario, as they only become available once the accident occurred. As such, our work will focus on only those factors that are available or could be made available using sensors in real-time. Beyond these, we also believe that social events can play an important role in accident predictions. In this work, as part of the objectives, we aim to validate this assumption.

A variety of techniques were employed in the literature for the purpose of accident modeling and prediction based on partial sets of the above-mentioned features, among which some are more often used than others, in particular neural networks and decision trees. This justifies the choice of these two techniques and their variants, namely gradient boosted trees and multi-layer perceptron in the context of the current work.

3. Methodology

The adoption of a well-established methodology increases the chances of success through the use of industry best practices. In this thesis, the *Cross-Industry Standard Process for Data Mining* (CRISP-DM) [73] was identified as a methodology to implement a proof-of-concept of the traffic collision framework.

The model defines six generic phases: (1) business understanding, (2) data understanding, (3) data preparation, (4) modeling, (5) evaluation; and (6) deployment that help to plan and execute a data mining/machine learning project. These phases are related and normally require several iterations. Figure 3-1 shows these phases of the reference model, where the arrows between the phases represent the typical dependency relationship and the circle that encloses it represents the typical cyclic relationship of a data mining project, in which the output of one phase provides an input to the other and generates a new iteration of the entire model.

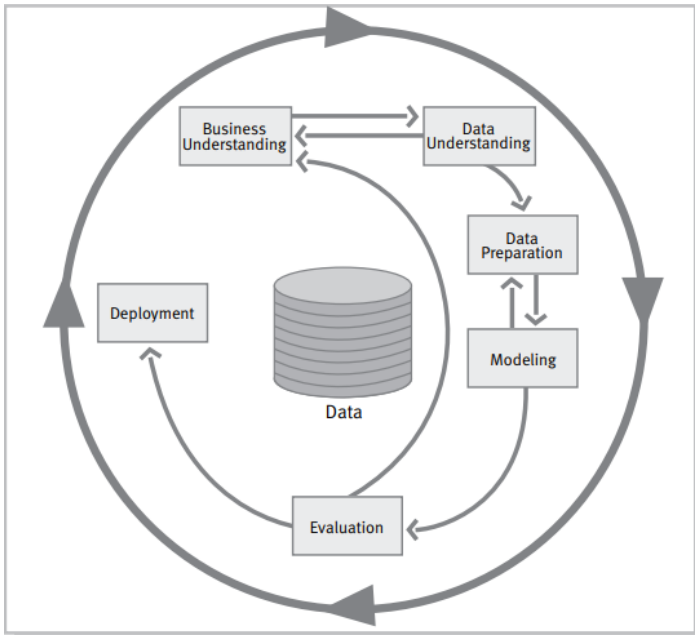


Figure 3-1 Phases of the CRISP-DM reference model (from [74])

Each phase includes different tasks and outputs, as illustrated in Figure 3-2. A description of each phase and task of the model is provided in the following sections.

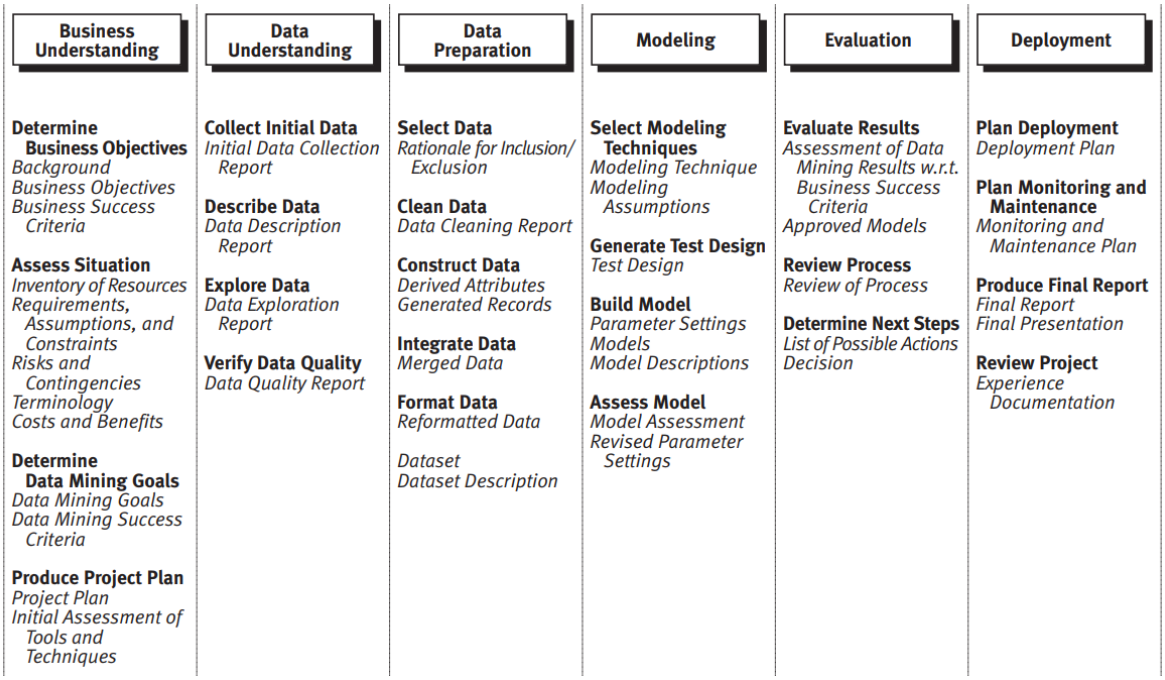


Figure 3-2 Generic tasks (bold) and outputs (italic) of the CRISP-DM reference model (from [74])

3.1.1. Business Understanding

As a model developed to guide the execution data mining projects, the first phase includes the understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data mining problem and a subsequent preliminary project plan designed to achieve the identified objectives [73].

This phase includes the following tasks [75]:

- Determine business objectives - this step focuses on the understanding of the client's goal for the project. The interaction with the stakeholders at this stage is crucial for future of the project.
- Assess the current situation - the objective is to create a list of the available resources (i.e. people, data, computing resources) to accomplish the project objective, as well as consider the different limitations, assumptions and constraints related to the project.

- Determine data mining goals: in this step, the project objectives are stated in business terms.
- Produce project plan - describes the intended plan for achieving the data mining goals, including outlining specific steps and a proposed timeline, an assessment of potential risks, and an initial assessment of the tools and techniques needed to support the project.

3.1.2. Data Understanding

The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information [73].

It includes the following tasks [75]:

- Collect initial data - includes the collection, loading and integration of the data if it is necessary.
- Describe data - the properties of the acquired data are analyzed and reports are produced on the results; several issues such as the format of the data, the quantity of the data, the number of records and fields in each table, the identities of the fields, and any other features of the data are examined.
- Explore data - tackles the data mining questions, which can be addressed using querying, visualization, and reporting.
- Verify data quality - the quality of the data is examined; some common items to verify include: missing attributes and blank fields; whether all possible values are represented; the plausibility of values; the spelling of values; and whether attributes with different values have similar meanings (i.e. '*BUSONLY*' and '*Bus only'*).

3.1.3. Data Preparation

The data preparation phase covers all activities to construct the final dataset (i.e. data that will be used for modeling) from the initial raw data. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order [73].

This phase includes the following tasks [75]:

- Select data – in this step, the data to be used for the analysis is chosen based on several criteria, including its relevance to the data mining goals, as well as quality and technical constraints such as limits on data volume or data types.
- Clean data – in this step, the selected data is cleaned and issues related to missing data and data quality are addressed in response to each problem reported earlier in the data quality verification step.
- Construct data - after the data is cleaned, data preparation operations are undertaken such as developing entirely new records (feature engineering) or producing derived attributes.
- Integrate data – this step involves combining information from multiple tables or records to create new records or values; it also covers aggregations.
- Format data - in some cases, it is necessary change the format or the design of data. These changes might be simple such as removing illegal characters from strings or trimming them to a maximum length, or more complex, such as those involving a reorganization of the information. Sometimes these changes are required to make the data suitable for a specific modeling tool.

3.1.4. Modeling

In this phase, various modeling techniques are selected and applied, and their parameters are calibrated to optimal values [73].

This phase includes the following tasks [75]:

- Select modeling technique(s) - one or more specific modeling techniques (i.e. machine learning algorithms) are chosen.

- Generate test design - in this stage, the process to test the model quality and validity is defined. In classification problems, it is common to use error rates as quality measures and also to split the dataset to create a train/test set.
- Build model: using the previous selected modeling techniques, in this stage, different models are built. A hyper-parameter optimization process can also be implemented as part of this stage.
- Assess model: once the results of each of the previous models are available, in this stage an interpretation of the models is performed, using the domain knowledge, the data mining success criteria and the different goals defined in the test design stage. A ranking of the models is also performed.

3.1.5. Evaluation

At this stage in the project, one or more models that appear to have high quality from a data analysis perspective have been built. Before proceeding to final deployment of the model, it is important to evaluate thoroughly the model, and review the steps executed to construct the model, to make sure that it properly achieves the business objectives. A key objective is to determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results should be reached [73].

3.1.6. Deployment

The creation of the model is generally not the end of the project. Usually, the knowledge gained will need to be organized and presented in a way that the customer can use it. Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as implementing a repeatable data mining process. In many cases, it will be the user, not the data analyst, who will carry out the deployment steps. In any case, it is important to understand up front what actions will need to be carried out in order to actually make use of the created models [73].

The following chapters of the thesis will describe the proposed framework for collision prediction and follow the development of its prototype through each of these phases.

4. Proposed Framework for Accident Prediction

This chapter discusses the proposed collision prediction framework that involves two main steps: (1) the identification and selection of a series of important features related with collisions (collision framework risk factors) that could be used for prediction and (2) the definition of a series of sequential steps required to create, apply and visualize through Geographic Information Systems (GIS) maps the results of the machine learning prediction model (framework process). It is important to mention that the proposed framework is theoretical and assumes that the identified risk factor data are available. As part of this work, we implemented a prototype of the framework for the city of Ottawa that uses only the available data. The details about the prototype are further described in the next chapter.

4.1. Identification and Selection of Important Features (Collision Framework Risk Factors)

Based on the findings in the literature (section 2.1), the proposed theoretical framework considers the following six categories of features:

(1) **Vehicle:** includes the type (i.e. automobile, truck, etc.), condition/maintenance, speed, location (real-time coordinates of the vehicle) and maneuver (i.e. turn right/left);

(2) **Roads:** this category includes information of the road segments such as: type (highway, residential, etc.), location (i.e. district/area, intersection, non-intersection), geometry (i.e. slope, sinuosity, alignment), surface condition (dry, snow, etc.), traffic control (traffic light, stop sign, etc.), traffic control condition (working, non-working), maximum speed and traffic volumes;

(3) **Events:** this category includes information (i.e. date, time, location) about different social events (statutory days, school days, sports, concerts, etc.);

(4) **Time/date:** the hour of the day, day of the week, day of the month, day of the year;

(5) **Weather:** this category includes the environment (snow, rain, dry, etc.) and light conditions (normal, artificial, dark, etc.); and

(6) **Driver:** this category includes the human related information: sex, age, driving experience, collision record, conditions (fatigue, etc.), distractions (radio, mobile phone, etc.).

These features are or could be (especially in the larger context of smart cities and connected vehicles) made available from conventional databases and/or from sensor data, i.e. the driver sex, age and collision record could be retrieved from an insurance database, a connected vehicle could report the information required about car maneuvers, a series of road sensors could report the traffic intersection volumes and road surface conditions, traffic signal sensors could be used to identify the traffic control conditions, weather conditions can be collected from weather sensors and a camera inside the car could be used to identify some driver conditions like fatigue and distractions.

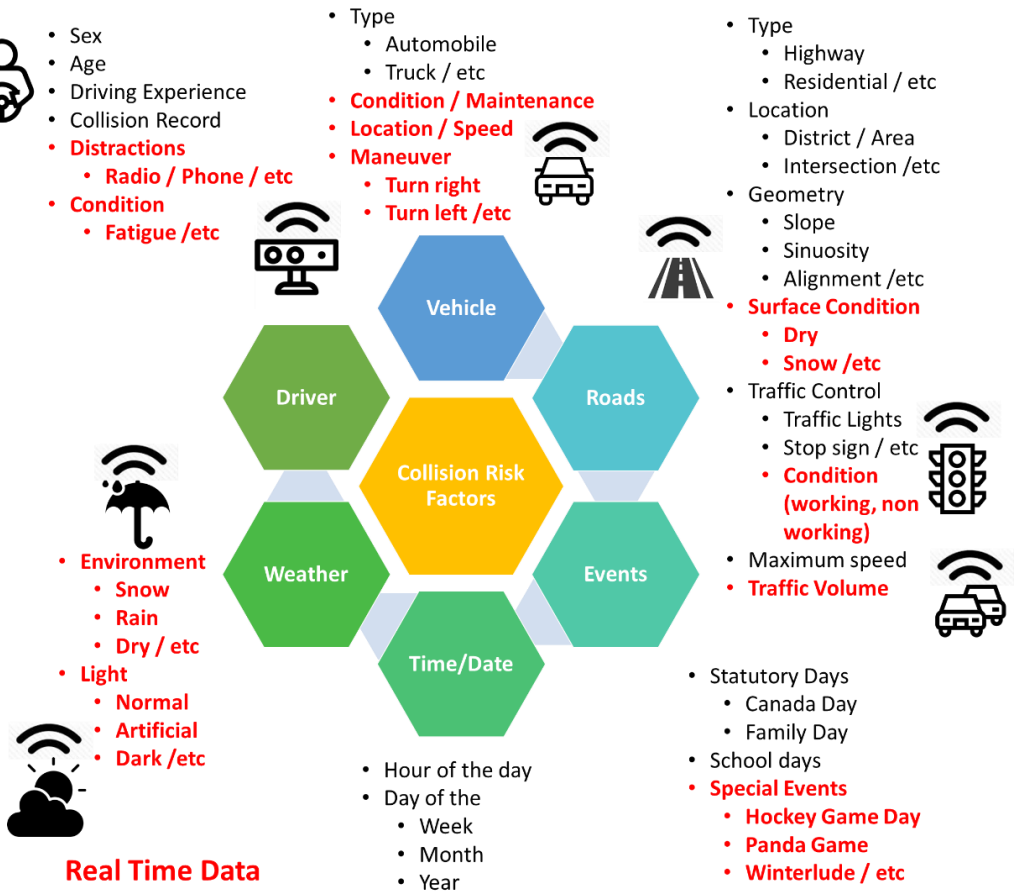


Figure 4-1 Collision risk factors per category with real-time ones highlighted in red (icons from [76])

It is important to note as well that some of the previous considered features must be handled in real-time (i.e. vehicle speed, road segment traffic volumes) and others could be considered nearly static (i.e. driver age, driving experience, social events) or completely static values (i.e. driver sex, road segment geometry). Figure 4-1 shows the different features considered with the real-time ones marked in red.

4.2. Definition of the Collision Framework Process

As it was mentioned before, the prediction framework includes the creation, application and visualization through GIS maps of the results of the machine learning prediction model. As part of the machine learning implementation, a series of well-known steps are considered as part of the process, related to the CRISP-DM model. The similarities with the latter will be described at the end of this section.

The framework considers the following twelve main steps, illustrated in Figure 4-2:

(1) **Collect Historical Data (Training Set)**: to generate the historical data, data regarding collisions, traffic intersection volumes, social events and road segments are collected.

(2) **Feature Engineering/Non-collision Samples Generation**: a typical step in any machine learning system is cleaning and feature selection process (*Feature Selection/Cleaning*, detailed in section 5.3.1); as well, in this step, new features are created and added to the historical data. Some of these are related with road segments (*Roads Feature Generation,* detailed in section 5.3.2.1), district / neighborhoods (*District / Neighborhoods Feature Generation*, detailed in section 5.3.2.2), date/time (*Date/Time Feature Generation,* detailed in section 5.3.2.5), social events (*Events Feature Generation*, detailed in section 5.3.2.6), location (*Location Feature Generation*, detailed in section 5.3.2.7), environment conditions (*Solar Azimuth / Elevation Feature Generation*, detailed in section 5.3.2.4), and others related with mathematical calculations/transformations (*Mathematical Feature Generation,* detailed in section 5.3.2.8). Because historical data only includes the information about different collisions, a mechanism to create non-

collision samples must also be implemented (*Non-Collision Samples Generation*, detailed in section 5.3.2.3).

(3) **Create Historical Dataset (Training Set)**: a historical dataset is created integrating all the original and previously engineered features (in step (2), as described in more detail in section 5.4).

(4) **Create Machine Learning Model**: using the previous data created (training set) the machine learning model is created including all the necessary data preprocessing and transformation (described in detail in sections 5.5, 7.1 and 7.2).

(5) **Collect Real Time Data (Test Set)**: real time data is generated/collected (i.e. vehicle, traffic volumes, events, solar azimuth/elevation and weather).

(6) **Feature Generation (Test Set)**: the same engineered features as the ones used for historical data in step (2) are created for the test set.

(7) **Create Real Time Dataset (Test Set)**: the existing and engineered features are integrated into the real time dataset.

(8) **Apply the Model to Test Set (Prediction):** the machine learning model created in step (4) is applied to the test set to generate predictions. The probabilities are extracted (details in section 7.1.5) to be used in the next step.

(9) **Multiplicative Correction Factor:** A post-decision step includes the addition of elements that improve the performance. The correction/weighting process is guided by understanding the importance of variables at the input (through statistical analysis) and also by using knowledge about the events that occur the day the accidents take place. As such, the proposed solution selectively increases the confidence level of predictions associated with given conditions that are known as more prone to accident occurrence (details in section 6.4).

(10) **Collision Probability per Road Segment:** the previous probability predictions are weighted to produce the collision probability per road segment (detailed in section 7.1.6).

(11) **Collision Probability Map per Road Segment:** to visualize the results in a more meaningful way, a collision probability map per road segment is generated in a GUI (detailed in section 7.1.6).

(12) **Update Collision Dataset with Real Data and Update the Machine Learning Model:** to keep the model updated, on a daily basis (or when additional information becomes available), the real collision information will be added to the historical dataset to retrain the model.

A matching between the different phases that are part of the CRISP-DM methodology and the previous proposed framework is presented in Figure 4-2.

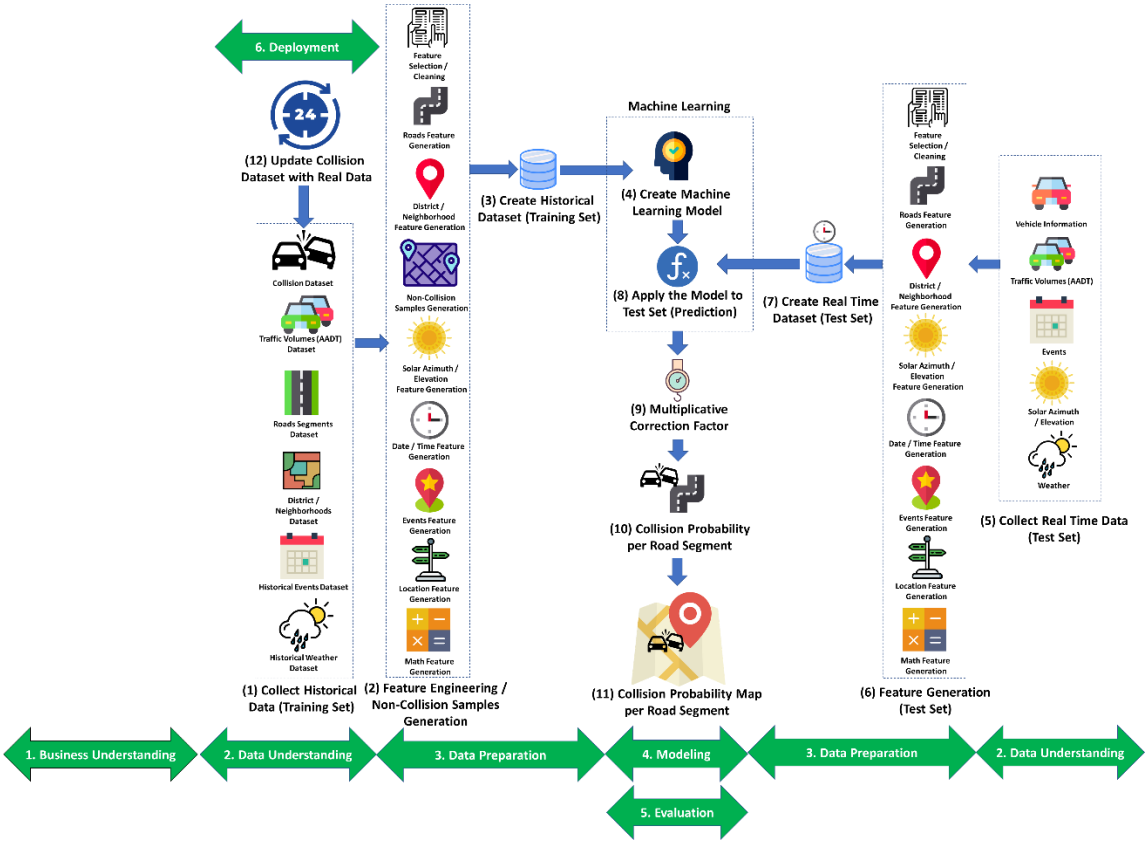


Figure 4-2 Matching between the proposed collision prediction framework and CRISP-DM reference model (icons from [76])

The next chapter describes a prototype implementation of the proposed framework for the city of Ottawa, Canada, following the phases of the CRISP-DM model.

5. Prototype Implementation: Business/Data understanding and Preparation

5.1. Business/Data understanding

The first two phases of the CRISP-DM methodology (described in chapter 3) are the business and data understanding. The main objective of the business understanding is establishing the objectives, that means, identifying the question to be answered and translating that question in a data mining/machine learning language. As a key factor, the understanding of the data will provide the necessary insights that could help us validate if the objectives are achievable with the existing resources. These two phases are normally related, and one determines/limits the other one.

5.1.1. Determine Business Objectives

After preliminary discussions with the Business Performance Section of the Ottawa Police Service, the main objective of the work in this thesis is to build a model to predict the probability of a collision by road segment in the City of Ottawa, in one-hour time-frame. We aim at an expected classification performance of at least 70% of precision and 70% of recall and at identifying models delivering good performance that can be updated through retraining with newly available data every 24 hours. As part of the objective, it was decided that it would be beneficial to visualize the results using GIS maps. The latter can be used for proactive traffic interventions schemes by the Ottawa Police Service.

5.1.2. Inventory of Available Resources

Having a clear idea about the available resources to accomplish the project (i.e. data, software, computing resources and people) is an important step that help us validate if the business objectives can be achieved. For this project, the resources available are the following:

- **Data:** through the Open data initiative [77], a series of datasets are made available, including the Tabular collision dataset [78], as well as geospatial

layers related with the City of Ottawa roads segments [79] and neighborhoods [80]. These will be described in detail in section 5.1.4.

- **Software:** different open source tools and libraries are available for data mining/machine learning and also through public licenses for research students. During the implementation phase, different software tools were used/tested including several programming languages (*R + RStudio* [81], [82], *Windows Excel Macros* [83], *Windows Batch Scripts* [84]) and also various software tools (*Tableau Desktop* [85], *RapidMiner Studio* [86], *ESRI ArcGIS Pro* [87]). These were chosen because of previous working knowledge (*R + RStudio, Windows Excel Macros*), of literature suggestions (*Tableau Desktop, RapidMiner Studio*), and of license availability (a license of *ESRI ArcGIS Pro* is available for Carleton Students for one year at the library). The use of these different tools creates a hybrid development/testing research environment that requires to execute/debug different software tools and thus reduces the implementation speed. For example, to generate an initial end-to-end process, the features were generated using *Windows Batch Scripts* (Coordinates Conversion), *Excel Macros* (Solar Azimuth / Elevation), *R + RStudio* (Roads Features) and *RapidMiner Studio* (Date/Time Features). After that, the machine learning model was created using *RapidMiner Studio*, the multiplicative correction factor was handled using *Tableau Desktop* and *R + RStudio* and the final collision probability road segment map visualization was performed using *ESRI ArcGIS Pro*. A special bottleneck in the process was related with the last step (collision probability map visualization) using *ESRI ArcGIS Pro*, considering the amount of data to handle and the lack of computing resources (a laptop GPU), the tool was almost unusable. Figure 5-1 shows the different programming languages and software tools used at the beginning of the implementation process.

  To optimize the end-to-end development process, Python was chosen as a main programming language due to the extensive available libraries that allow to unify the end-to-end process. *Jupyter Notebook* [88] was chosen as an Integrated Development Environment (IDE) based on its ability to create and

share documents based on live code and visualization. Because different Python libraries have different library dependencies, *Conda* [89] was chosen as package, dependency and environment management.

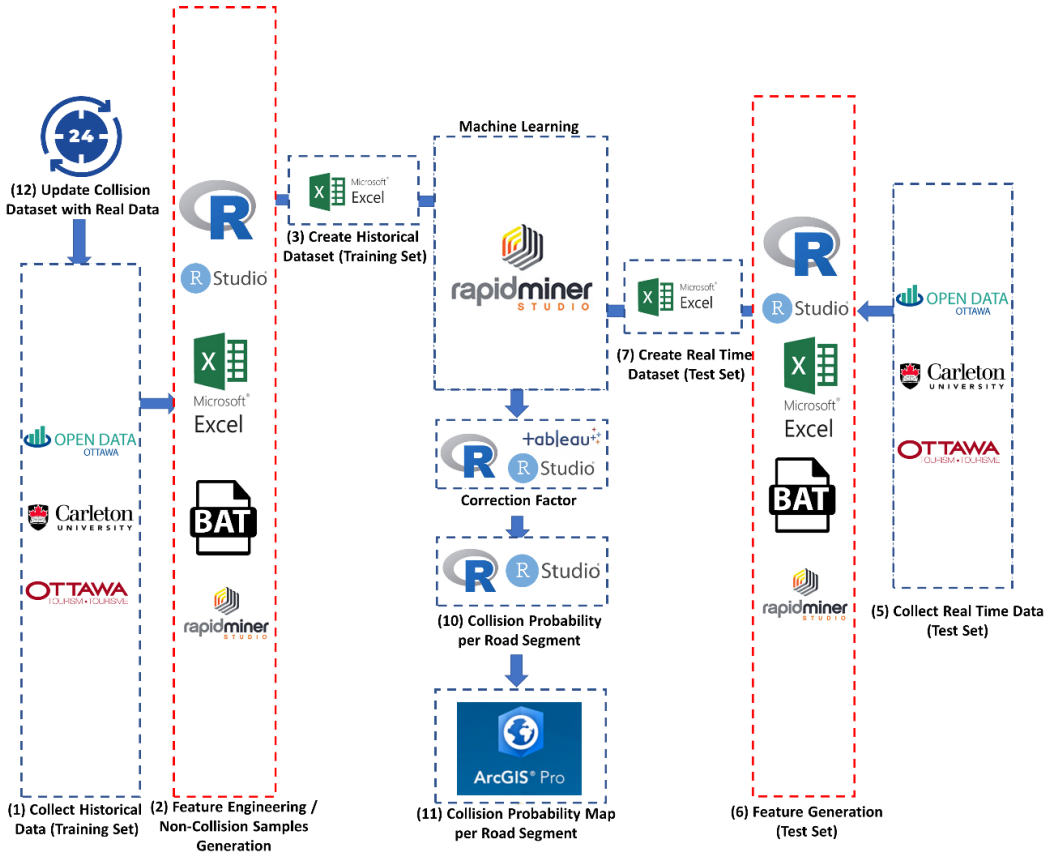


Figure 5-1 Initial selection of software tools for the proposed traffic collision prediction framework

For interactive map visualizations, an alternative open source solution was tested, namely *QGIS* [90] that works very well considering the lack of hardware resources available.

As a result, the initial hybrid development/testing research environment was reduced to four main components: (1) an end-to-end implementation in Python, using (2) *Jupyter Notebook* as IDE, (3) *Conda* as an environment manager and (4) *QGIS* as a map visualization tool.

Figure 5-2 shows the final software tools/data sources used in each of the steps related with the proposed traffic collision prediction framework.

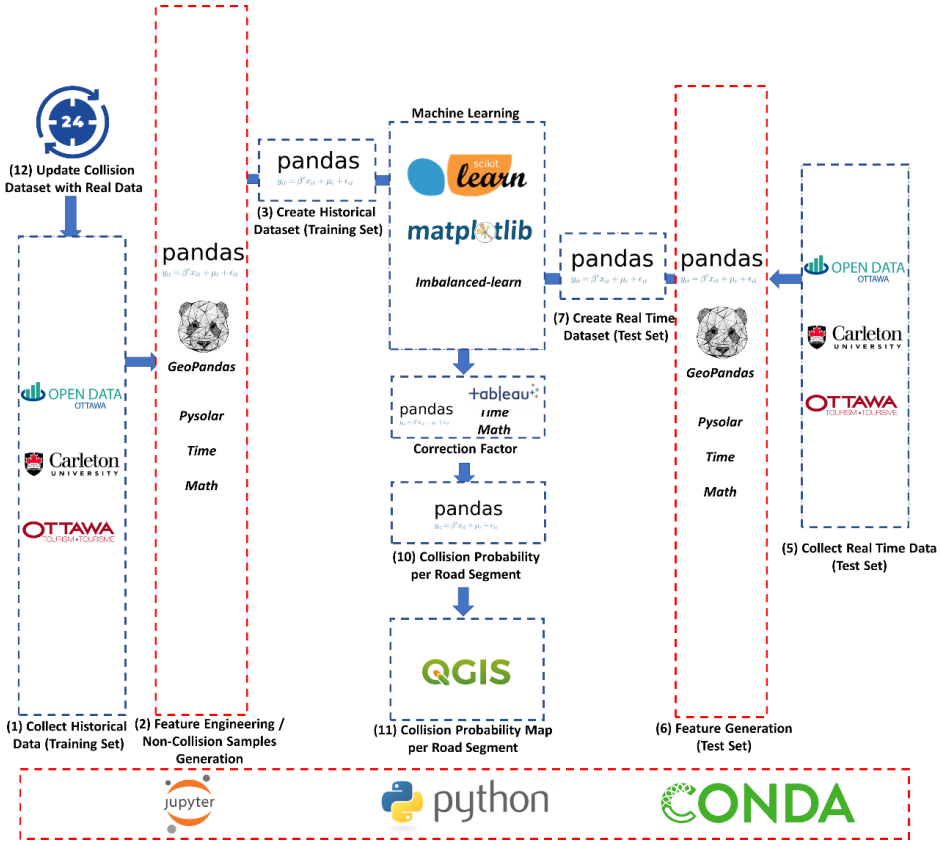


Figure 5-2 Final selection of software tools for the proposed traffic collision prediction framework

- **Computing Resources:** A special constraint is related to the limitation of the available computing resources, namely a laptop with an Intel Core i5-7200U 2.50GHz CPU, with 16GB of RAM, and equipped with an NVIDIA GeForce 940MX with 2GB VRAM GPU; Since the proposed collision prediction framework is considered to require a retraining of the model every 24 hours with newly available data, several design issues, such as the choice of the algorithms for learning are conditioned by the available computing resources and limitations.
- **People:** As stated previously, the domain knowledge in this project is provided through the collaboration and support of the Transportation Data Collection & Analytics Section of the City of Ottawa, as well as the Business

Performance Section of the Ottawa Police Service. The interaction with both was crucial to gain important insights into the problem domain and validate the suitability of the results.

5.1.3. Data Description

As part of the Canada Open Data initiative, the City of Ottawa provides a catalog of useful information to the public [77], including different kind of information (i.e. census, transportation, crime, collisions, etc.) in different electronic formats (csv, xlsx, xlm, GeoJSON, kmz, etc). This catalog represents the main repository used in the prototype of our framework.

The main dataset used in this work is the *Tabular Transportation Collision Data* from the city of Ottawa Transportation Services Department, Traffic Services Branch (Ottawa City - Traffic Department); it includes tabular information about each collision in the City of Ottawa from 2015 to 2017 [78].

Because the main dataset includes the spatial coordinates of the collision and motivated by the findings in the literature described in section 2.1, suggesting that the road type (i.e. highways and rural) has an effect on the collision trends, additional spatial layers datasets are used to create additional features including: roads [79], neighborhoods [80] and the city limits [91]. These new features are expected to help increase the predictive power of the machine learning model using the geo localization characteristics. Also, as part of the objectives, additional sources for social events are used [92] to validate if they contribute to accident collision rates.

The next sections describe in more detail each dataset.

5.1.3.1. Tabular Transportation Collision Data from Open Data Ottawa

As briefly mentioned in the previous section, this dataset contains collision occurrences across the city of Ottawa for three years, from 2015 to 2017 [78], a total of 43494 samples and 12 attributes for all three years together. Figure 5-3 shows a spatial visualization of this dataset using the *QGIS* software, where each blue dot represents a collision, while Figure 5-4 shows a more detailed view of the roads around Carleton University, in which an *Open Street Map* base layer was used to

identify the roads. A base layer or *basemap* is a background of geographical context added to the map that enhances the information displayed [93]. In this case, the *basemap* used includes the roads in the City of Ottawa.

Figure 5-3 Ottawa tabular collision data spatial visualization using QGIS, with blue dots representing collisions.

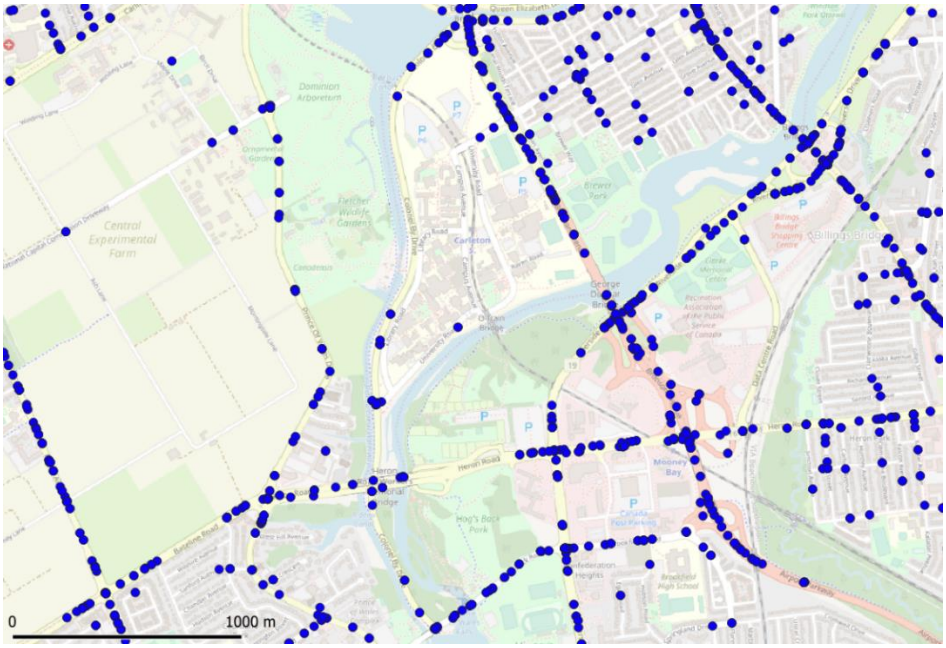


Figure 5-4 Carleton University area tabular collision data spatial visualization using QGIS

Table 5-1 shows the information related with the different dataset features, using their description from the Motor Vehicle Collision Report Manual [94]:

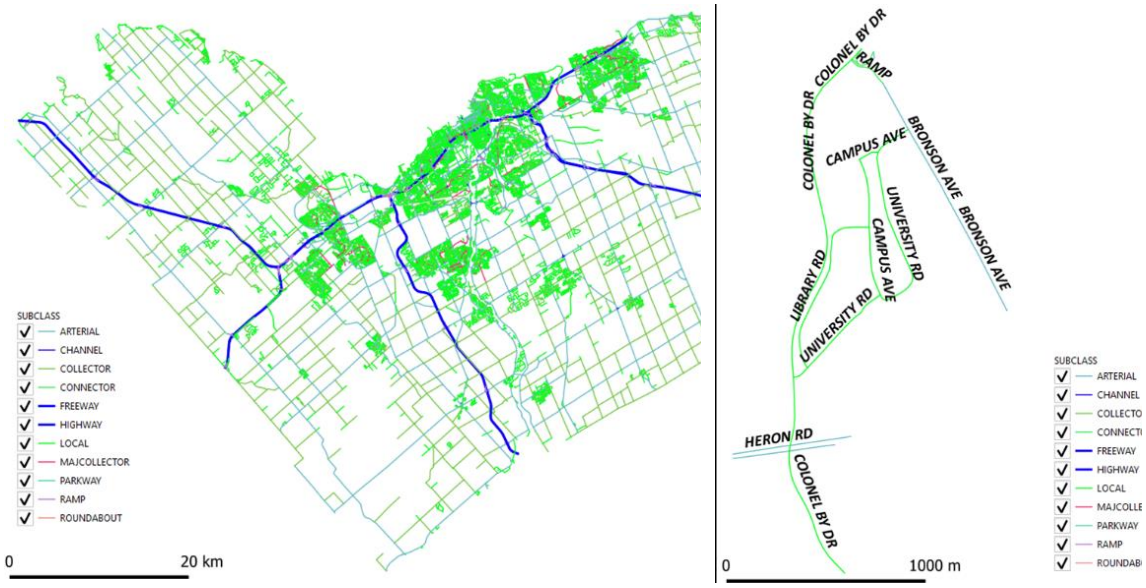
Table 5-1 Main Tabular Collision Dataset

Feature Name	Feature Description	Value Type / Example
LOCATION	A character string that identifies the street location (name) where the collision happened. Can be: a) An intersection of two streets: *STREET1 @ STREET2* b) On a main street, between the intersection of two adjacent streets: *STREET1 btwn STREET2 & STREET3*	String Example: *'ALTAVISTA RD @ BANK ST'* *'UNIVERSITY DR btwn COLONEL BY DR & LIBRARY RD'*
XCOORD YCOORD	Coordinates (longitude, latitude) where the collision happened.	Double Example: *367855.7101, 5031012.574*
ACCIDENT_ DATE	Date when the collision happened	Date in format: *YYYY-MM-DD* Example: *2017-06-30*
ACCIDENT_ TIME	Time when the collision happened	Time in format: *hh:mm AM/PM* Example: *12:30 PM* *11:50 AM*
ACCIDENT_LOCATION	*"Collision location is divided into two sub-categories, On Highway Codes 1 to 7 and 98 and Off Highway Codes 8 to 10 and 99. If collision occurred in a location which is not a public roadway, it is considered to be an Off Highway collision. The public roadway as defined under the HTA includes the roadway and shoulder."* [94] More details are available Appendix A.1.	***On Highway:*** *01 - Non intersection* *02 - Intersection related* *03 - At intersection* *04 - At/near private drive* *05 - At railway crossing* *06 - Underpass or tunnel* *07 - Overpass or bridge* *98 – Other* ***Off Highway:*** *08 - Trail* *09 - Frozen Lake or River* *10 - Parking Lot* *99 - Other*
CLASSIFICATION_ OF_ACCIDENT	*"The appropriate classification which describes the motor vehicle collision."* [94] More details are available in Appendix A.2.	*01 - Fatal injury* *02 - Non-fatal injury* *03 - P.D. only* *04 - Non-reportable* *99 – Other*

ENVIRONMENT_CONDITION	*"Designations are used to determine whether environmental conditions were a factor in the collision. Data is used to determine action for recurring conditions contributing to collisions."* [94] More details are available in Appendix A.3.	*00 – Unknown* *01 – Clear* *02 – Rain* *03 – Snow* *04 - Freezing rain* *05 - Drifting snow* *06 - Strong wind* *07 – Fog, mist, smoke, dust* *99 – Other*
LIGHT	*"The light conditions at the time of the collision are recorded to assess the need for artificial lighting and control devices."* [94] (MVCR 0304) More details are available in Appendix A.4.	*00 – Unknown* *01 – Daylight* *02 - Daylight, artificial* *03 - Dawn* *04 - Dawn, artificial* *05 - Dusk* *06 - Dusk, artificial* *07 - Dark* *08 - Dark, artificial* *99 – Other*
TRAFFIC_CONTROL	*"Identify any traffic control device at the collision scene. The device need not have been a factor in the collision. If more than one device exists, the traffic control device which had the greatest bearing on the collision is recorded.* *Speed limits and pavement markings are not traffic control devices for this field."* [94] More details are available in Appendix A.5.	*01 - Traffic signal* *02 - Stop sign* *03 - Yield sign* *04 - Ped. Crossover* *05 – Police control* *06 – School guard* *07 - School bus* *08 - Traffic gate* *09 - Traffic controller* *10 – No control* *11 – Roundabout* *99 - Other*
ROAD_SURFACE_CONDITION	*"The road surface condition at the collision site is recorded. For collisions occurring at or near an intersection, the condition for each road is entered. Where more than one condition applies, the most prevalent condition or the condition which had the greatest bearing on the collision is entered."* [94] More details are available in Appendix A.6.	*00 - Unknown* *01 - Dry* *02 - Wet* *03 - Loose snow* *04 - Slush* *05 - Packed snow* *06 - Ice* *07 - Mud* *08 - Loose sand or gravel* *09 - Spilled liquid*
IMPACT_TYPE	*"This field provides information to be retrieved from collision data systems to enable engineers to provide technical solutions unique to each selection. The code entered must best describe the general path of the vehicle (s) immediately before the first impact.* *Note: Where more than 1 impact occurred the first impact type is entered."* [94] More details are available in Appendix A.7.	*01 - Approaching* *02 - Angle* *03 - Rear end* *04 - Sideswipe* *05 - Turning movement* *06 - SMV unattended vehicle* *07 - SMV other* *99 - Other*

5.1.3.2. City of Ottawa Roads Segments Spatial Data Layer from *Open Data Ottawa*

The dataset contains a single polyline data showing operational roadways within geographic limits of City of Ottawa [79]. It includes a total of 26247 road segments with five features each, namely: name, subclass (i.e. collector, freeway, highway, etc., all subclasses are shown in Figure 5-5b), subtype (an integer from 1 to 9), length (the length of the road segment in km, i.e. 00031) and ID (i.e. __30305L*)*.



(a) (b)

Figure 5-5 Visualization using QGIS of road segments (a) in the city of Ottawa and (b) in the Carleton University area.

Figure 5-5a shows a spatial visualization of the roads segments using QGIS colored by subclass in the City of Ottawa, while Figure 5-5b shows the road segments in Carleton University area. Each road segment starts and ends in an intersection between at least two roads. An example of road segment (highlighted in red) is showed in Figure 5-6.

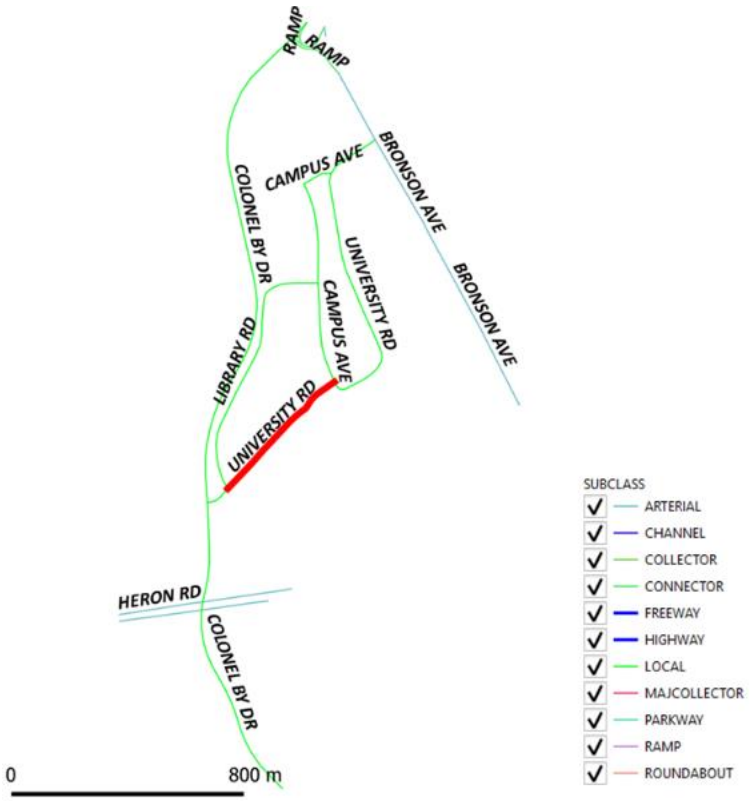


Figure 5-6 Road segment spatial visualization using QGIS (highlighted in red)

5.1.3.3. Ottawa Neighborhoods Spatial Data Layer from *Ottawa Neighborhood Study* (via *Open Data Ottawa*)

The ONS Neighborhood Boundaries were created by the Ottawa Neighborhood Study (ONS) to analyze population statistics and are not indicative of actual neighborhood limits [80]. The neighborhood boundaries were derived based on census tracts, physical and demographic similarities, physical barriers (e.g. waterways, highways, etc.), maps used by the real estate profession (e.g. the Ottawa Multiple Listing Service), consultations with community stakeholders, as well as fieldwork by ONS researchers. This spatial data layer is used by the Business Performance Section of the Ottawa Police Service as part of their analysis and deployment of personnel. It includes a total of 108 neighborhoods with two features each: name (i.e. Carleton University) and ID (an integer from 1 to 108). Figure 5-7a illustrates a spatial

visualization of the neighborhoods using QGIS in the City of Ottawa, while Figure 5-7b shows the neighborhoods around Carleton University area.

(a) (b)

Figure 5-7 Visualization using QGIS of neighborhoods in (a) City of Ottawa and (b) in the Carleton University area

5.1.3.4. Ottawa City Limits Spatial Data Layer from *Open Street Maps*

The limit boundaries of the city of Ottawa were obtained from Open Street Maps [91]. Figure 5-8 shows a spatial visualization of the City of Ottawa limits using QGIS.

Figure 5-8 City of Ottawa limits spatial visualization using QGIS

This spatial layer will be used later in section 5.3.2.2 to create a series of tiles over the City of Ottawa as part of the District/Neighborhood feature generation.

5.1.3.5. University Calendar from *Carleton University*

The Carleton University calendar (i.e. dates) [95] for the fall, summer and winter terms was used as part of the social events considered. The features used in the analysis are shown in the Table 5-2.

Table 5-2 Carleton University Calendar feature names, description and example values

Feature Name	Feature Description	Value Type / Example
CARLETON_CALENDAR_FALL CARLETON_CALENDAR_WINTER CARLETON_CALENDAR_SUMMER	The date ranges of the Carleton University term, including start, end and holidays (weekdays) dates during the term.	Date in format: *YYYY-MM-DD* Example: *Fall 2015* *START: 9/2/2015* *END: 12/7/2015* *HOLIDAYS:* *9/7/2015,* *10/12/2015,* *10/26/2015,* *10/27/2015,* *10/28/2015,* *10/29/2015,* *10/30/2015*

5.1.3.6. City of Ottawa Major Sport Events and Holidays from *Ottawa Tourism*

A selected group of the major sports events (in form of dates) including: hockey (Ottawa Senators, Ottawa 67s), football (Ottawa RedBlacks, Carleton Ravens, uOttawa Gee Gees), soccer (Ottawa Fury FC) and baseball (Ottawa Champions BBC); as well as the city holidays were used as additional information regarding the social events considered [92]. Details are shown in Table 5-3.

Table 5-3 City of Ottawa major sport events and holidays

Feature Name	Feature Description	Value Type / Example
OTTAWA_FURY_SOCCER	The dates when the Fury Soccer Team played a game in the City of Ottawa (TD Place Arena)	Date in format: *YYYY-MM-DD* Example: *4/18/2015*

OTTAWA_SENATORS_HOCKEY	The dates when the Senators Hockey Team played a game in Ottawa (Canadian Tire Centre Arena)	Date in format: *YYYY-MM-DD* Example: *1/4/2015*
OTTAWA_67S_HOCKEY	The dates when the 67s Hockey Team played a game in Ottawa (TD Place Arena)	Date in format: *YYYY-MM-DD* Example: *1/6/2015*
OTTAWA_REDBLACKS_FOOTBALL	The dates when the Redblacks Football Team played a game in Ottawa (TD Place Arena)	Date in format: *YYYY-MM-DD* Example: *6/8/2015*
OTTAWA_RAVENS_FOOTBALL	The dates when the Carleton Ravens Football Team played a game in Ottawa (MNP Park Arena, TD Place Arena – Panda Game)	Date in format: *YYYY-MM-DD* Example: *9/6/2015*
OTTAWA_GEEGEES_FOOTBALL	The dates when the uOttawa Gee-Gees Football Team played a game in Ottawa (Minto Sport Complex Arena)	Date in format: *YYYY-MM-DD* Example: *9/6/2015*
OTTAWA_CHAMPIONS_BBC	The dates when the Champions Baseball Team played a game in Ottawa (RCGT Park Arena)	Date in format: *YYYY-MM-DD* Example: *5/22/2015*
OTTAWA_STATUTORY_HOLIDAYS	The dates when a statutory holiday happened in Ottawa	Date in format: *YYYY-MM-DD* Example: *1/1/2015, #New Year's Day* *2/16/2015, #Family Day* *4/3/2015, #Good Friday* *5/18/2015, #Victoria Day* *7/1/2015, #Canada Day* *9/7/2015, #Labour Day* *10/12/2015, #Thanksgiving* *12/25/2015, #Christmas Day* *12/26/2015 #Boxing Day*

5.1.4. Data Exploration

In this part of the work, a series of visualization graphs were created for the main datasets used: (1) main collision dataset, (2) roads segments geospatial layer and (3) neighborhoods geospatial layer. Visualizations help to obtain very important insights about the data, for example to identify trends and the important features that could

have a contribution to accident collision rates. Also, they provide important information to be used as part of the multiplicative correction factor.

5.1.4.1. Main Collision Dataset

In the following visualizations, the features of the main collision dataset are classified in six categories: (1) date/time, (2) location, (3) environment, (4) road, (5) traffic, and (6) accident/collision.

Date/Time Related (ACCIDENT_DATE / ACCIDENT_TIME)

Figure 5-9 shows the number of records according to different date/time features (by year, month, weekday and hour).

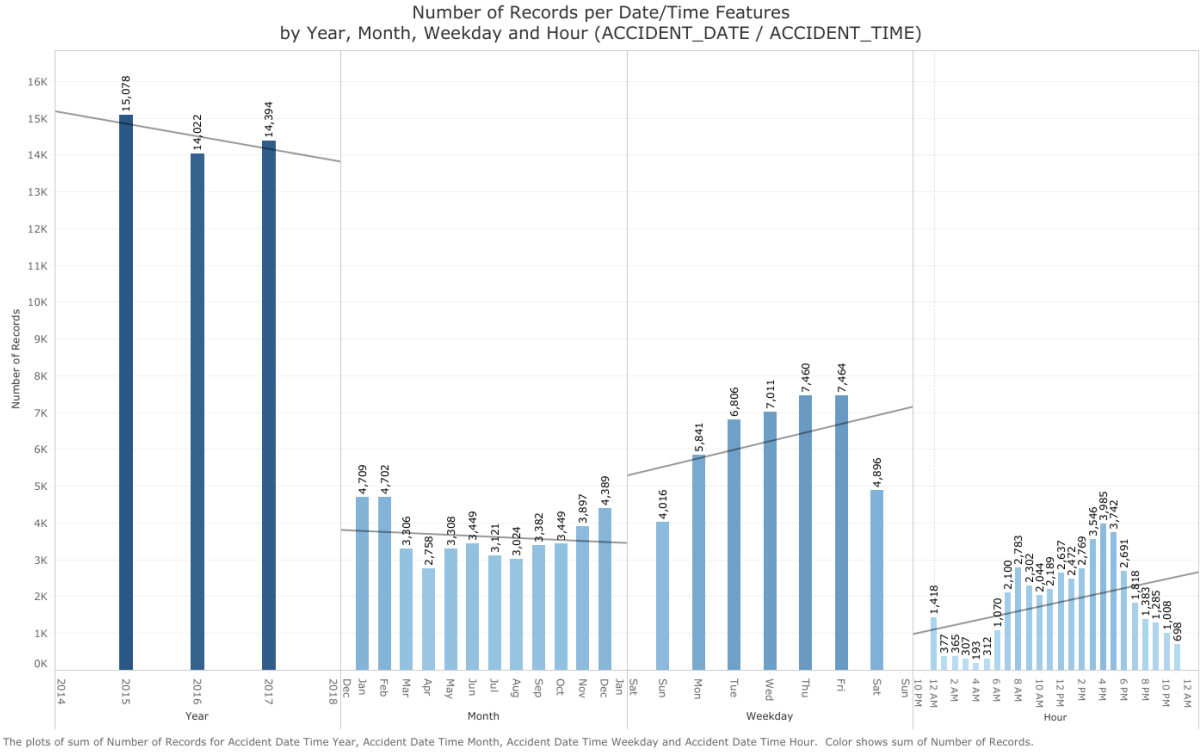


Figure 5-9 Number of records by year, month, weekday and hour (ACCIDENT_DATE / ACCIDENT_TIME)

We can notice that (1) the number of records per year shows a decreasing tendency; (2) the months with the highest/lowest number of records are *December, January* and *February* (the top-3) / *April, July* and *September* (the bottom-3); where *December* is the most dangerous and *April* is the safest one; (3) the number of records

increases from *Monday* to *Friday*, where the most dangerous days are *Thursday* and *Friday*; the day with the lowest number of records (the safest one) is *Sunday*, (4) the highest number of accidents happens between *7am-9am* in the morning and between *3pm-6pm* in the afternoon. These are observations fully aligned with the previous findings in the literature described in section 2.2.

Figure 5-10 shows the number of records by month colored by year. We can notice that: (1) the monthly distribution from *March* to *October* is almost the same; (2) from *November* to *January* there is an important increase and then a big decrease from *February* to *April*. These findings are aligned with the ones in the literature.

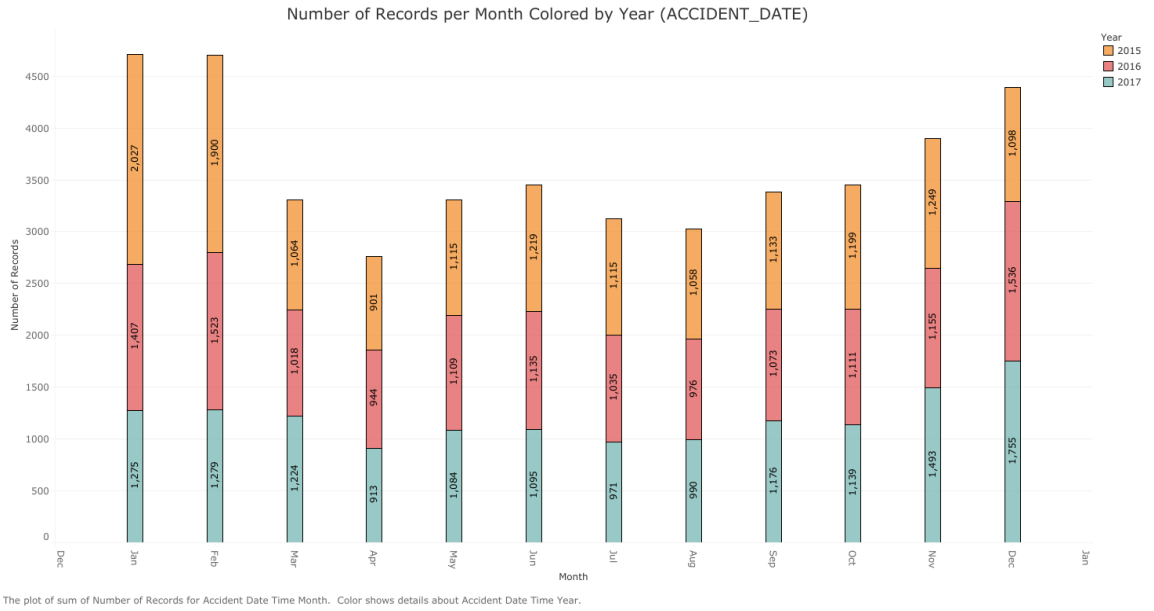


Figure 5-10 Number of records per month colored by year (ACCIDENT_DATE)

Figure 5-11 illustrates the number of records per week of the year. We can notice that the weeks of the year with the highest number of records are: 2 (middle of January), 7-8 (middle to end of February), and 50-51 (middle to end of December), while 53 (end of December to beginning of January), 1 (end of December to beginning of January), 17 (end of April) and 32 (beginning of August) have the lowest number of records. It is important to mention that the year 2015 is the only one that has 53 weeks, this explains why the week 53 is one of the lowest ones in terms of records.

The number of records by day of the month is illustrated in Figure 5-12. It can be observed that the days of the month with the highest number of records are: 14,

17 and 21 (top-3); while 31, 10, and 25 (bottom-3) are the ones with the lowest. Fewer months of the year have 31 days; this explains why 31 has the lowest number of records.

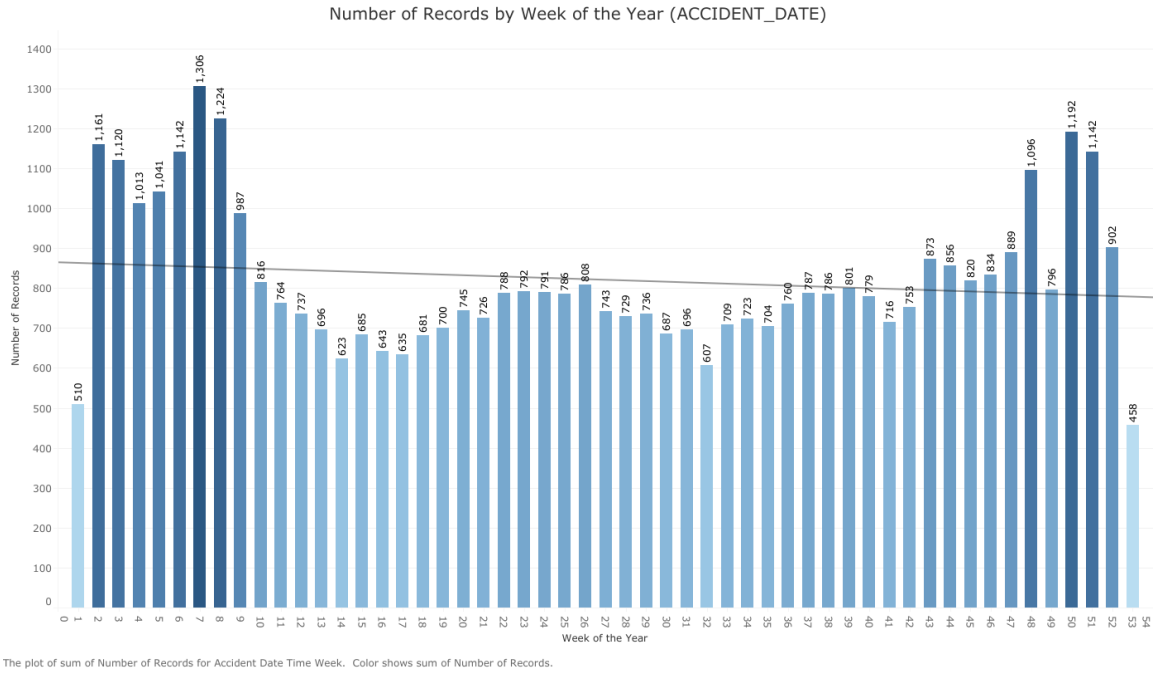


Figure 5-11 Number of records by week of the year (ACCIDENT_DATE)

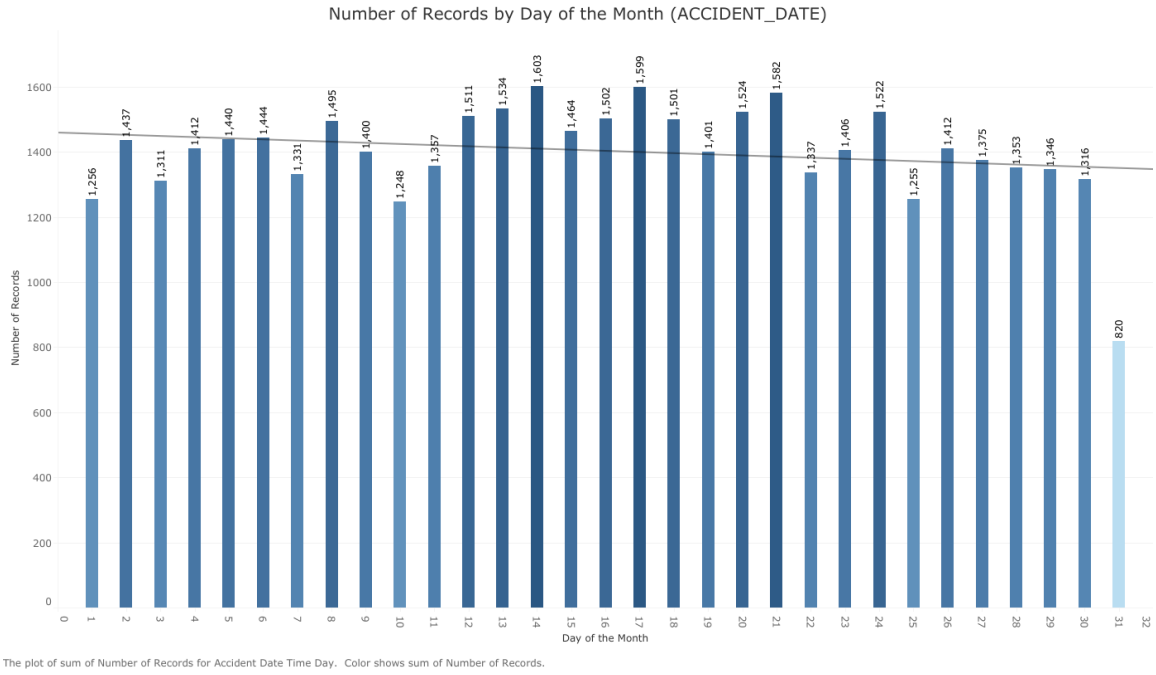


Figure 5-12 Number of records by day of the month (ACCIDENT_DATE)

A more detailed view of the records by hour of the day is shown in Figure 5-13. One can notice that (1) the hours of the day with the highest number of records are between 8 a.m.-9 a.m. and 3 p.m.-6 p.m., (2) after 7 p.m. the number of records is decreasing and (3) an important number of records appear at late night between 12 a.m.-1 a.m. These observations are supporting the previous findings in the literature.

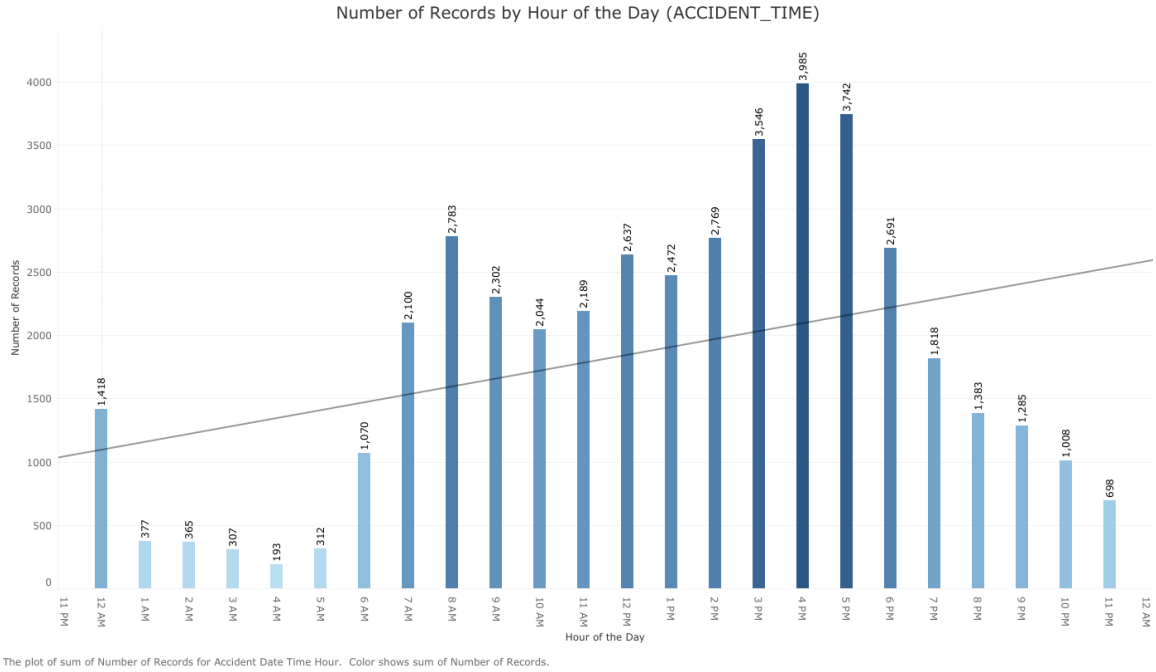


Figure 5-13 Number of records by hour of the day (ACCIDENT_TIME)

Location Related (LOCATION / XCOORD / YCOORD / ACCIDENT_LOCATION)

The dataset includes a total of 10283 different locations (LOCATION). To validate if some of the locations are more collision prone in comparison to others, we constructed a graph showing the distribution of the records by frequency. Figure 5-14 presents a Pareto diagram in which the *y*-axis contains some of the locations (picked up randomly) sorted by frequency from bottom to top, with the maximum frequency (167) at the bottom (*'ST. JOSEPH BLVD @ JEANNE D'ARC BLVD',* located at zero); and a double *x*-axis with the first at the bottom representing the frequency – Count (LOCATION) – identified by a light blue line, and the second axis at the top representing the total percentage of accumulative records, identified by a green line.

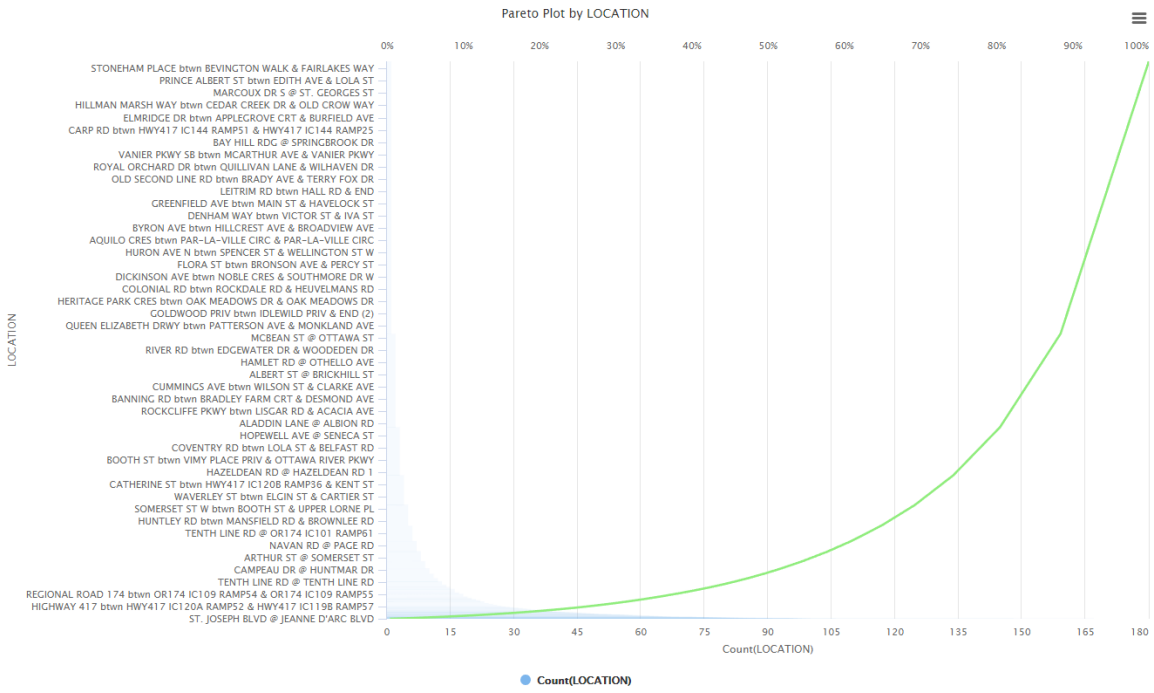


Figure 5-14 Pareto plot by location (LOCATION)

It can be observed that: (1) almost 25% of the total number of records are related with locations that have a frequency of at least 45 (top-25% locations by frequency) and (2) more than 40% of the total records are related with locations that have a frequency less than 7 (bottom-40% locations by frequency). Also, in terms of the total number of locations, the *top-25% locations by frequency* represents less than 5% of the records and (4) the *bottom-40% locations by frequency* represents more than 85% of the records. In short, 5% of the total locations (approximately 515) represents almost the 25% of the total number of records.

Figure 5-15 shows the top-30 locations by frequency. We can notice that *'ST. JOSEPH BLVD @ JEANNE D'ARC BLVD'* and '*HUNT CLUB RD @ RIVERSIDE DR*' are the top-2 most dangerous locations, and on the top-30 list, seven locations are related with the *Highway 417*.

Top-30 Records by Location (LOCATION)

Location	
ST. JOSEPH BLVD @ JEANNE D'ARC BLVD	167
HUNT CLUB RD @ RIVERSIDE DR	148
INNES RD @ TENTH LINE RD	106
PRINCE OF WALES DR @ WEST HUNT CLUB RD	100
BASELINE RD @ WOODROFFE AVE	99
WEST HUNT CLUB RD @ WOODROFFE AVE	98
HAWTHORNE RD @ HUNT CLUB RD	97
HIGHWAY 417 btwn HWY417 IC122 RAMP61 & HWY417 IC121B RAMP16	96
DONALD ST @ ST. LAURENT BLVD	95
HIGHWAY 417 btwn HWY417 IC127 RAMP25 & HWY417 IC126 RAMP51	91
CYRVILLE RD @ INNES RD	90
HIGHWAY 417 btwn HWY417 IC118 RAMP67 & HWY417 IC117 RAMP36	87
BANK ST @ HUNT CLUB RD	85
BLAIR RD @ INNES RD	85
HIGHWAY 417 btwn HWY417 IC126 RAMP61 & HWY417 IC124 RAMP76	84
ST. JOSEPH BLVD/OLD MONTREAL RD @ TRIM RD	83
MEADOWLANDS DR @ MERIVALE RD	82
RIVERSIDE DR @ TREMBLAY RD/HWY417 IC117 RAMP52	82
HIGHWAY 417 btwn HWY417 IC124 RAMP65 & HWY417 IC122 RAMP51	81
INDUSTRIAL AVE/INNES RD @ ST. LAURENT BLVD	81
BANK ST @ HERON RD	80
BELFAST RD @ ST. LAURENT BLVD	79
INDUSTRIAL AVE @ RIVERSIDE DR	78
MERIVALE RD @ WEST HUNT CLUB RD	77
HIGHWAY 417 btwn HWY417 IC117 RAMP51 & HWY417 IC117 RAMP35	76
MONTREAL RD @ VANIER PKWY	76
ST. LAURENT BLVD @ COVENTRY RD/OGILVIE RD	76
BASELINE RD @ CLYDE AVE	75
HIGHWAY 417 btwn HWY417 IC119A RAMP75 & HWY417 IC118 RAMP57	75
LAURIER AVE @ NICHOLAS ST	75

Sum of Number of Records broken down by Location. The view is filtered on Location, which keeps 30 of 6,161 members.

Figure 5-15 Top-30 location records by frequency (LOCATION)

Figure 5-16 and Figure 5-17 illustrate the previous top-30 location records by frequency, colored by year and month. One can observe a big increase in the number of records related with the location '*HIGHWAY 417 btwn HWY417 IC122 RAMP61 & HWY417 IC121B RAMP16*' (from 2015=17, 2016=35 to 2017=44) and a big decrease in *'HIGHWAY 417 btwn HWY417 IC118 RAMP57 & HWY417 IC118 RAMP35*' (from 2015=33, 2016=8 to 2017=8). As well, a big increase in the number of records takes place for the top location *'ST. JOSEPH BLVD @ JEANNE D'ARC BLVD'* between the months *April, May* and *June*.

Because each record includes the coordinates, we can use it to create a series of maps visualizations. Figure 5-18 presents a map of the records by coordinates and Figure 5-19 illustrates the top record coordinates by collision frequency. It is important to mention that a collision could have happened at the same physical location (i.e. an intersection) but could have slightly different coordinate values as a result of the collision reporting process (i.e. an officer needs to fill out the information and introduce the collision coordinates). Because of this, a spatial matching process

between the coordinates and the roads segments must be implemented to increase the prediction power of the machine learning algorithm.

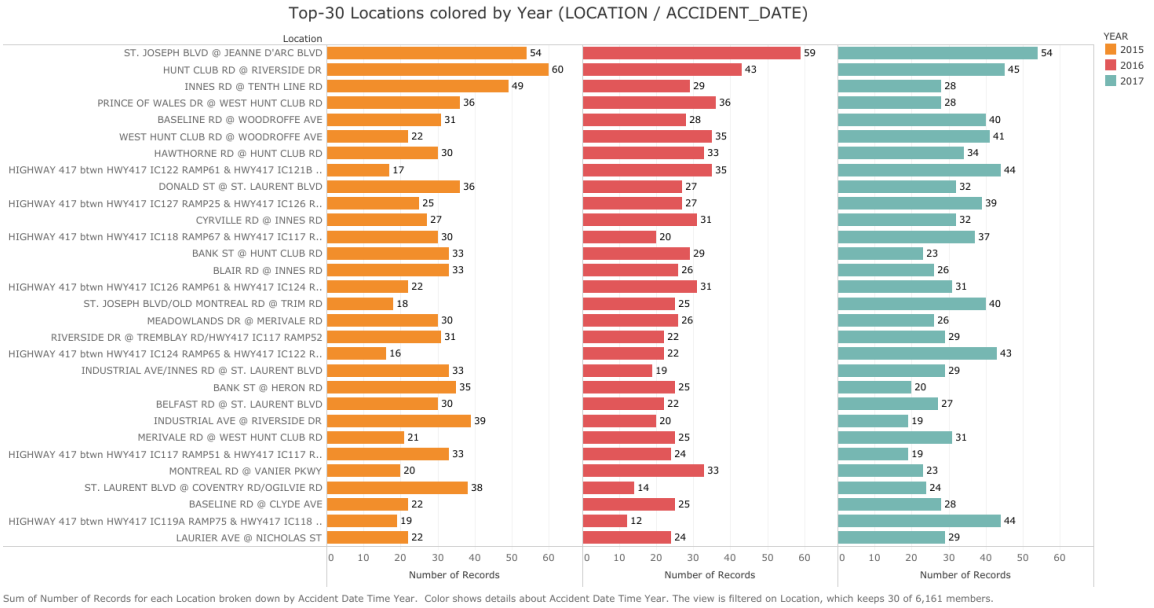


Figure 5-16 Top-30 locations by frequency colored by year (LOCATION / ACCIDENT_DATE)

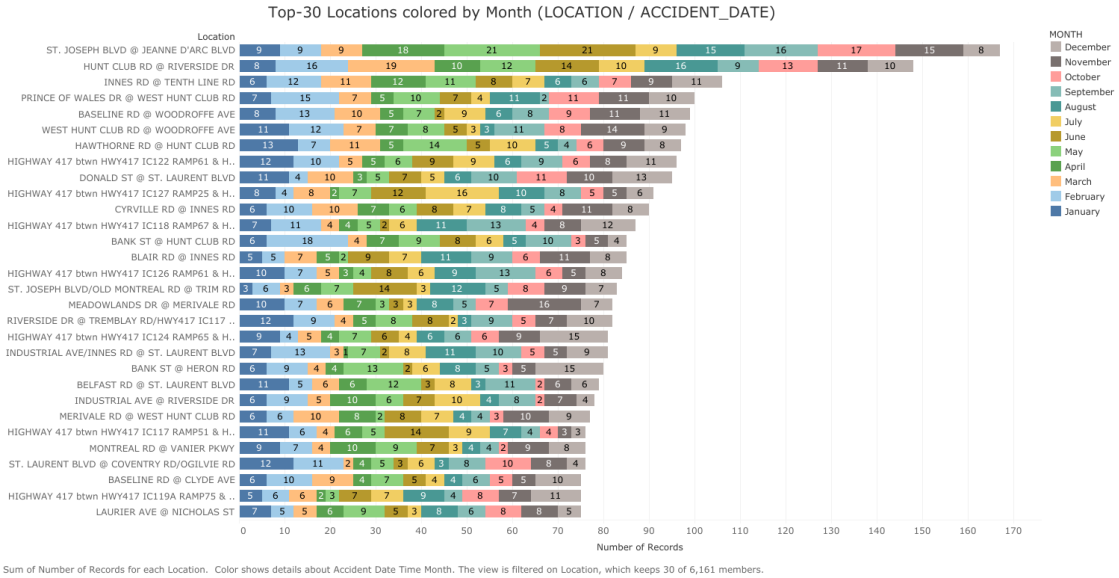


Figure 5-17 Top-30 locations by frequency colored by month (LOCATION / ACCIDENT_DATE)

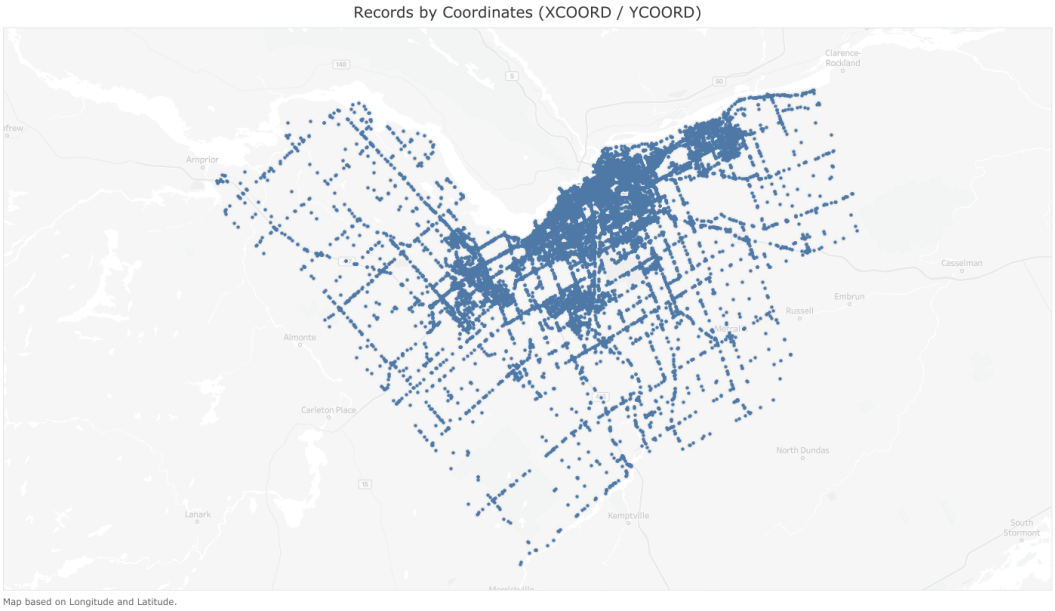


Figure 5-18 Map of records by coordinates (XCOORD / YCOORD)

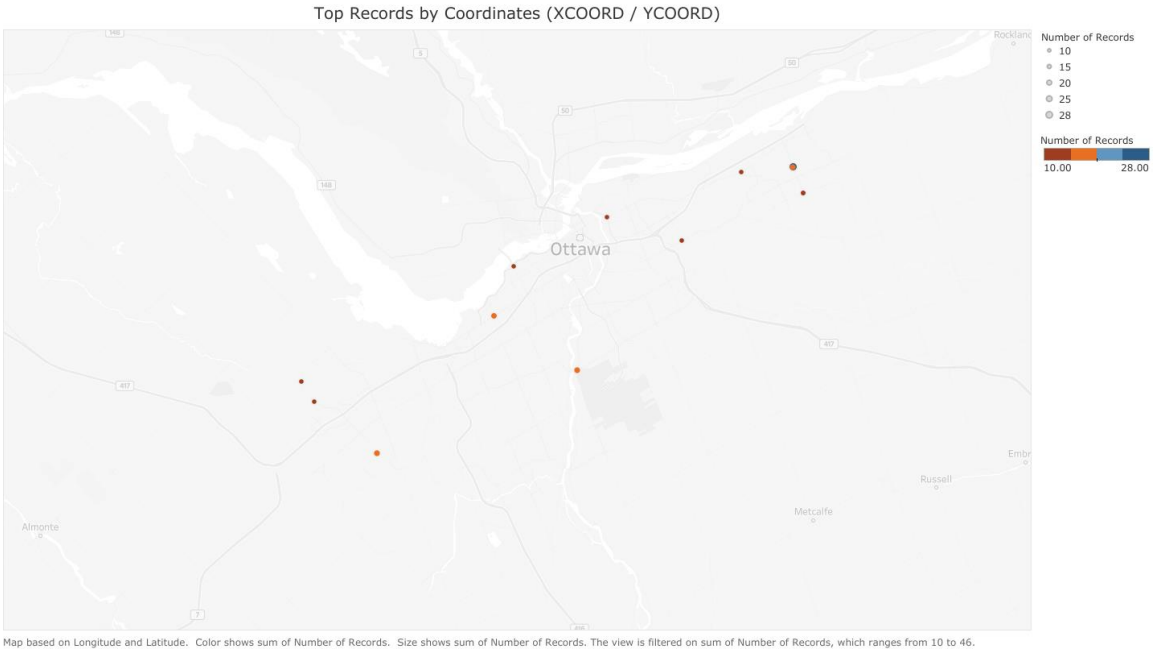


Figure 5-19 Top records by coordinates (XCOORD / YCOORD)

In terms of the accident location, Figure 5-20 depicts the distribution of the number of records. We can notice that: (1) the largest number of records is related with *'01 – Non Intersection'* and *'02 – Intersection Related'*; (2) considering together *'02 – Intersection Related'* and *'03 – At Intersection'* as a collision related with an

intersection, such collisions represent more than 50% of the total samples (a more detailed description about each value can be found in Appendix A.1).

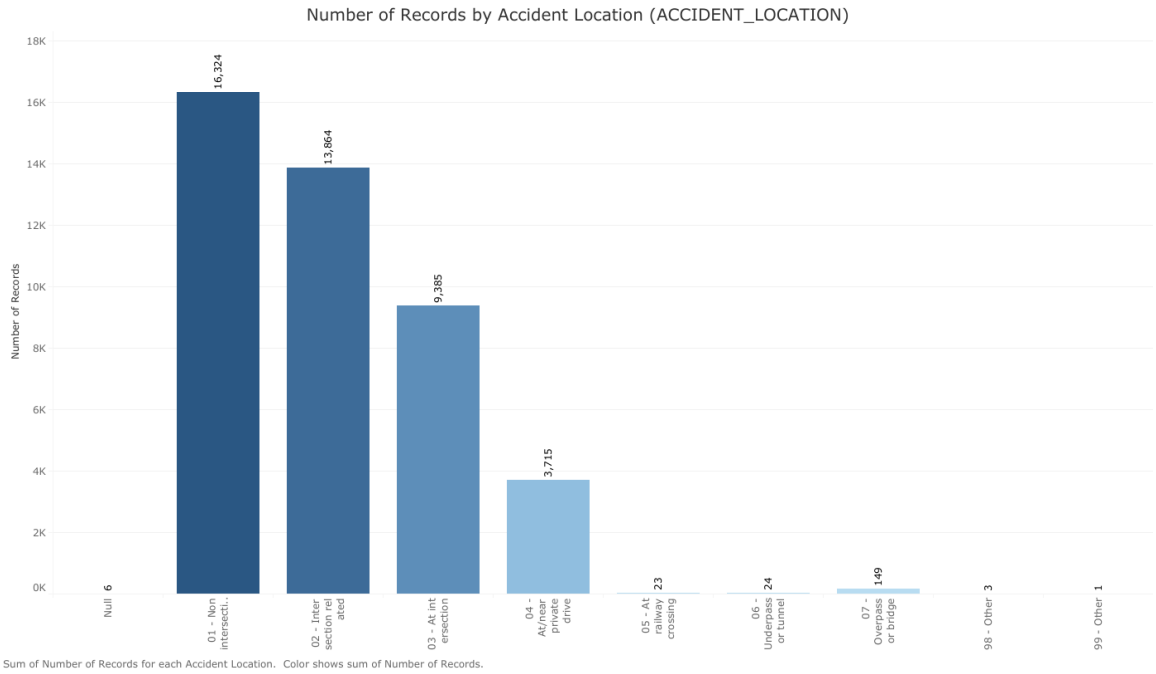


Figure 5-20 Number of records by accident location (ACCIDENT_LOCATION)

These visualizations identify the most dangerous locations, and can be used to build a series of map visualizations, as we will do in a later part of the thesis, that could be used for proactive traffic intervention.

Environment Related (ENVIRONMENT_CONDITION / LIGHT)

Additional visualizations were performed aimed at identifying trends related to the environment conditions. Considering both environment variables together (i.e. ENVIRONMENT_CONDITION and LIGHT), Figure 5-21 supports the previous findings in the literature described in section 2.1, in which the largest amount of collisions occurred under favorable environment conditions (i.e. daylight and clear environment). Interestingly, the snow and rain combined represent less than 20% of the total number of records (a more detailed description about each value of

environment and light condition variables can be found in Appendix A.3 and Appendix A.4).

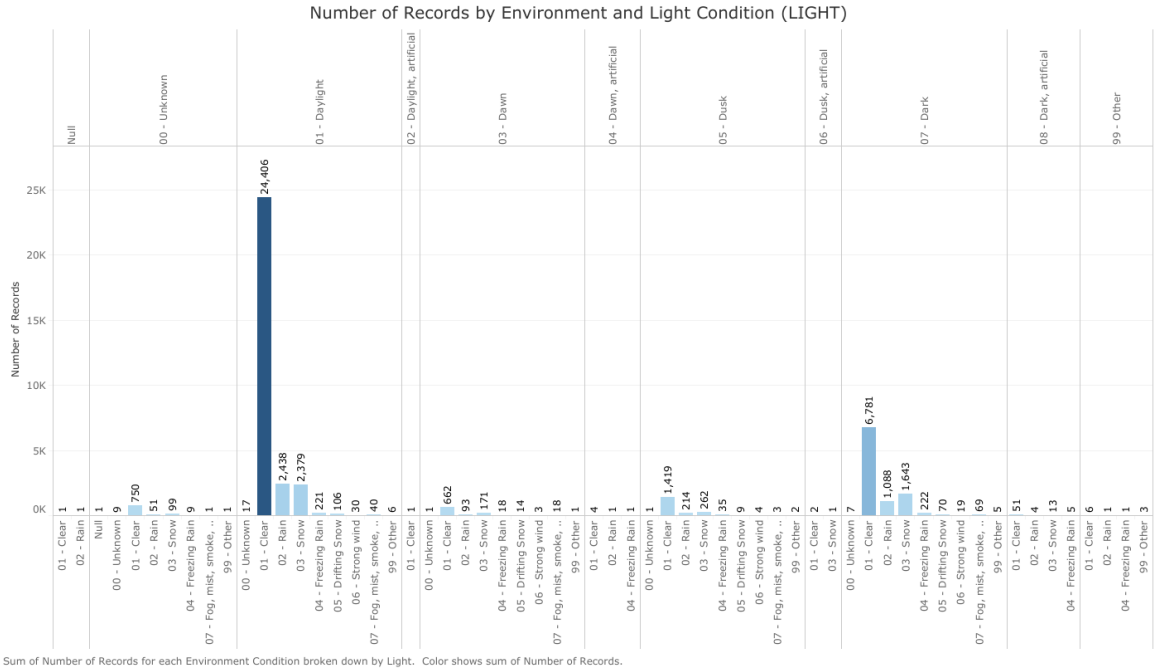


Figure 5-21 Number of records by environment and light conditions (ENVIRONMENT_CONDITION / LIGHT)

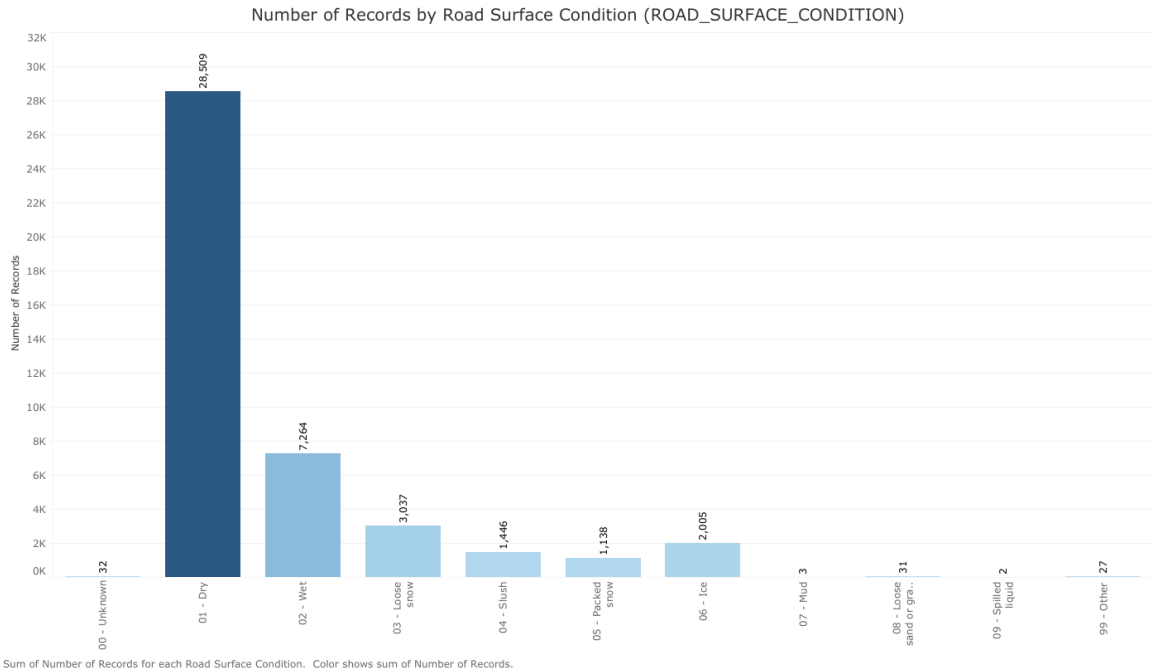


Figure 5-22 Number of records by road surface condition (ROAD_SURFACE_CONDITION)

Road Related (ROAD_SURFACE_CONDITION)

In order to validate if some trends exist related to the road conditions, Figure 5-22 shows the number of records by road surface condition. We can notice that the largest number of collisions occurred in good road conditions (*'01 – Dry'*). Again, unfavorable conditions such as snow *('03 – Loose snow', '04-Slush', '05 – Packet Snow'* and *'06 – Ice')* and rain *('02 – Wet')* account for less than 10% each one of the total number of records (a more detailed description about each value can be found in Appendix A.6). These observations are aligned with the findings in the literature described in section 2.1.

Traffic Related (TRAFFIC_CONTROL)

The traffic control elements (i.e. traffic signals, stop sign, yield sign, etc.) play an important role in regulating, guiding and warning users about road conditions. In order to identify a tendency related to these elements, Figure 5-23 shows the number of records by traffic control condition. We can notice that the largest number of traffic accidents occurred without traffic control (*'10 – No control*), followed very close by traffic signals (*'01 – Traffic Signal'*). As such, there is not a clear trend regarding traffic control. A more detailed description about each value of the TRAFFIC_CONTROL variable can be found in Appendix A.5.

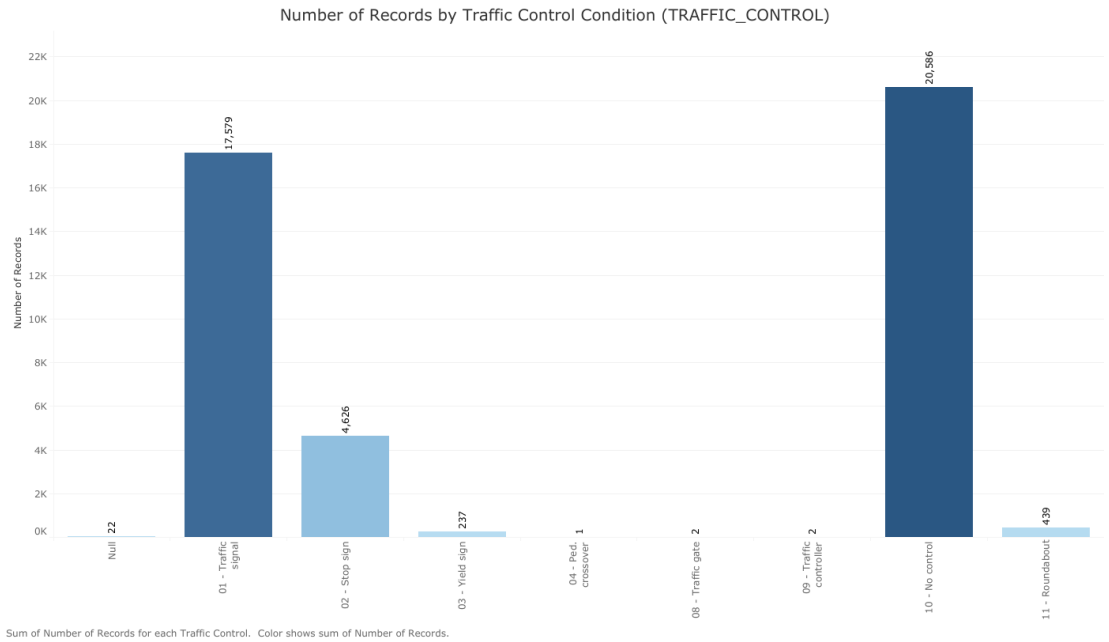


Figure 5-23 Number of records by traffic control (TRAFFIC_CONTROL)

Accident Related (INITIAL_IMPACT_TYPE / CLASSIFICATION_OF_ACCIDENT)

Figure 5-24 shows the number of records by initial impact type. We can notice that the biggest number of records happened by rear end (*'03 – Rear end'*) followed by single motor vehicle - other (*'07 – SMV other')* and with an almost equal distribution between the following three (*'02 – Angle', '04 – Sideswipe'* and *'05 – Turning movement')*. It is interesting to notice that a considerable amount of records are related to unattended vehicles (*'06 – SMV unattended vehicles')*. A detailed description about each value of this variable is presented in Appendix A.7.

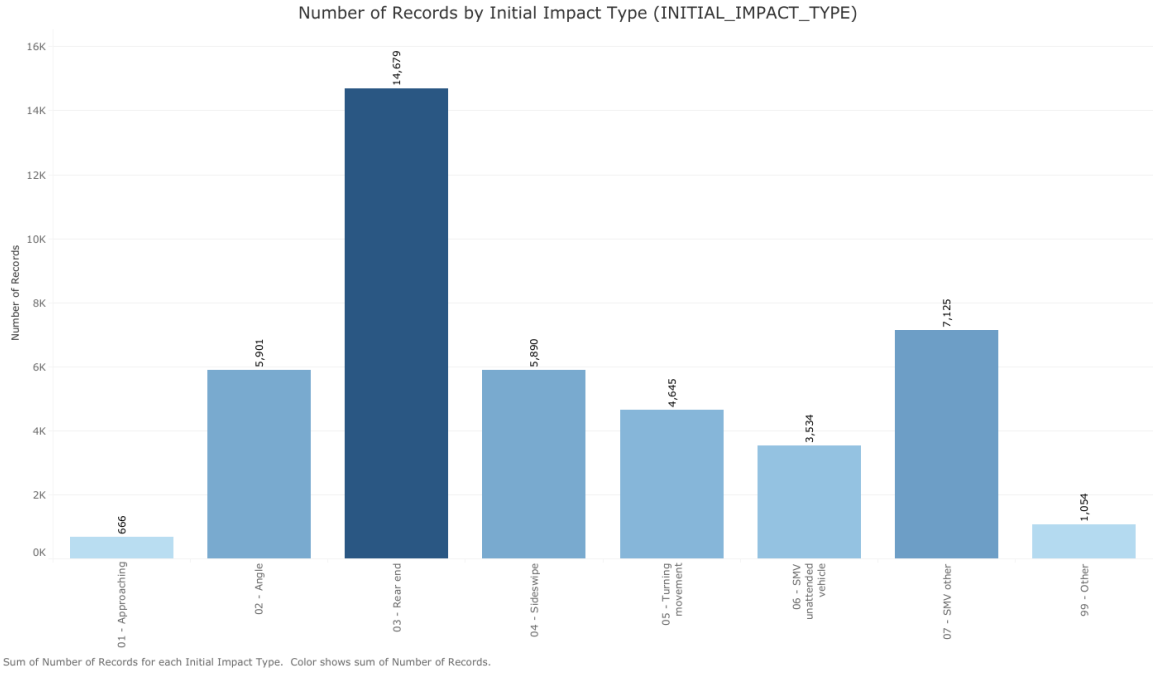


Figure 5-24 Number of records by initial impact type (INITIAL_IMPACT_TYPE)

In terms of the classification of accident, as one can notice in Figure 5-25, the largest number of accidents produce only property damage (*'03 – P.D. only'*), with only 99 from 43494 resulting in a fatal injury (*'01 – Fatal injury'*). Each of these variables are described in Appendix A.2. Is important to mention that, because the accident related variables become available after a collision takes place, these variables are removed from the dataset for the prediction of accident frequency (accident/no accident). For the prediction of accident severity, the variable CLASSIFICATION_OF_ACCIDENT will be used as a target variable.

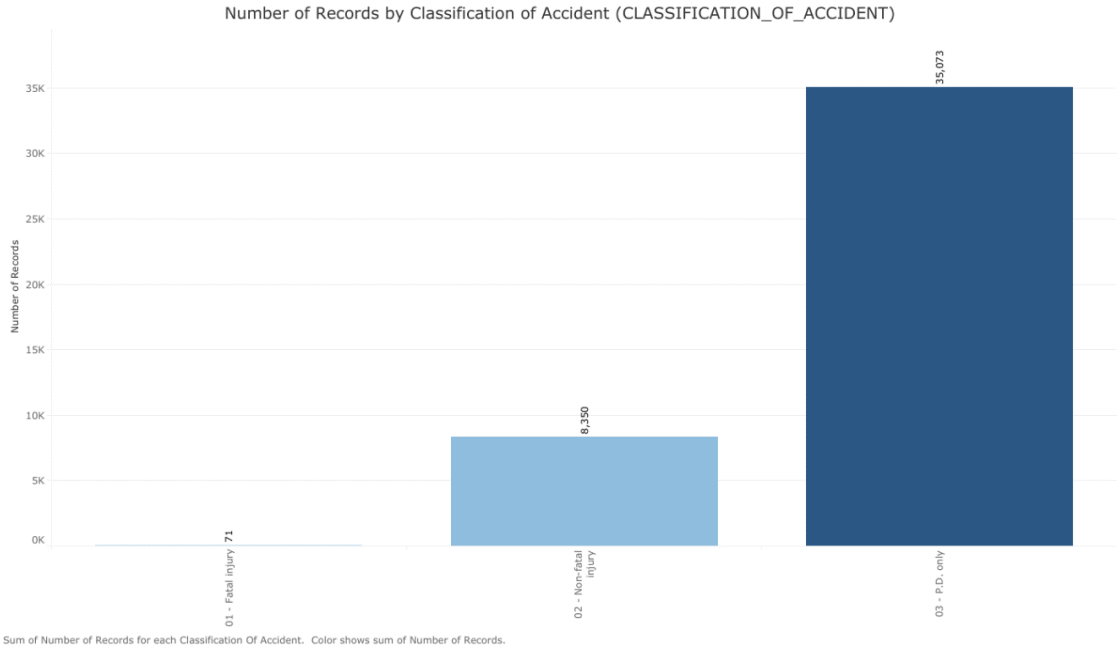


Figure 5-25 Number of records by accident type (CLASSIFICATION_OF_ACCIDENT)

5.1.4.2. Roads Segment Geospatial Layer

As it was mentioned in section 5.1.3, a spatial data layer with the road segments was added. Figure 5-26 and Figure 5-27 illustrate a map with the road segments colored by type and subclass, where all the subclass and road type are shown in the legend.

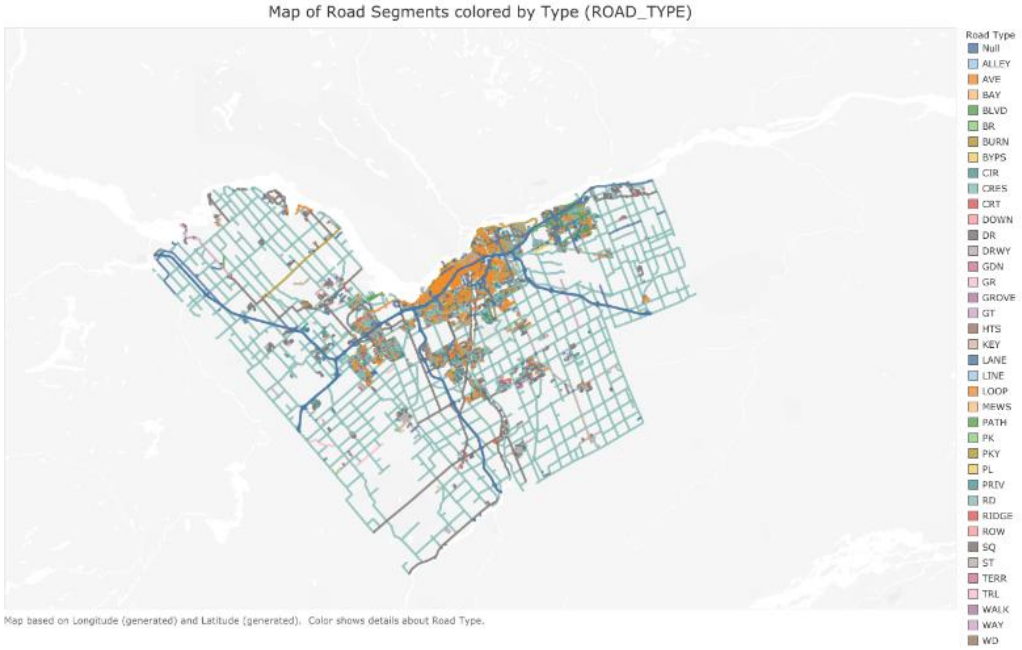


Figure 5-26 Map of road segments colored by type (ROAD_TYPE)

It is important to notice that some road segments have a *Null* road type (Figure 5-26). As well, some road segments belong to the public transport system (TRANSITWAY and BUSONLY in Figure 5-27).

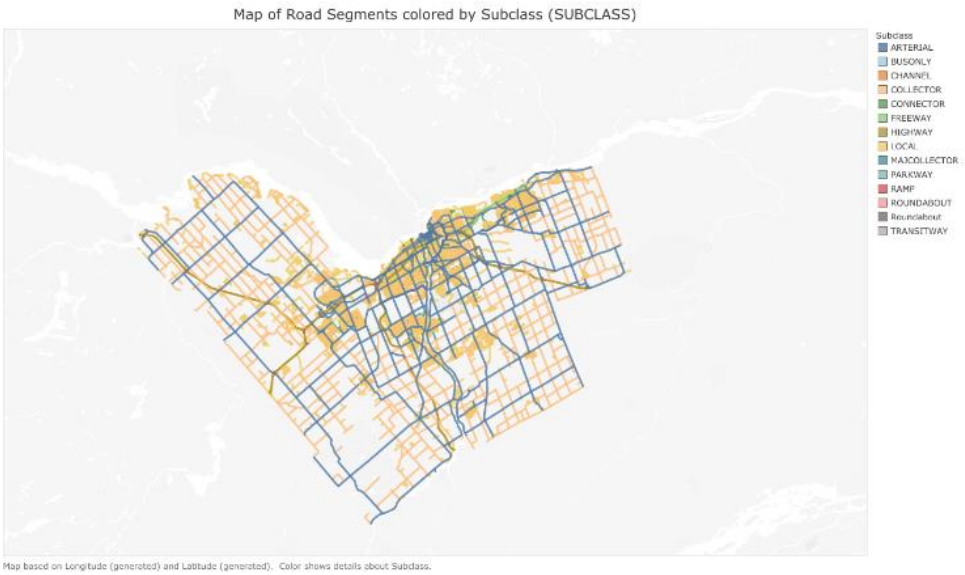


Figure 5-27 Map of road segments colored by subclass (SUBCLASS)

5.1.4.3. Neighborhoods Geospatial Layer

Additionally, a neighborhood geospatial layer used by the Business Performance Section of the Ottawa Police Service was added for analysis and prediction purposes. Figure 5-28 shows a map of Ottawa with its different neighborhoods.

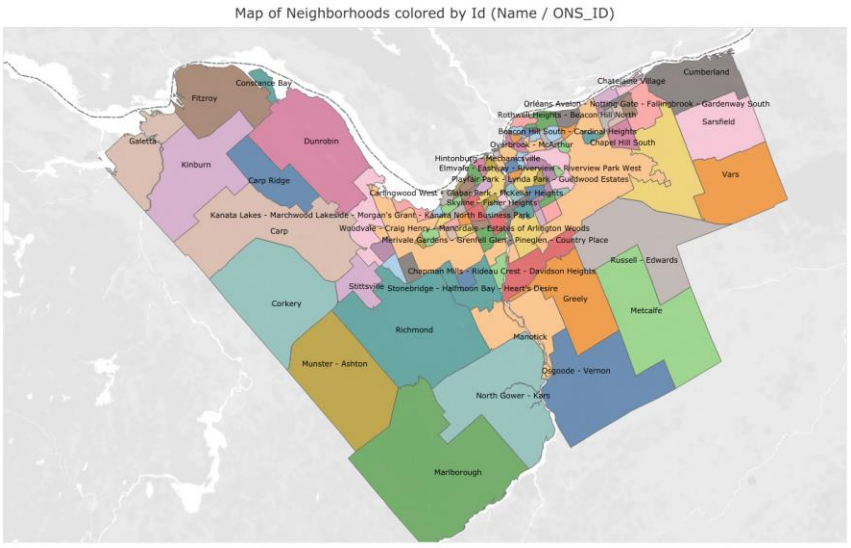


Figure 5-28 Map of neighborhoods colored by ID (Name / ONS_ID)

5.1.4.4. Most Important Statistical Factors

As part of the proposed framework and part of the objectives of this work, a series of statistics trends that can be used to create a multiplicative correction factor was also identified. Based on the previous visualizations, the main factors to be considered are: (1) hour of the day: a clear trend was identified in the morning 8 a.m. to 9 a.m.; and in the afternoon; 3 p.m. to 6 p.m. rush hour, (2) day of the week: Thursday and Friday were shown to be more accident prone and (3) week of the year: weeks 7,8, 48, 51 and 52 are also more accident prone. Using a similar approach to [96], the prediction probabilities for the records that match the previous conditions will be increased to reflect these statistics trends. This process is described in more detail in section 6.4.

5.2. Data Quality Verification

As part of the data methodology, the data quality verification phase includes verifying the dataset against missing attributes (*null/empty* values), special characters (i.e. with different character encoding schemes) and redundant attributes (i.e. same attribute with different lowercase / uppercase representation). This step will provide the necessary inputs to define the strategy for the next phase: data preparation.

5.2.1.1. Main Collision Dataset

Figure 5-29 illustrates general statistics on the main collision features. As we can notice, only few features (ENVIRONMENT_CONDITION, TRAFFIC_CONTROL, ACCIDENT_LOCATION and LIGHT) have missing (*Null*) values. Because the number is very small (maximum 31 records considering that they are not related) compared with the total number of records (43494), the strategy to handle it will be to simply delete them from the dataset.

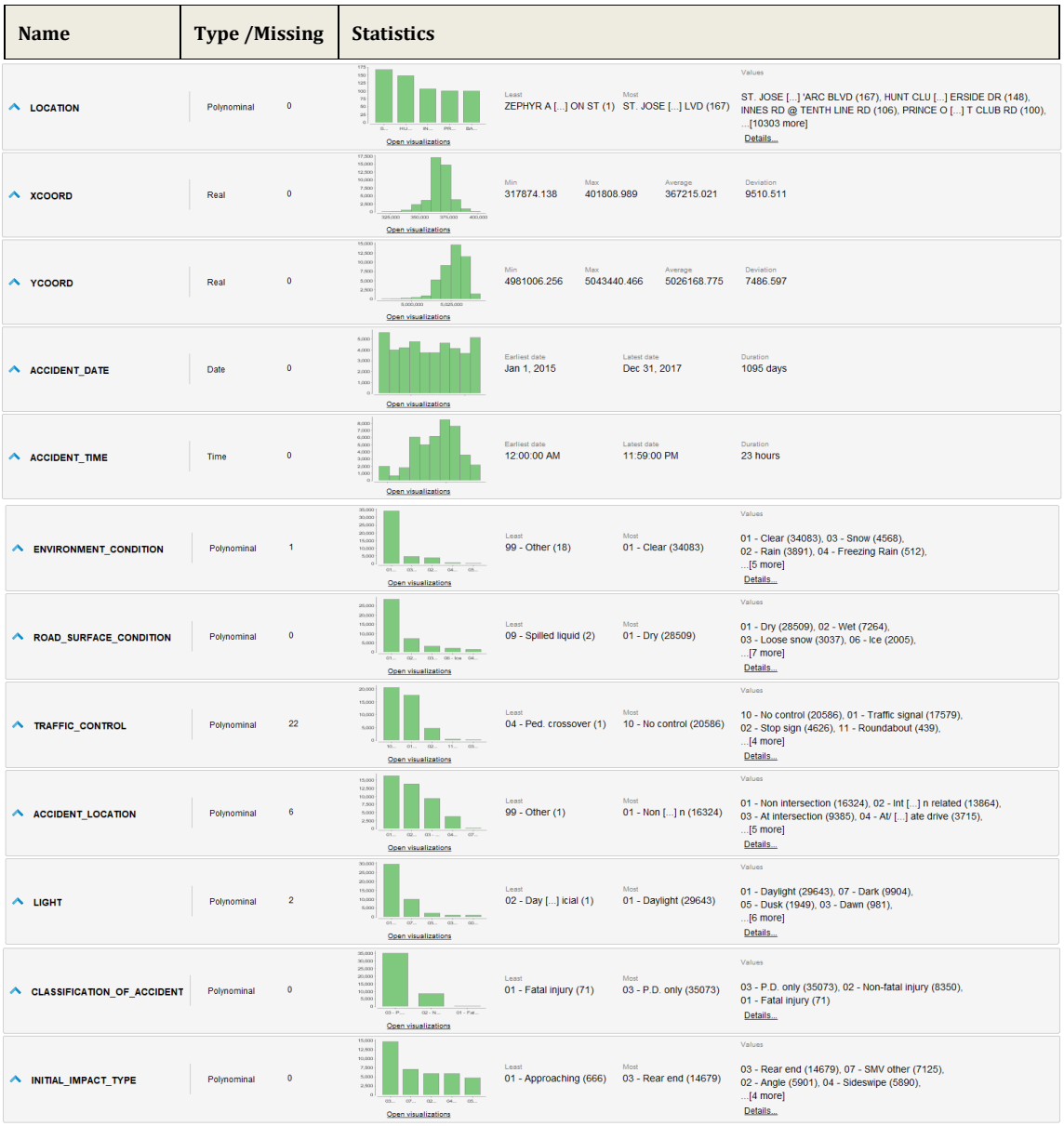


Figure 5-29 General statistics on the main collision dataset features

5.2.1.2. Road Segments Geospatial Layer

The general statistics about the road segment geospatial layer features are shown in Figure 5-30. We can notice as well that some features (i.e. ROAD_NAME and ROAD_TYPE) have missing (*Null*) values. Because the number is important (at least 1456) compared with the total (26675), it is necessary to define a strategy to handle this issue. In this case, we will fill the missing values.

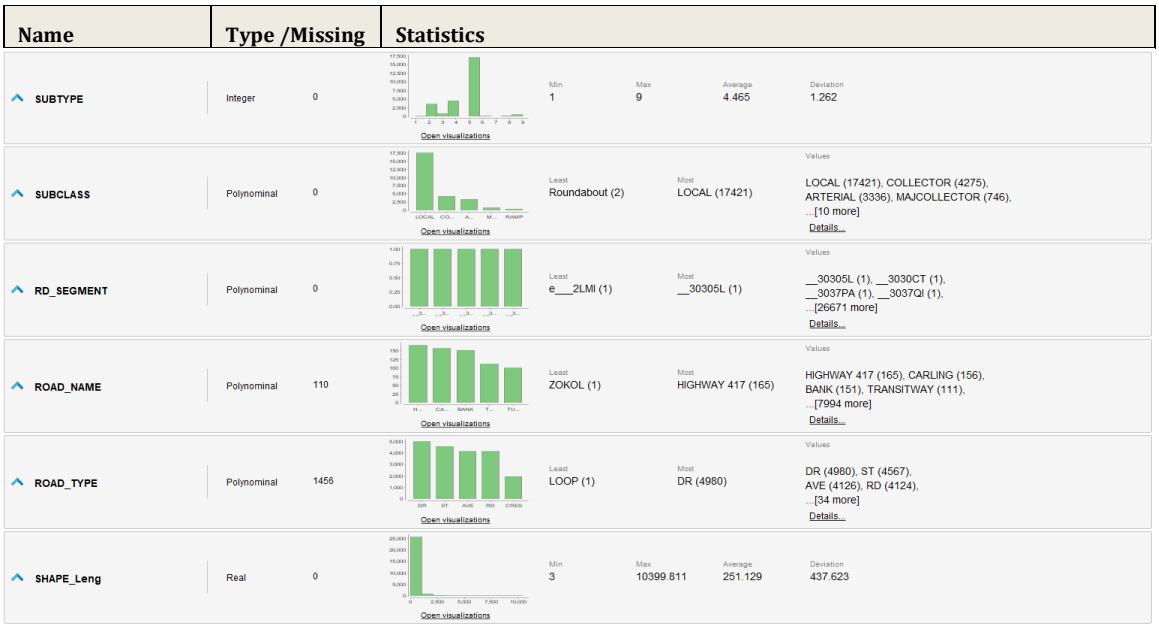


Figure 5-30 General statistics on the road segment geospatial layer features

Also, when looking in more detail at the ROAD_NAME values in Figure 5-31, we notice that some of them are not using the same formatting (i.e. road type first instead of road name) or include some string characters that could create some problems for automated processing (i.e. *'EMERILLON, CÔTE DE L''* has a comma and also French characters) that need to be handled.

Finally, regarding SUBCLASS, Figure 5-32 shows some values that need to be homologated ('*ROUNDABOUT* 'and '*Roundabout')* and others that are related with road segments that are used by the public transport system (*'TRANSITWAY'* and *'BUSONLY'*). A strategy to handle these values is defined in section 5.3.1.

5.2.1.3. Neighborhoods Geospatial Layer

For the neighborhoods geospatial layer, there are no missing or special values to handle. Figure 5-33 shows the general statistics related.

Special Road Name Records by Subclass (ROAD_NAME / SUBCLASS)

Road Name	Subclass	
Null	ARTERIAL	8
	BUSONLY	11
	COLLECTOR	1
	CONNECTOR	2
	LOCAL	79
	RAMP	1
	TRANSITWAY	8
ACADIE, COUR DE L'	LOCAL	1
AUBEPINES, PROMENA..	COLLECTOR	5
AUBRAIS, CROISSANT ..	LOCAL	1
AVE. DU PERE CHARLE..	LOCAL	2
AVE. PAUL EMILE LAMA..	LOCAL	2
AVE. VINCENT MASSEY	LOCAL	2
AVENUE	LOCAL	5
BAIE-DES-CASTORS, D..	LOCAL	5
BAIE-VERTE, CROIS DE..	LOCAL	2
CHANTS-D'OISEAUX,VO..	LOCAL	1
CHATEAU, CROISSANT ..	LOCAL	1
CHENE, VOIE DU	LOCAL	2
EAST	LOCAL	1
EMERILLON, CÔTE DE L'	LOCAL	1
FAMILLE-LAPORTE,AVE ..	COLLECTOR	7
JONQUILLE, VOIE DE LA	LOCAL	3
PEPIN, COUR	LOCAL	1

Sum of Number of Records broken down by Road Name and Subclass. The view is filtered on Road Name, which keeps 18 of 5,089 members.

(a)

Special Road Name Records by Subclass and Type (ROAD_NAME / SUBCLASS / ROAD_TYPE)

Road Name	Subclass	Road Type	
Null	ARTERIAL	Null	8
	BUSONLY	Null	11
	COLLECTOR	Null	1
	CONNECTOR	Null	2
	LOCAL	Null	79
	RAMP	Null	1
	TRANSITWAY	Null	8
ACADIE, COUR DE L'	LOCAL	Null	1
AUBEPINES, PROMENA..	COLLECTOR	Null	5
AUBRAIS, CROISSANT ..	LOCAL	Null	1
AVE. DU PERE CHARLE..	LOCAL	AVE	2
AVE. PAUL EMILE LAMA..	LOCAL	AVE	2
AVE. VINCENT MASSEY	LOCAL	AVE	2
AVENUE	LOCAL	RD	5
BAIE-DES-CASTORS, D..	LOCAL	ST	5
BAIE-VERTE, CROIS DE..	LOCAL	Null	2
CHANTS-D'OISEAUX,VO..	LOCAL	Null	1
CHATEAU, CROISSANT ..	LOCAL	Null	1
CHENE, VOIE DU	LOCAL	Null	2
EAST	LOCAL	ST	1
EMERILLON, CÔTE DE L'	LOCAL	Null	1
FAMILLE-LAPORTE,AVE ..	COLLECTOR	Null	7
JONQUILLE, VOIE DE LA	LOCAL	Null	3
PEPIN, COUR	LOCAL	Null	1

Sum of Number of Records broken down by Road Name, Subclass and Road Type. The view is filtered on Road Name, which keeps 18 of 5,089 members.

(b)

Figure 5-31 Special road name records by (a) subclass (ROAD_NAME / SUBCLASS) and (b) subclass and type (ROAD_NAME / SUBCLASS / ROAD_TYPE)

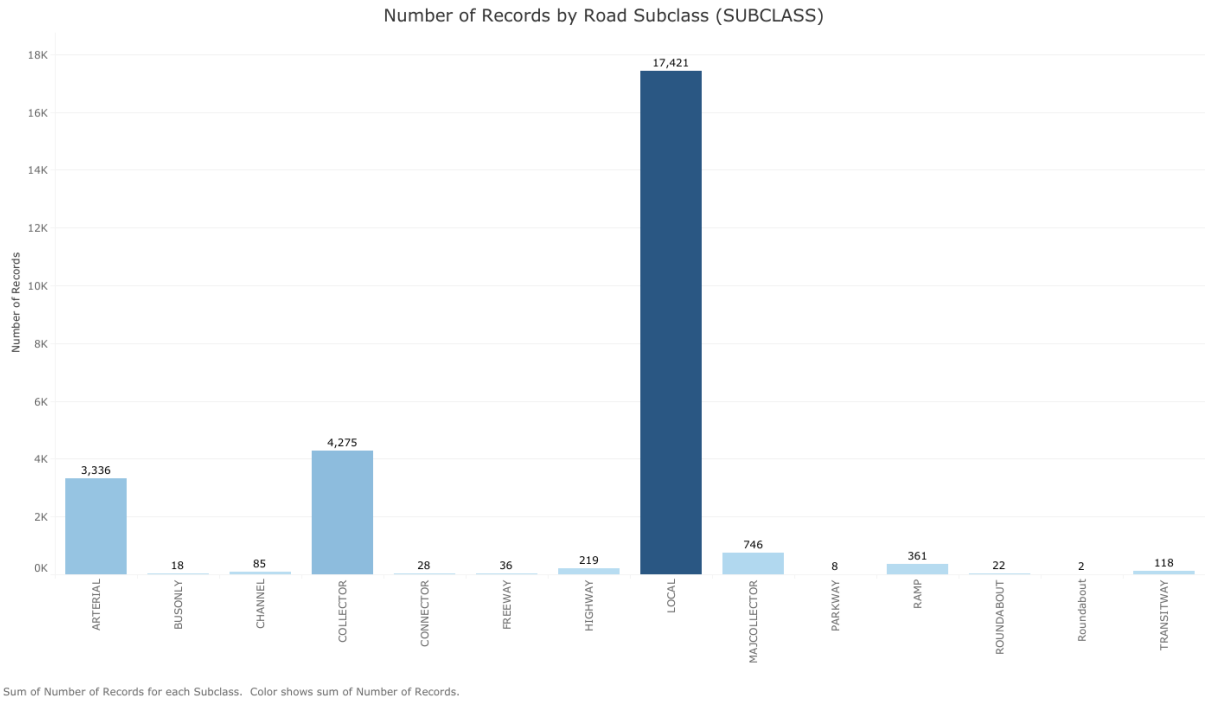


Figure 5-32 Number of records by road subclass (SUBCLASS)

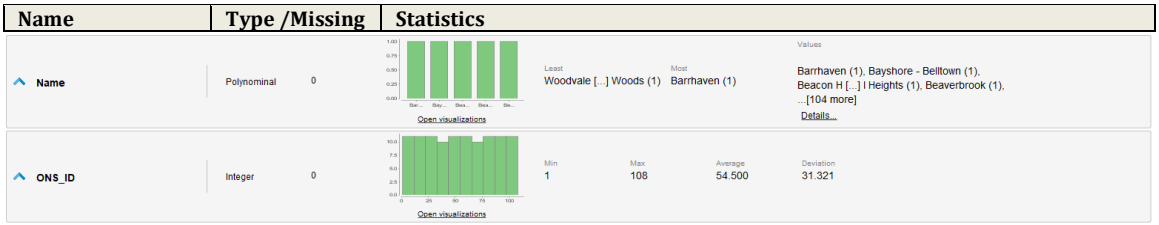


Figure 5-33 General statistics about neighborhoods geospatial layer features

5.3. Data preparation

The data preparation phase of the methodology, as its name suggests, defines the necessary steps to have the data ready to implement the machine learning models (modeling). Because some machine learning algorithms have special requirements for the data format (i.e. can only accept categorical data for example), it is important to apply the right transformations to the data. Also, certain transformations (i.e. standardization) will improve the prediction power of the algorithm. A special step included as part of this phase is the construction of the new data (known as feature engineering), where a series of new features are engineered based on statistics, visualizations, domain knowledge or simple a hunch or guess (i.e. our supposition that the social events could have an impact over collisions rates).

This stage includes the strategy defined to (1) select and clean, (2) construct, (3) integrate and (4) format the data.

5.3.1. Data Selection and Cleaning

The strategy to select and clean the data is based on the previous visualization and the output of the data quality verification.

Main Collision Dataset: Several features in the main collision dataset are related with information that will be available only after the collision happens (i.e. CLASSIFICATION_OF_ACCIDENT, INITIAL_IMPACT_TYPE) and cannot be used for collision prediction, and as such, they are eliminated from the model. Because there is vast amount of data, the samples with missing values were filtered (a maximum of 31 samples over a total of 43494, approximately 0.1%).

Roads Segment Geospatial Layer: As it was noticed previously in section 5.1.4.2, some of the road segments included are related with the public transport system ('TRANSITWAY' and 'BUSONLY'). Because of that, they were eliminated. Also, a strategy to handle the missing data was implemented. To fill the missing values related with ROAD_NAME (a total of 110), we considered an imputation approach using the value 'NONAME'.

Figure 5-34 shows the distribution of Null ROAD_NAME records by SUBCLASS. We can notice that the largest amount of road segments is related with the 'LOCAL' SUBCLASS.

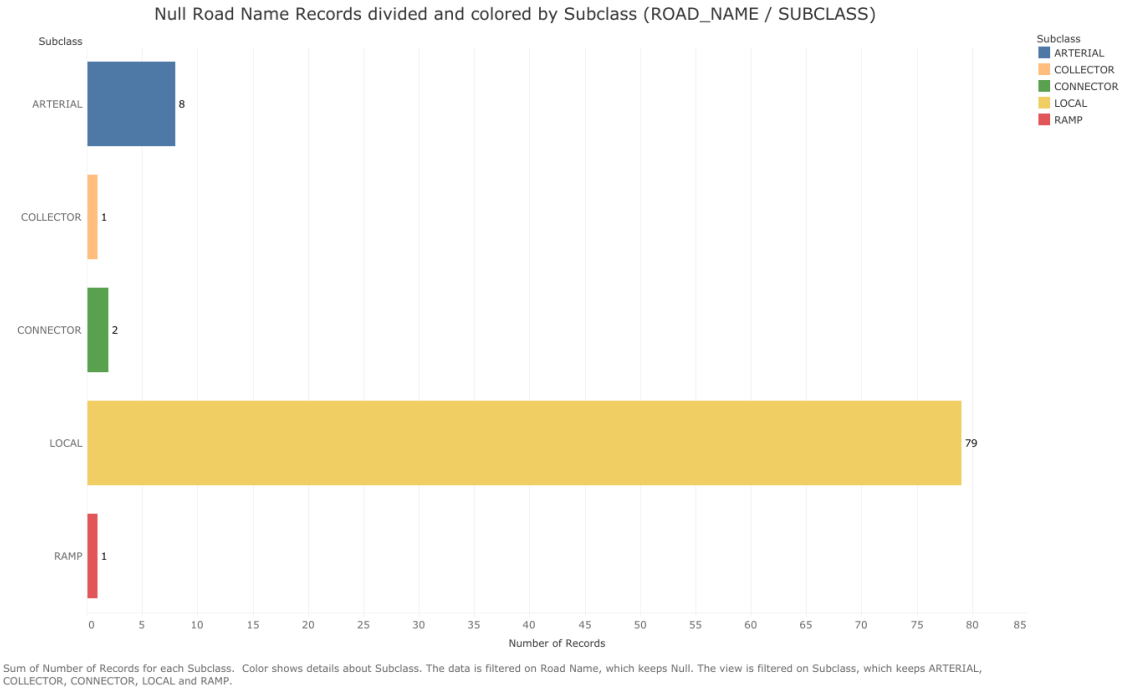


Figure 5-34 Number of Null road name records divided and colored by subclass (ROAD_NAME / SUBCLASS)

Regarding the SUBCLASS value *'Roundabout'* and *'ROUNDABOUT'*, the first was homologated to the second (in uppercase).

In regards to the roads segment without ROAD_TYPE (a total of 1456), after a more detailed inspection, we noticed that a large number of them were related with few types of roads like highways (i.e. *'HIGHWAY 417', 'REGIONAL ROAD 174', 'HIGHWAY 416'*), ramps (i.e. *'RAMP', 'HWY417 RAMP'*) as well as some specific roads

(i.e. *'TURN LANE'*, *'MONTREAL, CHEMIN DE'*, *'ORLEANS, BOULEVARD D''*, *'EPINETTES, AVENUE DES'*). A manual process was executed to fill out the right values.

Finally, in the case of the road segments with string characters that could generate problems (i.e. *'EMERILLON, CÔTE DE L''*, *'MONTREAL, CHEMIN DE'*, *'ORLEANS, BOULEVARD D''*, *'EPINETTES, AVENUE DES'*), a manual process using *QGIS* with a base map layer (*Open Street Map)* was executed to fill out the right values. Figure 5-35a shows an example of the road segment named *'EMERILLON, CÔTE DE L''* without the *Open Street Map* base layer, and Figure 5-35b the same road segment with the *Open Street Map* base layer, in which we can notice that the name related with the road should be *'EMERILLION RIDGE'*.

(a) (b)

Figure 5-35 Original road segment in QGIS (a) without Open Street Map base layer and (b) with Open Street Map base layer

Neighborhoods Geospatial Layer: The original neighborhood geospatial layer includes the *z* coordinates of the polygons (the neighborhoods). To avoid any possible issues at the time of executing geospatial queries, the *z* coordinates were removed.

5.3.2. Data Construction

To potentially increase the machine learning prediction accuracy, a series of new features was generated. Considering that the geospatial coordinates (XCOORD,

YCORD) for each collision are provided, previous related work findings in the literature suggesting that some spatial related features might have an effect on collision trends (i.e. road type, road sinuosity, etc.; described in section 2.1 and section 2.3) and also considering that different spatial data layers are available as part of the Ottawa Open Data initiative (i.e. roads and neighborhoods), it is possible to generate additional features that consider the geo-localization. Using the open source library *GeoPandas* [97], a series of Python scripts was created to calculate additional features (i.e. road sinuosity, road direction, etc.) and perform a geospatial matching between the collision samples coordinates and the different geospatial data layers (and between them). The following sections explain in more detail the related feature generation process.

5.3.2.1. Roads Features Generation

Using the spatial coordinates of each road segment, additional road features were calculated. Also, for each collision sample, a series of roads features was added through a geospatial road match (i.e. find the closest road to the collision location).

Considering that in terms of the road location, a collision can happen in an intersection or not, the roads feature generation process involves several steps: (1) calculate road segment features, (2) classify collision samples by accident location, (3) identify the road intersections, (4) match non-intersection collision samples with road segments; and (5) match intersection collision samples with road intersections.

Step 1: Roads Segment Calculated Features

For each road segment, four additional features were calculated: the *sinuosity* (ROAD_SINUOSITY), the direction/slope (ROAD_DIRECTION) and the vertex (ROAD_FIRST_POINT and ROAD_END_POINT, to be used later as part of the road features generation). Also, for road length and road sinuosity, because the samples have very similar values, a mathematical logarithm transformation was applied to generate new features (ROAD_SINUOSITY_LOG and ROAD_LEN_LOG) that can express the small differences and could be used by the machine learning algorithm to identify some patterns.

The Algorithm 5-1 shows the general pseudo-code related.

Algorithm 5-1: Adding road segments sinuosity, direction, vertex, sinuosity log and length log as features to the road segment layer (General)

```
1: For each road segment do
2:      Calculate and add the ROAD_SINUOSITY as a new feature
3:      Calculate and add the ROAD_DIRECTION as a new feature
4:      Calculate and add the road vertex (ROAD_FIRST_POINT and ROAD_END_POINT) as new features
5:      Calculate the log of ROAD_SINUOSITY and add ROAD_SINUOSITY_LOG as a new feature
6:      Calculate the log of ROAD_LEN and add ROAD_LEN_LOG as a new feature
7: End For
```

A more detailed pseudo-code version of the previous algorithm is provided in the Algorithm 5-2.

Algorithm 5-2: Adding road segments sinuosity, direction, vertex, sinuosity log and length log as features to the road segment layer (Detailed)

```
Inputs:
    road_segments // a geospatial layer with all road segments
Output:
    road_segments // a geospatial layer with all the road segments with the new features added

Function Create_Sinuosity_Direction_Vertex_Log(road_segments)
    For each rs in road_segments
        If rs.geometry is not Null  // avoid empty geometry items
            p0 = rs.geometry.start_point // start point coordinates of the road segment
            pn = rs.geometry.end_point  // end point coordinates of the road segment

            // calculate ROAD_DIRECTION
            If (abs(pn.y – p0.y) > abs(pn.x – p0.x))
                rs['ROAD_DIRECTION'] = "NS" // North or South
            Else
                rs['ROAD_DIRECTION'] = "EW" // East or West


            // calculate Euclidean distance
            euclid_distance = sqrt((p0.x – pn.x)**2 + (p0.y – pn.y)**2)

            // calculate ROAD_SINUOSITY
            if euclid_distance > 0  // avoid divide by zero
                rs['ROAD_SINUOSITY'] = rs['ROAD_LENGTH'] / euclid_distance
            Else
                rs['ROAD_SINUOSITY'] = 1
            End if

            // Add logarithm of the ROAD_SINOUSITY and ROAD_LENGTH
            rs['ROAD_SINUOSITY_LOG'] = math.log(rs['ROAD_SINUOSITY'])
            rs['ROAD_LEN_LOG'] = math.log(rs['ROAD_LEN'])

            // Add START and END road segment points (to be used later)
            rs['START_POINT'] = p0
            rs['END_POINT'] = pn
```

End if
End for
Return road_segments
End Function Create_Sinuosity_Direction_Vertex_Log

Step 2: Collision Samples Classification by ACCIDENT_LOCATION

Using the information from the MVCR Manual [94], we classify the collision samples into two groups using the ACCIDENT_LOCATION feature: *intersection* (ACCIDENT_LOCATION equals to *'02 - Intersection related'* OR *'03 - At intersection')* or *non-intersection* (all the other values). More detailed information about the ACCIDENT_LOCATION feature codes and description (MVCR 0301) can be found in the Appendix A.1. Figure 5-36 shows the collisions in the Carleton University area colored by the previous classification.

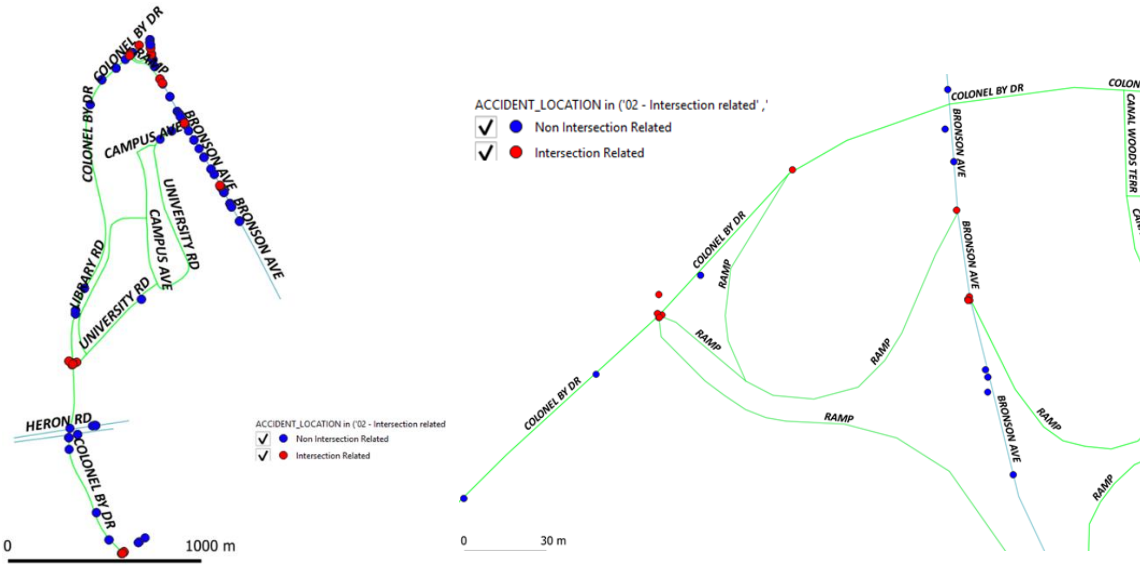


Figure 5-36 Collisions around Carleton University area colored by ACCIDENT_LOCATION

We can notice that, as expected, the collisions related with intersections (in red) are in fact, occurring at road intersections or very close to them (according with the MVCR 0301 definition, within a range of 100 meters).

Step 3: Identify / Create a Roads Intersection Layer

Each road segment starts and ends at a road intersection. As an example, Figure 5-6 showed the road segment *University Rd* around *Carleton University* area.

The Algorithm 5-3 shows the general pseudo-code to find the intersections.

Algorithm 5-3: Calculation of all road intersections (General)

```
1: For each road segment do
2:         Find vertex (start and end points)
3:         For each vertex (start/end points) do
4:                 Match the road segments
5:                 If at least two road segments exist with different road name then
6:                         Create the intersection
7:                 End If
8:         End For
9: End For
```

It is important to notice that the end point of a road segment matches the starting point of the following one. Using this observation, we can simplify our algorithm by only considering the start or end points (only one vertex) to find all the intersections. After the first implementation, we noticed that the processing time for matching the candidate intersection (one vertex) with the road segments (26113) is very long, taking a total of 60 minutes to create all the intersections. To address this issue, an improved version of the algorithm was implemented using a *spatial index* and *bounding boxes,* reducing the complete execution time to roughly 3 minutes. A *spatial index* is a method for indexing spatial databases to optimize spatial queries (i.e. *intersects, within*). In *GeoPandas*, the *spatial index* is implemented using *R-trees*. These are indexing structures in a tree form that represent the different spatial objects (i.e. lines, points, polygons) by their minimum *bounding boxes* (i.e. a minimum rectilinear shape that completely contains the bounded object or objects [98]). A bounding box can enclose data objects or other bounding boxes.

The following example explains the concept of bounding boxes and the R-tree representation. Given the spatial objects shown in Figure 5-37 (R8-R19): R8, R9 and R10 are the minimum bounding boxes that enclose the objects D1, D2 and D3; R10-R19 are quadrilaterals, the R-tree representation is shown in Figure 5-38a.

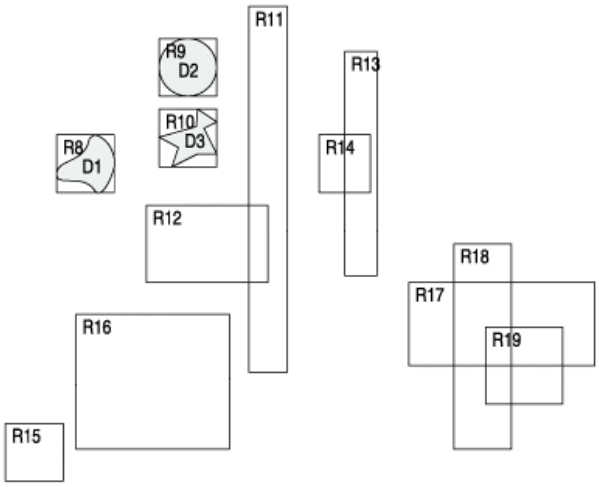


Figure 5-37 Spatial objects for R-tree example (from [98])

In Figure 5-38, at the bottom of the tree ("leaf pages"), we can find the different data objects (R8-R19). A series of minimum bounding boxes that enclosed the previous objects are created and added to the tree (as "branch pages"), i.e. R6 is the minimum bounding box that encloses R15 and R16; R7 is the minimum bounding box for R17, R18 and R19. At the top of the tree ("root page"), there are only two bounding boxes (R1 and R2). Each bounding box in the R-tree has attached its coordinates (the range related with the bounding box) that are used for spatial queries. Suppose now that we want to find the different spatial objects that *intersect* the red line (i.e. a road in our case), using the R-tree representation, the query will match from top to bottom R1-> R3 -> R9 (leaf) and R10 (leaf), the final result will be R9 and R10. Using this process, the task to match an object with others is improved. In our work, to create the road intersections, a matching process between each road segment vertex (two for each road segment) and the complete road segments (a total of 26113 road segments) is necessary.

An important additional improvement is also achieved when the data object that we want to match is represented by its *bounding box*; this means that instead of the red line in Figure 5-38b, we consider its *bounding box* (blue rectangle).

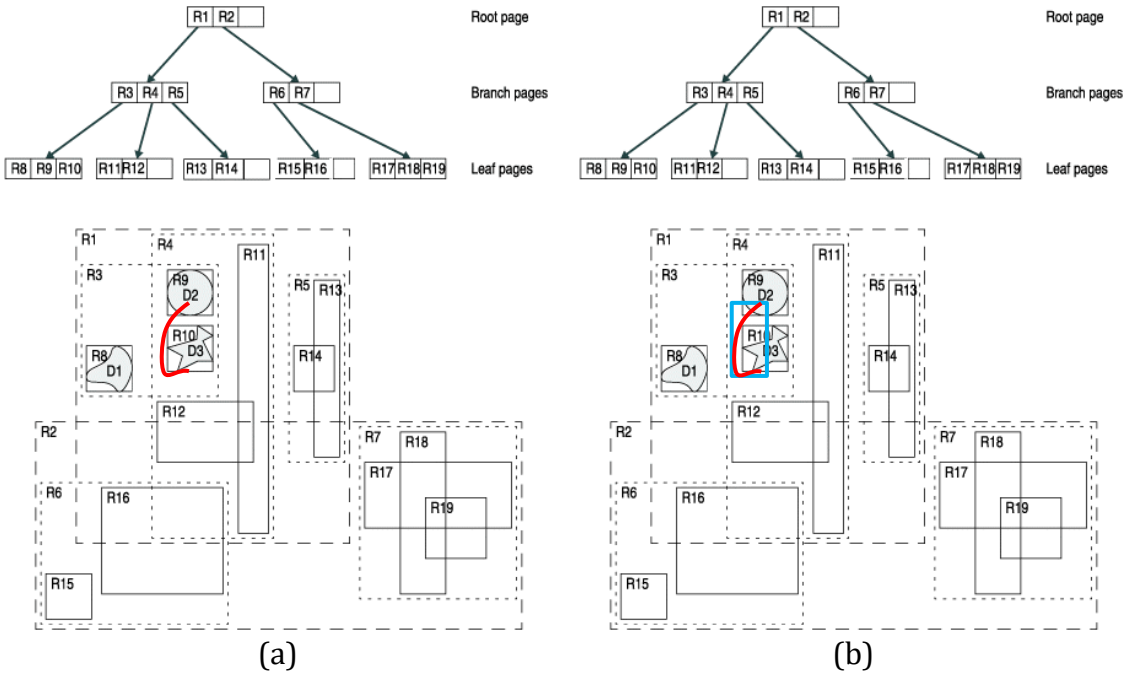
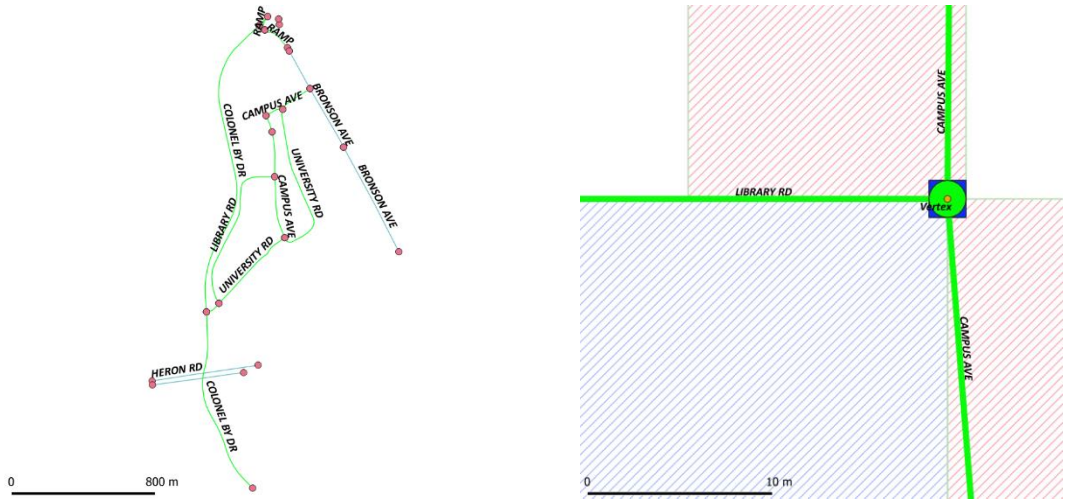


Figure 5-38 R-Tree example (from [98]): (a) Intersection of red line with existing spatial objects and (b) intersection of red line bounding box (blue rectangle) with existing spatial objects

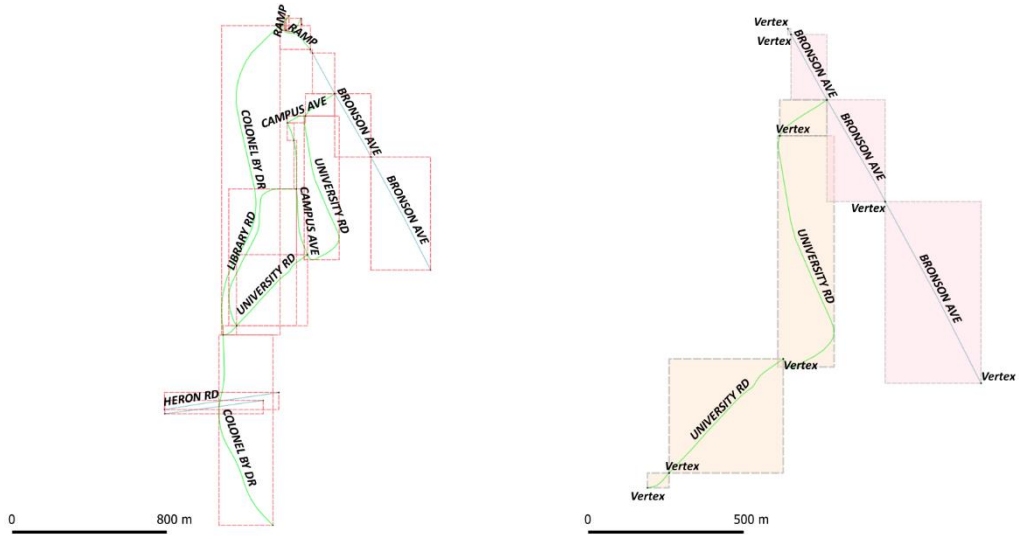
The calculated vertices (candidate intersections) of all the road segments around *Carleton University* are displayed in Figure 5-39a, while Figure 5-39b illustrates an example of the elements related with the matching process including a candidate intersection (orange dot); a 1m *buffer* around the candidate intersection (green circle); a *bounding box* around the buffered intersection (blue square) and the *bounding boxes* around each road segments related with *Campus Ave* and *Library Rd* inside Carleton University area (i.e. the green lines are the road segments, the blue and red marked areas are part of the road segments bounding boxes). The matching process between the *bounding box* around the *buffered* candidate intersection (blue square area) and the *bounding boxes* around the road segments using a *spatial index* returns the three green road segments, two related with *Campus Ave* and the last related with *Library Rd*.



(a) (b)

Figure 5-39 (a) Carleton University calculated road segment vertex (red dots) and (b) Calculated vertex (orange dot), buffer (green dot), bounding box (blue square) on *Library Rd* and *Campus Ave*

An example of bounding boxes related with all the road segments around *Carleton University* are shown in Figure 5-40a, while Figure 5-40b shows the *bounding boxes* and *vertices* around *University Dr* and *Bronson Av* road segments.



(a) (b)

Figure 5-40 Bounding boxes (a) around road segments in Carleton University and (b) and vertices around *University Rd* and *Bronson Ave* road segments in Carleton University

Using this approach, we obtained an important improvement in the execution time for calculating road intersections - from 60 minutes to about 3 minutes.

The final matching process includes: (1) create a *spatial index* to the *road segments* (2) create a *buffer* of 1m around the vertex point, and (3) create a spatial query to match the *intersection* between the road segment *spatial index* and the *bounding box* of the *buffered* vertex.

The Algorithm 5-4 provides a more detailed version of the geospatial matching pseudo-code.

Algorithm 5-4: Calculation of all road intersections (Detailed)

```
Inputs:
    road_segments // a geospatial layer with all road segments
Output:
    road_intersections // a geospatial layer with all the road intersections

Function Create_Intersections_Layer (road_segments)

    spatial_index = spatial_index(roads_segments) // create a spatial index for road segments
    road_intersections = [] // initialize road intersections

    // create a list with all unique road segment start points
    l_points = unique(road_segments['START_POINT'])

    For each rs in road_segments
        If rs.geometry is not Null // avoid empty geometry items
            If rs['START_POINT'] is in l_points
            // the road segment start point is part of the unique list of start points to validate

                // create a buffer area of 1 meter around the starting point
                buffer_point = rs['START_POINT'].buffer(1.0)

                // create a list with all the road segments that intersects the buffer point
                // bounds, for fast processing we use the spatial index
                l_road_inter = spatial_index.intersection(buffer_point.bounds)

                // filter the list to unique road segments names that have a
                // different name to the original
                l_road_inter = unique(l_road_inter['ROAD_NAME'])
                l_road_inter = l_road_inter.remove(rs['ROAD_NAME'])

                If len(l_road_inter) > 0
                // at least one road segment in the intersection with different road name
                // was found, create new intersection
                    new_intersection['INTER_NAME'] =
                        rs['ROAD_NAME']  + road_names['ROAD_NAME']
                    new_intersection['INTER_SEGMENT'] =
                        rs['ROAD_SEGMENT']  + road_names['ROAD_SEGMENT']
                    new_intersection['INTER_SUBTYPE'] = rs['ROAD_SUBTYPE']
                    new_intersection['INTER_SUBCLASS'] = rs['ROAD_SUBCLASS']
                    new_intersection['INTER_SINUOSITY'] = 1.0
                    new_intersection['INTER_SINUOSITY_LOG'] = 0.0
                    new_intersection['INTER_LEN'] = 1.0
                    new_intersection['INTER_LEN_LOG'] = 0.0
                    new_intersection['INTER_DIRECTION] = "NA"

                    // add the new intersection to the results
```

```
                                    road_intersections.append(new_intersection)
                                End if

                                // remove the start point to the list of points to validate
                                l_points.remove(rs['START_POINT'])
                            End if
                        End if
                End For
                Return road_intersections
```

End Function Create_Intersections_Layer

The calculated intersections around *Carleton University* area are shown in Figure 5-41.

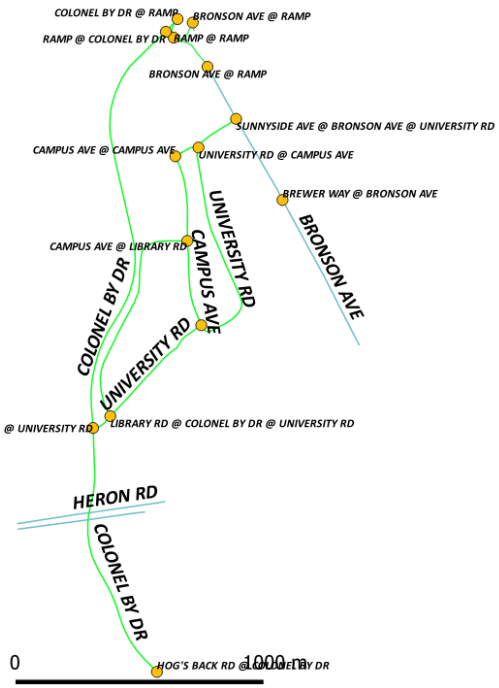


Figure 5-41 Intersections calculated around Carleton University area (orange dots)

Step 4: Match Non-Intersection Collision Samples with Road Segments

The Algorithm 5-5 shows the general pseudo-code to match non-intersection collision samples with the road segments.

Algorithm 5-5: Matching nearest road segment to each non-intersection collision sample by minimum distance (General)

```
1: For each non-intersection collision sample do
2:      Find the nearest (distance) road segment
3:          Add the ROAD_NAME feature to the non-intersection collision sample
4:          Add the ROAD_SEGMENT feature to the non-intersection collision sample
5:          Add the ROAD_SUBTYPE feature to the non-intersection collision sample
6:          Add the ROAD_SUBCLASS feature to the non-intersection collision sample
7:          Add the ROAD_DIRECTION feature to the non-intersection collision sample
8:          Add the ROAD_LEN feature to the non-intersection collision sample
9:          Add the ROAD_LEN_LOG feature to the non-intersection collision sample
10:         Add the ROAD_SINUOSITY feature to the non-intersection collision sample
11:         Add the ROAD_SINUOSITY_LOG feature to the non-intersection collision sample
12: End For
```

In this case, we also face long execution times because of the distance calculation between each non-intersection collision sample and each road segment (a total of 26113 road segments). To improve the execution time, the main objective is to minimize the number of road segments to be considered in the distance calculation. To achieve this, a similar approach to the previous one was implemented: (1) create a *spatial index* to the road segments; (2) create a *buffer* of 100 meters around each non-intersection collision sample; (3) create a spatial query to match the *intersection* between the road segment *spatial index* to the *bounding box* of the *buffered* non-intersection collision sample; and (4) from the previous selected road segments, find the nearest (minimum *distance*) road segment to the non-intersection collision sample.

Algorithm 5-6 shows a more detailed pseudo-code version of the Algorithm 5-5.

Algorithm 5-6: Matching nearest road segment to each non-intersection collision sample by minimum distance (Detailed)

```
Inputs:
        road_segments // a geospatial layer with all road segments
        non_intersection_collision_samples // a geospatial layer with all non-intersection collision samples
Output:
        non_intersection_collision_samples // a geospatial layer with all non-intersection collision samples with
                                           // the new features added

Function Match_nearest_road_segment_by_mindistance (road_segments, non_collision_samples)
        spatial_index = spatial_index(roads_segments) // create a spatial index for road segments

        For each cs in non_collision_samples
                If cs.geometry is not Null // avoid empty geometry items
```

```
                    // create a buffer area of 100 meter around the collision sample
                    buffer_point = cs.buffer(100.0)

                    // create a list with all the road segments that intersects the buffer point
                    // bounds, for fast processing we use the spatial index
                    l_road_around = spatial_index.intersection(buffer_point.bounds)

                    If len(l_road_around) > 0
                    // at least one road segment was found
                            // find the minimun distance from each of the road segments previously
                            // found to the original collision sample
                            l_road_distance = l_road_around.distance(cs.geometry)

                            // find the nearest road segment
                            d_min = l_road_distance.min()
                            road_id = l_road_distance['DISTANCE' == d_min].id()
                            closest_road = l_road_around.distance['ROAD_ID' == road_id]

                            // assign the closest road segment and the related road segment
                            // features to the collision sample
                            cs['ROAD_NAME'] = closest_road['ROAD_NAME']
                            cs['ROAD_SEGMENT']= closest_road['ROAD_SEGMENT']
                            cs['ROAD_SUBTYPE']= closest_road['ROAD_SUBTYPE']
                            cs['ROAD_SUBCLASS']= closest_road['ROAD_SUBCLASS']
                            cs['ROAD_DIRECTION']= closest_road['ROAD_DIRECTION']
                            cs['ROAD_LEN'] = closest_road['ROAD_LEN']
                            cs['ROAD_LEN_LOG'] = closest_road['ROAD_LEN_LOG']
                            cs['ROAD_SINUOSITY']= closest_road['ROAD_SINUOSITY']
                            cs['ROAD_SINUOSITY_LOG']= closest_road['ROAD_SINUOSITY_LOG']
                    End if
            End if
    End For
    Return non_collision_samples
End Function Match_nearest_road_segment_by_mindistance
```

Step 5: Match Intersection Collision Samples with Road Intersections

Algorithm 5-7 provides the pseudo-code to match intersection collision samples with road intersections, again a similar process to the one in Step 4, while Algorithm 5-8 shows the details.

Algorithm 5-7: Matching nearest road intersection to each intersection collision sample by minimum distance (General)

```
1: For each intersection collision sample do
2:      Find the nearest (distance) road intersection
3:          Add the ROAD_NAME feature to the intersection collision sample
4:          Add the ROAD_SEGMENT feature to the intersection collision sample
5:          Add the ROAD_SUBTYPE feature to the intersection collision sample
6:          Add the ROAD_SUBCLASS feature to the intersection collision sample
7:          Add the ROAD_DIRECTION feature to the intersection collision sample
8:          Add the ROAD_LEN feature to the intersection collision sample
9:          Add the ROAD_LEN_LOG feature to the intersection collision sample
10:         Add the ROAD_SINUOSITY feature to the intersection collision sample
11:         Add the ROAD_SINUOSITY_LOG feature to the intersection collision sample
12: End For
```

Algorithm 5-8: Matching nearest road intersection to each intersection collision sample by minimum distance (Detailed)

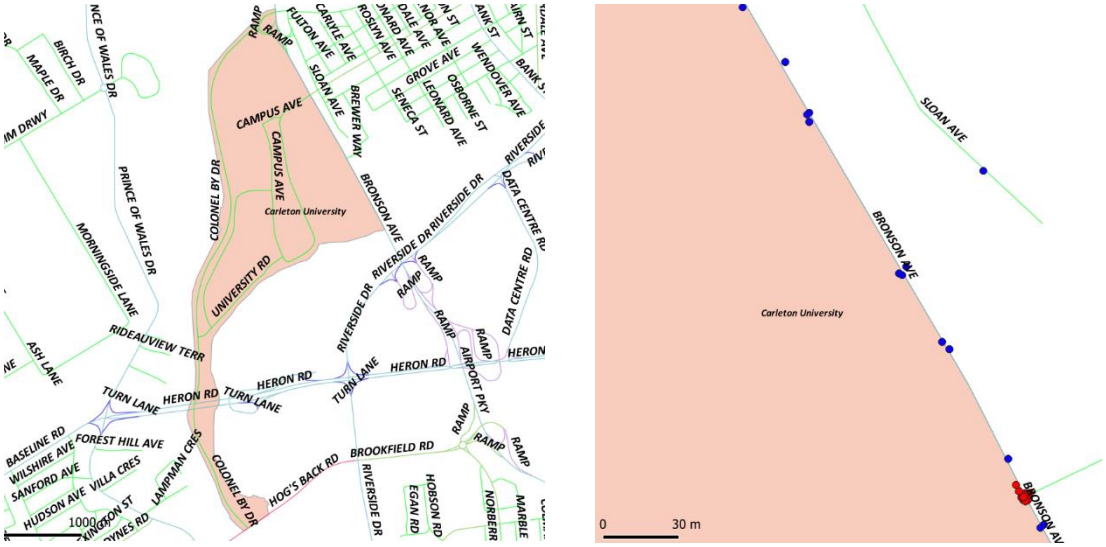
```
Inputs:
        road_intersections // a geospatial layer with all road intersections
        intersection_collision_samples // a geospatial layer with all intersection collision samples
Output:
        intersection_collision_samples // a geospatial layer with all intersection collision samples with the new
                                       // features added

Function Match_nearest_road_intersection_by_mindistance (road_intersections, collision_samples)
        spatial_index = spatial_index(roads_intersections) // create a spatial index for road intersections

        For each cs in collision_samples
                If cs.geometry is not Null // avoid empty geometry items

                        // create a buffer area of 100 meters around the collision sample
                        buffer_point = cs.buffer(100.0)

                        // create a list with all the road intersections that intersects the buffer point
                        // bounds, for fast processing we use the spatial index
                        l_inter_around = spatial_index.intersection(buffer_point.bounds)

                        If len(l_intersection_around) > 0
                        // at least one road segment was found
                                // find the minimum distance from each of the road intersections previously
                                // found to the original collision sample
                                l_inter_distance = l_inter_around.distance(cs.geometry)

                                // find the nearest road intersection
                                d_min = l_inter_distance.min()
                                inter_id = l_inter_distance['DISTANCE' == d_min].id()
                                closest_inter=l_inter_around.distance['INTER_ID' == inter_id]

                                // assign the closest road intersection and the related road intersection
                                // features to the collision sample
                                cs['ROAD_NAME'] = closest_inter['INTER_NAME']
                                cs['ROAD_SEGMENT'] = closest_ inter['INTER_SEGMENT']
                                cs['ROAD_SUBTYPE'] = closest_ inter['INTER_SUBTYPE']
                                cs['ROAD_SUBCLASS'] = closest_ inter['INTER_SUBCLASS']
                                cs['ROAD_DIRECTION'] = closest_ inter['INTER_DIRECTION']
                                cs['ROAD_LEN'] = closest_ inter['INTER_LEN']
                                cs['ROAD_LEN_LOG'] = closest_ inter['INTER_LEN_LOG']
                                cs['ROAD_SINUOSITY'] = closest_ inter['INTER_SINUOSITY']
                                cs['ROAD_SINUOSITY_LOG'] = closest_ inter['INTER_SINUOSITY_LOG']
                        End if
                End if
        End For
        Return collision_samples
End Function Match_nearest_road_intersection_by_mindistance
```

5.3.2.2. District/Neighborhood Feature Generation

Using the ONS neighborhood layer and the same idea of the section 5.3.2.1, a geospatial neighborhood match was performed.

Something important to notice is that, because the boundaries of some neighborhoods match road segments, and the collision samples are located around a road segment (not necessary on the top of it), the matching process could assign different neighborhoods to collision samples that are related with the same road segment (as a result of the related coordinates).

Figure 5-42a shows the road segments around the *Carleton University* neighborhood (area filled in orange). We can notice that the road segment *Bronson Ave* matches the boundaries of the neighborhood. In Figure 5-42b, we can notice that some of the collision samples around *Bronson Ave* are located around the road (in fact, some of the collision samples are outside of the orange area).

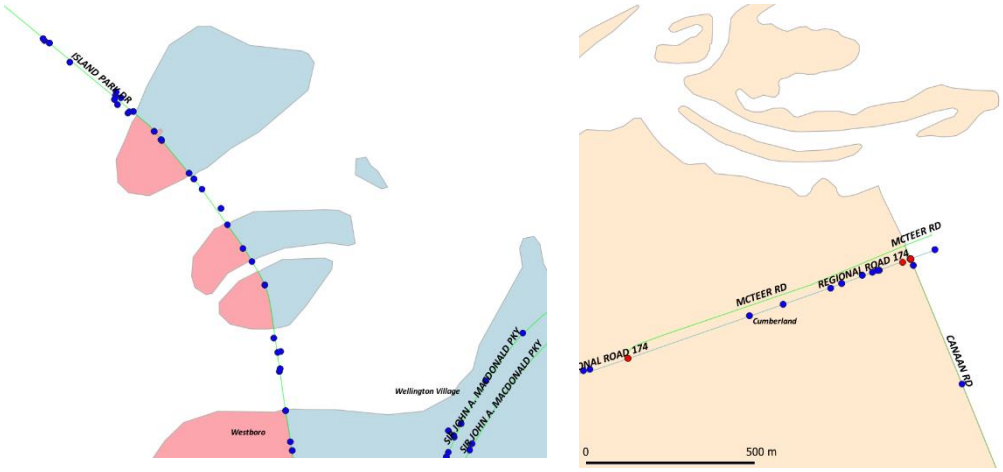


(a) (b)

Figure 5-42 (a) Road segments around *Carleton University* neighborhood and (b) Collision samples around *Bronson Ave* in *Carleton University* neighborhood

Another issue to consider is the fact that some of the collision samples could be located outside of the city limits because: (1) the road segments that match the boundaries of neighborhoods can also match the city limits or (2) the road segments are over (bridges) or close to water sources (lakes and rivers). Figure 5-43a and Figure 5-43b show examples of such situations. We can notice in Figure 5-43a that

several collision samples over *Island Park Dr* (a bridge over the Ottawa river) also match the boundaries of two neighborhoods *Westboro* and *Wellington Village*. In Figure 5-43b, we can notice collision samples outside of the city limits.



(a) (b)

Figure 5-43 Collision samples (a) over a bridge over Ottawa River around *Island Park Dr* in *Wellington Village* and *Westboro* neighborhoods and (b) outside the City Limits around *Regional Road 174* in *Cumberland* neighborhood

Because the neighborhoods are polygons, in this case we will use the *GeoPandas* function *within* for the geospatial matching. Using this approach, the issue with the collision samples outside the neighborhoods still needs to be addressed. A solution that was tested was to create a *buffer* around each neighborhood (to try to enclose all the collision samples around), but it creates additional problems, for example a collision sample could belong to two different neighborhoods. After trying several options, the final spatial matching process implementation considers two steps: (1) match the collision samples to the neighborhoods using the function *within*, and (2) for the non-matched collision samples (the ones outside the neighborhoods), use the function *distance* to match it to the nearest neighborhood. This process was also applied to road segments and road intersections.

The final district/neighborhood feature generation process involves several steps: (1) match collision samples to neighborhoods, (2) match road intersections to neighborhoods, and (3) match road segments to neighborhoods.

Step 1: Matching Collisions Samples to Neighborhoods

Algorithm 5-9, further detailed in Algorithm 5-10, shows the general pseudo-code to match collision samples to neighborhoods.

Algorithm 5-9: Matching collision samples with a neighbourhood (General)

```
1: For each neighborhood do
2:      Find the collision samples within its area
3:          Add the ONS_NAME feature to the collision sample
4:          Add the ONS_ID feature to the intersection collision sample
5: End For
6: For each non-previously matched collision samples do
7:       Find the nearest (distance) neighborhood
8:          Add the ONS_NAME feature to the collision sample
9:          Add the ONS_ID feature to the intersection collision sample
10: End For
```

Algorithm 5-10: Matching collision samples with a neighbourhood (Detailed)

```
Inputs:
    collision_samples // a geospatial layer with all the collision samples
    neighborhoods // a geospatial layer with all the neighborhoods
Output:
    collision_samples // a geospatial layer with all collision samples with the new features added

Function Match_neighborhoods_by_within(neighborhoods, collision_samples)

    For each n in neighborhoods
        If n.geometry is not Null // avoid empty geometry items

            // create a list with all the collision samples within the neighborhood
            l_collision_samples = n.within(collision_samples)

            For each cs in l_collision_samples
                // assign the neighborhood features to the collision samples
                cs['ONS_NAME'] = n['ONS_NAME']
                cs['ONS_ID'] = n['ONS_ID']
            End For
        End if
    End For
    Return collision_samples
End Function Match_neighborhoods_by_within

Function Match_nearest_neighborhoods_by_mindistance(neighborhoods, collision_samples)
    spatial_index = spatial_index(neighborhoods) // create a spatial index for road intersections

    For each cs in collision_samples
        If cs.geometry is not Null and cs['ONS_NAME'] == ""
        // avoid empty geometry items and collision samples with a neighborhood already assigned

            // create a buffer area of 100 meters around the collision sample
            buffer_point = cs.buffer(100.0)

            // create a list with all the neighborhoods that intersects the buffer point
            // bounds, for fast processing we use the spatial index
            l_neigh_around = spatial_index.intersection(buffer_point.bounds)
```

```
                    If len(l_neigh_around) > 0
                    // at least one neighborhood was found
                        // find the distance from the collision sample to each of the neighborhood
                        l_neigh_distance = l_neigh_around.distance(cs.geometry)

                        // find the nearest neighborhood
                        d_min = l_neigh_distance.min()
                        neigh_id = l_neigh_distance['DISTANCE' == d_min].id()
                        closest_neigh=l_neigh_around.distance['ONS_ID' == neigh_id]

                        // assign the neighborhood features to the collision sample
                        cs['ONS_NAME'] = closest_neigh['ONS_NAME']
                        cs['ONS_ID'] = closest_neigh['ONS_ID']
                    End if
                End if
            End For
            Return collision_samples
End Function Match_nearest_neighborhoods_by_mindistance

Function Main
            collision_samples = Match_neighborhoods_by_within(neighborhoods, collision_samples)
            collision_samples = Match_nearest_neighborhoods_by_mindistance(neighborhoods, collision_samples)
End Function Main
```

Step 2: Matching Road Intersections to Neighborhoods

Because a road intersection is similar to a collision sample (a point), to match road intersections with neighborhoods, the same approach from section 5.3.2.1 step 4 was implemented. Algorithm 5-11 provides the general pseudo-code to match road intersections to neighborhoods.

Algorithm 5-11: Matching road intersections to neighbourhoods (General)

```
1: For each neighborhood do
2:          Find the intersections within its area
3:                    Add the ONS_NAME feature to the intersection
4:                    Add the ONS_ID feature to the intersection
5: End For
6: For each non-previously matched intersection do
7:           Find the nearest (distance) neighborhood
8:                    Add the ONS_NAME feature to the intersection
9:                    Add the ONS_ID feature to the intersection
10: End For
```

Step 3: Matching Road Segments to Neighborhoods

In case of a road segment, because it can be part of different neighborhoods (i.e. could start in one and finish in another), we use the geometric centroid (the center of mass of the road segment) to identify the neighborhood to which a road segment belongs.

Using this idea, we use a point (geometric centroid) instead of a line to identify the road segment, and as a result, we can use the same approach from step 2. Figure 5-44 shows the road segments centroids around *Carleton University* area.

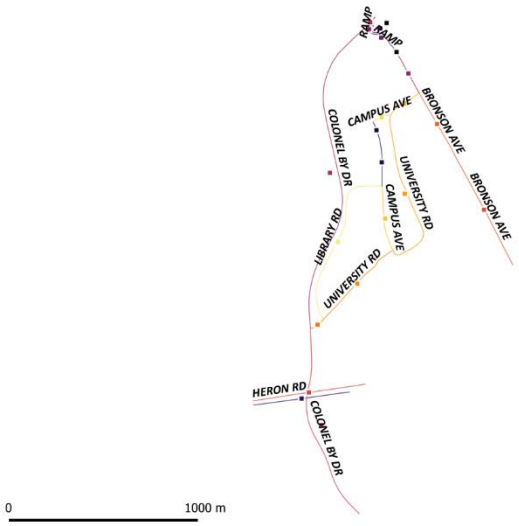


Figure 5-44 Road segment centroids around Carleton University area

The Algorithm 5-12 contains the general pseudo-code to match road segments to neighborhoods.

Algorithm 5-12: Matching road segments to neighbourhoods (General)

```
1: For each neighborhood do
2:          Find the road centroid within its area
3:                    Add the ONS_NAME feature to the road segment
4:                    Add the ONS_ID feature to the road segment
5: End For
6: For each non-previously matched road centroid do
7:          Find the nearest (distance) neighborhood
8:                    Add the ONS_NAME feature to the road segment
9:                    Add the ONS_ID feature to the road segment
10: End For
```

Because the neighborhood polygons have different sizes and forms, in an effort to create a feature that minimizes those geographical differences, a new geospatial layer was created. Using the *Ottawa city limits* spatial layer, a *tile* of 1 square kilometer was created, with a total of 3078 tiles. Figure 5-45a shows the new layer created. Using the same idea of the road segments, the centroids of each tile were used to match each tile with the neighborhoods. Figure 5-45b illustrates the tile

layer colored by neighborhoods. A detailed view of the collision samples in *Carleton University* neighborhood as well as the tiles around are presented in Figure 5-46.

(a) (b)

Figure 5-45 City of Ottawa (a) tile spatial layer and (b) tile spatial layer colored by neighborhood

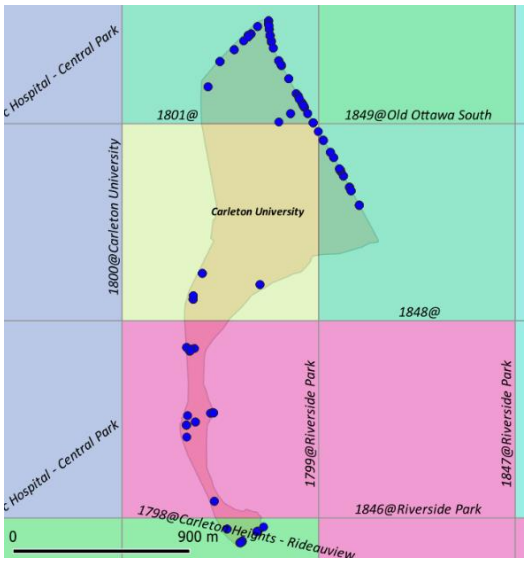


Figure 5-46 Carleton University neighborhood tiles and collision samples

The same previous process used for the neighborhoods layer was repeated to match *collision samples*, *road segments* and *road intersections* to each *tile* (Algorithm 5-13, Algorithm 5-14 and Algorithm 5-15).

Algorithm 5-13: Matching collision samples to tiles (General)

1: **For** each tile **do**
2: Find the collision samples within its area
3: Add the TILE_ID feature to the intersection collision sample
4: **End For**
5: **For** each non-previously matched collision samples **do**
6: Find the nearest (distance) tile
7: Add the TILE_ID feature to the intersection collision sample
8: **End For**

Algorithm 5-14: Matching road intersections to tiles (General)

1: **For** each tile **do**
2: Find the intersections within its area
3: Add the TILE_ID feature to the intersection
4: **End For**
5: **For** each non-previously matched intersection **do**
6: Find the nearest (distance) tile
7: Add the ONS_NAME feature to the intersection
8: Add the TILE_ID feature to the intersection
9: **End For**

Algorithm 5-15: Matching road segments to tiles (General)

1: **For** each tile **do**
2: Find the road centroid within its area
3: Add the TILE_ID feature to the road segment
4: **End For**
5: **For** each non-previously matched road centroid **do**
6: Find the nearest (distance) tile
7: Add the TILE_ID feature to the road segment
8: **End For**

In section 5.4, Table 5-4 contains the list of all the features added to each collision sample, including the ones related with this process.

5.3.2.3. Non-collision Samples Generation

Because the collision samples dataset contains only information about collisions and because in this work one of the objectives is classify the samples as *collisions* or *non-collisions*, a non-collision samples generation process is required. Drawing inspiration from the work of [53], the non-collision samples were created as described by the following pseudo-code (Algorithm 5-16). In step 3, the purpose of using a second sample (Sample2) to select randomly a different, but existing value in the dataset for the corresponding parameter, is to ensure a similar distribution on the

new non-collision samples, and thus avoid the generation of very different samples that could be easier for the classifier to identify.

Algorithm 5-16: Non-collision samples generation (General)

1: Choose randomly one sample from the original collision dataset (Sample1)
2: Choose randomly to change: road segment (RS) or hour of the day (HofD) or day of the year (DofY) (Change_Selection)
3: Select a sample with a different Change_Selection value (Sample2)
4: Create a non-collision sample (New_Sample) using:
5: **If** Change_Selection == RS **then**
6: Sample1(date/time, road surface, environment, light) features values and the rest of feature values from Sample2(X, Y, location, road segment, etc)
7: **Else** (Change_Selection == HofD OR Change_Selection == DofY):
8: Sample2(date/time, road surface, environment, location) and the rest of feature values from Sample1
9: **End If**
10: **If** New_Sample (date/time, RS) does not exist in the original collision dataset then
11: Include it as a new non-collision sample
12: **End If**
13: **Repeat** the steps 1-12 until the size of the dataset will contain twice the number of original samples.

One of the biggest issues that we faced with the implementation of the non-collision samples generation process was again the execution time. As an example, considering a dataset with 43949 collision samples, the function to calculate/add an equal number of non-collision samples took 90 minutes. The main cause was the function that appends the new non-collision sample on each iteration to the dataset (Pandas *DataFrame.append*). To address this issue, we used inside the non-collision samples loop generation a *Python* dictionary instead of a Pandas *DataFrame*, that collects all the non-collision samples generated. At the end of the generation process, the Python dictionary was transformed into a Pandas *DataFrame* using the function *DataFrame.from_dict* and finally was appended to the original collision dataset using the function *DataFrame.append*. This approach reduces the complete running time to 32 minutes.

As a result of this process, the complete dataset has 86988 samples (twice the original size).

5.3.2.4. Solar Azimuth / Elevation Feature Generation

As the weather changes during the different months of the year creating different seasons, one interesting feature that also changes over the year is the sun relative

position. As it is shown in Figure 5-47, the solar position is determined by the solar azimuth angle and the elevation.

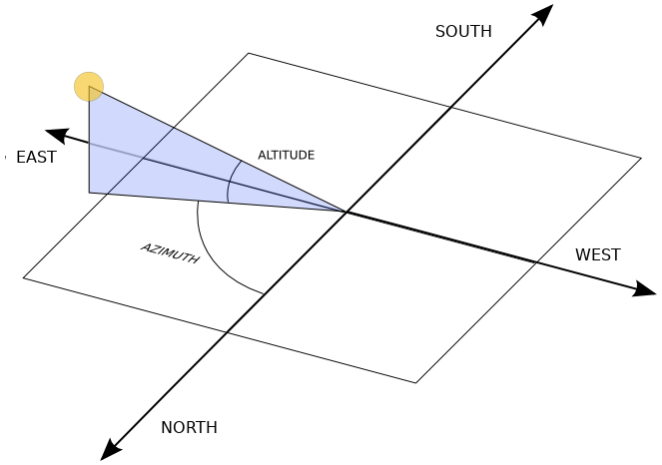


Figure 5-47 Solar azimuth / elevation from [99]

In countries in the north/south hemisphere like Canada, these changes are very noticeable (with the sun elevation being the most noticeable) and could be considered as an accident risk factor, as the sun position can impact the driver visibility (i.e. driving towards west in the afternoon).

Using the Python open source library *Pysolar* [99], the solar azimuth and elevation was calculated for each collision sample, using the coordinates and the date/time information. The general pseudo-code is presented in the Algorithm 5-17.

Algorithm 5-17: Generation of solar azimuth elevation features (General)

```
1: For each collision sample do
2:         Calculate and add the SOLAR_AZIMUTH as a new feature
3:         Calculate and add the SOLAR_ELEVATION as a new feature
4: End For
```

Figure 5-48 illustrates the distribution of records by solar elevation / azimuth per month and per hour of the day. One can notice very clearly that the sun azimuth / elevation changes during the year, with June having the longest days with sunlight (solar elevation > 0) and December the shortest. As expected, the solar azimuth changes a little during the year. In terms of the number of records, it is not easy to

identify a trend. The list of all the features added to each collision sample, including the ones related with this process, are shown later, in section 5.4, Table 5-4.

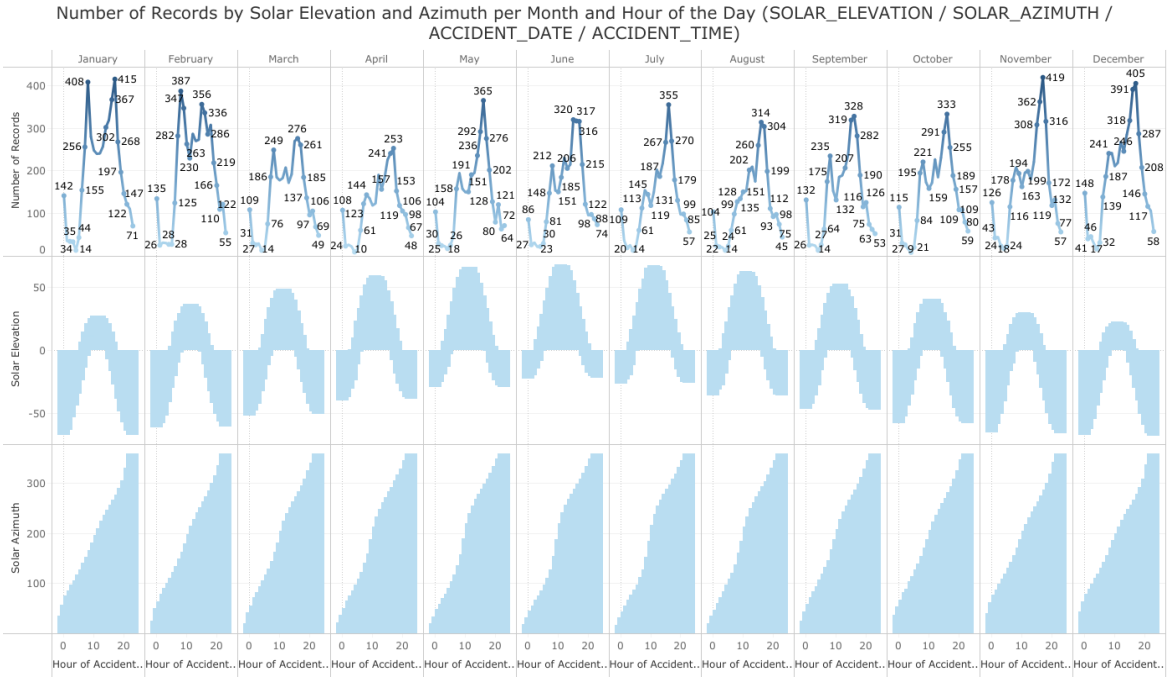


Figure 5-48 Number of records by solar elevation / azimuth per month and hour of the day (SOLAR_EVEVATION / SOLAR_AZIMUTH / ACCIDENT_DATE / ACCIDENT_TIME)

5.3.2.5. Date/Time Features Generation

As we noticed in the statistics visualizations, some date/time combinations are more prone to collisions: weekdays (more than weekends) between 7 – 9 a.m. and 3 – 5 p.m., when the largest number of people start/stop their journeys to study or work. In order to allow the classification model to identify and use this information, a series of new features was created related with the accident date and time.

The general pseudo-code to generate the date/time features is described in the Algorithm 5-18. Again, Table 5-4 in section 5.4 summarizes all the features added to each collision sample, including the ones related with this process.

Algorithm 5-18: Generation of date/time features (General)

1: **For** each collision sample **do**
2: Calculate and add the ACCIDENT_DATE_daym as a new feature
3: Calculate and add the ACCIDENT_DATE_dayw as a new feature
4: Calculate and add the ACCIDENT_DATE_dayy as a new feature
5: Calculate and add the ACCIDENT_DATE_month as a new feature

6: Calculate and add the ACCIDENT_DATE_weekm as a new feature
7: Calculate and add the ACCIDENT_DATE_weeky as a new feature
8: Calculate and add the ACCIDENT_DATE_year as a new feature
9: Calculate and add the ACCIDENT_TIME_hour as a new feature
10: **End For**

5.3.2.6. Events Features Generation

Because one of the objectives of this work is to validate if social events have an impact on the collisions, a series of different social events was used to tag the *collision samples* dataset with various features including university calendar (fall, winter, summer), major city sport events (hockey, football, baseball and soccer) and city holidays (i.e. Canada Day, Victoria Day, etc.)

Using the dates related with each event feature, a new binary feature was created. The general pseudo-code to generate the events features is described in the Algorithm 5-19. Please refer to section 5.4, Table 5-4 for the complete list of features added to each collision sample, including the ones related with this process.

Algorithm 5-19: Generation of events features (General)

1: **For** each event feature (ef) **do**
2: Add a new feature to all collision samples and initialize with 0
3: **For** each date in ef **do**
4: Find the collision samples (cs) that happened in that date
5: Update the feature value = 1 for the collision samples (cs)
6: **End For**
7: **End For**

5.3.2.7. Location Features Generation

As part of the features included in the main *collision sample* dataset, the intersection information related with the collision is provided (LOCATION). Because the intersection considers two or more streets, a split of the intersection information was made to create new features. The following example illustrates the split process:

LOCATION = *"HUNT CLUB RD btwn MALAK ST & ESSON ST"*

- STREET1 = *"HUNT CLUB RD"*
- STREET2 = *"btwn MAKAK ST"*
- STREET3 = *"& ESSON ST"*
- LOCATION_A = *"HUNT CLUB RD"*

- LOCATION_B = "*btwn MALAK ST & ESSON ST*"

The general process to generate the location features is described in the Algorithm 5-20.

Algorithm 5-20: Generation of location features (General)

```
1: For each collision sample do
2:         Calculate LOCATION_A and LOCATION_B and add them as new features
3:         Calculate STREET1, STREET2, STREET3 and add them as new features
4: End For
```

5.3.2.8. Math Features Generation

Using a series of existing features, the percentage of collisions samples, as a kind of collision weight factor, is calculated. For example, in the case of the ROAD_SEGMENT, each *collision sample* will have a new feature COLLISION_RATE_road_segment that counts the collision rate (number of collision per road segment / total collisions) over the same ROAD_SEGMENT. The general pseudo-code to generate the math features is listed in Algorithm 5-21.

Algorithm 5-21: Generation of mathematical features (General)

```
1: Calculate TOTAL_COLLISIONS
2: CONSIDERED_FEATURES = {ROAD_SEGMENT, ROAD_NAME, LOCATION_A, LOCATION_B, ONS_NAME, … etc.}
3: For each considered feature (cf) in CONSIDERED_FEATURES do
4:         For each unique value (uv) in cf do
5:                 Find the collision samples (cs) that belong to uv
6:                         Calculate NUMBER_OF_COLLISIONS
7:         Calculate COLLISION_RATE (NUMBER_OF_COLLISIONS / TOTAL_COLLISIONS) and add it to each
           collision sample (cs)
8:         End For
9: End For
```

5.4. Data Integration

As it was mentioned in section 5.1.2, the use of Python and of different software libraries in the complete development process (mainly *Pandas* and *GeoPandas*) gives us the benefit to handle from the beginning a unified dataset in all the different phases. As such, the final dataset is already integrated and it is not necessary to perform any additional processing.

Table 5-4 shows the entire set of features (83) related to each collision sample.

Table 5-4 Features related to each collision sample

Feature Type	Description

Original features (10)	• Date/Time Related (2):
	  ○ ACCIDENT_DATE
	  ○ ACCIDENT_TIME
	• Location Related (4):
	  ○ LOCATION
	  ○ XCOORD, YCOORD
	  ○ ACCIDENT_LOCATION
	• Environment Related (2):
	  ○ ENVIRONMENT_CONDITION
	  ○ LIGHT
	• Road Related (1):
	  ○ ROAD_SURFACE_CONDITION
	• Traffic Related (1):
	  ○ TRAFFIC_CONTROL

Road features generated (9)	• ROAD_NAME • ROAD_SEGMENT • ROAD_LEN • ROAD_SUBCLASS • ROAD_SUBTYPE • ROAD_SINOUSITY • ROAD_DIRECTION • ROAD_LEN_LOG • ROAD_SINUOSITY_LOG
Neighborhood / district features generated (3)	• ONS_NAME • ONS_ID • TILE_ID
Solar azimuth / elevation features generated (2)	• SOLAR_AZIMUTH • SOLAR_ELEVATION
Date/time features generated (8)	Related with ACCIDENT_DATE • ACCIDENT_DATE_daym: the day of the month (1-31) • ACCIDENT_DATE_dayw: the day of the week (1-7) • ACCIDENT_DATE_dayy: the day of the year (1-366) • ACCIDENT_DATE_month: the month of the year (1-12) • ACCIDENT_DATE_weekm: the week of the month (1-6) • ACCIDENT_DATE_weeky: the week of the year (1-55) • ACCIDENT_DATE_year: the year (2015-2017) Related with ACCIDENT_TIME • ACCIDENT_TIME_hour: the hour of the day (0-23)
Events features generated (11)	• CARLETON_CALENDAR_FALL • CARLETON_CALENDAR_WINTER • CARLETON_CALENDAR_SUMMER • OTTAWA_FURY_SOCCER • OTTAWA_SENATORS_HOCKEY • OTTAWA_67S_HOCKEY • OTTAWA_REDBLACKS_FOOTBALL • OTTAWA_RAVENS_FOOTBALL • OTTAWA_GEEGEES_FOOTBALL

	• OTTAWA_CHAMPIONS_BBC • OTTAWA_STATUTORY_HOLIDAYS
Location features generated (5)	• LOCATION_A • LOCATION_B • STREET1 • STREET2 • STREET3

Math features generated (36)	• Date/Time Related (8):
	  ○ COLLISION_RATE_accident_date_daym,
	  ○ COLLISION_RATE_accident_date_dayw,
	  ○ COLLISION_RATE_accident_date_dayy,
	  ○ COLLISION_RATE_accident_date_month,
	  ○ COLLISION_RATE_accident_date_weeky,
	  ○ COLLISION_RATE_accident_date_weekm,
	  ○ COLLISION_RATE_accident_date_hour,
	  ○ COLLISION_RATE_accident_date_year
	• Location Related (6):
	  ○ COLLISION_RATE_accident_location,
	  ○ COLLISION_RATE_location_a,
	  ○ COLLISION_RATE_location_b,
	  ○ COLLISION_RATE_street1,
	  ○ COLLISION_RATE_street2,
	  ○ COLLISION_RATE_street3
	• Environment Related (2):
	  ○ COLLISION_RATE_environment_condition
	  ○ COLLISION_RATE_light
	• Road Related (6):
	  ○ COLLISION_RATE_road_surface_condition
	  ○ COLLISION_RATE_road_segment,
	  ○ COLLISION_RATE_road_direction,
	  ○ COLLISION_RATE_road_name,
	  ○ COLLISION_RATE_road_subclass,
	  ○ COLLISION_RATE_road_subtype
	• Traffic Related (1)
	  ○ COLLISION_RATE_traffic_control
	• District/Neighborhoods Related (2):
	  ○ COLLISION_RATE_ons_name,
	  ○ COLLISION_RATE_tile_id
	• Carleton University Related (3):
	  ○ COLLISION_RATE_carleton_calendar_fall,
	  ○ COLLISION_RATE_carleton_calendar_winter,
	  ○ COLLISION_RATE_carleton_calendar_summer
	• Major Sport / Holidays Events Related (8):
	  ○ COLLISION_RATE_ottawa_fury_soccer,
	  ○ COLLISION_RATE_ottawa_senators_hockey,
	  ○ COLLISION_RATE_ottawa_67s_hockey,
	  ○ COLLISION_RATE_ottawa_redblacks_football,
	  ○ COLLISION_RATE_ottawa_ravens_football,
	  ○ COLLISION_RATE_ottawa_geegees_football,
	  ○ COLLISION_RATE_ottawa_champions_bbc,
	  ○ COLLISION_RATE_ottawa_statutory_holidays

Figure 5-49 shows the Python output related with the complete set of features included in the dataset and the counter of valid (not Null) values.

```
ACCIDENT_DATE                          86988 non-null object
ACCIDENT_LOCATION                      86988 non-null object
ACCIDENT_TIME                          86988 non-null object
ENVIRONMENT_CONDITION                  86988 non-null object
LIGHT                                  86988 non-null object
LOCATION                               86988 non-null object
ONS_ID                                 86988 non-null object
ONS_NAME                               86988 non-null object
ROAD_DIRECTION                         86988 non-null object
ROAD_LEN                               86988 non-null object
ROAD_NAME                              86988 non-null object
ROAD_SEGMENT                           86988 non-null object
ROAD_SINUOSITY                         86988 non-null object
ROAD_SUBCLASS                          86988 non-null object
ROAD_SUBTYPE                           86988 non-null object
ROAD_SURFACE_CONDITION                 86988 non-null object
TILE_ID                                86988 non-null object
TRAFFIC_CONTROL                        86988 non-null object
XCOORD                                 86988 non-null object
YCOORD                                 86988 non-null object
ACCIDENT_DATE_daym                     86988 non-null object
ACCIDENT_DATE_dayw                     86988 non-null object
ACCIDENT_DATE_dayy                     86988 non-null object
ACCIDENT_DATE_month                    86988 non-null object
ACCIDENT_DATE_weeky                    86988 non-null object
ACCIDENT_DATE_hour                     86988 non-null object
ACCIDENT_DATE_year                     86988 non-null object
OTTAWA_FURY_SOCCER                     86988 non-null object
OTTAWA_SENATORS_HOCKEY                 86988 non-null object
OTTAWA_67S_HOCKEY                      86988 non-null object
OTTAWA_REDBLACKS_FOOTBALL              86988 non-null object
OTTAWA_RAVENS_FOOTBALL                 86988 non-null object
OTTAWA_GEEGEES_FOOTBALL                86988 non-null object
OTTAWA_STATUTORY_HOLIDAYS              86988 non-null object
OTTAWA_CHAMPIONS_BBC                   86988 non-null object
SOLAR_AZIMUTH                          86988 non-null object
SOLAR_ELEVATION                        86988 non-null object
CARLETON_CALENDAR_WINTER               86988 non-null object
CARLETON_CALENDAR_SUMMER               86988 non-null object
CARLETON_CALENDAR_FALL                 86988 non-null object
ROAD_SINUOSITY_LOG                     86988 non-null object
ROAD_LEN_LOG                           86988 non-null object
LOCATION_A                             86988 non-null object
LOCATION_B                             86988 non-null object
STREET1                                86988 non-null object
STREET2                                86988 non-null object
STREET3                                40521 non-null object
ACCIDENT_DATE_weekm                    86988 non-null object
COLLISION_RATE_accident_date_daym      86988 non-null object
COLLISION_RATE_accident_date_dayw      86988 non-null object
COLLISION_RATE_accident_date_dayy      86988 non-null object
COLLISION_RATE_accident_date_month     86988 non-null object
COLLISION_RATE_accident_date_weeky     86988 non-null object
COLLISION_RATE_accident_date_weekm     86988 non-null object
COLLISION_RATE_accident_date_hour      86988 non-null object
COLLISION_RATE_accident_date_year      86988 non-null object
COLLISION_RATE_accident_location       86988 non-null object
COLLISION_RATE_location_a              86988 non-null object
```

```
COLLISION_RATE_location_b                  86988 non-null object
COLLISION_RATE_street1                     86988 non-null object
COLLISION_RATE_street2                     86988 non-null object
COLLISION_RATE_street3                     86988 non-null object
COLLISION_RATE_environment_condition       86988 non-null object
COLLISION_RATE_light                       86988 non-null object
COLLISION_RATE_road_surface_condition      86988 non-null object
COLLISION_RATE_road_segment                86988 non-null object
COLLISION_RATE_road_direction              86988 non-null object
COLLISION_RATE_road_name                   86988 non-null object
COLLISION_RATE_road_subclass               86988 non-null object
COLLISION_RATE_road_subtype                86988 non-null object
COLLISION_RATE_traffic_control             86988 non-null object
COLLISION_RATE_ons_name                    86988 non-null object
COLLISION_RATE_tile_id                     86988 non-null object
COLLISION_RATE_carleton_calendar_fall      86988 non-null object
COLLISION_RATE_carleton_calendar_winter    86988 non-null object
COLLISION_RATE_carleton_calendar_summer    86988 non-null object
COLLISION_RATE_ottawa_fury_soccer          86988 non-null object
COLLISION_RATE_ottawa_senators_hockey      86988 non-null object
COLLISION_RATE_ottawa_67s_hockey           86988 non-null object
COLLISION_RATE_ottawa_redblacks_football   86988 non-null object
COLLISION_RATE_ottawa_ravens_football      86988 non-null object
COLLISION_RATE_ottawa_geegees_football     86988 non-null object
COLLISION_RATE_ottawa_champions_bbc        86988 non-null object
COLLISION_RATE_ottawa_statutory_holidays   86988 non-null object
```

Figure 5-49 Complete dataset with features and counter of valid (not null) values

We can observe that the generated feature STREET3 (highlighted in red) includes many missing values. This is due to the fact that this feature is only present in the samples where the original LOCATION feature is related to a collision that happens at a three-street intersection or on a main street between two intersecting roads (less than 50% of the total samples). This is the reason why this feature and the related ones (STREET3, COLLISION_RATE_street3) were removed from the model.

5.5. Data formatting

A data preprocessing/transformation is necessary prior to the creation of different machine learning classifiers. Three main preprocessing operations were performed based on the feature type: (1) for numerical continuous features (i.e. XCOORD, YCOORD, ROAD_LEN, ROAD_SINUOSITY, SOLAR_AZIMUTH, SOLAR_ELEVATION, ROAD_LEN_LOG, ROAD_SINUOSITY_LOG, COLLISION_RATE_, etc.), a standardization process was applied (i.e. removing the mean and scaling to unit variance), (2) for categorical features (i.e. ACCIDENT_LOCATION, ENVIRONMENT_CONDITION, LIGHT, LOCATION_A, LOCATION_B, etc.), a label encoding was applied (encode label with

value between 0 and number of classes - 1), and (3) for binary features (i.e. CARLETON_CALENDAR_FALL, OTTAWA_FURY_SOCCER, OTTAWA_REDBLACKS, etc.), a binary transformation was applied (label encoded with 0, 1 values). All these transformations were performed using the *scikit-learn* functions *StandardScaler, LabelEncoder* and *Binarizer* [100]. In case of the features related with Date/Time (i.e. ACCIDENT_DATE_weekm, ACCIDENT_DATE_month, etc.), their values were kept in their original format.

A target variable (COLLISION*)* was generated for classification, as shown in Algorithm 5-22.

Algorithm 5-22: Generation of target classification feature (General)

```
1: For each collision sample do
2:         If CLASSIFICATION_OF_ACCIDENT is equal to '00 – No accident' then
3:                 COLLISION = 0
4:         Else
5:                 COLLISION = 1
6:         End If
           Add COLLISION as a new feature to the collision sample
7: End For
```

6. Prototype Implementation: Modeling, Model Comparison and Evaluation

Once the dataset is prepared, the next step is data modeling. As part of the objectives of this work, the task to predict traffic collisions and their type (accident severity) falls into the machine learning category of *classification*, and because the data already include the class to predict, is also considered a *supervised learning* task.

The case of traffic collisions prediction is a *binary classification* task (only two classes involved: no collisions/collisions, normally identified as class 0/1) and in the case of accident severity is a *multi-class classification* task (more than two classes involved: fatal, injury and property damage). Typically, a multi-class classification can be transformed into several binary classification problems using the well-known approach of *One vs Rest*. To do this, a series of new target features need to be created (one per class). In our case, three new target features, namely Fatal vs Rest (includes Injury and Property Damage), Injury vs Rest (includes Fatal and Property Damage) and Property Damage vs Rest (includes Fatal and Injury) were created.

The typical approach to solve any machine learning binary classification problem consists in choosing a series of algorithms (commonly referenced to as a *binary classifiers*) that identify pattern or trends (the *learning* process) from a subset of the data (training set) that could be used to *predict* the class for the unseen samples (test set). As outcome, the binary classifier provides the *probability* for each unseen sample as a measure of classification certainty; a *probability* value close to 0 means that the binary classifier has more certainty to *predict* 0 as the class (i.e. no collision), while a *probability* value close to 1 means that it is more likely to predict class 1 (i.e. collision). To classify each sample as class 0 or 1, a cut-off threshold (also named as *discriminant threshold*) is used. By default, the discriminant threshold is set as 0.5. This means, all the unseen samples with a *predicted* probability higher or equal to 0.5 are classified as class 1, and the others are classified as class 0. The default value of 0.5 for the discriminant threshold could however not be the optimal choice and could

increase the misclassification rate. Because of this, a method to choose the best value for the discriminant threshold is necessary.

To evaluate and compare the models to choose the *best classification performance*, it is necessary to use classification performance metrics. In this chapter, we will: (1) describe the performance classification metrics that could help us to decide how to select the best threshold and assess the classification performance of the different models, (2) define the strategy to compare the models and choose the discriminant threshold, (3) choose a series of machine learning algorithms to create different classifiers, and (4) define the way to test them and the process to build the classifiers and compare their results (build/assess).

6.1. Metrics to Evaluate the Performance of a Binary Classifier

In this section, we will discuss about the most important metrics that can be used to evaluate the performance of a classifier and choose the discriminant threshold.

6.1.1. Confusion Matrix

One of the most commonly used metrics, the confusion matrix, is a table that shows the performance of a classification algorithm through the relationship between the true (real) and predicted class. Table 6-1 shows a representation of a typical confusion matrix.

Table 6-1 Confusion matrix

		Predicted Class		
		Class 0	**Class 1**	
True Class	**Class 0**	TN	FP	TN + FP
	Class 1	FN	TP	FN + TP
		TN + FN	FP + TP	

The number of samples that belong to the Class 0 (true/real class) that were predicted *correctly* as Class 0, are called True Negatives (TN). The number of samples that belong to the Class 0 (true/real class) that were predicted *wrongly* as Class 1, are called False Positives (FP). Similar, the number of samples that belong to the Class 1 (true/real class) that are classified *correctly* as Class 1, are called True Positives (TP); and finally, the False Positives (FP) are the number of samples predicted *wrongly* as Class 1 (there are real Class 0). The use of *'Positive'* and *'Negative'* is because the Class 0 is considered the *Negative* class and the Class 1 the *Positive*. In this work, the Class 0 corresponds to '*NO COLLISION*' and Class 1 to '*COLLISION'.*

The total number of samples that belong to each *true/real* class can be calculated adding the rows of the table, with TN + FP representing the total number of samples that belongs to the true/real Class 0; and FN + TP the ones that belongs to the true/real Class 1. Considering the table columns, TN + FN provides the total number of samples *predicted* as Class 0; and FP + TP the ones *predicted* as Class 1. It is trivial to notice that a *perfect negative correlation* (-1) exists between TN and FP (a one-unit increase in TN produces a one-unit FP decrease) and also between FN and TP. Similarly, the number of *predicted positives* and *predicted negatives* are also *perfect negatively correlated*. The equations (1) to (6) show the previous definitions:

$$real\ Positives\ (Class\ 0) = TN + FP \tag{1}$$

$$real\ Negatives\ (Class\ 1) = FN + TP \tag{2}$$

$$Correlation\ (TN, FP) = Correlation\ (FN, TP) = -1 \tag{3}$$

$$predicted\ Positives\ (Class\ 0) = TN + FN \tag{4}$$

$$predicted\ Negatives\ (Class\ 1) = FP + TP \tag{5}$$

$$Correlation(predicted\ Positives, predicted\ Negatives) = -1 \tag{6}$$

It is clear to notice that the best classifier is the one that *maximizes* the number of *correctly predicted* samples (TN + TP) while *minimizing* the *wrong* ones (FN and FP).

Using these parameters (TN, TP, FN and TP), a series of classification rates and useful performance curves can be defined, as described in the following sections.

6.1.2. True Positive Rate (TPR - Sensitivity – Recall) and True Negative Rate (TNR - Specificity – Selectivity)

The *proportion* of *real* positives (Class 1) samples that was *correctly* classified as Class 1 (TP), is named *True Positive Rate (TPR)* (referenced also in the literature as *sensitivity* or *recall*). Similarly, the *proportion* of *real* negatives (Class 0) that was *correctly* classified as Class 0 (TN), is named *True Negative Rate (TNR)* (referenced in the literature as *specificity* and *selectivity)*. The equations (7) and (8) show the related definitions:

$$TPR = Recall = \frac{TP}{TP + FN} \tag{7}$$

$$TNR = Selectivity = \frac{TN}{TN + FP} \tag{8}$$

Substituting equations (1) and (2) in (7) and (8), the *recall* could be considered as a measure of TP over real positives; and *selectivity* a measure of TN over real negatives. The equations (9) and (10) illustrate these:

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{real\ Negatives\ (Class\ 1)} \tag{9}$$

$$Selectivity = \frac{TN}{TN + FP} = \frac{TN}{real\ Positives\ (Class\ 0)} \tag{10}$$

In our work, the *recall* represents the *proportion* of the real *COLLISION* samples that was classified *correctly* as a *COLLISION*; and the *selectivity* represents the *proportion* of the real *NO COLLISION* samples that was classified *correctly* as *NO COLLISION*, in other words, how good is the model in identifying the collisions vs. no collisions.

As the range of values of TPR/TNR is between 0 and 1, a bigger value is better (closer to 1). To maximize the value of recall in equation (9), the objective is to maximize the number of TP and minimize the number of FN; in the same context, maximizing the TN and minimizing the FP maximizes the value of selectivity in equation (10). If FN is equal to zero, the recall achieves the maximum score (1.0), the same applies for FP and selectivity.

Considering that each of these metrics scores the rate of *correctly* predicted per *real* class, they must be used along with other metrics to obtain a complete classification performance view of the model.

6.1.3. Positive Predictive Value and Negative Predictive Value

The *proportion* of samples *predicted as* Positives (Class 1) that was *correctly* classified as Class 1 (TP), is named *Positive Predictive Value (PPV)* (referenced also in the literature as *precision*). Equivalently the *proportion* of samples *predicted as* negatives (Class 0) that was *correctly* classified as Class 0 (TN), is named *Negative Predictive Value (NPV)*. The equations (11) and (12) show the related definitions:

$$PPV = Precision = \frac{TP}{TP + FP} \tag{11}$$

$$NPV = \frac{TN}{TN + FN} \tag{12}$$

Substituting equations (3) and (4) into (11) and (12), the *precision* could be considered as a measure of *TP* per *predicted positives*; and *NPV* a measure of *TN* per *predicted negatives*. The equations (11) and (12) illustrate these:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{predicted\ negatives\ (Class\ 1)} \quad (13)$$

$$NPV = \frac{TN}{TN + FN} = \frac{TN}{predicted\ positives\ (Class\ 0)} \quad (14)$$

In our work, the *precision* represents the *proportion* of samples *predicted* as *COLLISION* that were classified *correctly* as *COLLISION*; and the *NPV* represents the *proportion* of *predicted NO COLLISION* samples that were classified *correctly* as *NO COLLISION*, in other words, the probability that a *predicted* class is correct.

The range of values of *PPV/NPV* is between 0 and 1, where the *bigger is better* (closer to 1). To *maximize* the value of precision in equation (13), the objective is to maximize the number of TP and minimize the number of FP*;* in the same context, maximize the TN and minimize the FN will maximize the equation (14). If FP is equal to zero, the *precision* achieves the maximum score (i.e. 1), the same applies for FN and NPV.

Because according to equation (6) *predicted negatives* and *predicted positives* are *negative perfectly correlated*, an increase in precision will produce a decrease in NPV, and thus a trade-off needs to be considered.

It is important to mention that in the case of imbalanced datasets, the precision and NPV scores could be misleading. An example is shown in Table 6-2. Supposing a 1:10 relationship between no collision and collisions samples (10 no collisions and 100 collision samples), and assuming that the classifier is able to predict *correctly* 50% of the samples (TN=5, FP = 5 and FN = 50, TP = 50), the PPV and NPV will be equal to:

$$Precision = \frac{TP}{TP+FP} = \frac{50}{55} = 0.901 \quad (15)$$

$$NPV = \frac{TN}{TN+FN} = \frac{5}{55} = 0.0909 \quad (16)$$

Table 6-2 Confusion matrix example

		Predicted Class: no collision	Predicted Class: collision	
True Class	**no collision**	5	5	10
	collision	50	50	100
		55	55	

A value of 90% for PPV does not reflect the real classification performance (50%).

Also, as these metrics score the rate of *correctly* predicted per *predicted* class, they must be used with other metrics to obtain a complete view of the classification performance of the model.

6.1.4. Accuracy

The *accuracy (ACC)* is the proportion of real Negatives (Class 0) and real Positives (Class 1) classified correctly as Class 0 (TN) or Class 1 (TP) and measures the performance of the model considering both classes, as depicted in equation (17).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (17)$$

In this work, the accuracy represents the *proportion* of samples classified *correctly* as *COLLISION/NO-COLLISION*; in other words, a measure on how good is the model doing the correct classification or identifying the real class. It can be considered a combination of the previous rates.

Equivalent to previous measures, the range of values is between 0 and 1, where the *bigger is better* (closer to 1). To maximize the accuracy in equation (17),

the target is maximizing the number of TP and TN while minimizing the number of FP and FN. If FP and FN are equal to zero, the maximum accuracy is achieved (i.e. 1).

Because *accuracy* considers all elements in its calculation, it can be considered a good estimator of the complete model classification performance. The *accuracy* rate suffers from the same issues as precision and NPV in the case of imbalanced datasets, where the information provided could be misleading. Considering the example in section 6.1.3, the accuracy will be 50% (as seen in equation (18)).

$$Accuracy = \frac{55}{110} = 0.50 \tag{18}$$

6.1.5. F-score

The ***F-score*** is the *harmonic mean* of precision and recall. The *harmonic mean* is a measure used to calculate the average of two rates, as in equation (19) and (20).

$$harmonic\ mean\ (a_1, a_2, \dots, a_n) = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_n}} \tag{19}$$

$$\begin{aligned} F1\ score = harmonic\ mean\ (Precision, Recall) &= \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \\ &= 2 * \frac{Precision * Recall}{Precision + Recall} \end{aligned} \tag{20}$$

Because precision (true positives per predicted positives) and recall (true positives per real positives) are rates focused on the Positive Class (Class 1), the F-score is a measure of the classification performance for the Positive Class (Class 1); in our work, the COLLISION class.

The F1-score has a range between 0 and 1, where the bigger is better (closer to 1). To maximize the value of this score, the target is maximizing the product of precision and recall while minimizing their sum. It is important to notice that when precision is equal to recall, the F1 score is equal to them.

The F1 score is a better indicator than *accuracy* in case of imbalanced datasets. For the example of section 6.1.3, the F1 score will be 64.52%, as shown in equation (21).

$$F1\ score = 2 * \frac{0.901 * 0.091}{0.901 + 0.091} = 0.6452 \quad (21)$$

6.1.6. Area under Receiver Operating Characteristics Curve

The Receiver Operating Characteristic (ROC) curve shows the relationship between the recall and the False Positive Rate (FPR) for a range of different discriminant thresholds.

The *False Positive Rate (FPR)* is the proportion of wrongly predicted negatives (FP) over the total wrongly predicted positives (FP) and negatives (FN), is also referenced in the literature as *1 – Selectivity*, as in equation (22):

$$FPR = \frac{FP}{FP + FN} = 1 - Selectiviy \quad (22)$$

One can notice that the *FPR* range of values is between 0 and 1 and measures how far is the model to achieve a perfect *recall*, when the *lower* is the *better*.

Figure 6-1 shows an example of a ROC curve where the *x* axis represents the *FPR* and the *y* axis the recall. To plot the lines, for each value of *discriminant threshold* (a value between 0 and 1), the recall and FPR are calculated. The solid blue line represents the scores for the test set and the dotted blue line the scores for the train set. A higher recall and lower FPR identifies a good classification performance (minimum overlap between the prediction of two classes) with the best possible value at (recall = 1.0, FPR = 0.0), it means, a curve that is very close to the upper left corner of the graph (no overlap between the prediction of two classes). An orange solid line identified as *random guess* (recall = 0.5, FPR = 0.5), serves as base line (any classifier should be better than that). In this example, we can notice that the dotted blue line (train) is closer to the upper left corner than the solid blue line; because of that, the classification performance is better over the train set (as expected).

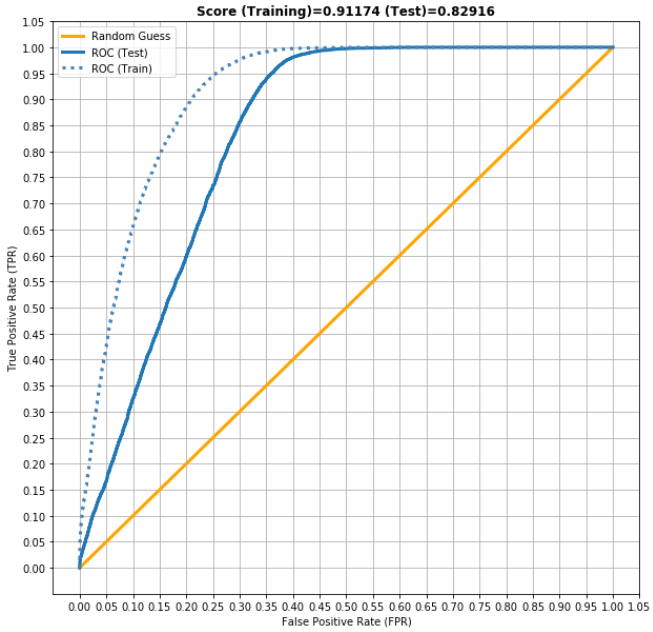


Figure 6-1 Receiver Operating Characteristic (ROC) Curve

As a measure of classification performance used to compare different models, the area under the ROC curve (AUC) provides a good indicator over the Positive Class, where the *bigger is better*, with a value of 1 for the perfect case (recall = 1.0 and FPR = 0.0). In this example, we can notice that the AUC for the dotted blue line (training) is higher (0.91) than the solid blue line (test, 0.83).

It is important to mention that because the ROC curve only considers the positive class (Class 1: collision), it could provide misleading information in case of imbalanced datasets. The AUC is threshold invariant; it measures the performance of the model without considering a specific discrimination threshold.

6.1.7. Area under Precision – Recall (PR) Curve

Similarly, the Precision – Recall (PR) curve, as its name suggest, shows the relationship between the precision and recall for a range of different discriminant thresholds.

Figure 6-2 shows an example of a Precision-Recall (PR) curve where the *x* axis represents the recall and the *y* axis the precision. Similar to previous case, to plot the lines for each value of the discriminant threshold (a value between 0 and 1), the precision and recall are calculated. The blue line represents the scores for the test set

and the orange line the scores for the train set. A higher recall and precision identify a good classification performance (minimum overlap between the prediction of two classes) with the best possible value at (precision = 1.0, recall = 1.0). It means that a good performance is denoted by a curve that is very close to the upper right corner of the graph (no overlap between the prediction of two classes). In this example, we can notice that the orange line (train) is closer to the upper right corner than the blue line, because of that, the classification performance is better over the train set (as expected).

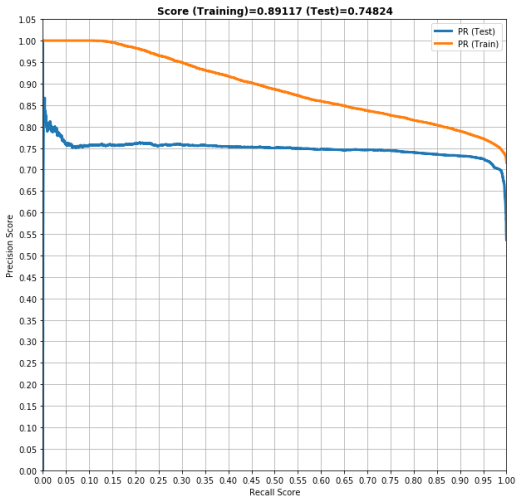


Figure 6-2 Precision – Recall (PR) Curve

As a measure of classification performance used to compare different models, the area under the precision-recall curve provides a good indicator, where *bigger is better*, with a value of 1 for the perfect case (precision = 1.0 and recall = 1.0). In this example, we can notice that the area under PR curve for the orange line (training) is higher (0.89) than the blue line (test, 0.74).

It is important to mention that, because the area under PR curve is based on precision and recall, it is a better indicator of performance over imbalanced datasets. Also, it is threshold invariant.

6.1.8. Youden J Index

Another important classification performance index is the Youden J index, as defined by the equation (23):

$$J = Recall + Selectivity - 1 = \frac{TP}{TP + FN} + \frac{TN}{TN + FP} - 1 \quad (23)$$

We can notice that if the number of FN and FP are equal to zero, the result is 1 (the best value). If the number of FN and FP is equal to TP and TN, or a value of 50% for recall and precision (i.e. a useless classifier, equivalent to random guess), the index value is 0 (the worst value).

The Youden J index is normally used as an indicator to select the optimum discrimination threshold value in the ROC curve. The J index is calculated for different thresholds and when it achieves its maximum score (closer to 1), the threshold associated is the one that provides the farthest vertical distance to random guess curve. Figure 6-3 shows the graphical representation of the Youden J index (in red) related with the ROC curve [101].

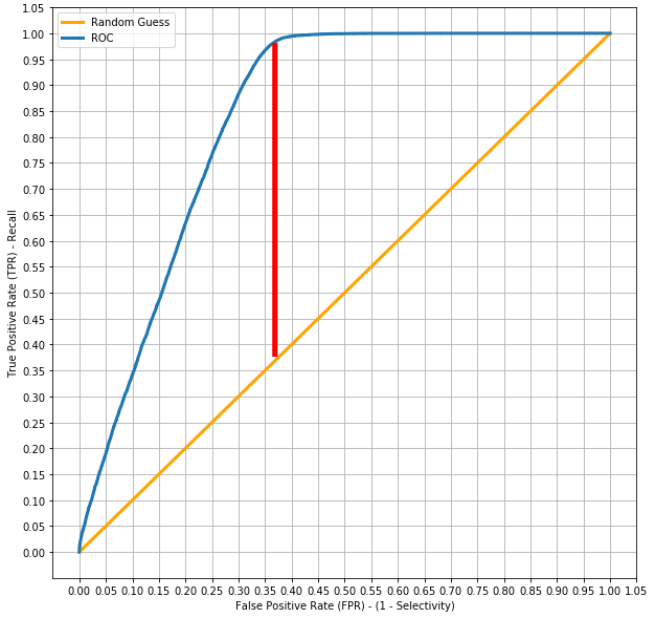


Figure 6-3 ROC curve Youden J index (red)

6.1.9. Matthews Correlation Coefficient (MCC)

The Matthews Correlation Coefficient (MCC) is a *correlation coefficient* between the *real* and *predicted* class, as shown in equation (24) [102]:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (24)$$

As a correlation coefficient, the range of values is between -1 and 1, where the higher its value, the better the performance. A value of 1 represents a *perfect positive correlation* between the *real* and *predicted* class (the best classifier), and a value of -1 a *perfect negative correlation* (the worst classifier). A result of 0 is equivalent to random guess.

Because it considers the *real* and *predicted* classification results for *both classes,* it can be considered a much better indicator, even for imbalanced datasets. It also can be used to choose the best discriminant threshold.

6.1.10. Maximum Precision – Recall Rate

The selection of the discrimination threshold that produces the best classification results (i.e. accuracy) could produce a classification model that gives more importance to the negative or positive class (or equal for both classes), and is totally dependent of the data and the problem to solve. One interesting classification scenario is the one where the positive and negative classes have the same importance, it means, we want to choose a discriminant threshold that produces the most balanced results: a very close value between TP and TN.

If we consider a problem where the number of positive and negative class samples are similar (i.e. a balanced dataset), we can notice that the threshold that produces the most balanced classification results is the one where precision and recall are equal, or graphically, where the precision and recall curves intersect.

To show this, let's assume that the number of TN is almost equivalent to TP, and because both classes are similar real positives are also almost equivalent to real

negatives, the number of FP and FN are also equivalent as shown in equations (25) and (26).

$$TP \sim TN \tag{25}$$

$$\begin{gathered} real\ Positives\ (Class\ 0) \sim real\ Negatives\ (Class\ 1) => FP + TN \sim FN + TP \\ => FP \sim FN \end{gathered} \tag{26}$$

Considering the equations (7) and (11), we can notice that TPR and PPV will be equal when FN = FP. To explain this using a graph, consider Figure 6-4. It includes two graphs, with the one on the top showing several classification curves (precision in yellow and recall in blue) and the one at the bottom representing the *TP Rate* (equation (27): number of positives compared with the total, represented with a dashed blue line) and *TN Rate* (equation (28), represented by a dashed yellow line). We can notice that when *TP* is equal to *TN*, *TP Rate* is equal to *TN Rate*, in the graph at the bottom we can see that both curves intersect at a specific threshold value (0.65). Also, we can see in the top graph that precision and recall curves intersect at the same threshold value.

$$TP\ Rate = \frac{TP}{TP + FN + TN + FP} \tag{27}$$

$$TN\ Rate = \frac{TN}{TP + FN + TN + FP} \tag{28}$$

Another interesting aspect to notice is that, in this case, any threshold value selected that is located to the left of the already mentioned threshold (< 0.65), will produce classification results with a higher number of *TP Rate* (a bigger number of *TP*) than *TN Rate* (a lower number of *TN*). In this case, the classification results will exhibit a larger recall value than the precision. As such, the classification model is giving more importance to predict correctly the positive class. Instead, if we choose any threshold larger than the one discussed (> 0.65), the results will be the opposite, a larger precision and a lower recall. This is important to be considered according to the classification objective.

6.2. Model comparison and discriminant threshold selection strategy

While a broad series of performance classification metrics are available with different advantages and disadvantages, none of them can be considered as a unique solution to decide which classification model is the best choice. Normally the selection of the metrics depends on the nature of the classification problem (i.e. existence of imbalance dataset). In this work, in order to compare the models, we will use two threshold independent metrics: the area under the ROC curve and the area under the precision-recall (PR) curve to compare the general model performance (without using a specific discrimination threshold) and a series of metrics to select the appropriate discriminant threshold to compare the classification performance. The metrics used to choose the thresholds are: the maximum accuracy score, the maximum F1 score, the maximum Youden J index score, the maximum Matthews Correlation Coefficient (MCC) score and the intersection between precision and recall scores, as described in section 6.1. The comparison of these will provide the basis to identify and choose the best classification model. Figure 6-4 shows at the top the different classification curves including five thresholds in the *x* axis identified by colored triangles: (1) maximum precision/recall, marked by a black triangle, (2) maximum accuracy, a brown triangle, (3) maximum F1 score, a green triangle, (4) maximum Youden J index, a red triangle, and (5) maximum Matthews coefficient, a pink triangle.

6.3. Modeling techniques selection

Considering the findings in the literature described in section 2.3, different machine learning classification algorithms were considered in the context of this thesis work, namely (1) random forest, (2) Adaboost, (3) gradient boosting tree and (4) multi-layer perceptron. These were identified among the most frequently used machine learning solutions in the context of traffic collision prediction.

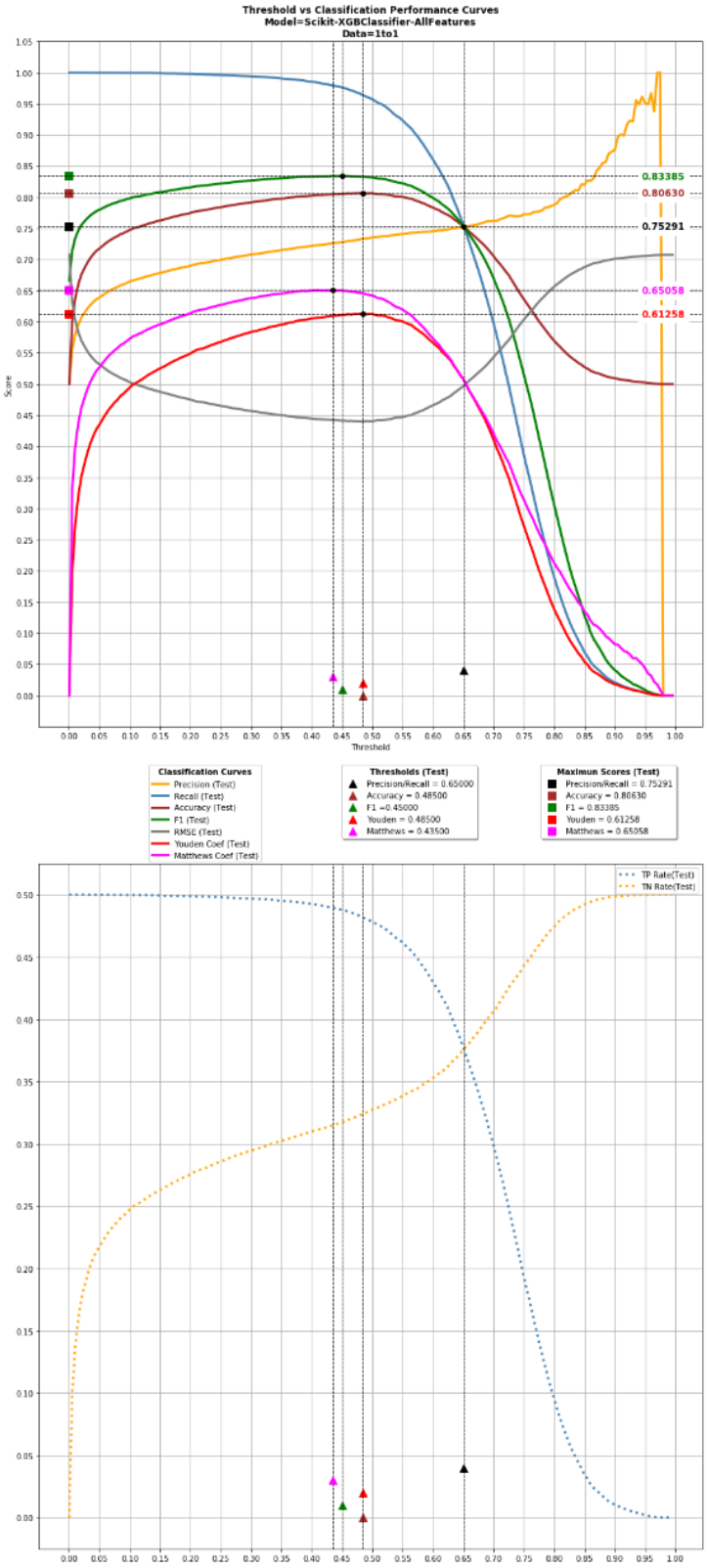


Figure 6-4 Classification performance plot curves - on the top: precision in yellow, recall in blue and at the bottom: TN rate (in dashed yellow) and TP rate (in dashed blue).

A *decision tree* (or classification tree), as its name suggests, is a machine learning method based on a tree structure constructed using a recursive partitioning of the data. During each partition, the objective is minimizing an error function between the child nodes. The idea behind decision trees is illustrated in Figure 6-5, where the graph on the left side shows the partitions and the tree on the right side, the classification model for each of the three classes (i.e. 1,2,3). The pseudo-algorithm for a decision tree is displayed in Algorithm 6-1. In this work, we make use of the scikit *DecisionTreeClassifier* to design and implement the decision tree classifier.

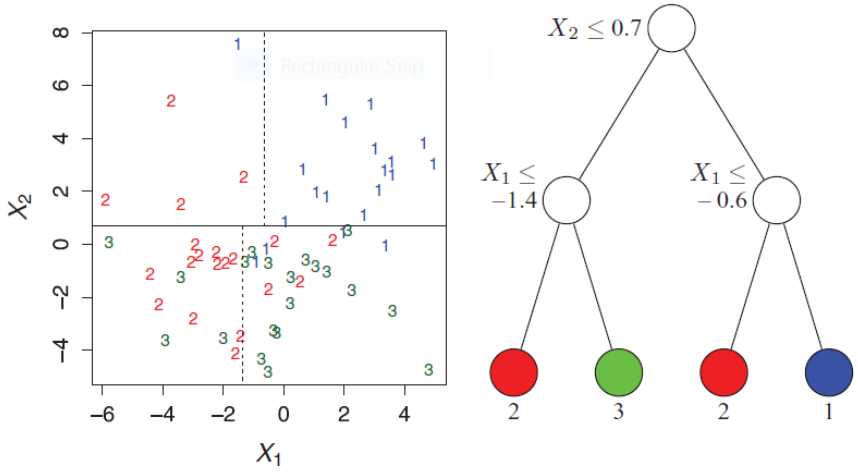


Figure 6-5 Decision tree graphical example (from [103])

Algorithm 6-1: Pseudo-code for tree construction by exhaustive search (adapted from [103])

1: Start at the root node
2: **For** each X **do**
3: Find the set S that minimizes the sum of the node impurities in the two child nodes and choose the split $\{X^* \in S^*\}$ that gives the minimum overall X and S.
4: **If** a stopping criterion is reached **then** Exit
6: **Else** apply step 2 to each child node in turn.
7: **End for**

Random forest (or random decision forest) is an ensemble-type of machine learning algorithm based on the construction of multiple decision trees at fitting time that grow in randomly selected subsets of the data. The final class (on classification problems) is chosen based on the majority of the votes. The pseudo code is displayed in Algorithm 6-2, while Figure 6-6 shows an illustration of the training and classification decision process. In Figure 6-6a, the complete training set which

contains positive (green labels) and negative (red labels) examples, is randomly subsampled to create each decision tree of the ensemble. Figure 6-6b shows the procedure related with the final class prediction. For each data point (X) the algorithm starts at the root node and traverses down each of the trees until a leaf node is reached, assigning positive (green) or negative (red) as final class to the instance. At the end, the majority of the votes (in this example red) is chosen as final class.

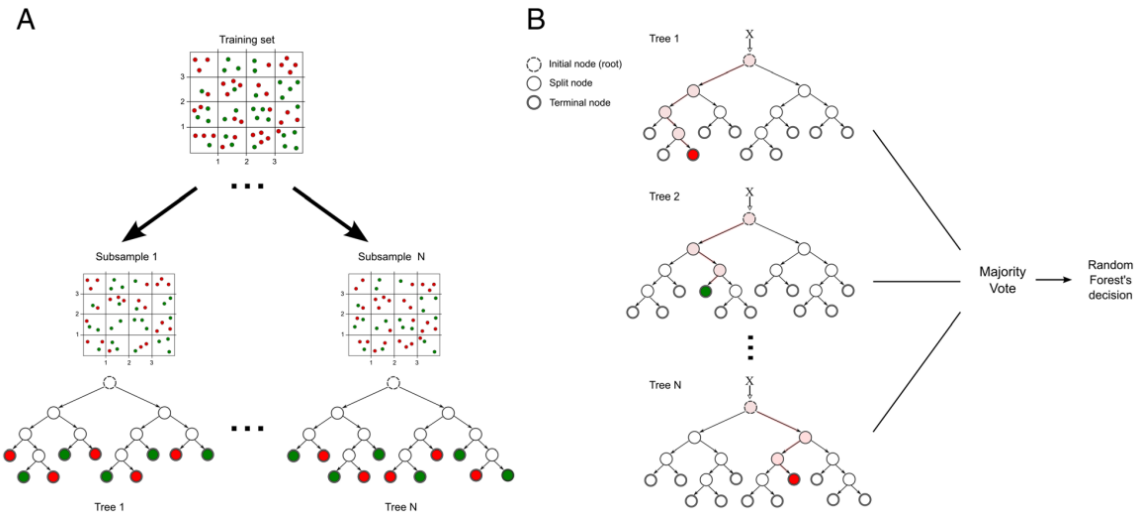


Figure 6-6 Random forest graphical example (a) training process and (b) classification decision process (from [104])

Algorithm 6-2: Random forest predicted value at x (adapted from [105])

Input: Training set D_n, number of trees $M > 0$, $m_{try} \in \{1,...,p\}$, $a_n \in \{1,...,n\}$, $t_n \in \{1,...,a_n\}$, and $x \in [0, 1]^p$.
Output: Prediction of the random forest at x.

For $j = 1,...,M$ **do**
  Select a_n points, without replacement, uniformly in D_n
  Set $P_0 = \{[0, 1]^p\}$ the partition associated with the root of the tree
  For all $1 \le l \le a_n$, set $P_l = \emptyset$
  Set $n_{nodes} = 1$ and level = 0
  While $n_{nodes} < t_n$ **do**
    If $P_{level} = \emptyset$ **Then**
      level = level + 1
    Else
      Let A be the first element in P_{level}
      If A contains exactly one point **Then**
        $P_{level} \leftarrow P_{level} \setminus \{A\}$
        $P_{level+1} \leftarrow P_{level+1} \cup \{A\}$
      Else
        Select uniformly, without replacement, a subset $M_{try} \subset \{1,...,p\}$ of cardinality m_{try}
        Select the best split in A by optimizing the CART-split criterion along the coordinates in M_{try} (see details below)

17: Cut the cell A according to the best split. Call A_L and A_R the two resulting cell
18: $P_{level} \leftarrow P_{level} \setminus \{A\}$
19: $P_{level+1} \leftarrow P_{level+1} \cup \{AL\} \cup \{AR\}$
20: $n_{nodes} = n_{nodes} + 1$
21: **End If**
22: **End If**
23: **End While**
24: Compute the predicted value $m_n(x; \Theta_j, Dn)$ at x equal to the average of the Y_i's falling in the cell of x in partition $P_{level} \cup P_{level+1}$
25: **End For**
26: Compute the random forest estimate $m_{M,n}(x; \Theta_1,\ldots, \Theta_M, D_n)$ at the query point x according to:

$$m_{M,n}(x;\ \Theta_1, \ldots, \Theta_M, D_n) = \frac{1}{M}\sum_{i=1}^{M} m_n(x;\ \Theta_j, D_n)$$

In this work, we make use of the scikit *RandomForestClassifier* to design and implement the random forest classifier.

Another example of ensemble learning-type machine learning model is *AdaBoost* (Adaptative Boosting). Boosting creates a strong classifier from multiple weak classifiers. It begins by fitting a weak classifier to the original dataset and then creating a second model to try to correct the error from the first one. Multiple instances are created and adjusted in order to focus on the more difficult cases. The algorithm and a graphical example are illustrated in Algorithm 6-3 and Figure 6-7. In Figure 6-7, the original dataset D_1 is used to train the first classifier, then for the examples that were wrongly classified (illustrated with an oval), a weighting process is applied to the data (D_2) such that the second classifier gives more importance to them. This process continues until the final classifier is produced.

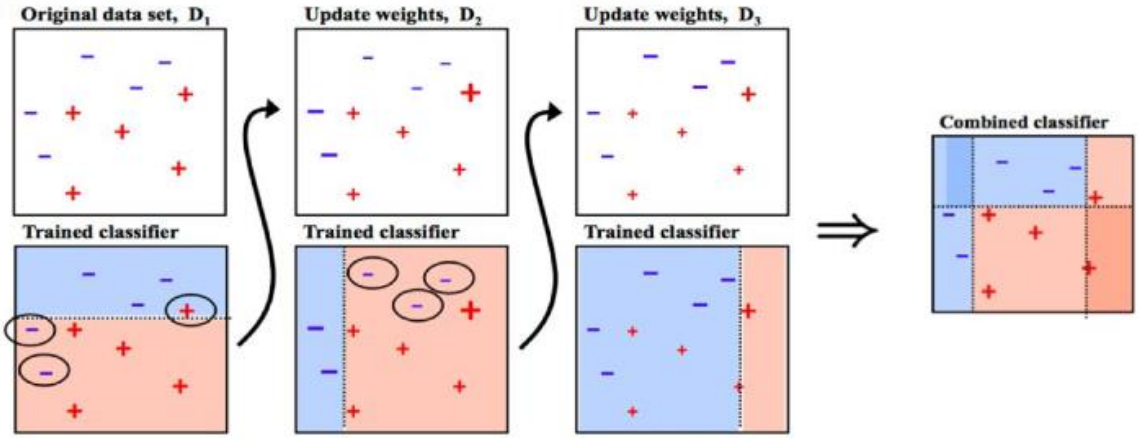


Figure 6-7 AdaBoost graphical example (from [106])

Algorithm 6-3: AdaBoost (adapted from [107])

Input:

sequence of N labeled examples $\{(x_1, y_1), ..., (x_N, y_N)\}$
distribution D over the N examples
weak learning algorithm *WeakLearn*
integer T specifying number of iterations

Output: the hypothesis

$$h_f(x) = \begin{cases} 1, & if \ \sum_{t=1}^{T} \log 1/\beta_t \ h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log 1/\beta_t \\ 0, & otherwise \end{cases}$$

1: Initialize the weight vector: $w_i^1 = D(i)$ for $i=1, ..., N$
2: **For** $t=1, 2, ..., T$ **do**
3: Set

$$p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}$$

4: Call **WeakLearn**, providing it with the distribution p^t; get back a hypothesis h_t: $X \rightarrow [0, 1]$
5: Calculate the error of h_t:

$$h_t: \varepsilon_t = \sum_{i=1}^{N} p_i^t \, |h_t(x_i) - y_i|$$

6: Set

$$\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$$

7: Set the new weights vector to be

$$w_i^{t+1} = w_i^t \beta_i^{1 - |h_t(x_i) - y_i|}$$

8: **End for**

In this thesis, we use as weak learner the scikit *DecisionTreeClassifier* as well as the *AdaBoostClassifier* implementation.

Gradient boosted tree is another ensemble-type of machine learning algorithm for a weaker prediction model like a decision tree. Just like any other boosting technique, gradient boosting algorithm combines the weak classifier into a stronger classifier in an iterative fashion. Gradient boosting builds the first decision tree on the training dataset to predict the samples, calculates the error as the difference between the real values and the outputs of the first decision tree and then uses this error to build an improved learner (decision tree) in the second stage. At every step, the residual of the error function is calculated using the gradient descent method and the new residual becomes a target variable for the subsequent iteration [108]. The idea behind gradient boosted tree is illustrated in Figure 6-8 and the algorithm pseudo-code in Algorithm 6-4.

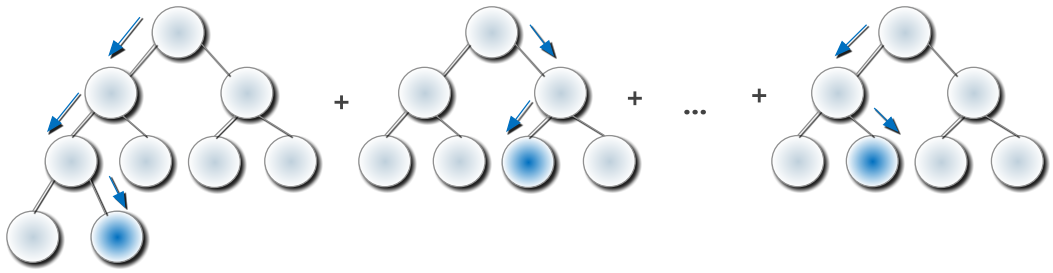


Figure 6-8 Gradient boosted tree structure (adapted from [109])

Algorithm 6-4: Algorithm for Gradient Boosted Decision Trees (adapted from [108] [110])

Input:

A set of data points $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ from the given dataset
A differentiate loss function $L(y, F(x))$
A number of iterations M
A week learner $h_m(x)$, usually decision trees

Output: An ensemble of weak learners, usually decision trees

1: Initialize the model with a constant value

$$F_0(x) = \arg\_\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$$ where $L(y, F(x))$ represents a loss function

2: **For** $m = 1$ *to* M **do**

3: Calculate pseudo-residuals $\hat{y}_i$

$$\hat{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad i = 1,2, \ldots, N$$

4: Fit a weak learner $h_m(x)$ to pseudo-residuals, train it using the training set $\{(x_1, \hat{y}_1), (x_2, \hat{y}_2), \ldots, (x_N, \hat{y}_N)\}$

5: Calculate the step magnitude multiplier ρ_m

$$\rho_m = \arg\_\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h_m(x_i))$$

6: Update the model

$$F_m(x) = F_{m-1}(x) + \rho_m h(x\,; a_m)$$

7: **End for**

8: Output $F_M(x)$

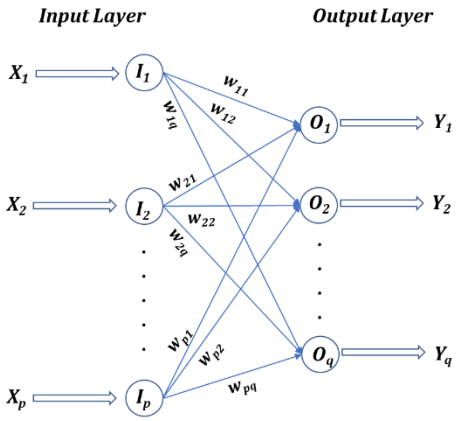
In this work, the scikit *XGBClassifier*, a gradient boosting tree implementation using the XGBoost [111] wrapper, is used to implement this machine learning algorithm.

Artificial Neural Networks, also known as Neural Networks (NNs) are designed to function like the human brain. Hence, the basic architecture of NNs is similar to that of the human brain, which is comprised of neurons and connections between them. NNs have neurons as processing units and connections among those processing units.

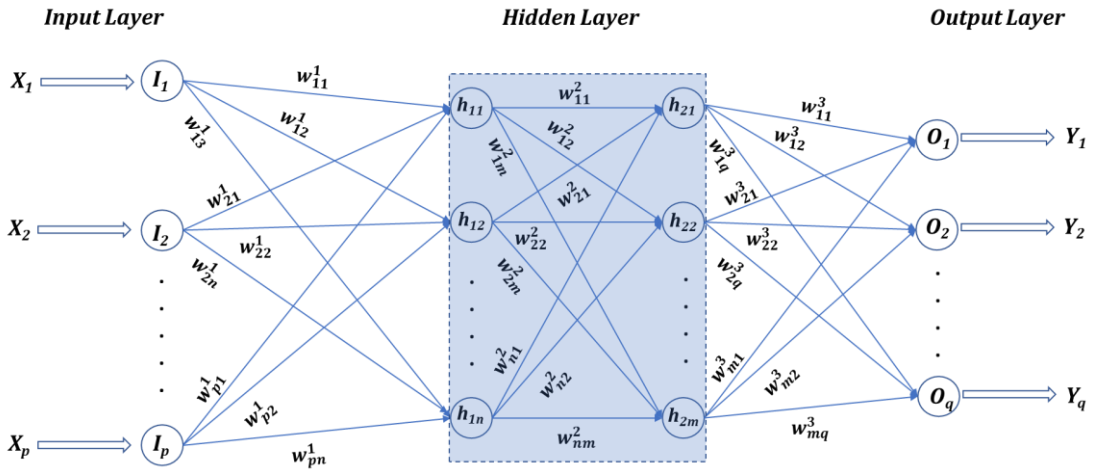
NNs are of two types: feed-forward NN and recurrent NN where the feed-forward NNs are commonly used. In feed-forward NNs, there are direct connections between the processing units of each layer at the hidden layer. Also, the output is not given as a feedback to the input. Whereas, in recurrent NNs, the output is given as feedback to the inputs and hence, the output at time t+1 depends on output at time t or on the state of the NN at time t.

A typical one layer fully connected feed-forward NN (perceptron) is shown in Figure 6-9a. Every processing unit in an NN has p inputs; $X_1, X_2, \ldots, X_p$ and q outputs, i.e., $Y_1, Y_2, \ldots, Y_q$. For feed-forward NNs, every processing unit of each layer is connected to a processing unit (i.e. neuron) in the next layer (input neurons $I_1, I_2, \ldots, I_p$ to output neurons $O_1, O_2, \ldots, O_q$). Each neuron assigns a weight to its input representing how correct or incorrect it is relative to the task being performed (w_{ij}). For classification problems, the number of neurons at the output corresponds to the number of class labels.

An extension of the basic one-layer perceptron network consists in adding hidden layers between input and output neurons, named multi-layer perceptron. Figure 6-9b illustrates and example of multi-layer perceptron network with two hidden layers and n and m neurons each one: $h_{11}, h_{12}, \ldots, h_{1n}$ and $h_{21}, h_{22}, \ldots, h_{2m}$. The weights between the *i-th* neuron at layer k and the *j-th* neuron and layer *k+1* is identified by w_{ij}^{k}.



(a)



(b)

Figure 6-9 Graphical representation of (a) perceptron network, and (b) multi-layer perceptron network with two hidden layers (adapted from [112], [113])

Multi-layer perceptrons are usually trained using the backpropagation algorithm, illustrated in Algorithm 6-5.

Algorithm 6-5: Algorithm for training neural networks using backpropagation (adapted from [112], [113])

Input:

A set of data points

Data = {$(x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)$} from the given dataset, where the input layer *I* has p neurons

I: $(x_1, x_2, \ldots, x_p)$

The number of hidden layers *s* and neurons per each layer *(m, n, …, t)*

H_1: $(h_{11}, h_{12}, \ldots, h_{1m})$

H_2: $(h_{21}, h_{22}, \ldots, h_{2n})$

. . . . .

H_s: $(h_{s1}, h_{s2}, \ldots, h_{st})$

The number of neurons *q* in output layer *O*

O: $(O_1, O_2, \ldots, O_q)$

A learning constant γ

Output: A trained neural network *NNET*

1: **Initialize** network weights w_{ij}^k often with small random values

$w_{ij}^k = random\ (0,1)$ for $k = 1, 2, \ldots, s+1$

Where $ij \in \{(11, 12, \ldots, 1m), (21, 22, \ldots., 2n), \ldots\ldots\ldots, (s1, s2, \ldots\ldots, st)\}$

2: **Do**

3: **For each** example *t* in $(x_1, x_2, \ldots, x_p)$ **do**

4: **Calculate** *predicted output* on *NNET* $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_p)$ *// forward phase*

5: **Calculate** Error function of the network *E*

$$E = \frac{1}{2} \sum_{i=1}^{p} |\hat{y}_i - y_i|^2$$

6: Minimize E by using iterative process and gradient descent, for which we need to calculate the gradient ∇E:

$$\nabla E = \left(\frac{\partial E}{\partial w_{11}^{1}}, \frac{\partial E}{\partial w_{12}^{1}}, \dots \dots, \frac{\partial E}{\partial w_{ij}^{k}}\right) \quad \text{for } k = 1,2, \dots, s+1$$
Where *ij* ∈ *{(11, 12, …, 1m), (21, 22, …., 2n), ……..,(s1, s2, ….., st)}*

7: **Calculate** Δw_{ij}^{k} for all weights in backward direction, starting from output layer O to hidden layer H_s, finishing with hidden layer H_1 to input layer I *// backward phase*

$$\Delta w_{ij}^{k} = -\gamma \frac{\partial E}{\partial w_{ij}^{k}} \quad \text{for } k = 1,2, \dots, s+1$$
Where *ij* ∈ *{(11, 12, …, 1m), (21, 22, …., 2n), ……..,(s1, s2, ….., st)}*

8: **Update** network weights w_{ij}^{k} on $NNET$

$$w_{ij}^{k+1} = w_{ij}^{k} + \Delta w_{ij}^{k}$$

9: **Until** all examples ($x_1, x_2, \dots, x_p$) classified correctly (*error == 0*) or another stopping criteria satisfied
10: **Return** $NNET$

In this thesis, we make use of the scikit *MLPClassifier* to implement the multi-layer perceptron architecture that is trained using backpropagation.

To tackle the class imbalance problem in the case of accident severity prediction (fatal, injury and property damage), in this work we make use of the *Synthetic Minority Over-sampling Technique (SMOTE)*. SMOTE is a statistical technique that increases the number of samples in the minority class by adding more instances for that class. The new samples added are not simply duplicated from the existing classes; instead the algorithm considers the feature space of the samples of each target class and its nearest neighbors. It produces new samples that have combined features of the target class and the nearest neighbor's features. This helps to improve the feature availability to each class and makes the samples more general. The algorithm works by taking the input and judiciously increasing the minority classes without touching the majority class. The algorithm is illustrated in Algorithm 6-6 and a graphical example in Figure 6-10. Figure 6-10a illustrates the SMOTE starting process: the green dots represents the positive examples and the blue dots the negative ones. Figure 6-10b shows the *k*-nearest selection process (*k*=3, identified as yellow dots) for a random selected positive example (black dot); and the final synthetic created example (red dot) is displayed in Figure 6-10c.

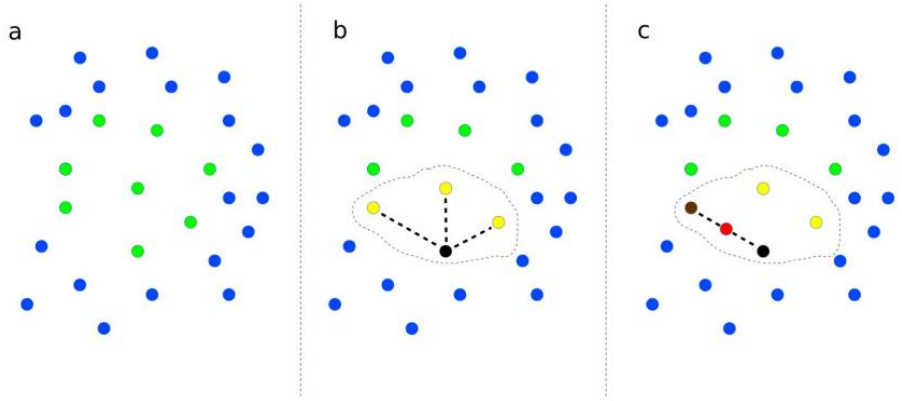


Figure 6-10 SMOTE graphical example (a) start process, (b) *k*-nearest selection process and (c) new synthetic positive sample added in red (from [114])

Algorithm 6-6: SMOTE (adapted from [115])

Input:

Number of minority class samples *T*
Amount of SMOTE *N%*
Number of nearest neighbors *k*

Output: *(N/100) * T* synthetic minority class samples

1: // If *N* is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd //
2: **If** *N < 100* **then**
3: Randomize the *T* minority class samples
4: *T = (N/100) * T*
5: *N = 100*
6: **End If**
7: *N = (int) (N/100)*
8: *k* = Number of nearest neighbors
9: *numattrs* = Number of attributes
10: *Sample[][]:* array for original minority class samples
11: *newindex:* keeps a count of number of synthetic samples generated, initialized to 0
12: *Synthetic[][]:* array of synthetic samples
13: **For** *i=1* to *T* **do**
14: Compute K nearest neighbors for *i*, and save the indices in the *nnarray*
15: *Populate(N, i, nnarray)*
16: **End For**

17: **Function Populate**(*N, i, nnarray*)
18: **While** *N <> 0* **do**
19: Choose a random number between 1 and *k*, call it *nn*. This step chooses one of the *k* nearest neighbors of *i*
20: **For** *attr =1* to *numattrs* **do**
21: *dif = Sample[nnarray[nn]][attr] – Sample[i][attr]*
22: *gap = random number between 0 and 1*
23: *Synthetic[newindex][attr] = Sample[i][attr] + gap * dif*
24: **End For**
25: *newindex++*
26: *N = N – 1*
27: **End While**
28: return
29: **End Function**

In this thesis, we use the *imbalanced-learn* [116] to implement SMOTE.

6.4. Model building and assessment

The process to test the model quality and validity involves several steps: (1) hyper-parameter tuning with cross-validation, (2) model general evaluation using the best hyper-parameters and cross-validation and (3) model specific evaluation using a split of test/train dataset.

In step 1, for each of the models selected (gradient boosted trees, random forest, Adaboost and multilayer perceptron), a hyper-parameter optimization was experimentally performed using a randomized search approach and cross-validation using the *scikit-learn* function *RandomizedSearchCV*. The folds are created by preserving the percentage of samples for each class, using the *scikit-learn* function *StratifiedKFold*. With this approach, a fixed number of parameter combinations are sampled from the specified distribution and the performance is evaluated using cross-validation, using as objective function the area under the receiver operating characteristic curve (AUC), to identify the best set of parameters. In the context of this work, to find the best set of parameters, we try 10 random combinations and 5 folds, and thus a total of 50 models are created. The selection of this random search approach instead of a typical grid search is mainly due to the existing computing resources limitations and the requirement to train the model every 24 hours, as described in sections 5.1.2 and 4.2.

In step 2, for a general classification performance evaluation, the best set of parameters are used to train a model using the complete dataset and 5 folds (with the *scikit-learn* function *cross_validate* [100]). Again, the folds are created by preserving the percentage of samples for each class and AUC is used as objective function.

In step 3, similarly to the process used for the proposed framework for accident prediction, the complete dataset is split into training (70%) and test (30%), and the final model is created using the training data and the set of best parameters obtained through hyper-parameter optimization in step 1. The prediction is

calculated using the unseen dataset (test set) and the created model. Instead of simply generating a final prediction class for each sample, a series of probabilities are also extracted (using the *scikit-learn* function *predict_proba*). As a post-processing step, a *multiplicative correction factor* is applied to each probability using the already identified statistics factors described in section 5.1.4.4 according to the Algorithm 6-7.

Algorithm 6-7: Application of multiplicative correction factor to prediction probabilities (General)

```
1: Calculate multiplicative correction factor for hour of the day (cf_hour_day)
2: Calculate multiplicative correction factor for day of the week (cf_day_week)
3: Calculate multiplicative correction factor for week of the year (cf_week_year)
4: For each collision sample (cs) in test set do
5:      If (cs.HOUR_DAY is in {8, 9, 3, 4, 5, 6} and (cs.PROBABILITY < cf_hour_day) then
6:          cs.PROBABILITY = cf_hour_day
7:      If (cs.DAY_WEEK is in {"THURSDAY", "FRIDAY"} and (cs.PROBABILITY < cf_day_week) then
8:          cs.PROBABILITY = cf_day_week
7:      If (cs.WEEK_YEAR is in {7, 8, 48, 51, 52} and (cs.PROBABILITY < cf_week_year) then
8:          cs.PROBABILITY = cf_week_year
7: End For
```

After this correction, a series of classification performance curves are generated: ROC curve, Precision/Recall (PR) curve; as well as plots of different classification performance scores to identify the best discriminant threshold candidates. These are described in section 6.2 and include: the maximum F1-score, maximum accuracy, maximum precision/recall, the maximum Matthews correlation coefficient and Youden J Index using the original and the corrected probabilities.

Using the five best discriminant threshold candidates identified, the models are rebuilt and the same series of classification performance metrics is used to compare the different models in sections 7.1.5 and 7.2.3.

7. Prototype Implementation: Experimental Results

The experimental results for each of the models created for accident frequency (collision/no collision) and accident severity (fatal, injury and property damage) are detailed in this chapter.

7.1. Prediction of Accident Frequency (collision/no-collision)

The following section details the different hyper-parameters ranges used, the best hyper-parameters obtained, the performance curves, the selected thresholds, and the final classification performance results for each of the models created for the classification of accident frequency (collision/no-collision).

7.1.1. Gradient Boosted Trees (*XGBClassifier*)

For the gradient boosted trees, the information about the different hyper-parameters tuned, including their description [111] and the ranges used, and the best value obtained as result of the optimization process are described in the Table 7-1.

Table 7-1 Gradient Boosting Trees hyper-parameters tuned, description, ranges of values and best value

Hyper-parameter Name	Description	Range of Values	Best Value
n_estimators	Number of trees to fit	[50,100,150]	150
max_depth	Maximum depth for base learners	[6,9,12]	12
objective	Specify the learning task and the corresponding learning objective	'binary:logistic'	'binary:logistic'
booster	Booster to be used	'gbtree'	'gbtree'
learning_rate	Boosting learning rate	[0.1, 0.2, 0.3]	0.1
min_child_weight	Minimum sum of instance weight(hessian) needed in a child.	[1.0, 5.0, 10.0]	5.0
subsample	Subsample ratio of the training instance.	[0.5, 0.7, 1.0]	1.0
colsample_bytree	Subsample ratio of columns when constructing each tree	[0.5, 0.7, 1.0]	1.0
reg_lambda	L2 regularization term on weights	[0.1, 1.0, 10.0]	10.0
reg_alpha	L1 regularization term on weights	0.0	0.0
early_stopping_ rounds	Activates early stopping. Validation error needs to decrease at least every <early_stopping_ rounds> round(s) to continue training.	10	10

Figure 7-1 displays the model features' importance. The plot shows, for each variable on the *y*-axis how important that variable is in classifying the data. The

importance of each variable is presented along the *x*-axis. The variables are presented in order from top to bottom, from the most to the least important. We can notice that the top-10 features are related with solar (SOLAR_ELEVATION, SOLAR_AZIMUTH), location (LOCATION_B, COLLISION_RATE_location_a, LOCATION, YCOORD, XCOORD), roads (ROAD_NAME, ROAD_SEGMENT) and date (COLLISION_RATE_ACCIDENT_ DATE_dayy) variables. These appear at the top of Figure 7-1. The bottom-10, appearing towards the bottom of the same figure, are related with the events generated features (COLLISION_RATE_ottawa_statutory_holidays, COLLISION_RATE_carleton_calendar_ fall, COLLISION_RATE_ottawa_champions_bbc, OTTAWA_RAVENS_FOOTBALL), location (COLLISION_RATE_street1, STREET2, STREET1) and roads (COLLISION_RATE_road_direction, ROAD_LEN_LOG) and date (ACCIDENT_DATE_ weekm).

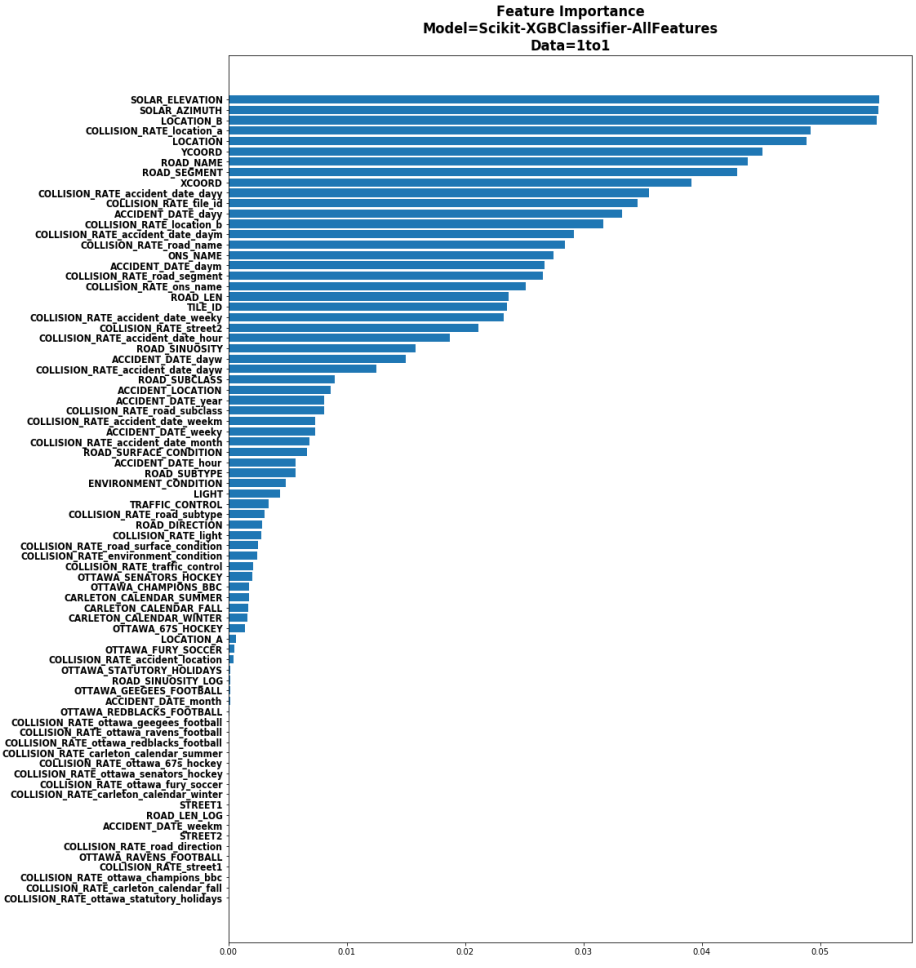


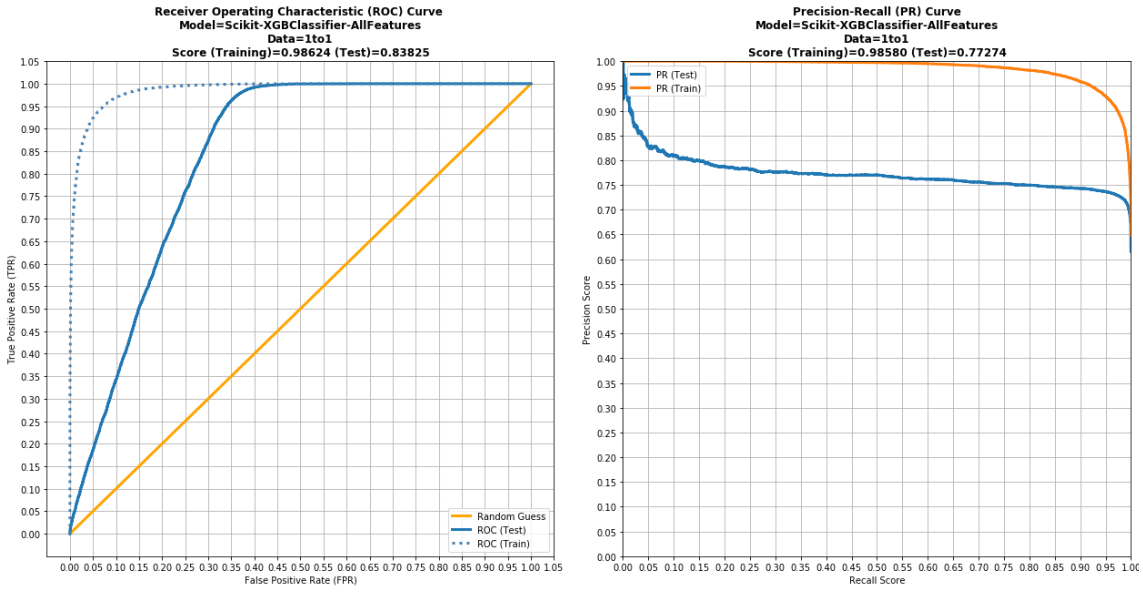
Figure 7-1 Gradient Boosted Trees feature importance

The Table 7-2 contains the general performance evaluation over the test set using cross-validation.

Table 7-2 Gradient Boosting Trees general model performance over the test set using best hyper-parameters and cross-validation (5 folds)

AUC (mean)	Accuracy (mean)	F1 Score (mean)	Precision (mean)	Recall (mean)
83.91%	80.85%	83.43%	73.53%	96.41%

The ROC curve is illustrated in Figure 7-2a, with the solid blue line representing the score over the test set and the dotted blue line over the training set.



(a) (b)

Figure 7-2 Gradient Boosted Trees: (a) ROC curve and (b) precision-recall curve

We can notice that the performance (AUC) is better for the training set (as expected), while the AUC for test set is 0.83825. Figure 7-2b shows the precision-recall curve where the blue line represents the score for the test set and the orange for the training. Similarly, we can notice that the performance for the training set is better than over the test set, achieving a performance (AUC, area under the PR curve) of 0.77274 for the test set.

The plot of the different classification performance curves (i.e. recall, precision, accuracy, F1, Youden J index and Matthews correlation coefficient) for the test set is shown in Figure 7-3. The *x*-axis represents the discriminant threshold (cut-

off value) and the *y*-axis the score. A series of five thresholds was chosen that maximize the corresponding score. These are, in the order from highest to lowest: F1 (in green at [threshold = 0.4500, score = 0.83385]), accuracy (in brown at [threshold = 0.4850, score = 0.80630]), precision and recall (in black at [threshold = 0.65, score = 0.75291]), Matthews (in magenta at [threshold = 0.4350, score = 0.65058]) and Youden (in red at [threshold =0.4850, score = 0.61258]). We can notice also that the Youden J index and the accuracy-based thresholds match.

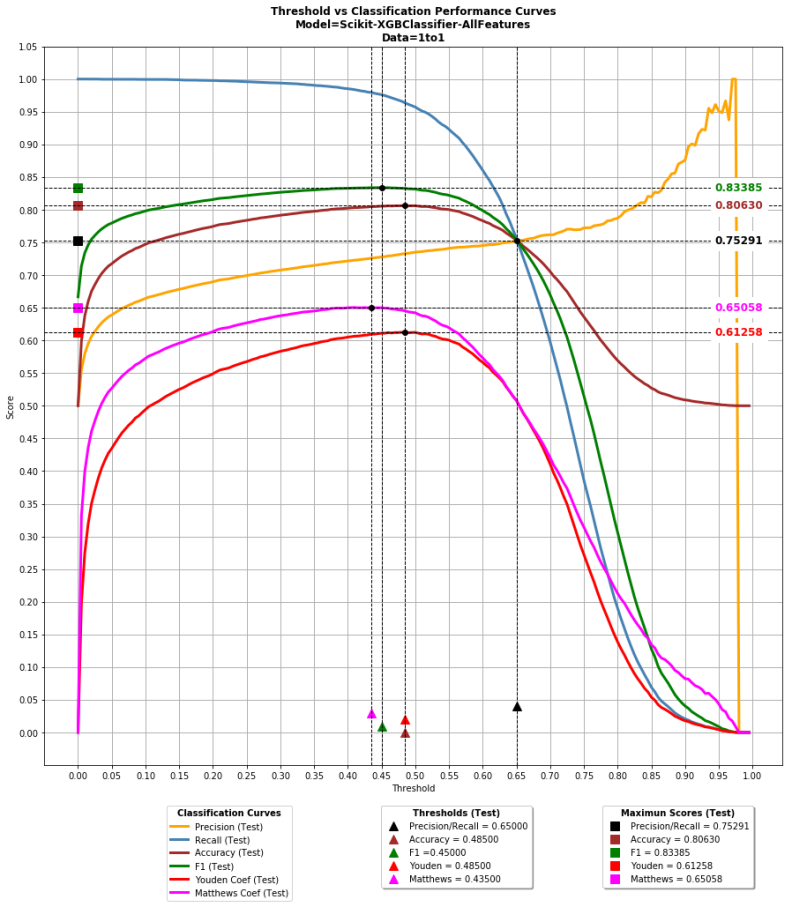
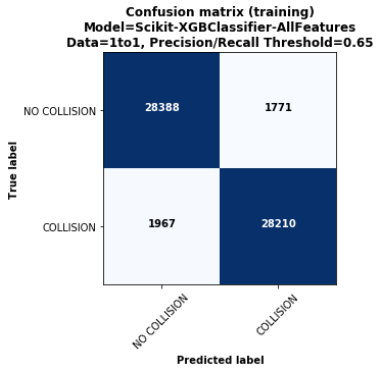


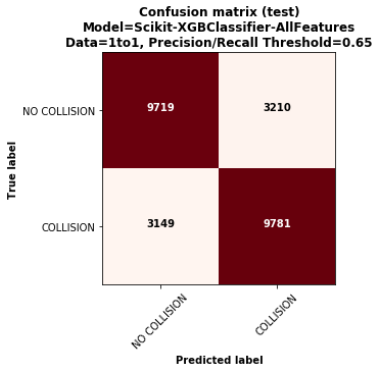
Figure 7-3 Gradient Boosted Trees: classification performance curves vs discriminant thresholds over the test set

The confusion matrices using the four different thresholds: 0.4350 (Matthews), 0.4500 (F1), 0.48500 (Youden and accuracy, only Youden is shown) and 0.6500 (precision-recall) over the test and training set are showed in the Figure 7-4a to Figure 7-4h. We can notice that, as it was discussed in section 6.1.10, using as discriminant threshold the value related with the intersection between

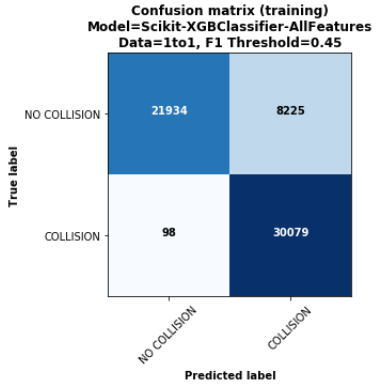
precision/recall produces a more balanced relationship between TP and TN, as well as between FP and FN (Figure 7-4a and Figure 7-4b). Because all the other thresholds considered are located to the left side of the precision/recall threshold (where the number of TP is larger than that of TN), the resulting confusion matrix shows a more imbalanced relationship giving more importance to the positive class (collision). As such, the number of TP is increased and the number of FN is reduced. This affects the distribution of the negative class (no collision) reducing the number of TN with an expected increase of FP (a negative correlation relationship exists between TP and FP), with Matthews coefficient (Figure 7-4g, h) providing the biggest increase in the TN (12665).



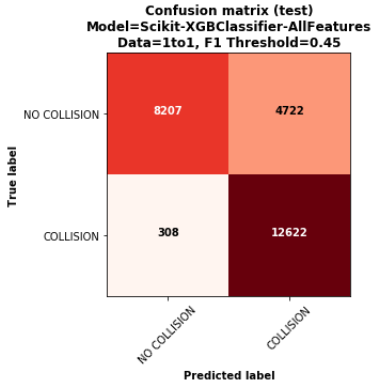
(a)



(b)



(c)



(d)

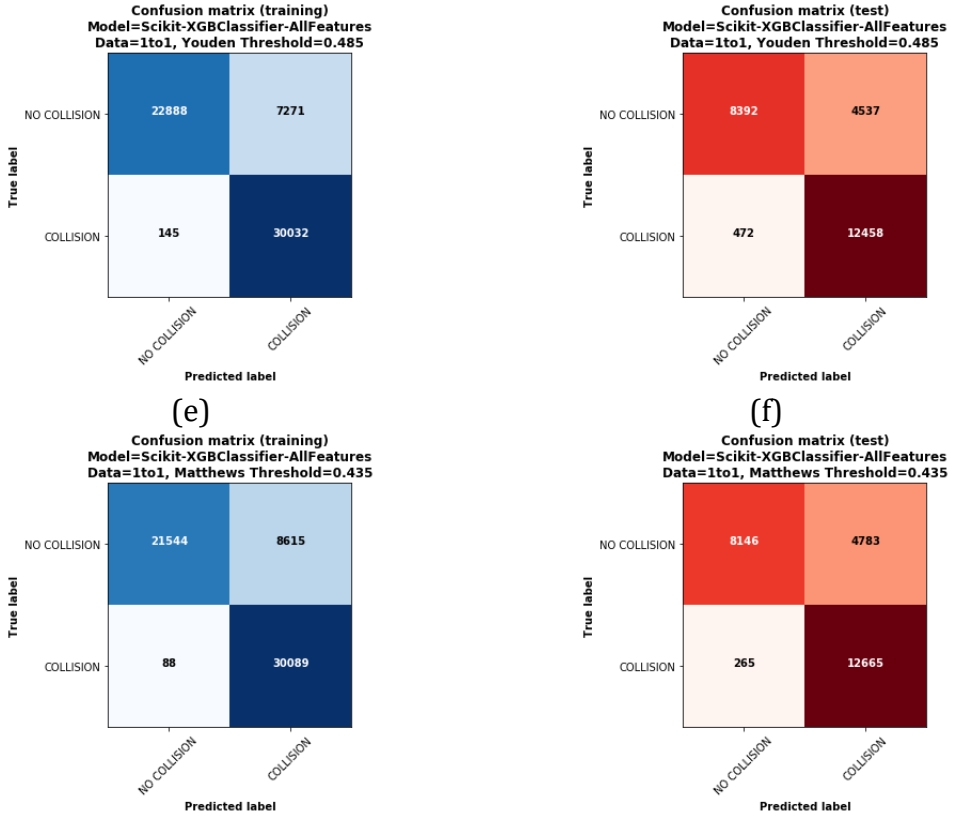
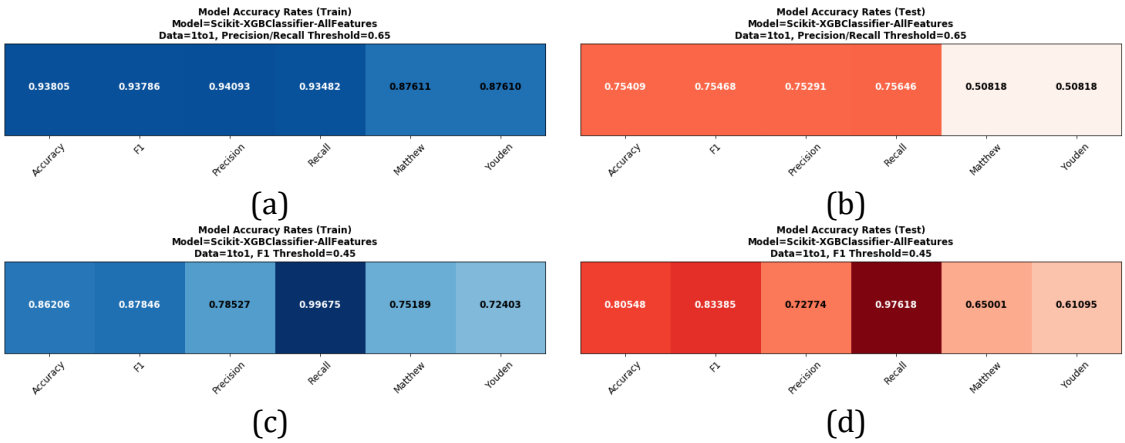


Figure 7-4 Gradient Boosted Trees: confusion matrix using different discriminant thresholds over training/test set: precision-recall (a/b), F1 (c/ d), Youden (e/f) and Matthews (g/h)

The model accuracy using each of the four thresholds for the test and training set is shown in Figure 7-5a to Figure 7-5h.



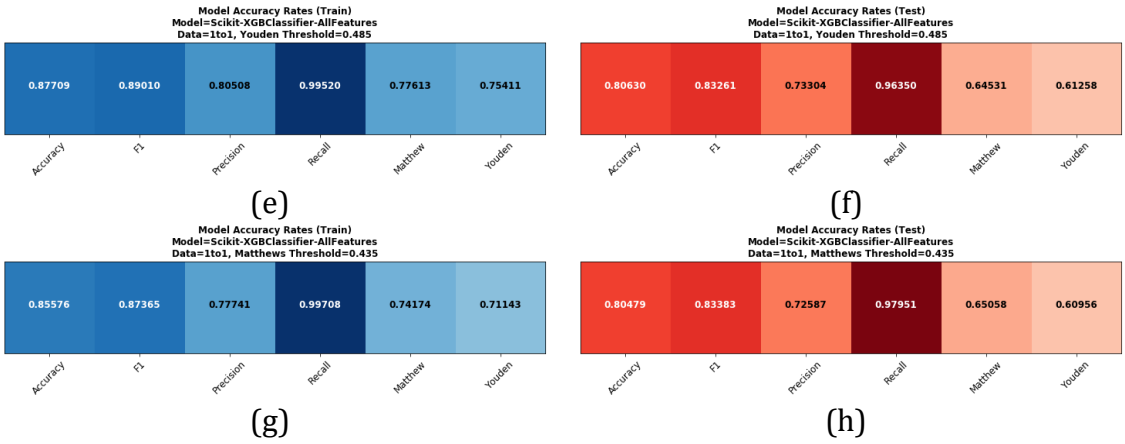


Figure 7-5 Gradient Boosted Trees: metrics over training/test set using different discriminant thresholds precision-recall (a/b), f1 (c/d), Youden (e/f) and Matthews (g/h)

As noticed before, the threshold related to the intersection between the precision and recall curves produces more balanced values between accuracy, F1, precision and recall (over 0.75409 over the test set, in Figure 7-5b), while using the other thresholds produces more heterogeneous results, in particular with a higher accuracy and recall scores (over 0.80479 and 0.96350 over the test set respectively, Figure 7-5d, Figure 7-5f and Figure 7-5h) as expected (as accuracy and recall are metrics defined around the TP). As identified previously, the threshold related to Matthews coefficient produces the best recall score over the test set (0.97951, Figure 7-5h).

As it can be noticed in Table 7-3, if the classification objective is creating a model that considers both classes with equal importance, the best results (a more balanced ratio between TP and TN; and between precision, recall and accuracy scores) are obtained by using the threshold related with intersection between precision and recall. If the objective is creating a model that provides the maximum accuracy over both classes, as expected, the threshold related with the maximum accuracy score produces the best results (the maximum number of TP + TN). If the objective is maximizing the accuracy over the positive class (collision), the maximum Matthews related threshold produces the best results (a higher recall score sacrificing accuracy and precision). Because predicting a collision is more important is this work, we will choose as the best threshold the one related with the maximum

Matthews score (0.435 in this case). In the following tables the accuracy, precision, recall and F1 scores are reported as percentage.

Table 7-3 Gradient Boosted Trees classification scores over the test set

Threshold	TP	TN	TP + TN	Accuracy Score (%)	Precision Score (%)	Recall Score (%)	F1 Score (%)
Matthews (0.435)	**12665**	8146	20811	80.479	72.587	**97.951**	83.383
F1 (0.450)	12622	8207	20829	80.548	72.774	97.618	**83.385**
Accuracy, Youden (0.485)	12458	8392	**20850**	**80.630**	73.304	96.350	83.261
Precision/ Recall (0.650)	9781	**9719**	19500	75.409	**75.291**	75.646	75.409

7.1.2. Adaboost (AdaBoostClassifier)

For the AdaBoost classifier, the identified hyper-parameters are described in Table 7-4. Because we use as a base estimator a *DecisionTreeClassifier*, some of the parameters in the table are related with it.

Table 7-4 AdaBoost hyper-parameters tuned, description, ranges of values and best value

Hyper-parameter Name	Description	Range of Values	Best Value
n_estimators	The maximum number of estimators to be used (*DecisionTrees*)	[50,100,150]	100
base_estimator_ max_depth	The maximum depth of the tree	[3,6,9]	6
base_estimator_learning_rate	The learning rate serves as a	[0.1, 0.5, 1.0]	0.1
base_estimator_criterion	The function used to measure the quality of a split	[‘gini’, ‘entropy’],	‘gini’
Base_estimator_splitter	The strategy used for the splitting	[‘best’, ‘random’]	‘best’

Figure 7-6 shows the model feature importance obtained by this model. We can observe that the top-10 features are related with roads (ROAD_NAME, COLLISION_RATE_road_segment), solar (SOLAR_AZIMUTH, SOLAR_ELEVATION), location (LOCATION_B, COLLISION_RATE_street1, XCOORD, COLLISION_RATE_location_a, YCOORD, COLLISION_RATE_location_b); and the

bottom-10 are related with the events generated features (COLLISION_RATE_ottawa_redblacks_football, CARLETON_CALENDAR_WINTER, COLLISION_RATE_ottawa_statutory_holidays, etc). These variables are similar to those identified by the gradient boosted trees (section 7.1.1).

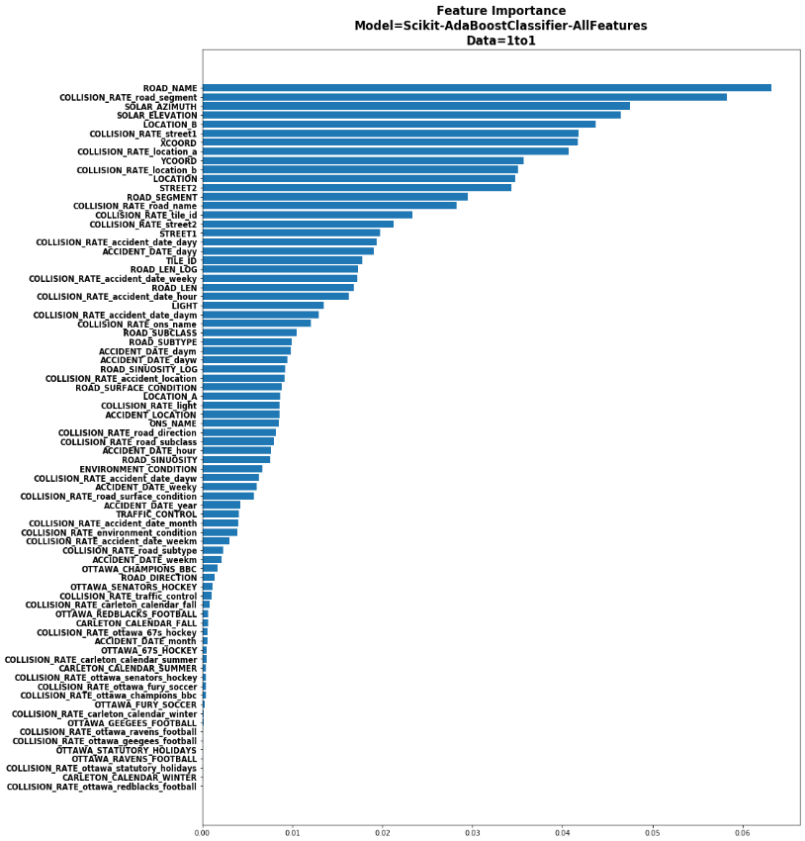
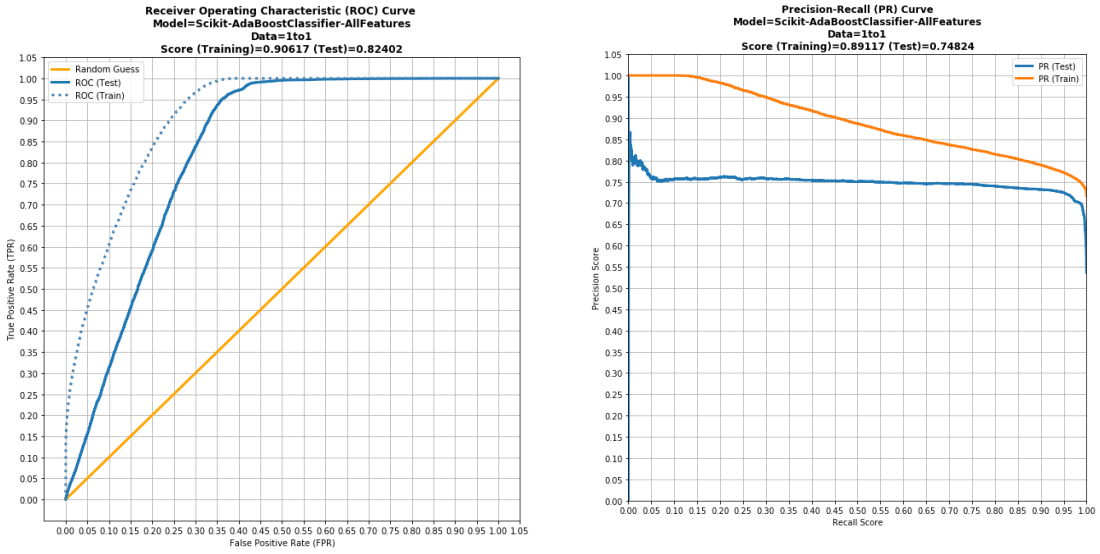


Figure 7-6 AdaBoost feature importance

The general classification performance scores are included in Table 7-5, and we can notice that the results are inferior compared to the ones obtained with gradient boosted trees. The ROC and precision-recall curves are displayed in Figure 7-7a and Figure 7-7b. One can notice that the AUC score over the test set is 0.82402 and the area under precision-recall curve is 0.74824.

Table 7-5 AdaBoost general model performance over the test set using best hyper-parameters and cross-validation (5 folds)

AUC (mean)	Accuracy (mean)	F1 Score (mean)	Precision (mean)	Recall (mean)
83.06%	79.79%	82.67%	72.36%	96.41%



(a) (b)

Figure 7-7 AdaBoost: (a) ROC curve and (b) Precision-Recall (PR) curve

The different classification performance curves over the test set are illustrated in Figure 7-8. We can notice that three thresholds can be considered: 0.5000 (related with maximum Matthews and F1 scores), 0.50500 (related with Youden and accuracy maximum scores) and 0.5200 (precision/recall).

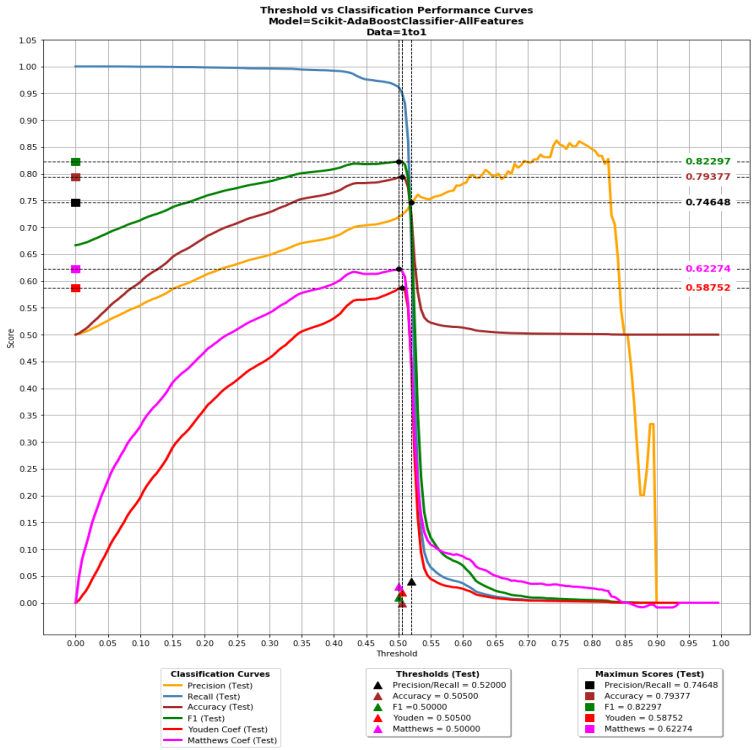


Figure 7-8 AdaBoost: classification performance curves vs discriminant thresholds

The confusion matrices and the classification reports using each of the three thresholds for the test set are displayed in the Figure 7-9. Again, the Matthews threshold provides the best results. However, the results obtained are not as good as those obtained by the gradient boosted trees.

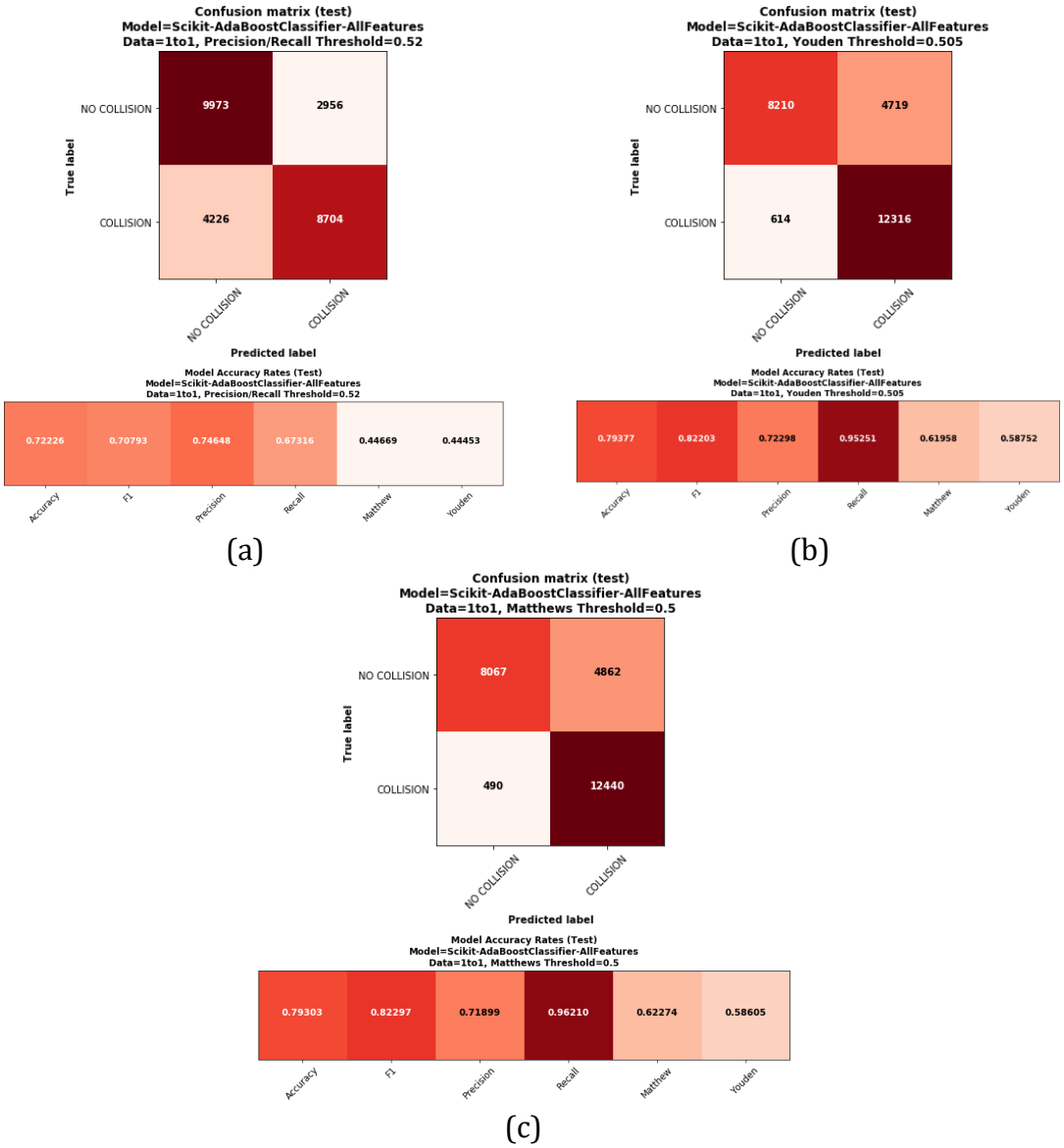


Figure 7-9 AdaBoost: confusion matrices using different discriminant thresholds for the test set: (a) precision/recall (b) Youden/accuracy and (c) Matthews/F1

7.1.3. Random Forest (RandomForestClassifier)

Similar to the previous two classifiers, the hyper-parameters and their best values obtained as result of the optimization process are described in Table 7-6 for the random forest classifier.

Table 7-6 Random Forest hyper-parameters tuned, description, ranges of values and best value

Hyper-parameter Name	Description	Range of Values	Best Value
n_estimators	Number of trees in random forest	[50,100,150]	150
max_depth	Maximum number of levels in tree	[6,9,12]	12
max_features	Number of features to consider at every split	['auto', 'sqrt']	'auto'
min_samples_split	Minimum number of samples required to split a node	[2, 5, 10],	10
min_samples_leaf	Minimum number of samples required at each leaf node	[1, 2, 4]	2
bootstrap	Method of selecting samples for training each tree	[True, False]	True
criterion	The function used to measure the quality of a split	['gini']	'gini'

Figure 7-10 displays the model feature importance. We can notice that the top-10 features are related with location (LOCATION_B, STREET2, COLLISION_RATE_street2, COLLISION_RATE_location_a, COLLISION_RATE_location_b) and roads (ROAD_NAME, COLLISION_RATE_road_name, COLLISION_RATE_road_segment, ROAD_LEN, ROAD_LEN_LOG); and the bottom-10 are related with the events generated features (OTTAWA_RAVENS, OTTAWA_STATUTORY_HOLIDAYS, etc.), sensibly different from those identified by the gradient boosted trees (section 7.1.1) and AdaBoost (section 7.1.2). In particular, the solar features (SOLAR_ELEVATION and SOLAR_AZIMUTH) are identified as less important. As in the previous cases, the social events generated features are situated towards the bottom of the feature importance graph, thus demonstrating that they bring only a very small contribution.

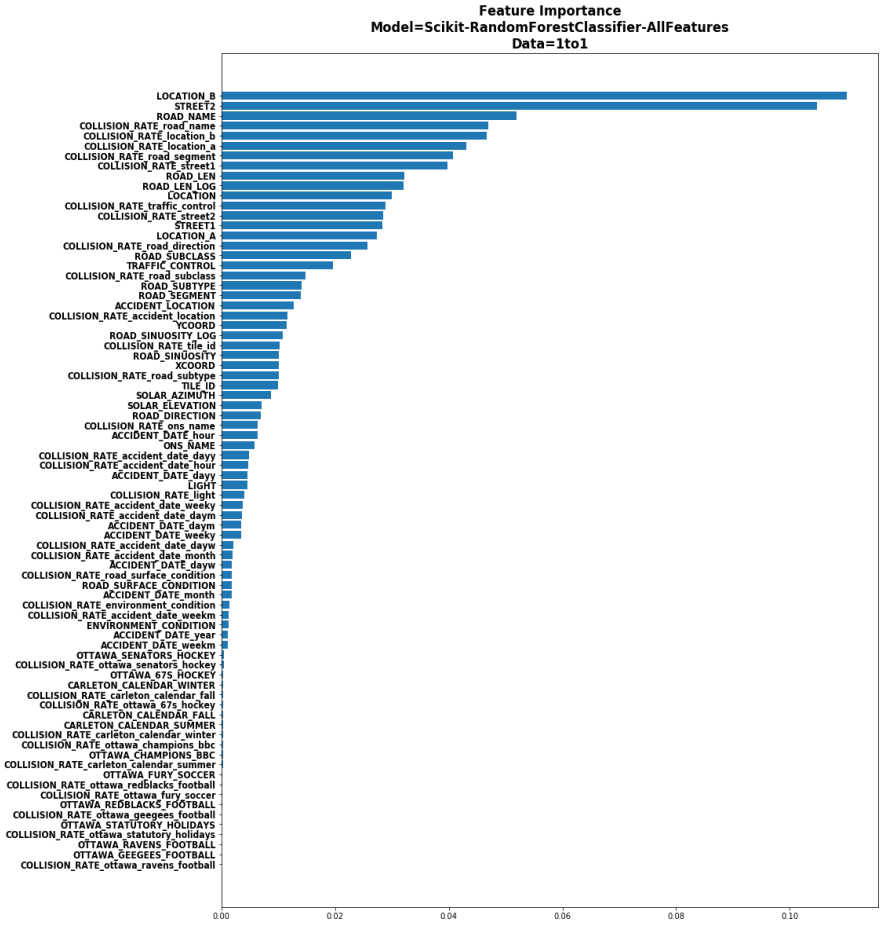


Figure 7-10 Random Forest feature importance

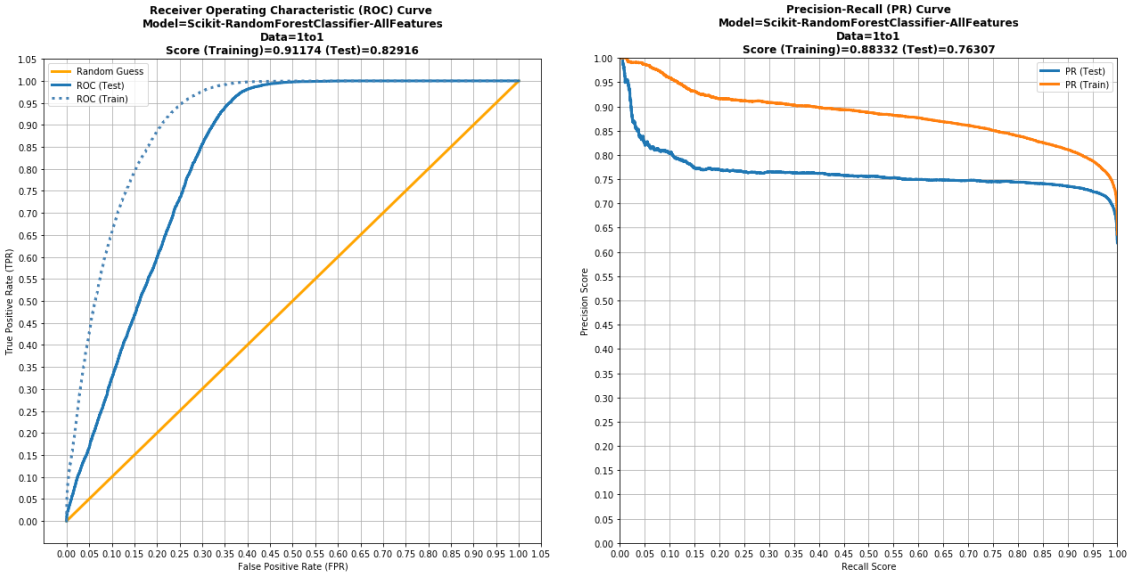
Table 7-7 Random Forest: general model performance over the test set using best hyper-parameters and cross-validation (5 folds)

AUC (mean)	Accuracy (mean)	F1 Score (mean)	Precision (mean)	Recall (mean)
82.87%	77.54%	81.55%	69.22%	99.22%

Again, the general performance evaluation results are lower than those of the previous models (Table 7-7).

The ROC and precision-recall curves are illustrated in Figure 7-11a and Figure 7-11b; we can notice that the AUC score over the test set is 0.82916 and the area under precision-recall curve is 0.76307.

The plot of the different classification performance curves over the test set is shown in Figure 7-12.



(a) (b)

Figure 7-11 Random Forest: (a) ROC curve and (b) precision-recall curve

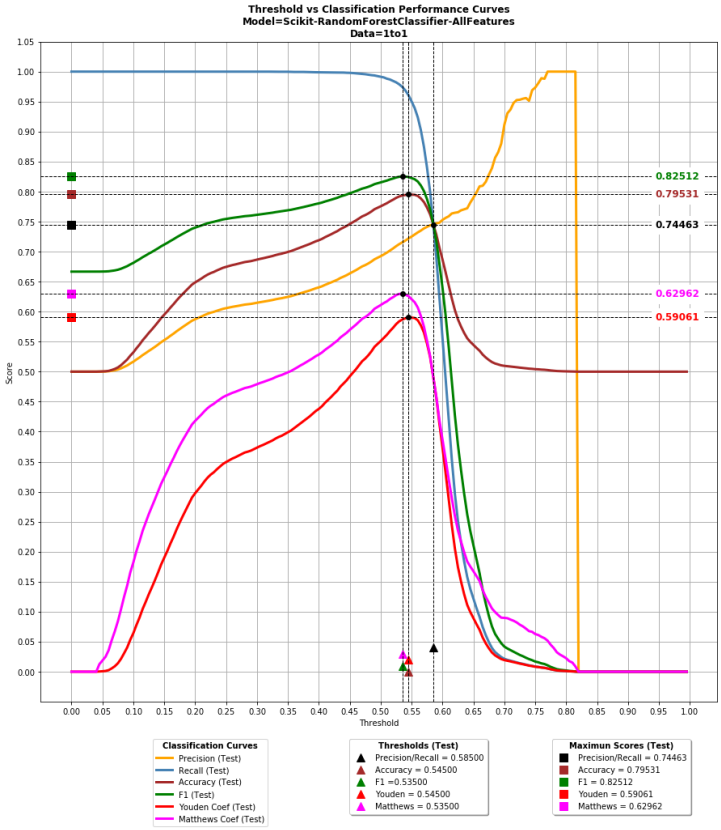


Figure 7-12 Random Forest: classification performance curves vs discriminant thresholds

We can notice that three thresholds identified are: 0.53500 (related with maximum Matthews and F1 scores), 0.54500 (related with Youden and accuracy maximum scores) and 0.58500 (precision/recall).

The confusion matrices and the classification performance metrics using each of the three different thresholds over the test set are illustrated in the Figure 7-13a and Figure 7-13b. Again, the Matthews threshold provides the best results. The results are a little better than the ones obtained with AdaBoost (section 7.1.2), but not as good as those achieved by the gradient boosted trees (section 7.1.1).

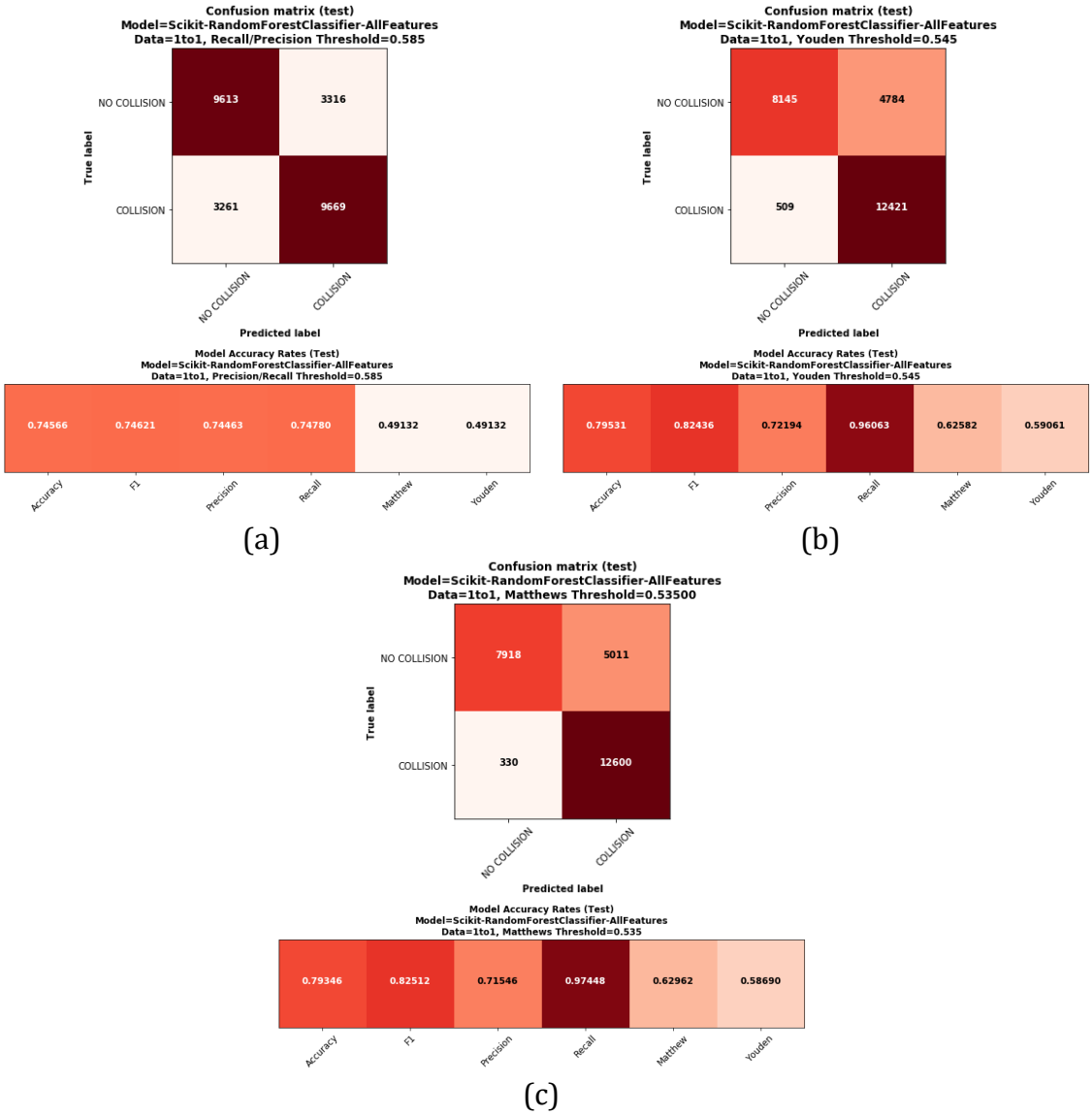


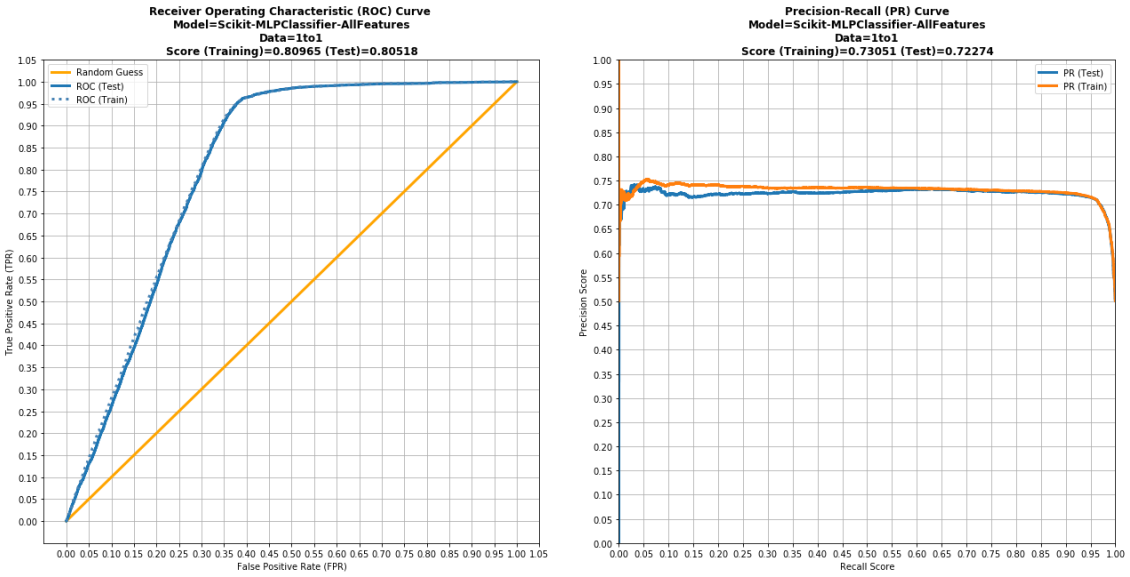
Figure 7-13 Random Forest: confusion matrices using different discriminant thresholds: (a) precision/recall (b) Youden and (c) Matthews

7.1.4. Neural Networks (*MLPClassifier*)

Finally, for the multi-layer perceptron classifier, the hyper-parameters along with their best values are listed in the Table 7-8.

Table 7-8 Multi-layer perceptron hyper-parameters tuned, description, ranges of values and best value

Hyper-parameter Name	Description	Range of Values	Best Value
hidden_layer_sizes	Number of neurons in each hidden layer (layer 1, layer2, layer3) or (layer1,)	[(50,50,50), (50,100,50), (100,)]	(50,100,50)
activation	Activation function used for the hidden layer	['relu','logistic']	'relu'
solver	The solver used for weight optimization	['sgd', 'adam']	'adam'
early_stopping	Used to terminate the training process when the validation score is not improving	[True]	True
learning_rate	The type of learning rate to be used	['constant','adaptive']	'constant'
learning_rate_init	The initial learning rate used	[0.001, 0.01, 0.03]	0.001
alpha	Regularization term (L2)	[0.0001, 0.01, 0.05]	0.05
tol	Tolerance of the optimization	[1e-4]	[1e-4]
n_iter_no_change	Maximum number of epochs allowed that do not meet a tol improvement	[10]	10



(a) (b)

Figure 7-14 Multi-layer perceptron: (a) ROC curve and (b) precision-recall curve

One can notice that the best neural network architecture consists of three hidden layers with 50, 100 and 50 neurons in each, respectively.

Table 7-9 Multi-layer perceptron: general model performance over the test set using best hyper-parameters and cross-validation (5 folds)

AUC (mean)	Accuracy (mean)	F1 Score (mean)	Precision (mean)	Recall (mean)
80.55%	78.14%	81.14%	71.36%	94.05%

Table 7-9 contains the general model performance scores. Again, the results are worse than those obtained by the other algorithms.

The ROC and precision-recall curves, illustrated in Figure 7-14a and Figure 7-14b, show that the AUC score over the test set is 0.80518 and the area under precision-recall curve is 0.72274, again, not as good as the previous three models.

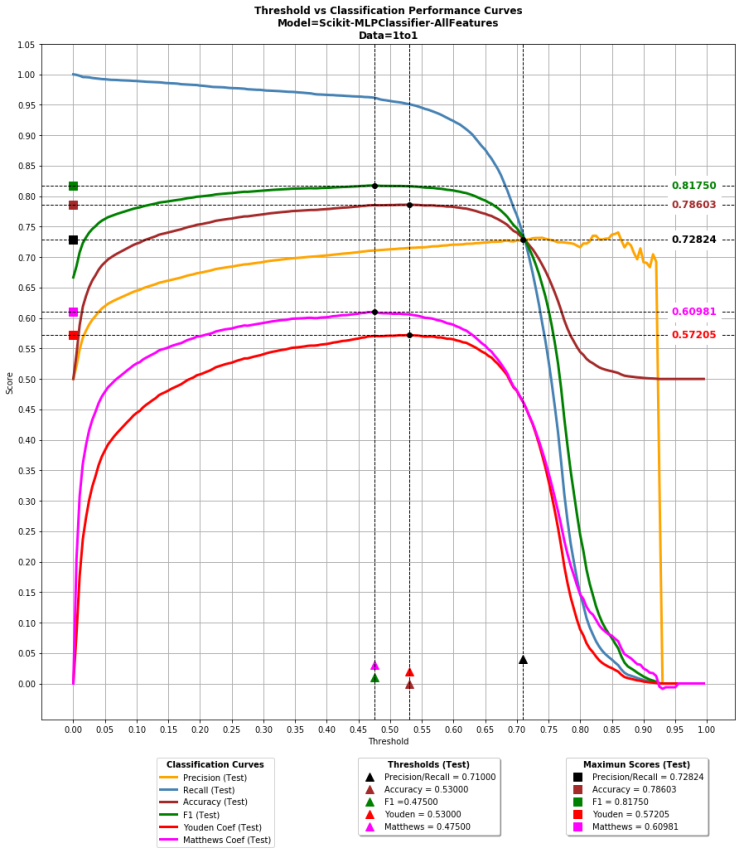


Figure 7-15 Multi-layer perceptron: classification performance curves vs discriminant thresholds

The plot of the different classification performance curves (i.e. recall, precision, accuracy, F1, Youden J Index and Matthews Correlation Coefficient) over the test set is shown in Figure 7-15. We can notice that three thresholds to be considered are: 0.4750 (related with maximum Matthews and f1 scores), 0.53000 (related with Youden and accuracy maximum scores) and 0.7100 (precision/recall).

The classification scores and the confusion matrices displayed in Figure 7-16a to Figure 7-16c demonstrate as well that the results are not as good as those achieved by the other models.

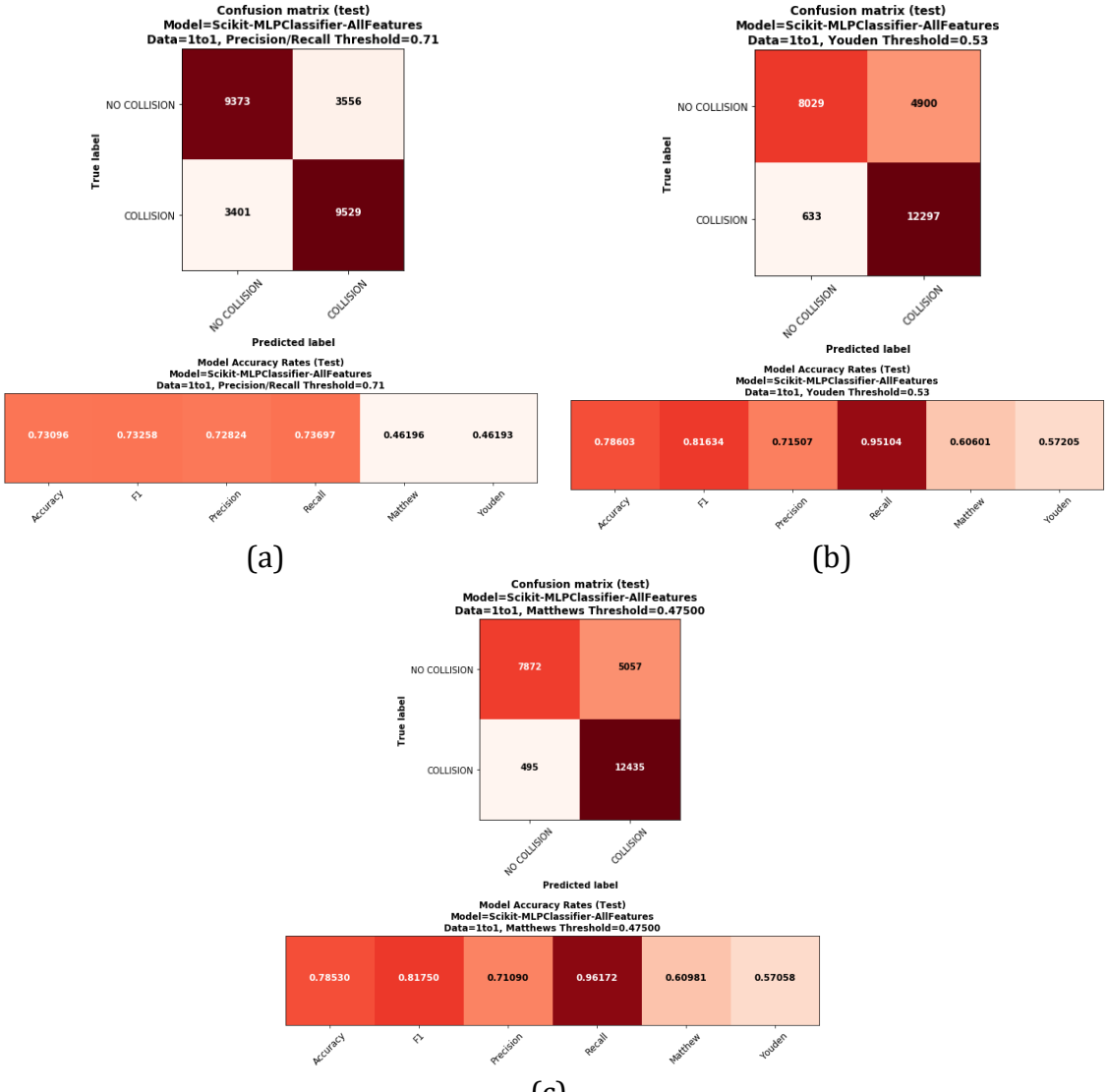


Figure 7-16 Multi-layer perceptron: confusion matrices using different discriminant thresholds over the test set (a) precision/recall (b) Youden and (c) Matthews

7.1.5. Model Comparison and Evaluation

In this section, we compare the various classifiers described in sections 7.1.1 to 7.1.4. Using the Matthews related threshold as the best discrimination threshold, Table 7-10 displays the classification performance scores and Table 7-11 the total execution time (including the time required for hyper-parameter optimization).

Table 7-10 Model comparison in terms of performance metrics over the test set using the best threshold (Matthews)

Model	Number of TP + TN	Accuracy Score (%)	Precision Score (%)	Recall Score (%)	F1 Score (%)	Matthews Score
GBT	**20811**	**80.479**	**72.587**	**97.951**	**83.383**	**0.65058**
AdaBoost	20507	79.303	71.899	96.210	82.297	0.62274
Random Forest	20518	79.346	71.546	97.448	82.512	0.62962
Multi-layer Perceptron	20307	78.530	71.090	96.172	81.750	0.60981

Table 7-11 Model comparison in terms of execution time over test set using the best threshold (Matthews)

Model	Total Execution Time
GBT	63 minutes
AdaBoost	150 minutes
Random Forest	21 minutes
Multi-layer Perceptron	40 minutes

In terms of classification performance scores, it is clear that the best model is gradient boosted trees, followed by the random forest classifier and AdaBoost. Considering the execution time, random forest is the best one (21 minutes), followed by multi-layer perceptron (40 minutes) and gradient boosted trees (63 minutes). Because we give more importance to the classification scores, we chose the gradient boosted trees as the final model.

Once the best model is selected, in order to simplify it and reduce the total execution time, a feature selection process was implemented. To do this, we trained a new model using the best set of hyper-parameters obtained for the complete model (in section 7.1, to save time) and using as features the ones with an importance greater than the median in the original model (a total of 26 features in our work) and compared the performance with the original model. If the performance did not

change significantly, we removed more features and reapplied the process, using the top-n important features. The models tested were: top-over the median (top-26), top-20, and top-10 to top-5, where a significant performance decline was encountered. The best model identified was the one using the top-7 features and that uses as variables, in order, YCOORD, ROAD_NAME, SOLAR_AZIMUTH, LOCATION, LOCATION_B, SOLAR_ELEVATION and COLLISION_RATE_LOCATION_A.

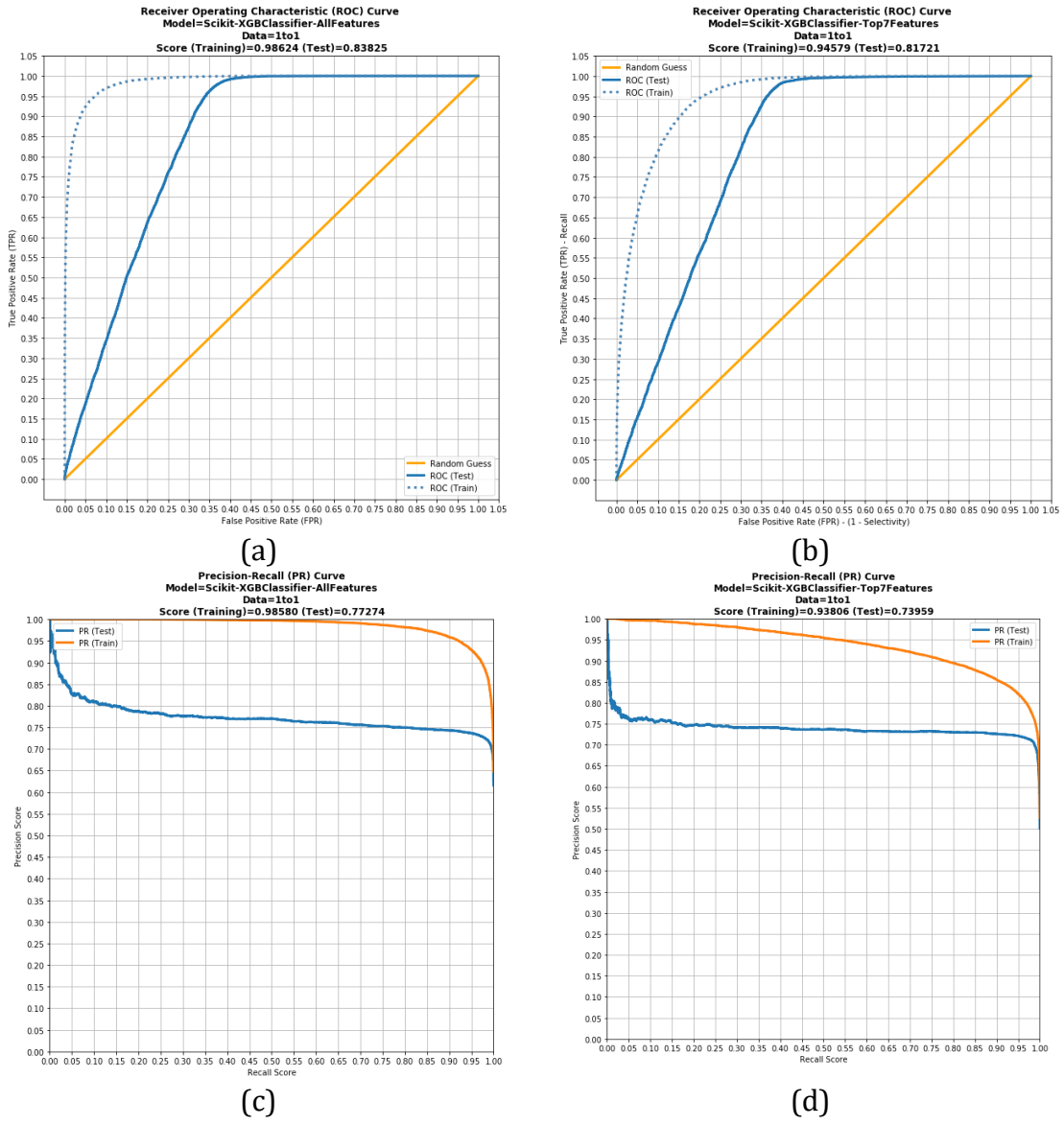


Figure 7-17 Gradient Boosted Trees: ROC curves for (a) all features and (b) the top-7 most important features, and precision-recall curves for (c) all features and (d) top-7 most important features

To compare it with the original model created based on all features, a new complete run was executed including hyper-parameter optimization with only the top-7 features. Figure 7-17 displays the ROC and PR curves (in Figure 7-17a and c for the model with all features and in Figure 7-17b and d the ones related with the top-7 features). We can notice that the curves look very similar, but with a slightly lower score for the model based on the most important top-7 features only (i.e. the AUC score over the test set is 0.817 vs. 0.838 and the area under precision-recall curve is 0.740 vs. 0.772).

The confusion matrices and the classification results over the test set using the Matthews related threshold are illustrated in Figure 7-18. As it can be noticed, because the simplified model (with top-7 most important features) achieves similar classification results to the original, with a slightly better recall score, the simplified model was selected as the final model.

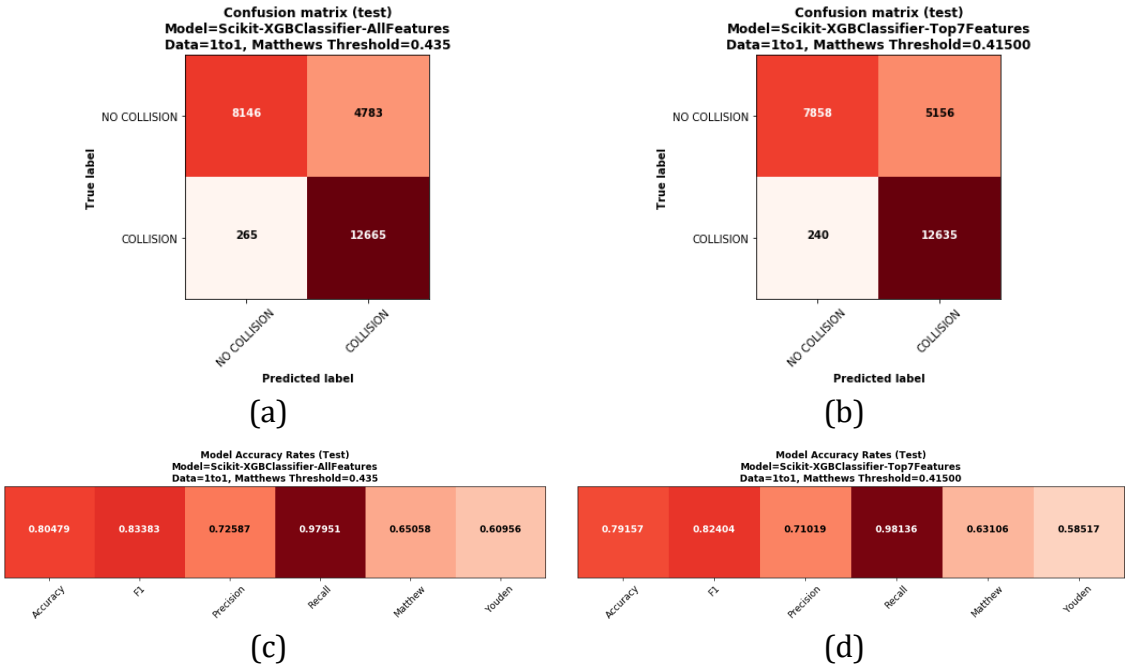


Figure 7-18 Gradient Boosted Trees: confusion matrices and performance metrics using Matthews discriminant threshold over the test set with all features (a and c) and with top-7 important features only (b and d);

The total execution time for the model with the top-7 most important features takes only 12 minutes, a big improvement compared with the model using the

complete set of features (63 minutes). The final model comparison tables are Table 7-12 and Table 7-13. It is important to mention that for all the models used, the results after applying the multiplicative correction factor were not improved with respect to the original probabilities. This is due in this thesis to the manner in which the non-collision dataset was created (i.e. close variations of accident samples). Because of this, the following results presented are based only on the original probabilities.

Table 7-12 Final model comparison in terms of performance metrics over the test set using the best threshold (Matthews)

Model	Number of TP + TN	Accuracy Score (%)	Precision Score (%)	Recall Score (%)	F1 Score (%)	Matthews Score
Gradient Boosted Trees	20811	80.479	72.587	97.951	83.383	0.65058
AdaBoost	20507	79.303	71.899	96.210	82.297	0.62274
Random Forest	20518	79.346	71.546	97.448	82.512	0.62962
Multi-layer Perceptron	20307	78.530	71.090	96.172	81.750	0.60981
Gradient Boosted Trees (top-7)	**20493**	**79.157**	**71.019**	**98.136**	**82.404**	**0.63106**

Table 7-13 Final model comparison in terms of execution time over test set using the best threshold (Matthews)

Model	Total Execution Time
Gradient Boosted Trees	63 minutes
AdaBoost	150 minutes
Random Forest	21 minutes
Multi-layer Perceptron	40 minutes
Gradient Boosted Trees (top-7)	**12 minutes**

7.1.6. GIS Map Visualizations of Model Predictions

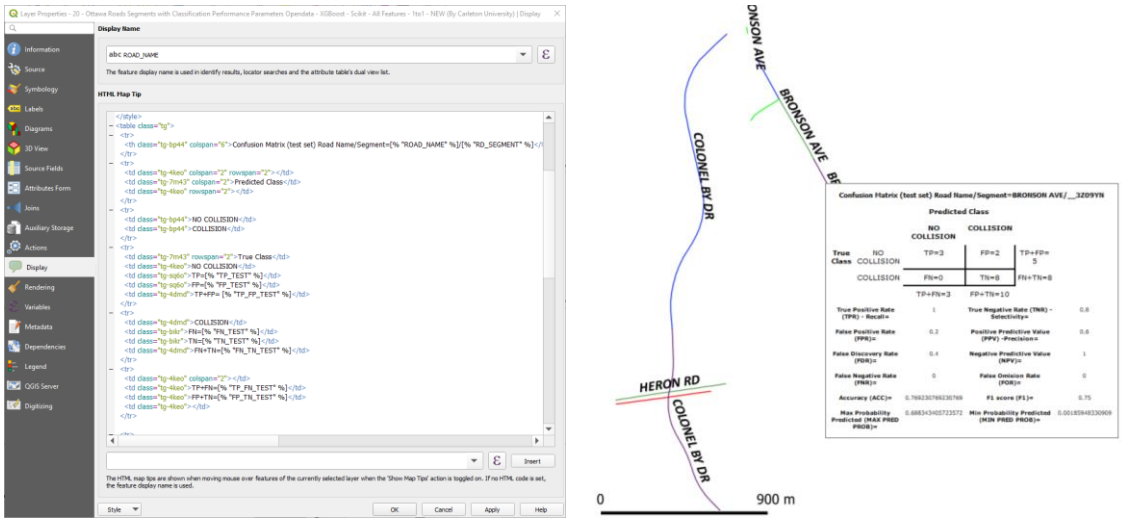
Using the best model (Gradient Boosted Trees with top-7 features) and the selected discrimination threshold (0.4150), the series of classification performance scores (i.e. accuracy, recall, precision, F1, Matthews MCC, Youden J Index) were calculated over the test set using different types of geospatial layers. To enhance visualization capabilities, two new geospatial layers were created from the original road segments: roads merged by name and roads merged by subclass. To create the geospatial layer

roads segment merged by name, all the road segments that belong to the same road name were merged into one geospatial feature. Using this, we calculate and show the classification scores per road name (i.e. considering all the *Bronson Avenue* segments together). Similarly, the roads segments merged by subclass were also created. The final list of geospatial layers with classification performance scores are: road segments, road intersections, neighborhoods, tiles, roads merged by name; and roads merged by subclass.

As part of the objectives, the final spatial layers along with the classification performance metrics are loaded into QGIS to perform several visualizations. Using the QGIS HTML Map Tip layer property, it is possible to display in real-time information when the mouse is placed over the element (HTML Tooltip). Using this property, an HTML code was developed to show the confusion matrix and classification scores. The QGIS HTML Map Tip layer windows is illustrated in Figure 7-19a. These visualizations can be used for proactive traffic intervention schemes.

The following figures show examples of classification performance metrics over the test set using different types of geospatial layers: by road segment (*road segment id =__3Z09YN* that belongs to *Bronson Ave*, in Figure 7-19b), by road name (all the road segments that belongs to *Bronson Ave*, in Figure 7-20), by neighborhood (*Carleton University*, in Figure 7-21) and by tile (*tile id=1800* that belongs to *Carleton University* neighborhood, in Figure 7-22). These advanced visualization capabilities go beyond the currently available evaluation tools, offering specific, localized details on the performance of the classifiers and thus representing one of the important contributions of this work.

The next figures show examples of heat maps over the test set using different classification metrics and types of geospatial layers. Figure 7-23 shows the maximum predicted collision probability by tile, where the high probability is shown in yellow color (related with a collision) and the minimum in magenta (related with non-collisions) and Figure 7-24, the total number of predicted collisions over the test set by neighborhood, where the areas in blue denote high risk area for accident occurrence. Finally, Figure 7-25 illustrates the most dangerous predicted intersections and Figure 7-26 displays the most dangerous predicted roads by name.



(a) (b)

Figure 7-19 (a) QGIS HTML Map Tip layer with HTML code example (b) classification performance metrics by road segment: *BRONSON AVE__3Z09YN*

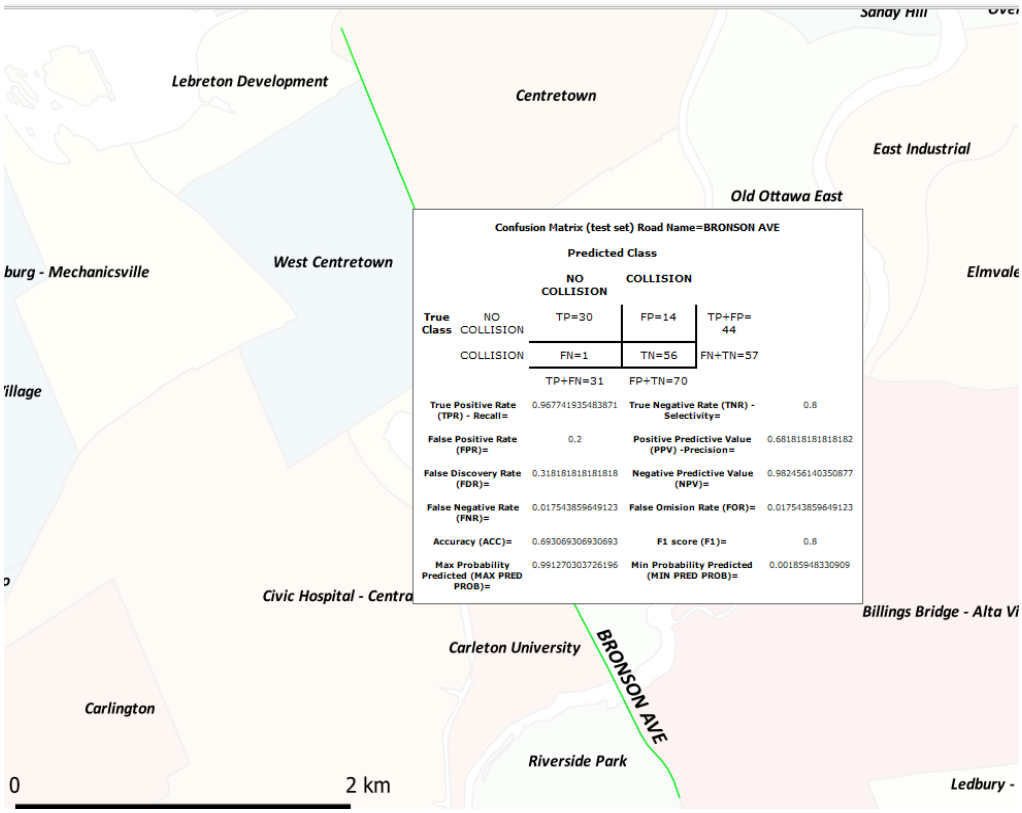


Figure 7-20 Classification performance metrics by road name: *BRONSON AVE*

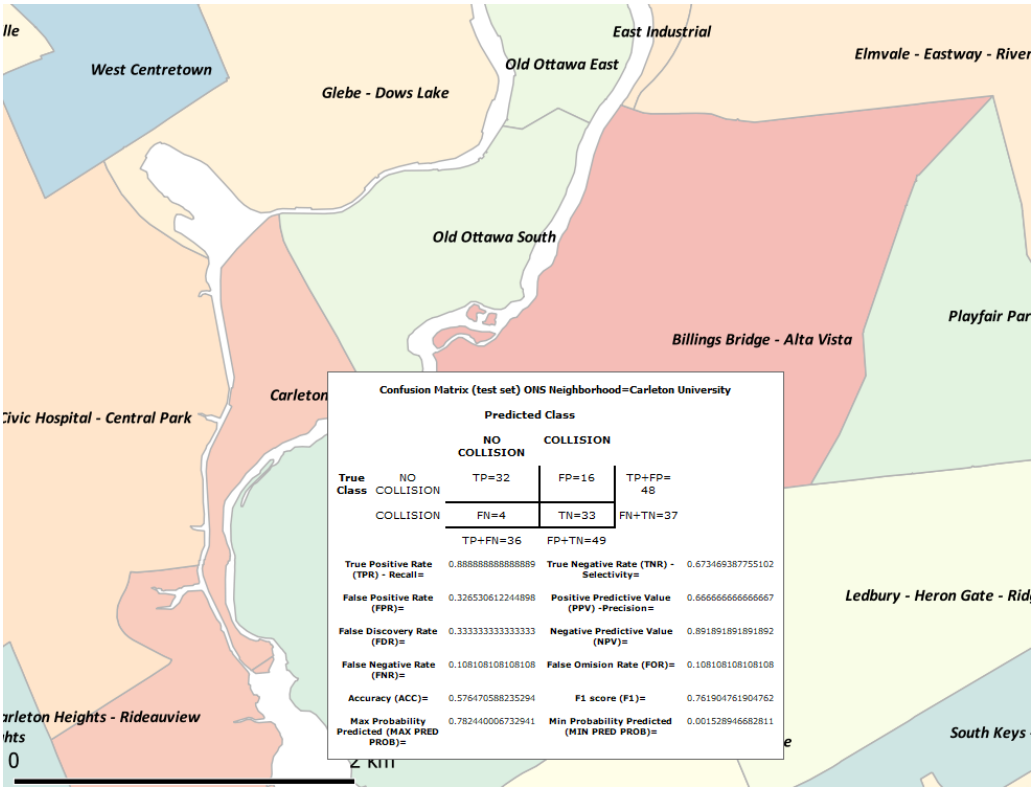


Figure 7-21 Classification performance metrics by neighborhood: *Carleton University*

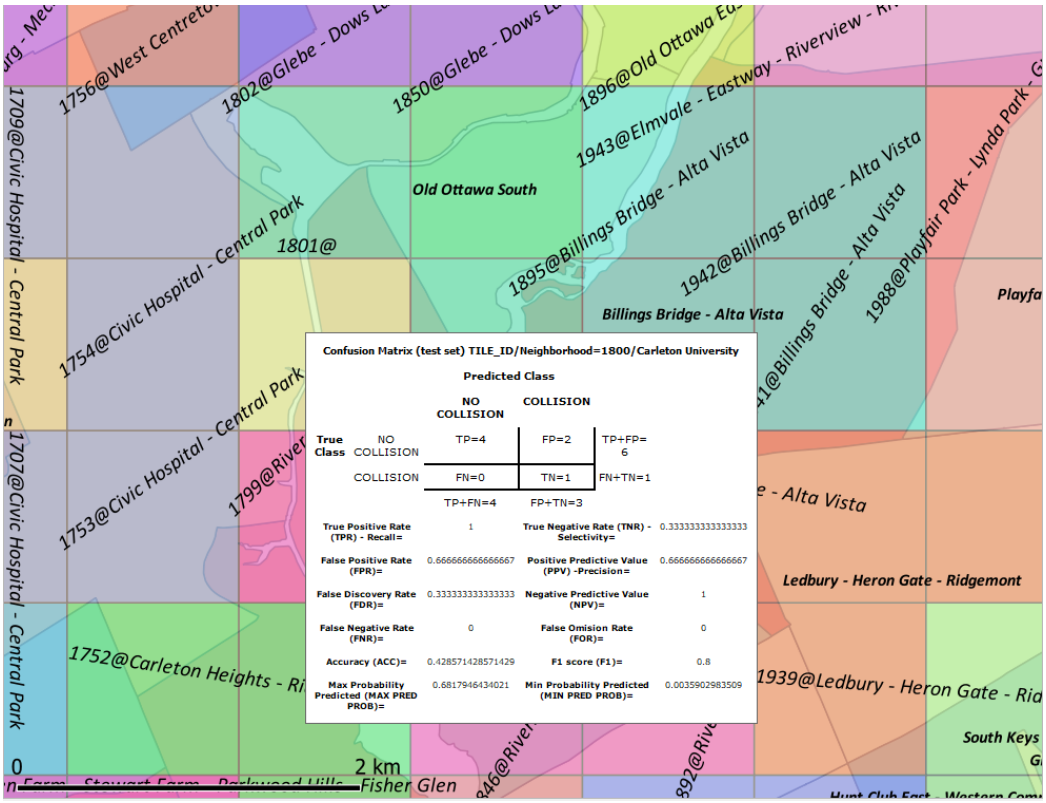


Figure 7-22 Classification performance metrics by tile: *1800/Carleton University*

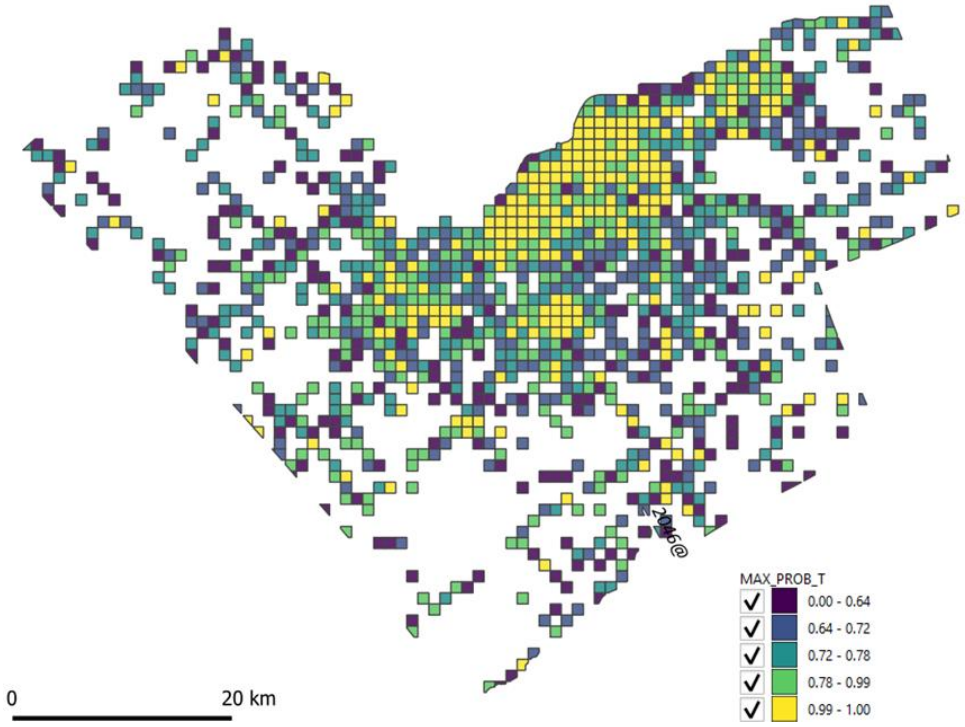


Figure 7-23 Maximum predicted collision probability heat map by tile

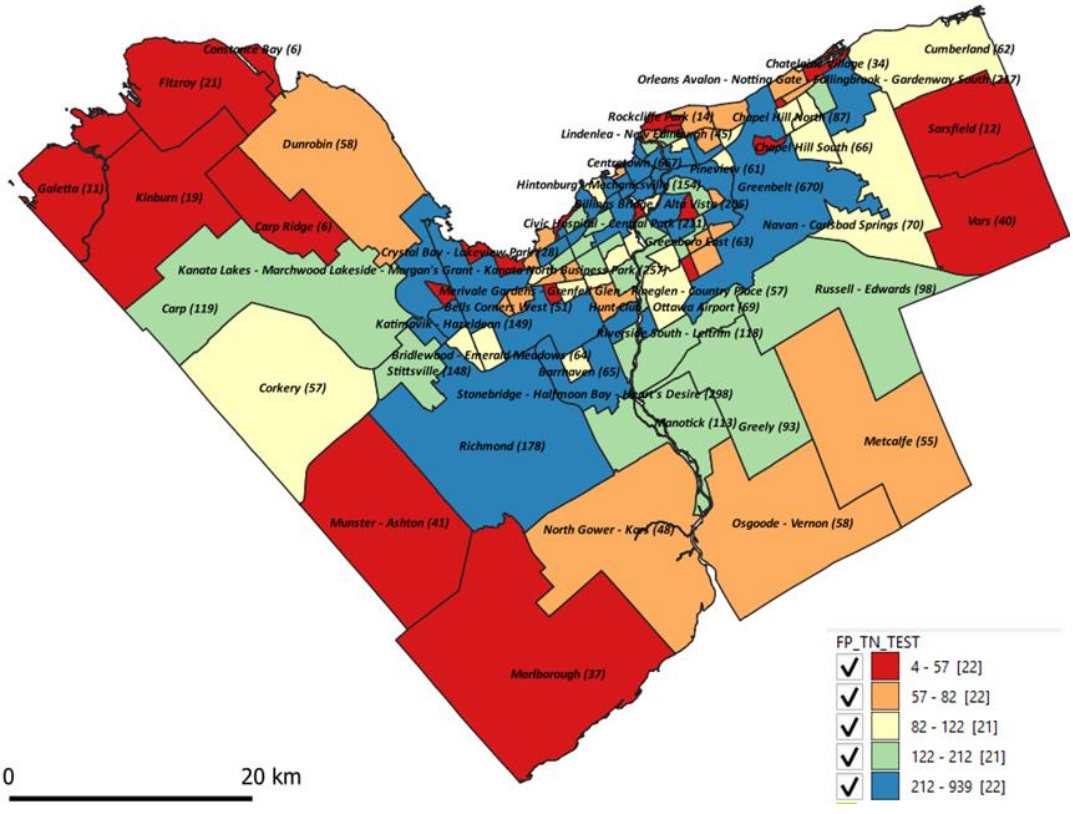


Figure 7-24 Heat map for the total number of collisions predicted by neighborhood

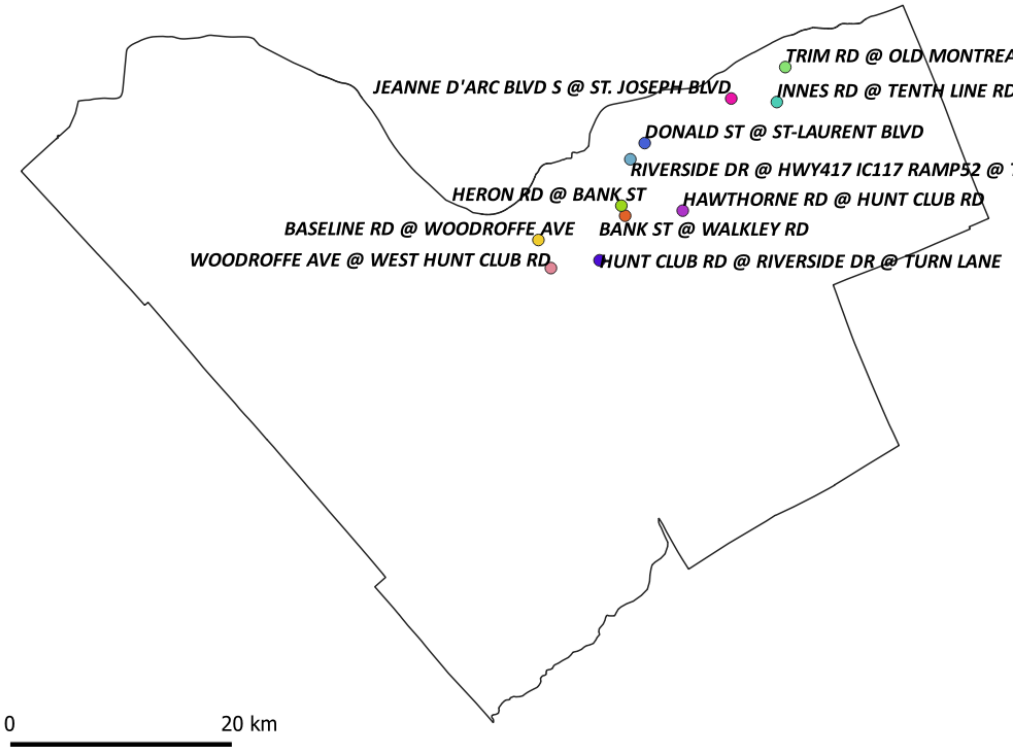


Figure 7-25 Top-10 most dangerous predicted intersections

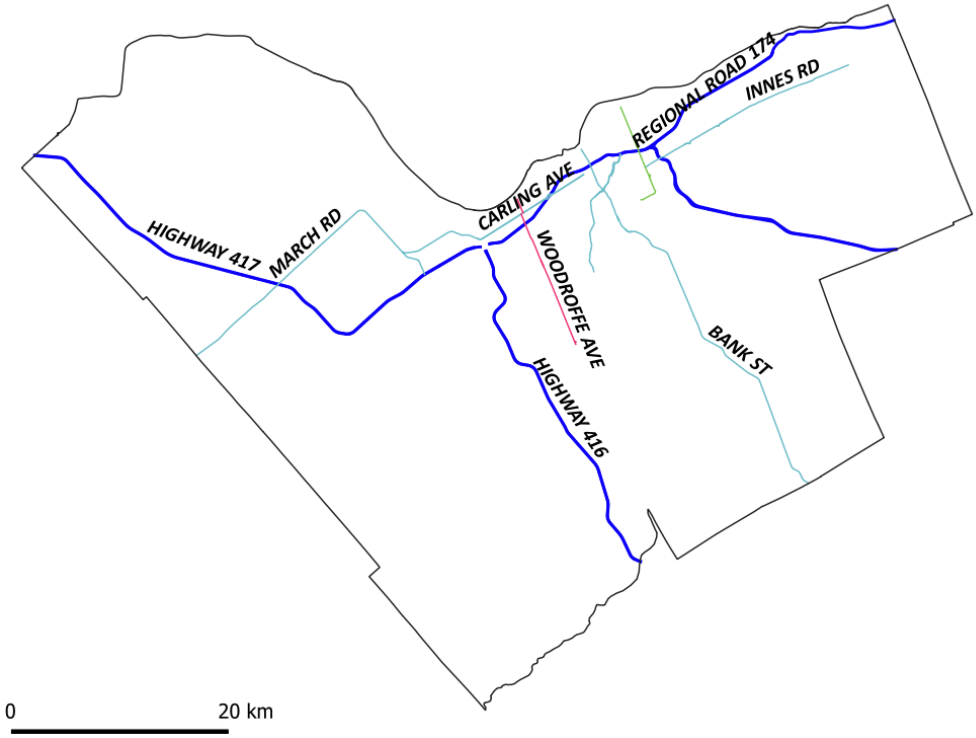


Figure 7-26 Top-10 most dangerous roads predicted, by name

7.2. Prediction of Accident Severity (fatal, injury and property damage)

The original collision dataset includes as a feature the accident severity (CLASSIFICATION_OF_ACCIDENT: 'Fatal', 'Non-Fatal' and 'Property Damage'), feature that was not used in the COLLISION/NO-COLLISION prediction model because it only becomes available after the collision happens. Because we found several research studies related with accident severity, to evaluate and compare the proposed collision prediction framework, a classification model was built for this purpose. For this model, the non-collision samples previously generated (section 5.3.2.3) are not necessary and were thus removed from the dataset.

A class imbalance problem was found in the target variable (CLASSIFICATION_OF_ACCIDENT), as shown in Figure 7-27. The number of samples related with '*0 – Fatal'* is only 71 (0.16%), with the '*1-Injury (Non-fatal)*' and *'2-Property Damage (PD)'* representing 8241 (19.12%) and 34795 (80.72%) respectively. To handle the class imbalance problem, the Synthetic Minority Over-sampling Technique (SMOTE) described in section 6.3 was used.

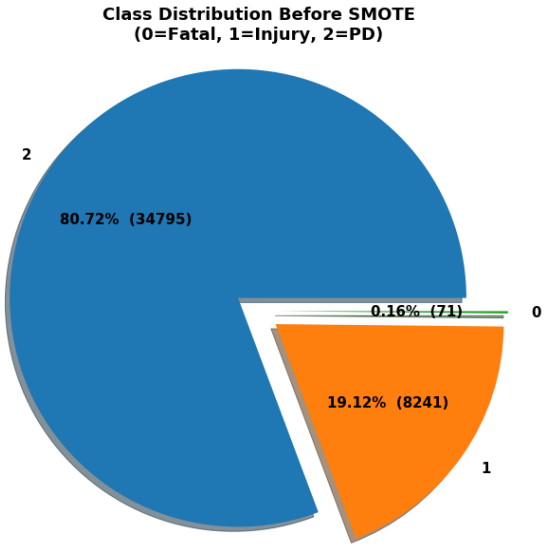


Figure 7-27 Accident severity class distribution (CLASSIFICATION_OF_ACCIDENT)

Because the target variable includes three different values: '*0 – Fatal'*, '*1-Injury'* and '*2-Property Damage'*; we are facing a multiclass classification problem. To solve it, a *One vs Rest* approach was applied, and thus the original multiclass problem

was transformed in three binary classification problems: *Fatal vs Rest*, *Injury vs Rest* and *PD vs Rest*. Based on this, three new target variables were created: (1) CLASSIFICATION_OF_ACCIDENT_0 that classifies the samples as *'0 – Fatal'* or *'Rest'* (includes '1-Injury' and '2-PD'), (2) CLASSIFICATION_OF_ACCIDENT_1 that classifies the samples as *'1-Injury'* or *'Rest'* (includes '*0-Fatal'* and '*2-PD'*) and (3) CLASSIFICATION_OF_ACCIDENT_2 that classifies the samples as '*2-PD'* or *'Rest'* (includes *'0-Fatal'* and *'1-Injury'*). In the following sections we will use *Fatal vs Rest*, *Injury vs Rest* and *PD vs Rest* as a convention to refer to these models.

As such, three models were created using the machine learning algorithm that provides the best results in the prediction of collision/no-collision, the gradient boosted trees (GBT). The results shown here correspond to the step 3 (described in section 6.4), were the original dataset was split into training (70%) and test (30%) and the discriminant threshold related with the maximum Matthews score is used. The same training and test set is used to build and test the three different models.

The classification results using the original dataset and the ones transformed using SMOTE are described in the following sections.

7.2.1. Accident severity models using the original dataset (with class imbalance)

As a baseline, we created the three models using the original dataset.

7.2.1.1. Fatal vs Rest

As a result of the class imbalance problem, as expected, the classification performance metrics over test set for this model is very low. Figure 7-28a shows a AUC score of 0.65, while Figure 7-28b shows an area under PR curve of 0.00289, the confusion matrix in Figure 7-28d shows that the model was only able to identify 4 of the 21 samples, achieving a recall of 19%. The top 5 most important features identified are:

XCOORD, COLLISION_RATE_accident_date, ROAD_LEN, SOLAR_AZIMUTH and ROAD_NAME.

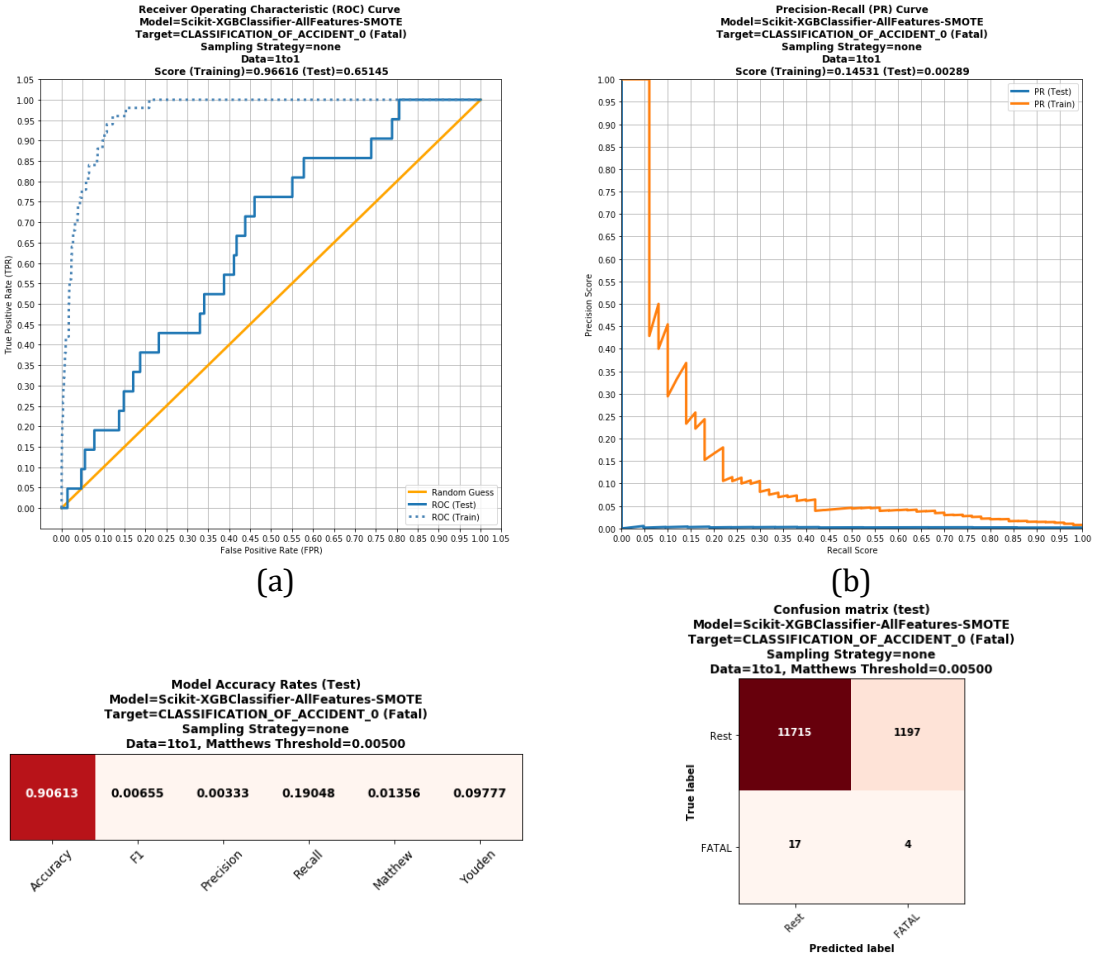
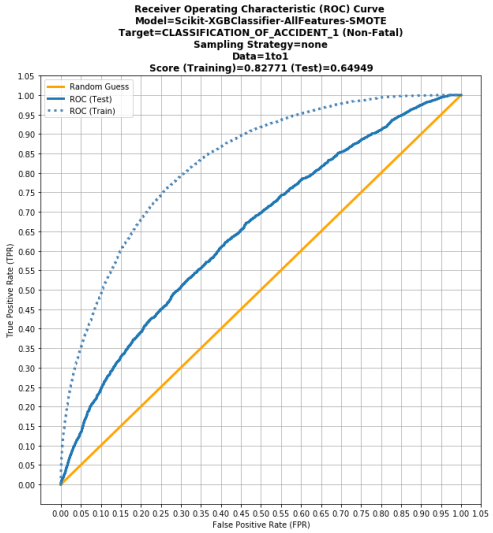


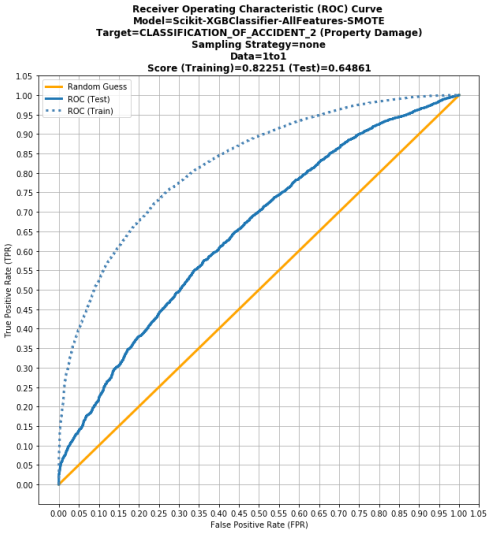
Figure 7-28 *Fatal vs Rest* model using original dataset: (a) ROC curve, (b) precision-recall curve, (c) classification report, and (d) confusion matrix

7.2.1.2. *Injury vs Rest* and *PD vs Rest*

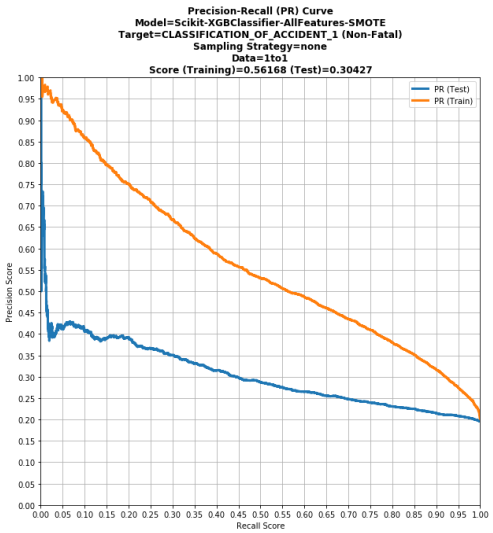
The classification performance results regarding the *Fatal vs Rest*, and *PD vs Rest* models are displayed in Figure 7-29. In general, we can notice a better classification performance for the *PD vs Rest* model, achieving a better recall (98% vs 37%, Figure 7-29g and Figure 7-29h), product of the same class imbalance issues (the number of property damage samples is very high compared with the non-fatal ones).


Receiver Operating Characteristic (ROC) Curve
Model=Scikit-XGBClassifier-AllFeatures-SMOTE
Target=CLASSIFICATION_OF_ACCIDENT_1 (Non-Fatal)
Sampling Strategy=none
Data=1to1
Score (Training)=0.82771 (Test)=0.64949
Random Guess
ROC (Test)
ROC (Train)
True Positive Rate (TPR)
False Positive Rate (FPR)

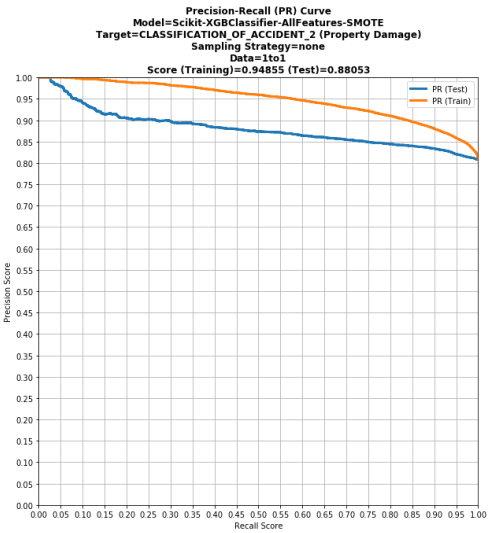
(a)


Receiver Operating Characteristic (ROC) Curve
Model=Scikit-XGBClassifier-AllFeatures-SMOTE
Target=CLASSIFICATION_OF_ACCIDENT_2 (Property Damage)
Sampling Strategy=none
Data=1to1
Score (Training)=0.82251 (Test)=0.64861
Random Guess
ROC (Test)
ROC (Train)
True Positive Rate (TPR)
False Positive Rate (FPR)

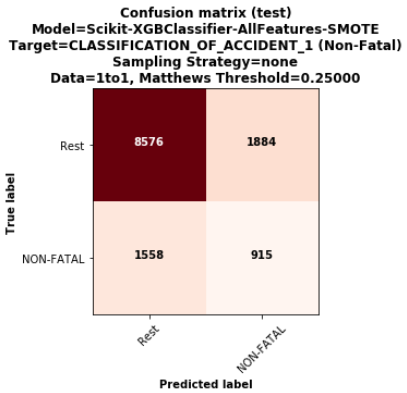
(b)


Precision-Recall (PR) Curve
Model=Scikit-XGBClassifier-AllFeatures-SMOTE
Target=CLASSIFICATION_OF_ACCIDENT_1 (Non-Fatal)
Sampling Strategy=none
Data=1to1
Score (Training)=0.56168 (Test)=0.30427
PR (Test)
PR (Train)
Precision Score
Recall Score

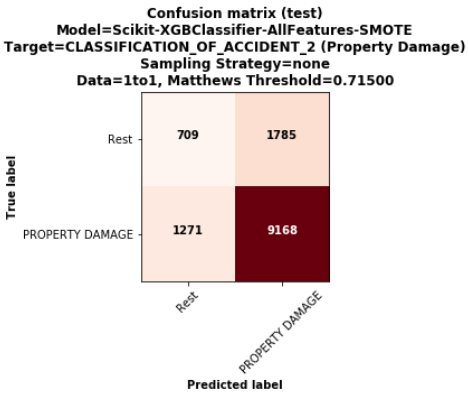
(c)


Precision-Recall (PR) Curve
Model=Scikit-XGBClassifier-AllFeatures-SMOTE
Target=CLASSIFICATION_OF_ACCIDENT_2 (Property Damage)
Sampling Strategy=none
Data=1to1
Score (Training)=0.94855 (Test)=0.88053
PR (Test)
PR (Train)
Precision Score
Recall Score

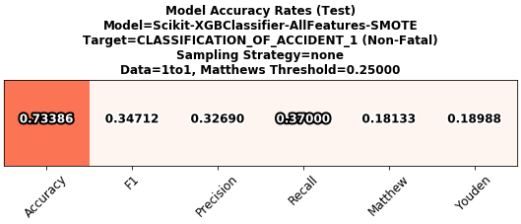
(d)


Confusion matrix (test)
Model=Scikit-XGBClassifier-AllFeatures-SMOTE
Target=CLASSIFICATION_OF_ACCIDENT_1 (Non-Fatal)
Sampling Strategy=none
Data=1to1, Matthews Threshold=0.25000
Rest
NON-FATAL
8576
1884
1558
915
True label
Predicted label

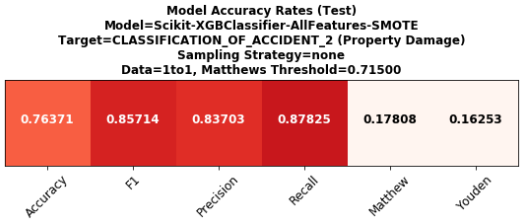
(e)


Confusion matrix (test)
Model=Scikit-XGBClassifier-AllFeatures-SMOTE
Target=CLASSIFICATION_OF_ACCIDENT_2 (Property Damage)
Sampling Strategy=none
Data=1to1, Matthews Threshold=0.71500
Rest
PROPERTY DAMAGE
709
1785
1271
9168
True label
Predicted label

(f)



(g)



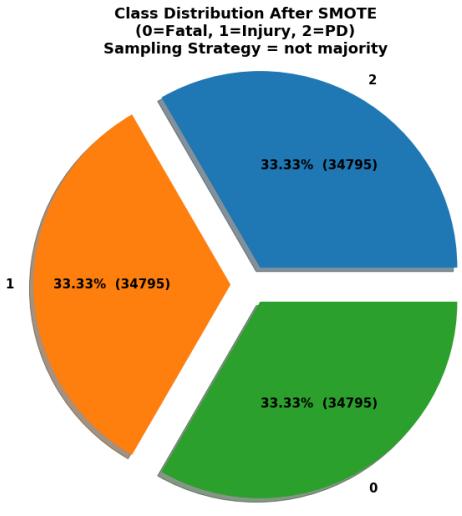
(h)

Figure 7-29 *Injury vs Rest* model using original dataset (a) ROC curve, (c) precision-recall curve, (e) confusion matrix, and (g) classification report; *PD vs Rest* using original dataset (b) ROC curve, (d) precision-recall curve, (f) confusion matrix, and (h) classification report

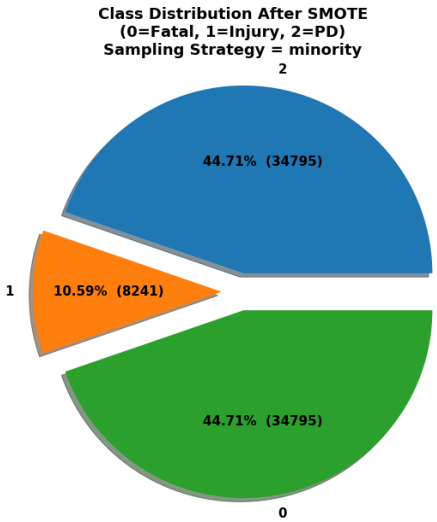
Both models share the same top-5 features: SOLAR_AZIMUTH, SOLAR_ELEVATION, YCOORD, XCOORD and ROAD_SEGMENT.

7.2.2. Accident severity models using SMOTE

Two approaches were used to balance the dataset using SMOTE: (1) resample all classes but the majority class ('not majority') and (2) resample only the minority class ('minority').



(a)



(b)

Figure 7-30 Class distribution after SMOTE using: (a) not majority strategy and (b) minority strategy

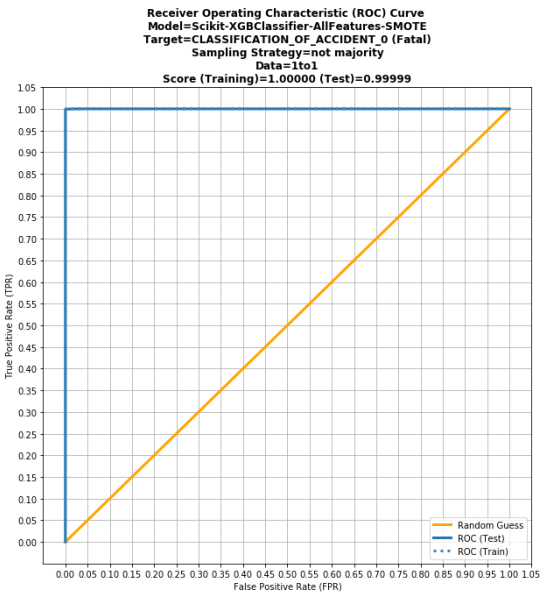
The class distribution after each of the resampling techniques is illustrated in Figure 7-30, where Figure 7-30a shows the class distribution after resampling classes 0

(Fatal) and 1 (Non-fatal, Injury) to be equal to class 2 (PD). Figure 7-30b shows the class distribution after resampling the minority class (0 – Fatal) to be equal to the class 2.

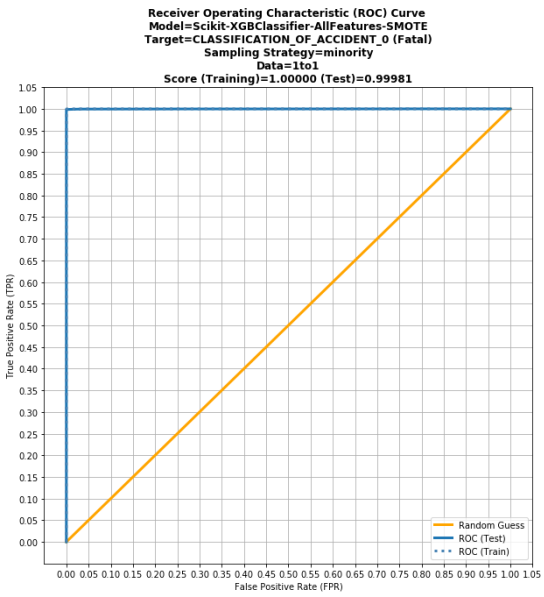
For each class, the classification performance results are described considering each of the two resampling strategies in the following sections.

7.2.2.1. *Fatal vs Rest* using SMOTE

After applying the two SMOTE resampling strategies, the classification performance results related with the model *Fatal vs Rest* are displayed in Figure 7-31. We can notice an important improvement for both SMOTE resampling strategies compared with the original imbalanced dataset, while there is no significant difference in results between them – both are characterized by an almost perfect AUC score (0.999, Figure 7-31a and Figure 7-31b), an area under the precision-recall curve (0.99, Figure 7-31c and Figure 7-31d) and a very high accuracy (99.9%, Figure 7-31g and Figure 7-31h). Any of the models could be selected as the best one.



(a)



(b)

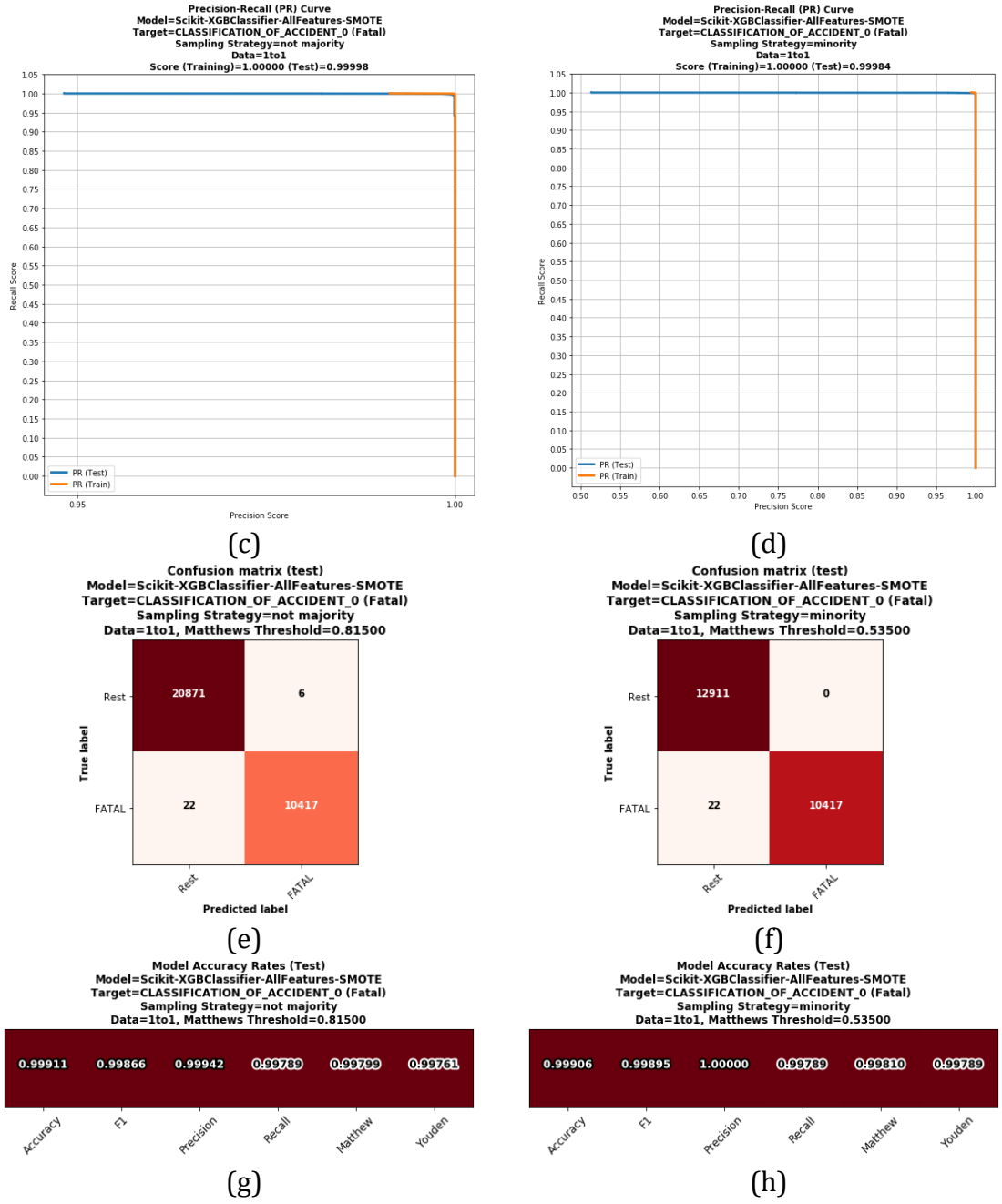
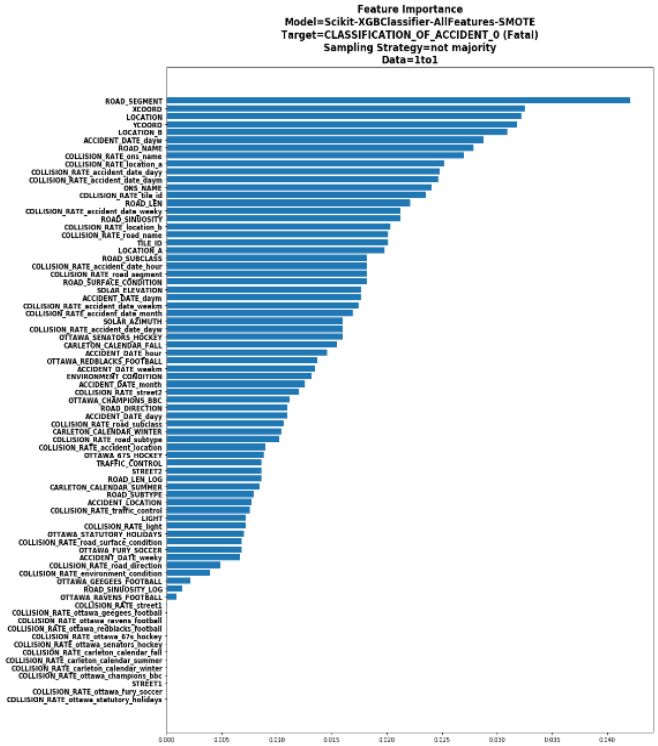


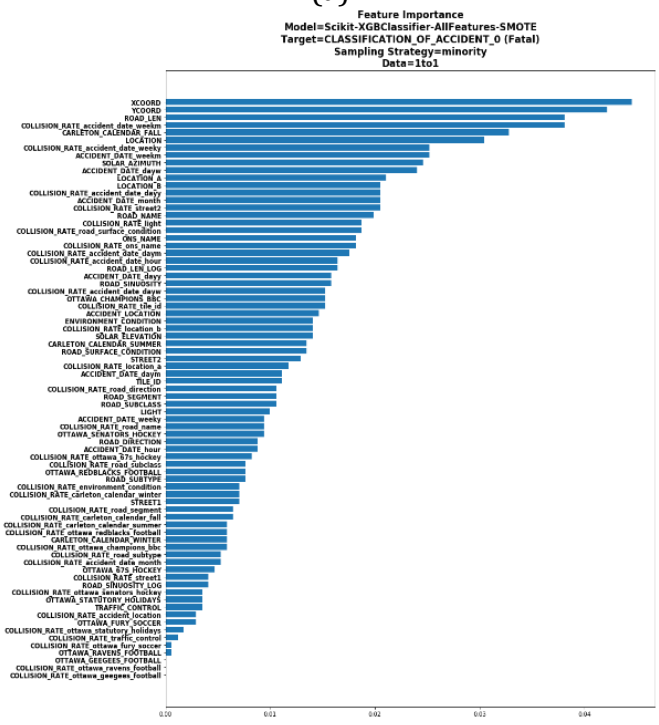
Figure 7-31 ***Fatal vs Rest*** **model using SMOTE resampling to the majority class (a) ROC curve, (c) precision-recall curve, (e) confusion matrix, (g) classification report; and using SMOTE resampling only the minority class (b) ROC curve, (d) precision-recall curve, (f) confusion matrix, and (h) classification report**

In terms of the most important features, we can notice that both models exhibit differences, where the most noticeable are the CARLETON_CALENDAR_FALL as part of the top-5 features for the one that resamples the minority (Figure 7-32b)

and LOCATION/LOCATION_B as part of the second resampling strategy (Figure 7-32a).



(a)

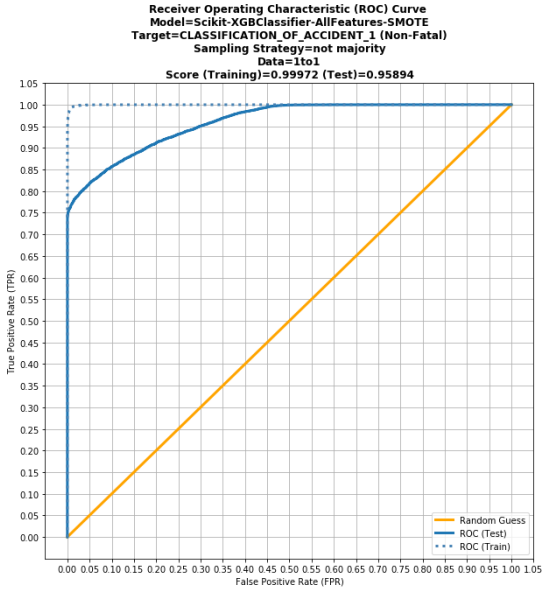


(b)

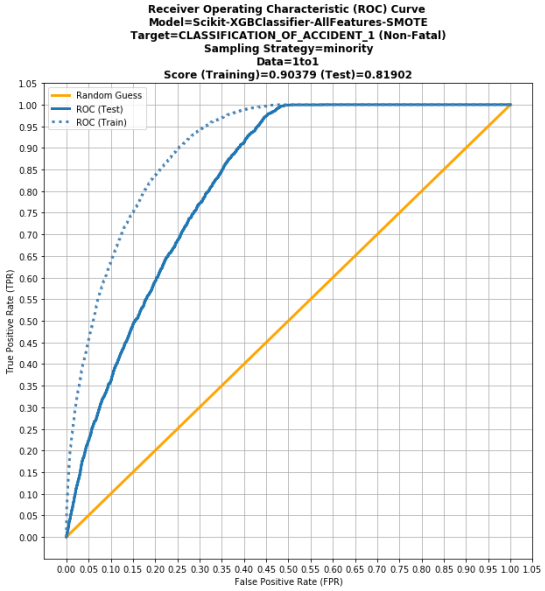
Figure 7-32 Most important features for *Fatal vs Rest* model using SMOTE with (a) not-majority resampling and (b) minority resampling

7.2.2.2. *Injury vs Rest* using SMOTE

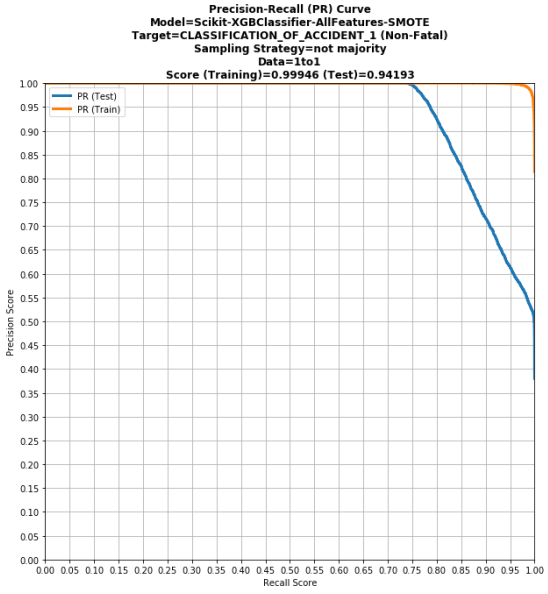
Considering the *Injury vs Rest* model, we can notice a big difference in classification results related with the resampling to not-majority strategy.



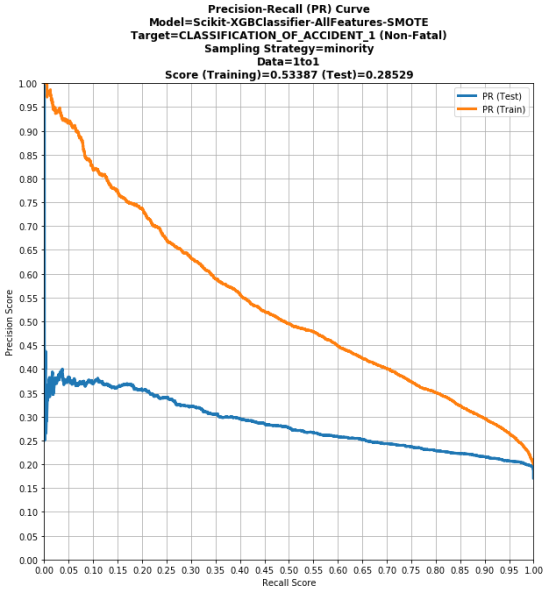
(a)



(b)



(c)



(d)

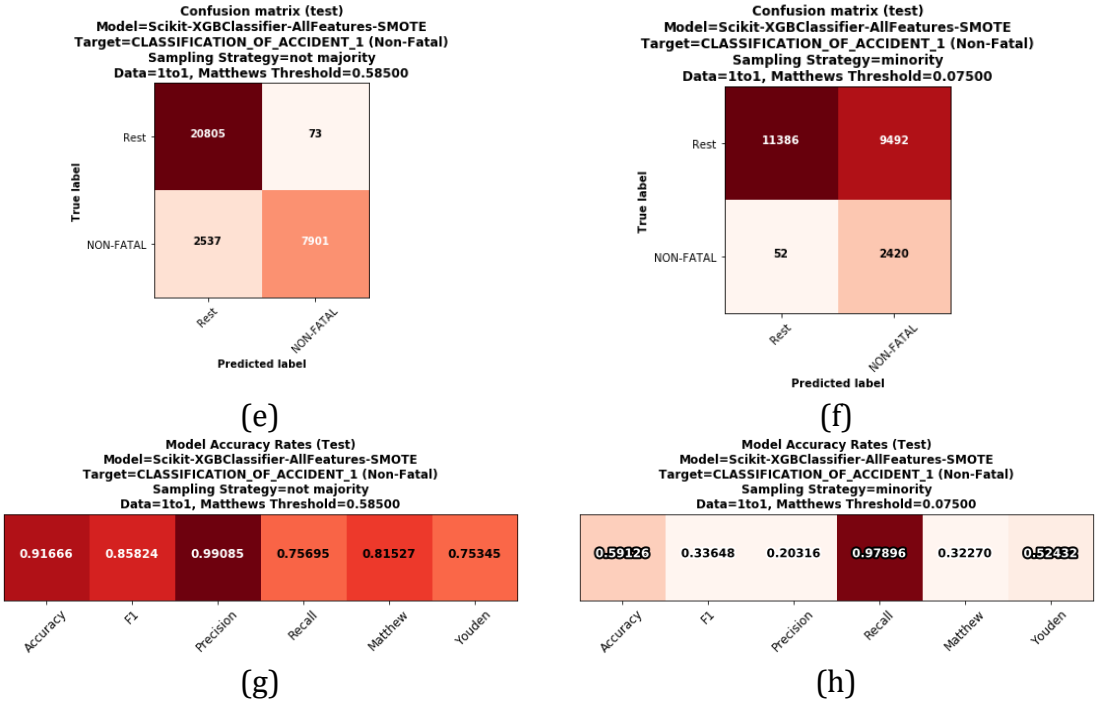
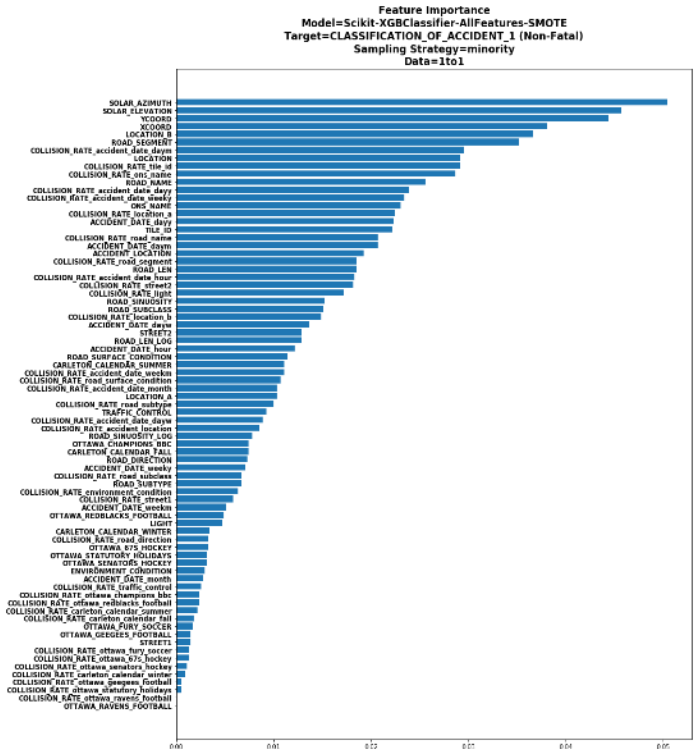


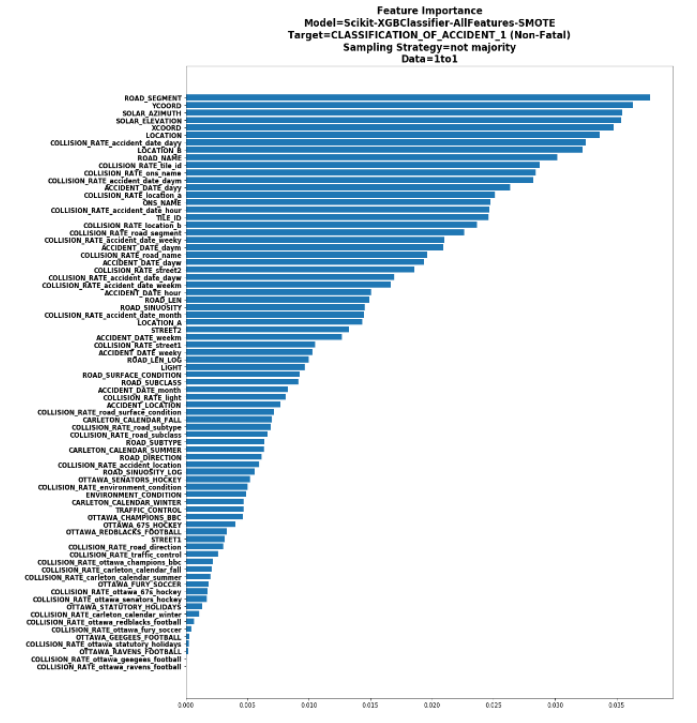
Figure 7-33 ***Injury vs Rest*** **model using SMOTE resampling to the majority class (a) ROC curve, (c) precision-recall curve, (e) confusion matrix, (g) classification report; and using SMOTE resampling only the minority class, (b) ROC curve, (d) precision-recall curve, (f) confusion matrix, and (h) classification report**

It is indicated by a higher AUC (0.95 vs 0.81, Figure 7-33a and Figure 7-33b) and a higher area under precision-recall curve (0.94 vs 0.26, in Figure 7-33c and Figure 7-33d), and a very high accuracy (91% vs 59%), as it can be noticed in the confusion matrix (Figure 7-33e and Figure 7-33f) and also in the classification report (Figure 7-33g and Figure 7-33h). Without any doubt, the best model is the one related with the not-majority resampling.

The top-5 most important features are the same, but in a different order (SOLAR_AZIMUTH, SOLAR_ELEVATION, XCOORD, YCOORD and LOCATION_B), as it can be observed in Figure 7-34a and Figure 7-34b.



(a)

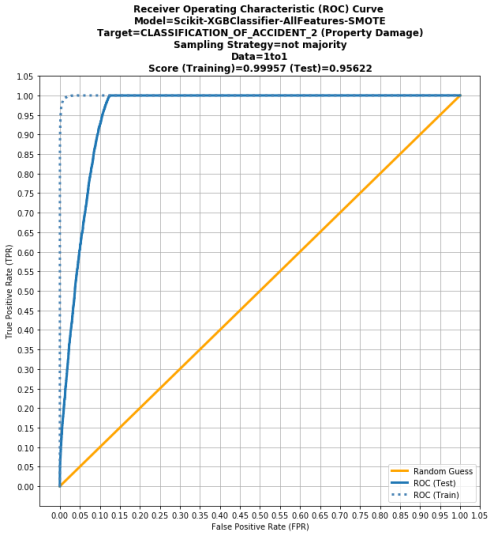


(b)

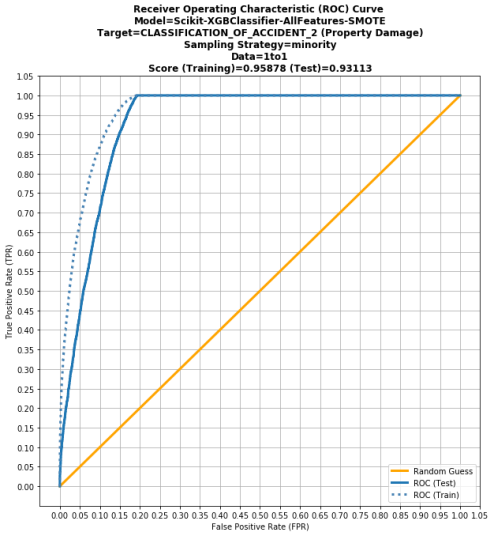
Figure 7-34 Most important features for *Injury vs Rest* model using SMOTE with (a) not-majority resampling and (b) minority resampling

7.2.2.3. *PD vs Rest* using SMOTE

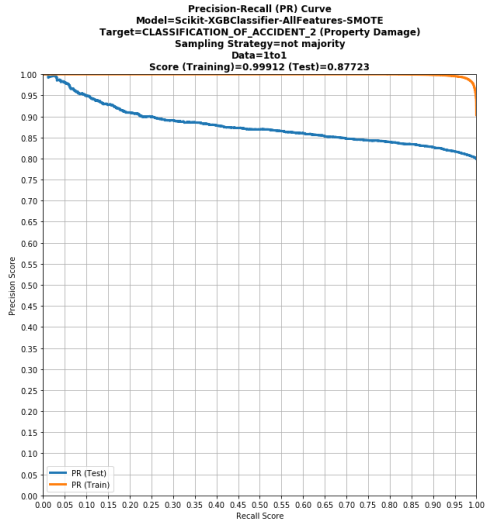
The last model is related to the classification of property damage accidents, *PD vs Rest*.



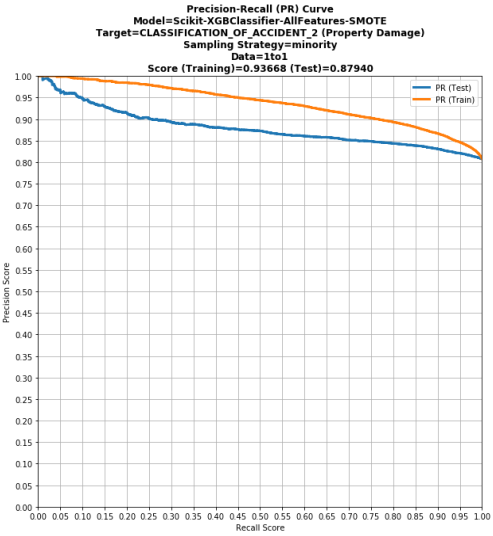
(a)



(b)



(c)



(d)

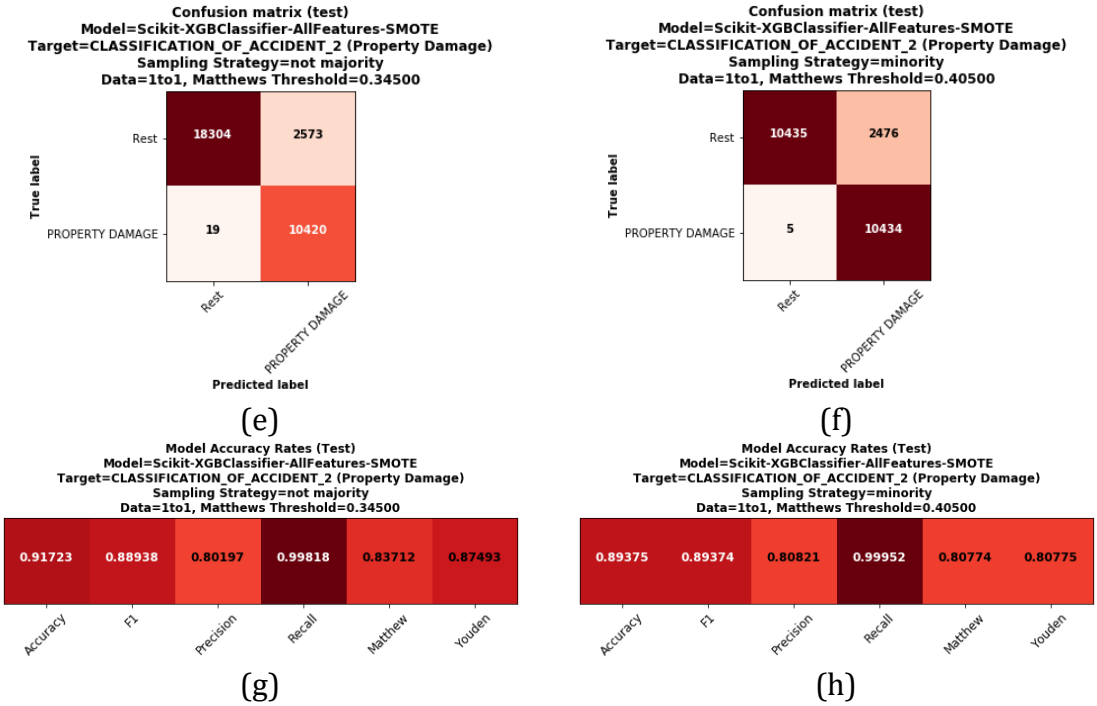


Figure 7-35 *PD vs Rest* model using SMOTE resampling to the majority class (a) ROC curve, (c) precision-recall curve, (e) confusion matrix, (g) classification report; and using SMOTE resampling only the minority class (b) ROC curve (d) precision-recall curve, (f) confusion matrix, and (h) classification report

The not-majority SMOTE resampling strategy, as in the previous case, provides the best results, including a higher AUC score (95% vs 93%, in Figure 7-35a and b), a slightly lower area under precision-recall curve (87.7% vs 87.9%, Figure 7-35c and d) and a higher accuracy (91% vs 89%, Figure 7-35g and h). The best model is therefore the one related to the not-majority SMOTE sampling strategy.

As for the previous model, the top-5 most important features are the same, but in different order: SOLAR_ELEVATION, SOLAR_AZIMUTH, XCOORD, YCOORD and ROAD_SEGMENT (Figure 7-36a and Figure 7-36b).

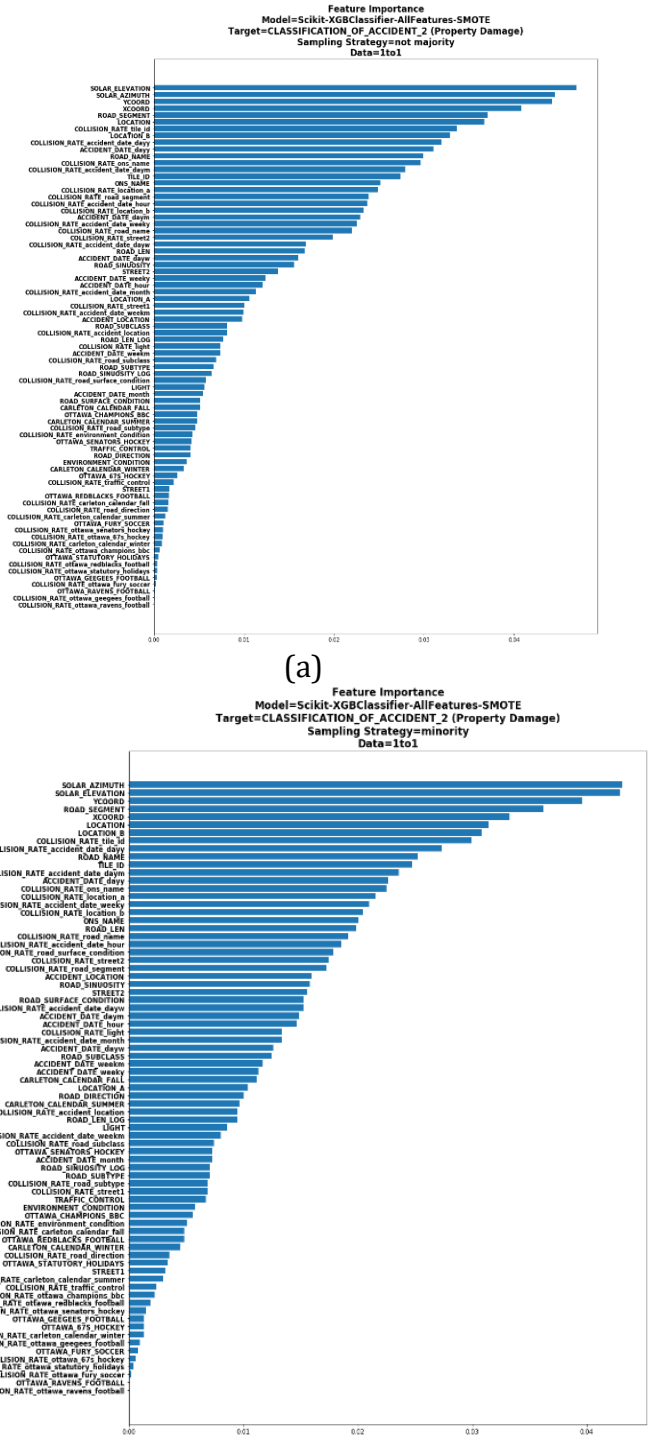


Figure 7-36 Most important features for *PD vs Rest* model using SMOTE with (a) non-majority resampling and (b) minority resampling

7.2.3. Model Comparison and Evaluation

Considering the different models created in section 7.2.1 and 7.2.2, the best model was obtained using the SMOTE sampling strategy to the not-majority class as shown in Table 7-14.

We can notice that the *Injury vs Rest* and *PD vs Rest* models achieve similar results in terms of accuracy (91%), but the first is better in terms of precision (99% vs 80%) and worse in term of recall (75% vs 99%). In general, the *Fatal vs Rest* model achieves a very high performance. In terms of the execution time, the ones related with SMOTE sampling strategy to the not-majority class are the most expensive (Table 7-15). After that, and using the approach described in section 7.1.5, a feature selection process was implemented to reduce the number of features per model and thus reduce the execution time.

Table 7-14 Model comparison in terms of performance metrics over the test set using the best threshold (Matthews)

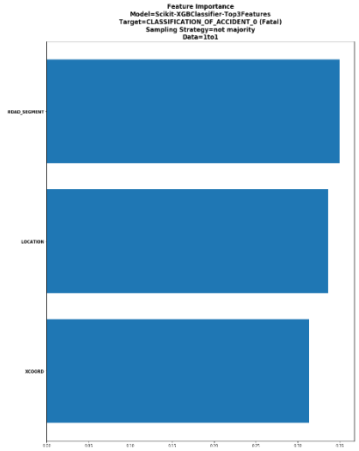
Model	Number of TP + TN	Accuracy Score (%)	Precision Score (%)	Recall Score (%)	F1 Score (%)	Matthews Score
GBT - Fatal vs Rest						
NO SMOTE	11719	90.613	0.333	19.048	0.655	0.01356
SMOTE NOT MAJORITY	**31288**	**99.911**	**99.942**	**99.789**	**99.866**	**0.99799**
SMOTE MINORITY	23328	99.906	100.000	99.789	99.985	0.99810
GBT – Injury vs Rest						
NO SMOTE	9491	73.386	32.690	37.000	34.712	0.18133
SMOTE NOT MAJORITY	**28706**	**91.666**	**99.085**	**75.695**	**85.824**	**0.81527**
SMOTE MINORITY	13806	59.126	20.316	97.896	33.648	0.52432
GBT - PD vs Rest						
NO SMOTE	9877	76.371	83.703	87.825	85.714	0.17808
SMOTE NOT MAJORITY	**28724**	**91.723**	**80.197**	**99.818**	**88.938**	**0.83712**
SMOTE MINORITY	20869	89.375	80.821	99.952	89.374	0.80774

Table 7-15 Model comparison in terms of execution time over test set using the best threshold (Matthews)

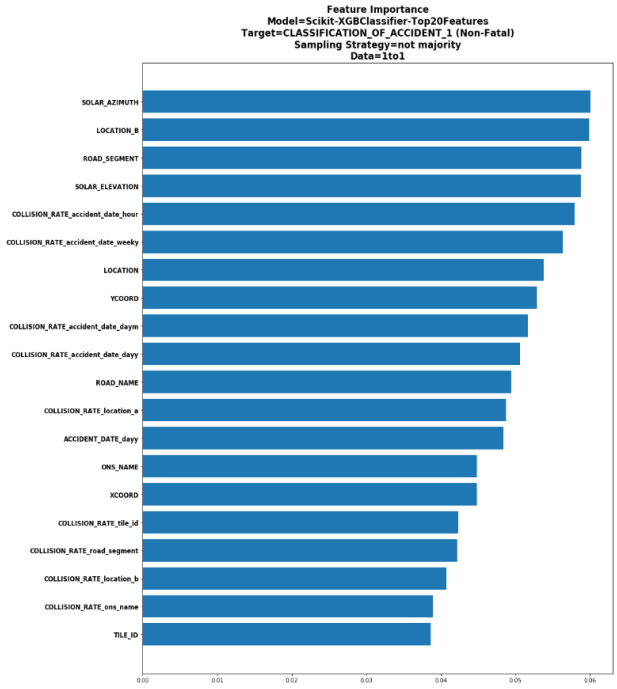
Model	SMOTE	Total Execution Time
GBT – Fatal vs Rest	NO SMOTE	5 minutes
	NOT MAJORITY CLASS	**36 minutes**

	MINORITY CLASS	19 minutes
GBT - Injury vs Rest	NO SMOTE	21 minutes
	NOT MAJORITY CLASS	**71 minutes**
	MINORITY CLASS	44 minutes
GBT - PD vs Rest	NO SMOTE	21 minutes
	NOT MAJORITY CLASS	**68 minutes**
	MINORITY CLASS	46 minutes

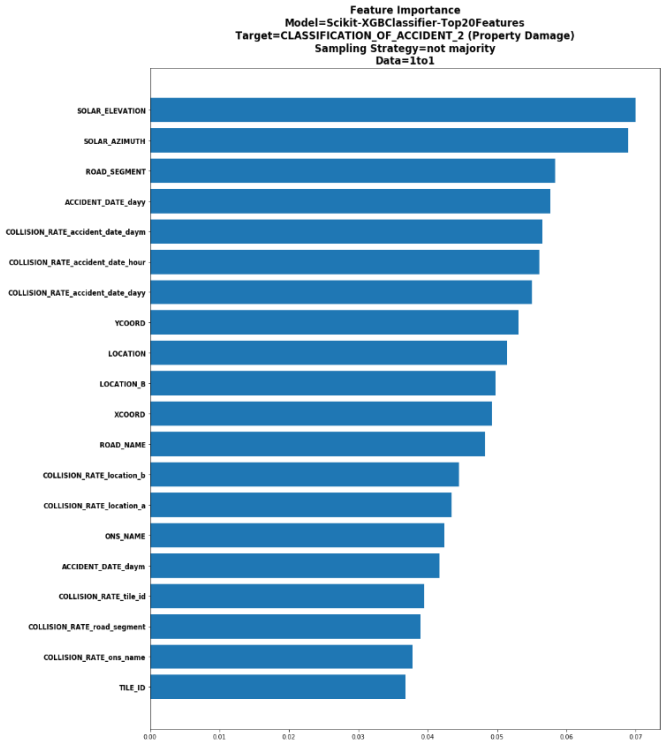
The best models were obtained using: top-3 features for *Fatal vs Rest* (ROAD_SEGMENT, LOCATION and XCOORD; in Figure 7-37a), top-20 features for *Injury vs Rest* (SOLAR_AZIMUTH, LOCATION_B, ROAD_SEGMENT, SOLAR_ELEVATION, COLLISION_RATE_accident_date_hour, COLLISION_RATE_accident_date_weeky, LOCATION, YCOORD, COLLISION_RATE_accident_date_daym, COLLISION_RATE_accident_date_dayy, ROAD_NAME, COLLISION_RATE_location_a, ACCIDENT_DATE_dayy, ONS_NAME, XCOORD, COLLISION_RATE_tile_id, COLLISION_RATE_road_segment, COLLISION_RATE_location_b, COLLISION_RATE_ons_name and TILE_ID; in Figure 7-37b) and top-20 features for *PD vs Rest* (SOLAR_ELEVATION, SOLAR_AZIMUTH, ROAD_SEGMENT, ACCIDENT_DATE_dayy, COLLISION_RATE_accident_date_daym**,** COLLISION_RATE_accident_date_hour, COLLISION_RATE_accident_date_dayy, YCOORD, LOCATION, LOCATION_B, XCOORD, ROAD_NAME, COLLISION_RATE_location_b, COLLISION_RATE_location_a, ONS_NAME, ACCIDENT_DATE_daym, COLLISION_RATE_tile_id, COLLISION_RATE_road_segment, COLLISION_RATE_ons_name and TILE_ID; in Figure 7-37c).



(a)



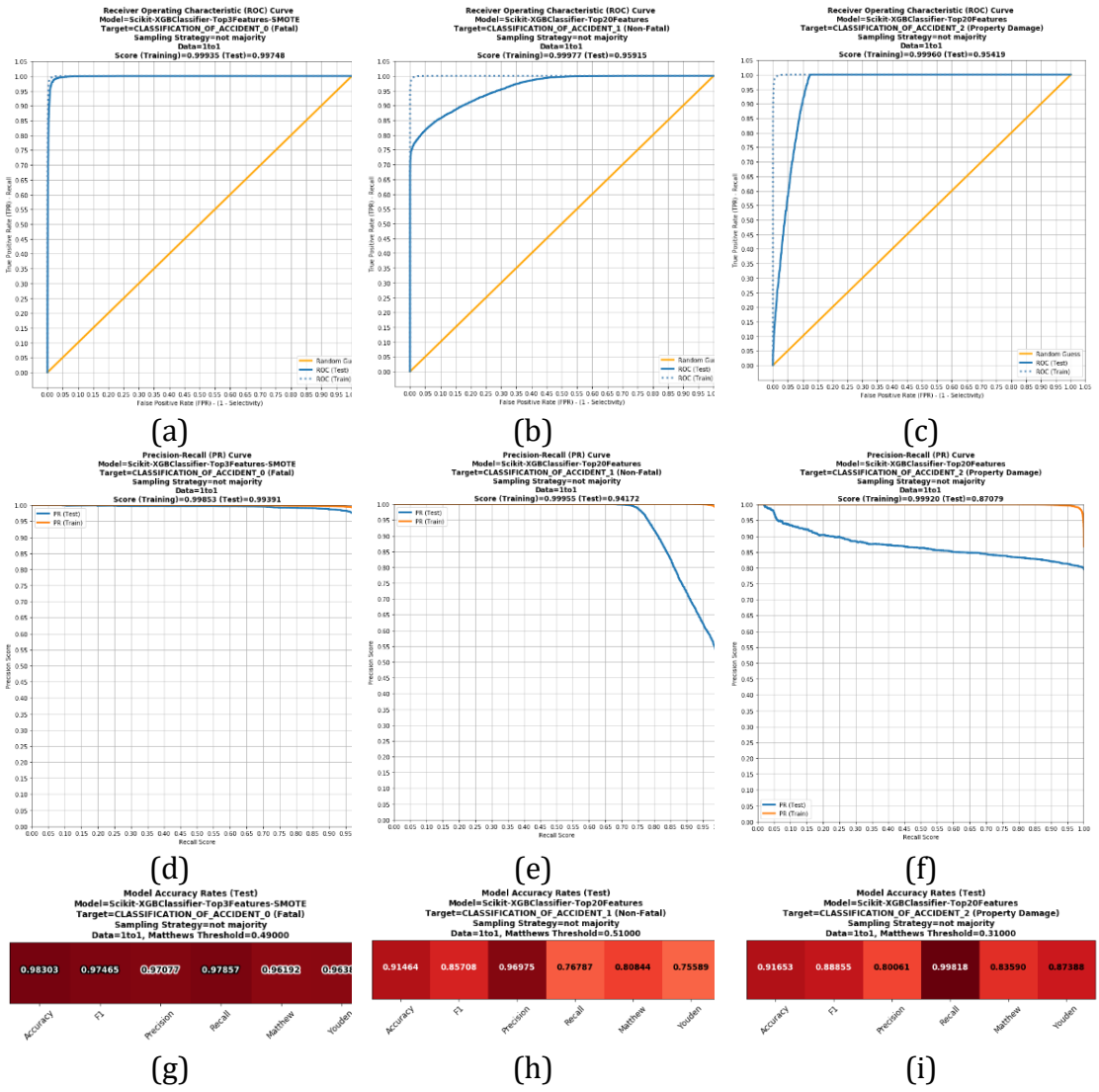
(b)

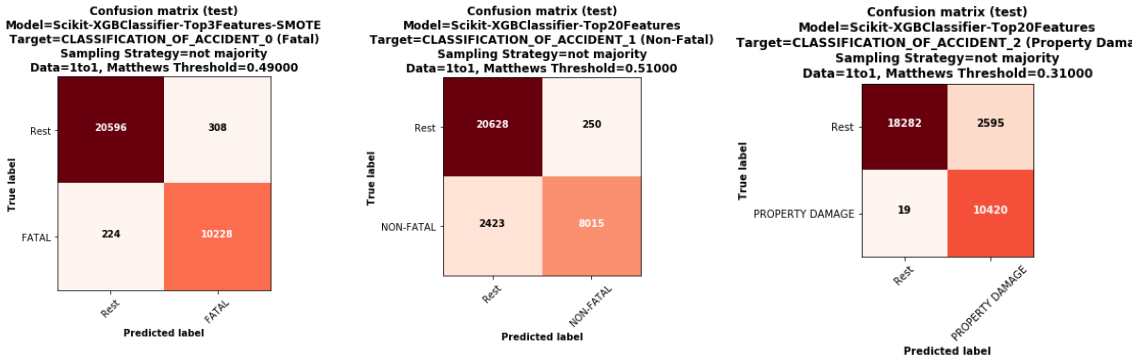


(c)

Figure 7-37 Most important features using SMOTE and resampling to non-majority class for (a) *Fatal vs Rest* model with top-3 features, (b) *Injury vs Rest* model with top-20 features, and (c) *PD vs Rest* model with top-20 features

We can notice that three variables are present in all the models with different importance (ROAD_SEGMENT, XCOORD and LOCATION) and also the top-20 features for *Injury vs Rest* and *PD vs Rest* and are practically the same.





(j) (k) (l)

Figure 7-38 Classification performance using SMOTE and resampling to non-majority class for *Fatal vs Rest* model with top-3 features (ROC curve in a, Precision-Recall curve in d, classification report in g, and confusion matrix in j), *Injury vs Rest* model with top-7 features (ROC curve in b, Precision-Recall curve in e, classification report in h, and confusion matrix in k) and *PD vs Rest* model with top-6 features (ROC curve in c, Precision-Recall curve in f, classification report in i, and confusion matrix in l)

Regarding the classification performance, Figure 7-38 illustrates the ROC and precision-recall curves, as well as the classification report and the confusion matrices for each of the model created based on the top-n features. We can notice an almost perfect score for *Fatal vs Rest* model in terms of AUC (Figure 7-38a) and area under precision-recall curve (Figure 7-38b).

For *Injury vs Rest* and *PD vs Rest* our model is not as good as *Fatal vs Rest*. The final model comparison is shown in Table 7-16 and Table 7-17.

Table 7-16 Final model comparison in terms of performance metrics over the test set using the best threshold (Matthews)

Model	Number of TP + TN	Accuracy Score (%)	Precision Score (%)	Recall Score (%)	F1 Score (%)	Matthews Score
GBT - Fatal vs Rest						
NO SMOTE – All Features	11719	90.613	0.333	19.048	0.655	0.01356
SMOTE NOT MAJORITY – All Features	31288	99.911	99.942	99.789	99.866	0.99799

SMOTE MINORITY – All Features	23328	99.906	100.000	99.789	99.985	0.99810
SMOTE NOT MAJORITY – top3 features	**30824**	**98.303**	**97.077**	**97.857**	**97.465**	**0.96192**
GBT – Injury vs Rest						
NO SMOTE – All Features	9491	73.386	32.690	37.000	34.712	0.18133
SMOTE NOT MAJORITY – All Features	28706	91.666	99.085	75.695	85.824	0.81527
SMOTE MINORITY – All Features	13806	59.126	20.316	97.896	33.648	0.52432
SMOTE NOT MAJORITY – top20 features	**28643**	**91.464**	**96.975**	**76.787**	**85.708**	**0.75589**
GBT - PD vs Rest						
NO SMOTE – All Features	9877	76.371	83.703	87.825	85.714	0.17808
SMOTE NOT MAJORITY – All Features	28724	91.723	80.197	99.818	88.938	0.83712
SMOTE MINORITY – All Features	20869	89.375	80.821	99.952	89.374	0.80774
SMOTE NOT MAJORITY – top20 features	**28702**	**91.653**	**80.061**	**99.818**	**88.855**	**0.83590**

Table 7-17 Final model comparison in terms of execution time over test set using the best threshold (Matthews)

Model	SMOTE	Total Execution Time
GBT – Fatal vs Rest	NO SMOTE - All Features	5 minutes
	NOT MAJORITY CLASS – All Features	36 minutes
	MINORITY CLASS – All Features	39 minutes
	NOT MAJORITY CLASS – top3 Features	**9 minutes**
GBT - Injury vs Rest	NO SMOTE – All Features	21 minutes
	NOT MAJORITY CLASS – All Features	71 minutes
	MINORITY CLASS – All Features	44 minutes
	NOT MAJORITY CLASS – top20 Features	**30 minutes**
GBT - PD vs Rest	NO SMOTE – All Features	21 minutes

	NOT MAJORITY CLASS – All Features	68 minutes
	MINORITY CLASS – All Features	46 minutes
	NOT MAJORITY CLASS – top20 Features	**32 minutes**

7.3. Final Comparison Results

Comparing our results with the ones in the literature that have similar classification objectives (i.e. accident frequency and accident severity classification), Table 7-18 summarizes the results. It is important to mention that the comparison results illustrated are based on the reported results (we did not run the models by ourselves), using the order of magnitude of the different classification performance parameters (i.e. accuracy, recall, precision, AUC).

We can notice that our results are better for the accident frequency (accidents / no accidents) compared with existing models ([31], [35] and [53]), as illustrated in Table 7-18 (Classification Objective = Accident frequency (accident / no accidents).

For accident severity (or fatal, injury and PDO), the obtained results in this thesis are mixed: (1) for the *'fatal'* class, they are better than the others (98.3% vs 87.6% [50] and 0% [51]), (2) for the *'injury'* type of accidents, they are better than the ones achieved by [50] (91.5% vs 78.4%), but not as good as the ones obtained by [51] (96.4%), and (3) for the *'non-injury'* class, they are better than [51] (91.7% vs 88.5%), but not as good as [50] (100%).

Table 7-18 Comparison of results with the literature

Literature Work (Paper)			**Our Work**	
Reference	**Method Used / Variant**	**Results**	**Method Used / Variant**	**Results**
Classification Objective = Accident frequency (accidents / no accidents)				
[31]	Neural Networks / Support Vector Machines	Accuracy = 76.7% (using 6 variables)	Gradient Boosted Trees using maximum Matthews Correlation Coefficient related threshold and top-7 features	Accuracy = 79.2% AUC = 0.817 Recall = 98.2% Precision = 71.0%
[35]	Neural Networks / K-Means attribute weighting	Accuracy = 74.15%, AUC = 0.742 (using 5 variables and PCA)		

[53]	Gradient Boosted Trees / 84	Accuracy = 68.59% Recall = 90% Precision = 30% (using 84 variables)		
Classification Objective = Accident severity (fatal, injury and PDO)				
[50]	Ensemble of CART Decision Trees, TreeNet and Random Forest	Accuracy = 87.6% (fatal), 78.4% (injury) and 100.0% (non-injury) (using 48 variables)	Gradient Boosted Trees using SMOTE to not-majority class strategy sampling and maximum Matthews Correlation Coefficient related threshold	Accuracy = 98.3% (fatal) using top-3 features, 91.5% (injury, non-fatal) using top-20 features, and 91.7% (non-injury) using top-20 features
[51]	CART Decision Trees	Accuracy = 0% (fatal), 96.4% (injury) and 88.5% (non-injury) (using 21 variables)		

8. Conclusions

8.1. Summary

The fundamental objective of this research is to propose an innovative general framework for traffic accident collision using machine learning algorithms and GIS visualizations and demonstrate its validity by implementing a proof-of-concept of such a framework using public available datasets for the City of Ottawa, Canada.

The proposed general framework for traffic collision integrates well-known collision prediction risks factors with social events and engineered (generated) features, while also considering which of them could be obtained in real-time using existing sensor technology. As such, beyond the existing historical datasets, in the feature engineering process, a series of generated geospatial (i.e. roads, neighborhoods, tile), solar (solar azimuth and solar elevation), social events (i.e. statutory holidays, sport events, etc.), location and mathematical features (i.e. collision rate per road segment, road name, neighborhood, day of the year, day of the month, tile, etc.) features are identified to contribute with various strengths to the classification performance.

Two different classification models were created: (1) a model to predict accident frequency (collision/no-collision) and (2) a model to predict accident severity (fatal, injury and property damage). In the first model (accident frequency: collision/no-collision), a process to create the negative class (no-collision) was necessary, while for the second model (accident severity prediction: fatal, injury and property damage), a process to handle the imbalance in the target class was implemented (SMOTE). To build the best model, first a randomized hyper-parameter optimization process was implemented using the complete model features, followed by the implementation of a feature selection process to reduce the complexity and improve the execution time. The best classification scores were achieved using Gradient Boosted Trees using as discriminant threshold the one related with the maximum Matthews correlation coefficient. The proposed solution achieves a classification rate for accident frequency (collision/no-collision) of 79.2% using the

top-7 most important features; and considering accident severity (fatal, injury and property damage): 98.3% for fatal using top-3 most important features, 91.5% for injury using top-20 features, and 91.7% for property damage using top-20 most important features.

To improve data visualization and thus enable proactive intervention schemes by the Ottawa Police Service, a series of GIS map visualizations was implemented including heat maps for intersections, roads, neighborhoods and tiles.

8.2. Contributions

This research makes the following contributions to the field of accident modeling and prediction, especially to the areas of prediction of accident frequency (collision / no-collision) and accident severity (fatal, injury and property damage):

- Design of a collision prediction framework based on machine learning and GIS visualizations that uses a broad series of risk factors (including social events, geospatial information, location and mathematical features).
- Development of an innovative feature engineering process that includes geospatial (i.e. road segment, road intersection, neighborhood, tile), solar (solar azimuth and solar elevation), social events (i.e. statutory holidays), location and mathematical features (i.e. collision rate per road segment, road name, neighborhood, day of the year, etc.) that are demonstrated to contribute to the classification performance.
- Development of a process to generate the negative class (non-collision) that helps to create the dataset for classification.
- Development of a classification model using Gradient Boosted Trees with only top-7 most important features that outperforms the literature results in terms of accuracy for prediction of accident frequency (collision/no-collision).
- Development of a classification model using Gradient Boosted Trees for prediction of accident severity (fatal, injury and property damage) that

outperforms the related work in the literature for the *'fatal'* class and is in line with the results in the literature for the *'injury'* and *'non-injury'* classes.

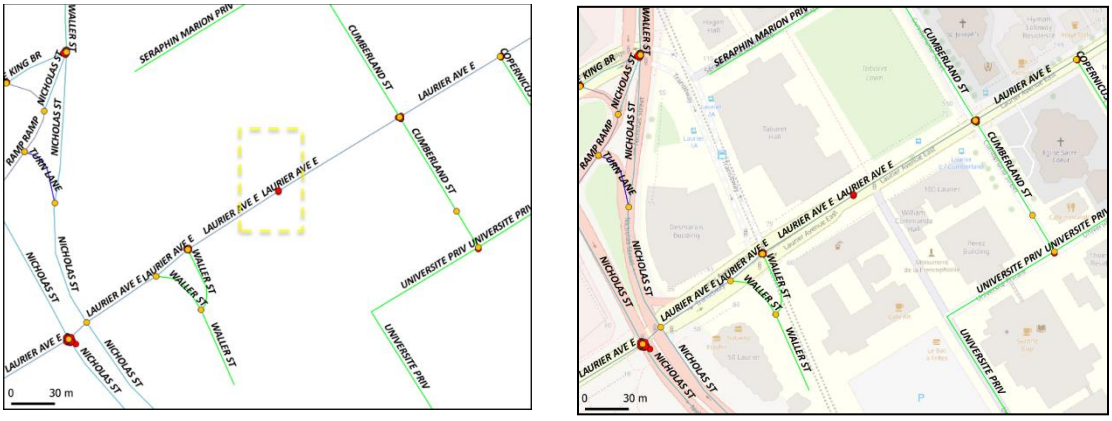
- Development of GIS heat maps based on several geospatial layers (i.e. road segments, road name, neighborhoods, tiles) to improve accident prediction visualization, including the visualization of the confusion matrix and of the classification scores at specific segments, road names, neighborhoods, tiles.
- Experimental demonstration of the fact that the best discrimination threshold for the explored dataset is the related with the maximum Matthew correlation coefficient.
- Experimental validation of the fact that social events do not contribute as much as the location, road, solar features and location to accident occurrence.

The research work in this thesis led to the publication of a conference paper [117].

8.3. Future Work

In section 5.1.3.2, the Open Data Roads Segments spatial data layer was described and used as part of the feature engineering process. The information that the layer provides is limited, i.e. it does not include the speed limit, the number of lines, the direction of the road and the existing traffic control elements (i.e. traffic lights, stop signs, yield signs, etc.). Also, there is lack of information that generates missing intersections, creating wrong information when the collision samples related with intersections are matched. An example of lack of information can be seen in Figure 8-1. In Figure 8-1a, a visualization using QGIS of the road segments (lines) and the road intersection identified (dots in orange) and the collision related with intersections (dots in red) is illustrated. We can notice that an intersection related collision (identified by a dashed yellow box) around *LAURIER AVE E* looks like is not really related with an intersection. In Figure 8-1b, the same information from the previous figure is used adding as a background layer the *Open Street Map* (OSM)

roads. We can notice that according with the OSM background layer, there is a missing road segment close to the collision. Using the OSM roads information instead of the Open Data Road Segments could therefore help to improve the results.



(a) (b)

Figure 8-1 (a) Open data roads segment geospatial layer (lines) with intersections (dots in orange) and collision related with intersection (red dots) and (b) same previous geospatial layers using OSM roads as background layer

In section 6.3, a series of machine learning algorithms was selected considering the literature findings. Future research could focus on the implementation and testing of additional classification algorithms, such as deep learning using the CUDA parallel processing platform on GPUs to achieve a reasonable execution time and thus allow the model to be retrained every 24 hours.

In section 5.3.2.6, the event related features were created. The approach used tagged the dataset by day of the year. This means that if we consider a statutory day, all the collision samples that match that day will be tagged. This tagging process is good for events that happened for a complete day, but is not realistic for events that happened in an area and at a specific time (i.e. sport events). Future research could develop an improved event tagging process that considers a neighborhood and a specific time frame.

In section 5.3.2.3, the non-collision samples were generated. As we are basing the algorithm on the generation of non-collision samples using a random procedure,

some biasing could be introduced in the system, such as the potential creation of data that is not coherent with the real geospatial attributes for the generated non-collision samples. Future research could perform a thorough investigation on these cases to study their impact on the classification performance. Also, different approaches for the selection of the attribute value (Sample2) in the non-collision samples algorithm could be considered using the distance, for example where far samples have preference over the closest ones.

In section 6.4, as part of the proposed traffic collision framework, a multiplicative correction factor was implemented in an attempt to improve the classifier performance. The results obtained after applying this process did not improve the performance due to the manner in which the non-collision dataset was created in this thesis (i.e. close variations of collision samples). As such, future research could be focused towards the development of an improved multiplicative corrective factor, for example using weighting schemes based on the frequency of features. Using this idea, the percentage of days per year when the surface is dry and the percentage of the time when the road is in daylight should be considered in order to avoid that the accident probability of those normal conditions will be higher than the others.

Appendices

Appendix A Motor Vehicle Collision Report (MVCR) Codes and Descriptions

In this section we will provide a more detail description from the MVCR Manual [94] of some of the feature's codes related with the features in the *main collision dataset:* (1) ACCIDENT_LOCATION, (2) CLASSIFICATION_OF_ACCIDENT, (3) ENVIRONMENT_CONDITION, (4) LIGHT, (5) TRAFFIC_CONTROL, (6) ROAD_SURFACE_CONDITION and (7) IMPACT_TYPE. The information provided here is a copy of the description in the manual.

A.1 ACCIDENT_LOCATION (MVCR 0301)

Table A-1 ACCIDENT_LOCATION (MVCR 0301) Codes and Descriptions

Code	Description
01 – Non-Intersection	There are no intersections, underpasses, overpasses, tunnels, bridges, private drives or railway crossings. The cause of the collision is not related to activity at a nearby intersection. For a definition of intersection see Code 03.
02 – Intersection Related	A collision is intersection related in any of the following situations: • a motor vehicle is moving toward an intersection, is within 100 m of the intersection and not turning into a private driveway • a motor vehicle is moving away from an intersection in a turning action • a motor vehicle is moving away from an intersection, is not turning and is within 100 m of the intersection
03 – At Intersection	The area within the outermost lines of the crosswalks. If there are no crosswalks, the intersection is the area within an imaginary line extending from the curb or highway boundary lines. **Note:** Right turn channels are not part of the intersection.
04 – At/near Private Drive	Private drives are all entries or exits which are not public roadways, e.g.: • entrance to plazas • schools • hospitals • homes • factories For use when the cause of the collision is related to a nearby private drive, ie., vehicle is turning into or out of a drive.
05 – At Railway Crossing	Collision occurred at a railway crossing. Includes railway tracks but not trolley or street car tracks located in the travelled lanes.
06 – Underpass or Tunnel	Collision occurred in a tunnel or on a roadway underneath a structure.
07 – Overpass or Bridge	Collision occurred on a bridge or on a roadway on a structure.
98 – Other	Collision occurred on a public roadway not described above.
08 – Trail	Collision occurred adjacent to trails or paths.

09 – Frozen Lake or River	Collision occurred on the surface of a frozen lake or river. This will often apply to off-road vehicles and snowmobiles but includes any vehicle operated on a frozen watercourse.
10 – Parking Lot	Collision occurred on private property designated for vehicular use. Includes driveways to parking lots and parking garages, but not residential drives.
99 – Other	Collision occurred off highway not described above, e.g.: • field • parkland • residential drive

A.2 CLASSIFICATION_OF_ACCIDENT (MVCR 0322)

Table A-2 CLASSIFICATION_OF_ACCIDENT (MVCR 0322) Codes and Descriptions

Code	Description
01 - Fatal injury	A collision which results in a fatality within 30 days of the date of the motor vehicle collision.
02 - Non-fatal injury	A collision which results in injury to one or more persons which does not result in a fatality within 30 days of the date of the collision. Injury is defined as any bodily harm visible or complained of resulting from the collision.
03 - P.D. only	A property damage only collision is a collision in which no injury occurs and total damage including load damage is in excess of $1000. Includes collisions involving motorized snow vehicles where total damages exceeds $400.
04 - Non-reportable	A collision in which no injury occurs and total damage including load damage is less than $1000 or $400 for motorized snow vehicles.
99 – Other	A collision is the intentional contact of a motor vehicle to oneself, others, property, buildings, etc. resulting in a death (includes homicides).

A.3 ENVIRONMENT_CONDITION (MVCR 0322)

Table A-3 ENVIRONMENT_CONDITION (MVCR 0303) Codes and Descriptions

Code	Description
00 – Unknown	Self-explanatory.
01 – Clear	Dull, overcast or bright conditions are recorded as clear provided no precipitation or airborne matter, obscures visibility.

02 – Rain	Self-explanatory.
03 – Snow	Self-explanatory.
04 - Freezing rain	Includes sleet and hail.
05 - Drifting snow	Snow drifting on or above roadway which obscures visibility of the roadway, road markings, traffic devices or highway fixtures.
06 - Strong wind	If wind was a contributing factor in the collision.
07 – Fog, mist, smoke, dust	Airborne matter obscuring visibility, whether of natural or industrial origin.
99 – Other	Environmental conditions not described above.

A.4 LIGHT (MVCR 0304)

Table A-4 LIGHT (MVCR 0304) Codes and Descriptions

Code	Description
00 – Unknown	Self-explanatory.
01 – Daylight	The light conditions which normally occur between one half hour after sunrise and one half hour before sunset.
02 - Daylight, artificial	The light conditions which normally occur between one half hour after sunrise and one half hour before sunset. Artificial illumination devices were functioning at the collision site.
03 – Dawn	The light conditions which normally occur between one half hour before and one half hour after sunrise.
04 - Dawn, artificial	The light conditions which normally occur between one half hour before and one half hour after sunrise. Artificial illumination devices were functioning at the collision site.
05 – Dusk	The light conditions which normally occur between one half hour before and one half hour after sunset.
06 - Dusk, artificial	The light conditions which normally occur between one half hour before and one half hour after sunset. Artificial illumination devices were functioning at the collision site.
07 – Dark	The light conditions which normally occur between one half hour after sunset and one half hour before sunrise.
08 - Dark, artificial	The light conditions which normally occur between one half hour after sunset and one half hour before sunrise. Artificial illumination devices were functioning at the collision site.
99 – Other	The collision occurred under light conditions not defined above. Includes non-normal occurrences such as a solar eclipse, major storm on location at which artificial illumination is not functioning e.g. tunnel

A.5 TRAFFIC_CONTROL (MVCR 0305)

Table A-5 TRAFFIC_CONTROL (MVCR 0305) Codes and Descriptions

Code	Description
01 - Traffic signal	Properly operating traffic signals or intersections under the manual control of a police officer are included. Includes railway warning lights without gates, pedestrian walk lights and lane control signals. Note: If traffic signals are not functioning correctly, enter under code 99. Auxiliary flashing lamps installed in conjunction with yield, stop or other regulatory and warning signs are not traffic signals.

02 - Stop sign	A regulatory sign requiring driver to halt.
03 - Yield sign	A regulatory sign requiring driver to reduce speed and give right of-way to approaching traffic before proceeding.
04 - Ped. crossover	School crosswalks or the normal painted line crosswalk delineation provided at signalized urban intersections are not crossovers. A pedestrian crossover may be located at an intersection or elsewhere and is designated by: • mandatory X marks in each lane of road way 30 meters in advance of crossover • mandatory overhead sign • mandatory signs prohibiting passing optional pedestrian-activated warning lights
05 – Police control	Traffic flow controlled by a police officer supersedes all other traffic controls. Does not include manual control of a traffic signal by a police officer.
06 – School guard	Control of traffic by a person other than a police officer to assist school children in crossing a road. Does not include child acting in a "safety patrol" capacity or a school guard at a location governed by a traffic signal or pedestrian crossover.
07 - School bus	A chrome yellow and black bus, van or mini bus used to transport children, or mentally challenged adults to or from a training centre. This vehicle functions as a traffic control device when it is stationary, has red signal lights flashing and stop arm activated.
08 - Traffic gate	A barrier to prevent vehicles from passing a point on the roadway. Normally, they are located at railway crossings, entrances to private parking lots and ramps used in conjunction with reversible lanes.
09 - Traffic controller	Any person directing or controlling traffic who is not a police officer or school crossing guard. Includes construction or railway flag persons and persons directing traffic when vehicles are backing from a driveway or public access, or persons at collision locations or other temporary locations.
10 – No control	None of the traffic control devices defined above was present at collision site.
11 – Roundabout	A roundabout is a circular intersection in which the traffic flow in only one direction. There are no traffic signals.
99 - Other	Traffic control device is not described above.

A.6 ROAD_SURFACE_CONDITION (MVCR 0310)

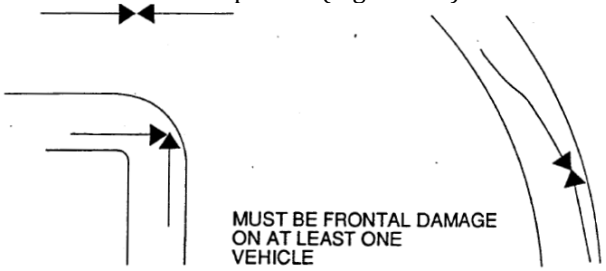
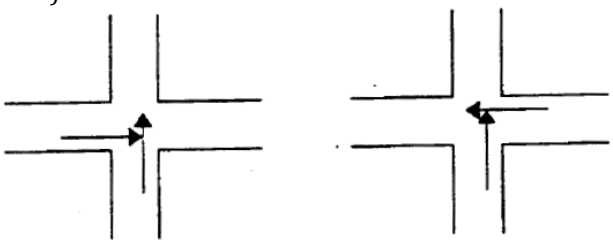
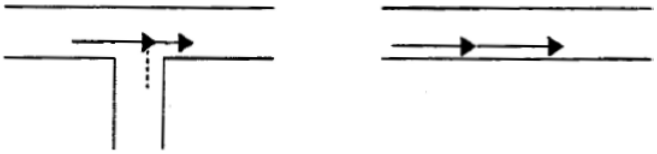
Table A-6 ROAD_SURFACE_CONDITION (MVCR 0310) Codes and Descriptions

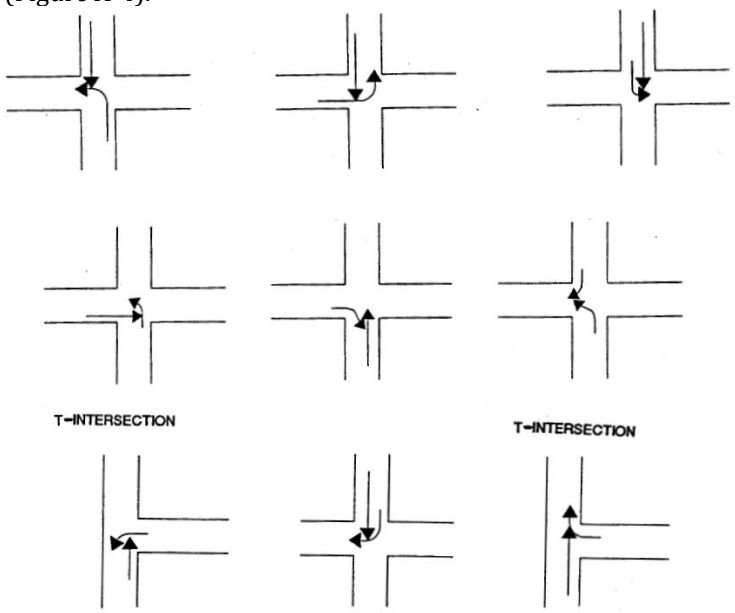
Code	Description
00 - Unknown	Self-explanatory.
01 – Dry	Road surface is free of any impediment to traction and tire adhesion.
02 – Wet	Includes flooding
03 - Loose snow	Recently fallen or blown snow covers a large portion of the road but not yet packed by traffic.
04 – Slush	A mixture of heavy, wet snow and water covers a substantial portion of the road.
05 - Packed snow	Snow has been packed by traffic after falling or drifting onto road
06 – Ice	Ice includes freezing rain or black ice on the road.

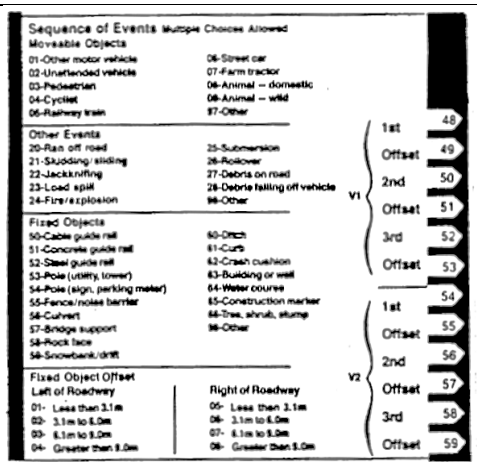
07 – Mud	Wet soil deposited on road by construction, off-road vehicles, farm equipment, animals, precipitation.
08 - Loose sand or gravel	Loose granular material on the road surface. This can be under wet or dry conditions.
09 - Spilled liquid	Liquid other than water on the road from vehicle spillage, e.g.: • diesel fuel • gasoline • oil • chemicals

A.7 INITIAL_IMPACT_TYPE (MVCR 0324)

Table A-7 IMPACT_TYPE (MVCR 0324) Codes and Descriptions

Code	Description
01 - Approaching	Initial direction of travel of each vehicle is opposite the other and at least 1 vehicle was impacted on the front. One vehicle may be stopped but not disabled or parked (Figure A-1).  **Figure A-1 IMPACT_TYPE Code 01 - Approaching (from [94])**
02 - Angle	Included are collisions which occur at intersections and/or private drives, where the initial directions of travel are approximately 90 to one another and neither vehicle is in the act of turning, e.g., Vl-EB, V2-NB. Normally a vehicle entering a roadway from a private drive is in the act of turning and this is not considered an angle impact (Figure A-2).  **Figure A-2 IMPACT_TYPE Code 02 - Angle (from [94])**
03 - Rear end	Collisions where vehicles are travelling in the same direction and the lead vehicle is struck in the rear (Figure A-3).  **Figure A-3 IMPACT_TYPE Code 03 – Rear end (from [94])**
04 - Sideswipe	Collisions involving side impacts where vehicles are travelling in

	the same or opposite direction. Vehicles which sideswipe while approaching, ie.: no frontal impacts are coded as sideswipes.
05 - Turning movement	Collisions in which vehicles are turning and impact location of one of the vehicles is on the side, e.g., V-1 is SB and V-2 is NB to WB (Figure A-4).  **Figure A-4 IMPACT_TYPE Code 05 – Turning movement (from [94])** Note: When two vehicles are travelling in the same direction and one of them is indicating a turning movement of either 04, 05 or 06. The initial impact will always be 05. Not a rear end (03).
06 - SMV unattended vehicle	Single motor vehicles (SMV) collisions occur when a vehicle strikes a vehicle unattended by its driver. Includes parked, stopped, disabled, abandoned and runaway vehicles, provided it was not under the care and control of a driver. Does not include vehicles stopped for traffic, standing while loading, unloading passengers, or cargo (Figure A-5).  **Figure A-5 IMPACT_TYPE Code 06 – SMV unattended vehicle (from [94])**
07 - SMV other	Single Motor Vehicle (SMV) initially collides with a fixed object, pedestrian or animal. Includes occurrences of other Events provided in the Sequence of Events (Figure A-6).

	 **Figure A-6 IMPACT_TYPE Code 07 – SMV other (from [94])**
99 – Other	Impact type not described above.

Appendix B Software Tools and Libraries

This section includes a brief introduction of each of the software tools and libraries used in this thesis.

- **Tableau Desktop** [85] is a data visualization software developed by *Tableau Software* that allows to connect to a spreadsheet or file and create data visualizations. Using a simple Graphic User Interface (GUI) is possible to create different data visualization using drag and drop that can be also exported to the web.
- **RStudio** [82] is a free and open-source integrated development environment (IDE) for *R* [81], a programming language for statistical computing and graphics. It includes a console, a syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. The number of libraries available for *R* makes it an important tool for data mining tasks.
- **RapidMiner Studio** [86] is a data science software platform developed by the company of the same name that provides an integrated environment for data preparation, machine learning, deep learning, text mining, and predictive analytics. A visual workflow designer and a series of available libraries make easy the creation of machine learning models.
- **ESRI ArcGIS Pro** [87] is a Geographic Information System (GIS) developed by *ESRI* that allows to work with geospatial data. It provides a series of integrated tools to perform geospatial analysis and includes a Python library that can be used for developing script.
- **QGIS** [90] is a free and open source GIS that allows to work with geospatial data. QGIS is an official project of the Open Source Geospatial Foundation. It provides a series of integrated tools to perform geospatial analysis and includes a Python library that can be used for developing script. It can be considered as an open source alternative to ArcGIS Pro.

- **Pandas** [118] is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
- **GeoPandas** [97], [119] is an open source project to make working with geospatial data in Python easier. GeoPandas extends the datatypes used by *Pandas* to allow spatial operations on geometric types. Geometric operations are performed by *Shapely*. GeoPandas further depends on *Fiona* for file access and *Descartes* and *Matplotlib* for plotting.
- **Shapely** [120] is a BSD-licensed Python package for manipulation and analysis of planar geometric objects.
- **Fiona** [121] is an open source package that can read and write real-world data using multi-layered GIS formats and zipped virtual file systems and integrates readily with other Python GIS packages such as *Rtree*, and *Shapely*.
- **Rtree** [122] is an open source package that provides a number of advanced spatial indexing features.
- **Descartes** [123] allows to use *Shapely* or GeoJSON-like geometric objects as *Matplotlib* paths and patches.
- **Matplotlib** [124], [125] is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. *Matplotlib* can be used in Python scripts, the Jupyter notebook, web application servers, and graphical user interface toolkits.
- **Pysolar** [99] is a collection of Python libraries for simulating the irradiation of any point on earth by the sun.
- **Jupyter Notebook** [88], [126] is a free and open source web application that allows to create and share documents that contain live code, equations, visualization and narrative text.
- **Conda** [89] is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly

installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on a local computer.

- **Scikit-learn** [100] is a free and open source Python library that provides simple and efficient tools for data mining and data analysis, including tools for classification, regression, clustering, dimensionality reduction, model selection and preprocessing.
- **Imbalance-learn** [116] is a free and open source Python library that provides methods to handle the imbalance problem including under-sampling, over-sampling methods and SMOTE.

References

[1] World Health Organization, “Global Status Report on road safety 2018”. Available online, accesed 09-August-2019. https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/

[2] World Health Organization, “World report on road traffic injury prevention - Main messages”, 2004. Available online, accesed 09-August-2019. https://www.who.int/violence_injury_prevention/publications/road_traffic/world_report/main_messages_en.pdf?ua=1

[3] Transport Canada, “Canadian Motor Vehicle Traffic Collision Statistics:2017”. Available online, accesed 09-August-2019. https://www.tc.gc.ca/eng/motorvehiclesafety/canadian-motor-vehicle-traffic-collision-statistics-2017.html

[4] Ontario Ministry of Transportation, “Ontario Road Safety Annual Reports (ORSAR)”. Available online, accesed 09-August-2019. http://www.mto.gov.on.ca/english/publications/ontario-road-safety-annual-report.shtml

[5] Parachute, “The cost of injury in Canada”, June 2015. Available online, accesed 09-August-2019. http://www.parachutecanada.org/downloads/research/Cost_of_Injury-2015.pdf

[6] Canadian Association of Chiefs of Police, “Canada Road Safety Week – The Facts and Stats”, May 2018. Available online, accesed 09-August-2019. https://www.cacp.ca/index.html?asst_id=1626

[7] Saskatchewan Government Insurance (SGI), “2017 Saskatchewan Traffic Accident Facts”, 2017. Available online, accesed 09-August-2019. https://www.sgi.sk.ca/documents/625510/627017/TAIS+2017+Annual+Report/6112c88a-c088-44e6-9aa5-415450cc0ebf

[8] P. Moiseets, and Y. Tanaka, "Road Danger Estimation for Winter Road Management", 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 304-311, 2014.

[9] S. Afrin, "The influence of winter weather on high-crash days in Southern Ontario", Master thesis, University of Waterloo, 2013.

[10] R. Yu, and M. Abdel-Aty, "Analyzing crash injury severity for a mountainous freeway incorporating real-time traffic and weather data", Safety Science 63 (2014) 50–56.

[11] C. Xu, W. Wang, and P. Liu, "Identifying crash-prone traffic conditions under different weather on freeways", Journal of Safety Research 46 (2013) 135–144.

[12] R. Tian, Z. Yang, and M. Zhang, "Method of Road Traffic Accidents Causes Analysis Based on Data Mining", International Conference on Computational Intelligence and Software Engineering, pp. 1-4, 2010.

[13] A. Gregoriades, "Road Safety Assessment using Bayesian Belief Networks and Agent-based Simulation", 2007 IEEE International Conference on Systems, Man and Cybernetics, pp.615-620.

[14] World Health Organization, "World report on road traffic injury prevention", 2004. Available online, accesed 09-August-2019. https://apps.who.int/iris/bitstream/handle/10665/42871/9241562609.pdf?sequence=1

[15] Z. Yan, X. Wang, and L. Du, "Design Method of Highway Traffic Safety Analysis Model", 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), pp. 151-154, 2011.

[16] B. Aneurin, "Collision-prone: When are you statistically most likely to get in an accident?", Wheel.ca, 2014. Available online, accesed 09-August-2019. http://www.wheels.ca/news/49355/

[17] CityNews Staff, "Car accidents on the rise in Canada, Toronto: study", 2015. Available online, accesed 09-August-2019. http://www.citynews.ca/2015/11/26/car-accidents-on-the-rise-in-canada-toronto-study/

[18] InsuranceHotline.com Staff, "5 Times When You're More Likely to Get in a Car Accident", 2018. Available online, accesed 09-August-2019. https://www.insurancehotline.com/five-times-when-youre-more-likely-to-get-in-a-car-accident

[19] Good Hans Advice Team, "Safe Driving Study 2018: Ten Years of Road Safety Awareness", 2018. Available online, accesed 09-August-2019. https://blog.allstate.ca/safe-driving-study-2018/

[20] A. Weight-Bienzle, "2016 Safe Driving Study: The Safest Days on the Road[Infographic]", 2016. Available online, accesed 09-August-2019. https://blog.allstate.ca/2016-safe-driving-study-the-safest-days-on-the-road-infographic/

[21] HG.org Legal Resources Staff, "Fatal Car Accident Statistics", 2015. Available online, accesed 09-August-2019. http://www.hg.org/article.asp?id=29836

[22] Forbes Staff, "Car Accident Times", 2009. Available online, accesed 09-August-2019. http://www.forbes.com/2009/01/21/car-accident-times-forbeslife_cx_he_0121driving_slide_2.html?thisspeed=25000

[23] M. Presta, "June Most Deadliest Month On Ontario Roads", Young Drivers Blog, 2015. Available online, accesed 09-August-2019. https://www.yd.com/blog/june-most-deadliest-month-on-ontario-roads/

[24] Transport Canada, "Canadian Motor Vehicle Traffic Collision Statistics: 2017", 2017. Available online, accesed 09-August-2019. https://www.tc.gc.ca/eng/motorvehiclesafety/canadian-motor-vehicle-traffic-collision-statistics-2017.html

[25] Z. Li, I. Kolmanovsky, E. Atkins, J. Lu, D. Filev, and J. Michelini, "Road Risk Modeling and Cloud-Aided Safety-Based Route Planning", IEEE Transactions on Cybernetics, pp. 2473-2483, 2016.

[26] Y. Lv , Z. Haixia, Z. Xing-lin, L. Ming and L. Jie, "Research on Accident Prediction of Intersection and Identification Method of Prominent Accident Form Based on Back Propagation Neural Network", 2010 Int. Conference on Computer Application and System Modeling (ICCASM), pp. VI-434-VI-438, 2010.

[27] P. Liu, S.-H. Chen, and M.-D. Yang, "Study of Signalized Intersection Crashes Using Artificial Intelligence Methods", 2008 Mexican International Conference on Artificial Intelegence (MICAI), pp. 987–997, 2008.

[28] W. Huiying, L. Jun, C. Xiaolong, and G. Xiaohui, "Real-time Highway Accident Prediction Based on Grey Relation Entropy Analysis and Probabilistic Neural Network", 2011 International Conference on Electric Technology and Civil Engineering (ICETCE), pp. 1420-1423.

[29] J.-W. Hwang, Y.-S. Lee, and S.-B. Cho, "Hierarchical Probabilistic Network-based System for Traffic Accident Detection at Intersections", 2010 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, pp. 211-216, 2010.

[30] S. Li, and D. Zhao, "Prediction of Road Traffic Accidents Loss Using Improved Wavelet Neural Network", IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering, pp. 1526-1529, 2002.

[31] Y. Lv, S. Tang, H. Zhao, and S. Li, "Real-time Highway Accident Prediction based on Support Vector Machines", Chinese Control and Decision Conference, pp. 4403-4407, 2009.

[32] R. Yu, and M. Abdel-Aty, "Utilizing support vector machine in real-time crash risk evaluation", Accident Analysis and Prevention 51 (2013) 252– 259.

[33] N. Dong, H. Huang, and L. Zheng, "Support vector machine in crash prediction at the level of traffic analysis zones: Assessing the spatial proximity effects", Accident Analysis and Prevention 82 (2015) 192–198.

[34] R. Gang, and Z. Zhuping, "Traffic safety forecasting method by particle swarm optimization and support vector machine", Expert Systems with Applications, 38 (2011) 10420–10424.

[35] K. Polat, and S.S. Durduran, "Automatic determination of traffic accidents based on KMC-based attribute weighting", Neural Computing and Application, 21, pp. 1271–1279, 2012.

[36] M. Hosseinpour, A. S. Yahaya, S. M. Ghadiri, and J. Prasetijo, "Application of Adaptive Neuro-Fuzzy Inference System for Road Accident Prediction", KSCE Journal of Civil Engineering (2013) 17(7), pp. 1761-1772, 2013.

[37] X. Zhu, “Application of Composite Grey BP Neural Network Forecasting Model to Motor Vehicle Fatality Risk”, 2010 Second International Conference on Computer Modeling and Simulation, pp. 236-240, 2010.

[38] X. Xu, B. Chen, and F. Gan, “Traffic Safety Evaluations Based on Grey Systems Theory and Neural Network”, 2009 World Congress on Computer Science and Information Engineering, pp. 603-607, 2009.

[39] Q. Wuyong, D. Yaoguo, M. Sen, and L. Xuemei, “The Intelligent Optimization of GM(1,1) Power Model and its Application in the Forecast of Traffic Accident”, IEEE International Conference on Grey Systems and Intelligent Services (GSIS), pp. 385-389, 2011.

[40] X. Binglei, H. Zheng, and M. Hongwei, “Fuzzy-Logic-Based Traffic Incident Detection Algorithm for Freeway”, Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, pp. 1254-1259, 2008.

[41] H. Wang, L. Zheng, and X. Meng, “Traffic Accidents Prediction Model Based on Fuzzy Logic”, Advances in Information Technology and Education, pp. 101–108, 2011.

[42] Y. Luo, and S. Zhang, “The Fuzzy Regression Prediction of the City Road Traffic Accident”, 2009 International Conference on Industrial Mechatronics and Automation, pp.121-124, 2009.

[43] A.-Z. Li, and X.-H. Song, “Traffic Accident Characteristics Analysis Based on Fuzzy Clustering”, 2012 IEEE Symposium on Electrical & Electronics Engineering (EEESYM), pp.468-470.

[44] P. Kromer, T. Beshah, D. Ejigu, V. Snasel, J. Platos, and A. Abraham, “Mining Traffic Accident Features by Evolutionary Fuzzy Rules”, 2013 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS), pp. 38-43, 2013.

[45] T. Beshah, D. Ejigu, P.Kromer, V. Snasel, J. Plato, A. Abraham, “Learning the Classification of Traffic Accident Type”, 2012 Fourth International Conference on Intelligent Networking and Collaborative Systems, pp.463-468, 2012.

[46] H. K. Moghaddam, and X. Wang, "Vehicle Accident Severity Rules Mining Using Fuzzy Granular Decision Tree", International Conference on Rough Sets and Current Trends in Computing, pp. 280–287, 2014.

[47] X.-F. Zhang and L. Fan, "A Decision Tree Approach for Traffic Accident Analysis of Saskatchewan Highways", 2013 26th IEEE Canadian Conference Electrical And Computer Engineering, pp. 1-4, 2013.

[48] T. Beshah and S. Hill, "Mining Road Traffic Accident Data to Improve Safety: Role of Road-related Factors on Accident Severity in Ethiopia", 2010 AAAI Spring Symposium Series, 2010.

[49] G. Khan, A. R. Bill, and D. A. Noyce, "Exploring the feasibility of classification trees versus ordinal discrete choice models for analyzing crash severity", Transportation Research Part C: Emerging Technologies 50, pp. 86–96, 2015.

[50] T. Beshah, D. Ejigu, A. Abraham, V. Snasel, and P. Kromer, "Knowledge Discovery from Road Traffic Accident Data in Ethiopia: Data Quality, Ensembling and Trend Analysis for Improving Road Safety", Neural Network World 22(3), pp. 215, 2012.

[51] L.-Y. Chang, and H.-W. Wang, "Analysis of traffic injury severity: An application of non-parametric classification tree techniques", Accident Analysis and Prevention 38, pp. 1019–1027, 2006.

[52] D. Saha, P. Alluri, and A. Gan, "Prioritizing Highway Safety Manual's crash prediction variables using boosted regression trees", Accident Analysis and Prevention 79, pp. 133–144, 2015.

[53] D. Wilson, "Using Machine Learning to Predict Car Accident Risk". Available online, accesed 09-August-2019. https://medium.com/geoai/using-machine-learning-topredict-car-accident-risk-4d92c91a7d57

[54] Q. Guo, X. Pei, D. Yao, and S. Wong, "Role of street patterns in zone-based traffic safety analysis", Journal of Central South University, pp. 2416–2422, 2015.

[55] T. Sayed, and P. de Leur, "Collision Prediction Models for British Columbia", Technical Report for BC Ministry of Transportation & Infrastructure, 2008.

[56] Z. Ma, C. Shao, H. Yue, and S. Ma, “Analysis of the Logistic Model for Accident Severity on Urban Road Environment”, 2009 IEEE Intelligent Vehicles Symposium, pp. 983-987, 2009.

[57] J. Pahukula, S. Hernandez, and A. Unnikrishnan, “A time of day analysis of crashes involving large trucks in urban areas”, Accident Analysis and Prevention 75, pp. 155–163, 2015.

[58] S. Park, K. Jang, S. H. Park, D.-K. Kim, and K. S. Chon, “Analysis of Injury Severity in Traffic Crashes: A Case Study of Korean Expressways”, KSCE Journal of Civil Engineering (2012), pp. 16(7):1280-1288, 2012.

[59] E. Moons, T. Brijs, and G. Wets, “Improving Moran’s Index to Identify Hot Spots in Traffic Safety”, Geocomputation and Urban Planning, pp. 117–132, 2009.

[60] C. Xu, P. Liu, W. Wang, and X. Jiang, “Development of a Crash Risk Index to Identify Real Time Crash Risks on Freeways”, KSCE Journal of Civil Engineering (2013) 17(7), pp. 1788-1797, 2013.

[61] X. Zhan, H.M.A. Aziz, and S. V. Ukkusuri, “An efficient parallel sampling technique for Multivariate Poisson-Lognormal model: Analysis with two crash count datasets”, Analytic Methods in Accident Research 8, pp. 45–60, 2015.

[62] X. Hongguo, Z. Huiyong, and Z. Fang, “Bayesian Network-Based Road Traffic Accident Causality Analysis”, 2010 WASE International Conference on Information Engineering, pp. 413-417, 2010.

[63] E. Paikari, M. Moshirpour, R. Alhajj, and B.H. Far, “Data Integration and Clustering for Real Time Crash Prediction”, IEEE IRI 2014, pp. 537-544, San Francisco, California, USA, 2014.

[64] X. Qin, J. N. Ivan, N. Ravishanker, J. Liu, and D. Tepas, “Bayesian estimation of hourly exposure functions by crash type and time of day”, Accident Analysis and Prevention 38, pp. 1071–1080, 2006.

[65] R. Marukatat, “Structure-Based Rule Selection Framework for Association Rule Mining of Traffic Accident Data”, 2006 International Conference on Computational and Information Science, pp. 231–239, 2006.

[66] Y. Lv, S. Tang, and H. Zhao, “Real-time Highway Traffic Accident Prediction Based on the k-Nearest Neighbor Method”, 2009 International Conference on Measuring Technology and Mechatronics Automation, pp. 547-550, 2009.

[67] J. Wang, and X. Wang, “An Ontology-Based Traffic Accident Risk Mapping Framework”, International Symposium on Spatial and Temporal Databases, pp. 21–38, 2011.

[68] R. Jagannathan, S. Petrovic, G. Powell, and M. Roberts, “Predicting Road Accidents Based on Current and Historical Spatio-temporal Traffic Flow Data”, International Conference on Computational Logistics, pp. 83-79, 2013.

[69] K. Li, and N. M. Waters, “Transportation Networks, Case-Based Reasoning and Traffic Collision Analysis: A Methodology for the 21st Century”, Methods and Models in Transport and Telecommunications, pp. 63-92, 2005.

[70] J. Wua, M. Abdel-Aty, R. Yu, and Z. Gao, “A novel visible network approach for freeway crash analysis”, Transportation Research Part C 36 (2013), pp. 72–82, 2013.

[71] D.N. Reshef, Y.A. Reshef, H.K. Finucane, S.R. Grossman, G. McVean, P.J. Turnbaugh, E.S. Lander, M. Mitzenmacher, and P.C. Sabeti, “Detecting novel associations in large data sets”, Science 334, pp. 1518–1524, 2011.

[72] L. Breiman, “Some Infinity Theory for Predictor Ensembles”, Techical Report 579, Statistics Department UCB, 2000.

[73] R. Wirth, and J. Hipp, “CRISP-DM: Towards a standard process model for data mining”, Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining, pp. 29–39, 2000.

[74] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, “CRISP-DM 1.0 Step-by-step data mining guide”, SPSS Inc. CRISPMWP-1104, 2000.

[75] C. Shearer, "The CRISP-DM model: the new blueprint for data mining", Journal of data warehousing 5.4, pp. 13-22, 2000.

[76] FlatIcon Development Team, “The largest database of free icons available in PNG, SVG, EPS, PSD and BASE 64 formats”. Available online, accesed 09-August-2019. http://www.flaticon.com

[77] Open Data Ottawa Staff, “Ottawa’s Open Data Catalogue”. Available online, accesed 09-August-2019. http://data.ottawa.ca

[78] Open Data Ottawa Staff, “Tabular Transportation Collision Data”. Available online, accesed 09-August-2019. http://data.ottawa.ca/dataset/collisiondata2017

[79] Open Data Ottawa Staff, “Road Segments”. Available online, accesed 09-August-2019. http://data.ottawa.ca/dataset/roads

[80] Open Data Ottawa Staff, “ONS Neighborhood Boundaries”. Available online, accesed 09-August-2019. http://data.ottawa.ca/dataset/neighbourhoodboundaries

[81] R Project Development Team, “The R Project for Statistical Computing”. Available online, accesed 09-August-2019. http://www.r-project.org

[82] RStudio Development Team, “RStudio”. Available online, accesed 09-August-2019. http://www.rstudio.com/products/rstudio

[83] Microsoft Excel Development Team, “Quick start: Create a Macro”. Available online, accesed 09-August-2019. https://support.office.com/en-us/article/quick-start-create-a-macro-741130ca-080d-49f5-9471-1e5fb3d581a8

[84] Microsoft Windows Development Team, “Windows Commands”. Available online, accesed 09-August-2019. https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands

[85] Tableau Development Team, “Tableau Desktop”. Available online, accesed 09-August-2019. https://www.tableau.com/products/desktop

[86] RapidMiner Development Team, “RapidMiner Studio”. Available online, accesed 09-August-2019. https://rapidminer.com/products/studio/

[87] ArcGIS Pro Development Team, “ArcGIS PRO”. Available online, accesed 09-August-2019. https://pro.arcgis.com

[88] Jupyter Notebook Development Team, “Jupyter Notebook”. Available online, accesed 09-August-2019. https://jupyter.org/

[89] Conda Development Team, "Conda". Available online, accesed 09-August-2019. https://conda.io

[90] QGIS Development Team, "QGIS Geographic Information System. Open Source Geospatial Foundation Project". Available online, accesed 09-August-2019. https://qgis.org

[91] OpenStreetMap Development Team, "OpenStreetMap". Available online, accesed 09-August-2019. http://openstreetmaps.org/

[92] Ottawa Tourism Development Team, "Ottawa Tourism, Events". Available online, accesed 09-August-2019. http://www.ottawatourism.ca/events.

[93] Argis Development Team, "Choose basemap". Available online, accesed 09-August-2019. https://doc.arcgis.com/en/arcgis-online/create-maps/choose-basemap.htm

[94] Ministry of Transportation Ontario (MTO), "Motor Vehicle Collision Report (MVCR) Manual", 2011.

[95] Carleton University, "Carleton University Calendar". Available online, accesed 09-August-2019. http://calendar.carleton.ca.

[96] N.P. Bobbili, and A.-M. Cretu, "Adaptive Weighting with SMOTE for Learning from Imbalanced Datasets: A Case Study for Traffic Offence Prediction", IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications, pp. 40-45, Ottawa, Canada, 2018.

[97] GeoPandas Development Team, "A Open Source project to make working with geospatial data in Python easier". Available online, accesed 09-August-2019. http://geopandas.readthedocs.io

[98] IBM, "IBM Informix R-Tree Index User's Guide", Available online, accesed 09-August-2019. https://www.ibm.com/support/knowledgecenter/en/SSGU8G_11.50.0/com.ibm.rtree.doc/sii-overview-27706.htm

[99] Pysolar Development Team, "A collection of Open Source Python libraries for simulating the irradiation of any point on earth by the sun". Available online, accesed 09-August-2019. https://pysolar.readthedocs.io

[100] F. Pedregosa, *et al.*, "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research, pp. 2825-2830, 2011. Available online, accesed 09-August-2019. https://scikit-learn.org

[101] Wikipedia Staff, "Youden's J statistic". Available online, accesed 09-August-2019. https://en.wikipedia.org/wiki/Youden%27s_J_statistic

[102] Wikipedia Staff, "Matthews correlation coefficient", Available online, accesed 09-August-2019. https://en.wikipedia.org/wiki/Matthews_correlation_coefficient

[103] W.Y. Loh, "Classification and regression trees", Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1), 14-23, 2011.

[104] G. Machado, M.R. Mendoza, and L.G. Corbellini, "What variables are important in predicting bovine viral diarrhea virus? A random forest approach", Veterinary research, 46(1), pp. 85, 2015.

[105] G. Biau, L. Devroye, and G. Lugosi, "Consistency of random forests and other averaging classifiers", Journal of Machine Learning Research, pp. 2015-2033, 2008.

[106] B. Marsh, "Multivariate Analysis of the Vector Boson Fusion Higgs Boson", Georg-August-Universität Göttingen · II. Physical Institute · Higgs Group Germany · Göttingen, Lower Saxony, 2016.

[107] F. Yoav, and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Journal of computer and system sciences 55.1, pp. 119-139, 1997.

[108] J.H. Friedman, "Greedy function approximation: a gradient boosting machine", Annals of statistics, pp. 1189-1232, 2001.

[109] A. Rogozhnikov, "Gradient Boosting explained [demonstration]". Available online, accesed 09-August-2019. http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

[110] S. Manna, S. Biswas, R. Kundu, S. Rakshit, P. Gupta, and S. Barman, “A statistical approach to predict flight delay using gradient boosted decision tree”, 2017 International Conference on Computational Intelligence in Data Science (ICCIDS), pp. 1–5, 2017.

[111] XGBoost Development Team, “A Open Source software optimized distributed gradient boosting library in Python”. Available online, accesed 09-August-2019. https://xgboost.readthedocs.io

[112] N. Ye, “Data mining: theories, algorithms, and examples”, CRC press, 2013.

[113] Ch. C. Aggarwal, “Neural networks and deep learning”, Springer, 2018.

[114] M. Schubach, M. Re, P.N. Robinson, and G. Valentini, "Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants", Scientific reports 7, no. 1, pp. 2959, 2017.

[115] N. Chawla, K.W. Bowyer, L.O. Hall, and W. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique", Journal of artificial intelligence, research16, pp. 321-357, 2007.

[116] Imbalanced-Learn Development Team, “Imbalance Learn Documentation”. Available online, accesed 09-August-2019. https://imbalanced-learn.readthedocs.io/en/stable/index.html

[117] E. Reveron, and A.M. Cretu, "A Framework for Collision Prediction Using Historical Accident Information and Real-time Sensor Data: A Case Study for the City of Ottawa”, IEEE International Symposium on Robotic and Sensors Environments, pp. 143-149, 2019, Ottawa, Canada.

[118] Pandas Development Team, “A Open Source Python Data Analysis Library”. Available online, accesed 09-August-2019. http://pandas.pydata.org

[119] K. Jordahl, "GeoPandas: Python tools for geographic data", Available online, accesed 09-August-2019. https://github. com/geopandas/geopandas.

[120] Shapely Development Team, “Shapely Documentation”. Available online, accesed 09-August-2019. http://shapely.readthedocs.io

[121] Fiona Development Team, “Fiona Project Description”. Available online, accesed 09-August-2019. http://pypi.org/project/Fiona

[122] RTree Development Team, "RTree Project Description". Available online, accesed 09-August-2019. https://pypi.org/project/Rtree

[123] Descartes Development Team, "Descartes Project Description". Available online, accesed 09-August-2019. http://pypi.org/project/Descartes

[124] Matplotlib Development Team, "Matplotlib Project Description". Available online, accesed 09-August-2019. http://matplotlib.org

[125] J. Hunter, *et al.*, "Matplotlib: A 2D graphics environment", Computing in Science and Engineering 9.3, pp. 90, 2007.

[126] F. Pérez, B.E. Granger, "IPython: A System for Interactive Scientific Computing", Computing in Science and Engineering, vol. 9, no. 3, pp. 21-29, 2007, Available online, accesed 09-August-2019. http://ipython.org