Artificial Intelligence driven optimal route planning for urban transit.

by

Ogechukwu (Leroy) Patrick Ngene

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

Master of Information Technology

in

Digital Media with Specialization in Data Science

Carleton University
Ottawa, Ontario

Abstract

This thesis proposes an ensemble approach by combining machine learning and deep learning techniques for predicting taxi trips arrival times and predicting shared rides. This research work considers the capabilities of machine learning and deep learning models to predict arrival times and shared rides as an essential aspect of developing an intelligent transportation system offering optimized and efficient ride-sharing schemes. The dataset was made publicly available by the New York City transport, and limousine company (TLC) consisting of roughly 1 million records of New York City green taxi trip data for January, February 2017 and January to June 2019. We compare the results of our ensemble approach and observe that the Linear regression model and Convolutional Neural Network (CNN) perform admirably for arrival time prediction while the Capsule Network (CapsNet) provides the best results for ride-sharing prediction.

Keywords—Arrival time prediction; Ride-sharing prediction, Machine learning; Deep learning, Ensemble techniques.

Acknowledgements

Firstly, we give thanks to God almighty the creator for blessings, support and eternal guidance throughout the execution of this research work, I thank and dedicate this research work to my mother for her unwavering strength and support throughout my time as a graduate student at Carleton University, her daily prayers continue to strengthen me and guide me towards the path of light and humility. I thank every member of my family for the text messages, phone calls, show of support and love, this has been very instrumental in ensuring I complete this journey, I thank my partner for her tenderness, care and loving support throughout the duration of this project, you have been ever present from start to finish.

I would like to give special thanks to my thesis supervisor Professor M. Omair Shafiq, for his technical guidance and provision of the technical expertise required to execute a project of this scale, the weekly meetings throughout the duration of this research work ensured that I stayed on track and my numerous questions were answered in a timely manner. I appreciate the countless and immeasurable advice you offered throughout the execution process of this thesis work, the outlines and methods applied to ensure the implementation process is a success cannot be understated. I believe that the approach you employed made the entire work very enjoyable, and the tasks executed were appropriately spaced to ensure a high level of productivity and focus throughout the thesis research year.

I also thank the faculty members of the information technology department at Carleton University for providing an excellent foundation for me through excellent course work via individual and group projects, this served as a fantastic pedestal in preparing me in the best way possible to carry out my thesis research work.

Table of Contents

List of Acronyms

1) ML: Machine Learning
2) DL: Deep Learning
3) LR: Linear Regression
4) RF: Random Forest
5) ANN: Artificial Neural Networks
6) CNN: Convolutional Neural Networks
7) LSTM: Long short-term Memory
8) RNN: Recurrent Neural Network.
9) CapsNet: Capsule Networks.
10) DT: Decision Trees.
11) ITS: Intelligent Transportation Systems.
12) ATIS: Advanced Traveler Information Systems.
13) AV: Autonomous Vehicles.
14) CV: Connected Vehicles.
15) EM: Ensemble Model.
16) EB: Ensemble Boosting
17) MSE: Mean Squared Error
18) MAE: Mean Absolute Error
19) ABRM: activity-based ride matching
20) STaRS: Simulating Taxi Ride-sharing
21) POI: Point of Interest
22) MMTP: Multi-modal trip planning
23) XAR: Xhare a Ride
24) HDFS: Hadoop Distributed File System.
25) ARIMA: Autoregressive integrated moving average
26) XGBoost: Extreme Gradient Boosting.
27) AdaBoost: Adaptive Boosting.
28) Bagging: Bootstrap Aggregating.
29) TP: True Positive
30) FP: False Positive
31) FN: False Negative
32) TN: True Negative

List of Figures

List of Tables

Chapter 1: Introduction and Background

The capability to conveniently move from one location to another remains a key aspect of living in today's urban cities hence the need to have a dependable transportation system cannot be understated. Many urban cities have delved into the concept of intelligent transportation systems which offer a new approach to creating transportation systems suitable to the demands of everyday users. The year 1984 saw the introduction of an automated traffic surveillance and control system in Los Angeles, USA along with closed-circuit TV (CCTV) and coordinated signal timing data as shown by the authors in (Auer et. al., 2016). The implementation of such system created an avenue where large volumes of transportation data can be captured, managed and analyzed to extract valuable informative trends and patterns.

This research work involves utilizing modern artificial intelligence (AI) tools, machine learning and deep learning techniques that can contribute to the development of a reliable and sophisticated transportation system, we execute an extensive study of the related works in developing intelligent transportation systems (ITS) and advance traveler information systems (ATIS) to identify possible trends and also acquire more information and understanding on the industrial and technological standards for building such solutions. We are excited at the amount of transportation data openly available which creates a healthy environment for executing research in various modes of transportation and identifying key informative patterns in such large datasets, it is also important to highlight that the technological resources available in developed and under-developed cities vary therefore highlighting the need for any proposed solution to be feasible and scalable for possible implementation in any transportation environment.

A very large number of the related works executed simulations as a form of experiments and although the prospect of running simulations seemed exciting, the resources currently available

meant we focused on executing machine learning and deep learning experiments to identify possible solutions to our research questions. Our literature review led us to identify trip arrival time prediction and the potential of sharing rides as the key factors in developing an efficient ride-sharing scheme therefore for this research work, we proceed to develop deep learning and machine learning models to predict taxi trip arrival times and predict shared rides. We believe that the information from such a sophisticated system can help an end-user execute optimal route planning activities by identifying the preferred ride and route options based on real-time information on travel time and ride sharing options. The capability of our ride sharing prediction model to predict rides with the potential to be shared in real-time, could provide important information with regards to identifying the key locations where rides are mostly shared for proper vehicle allocation. This could ultimately lead to better vehicle efficiency, fewer traffic congestions, carbon emissions and avoiding road accidents.

1.1 Arrival Time Prediction

For this research work, arrival time prediction refers to the utilization of artificial intelligence and machine learning tools to execute predictions on the duration of taxi trip times. Arrival time prediction is a key aspect of building intelligent transportation systems, this ensures that in a real-time scenario, an end-user can make informed transportation decisions based on real-time traffic information e.g. real-time information of travel time from home to work. We aim to contribute to the development of ride-sharing induced intelligent transportation systems capable of accurately predicting taxi trip arrival times as well as accurately predicting shared rides.

We highlight that taxi trip arrival times are generally affected by several factors that can be classified as either internal or external factors. Some internal factors affecting a taxi trip arrival time includes the vehicles physical features, speed, trip distance to name a few while the external

factors refer to events that cannot be directly controlled by the vehicle but have a direct impact on taxi trip travel times such as traffic congestions, weather conditions, carnivals, etc.

For our research, we believe that the dataset utilized covers a long enough period capable of identifying periodic patterns possibly created by the external traffic factors, hence we choose to solely focus on the internal traffic factors present in our dataset for arrival time prediction. For this research, arrival time is simply the time it takes for a taxi to travel from a given pick-up point to a drop-off point, as earlier discussed this can be affected by internal and external factors. However, for our research we would be utilizing the internal factors captured in our dataset for taxi trip arrival time prediction.

1.2 Ride-Sharing Prediction

The vast amount of data available creates an environment whereby researchers can carry out valuable research on the collected traffic data and using advanced technological traffic analytic tools to identify meaningful insights and trends. For this research, ride-sharing prediction refers to the utilization of machine learning and deep learning tools to predict whether a taxi trip ride was shared or not with or without utilizing the passenger count information recorded during the trip. This research work supports the widely accepted view that an increase in shared rides can potentially have a direct impact on the current adverse traffic conditions being experienced in urban environments. We believe that an effective ride-sharing scheme can result in reduced cost to both the passenger and the drivers involved, the following is a brief discussion of a potential ride-sharing scenario describing how cost can be minimized for both parties involved, this representation doesn’t not depict a real-life transportation route.

Figure 1. Ride-sharing trip trajectory.

Fig 1. Is an illustration of a simple ride-sharing trajectory, the travel trajectory consists of 3 pick-up point as well as 3 different routes, the driver is presented with the following 3 scenarios.

- Scenario 1: a driver picks up passenger 1 at pick-up point A, travels through route 1 to the drop-off point.
- Scenario 2: a driver picks up passenger 1 at pick-up point A, driver travels to pick-up point B to pick-up passenger 2 and travels through route 2 to drop off point.
- Scenario 3: a driver picks up passenger 1 at pick-up point A, driver travels to pick-up point B to pick-up passenger 2, driver travels to pick-up point C to pick-up passenger 3, and travels through route 3 to the drop off point.

The scenarios discussed above presents the possible taxi trip trajectory whereby the choice to share the ride could result in different outcomes. Scenario 1 is an unshared ride whereby the trip is

completed at a set price and there is no cost-saving for driver or passenger, scenario 2 offers a different perspective whereby the driver can offer passenger 1 a lower travel cost at the expense of picking up another passenger at pick-up point B, this increases passenger one's travel time but would, in turn, increase the driver's income depending on the travel distance to the drop-off point. Scenario 3 offers an even trickier situation whereby the driver can offer a lower travel cost to passenger 1 and passenger 2, this is at the expense of picking up passenger 3 at pick-up point C, even though the travel distance from pick-up point B to the drop-off location is the same regardless of if passenger 3 is picked up or not.

The major highlights from the scenarios discussed offer a transparent view on how ride-sharing can serve as an effective means for simultaneously reducing passenger and driver travel costs while also showcasing better vehicle efficiency due to fewer vehicles being required for transportation activities.

Figure 2. January 2017 green taxi data.

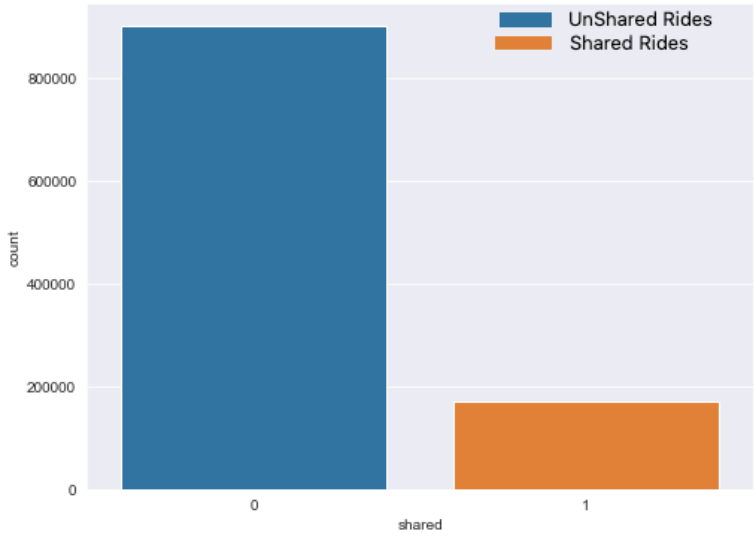
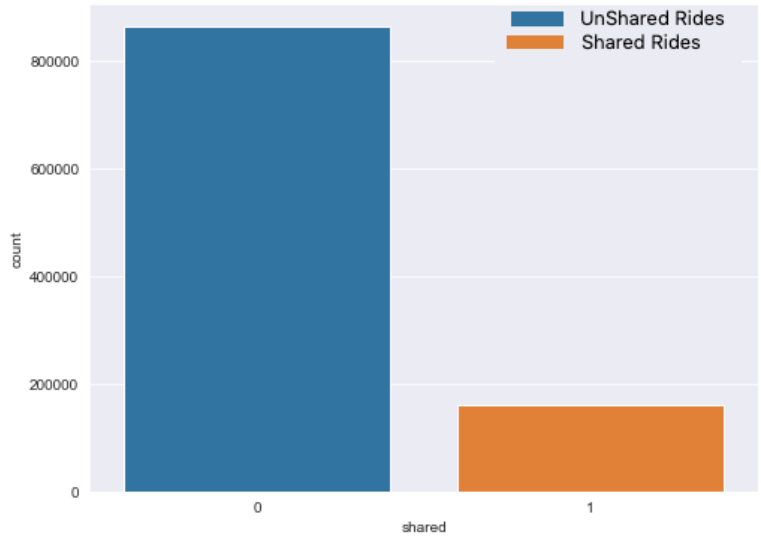


Figure 3. February 2017 green taxi data



From our exploratory data analysis Fig 2 and 3 show that just about 20% of the New York City green taxi trip data for January and February 2017 were shared, this trend has been observed across many periods highlighting the lack of ride-sharing and high vehicle utilization, considering each trip requires a vehicle. We also observe that a large number of taxi trips had the same pick-up locations and similar drop-off locations as shown in Fig 4 and 5.

Figure 4.January 2017 taxi trip pick-up locations

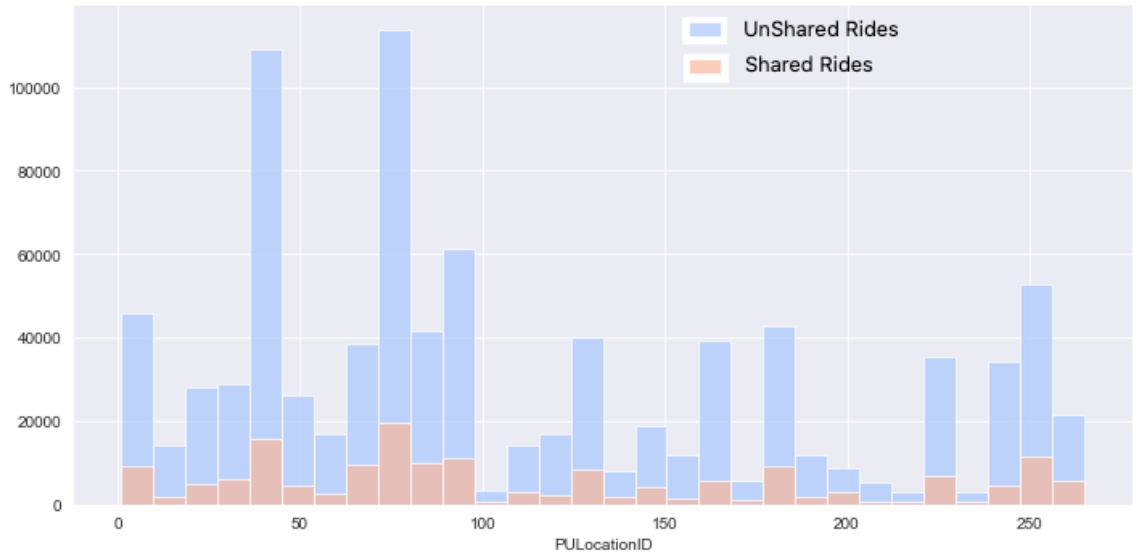
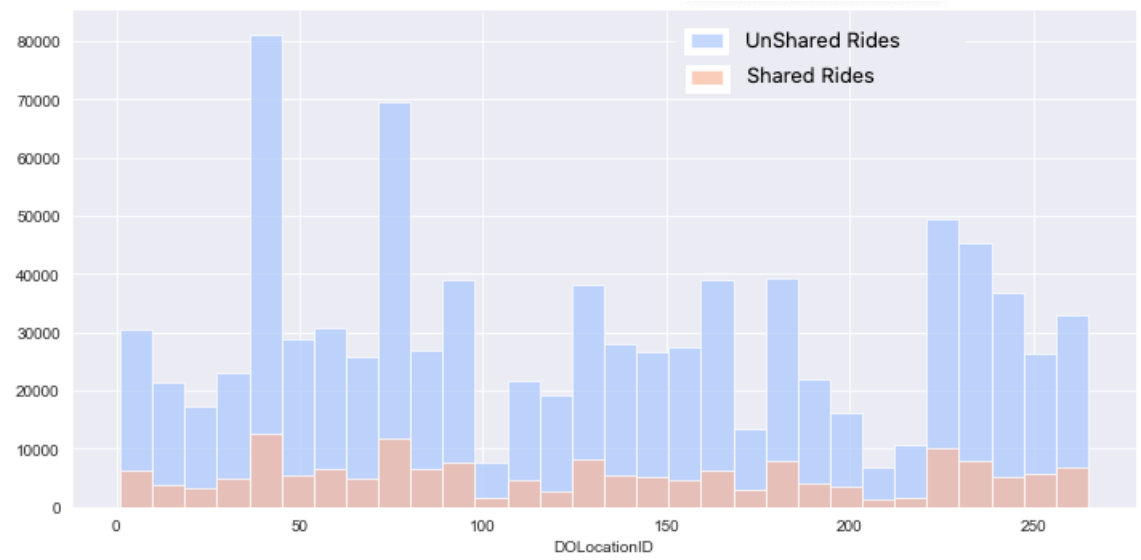


Figure 5. January 2017 taxi trip drop-off locations.



The glaring relationship between the number of taxi trips with similar pick-up and drop-off locations as shown in figures 4 & 5 depicts an environment whereby there is a high potential to reduce transportation costs via efficient ride-sharing schemes, however, it is worth noting that for ride-sharing to be successfully implemented in today's urban society, there has to be a collective

effort from every involved party to create a sustainable ride-sharing environment i.e. both drivers and passengers have to be open to the idea of sharing rides.

1.3 Motivation

The author of this research work believes that an efficient intelligent transportation system (ITS) can conveniently have a positive impact on current traffic problems i.e. carbon emissions, operational costs, traffic congestions, avoid road accidents, vehicle efficiency. The observations from our experiments and proposed solution offer a promising insight as to how much operational costs can be reduced via implementing effective arrival time prediction and ride-sharing schemes. From our literature review, we observe that the related works in the field of ride-matching and sharing generally accept that fewer number of vehicles are capable of meeting the current transportation demands in most urban cities. The capability to accurately predict trip arrival times offers a healthy environment for efficient transportation management planning at the highest levels of decision making. The capability to predict shared-rides without the passenger-count information using historical trip data shows that in a real-time environment, we can identify rides exhibiting ride sharing characteristics i.e. rides with potential to be shared in real time. The information from such rides can also identify the preferred hotspots where ride-sharing is more prevalent which ultimately boots informed decision making for efficient vehicle allocation. This also shows that in an autonomous vehicle environment, the vehicles can make informed decisions on the best locations to pick-up passengers without an external party selecting preferred vehicle dispatch locations.

In (Huang, 2019), the authors believed that connected vehicles and high-precision real-time location services are at the heart of Intelligent transportation systems (ITS) research, and the emergence of 5G networks promise better data rates, reduced delays and easy connection for large scale equipment's in (Kljaic et al. 2016). This presents a new era of opportunities for building

advanced traveler information systems (ATIS) capable of taking full advantage of the burgeoning technology. We remain excited and optimistic about how the introduction of autonomous and connected vehicles into our intelligent transportation system (ITS) concept would boost performance and even better vehicle efficiency through eliminating the driver requirements and centralizing the entire system on the needs of the passengers.

1.4 Summary of Gap Analysis and Problem Statement

The key takeaways from our gap analysis and problem statements are the continuous lifecycle of developing and improving intelligent transportation systems, it is important to emphasize that continuous advancement in technology has a parallel relationship with the development of intelligent transportation systems. However, we observe that the majority of arrival time prediction research is centered towards buses and public transit systems while the concept of ride-sharing is approached as a linear optimization problem and utilizes shortest path calculations such as Dijkstra's shortest path algorithm or using Euclidean distance for ride-matching calculations.

Some generally observed problems in the field of intelligent transportation systems include the continuously observed traffic congestions in densely populated urban cities, high operational costs for vehicle management systems, low vehicle efficiency, and road accidents. The impact of utilizing too many vehicles for executing the daily transportation needs also results in an adverse impact on the environment via carbon emissions, this remains a major problem that needs to be addressed and various solutions have been proposed.

1.5 Summary of Contributions

This research work proposes a ride-sharing induced intelligent transportation system capable of predicting taxi trip arrival times and rides with the potential to be shared to assist with making informed decisions on efficient vehicle allocation. We develop deep ensembles, combining the

predictive capability of deep learning models and ensemble boosting techniques, for arrival time prediction we utilize modern stacking techniques to combine a convolutional neural network (CNN), an AdaBoost model, Bagging model, and Gradient Boosting models. We execute a similar deep ensemble approach for ride-sharing prediction, utilizing a Capsule Network (CapsNet) in place of a convolutional neural network (CNN) for deep learning.

The proposed solutions presented in this thesis offer a deep learning approach for executing accurate predictions which is more in line with the modern standards of artificial intelligence and machine learning. The proposed solutions herein, can contribute to real time traffic prediction for taxi trip arrival times and shared ride prediction using real-time traffic information, our proposed solutions have the potential to reduce road accidents, traffic congestions, carbon emissions, reduce transportation costs for all parties involved i.e. passengers, drivers, and operations management.

1.6 Structure of this Thesis

The chapter 1 of this these provides an introduction and background of the purpose of this research work, we present a detailed overview of intelligent transportation systems, arrival time prediction and ride-sharing prediction, we also offer a brief discussion on a possible ride-sharing scenario and a brief overview of the dataset utilized for executing experiments and developing our proposed solution, chapter 2 is a summary of the related works in intelligent transportation systems (ITS), arrival time prediction, ride-sharing and ride-matching, autonomous and connected vehicles, we provide a comparative analysis of the related works to deliver a summary of the approaches executed in the related works studied.

Chapter 3 offers insight on the kind of dataset required for developing intelligent transportation systems predicting trip arrival times and building ride-matching and ride-sharing systems, chapter 4 highlights the gap analysis in the related works discussed, our problem statement and research questions.

In chapter 5 we execute regression and classification experiments, utilizing machine learning, deep learning techniques and ensemble techniques, our experiments led us to the identification of the preferred models for our proposed solutions for arrival time and ride-sharing prediction. Chapter 6 provides a detailed insight into the design of our proposed solutions, in chapter 7 we evaluate our proposed solutions and compare our results with other related works.

In chapter 8, we conclude with the key strengths of our proposed solution, chapter 9 discusses our future work to possibly improve our solutions and in chapter 10 we cite all the research works referenced in this thesis.

Chapter 2: Literature Review

In chapter 2 we carry out a literature review of some research works in various fields of transportation, our major focus, however, is on intelligent transportation systems (ITS), advanced traveler information systems (ATIS), arrival time prediction, ride-matching, ride-sharing, autonomous vehicle (AV) and connected vehicles. We learn about linear optimization problems, shortest path calculations, the modern machine learning and deep learning techniques utilized for building industry-standard transportation system, we also gain more information on the need to build intelligent systems capable of scalability and also applicable in any transport environment. The related works studied are in the field of ride-matching, ride-sharing, arrival time prediction and autonomous vehicles, hence, we segregate the related works to capture more focus on these works.

2.1 Ride Sharing and Ride matching

In (de Lira et al. 2018), the authors proposed an activity-based ride matching (ABRM) algorithm for matching ride requests with ride offers with the possibility of proposing alternative destinations where the required activity can be performed, they believed that ride-sharing and car-pooling services have the potential to considerably reduce urban traffic levels, their proposed system collected ride-sharing offers of users and matched the offers with ride requests of users with similar destinations, the matches were then ranked based on the anticipation or delay of the trip, the distance a passenger is expected to walk to a pick-up point, the estimated duration of the ride from pick-up point to the destination and the length of the ride.

They conducted experiments on publicly available semi-synthetic datasets of Foursquare users in New York City and Tokyo from April 12, 2012, to February 16, 2013, the data features recorded

include time-stamps of check-ins and GPS coordinates, the results of their experiments showed that the ABRM algorithm results in a 54.69% increase in the number of rides that can be satisfied in comparison with traditional ride-sharing systems.

In (James, 2018) the author developed a two-staged scheduling methodology that handles the vehicle allocation and vehicle charging phase of ride-sharing activities, the identification of the points of interest (POI) on the trip using Dijkstra's algorithm for calculating shortest paths and the routing stage to ensure the trip route covers all the stated points of interest (POI).

The problems they intended to address in their research involved developing schedules for delivering logistic requests especially for last-mile requests and considered request allocation, vehicle routing, and battery charging as the major problems to be addressed by their system. They believed that the sensing capability and full-fledged controllability of autonomous and electric vehicles mean they can handle these tasks and are a better alternative due to their ability to adapt and respond to different transportation situations.

Their proposed methodology separated the battery charging and allocation of request aspect of each ride from the request scheduling stage and utilized the time it took for each vehicle to service a selected request to obtain a detailed route for the trip. For the problem formulation they utilized a mixed-integer nonlinear program and made use of linear transformation techniques to reduce the problem of computational complexity, they tested their solution on the real-world transportation network and traffic data of Cologne, Germany, and extracted the map of the city from OpenStreetMap.

In (Ota et al. 2016), the authors believed that ride-sharing occurred in real-time therefore there's a need to have a system that accommodates online spontaneous requests. They proposed a Simulating Taxi Ride-sharing (STaRS) data-driven framework capable of analyzing a wide range of ride-sharing scenarios where ride-sharing trip information don't need to be known in advance. They developed an optimization algorithm linear in trip numbers and used an efficient indexing scheme utilizing parallelization to make their proposed system scalable.

They obtained their dataset from New York City taxis containing 360 million trips captured from 13,000 taxis in 2011 and 2012, the goal of their algorithm was to minimize total cost or utility of ride-sharing while meeting several constraints dictated by the taxi vendor including the maximum number of stops and wait times, number of passengers allowed, the travel distance and the road network.

They used Dijkstra's shortest path algorithm to determine the distance between locations and adjusted traffic conditions using a weight matrix that changes over time. For their simulations, they implemented a thread pool model that distributes ride requests across multiple machine cores and reduced inter-thread communication while processing multiple taxis at a time.

They ran multiple experiments of their algorithm using an open-source Apache Hadoop software library on 1200 clusters and showcased a MapReduce Simulation of their algorithm.

In (Kaplunovich and Yesha, 2018), they identified some data analysis techniques for their research on ride consolidation, they looked at many alternatives including SQL database querying, AWS Glue, and Athena, EMR with Hadoop. These tools had some shortcoming with regards to their needs and therefore there decided to utilize EMR with Spark for their data analysis activities thanks

to the ability of Spark to offer full control over data processing and transformation, as well as selecting server choices and auto-scaling.

The purpose of their research was to identify the scenarios that can lead to rides being shared based on several conditions including the distance of the rides start location, the start time of the rides, how close the ride destinations were to each other and the total number of passengers that can fit into a vehicle.

They used zeppelin workbooks to execute Scala spark commands and used a ganglia web application integrated with EMR to monitor their clusters. In using zeppelin, they were able to identify the parameters that needed optimization to improve performance, and also identify the need for more instances to speed up the data processing,

In (Pedro and Ferreira, 2014), the authors designed a taxi-sharing system with the following attributes i.e. automatic decision making, constrainable, cost-effective, distributed, door-to-door and dynamic, they noted that for the system to be effective the following requirement has to be met i.e. GPS coordinates of the passenger and driver, wireless communication hardware (smartphone) and routing and navigation capabilities available in the vehicle.

They noted that for the proposed system to be effective more emphasis has to be placed on the needs of the passenger and the utilization of the available vehicle fleet, they also showed how the system can be cost-effective to passengers by proportionally sharing the cost of the rides based on the distance covered, the number of passengers, the time of the day and also the day of the week.

They utilized DIVERT, a sophisticated microscopic traffic model simulator based on an intelligent driver model to carry out simulations of their concept and used a MOBIL model to simulate lane changes, these tools were used to develop their taxi operation model capable of executing single

rides as well as taxi-sharing rides when necessary. They used a list of real taxi-requests obtained from a taxi fleet in the city of Porto, Portugal to carry out over 360 simulations of their proposed algorithm.

In (Golpayegani, 2018), the author developed a Co-ride model that considers passengers and drivers as autonomous agents with multiple preferences. Their model includes a taxi-depot containing all the vehicles available in a distributed context whereby all taxis have the information of all the ride requests, and a taxi-depot manager that facilitates the process of dispatching taxis to appropriate rides. They noted that passengers choose to share rides based on various reasons such as travel cost, environmentally friendly preferences and also the travel time.

They used a dataset obtained from the annual taxi statistics for Dublin, Ireland to identify the preferences of the drivers including the preferred revenue per day and the desired number of working hours. To simplify the entire process their Co-ride-sharing model was further broken down into three phases i.e. information sharing phase, option calculation and the decision-making phase.

The information-sharing phase involved sharing the matched rides with the taxi drivers and passengers based on their preferences, the calculations involved identifying the shortest path for routing and identifying the preference of the passenger with regards to the number of passengers they are willing to share a ride with. The decision-making phase involves the selection or acceptance of the rides offered based on the Co-ride model which can either be accepted or rejected by either the passenger or driver.

In (Thangaraj et al, 2017), the authors developed a dynamic ride-sharing system (Xhare-a-Ride, XAR) that can be integrated with a multi-modal trip planning (MMTP) system and ensures high levels of accuracy while considering users constraints like detours and required walking distances, the system also provides several options based on the selected sharing preferences. They developed a three-tiered discretization system for in-memory indexing for updating and retrieval of spatiotemporal ride-sharing information, this eliminates the need to carry out the shortest path calculation for real-time searches.

The three-tiered region discretization consists of grids i.e. a bounded square geographic region, a landmark (point of interest in a grid) and a cluster which is a collection of landmarks, the XAR system is further broken down into three components i.e. XAR pre-processing, XAR run time unit and the XAR in-memory indexing unit.

They identified regional clusters via their cluster-minimization problem using an integer linear program, they then used a greedy search algorithm to approximate the distance between the identified clusters. They were able to identify the location of vehicles while in route by updating the clusters that were reachable to the vehicle based on its current GPS and marking unreachable clusters as obsolete. They noted the entities that characterized XAR rides as a source and destination location, seats, departure time, via points, segments, detours limits, pass through and reachable clusters.

In (Bathla et al, 2018), the authors developed a dynamic algorithm for taxi ride-sharing (TRS) based on the terms of ride-sharing flexibility, they proposed a novel distributed algorithm for ride-sharing that utilizes localized communication networks between close-by users and taxi drivers to match possible ride requests. They believed that the increasing number of taxis in urban cities

could lead to higher levels of traffic congestion and carbon emissions and therefore the problem needed to be addressed through efficient ridesharing.

Their analysis of traffic data from Shanghai, San Francisco, and New York City showed a high volume of short trips during office hours meaning ride-sharing could effectively reduce the required number of vehicles on the road, reduce passengers' wait time and travel cost.

Their system accepts several ride-sharing requests and stores the information into a temporary schedule, upon analyzing the eligibility requirements of the request the taxi responds with a false status for rides that don't match and true for rides that meet the eligibility requirements i.e. costs, available seats, distance.

In (Duo et al, 2017), they analyzed the ride-sharing services of three TNCs based on their temporal, spatial and distribution characteristics. They noted that TNCs can effectively reduce the time it normally takes to find a client as well as providing a secure and automated way for trip cost handling. They believed that implementation of efficient ride-sharing techniques could reduce congestion, fuel consumption, pollution and save parking spaces.

They combined online behaviors of users on ride-sharing applications and offline behaviors to study how residents prefer to travel and used the theory of human dynamics to explain the frequency of ride-sharing usage and times of the day. They used clustering analysis to obtain information on how people move around.

They obtained data from Chinese 2G and 3G service providers in northern China, the major sources of the data were from mobile devices, wireless access networks, and core networks. They identified keywords in their data i.e. call a car, cancel the order and create actions from the keywords.

In (Edoardo et al, 2014), the authors believed that dynamic ride-sharing techniques could lead to an improved transportation system. They outlined that beyond environmental sustainability, the reduction of vehicles in urban areas via shared supply systems is the best way to reduce traffic congestions. They develop a system architecture consisting of a supply manager that accepts and processes all ride requests and stores the current state of all the ride options i.e. bus, dynamic ride-sharing, and one-way car sharing.

Their optimization problem focused on outlining their notations, developing a network model to identify the location of users in a trip, the departure and arrival times of the ride-sharing trip, a virtual driver for ride-sharing, and car-sharing drivers and also analysis of the pedestrian and bus modes.

In (Bicocchi et al, 2015), the authors present a methodology that extracts suitable information via a user's mobility traces and estimates the advantages with regards to mobility reshaping of ride-sharing opportunities. They noted that the increased adoption of mobile phones and social media tools offers the opportunity to collect voluminous amounts of data with regards to the mobility habits of people. They believed that current pervasive technologies allow the automation of identifying and negotiating matching rides among users.

They proposed a fully autonomous system where users are not required to explicitly state their needs, they utilized previously analyzed mobility patterns to recommend suitable offers to users when available. They obtained a large dataset of user mobility information from an Italian telecom operator and analyzed the call data records (CDR) of over 3.5 million people.

They utilized clustering techniques to identify home-work and general mobility routines of users, highlighting the average commuting hours and the number of rides that could be shared based on

the flexibility and preferences of the users like small detours and delays. They adopted a simple agglomerative algorithm using the geographic distance between multiple CDRs as against K-Means clustering because K-Means requires a set number of clusters for analysis.

In (Cao et al, 2015), they propose a dynamic ride-sharing system that matches rides based on cost and temporal constraints, their system SHAREK allows drivers outline their ride preferences before they pick up rides and calculates the cost of rides based on detours and trip distance. They believed that SHAREK offers a socially sustainable way to handle global transportation problems like high gas prices and jam-packed traffic.

They executed SHAREK in three phases, first they reduced the required number of shortest path computations by using Euclidian temporal pruning to identify a set of drivers capable of arriving at the trip pick up point on time based on Euclidean distance computations. They then carried out conservative Euclidian computations to select the optimal driver based on cost constraints and finally, they used semi-Euclidean skyline-aware pruning to compute the shortest path of the road network.

In (Pandit et al, 2019), the authors carried out an extensive study on methods to identify pricing strategies for static and dynamic ride-sharing, they developed a system that uses a queuing framework where customers and drivers arrive into two separate queues and matched based on cost constraints. The goal of their work, however, was to maximize the revenue generated by their platform, therefore, the pricing of the rides was set by their system, they used the volume of available matching rides to determine the price of rides i.e. the higher the options the lower the price.

They utilized Bernoulli and Poisson techniques to analyze the relationships and differences between static and dynamic ridesharing prices, prices were deemed static when the price per transaction was independent of the state of the system and dynamic when the price per transaction was a function of the current state of the framework. They however noted that dynamic pricing could perform better than static pricing when customers and drivers make decisions based on an instantaneous price.

In (Wei et al, 2019), the authors studied how the inclusion of autonomous vehicles can lead to higher profit margins, they proposed a model that utilized both human and autonomous vehicles in a multi-location network for executing ride-sharing. The proposed platform handles drivers compensation, price selection and operation of autonomous vehicles at a fixed price, they noted that due to the technological hurdles as at the time of their research it could take a reasonable amount of time to transition ride-sharing platforms into an era of 100% autonomous fleets.
The proposed system considers human drivers for trips before engaging autonomous vehicles ignorer to create an attraction for human drivers using the platform. For scenarios where there's no suitable match for a human driver due to conditions like price and distance constraints the rides are thereby executed by autonomous vehicles.

In (Duan et al, 2019), the authors highlighted that the most important problem of ride-sharing is the process of matching rides efficiently between passengers and drivers. They noted that the spatial-temporal nature of passenger requests and the benefits available to a driver have to find an equilibrium for the rides to be efficient to both parties. They outlined the approaches currently used by ride-sharing service providers i.e. the system assigning and driver grabbing approaches.

They noted how the preferences of passengers are placed in higher regards considering the users directly pay money to the service providers thereby underestimating the drivers' importance.

They proposed an order dispatch scheme that considered both the needs of the passenger and the driver though obtaining GPS information on both parties to make an informed decision for ride-sharing matching. This creates an environment where multiple drivers evaluate the same ride at the same time resulting to a reduced wait time for passengers, they observed that when ride information is dispatched to drivers in a geographically wider region the number of drivers reachable is larger which also results in possibly longer rides. They proposed a dynamic programming algorithm that adaptively reduces the geospatial range of the dispatch messages resulting in a reduction of idle driving time and waiting time.

They noted that drivers within the geospatial range of multiple user requests could face a problem of overlapping requests and adjusted their algorithm handle overlapping scenarios. For their experiments they obtained synthetic datasets and real-world dataset from Didi's GAIA open data program containing passenger request and trajectory data of Chengdu City from Nov 2016.

In (Zhong et al, 2018), they carried out extensive studies to prove that optimal matching algorithms sometimes become unstable due to passengers and drivers refusing their matching options due to their ride-sharing preferences. They noted that the increased use of mobile devices facilitates an environment where ride-sharing can serve as an effective mode of transportation while contributing to reduction in environmental pollution, traffic congestion, and energy consumption. They proposed a novel-auction based decentralized dispatching system where drivers selected their preferred passengers, the virtual price of the ride increases based on the number of drivers interested until only one driver is willing to accept the trip. They highlighted that most ride-sharing

systems utilize a centralized dispatching system which could require a lot of computation due to the voluminous amount of taxi fleets in urban areas.

Their research was based on 3 main goals, maximizing social income through effective taxi dispatching, reduced computation complexity, and workload on a central server and maintaining the stability of ride-sharing matching results. They discussed the importance of considering virtual costs of rides like the time and energy expended by the driver based on the distance covered and other factors as well as physical costs like the cost of gas.

In (Teung et al, 2019), the authors proposed an activity-based shared mobility (ASM) model that determines ride-sharing schedules based on the preferred activity type carried out by the passengers. They noted that information gotten from users mobility activities could be summed up into their social comfort, energy savings appetite and result in a higher vehicle space utilization. They highlighted that if executed efficiently, the framework could reduce the number of vehicles on the road and travel costs, and also the introduction of autonomous vehicles into such an environment would further minimize the usage of resources, alleviate traffic congestions and reduce overall energy consumption.

Their proposed ASM model was made up of ride-sharing group formations achieved using clustering techniques to merge rides within a specified distance. They were able to execute schedule generation using a combination of rides clustered based on distance and also departure times. To address the need to reduce the number of vehicles required for trips, they implemented combination and trip integration techniques by transforming trip schedules into a minimum weight path cover problem solved by polynomial-time, through the generation of a complete bipartite graph utilizing the Hungarian method.

In (Silwal et al, 2019), they carried out a survey on the architecture of ride-sharing systems, they categorized ride-sharing into static and dynamic and discussed the current designs including the central, distributed, and hybrid designs. They pointed out how traffic congestion is simply a result of the volume of vehicles on urban roads, they believed ride-sharing could increase productivity levels and conserve energy and highlighted how the introduction of autonomous vehicles can boost the environment through reduced carbon emissions.

They discussed how ride-sharing is generally considered an optimization problem and discussed some algorithms used to solve both static and dynamic ride-sharing problems. They noted how static ride-sharing system utilizes central systems while dynamic ride-sharing can be distributed, decentralized or hybrid. They believed that the dynamic systems offer a better solution for solving today's transportation problems.

In (Tsao et al, 2019), the authors proposed an MPC algorithm for ride-sharing autonomous mobility on demand (RAMoD) system, they developed a multi-commodity network for capturing double occupancy operations. The MPC algorithm handles assigning of multiple passengers to vehicles, design of vehicle routes and anticipation of future requests through vehicle re-balancing, the aim of their proposed model was to reduce the number of vehicles on the road through high-quality mobility service.

They believed that their proposed algorithm can combine the capabilities of ride-sharing and autonomous vehicles to address traffic congestion issues and generally improve traffic. They focused their studies on double occupancy considering most vehicles have at least two passenger seats and the computational complexity of simulating more than two passengers could result in diminishing returns on performance. They measured customer satisfaction using the waiting and

journey times for trips and used the total distance driven by the vehicle to obtain the operational cost.

They tested their algorithm using a publicly available dataset of the city of San Francisco containing over 464,000 trips and used an open-source simulator "AMoDeus" to validate their algorithms.

In (Luo et al, 2018), the authors proposed a privacy-preserving ride-sharing framework called pRide that carries out ride-matching efficiently based on road network distances without learning about information related to the driver or passenger's location. They utilized road network embedding (RNE) techniques alongside cryptographic primitives to estimate the shortest distance between the drivers and passengers. They implemented a prototype of their proposed scheme using homomorphic encryption and Yao's garbed circuit for distance calculations.

They made the following assumptions in developing their system i.e. the map is a public area that can be accessed by all entities, secure channels exist between all entities and there is no collusion between the Online ride hailing (ORH) server and the crypto provider used for setting cryptographic parameters and ride-matching computations. They developed a 2-scheme methodology using somewhat homomorphic encryption (SHE) and garbled circuit to achieve high accuracy for ride-matching and used partially homomorphic encryption (PHE) alongside partitioning and data packing techniques to achieve optimal efficiency.

In (Cangialosi et al, 2016), the authors introduced a generalized ride-sharing system (GRS) that combined multimodal transport systems including buses, taxies, dynamic ride-sharing, one-way car-pooling, and pedestrians to allow users schedule the different trip modes as a single task. They

noted that the ability of users to receive real-time information about travel times and bus waiting times as well as communicate with available drivers create new ways for planning transportation mobility.

Their architecture consisted of a supply manager (SM) that receives and processes ride requests from riders and drivers obtained from a real-time external module. The speed and position of all users was known by the supply manager making it possible to evaluate the possibility of users missing rides and also proposing ride matches. The SM worked in a rolling horizon framework meaning it is possible to identify new solutions for users unable to meet up with ride matches. Their model core was represented as a mixed-integer linear programming (MILP) problem that handled minimizing deviations from the users desired departure and arrival times, their proposed system provided multimodal solutions among a set of transportation options based on travel costs, vehicle capacity and path flexibility.

In (Singh et al, 2019), the authors propose a novel multi-hop ride-sharing (MHRS) algorithm that interacts with external environments and utilizes deep reinforcement learning via a deep neural network to dynamically learn the best options for vehicle dispatch. They believed that the changing distributions of vehicle locations, active vehicles, customer arrivals, and total travel time required a model-free distributed approach. Their proposed algorithm handled matching vehicles to customers, dispatch of vehicles in locations where high demands are anticipated, their system allowed users to change vehicles on their way to their destination to incur less cost at the expense of experiencing a slightly longer travel path.

The purpose of their research was to minimize passenger waiting time, reduce the demand and supply gap, minimize the number of fleets, number of transfers, control travel time and less fuel

consumption. They highlighted that their MHRS model resulted in better vehicle utilization of 20%, idle and waiting time reduced by 40%, 30% less operational costs 99% accept rate, reduced traffic congestion and pollution.

In (Alisoltani et al, 2020), the authors propose a multi-agent real-time ride-sharing system capable of dealing with dynamic traffic conditions, they used the average speed in the network to predict travel times for calculating vehicle fleet scheduling and developed a plant model that represents real traffic dynamics. They noted that the main problems of dynamic ridesharing were efficient management of the vehicle fleet and accurate prediction of pickup and drop-off times, their multi-agent system consisted of a passenger agent that sends requests to an optimizer agent that handles computation of optimal routes for all trip requests. This handles the problem of minimizing travel distance and passenger wait times then provides such trip information to an operator agent and passengers.

Their system was separated into demand characteristics, highlighting the ride preferences of the passenger and the service characteristics based on the preferences of the autonomous ride service provider, the mentioned ride constraints include vehicle capacity, time window, quality of service and sharing number. They highlighted that careful observation of the trip constraints could help identify the best route alternatives, they broke their algorithm into two parts. The first part handles route schedules based on the set constraints and the second part handles information regarding vehicle capacity, passenger allocation and possible re-routing of vehicles to pick up a passenger that already meets route requirements.

They used a realistic O-D matrix to obtain the transportation network the city of Lyon, France, consisting of 1883 nodes and 3383 links containing 62,000 trip requests, they used the current

mean speed in the network over 10mins to predict travel times and utilized a trip-based Macroscopic Fundamental Diagram (MFD) model updated every 10secs to simulate traffic dynamics.

2.2 Arrival Time Prediction and Autonomous Vehicles

In (Sherif et al. 2016), the authors believed that the elimination of the driver effort constraint by autonomous vehicles can make ride-sharing more convenient, popular and in the long run economically cheaper, they developed a system that protects a user's sensitive information when searching for trips, using Visual C on a real-life map of Cookeville city, Tennessee obtained from OpenStreetMap, they simulated their research and the results showed that the system would be effective when ride-sharing with autonomous vehicles becomes popular.

They collected trip data using a similarity measurement technique that represents data in a binary vector. The system further divides ride-sharing regions into cells represented by a bit in a vector, the pick-up, and drop-off locations are sent as packets from a primary user (driver) to a secondary user (passenger). They noted that for human-driven cars ride-sharing could become inconvenient if the distance between trip locations are too far but that would not be a problem for autonomous vehicles, they developed three ride-sharing cases based on the primary users travel distance and preferred location requirements to carry out tests and simulations. They evaluated their proposed system on trip data privacy, primary-secondary user's unlinkability, identity anonymity and authentication, and access control.

In (Al Najada and Mahgoub, 2016), the authors proposed a trajectory planning system for fully autonomous vehicles to predict a safe trajectory of autonomous vehicle rides represented by the position, ETA, distance, and estimated fuel consumption of AV trips. They believe that data

collected from historical accident datasets could help identify the events, timing, and attributes that could lead to a potential accident.

They outlined the various levels of vehicle automation and described by the U.S. Department of Transportation's National Highway Traffic Safety Administration (NHTSA) consisting of No-Automation (Level 0), Function-specific Automation (Level 1), Combined Function Automation (Level 2), Limited Self-Driving Automation (Level 3), Full Self-Driving Automation (Level 4). Their research work focused solely on fully automated vehicles and they noted that data obtained from autonomous connected vehicles (ACV) can help reduce the number of road accidents, casualties, and fatalities.

They developed a system architecture that predicts the possibility of accidents and congestions on a trajectory selected by an autonomous vehicle using big data analytics, they noted that the challenges their system includes connectivity, security, privacy, data transmission, and data aggregation. They believed that the best way to optimize the capabilities of autonomous vehicles was to have them all connected in a single environment.

They obtained their dataset from Florida Department of Transportation and executed their proposed algorithm which selects the safest route based on the desired preference of a user, for accident prediction they made use of a distributed random forest (DRF) classification algorithm and used a linear regression algorithm to predict the expected time of arrival (ETA) of each trip.

In (Kumar and Goel, 2018), the authors believed that for autonomous vehicles to work in a real-world environment, the use of highly intensive parallel computation mechanisms have to be implemented due to the vast amount of data collected by different sensors on autonomous vehicles such as roof-top Lidar, ultrasonic sensors, rear-mounted radar, gyroscope, and tachometers.

The architecture of their proposed model includes data sources via sensors and GPS satellites, a flume for collecting and transferring large amounts of data into a Hadoop Distributed File System (HDFS), a flume event considered a basic unit of data, a flume Agent acting as an independent JVM receives data and transferring to its destination. They used Kafta for log processing and Zookeeper for coordination between consumers.

They proposed Apache Spark as their data processing tool due to its renowned capacity for performing real-time cluster computing and utilized a convolutional deep learning neural network for object recognition, sparkNets as the training tool for deep neural networks.

In (Seungwoo et al, 2014), the authors proposed a new system that generates long-term traffic time predictions based on accumulated historical data. They suggested the use of vertical data arrangements and collection of historical traffic data in a single time slot. They highlighted that in forecasting time series data the various traffic patterns have to be considered to yield accurate results, the noted patterns include seasonality, cyclicality, and irregularity. They developed a spatiotemporal prediction map where the elements in the map are represented as a time-series forecasting method and for identification of prediction accuracy they utilized the R-squared value. They utilized various big data tools for statistical modeling, RHive served as an intermediate layer and used Hadoop Distributed File System (HDFS) for data storage and creation of clusters, their framework automatically analyzed traffic patterns, generated and verified predicted traffic data. They obtained traffic data using a regional Intelligent transportation system (ITS) that collected traffic data from buses, taxis, sensors using dedicated short-range communications (DSRCs).

In (Huang and Peng, 2018), they described a network partition algorithm that required fewer vehicles to serve the transportation need of a higher number of customers by considering future

travel demands, they noted that connected autonomous vehicles provide information about users travel demands and also a platform that enables centralized coordination. They proposed an algorithm based on multidimensional scaling (MDS) that projected road network locations in a Euclidean space. Travel locations were characterized using Dirichlet Process Gaussian Mixture Model (DPGMM), using these road network projections they were able to identify better clustering results in comparison to geometric coordinate-based methods.

They developed a fleet control algorithm that handles the demand distribution information of the vehicle fleets and proposed a Kullback-Leibler (KL) divergence regularization-based control policy that handles live trip requests. This creates a proper distribution for future travel demands thereby leading to less travel distance of their fleet and an overall reduction in operational costs for service providers.

In (Salazar-Cabrera et al, 2019), they developed a fleet management control system (FMCS) that executes tasks related to vehicle operation, compliance evaluation and scheduling services. They believed FMCS systems reduced risk levels, improved quality of service and overall operational efficiency, they believed that information obtained from fleet management of public transport can help transport companies improve safety, efficiency, and productivity even with lower budgets.

Their proposed FMCS utilized vehicle-to-infrastructure (V2I) communication to transfer information about vehicles real-time performance to a central module. They believed that the information obtainable from the vehicles i.e. location, speed, schedule performance can be used to improve road traffic conditions in intermediate cities. They noted that according to SINITT, a tracking service owned by the government of Columbia, the benefits of real-time traffic information include reduced passenger wait times, improved system efficiency, fleet optimization, trip planning and an overall reduction in operational costs.

In (Erika Ritzelle et al, 2018), they developed a real-time traffic information system using traffic-related tweets obtained from the metropolitan manila development authority's (MMDA) official twitter account. They used Twitters Streaming API filtered by named entity recognition, preprocessing, frequency counting and cleaning, they believed that the information obtained about the road networks could increase the accuracy of detecting real-time traffic updates in Manila, Philippines.

They divided their system into five main modules, the process of data gathering involved collecting and storing traffic updates related tweets from the official MMDA account using Twitter streaming API and filtered using named entity recognition. Data pre-processing involved tokenization to identify the parameters required for feature extraction and frequency counting to identify the terms tweeted the most in the dataset. For feature extraction, they created a bag of words for the following parameters, day, time, lane of the road, road direction, location, and traffic mode, this was required for their implementation of Latent Dirichlet allocation algorithm, they further classified the tweets into 5 traffic categories namely light, light to moderate, moderate, moderate to heavy and heavy traffic conditions.

In (Zhu et al, 2018), the authors believed that big data analytics is the future direction of intelligent transportation systems (ITS), they noted that intelligent transportation systems combine advanced technologies for electronic sensors, data transmission, and intelligent control. They believed that intelligent transportation systems would result in better services for both drivers and passengers. They noted the major big data sources including smart cards via automatic collection fare systems, GPS combining geographic information systems and map technologies, data from video cameras

deployed at strategic locations, sensor data from equipment's installed in ITS collecting data like speed, traffic flow and trip times, data collected from autonomous and connected vehicles shared via wireless networks and data passively collected through mobile phone usage.

They highlighted that the vast amounts of data produced and collected by ITS can be efficiently handled using big data analytics tools, they noted that big data analytics can handle previous problems related to data storage, analysis and management. They stated that new techniques can be used to identify ride patterns in transportation networks, road traffic flow prediction, personal travel route planning and help with public transportation planning, prediction of possible traffic accidents and real-time response in the event of accidents.

They describe the architecture of big data systems as having a data collection layer, a data analytics layer where processing and analysis of the collected data are executed and an application layer for visualizing information derived from analysis. They noted the use of machine learning techniques for big data analysis and how the adoption of artificial intelligence and deep learning models have brought about rapid developments in ITS.

In (Petersen et al, 2019), the authors propose a system that takes advantage of non-static Spatio-temporal correlations in urban bus networks to execute bus travel prediction times, they developed a model that combines convolutional and long short-term memory (LSTM) for a multi-output, multi-time-step deep neural network. They focused their research on urban bus networks considering buses have to share their traffic routes with other vehicles and are prone to the leading ripple effects. They highlighted how automated systems already utilize big data from intelligent transportation systems (ITS) and automatic vehicle location (AVL) systems to propose a suitable alternate ride option to customers in real-time.

The neural network topology consisted of input-to-state and state-to-state convolutions, they utilized a sequence encoder/decoder technique consisting of multiple convolutional LSTM's allowing them to predict the next 3 time-steps based on data from 20 previous time-steps. They observed that urban bus travel times vary based on time of day and day of week conditions hence the need to execute normalization techniques on the dataset to focus on the deviations from the expected patterns. They implemented their proposed model in Python using a Keras Framework and executed training using an RMSprop algorithm.

They obtained the publicly available dataset from Copenhagen's public transport authority Movia, containing 1.2 million records of bus travel time observation in 2017, and evaluated their proposed model on mean absolute error (MAE), root mean squared error (RMSE) and mean absolute percentage error (MAPE). They compared their model performance with Google Traffic maps, a naive historical rage model, Movia's traffic prediction model and a pure LSTM model for traffic prediction.

In (Agafonov et al, 2019), the authors develop a Long Short-term Memory (LSTM) neural network model that predicts bus arrival times based on heterogenous traffic information. They described traffic situations based on heterogeneous information consisting of day and time of the week, bus travel speed to have an idea of traffic congestions, the travel time of a preceding bus on the same route, and the weighted and historical travel times of all buses operating on the same route. They highlighted that LSTM networks could deal with long term dependencies by transferring cell state from one time-step to another and also control information flow using its input gate, forget gate, and output gate cell structure.

They carried out experiments using bus operations data of September 2018 from Samara, Russia, the GPS trajectories for the period were processed and converted into bus travel times, for comparative analysis they tested their LSTM model against a Linear regression model and an Artificial Neural Network using mean absolute error (MAE) and mean absolute percentage error (MAPE) evaluation metrics.

In (Chen et al, 2019), they propose a deep belief network (DBN) for bus travel time predictions, they utilized Gaussian-Bernoulli restricted Boltzmann machines (GBRBM) to extract the features of continuous data and used backpropagation techniques to further improve the performance of the model in a supervised manner. They noted that traffic conditions are the major influence of bus time predictions and believed that traffic information could be reflected from a vehicles travel speed, travel times, travel distance and dwell times. They utilized the temporal and spatial data obtained from preceding buses to predict travel times of the following buses. They utilized the final output of the GBRBM's as the input of the Back Propagation (BP) neural network and the output of the BP was considered the final result.

They obtained the dataset from vehicle recognition video data and 50,876 bus route observations in the city of Shenyang, the capital city of Liaoning Province in China, they evaluated the performance of their model against a k-Nearest Neighbors (k-NN) model, Artificial neural network, Support Vector Machine (SVM) and a Random Forest (RF) model, their evaluations were based on mean absolute error (MAE), mean absolute percent error (MAPE) and root mean squared error (RMSE) metrics.

In (Min et al, 2019), the authors propose a new approach for obstacle vehicle path prediction using deep ensemble techniques to train different networks of the same structure, they obtained sensor

data via executing experiment using an SUV with LIDAR sensors, cameras and GPS technology installed. The utilization of deep ensemble methods ensures the robustness of the estimation, acquisition of uncertainties and reduce overfitting, they believed that the travel path of a vehicle can be modified if the path of an obstacle vehicle is obtained in advance.

The input features obtained from the dataset include the vehicle's longitudinal and lateral speed, the position of the obstacle vehicle, obstacle vehicle velocity parallel to lane center obstacle vehicle velocity, and the distance perpendicular to the lane center.

They employed a deep ensemble with 5 RNNs of the same structure that collected inputs into three different fully connected layers, to produce trajectory predictions, a lateral distance of predicted path and standard deviation of lateral distance from the obstacle vehicles predicted trajectory to ultimately produce the coefficients of a 3rd order polynomial of an obstacle vehicle trajectory for 2 seconds. They utilized stochastic gradient descent (Adam optimizer) for weight and bias optimization. For comparison purposes, they tested a Long-short Term Memory (LSTM) model, Gated Recurrent Unit (GRU), Bidirectional RNN with LSTM, and Minimal RNN (MRNN) models to obtain the best results.

2.3 Comparative analysis of related works

Our comparative analysis consists of a wide spectrum of research on intelligent transportation systems, we study several alternative solutions for matching trip rides, identify the key factors that make sharing rides a more convenient, popular and economic option for both passengers and drivers. Some researchers employed multi-staged algorithms that divided the prediction tasks into separate smaller tasks and helped address the problem of computational resources, some of the research works also focused on privacy policies and how customers and driver's information can be protected using advanced encryption technology. Transportation data is generally spatial-

temporal in nature, particularly exciting research work by the authors in (Duo et al, 2017) involved utilizing users online behaviors on ride-sharing applications to identify patterns on how users prefer to travel which could be used to build either a single or multi-modal intelligent transportation systems, the identified patterns were used to determine the time of day users preferred to share rides. The work by (Bicocchi et al, 2015) highlighted that the current number of mobile phone users alongside the pervasive nature of mobile devices create an environment where social media tools can be used to collect voluminous amounts of data for understanding users mobility habits. The research works by (Wei et al, 2019) involved utilizing autonomous for executing rides thereby eliminating possible human errors and also proposed that autonomous systems would eventually yield more profit due to the possibility of executing more trips.

The widely accepted key features for an effective ride-sharing scheme from the study of our related works consists of the distance of the rides start location, the rides start time and number of passengers capable of fitting into the vehicles, there were some polarizing observations however as some researchers focused their research on satisfying the needs of the customers while some focused on the need on the driver and some focused on empowering the taxi company business as a whole. We believe that for ride-sharing to be effective and yield the required benefits, the efficient ride-sharing scheme would need to meet the requirements of every invested party.

The related work studied consists of research carried out from 2014, to identify the newest techniques, practices, and approaches being executed in building intelligent transportation systems using artificial intelligence, machine learning and deep learning. The datasets utilized was captured from various sources including taxi trip datasets, transportation network, and traffic data, taximeter surveys, sensors, LIDAR and GPS satellites information, 2G and 3G mobile data, trajectory data, road network data, smart cards, passive collections, social media feed, and bus route observations.

The research works utilized big data tools like apache spark, Hadoop distributed file systems (HDFS), autoregressive integrated moving average (ARIMA) models, the common deep learning models utilized include convolutional neural networks (CNN) and long short-term memory (LSTM) models and deep belief networks (DBN). The other discussed approaches include multi-modal transport systems combining multiple modes of transportation, the overall goals of our related works comprise within reducing carbon emissions, roadside accidents, traffic congestions and improving overall vehicle efficiency and real-time traffic prediction.

Table 1. Comparative analysis of literature review

No:	Big Data Source	Model/Technique	System	AV	Strengths	Weakness
(de Lira et al. 2018)	Foursquare Online Data from New York and Tokyo	Activity-based ride matching (ABRM) algorithm.	Dynamic	No	Creates ride-sharing matches based on a user's frequent activity.	Evaluations was based on a synthetic dataset, tests from real users required for tests.
(Sherif et al. 2016)	NA	NA	Static	Yes	Preserves private information of drivers and passengers.	Experiments were carried out without real life data.
(James, 2018)	Transport network and traffic data of Cologne, Germany.	Autonomous Vehicle Logistic System (AVLS) model.	Dynamic	Yes	Two staged methodology that separates the battery charging and allocation of request aspect of each ride from the request scheduling stage.	Yet to be applied in a distributed ride-sharing system.
(Ota et al. 2016)	360 million trips of NYC taxis captured from 13,000 taxis in 2011 and 2012.	Simulating Taxi Ride-sharing (STaRS) large scale system.	Dynamic	No	Achieved scalability using an efficient indexing scheme and a linear optimization algorithm.	Their utilization of Dijkstra's shortest path algorithm led to high computational complexity.
(Kaplunovich and Yesha, 2018)	New York taxi rides data from 2009 till 2017, 15 million rides per file.	EMR with Spark, zeppelin workbooks.	Static	No	Experimental results showed that 35% of rides could be potentially consolidated.	Implementation does not discuss any modern machine learning technique.
(Pedro and Ferreira, 2014)	Taxi fleet data from the city of Porto, Portugal.	DIVERT, microscopic traffic model and MOBIL model.	Dynamic	No	Improved performances of 8% reduced trip fares, 9% reduced travel time and operational costs.	Increased travel and wait times to passengers and possible degradation of quality of service.
(Al Najada and Mahgoub, 2016)	Florida Ministry of Transportation.	Trajectory planning algorithm.	Dynamic	Yes	Reduction of road accidents, casualties and fatalities.	Evaluated on lightweight and straightforward algorithms.
(Golpayegani , 2018)	Dublin Taximeter survey 2015.	Co-Ride algorithm.	Dynamic	Yes	Preferences of both passengers and drivers are considered for trip planning.	Untested with real road networks and on-line traffic data.
(Thangaraj et al, 2017)	350,000 records of taxi trips.	Xhare-a-Ride (XAR) algorithm.	Dynamic	No	XAR can be integrated with a multimodal trip planning (MMTP) system.	Under-performed in ride booking scenarios.
(Bathla et al, 2018)	Trip data of 4,000 taxi trips in Shanghai, China.	Taxi Ride-sharing (TSR) algorithm.	Dynamic	No	Cost savings for both passenger and drivers and less travel distance.	More data required for experimental evaluations.
(Kumar and Goel, 2018)	Sensors and GPS satellites.	HDFS, Apache Spark, Kafta, SparkNet, flume and Zookeeper.	Dynamic	Yes	Capability to process large volumes of data due to Apache Spark.	No experimental evaluations for proposed model.

(Duo et al, 2017)	Mobile data of 2G and 3G users in Northern China.	Clustering analysis.	Dynamic	No	Users mobility analysis provides user behavioral dynamics information for ride-sharing.	Experiments should be executed to show how data obtained improves traffic conditions and affects ride-sharing.
(Edoardo et al, 2014)	Transportation network of Genoa, Italy.	Generalized Ride-sharing (GRS) algorithm	Dynamic	No	Proposed a multimodal system for ride-sharing.	Modern machine learning techniques have proven to outperform optimization problems.
(Bicocchi et al, 2015)	Mobility data from Italian telecoms operator, 3.5 million call data records.	Agglomerative clustering algorithm.	Static	No	Proposed solution shows that up to 60% of trips can be saved via ride-sharing.	Proposed model employs pervasive technology which could violate user privacy policies.
(Cao et al, 2015)	223,606 edges and 175,343 nodes of a road network in San Francisco, USA.	SHAREK model.	Dynamic	No	SHAREK takes a few milliseconds to respond to a ride-sharing requests with over 10000 drivers close-by.	Pruning techniques could result in selecting a driver not preferred by the passenger.
(Pandit et al, 2019)	NA	Bernoulli and Poisson distribution techniques.	Both	No	Proposed model can be used to optimize average waiting time for drivers and passengers.	Focuses on maximizing platform as opposed to customer satisfaction.
(Wei et al, 2019)	NA	Mixed autonomy ride-sharing model	Dynamic	Yes	NA	Proposed platform ignores autonomous vehicles for human divers despite costing less.
(Duan et al, 2019)	Didi's trajectory data in Chengdu City	Dynamic Programming algorithm.	Dynamic	No	Reduced driver pickup distances and reduced dispatching time.	Possibility of increased passenger waiting time.
(Zhong et al, 2018)	MATLAB taxi scenario simulations.	Distributed algorithm.	Dynamic	No	De-centralized dispatching system reduces computational complexity.	The proposed algorithm could result in passengers paying more for rides.
(Teung et al, 2019)	NA	Activity-based shared mobility (ASM) model	Dynamic	No	Cost-efficient shared route pattern in an offline setting.	No simulations or evaluations with real world datasets.
(Seungwoo et al, 2014)	8769 traffic data files, each file contains about 50,000 records.	Autoregressive integrated moving average (ARIMA), RHive, HDFS.	Static	No	Vertical data arrangement provided a suitable platform for analysis and prediction of future traffic conditions.	Difficulty in predicting next day traffic due to changes in traffic conditions.
Tsao et al, 2019)	464,045 taxi trip records from San Francisco, USA	Ride-sharing autonomous mobility on demand (RAMoD).	Dynamic	Yes	20% less distance as opposed to single trips and 40% reduction in passenger waiting times.	Possibility of increased customer wait time due to the need to have a large number of customers present before vehicle dispatch.
(Luo et al, 2018)	21,048 nodes and 21,693 edges of a road network in California, USA.	pRide, Road Network Embedding (RNE).	Dynamic	No	Proposed scheme took less than 10 seconds and boasted a 99% accuracy for ride matching.	Privacy preserving means no information is obtained from the activity therefore nothing can be learnt from historical records.
(Huang and Peng, 2018)	Trip information of over 2,800 vehicles from a safety pilot model deployment (SPMD) project.	Dirichlet Process Gaussian Mixture Model (DPGMM), multidimensional scaling (MDS).	Dynamic	Yes	Serves more passengers per vehicle, meaning more idle vehicles are available for possible future travel demands.	Proposed model shows degraded performance when tested with dynamic traffic situations.
(Cangialosi et al, 2016)	Transportation network of the city of Genoa, Italy.	Generalized ride-sharing system (GRS), Supply Manager (SM).	Dynamic	No	Proposed system always offers multimodal trip solutions to passengers.	Proposing multimodal trips required higher computational resources.
(Salazar-Cabrera et al, 2019)	NA	Fleet Management Control Systems (FMCS).	Dynamic	No	Reduced risk levels, improved quality of service and overall operational efficiency	Proposed system developed for intermediate cities could be un-applicable in an urban environment.
(Erika Ritzelle et al, 2018)	Traffic related tweets from the metropolitan manila development	Latent Dirichlet allocation algorithm, K-	Static	No	Increased accuracy of detecting real time traffic updates.	Combination of traffic related tweets with actual traffic data could provide better analysis.

	authority's (MMDA) official twitter accounts.	Nearest Neighbors (k-NN) algorithm.				
(Zhu et al, 2018)	Smart cards, GPS, Videos, Sensors, Passive collection.	Machine Learning, Apache Hadoop, Apache Spark.	Dynamic	No	High level description of Big data architecture required for developing intelligent transportation systems (ITS).	Research work does not include any implementation of discussed techniques.
(Singh et al, 2019)	New York City taxi trips data.	Multi-Hop Ride-sharing (MHRS) algorithm.	Dynamic	No	MHRS model resulted in better vehicle utilization of 20%, idle and waiting time reduced by 40%, 30% less operational costs 99% accept rate.	Ride-sharing with more than 2 hops don't provide optimal results.
(Petersen et al, 2019)	1.2 million records of bus travel time observations in 2017 from the city of Copenhagen, Germany.	Convolutional and Long Short-term Memory (LSTM) neural networks.	Dynamic	No	LSTMConv model outperforms most traditional models in prediction of traffic conditions during peak periods.	High computational complexity of training activities.
(Alisoltani et al, 2020)	Transportation network in Lyon, France, 1883 nodes and 3383 links. 62,000 ride requests.	O-D Matrix, Macroscopic Fundamental Diagram (MFD) model.	Dynamic	Yes	Reduced required vehicle fleet by half and travel time by 226hrs.	Method for predicting travel time could be improved i.e. machine learning or deep learning.
(Agafonov et al, 2019)	Bus operations data of September 2018, Samara, Russia.	Long short-term Memory (LSTM) neural networks.	Dynamic	No	Real time bus arrival time prediction.	The proposed model was tested on a single bus route, might not be effective on different routes.
(Chen et al, 2019)	50,876 bus route observations in the city of Shenyang, the capital city of Liaoning Province in China	Deep belief network (DBN), Back Propagation (BP) neural network, Gaussian–Bernoulli restricted Boltzmann machines (GBRBM).	Static	No	Maintains prediction accuracy during peak periods and showed superior prediction accuracy.	Tested on a single bus route and not compared to other deep learning models.
(Min et al, 2019)	Experimental sensors result from an SUV with cameras, LIDAR and GPS technology installed.	Deep Ensemble Learning using RNNs.	Static	Yes	Deep ensemble learning ensures robustness of prediction accuracy, acquisition of uncertainties and reduced overfitting.	Dataset utilized for tests doesn't represent big data, additional data would be required.

Chapter 3: Dataset Review and Analysis

The dataset literature review provides highly enlightening information with regards to understanding the kind of data required for developing intelligent transportation systems, executing arrival time prediction, identifying and predicting shared rides, simulating ride-sharing activities and autonomous vehicle data requirements. The main observation generally is the use of taxi trip data for AV simulations and predictions due to the similarity in the behaviors and expected activities to be carried out by an autonomous vehicle.

In (Cui et al, 2018), they obtained their dataset from a trajectory of 12,509 taxis in Beijing for 27 days, they thought that the similarities in trip behaviors of taxis and that of autonomous vehicles were similar with regards to moving from a starting site to a destination site, the major components of the data was the time it took to complete a trip, taxi location, taxi ID and taxi state.

Strengths: The use of taxi data gives them a realistic idea with regards to the expected behavior of an AV concerning to the captured attributes, the expectation is that the results obtainable from this dataset should be able to conveniently predict the expected outcomes of AV trips.

Weaknesses: The data captured also observe just a portion of Beijing which in respect to the entire city might not offer in detail the actual real-life traffic experiences of individuals in Beijing, it would have also been great to capture some additional attributes such as the average taxi speed, distance covered, passenger capacity and the number of stops in a trip. We believe that additional information if included could result in a better performing model for predicting the behaviors of AVs more accurately.

In (Xu et al, 2017), the authors propose a system that obtains data from the sensors accompanied in a vehicle whether inbuilt or via additional installations. They suggested that real-time big data could be obtained via the internet of vehicles (IOV) network and divided their data into 2 categories. On-board data and On-road data, with the later representing the real-time status of the vehicle such as velocity, engine parameters, brake status, etc. while On-road data consists of the event information happening on the road around the vehicle obtainable from both onboard sensors and sensors from surrounding vehicles.

Strengths: The possibility of capturing such large amounts of data is highly fascinating and promises to provide so much detail into the travel experience of the AV which is expected to come in handy during training and machine learning processes.

Weaknesses: the research did not include the actual capture of the said data and execution of any form of analysis so is therefore still considered a proposed solution.

In (Alonso-Mora et al, 2017), they evaluated their solution with historical taxi data obtained from the Illinois data bank. The data was made up of 13,586 active taxi trips in Manhattan, New York, USA, containing the geographical coordinates for trip origins and destination locations as well as the date and time when the trips were executed.

Strengths: The location where the data was obtained for the research is renowned globally and is densely populated with taxis therefore the information that can be obtained to train an AV would be of the highest standard and therefore sustainable and applicable in any urban environment.

Weaknesses: The shortcomings of the dataset were their inability to capture the actual time the request was made which led them to consider the pickup time equal to the request time which is not an actual representation of the experience of the end-user with regards to the entire length of

the trip.

In (Dandl et al, 2017), the authors obtain their dataset from executing microsimulations on the A99 Federal highway in Munich, Germany. The attributes of their dataset included lane numbers, capacities, and speed limits, and data of 612 loop detectors located at 174 intersections on the road networks were used to calibrate their simulation. The simulation was done for a period of 6 hours and consisted of data from 4000 taxi fleets and 40,000 aTaxi requests.

Strengths: to ensure that the data was a proper representation of actual working day scenarios they only made use of data captured from Tuesday to Thursday during the hours of 5 am – 11 am which in most cities can be considered a peak period due to work schedule responsibilities.

Weaknesses: the behavioral features of the data used for the research was either not captured or properly outlined in the research work.

In (Bischoff et al, 2017), they applied their proposed shared taxi algorithm on a real-world dataset of taxi requests obtained from Berlin, Germany. The dataset was collected on a Tuesday in 2013 which represents a typical workday situation in the city of Berlin and was also similar to the kind of taxi fleet requests comparable to that handle by Taxi Berlin i.e. the cities' major taxi organization which is made up of an estimated 5000 vehicles. For comparison purposes they developed a base case for simulating non-shared vehicles and executed a simulation within the hours of 4 am to 4 am recording 27,336 requests handled by 4,212 vehicles.

Strengths: The volume of data collected was highly impressive for vehicle and requests volume

Weaknesses: however, a lack of insight into the actual nature of the data collected leaves a gap with regards to how the data would be applied for research and development of models capable of

training AVs.

In (Ota et al, 2015), the authors obtained the NYC Taxi trip data set from 2011 - 2012 containing over 360 million trips taken by 13000 taxis, the data was provided by Taxi & Limousine Commission (TLC) via a FOIL request.

Their data features were separated into multiple parts (taxi & passenger data), the taxi data consists of the taxi identification number, the number of passengers, current speed, list of stops, odometer reading and stop locations while the passenger data was made up of request time, pick-up and drop-off locations and a maximum number of trips to be shared.

Strengths: The data collected was extremely rich and would be capable of simulating the activities executed by an AV expected to carry-out ride-sharing activities, the fact that they went ahead to capture the latitude and longitude of the start and pick up locations gives a real-life representation of the actual travel distance covered in miles.

Weaknesses: the lack of traffic conditions on the dataset under-represents real-life traffic scenarios.

In (Kaplunovich and Yesha, 2018), the authors obtained their dataset from New York taxi data rides spanning from 2009 to 2017, their dataset was made up of multiple taxi companies separated into 3 different categories (yellow, green or fhz), the data made up of 189 files were downloaded using a shell script on GitHub with a total size of 229.5 GB.

The average file size was 2GB containing over 15 million records per file, the attributes of the dataset wer- the Start location, stop location, pick up date time, drop off date-time, number of

passengers, cab and driver identification and also the longitude and latitude of the pick-up and drop off locations.

Strengths: The data collected was of the required caliber for simulation of AV trips and although the essence of their data collection was to identify how many ride-sharing trips can be consolidated to yield a higher efficiency of taxi vehicles utilization the expectation is that the same consolidation metrics should apply to AVs expected to execute ride-sharing activities.

In (Poulsen et al, 2016), they obtained the datasets for New York green taxis and Uber rides, the green taxi data used for their research was collected from NYCTLC's website consisting of taxi trip rides from April to September 2014 adding up to over 8 million rides. The selected period was due to the need to match up with the available Uber data of the same period, the original Uber data set consisted of Uber pick-ups in New York City from April - September 2014 and January - June 2015 however only the data from the 2014 period was used for analysis containing 18.8 million Uber rides.

Weaknesses: The data features mentioned were the longitude and the latitude of the Uber rides and we believe that not enough information with regards to the data features was provided considering the amount of the data captured.

Strengths: The strengths however remain the volume of the data collected because the information that can be obtained via big data analysis for such a huge dataset could offer a close to a realistic representation of possible real-life scenarios for AVs.

In (Fényes et al, 2018), the authors created their dataset from training data collected from simulations using the high-fidelity simulation software CarSim. The simulations involved the use

of a D-class sedan passenger car weighing 1320kg, the vehicle was driven along the Michigan Waterford hill course several at different longitudinal velocities. The simulations resulted in over 2 million instances and the data attributes captured were the longitudinal acceleration, lateral acceleration, yaw rate, angular velocity of wheels, angle of the steering wheel, and the side-slip angle.

The dataset was further divided into 5 categories using the different attributes captured to run varying simulations for comparison purposes and also to identify the best performing model based on selected data attributes.

Strengths: The details of the data collected could provide valuable insights into the movement mechanics of an AV which could lead to improved safety levels and ultimately eliminate accidents.

Weaknesses: dataset was obtained from simulations that could misrepresent real-life conditions.

In (Shen et al, 2019), they evaluated their roo algorithm with a real-world trajectory of taxi data and road networks from New York City and Beijing, the Beijing urban road network contained 171,078 road network nodes and 462,178 road network edges while the New York City Road Network contained 264,346 road

network nodes and 730,100 road network edges. The Beijing taxi data was made up of 130,794 real taxi requests on March 15, 2009, the New York taxi requests were captured on May 10, 2016 contained 400,088 records.

Strengths: The results of their root algorithm on the described dataset showed that they could save 50% of mileage by 1000 vehicles handling around 700 trip requests in New York City between 7:40 am to 8:00 am with an average waiting time of 4 minutes.

Weaknesses: The data features weren't discussed extensively so there is little to no information on the exact nature the data captured so suggestions cannot be made on how the dataset can be effective for creating models for AV ride-sharing and trip requests.

In (Bathla et al, 2018), they noted that taxi ride-sharing data was unavailable so they obtained their dataset from a large scale taxi GPS covering the major urban areas of the city of Shanghai, China. The data consisted of GPS traces of 4000 taxis derived from two years covering over 120 square kilometers.

The stated features of the dataset consist of the taxi id, latitude, longitude, taxi speed, angle, trip date and time, and the Boolean status of the taxi either with a passenger or not. The data used to perform their experiments was from 5 days spanning from February 20 - 26, 2017.

Strengths: Their experiments on the dataset also showed that their TSR algorithm accommodates a 33% higher ride share among passengers while dealing with 44,241 requests.

The attributes of the data captured were of the required caliber for simulating ride-sharing activities and the expectation is that due to the similarity in taxi ride-sharing and AV ride-sharing, the same data set can be applicable on AV ride-sharing predictions and simulations.

Weaknesses: the volume of the dataset used is insufficient in this big data era, additional data would be preferred.

In (Thangaraj et al, 2017), the authors obtained their dataset from publicly available New York City taxi trip data, the data attributes mentioned where the pickup time, pickup location, and drop-off locations, for their experiments they selected one day's data (March 7th, 2013). The taxi trip data obtained for that day contained 350,000 taxi trips, although, in the original dataset they were

individual taxi trips, they were able to generate ride-sharing data from the dataset using the distance between multiple requests as the form of trip request aggregation.

The methods applied for generating ride-sharing data were highly impressive considering the lack of actual ride-sharing data mentioned previously.

Weaknesses: the expectation is that the lack of insight into more features of the dataset could impact the amount of information obtainable with regards to AV ride-sharing behavior and characteristics.

Strengths: The advantage however is that current predictive systems are created to be scalable meaning an influx of additional data features can be applied to the same model to produce better predictions and results with regards to AV behaviors.

In (Duo et al, 2017), they obtained mobile phone data collected by a traffic monitoring system (TMS) used by a large number of internet service providers (ISP) to monitor traffic scenarios in a production environment. The data contained information from all 2G and 3G users in Northern China from April 18 to April 24, the data traffic collected was generated by software installed on smartphones or via the smartphone itself, the data is sent to a close-by base station for transmission.

Strengths: The data collected meant they had a good understanding of the mobility habits of mobile phone users which is instrumental in traffic and route planning.

Weaknesses: The focus of their research was to visualize the number of individuals that open ride-sharing applications but end up not booking a ride hence the data features collected don't apply to the purpose of this research however the concept discussed remains enlightening with regards to ride-sharing users behavior.

In (Al-Abbasi et al, 2019), the authors obtained a real-world dataset of taxi trips in Manhattan, New York City, they used the data for June 2016 for training neural networks and used one week's data form July 2017 for evaluation purposes, the dataset contained numerous attributes however for the research they made use of the pick-up time, location, passenger count and drop-off locations to develop a travel demand predictive model.

Strengths: They performed 20 minutes simulations with 6000 vehicles and noted that the initial locations of the vehicles corresponded with the first 6000 rides, their results also showed that their DeepPool framework performed better than other techniques proposed in their literature review that didn't consider ride-sharing in specific regions where they expected increased demand in the future.

Weaknesses: The dataset contained a lot of features that were not utilized in this research due to reasons not outlined by the researchers however these features can still be highly instrumental in constructing a predictive model for AV trips.

In (de Lira et al, 2018), they obtained their data through enriching two publicly available four square datasets that recorded the check-ins of users on the foursquare application in New York City and Tokyo, the data captured spanned from April 12, 2012 to February 16, 2013. The data features mentioned include the time stamp of the check-in, the GPS coordinates, and a fine-grained venue-category, they noted that their dataset wasn't an outright representation of the traffic situation considering most trips don't start and end at the driver check-in locations used for their analysis.

Strengths: They noted however that the purpose of the research was to simulate activity-based ride requests and offers, therefore the data captured was sufficient for their research which showed

an increase of up to 54.69% of ride-sharing opportunities with regards to a destination-oriented approach.

Weaknesses: The dataset captured although relevant to the purpose of this research provides significantly less information in comparison to the other papers covered in this literature review and would not provide any relevant information with regards to predicting AV characteristics and behaviors in a ride/sharing environment.

Table 2. Comparative analysis of dataset review

Paper	Year	Data Volume	Data Source	Data Features	Real-world data	Features	AV Data
(Cui et al, 2018)	2019	12,509 trip records	Taxi trip trajectories from Beijing, China.	Time, location, taxi ID and taxi state	Yes	4	Yes
(Xu et al, 2017)	2018	NA	NA	NA	NA	NA	Yes
(Alonso-Mora et al, 2017)	2017	13,586 trips	Historical taxi trip data from Manhattan, New York, USA.	Date, time and geographical coordinates for pickup and drop off locations and	Yes	4	Yes
(Dandl et al, 2017)	2017	40,000 trips	A99 Federal highway in Munich, Germany.	lane numbers, lane capacities and speed limits.	Yes	NA	No
(Bischoff et al, 2017)	2017	27,336 trips	Taxi request data from Berlin, Germany.	NA	Yes	NA	Yes
(Ota et al, 2015)	2015	360,000,000 trips	Taxi trip data from New York City.	Taxi ID, passenger count, current speed, list of stops, odometer reading, stop locations, request time, pick-up and drop-off locations and maximum number of trips.	Yes	10	Yes
(Kaplunovich and Yesha, 2018)	2018	15,000,000 trips per file	New York taxi data.	Start & stop location, pick up date time, drop off date time, number of passengers, cab and driver identification, longitude and latitude of the pick-up and drop off locations.	Yes	10	Yes
(Poulsen et al, 2016)	2016	18,800,000 trips	New York green taxis data and Uber rides data.	Longitude and latitudinal ride information.	Yes	2	No
(Fényes et al, 2018)	2018	2,000,000 trips	CarSim ride simulations of Waterford Hill road racing course in Michigan, USA.	Longitudinal acceleration, lateral acceleration, yaw rate, angular velocity of wheels, angle of steering wheel and the side-slip angle.	No	6	Yes
(Shen et al, 2019)	2019	400,088 trips	Taxi data and road networks from New York City and Beijing.	NA	Yes	NA	No
(Bathla et al, 2018)	2018	44,241 trips	Taxi GPS data from the city of Shanghai China.	Taxi id, latitude, longitude, taxi speed, angle, trip date and time, passenger status.	Yes	8	Yes
(Thangaraj et al, 2017)	2017	350,000 trips	New York city taxi trip data.	Pickup time, pickup location and drop-off location.	Yes	3	No
(Duo et al, 2017)	2017	NA	Mobile phone data of 2G and 3G users in Northern China.	NA	Yes	NA	No
(Al-Abbasi et al, 2019)	2019	6000 taxis	Taxi trips data in Manhattan, New York	Pick-up time, location, passenger count and drop-off	Yes	4	Yes

			City.	locations			
(de Lira et al, 2018)	2018	NA	FourSquare mobility data in New York and Tokyo.	Check in time stamps, GPS coordinates and a fine-grained venue-category.	Yes	3	No

The dataset review provides a detailed insight into the sort of data required for building intelligent transportation systems, we focus on datasets obtained from taxi trips, autonomous vehicles, and experimental simulations. The data features utilized for each research work was also in line with the purpose of the research content, we actively laid a keen interest in research works utilizing autonomous vehicles (AV) data and simulations due to the nature of our research work.

We observe that several researchers chose to utilize New York City taxi trip data sets, and further analysis leads us to understand the rich context of the New York City taxi trip data as well as being open source and readily available.

3.1 Dataset Used

The dataset for this research was obtained from the open-source New York Taxi & Limousine Commission (New York Taxi, 2020) TLC trip record data, the data collected include green taxi trip data records for January and February 2017 and January to June 2019, the green taxi data consists of roughly 1 million records each. The features of the dataset used are made up of internal traffic factors captured by the taximeter including the trip pick-up time, drop-off time, pick-up location, drop-off location, passenger count, trip distance, total amount, payment type, and trip type. The New York City green taxi dataset offers valuable data features suitable for executing experiments for predicting trip arrival times as well as predicting shared rides. The dataset also covers a long period of time therefore, we believe that other external factors i.e. seasonal weather conditions, social events, social media feeds etc. can still be captured using deep learning we utilize a dataset spanning an entire year's period.

3.2 Data Dictionary

The following data dictionary provides a detailed description of the New York City taxi trip dataset, the data dictionary was obtained from the New York Taxi & Limousine Commission (TLC) website. The TLC taxi-zones represent the exact location in a city where a taxi ride was initiated by the taximeter e.g. Astoria, Baisley Park and Bayside are taxi zones in Queens, New York City. The presence of the location ID information offers precise detail with regards to the taxi trip pick-up location and helps ensure accurate predictive activities represent a close to real-life scenario. The trip distance presents the actual distance of the trip in miles captured by the taximeter in the taxi vehicle, and trip type information refers to whether the ride was a dispatch or street-hailing, the details of the data dictionary is publicly available on (LPEP Trip Records).

Chapter 4: Gap Analysis, Research Questions & Problem Statement

In chapter 4, we provide an intuitive description of the research works delivered in chapter 2. The gap analysis provides information on the observed gap between the related works already executed in developing intelligent transportation systems and the proposed solution presented in this research work. For the problem statement, we highlight the key problems observed by other researchers in building smart transportation systems and provide details on the current problems experienced in the process of building intelligent transportation systems, and how these problems are in line with our research questions.

4.1 Gap Analysis

Intelligent transportation systems (ITS) create an advanced technological environment that enables ride-sharing activities, seamless communication between driver and rider, safe execution of the payment of rendered service, and most essentially the logging and documentation of trip/travel data. The open-source nature of these large datasets means that valuable information and patterns can be observed and extracted using modern machine learning techniques, the research works covered in the literature review of this thesis show that extensive research has been undergone in the area of artificial intelligence, autonomous vehicles, and ride-sharing. Various problems have been identified by multiple researchers including valuable observations, however, the major problems identified based on our observations is the level of efficiency of vehicles used for daily transportation activities. The general consensus is that fewer number of vehicles could achieve the tasks needed for daily general transportation activities and possibly perform better with the implementation of effective ride-sharing schemes, ride-sharing could also lead to less cost for all parties involved i.e. passenger, driver, manufacturer and ultimately reduced carbon emissions thereby being environmentally friendly.

The study of the related works on arrival time prediction led us to the understanding that most research works focused on buses and public transit systems, more research should be done on taxi trip arrival times considering the data captured from taxi trips can be used to simulate private vehicle rides as well as autonomous vehicles simulations. The gap, however, is a continuous life cycle whereby intelligent transportation systems (ITS) need to be continuously optimized and improved using real-time data obtained from real-time rides. The emergence of 5G telecommunication networks also means that the transfer of large volumes of real-time traffic data becomes even more feasible on a global scale meaning drivers and passenger can be provided with real-time information with regards to the possibility of rides that can be shared that suits the needs of both driver and passenger based on real-time traffic conditions and demands.

4.2 Problem statement

The number of vehicles currently vying the roads in urban cities leads to extreme levels of traffic congestion, increased transportation costs, excessive fuel consumption, road accidents, carbon emissions etc. These unwanted traffic conditions ultimately impact the quality of life and productivity of the individuals inhabiting urban cities. In (Shi et al, 2016), they noted that traffic congestion is becoming an increasingly serious problem and the need to develop intelligent solutions to address congestion problems in line with the rapid development of science and technology, they highlighted that intelligent transportation systems (ITS) will be a key point for solving traffic problems. About the 2011 National Household Survey (NHS), Statistics Canada states “roughly 15.4 million Canadians commuted to work, of those who commuted, 13.5 million went to a usual place of work”. According to the details provided in (Statistic Canada, 2011), it is logical to assume a high percentage of these numbers have similar work and residential locations. This shows that there is a constant demand daily to transport individuals to their places of work,

therefore the need to create efficient intelligent transportation systems cannot be understated. The public transit systems provide a sustainable means of meeting these transportation requirements, however, the advancements of technology in the field of transportation means that new intelligent systems have to be developed to take advantage of loads of data collected daily.

4.3 Research questions

The information obtained from our problem statement highlights the major problems in building intelligent transportation systems, our aim is to propose a solution capable of meeting the daily requirements of intelligent transportation system in urban cities, we also consider cost savings to every invested party as an integral aspect of a proposed solution. The ability to accurately predict trip travel and arrival times is seen by most researchers as an important proponent of an efficient operational management system, hence, building a system capable of addressing arrival-time prediction is a key aspect of our thesis research.

The data visualization showcased in chapter 1 highlighted the similarity in pick-up and drop-off locations for several rides, therefore it is safe to assume that a high number of rides have the same pick-up and drop-off locations hence backing the need to effectively share more rides, the observations from the related works studied and our exploratory data analysis lead us to the following research questions.

1. How can AI and machine learning techniques guarantee accurate arrival/travel time predictions to support operational management systems in real time?
2. How can we predict ride-sharing accurately to enable high levels of vehicle efficiency, and promote ride sharing as a preferred option?

4.4 Research Contributions

The research questions discussed in chapter 4.3 creates a clear direction for the activities executed in this thesis research. We showcase how several machine learning and deep learning techniques can be used to predict taxi trip arrival times and predict shared rides, in an attempt to answer our research questions, our major research contributions include the proposed solutions presented in chapter 6.

For our first research contribution, we build an ensemble of deep learning approaches including a convolutional neural network (CNN) boosted using extreme gradient boosting (XGBoost), a Bagging, AdaBoost and Gradient Boosting model. The model's prediction results are stacked together using a simple numerical processing (NumPy) function and in combining the 4 models we are able to optimize weight and bias as well as reducing our error.

For our second research contribution, we propose a deep learning-based approach for predicting shared rides, we implement a similar stacking approach whereby we utilize the same ensemble boosting techniques i.e. AdaBoost, Bagging and Gradient Boosting and for deep learning we build a capsule network (CapsNet) boosted using extreme gradient boosting. We believe that the capability of our ride-sharing prediction model to accurately predict and identify shared rides shows that in a real-time environment, we can accurately predict rides that offer potential for ride-sharing. This information can be used to pinpoint the potential hotspots for ride-sharing based on the vehicle locations and execute efficient vehicle allocation policies based on the identified hotspots.

Chapter 5: Experiments

In chapter 5 of our research work, we aim to accurately predict taxi trip arrival times and also identify and predict taxi trip rides that were shared, we execute building regression models for prediction of taxi trips arrival time and classification models for predicting shared rides in our New York City taxi trip dataset. We execute multiple machine learning and deep learning experiments for the purpose of predicting taxi trip arrival times and prediction of shared rides. For arrival time predictions, we employ a number of machine learning regression techniques including linear regression, random forest regression, decision tree regression, k-nearest neighbors regression and also build artificial neural networks (ANN) and convolutional neural networks (CNN) for both arrival time prediction and ride-sharing classification for comparative analysis purposes. For prediction of shared rides, we execute classification prediction techniques such as decision tree classifiers, random forest classifiers, extreme gradient boosting (XGBoost) classifiers, long short-term memory (LSTM) neural network and Capsule Network (CapsNet). We take a look at merging some machine learning models using Scikit-Learn ensemble stacking and voting libraries and also employ ensemble boosting techniques like gradient boosting, adaptive boosting (AdaBoost), and extreme gradient boosting (XGBoost) and bootstrap aggregative (Bagging) methods. We execute these experiments on the basis that the key factors required in an efficient ride-sharing scheme are the ability to accurately and effectively predict how long each taxi trip would take and also the ability to predict shared rides allows us to identify and predict the preferred locations where to dispatch taxi vehicles based on ride-sharing possibilities.

5.1 Arrival Time Prediction Experiments

The following documentation consists of the regression experiments carried out to predict taxi trip arrival times, our arrival time experiments consist of both machine learning and deep learning

techniques to identify the preferred regression technique to be implemented as our proposed solution.

5.1.1 Data Preparation

Dataset was obtained from New York City green taxi data trips for January & February 2017, the trip records span just over 1 million records per month, the data features used for this experiment include pick-up time, drop-off time, pick-up location ID, Drop-off location ID, passenger count, trip distance, total amount, payment type and trip type.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
data_jan = pd.read_csv('data/nyc_g_jan17.csv')
```

```
data_feb = pd.read_csv('data/nyc_g_feb17.csv')
```

5.1.2 Regression Data Features

For our regression experiments data features we utilize the taxi trip pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type as our independent data features and to predict taxi trip arrival times we set the trip drop-off time as our arrival time and the target variable.

- Convert pick-up and drop-off columns for both the January and February 2017 dataset into a date-time format using pandas.

```
data_jan['lpep_pickup_datetime'] = pd.to_datetime(data_jan.lpep_pickup_datetime)
```

```
data_jan['lpep_dropoff_datetime'] = pd.to_datetime(data_jan.lpep_dropoff_datetime)
```

```
data_feb['lpep_pickup_datetime'] = pd.to_datetime(data_feb.lpep_pickup_datetime)
```

```
data_feb['lpep_dropoff_datetime'] = pd.to_datetime(data_feb.lpep_dropoff_datetime)
```

- Select the required data features.

```
data_jan = data_jan[['lpep_pickup_datetime','PULocationID','DOLocationID',
                     'passenger_count','trip_distance','total_amount','trip_type','lpep_dropoff_datetime']]
```

```
data_feb = data_feb[['lpep_pickup_datetime','PULocationID','DOLocationID',
                     'passenger_count','trip_distance','total_amount','trip_type','lpep_dropoff_datetime']]
```

- Rename pick-up and drop-off time fields.

```
data_jan = data_jan.rename(columns={'lpep_pickup_datetime': 'pickup_time'})
```

```
data_feb = data_feb.rename(columns={'lpep_pickup_datetime': 'pickup_time'})
```

```
data_jan = data_jan.rename(columns={'lpep_dropoff_datetime': 'arrival_time'})
```

```
data_feb = data_feb.rename(columns={'lpep_dropoff_datetime': 'arrival_time'})
```

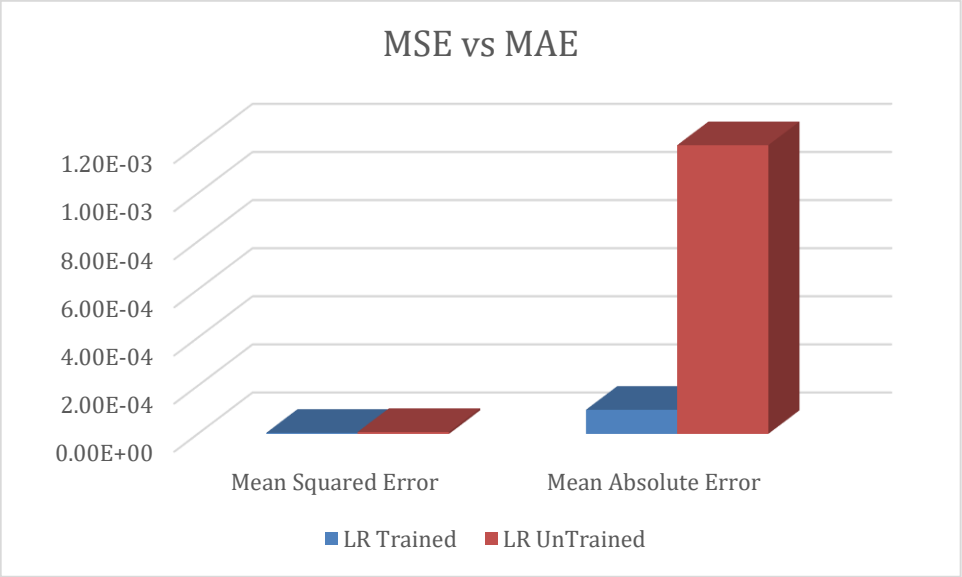
5.1.3 Experiment 1: Arrival Time prediction using Multiple Linear Regression

Linear regression is a machine learning approach that models the linear relationship between one or several informative variables, models with more than one input are considered multiple linear regression and as the name suggests, the relationship is linear in nature according to (Schmidt and Finan, 2018). We create a linear regression model using Scikit-Learn (Scikit-learn) library and for the integration of the linear regression model with our dataset, we utilize the pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variable. We split our dataset into 70% training and 30% test sets and the linear regression model is trained using our training set consisting of our independent variables. We then evaluate our linear model using Scikit-Learn regression evaluation metrics and carry-out prediction using our test set made up of drop-off/arrival-times.

Table 3. Linear regression model evaluation

Linear Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	4.606e-06	0.0001	99%
February 2017 test set predictions (Un-Trained)	99%	8.070e-06	0.0012	99%

Figure 6. Linear Regression MSE and MAE evaluation results



We evaluate our linear regression model using Scikit-Learn regression metrics, we aim to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible. From our evaluation results, our model records 99% accuracy for explained variance score (EVS) and R2 Score for the January 2017 taxi trip dataset, we also record an exponential MSE and low MAE which confirms very little error and hence high predictive accuracy. To test the performance of our linear model on an unknown dataset, we also carry out predictions using the February 2017 test set, the results are quite similar with the explained variance score and r2 score also recording 99% accuracy showing that our linear model still performs admirably when tested on untrained data.

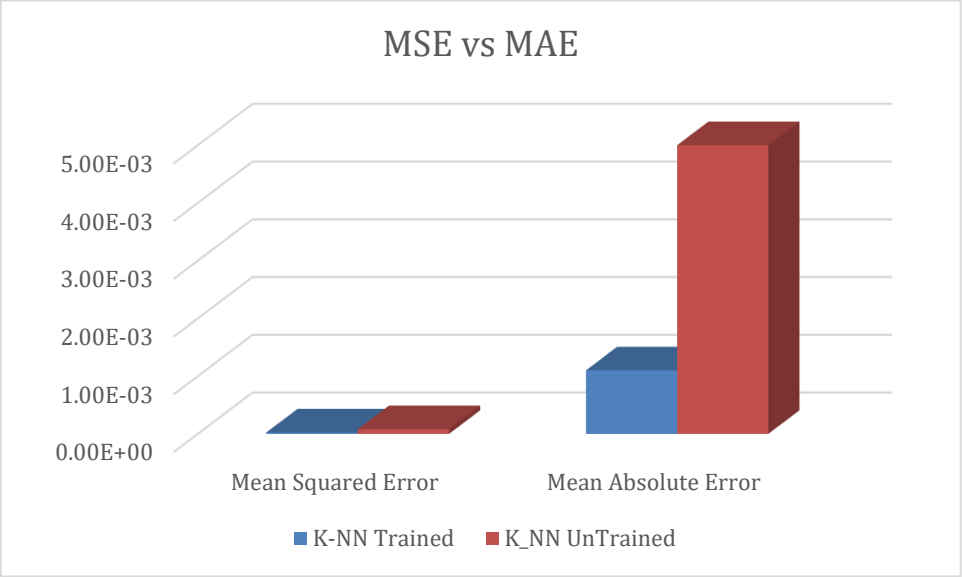
5.1.4 Experiment 2: Arrival Time Prediction using K-Nearest Neighbors (K-NN)

In (Gazalba and Reza, 2017), the authors noted that the k-nearest neighbors (k-NN) is carried out by searching a group of similar or closer objects with K as a threshold. We create a k-NN regression model using Scikit-Learn (Scikit-learn) library and for the integration of the k-NN regression model with our dataset, we utilize the pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variable. We split our dataset into 70% training and 30% test sets and the linear regression model is trained using our training set. We then evaluate our k-NN model using Scikit-Learn regression evaluation metrics and carry-out prediction using our test set made up of arrival times.

Table 4. k-NN regression model evaluation.

K-NN Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	2.478e-05	0.0011	99%
February 2017 test set predictions (Un-Trained)	99%	8.039e-05	0.0050	99%

Figure 7. k-NN Regression MSE and MAE evaluation results



We evaluate our k-NN regression model using Scikit-Learn regression metrics, we aim to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible. From our evaluation results, our model records 99% accuracy for explained variance score (EVS) and R2 Score for the January 2017 taxi trip dataset which is just a little lower than the performance of our linear model, we record a low MSE and MAE which can be considered good enough but is still outperformed by our linear model. To test the performance of our k-NN regression model on an unknown dataset, we also carry out predictions using the February 2017 test set, the results are not as accurate at the trained set but with explained variance score and r2 score recording 99% accuracy our k-NN regression model still performs admirably when tested on untrained data.

5.1.5 Experiment 3: Arrival Time Prediction using Decision Trees

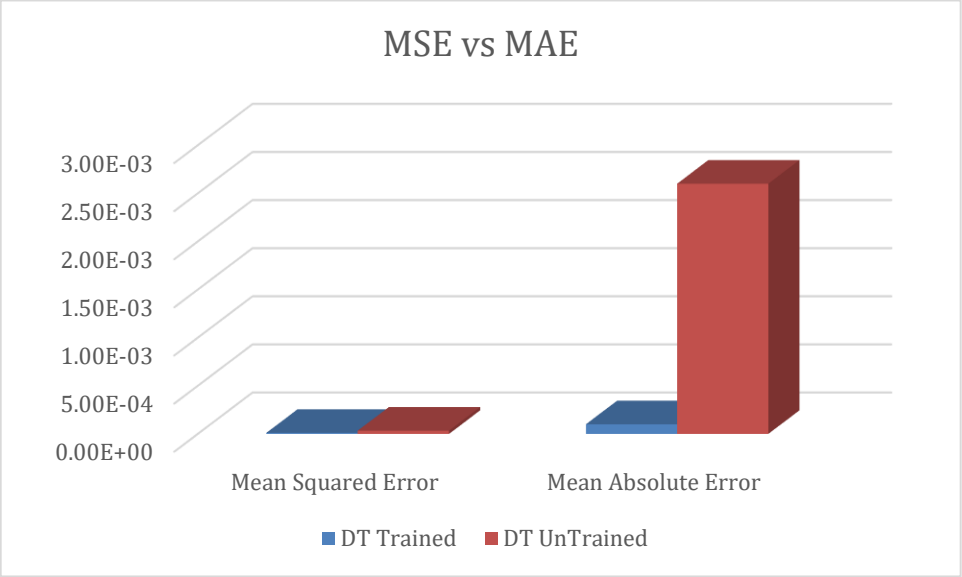
In (Nwulu, 2017), the author noted that a decision tree is a tree-like machine learning model that is used to carry out classification and numeric prediction. We create a decision tree regression model using Scikit-Learn (Scikit-learn) library and for the integration of the decision tree

regression model with our dataset, we utilize the pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variable. We split our dataset into 70% training and 30% test sets and the decision tree regression model is trained using our January 2017 taxi trip training set consisting of our independent variables. We then evaluate our decision tree regression model using Scikit-Learn regression evaluation metrics and carry-out prediction using our January and February 2017 test sets made up of arrival times of taxi trips for both months.

Table 5. Decision tree regression model evaluation

Decision Tree Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	1.044e-05	0.0001	99%
February 2017 test set predictions (Un-Trained)	99%	3.342e-05	0.0026	99%

Figure 8. Decision Trees Regression MSE and MAE evaluation results



We evaluate our decision tree regression model using Scikit-Learn regression metrics, we aim to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible, from our evaluation results, our model records 99% accuracy for explained variance score (EVS) and R2 Score for the January 2017 taxi trip dataset which is relatively similar to the performance of our linear model. We record a low MSE and MAE which can be considered good enough but still slightly underperforms compared to our linear model, our initial expectations, however, was for the linear regression model to outperform tree regression models if we consider our arrival time prediction as a linear problem. To test the performance of our decision tree regression model on an unknown dataset, we also carry out predictions using the February 2017 test set, we record similar explained variance score (EVS) of 99% and a very low mean squared error (MSE).

5.1.6 Experiment 4: Arrival Time Prediction using Random Forest Regressor

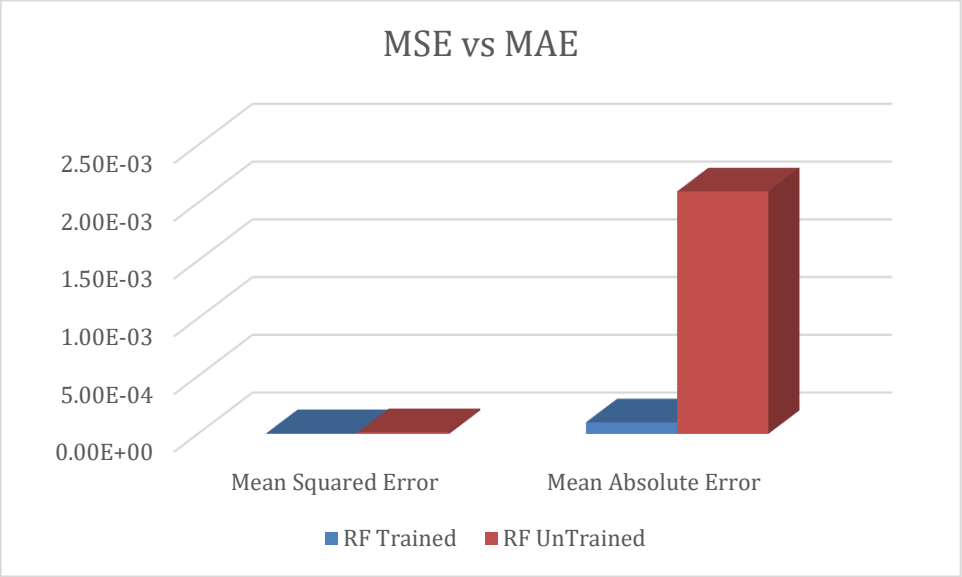
Random Forest algorithm is an ensemble of Classification and Regression Decision Trees (CARTs) which are based on the principle of a technique called bagging (Georganos et al, 2019). In (Krishni, 2018), the author highlights how random forest combines the results of multiple decision trees for a final prediction result via averaging.

We develop a random forest regression model using Scikit-Learn (Scikit-learn) ensemble library and for the integration of the random forest regression model with our dataset, we utilize the pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variable, we proceed to split our dataset into 70% training and 30% test sets and the Random forest regression model is trained using our January 2017 taxi trip training set consisting of our independent variables. We then evaluate our Random forest regression model using Scikit-Learn regression evaluation metrics and carry-out prediction using our January and February 2017 test sets made up of arrival times of taxi trips for both months.

Table 6. Random forest regression model evaluation.

Forest Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	5.636e-06	0.0001	99%
February 2017 test set predictions (Un-Trained)	99%	1.442e-05	0.0021	99%

Figure 9. Random Forest Regression MSE and MAE evaluation results.



We evaluate our random forest regression model using Scikit-Learn regression metrics, we aim to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible, from our evaluation results, our model records 99% accuracy for explained variance score (EVS) and R2 Score for the January 2017 taxi trip dataset which is relatively lower than the performance of our linear model. We record relatively similar MSE and MAE for both trained (January 2017) and untrained (February 2017) test sets in comparison to our decision tree regression model although the random forest trees performed slightly better in comparison to decision trees however our random forest model still outperformed by our linear model.

5.1.6.1 Extreme Gradient Boosting (XGBoost).

Extreme gradient boosting (XGBoost) is an advanced form of the well-known gradient boosting algorithm and framework by (Friedman, 2016). The XGBoost implementation package we used supports' various objective functions, including regression, classification, and rankings as stated by the authors in (Chen et al). Boosting is an ensemble technique whereby we add a new model to

correct errors made by older models, and then add models sequentially until no more corrections are required, In gradient boosting, we create new models to predict the errors of older models, then our final result is based on the average of all our models. XGBoost permits distributed computing and is very efficient for large training sets. For our experiment we train an XGBoost regression model with 200 weak random forest regression models serving as base estimators then train our January 2017 taxi trip training dataset, we evaluate our XGBoost using the same regression metric for comparison purposes.

Figure 10. Extreme Gradient Boosting regression (XGBoost model architecture).

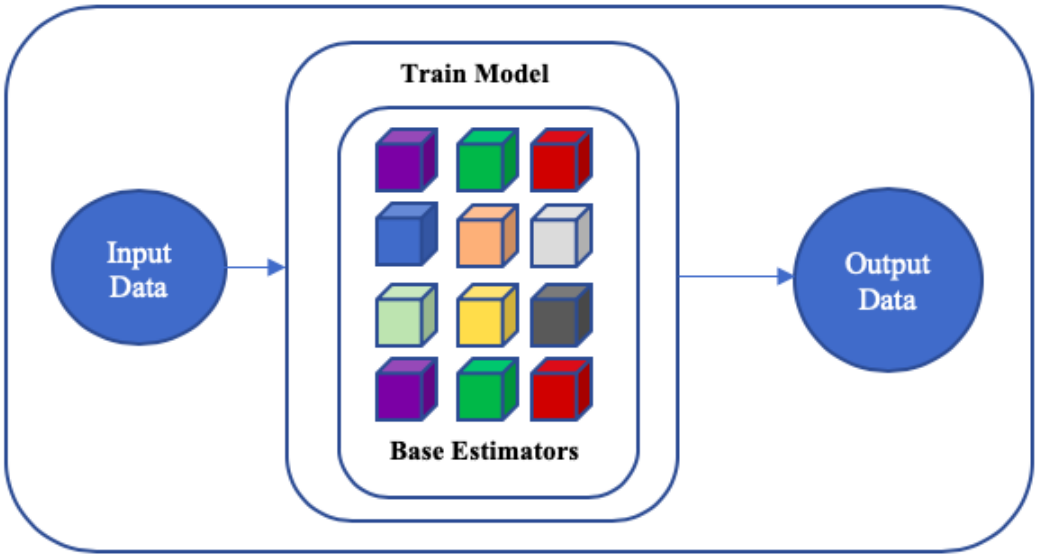
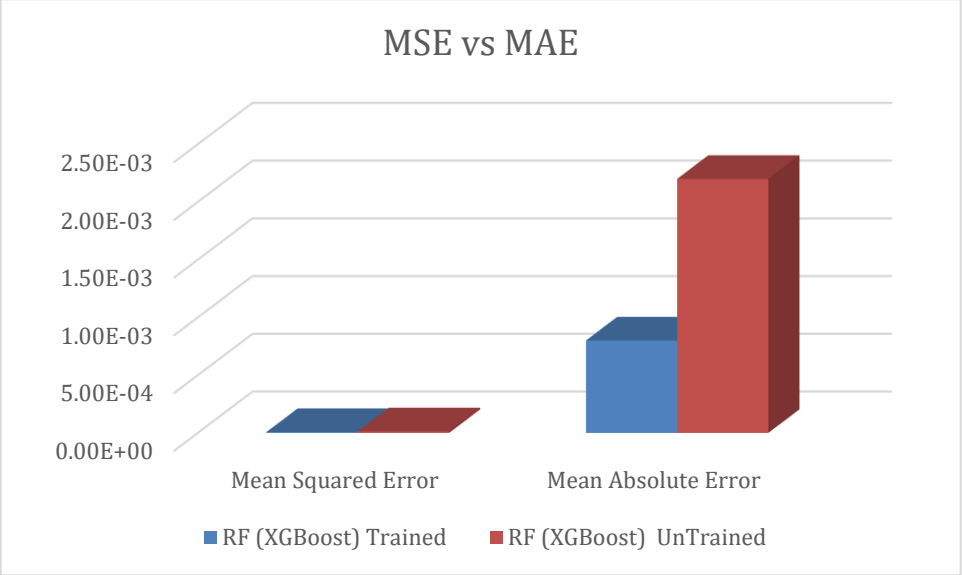


Table 7. XGBoost regression model evaluation.

XGBoost Regression Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	r2 Score
January 2017 test set predictions (Trained)	99%	6.711e-06	0.0008	99%
February 2017 test set predictions (Un-Trained)	99%	1.353e-05	0.0022	99%

Figure 11. Random Forest (XGBoost) Regression MSE and MAE evaluation results



We evaluate our XGBoost regression model using Scikit-Learn regression metrics, we aim to record a high explained variance score, and r2 score, and record a mean squared error (MSE) and mean absolute error (MAE) as low as possible, from our evaluation results, our model records 99% accuracy for explained variance score (EVS) and r2 Score for both January 2017 (trained) and February 2017 (untrained) taxi trip dataset but is still relatively lower than the performance of our linear regression model, so for our general observation is that the linear regression model outperforms all our machine learning models for arrival time prediction.

5.1.7 Experiment 5: Arrival Time prediction using Ensemble Techniques

From the research work in (Smolyakov, 2017), the authors note that ensemble methods are known as methods in which different machine or deep learning models are combined to capture more variance and improve accuracy. For comparative purposes, cross-validation and also to capture the predictive power of all our created models we develop an ensemble model combining the four (4) models from our previous experiments i.e. our decision tree regression model, random forest regression model, k-NN regression model, and the linear regression model, the models are merged

using Scikit-learn ensemble voting regression library that utilizes a uniform weight to execute predictions before carrying out averaging for a final prediction result. We train our ensemble model using the January 2017 taxi trip training set consisting of the pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount, and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variables. The implementation of this experiment is done using (Scikit-learn).

Figure 12. Ensemble model Architecture.

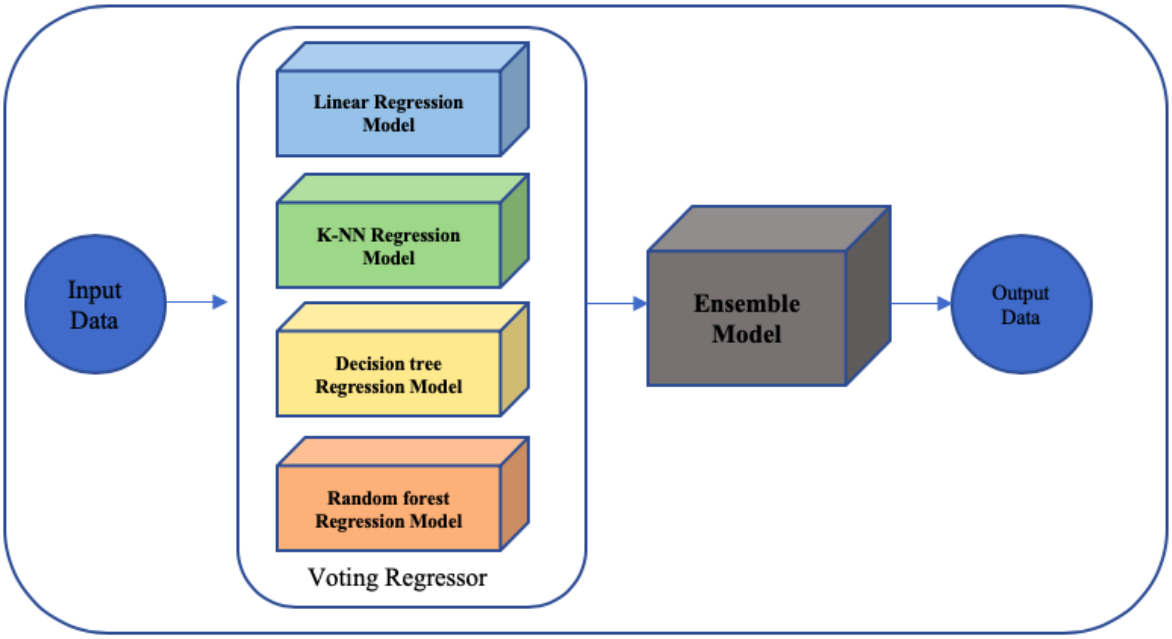


Table 8. Machine learning regression models evaluation.

Ensemble Regression Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	6.480e-06	0.0003	99%
February 2017 test set predictions (Un-Trained)	99%	1.627e-05	0.0026	99%

Decision Tree Regression Model Evaluation.				
January 2017 test set predictions (Trained)	99%	1.044e-05	0.0001	99%
February 2017 test set predictions (Un-Trained)	99%	3.342e-05	0.0026	99%
Random Forest Regression Model Evaluation.				
January 2017 test set predictions (Trained)	99%	5.636e-06	0.0001	99%
February 2017 test set predictions (Un-Trained)	99%	1.442e-05	0.0022	99%
K-NN Regression Model Evaluation.				
January 2017 test set predictions (Trained)	99%	2.479e-05	0.0011	99%
February 2017 test set predictions (Un-Trained)	99%	8.039e-05	0.0050	99%
Linear Regression Model Evaluation.				
January 2017 test set predictions (Trained)	99%	4.606e-06	0.0001	99%
February 2017 test set predictions (Un-Trained)	99%	8.071-06	0.0012	99%

Figure 13. Machine Learning regression models MSE evaluation results.

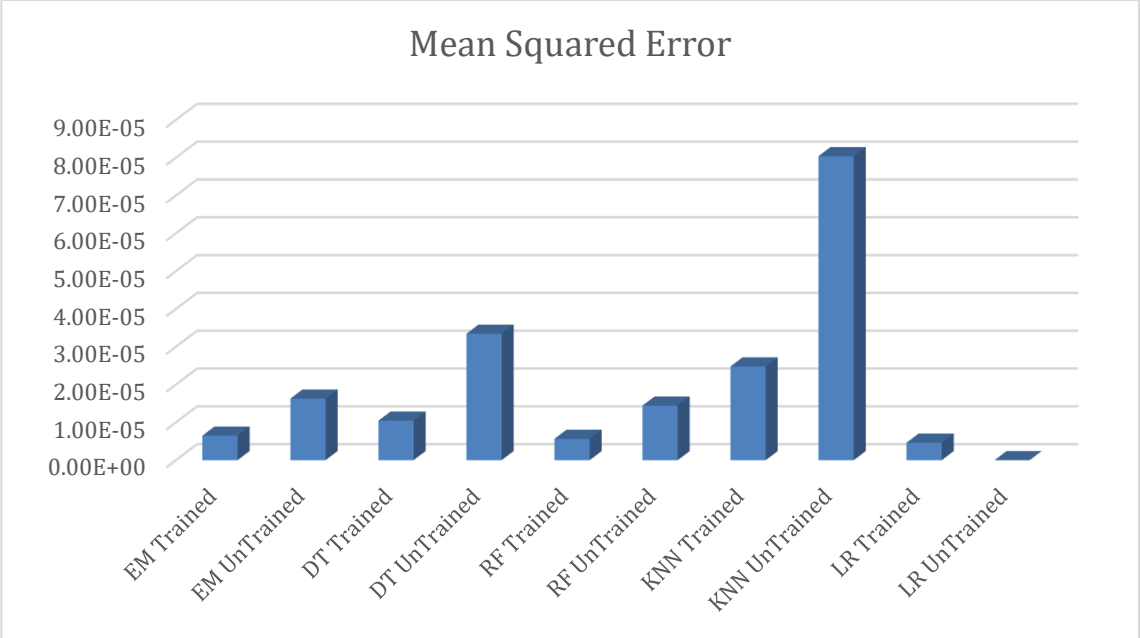
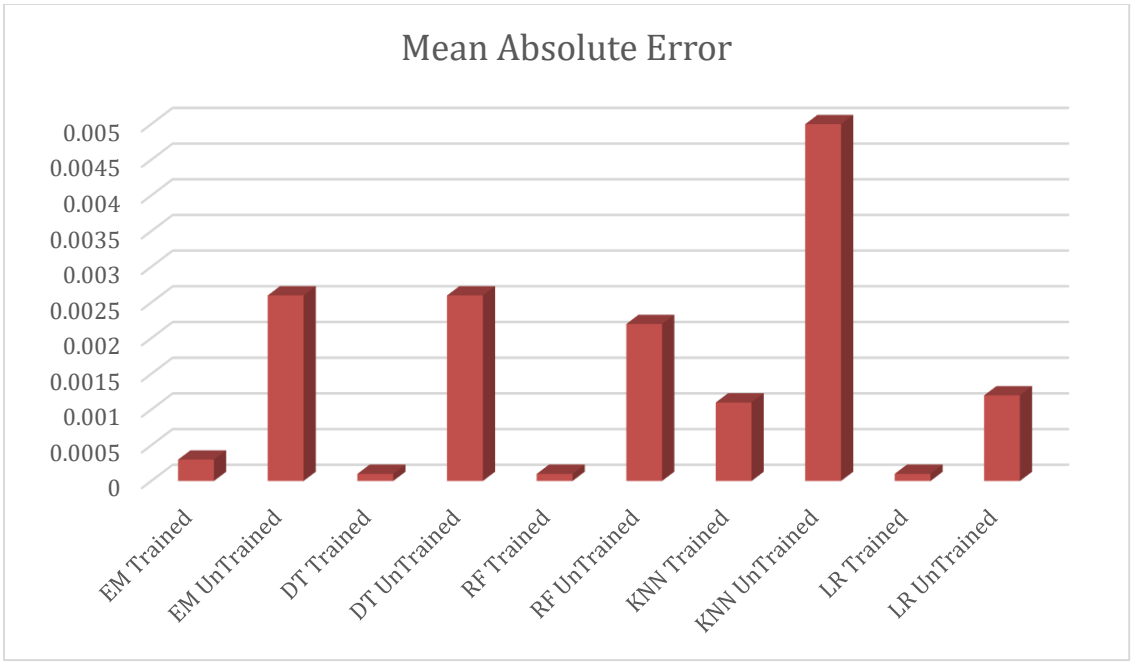


Figure 14. Machine Learning regression models MAE evaluation results.



From the comparative analysis above, we can identify a similar trend in the prediction results of our regression algorithms all the models record 99% accuracy for explained variance score and r2

score, the ensemble regression model performs relatively better that all the single models excluding the linear regression model which records lower mean squared error (MSE) and mean absolute error (MAE). The results of employing machine learning techniques for arrival time prediction are general impressive however for comparison purposes we proceed to utilize deep learning techniques by building neural networks for arrival time prediction.

5.1.8 Experiment 6: Arrival Time Prediction using an Artificial Neural Network (ANN)

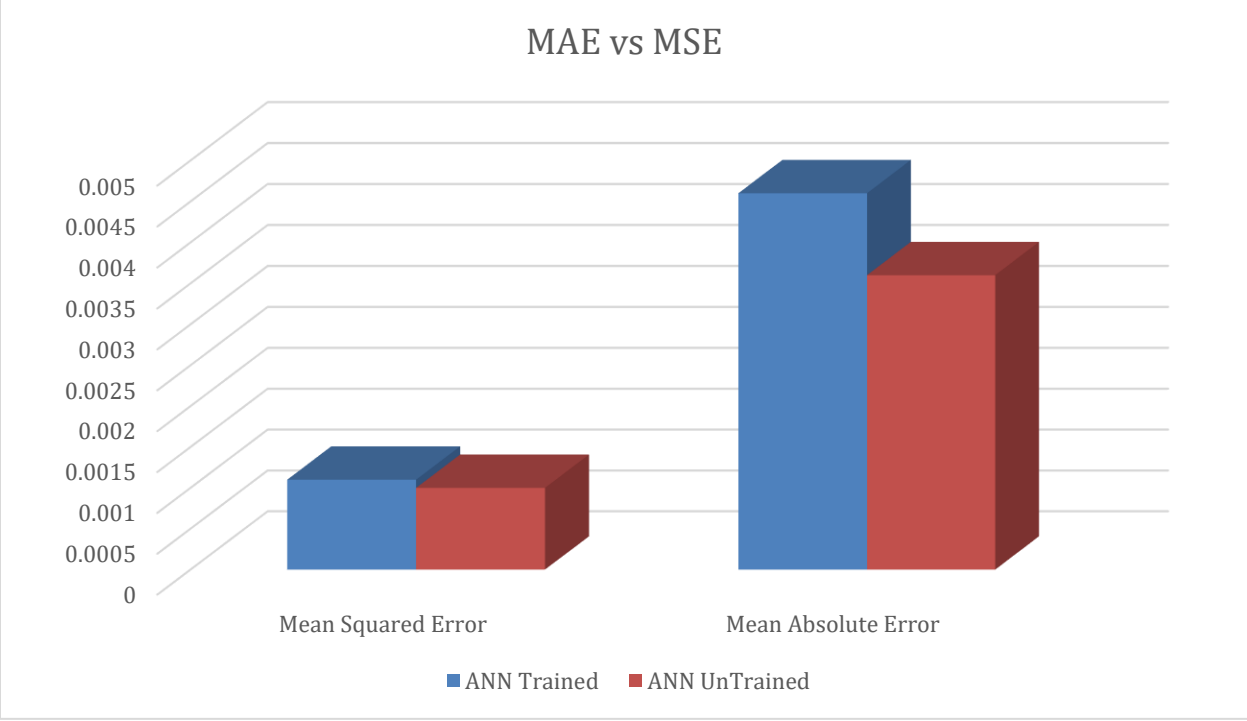
Artificial Neural networks are composed of different interconnected neurons and imitate the functionality of the human brain, inner layers other than the input and output layers are called hidden layers. Artificial neural networks feed the outputs of the first layer as the Outputs of one layer are fed to the inputs of another layer (Wickramarachchi, 2017). We build an artificial neural network (ANN) model using TensorFlow Keras libraries and for the integration of the artificial neural network (ANN) model with our dataset, we utilize the pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variable. We create our ANN model using Keras Sequential library and construct our fully connected network layers using the Dense library, we initialize our artificial neural network (ANN) model layers with a uniform initializer utilizing 32 neurons each and the input layer of our artificial neural network (ANN) consists of 7 input dimensions to accommodate our independent data features. We activate each network layer using the ReLu rectifier function and the output layer is captured using the same dense library, each network layer has a dropout rate of 20% to discard redundant independent data features observed by the ANN regression model, the model is compiled using stochastic gradient descent (adam) optimizer while loss is calculated and reduced using the mean squared error. We proceed to split our dataset into 70% training and 30% test sets, and the artificial neural network

(ANN) model is trained on 50 epochs with a batch size of 1000 records each using our January 2017 taxi trip training set consisting of our independent variables. We then evaluate our ANN model using Scikit-Learn regression evaluation metrics and carry-out prediction using our January and February 2017 test sets made up of arrival times of taxi trips for both months. The implementation of this experiment was adapted from (Kirill Eremenko at Udemy).

Table 9.Artificial Neural Network (ANN) regression model evaluation.

Artificial Neural Network (ANN) Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	98%	0.0011	0.0046	98%
February 2017 test set predictions (Un-Trained)	98%	0.0010	0.0036	98%

Figure 15. ANN regression models MAE evaluation results.



We evaluate our artificial neural network (ANN) model using Scikit-Learn regression metrics, our aim is to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible. From our evaluation results, our ANN model records 98% accuracy for explained variance score (EVS) and R2 Score for the January 2017 taxi trip dataset which is relatively lower than the performance of our machine learning model, we also record lower mean squared error (MSE) and mean absolute error (MAE) scores for both trained (January 2017) and untrained (February 2017) test sets in comparison to our machine learning regression models. Our initial expectation was for the deep learning to perform better than the machine models but the results of our artificial neural network (ANN) regression model proves otherwise, however we proceed to boosting our ANN regression model with an extreme gradient boosting (XGBoost) regression model before building a convolutional neural network (CNN) to compare performance.

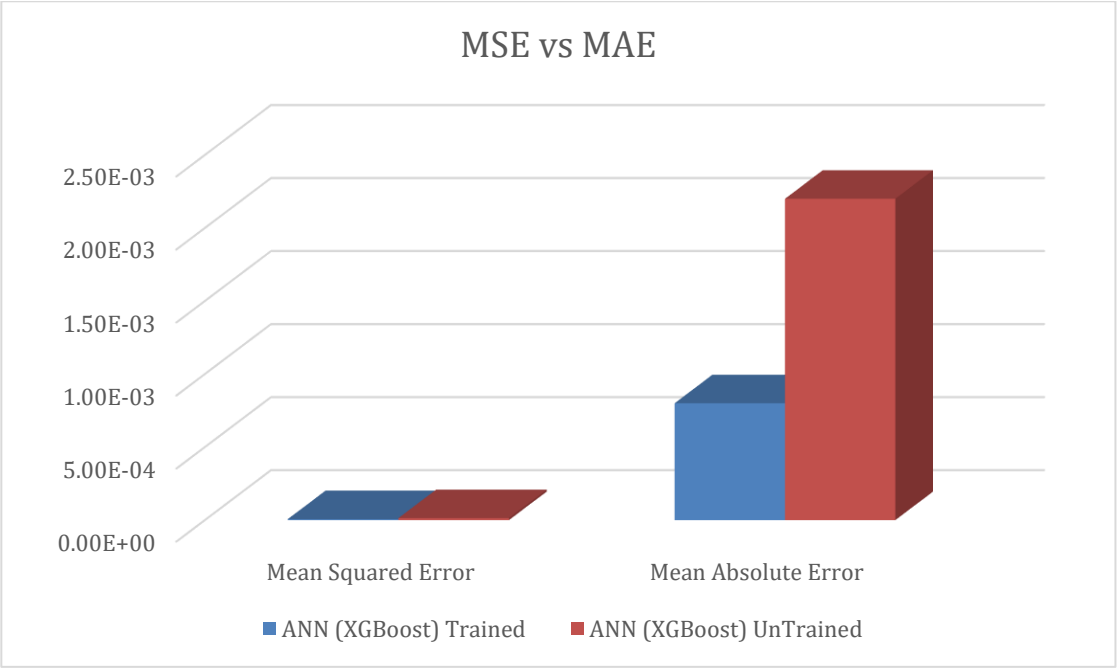
5.1.8.1 Extreme Gradient Boosting (XGBoost)

For our experiment we train an XGBoost regression model with 200 base estimators with our artificial neural network (ANN) serving as the base estimator, we then train our XGBoost regression model with the January 2017 taxi trip training dataset consisting of our independent data features and carry out predictions with our XGBoost regression model using the January and February 2017 test sets, the model is evaluated using Scikit-learn regression metrics for comparison purposes

Table 10. XGBoost regression model evaluation.

Extreme Gradient Boosting (XGBoost) Model Evaluation.				
January 2017 test set predictions (Trained)	99%	6.711e-06	0.0008	99%
February 2017 test set predictions (Un-Trained)	99%	1.352e-05	0.0022	99%

Figure 16. ANN (XGBoost) regression models MAE evaluation results.



We evaluate our XGBoost regression model using Scikit-Learn regression metrics, we aim to record a high explained variance score and r2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible. From our evaluation results, we can improve the prediction accuracy of the ANN model on every regression metric, our model records 99% accuracy for explained variance score (EVS) and r2 Score for both January 2017 (trained) and February 2017 (untrained) taxi trip dataset which is similar to the performance of our machine learning regression models, our mean squared error (MSE) score and mean absolute error (MAE) score is also similar to that of our machine learning regression models.

5.1.9 Experiment 7: Arrival Time Prediction using a Convolutional Neural Network (CNN)

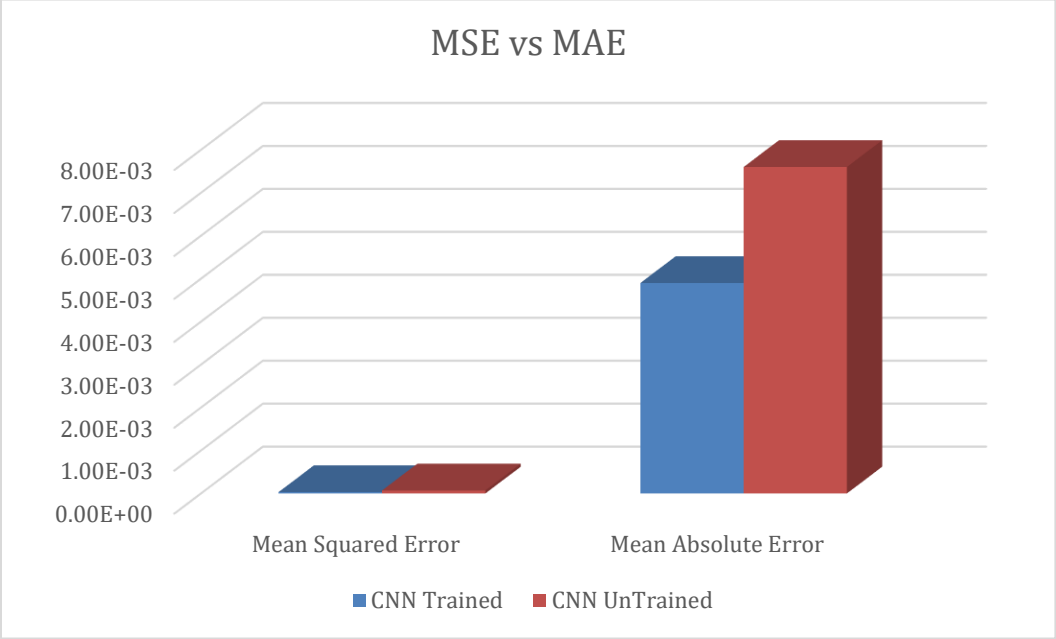
According to the authors in (Al-Saffar et al, 2017), a convolution neural network (CNN) is a multi-layer artificial neural network that has been designed for two-dimensional input data, which can be mostly imaged. Such networks utilize a mathematical operation called a “convolution” which

are complex types of operations as opposed to typical artificial neural networks using matrix multiplication. We build a convolutional neural network (CNN) model using TensorFlow Keras libraries and for the integration of the convolutional neural network (CNN) model with our dataset, we utilize the pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variable. We create our CNN model using Keras Sequential library and build our network layers using a one dimensional convolutional Keras layer, each convolutional layer consists of 32 filters with a kernel size of 2, the input shape of our convolutional neural network (CNN) is a 7 by 1, 2 dimensional matrix to accommodate our independent data features. We utilize the max pooling Keras library to add a one (1) dimensional pooling layer to down sample our input data representation and capture the key features in our dataset, we utilize Keras flatten library to convert our data features matrix into a vector and using the Keras dense layer library we add a fully connected layer consisting of 128 neurons that uses the flattened results of the pooling and convolution layers to make a final prediction. Each network layer is activated using the rectifier activation function and the output layer is captured using a fully connected dense layer, the model is compiled using stochastic gradient descent "adam" optimizer while loss is reduced using the mean squared error. We proceed to split our dataset into 70% training and 30% test sets, and the convolutional neural network (CNN) regression model is trained on 50 epochs with a batch size of 1000 records each using our January 2017 taxi trip training set consisting of our independent data variables. We then evaluate our CNN model using Scikit-Learn regression evaluation metrics and carry-out prediction using our January and February 2017 test sets made up of arrival times of taxi trips for both months. The implementation of this experiment was adapted from (Kirill Eremenko at Udemy).

Table 11. Convolutional Neural Network (CNN) regression model evaluation.

Convolutional Neural Network (CNN) Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	3.0927e-05	0.0049	99%
February 2017 test set predictions (Un-Trained)	99%	6.881e-05	0.0076	99%

Figure 17. CNN regression models MAE evaluation results.



We evaluate our convolutional neural network (CNN) model using Scikit-Learn regression metrics, we aim to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible. From our evaluation results, our CNN regression model records 99% accuracy for explained variance score (EVS) and R2 Score for the January and February 2017 taxi trip test sets which is similar to the performance of our machine learning models. We record relatively similar mean squared error (MSE) and mean absolute error (MAE) scores for both trained (January 2017) and untrained (February 2017) test

sets in comparison to our machine learning regression models, our initial expectation was for the deep learning to perform better than the machine models but the results of our artificial neural network (CNN) regression model proves otherwise. We boost our CNN regression model with an extreme gradient boosting (XGBoost) regression model for comparative purposes.

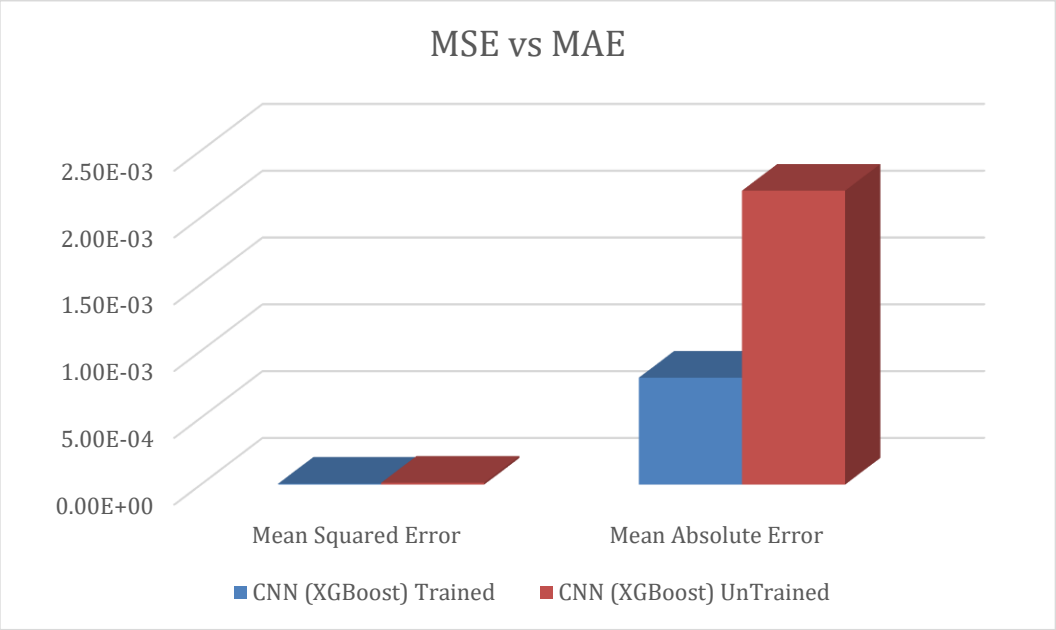
5.1.9.1 Extreme Gradient Boosting (XGBoost)

We create an XGBoost classifier model that utilizes our CNN regression model as a base estimator with 800. We train the XGBoost model using the January 2017 training set and carry out predictions using the January and February 2017 test sets, the XGBoost classifier model is evaluated using Scikit-learn classification report and confusion matrix.

Table 12. XGBoost regression model evaluation.

Extreme Gradient Boost (XGBoost) Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	6.711e-06	0.0008	99%
February 2017 test set predictions (Un-Trained)	99%	1.352e-05	0.0022	99%

Figure 18. CNN (XGBoost) regression models MAE evaluation results.



We evaluate our XGBoost regression model using Scikit-Learn regression metrics, we aim to record a high explained variance score and r2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible. From our evaluation results, we improve the prediction accuracy of the CNN regression model concerning the mean squared error (MSE) and mean absolute error (MAE), our model records 99% accuracy for explained variance score (EVS) and r2 Score for both January 2017 (trained) and February 2017 (untrained) taxi trip dataset. This is similar to the performance of the original CNN regression model and our machine learning regression models, our mean squared error (MSE) score and mean absolute error (MAE) score is also similar to that of our machine learning regression models.

5.1.10 Experiment 8: Arrival Time Prediction using a Long Short-Term Memory (LSTM) Network

LSTM's are recurrent neural networks consisting of an input layer, a recurrent hidden layer, and an output layer according to the authors in (Gers et al, 1999). In (Liu et al, 2018) they highlight

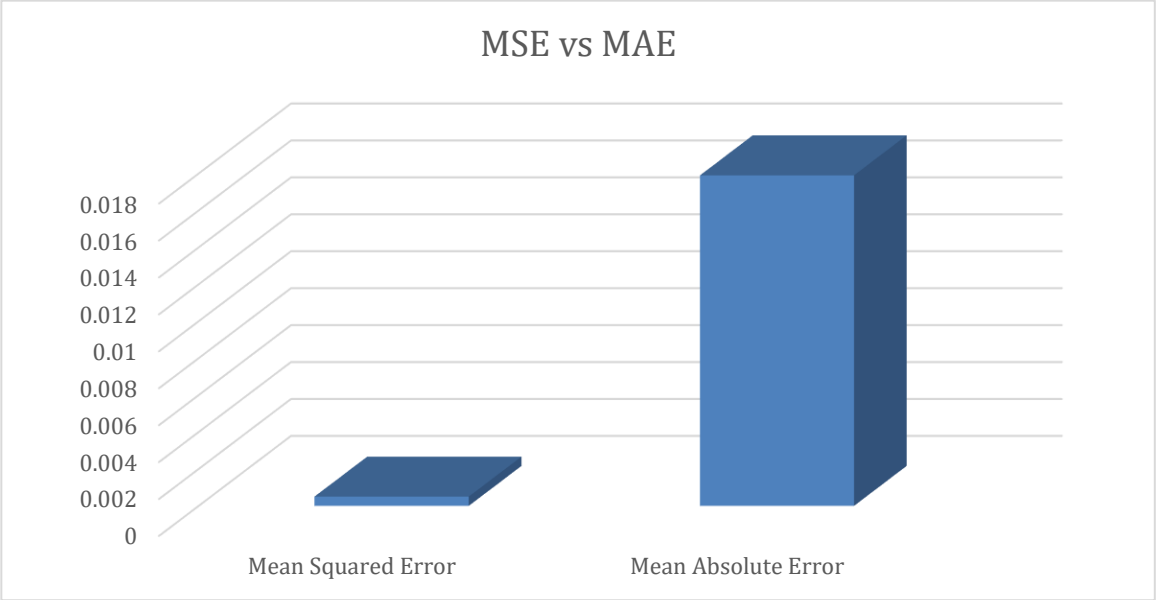
that LSTM's are composed of an input gate, a memory cell with a connection to itself, a forget and output gate, these components make up a memory block.

We build a long short-term memory (LSTM) model using TensorFlow Keras libraries and for the integration of the long short-term memory (LSTM) model with our dataset, we utilize the pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variable. We create our LSTM classifier model using Keras Sequential library and build our network layers using LSTM Keras layers, each LSTM layer consists of 50 neurons and we utilize a dropout rate of 20% for each layer, we utilize a dense fully connected layer to capture our output and the model is compiled using stochastic gradient descent "adam" optimizer which splits a dataset into multiple subsets of training cases and executes modification of the data features of each training case to identify the features that produce the best results, the loss is measured using the mean squared error. We proceed to split our dataset into 70% training and 30% test sets, and the LSTM model is trained on 50 epochs with a batch size of 1000 records each using our January 2017 taxi trip training set consisting of our independent data variables. We then evaluate our LSTM regression model using Scikit-Learn regression metrics and carry-out predictions using our January test set made up of taxi trip drop-off/arrival-times. The implementation of this experiment was adapted from (Kirill Eremenko at Udemy).

Table 13. LSTM regression model evaluation.

Long Short-Term Memory (LSTM) Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	0.0005	0.0179	99%

Figure 19. Long Short-Term Memory (LSTM) regression model evaluation.



We evaluate our long short-term memory (LSTM) model using Scikit-Learn regression metrics, we aim to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible. From our evaluation results, our LSTM regression model maintains 99% accuracy for explained variance score (EVS) and R2 Score for the January taxi trip test sets but underperforms in relation to the CNN model for mean squared error (MSE) and mean absolute error (MAE) scores.

5.1.11 Experiment 9: Arrival Time prediction using Ensemble boosting techniques

The ensemble technique employed herein utilizes a linear regression model as a week learner for an AdaBoost and Bagging model and combines an adaptive boosting (AdaBoost) regression model, bootstrap aggregation (Bagging) regression model, and gradient boosting regression model. The models are trained individually and evaluated before being combined using the averaging voting technique offered by the scikit-learn voting regressor function. We train our ensemble regression model using the January 2017 taxi trip training set consisting of the pick-up

time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variables. The implementation of this experiment was carried out using (Scikit-learn).

Figure 20. Ensemble Boosting Architecture.

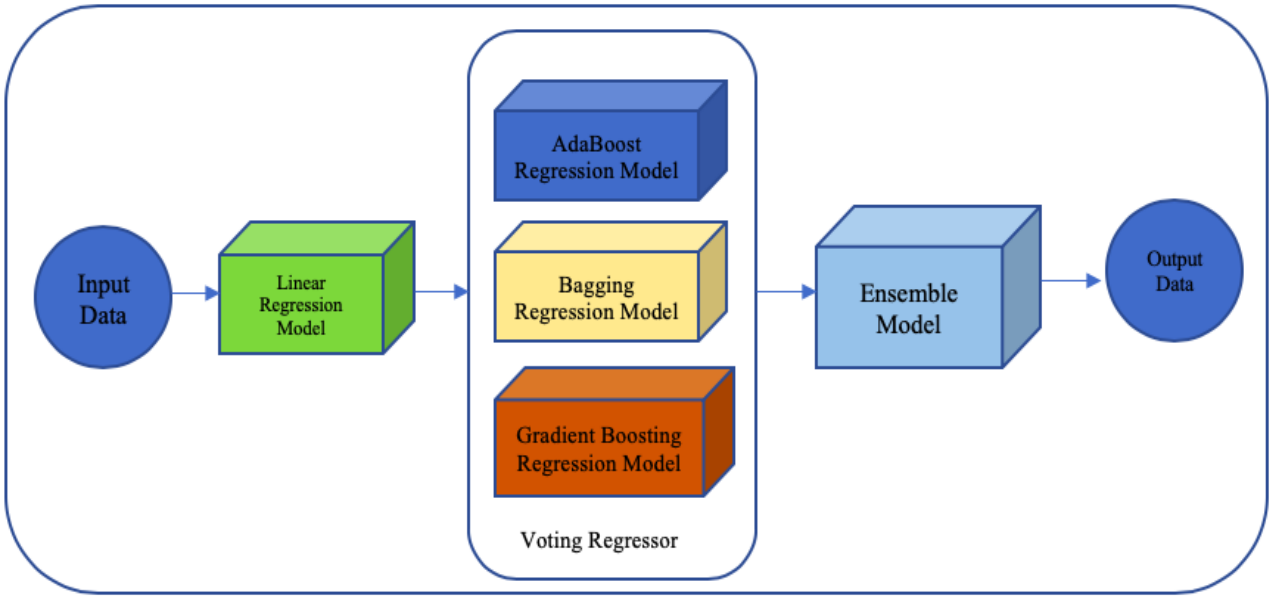


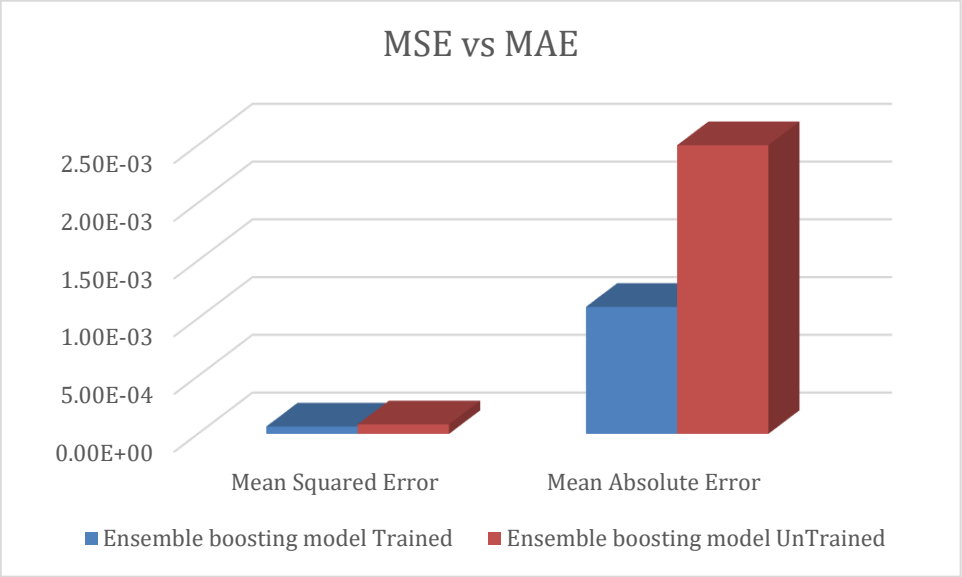
Table 14. Boosting regression models evaluation.

Models Evaluation Score.	
AdaBoost Model	99%
Bagging Model	99%
Gradient Boosting Model	99%

Table 15. Ensemble boosting regression model evaluation.

Ensemble Boosting Regression Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	6.398e-05	0.0011	99%
February 2017 test set predictions (Un-Trained)	99%	8.262e-05	0.0025	99%

Figure 21. Ensemble Boosting regression models MAE evaluation results.



We evaluate our ensemble boosting model using Scikit-Learn regression metrics, we aim to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible. From our evaluation results, our ensemble boosting regression model records 99% accuracy for explained variance score (EVS) and R2 Score for the January and February 2017 taxi trip test sets which is similar to the performance of our machine learning models, we also record relatively similar mean squared error (MSE) and mean absolute error (MAE) scores for both trained (January 2017) and untrained (February 2017) test sets in comparison to our machine learning regression models, the results of our ensemble boosting model are impressive however we proceed to apply extreme gradient boosting (XGBoost) on our ensemble boosting model to confirm if we can attain even better evaluation results.

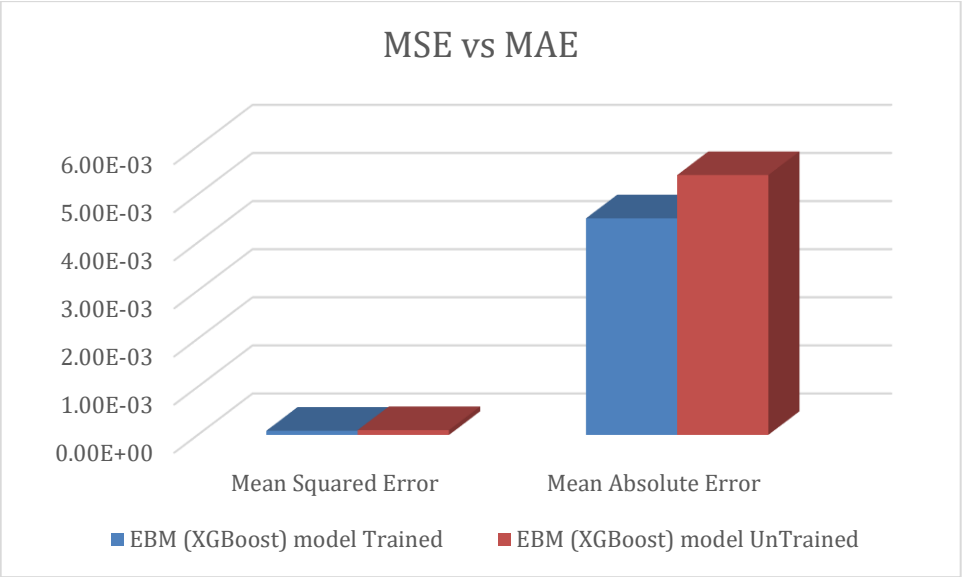
5.1.11.1 Extreme Gradient Boosting (XGBoost)

We create an XGBoost classifier model that utilizes our ensemble boosting regression model as a base estimator with 200. We train the XGBoost model using the January 2017 training set and carry out predictions using the January and February 2017 test sets, the XGBoost classifier model is evaluated using Scikit-learn classification report and confusion matrix.

Table 16. XGBoost regression model evaluation.

Extreme Gradient Boosting (XGBoost) Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	8.652e-05	0.0045	99%
February 2017 test set predictions (Un-Trained)	99%	0.0001	0.0054	99%

Figure 22. Ensemble Boosting (XGBoost) regression models MAE evaluation results.



From the above evaluation, we maintain an explained variance score (EVS) and r2 score of 99%, however, we observe a slight reduction in mean squared error (MSE) and mean absolute error (MAE) rendering the application of extreme gradient boosting on our ensemble boosting regression model as redundant.

5.2 Ride-sharing Prediction Experiments

We proceed to execute further experiments to predict shared rides, we utilize the classification data features discussed in our data Preparation section for executing classification predictions and perform experiments using a random forest classifier, decision tree classifier, artificial neural network (ANN), convolutional neural network (CNN), long short term memory (LSTM) i.e. recurrent neural network (RNN), a capsule network (CapsNet) and also employ ensemble methods and boosting techniques.

5.2.1 Data Preparation

Dataset was obtained from New York City green taxi data trips for January 2017, the trip records span just over 1 million records per month, the data features used for this experiment include pick-up time, drop-off time, pick-up location ID, Drop-off location ID, passenger count, trip distance, total amount, payment type and trip type.

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

```
data_jan = pd.read_csv('data/nyc_g_jan17.csv')
```

5.2.2 Classification Data Features

A new data field was introduced into the dataset using the passenger count field to signify whether a trip record was shared or not, therefore, trip records where passenger count is greater than one is

updated in the new field ‘shared’ as 1 and where less than equals to 1 updated as 0. This creates a new form of binary data that can be used for classification prediction for shared rides.

```
data_jan['shared'] = np.where(data_jan['passenger_count'] > 1, 1, 0)
```

```
data_jan = data_jan[['lpep_pickup_datetime', 'lpep_dropoff_datetime','PULocationID', 'DOLocationID','fare_amount',
                     'tip_amount','trip_distance', 'total_amount', 'payment_type', 'trip_type', 'shared']]
```

5.2.3 Experiment 10: Ride-sharing Prediction using Random Forest classification

As discussed in experiment 4, random forest can be used to execute both classification and regression algorithms. For experiment 5, we utilize a random forest classifier to predict whether a ride was shared or not, we utilize our classification data features discussed in our data preparation section. Our independent variables consist of the trip pick-up time, drop-off time, pick-up location ID, Drop-off location ID, tip amount, trip distance, total amount, payment type and trip type while our target variable is the newly created column signifying if the ride was shared or not represented in a binary form where shared rides are represented as 1 and ride not shared represented as 0. We split our dataset into 70% training and 30% test sets and the Random forest classifier model is trained using our January 2017 taxi trip training set consisting of our independent variables. We then carry-out predictions using our January 2017 test sets made up of arrival times of taxi trips for the month and evaluate our Random forest classifier model using the Scikit-Learn classification report and confusion matrix. The implementation of this experiment was carried out using (Scikit-learn).

Table 17. Random forest classifier model classification report.

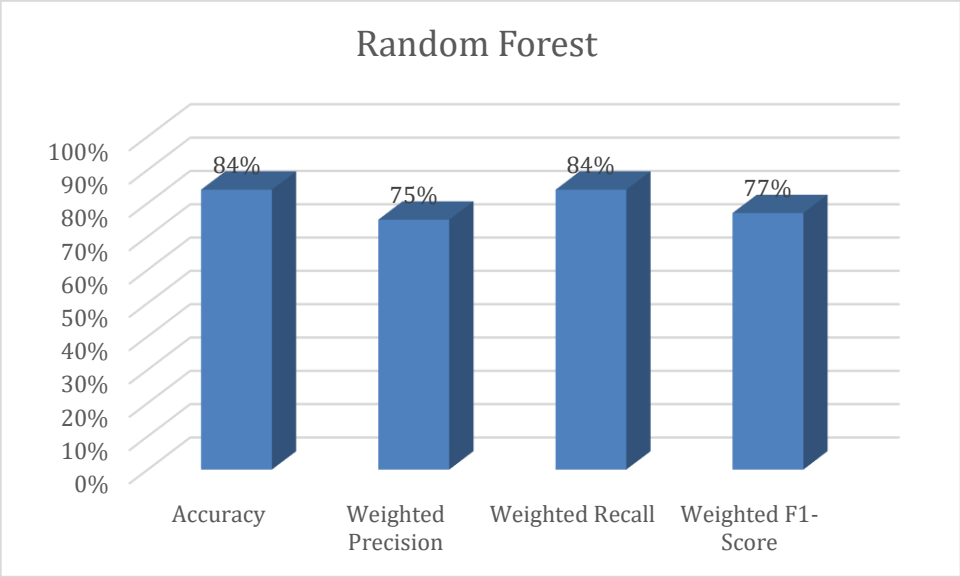
Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	99%	91%	270,197
1	28%	1%	2%	50,882
Accuracy			84%	321,079

Macro Avg	56%	50%	47%	321,079
Weighted Avg	75%	84%	77%	321,079

Table 18. Random forest classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 268,574	FN = 1,623
Actual: NO	FP = 50,247	TN = 635

Figure 23. Random Forest classifier model January evaluation results.



We evaluate our random forest classification model using Scikit-Learn confusion matrix and classification report, we aim to record a high accuracy, precision, recall, and F1 scores, the precision score is a measurement of the ability of our model to accurately not classify our positive target labels as negative labels, recall refers to the ability of our model to capture all our positive labels and f1-score is the harmonic mean of the precision and recall, from our evaluation results, the classification report shows that our random forest classifier model records 84% accuracy, the

weighted precision of 75% is relatively low and performs below our expectations, we also record an f1-score of 77%. In a ride-sharing perspective our random forest model only predicts 635 shared rides for January 2017, in an attempt to boost our prediction results, we proceed to apply extreme gradient boosting (XGBoost).

5.2.3.1 Extreme Gradient Boosting (XGBoost).

As discussed in experiment 4 we also employ extreme gradient boosting techniques for our classification algorithm, we create an XGBoost classifier model that utilizes our random forest classifier model as a base estimator with 400 estimators. We train the XGBoost model using the January 2017 training set and carry out predictions using the January 2017 test sets, the XGBoost classifier model is evaluated using Scikit-learn classification report and confusion matrix.

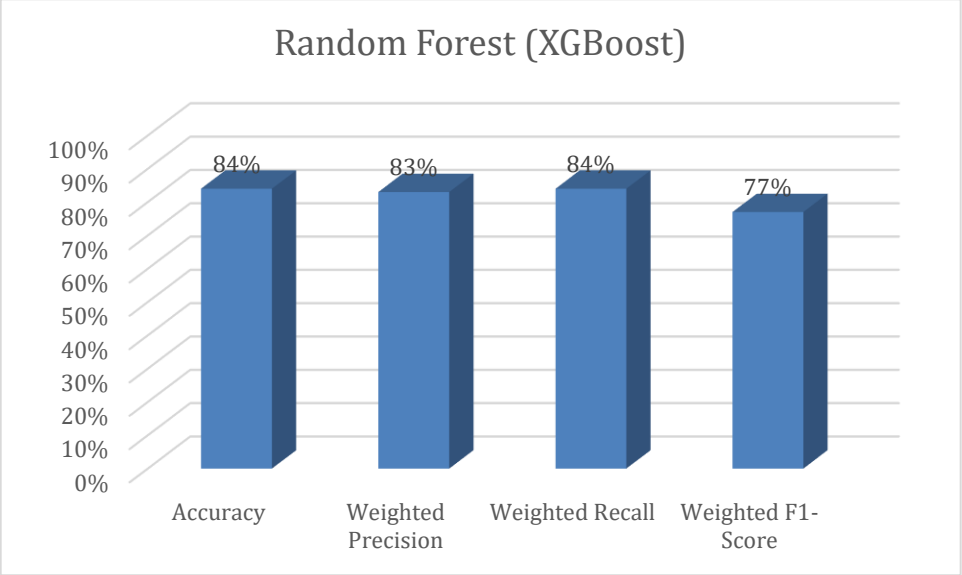
Table 19. Extreme gradient boosting (XGBoost) classifier model classification report.

Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	100%	91%	270,197
1	79%	0%	0%	50,882
Accuracy			84%	321,079
Macro Avg	82%	50%	46%	321,079
Weighted Avg	83%	84%	77%	321,079

Table 20. Extreme gradient boosting (XGBoost) classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 270,115	FN = 82
Actual: NO	FP = 50,727	TN = 155

Figure 24. Random Forest (XGBoost) classifier model January evaluation results.



We evaluate our random forest XGBoost classification model using Scikit-Learn confusion matrix and classification report, our XGBoost classifier model maintains the 84% accuracy, we record a higher weighted precision of 83% for the trained test set but we are unable to improve on the f1-score of 77%. We are also unable to predict more shared rides.

5.2.4 Experiment 11: Ride-sharing Prediction using an Artificial Neural Network (ANN) Classifier

We build an artificial neural network (ANN) classifier model using TensorFlow Keras libraries and for the integration of the artificial neural network (ANN) model with our dataset, we utilize the trip pick-up time, drop-off time, pick-up location ID, Drop-off location ID, fare amount, tip amount, trip distance, total amount, payment type and trip type while our target variable is the newly created column signifying if the ride was shared or not represented in a binary form where shared rides are represented as 1 and rides not shared represented as 0. We initialize our ANN classifier model using Keras Sequential library and construct our network layers using the Dense

library for a fully connected layer, we initialize our artificial neural network (ANN) model layers with a uniform initializer utilizing five (5) neurons each and the input layer of our artificial neural network (ANN) consists of nine (9) input dimensions to accommodate our independent data features, we activate each network layer using the ReLu rectifier function and the output layer is captured using the dense library, each network layer has a dropout rate of 20% to discard redundant independent data features observed by the ANN classifier model to address overfitting, for the classifiers output layer we utilize a sigmoid activation function because our target variable is between 1 and 0, the model is compiled using stochastic gradient descent "adam" optimizer which splits a dataset into multiple subsets of training cases and executes modification of the data features of each training case in order to identify the features that produce the best results, stochastic gradient descent has proven to be an efficient algorithm for modern day intelligent systems and finally we generate model loss using binary cross entropy for our binary classification algorithm. We proceed to split our dataset into 70% training and 30% test sets, and the artificial neural network model is trained on 50 epochs with a batch size of 1000 records each using our January 2017 taxi trip training set consisting of our independent variables. We then evaluate our ANN classifier model using the Scikit-Learn classification report and confusion matrix and carry-out predictions using our January 2017 test sets made up of shared ride multi-label information. The implementation of this experiment was adapted from (Kirill Eremenko at Udemy).

Model Evaluation using Scikit-learn libraries Accuracy Score, Classification Report and Confusion Matrix

Table 21. Artificial Neural Network (ANN) classifier model classification report.

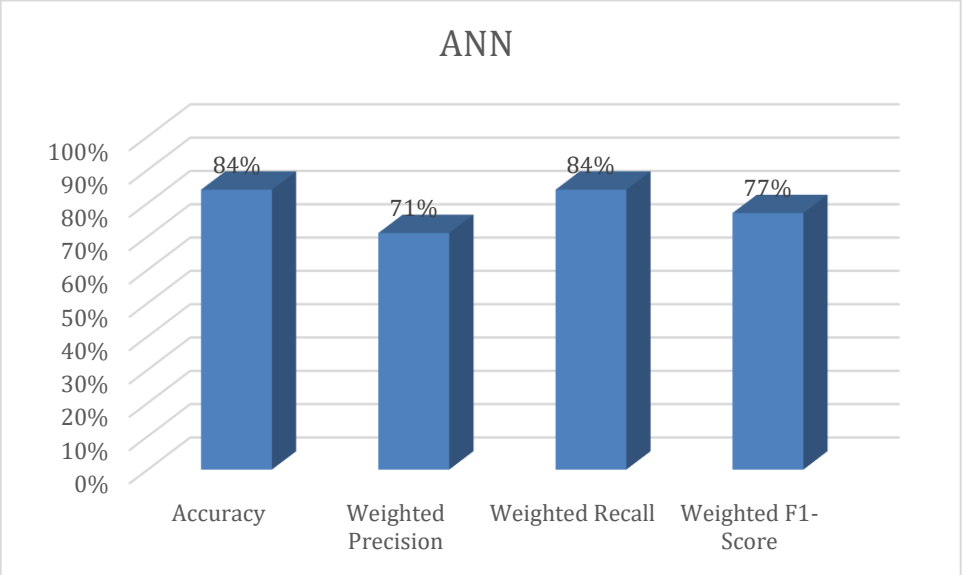
Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	100%	91%	270,423

1	0%	0%	0%	50,656
Accuracy			84%	321,079
Macro Avg	42%	50%	46%	321,079
Weighted Avg	71%	84%	77%	321,079

Table 22. Artificial Neural Network (ANN) classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 270,423	FN = 0
Actual: NO	FP = 50,656	TN = 0

Figure 25. Artificial Neural Network (ANN) classifier model evaluation results.



We evaluate our ANN classification model using Scikit-Learn confusion matrix and classification report, we aim to record a high accuracy, precision, recall, and F1 scores, the precision score is a measurement of the ability of our model to accurately not classify our positive target labels as negative labels, recall refers to the ability of our model to capture all our positive labels and f1-score is the harmonic mean of the precision and recall, from our evaluation results, the

classification report shows that our ANN classifier model records 84% accuracy, the weighted precision of 71% is lower than our random forest model performing well below our expectations, we maintain our weighted recall score of 84% and f1-score of 77%. Our ANN classifier model is unable to accurately predict any shared rides therefore in an attempt to boost our evaluation results, we proceed to apply extreme gradient boosting (XGBoost).

5.2.4.1 Extreme Gradient Boosting (XGBoost)

We create an XGBoost classifier model that utilizes our ANN classifier model as a base estimator with 200 estimators. We train the XGBoost model using the January 2017 training set and carry out predictions using the January test sets, the XGBoost classifier model is evaluated using scikit-learn classification report and confusion matrix.

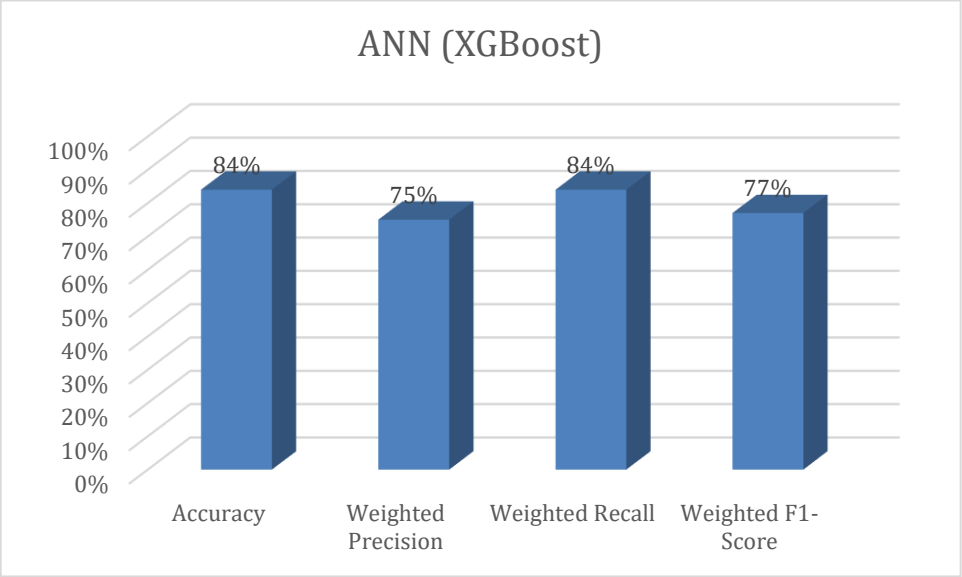
Table 23. ANN (XGBoost) classifier model classification report.

Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	99%	91%	270,423
1	26%	2%	4%	50,656
Accuracy			84%	321,079
Macro Avg	55%	51%	48%	321,079
Weighted Avg	75%	84%	77%	321,079

Table 24. ANN (XGBoost) classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 267,162	FN = 3,261
Actual: NO	FP = 49,525	TN = 1,131

Figure 26. Artificial Neural Network (ANN) (XGBoost) classifier model evaluation results.



We evaluate our XGBoost classification model using the Scikit-Learn confusion matrix and classification report, our XGBoost classifier model maintains the 84% accuracy, we record a higher weighted precision of 75% but we are still unable to improve on the f1-score of 77%. It is worth noting that our boosted model can predict 1,131 shared rides.

5.2.5 Experiment 12: Ride-sharing Prediction using a Convolutional Neural Network (CNN) classifier

We build a convolutional neural network (CNN) model using TensorFlow Keras libraries and for the integration of the convolutional neural network (CNN) model with our dataset, we utilize the trip pick-up time, drop-off time, pick-up location ID, Drop-off location ID, fare amount, tip amount, trip distance, total amount, payment type and trip type while our target variable is the newly created column signifying if the ride was shared or not represented in a binary form where shared rides are represented as 1 and rides not shared represented as 0. We create our CNN classifier model using Keras Sequential library and build our network layers using a one dimensional convolutional Keras layer, each convolutional layer consists of 32 filters with a kernel

size of 3, the input shape of our convolutional neural network (CNN) is a 10 by 1, 2 dimensional matrix to accommodate our independent data features, we utilize the max-pooling Keras library to add a one (1) dimensional pooling layer to down sample our input data representation and capture the key features in our dataset, we utilize Keras flatten library to convert our data features matrix into a vector and using the Keras dense layer library we add a fully connected layer consisting of 128 neurons that uses the flattened results of the pooling and convolution layers to make a final prediction, each network layer is activated using the rectifier activation function and the output layer is captured using a dense layer, the model is compiled using stochastic gradient descent "adam" optimizer while loss is reduced using the mean squared error. We proceed to split our dataset into 70% training and 30% test sets and the convolutional neural network (CNN) classification model is trained on 50 epochs with a batch size of 1000 records each using our January 2017 taxi trip training set consisting of our independent data variables. We then evaluate our CNN classifier model using the Scikit-Learn classification report and confusion matrix and carry-out prediction using our January 2017 test sets made up of shared trips multi-label information. The implementation of this experiment was adapted from (Kirill Eremenko at Udemy).

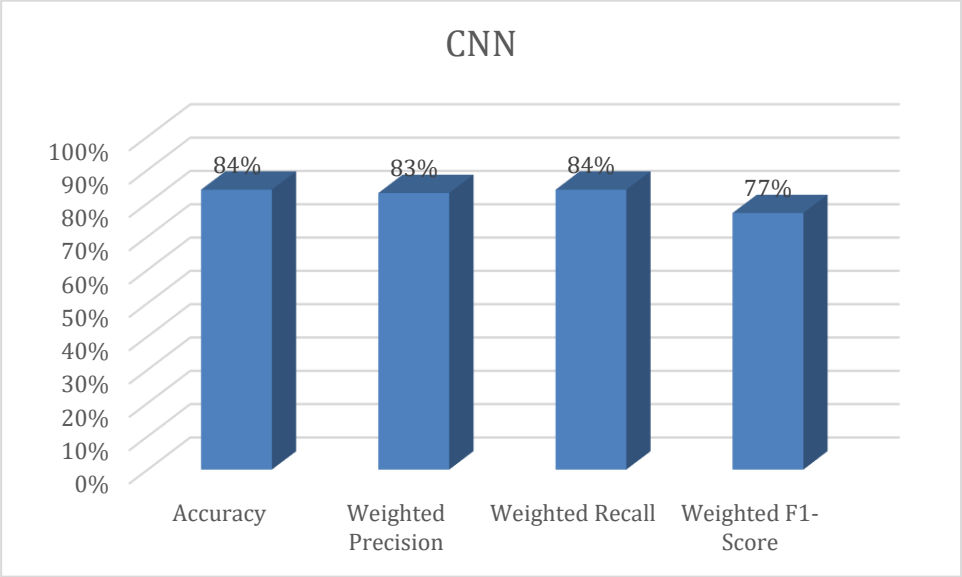
Table 25. Convolutional Neural Network (CNN) classifier model classification report.

Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	100%	91%	270,197
1	79%	0%	1%	50,882
Accuracy			84%	321,079
Macro Avg	82%	50%	46%	321,079
Weighted Avg	83%	84%	77%	321,079

Table 26. Convolutional Neural Network (CNN) classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 270,163	FN = 34
Actual: NO	FP = 50,754	TN = 128

Figure 27. Convolutional Neural Network (CNN) classifier model evaluation.



We evaluate our CNN classification model using Scikit-Learn confusion matrix and classification report, we aim to record a high accuracy, precision, recall, and F1 scores, the precision score is a measurement of the ability of our model to accurately not classify our positive target labels as negative labels, recall refers to the ability of our model to capture all our positive labels and f1-score is the harmonic mean of the precision and recall, from our evaluation results, the classification report shows that our CNN classifier model records 84% accuracy, the weighted precision of 83% is higher than our random forest and ANN models, we maintain our weighted recall score of 84% and f1-score of 77%. Our CNN is only able to predict 128 shared rides therefore

in an attempt to boost our evaluation results, we proceed to apply extreme gradient boosting (XGBoost).

5.2.5.1 Extreme Gradient Boosting (XGBoost)

We create an XGBoost classifier model that utilizes our CNN classifier model as a base estimator with 200 estimators. We train the XGBoost model using the January 2017 training set and carry out predictions using the January test set, the XGBoost classifier model is evaluated using the Scikit-learn classification report and confusion matrix.

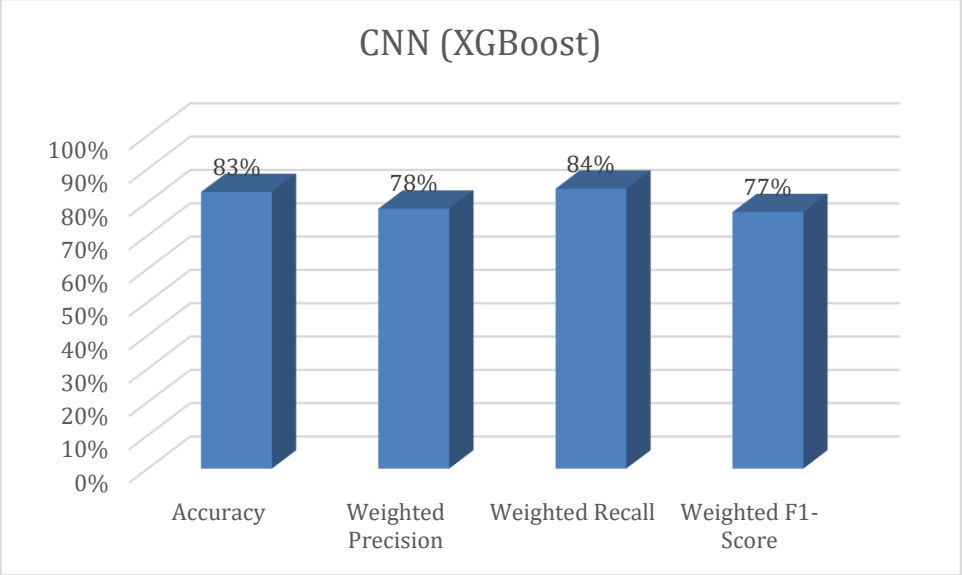
Table 27. CNN (XGBoost) classifier model classification report.

Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	100%	91%	270,197
1	44%	1%	1%	50,882
Accuracy			83%	321,079
Macro Avg	64%	50%	46%	321,079
Weighted Avg	78%	84%	77%	321,079

Table 28. CNN (XGBoost) classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 269,739	FN = 458
Actual: NO	FP = 50,528	TN = 354

Figure 28. Convolutional Neural Network (CNN) (XGBoost) classifier model evaluation.



We evaluate our XGBoost classification model using Scikit-Learn confusion matrix and classification report, our XGBoost classifier model results in lower accuracy of 83%, we also record a lower weighted precision of 78% and we are still unable to improve on the f1-score of 77% and similar to the ANN classifier model performs extremely poorly and predicts less shared rides rendering our boosting attempts redundant.

5.2.6 Experiment 13: Ride-sharing Prediction using a Long Short-Term Memory (LSTM) Classifier

We build a long short term memory (LSTM) model using TensorFlow Keras libraries and for the integration of the long short term memory (LSTM) model with our dataset, we utilize the trip pick-up time, drop-off time, pick-up location ID, Drop-off location ID, fare amount, tip amount, trip distance, total amount, payment type and trip type while our target variable is the newly created column signifying if the ride was shared or not represented in binary form where shared rides are represented as 1 and rides not shared represented as 0, we create our LSTM classifier model using Keras Sequential library and build our network layers using LSTM Keras layers, each LSTM layer

consists of 50 neurons and we utilize a dropout rate of 20% for each layer, we utilize a dense fully connected layer to capture our output and the model is compiled using stochastic gradient descent "adam" optimizer which splits a dataset into multiple subsets of training cases and executes modification of the data features of each training case in order to identify the features that produce the best results, loss is measured using the binary cross entropy for multi-label classification. We proceed to split our dataset into 70% training and 30% test sets, and the LSTM classification model is trained on 30 epochs with a batch size of 1000 records each using our January 2017 taxi trip training set consisting of our independent data variables. We then evaluate our LSTM classifier model using the Scikit-Learn classification report and confusion matrix and carry-out prediction using our January test set made up of shared trips multi-label information. The implementation of this experiment was adapted from (Kirill Eremenko at Udemy).

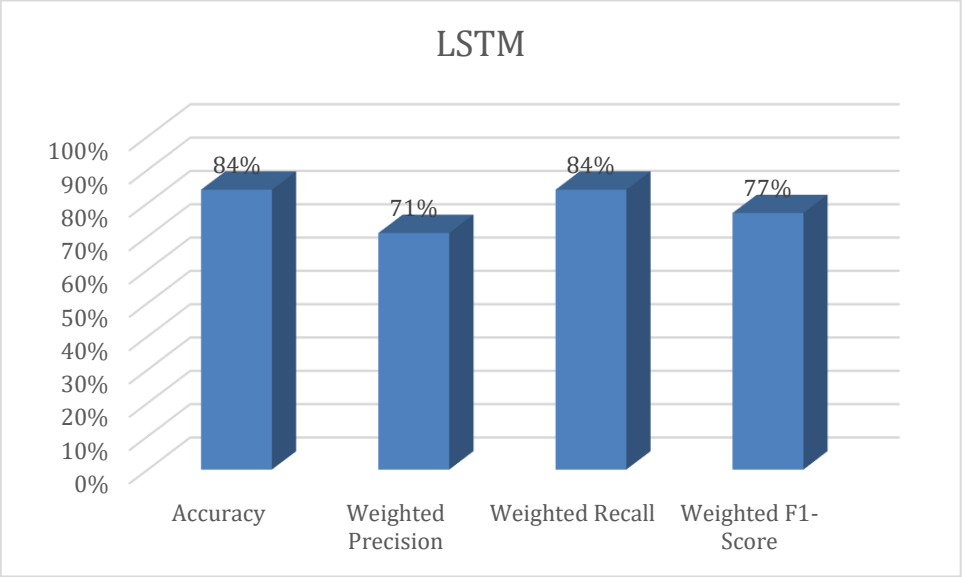
Table 29. Long Short-Term Memory (LSTM) classifier model classification report.

Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	100%	91%	126,474
1	0%	0%	0%	23,526
Accuracy			84%	150,000
Macro Avg	42%	50%	46%	150,000
Weighted Avg	71%	84%	77%	150,000

Table 30. Long Short-Term Memory (LSTM) classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 126,474	FN = 0
Actual: NO	FP = 23,526	TN = 0

Figure 29. Long Short-Term Memory (LSTM) classifier model evaluation.



We evaluate our LSTM classification model using Scikit-Learn confusion matrix and classification report, we aim to record a high accuracy, precision, recall, and F1-score. The precision score is a measurement of the ability of our model to accurately not classify our positive target labels as negative labels, recall refers to the ability of our model to capture all our positive labels and f1-score is the harmonic mean of the precision and recall, from our evaluation results, the classification report shows that our LSTM classifier model records 84% accuracy, the weighted precision is a lowly 71% similar to our ANN classifier model, we maintain our weighted recall score of 84% and f1-score of 77%. Our LSTM model is unable to predict a single shared-ride hence in an attempt to boost our evaluation results, we proceed to apply extreme gradient boosting (XGBoost).

5.2.6.1 Extreme Gradient Boosting (XGBoost)

We create an XGBoost classifier model that utilizes our LSTM classifier model as a base estimator with 200 estimators and set our learning rate to 0.5. We train the XGBoost model using the January 2017 training set and carry out predictions using the January and February 2017 test sets, the

XGBoost classifier model is evaluated using Scikit-learn classification report and confusion matrix.

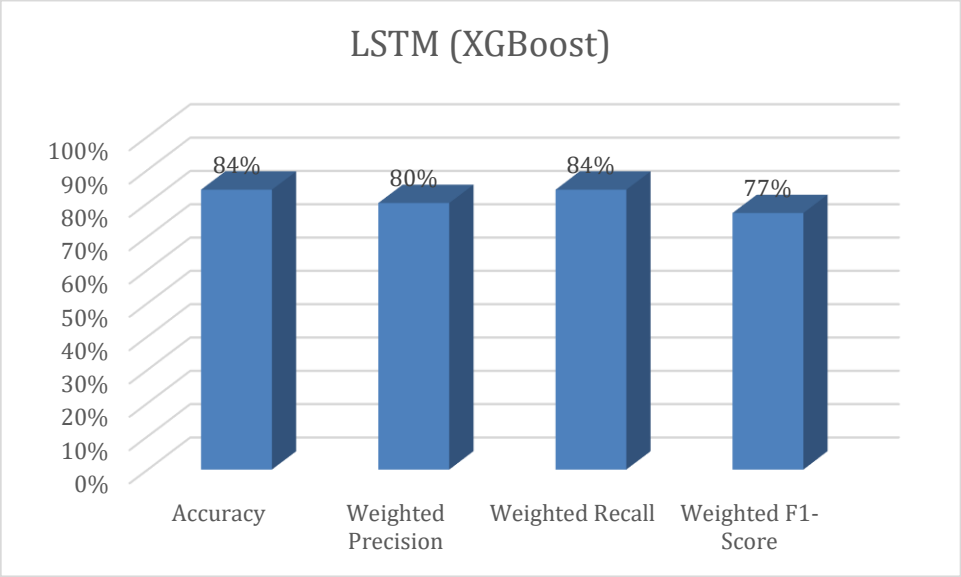
Table 31. LSTM (XGBoost) classifier model classification report.

Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	100%	91%	126,474
1	55%	0%	1%	23,526
Accuracy			84%	150,000
Macro Avg	69%	50%	46%	150,000
Weighted Avg	80%	84%	77%	150,000

Table 32. LSTM (XGBoost) classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 126,419	FN = 55
Actual: NO	FP = 23,460	TN = 66

Figure 30. Long Short-Term Memory (LSTM) (XGBoost) classifier model evaluation.



We evaluate our XGBoost classification model using the Scikit-Learn confusion matrix and classification report, our XGBoost classifier model maintains the accuracy of 84%, we boost the weighted precision to 80% and maintain our weighted recall and f1-scores.

5.2.7 Experiment 14: Ride-sharing Prediction using a Capsule Network (CapsNet)

Capsule networks (CapsNet) group neutrons together in layers of capsules, CapsNet utilizes dynamic routing and implicit regularization according to the authors in (Xiong et al, 2019). We build a capsule network (CapsNet) classifier model using TensorFlow Keras libraries and online documentation, for the integration of the CapsNet model with our dataset, we utilize the trip pick-up time, drop-off time, pick-up location ID, Drop-off location ID, passenger count, trip distance, total amount, payment type and trip type while our target variable is the newly created column signifying if the ride was shared or not represented in binary form where shared rides are represented as 1 and rides not shared represented as 0, we create our CapsNet classifier model using the Keras CIFAR-10 CNN-CAPSULE documentation (CIFAR-10), this involves building a

function to squash our input data in reference to Geoffrey Hinton's paper, however we utilize 0.5 to calculate our squash scale instead of 1, define a margin loss function and SoftMax function and build a capsule layer activated by the squash function, the capsules input shape consists of a batch size, number of input capsules and number of input dimensions, we build a one (1) dimensional convolution model consisting of 2 64 neuron layers a pooling layer with a pool size of two (2) and is activated by the rectifier activation function to serve as input for our capsule layer. The capsules final output is the length of two (2) capsules and 16 dimensions and is compiled using stochastic gradient descents “adam” optimizer which splits a dataset into multiple subsets of training cases and executes modification of the data features of each training case to identify the features that produce the best results, the loss is measured using the defined margin loss function. We proceed to split our dataset into 70% training and 30% test sets and the CapsNet classifier model is trained on 30 epochs with a batch size of 1000 records each using our January 2017 taxi trip training set consisting of our independent data variables. We then evaluate our CapsNet classifier model using the Scikit-Learn classification report and confusion matrix and carry-out prediction using our January 2017 test set made up of shared trips multi-label information.

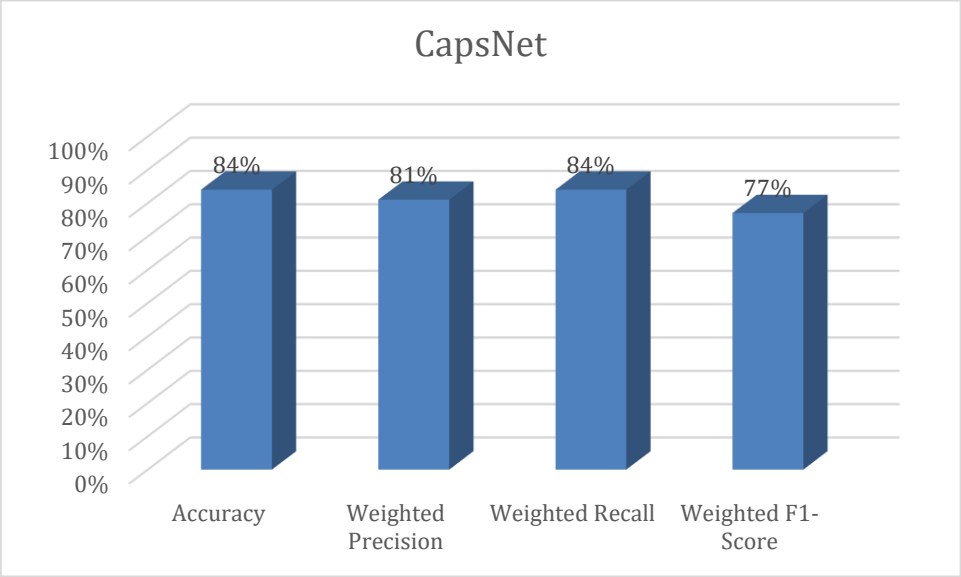
Table 33. Capsule Network (CapsNet) classifier model classification report.

Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	100%	91%	270,197
1	66%	0%	1%	50,882
Accuracy			84%	321,079
Macro Avg	75%	50%	46%	321,079
Weighted Avg	81%	84%	77%	321,079

Table 34. Capsule Network (CapsNet) classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 258,386	FN = 66
Actual: NO	FP = 48,400	TN = 127

Figure 31. Capsule Network (CapsNet) classifier model evaluation.



We evaluate our CapsNet classification model using Scikit-Learn confusion matrix and classification report, we aim to record a high accuracy, precision, recall, and F1-scores, the precision score is a measurement of the ability of our model to accurately not classify our positive target labels as negative labels, recall refers to the ability of our model to capture all our positive labels and f1-score is the harmonic mean of the precision and recall, from our evaluation results, the classification report shows that our CapsNet classifier model records a similar 84% accuracy, we record 81% weighted precision slightly lower than our CNN classifier model, we maintain our weighted recall score of 84% and f1-score of 77%. In an attempt to boost our evaluation results, we proceed to apply extreme gradient boosting (XGBoost).

5.2.7.1 Extreme Gradient Boosting (XGBoost)

We create an XGBoost classifier model that utilizes our CapsNet classifier model as a base estimator with 200 estimators and set our learning rate to 0.5. We train the XGBoost model using the January 2017 training set and carry out predictions using the January and February 2017 test sets, the XGBoost classifier model is evaluated using Scikit-learn classification report and confusion matrix.

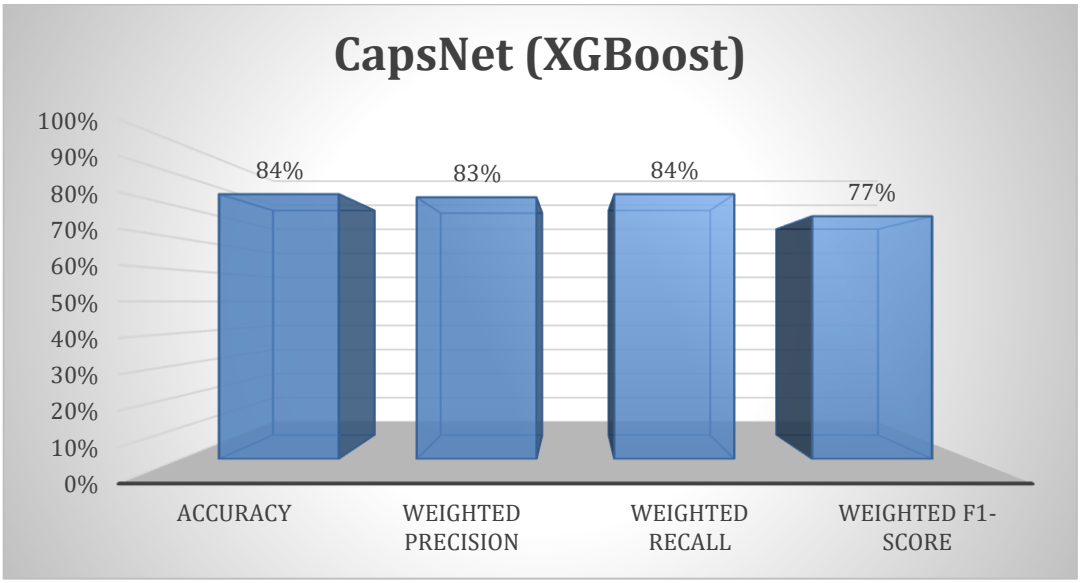
Table 35. CapsNet (XGBoost) classifier model classification report.

Classification report for January 2017 test set for shared rides.				
	Precision	Recall	F1-Score	Support
0	84%	100%	91%	270,197
1	78%	0%	1%	50,882
Accuracy			84%	321,079
Macro Avg	81%	50%	46%	321,079
Weighted Avg	83%	84%	77%	321,079

Table 36. CapsNet (XGBoost) classifier model confusion matrix.

Confusion Matrix for January 2017 test set for shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 270,139	FN = 58
Actual: NO	FP = 50,671	TN = 211

Figure 32. Capsule Network (CapsNet) (XGBoost) classifier model evaluation.



We evaluate our XGBoost classification model using the Scikit-Learn confusion matrix and classification report, our XGBoost classifier model maintains the accuracy of 84%, we can boost the weighted precision to 83% and maintain our weighted recall and f1-scores but still we are only able to predict 211 shared rides.

5.2.8 Comparative Analysis of Arrival Time Evaluation Results.

Table 37. Comparative analysis of Arrival time models evaluation (Trained).

	Explained Variance Score (EVS	**Mean Squared Error (MSE)**	**Mean Absolute Error (MAE)**	**r2 Score**
Linear Regression	99%	4.606e-06	0.0001	99%
Decision Trees	99%	1.044e-05	0.0001	99%

Random Forest	99%	5.636e-06	0.0001	99%
Random Forest (XGBoost)	99%	6.711e-06	0.0008	99%
K-NN	99%	2.478e-05	0.0011	99%
Ensemble Model	99%	6.480e-06	0.0003	99%
Artificial Neural Network (ANN)	98%	0.0010	0.0036	98%
Artificial Neural Network (ANN) (XGBoost)	99%	1.352e-05	0.0022	99%
Convolutional Neural Network (CNN)	99%	3.0927e-05	0.0049	99%
Convolutional Neural Network (CNN) (XGBoost)	99%	6.711e-06	0.0008	99%
Ensemble Boosting	99%	6.398e-05	0.0011	99%
Ensemble Boosting (XGBoost)	99%	8.652e-05	0.0045	99%
Long short-term memory (LSTM)	99%	0.0005	0.0179	99%

Figure 33. Comparative analysis of Mean Squared Error Trained.

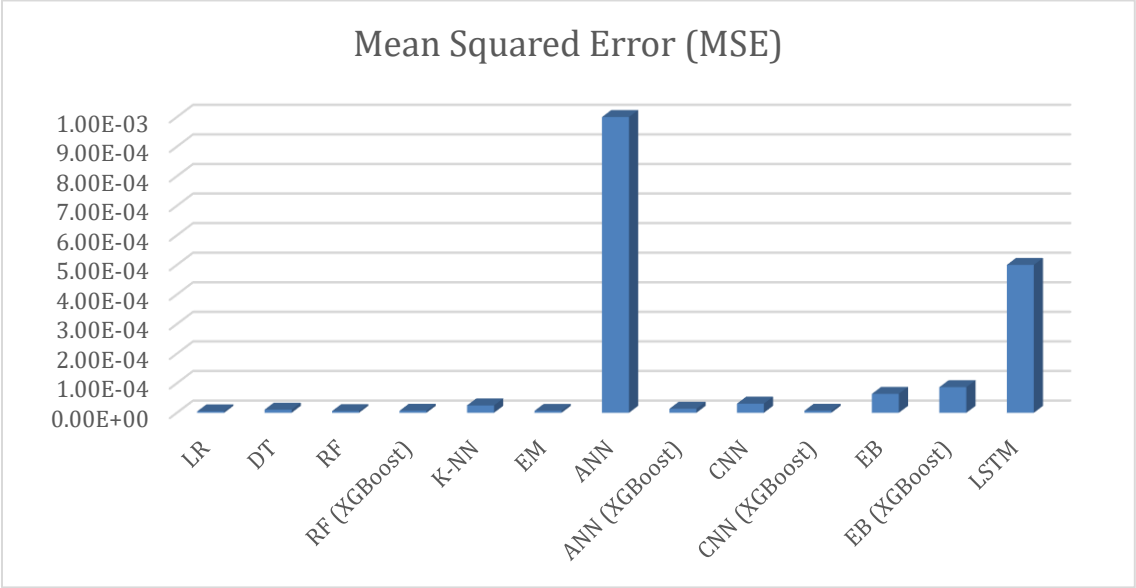


Figure 34. Comparative analysis of Mean Absolute Error (MAE) (Trained).

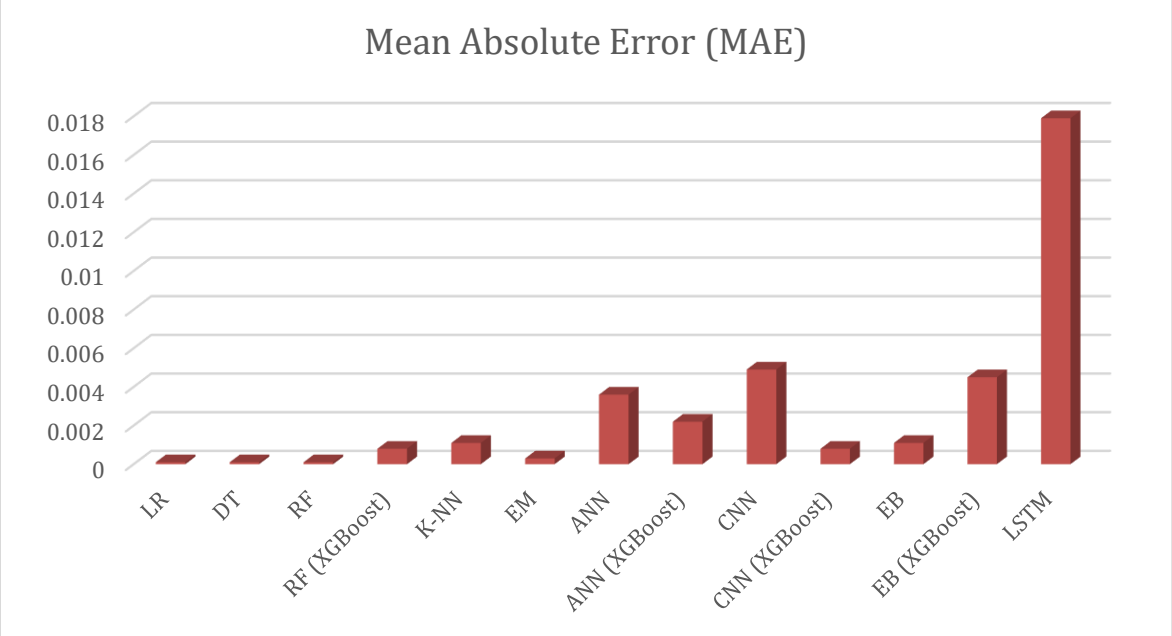


Table 38. Comparative analysis of Arrival time models evaluation (Untrained).

	Explained Variance Score (EVS	**Mean Squared Error (MSE)**	**Mean Absolute Error (MAE)**	**r2 Score**
Linear Regression	99%	8.070e-06	0.0012	99%
Decision Trees	99%	3.342e-05	0.0026	99%
Random Forest	99%	1.442e-05	0.0021	99%
Random Forest (XGBoost)	99%	1.353e-05	0.0022	99%
K-NN	99%	8.039e-05	0.0050	99%
Ensemble Model	99%	1.627e-05	0.0026	99%
Artificial Neural Network (ANN)	98%	0.0011	0.0046	98%
Artificial Neural Network (ANN) (XGBoost)	99%	6.711e-06	0.0008	99%
Convolutional Neural Network (CNN)	99%	6.881e-05	0.0076	99%
Convolutional Neural Network (CNN) (XGBoost)	99%	1.352e-05	0.0022	99%
Ensemble Boosting	99%	8.262e-05	0.0025	99%
Ensemble Boosting (XGBoost)	99%	0.0001	0.0054	99%

Figure 35. Comparative analysis of Mean Squared Error (Untrained).

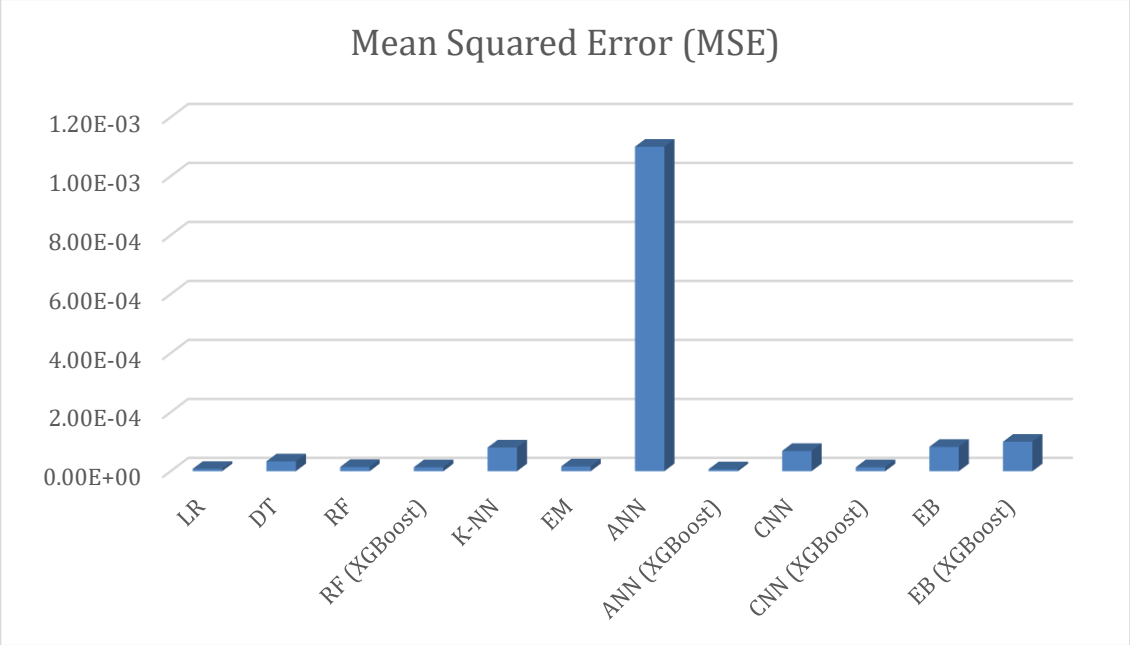
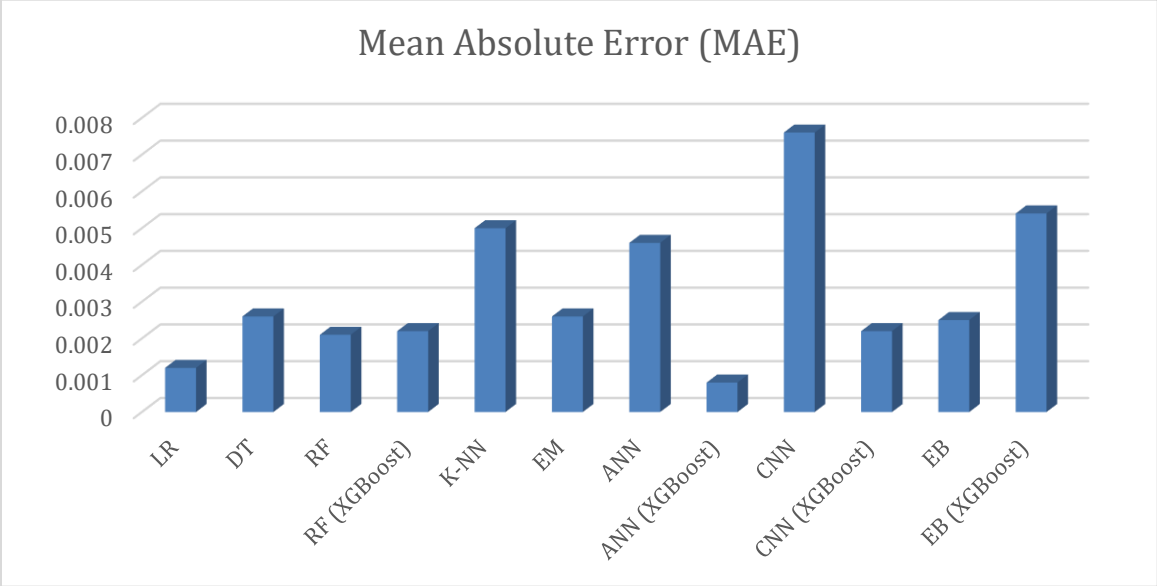


Figure 36. Comparative analysis of Mean Absolute Error (MAE) (Untrained).



5.2.9 Evaluations Discussion

From our comparative analysis of our regression models evaluation results, we observe a clear trend where all our arrival time prediction models record 99% explained variance score and r2

score for trained (January 2017) and untrained (February 2017) test sets excluding the artificial neural network model (ANN), overall all our arrival time prediction regression models perform relatively well concerning the mean squared error (MSE) and mean absolute errors (MAE) recorded, we get the lowest MSE and MAE for trained test sets from our linear regression model and obtain the lowest MSE and MAE for our untrained test sets from our boosted (XGBoost) artificial neural network (ANN), our general observation is that our machine learning models outperform our deep learning models for arrival time prediction and we were a bit disappointed in the prediction accuracy of our ensemble model.

5.2.10 Comparative Analysis of Ride-sharing Prediction Evaluation Results.

Table 39. Comparative analysis of ride-sharing prediction models evaluation.

	Accuracy	**Weighted Precision**	**Weighted Recall**	**Weighted F1-Score**
Decision Trees	73%	74%	73%	73%
Random Forest	84%	75%	84%	77%
Random Forest (XGBoost)	84%	83%	84%	77%
ANN	84%	71%	84%	77%
(ANN) (XGBoost)	84%	75%	84%	77%
CNN	84%	83%	84%	77%

CNN (XGBoost)	83%	78%	84%	77%
LSTM	84%	71%	84%	77%
LSTM (XGBoost)	84%	80%	84%	77%
CapsNet	84%	81%	84%	77%
CapsNet (XGBoost)	84%	83%	84%	77%

Figure 37. Comparative analysis of ride-sharing prediction models accuracy.

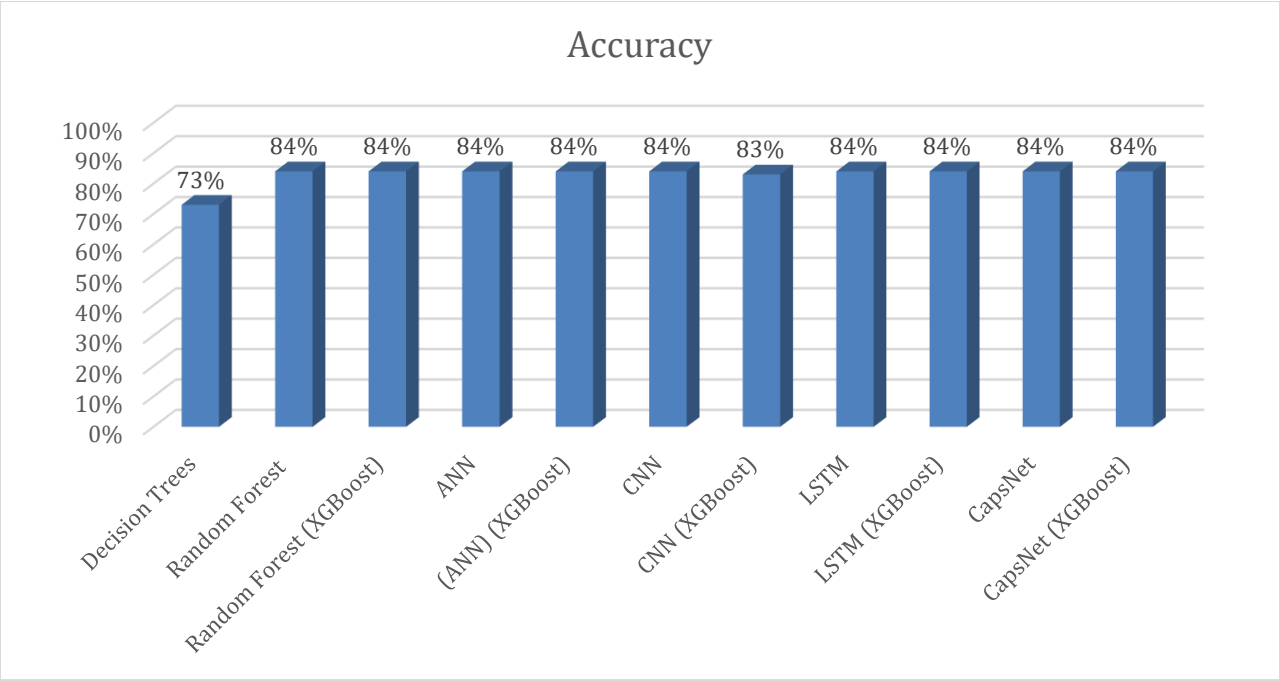


Figure 38. Comparative analysis of ride-sharing prediction models weighted precision.

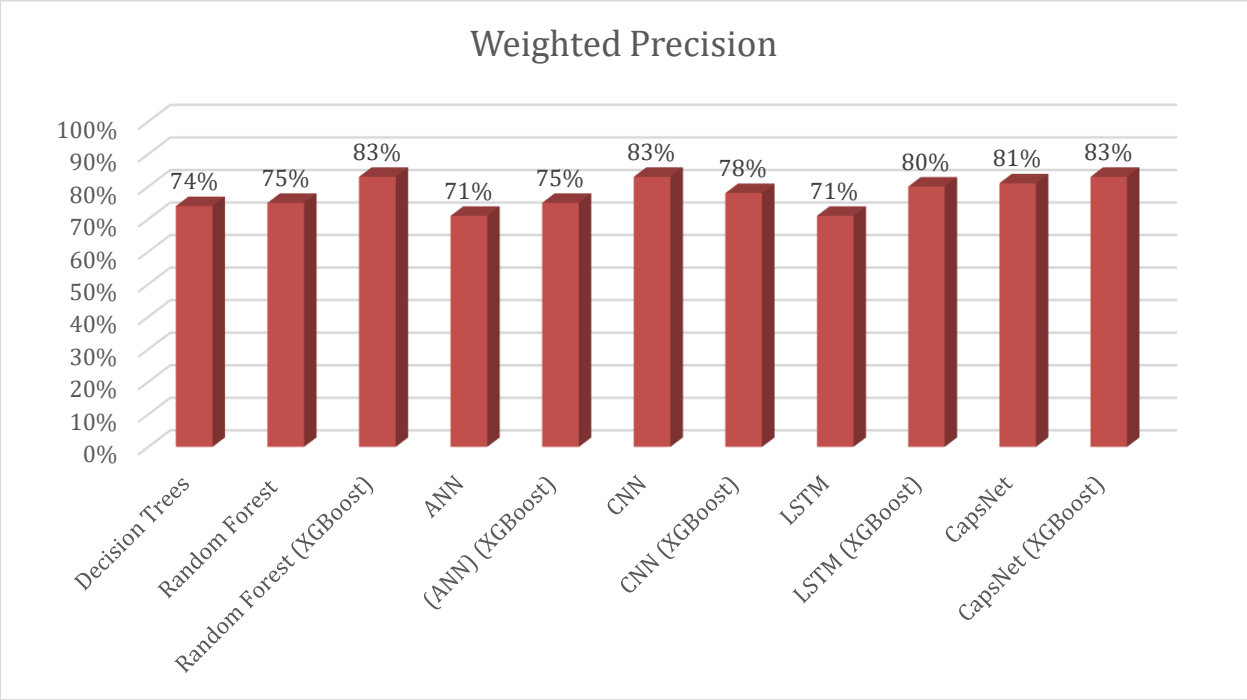


Figure 39.Comparative analysis of ride-sharing prediction models weighted recall.

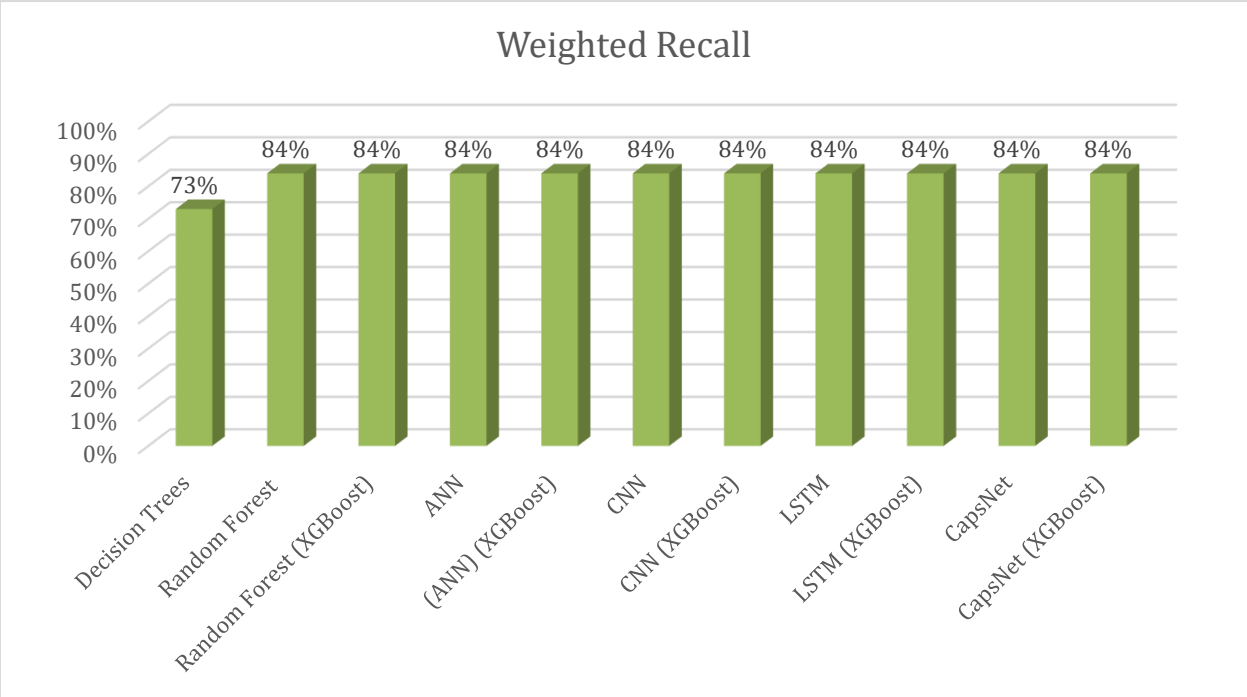
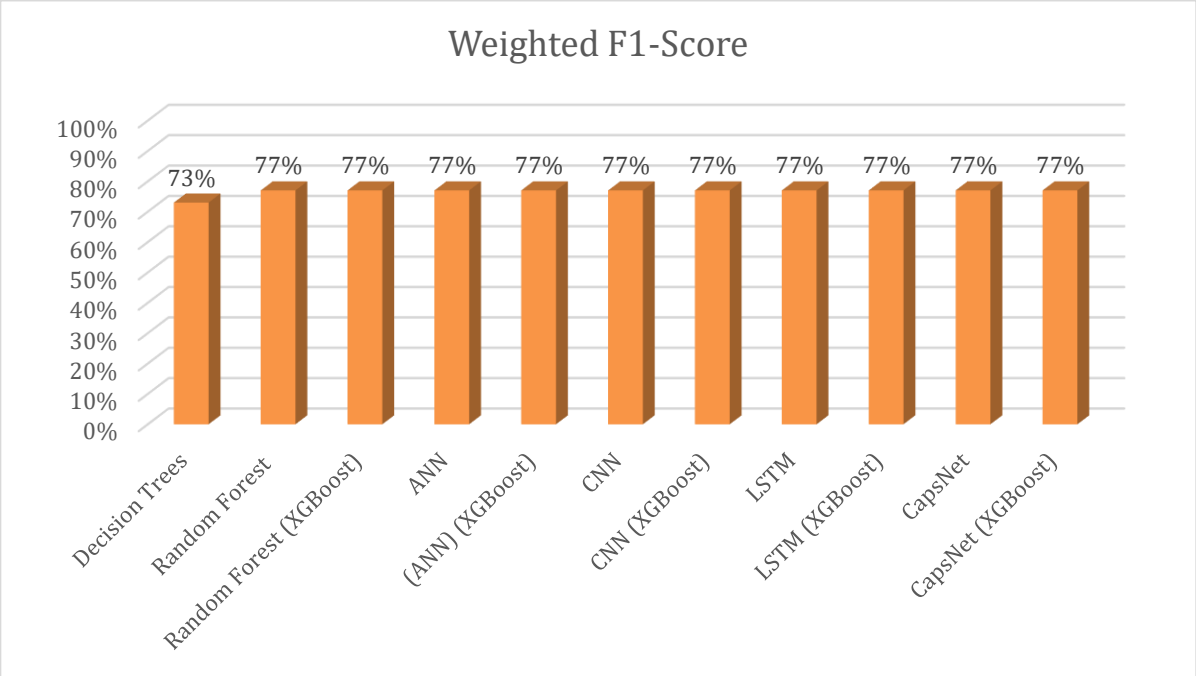


Figure 40. Comparative analysis of ride-sharing prediction models weighted f1-score.



5.2.11 Evaluation Discussion

From our comparative analysis of our classifier models evaluation results, we record similar results for both machine learning and deep learning models except the decision tree model. The performance of our classification models are generally substandard because we are still unable to predict the shared rides, for comparison, however, we obtain the best results from the random forest classifier model boosted using extreme gradient boosting (XGBoost) and the capsule network (CapsNet) also boosted by XGBoost, and the convolutional neural network (CNN) based on the weighted precision calculated, the comparison analysis also shows that we are unable to achieve an evaluation score higher than 84% for every metric which is well below our expectations. The overall observation herein is that the model is currently unable to accurately differentiate between a shared and an unshared ride meaning our classification algorithm needs to be improved or better data engineering needs to be executed.

Chapter 6: Design and Implementation of Proposed Solution

The experiments executed in chapter 5 were to identifying the preferred techniques for building our proposed solution, we develop a taxi trip arrival time prediction model utilizing a stacked ensemble of a convolutional neural network (CNN) boosted using extreme gradient boosting techniques and a stacked ensemble of an AdaBoost, Bagging and gradient boosting regression models. We utilize both Scikit-learn ensemble technique and numerical processing stacking function, we also develop a classification algorithm to predict shared rides to meet the requirements of building an intelligent transportation system capable of predicting taxi trip arrival times and also predict shared rides, we believe that the stated requirements are essential for developing an intelligent and efficient ride-sharing scheme. The ride-sharing prediction model is a combination of a capsule network (CapsNet) classification algorithm boosted using extreme gradient boosting (XGBoost) and a Bagging, AdaBoost, and gradient boosting classification algorithms stacked together using a stacking function. We proceed with a technical breakdown of the implementation of our proposed solutions for ride-sharing prediction and taxi trip arrival time prediction.

6.1 Formal System Model (Arrival Time Prediction)

Definitions: the definitions herein describe the independent and target data features utilized for training our arrival time prediction model, the dataset is obtained from the New York Taxi & Limousine Commission (TLC) consisting of roughly 1 million green taxi rides. The definitions are adapted from (LPEP Trip Records).

6.1.1 Independent Data Features

Pick-up time: this refers to the date and time in which the taxi trip meter was engaged at a pick-up location, it is denoted as PT, we believe that the trip pickup time is an essential factor for arrival time prediction.

Pick-up location ID: this refers to the TLC taxi zone where a taximeter was engaged, it is denoted as PL.

Drop-off location ID: this refers to the TLC taxi zone where the taximeter was disengaged, it is denoted as DL.

Passenger Count: this refers to the number of passengers present in the taxi during the trip, it is denoted as PC, we believe that the number of passengers could have an impact on arrival times due to the possibility of some rides being shared having a longer trip distance.

Trip distance: this refers to the elapsed trip distance in miles recorded by the taximeter during the trip, it is denoted as TD, we expect the trip distance to have a relative impact on the trip arrival time.

Total Amount: this refers to the total amount charged to the passengers and does not include the cash tips, it is denoted as TA.

Trip Type: We are using the definition of trip type from (LPEP Trip Records) as a code indicating whether the trip was a street-hail or a dispatch that is automatically assigned based on the metered rate in use but can be altered by the driver. 1 indicates street hail and 2 indicates a dispatch, it is denoted as TT.

6.1.2 Target Data Feature

Arrival time: this refers to the date and time in which the taxi trip meter was disengaged at a drop-off location, it is denoted as AT, we are of the opinion that the trip pickup time is an essential factor for arrival time prediction.

6.1.3 Calculations

Pick-up time = PT . . . (where PT = pick-up time in datetime format) . . . . . x_1

Pick-up location = PL . . . (where PL = pick-up location ID) . . . . . x_2

Drop-off location = DL . . . (where DL = drop-off location ID) . . . . . x_3

Passenger = P

Passenger Count = $P_1 + P_2 + P_3 + \ldots P_n$ (where n = number of passengers) ... . x_4

Trip distance = TD (where TD = distance in miles from PL to DL) . . . . x_5

Total amount = TA (where TA = total amount charged to P) . . . . x_6

Trip type = TT (where $1 < TT \leq 2$ "TT is between 1 and 2") . . . . x_7

Arrival time = AT (where AT = drop-off time in datetime format) . . . . x_8

6.1.3.1 Data Preprocessing

We convert our dataset into a machine-readable form to improve our models' predictive accuracy, we utilize the Minmax scaler library to scale our dataset using the below function *(eq1)* where X_{sc} denotes our scaled output, X represents our input data, the lowest input value is denoted as X_{min} and the max input value denoted as X_{max}, we proceed to split our data into training and test set where the test set is represented as X * 0.3.

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad , \ldots, \textit{(eq1)}$$

6.1.3.2 Convolution layer

Our input data goes through two (2) convolution *(eq2)* and max pooling *(eq3)* layers, our convolution process feeds a set number of feature maps into the max-pooling layer creating a pooled feature map, We feed the second convolution layer with the pooled feature map from the first layer. The convolutional neural network (CNN) accepts our input data denoted as $\{x_1, x_2, \ldots, x_n\}$, our CNN consists of an input layer and a hidden layer, the convolutional neural network layers are denoted as $\{h_1, h_2, \ldots, h_n\}$, the layers of the CNN consists of feature detection activation nodes denoted as $\{w_1, w_2, \ldots, w_n\}$ activated using the rectifier activation function *(eq5)* denoted as *{ReLu}*, the output from each layer is denoted as $\{z\}$.

$$z^1 = h^{1-1} * w^1 \quad , \ldots, \textit{(eq2)}$$

$$h^1{}_{xy} = max_{i=0..s,j=0..s}\, h^{1-1}\,(x+y)(y+j) \quad , \ldots, \textit{(eq3)}$$

6.1.3.3 Fully Connected layer

We utilize a fully connected layer *(eq4)* for predicting our final results, this involves a forward propagation process and requires performing a linear transformation on the data passed into the fully connected layer.

$$z_l = W_l * \; h_{l-1} \quad , \ldots, \textit{(eq4)}$$

$$ReLu\,(z_i) = \max\,(0, z_i) \quad , \ldots, \textit{(eq5)}$$

We also execute backward propagation in an attempt to update our randomly initialized weights and bias to improve our prediction accuracy and perform a non-linear transformation of our data in the fully connected layer. Our convolutional neural network calculations are inspired by the work done in (Alajrami et al, 2020).

6.1.3.4 Gradient Boosting & Extreme Gradient Boosting (XGBoost)

Some models include functions as parameters and cannot be optimized using traditional optimization methods in Euclidean space.

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \boldsymbol{l}(\boldsymbol{y_i}, \hat{\boldsymbol{y}}_{\boldsymbol{i}}^{(\boldsymbol{t-1})} + \;\; \mathfrak{f}_{\boldsymbol{t}}\,(\boldsymbol{x_i})) + \boldsymbol{\Omega}(\mathfrak{f}_{\boldsymbol{t}}) \;, \ldots, \text{(eq6)}$$

The model is trained in an additive manner, this means we greedily add the base learner denoted as $\mathfrak{f}_{\boldsymbol{t}}$ that most improves our model, the loss function is denoted as $\boldsymbol{l}$ and $\boldsymbol{\Omega}$ denotes our regularisation function. In (Chen and Guestrin, 2016), utilizing Taylors approximation theorem they obtained the first and second order gradient statistics on the loss function to obtain the following XGBoost simplified objective.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i\, \mathfrak{f}_t(x_i) + \frac{1}{2} h_i \mathfrak{f}_t^2(x_i)] + \; \boldsymbol{\Omega}(\mathfrak{f}_{\boldsymbol{t}}) \;, \ldots, \text{(eq7)}$$

6.1.3.5 Bootstrap Aggregative method (Bagging)

According to (Rocca, 2019), bagging involves creating a bootstrap of different data samples so that each new bootstrap sample acts as an independent dataset drawn from the true distribution. We then fit the weak learners $\mathcal{L}$ on the new samples and average their outputs in an aggregative manner, this creates an ensemble model with less variance than its components. Where our weak learners are denoted in (eq8) for simple averaging in (eq9).

$$w_1(.), w_2(.), \dots, w_L(.) \ , \ . \ . \ ., \text{(eq8)}$$

$$s_L(.) = \frac{1}{L} \sum_{l=1}^{L} w_l(.) \ , \ . \ . \ ., \text{(eq9)}$$

6.1.3.6 Adaptive Boosting method (AdaBoost)

In AdaBoost, the weights are set for the sample data selected and the model, this ensures the model focuses more on the observations with inaccurate predictions, the weights ω are adjusted sequentially as the algorithm iterates according to the author in (Jung, 2018).

$$\widehat{MME}_{emp}^{(j)} = \frac{\sum_{i=1}^{N} \omega_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^{N} \omega_i} \ , \ . \ . \ ., \text{(eq10)}$$

6.1.3.7 Stacking

For the stacking technique utilized in our arrival time prediction solution, we simply calculate the mean of the prediction results from our XGBoost model, AdaBoost model, Gradient Boosting model, and Bagging model.

$$m = \frac{sum\ of\ model\ results}{number\ of\ models} \ , \ . \ . \ ., \text{(eq11)}$$

6.1.4 Pseudocode of Algorithm

Input Data:

Pick-up time, datetime64.

Pick-up location ID, int64.

Drop-off location ID, int64.

Passenger Count, int64.

Trip distance, float64.

Total amount, float64.

Trip type, int64.

Algorithm: Arrival time prediction

Input: taxi trip data $\{T_1, T_2, \ldots, T_n\}$
Output: taxi trip arrival time, . . . , AT

$$AT \leftarrow \bigcup_{i=1}^{n} T_i$$

1.
2. Utilizing training set T, we use n base model to train n regressors, $\{f_1(x), \ldots, f_n(x)\}$
3. **for** arrival time $i = 1$ to m **do**
4. Select a sample x_n from T_n
5. Set $R = 0$;
6. **for** regressor $r = 1$ to n **do**
7. **if** x_n is observed as taxi trip data by $f_r(x)$ ***then***
8. $R{+}{+}$
9. ***end if***
10. ***end for***
11. **if** $R \neq n$ then
12. Insert *at* into meta learner EM_r
13. ***end if***
14. *arrival time prediction*
15. **end for**

Description of the algorithm

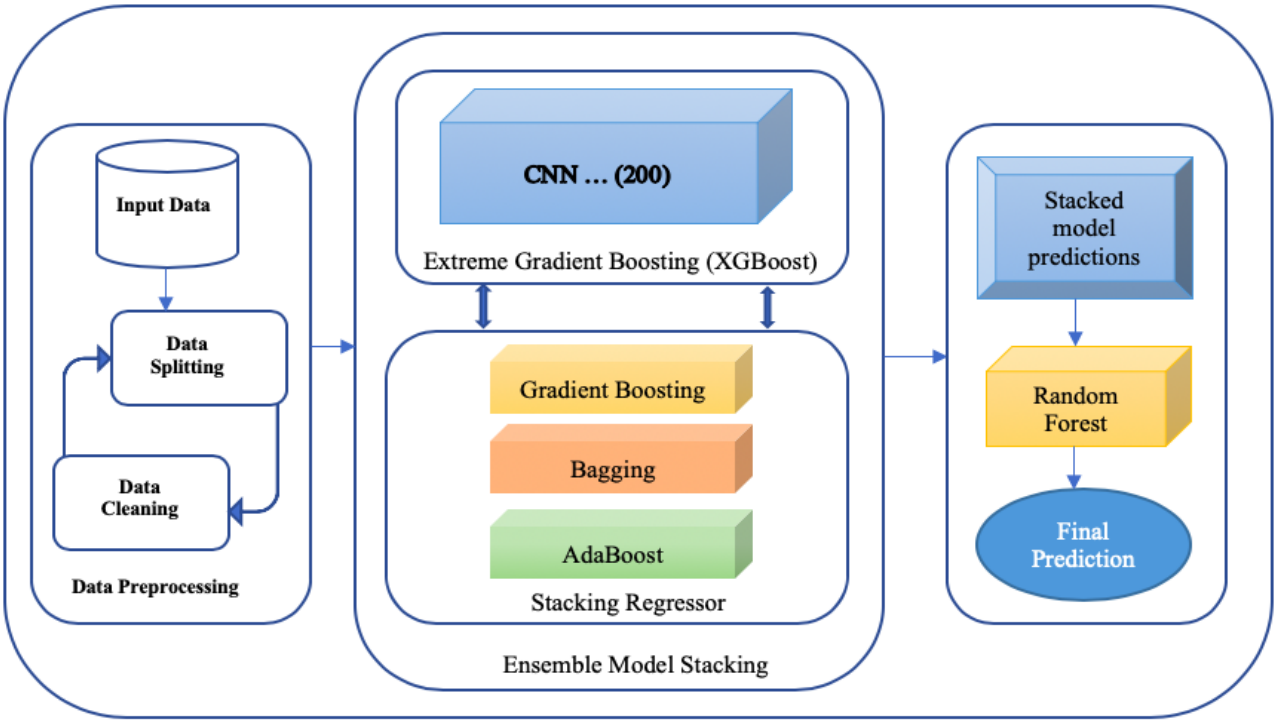
Our algorithm represents the steps to be taken for predicting taxi trips arrival times, our input data $\{T_1, T_2, \ldots, T_n\}$ consists of taxi trip data-independent features described in the input data section,

our base models are represented as $\{f_1(x), \ldots, f_n(x)\}$ which will be used to execute model training and prediction activities. X_n represents one taxi trip record selected from our dataset and *R* represents our regression models i.e the convolutional neural network (CNN) model boosted using extreme gradient boosting (XGBoost) and the ensemble regression model stacking our Bagging, gradient boosting and AdaBoost regression models. We represent the arrival time predictions from our regression models as *at,* our arrival time predictions are used to train a random forest model serving as a meta learner model represented in our algorithm as EM_r. The algorithm implemented for our proposed solution is inspired by the work done in (Wang et al, 2017).

6.1.5 Implementation of Proposed Solution for Arrival Time Prediction.

Our proposed solution herein for predicting taxi trip arrival times is a deep ensemble of a convolutional neural network (CNN) regression model boosted using extreme gradient boosting (XGBoost) and an ensemble of a bootstrap aggregation (Bagging) regression model, adaptive boosting (AdaBoost) regression model and a gradient boosting regression models combined using Scikit-learns stacking regressor library.

Figure 41. Arrival time prediction Model Architecture.



The above architecture describes the implementation of our arrival time prediction solution, we proceed to deliver a detailed presentation of all the steps involved in our architecture.

The first phase of building our proposed solution involves the cleaning and preprocessing of our dataset, this includes the identification of our independent and target data features, conversion of our data features into a machine-readable form, and training and testing set for training and prediction activities.

6.1.5.1 Data Preprocessing

Using the preprocessing tools available on Scikit-learn libraries, a Min-max scaler is used to convert the time series data into a machine-readable form transforming the data records into a scaled form between 0 and 1.

6.1.5.2 Building a Convolution Neural network (CNN) using TensorFlow

We build a convolutional neural network (CNN) model using TensorFlow Keras libraries and for the integration of the convolutional neural network (CNN) model with our dataset, we utilize the pick-up time, pick-up location, drop-off location, passenger count, trip distance, total amount and trip type present in our taxi trip dataset as independent variables and the drop-off/arrival-time as the target variable, we create our CNN model using Keras Sequential library and build our network layers using a one dimensional convolutional Keras layer, for feature mapping and detection each convolutional layer consists of 32 filters with a kernel size of 2, the input shape of our convolutional neural network (CNN) is a 7 by 1, 2 dimensional matrix to accommodate our independent data features, we utilize the max pooling Keras library to add a one (1) dimensional pooling layer to down sample our input data representation and capture the key features in our dataset, we utilize Keras flatten library to convert our data features matrix into a vector and using the Keras dense layer library we add a fully connected layer consisting of 128 neurons that uses the flattened results of the pooling and convolution layers to make a final prediction, each network layer is activated using the rectifier activation function and the output layer is captured using a fully connected dense layer, the model is compiled using stochastic gradient descent “adam” optimizer which splits a dataset into multiple subsets of training cases and executes modification of the data features of each training case in order to identify the features that produce the best results and loss is calculated and minimized using the mean squared error. We proceed to split our dataset into 70% training and 30% test sets, and the convolutional neural network model is trained

on 50 epochs with a batch size of 1000 records each using our January 2017 taxi trip training set consisting of our independent data variables. We then evaluate our CNN model using Scikit-Learn regression evaluation metrics and carry-out prediction using our January and February 2017 test sets made up of arrival times of taxi trips for both months. The implementation of this solution was adapted from (Kirill Eremenko at Udemy).

6.1.5.3 Extreme Gradient Boosting (XGBoost)

Extreme gradient boosting (XGBoost) is an advanced form of the well-known gradient boosting algorithm and framework by (Friedman, 2016). The XGBoost implementation package we used supports' various objective functions, including regression, classification, and rankings as disclosed in (Chen et al, 2016). Boosting is an ensemble technique whereby we add a new model to correct errors made by older models, and then add models sequentially until no more corrections are required, In gradient boosting, we create new models to predict the errors of older models, then our final result is based on the average of all our models. XGBoost permits distributed computing and is very efficient for large training sets. For our solution we train an XGBoost regression model with 200 weak random forest regression models serving as base estimators then train our January 2017 taxi trip training dataset, we evaluate our XGBoost using the same regression metric for comparison purposes.

6.1.5.4 Building an Ensemble Regression model using Scikit-Learn Stacking Regressor

We proceed to create an ensemble regression model combining the predictive attributes of an adaptive boosting (AdaBoost) regression model, a bootstrap aggregation (Bagging) regression model, and a gradient boosting regression model, the following is a step by step implementation of our ensemble regression model.

6.2 Formal System Model: Ride-sharing Solution Prediction

Definitions: the definitions herein describe the independent and target data features as taken from (LPEP Trip Records) utilized for training our ride-sharing prediction model, the dataset is obtained from the New York Taxi & Limousine Commission (TLC) consisting of roughly 1 million green taxi rides per month from January to June 2019.

6.2.1 Independent Data Features

Pick-up time: this refers to the date and time in which the taxi trip meter was engaged at a pick-up location, it is denoted as *PT*, we believe that the trip pickup time is an essential factor for predicting shared rides.

Drop-off time: this refers to the date and time in which the taxi trip meter was disengaged at a drop-off location, it is denoted as *DT*, we believe that the trip pickup time is an essential factor for predicting shared rides.

Pick-up location ID: this refers to the TLC taxi zone where a taximeter was engaged, it is denoted as *PL*.

Drop-off location ID: this refers to the TLC taxi zone where the taximeter was disengaged, it is denoted as *DL*.

Fare Amount: this refers to the time and distance fare calculated by the taximeter, denoted as *FA*.

Total Amount: this refers to the total amount charged to the passengers and does not include the cash tips; it is denoted as TA.

Tip Amount: this refers to credit card tips for each trip, it is denoted as *TI*.

Trip Type: We are using the definition of trip type from (LPEP Trip Records) as a code indicating whether the trip was a street-hail or a dispatch that is automatically assigned based on the metered rate in use but can be altered by the driver. 1 indicates street hail and 2 indicates a dispatch, it is denoted as TT.

Payment Type: this refers to a numerical code signifying hoe the trip was paid for, it is denoted as *PA*.

6.2.2 Target Data Feature

Shared: this is signified as either 1 or 0, where 0 signifies a shared ride and 1 represents unshared rides, it is denoted as *S*.

6.2.3 Calculations

Pick-up time = PT . . . (where PT = pick-up time in datetime format) . . . . . x_1

Drop-off time = DT . . . (where DT = drop-off time in datetime format) . . . . . x_2

Pick-up location = PL . . . (where PL = pick-up location ID) . . . . . x_3

Drop-off location = DL . . . (where DL = drop-off location ID) . . . . . x_4

Trip distance = TD (where TD = distance in miles from PL to DL) . . . . . x_5

Fare amount = FA (where FA = total amount calculated by taximeter) . . . . . x_6

Total amount = TA (where TA = total amount charged to P) . . . . . x_7

Tip amount = TI (where TI = total tip payed) . . . . . x_8

Trip type = TT (where 1 < TT ≤ 2 "TT is between 1 and 2") . . . . . x_9

Payment Type = PA (where PA = payment mode utilized) . . . . . x_{10}

Shared = S (where S is either 1 or 0) . . . . . x_{11}

6.2.3.1 Data Preprocessing

We convert our dataset into machine-readable form to improve our models' predictive accuracy, we utilize the Minmax scaler library to scale our dataset using the below function *(eq1)* where X_{sc} denotes our scaled output, X represents our input data, the lowest input value is denoted as X_{min}

and the max input value denoted as X_{max}, we proceed to split our data into training and test set where the test set is represented as X * 0.3.

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad , \ldots, \textit{(eq1)}$$

6.2.3.2 Capsule Network (CapsNet) layer

Capsule networks (CapsNet) make use of a standard convolutional neural network (CNN), a primary capsule layer for applying convolution operations to the input and re-allocation of the output in several capsules and executes dynamic routing for full connection of the capsule layers in the network. The capsule network utilizes a non-linear activation function called squash (eq2) to shrink the length of the output vector to between zero (0) and one (1), v_j is the vector output of the capsule j and s_j is the total output as described by the authors in (Xiong et al, 2019).

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \ , \ldots, \textit{(eq2)}$$

The total input vector s_j of the j^{th} capsule layer L+1 is generated via summing the weights $\hat{u}_{j|i}$ of every capsule layer L.

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \ , \ldots, \text{(eq3)}$$

$$\hat{u}_{j|i} = W_{ij} u_i \ , \ldots, \text{(eq4)}$$

W_{ij} denotes a weight matrix and u_i is the output vector of the i^{th} capsule of layer L, we utilize a SoftMax function (eq5) to ensure the sum of our coefficients c_{ij} is always equal to one (1).

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \ , \ldots, \text{(eq5)}$$

We calculate the total margin loss for each fully connected capsule with (eq6).

$$L_k = T_k \max(0, m^+ - \| v_k \|^2) + \lambda(1 - T_k) \max(0, \| v_k \| - m^-)^2$$

$T_k = 1$ if and only if a digit of class k is present, $m^+ = 0.9, m^- = 0.1, \lambda = 0.5$. (Xiong et al, 2019).

6.2.3.3 Gradient Boosting & Extreme Gradient Boosting (XGBoost)

Some models include functions as parameters and cannot be optimized using traditional optimization methods in Euclidean space.

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \boldsymbol{l}(\boldsymbol{y_i}, \hat{\boldsymbol{y}}_{\boldsymbol{i}}^{(\boldsymbol{t-1})} + \quad \mathfrak{f}_{\boldsymbol{t}}(\boldsymbol{x_i})) + \boldsymbol{\Omega}(\mathfrak{f}_{\boldsymbol{t}}) \text{ , . . ., (eq7)}$$

In (Chen and Guestrin, 2016), the authors state how the model is trained in an additive manner, this means we greedily add the base learner denoted as $\mathfrak{f}_{\boldsymbol{t}}$ that most improves our model. The loss function is denoted as $\boldsymbol{l}$ and $\boldsymbol{\Omega}$ denotes our regularisation function. In (Chen and Guestrin, 2016), utilizing Taylors approximation theorem they obtained the first and second order gradient statistics on the loss function to obtain the following XGBoost simplified objective.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i \, \mathfrak{f}_t(x_i) + \frac{1}{2} h_i \mathfrak{f}_t^2(x_i)] + \boldsymbol{\Omega}(\mathfrak{f}_{\boldsymbol{t}}) \text{ , . . ., (eq8)}$$

6.2.3.4 Bootstrap Aggregative method (Bagging)

In (Jung, 2018), the authors outline that bagging involves creating a bootstrap of different data samples so that each new bootstrap sample acts as an independent dataset drawn from the true distribution. We then fit the weak learners $\mathcal{L}$ on the new samples and average their outputs in an aggregative manner, this creates an ensemble model with less variance than its components. Where our weak learners are denoted in (eq9) for simple majority vote in (eq10).

$$w_1(.), w_2(.), \dots, w_L(.) \text{ , . . ., (eq9)}$$

$$s_L(.) = arg \max_k [card(l | w_l(.) = k)] \text{ , . . ., (eq10)}$$

6.2.3.5 Adaptive Boosting method (AdaBoost)

In (Rocca, 2019), the author discusses AdaBoost algorithms, the weights are set for the sample data selected and the model, this ensures the model focuses more on the observations with inaccurate predictions, the weights ω are adjusted sequentially as the algorithm iterates.

$$\widehat{MME}_{emp}^{(j)} = \frac{\sum_{i=1}^{N} \omega_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^{N} \omega_i}, \ldots, \text{(eq11)}$$

6.2.3.6 Stacking

For the stacking technique utilized in our ride sharing prediction solution, we utilize a NumPy function to stack the prediction results together and we simply utilize a random forest model to execute majority vote predictions based on the prediction results from our XGBoost model, AdaBoost model, Gradient Boosting model, and Bagging model.

6.2.4 Pseudocode of Algorithm

Input Data:

Pick-up time, datetime64.

Drop-off time, datetime64.

Pick-up location ID, int64.

Drop-off location ID, int64.

Trip distance, float64.

Fare amount, float64.

Total amount, float64.

Tip Amount, float64.

Trip type, int64.

Payment type, int64.

Algorithm: Ride-sharing prediction

Input: taxi trip data $\{T_1, T_2, \ldots, T_n\}$
Output: shared ride, . . ., S

1. $S \leftarrow \cup_{i=1}^{n} T_i$
2. Utilizing training set T, we use n base model to train n classifiers, $\{f_1(x), \ldots, f_n(x)\}$
3. **for** shared ride i = 1 to m **do**
4. Select a sample x_n from T_n
5. Set C = 0;
6. **for** classifier c = 1 to n **do**
7. **if** x_n is observed as taxi trip data by $f_c(x)$ ***then***
8. C++
9. ***end if***
10. ***end for***
11. **if** $C \neq n$ then
12. Insert s into meta learner EM_c
13. ***end if***
14. *ride-sharing prediction*
15. **end for**

Description of the algorithm

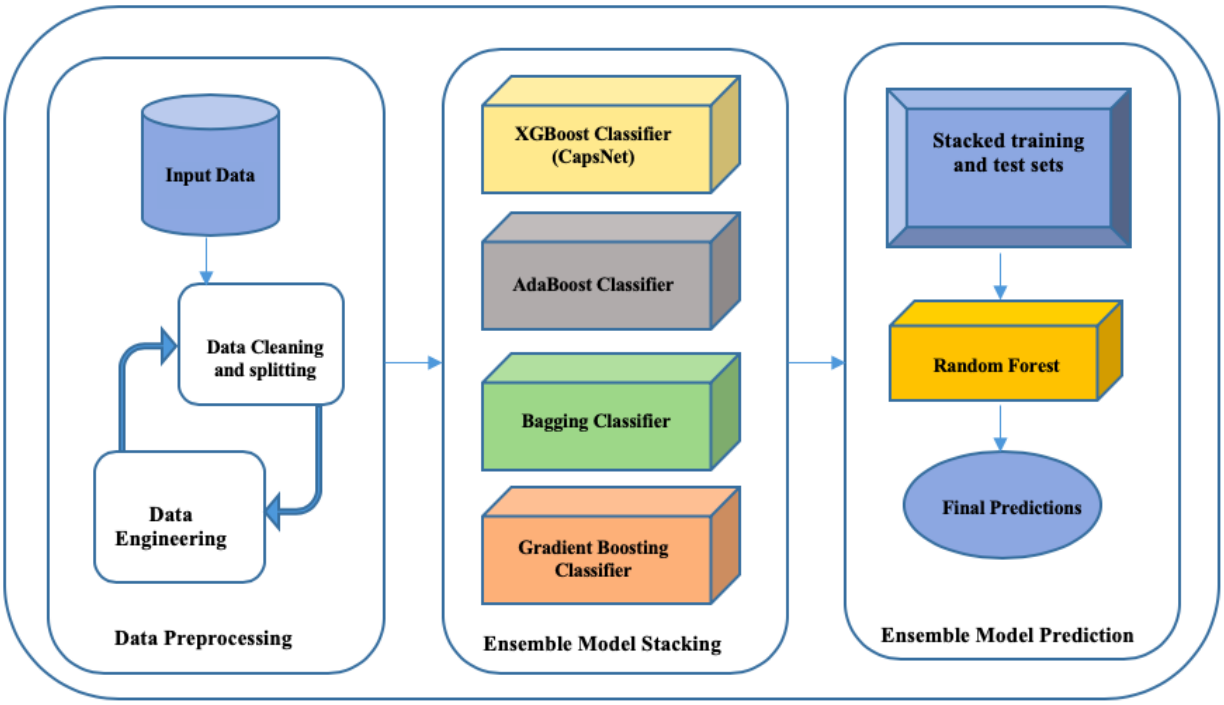
Our algorithm represents the steps to be taken for predicting shared taxi trips, our input data $\{T_1, T_2, \ldots, T_n\}$ consists of the taxi trip independent data features described in the input data section, our base models are represented as $\{f_1(x), \ldots, f_n(x)\}$ which will be used to execute model training and prediction activities. X_n represents one taxi trip record selected from our dataset and C represents our classifier models i.e the capsule network (CapsNet) model boosted using extreme gradient boosting (XGBoost), our Bagging, gradient boosting and AdaBoost classifier models. We represent the shared-ride predictions from our classifier models as s, our shared ride predictions are used to train a random forest model serving as a meta learner model represented in our algorithm as EM_c.

6.2.5 Implementation of Proposed solution for Ride-Sharing Prediction

The experiments executed in chapter 5 of this research involved several techniques including machine learning, deep learning, ensemble learning and ensemble boosting techniques, several machine learning techniques showed impressive results i.e. random forest for classification algorithms and linear regression for regression algorithms, however, the proposed solution based on the evaluation results would be a combination of experiment (11) Ensemble Boosting technique and experiment (14) Capsule Network (CapsNet) for ride-sharing classification algorithms, ensemble methods can consist of bagging, boosting or stacking techniques. We build a capsule network (CapsNet) classifier model using TensorFlow Keras libraries and online documentation, for the integration of the CapsNet model with our dataset, we utilize the trip pick-up time, drop-off time, pick-up location ID, Drop-off location ID, passenger count, trip distance, total amount, payment type and trip type while our target variable is the newly created column signifying if the ride was shared or not represented in binary form where shared rides are represented as 0 and rides not shared represented as 1, we create our CapsNet classifier model using the Keras CIFAR-10 CNN-CAPSULE documentation (CIFAR-10). This involves building a function to squash our input data in reference to Geoffrey Hinton's paper, however we utilize 0.5 to calculate our squash scale instead of 1, define a margin loss function and SoftMax function and build a capsule layer activated by the squash function, the capsules input shape consists of a batch size, number of input capsules and number of input dimensions, we build a one (1) dimensional convolution model consisting of two 32 neurons layers and two 64 neurons layers, a pooling layer with a pool size of two (2) and is activated by the rectifier activation function to serve as input for our capsule layer. The capsules final output is the length of two (2) capsules and 16 dimensions and is compiled using stochastic gradient descents “adam” optimizer which splits a dataset into multiple subsets of training cases and executes modification of the data features of each training case to identify the

features that produce the best results, the loss is measured using the defined margin loss function. We proceed to split our dataset into 70% training and 30% test sets and the CapsNet classifier model is trained on 100 epochs with a batch size of 1000 records each using our taxi trip training set consisting of our independent data variables.

Figure 42. Ride-sharing Classification Model Architecture.



Bagging considers homogenous weak learners and executes learning by training the weak learners independently in a parallel manner then combines the results of the weak learners using a deterministic averaging process, bagging creates several subsets of data from a training sample chosen randomly with replacement and uses a decision tree to train the individual subset, boosting ensemble techniques learn sequentially and adaptively whereas a base model depends on the previous base model, models are then combined using a deterministic strategy, stacking combines the training and testing predictions of several models and uses the stacked training and test data to train a meta-model to output a prediction based on the training and test predictions of the base

models. For our ride-sharing prediction proposed solution we employ bagging and boosting on our base models to reduce variance and bias and then utilize stacking ensemble techniques over voting to improve our models' prediction accuracy. Finally, we utilize a random forest model to make the final predictions due to the performance from our experiments.

From our ride-sharing prediction experiments, we observe that our classification models predictive accuracy is limited due to our data being highly imbalanced, we observe that the shared rides data in our January 2017 dataset was barely 20% of the training and test datasets, we proceed to exercise some data engineering techniques to balance our dataset. We extract shared rides information from January 2019 to June 2019 to boost or classification models predictive accuracy.

Fig 42 is an architecture diagram illustrating the implementation of the ride-sharing classification algorithm employed for predicting shared rides.

Chapter 7: Evaluation

In chapter 7, we provide detailed information on the evaluation metrics utilized for validating the proposed solutions, we evaluate our classification and regression solutions for ride-sharing prediction and arrival time prediction respectively, and we execute functional testing, efficiency testing, and complexity analysis to provide a detailed understanding of our proposed solution's performance. We also compare the evaluation results of our thesis work with the evaluation results of some of the related works in arrival time prediction and ride sharing/matching.

7.1 Functional testing.

The functional testing of our proposed solutions involves evaluating our classification and regression models using the industry-standard metrics, for arrival time prediction we utilize Scikit-learns' regression evaluation metrics and employ Scikit-learns' confusion matrix and classification report for evaluating our ride-sharing prediction model. The following is a description of the evaluation metrics utilized for both solutions to provide more information as to why they were employed.

7.1.1 Arrival time prediction evaluation metrics.

Explained variance score: variance refers to how far the actual values are from the average of the predicted values, we utilize the Scikit-learn regression metrics library for our calculation.

R^2 score: this refers to the percentage of variation between two variables, the best fit is denoted as the (*line*), a high R2 score signifies that the independent data features are good predictors of the target output.

$$R^2 = \big(var(mean) - var(line)\big) \,/\, var(mean)$$

Mean squared error (MSE): this refers to the average of the squares of the errors, the larger the

MSE recorded the higher the errors of our regression model.

$$MSE = \frac{1}{n}\sum(y - \hat{y})^2$$

Mean absolute error (MAE): this refers to the average magnitude of the errors in our model prediction.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} | y_i - \hat{y}_i |$$

We evaluate our convolutional neural network (CNN) regression model using the Scikit-learn regression evaluation metrics using the January and February 2017 test set.

Table 40. Convolutional Neural Network (CNN) regression model evaluation.

Convolutional Neural Network (CNN) Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	8.877e-06	0.0016	99%
February 2017 test set predictions (Un-Trained)	99%	1.108e-05	0.0008	99%

Figure 43. CNN regression models MSE evaluation results.

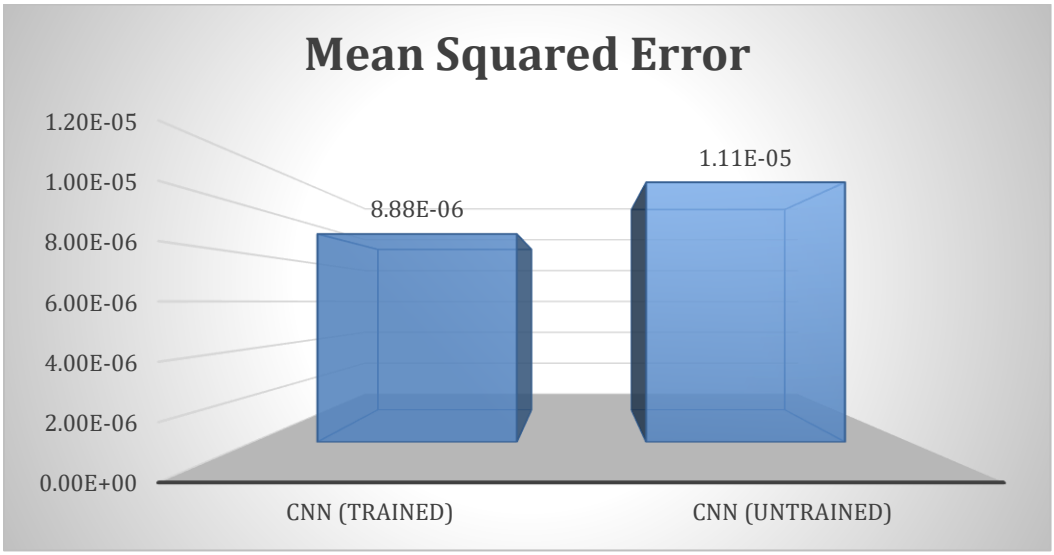
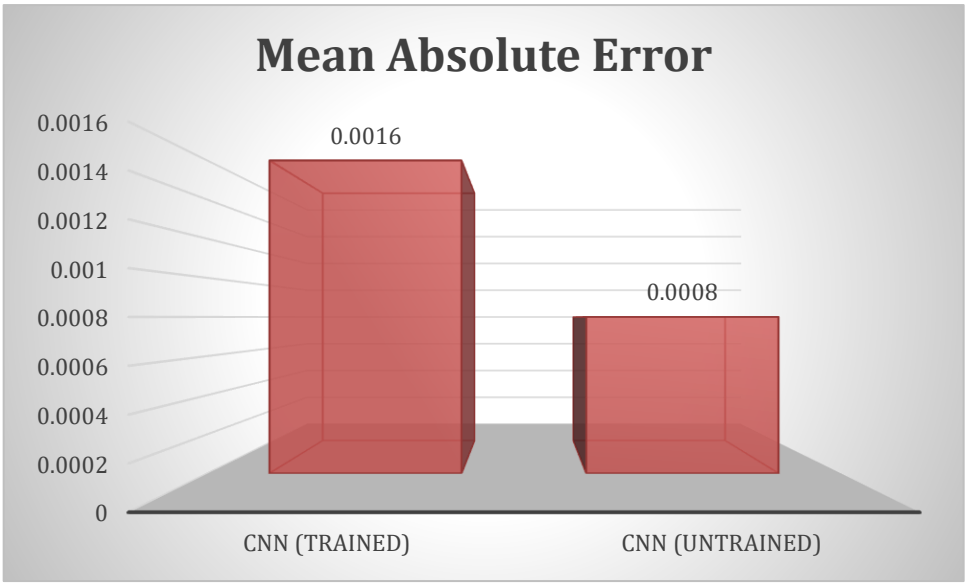


Figure 44. CNN regression models MAE evaluation results.



We evaluate our convolutional neural network (CNN) model using Scikit-Learn regression metrics, our aim is to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible, from our evaluation results, our CNN regression model records 99% accuracy for explained variance score (EVS) and R2 Score for the January and February 2017 taxi trip test sets which is similar to the performance

of our machine learning models, we also record relatively similar mean squared error (MSE) and mean absolute error (MAE) scores for both trained (January 2017) and untrained (February 2017) test sets in comparison to our machine learning regression models, we proceed to boosting our CNN regression model with an extreme gradient boosting (XGBoost) regression model to improve prediction accuracy.

- Evaluate XGBoost model using January 2017 test set.

Table 41. Convolutional Neural Network (CNN) XGBoost regression model evaluation.

Convolutional Neural Network (CNN) (XGBoost) Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	6.711e-06	0.0008	99%
February 2017 test set predictions (Un-Trained)	99%	1.353e-05	0.0022	99%

Figure 45. CNN (XGBoost) regression models MSE evaluation results.

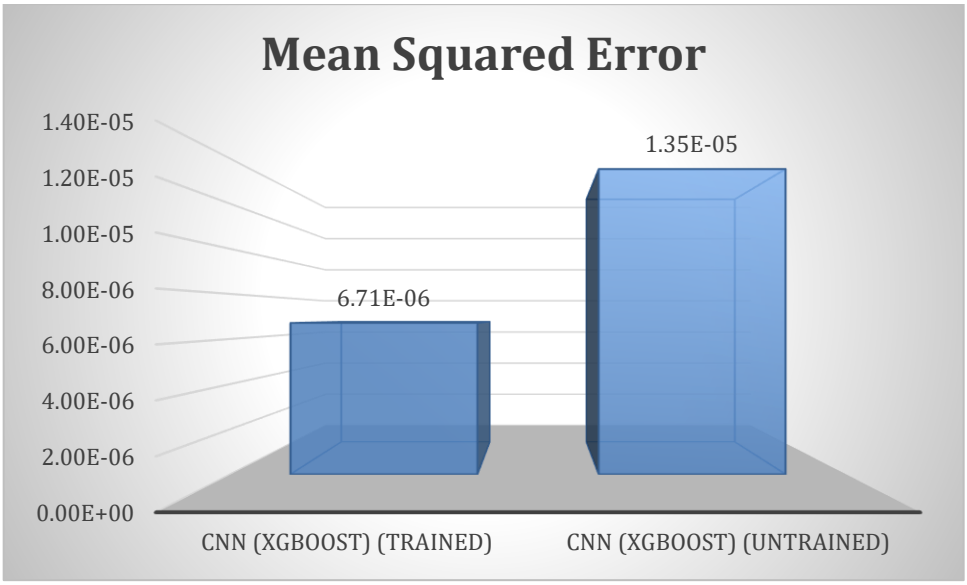
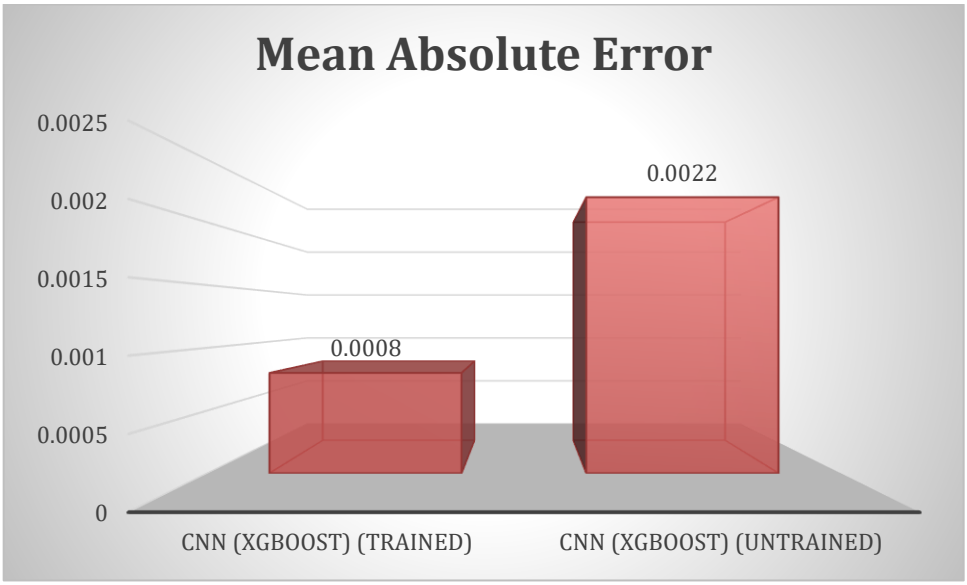


Figure 46. CNN (XGBoost) regression models MAE evaluation results.



We evaluate our convolutional neural network (CNN) model using Scikit-Learn regression metrics, we aim to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible, from our evaluation results, our CNN regression model records 99% accuracy for explained variance score (EVS) and R2 Score for the January and February 2017 taxi trip test sets, we also observe better uniformity with regards to trained and untrained test sets, the MAE and MSE recorded for trained sets are lower in comparison to the original CNN regression model.

The following represents our proposed solution final evaluation which is a stacked ensemble of our four (4) regression models i.e. a convolutional neural network (CNN) boosted using XGBoost, Bagging, AdaBoost and gradient boosting regression models stacked together using Scikit-learn stacking library.

- Evaluate the stacked ensemble regression model with our January 2017 test set.

Table 42.Stacked ensemble regression model evaluation.

Stacked Ensemble Model Evaluation.				
	Explained Variance Score	Mean Squared Error	Mean Absolute Error	R2 Score
January 2017 test set predictions (Trained)	99%	6.796e-06	0.0001	99%

Figure 47. Comparison of regression models MSE evaluation results.

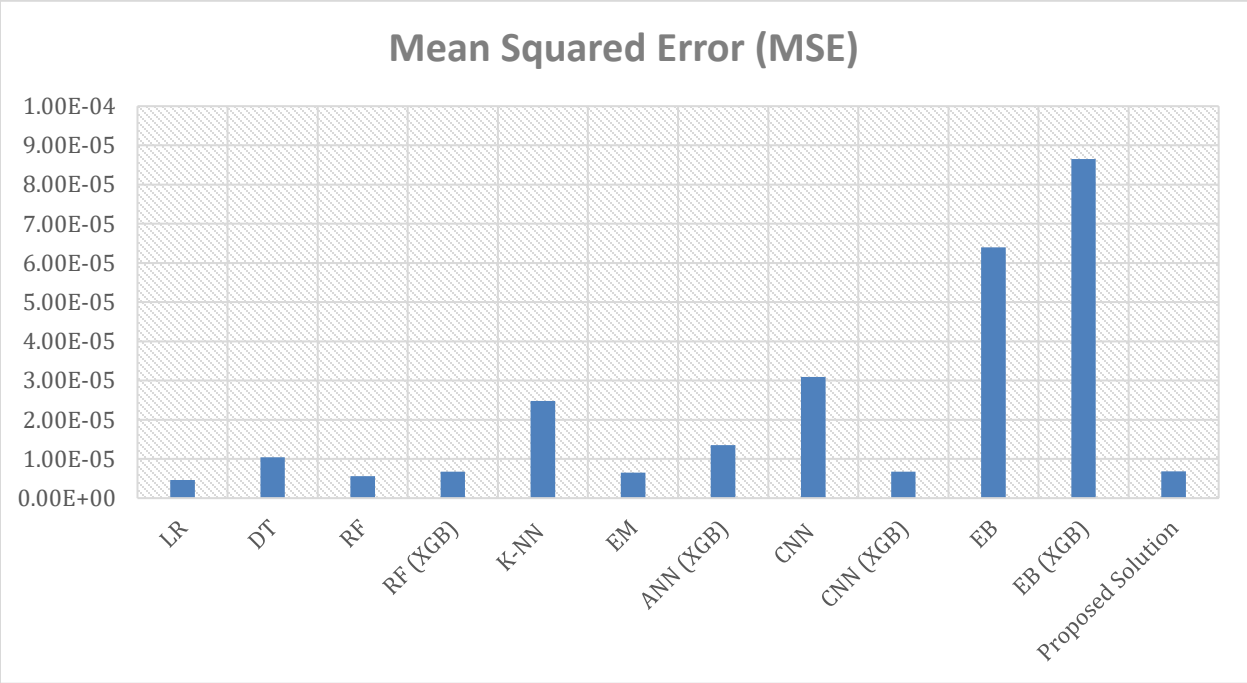
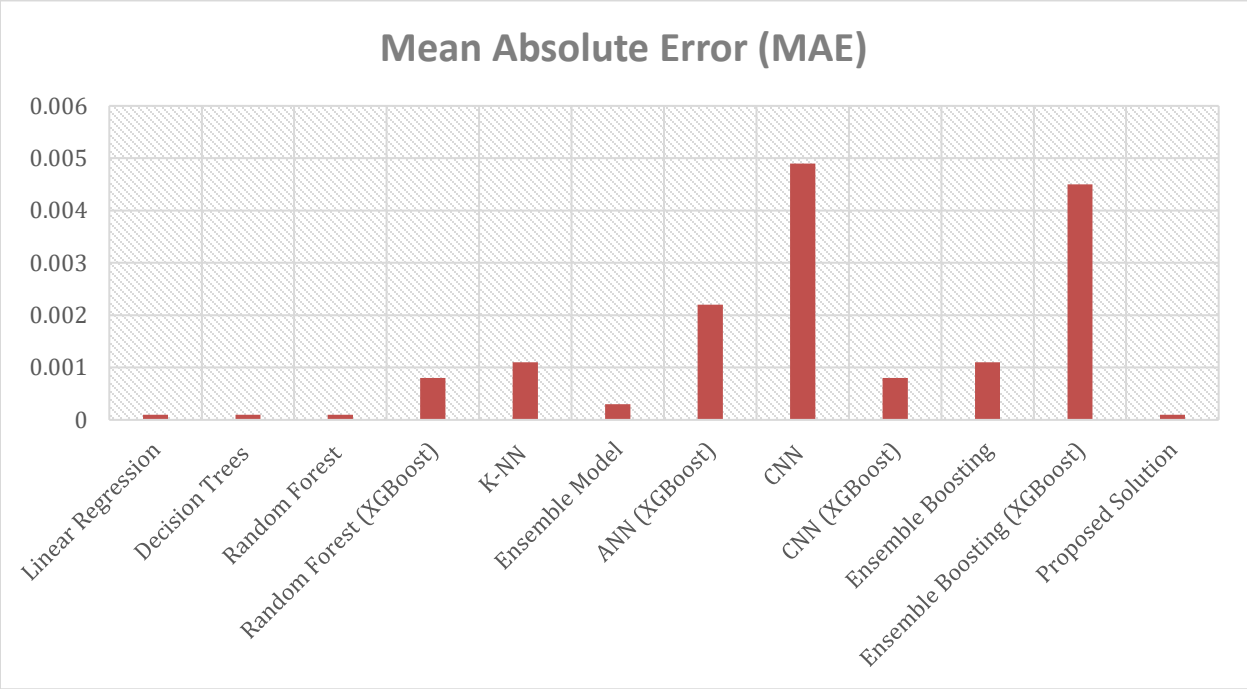


Figure 48. Comparison of regression models MAE evaluation results.



We evaluate our stacked ensemble model using Scikit-Learn regression metrics, we aim to record a high explained variance score and R2 score as well as to record a mean squared error (MSE) and mean absolute error (MAE) as low as possible, from our evaluation results, our proposed solution maintains 99% accuracy for explained variance score (EVS) and R2 Score for the January 2017 taxi trip test sets, we also observe that the machine learning models record similar accuracy to our proposed solution however we prefer to utilize our ensemble model which is a combination of machine learning and deep learning models. The extreme gradient boosting (XGBoost) model employed also ensures loss is minimally reduced and the ensemble techniques guarantee reduced variance and bias, the ensemble model also provides a robust model capable of maintaining high accuracy levels in the event of data scaling and possible new input data features.

7.1.2 Ride-sharing prediction Evaluation

We evaluate our ride-sharing prediction model using Scikit-learns' classification report and confusion metrics to observe the predictive capabilities of our classification model.

Classification report: we utilize the classification report available on Scikit-learn to evaluate the classification models using the following metrics.

Precision: this refers to our model's ability to accurately not classify our positive target labels as negative labels, this can be calculated using the below formula.

$$Precision = \frac{True\ positive}{True\ Positive + False\ positive}$$

Recall: this refers to the ability of our classification model to accurately predict all our positive labels and is calculated using the below formula.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

F1-Score: this refers to the harmonic mean of the precision and recall, the f1-score is calculated using the below formula.

$$F1score = 2\ \times \frac{Precision * Recall}{Precision + Recall}$$

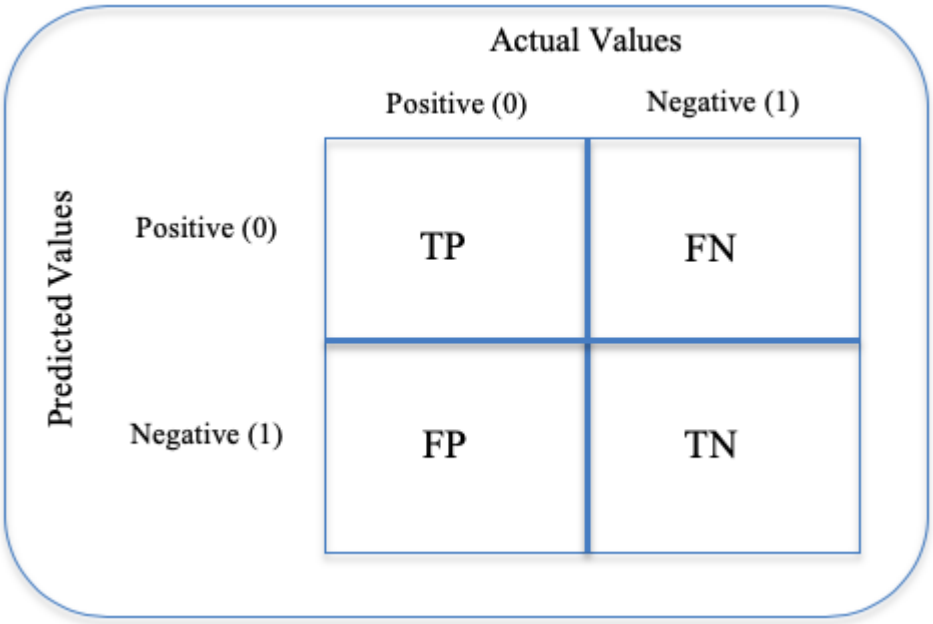
Receiver Operating Curve (ROC): this refers to a probability curve that represents the accuracy of a classification model in properly identifying classes with regards to the true-positive and false-positive rates.

Area Under Curve (AUC): this refers to the area under our ROC curve ranging from 0 to 1.

Confusion Matrix: this refers to a performance measure of classification algorithms where the output can be either binary or categorical, it is made up of a table with 4 different combinations of predicted and actual values. Utilizing the confusion matrix on Scikit-learn we represent the

predicted shared rides as our positive value denoted as (0) and the predicted values of the unshared rides are presented as the negative predictions denoted as (1).

Figure 49. Confusion matrix for the classification algorithm.



True Positive: this refers to all the positive values (shared rides) predicted as positive values (shared rides) by our classification model, this is denoted as TP.

False Negative: this refers to the positive values (shared rides) predicted as negative values (unshared rides), this is denoted as FN.

False Positive: this refers to negative values (unshared rides) predicted as positive values (shared rides), it is denoted as FP.

True Negative: this refers to negative values (unshared rides) predicted as negative values (unshared rides), it is denoted as TN.

We proceed to evaluate the individual models leading to our ride-sharing prediction proposed solution, the evaluation is executed based on the implementation steps discussed in detail in chapter 6, we begin by evaluating our capsule network (CapsNet) classification model.

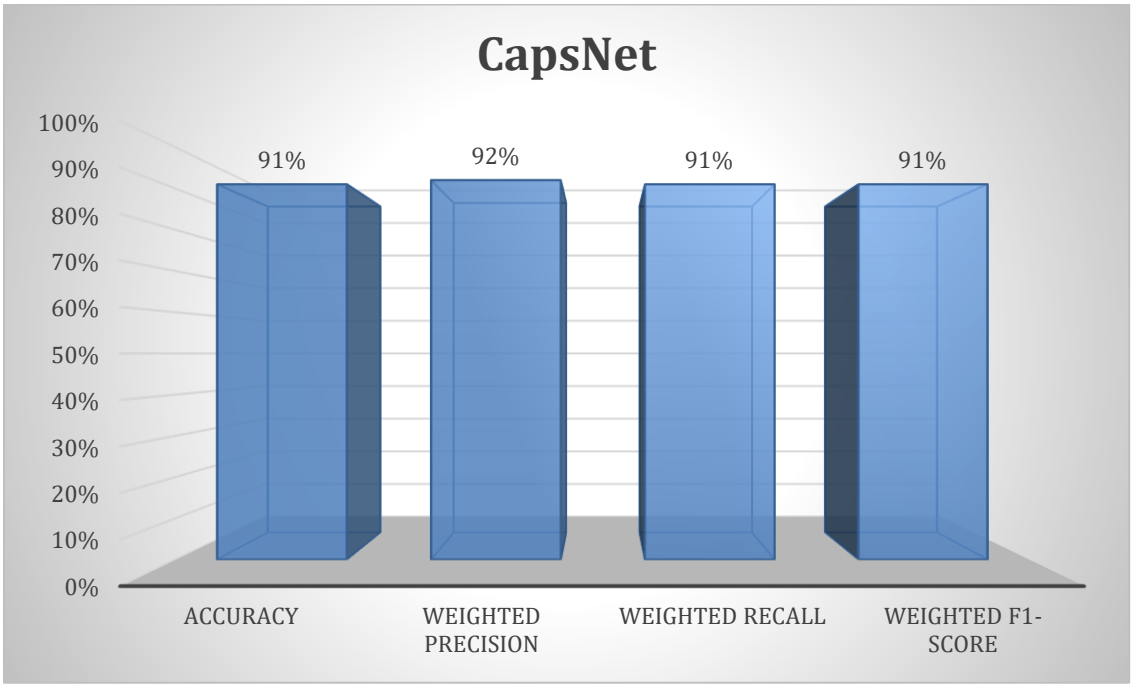
Table 43. Capsule Network (CapsNet) model classification report.

Classification report of CapsNet on the test set				
	Precision	Recall	F1-Score	Support
0	100%	81%	89%	135,032
1	86%	100%	93%	163,748
Accuracy			91%	298,780
Macro Avg	93%	90%	91%	298,780
Weighted Avg	92%	91%	91%	298,780

Table 44. Capsule Network (CapsNet) model confusion matrix.

Confusion Matrix of CapsNet on the test set		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 108,799	FN = 26,233
Actual: NO	FP = 50	TN = 163,698

Figure 50. Evaluation of the CapsNet classification model.



We evaluate our CapsNet classification model using Scikit-Learn confusion matrix and classification report, we observe an impressive improvement in the predictive accuracy of our ride-sharing model, notably, we can accurately predict shared and unshared rides in comparison to our ride-sharing prediction experiments in chapter 5. We record overall model accuracy of 91% meaning our model still misclassifies some target labels however the ability of our model to accurately predict 108,799 shared rides from 135,032 is a good step in the right direction, we also observe that only 50 un-shared rides are misclassified, we proceed to apply extreme gradient boosting (XGBoost) to improve or ride-sharing prediction accuracy.

Table 45.Capsule Network (CapsNet) (XGBoost) classification model classification report.

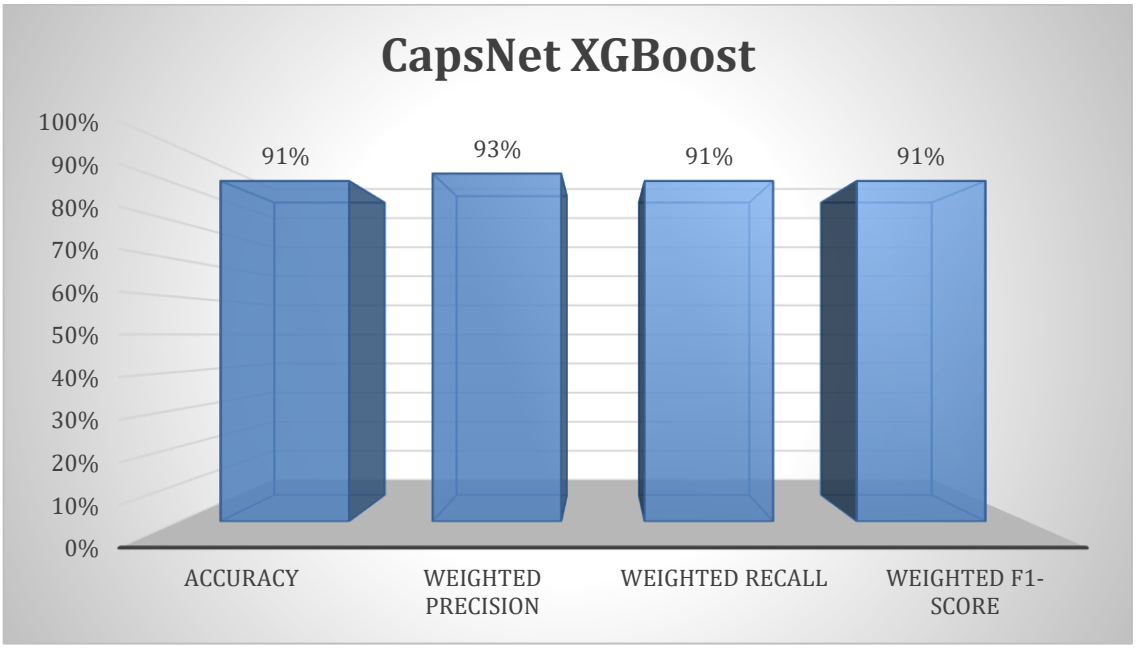
Classification report of XGBoost (CapsNet) on the test set				
	Precision	Recall	F1-Score	Support
0	100%	81%	89%	135,032
1	86%	100%	93%	163,748

Accuracy			91%	298,780
Macro Avg	93%	90%	91%	298,780
Weighted Avg	93%	91%	91%	298,780

Table 46. Capsule Network (CapsNet) (XGBoost) classification model confusion matrix.

Confusion Matrix of XGBoost (CapsNet) on the test set		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 109,289	FN = 25,743
Actual: NO	FP = 28	TN = 163,720

Figure 51.Evaluation of CapsNet (XGBoost) classification model.



The notable observations from applying extreme gradient boosting (XGBoost) on the CapsNet model is the improvement of our weighted precision from 92% to 93%, this leads to reduced misclassification hence we boost the predicted shared rides from 108,799 to 109,289, the number of misclassified unshared rides also drops from 50 to 28 rides.

We proceed to evaluate our stacked ensemble boosting model of an adaptive boosting (AdaBoost) classifier model, bootstrap aggregation (Bagging) classifier model, and our gradient boosting (GB) classifier model combined using Scikit-learns' ensemble library stacking classifier.

Table 47. Stacked ensemble boosting model classification report.

Classification report of Stacked Ensemble boosting on the test set				
	Precision	Recall	F1-Score	Support
0	100%	81%	89%	135,032
1	86%	100%	93%	163,748
Accuracy			91%	298,780
Macro Avg	93%	90%	91%	298,780
Weighted Avg	93%	91%	91%	298,780

Table 48. Stacked ensemble boosting model confusion matrix.

Confusion Matrix of Stacked Ensemble boosting on the test set		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 109,297	FN = 25,735
Actual: NO	FP = 46	TN = 163,702

Our stacked ensemble boosting classifier model records similar prediction accuracy to our XGBoost (CapsNet) classifier model concerning the weighted precision, recall, and F1-score, from the results of our confusion matrix we observe that it is outperformed by the XGBoost (CapsNet) classifier model for predicting unshared rides with 46 misclassifications compared to 28 but slightly performs better for predicting shared rides with 109,297 compared to 109,289 from the XGBoost (CapsNet)model.

We proceed to evaluate our proposed solution which is a stacked ensemble of our XGBoost (CapsNet) model and our stacked ensemble boosting classifier models using numerical processing (NumPy) column stacking function. The following is a representation of the evaluation results of our proposed solution utilizing the above-discussed evaluation metrics available on Scikit-learn, we outline the results as obtained from the classification report and confusion matrix.

Table 49. Proposed solution classification report.

Classification report of proposed solution for predicting shared rides.				
	Precision	Recall	F1-Score	Support
0	100%	81%	89%	135,032
1	86%	100%	93%	163,748
Accuracy			91%	298,780
Macro Avg	93%	90%	91%	298,780
Weighted Avg	93%	91%	91%	298,780

Table 50. Proposed solution confusion matrix.

Confusion Matrix of proposed solution for predicting shared rides		
	Predicted: YES	Predicted: NO
Actual: YES	TP = 109,273	FN = 25,759
Actual: NO	FP = 6	TN = 163,742

Figure 52. ROC curve for the classification algorithm.

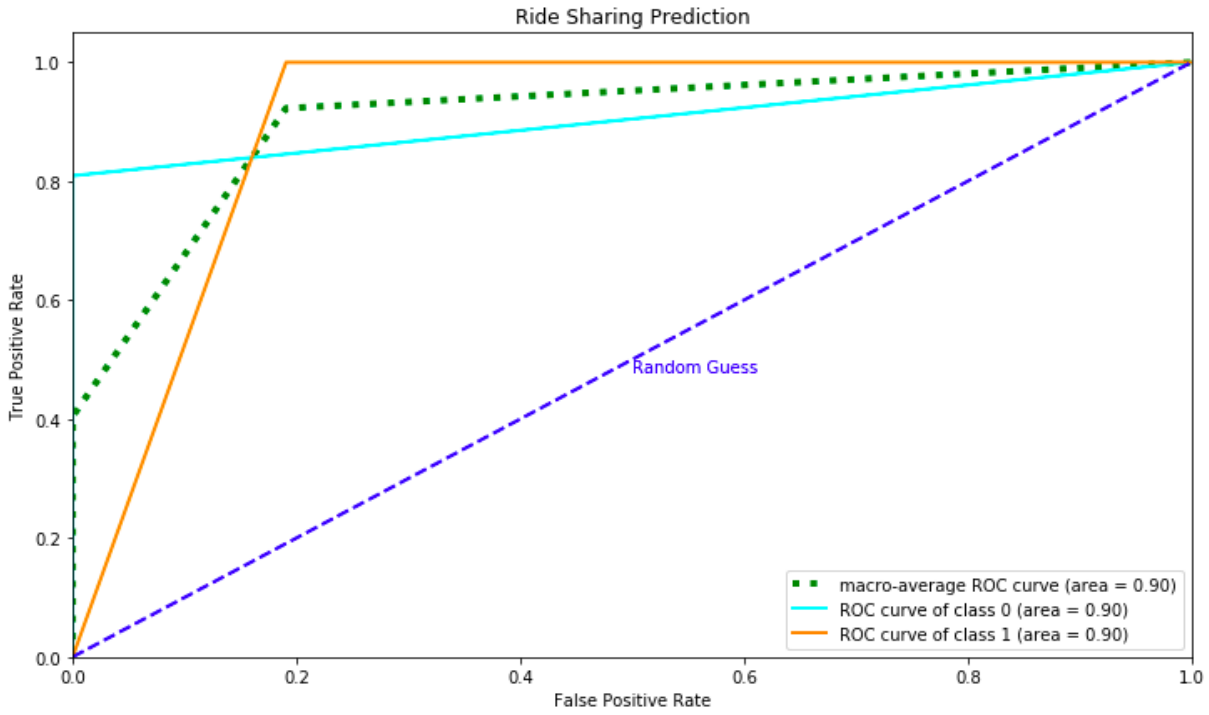


Figure 53. Accuracy comparison for classification algorithms.

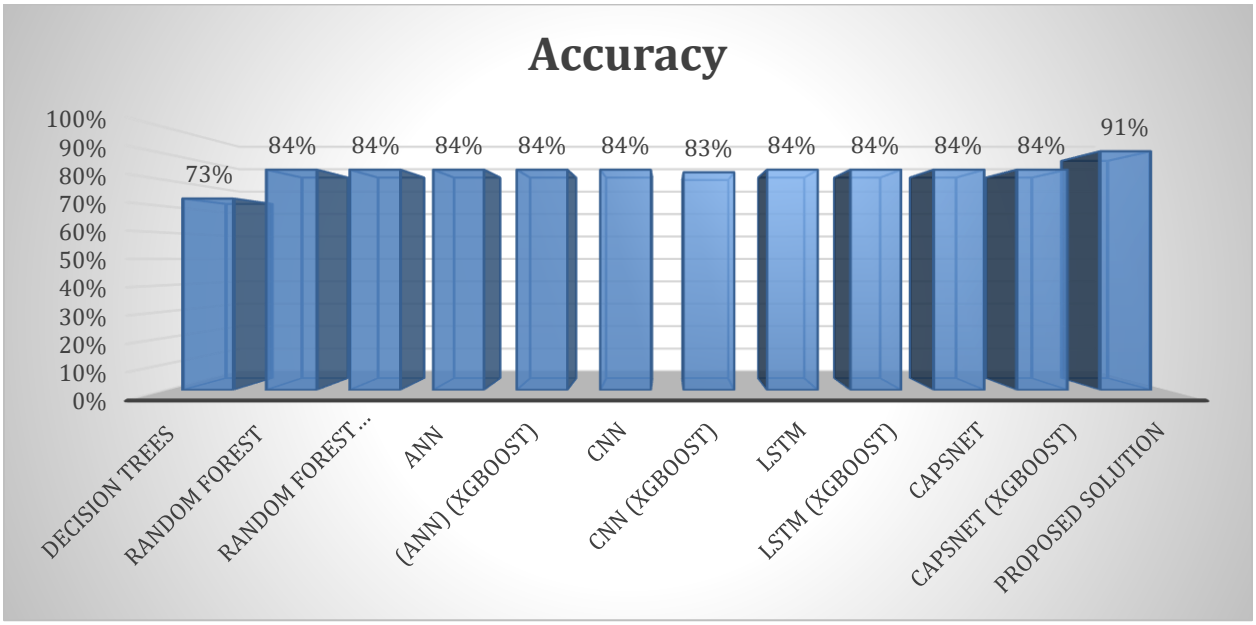


Figure 54. Weighted precision comparison for classification algorithms.

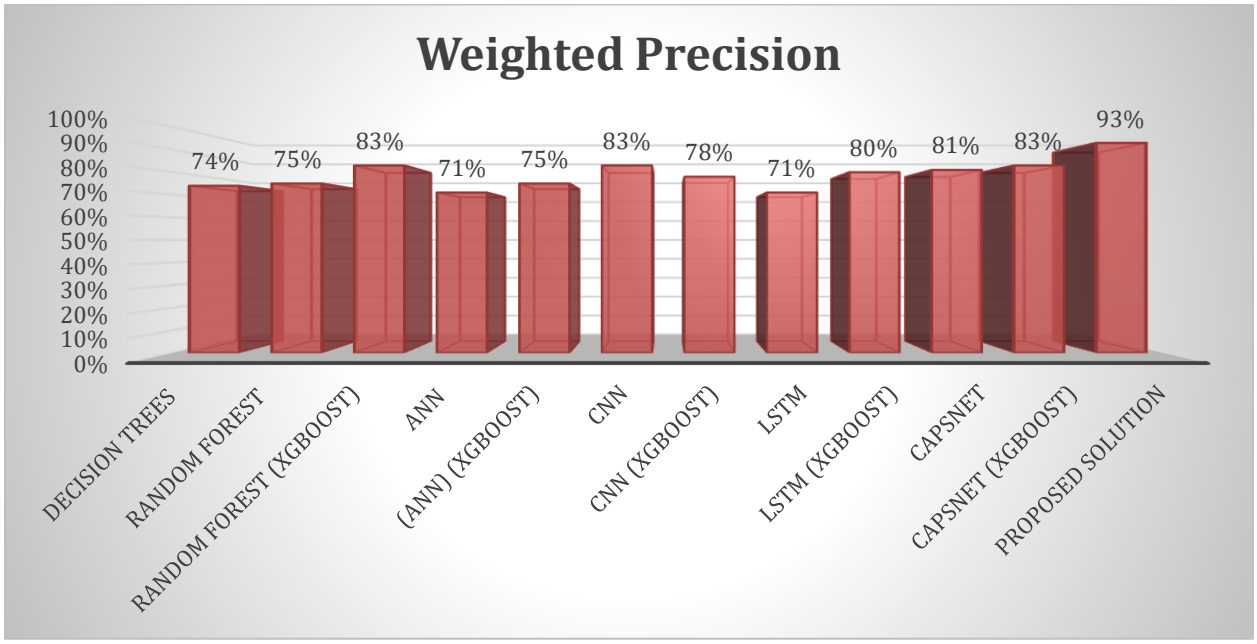


Figure 55. Weighted recall comparison for classification algorithms.

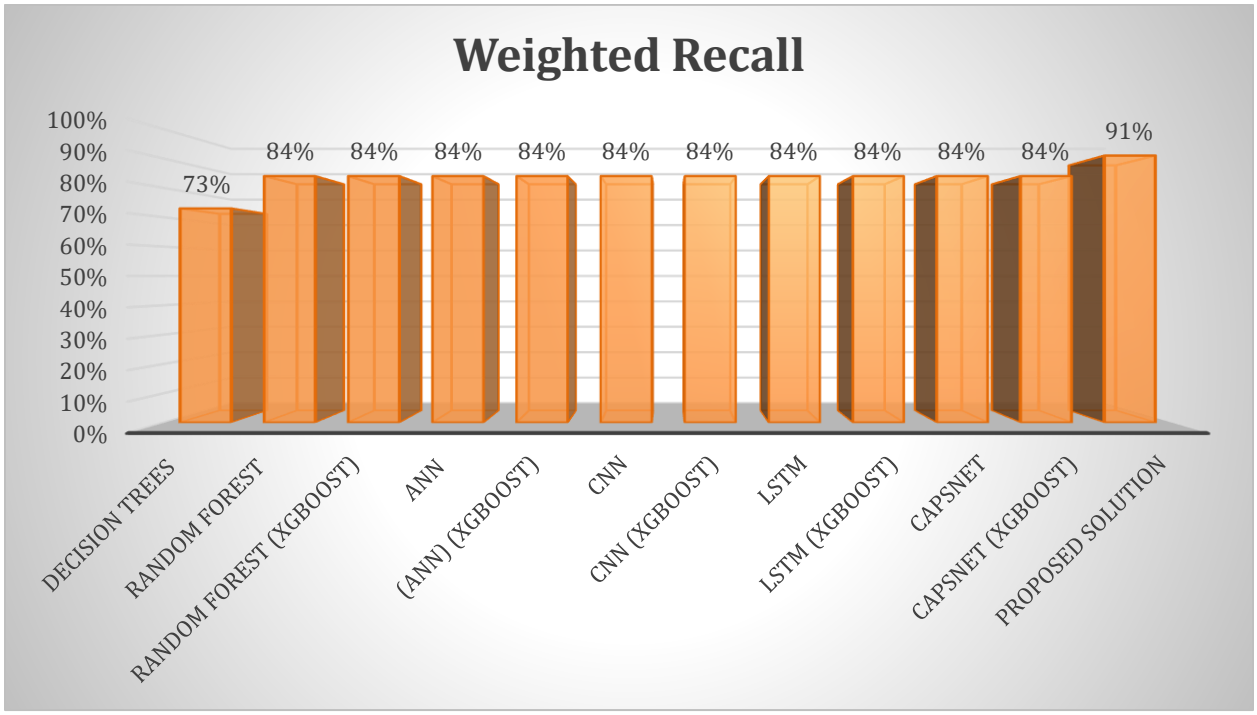
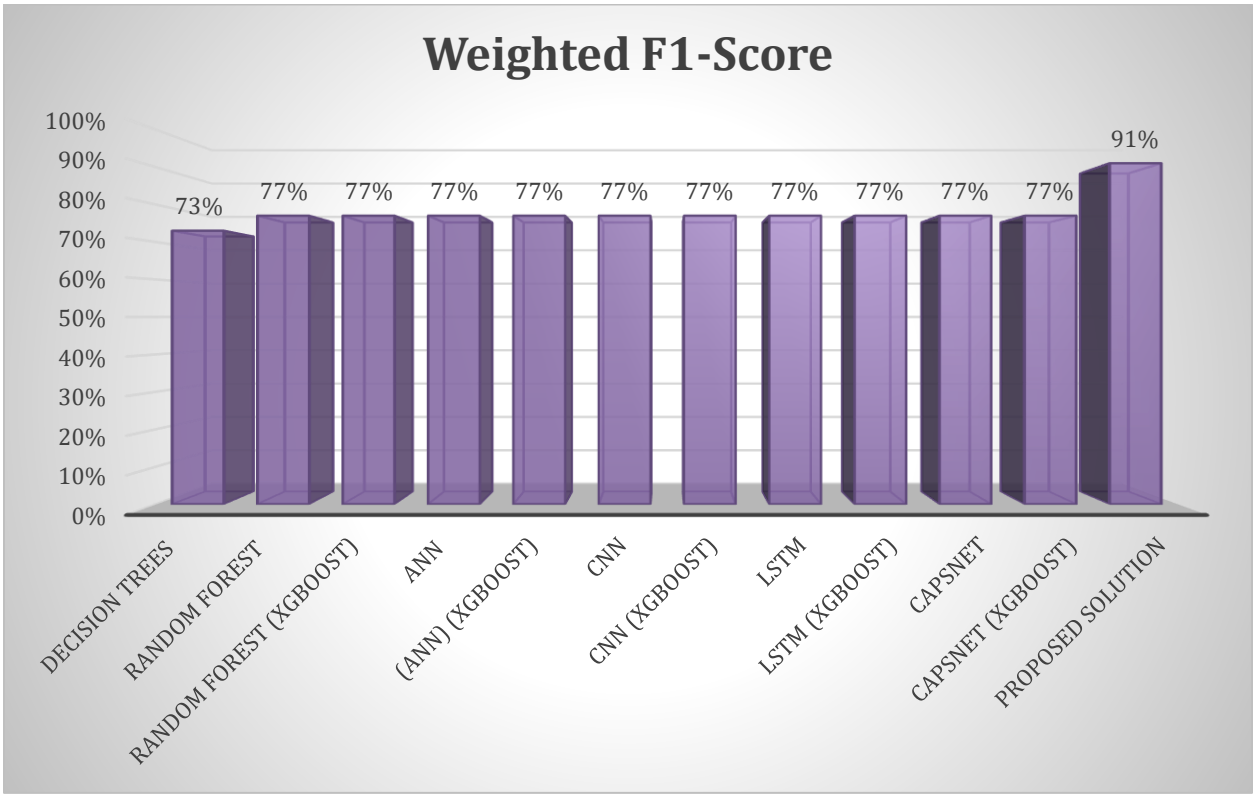


Figure 56. Weighted F1-score comparison for classification algorithms.



We evaluate our proposed solution classification model using Scikit-Learn confusion matrix and classification report, we aim to record a high accuracy, precision, recall, and F1 scores, the precision score is a measurement of the ability of our model to accurately not classify our positive target labels as negative labels, recall refers to the ability of our model to capture all our positive labels and f1-score is the harmonic mean of the precision and recall, from our evaluation results, the above classification report and confusion matrix highlights the improvement of our ride-sharing prediction proposed solution in comparison to our experiments in chapter 5, it is worth noting that we employ the same ride-sharing classification model as utilized in our experiments, however, we boost our prediction accuracy for every metrics thanks to the data engineering employed to have a more balanced dataset in comparison to our experiments. The overall outcome of our proposed solution is the ability to accurately predict shared rides without passenger count

information even though we still have an acceptable level of misclassification in our confusion matrix, we observe that our proposed solution only misclassifies 6 unshared rides and we can predict 109,273 shared rides from 135,032 which is a great improvement in comparison to our experiments.

7.2 Efficiency testing

We execute efficiency testing of our arrival time and ride-sharing prediction models to have a clear understanding of the computational resources required to execute our regression and classification algorithms, we also provide an astute breakdown of the runtime of all the key processes i.e. model training, data preprocessing. The proposed system offers a reasonable level of scalability, and for comparison purposes, we also execute the proposed solutions on a MacBook Air laptop, running on 250GB hard disk and 8GB memory, the runtime and evaluation results recorded are similar to the results on Google Colab, however, we prefer utilizing Google Colab for executing our solutions due to its cloud computing capacity.

Table 51. Arrival time prediction efficiency testing.

Arrival time prediction efficiency testing	
Data preprocessing	30 seconds
Convolutional neural network (CNN) model training	6 minutes 57 seconds
XGBoost (CNN) model training.	1 minute 9 seconds
Ensemble Model (StackingRegressor) model training	14 minutes 24 seconds
XGBoost (CNN) model stacking	10 seconds
Ensemble model stacking	15 seconds
Meta learner model training	1 minute 6 seconds

Total algorithm runtime	23 minutes 6 seconds
Memory/RAM consumption	1.37 GB
Resource consumption	Google Colab 20 GB
CPU / computing cycles consumption	12 GB RAM

Table 52. Ride-sharing prediction efficiency testing.

Ride-sharing prediction efficiency testing	
Data preprocessing	1 minute
Capsule Network (CapsNet) model training	42 minutes
XGBoost (CapsNet) model training.	53 seconds
AdaBoost model training	2 minutes
Bagging model training	1 minute 11 seconds
Gradient boosting model training	4 minutes 4 seconds
Ensemble Model (StackingClassifier) model training	40 minutes
XGBoost (CapsNet) model stacking	3 minutes 38 seconds
AdaBoost model stacking	7 minutes 24 seconds
Bagging model stacking	5 minutes 10 seconds
Gradient Boosting model stacking	16 minutes 49 seconds
Meta learner model training	16 seconds
Total algorithm runtime	2 hours 10 minutes
Memory/RAM consumption	2.19 GB
Resource consumption	Google Colab 20 GB
CPU / computing cycles consumption	12 GB RAM

We execute both solutions on Google Colab utilizing the free computational resources provided by Google, we run our solution on a cloud computing GPU server hosting Python 3 and consisting of 12GB memory and 100GB disk space. The arrival time prediction model takes about 23 minutes and the ride-sharing solution took about 2 hours and 10 minutes, however, we need to highlight that the Scikit-learn ensemble model was only created for the comparison and is therefore not an integral part of our ride-sharing prediction solution hence the overall runtime for the ride-sharing prediction solution comes down to 1 hour and 30 minutes.

7.3 Complexity analysis

Big O notation for the worst-case scenario (time-complexity, and space-complexity): time complexity refers to an approach for measuring how an algorithm runtime increases in response to the input data, for our proposed solutions for ride-sharing and arrival time prediction we observe a constant time complexity denoted as O(*1*) for preprocessing our dataset into a machine-readable form using a Minmax scaler for arrival time prediction and the label encoder from Scikit-learns' libraries for ride-sharing prediction, the constant time complexity of O(*1*) expresses a scenario whereby an increase in our input data would not increase the time required for data preprocessing. The process of training our arrival time prediction model represents a linear time complexity denoted as O(*n*) where n is the size of our input data, considering our final regression model is an ensemble of two (2) models, we denote the time complexity of the CNN (XGBoost) regression model as O(n), we also denote the stacked ensemble model as O(n). The total time complexity for training our arrival time prediction model is denoted as T_1.

$$T_1 = O(n) + O(n)$$

$$T_1 = O(n)$$

We observe a similar time complexity for our ride-sharing prediction model, our final ensemble is a stack of a CapsNet (XGBoost) classification model with time complexity of O(n), an AdaBoost model O(n), Gradient Boosting O(n) and Bagging model O(n), the total time complexity of our ride-sharing prediction model is denoted as T_2.

$$T_2 = O(n) + O(n) + O(n) + O(n)$$

$$T_2 = O(n)$$

A linear time complexity expresses a linear relationship between the size of our input data and the time required to completely execute our algorithms whereby an increase in the number of taxi trips data utilized for our arrival time prediction and ride-sharing prediction would result in increased runtime for training our regression and classification prediction algorithms, hence the overall time complexity for training our proposed solutions is denoted as O(n).

7.4 Comparison of the results of our proposed solution with the results of other related works

We carry out a comparative analysis of our evaluation results and other related works, we compare research works in ride-sharing and arrival time prediction to observe how our proposed solutions compare to other works in building intelligent transportation systems (ITS).

7.4.1 Comparison of Complexity analysis

The complexity analysis of our arrival time prediction and ride-sharing prediction proposed solutions presents a linear time complexity denoted as $O(n)$, this shows that our proposed solutions are scalable, and the processing time and computational resources required are to be configured based on the input data features and parameters, in (de Lira et al. 2018) the complexity analysis

was not discussed, however, the proposed solution is an activity-based ride matching (ABRM) algorithm whereby the transportation system proposes an alternative location where the selected activity can be performed, this action requires re-computation of the trip trajectory, therefore we believe the ABRM algorithm presents a quadratic time complexity and could be denoted as $O(N^2)$.

7.4.2 Comparison of Functional analysis

We compare the evaluation results of our related works to identify possible trends in our functional testing and that of related works in transportation travel time prediction and ride-sharing activities. Our arrival time prediction solution utilizes the internal transportation factors captured during the trip by a taximeter, we utilize a stacked deep ensemble of a convolutional neural network (CNN), boosted using extreme gradient boosting (XGBoost), a Bagging, AdaBoost and Gradient boosting model. We record a mean squared error of 6.796e-06, a mean absolute error of 0.0001 an explained variance score of 99% and r^2Score of 99%. In (Chen et al, 2019) they employ a convolutional and long short-term memory neural network (ConvLSTM) for bus travel time prediction, they noted that their proposed solution required high computational resources for their training activities, they recorded a mean absolute error of 2.79. In (Seungwoo et al, 2014), the authors developed a spatiotemporal prediction map where the elements in the map are represented as a time-series forecasting method, they recorded an R-squared score of 75%. In (Agafonov and Yumaganov, 2019) they evaluate their bus arrival time prediction using the mean squared error in seconds, they record 26.25 for their LSTM network, 34.186 for the artificial neural network and 42.77 for their linear regression model.

We observe that our taxi arrival time prediction model records a very low mean squared error in comparison to the related works, we believe that utilizing only the internal traffic factors recorded in our dataset led to very low errors, the related works focused on arrival time prediction for buses

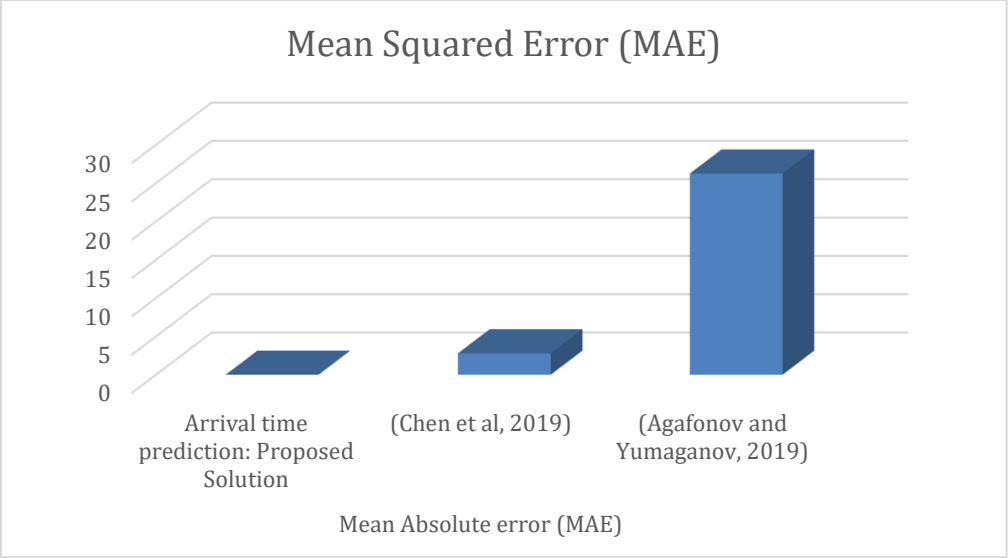
and several research works utilize external traffic factors i.e. weather condition, social media feed etc.

In (Singh et al, 2019) the authors validate better vehicle utilization by 20%, reduce idle time and wait times by 40% and 30% less operational costs, we are yet to execute simulations of our proposed solution to have a realistic view of the exact cost-saving metrics of our ride-sharing prediction solution.

Table 53. Comparison of Arrival time prediction model MAE with related works.

Mean Absolute error (MAE)		
Arrival time prediction: Proposed Solution	(Chen et al, 2019)	(Agafonov and Yumaganov, 2019)
0.0001	2.79	26.25
$\boldsymbol{r^2 Score}$		
Arrival time prediction: Proposed Solution	(Seungwoo et al, 2014)	NA
99%	75%	NA

Figure 57.Comparison of Arrival time prediction model MAE with related works.



It should be noted that the dataset and the internal and external traffic factors utilized in our analysis and prediction activities vary from the datasets utilized in the related works. However, we proceed to carry-out the comparative analysis to have an understanding from a broader perspective about the variations in the evaluation results for different modes of transportation, i.e., buses and taxis.

7.4.3 Comparison of Efficiency analysis

We execute our proposed solution on Google Colab, utilizing the open-source online Python 3 environment, the online IDE provides 12 GB RAM and 100GB disk storage space, our arrival time prediction algorithm executes for 23 minutes utilizing 1.37GB of memory while the ride-sharing prediction algorithm executes for 1hour 30 minutes, consuming 2.19GB of memory. While we utilized Google Colab for executing our algorithms, In (Erika Ritzelle et al, 2018), the researchers used RapidMiner Studio to observe four machine learning algorithms (Naïve Bayes, Decision Tree, Random Forest, k-Nearest Neighbor).

Chapter 8: Conclusions and Future Work

In this thesis, we develop techniques that can contribute to a ride-sharing induced intelligent transportation system capable of predicting trip travel times and also predicting shared rides, we demonstrate the capability of our model to predict shared rides with 91% accuracy and 93% precision. The ability to predict taxi trip arrival times ensures that high level transportation planning can be executed by end-users and business operations, and the information we obtain from predicting shared rides can help identify the hotspots where ride-sharing is a more prevalent option in any urban or rural city. We highlight that this information could be key in boosting vehicle allocation and efficiency including operational management costs. The emergence of smart cities guarantees that developing intelligent transportation systems are no longer a thing of the future, the benefits of implementing these intelligent systems include reduces traffic congestions, reduced road accidents, sustainable environment via less co2 emissions, and lower operational costs.

We begin by providing an introduction and background into our understanding of arrival time prediction and ride-sharing prediction, the introduction is followed by an extensive literature review into the related works in developing intelligent transportation systems (ITS) and advanced traveler information systems (ATIS), we also execute a dataset review to gain insight into the preferred data set for executing machine learning and deep learning experiments, our experiments involved building both regression and classification algorithms for arrival time prediction and ride-sharing prediction respectively and we attain more knowledge on the preferred techniques to be utilized via our experiments evaluation results.

For arrival time prediction, we propose a stacked ensemble of a convolution neural network (CNN) boosted via extreme gradient boosting (XGBoost), an AdaBoost, Bagging, and Gradient Boosting

regression models. The stacked ensemble certifies that we can take full advantage of the predictive capability of the four (4) individual models, we reduce bias and variance via bagging and boosting and improve our arrival time prediction solutions accuracy via stacking.

In the case of ride-sharing prediction, we utilize a similar stacked ensemble technique, we propose a stack of the classifiers, utilizing the same ensemble boosting techniques, and for deep learning, we implement a capsule network (CapsNet) classifier algorithm. The key contributions of this research work include our ability to accurately predict taxi trip arrival times with minimal error as well as predicting shared rides without the passenger count information, we also implement a simple approach for stacking deep learning and machine learning regression and classification models together for higher prediction accuracy.

The experiments carried out in chapter five (5) provides a good platform for students interested in executing research utilizing machine learning and deep learning techniques to understand the various requirements for building both regression and classification algorithms, the documentation of the detailed experiments executed provides a step by step implementation of all the techniques attempted serving as a guide for future Data Science students, the entire thesis research work has provided valuable insight on the industry standards for building intelligent transportation system and has been a great learning experience.

In Chapter six (6) of this thesis, we propose an intelligent approach for predicting taxi trip arrival times and predicting shared rides without passenger count information, the techniques herein employ machine learning, deep learning as well as ensemble techniques. For our future work, we envision building a system that utilizes our arrival time prediction model and ride-sharing prediction model to execute trip simulations, we believe that via executing simulations of the

proposed solution, we can prove that there is room to share more rides, save cost for both driver and passenger and improve vehicle efficiency.

We would also look into synchronizing the taxi trip dataset with external traffic elements of the same period i.e. social media feeds, public events itinerary, weather conditions. We believe that these external factors could lead to novel information as well as provide a more accurate representation of real-life scenarios. The author of this work also believe that further research should be executed on how the proposed intelligent transportation system (ITS) can be implemented in an autonomous vehicle environment, we believe that AV's could produce better results and cost savings through the elimination of the driver requirements.

References

(Agafonov et al, 2019) Agafonov, A., & Yumaganov, A. (2019, July). Bus Arrival Time Prediction with LSTM Neural Network. In International Symposium on Neural Networks (pp. 11-18). Springer, Cham.

(Al-Abbasi et al, 2019) Al-Abbasi, A. O., Ghosh, A., & Aggarwal, V. (2019). Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 20(12), 4714-4727.

(Alajrami et al, 2020) Alajrami, E., Ashqar, B. A., Abu-Nasser, B. S., Khalil, A. J., Musleh, M. M., Barhoom, A. M., & Abu-Naser, S. S. (2020). Handwritten Signature Verification using Deep Learning.

(Ali, 2019) Arish Ali. Ensemble Machine Learning Techniques. 2019. Packt Publishing.

(Alisoltani et al, 2020) Alisoltani, N., Zargayouna, M., & Leclercq, L. (2020). A Multi-agent System for Real-Time Ride-sharing in Congested Networks. In Agents and Multi-agent Systems: Technologies and Applications 2019 (pp. 333-342). Springer, Singapore.

(Al Najada and Mahgoub, 2016) Al Najada, H., & Mahgoub, I. (2016, October). Autonomous vehicles safe-optimal trajectory selection based on big data analysis and predefined user preferences. In 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) (pp. 1-6). IEEE.

(Alonso-Mora et al, 2017) Alonso-Mora, J., Wallar, A., & Rus, D. (2017, September). Predictive routing for autonomous mobility-on-demand systems with ride-sharing. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 3583-3590). IEEE.

(Al-Saffar et al, 2017) Al-Saffar, A. A. M., Tao, H., & Talab, M. A. (2017, October). Review of deep convolution neural network in image classification. In 2017 International Conference on

Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET) (pp. 26-31). IEEE.

(Auer et. al., 2016) Auer, A., Feese, S., Lockwood, S., & Hamilton, B. A. (2016). History of intelligent transportation systems (No. FHWA-JPO-16-329). United States. Department of Transportation. Intelligent Transportation Systems Joint Program Office.

(Bathla et al, 2018) Bathla, K., Raychoudhury, V., Saxena, D., & Kshemkalyani, A. D. (2018, November). Real-time distributed taxi ride-sharing. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 2044-2051). IEEE.

(Bicocchi et al, 2015) Bicocchi, N., Mamei, M., Sassi, A., & Zambonelli, F. (2015, September). Opportunistic ride sharing via whereabouts analysis. In 2015 IEEE 18th International Conference on Intelligent Transportation Systems (pp. 875-881). IEEE.

(Bischoff et al, 2017) Bischoff, J., Maciejewski, M., & Nagel, K. (2017, October). City-wide shared taxis: A simulation study in Berlin. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (pp. 275-280). IEEE.

(Cangialosi et al, 2016) Cangialosi, E., Di Febbraro, A., & Sacco, N. (2016). Designing a multimodal generalised ride-sharing system. IET Intelligent Transport Systems, 10(4), 227-236.

(Cao et al, 2015) Cao, B., Alarabi, L., Mokbel, M. F., & Basalamah, A. (2015, June). Sharek: A scalable dynamic ride sharing system. In 2015 16th IEEE International Conference on Mobile Data Management (Vol. 1, pp. 4-13). IEEE.

(Chen and Guestrin, 2016) Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

(Chen et al, 2019) Chen, C., Wang, H., Yuan, F., Jia, H., & Yao, B. (2019). Bus travel time prediction based on deep belief network with back-propagation. Neural Computing and Applications, 1-15.

(Chen et al, 2016) Chen, T., He, T., Benesty, M., Khotilovich, V., & Tang, Y. (2016). Xgboost: Extreme Gradient Boosting; R Package Version 0.4-2; 2015.

(Cifar-10) CNN-Capsule Networks, Capsule Neural Network as implemented using online Keras documentation. https://keras.io/zh/examples/cifar10_cnn_capsule/

(Cui et al, 2018) Cui, Q., Wang, Y., Chen, K. C., Ni, W., Lin, I. C., Tao, X., & Zhang, P. (2018). Big data analytics and network calculus enabling intelligent management of autonomous vehicles in a smart city. IEEE Internet of Things Journal, 6(2), 2021-2034.

(Dandl et al, 2017) Dandl, F., Bracher, B., & Bogenberger, K. (2017, June). Microsimulation of an autonomous taxi-system in Munich. In 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS) (pp. 833-838). IEEE.

(de Lira et al, 2018) de Lira, V. M., Perego, R., Renso, C., Rinzivillo, S., & Times, V. C. (2018). Boosting ride-sharing with alternative destinations. IEEE Transactions on Intelligent Transportation Systems, 19(7), 2290-2300.

(Duan et al, 2019) Duan, Y., Wang, N., & Wu, J. (2019, July). Optimizing Order Dispatch for Ride-Sharing Systems. In 2019 28th International Conference on Computer Communication and Networks (ICCCN) (pp. 1-9). IEEE.

(Duo et al, 2017) Duo, F., Qiao, Y., & Yang, J. (2017, October). Analysis and modeling of ride-sharing service user behavior in urban area. In 2017 IEEE/CIC International Conference on Communications in China (ICCC) (pp. 1-6). IEEE.

(Edoardo et al, 2014) Edoardo Cangialosi, Angela Di Febbraro, Nicola Sacco, Alberto Baud. Generalized Ride-Sharing: An Enhanced Model and New Results. 2014 International Conference on Connected Vehicles and Expo (ICCVE).

(Erika Ritzelle et al, 2018) Erika Ritzelle P. Bondoc, Francis Percival M. Caparas, John Eddie D. Macias, Vileser T. Naculangga. An Intelligent Road Traffic Information System using Text Analysis in the Most Congested Roads in Metro Manila. 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM).

(Fényes et al, 2018) Fényes, D., Németh, B., Asszonyi, M., & Gáspár, P. (2018, June). Side-slip angle estimation of autonomous road vehicles based on big data analysis. In 2018 26th Mediterranean Conference on Control and Automation (MED) (pp. 849-854). IEEE.

(Friedman, 2016) Friedman, J. H. (2016). Greedy Function Approximation: A Gradient Boosting Machine. 1999. DOI= http://wwwstat. stanford. edu/~ jhf/ftp/trebst. pdf.

(Gazalba and Reza, 2017) Gazalba, I., & Reza, N. G. I. (2017, November). Comparative analysis of k-nearest neighbor and modified k-nearest neighbor algorithm for data classification. In 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE) (pp. 294-298). IEEE.

(Georganos et al, 2019) Georganos, S., Grippa, T., Gadiaga, A., Vanhuysse, S., Kalogirou, S., Lennert, M., & Linard, C. (2019, May). An Application of Geographical Random Forests for Population Estimation in Dakar, Senegal using Very-High-Resolution Satellite Imagery. In 2019 Joint Urban Remote Sensing Event (JURSE) (pp. 1-4). IEEE.

(Gers et al, 1999) Gers, F. A., Schmidhuber, J., & Cummins, F. Learning to forget: Continual prediction with LSTM. 1999.

(Golpayegani, 2018) Golpayegani, F. (2018, November). Co-ride: Collaborative preference-based taxi-sharing and taxi-dispatch. In 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 864-871). IEEE.

(Huang, 2019) Huang, S. (2019, November). 5G-based intelligent transportation system construction. In 2019 2nd International Conference on Safety Produce Informatization (IICSPI) (pp. 468-472). IEEE.

(Huang and Pend, 2018) Huang, X., & Peng, H. (2018, November). Efficient mobility-on-demand system with ride-sharing. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 3633-3638). IEEE.

(James, 2018) James, J. Q. (2018). Two-stage request scheduling for autonomous vehicle logistic system. IEEE Transactions on Intelligent Transportation Systems, 20(5), 1917-1929.

(Jung, 2018) Haebichan Jung. 2018. Adaboost for Dummies: Breaking Down the Math (and its Equations) into Simple Terms. 2018. https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

(Kaplunovich and Yesha, 2018) Kaplunovich, A., & Yesha, Y. (2018, December). Consolidating billions of Taxi rides with AWS EMR and Spark in the Cloud: Tuning, Analytics and Best Practices. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 4501-4507). IEEE.

(Kljaic et al. 2016) Kljaic, Zdenko et al. “The challenge of cellular cooperative ITS services based on 5G communications technology.” 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (2016): 587- 594

(Krishni, 2018) Krishni “A Beginners Guide to Random forest Regression” 2018 https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb

(Kirill Eremenko at Udemy) Kirill Eremenko, Deep Learning A-Z: Hands-On Artificial Neural Networks https://www.udemy.com/course/deeplearning/learn/lecture/6743222#overview

(Kumar and Goel, 2018) Kumar, S., & Goel, E. (2018, April). Changing the world of Autonomous Vehicles using Cloud and Big Data. In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 368-376). IEEE.

(Liu et al, 2018) Liu, Y., Qin, Y., Guo, J., Cai, C., Wang, Y., & Jia, L. (2018, December). Short-term forecasting of rail transit passenger flow based on long short-term memory neural network. In 2018 International Conference on Intelligent Rail Transportation (ICIRT) (pp. 1-5). IEEE.

(LPEP Trip Records) Data Dictionary - LPEP Trip Records May 1, 2018. https://www1.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_green.pdf

(Luo et al, 2018) Luo, Y., Jia, X., Fu, S., & Xu, M. (2018). pRide: Privacy-preserving ride matching over road networks for online ride-hailing service. IEEE Transactions on Information Forensics and Security, 14(7), 1791-1802.

(Min et al, 2019) Min, K., Kim, D., Park, J., & Huh, K. (2019). RNN-Based Path Prediction of Obstacle Vehicles With Deep Ensemble. IEEE Transactions on Vehicular Technology, 68(10), 10252-10256.

(New York Taxi, 2020) New York City Taxi & Limousine Commission, TLC Trip Record Data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

(Nwulu, 2017) Nwulu, N. I. (2017, September). A decision trees approach to oil price prediction. In 2017 International Artificial Intelligence and Data Processing Symposium (IDAP) (pp. 1-5). IEEE.

(Ota et al, 2015) Masayo Ota, Huy Vo, Claudio Silva, Juliana Freire. A Scalable Approach for Data-Driven Taxi Ride-Sharing Simulation. 2015 IEEE International Conference on Big Data (Big Data).

(Ota et al. 2016) Ota, M., Vo, H., Silva, C., & Freire, J. (2016). Stars: Simulating taxi ride-sharing at scale. IEEE Transactions on Big Data, 3(3), 349-361.

(Pandit et al, 2019) Pandit, V. N., Mandar, D., Hanawal, M. K., & Moharir, S. (2019, January). Pricing in Ride Sharing Platforms: Static vs Dynamic Strategies. In 2019 11th International Conference on Communication Systems & Networks (COMSNETS) (pp. 208-215). IEEE.

(Pedro and Ferreira, 2014) Pedro M. d'Orey, Michel Ferreira. Can ride-sharing become attractive? A case study of taxi-sharing employing a simulation modelling approach. IET Intelligent Transport Systems 7th June 2014.

(Petersen et al, 2019) Petersen, N. C., Rodrigues, F., & Pereira, F. C. (2019). Multi-output bus travel time prediction with convolutional LSTM neural network. Expert Systems with Applications, 120, 426-435.

(Poulsen et al, 2016) Poulsen, L. K., Dekkers, D., Wagenaar, N., Snijders, W., Lewinsky, B., Mukkamala, R. R., & Vatrapu, R. (2016, June). Green cabs vs. uber in new york city. In 2016 IEEE International Congress on Big Data (BigData Congress) (pp. 222-229). IEEE.

(Rocca, 2019) Joseph Rocca. "Ensemble methods: bagging, boosting and stacking, Understanding the key concepts of ensemble learning" 2019. https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

(Salazar-Cabrera et al, 2019) Salazar-Cabrera, R., De La Cruz, Á. P., & Molina, J. M. M. (2019, March). Fleet management and control system from intelligent transportation systems perspective.

In 2019 2nd Latin American Conference on Intelligent Transportation Systems (ITS LATAM) (pp. 1-7). IEEE.

(Schmidt and Finan, 2018) Schmidt, A. F., & Finan, C. (2018). Linear regression and the normality assumption. Journal of clinical epidemiology, 98, 146-151.

(Scikit-learn) Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., "Scikit-learn: Machine Learning in Python", in the Journal of Machine Learning Research, volume 12, pages=2825-2830, 2011.

(Seungwoo et al, 2014) Seungwoo Jeon, Bonghee Hong, and Byungsoo Kim. Big Data Processing for Prediction of Traffic Time based on Vertical Data Arrangement. 2014 IEEE 6th International Conference on Cloud Computing Technology and Science.

(Shen et al, 2019) Shen, B., Cao, B., Zhao, Y., Zuo, H., Zheng, W., & Huang, Y. (2019, February). Roo: Route Planning Algorithm for Ride-sharing Systems on Large-Scale Road Networks. In 2019 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 1-8). IEEE.

(Sherif et al. 2016) Sherif, A. B., Rabieh, K., Mahmoud, M. M., & Liang, X. (2016). Privacy-preserving ride-sharing scheme for autonomous vehicles in big data era. IEEE Internet of Things Journal, 4(2), 611-618.

(Shi et al, 2016) Shi, M. K., Jiang, H., & Li, S. H. (2016, November). An intelligent traffic-flow-based real-time vehicles scheduling algorithm at intersection. In 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV) (pp. 1-5). IEEE.

(Silwal et al, 2019) Silwal, S., Gani, M. O., & Raychoudhury, V. (2019, June). A Survey of Taxi Ride-sharing System Architectures. In 2019 IEEE International Conference on Smart Computing (SMARTCOMP) (pp. 144-149). IEEE.

(Singh et al, 2019) Singh, A., Alabbasi, A., & Aggarwal, V. (2019). A distributed model-free algorithm for multi-hop ride-sharing using deep reinforcement learning. arXiv preprint arXiv:1910.14002.

(Smolyakov, 2017) Vadim Smolyakov. 2017. Ensemble Learning to Improve Machine Learning Results. 2017. https://blog.statsbot.co/ensemble-learning-d1dcd548e936

(Statistic Canada, 2011) Statistic Canada "Commuting to work, National Household Survey" https://www12.statcan.gc.ca/nhs-enm/2011/as-sa/99-012-x/99-012-x2011003_1-eng.cfm

(Teung et al, 2019) Yeung, S., Aziz, H. A., & Madria, S. (2019, June). Activity-Based Shared Mobility Model for Smart Transportation. In 2019 20th IEEE International Conference on Mobile Data Management (MDM) (pp. 599-604). IEEE.

(Thangaraj et al, 2017) Thangaraj, R. S., Mukherjee, K., Raravi, G., Metrewar, A., Annamaneni, N., & Chattopadhyay, K. (2017, April). Xhare-a-ride: A search optimized dynamic ride-sharing system with approximation guarantee. In 2017 IEEE 33rd International Conference on Data Engineering (ICDE) (pp. 1117-1128). IEEE.

(Tsao et al, 2019) Tsao, M., Milojevic, D., Ruch, C., Salazar, M., Frazzoli, E., & Pavone, M. (2019, May). Model predictive control of ride-sharing autonomous mobility-on-demand systems. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 6665-6671). IEEE.

(Wang et al, 2017) Wang, B., Zhang, J., Zhang, Z., Pan, L., Xiang, Y., & Xia, D. (2017). Noise-resistant Statistical Traffic Classification. IEEE Transactions on Big Data.

(Wei et al, 2019) Wei, Q., Rodriguez, J. A., Pedarsani, R., & Coogan, S. (2019, July). Ride-sharing networks with mixed autonomy. In 2019 American Control Conference (ACC) (pp. 3303-3308). IEEE.

(Wickramarachchi, 2017) Anuradha Wickramarachchi Introduction to Neural Networks. 2017. https://towardsdatascience.com/introduction-to-neural-networks-ead8ec1dc4dd

(Xiong et al, 2019) Xiong, Y., Su, G., Ye, S., Sun, Y., & Sun, Y. (2019, July). Deeper capsule network for complex data. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

(Xu et al, 2017) Xu, W., Zhou, H., Cheng, N., Lyu, F., Shi, W., Chen, J., & Shen, X. (2017). Internet of vehicles in big data era. IEEE/CAA Journal of Automatica Sinica, 5(1), 19-35.

(Zhong et al, 2018) Zhong, Y., Gao, L., Wang, T., Gong, S., Zou, B., & Yu, D. (2018, October). Achieving Stable and Optimal Passenger-Driver Matching in Ride-Sharing System. In 2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS) (pp. 125-133). IEEE.

(Zhu et al, 2018) Zhu, L., Yu, F. R., Wang, Y., Ning, B., & Tang, T. (2018). Big data analytics in intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems, 20(1), 383-398.