An Adaptive and Diversity-Based Ensemble Method for Binary Classification

by

Xing Fan

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

Master of Applied Science

in

Electrical and Computer Engineering (Data Science)

Carleton University
Ottawa, Ontario

© 2016, Xing Fan

Abstract

In recent years, machine learning techniques have been rapidly developed and widely applied to many industrial and academic fields. Moreover, as an important part of the machine learning techniques, ensemble techniques have drawn significant attention in both academic researches and practical applications, which can make use of multiple single models to construct a hybrid model. Usually, compared to each individual model, a better performance can be achieved by applying ensemble methods. In this thesis, a novel ensemble method is proposed to improve the performance for binary classification. The proposed method can non-linearly combine the base models by adaptively selecting the most suitable one for each data instance. The new approach has been validated on two datasets, and the experiments results show an up to 18.5% improvement on F_1 score compared to the best individual model. In addition, the proposed method outperforms two other commonly used ensemble methods (Averaging and Stacking) in improving F_1 score.

Acknowledgements

I would like to express my sincere thanks to my thesis supervisor, Professor Lung and Professor Ajila, for all their guidance and support throughout the completion of this thesis.

Thanks for the support of my families and friends, the accompaniment of A-meow and the assistance of Hermit.

Table of Contents

Abstract	i
Acknowledgements	ii
List of Figures	v
List of Tables.....	vii
Chapter 1: Introduction.....	1
1.1 Motivations and Objectives	1
1.2 Thesis Contributions	4
1.3 Thesis Organization	4
Chapter 2: Background and State of the Art.....	6
2.1 Background Review	6
2.1.1 Overview of Problem Types in Machine Learning.....	6
2.1.2 Classification.....	7
2.1.3 Feature Engineering.....	10
2.1.4 Metrics for Binary Classifiers	12
2.1.5 Machine learning Algorithms for Classification.....	19
2.1.6 Tools and Environments	34
2.2 Related Work.....	36
Chapter 3: System Design and Implementation	39
3.1 Overview	39
3.2 System Design	43
3.2.1 System Architecture and Data Flow	44
3.2.2 Pre-processor.....	46
3.2.3 Base Model Selector	47
3.2.4 Relabeler	51

Chapter 4: Experiments and Evaluation.....	54
4.1 Overview of Datasets.....	54
4.1.1 Repeat Buyer Prediction (RBP) [IJCAI, 2015].....	54
4.1.2 Census Income Prediction (CIP).....	56
4.2 System Implementation	57
4.2.1 Pre-processing.....	57
4.2.2 Single Model Training	65
4.2.3 Ensemble Construction	65
4.3 Experiment Design	69
4.4 Results and Analysis.....	71
4.4.1 Analysis of Single Models	72
4.4.2 Analysis of Ensemble Model	83
4.5 Comparison of AUC and F_1 Score	88
Chapter 5: Conclusion	92
5.1 Accomplishments.....	92
5.2 Limitations	93
5.3 Future Research Directions.....	94
References	95

List of Figures

Figure 1.1 General Ensemble Learning Workflow	2
Figure 2.1 Workflow for Building a Machine Learning System	7
Figure 2.2 An Example for Binary Classifier Training and Predicting	9
Figure 2.3 Confusion Matrix for Binary Classification [Kohavi, 1998].....	14
Figure 2.4 ROC Space and Four Plots of Prediction Examples [ROC, 2016].....	16
Figure 2.5 Examples of ROC Curves in ROC Space [ROC, 2016].....	17
Figure 2.6 Main Categories of Machine Learning Algorithms for Classification [Brownlee, 2013]	21
Figure 2.7 The Logistic Function [LogisticFuction, 2016].....	24
Figure 2.8 The Trade-off between Bias and Variance [Fortmann-Roe, 2012]	26
Figure 2.9 The AdaBoost Algorithm: an Illustration [Schapire, 2012]	28
Figure 2.10 A 2D Example of Optimization of Loss Function Using Gradient Descent [Sebastian, 2016].....	30
Figure 2.11 The Process of Stacking	33
Figure 3.1 An Example of Base Model Selection According to Pairwise Diversity	40
Figure 3.2 An Example of Constructing Ensemble Model by Averaging	41
Figure 3.3 System Architecture and Data Flow	44
Figure 4.1 Working Principle of the Constructed Ensemble Model.....	68
Figure 4.2 Result of AdaBoost Performance in RBP	73
Figure 4.3 Result of AdaBoost Performance in CIP	74
Figure 4.4 Result of Logistic Regression Performance in RBP.....	74
Figure 4.5 Result of Logistic Regression Performance in CIP	75

Figure 4.6 Result of Xgboost Performance in RBP	75
Figure 4.7 Result of Xgboost Performance in CIP	76
Figure 4.8 Result of Random Forest Performance in RBP	76
Figure 4.9 Result of Random Forest Performance in CIP	77
Figure 4.10 Result of AdaBoost Overhead in RBP	78
Figure 4.11 Result of AdaBoost Overhead in CIP	78
Figure 4.12 Result of Logistic Regression Overhead in RBP	79
Figure 4.13 Result of Logistic Regression Overhead in CIP	79
Figure 4.14 Result of Xgboost Overhead in RBP	79
Figure 4.15 Result of Xgboost Overhead in CIP	80
Figure 4.16 Result of Random Forest Overhead in RBP	80
Figure 4.17 Result of Random Forest Overhead in CIP	80
Figure 4.18 Comparison of single model's best performance on RBP.....	82
Figure 4.19 Comparison of single model's best performance on CIP	82
Figure 4.20 AUC result comparison between different ensemble methods for RBP	84
Figure 4.21 F_1 score result comparison between different ensemble method for RBP	85
Figure 4.22 AUC result comparison between different ensemble method for CIP	86
Figure 4.23 F_1 score result comparison between different ensemble method for CIP	86

List of Tables

Table 3.1 Example of Normalizing Probabilities Based on Ranks.....	48
Table 3.2 Example of Calculating Two Types of Pairwise Diversity	50
Table 3.3 Example of Relabeling.....	52
Table 4.1 The Definitions of Data Fields in User Behavior Logs in RBP.....	55
Table 4.2 The Definitions of Data Fields in User Profile Information in RBP	56
Table 4.3 The Definitions of Data Fields in Training and Testing Data in RBP.....	56
Table 4.4 The Definitions of Data Fields in CIP	57
Table 4.5 An Example of the Merged Data in RBP	58
Table 4.6 Statistics of ID Fields in RBP	59
Table 4.7 Statistics of Action Types in RBP	59
Table 4.8 Statistics of Age Ranges in RBP.....	59
Table 4.9 Statistics of Gender in RBP	59
Table 4.10 Statistics of Labels in RBP	59
Table 4.11 Examples of CIP raw data [UCI, 1996].....	61
Table 4.11 Examples of CIP raw data (cont'd).....	61
Table 4.12 Statistic of Labels in CIP	61
Table 4.13 The Definitions of Extracted Features in RBP	63
Table 4.14 Probabilities and Ranks for Testing Instances	66
Table 4.15 Relabeled Training Instances.....	67
Table 4.16 Logistic Regression for RBP	69
Table 4.17 Adaboost for RBP.....	70
Table 4.18 Xgboost for RBP.....	70

Table 4.19 Random Forests for RBP	70
Table 4.20 Logistic Regression for CIP	70
Table 4.21 Adaboost for CIP	70
Table 4.22 Xgboost for CIP	70
Table 4.23 Random Forests for CIP	71
Table 4.24 Diversity between single models in RBP	83
Table 4.25 Diversity between single models in CIP	83
Table 4.26 Ensemble Overhead in RBP	88
Table 4.27 Ensemble Overhead in CIP	88
Table 4.28 Probabilities from Two Classifiers and Labels for 20 Example Instances	89
Table 4.29 Ranked Instances with Real Labels and Predicted Labels.....	89
Table 4.30 Metrics for the Two Classifiers	90

Chapter 1: Introduction

This chapter will firstly go through the motivation and objectives of the proposed research question, followed by an overall introduction of this thesis' contribution. Finally, the thesis organization is described.

1.1 Motivations and Objectives

In supervised machine learning, ensemble learning is the technique that aims to make use of multiple single models to construct a hybrid model to achieve a better performance than that of each single integrated model [Opitz, 1999]. Figure 1.1 indicates a general workflow for solving classification problems by applying ensemble methods. Firstly, raw data usually needs to be pre-processed for initializing a training dataset, during which processes such as feature extraction and normalization could be involved. Secondly, training sets for each individual single model are derived from the initialized dataset. Thirdly, single models are trained from different training datasets or by different algorithms. Parameter tuning and validation for each individual model are included in this phase as well. Finally, all the single models are combined to construct the ensemble, then the final model could be further validated and tested.

There exists a number of ensemble learning techniques proven to be successful in practice. Based on our understanding, most of the techniques could be generally categorized into two types according to their different concerns:

Type I. Techniques that focus on deriving new training sets from the initial training set to train diverse single models, which is the phase marked as red in Figure 1.1 [Opitz, 1999];

Type II. Techniques that focus on finding ways to blend the individual models, which is the phase marked as blue in Figure 1.1 [Canuto, 2005]. In this thesis we will mainly focus on the discussion of Type II techniques for improving predictive performance in binary classification problems.

Figure 1.1 General Ensemble Learning Workflow

For a particular dataset, due to the various principles and implementations of the applied machine learning algorithms, single models trained by different algorithms may have diverse abilities in recognizing the pattern of a given training sample. Therefore, in order to achieve the complementation across individual models by applying ensemble

methods, it is important to select those highly-diverse single models, and to blend them properly with an efficient utilization of the diversity [Kuncheva, 2005a].

Most of popular *Type II* ensemble techniques assign weights to all trained single models and then linearly combine them [Canuto, 2005]. Thus, for all unknown instances, the contribution of each individual model to the final prediction is fixed, which may result in a limitation of performance enhancement. Therefore, dynamically adjustments of each single model's contribution for different instances could be helpful, which requires methods of non-linear blending [Canuto, 2005].

In other words, as the existing *Type II* ensemble methods takes all the outputs of the combined single models into account, the weaknesses of the single models are also kept in the hybrid model. If an ensemble method can adaptively select the most suitable single model to predict for each instance, then the weaknesses of the single models are avoided. In addition, it is not necessary to combine all the single models. In order to maximize the complementation effect, only those single models with highest pairwise diversities should be selected to construct the ensemble model.

Based on all the motivations described above, the objectives of this thesis is to find an ensemble method that can 1) determine base models for combination according to their pairwise diversities; 2) recognize the suitable base model for each data instance; 3) each unknown instance is predicted only by the identified suitable base model. As a result, the overall performance of the predictions by the ensemble model can be improved by non-linearly combining the base models. In order to perform the adaptive selection of base models for each unknown instance, a meta-level model should be trained from the relabeled training set, which can indicate the suitable base model of each training sample.

1.2 Thesis Contributions

The process of achieving the goal described above has led to the following contributions:

- 1) A novel ensemble method for solving binary classification problems is proposed. the core idea of the proposed ensemble method is to train a meta-level model that can dynamically choose a more suitable base classifier for each unknown instance;
- 2) The training data is relabeled by a new method, and then the relabeled training data is used to train the meta-level model;
- 3) The pairwise diversities of single classifiers are measured by two methods, according to which the base classifiers for combination are selected, and all the applied methods of measurement are compared;
- 4) Multiple machine learning algorithms are applied to train a group of different base models, and a comprehensive framework including all the methods mentioned above are designed and implemented;
- 5) The proposed ensemble method is validated on two datasets (Repeat Buyer Prediction [IJCAI, 2015] and Census Income Prediction [UCI, 1996]), and the experiments results indicate that an up to 18.5% improvement on performance is achieved compared to those of two other *Type II* ensemble methods (Averaging and Stacking).

1.3 Thesis Organization

This thesis consists of 5 chapters including the introductory chapter. Chapter 2 introduced required background information, which includes a basic description of classification problems in data science, the general workflow for problem solving, and the knowledge of several common types of supervised machine learning algorithms. Then a

number of popular ensemble methods, as well as the concept of feature engineering are introduced. It continues by covering the tools utilized in this research, plus a discussion on related works.

Chapter 3 describes the design and implementation of the framework that applies the proposed ensemble method. The contents of this chapter are organized as the workflow of this framework. Firstly, data pre-processing is introduced including feature extraction and normalization. Secondly, single classifiers trained by different algorithms are discussed and compared respectively. At last, a detailed description of the proposed ensemble method is presented.

The experiments design and results are presented in Chapter 4, which also includes the evaluations of performances and the associated analysis. The comparison of the proposed ensemble method and two others are discussed as well.

Chapter 5 is a conclusion that summarizes the main accomplishments of this thesis along with the limitations of the proposed ensemble method and the potential directions for future works.

Chapter 2: Background and State of the Art

This chapter introduces the background for this proposed ensemble method, which includes the problem types in machine learning, general process of building machine learning system, basic concepts of feature engineering, metrics for evaluating classifiers and the review of multiple machine learning algorithms. In addition, several important tools are introduced as well as a brief description of the experiments environments. Finally, related works, especially the approaches regarding to ensemble techniques are reviewed.

2.1 Background Review

2.1.1 Overview of Problem Types in Machine Learning

Supervised learning and unsupervised learning are two typical types of machine learning. Other types, such as semi-supervised learning and reinforcement learning, are usually evolved from those two types [Barber, 2012]. For supervised learning, its main target is to train a model from the training data with labels so that the model could make predictions for unknown data. Moreover, unsupervised learning techniques are applied to allow us to reveal and explain the unknown patterns of the unlabeled data. The main difference between these two types of learning technique is that in supervised learning, the correct output of the given training dataset is already known, having the idea that there is a relationship between the input and the output. Unsupervised learning, on the other hand, focuses on approaching problems with little or no idea what the results should be.

In supervised learning, most problems could fall into two areas, classification and regression. Most parts of these two types of tasks are similar, except for the outputs of the trained models. The outputs of classification tasks are discrete, which are the categorical

class labels, whereas those of regression are continuous. In this thesis, the proposed ensemble method is for solving classification problems.

2.1.2 Classification

2.1.2.1 Workflow

Figure 2.1 shows the process to build a machine learning system for solving classification problems, during which training and predicting are two fundamental stages. Training is the process of using existing data to train a model, followed by predicting, which is the application of trained model for unknown data. This research focuses on the training stage, in which three phases are involved: pre-processing, learning and evaluation.

Figure 2.1 Workflow for Building a Machine Learning System

Raw data rarely comes in a required form or scale. In some cases, the way raw data organized is not capable for presenting any meaningful information. Therefore, pre-processing, which includes sampling, dimensionality reduction, transformation, normalization and feature engineering, is usually a necessary and important process for the subsequent training phase. At the end of pre-processing, raw data is cut into the shape that

fits the learning algorithm and the processed data is then split into two part: train dataset and test dataset.

In the learning phase, the derived train data and associated labels are fed to multiple machine learning algorithms for model training, during which techniques such as feature selection, parameter tuning and cross-validation might be utilized to improve the model.

During the final evaluation phase, one or more suitable metrics for specific problems are selected for evaluating models, the trained model is then applied to make predictions for test data. Afterwards selected metrics are calculated based on the predictions and the associated labels of test data. Finally, the model with best performance is chosen as the final model for predicting stage.

2.1.2.2 Types of Classification

Classification problems can be categorized into two types according to the number of classes to be identified, which are binary classification and multiclass classification [Har-Peled, 2002]. The former identifies instances as one of the two pre-defined classes, whereas the latter classifies instances into one of the more than two classes. Numerous classification methods and related techniques have been tailored to binary classification, which also lays a foundation for solving multiclass classification problems. In this thesis for simplicity we mainly focus on binary classification.

2.1.2.3 Binary Classifier Training and Predicting

Figure 2.2 An Example for Binary Classifier Training and Predicting

Figure 2.2 describes the training and predicting detail of single binary classifier. First of all, a dataset consists of a group of instances (the rows in the table) that are presented as feature vectors plus corresponding labels. A feature vector is a set of individual and measurable attributes that describe the instance. The attributes are also commonly known as features, predictors, and explanatory or independent variables, especially in statistics. In addition, the values of features could be of various types such as nominal (e.g. male and female) and ordinal values (e.g. large, medium and small), both of which could be considered as categorical values. The slight difference is that ordinal values could be sorted whereas no order is implied in nominal values. The features could be also integer-valued (e.g. the number of users who purchased a certain item) or real-valued (e.g. height and

weight). Furthermore, each instance in the given dataset is labeled as one of actual classes or targets. The known dataset could be split into train data and test data according to a certain proportion.

Essentially, a classifier is a derived function that maps instances to targets, of which all parameters are determined. A machine learning algorithm is a process that estimates the parameters of the function by learning the train data, so that the classifier could fit the train data. In other words, a classifier is the output of an algorithm that is executed on the train data including the actual classes. The feature vector of each instance in training data can be plugged into the classifier and then a group of predicted class labels are obtained. All the parameters of the classifier are settled by the algorithm-specified method so that the sum of errors in the predictions, compared to the actual classes, is minimized, therefore this is the meaning of “fit”. In general, each algorithm has its individual formulation of model, which may require diverse methods to learn its parameters.

During the process of testing, all feature vectors in test data are plugged into the trained classifier to obtain the predictions and then one or multiple metrics are selected to evaluate the performance of the classifier on this given dataset, based on the predictions and the corresponding actual class labels. When utilizing the classifier, the unknown instances come without labels, but with the same form of the feature vectors as those in the train and test data, thus the outcomes of the classifier are the predictions for the unknown instances.

2.1.3 Feature Engineering

Feature engineering is generally the process of manually designing a group of new features that could better match the machine learning algorithms and describe the specific

problems, and is not a formally defined term [Bouchard-Côté, 2009]. As mentioned previously, the quality of the preliminarily collected data is usually not good enough to be learned directly. Specifically, not only the normalization of the data is required, but sometimes extra features are also needed to achieve a better performance, since the original features in raw data might be either redundant on describing the properties of instances, or short of meaningful expression.

In order to reduce the relevance between the initial features, and to extract even more informative features on the basis of the initial features, it is significant to understand the design principle of more proper features, which usually requires domain knowledge of the certain problems. In many of these cases, feature engineering could even be considered as essential for solving such problems. The initial features are normally the most intuitive expression of the instances and easy to be collected, while the engineered features are usually obtained by combining two or more primary features or expressing those features from different perspective.

In spite of the performance improvement, there are other elements should be considered among feature engineering. For instance, extra workloads are always brought into the system and it could be very expensive in both time and space, especially when the size (number of features and instances) of data is relatively large. In addition, the number of new features could be much larger than that of the original ones, which requires more space to store the data and more memory for computation. These problems are worth being carefully considered in practice.

2.1.4 Metrics for Binary Classifiers

2.1.4.1 Types of Binary Classifiers

A binary classification model, or a binary classifier could be regarded as a function that maps instances to positive or negative class label [Alpaydin, 2010]. According to the types of outcomes, classifiers could be recognized as a discrete classifier or probabilistic classifier. A discrete classifier directly generates a predicted class label for an instance whereas a probabilistic classifier assigns each instance a score to indicate its degree of confidence of belonging to one class [Friedman, 2009]. Moreover, for the probabilistic classifiers, the assigned score could either be an absolute probability, or could be any uncalibrated real value that a higher value still represents a higher probability. In other words, probabilistic classifiers rank instances according to their probabilities of being a member of class in ascending or descending order. Afterwards, compared to the absolute predictions by discrete classifiers, a relative threshold is assigned to probabilistic classifiers to determine the predicted classes of instances. The instances with a score higher than the threshold is considered as positive, while those with a score lower than the threshold is classified as negative.

2.1.4.2 Confusion Matrix

Confusion Matrix is generally used for presenting and interpreting the performance of a classifier on a certain dataset. It is a $n \times n$ matrix (n represents the number of classes) constructed from the Cartesian product of actual classes and predicted classes [Kohavi, 1998]. Each entry of the matrix indicates the number of instances in each ordered pair of the Cartesian product. For binary classification, n equals to 2, therefore represents the two classes of Positive and Negative. As shown in Figure 2.2, the table in solid lines represents

the confusion matrix for binary classifiers. Additionally, 9 common rates calculated from related entries are presented in dash lines. Some important terms in Figure 2.3 are explained in detail as following [Kohavi, 1998]:

- Total: the total number of instances that have been classified by the classifier;
- True Positive (TP): the number of instances that have been correctly classified as Positive by the classifier;
- False Positive (FP): the number of instances that have been incorrectly classified as Positive by the classifier;
- True Negative (TN): the number of instances that have been correctly classified as Negative by the classifier;
- False Negative (FN): the number of instances that have been incorrectly classified as Negative by the classifier;
- True Positive Rate (TPR): also called Sensitivity or Recall, the proportion of real positive instances that are correctly identified;
- True Negative Rate (TNR): also called Specificity, the proportion of real negative instances that are correctly identified;
- False Positive Rate (FPR): also called Fall-out, the proportion of real negative instances that are incorrectly identified, can be calculated from $(1 - \text{Specificity})$ as well;
- False Negative Rate (FNR): also called Miss Rate, the proportion of real positive instances that are incorrectly identified;
- Positive Predictive Value (PPV): also called Precision, the proportion of predicted positive instances that are correctly identified;

- Accuracy: The proportion of all instances that are correctly identified.

Total		Predicted Classes			
		Positive	Negative		
Actual Classes	Positive	True Positive (TP)	False Negative (FN)	True Positive Rate (TPR), Sensitivity, Recall = $TP/(TP+FN)$	False Negative Rate (FNR), Miss Rate = $FN/(TP+FN)$
	Negative	False Positive (FP)	True Negative (TN)	False Positive Rate (FPR), Fall-out = $FP/(FP+TN)$	True Negative Rate (TNR), Specificity = $TN/(FP+TN)$
		Positive Predictive Value (PPV), Precision = $TP/(TP+FP)$	False Omission Rate (FOR) = $FN/(FN+TN)$	Accuracy = $(TP+TN)/(TP+TN+FP+FN)$	
		False Discovery Rate (FDR) = $FP/(TP+FP)$	Negative Predictive Value (NPV) = $TN/(FN+TN)$		

Figure 2.3 Confusion Matrix for Binary Classification [Kohavi, 1998]

Among these terminologies listed above, Precision (PPV) and Recall (TPR) are two universal metrics describing the performance of a classifier on a certain dataset. Further complex metrics can be also calculated on the basis of the confusion metrics, thus to evaluate a classifier from different aspects. However, it is worth mentioning that the accuracy may not be an adequate measure for performance because it is very sensitive to class distribution of the given dataset [Fawcett, 2006]. When it comes to an imbalance dataset, for example, if 99% of the instances are negative and only 1% is positive, a classifier could even achieve a high accuracy of 99% by simply identifying all instances as negative, which is meaningless in practice. In this thesis, Area Under the receiver operating characteristic Curve (AUC) and F_1 score are selected as the metrics to evaluate the

performance of models. A more detailed description about AUC and F_1 score will be introduced in the next few sub-sections.

2.1.4.3 Receiver Operating Characteristic (ROC) Space

Basically, a receiver operating characteristic, or ROC curve, is a plot that visualizes the performance of a binary classifier. It is drawn in a 1×1 square area called ROC space, of which x-axis and y-axis are defined as False Positive Rate (FPR) and True Positive Rate (TPR) respectively [Fawcett, 2006]. Therefore, the plots in ROC space could describe the trade-off between the benefit (TPR) and the cost (FPR) of a classifier. Figure 2.4 explains ROC space with several significant points (Receiver operating characteristic, 2016). The lower left point (0, 0) represents a classification that only generates negative outcomes, while the upper right point (1, 1) represents a classification with only positive outcomes. The (0, 1) point represents a perfect classification when all actual positives are correctly predicted without any error. The points on the diagonal line represents the classifications that randomly guess one class with a certain probability. The points above the diagonal line represents the classifications with better performance compared to random guess. In practice, the area under the diagonal line is always considered as empty because a performance better than random guess could be achieved by simply reversing the predicted classes whenever the classification is plotted under the diagonal line.

Figure 2.4 ROC Space and Four Plots of Prediction Examples [ROC, 2016]

2.1.4.4 ROC Curve

For each discrete classifier on a given dataset, there is only one corresponding confusion matrix, which are plotted as a single point in ROC space [Fawcett, 2006]. On the other hand, probabilistic classifiers could yield different confusion matrixes as threshold varies, so that multiple points are plotted in ROC space, which traces the ROC curves as shown in Figure 2.5 [ROC, 2016].

Figure 2.5 Examples of ROC Curves in ROC Space [ROC, 2016]

2.1.4.5 AUC

Simply to say, a threshold represents the strictness of a probabilistic classifier making a positive prediction. Normally a higher threshold could result in a lower FPR but at the same time sacrifice the TPR, because a strict classifier rarely generates errors in predicting a positive class whereas only a small proportion of actual positive instances could be recognized. Consequently, the points plotted from higher thresholds are relatively close to the (0, 0) point in ROC space. On the contrary, lower thresholds could lead to both increased FPR and TPR, from which the points plotted are close to the (1, 1) point in ROC space.

If more actual positive instances could be assigned relatively higher scores by the probabilistic classifier, then a higher TPR with a lower FPR could be achieved even when a relatively higher threshold is set, so that the points plotted from higher thresholds are close to the (0, 1) point in ROC space. The ROC curve, in this case, is pulled up to the perfect classification point, which leads to a larger area under the ROC curve (AUC). Therefore, a larger AUC indicates that a better performance is achieved by this classifier. In other words, AUC represents the probability that a classifier assigns a higher score to a randomly selected positive instance than a negative one [Fawcett, 2006]. In this thesis, the main reasons of selecting AUC as the evaluation metric are: 1) AUC is a popular metrics for evaluating model performance in binary classification; 2) It is insensitive to imbalanced class distribution; 3) The two datasets used in this thesis have imbalanced class problem to different extents.

2.1.4.6 F₁ Score

In analysis of binary classification, F₁ score [Cohen, 2003] is also a commonly used metric to evaluate the performance of a classifier. F₁ score is defined as the harmonic mean of precision and recall:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The best value of F₁ score is 1 and the worst is 0. F₁ score conveys the balance between precision and recall. For example, an F₁ score of 0.45 is calculated from a precision of 0.9 and a recall of 0.3, while another F₁ score of 0.5 is obtained with both precision and recall being 0.5. Therefore, for a certain classifier, a tradeoff between its precision and recall is beneficial for achieving a higher F₁ score. Moreover, the reasonable tradeoff can be accomplished by tuning the threshold, which is highly correlated with the

class distribution of the given dataset. For example, a threshold of 0.5 may be appropriate for the dataset with balanced class, while a higher threshold could be better if the proportion of positive instances is very small. According to the definitions, precision can denote the degree of confidence for a classifier to predict an instance as positive, while recall indicates the ability of a classifier to identify the positive instances. For a certain classifier, a higher threshold usually leads to a higher precision but a lower recall, while a lower threshold results in a lower precision but a higher recall. In other words, both aggressive (a lower threshold) and conservative (a higher threshold) attitude in predicting positive harm the F_1 score.

In this thesis, the F_1 score is also selected as the performance evaluation metric, as the F_1 score is considered more suitable than AUC when suffering from imbalanced class problem in [Davis, 2006]. In addition, AUC is a “curve” metric to indicate the general or average performance of a classifier, while the F_1 score is a “point” metric that could be more meaningful in practice in some cases.

2.1.5 Machine learning Algorithms for Classification

2.1.5.1 Overview

In general, most machine learning algorithms, especially for solving classification problems, can be categorized as Figure 2.6 below [Brownlee, 2013]. On the basis of Brownlee’s method of categorization, those algorithms that can be utilized for classification are particularly selected to introduce. Plus, a category of Factorization Modeling that is widely applied in recommender systems is added, which was mentioned in [Brownlee, 2013] but not presented in the figure.

All these listed algorithms have their own characteristics that fit different specific classification problems in diverse fields. For example, logistic regression is commonly used for prototype validation for its efficiency in achieving an acceptable performance [Schutt, 2013]. In addition, Factorization Machine (FM) is more suitable for recommender systems, of which the collected data for training is usually under high sparsity, and FM has its unique character of learning this type of data in linear time [Rendle, 2010]. Each potential algorithm for a certain practical problem should be carefully considered from both aspects of design and implementation, because in many situations, the design concepts of a particular algorithm may satisfy the requirements, but the implementation of the algorithm may not.

In this thesis, these factors are taken into account when selecting algorithms for the given datasets: 1) The algorithm should have the ability of learning a probabilistic classifier, as AUC and F_1 score are selected as the metrics for evaluation. 2) The algorithm should be capable to learn sparse data since the given dataset is under high sparsity. 3) Since the main concern of this thesis is the method of ensemble, it is better for the selected algorithms to be simple so that the parameter tuning for models could be easier. This is the main reason why those complex algorithms such as deep learning are not selected. 4) The algorithm should achieve a proper balance between the performance of the single model and its training time consumption as well as its resource requirement.

Those algorithms that contributes to the proposed approach is marked as red, and both the designs and implementations of them will be discussed in detail during next several paragraphs. For more explicitly explaining the underlying mathematics, bold $\mathbf{x}$ is used to denote feature vectors, regular x_i with subscript i to refer to the i th feature of an

instance, bold $\mathbf{w}^T$ with superscript T to represent the transpose of the parameter vector, regular w_i with subscript i to stand for the weight of x_i , y for actual classes of the instances, $\hat{y}$ for predicted classes, P for the probability of an instance to be positive, and L for the loss function of a model.

Figure 2.6 Main Categories of Machine Learning Algorithms for Classification [Brownlee, 2013]

In this thesis, despite the algorithms that are introduced in detail in the following sections, other algorithms such as Support Vector Machines (SVM) [Cortes, 1995] and Factorization Machines are also considered when selecting algorithms for each given dataset. SVM is not selected as the single model mainly because: 1) it is not a probabilistic classifier; 2) the selected datasets in this thesis are highly sparse, and it is very time consuming for SVM to learn from sparse data, especially when the size of datasets is large. On the other hand, for the given datasets in this thesis, too much feature engineering is harmful to the performance of Factorization Machines, thus, the features for training the FM model are different from those for other single models. As the proposed method in this

thesis cannot properly deal with the single models that are trained from different features, FM is not selected as the single model, either.

2.1.5.2 Logistic Regression

Logistic regression can learn a model which is a function mapping an instance to a predicted class. The function is a linear combination of every feature with an estimated weight that indicates the contribution of this feature to the target variable. The output of the function is a real value between 0 and 1 that indicates the probability of this instance being one of the binary classes. The instance could then be classified as positive if its predicted probability is larger than the pre-determined threshold [Barber, 2012].

Logistic regression can be regarded as a generalization of linear regression, which is called Generalized Linear Model (GLM) in statistics. The GLM is capable for accepting the dependent variables that have other distributions rather than normal distributions for ordinary linear regression, such as Bernoulli distribution for logistic regression. It is accomplished by mapping the linear regression model to the response variable through a link function and a quantizer.

For logistic regression, the standard logistic function is applied as the link function, which is defined as following [Schutt, 2013]:

$$P(t) = \frac{1}{1 + e^{-t}}$$

where t represents the ordinary linear regression model that can be formulated as following:

$$t = \mathbf{w}^T \mathbf{x} + \beta$$

where β denote an unobservable bias from the actual value. Therefore, the final logistic regression model can be formulated as following:

$$F(\mathbf{x}) = P(y_i = 1) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + \beta)}}$$

by which $F(\mathbf{x})$ is interpreted as the probability of the i th instance being positive. Furthermore, the logistic function is also known as the inverse-logit function, which is often referred to a sigmoid function because of its “S” shaped curve as shown in Figure 2.7. A logistic function (inverse-logit function) can transform any real values into the range between 0 and 1, while a logit function does the reverse. Base on the information above, the following equation is obtained:

$$t = \text{logit}(P) = \ln\left(\frac{P}{1-P}\right) = \ln(P) - \ln(1-P) = \mathbf{w}^T \mathbf{x} + \beta$$

In other words, logistic regression can find a hyperplane $\mathbf{w}^T \mathbf{x} + \beta = 0$ as the decision boundary that slices the train data into two parts: the points above the hyperplane denote the instances with higher probabilities of being positive than that of being negative, and the points under the hyperplane are considered as the instances more likely to be negative. Additionally, the points on the hyperplane represents the instances with equal probabilities of being positive and negative, noted that the hyperplane is precisely the quantizer mentioned before. In summary, logistic regression is a probabilistic classifier in terms of design. Because of its simplicity (linear model) and expressive interpretability, logistic regression is usually selected as the classifier with baseline performance. In this thesis, logistic regression is performed by applying an implementation provided by Scikit-learn [Scikit, 2016], which can take sparse data as input.

Figure 2.7 The Logistic Function [LogisticFuction, 2016]

2.1.5.3 Ensemble Learning Algorithms

2.1.5.3.1 Overview

In classification, based on the train data, a model learned by a certain algorithm generates a hypothesis that is used for predictions. In terms of train data, residual is defined as the deviation of approximation given by the hypothesis from the actual value, while the error with respect to test data and unknown data is defined as the gap between the predictions and the actual values, which consists of bias, variance and noise [Friedman, 1997]. The bias appears when the model underfits the training data, thus it indicates the amount of data that is not properly learnt. A lower bias leads to a less residual and then contributes less to the error. On the other hand, the variance results from the model overfitting the train data, therefore, the variance indicates the quantity of noise that is incidentally modeled. The variance is not reflected in the residual but in the error. If a model is too sensitive to the noise in train data, then it is usually not well generalized for test data and unknown data. The bias and variance are generated by the model and could

be changed as the model varies, while the noise is the natural part of the samples, so that it cannot be measured or eliminated.

In general, it is very hard to minimize bias and variance simultaneously, hence, there is a trade-off between these two sources of error. Typically, the bias decreases as the complexity of the model increases (e.g. more features), while the variance increases accordingly. However, the variance could decrease as the size of train data increases (more samples). As a result, the expected prediction error goes down first and then up as the complexity grows. The trade-off is graphically illustrated in Figure 2.8 [Fortmann-Roe, 2012]. In order to improve the performance of a classifier, ensemble methods are used to reduce either or both of the two sources of error by combining multiple simple, or even “weak” base classifiers. In this thesis, several popular ensemble learning algorithms are selected as the base classifiers in the proposed ensemble model, and the core concepts of each of them are described in the following sections.

Although ensemble learning can help to enhance the performance, in practice, the application of ensemble methods leads to the increased computation during both training and predicting phases, as well as the increased storage for more model parameters. Also, a complex ensemble model may even loss more comprehensibility and interpretability.

Figure 2.8 The Trade-off between Bias and Variance [Fortmann-Roe, 2012]

2.1.5.3.2 Bootstrap Aggregation (Bagging)

Bagging was firstly proposed by [Breiman, 1996]. The core concept of Bagging is to train multiple base classifiers from different new training sets that are randomly sampled from the original train data with replacement, then to make the final prediction for each instance by averaging the scores (for a probabilistic classifier) or majority voting (for a discrete classifier). With this sampling method, the new sampled sets, which are known as the bootstrap samples in statistics, could have the same size with that of the original set, but only a fraction of a new set are unique samples and the rest are duplicates. This method can improve performance because it artificially creates diversities among the base classifiers, which is very helpful when constructing an ensemble, and reduces variance by averaging, but does not help much with reducing bias. In practice, Bagging is more helpful regarding to unstable algorithms, which are relatively more sensitive to the noise

(overfitting) so that the variance of a base classifier is higher accordingly. Decision tree is usually selected as the base model for bagging, but theoretically, any type of algorithms could be applied to train the base classifiers.

In this thesis, Random Forests provided by Scikit-learn is applied, which is a popular implementation of Bagging and uses decision tree as the base model. In spite of the randomly sampled training sets, Random Forests also integrates the idea of randomly selecting features when splitting in each single tree in order to further increase the diversity.

2.1.5.3.3 Boosting

Boosting is a family of ensemble learning algorithms that combines a set of weak learners to construct a relatively strong one. A weak learner here is defined as a classifier that makes predictions with a slightly better performance than random guess. However, in practice, it is not necessary for every base model to be weak. Compared to Bagging method, Boosting algorithms intended to improve the performance mainly by reducing bias.

Adaptive Boosting (AdaBoost) and Gradient Boosting Machine (GBM) are two of the most popular Boosting algorithms proposed by [Freund, 1997] and by [Friedman, 2001] respectively. The shared core concepts of them are: 1) all base classifiers are incrementally trained on different training sets in a forward stage-wise manner; 2) In each stage, a new base model is trained to overcome the shortcomings of the last one, which is always achieved by highlighting the errors made by the last base model for guiding the new one to focus more on those misclassifications; 3) The final hypothesis generated by the ensemble model is derived from the sum of hypothesis of each base model. The key divergence between AdaBoost and GBM is that, AdaBoost highlights the errors by adjusting the weights of training samples: increase the weights of the misclassified ones

and decrease the weights of the correctly identified ones, as shown in Figure 2.9 [Schapire, 2012], while GBM highlights the errors by calculating the residuals of the existing base models, which is exactly the negative gradient descent of the loss function.

Figure 2.9 The AdaBoost Algorithm: an Illustration [Schapire, 2012]

In classification, each model trained from a certain algorithm has its own formula to present the calculation of the approximation to the probability or discrete class, which is defined as $\hat{y} = f(x)$. Therefore, a loss function (or cost function) $L(y, f(x))$ of the model is defined as a function that quantizes the cost for the deviation of predictions from the actual values, such as squared loss $L(y, f(x)) = \frac{1}{2}(y - f(x))^2$ [Friedman, 2009]. Given a

train data, if its loss function is convex and differentiable, then a optimization algorithm can be applied to find a set of the parameters for $f(x)$, so the loss function is minimized on that given data. The convexity of the loss function could guarantee that the found minimum is the global one. This process of optimization is to accomplish parameter estimation, and different selection of loss functions could lead to different estimations of the parameters.

As shown in Figure 2.10, gradient descent, also known as steepest descent, is a type of optimization algorithm that searches the global minimum of a function by taking steps to slide down to the minimum along the negative direction of gradient of the function (the steepest direction) at each current point. Therefore, for a differentiable loss function of n variables, the gradient at the current point is the vector of n partial derivatives of the loss function at the point. Furthermore, the steps are taken as proportional to the gradient, and in order to avoid missing the minimum point, a relatively smaller step is preferred as the algorithm approaches the minimum point. To achieve it, a parameter called learning rate is involved in, which is a real value between 0 and 1. Then the length of each step is determined by its previous one multiplied by the ratio representing the shrinkage of steps, in this way the speed of sliding down is controlled. Empirically, a lower learning rate can help to obtain an approximation closer to the minimum, which usually implies a better performance, but also leads to an increased number of iterations (steps). This trade-off is worth being carefully considered when tuning the model parameters for a higher efficiency of time and space utilization in practice.

Figure 2.10 A 2D Example of Optimization of Loss Function Using Gradient Descent [Sebastian, 2016]

Based on the descriptions above, in general, an ensemble model trained by a certain boosting algorithm can be formulated as follows [Friedman, 2001]:

$$F_T(x) = \sum_{t=1}^T \alpha_t h_t(x) + \alpha_0$$

where T denotes the number of base models, so that $F_T(x)$ represents the final ensemble model, and $h_t(x)$ represents the hypothesis generated by the t th base classifier, plus, α_t is the weight assigned to the t th base classifier and α_0 is a constant determined in the first stage for initializing the model. For gradient boosting, each additive base model is trained to fit the residual of the existing ensemble model, so that the final model becomes more accurate by taking the hypothesis generated by this newly added base classifier into

account. As the residual is precisely the negative gradient, in other words, the gradient boosting algorithm is actually a process to minimize the loss function of the final ensemble model without prior knowledge to the final loss function, which is achieved by only minimizing the loss function of the existing ensemble model whenever a new base model is added. Therefore, with the accumulation of the base models, the loss of the implied final function declines gradually. In other words, with a training set $\{(x_i, y_i)\}_{i=1}^n$, a convex and differentiable loss function $L(y, F(x))$ and the number of iterations (base classifiers) T , the underlying mathematics of this process can be defined as follows [Friedman, 2001]:

1. Initialize the model with a constant:

$$F_0 = \alpha_0 = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, \alpha)$$

2. For $t = 1$ to M :

- 1) Calculate the residuals as the negative gradient:

$$-\frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)} = y_i - F_{t-1}(x_i) = r_{it}$$

- 2) Fit a new base model $h_t(x)$ with the new training set $\{(x_i, r_{it})\}_{i=1}^n$:

$$h_t(x) = \underset{h}{\operatorname{argmin}} \sum_{i=1}^n L(r_{it}, h(x_i))$$

- 3) Determine the multiplier α_t of the newly added base model $h_t(x)$:

$$\alpha_t = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, F_{t-1}(x_i) + \alpha h_t(x_i))$$

- 4) Update the existing ensemble model:

$$F_m(x) = F_{m-1}(x) + \alpha_m h_m(x)$$

3. Output the final ensemble model as follows:

$$F_T(x) = \sum_{t=1}^T \alpha_t h_t(x) + \alpha_0$$

Additionally, in particular, AdaBoost can be also regarded as a case of the GBM with a differently defined loss function.

As explained above, the bias of the model is reduced by incrementally training the base models to learn the errors of their predecessor ensemble models. However, the concept of correcting the errors made by the predecessor model may be overfitting-prone, especially for the relatively noisy data. Accordingly, several regularization techniques are introduced into the model for constraining the learning procedure, so that the final ensemble model is prevented from overfitting the training set. Consequently, Boosting methods can reduce the bias of model with keeping variance under control.

In this thesis, an implementation of AdaBoost provided by Scikit-learn and an implementation of GBM called Extreme Gradient Boosting (Xgboost) is applied, which is optimized to be highly efficient, flexible, portable and scalable. Xgboost was initially released as an open source project by [Chen, 2016]. Both of the selected implementations of AdaBoost and GBM are probabilistic classifiers and able to deal with sparse data.

2.1.5.3.4 Stacked Generalization

Stacked Generalization, which is also known as Stacking, was firstly proposed by [Wolpert, 1992]. It is a high level ensemble method that introduces a meta-level model to combine a group of base models, which are usually of diverse types. Figure 2.11 demonstrates the process of Stacking, during which all base models are trained from the training set and tested on the testing set, then the meta model takes the outputs of all base models for testing data as new feature vectors and the corresponding actual classes values in testing data as targets, so that the meta model can learn a combination of all base models

to reduce the overall error. Theoretically, the meta-level model could be learnt by any type of supervised machine learning algorithms, but logistic regression is usually selected to be the meta-level learner because of its simplicity and efficiency, therefore, all the base models are linearly combined. In addition, non-linear algorithms can also be applied as the meta model, thus, Stacking can also be regarded as a more complex version of cross-validation, which can produce a more generalized ensemble model to improve the final performance. Overall, Stacking is a type of more generalized and flexible method of ensemble, which could be easily applied together with other ensemble methods to further enhance the total performance.

Figure 2.11 The Process of Stacking

Since both Stacked Generalization and this proposed method could be considered as *Type II* ensemble techniques, which is defined previously in Chapter 1, they are compared

in terms of combination method in this thesis. For Stacking, each of linear and non-linear models is selected as the meta model to combine a group of base models that are the same as those in this proposed method.

2.1.6 Tools and Environments

2.1.6.1 Python

The language used in this thesis for data operation and system implementation was Python, which is a high-level, general-purpose, interpreted and dynamic programming language. Python is well-known for its illustrious code readability and expressiveness. In addition, various programming paradigms are supported in python, including object-oriented and functional programming styles. Furthermore, Python interpreter is a free, cross-platform and open-source software, which features dynamic typing and automatic memory management. The kernel philosophy of Python, which is widely known as The Zen of Python [Python, 2016], plus its open-source feature, has attracted lots of outstanding developers forming a powerful community to make continuous contribution to the Python ecosystem. Therefore, lots of effective third-party tools have been developed and can be easily integrated into Python as extensions, which allows Python to be utilized under many circumstances such as web development and data science.

2.1.6.2 Anaconda

Anaconda is an open source distribution of Python, which packs numerous popular Python-based tools for data science covering data processing, data analysis, machine learning, data visualization, and scientific computing [Continuum, 2016]. Being an open data science platform with considerable number of users from diverse fields particularly in

education and research, Anaconda helps accelerate the development of data science ecosystem powered by Python.

2.1.6.3 Data Science-Related Python Libraries

Multiple popular data science-related Python libraries are utilized in this thesis. Firstly, for data pre-processing, Pandas, which is an open source library providing high performance and easy-to-use data structures for Python [Pandas, 2016], is used. Various interfaces in charge of interacting with other common data storage system, such as databases or file systems, are also supported. With Pandas, rapid accessing to data and intuitive data operation could be easily achieved. Secondly, Scikit-learn is applied during model training and evaluation phases, which is also an open source library providing comprehensive tools for machine learning, and already widely applied in various fields. In spite of diverse built-in implementations of various machine learning algorithms, it is also compatible with other standalone algorithms.

Pandas and Scikit-learn are both built on NumPy [NumPy, 2016], SciPy [SciPy, 2016] and Matplotlib [Matplotlib, 2016], which are three fundamental libraries for scientific computing with Python. NumPy provides a high performance, powerful, and multiple dimension array object, and many common functions that execute computations with those arrays. SciPy provides a large number of tools for scientific computing, which take the advantages of NumPy. Matplotlib is also a well-known Python library for visualizing data in 2D plots, which can yield high quality figures in various formats.

2.1.6.4 Experiments Environments

All the developments and experiments are conducted on a customized compute engine from Google Cloud Platform (Google cloud computing, hosting services & APIs,

2016), which is operated by Ubuntu 16.04 LTS with 6 cores (2.3GHz) virtualized from Haswell processors, 32 GB memory and 128 GB SSD.

2.2 Related Work

Ensemble methods such as Bagging, Boosting and Stacking which combine the decisions of multiple hypotheses are some of the strongest existing ensemble methods, therefore provide a reliable foundation for this proposed ensemble method. Among them, [Ham, 2005] applied Random Forests to classification of hyperspectral data on the basis of a binary hierarchical multi-classifier system. [Berstra, 2006] presented an algorithm for musical style and artist prediction from an audio waveform by utilizing AdaBoost for the selection and aggregation of audio features. Gradient Boosting Machine (GBM) was applied in [Atkinson, 2012] for incorporating diverse measurements of bone density and geometry so as to improve the accuracy of fracture prediction compared to standard measurement. Moreover, Stacked generalization is a universal method of applying a high level model to integrate lower level models for better predictive accuracy. It is usually achieved by utilizing different classifiers and combining the outputs. [Ghorbani, 2001] examined the generalization behavior by comparing single level learning models to multiple level learning models (stacked generalization method) on a multilayer neural network. Result shows that the stacked generalization scheme could improve classification performance and accuracy compared to the single level model. The method is also demonstrated through Wang's approach [Wang, 2006] in which the stacking approach performed successfully in predicting membrane protein types based on pseudo-amino acid composition.

Further ensemble methods are designed for the purpose of improving predictive accuracy and robustness by blending the predictions of multiple models in a more rational way. There are several schemes proposed for the customized selection of classifiers, thus to find the best base classifier for each individual instance on the basis of local accuracy. For example, [Tysmbal, 2008] proposed the method of Dynamic Integration, in which weighted voting were applied by giving higher weight to a classifier if its training data were in the same region as the testing example. [Giacinto, 2000] introduced an approach to group classifiers by their similarities and to retain one representative classifier per cluster. The core notion of this type of methods is to enhance the overall performance by choosing and/or assigning more weight to those classifiers that perform best in instances that are similar to the one that is being classified. On the other hand, static ensemble methodologies are gradually developing, [Bhatnagar, 2014] proposed a linear ensemble algorithm that takes into account both the accuracy of individual classifiers and the diversity among classifiers, which are also vital factors where importance should be attached when designing ensemble methods.

Moreover, since the dataset applied in this research was from the Repeat Buyers Prediction Competition [IJCAI-15, 2015]. Approaches based on this certain repeated buyer dataset were reviewed as references for this thesis as well. In this two-stage competition with different amount of data provided, the work [Liu, 2015] won the first place in Stage 1, with the progress in both phases of feature engineering and model training. Various techniques for extracting a large number of features were applied in feature engineering, while a blending algorithm was proposed to further boost the prediction performance in the training phase. The work [He, 2015] won the first place in Stage 2, in which a four-step

solution was introduced including 1) characteristics analysis and strategy design, 2) feature extraction and selection, 3) data training, 4) hybrid ensemble on both models and features. Both of the award winners suggested that feature engineering is the key element to this work. and the ensemble method applied in these two approaches perform better than any of the individual classifier. Furthermore, only linear ensemble methods were utilized in their works, so the results of these works can be considered as a reasonable quantitative comparison to this proposed non-linear method.

Chapter 3: System Design and Implementation

This chapter describes the system design and implementation of the proposed ensemble technique. The chapter starts with an overview of a problem of existing ensemble methods and a proposed solution. Section 3.2 discusses the design of the system, which is followed by system implementation as presented in Section 3.3. In addition, in the following sections, *positive (1)* and *negative (0)* are used to denote the two classes in binary classification. In binary classification, a classifier generates a value between 0 and 1 for each certain instance that represents the probability of this instance being classified as the *positive* class.

3.1 Overview

To solve classification problems, a classifier is learnt by a certain algorithm, so that a hypothesis can be proposed based on the classifier that best fits the given training set. For a certain dataset, a single classifier may have its own “blind area”, i.e., some points cannot be clearly identified, especially when the dataset is high-dimensional or non-linear. For example, in 2-dimensional space, as shown in Figure 3.1, some points (circled in red) cannot be correctly classified. Those points are points 1–5 in Figure 3.1 (a) based on the yellow classifier (as shown in a yellow line), and point 1 and points 3–6 using the blue classifier (as depicted with a blue line), and so on. A plus sign in Figure 3.1 indicates an instance belonging to the positive class, whereas a minus represents the negative class.

By applying the ensemble technique, multiple models can be integrated to reduce those blind areas as indicated in the lower three diagrams, e.g., Figure 3.1 (d), (e), and (f). Each data instance in the blind areas is then classified by the capable base model. For example, with the combination of H1 and H2 as depicted in Figure 3.1 (d), point 6 can be

identified by H1 generated by classifier 1, not circled in red, as shown in (a), whereas point 2 can be identified by H2 from classifier 2, not circled in red, as shown in (b).

Figure 3.1 An Example of Base Model Selection According to Pairwise Diversity

Furthermore, the ensemble models consisting of variously selected single classifiers have different recognized areas. As shown in (d), (e) and (f) of Figure 3.1, two more diverse single models can be integrated into an ensemble with a larger recognized area. In this example, the pairwise diversity is defined in terms of the number of points that are identified as different classes by two single classifiers. Two points (2 and 6) are distinctly classified by H1 and H2, respectively; hence, the pairwise diversity for H1 and H2 is 2. Similarly, six points (1, 3, 4, 7, 8 and 9) for H2 and H3, and eight points (1, 2, 3, 4, 6, 7, 8,

9) for H1 and H3, respectively, are differentiated. Therefore, H1 and H2 are relatively similar, while H3 is more different.

Accordingly, compared to the combination of H1 and H2 (with four unrecognized points), the combination of H2 and H3 has two less unrecognized points, and the combination of H1 and H3 has only one unrecognized point. As a result, classifiers 1 and 3 are considered the proper group for constructing the ensemble model. In summary, higher diversities among base models can improve the ability of the ensemble model by satisfying more types of data instances.

With the ensemble model consisting of classifiers 1 and 3, for each unknown data instance, if it is a positive one close to points 8 and 9 (or a negative one close to points 6 and 7), then classifier 1 should be used to make the prediction. The reason for the model selection policy is that those known instances (points 6, 7, 8, 9), which are homogeneous as this unknown instance, can be correctly classified by classifier 1. Similarly, if it is a positive one close to points 3 and 4 (or a negative one close to points 1 and 2), classifier 3 should be used.

Figure 3.2 An Example of Constructing Ensemble Model by Averaging

In addition, the problem in this example may not be properly solved with the existing ensemble techniques, such as averaging and stacking. As illustrated in Figure 3.2, the ensemble model is built by simply averaging all the hypotheses of the three classifiers, but there are still five unrecognized points for the final hypothesis.

To solving binary classification problems, in this thesis, a novel ensemble method is proposed and implemented. In the proposed ensemble method, a meta-level model is learnt, which recognizes the capable base classifier for each data instance. In addition to the meta model, from all trained single classifiers, only those with high pairwise diversities are selected as base models. Finally, the constructed ensemble model is able to support adaptive selection of the proper base model for each data instance.

To accomplish all the points mentioned above, two key problems demand prompt solutions:

- 1) The pairwise diversity needs to be defined for measurement.
- 2) A training set needs to be designed for learning the meta model.

As explained in Section 2.1.4.1, a probabilistic classifier performs the classification by ranking the instances according to their predicted probabilities of being positive. Therefore, for the first problem, the pairwise diversity is defined as the difference between the ranks of instances provided by each two classifiers.

For the second problem, in order to map each data instance to its appropriate base model, the meta model should be trained from the same feature vectors that have been used to train the single model. Additionally, each data instance in the training set needs to be relabeled to indicate its proper base model. A base model is considered proper for a data instance, if it can generate the most certain prediction. By ranking all the instances

according to their predicted probabilities, the certainty of a prediction is defined as the relative position of a certain instance in the ordered queue. For example, a prediction for a data instance is considered more certain if a positive instance is ranked at a lower position (or a negative instance at a higher position) in an ascent sorting. Then, this data instance should be relabeled as the base model which gives this prediction.

Overall, the pairwise diversity is measured on a group of instances for base model selection, while the certainty is on the basis of a single instance for relabeling the training set.

3.2 System Design

This section will firstly go through the overall system architecture and the data flow, which is followed by a detailed introduction of the design of three vital system modules: *Pre-processor*, *Base Model Selector*, and *Relabeler*.

3.2.1 System Architecture and Data Flow

Figure 3.3 System Architecture and Data Flow

Based on the concept described in Section 3.1, a system is designed to build the proposed ensemble model. The system consists of seven modules: *Pre-processor*, *Relabeler*, *Base Model Selector*, *Single Model Learner*, *Meta Model Learner*, *Meta Model* and *Base Models*, which are organized as shown in Figure 3.3. Each module is briefly described as follows.

- The *Pre-processor* is responsible for data transformation, data normalization and feature engineering, as well as splitting the known data into a training set and a testing set.

- The *Single Model Learner* learns single classifiers from the training set with the original labels for data instances.
- The *Base Model Selector* takes both the testing set and the outputs of the *Single Model Learner* as inputs, to determine the *Base Models* for constructing the ensemble.
- The *Relabeler* receives the training set with original labels for data instances, and then relabels the training set with the outputs of the *Base Model Selector*. The relabeling of each instance is performed according to the behaviors of *Base Models* on the original training set.
- The *Meta Model Learner* uses the relabeled training set to train the meta model.
- The *Meta Model* takes the feature vector of each unknown instance to predict its most suitable base model, and then passes the feature vector to the selected *Base Model* for the final prediction, which is the output of the system.

Differing from other modules in the system, the *Meta Model* and *Base Model* are not only considered as parts of the system, but also the outputs of other modules. Also, they are the ones forming the final ensemble model, which is then used for predicting. As the specific problem varies, each module of the system can be modified to adapt to the practical requirements, especially for the *Pre-processor*, which heavily depends on the problem-related domain knowledge.

In addition to the system architecture, the data flow as indicated in Figure 3.3 also demonstrates the workflow of the system. There are two types of input to the system: known raw data and unknown raw data, and both of them need to be pre-processed before further operations. The known raw data is a group of instances that has been already

collected, so that the size of the data is fixed. The unknown data are those without labels and will be used for prediction. Meanwhile, the unknown data usually enter the system randomly so that the data size is not fixed.

The next sub-sections discuss *Pre-processor*, *Base Model Selector* and *Relabeler* modules in more detail, since they have covered most of the design and implementation of the proposed method. Other modules are similar to the one described in Section 2.1.5.

3.2.2 Pre-processor

The task of the *Pre-processor* module mainly contains four parts: transformation, normalization, feature engineering and splitting. Firstly, transformation is a general process to restructure the raw features in another format. The purposes of transformations are highly diverse. In this thesis, the main focuses are data type conversion, text parsing and feature vectorization. For example, the log of users is initially in strings which cannot be directly computed. Therefore, the text has to be parsed into meaningful and computable real-valued features. In general, transformation is conducted to make the features capable of describing the instances more properly and informatively, as well as more compatible with the applied machine learning algorithms.

Secondly, normalization is usually known as the process of scaling the values of feature vectors into a unified range. The main goal is to standardize different variances of features. Normalization mostly contributes to the performance of those algorithms that are based on Euclidean metric space, such as k-means and k-nearest neighbors [Fukunaga, 1975]. But normalization has little benefit for decision tree-based algorithms [Friedl, 1997]. Although normalization is not always required by any algorithm, it seldom brings adversely effects to the overall performance. In addition to performance improvement, normalization

is also used for speeding up the learning process for the algorithms using gradient descent-based loss function solvers [Varma, 2009]. Accordingly, in this thesis, normalizations were applied for logistic regression to speed up the learning process.

Thirdly, for many practical problems, feature engineering or feature extraction is a vital process. Feature engineering is sometimes even more important than the algorithm selection, because the original features in the raw data are not adequate for characterizing the instances. Feature engineering often means the artificial creation of new features by the computation of two or more existing features, based on the background knowledge of the specific problem. In this thesis, feature engineering is the major workload of pre-processor, in which many new features were created through the computation on multiple original features in the raw data, so that the interactions between multiple features could be represented based on the understanding of this specific problem.

Finally, the splitting task is to divide the processed data into training set and testing set according to a certain ratio determined by *Pre-processor*.

3.2.3 Base Model Selector

The *Base Model Selector* is the module that determines the base models for constructing the final ensemble model. Initially, various single models learnt by different machine learning algorithms are tested on the testing set obtained from the *Pre-processor*. Each single model generates a group of probabilities for data instances in the testing set. Then the performance of each single model is evaluated. However, the main focus of *Base Model Selector* is the behaviors of the single models, rather than their performances. In other word, it is acceptable if the single models are relatively “weak”, but the final performance can be improved by combining these single models as long as the behaviors

of the single models are diverse. Therefore, the main goal of *Base Model Selector* is to measure the pairwise diversities between each two single models based on their predictions for the testing set.

The probabilities generated by single models usually cannot be used directly for calculating the pairwise diversities, as the probabilities generated by the single models have different distributions. For example, the probabilities generated by AdaBoost are all around 0.5 on the Repeat Buyer Prediction dataset, whereas the ones from Logistic Regression are up to 0.9. Therefore, the probabilities generated by different single models need to be normalized into identical scale based on their corresponding ranks.

Table 3.1 Example of Normalizing Probabilities Based on Ranks

Instance	lr_proba	lr_rank	lr_normalized	ab_proba	ab_rank	ab_normalized
0	0.245	2	0.4	0.501	3	0.6
1	0.781	4	0.8	0.534	4	0.8
2	0.344	3	0.6	0.496	2	0.4
3	0.106	1	0.2	0.558	5	1
4	0.983	5	1	0.472	1	0.2

Table 3.1 shows an example of the normalization. The probabilities generated by Logistic Regression (lr_proba) and AdaBoost (ab_proba) are not in unified scale, so the difference between the probabilities cannot represent the diversity between the behaviors of two classifiers. To normalize the probabilities into the same scale, firstly each testing instance is assigned two ranks (lr_rank, ab_rank) corresponding to its positions in the ascent order of each group of predicted probabilities. Secondly, both groups of ranks are divided by the number of data instances to obtain two groups of normalized probabilities (lr_normalized and ab_normalized). The rank will be used to calculate the pairwise diversity.

With the normalized probabilities, a global threshold can be set to determine the predicted classes by each single classifiers. For example, if 0.7 is set as the global threshold, then instances 1 and 4 are identified as positive class by Logistic Regression, while instances 1 and 3 are classified as positive by AdaBoost. At last, the pairwise diversities are calculated from the ranks, and the algorithm of determining the single models based on diversities is described as follows:

Algorithm 1: Base Model Selection

1. Given a testing set $S = \{s_i | i = 1, 2 \dots n\}$ and a group of trained single classifiers $C = \{c_j | j = 1, 2 \dots m\}$, where n and m denote the number of samples in testing set and the number of trained single classifiers, respectively;
 2. **For** each classifier c_j in C :
 3. Use c_j to make predictions for each s_i in S to obtain a set of probabilities $P_j = \{p_{ij} | i = 1, 2 \dots n\}$, where p_{ij} represents the probability of s_i being positive generated by c_j ;
 4. Sort all s_i according to their p_{ij} in ascending order, and then obtain a set of ranks $R_j = \{r_{ij} | i = 1, 2 \dots n\}$, where r_{ij} denotes the rank of s_i in all ordered test samples given by c_j ;
 5. **End For**
 6. Calculate the root-mean-square deviation (RMSD) of each two R_j as the metric of pairwise diversity $D_{pq} = \sqrt{\frac{\sum_{i=1}^n (r_{ip} - r_{iq})^2}{n}}$;
 7. Choose the k single classifiers c_j with highest RMSDs as the base models for constructing the ensemble.
-

In addition to RMSD described in the above algorithm, another metric called Zero-One Loss (ZOL) [Domingos, 1997] is also calculated to represent the pairwise diversity. In binary classification, the two classes are usually named as positive and negative. Simply, 1 and 0 are used to represent positive and negative, respectively. Thus, ZOL is defined as the number of instances with different predicted classes (sum of (1-0)). RMSD is used to indicate the difference between the ranks, whereas ZOL denotes the difference between the

predicted classes. An example for calculating the two types of pairwise diversity is presented in Table 3.2 as follows:

Table 3.2 Example of Calculating Two Types of Pairwise Diversity

Instance	lr_rank	lr_normalized	lr_pred	ab_rank	ab_normalized	ab_pred
0	2	0.4	0	3	0.6	0
1	4	0.8	1	4	0.8	1
2	3	0.6	0	2	0.4	0
3	1	0.2	0	5	1	1
4	5	1	1	1	0.2	0

0: negative; 1: positive

Similar to the example shown in Table 3.1, assuming that the global threshold is set to 0.7, so that the predicted classes by each single classifier is are shown as the two columns lr_pred and ab_pred in Table 3.2. Based on Algorithm 1, from the data in Table 3.2, The RMSD of Logistic Regression and AdaBoost is calculated as the following equation:

$\sqrt{\frac{(2-3)^2+(4-4)^2+(3-2)^2+(1-5)^2+(5-1)^2}{5}} = 2.61$. The ZOL is calculated as $|0 - 0| + |1 - 1| + |0 - 0| + |0 - 1| + |1 - 0| = 2$, which means that the two classifiers have different predictions on 2 data instances.

In summary, ZOL describes the diversity in terms of the predicted class, whereas RMSD measures the diversity in terms of the rank (equivalent to the normalized probability), which represents the confidence of the prediction. Some insignificant differences are taken into account by RMSD. For example, although the ranks (lr_rank and ab_rank) of instance 0 in Table 3.2 are different (2 and 3 respectively), both classifiers identify this instance as negative. Therefore, ZOL depicts the explicit diversity, while RMSD also consider the implied diversity and the measurement is relatively more fine-grained. Finally, all base models are determined by the method described above in the Base Model Selector.

3.2.4 Relabeler

The function of *Relabeler* is to design a new training set for training the meta-level model. The aim is to identify the proper base model for a given instance. A training set consists of two parts: feature vectors and labels. The former describes the characteristics of data instances, while the latter indicates the class of instances in a certain class space.

For example, feature vectors consisting of age, gender, height, weight and income could describe persons. They can be mapped to a class space of “rich” and “poor” by a certain classifier, and also to another class space of “active” and “passive” by another classifier. The same group of feature vectors could be used for solving different classification problems as the labels vary. Therefore, in order to obtain such a meta model, the feature vectors of the training set for single models need to be mapped to another class space with base models being the labels. In other words, a new training set is derived from relabeling the consistent feature vectors with labels of different base models, so that the meta model learnt from this training set could assign a proper base classifier for each unknown instance.

The new label for each instance in the training set is determined by comparing the predictions of base models for this instance. Similar to base model selector, the rank of each instance is used for comparison rather than directly probabilities comparison. This is also because the AUC score is selected as the evaluation metric. Therefore, based on the previous explanation of AUC (see Section 2.1.4.5), a base classifier is considered to be more suitable for a certain instance if it could assign a higher rank to the instance in descending order than other base models when the instance is positive (or a lower rank when negative). Accordingly, the algorithm of relabeling is designed as follows:

Algorithm 2: Relabeling Training Set for Meta Model

1. Given a training set $S = \{s_i | i = 1, 2 \dots n\}$ obtained from pre-processor and a group of base classifiers $B = \{b_j | j = 1, 2 \dots m\}$ determined by base model selector;
 2. **For** each base classifier b_j in B :
 3. Use b_j to make predictions for each s_i in S to obtain a set of probabilities $P_j = \{p_{ij} | i = 1, 2 \dots n\}$, where p_{ij} represents the probability of s_i being positive generated by b_j ;
 4. Sort all s_i according to their p_{ij} in ascent order, and then obtain a set of ranks $R_j = \{r_{ij} | i = 1, 2 \dots n\}$, where r_{ij} denotes the rank of s_i in all ordered samples given by b_j ;
 5. **End For**
 6. **For** each training sample s_i in S :
 7. **If** s_i is positive:
 8. relabel s_i as b_j which generates the largest r_{ij} ;
 9. **End If**
 10. **If** s_i is negative:
 11. relabel s_i as b_j which generates the smallest r_{ij} ;
 12. **End If**
 13. **End For**
 14. **Return** the relabeled training set.
-

The training set derived from the algorithm above indicates the proper base model for each instance, and then it is further used for training meta-level model, so that the meta model can be used to predict the appropriate base models for unknown instances. An example of relabeling according to Algorithm 2 is presented in Table 3.3.

Table 3.3 Example of Relabeling

Instance	lr_rank	ab_rank	original_label	new_label
0	2	3	Positive	AdaBoost
1	4	2	Positive	Logistic Regression
2	3	4	Negative	Logistic Regression
3	1	5	Positive	AdaBoost
4	5	1	Negative	AdaBoost

In Table 3.3, the original label of instance 1 is positive, and Logistic Regression can assign it a higher rank than that of Ababoost, i.e., $4 > 2$, in the ascent order, which represents that Logistic Regression has more confidence to predict instance 1 as positive. Therefore, instance 1 is relabeled as “Logistic Regression” to indicate that Logistic Regression is more proper for this instance. On the contrary, for instance 4 in Table 3.3, its original label demonstrates that it belongs to the negative class. AdaBoost can assign it a lower rank than that of the Logistics Regression in the ascent order, i.e., $1 < 5$. That means that AdaBoost is more confident to predict instance 4 as negative. Thus, instance 4 is relabeled as “AdaBoost” to indicate that AdaBoost is more proper for this instance. In summary, the ranks assigned to each instance by the single classifiers imply the probabilities of this instance to be positive class. Hence, a higher probability indicates a higher confidence of the classifier for its prediction. For a certain instance, the more confident classifier is considered the more proper one.

Chapter 4: Experiments and Evaluation

This chapter will firstly go through the datasets utilized to validate the proposed ensemble method, which is followed by a brief introduction of the system implementation process. The experiments design is then discussed, as well as the analysis of the results.

4.1 Overview of Datasets

Two binary classification datasets were selected to validate the proposed ensemble method. The first dataset is the Repeat Buyer Prediction dataset, obtained from a machine learning competition, which was held by International Joint Conference on Artificial Intelligence (IJCAI) and Alibaba Group in 2015 [IJCAI, 2015]. The other dataset, Census Income Prediction, was obtained from University of California, Irvine (UCI) machine learning repository [UCI. 1996]. More detailed information of these two datasets are introduced below.

4.1.1 Repeat Buyer Prediction (RBP) [IJCAI, 2015]

This dataset is provided by Tmall, which is the largest business-to-consumer (B2C) online shopping website of China under Alibaba Group. Tmall is a platform for merchants to sell their branded goods to customers, therefore, it works as a coordinator between merchants and customers. The activities between merchants and customers including each transaction and all types of user behaviors are recorded for data mining and building predictive system through machine learning techniques, which could help merchants for solving practical problems so as to improve their business.

Merchants usually provide various promotion sales on particular dates (such as Boxing Day, Black Friday in western countries and Double 11 in China), to promote their brand, acquire new customers, and strengthen customer loyalty. Since many of new

customers are so-called one-time deal hunters, the promotions are considered rarely produce long-term effects on businesses. The return on investment (ROI) of merchants can be enhanced if the potential loyal customers for each merchant could be targeted (DataSet, 2015). Overall, the question of the competition IJCAI, as well as extended to this research, is to identify the potential loyal customers for each merchant based on the given dataset by building a predictive system which is capable to recognize whether a customer would be a repeat buyer of a merchant or not.

Table 4.1 The Definitions of Data Fields in User Behavior Logs in RBP

Data Field	Data Type	Definition
user_id	integer	A unique id for the shopper
merchant_id	integer	A unique id for the merchant
item_id	integer	A unique id for the item
brand_id	float	A unique id for the brand of the item
cat_id	integer	A unique id for the category of the item
time_stamp	integer	Date the action took place (format: mmdd)
action_type	integer	Type of the action, which is enumerated as the set {0, 1, 2, 3}, where 0 represents click, 1 is for add-to-cart, 2 denotes purchase and 3 is for add-to-favorite

The dataset consists the behavior log of anonymized users accumulated during the 6 months before and on the “Double 11” day, with labels indicating whether a customer is a repeat buyer of a merchant or not. As we did not take part in the competition, the dataset we have is only a subset of the complete dataset that the competitors used. The dataset is 1.92 GB in total and stored as three comma-separated values (CSV) files: user behavior logs, user profile information and training and testing. The definitions of data fields in each file are described as shown in Table 4.1, 4.2 and 4.3, respectively.

Table 4.2 The Definitions of Data Fields in User Profile Information in RBP

Data Field	Data Type	Definition
user_id	integer	A unique id for the shopper
gender	float	Gender of the shopper: 0 for female, 1 for male, 2 and NULL for unknown
age_range	float	Age range of the shopper: 1 for less than 18, 2 for [18, 24], 3 for [25, 29], 4 for [30, 34], 5 for [35, 39]; 6 for [40, 49], 7 and 8 for more than 49, 0 and NULL for unknown

Table 4.3 The Definitions of Data Fields in Training and Testing Data in RBP

Data Field	Data Type	Definition
user_id	integer	A unique id for the shopper
merchant_id	integer	A unique id for the merchant
label	integer	A binary label {0, 1} indicating whether a shopper is a repeat buyer of a merchant, where 1 represents repeat buyer and 0 is for non-repeat buyer

As the tables shown above, some data types in the raw data are not appropriate for this research, such as gender, age range, and brand_id, which are supposed to be integer. Hence, data transformation is performed in the pre-processing phase.

4.1.2 Census Income Prediction (CIP)

This data was extracted from the census bureau database for census income prediction. It is popular used in research area for validating and evaluating binary classification algorithms and methods, as it does not require feature engineering and it is relatively convenient to use. There are 6 continuous attributes and 8 categorical attributes in the raw data. The labels indicate whether an anonymous person makes an income over \$50K/yr. Table 4.4 lists the definitions of each data field. Note that the fields in data type of string are all categorical, therefore they need to be transformed to proper formats during pre-processing for further operations.

Table 4.4 The Definitions of Data Fields in CIP

Data field	Data type	Definition
age	integer	The age of an anonymous person
workclass	string	The work class of an anonymous person
fnlwtg	integer	Independent estimates of the civilian non-institutional population of the US
education	string	The highest degree of an anonymous person
education_num	integer	Number of years an anonymous person is under education
marital_status	string	The marital status of an anonymous person
occupation	string	The occupation of an anonymous person
relationship	string	The role of an anonymous person in a family
race	string	The race of an anonymous person
sex	string	The gender of an anonymous person
capital_gain	integer	The capital gain of an anonymous person
capital_loss	integer	The capital loss of an anonymous person
hours_per_week	integer	The working hours of an anonymous person per week
Native_country	string	The native country of an anonymous person
Labels	string	Indicates whether an anonymous person makes over \$50K/yr

4.2 System Implementation

4.2.1 Pre-processing

In this thesis, pre-processing is mainly performed by Pandas, and the task could be generally summarized as two parts: data transformation and feature engineering.

4.2.1.1 Transformation

Two datasets have been used. They require different data transformation operations. The following highlights the transformation for each dataset.

4.2.1.1.1 RBP Dataset

For the RBP dataset, firstly, the data stored in three separate tables need to be merged, so that further operations can be conducted on the merged data. To avoid producing redundant rows, the three tables are merged by inner join, during which the user behavior

logs table and the user profile information table are joined on “user_id”, and then joined with the training and testing data table on pair of “user_id” and “merchant_id”. Secondly, the data types of some data fields need to be converted to more proper ones. For example, brand_id is originally in float type, then it is converted to integer which is more proper to represent ID fields. Thirdly, the time_stamp is parsed into the data type of datetime64, which is defined in NumPy to indicate date and time. At last, the rows containing missing values are discarded directly (1.41%). The final merged data is in the shape as shown in Table 4.5.

Table 4.5 An Example of the Merged Data in RBP

	user_id	mer_id	gender	age_range	item_id	brand_id	cat_id	time_stamp	action_type	label
0	255390	4585	1	6	934988	8011	611	2015-11-11	0	1
1	255390	4585	1	6	616588	8011	611	2015-11-11	0	1
2	255390	4585	1	6	934988	8011	611	2015-11-11	0	1
3	255390	4585	1	6	934988	8011	611	2015-11-11	0	1
4	255390	4585	1	6	183925	8011	1075	2015-11-11	0	1

The first column of the table is the indexes of rows, which is only an indicator. From the table it can be observed that many duplicated rows in the merged data (for example, rows 0, 2 and 3), and they are the real logs of the repeated activities. In this dataset, around 37.1% of the whole data instances are duplicated and they are representing actual repeated user behaviors. Therefore, these data instances can be further processed for feature engineering, e.g., the number of times a user has browsed the particular merchant. Those data instances, though duplicated, should not be deleted. Some data statistics from various perspectives are presented in Table 4.6, Table 4.7, Table 4.8, Table 4.9 and Table 4.10, respectively.

Table 4.6 Statistics of ID Fields in RBP

	Pair	User	Merchant	Item	Brand	Category
Number	257088	208839	1993	163300	3170	1086

Table 4.7 Statistics of Action Types in RBP

	Click	Favor	Cart	Purchase	Total
All Stage	2340939	99947	15	343541	2784442
Double 11	1267836	17088	4	343541	1628469

Table 4.8 Statistics of Age Ranges in RBP

	<18	18~24	25~29	30~34	35~39	40~49	>49	Unknown
Number	11	26148	55511	39742	20347	17680	4019	45381

Table 4.6 indicates the number of unique IDs for each ID field. The column ‘Pair’ represents the unique pairs of user and merchant. Table 4.7 demonstrates the numbers of different action types both during the whole period and on the promotion day, in which it can be found that all the purchases were made on promotion day. Table 4.8 shows the distribution of different age groups, from which it can be observed that people at age from 25 to 34 are the majority of online shoppers.

Table 4.9 Statistics of Gender in RBP

	Female	Male	Unknown
Number	143064	60670	5105

Table 4.10 Statistics of Labels in RBP

	Repeat Buyer	Non-Repeat Buyer
Number	15833	241255

Table 4.9 presents the distribution of gender, and it shows that women are the majority of the online shoppers. Table 4.10 indicates the numbers of pairs of merchant and user which are labeled as positive or negative. As shown in this table, the dataset suffers a serious imbalanced class problem, i.e., the ratio between the number of repeat buyers (positive instance) and non-repeat buyers (negative instances) is around 1:15.

4.2.1.1.2 CIP Dataset

For the CIP dataset, to begin with, the categorical features in datatype of string need to be transformed into integers, which are used to represent different categories. For instance, female, under the data field of sex, is transformed to 0 while male is transformed to 1. Secondly, all the categorical features are vectorized as arrays to properly express themselves, since the computation on the cardinal number seldom has meaning. The vectorization is accomplished by a method called One Hot Encoding (OneHotEncoder, 2010), which turns each value of a categorical feature to a n-dimension vector, where n denotes the number of unique values of this feature. Therefore, the vector can represent a certain value by presenting the corresponding element of the vector as 1 and leaving the rest elements as 0. For example, female (0) is vectorized as [1, 0], while male is vectorized as [0, 1]. Table 4.11 shows some examples of the raw data in CIP dataset.

Table 4.11 Examples of CIP raw data [UCI, 1996]

	age	work_class	fnlwgt	education	education_num	marital_status	occupation
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial
8	31	Private	45781	Masters	14	Never-married	Prof-specialty
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial

Table 4.11 Examples of CIP raw data (cont'd)

	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	label
0	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	Husband	White	Male	0	0	13	United-States	<=50K
2	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	Husband	Black	Male	0	0	40	United-States	<=50K
4	Wife	Black	Female	0	0	40	Cuba	<=50K
5	Wife	White	Female	0	0	40	United-States	<=50K
6	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
7	Husband	White	Male	0	0	45	United-States	>50K
8	Not-in-family	White	Female	14084	0	50	United-States	>50K
9	Husband	White	Male	5178	0	40	United-States	>50K

Table 4.12 Statistic of Labels in CIP

	<= 50K	>50K
Number	33988	11206

From Table 4.12, it can be observed that the CIP dataset still faces an imbalanced class problem, but it is not as serious as that of RBP (the ratio between the number of instances > 50K (positive instances) and the number of instances <= 50K (negative instances) is around 1: 3). In addition, the instances with missing values and the duplicated instances are directly dropped for CIP.

4.2.1.2 Feature Engineering

As mentioned before, CIP dataset does not require feature engineering, so that in this thesis, feature engineering is only performed for RPB dataset. New features extracted from

the original ones of raw data are mainly in two types: count features and ratio features. The former ones are derived from accumulating the related instances, and the latter ones are obtained from one feature divided by another. Furthermore, similar to CIP dataset, vectorization is performed on categorical features. In this thesis, some count features are derived from counting multiple vectorized categorical values and then integrating those counts. For example, a feature was designed to represent the number of clicks on each brand that is browsed by a certain shopper. Firstly, all brand IDs are vectorized as 3170-dimension vectors. Secondly, the numbers of clicks on each brand browsed by each user are counted, then the vectors representing the counted brand IDs are multiplied by the corresponding numbers respectively. At last, all vectors are added up to obtain the expected vector for each user. The final created feature consists of vectors with multiple elements being different integers to indicate the counts of clicks on the corresponding brands. In this way, the final design matrix could be highly sparse, and thus it is formatted as a sparse matrix, which is a data structure defined in SciPy and contributes to reducing storage space and speeding up the subsequent computations.

On the other hand, all the new features are created in terms of three aspects: users, merchants and interactions between user and merchant. In other words, features are extracted from the raw data to describe the behaviors of users, the characteristics of merchants and the interactive behaviors between users and merchants. In this thesis, 74 new features were designed on the basis of the four original features of the raw data, and the definitions of the new features are summarized as shown in Table 4.13.

After all the process described above, the processed data are randomly split into training and testing set according to a certain ratio. Usually, the ratio has to be in the range

between 0.6 and 0.8, so that the trained models are generalized. In this thesis, 0.8 was taken as the ratio which means 80% of the whole data are used for training and the rest 20% are used for testing. As the given dataset is only part of the complete dataset with imbalanced classes, and the number of features to deal with was comparatively large, therefore a relatively larger ratio was chosen so that the models are better fitted in this thesis.

Table 4.13 The Definitions of Extracted Features in RBP

Category	Feature Names	Data Type	Definition
Statistics	major_gender, major_age_range	vector	The major gender and age range of the user who browsed a certain merchant
Counts of IDs	u_mer_cnt, u_item_cnt, u_brand_cnt, u_cat_cnt	integer	The numbers of merchants, items, brands and categories that a certain user browsed
	m_rebuyer_cnt, m_user_cnt		The numbers of repeat buyers and total users who browsed a certain merchant
	m_item_cnt, m_brand_cnt, m_cat_cnt		The numbers of items, brands and categories that were browsed by users who browsed a certain merchant
	p_item_cnt, p_brand_cnt, p_cat_cnt		The numbers of items, brands and categories that were browsed by a certain user who browsed a certain merchant
Counts of Actions	u_actype_cnt, u_activity_cnt, u_purchase_cnt, u_cart_cnt, u_fav_cnt, u_actonpd_cnt	integer	The numbers of action types, total activities, purchases, add-to-carts, add-to-favorites and actions on promotion day that were made by a certain user
	m_activity_cnt, m_purchase_cnt, m_cart_cnt, m_fav_cnt		The numbers of total activities, purchases, add-to-carts and add-to-favorites that were made by users who browsed a certain merchant
	p_actype_cnt, p_activity_cnt, p_purchase_cnt, p_cart_cnt, p_fav_cnt		The numbers of action types, total activities, purchases, add-to-carts and add-to-favorites that were made by a certain user who browsed a certain merchant

Category	Feature Names	Data Type	Definition
Counts of Actions on IDs	u_iact_cnt, u_icart_cnt, u_ifav_cnt, u_ipur_cnt, u_bact_cnt, u_bcart_cnt, u_bfav_cnt, u_bpur_cnt, u_cact_cnt, u_ccart_cnt, u_cfav_cnt, u_cpur_cnt	vector	The numbers of total actions, add-to-carts, add-to-favorites, purchases that were made by a certain user for items, brands and categories
	u_mact_cnt		The numbers of total activities made by a certain user to merchants
	m_iact_cnt, m_icart_cnt, m_ifav_cnt, m_ipur_cnt, m_bact_cnt, m_bcart_cnt, m_bfav_cnt, m_bpur_cnt, m_cact_cnt, m_ccart_cnt, m_cfav_cnt, m_cpur_cnt		The numbers of total actions, add-to-carts, add-to-favorites, purchases that were made by users who browsed a certain merchant for items, brands and categories
	p_bcart_cnt, p_bfav_cnt, p_bpur_cnt, p_ccart_cnt, p_cfav_cnt, p_cpur_cnt		The numbers of add-to-carts, add-to-favorites and purchases that were made by a certain user who browsed a certain merchant for brands and categories
Counts of Days	u_day_cnt	integer	The number of days on which a certain user was active
	m_day_cnt		The number of days on which a certain merchant was browsed
	p_day_cnt		The number of days on which a certain user had activities with a certain merchant
Ratios	u_cact_ratio, u_fact_ratio, u_pact_ratio, u_actonpd_ratio	float	The ratios of u_cart_cnt, u_fav_cnt, u_purchase_cnt and u_actonpd_cnt to u_activity_cnt, respectively
	m_cact_ratio, m_fact_ratio, m_pact_ratio		The ratios of m_cart_cnt, m_fav_cnt and m_purchase_cnt to m_activity_cnt, respectively
	p_pact_ratio		The ratio of p_purchase_cnt to p_activity_cnt

Category	Feature Names	Data Type	Definition
Duration	u_duration	integer	The duration from the promotion day back to the day when a certain user became active for the first time
	m_duration		The duration from the promotion day back to the day when a certain merchant was browsed for the first time
	p_duration		The duration from the promotion day back to the day when a certain user browsed a certain merchant for the first time

4.2.2 Single Model Training

After pre-processing, single models are trained on the designed training set by various machine learning algorithms including logistic regression, gradient boosting machine, AdaBoost, random forests and factorization machine. Then the trained models are tested on the testing set for evaluations. The process of learning and testing usually needs repeated trials so that a group of relatively optimal hyperparameters for a certain model is determined. Due to the large size of the dataset, exhaustive search for parameter tuning is extremely time consuming. Therefore, for each time of training and testing, only one parameter was tuned within an empirical range by a relatively small and fixed step, and leave the rest parameters unchanged. At last, only the models that generate the best performances for each algorithm are passed to base model selector for further operations.

4.2.3 Ensemble Construction

The final ensemble model was constructed based on the trained single models. This section is organized according to the process of construction, which is summarized as steps below:

1. Determine base models for combining;

2. Relabel the training set and learn the meta model;
3. Use meta model to combine the selected base models.

4.2.3.1 Base Model Selection

Table 4.14 Probabilities and Ranks for Testing Instances

	lr	lr_rank	ab	ab_rank	xgb	xgb_rank
0	0.326982	9290	0.493181	11474	0.318909	9549
1	0.387424	19571	0.493652	18514	0.427959	23139
2	0.373874	17092	0.493867	24173	0.439905	25273
3	0.361815	14987	0.491878	5222	0.267688	6449
4	0.327224	9339	0.493537	16329	0.403704	18644
5	0.378829	17915	0.494864	40616	0.543609	39324

As explained in system design, for determining the base models, firstly, all trained single models were tested on the testing set, thus, each single model generated a group of probabilities for the testing instances. Secondly, all the testing instances were ascent ordered according to their predicted probabilities, and the corresponding rank was assigned to each instance. The instance located at a higher position was assigned a lower rank. Table 4.14 shows a proportion of the probabilities with the associated ranks, generated by logistic regression, AdaBoost and GBM. The first column of the table is the index of testing instances. Thirdly, the RMSDs between each two single models were calculated to represent the pairwise diversities. Finally, the single models with first n highest RMSDs were selected as the base models. If $n = 1$, only two single models are selected, accordingly, the meta model is trained to be a binary classifier. Otherwise, the meta model is a multiclass classifier.

4.2.3.2 Relabeling and Meta Model Training

With the base models determined, the training set that had been used for training single models was relabeled, so that a meta-level classifier can be trained on this relabeled training set, and it is capable to recognize the proper base model for each testing instance. Similar to calculating the pairwise diversities, the new labels are also determined based on ranking, except for determining labels, base models are applied on the training set rather than the testing set. For simplicity, it is assumed that the number of base models is two ($n = 1$), and the two base models are logistic regression and GBM (lr and xgb). Therefore, the new class space consists of two class labels (lr and xgb), and the trained meta model is a binary classifier.

For each training instance, if its original label is 0 (negative), then it is relabeled as the base model which generates a lower rank, otherwise, it is relabeled as the base model which generates a higher rank. Table 4.15 shows a portion of the probabilities and ranks with both original labels and relabeled labels to illustrate the relabeling.

Table 4.15 Relabeled Training Instances

Instance	lr	lr_rank	xgb	xgb_rank	original_label	meta_label
0	0.225102	658	0.698827	49038	1	xgb
1	0.267478	2374	0.448182	26904	0	lr
2	0.509196	37998	0.002849	2295	0	xgb
3	0.494904	36388	0.383315	15742	1	lr
4	0.531969	40124	0.404014	18726	1	lr
5	0.471655	33466	0.251891	5478	0	xgb
6	0.297399	5204	0.564898	41307	1	xgb
7	0.381879	18495	0.634423	46109	0	lr

The process of meta model training is same with that of training single models. In this thesis, both a linear algorithm (logistic regression) and a non-linear algorithm (GBM)

were applied on the relabeled training set to learn the meta model, and the ensemble models constructed with these two different meta models were compared.

4.2.3.3 Combination of Base Models

The last step of the ensemble model construction is utilizing the trained meta model to combine the base models. Figure 4.1 demonstrates the principle of the combination. The meta model works as a consultant of the unknown instances, it recommends a proper base model for each unknown instance, and then the recommended base model makes the prediction for this unknown instance. By applying this method, all the base models are integrated to make predictions for unknown instances.

Figure 4.1 Working Principle of the Constructed Ensemble Model

4.3 Experiment Design

In this thesis, experiments are designed to evaluate performance (AUC and F_1 score) and overhead that is introduced to the system in terms of time consumption. In addition, both single models and ensemble models are taken into account for evaluation. For each single model, the changing of performance is traced as its parameters vary. Multiple parameters are selected for each single model, and only one of them is tuned each time with other parameters unchanged. The combination of a set of parameters leading to a relatively best performance for each single model is considered as the best parameters of this single model. Then single models with best parameters are compared horizontally.

Four machine learning algorithms (Logistic Regression, AdaBoost, Xgboost and Random Forests) are selected to train single models, and the list of tuned parameters for these various single models is highlighted as follows:

- C: Inverse of regularization strength, is used to prevent the model from overfitting. Smaller values specify stronger regularization;
- max_iter: the maximum number of iterations for Logistic Regression;
- n_estimator: the number of constructed decision trees for the tree-based models (AdaBoost, Xgboost, Random Forests);
- learning_rate: a rate indicating the learning rate of loss function solver.
- max_depth: the maximum tree depth of the constructed decision tree for the tree-based models

Table 4.16 Logistic Regression for RBP

Parameter	C	max_iter
Value	0.1, 0.2 , 0.3, 0.4, 0.5	100 , 150, 200, 250, 300

Table 4.17 Adaboost for RBP

Parameter	n_estimators	learning_rate
Value	50 , 100, 150, 200, 250	0.2, 0.4, 0.6 , 0.8, 1.0

Table 4.18 Xgboost for RBP

Parameter	n_estimators	learning_rate	max_depth
Value	50 , 100, 150, 200, 250	0.1 , 0.2, 0.3, 0.4, 0.5	3, 4 , 5, 6, 7

Table 4.19 Random Forests for RBP

Parameter	n_estimators	max_depth
Value	200, 250, 300, 350 , 400	3, 4, 5 , 6, 7

For RBP, the selected parameters and values are listed in Table 4.16 to 4.19 above, the value in bold indicates the default value of that parameter. (Default value is set as the fixed one when tuning other parameters which are selected randomly.)

Table 4.20 Logistic Regression for CIP

Parameter	C	max_iter
Value	0.1, 0.2 , 0.3, 0.4, 0.5	50, 100 , 150, 200, 250

Table 4.21 Adaboost for CIP

Parameter	n_estimators	learning_rate
Value	50 , 100, 150, 200, 250, 300	0.2, 0.4, 0.6, 0.8, 1

Table 4.22 Xgboost for CIP

Parameter	n_estimators	learning_rate	max_depth
Value	50 , 100, 150, 200, 250	0.1, 0.2 , 0.3, 0.4, 0.5	3, 4, 5, 6, 7, 8 , 9

Table 4.23 Random Forests for CIP

Parameter	n_estimators	max_depth
Value	50, 100, 150, 200, 250	5, 6, 7, 8 , 9

For CIP, the selected parameters and values are listed in table 4.20 to 4.23 above. For analyzing the impact of each individual parameter on performance and overhead, only one parameter is tuned each time while the other parameters are set as the default values. The default value of each parameter is determined with the consideration of both performance and overhead. In other words, only those values that lead to acceptable performances with less overheads are set as the default values of the parameters. In addition to the presented values, much more combinations of parameters are tested in this thesis, and only part of them are used for visualization.

4.4 Results and Analysis

This section introduces the experimental results, which include the evaluation and analysis of each of four selected single models (Logistic Regression, AdaBoost, Xgboost, Random Forests) and a comparison of multiple ensemble methods. Section 4.4.1 discusses the performance and overhead of each single model, while section 4.4.2 summarizes the result of ensemble methods. In this thesis, for simplicity, only two single models are selected as base models for combining. Three methods of base model selection are evaluated and compared, and they are listed as follows:

- RMSD diversity-based: the two single models with highest pairwise RMSD of all are selected as the base models for combining;
- ZOL diversity-based: the two single models with highest pairwise ZOL of all are selected as the base models for combining;

- Performance-based: the two single models with highest performance (AUC and F_1 score) of all are selected as the base models for combining;

In addition, three other ensemble methods are adopted in the thesis for a comparison with the proposed ensemble method, and they are listed as follows:

- Averaging: final predictions are obtained from averaging the outputs of the two base models;
- Stacking with Logistic Regression as the meta model: Logistic Regression is used as the meta model. There is a holdout validation dataset for base models, and the training set for meta model consists of the outputs of the two base models on the validation set (as features) and the original labels (as label);
- Stacking with Xgboost as the meta model: same with previous one except for Xgboost as meta model.

Note that all the three methods of base model selection are applied in each of the four ensemble methods.

4.4.1 Analysis of Single Models

4.4.1.1 Performance

Figure 4.2 shows the result of the AdaBoost performance for the RBP dataset, from which no direct connection was found between the value of AUC and F_1 score. A higher AUC score does not necessarily correspond to a higher F_1 score. In Adaboost, $n_estimator$ represents the number of decision trees. For the RBP dataset, as the value of $n_estimator$ grows, both AUC and F_1 score increased at first and then decreased, but with the different inflection point. When the value of $n_estimator$ reaches a certain value, the complexity of the model becomes too high, which may lead to the model overfitting the training data so

that the performance could be affected. Here a relatively smaller value of $n_estimator$ (50 representing 50 trees) was chosen as the default value for tuning learning rate. The principle of Adaboost [Collins, 2002] suggests that when the learning rate is too small, the loss function may not able to converge within the increment of 50 trees. Hence, a relatively bigger learning rate (≥ 0.6) could better fit this dataset for AUC. On the other hand, the F_1 score is always close to zero.

Figure 4.2 Result of AdaBoost Performance in RBP

Figure 4.3 shows the result of AdaBoost performance in CIP dataset, from which the data suggested that a larger $n_estimator$ leads to a higher AUC score. However, as $n_estimator$ grows, the growth of AUC became slower. On the other hand, a larger learning rate also leads to a better performance for both AUC and F_1 score on CIP.

Figure 4.3 Result of AdaBoost Performance in CIP

Figure 4.4 shows the performance of Logistic Regression on RBP dataset. A relatively smaller C (stronger regularization) indicated a very strict penalty on the error instance when learning the classifier. It is observed that a smaller C has benefits for both AUC and F_1 score on RBP. On the other hand, as max_iter increases, a growth in performance is observed to a certain extent.

Figure 4.4 Result of Logistic Regression Performance in RBP

For CIP, (see Figure 4.5), it is observed that C contributed less to overall performance, and the impact is relatively random. In addition, since the number of instances is not big in CIP dataset and the number of features is also limited, the model could completely

converge within 100 iterations. Thus, when max_iter further increases, almost no effect is observed to both AUC and F₁ score.

Figure 4.5 Result of Logistic Regression Performance in CIP

Figure 4.6 and Figure 4.7 show the performance of Xgboost on RBP and CIP, respectively. For both RBP and CIP, a relative lower learning rate is preferred. Both n_estimator and max_depth represent the complexity of a model, in which max_depth indicates the maximum allowable depth of each tree. Normally, a model with relatively lower complexity could benefit AUC, on the other hand, a model with relatively higher complexity could increase F₁ score. Specially in CIP, as shown in Figure 4.7, a higher max_depth could increase both AUC and F₁ score.

Figure 4.6 Result of Xgboost Performance in RBP

Figure 4.7 Result of Xgboost Performance in CIP

Figure 4.8 and Figure 4.9 present the performance of Random Forest in RBP and CIP datasets. It is observed that the increase of n_estimator helps improving AUC, on the contrary, higher F₁ scores were mostly acquired among low n_estimator values. The increasing of max_depth has positive impact on both AUC and F₁ score for both datasets. Note that there is a relatively large fluctuation in performance regarding Random Forests. Different models could be generated even with the same set of parameters in repeatedly training. This is caused by the principle of the model itself.

Figure 4.8 Result of Random Forest Performance in RBP

Figure 4.9 Result of Random Forest Performance in CIP

In summary, both the AUC and F_1 score results obtained from Xgboost are better than that for other methods, and AdaBoost does not perform well on F_1 score. On the other hand, it is worth mentioning that the performance of the Random Forest on RBP is not stable. The variance of both AUC and F_1 score is relatively high, when the model is trained with the same parameter values several times by Random Forests. The general performance of Random Forests is also lower than that of those three other models, especially for RBP. This results indicates that it might be inappropriate for the datasets in high complexity.

4.4.1.2 Overhead

Time consumption is considered as a typical overhead regarding any engineering problem. In this thesis, time spent in both learning and predicting are evaluated as index to all models.

Figure 4.10 and Figure 4.11 show the overhead of AdaBoost in RBP and CIP datasets, separately. It can be observed that the increasing of $n_estimators$ leads to a notable increasing in overhead. On the other hand, learning rate seems contribute less to the computational overhead.

Figure 4.10 Result of AdaBoost Overhead in RBP

Figure 4.11 Result of AdaBoost Overhead in CIP

Figure 4.12 and Figure 4.13 present the overhead in Logistic Regression. Overhead was affected little by C. This is reasonable as C indicates the degree of penalty rather than the learning rate. When the data size becomes bigger and complexity is higher. i.e., RBP, max_iter becomes the vital determining factor of learning time. Different from tree-based methods, the complexity of the Logistic Regression model does not increase as those two parameters (C and max_iter) vary. Hence, the predicting time was seldom affected, and it always maintains in a stable magnitude.

Figure 4.12 Result of Logistic Regression Overhead in RBP

Figure 4.13 Result of Logistic Regression Overhead in CIP

Figure 4.14 and Figure 4.15 show the overhead of Xgboost on RBP and CIP. Overall, the increase of learning rate, n_estimator and max_depth could all lead to a slight increasing trend of overhead in time. However, there is an exception that for CIP, the growth of learning rate can lead to a decrease of overhead. Since the magnitude was too small (<10 ms), this impact is negligible.

Figure 4.14 Result of Xgboost Overhead in RBP

Figure 4.15 Result of Xgboost Overhead in CIP

Figure 4.16 and Figure 4.17 shows the overhead of Random Forest in RBP and CIP. It is observed that both `n_estimator` and `max_depth` can affect learning time to a large extent, while `max_depth` has little impact on predicting time, and the effect is relatively random.

Figure 4.16 Result of Random Forest Overhead in RBP

Figure 4.17 Result of Random Forest Overhead in CIP

In summary, the learning time of Logistic Regression can be extremely high on the datasets in large size and high complexity, as well as AdaBoost but better than Logistic Regression. However, the predicting time of Logistic Regression is stable on a low level. The complexity of the model has effects for all three tree-based models on both learning time and predicting time. On the other hand, the overall overhead introduced by Xgboost maintains on a low level, and a better scalability is observed for Xgboost. In addition, all these four single models introduce much less overhead when working on the smaller and simpler dataset (CIP).

4.4.1.3 Best Performance of Single Models

The best performance results (AUC and F_1 score) for each single model are selected for a comparison. Figure 4.18 depicts the best results for those single models on RBP, and Figure 4.19 shows the best results for those single models on CIP.

As shown in those two figures, the differences among the single models are larger for RBP than that of CIP. The main reason is likely to do with the data size and the complexity of data.

Figure 4.18 Comparison of single model's best performance on RBP

Figure 4.19 Comparison of single model's best performance on CIP

In summary, Xgboost performs the best for both AUC and F_1 score on both RBP and CIP. All these four single models can deliver a considerable performance for both AUC and F_1 score on CIP. In addition, to some extent, the wide difference between AUC and F_1 score of those four single models for RBP can demonstrate the various concepts of these

two metrics in performance evaluation, as well as the practical value of F_1 score, which is elaborated in the next section.

4.4.2 Analysis of Ensemble Model

4.4.2.1 Base Model Selection

To determine base models for combining, pairwise diversities are measured by two methods: RMSD and ZOL, which have been described in Section 3.2.3. The diversities in RBP and CIP are presented in Table 4.24 and Table 4.25, respectively. As discussed in Section 3.2.3, the base models with the largest pairwise diversity value are selected for combination in the ensemble method. Also, in this thesis, as only 2 single models are selected as base models for combining, the meta model is learnt as a binary classifier.

Table 4.24 Diversity between single models in RBP

Diversity	lr & ab	lr & xgb	lr & rf	ab & xgb	ab & rf	xgb & rf	overhead
RMSD	6719	5969	12827	3773	11821	11875	1min 43s
ZOL	18859	5921	15086	18292	3987	14779	1min 41s

Table 4.25 Diversity between single models in CIP

Diversity	lr & ab	lr & xgb	lr & rf	ab & xgb	ab & rf	xgb & rf	overhead
RMSD	896	1036	759	590	909	907	638ms
ZOL	1458	808	876	1350	2024	944	619ms

For RBP, Logistic Regression and Random Forests are selected as base models according to the RMSD diversity, while Logistic Regression and AdaBoost are selected according to the ZOL diversity. For CIP, logistic regression and Xgboost are selected according to the RMSD diversity, while AdaBoost and random forests are selected according to the ZOL diversity. In addition, AdaBoost and Xgboost are also selected, since they generate the best performances among all the single models, for both RBP and CIP,

see Section 4.4.1.3. At last, the overhead introduced by calculating diversities are also presented in the tables. The time for calculating RMSD and ZOL are similar on both RBP and CIP.

In the following sections, the proposed ensemble method is compared to Averaging, Stacking and the best base model on both RBP and CIP. Furthermore, a linear (Logistic Regression) and a non-linear (Xgboost) meta models are both tested for Stacking, as described in Section 2.1.5.3.4. Figure 4.12 is an illustration of experimental design for those four ensemble methods.

4.4.2.2 Performance

Four ensemble methods have been evaluated. This section presents the performance results of those four ensemble methods and also shows a comparison with the results obtained from the best base model. Those four models are described as follows:

Figure 4.20 AUC result comparison between different ensemble methods for RBP

Figure 4.20 shows the performance comparison among different ensemble methods for RBP. As can be seen in the figure, the averaging of base models, which are selected based on RMSD diversity, generates a little higher AUC performance result than the best base model (Logistic Regression). The model from the proposed method based on RMSD diversity also achieves a slightly higher performance than that of the best base model (but slightly lower than averaging). However, the other two methods, i.e., Stacking (lr) and Stacking (xgb), failed to further enhance AUC. In addition, all these four ensemble methods with other two base model selections (ZOL and best performance) have no contribution to the increase of AUC.

Figure 4.21 F₁ score result comparison between different ensemble method for RBP

As shown in Figure 4.21, the proposed method and averaging increases the F₁ score with all three base model selections, compared to best base model (Logistic Regression, AdaBoost, Xgboost). The combination of two single models with the highest ZOL diversity

leads to the highest improvement on F_1 score, especially for the proposed method (up to 18.5%). Furthermore, the combination of two single models with the best performance results in the highest F_1 score. The two Stacking methods fail to improve the F_1 score.

Figure 4.22 AUC result comparison between different ensemble method for CIP

Figure 4.23 F_1 score result comparison between different ensemble method for CIP

Figure 4.22 and Figure 4.23 show the result comparison between different ensemble method for CIP dataset. Similar to RBP, these ensemble methods do not help much in improving AUC score. The proposed method with different base model selections all contribute to the increase of F_1 score. Compared to other ensemble methods, this proposed method performs better in combining base models based on diversity. When utilizing the stacking method with linear meta model, the combination of two single models with highest ZOL diversity shows an obvious negative impact on AUC result.

In summary, Averaging and Stacking methods perform slightly better in combining single models with best performance, while the proposed method contributes more to the increase of F_1 score, compared to the other two ensemble methods. The selection of base models based on ZOL diversity improves the performance most compared to the best base model. For RBP, the largest improvement achieved by the proposed ensemble method is around 18.5% on F_1 score. This can be valuable for solving practical engineering problems, as the improvement on F_1 score means that the predictive system can help the merchants reach more real repeat buyers with the same cost of promotions. Moreover, this level of improvement is achieved on part of the offline version training set. The competitors of the [IJCAI, 2015] competition had more data including the complete offline version training set and another online version training set, which is five times the offline version. The highest AUC results achieved by the two best teams of the competition are close to 0.712, respectively. On the other hand, the best AUC result for RBP achieved in this thesis is up to 0.718 and the result would be higher with more training data instances.

4.4.2.3 Overhead

Table 4.26 Ensemble Overhead in RBP

Base Model	Proposed Method		Stacking (lr)	Stacking (xgb)
	Relabel	Meta Model Training		
lr & rf	6min 51s	3min 8s	444ms	1.39s
lr & ab	5min 41s	3min 8s	418ms	1.43s
ab & xgb	6min 28s	3min 10s	488ms	1.29s

Table 4.27 Ensemble Overhead in CIP

Base Model	Proposed Method		Stacking (lr)	Stacking (xgb)
	Relabel	Meta Model Training		
lr & xgb	33.8s	3.45s	27.8ms	323ms
rf & ab	35.46s	3.38s	27.8ms	309ms
ab & xgb	34.77s	4.46s	25.5ms	291ms

Table 4.26 and Table 4.27 show the computational overhead of the ensemble methods for both RBP and CIP datasets. It can be observed that the proposed ensemble method introduced higher overhead in both relabel and meta model training, but the time consumption is still within an acceptable range. Moreover, when the time spent on combining different base models (both for relabel and meta model training) does not have much difference. For instance, results for lr & rf and ab & xgb, as shown in Table 4.26.

4.5 Comparison of AUC and F_1 Score

When solving binary classification problems in practice, the interested instances (positive class, for example, repeat buyer) are usually the minority of the dataset. The class imbalance can be very serious. For these cases, AUC may not accurately reflect the ability of a classifier, while F_1 score is considered more suitable. A simple example is taken to illustrate the comparison of AUC and F_1 score, as shown in Table 4.28.

Table 4.28 Probabilities from Two Classifiers and Labels for 20 Example Instances

Instance	Probability from Classifier 1	Probability from Classifier 2	Real Label
0	0.05	0.15	0
1	0.1	0.2	0
2	0.15	0.25	0
3	0.2	0.3	0
4	0.25	0.35	0
5	0.3	0.4	0
6	0.35	0.45	0
7	0.6	0.5	0
8	0.65	0.6	0
9	0.7	0.65	0
10	0.75	0.7	0
11	0.8	0.75	0
12	0.85	0.8	0
13	0.9	0.85	0
14	0.95	0.9	0
15	1	0.95	0
16	0.4	1	1
17	0.45	0.55	1
18	0.5	0.05	1
19	0.55	0.1	1

The first column is the index of instances (20 in total). The second and third columns are predicted probabilities from Classifier1 and Classifier2. The last column is the labels for those 20 instances (0 represents negative, 1 represents positive, 16 negative instances and 4 positive instances). Then all the 20 instances are sorted according to their predicted probabilities in ascent order as shown in Table 4.29.

Table 4.29 Ranked Instances with Real Labels and Predicted Labels

Rank from Classifier 1	Instance	Rank from Classifier 2	Instance	Predicted Label
0	0	1	18	0
0	1	1	19	0
0	2	0	0	0
0	3	0	1	0
0	4	0	2	0
0	5	0	3	0
0	6	0	4	0

1	16	0	5	0
1	17	0	6	0
1	18	0	7	0
1	19	0	17	1
0	7	0	8	1
0	8	0	9	1
0	9	0	10	1
0	10	0	11	1
0	11	0	12	1
0	12	0	13	1
0	13	0	14	1
0	14	1	15	1
0	15	1	16	1

The first and third columns are the ranked instances with their real labels. The second and forth columns are the index of instances same with that in Table 4.28. The last column is the predicted labels when threshold is set as 0.5 (50% of the instances are predicted as positive). In this case, the calculated AUC and F_1 score are shown as Table 4.30.

Table 4.30 Metrics for the Two Classifiers

	AUC	F_1 Score	Precision	Recall
Classifier 1	0.4375	0.1429	0.1	0.25
Classifier 2	0.375	0.2857	0.2	0.5

The AUC of Classifier 1 is higher than that of Classifier 2, while the F_1 score of Classifier 2 is two times than that of Classifier 1. Moreover, 2 positive instances are correctly predicted by Classifier 2, while only 1 positive instance is correctly identified by Classifier 1. In practice, for example in the problem of repeat buyer prediction, classifier 2 can help improve the ROI of merchants more. For the datasets with imbalanced class problem, the threshold can be set higher than 0.5 according to practical situation, then the F_1 scores of the classifiers similar to Classifier 1 can be even much lower with AUC unchanged (This also explains why the F_1 score of AdaBoost for RBP is close to zero).

Therefore, AUC is more suitable to measure the general ability of a classifier, and this also explains why Averaging performs better in improving AUC, because it can average the extremely predicted instances (at the start or the end of the ranked queue) to the middle part of the ranked queue.

In practice, the precision of a classifier can represent the cost for solving the problem (a lower precision leads to a higher cost, for example in Table 4.29, Classifier 1 needs to predict more instances as positive to identify more real positive instances). On the other hand, the recall can represent the profit and the threshold can represent the limited investment. For repeat buyer prediction, the investment of a merchant on promotion is limited, for example, a merchant can provide promotions for only 50 customers, then only the 50 customers at the end of the ranked queue are classified as repeat buyer. A classifier with a higher recall can help the merchant invest on more real repeat buyers. Meanwhile, a higher precision is beneficial to get a higher recall even when the threshold is stricter (higher than 0.5). A tradeoff is needed among precision, recall and the threshold, especially when the interested positive instance is the minority and extremely imbalanced. Therefore, F_1 score is more suitable for imbalanced datasets in practice, as it can naturally reflect the balance between precision and recall according to its definition.

Chapter 5: Conclusion

5.1 Accomplishments

In this thesis, a novel ensemble method is proposed, which is capable of performing adaptive selection of the proper base model for each unknown instance. Furthermore, the base models are selected according to the pairwise diversities of all trained single models. Two metrics (RMSD and ZOL) are calculated as the measure of diversity. In addition, a relabeled training set is used for learning a meta model, which can identify the proper base model for each unknown instance. The relabeling is performed according to the behaviors of the base models on the training set. Moreover, two algorithms were introduced for base model selection and relabeling, respectively. Finally, the proposed method is validated on two datasets, and the performance results of AUC and F_1 score were compared with those of other two existing ensemble techniques (Averaging and Stacking).

Based on the experimental results, the proposed ensemble method can improve the overall performance in terms of F_1 score, which is considered as a more valuable metric in practical engineering problems. There is a significant performance improvement of 18.5% on F_1 score for the RBP dataset using the proposed ensemble method compared with the best base model. This improvement can lead to a great benefit for merchants by helping them reach more real repeat buyers with a limited cost so as to effectively improve the return on investment to a notable extent. This improvement also indicates that the proposed ensemble method has an exceptional ability of dealing with imbalanced datasets. In addition, this proposed ensemble method outperformed all other methods that were applied in this thesis on CIP dataset.

The experiment results also indicate that the base model with better performance contributes notable impact to the overall performance increase of the ensemble model. In other word, the combination of two base models with better individual performance could result in a relatively higher absolute performance of the ensemble model. Moreover, a higher pairwise diversity of combined base models can lead to a further improvement toward the performance of ensemble model. Above all, the individual performances of single models and their pairwise diversities are both important for achieving a higher overall performance of ensemble model. This insight can be regard as a guidance when solving practical problems.

5.2 Limitations

There are some limitations regarding this thesis. First of all, the proposed ensemble method is designed for general purpose but only validated on two datasets. Additional datasets can be used for further validation. Secondly, the meta model is trained as a binary classifier, thus, only two base models can be combined in this proposed method. There may be a further improvement on the performance of ensemble techniques when combining more than two base models, which is not addressed in this thesis. Thirdly, only AUC and F_1 score are used for evaluating the performance and only two types of diversity are measured for base model selection. Different measurements of diversity could result in different combinations of base models. Hence, other methods for measuring the pairwise diversity can be considered and compared for quantifying the contribution of each type of diversity. Finally, the proposed ensemble method is limited to combining only probabilistic classifiers. Relabeling methods for discrete classifiers can be investigated, so that the meta model is able to combine more types of single models.

5.3 Future Research Directions

Some potential directions for the future work of this research include:

- 1) The combination of base models could be determined on the basis of predicted probabilities from the meta model, in this way bias and variance could be eliminated to some extent.
- 2) For imbalanced dataset, more effort could be put in finding a solution to increasing the performance of meta model in minority positive instances, rather than overall performance. For example, the meta model could be trained not only to alter the original labels, but also to weight each sample so that the meta model could be more sensitive to the minority instances.
- 3) A method for meta model training could be designed so that base models trained different features could be properly combined.

References

- [Alpaydin, 2010] Alpaydin, Ethem. “Introduction to Machine Learning.” (2010): 249-256.
- [Atkinson, 2012] Atkinson, Elizabeth J., et al. “Assessing fracture risk using gradient boosting machine (GBM) models.” *Journal of Bone and Mineral Research* 27.6 (2012): 1397-1404.
- [Barber, 2012] Barber, David. “Bayesian reasoning and machine learning.” Cambridge University Press, (2012).
- [Bergstra, 2006] Bergstra, James, et al. “Aggregate features and AdaBoost for music classification.” *Machine Learning* 65.2-3 (2006): 473-484.
- [Bhatnagar, 2014] Bhatnagar, Vasudha, et al. “Accuracy–diversity based pruning of classifier ensembles.” *Progress in Artificial Intelligence* 2.2-3 (2014): 97-111.
- [Bouchard-Côté, 2009] Bouchard-Côté, Alexandre, Thomas L. Griffiths, and Dan Klein. “Improved reconstruction of protolanguage word forms.” *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (2009).
- [Breiman, 1996] Breiman, Leo. “Bagging predictors.” *Machine Learning* 24.2 (1996): 123-140.
- [Canuto, 2005a] Canuto, Anne MP, et al. “Performance and diversity evaluation in hybrid and non-hybrid structures of ensembles.” *Fifth International Conference on Hybrid Intelligent Systems*. IEEE, (2005).
- [Canuto, 2005b] Canuto, Anne MP, et al. “Using weighted dynamic classifier selection methods in ensembles with different levels of diversity.” *International Journal of Hybrid Intelligent Systems* 3.3 (2006): 147-158.
- [Chen, 2016] Chen, Tianqi, and Carlos Guestrin. “Xgboost: A scalable tree boosting system.” (2016).
- [Cohen, 2003] Cohen, William, Pradeep Ravikumar, and Stephen Fienberg. “A comparison of string metrics for matching names and records.” *Kdd Workshop on Data Cleaning and Object Consolidation*. Vol. 3. (2003).
- [Collins, 2002] Collins, Michael, Robert E. Schapire, and Yoram Singer. “Logistic regression, AdaBoost and Bregman distances.” *Machine Learning* 48.1-3 (2002): 253-285.
- [Continuum, 2016] *Continuum*. [online] Available at: <https://www.continuum.io/> [Accessed 16 Sep. 2016].

- [Cortes, 1995] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." *Machine Learning* 20.3 (1995): 273-297.
- [Davis, 2006] Davis, Jesse, and Mark Goadrich. "The relationship between Precision-Recall and ROC curves." *Proceedings of the 23rd International Conference on Machine Learning*. ACM, (2006).
- [Domingos, 1997] Domingos, Pedro, and Michael Pazzani. "On the optimality of the simple Bayesian classifier under zero-one loss." *Machine Learning* 29.2-3 (1997): 103-130.
- [Fawcett, 2006] Fawcett, Tom. "An introduction to ROC analysis." *Pattern Recognition letters* 27.8 (2006): 861-874.
- [Freund, 1997] Freund, Yoav, et al. "Using and combining predictors that specialize." *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*. ACM, (1997).
- [Friedl, 1997] Friedl, Mark A., and Carla E. Brodley. "Decision tree classification of land cover from remotely sensed data." *Remote Sensing of Environment* 61.3 (1997): 399-409.
- [Friedman, 1997] Friedman, Nir, Dan Geiger, and Moises Goldszmidt. "Bayesian network classifiers." *Machine Learning* 29.2-3 (1997): 131-163.
- [Friedman, 2001] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." *Annals of Statistics* (2001): 1189-1232.
- [Friedman, 2009] Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. "glmnet: Lasso and elastic-net regularized generalized linear models." *R Package Version 1* (2009).
- [Fortmann-Roe, 2012] Fortmann-Roe, Scott. "Understanding the bias-variance tradeoff." (2012).
- [Fukunaga, 1975] Fukunaga, Keinosuke, and Patrenahalli M. Narendra. "A branch and bound algorithm for computing k-nearest neighbors." *IEEE Transactions on Computers* 100.7 (1975): 750-753.
- [Ghorbani, 2001] Ghorbani, Ali A., and Kiarash Owrangh. "Stacked generalization in neural networks: generalization on statistically neutral problems." *Proceedings of the International Joint Conference Neural Networks*. IEEE, (2001).
- [Giacinto, 2001] Giacinto, Giorgio, and Fabio Roli. "Dynamic classifier selection based on multiple classifier behaviour." *Pattern Recognition* 34.9 (2001): 1879-1881.
- [Ham, 2005] Ham, Jisoo, et al. "Investigation of the random forest framework for classification of hyperspectral data." *IEEE Transactions on Geoscience and Remote Sensing* 43.3 (2005): 492-501.

[Har-Peled, 2002] Har-Peled, Sarel, Dan Roth, and Dav Zimak. "Constraint classification: A new approach to multiclass classification." *International Conference on Algorithmic Learning Theory*. Springer Berlin Heidelberg, (2002).

[He, 2015] He, B, et al. "Repeat Buyers Prediction after Sales Promotion for Tmall Platform." In *Proceedings of the 1st International Workshop on Social Influence Analysis SocInf* 2015.
<https://docs.google.com/viewer?a=v&pid=sites&srcid=aXNpc3Rhbi51bmljZW4uZWR1LmFyfHNvY2luZjIwMTV8Z3g6M2NmMmNjZDlkNTI2ZTVjOA> [Accessed 19 Aug. 2016]

[IJCAI-15, 2015] *Repeat Buyers Prediction Competition*. [online] Ijcai-15.org. Available at: <http://ijcai-15.org/index.php/repeat-buyers-prediction-competition> [Accessed 10 Sep. 2016].

[Kohavi, 1998] Kohavi, Ron, and Foster Provost. "Glossary of terms." *Machine Learning* 30.2-3 (1998): 271-274.

[Kuncheva, 2005a] Kuncheva, Ludmila I. "Diversity in multiple classifier systems." *Information Fusion* 6.1 (2005): 3-4.

[Kuncheva, 2005b] Kuncheva, Ludmila I. "Using diversity measures for generating error-correcting output codes in classifier ensembles." *Pattern Recognition Letters* 26.1 (2005): 83-90.

[Liu, 2015] Liu, G, et al. "Report for Repeated Buyer Prediction Competition by Team 9*TAR." In *Proceedings of the 1st International Workshop on Social Influence Analysis SocInf* 2015.
<https://docs.google.com/viewer?a=v&pid=sites&srcid=aXNpc3Rhbi51bmljZW4uZWR1LmFyfHNvY2luZjIwMTV8Z3g6NmNmYTdkNzQxMTYxYzNjZQ> [Accessed on 19 Aug. 2016]

[LogisticFuction, 2016] Wikipedia. (2016). *Logistic function*. [online] Available at: https://en.wikipedia.org/wiki/Logistic_function [Accessed 16 Sep. 2016].

[Matplotlib, 2016] *Matplotlib 1.5.3 documentation*. [online] Available at: <http://matplotlib.org/> [Accessed 16 Sep. 2016].

[Nascimento, 2014] Nascimento, Diego SC, André LV Coelho, and Anne MP Canuto. "Integrating complementary techniques for promoting diversity in classifier ensembles: A systematic study." *Neurocomputing* 138 (2014): 347-357.

[Numpy, 2016] *NumPy*. [online] Available at: <http://www.numpy.org/> [Accessed 16 Sep. 2016].

[Opitz, 1999] Opitz, David, and Richard Maclin. "Popular ensemble methods: An empirical study." *Journal of Artificial Intelligence Research* 11 (1999): 169-198.

- [Pandas, 2016] *Python Data Analysis Library*. [online] Available at: <http://pandas.pydata.org/> [Accessed 16 Sep. 2016].
- [Polikar, 2006] Polikar, Robi. "Ensemble based systems in decision making." *IEEE Circuits and Systems Magazine* 6.3 (2006): 21-45.
- [Python, 2016] *Welcome to Python.org*. [online] Available at: <https://www.python.org/> [Accessed 16 Sep. 2016].
- [Rendle 2010] Rendle, Steffen, and Lars Schmidt-Thieme. "Pairwise interaction tensor factorization for personalized tag recommendation." *Proceedings of the Third ACM International Conference on Web Search and Data Mining*. ACM, (2010).
- [ROC, 2016] Wikipedia. (2016). *Receiver operating characteristic*. [online] Available at: https://en.wikipedia.org/wiki/Receiver_operating_characteristic [Accessed 16 Sep. 2016].
- [Rokach, 2009] Rokach, Lior. "Ensemble-based classifiers." *Artificial Intelligence Review* 33.1-2 (2010): 1-39.
- [Schapire, 2012] Schapire, Robert E., and Yoav Freund. "*Boosting: Foundations and algorithms*." MIT Press, (2012).
- [Schutt, 2013] Schutt, Rachel, and Cathy O'Neil. "*Doing data science: Straight talk from the frontline*." O'Reilly Media, Inc., (2013).
- [Scikit-learn, 2016] Scikit-learn.org. (2016). *scikit-learn: machine learning in Python* [online] Available at: <http://scikit-learn.org/stable/> [Accessed 16 Sep. 2016].
- [Sebastian, 2016] Sebastian Raschka's Website. (2016). *Machine Learning FAQ*. [online] Available at: <http://sebastianraschka.com/faq/docs/closed-form-vs-gd.html> [Accessed 16 Sep. 2016].
- [Shipp, 2002] Shipp, Catherine A., and Ludmila I. Kuncheva. "Relationships between combination methods and measures of diversity in combining classifiers." *Information Fusion* 3.2 (2002): 135-148.
- [Sill, 2009] Sill, Joseph, et al. "Feature-weighted linear stacking." (2009).
- [Sirlantzis, 2008] Sirlantzis, Konstantinos, Sanaul Hoque, and Michael C. Fairhurst. "Diversity in multiple classifier ensembles based on binary feature quantisation with application to face recognition." *Applied Soft Computing* 8.1 (2008): 437-445.
- [Tysmbal, 2008] Tysmbal, Alexey, et al. "Dynamic integration of classifiers for handling concept drift." *Information Fusion* 9.1 (2008): 56-68.
- [UCI, 1996] Archive.ics.uci.edu. (2016). *UCI Machine Learning Repository*. [online] Available at: <http://archive.ics.uci.edu/ml/> [Accessed 16 Sep. 2016].

- [Wang, 2006] Wang, Shuang-Quan, Jie Yang, and Kuo-Chen Chou. “Using stacked generalization to predict membrane protein types based on pseudo-amino acid composition.” *Journal of Theoretical Biology* 242.4 (2006): 941-946.
- [Windeatt, 2005] Windeatt, Terry. “Diversity measures for multiple classifier system analysis and design.” *Information Fusion* 6.1 (2005): 21-36.
- [Wolpert, 1992] Wolpert, David H. “Stacked generalization.” *Neural Networks* 5.2 (1992): 241-259.
- [Varma, 2009] Varma, Manik, and Bodla Rakesh Babu. “More generality in efficient multiple kernel learning.” *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, (2009).
- [Zouari, 2005] Zouari, Hla, Laurent Heutte, and Yves Lecourtier. “Controlling the diversity in classifier ensembles through a measure of agreement.” *Pattern Recognition* 38.11 (2005): 2195-2199.