UNIVERSITY OF CALIFORNIA,
IRVINE

# Supply-demand Forecasting For a Ride-Hailing System

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER

in Transportation Science

by

Runyi Wang

Thesis Committee:
Professor Amelia Regan, Chair
Professor R. Jayakrishnan
Professor Wenlong Jin

2017

# TABLE OF CONTENTS

# LIST OF FIGURES

Page

# LIST OF TABLES

# ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Professor Amelia Regan of the ITS at UC Irvine. The door to Prof. Regan's office was always open whenever I ran into a trouble spot or had a question about my research or writing. She consistently allowed this paper to be my own work, but steered me in the right the direction whenever she thought I needed it.

I would also like to thank my thesis committee who were involved in this research project: Professor R. Jayakrishnan and Professor Wenlong Jin. Without their passionate participation and input, the paper could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

# ABSTRACT OF THE THESIS

Supply-demand Forecasting For a Ride-Hailing System

By

Runyi Wang

Master in Transportation Science

University of California, Irvine, 2017

Professor Amelia Regan, Chair

Ride-hailing or Transportation Network Companies (TNCs) such as Uber, Lyft and Didi Chuxing are gaining increasing market share and importance in many transportation markets. To estimate the efficiency of these systems and to help them meet the needs of riders, big data technologies and algorithms should be used to process the massive amounts of data available to improve service reliability. The model developed predicts the gap between rider demands and driver supply in a given time period and specific geographic area using data from Didi Chuxing, the dominant ride-hailing company in China. The data provided includes car sharing orders, point of interest (POI), traffic, and weather information. A passenger calls a ride (makes a request) by entering the place of origin and destination and clicking Request Pickup on the Didi phone based application. A driver answers the request by taking the order. Our training data set contains three consecutive weeks of data in 2016, for large Chinese city which is referred to as City M. Though the training set is relatively small when compared to the whole of Didi's ride sharing market, it is large enough so that patterns can be discovered and generalized. These data were made available to researchers and entrepreneurs by Didi after removal of some identifying information.

# Chapter 1

# Introduction

Ride-hailing or Transportation Network Companies (TNCs) such as Uber, Lyft and Didi Chuxing are gaining increasing market share and importance in many transportation markets.

To estimate the efficiency of these systems and to help them meet the needs of riders, big data technologies and algorithms should be used to process the massive amounts of data available to improve service reliability.

Using data from the dominant ride-hailing company in China, this thesis develops a model to predict supply and demand mismatches.

Because few of Chinas 1.4 billion citizens own automobiles, the frequency with which Chinese citizens commute on taxis, buses, trains is among the highest in the world. Didi Chuxing, the dominant ride-hailing company in China processes over 11 million trips and collects over 50GB of data per day. [6]

A common way to predict ride demand is to use traditional statistical methods. These rely on current traffic information and historical traffic data, and uses models such as Auto-

Regression, Moving Average, Autoregressive Moving Average etc. We can also use machine learning methods to train regression models, inferring the relationship between future traffic demands and related big data, such as previous traffic demands and supplies, POI (Place Of Interest) information etc, to optimize the level of accuracy of the prediction. However, it can be very time-consuming to train these models, so it is crucial to choose the right model and attributes in order to balance cost and accuracy.

The model developed predicts the gap between rider demands and driver supply in a given time period and specific geographic area. The data provided includes car sharing orders, point of interest (POI), traffic, and weather information. A passenger calls a ride (makes a request) by entering the place of origin and destination and clicking Request Pickup on the Didi app. A driver answers the request by taking the order. Our training data set contains three consecutive weeks of data in 2016, for large Chinese city which is referred to as City M. Though the training set is relatively small when compared to the whole of Didi's ride sharing market, it is large enough so that patterns can be discovered and generalized. These data were made available to researchers and entrepreneurs by Didi after removal of some identifying information.

Mean absolute error was chosen as the measure for evaluation of the models. Several machine learning models including Support Vector Machine and Decision Trees were tested and compared. We also tuned the model's parameters with cross validation to get the best result. To improve the performance of our model, we combined a set of algorithms to reduce overfitting and noise.

The rest of this thesis is organized as follows. In chapter 2, we introduce the classic approach to predict rider demand. In chapter 3, we introduce the models. Chapter 4 covers the training set and testing set. In chapter 5 and 6 we discuss our experiments on different models and results. In chapter 7 we discuss how we set XGBoost hyperparameters. In chapter 8 we present our conclusions and extensions to do in the future.

# Chapter 2

# Previous Work

Traffic demand prediction is an important topic in transportation. Accurate prediction of near-future traffic demand makes it possible to generate and evaluate response strategies, which is very helpful for traffic assignment, vehicle routing and congestion reduction. Currently there exists a number of techniques to estimating traffic demands in a network. Most of them are based on historical traffic demand and current traffic situation.

In 1979, Alfa and Minh suggested a stochastic model for traffic demand distribution based on the commuters' departure time from home [1]. Heuristic rules describing travel time prediction and departure time adjustment contributes to the dynamics of urban traffic demand prediction (Chang et al. 1988 [4]). The Dynasmart evaluation model incorporating drivers response to information, traffic flow behavior, and the resulting changes in the characteristics of network paths, into an integrated simulation framework was introduced in 1994 [10]. DynaMIT introduced a mesoscopic demand and supply simulation, with reference to departure time, pre-trip path and mode choice decisions and en-route path choice decisions (Moshe et al. 1998 [3]).A rolling horizon framework was implemented, which makes it possible to re-estimate the state of the network frequently, such that the quality of the prediction is

maximized. State-space modeling, as a classical technique for dynamic systems has also been used for extended OD estimation and prediction framework (Antoniou and Koutsopoulos, 2006 [2]). Also machine learning method such as neural network can also serve as an approach to reach our goal with quite expensive cost (Min Zhang et al. 1999 [11]). However, these methods can predict the amount of traffic in the next period of time, but they barely provide any information on how close their predictions are to the real traffic demand. To tell that, a technique providing predictions with confidence which can reflect how reliable the predictions are.(Dashevskiy and Luo, 2008 [7]). Conformal Predictors are introduced to guarantee that the probability of making erroneous predictions equals to a pre-defined significance level.

Because of the explosive developments in data collection technologies during the last few decades, the foundation of decision making has been transformed. Many decision-making protocols in road traffic management are no longer effective in the sense that they do not fully utilize the recently increased availability of traffic data. The problem has turned into how to select significant features and optimize with respect to all available information.

# Chapter 3

# Model Description

Machine learning is a relatively new computing technology that uses artificial intelligence tools to develop systems that learn from data, rather than simply performing programmed instructions. Machine learning is now widely used by researchers and industry analysts to build predictive models from a wide variety of data. As models are fed new data, they are able to independently adapt. They can learn from historical patterns and computations to produce reliable predictions and results. In this chapter we introduce three techniques that we will use to build our traffic demand prediction model.

## 3.1 Linear Regression

As one of the most basic and commonly used type of predictive analysis, linear regression is an approach to build relationship between a set of explanatory variables and a dependent variable from data.[13] It will find the straight line that best fits the plotted data. The

general equation for linear regression is

$$y_i = x_i^T \beta + \epsilon_i, \qquad i = 1, ..., n,$$

where $x_i^T \beta$ is the inner product between explanatory vector $x_i$ and parameters $\beta$.

The model will be trained to minimize the squared residuals, which can be calculated as

$$\sum_{i=1}^{n} (X_i^T \beta - Y_i)^2$$

The parameters of the model can be estimated using the formula:

$$\beta = (X^T X)^{-1} X^T y$$

## 3.2 Support Vector Machine

A Support Vector Machine (SVM) by definition is a discriminative classifier formally defined by a separating hyperplane. An SVM will give a score:

$$f(x_i, W, b) = W x_i + b$$

in which $x_i$ is the data vector, $W$ and $b$ are model's parameters. It is calculated from the raw data to identify its class. SVM is evaluated through a loss function:

$$L_i = \sum_{j \neq y_i} max(0, w_j^t x_i - w_{y_i}^T x_i + \Delta) + \lambda \sum_k \sum_l W_{k,l}^2$$

in which $\Delta$ is a fixed margin and $R(W) = \lambda \sum_k \sum_l W_{k,l}^2$ is a regularization penalty. It tells how close the predicted scores to the true class is. Thus the process of training SVM model can be considered as an optimization problem, in which we tuning its parameters to minimize the loss function.

SVM has a regularization parameter, which helps us avoid over-fitting. Also the training process of SVM is defined as a convex optimization problem, so that we can easily use methods such as Gradient decent. What's more, SVM comes with a kernel function by which we can use our domain knowledge. However to overcome the problem of over-fitting, SVM is more about model selection than finding the best parameters.

To use SVM in regression, slack variables $\xi$, $\xi^\star$are introduced to deal with infeasible constraints.[14] Slack variables allow regression error to exist up to the value. The primal function of SVM regression can be written as

$$j(W) = \frac{1}{2} W'W + C \sum_{n=1}^{N} (\xi_n + \xi_n^*)$$

Subject to:

$$\forall n : y_n - (x_n' \beta + b) \leq \epsilon + \xi_n$$

$$\forall n : (x_n' \beta + b) - y_n \leq \epsilon + \xi_n$$

$$\forall n : \xi_n^\star \geq 0$$

$$\forall n : \xi_n \geq 0$$

## 3.3 Extreme Gradient Boosting Tree

Quite different from the previous two models, a decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences. Using it as a predictive model is one of the most popular and practical method for inductive inference.

To train the model, we will use Xgboost, which stands for extreme gradient boosting.[5] This is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It used a more regularized model formalization to control over-fitting, and gives it better performance. It pushes the limit of computations resources for boosted tree algorithms.

Prediction trees can be summed up to get better representation. A tree ensemble can be written in the form of

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F$$

where K is the number of trees, f is a function and F is the set of possible trees.

# Chapter 4

# Data Handling

We have three groups of data. The first group is the order group. Each of the orders has an order ID, driver ID, user ID, departure location, destination location, price and time. The second group includes district level information, like POI class, traffic level etc. The last group shows the weather information every 10 minutes for each district.

First, we explored the information related to the second group of information – the district information for each order. The city is divided into 66 non-overlapping square districts $D = \{d_1, d_2, ..., d_{66}\}$. After calculating the frequency of each district, which is plotted in figure 4.1, it shows that the number of orders in districts is not quite evenly distributed. 15% of the orders took place in district 51, 80% of the orders took place in the downtown area and rest of the orders are distributed evenly in the rest of the 45 districts.

Then we divide each day uniformly into 144 slots with a length of 10 minutes. In district $d_i$ and time slot $t_j$, the demand $demand_{ij}$ is defined as the number of passengers' requests, and the supply $supply_{ij}$ is defined as the drivers' answers. The gap can be represented as: $gap_{ij} = demand_{ij} - supply_{ij}$. We're trying to predict $gap_{ij}$ given the historical data of every district. Most of the orders are made from 7am to 9pm, and there are 2 peaks in the morning

and evening rush hours.

Interestingly, as shown in figure 4.2, the frequency of gaps are monotonically decreasing with the increase of gap length. 30% of the time all user demands are satisfied.

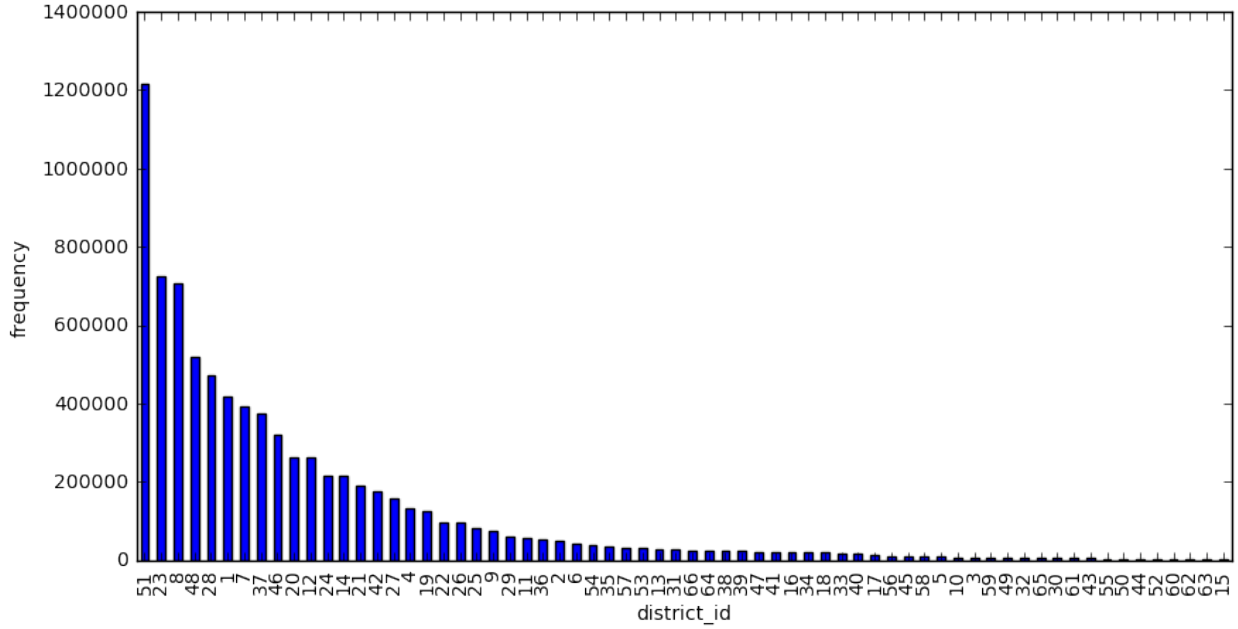



Figure 4.1: Frequency of Start District

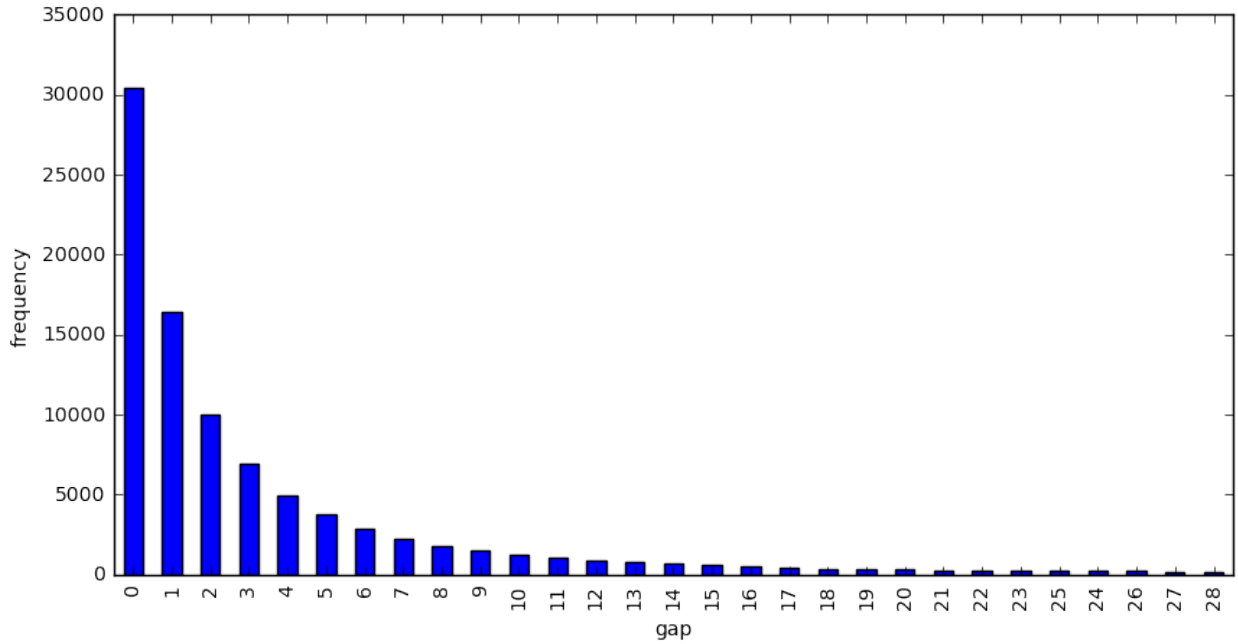



Figure 4.2: Frequency of Gaps

There are multiple ways to build the samples from the data. We can train the model, using the data several time slots ahead as the input to predict the target time slot's gap. Physically it means that we're predicting according to changing tendency but not historical record. The reason we choose this approach is that the data space is relatively small, and there's no direct proof that using previous day or week's data will boost the performance of our model. In other words, we can make the conclusion that in this data set the main factor affecting the prediction of gap is the related data in three time slots ahead.

Since we have 144 time slots everyday and 21 days, we have 3024 samples for each district. We used the first two week's data as training set and last week's data as our validate set.

For all categorical information, one hot encoding (a process by which categorical variables are converted into a form that can be provided to ML algorithms to do a better job in prediction) should be used for indicating the state.

There are lot of strategies to do feature engineering. We can use Cross Validation (CV) onto different permutation of features, testing out which one has the greatest influence on the data. CV involves partitioning a sample of a data into complementary subsets, training some subsets and validating the performance on the other sets. The data set is split into 10 folds and predicts gap for each part using the other parts to train with. Multiple rounds of CV should be performed onto the model to avoid overfitting problem.[12]

Also, XGBoost offers calculation of importance, which can be used as a reference on the data selection. The features of top 30 importance are listed in figure 4.3. We use 190 dimensions of features and some subsets of the features to build a hybrid model.

In chapter 5, to determine the hyperparameters of the model, CV will also be used for help. Using CV onto different hyperparameter combinations is called Grid Search.

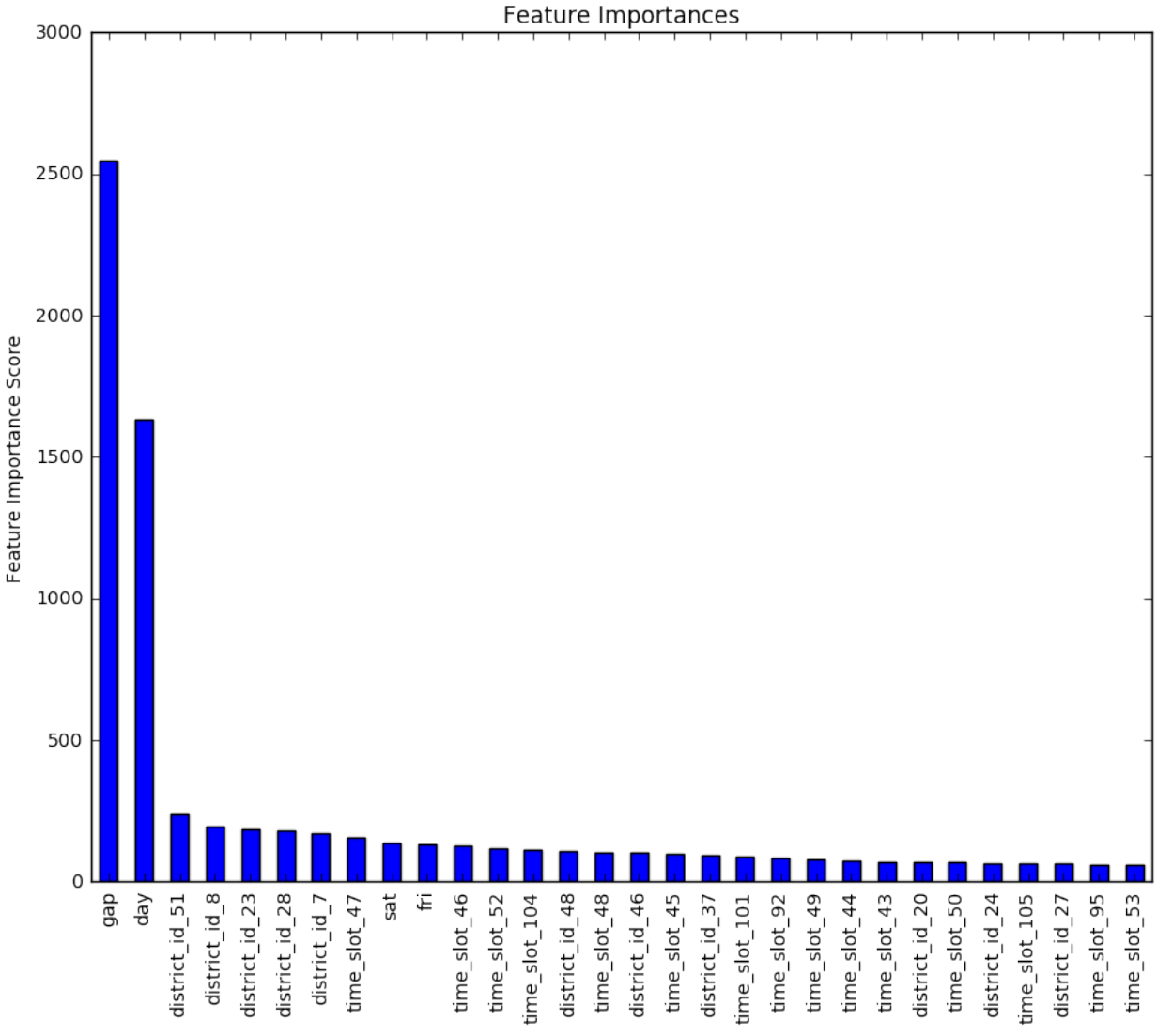



Figure 4.3: Top 30 importance of features

# Chapter 5

# Experiments

This chapter provides details on how we conducted experiments on the data set of the three different models.

## 5.1 Support Vector Machine

When handling huge amount of data, Support Vector Machine has the problem of slow training speed and enormous memory usage. Kernelized SVMs require the computation of a distance function between each point in the dataset, which is the dominating cost of $O(n_{features} \times n^2_{observations})$. Also, the storage of such number of distance is a huge burden on memory, so that most of the time we recompute the distance when we need them. Hence the running time can go up to $O(n_{features} \times n^2_{observations})$. Hence to solve problem with large-scale data set, methods like working set selection, divide and conquer strategy etc.

In this thesis, we use the idea of ensemble SVM[15] to train 30 different SVM models with bootstrapped subsets of the data set. Though it might seem like we're increasing the complexity of the algorithm, the complexity is actually reduced. Assume we divide the data into

p subsets, then the size of each set is $n/p$, the complexity of training each sub SVM model is $O((n/p)^2)$. The total complexity can be calculated as:

$$\sum_{i=1}^{p} O((n/p)^2) = O(n^2/p)$$

which is stepped-down by p times.

After the submodels are trained, we take average of the predictions as a result. This method greatly reduced the training time without sacrifice of accuracy. The whole architecture of this model is shown in the figure below .

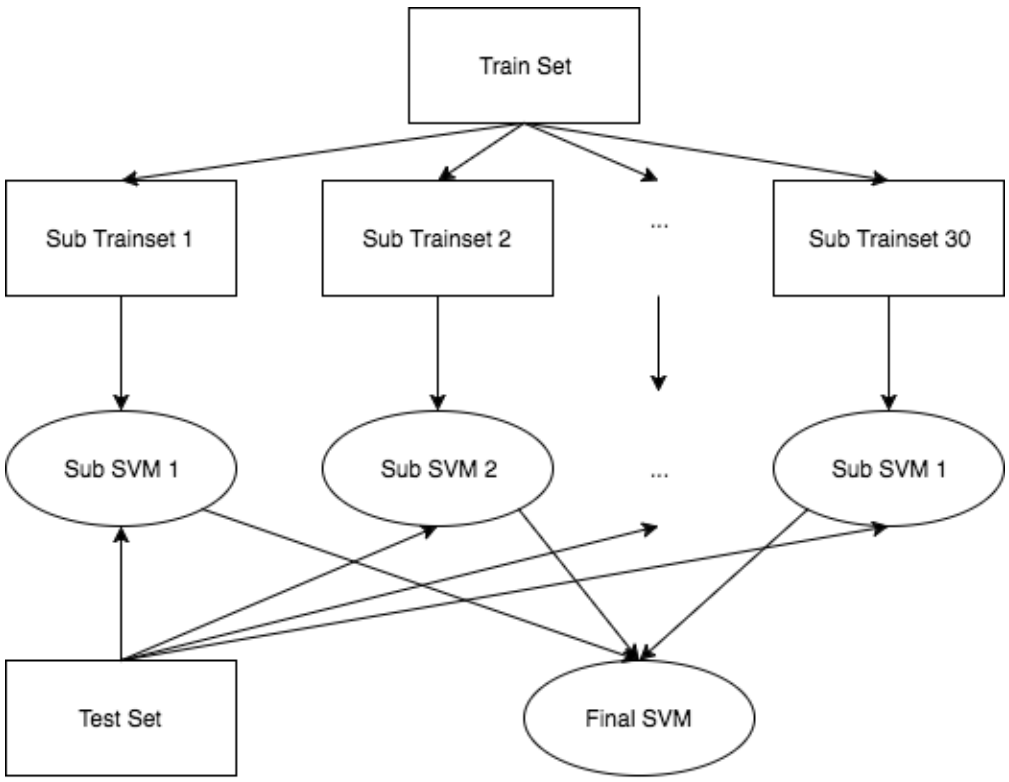



Figure 5.1: Structure of SVM

## 5.2 Improving single XGB model

Hyper-parameters are parameters those will not be learnt within estimators. To build a well-performed XGB model, we used the technique of grid search and cross validation to find out the best hyper-parameters. There are three types of parameters in an XGBoost model according [5]. General parameters define the boosting model, booster parameters are the detail parameters of the chosen model and learning task parameters guide the optimization performed. In this thesis we consider only the tree booster.

To determine the best boosting parameters, we need to set some initial values of other parameters. We started with the following values:

| | |
|---|---|
| max_depth | 3 |
| learning_rate | 0.1 |
| gamma | 0 |
| min_child_weight | 1 |
| subsample | 1 |
| colsample_bytree | 1 |
| max_depth | 3 |
| reg_alpha | 0 |
| reg_lambda | 1 |
| min_child_weight | 1 |
| scale_pos_weight | 1 |
| base_score | 0.5 |

Table 5.1: Single XGBoost initial parameters

We tuned the max_depth and min_child_weight first as they will have the major impact on model performance. Wider ranges were set first and then we performed iterations for smaller ranges. For the other parameters we followed very similar steps. A matrix of two grids was built in the form of:

As the model performance increases, it becomes much more difficult to achieve even marginal gains in performance.

| | |
|---|---|
| max_depth | [3, 5, 7, 9] |
| min_child_weight | [1, 3, 5] |

Table 5.2: Grid Search Matrix of max_depth and min_child_weight

By doing these iterations on different combinations of parameters, we obtained the best set. We show these in table 5.1:

| | |
|---|---|
| max_depth | 8 |
| learning_rate | 0.1 |
| gamma | 0.2 |
| min_child_weight | 2 |
| subsample | 0.7 |
| colsample_bytree | 0.85 |
| reg_alpha | 100 |
| scale_pos_weight | 1 |

Table 5.3: Single XGBoost final parameters

A significant boost in performance can be observed. However, it is difficult to get a very big improvement by just tuning hyperparameters. To achieve a significant boost, we have to build an ensemble model.

## 5.3 Bagging XGBoost

Averaging is very helpful for a wide range of problems, including both classification and regression. Also it works well on usual metrics, for example accuracy or mean square error. Bootstrap aggregating, also known as bagging, is a very frequently used machine learning ensemble meta-algorithm, and a special case of the model averaging approach. It is designed to the stability and accuracy of machine learning models and reduce overfitting.[9]

Inspired by research on bagging, we allowed these parameters to fluctuate randomly within a certain small range while randomly sampling the characteristics and training multiple

xgboost sub-models for averaging. 30 sub-models are trained using a subset of the training set and the parameters will be randomly generated randomly in a small range around the parameters of the best single xgboost model. The details of the parameters is listed in the following table:

| | |
|---|---|
| max_depth | 7 9 |
| gamma | 0 0.5 |
| min_child_weight | 1.5 2.5 |
| subsample | 0.65 0.75 |
| colsample_bytree | 0.8 0.9 |
| min_child_weight | 1.5 2.5 |

Table 5.4: Bagging XGBoost parameters

After we acquired the models, we calculate the average of their predictions. Bagging brought diversity into our model, which will generalize our model well to new unseen data. The graph below illustrate the basic principle of bagging being used in our model.

We took 50 samples from the training set to illustrate how bagging works. The gray lines are very wiggly and show that the model is overfitting the data. By taking the average of 30 sampled parameter values each is fitted to a subset of the entire data set. The new prediction is more stable and has less overfitting.

## 5.4 Multi-model blending

Model ensembling is a very powerful technique to improve the performance of ML tasks. In this thesis we use blending ensemble to build a hybrid model based on Random Forest, Extra Trees and AdaBoost.

The blending process is actually divide the training set into smaller training sets $train-train$ and validating sets $train - valid$ during cross validation. Linear regression has been chosen

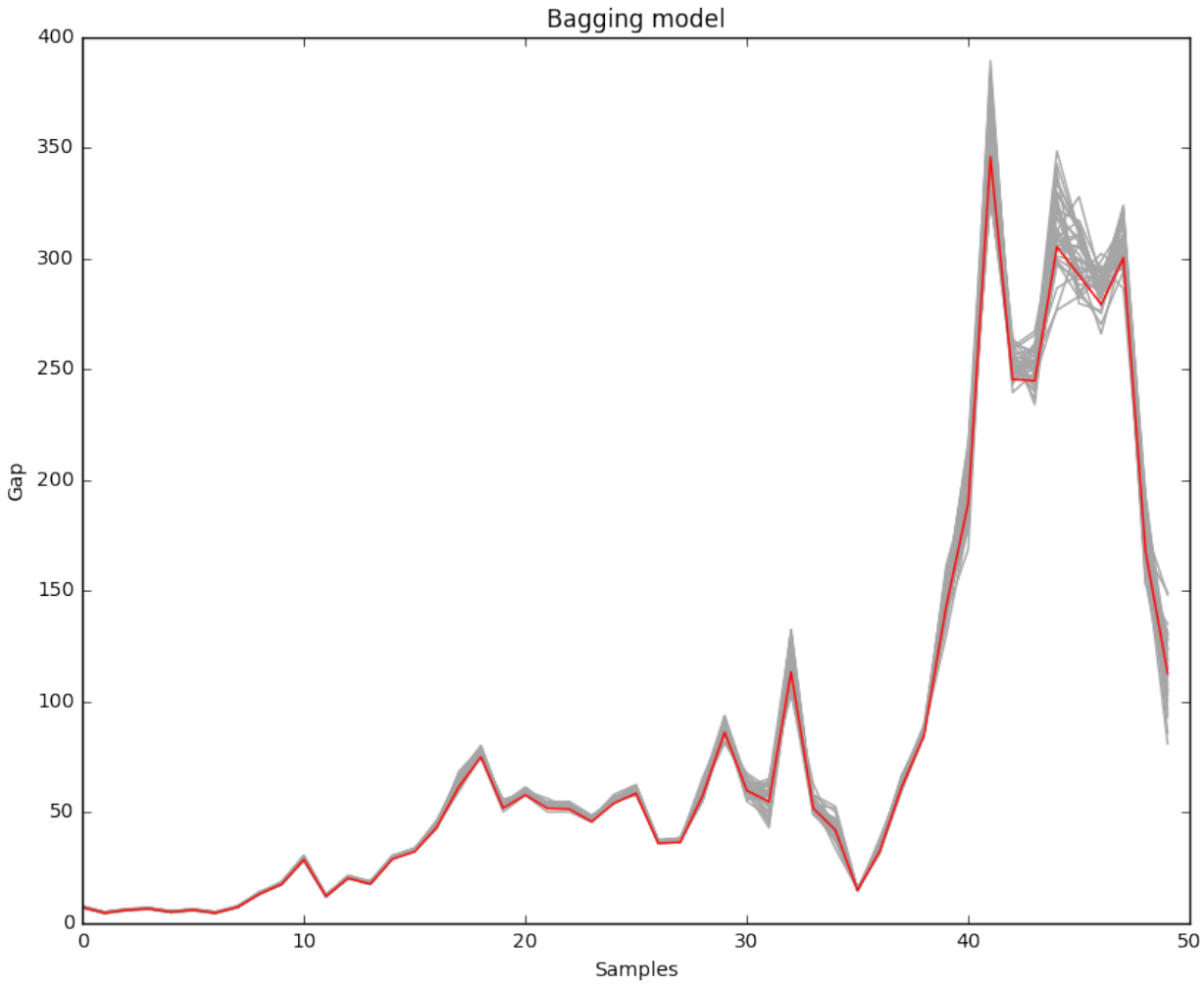



Figure 5.2: Bagging XGBoost

to be the upper-level model, the whole structure are as follows in figure 5.3.

Random forest and extra trees are very similar models. Both of are estimators that fit a number of decision trees. When choosing variables at a split, extra trees draw samples from the entire training set instead of a bootstrap sample of the training set. Also the splits are chosen completely at random from the range of values in the sample at each split. Random forests are generally more compact than extra trees in practice, but they are also more expensive to train.

AdaBoost is a meta-estimator that begins by fitting from original data set, and then Ad-

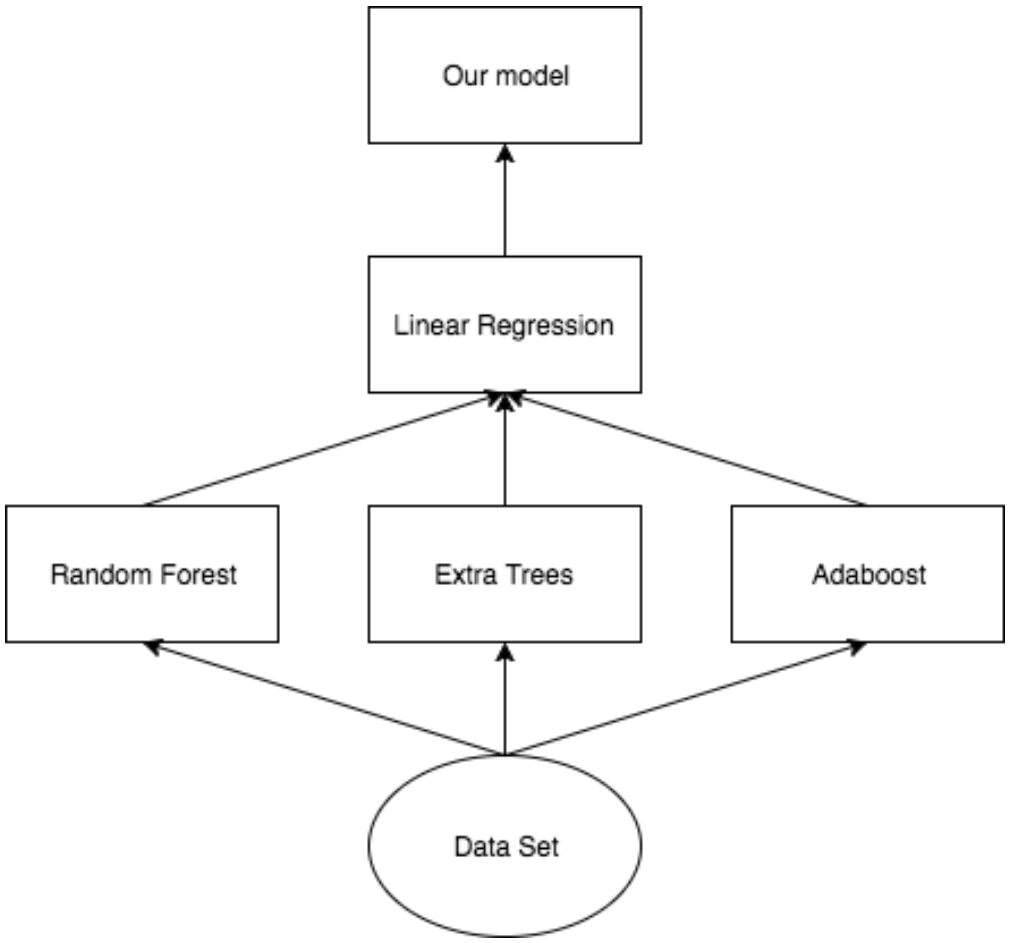



Figure 5.3: Structure of Blending Emsemble Model

aBoost training process selects only those features known to improve the predictive power of the model. Unlike other models, AdaBoost can achieve similar results with much less tweaking of parameters or settings.

The blending process in our model was according to the following steps:

1. Split the training set into 5 sub train and test sets, named $train - train_i$ and $train - valid_i$.

2. Train and cross-validate 3 models from each of the subsets. Then use the submodels to predict on $train - train_i$, get $train - pred_{ij}$

3. For each model j, merge $train - pred_{ij}$ into one prediction set $train - pred_j$.

4. Use $train - pred_j$ as input and the true label of training set as output to train a linear regression model as our final ensemble model.

## 5.5 Model ensemble

Once we acquired all three models, we build final ensemble model based on them. The structure is shown in figure 5.3:

To take the benefits of having multiple models, we build the ensemble by taking their weighed averages. To get a good result from ensemble, diversity is highly required. To observe the difference of different single models, we calculate the Maximum Information Constant (MIC) between the models, and visualize them in matrix (brighter colors means less correlation).

From figure 5.4, we show that the correlation of MIC between XGBoost and SVM, XGBoost and blending, SVM and blending are relatively lighter. These combinations of models show great diversity and so that we ensemble these models by taking weighed average.

In summary, as shown in figure 5.6, we implemented a multi-layer ensemble model framework from 4 different submodels:

1. SVM: Randomly divide the whole data set into 30 subsets such that the complexity will be decreased. We take the average of scores to be M1's prediction.

2. Bagging XGBoost: Generate 30 different XGB models by randomly adjusting parameter in a small range. We also take the average of scores to be M2's prediction.

3. Blending Model: Train different models on the same data set and use the parameters as a starting point, and then continue training with Logistic Regression to reach the weight of

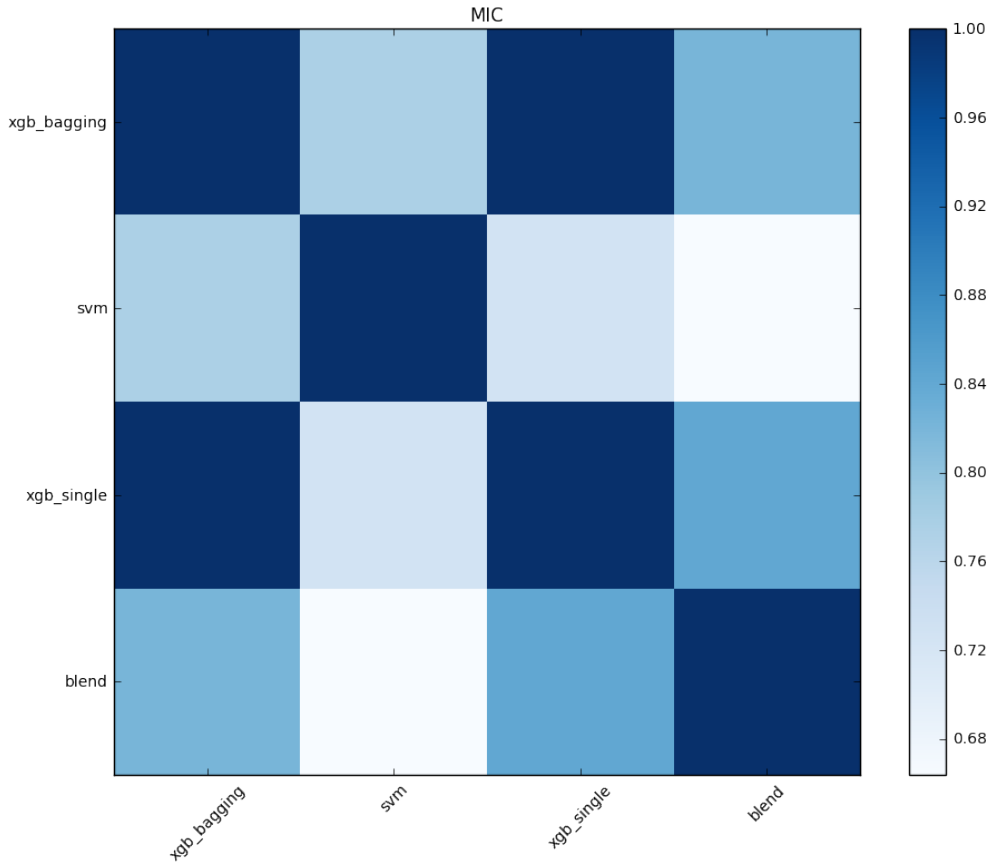
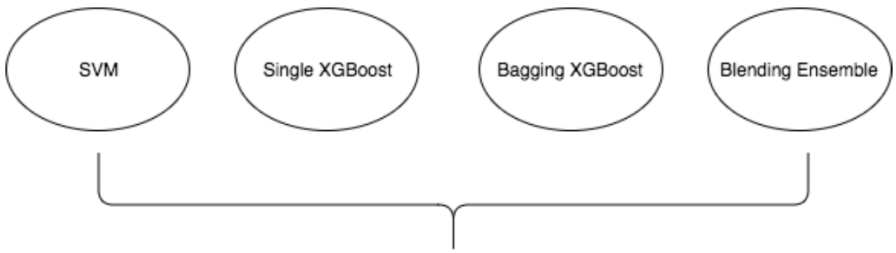



Figure 5.4: Maximum Information Constant

each model.

4. Sngle XGBoost: Single XGBoost model with best performance given by CV.

On the top level we can use averaging to get our final model. We chose rank averaging as figure 5.5, since it gives the best result.




$$score = \sum_{i=1}^{n} \frac{w_i}{rank_i}$$

Figure 5.5: Rank Average

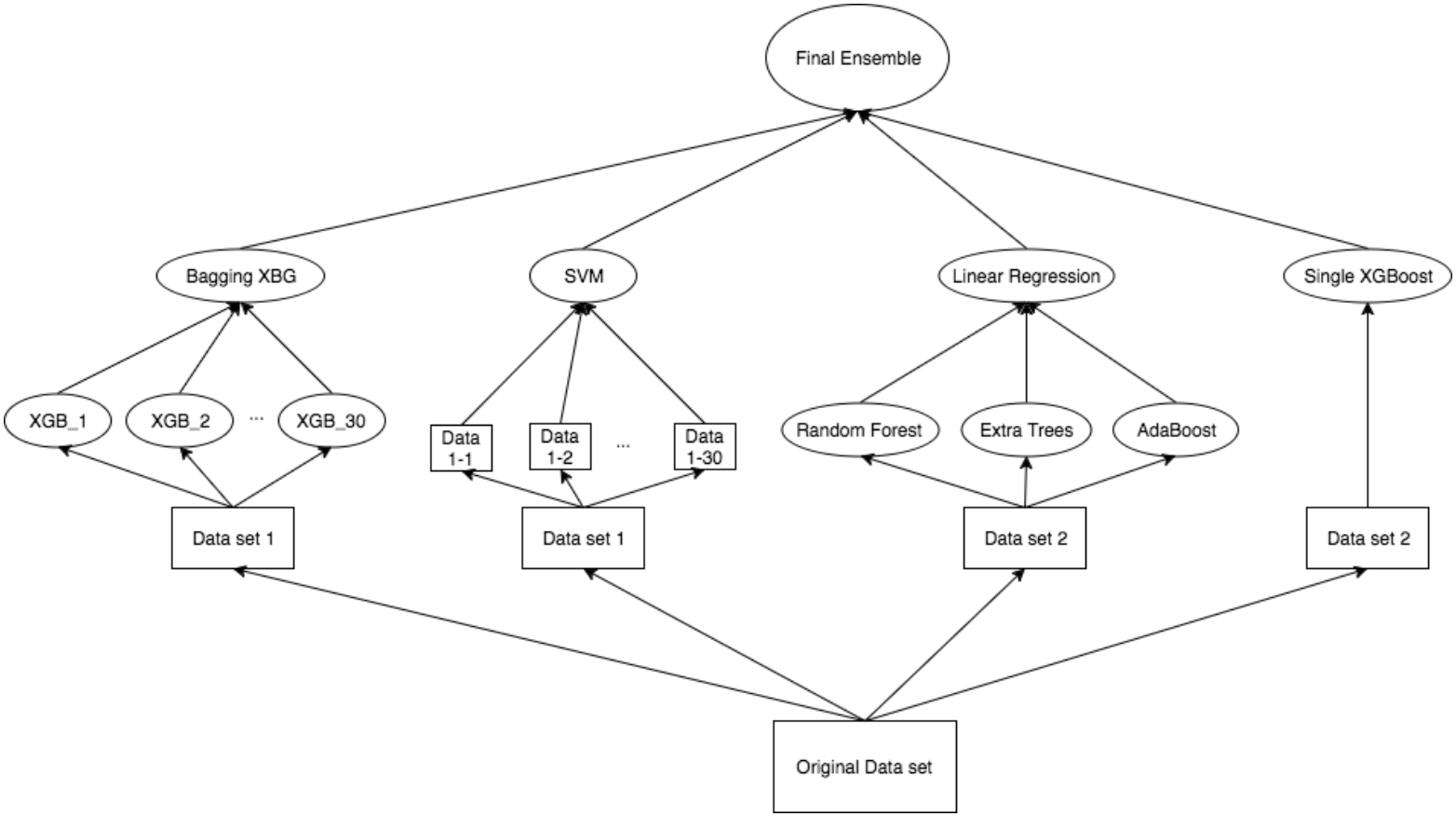



Figure 5.6: Structure of Final Model

# Chapter 6

# Result

The parameters and result of all models are listed in this Chapter.

## 6.1 Support Vector Machine

| | |
|---|---|
| Number of Submodels | 30 |
| C | 1 4 |
| gamma | 0.001 0.031 |
| min_child_weight | 1 |
| cache_size | 7000 |
| verbose | true |
| rbf | true |
| Mean Absolute Error | 7.9957 |
| Explained Variance Score | 0.3793 |

Table 6.1: SVM parameters

## 6.2 Single XGBoost

| | |
|---|---|
| max_depth | 8 |
| learning_rate | 0.1 |
| gamma | 0.2 |
| min_child_weight | 2 |
| subsample | 0.7 |
| colsample_bytree | 0.85 |
| reg_alpha | 100 |
| scale_pos_weight | 1 |
| base_score | 0.5 |
| Mean Absolute Error | 4.6319 |
| Explained Variance Score | 0.8881 |

Table 6.2: Single XGBoost parameters

## 6.3 Bagging XGBoost

| | |
|---|---|
| Number of Submodels | 30 |
| max_depth | 7 9 |
| gamma | 0 0.5 |
| min_child_weight | 1.5 2.5 |
| subsample | 0.65 0.75 |
| colsample_bytree | 0.8 0.9 |
| min_child_weight | 1.5 2.5 |
| Mean Absolute Error | 4.6019 |
| Explained Variance Score | 0.8856 |

Table 6.3: Bagging XGBoost parameters

## 6.4 Random Forest

| | |
|---|---|
| max_depth | 12 |
| max_features | 0.14357 |
| min_impurity_split | 1e-7 |
| min_samples_leaf | 3 |
| min_samples_split | 6 |
| n_estimators | 6 |
| Mean Absolute Error | 6.3750 |

Table 6.4: Random Forest parameters

## 6.5 Extra Trees

| | |
|---|---|
| max_depth | 20 |
| max_features | 0.4 |
| min_impurity_split | 1e-7 |
| min_samples_leaf | 2 |
| min_samples_split | 5 |
| Mean Absolute Error | 4.9709 |

Table 6.5: Extra Trees parameters

## 6.6 Adaboost

| | |
|---|---|
| n_estimators | 10 |
| Mean Absolute Error | 5.7845 |

Table 6.6: Adaboost parameters

## 6.7 Final Model

| | |
|---|---|
| Single XGBoost weight | 0.5 |
| Bagging XGBoost weight | 0.25 |
| Blending Ensemble | 0.16 |
| SVM | 0.09 |
| Mean Absolute Error | 4.6657 |
| Explained Variance Score | 0.8813 |

Table 6.7: Final Model parameters

# Chapter 7

# Conclusion

A traffic demand-supply prediction model has been developed that is based on historical record and traffic information. A combination of multiple machine learning algorithms has been used in the development of the model.

1. The best approach to build a prediction regressor right now can be categorized as Decision Trees and Neural Networks.

    - Among single models, XGBoost outperformed any other models. But there are still a lot of choices out there, such as multilayer perceptron, one-shot memory network, feedforward neural network etc.
    - The Decision Tree's structure, learning algorithms and parameters tuning has a great influence on the final result.
    - With respect to efficiency of compute time and memory resources, XGBoost has have some significant advantages over neural network.
    - The ensembles of different models brought diversity and made the predictions smoother, which reduced variance and helped to avoid overfitting.

2. Feature engineering is the foundation of models' performance.

    - Data visualization can greatly help feature extraction.
    - Categorizing samples according to their similarity on some features can greatly improve the performance of the model.
    - The key feature should be highly related to our target physically. In our case, it's the gap from the previous time slot.
    - Always use one hot encoding on categorical features.

3. There is still room to improve on the way we split training and testing set. A shifting time window can be used to increase the number of samples.

4. Once we can precisely predict the future demand-supply gap, we can build an algorithm to optimize the geographic and temporal distribution of drivers', so that there are always enough cars available for the riders in any zone during any time interval. According to Wardrop's Principles, we can design a system optimization heuristic such that most of the demand can be satisfied. We can start with Dial's multipath traffic assignment model (R. Dial, 1970 [8]), improve its complexity and result.

# Bibliography


[1] A. S. Alfa and D. L. Minh. A stochastic model for the temporal distribution of traffic demandthe peak hour problem. *Transportation Science*, 13(4):315–324, 1979.

[2] C. Antoniou and H. Koutsopoulos. Estimation of traffic dynamics models with machine-learning methods. *Transportation Research Record: Journal of the Transportation Research Board*, (1965):103–111, 2006.

[3] M. Ben-Akiva, M. Bierlaire, H. Koutsopoulos, and R. Mishalani. Dynamit: a simulation-based system for traffic prediction. In *DACCORD Short Term Forecasting Workshop*, pages 1–12, 1998.

[4] G.-L. Chang and H. S. Mahmassani. Travel time prediction and departure time adjustment behavior dynamics in a congested traffic system. *Transportation Research Part B: Methodological*, 22(3):217–232, 1988.

[5] T. Chen and T. He. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 2015.

[6] D. Chuxing. The di-tech challenge. *http://research.xiaojukeji.com/com petition/main.action?competitionId=DiTech2016*, 2016.

[7] M. Dashevskiy and Z. Luo. Network traffic demand prediction with confidence. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5. IEEE, 2008.

[8] R. B. Dial. A probabilistic multipath traffic assignment model which obviates path enumeration. *Transportation research*, 5(2):83–111, 1971.

[9] T. Hothorn and B. Lausen. Double-bagging: Combining classifiers by bootstrap aggregation. *Pattern Recognition*, 36(6):1303–1309, 2003.

[10] R. Jayakrishnan, H. S. Mahmassani, and T.-Y. Hu. An evaluation tool for advanced traffic information and management systems in urban networks. *Transportation Research Part C: Emerging Technologies*, 2(3):129–147, 1994.

[11] Z. Y. Y. H. R. Min. Study on transportation demand forecast using neural network model [j]. *Journal of Southwest Jiaotong University*, 5:022, 1999.

[12] A. W. Moore. Cross-validation for detecting and preventing overfitting. *School of Computer Science Carneigie Mellon University*, 2001.

[13] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.

[14] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[15] M. Tian, W. Zhang, and F. Liu. On-line ensemble svm for robust object tracking. *Computer Vision–ACCV 2007*, pages 355–364, 2007.