Palestine Polytechnic University
Deanship of Graduate Studies and Scientific Research
Master of informatics

Machine Learning Based Taxi Demand Prediction

Submitted by:

Samah Amer Al-Aloul

Thesis submitted in partial fulfillment of requirements of the degree Master of Science in Informatics
September, 2020

The undersigned hereby certify that they have read, examined and recommended to the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University the approval of a thesis entitled: **Machine learning based Taxi Demand Prediction**, submitted by **Samah Amer Al-Aloul** in partial fulfillment of the requirements for the degree of Master in Informatics.

Graduate Advisory Committee:

Dr. Hashem Tamimi (Supervisor), Palestine Polytechnic University.

Signature:_____ Date:_____

Dr. Mohammad Aldasht (Internal committee member), Palestine Polytechnic University.

Signature:_____ Date:_____

Dr. Wesam Herbawi(External committee member) Leam GmbH

Signature:_____ Date:_____

Thesis Approved

Dr. Murad Abu Subaih
Dean of Graduate Studies and Scientific Research
Palestine Polytechnic University

Signature:_____ Date:_____

DECLARATION

I declare that the Master Thesis entitled **"Machine learning based Taxi Demand Prediction"** is my original work, and hereby certify that unless stated, all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgement is made in the text.

Samah A. Al-Aloul

Signature:_____ Date:_____

STATEMENT OF

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for the master degree in Informatics at Palestine Polytechnic University, I agree that the library shall make it available to borrowers under rules of the library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from, reproduction, or publication of this thesis may be granted by my main supervisor, or in his absence, by the Dean of Graduate Studies and Scientific Research when, in the opinion of either, the proposed use of the material is for scholarly purposes.

Any coping or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Samah A. Al-Aloul

Signature:_____ Date:_____

الملخص

تشكل حركة المرور النبض لاي مدينة، حيث تؤثر على الحياة الاجتماعية والاقتصادية والصحية لملايين من البشر. يلعب التنبؤ بالطلب، وخاصة التنبؤ بطلب على سيارات الأجرة، دورًا هاماً في أنظمة النقل الذكية في المدن الذكية. مشكلة التنبؤ بطلب على سيارات الأجرة هي عملية التنبؤ بعدد طلبات سيارات الأجرة في منطقة معينة خلال فترة زمنية معينة باستخدام معلومات حول طلبات سيارات الأجرة السابقة. في هذا البحث، استخدمنا خوارزميات التعلم الآلة مثل random forest regression RFR, support vector regression SVR, artificial neural network ANN للتنبؤ بطلب على سيارات الأجرة بدقة. هدفنا هو تحسين دقة التنبؤ بطلب على سيارات الأجرة باستخدام طرق مختلفة لعملية التنبؤ. لذلك، قمنا بدراسة تأثير الموقع على عميلة التنبؤ بطلب على سيارات الأجرة. حيث اعتبرنا ان الموقع الذي تم فيه الطلب على سيارة الاجرة بمثابة: three decimal rounded coordinates و neighborhood و cluser .

استخدمنا في هذا البحث البيانات الخاصة بطلبات سيارات الأجرة الصفراء في مدينة نيويورك. و قمنا بدعم بياناتنا بإضافة المعلومات الخاصة بالطقس. وجدنا أن استخدام معلومات الموقع بطرق مختلفة كان له تأثير واضح على نتائج التنبؤ وأداء الخوارزميات. افضل اداء للخوارزميات كان عند اعتبارنا للموقع كأحد الاحياء في مدينة نيويورك، كما هو الحال في الشبكات العصبونية الذكية NN . تم ملاحظة

أهمية معلومات الطقس عند استخدام احداثيات الموقع three decimal rounded coordinates . و بشكل عام كان اداء الشبكات العصبونية الذكية هو الأفضل أداء تقريبًا.

Abstract

Traffic is the pulse of any city that impacts the social, economic and healthy life of millions of people. Demand prediction, mainly taxi demand prediction, plays an important role in intelligent transportation systems in smart cities. The taxi demand prediction problem is to precisely predict the number of taxi requests for a certain region at a certain time-bin based on the information provided by previous taxi requests.

In this research, we used machine learning methods, such as random forest regression RFR, support vector regression SVR, and artificial neural network ANN, to accurately predict taxi demand. Our goal is to improve the accuracy of taxi demand prediction by using different ways of prediction. Therefore, we studied the effect of the location on taxi demand prediction. We consider the location of the pickup as a cluster, three decimal rounded coordinates or neighborhood.

The research uses yellow-taxi requests in New York City NYC as a dataset. We refined our data by adding weather information. We found that location features affect the prediction result and the performance of the model. NYC neighborhoods as a location feature are the best in the model's per-

formance, as in the NN model. We noticed the importance of the weather features with the three decimal rounded location coordinates. The NN model approximately gives the best performance.

DEDICATION

To Shadi,

my great husband,

who stood beside me throughout the years.

His beneficial care for me and our daughters made it possible to complete this search.

To my angels Yasmin and Laya,

they are indeed a treasure from Allah.

To my parents,

I could never have done this without your faith, support, and constant encouragement.

ACKNOWLEDGEMENT

I am grateful to the thesis's supervisor, Dr. Hashem Tamimi, for his great guidance and immense knowledge. I am thankful to Dr. Zain Salah and Dr. Alaa Halawani for their support and useful comments. Also, I would like to express my sincere gratitude to the committee members, Dr. Mohammad Aldasht and Dr. Wesam Herbawi. Moreover, I would like to thank all members of the university computer center for their help.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
SVM	Support Vector Machine
SVR	Support Vector Regression
ANN	Artificial Neural Network
RFR	Random Forest Regression
Sklearn	Scikit-learn
LR	Learning rate

Chapter 1

Introduction

Intelligent transportation systems are the future of smart cities. Taxi demand prediction is an important issue in traffic and transportation systems. The problem of taxi demand prediction is to predict the number of pickups in different locations at different times. Several studies focused on utilizing big data to get accurate predictions to help drivers to get to passengers on time. Nowadays, we can get a huge demand data history based on the popularity of taxi requesting services and applications, such as Uber and Left. Several approaches can be used to deal with this problem. This paper will use machine learning approaches. This chapter introduces the problem and the suggested solutions.

1.1 Problem Statement

The taxi demand prediction problem is to predict the number of taxi requests for a certain region at a certain time-bin by using historical taxi requesting data. We consider the taxi demand prediction problem as a regression problem. As shown in Figure 1.1, given the **inputs**: time, region, and others such as temperature, passenger count, etc. Then, the **output** is the pre-

dicted number of pickups. We focus on using historical taxi pickup data and weather data of NYC to train and test our models.

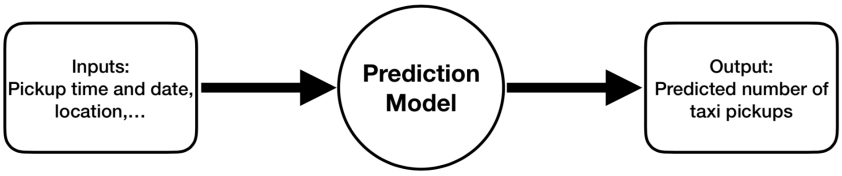


Figure 1.1: Taxi demand prediction problem.

Taxi demand looks like random behavior. Therefore, we need a model that can capture the relation between the features and the target value.

1.2 Thesis objective and contribution

In this research, we edit the taxi data set to be usable with machine learning methods. We will refine it by adding the weather information for each day. Our goal is to determine the relationship between the problem features and the taxi demand to predict the number of pickups in a location at a time-bin. We apply different machine learning models, such as RFR, SVR and ANN. We train each method on three different considerations of the pickup location. We introduce in detail the effect of the pickup location features on the taxi demand prediction model performance.

1.3 Thesis Organization

The thesis organized as follows: Chapter 2 presents overviews of the background of the research preliminaries. Section 2.1 presents the definition of the regression problem. The methods of machine learning are introduced in section 2.2. Chapter 3 presents the literature review about taxi demand

prediction. Chapter 4 describes the dataset, the data pre-processing, data visualization, the creation of the workspace and general description of the proposed methods. Chapter 5 contains the details about the experiment methodology for taxi demand prediction and results analysis. Finally, we conclude the research and the future work in Chapter 6.

Chapter 2

Background

2.1 Regression Definition

Regression uses in understanding the relationship between variables. The regression problem is to find the function used to predict future values. The regression can be simple or multiple linear. In simple linear regression, we have a single independent variable to predict the dependent variable. However, in multiple linear regression, we use more than one independent variable to predict the dependent variable. The general form of regression types are: Simple linear regression

$$Y = \alpha + \beta X \tag{2.1}$$

Multiple linear regression

$$Y = \alpha + \sum_{i=1}^{n} \beta_i X_i \tag{2.2}$$

where Y is the target (dependent variable), X is the feature (independent variable), α is the intercept with x-axis, β is the slope of the line and n is

the number of independents. The machine learning field uses to solve non-linear regression problems. We have types of regression methods in ML; For example, Support Vector Regression, Random Forest Regression, etc. This chapter discusses some of the types of regressions.

2.2 Machine Learning

Machine learning ML [19] is an AI science. It gives agents the ability to learn without being explicitly programming. ML enables computer programs to learn themselves, to grow or change when they get to new data. Machine learning deals with decision problems with sensitivity, noise, or missing data values. Machine learning split into two main fields, supervised learning: the model trains of an input-output dataset (labeled data), to estimate new output for new input. Unsupervised learning: the model trains on a dataset with no pre-existing labels. The goal is to understand the dataset, create some description of boundaries, categorize, which are initially unknown. In our problem, we focus on supervised learning methods. Taxi demand prediction is considered as a regression problem. The dataset contains the features of the correct number of pickups. This chapter reviews the ML methods which are used in the prediction of taxi demand.

2.2.1 Support Vector Regression SVR

Support Vector Machine SVM uses to map the input space data to a higher feature dimensional space by using the function Φ. Then it creates a separating hyperplane in the feature space. SVM model train on a set of data $x_i \subset R^n$ and class labels $y_i = -1, +1$, where $i = 1, 2, \ldots, n$, where n is the size of training data. SVM train to find the direction w and b of the

hyperplane such that $f(x) = w.\Phi(x) + b > 0$ for positive examples and $f(x) = w.\Phi(x) + b \leq 0$ for negative examples. Therefore, it is easy to find an optimal hyperplane that can distinguish between two types of datasets. The basic idea of SVR such as SVM, but the hyperplane is to do linear regression in the feature space [18]. The regression problem is to find a function that can estimate future values accurately based on previous values. The generic SVR forecasting function takes the form

$$f(x) = w.\Phi(x) + b \qquad \Phi : R^n \to \mathcal{F}, \quad w \in \mathcal{F} \tag{2.3}$$

Where $\mathcal{F}$ is the feature space and b is a threshold. Linear regression in a high dimensional space corresponds to a non-linear regression in the low dimensional space (i.e. map non-linear input space to linear in feature space). Consequently, the goal is to find the optimal value of w and b by minimizing the regression risk

$$R(f) = C \sum_{i=1}^{n} \Gamma(f(x_i) - y_i) + \frac{1}{2}||w||^2 \tag{2.4}$$

where $\Gamma(.)$ is a cost function, C is a constant determines penalties to estimation errors. and the vector w can be written in terms of data points as

$$w = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)\Phi(x_i) \tag{2.5}$$

where the pairs (α_i, α_i^*) are Lagrange multipliers [10], Lagrange is a mathematical method uses to find the local maximum and minimum of a function subject to equality constraints. These multipliers represent the solutions that act as forces pushing predictions toward target value y_i.

Substitute Equation (2.5) into Equation (2.3), we get

$$\begin{aligned} f(x) &= \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)(\Phi(x_i).\Phi(x)) + b \\ &= \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)k(x_i, x) + b \end{aligned} \tag{2.6}$$

$k(x_i, x)$ in Equation (2.6) is the kernel function. By using Kernel functions, the dot product can performe low-dimensional space data input to high-dimensional feature space without knowing the transformation Φ. Tabel 2.1 shows some popular kernel functions. Now, The cost function $\Gamma(.)$. Typically

Table 2.1: Popular Kernel function

Kernel	Function
Linear	$x.y$
Polynomial	$(x.x_i + 1)^d$
Radial Basis Function RBF	$exp(-\gamma\|x - x_i\|^2)$

the ε-insensitive loss function is used, and it is in the form

$$\Gamma(f(x) - y) = \begin{cases} |f(x) - y| - \varepsilon, & \text{for } |f(x) - y| \geq \varepsilon \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

The risk in Equation 2.4 and loss function in Equation 2.7, by solving the quadratic optimization problem, they can be minimized

$$\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)k(x_i, x_j) - \sum_{i=1}^{n}\alpha^*(y_i - \varepsilon) - \alpha(y_i + \varepsilon)$$

$$\sum_{i=1}^{n}(\alpha^* - \alpha) = 0, \qquad \text{where } \alpha^*, \alpha \in [0, C]. \tag{2.8}$$

Important notice, only the nonzero values of the Lagrange multipliers are useful in forecasting the regression line. These values are known as support

vectors. Figure 2.1 shows the parameters of SVR.

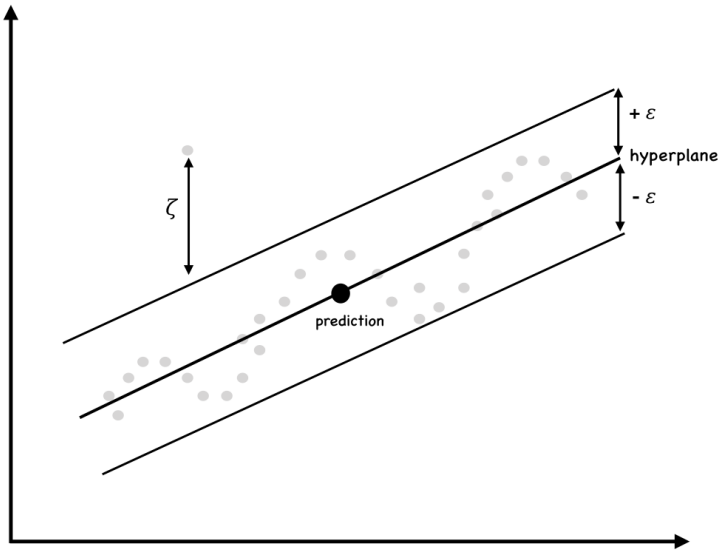


Figure 2.1: The parameters of SVR. SVR fit a tube with radius ε, and ζ_i measuring the points lying outside of the tube. The point on the hyperplane is the prediction value for an observation value.

As for parameter C in Equation 2.4, when C is large, there is more emphasis placed on the error, which means the regression has few errors but a lower generalization. While if C is small, less emphasis is placed on the error, which leads to a better generalization. If C goes to zero, the SVR model will have a large number of errors. Whereas the C value goes to infinity, the SVR model will have no errors but a very complex model.

Now, how to find the value of b?. Karush–Kuhn–Tucker (KKT) conditions is used to choose the proper value of b. "The final KKT condition states that the product of the Lagrange multipliers and the constraints is equal to zero" (the complete derivation can found in [3]), this implies to

$$\begin{aligned} \alpha_i(\varepsilon + \zeta - y_i + (w, x_i) + b) = 0 \\ \alpha_i^*(\varepsilon + \zeta^* - y_i - (w, x_i) - b) = 0 \end{aligned} \tag{2.9}$$

and

$$\begin{aligned}(C-\alpha_i)\zeta_i &= 0\\ (C-\alpha_i^*)\zeta_i^* &= 0\end{aligned} \tag{2.10}$$

ζ and ζ^* are slack variables, which are variables that added to an inequality constraint to transform it into equality. Solve Equation 2.9 with respect to b to obtain

$$\begin{aligned}b &= y_i - (w, x_i) - \varepsilon \quad \text{for } \alpha_i \in (0, C)\\ b &= y_i - (w, x_i) + \varepsilon \quad \text{for } \alpha_i^* \in (0, C)\end{aligned} \tag{2.11}$$

Consequently, now we can use SVM and SVR without knowing the transformation.

2.3 Artificial Neural Network

The artificial neural network ANN modeled on the biological neural network (human brain functioning). Such as in the biological neural network, the ANN has three components: 1) Node character, which controls the signals, such as the number of inputs and outputs, the weights, and the activation function. 2) Network topology and how to organize and connect the nodes. 3) Learning rules, use to initialize and adjust the weights. Figure 2.2 shows a comparison between a human neuron and an ANN neuron. This section reviews these part of ANN.

Node Character

Figure 2.2(B) shows a basic model for a node in the ANN. A basic model of a single node contains: input x_i , weight w_i, transfer function f and output y. Each node receives inputs from other nodes via connections that have

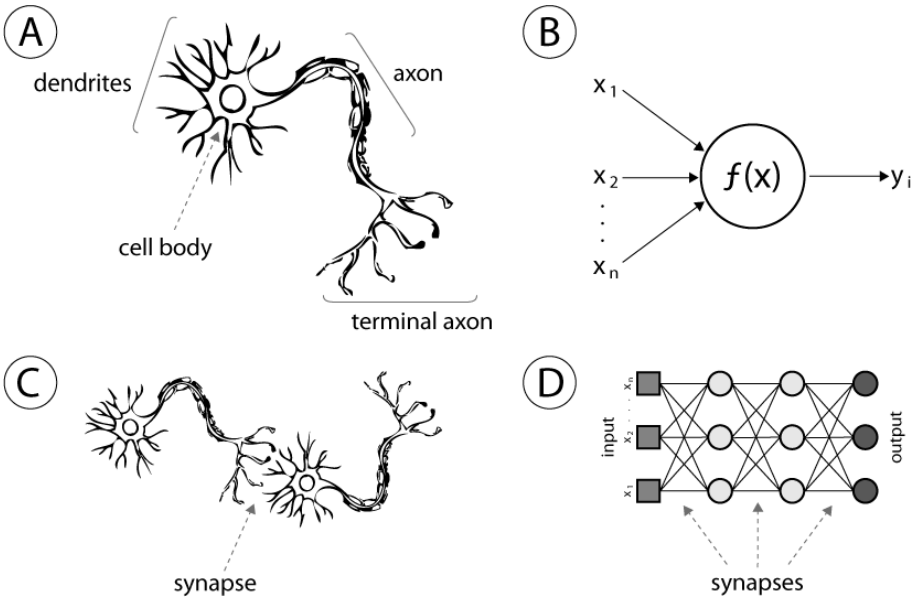


Figure 2.2: (A) Human neuron (B) Artificial neuron (C) Biological synapse (D) ANN synapses. (reprinted from Maltarollo et. al. 2013 [16]).

associated weights, these connections such as the strength of the synapse. The weights are summed then activate and pass the signal through a transfer function and sends it to neighboring nodes. This process can be formed as a mathematical model as

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right), \qquad b \text{ is a Bias.} \tag{2.12}$$

Where f is the transfer function (also, known as activate function). Figure 2.3 shows some of activation functions.

Network Topology

Network topology is how the nodes organized and connected. The network has three layers: the input layer, hidden layer, and output layer. Also, There can be no hidden layers (only the main layers: input and output layers). Designing the topology of the network is initially set by intuition and exper-

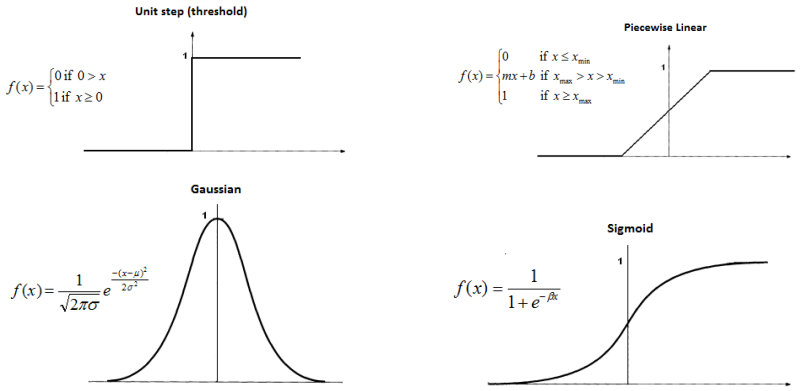


Figure 2.3: Different types of activation functions.

iment. We are doing experiments to obtain the best number of nodes in each layer, the number of layers in the network, and the path of connection between the nodes. The connections among nodes can be a one-way connection, or with some loop-back connections. Consequently, we have two categories of ANN: feedforward network and feedback network (see Figure 2.4). The feedforward network is static, which means the input is associated with one particular output (the information move in one direction). But, the feedback network is dynamic, the one input provides a series of outputs. These series get from the cycles changes in the state, which is loop until reaching a stable point. The simplest example of feedforward ANN is Perceptron. The most popular feedforward ANN is back-propagation ANN, which based on back-propagation learning.

Learning

As we said before, machine learning is classified into two major types: supervised learning and unsupervised learning. Our research focuses on ANN models based on supervised learning algorithms. ANN uses learning to train the model to adjust weights into wanted values. The adjustment of weights

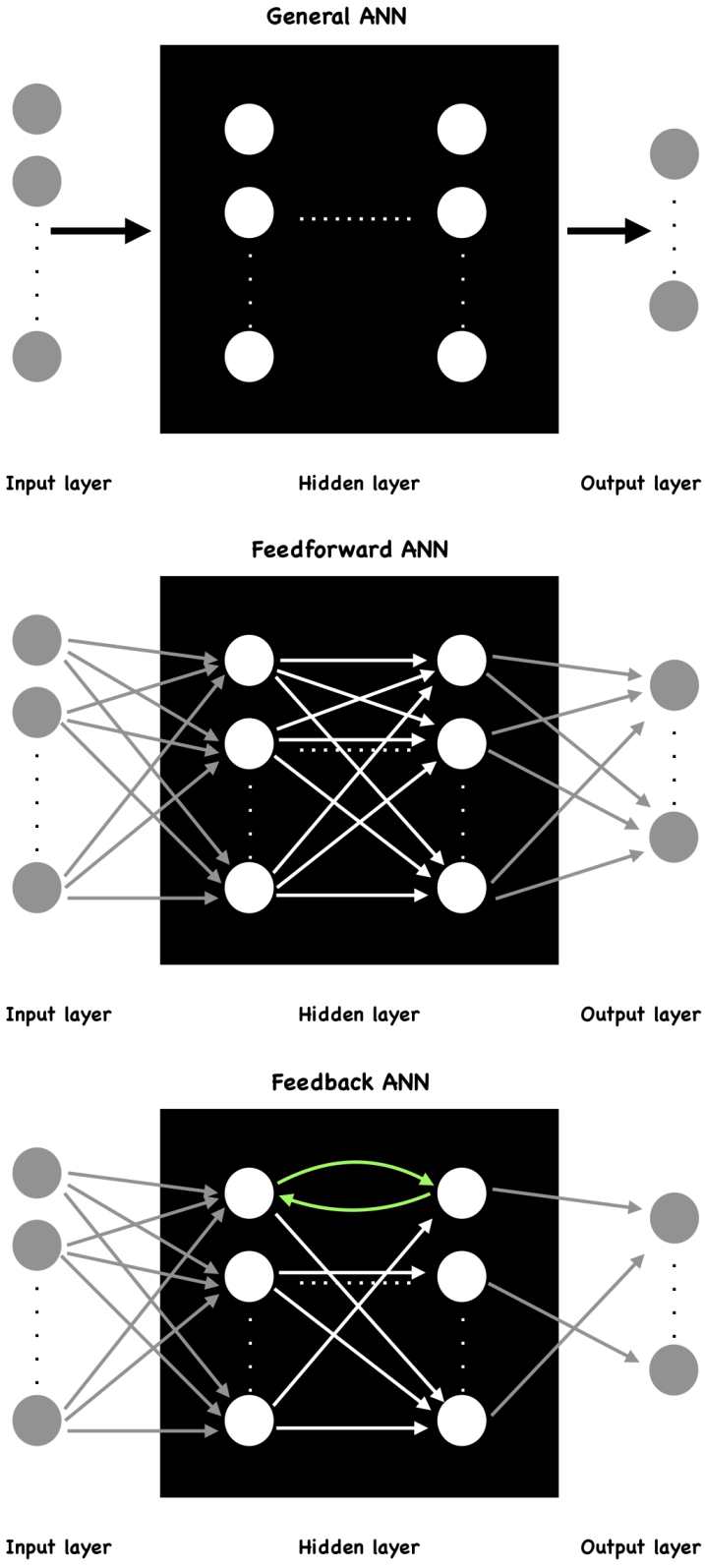


Figure 2.4: ANN different Topology. Inspire by [31].

done by minimizes the error between the output of the network and the actual value. Thus, the network learns by error correction. The most popular method for error correction is backpropagation. We can define the error function as the variation between the node output and actual output as follows:

$$e_n = y'_k - y_k$$

where e_n, is the error at step n, between the output of nodes k actual output value of the same node. Now, which fast the network will learn, or the weights are adjusted. This fast called the **learning rate**, which has a sharp impact on the system convergent. Let this learning rate be a constant λ for adjustment the new weights such that for input x_i the new weight is calculated as

$$w_{k,j,n+1} = w_{k,j,n} - \lambda e_k x_i$$

After training the model and get the optimal weights, the weights are fixing, and the network can use to predicate or operate for new values.

2.4 Random Forest

Random Forest RF invented by Breiman [6], which is based on random decision forest [12] and it is an instance of bagging of predictors [5]. The main problem with the decision tree is the high variance of the resulting predictor which means we have over-fitting in the model. The bagging technique solved this problem by reducing the variance of the model. The idea of bagging is draw with replacement from the data set D multiple sub-sample sets $d_1, d_2, ..., d_n$, and train the model (such as decision tree DT) on the data set as shown on Figure 2.5. The final model is the average of all $(m_1, m_2, ..., m_n)$

such that

$$m(x) = \frac{1}{n}\sum_{i=1}^{n} m_i(x)$$

The random forest model is easy and suitable to use most of the time. In this model, we draw different datasets from the main data point. Every dataset will be used to train a decision tree. Then we take the average of results to be the final random forest model. The trees in the random forest make different mistakes in test time. We take the average of the mistakes to divide their effects. In our research, we use RFR sklearn. In the RFR sklearn, the decision trees in the ensemble are built from a sample drawn with replacement from the training data [24].

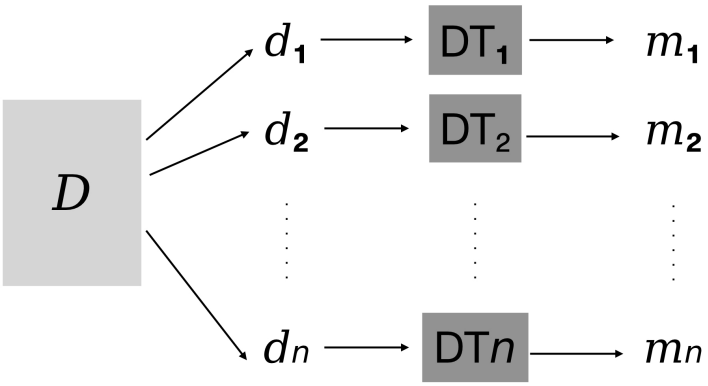


Figure 2.5: Bagging and random forest idea.

The random forest has two hyper-parameters. First, the max number of features parameter. It works with high dimensional data with a large number of features to reduce them. Second, we need to choose a large number of decision trees (or the number of sub-samples). But it is preferable not to be overused because this will end up with a long training run time.

Random Forest finds the importance of explanatory variables in prediction which is the most complex part of this method. It is enough to know that

RF estimates the importance of the random variable by observing the effect on the prediction accuracy when the value of the variable changes. Consequently, remove the variables that lead to low accuracy.

2.5 K-mean clustering

Clustering is the capability of grouping similar objects. Clustering is an unsupervised method that groups unlabeled data into clusters of similar inputs. K-mean clustering invented by Stuart Lloyd [14, 15]. The main idea of K-clustering is to find the centroids, which is the mean average of K clusters. K-means means how many clusters we have. This method works as follow

- We initially start by randomly place K centroids.
- We assign each data point to its closest cluster k.
- We update the centroids.

Figure 2.6 shows the procedure visualization of the K-mean clustering algorithm.

We get the grouping similarity by distance measurement. If we have a point x_i and another point x_j the distance computes by the following equation

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i1} - x_{j1})^2 + \dots} \quad = \quad ||x_i - x_j||$$

This is the Euclidean distance. We use it to find the distance between the k^{th} cluster centroid c_k and the data point. If the distance is small, the point is similar to the cluster. But if the distance is large, it should not be in this cluster. The objective of the distance calculation is to reduce the error and

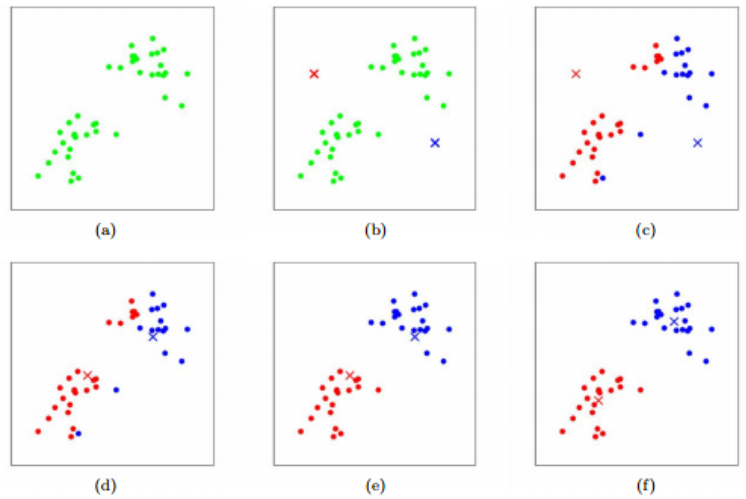


Figure 2.6: K-mean clustering algorithm. Reprinted from [9], "Training examples are shown as dots, and cluster centroids are shown as crosses. (a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations of k-means".

is the sum of distances across all clusters

$$E = \sum_{k=1}^{K} \sum_{x \in c_k} ||x - m_k||$$

where m_k is the centroid of the kth cluster. We need to minimize E because we want to have similar objects in the same cluster. Now, the main role in K- means is to update the centroids. In simple, updated centroids are the average of all $x \in c_k$ for $k \in \{1, 2, \ldots K\}$. We started with some initialization. Then we repeat the calculation of the distance and the update of centroids. The question now is when should the model stop working? We can stop after some iteration or when centroids do not change anymore. We usually have a problem with K-means clustering. That is we need to know the best value of K because the K-means algorithm is outlier sensitive.

Chapter 3

Literature Review

3.1 Limit of predictability approache

In general, we do not feel that our activities behaviors are random or unorganized actions. But, an external observer sees that we act randomly and unpredictably. This is because he is unaware of our motives and schedule. So, when we design a model in humane dynamic fields, we have a gap between our intuition and the actual modeling diagram. That is the approach of Song et. al. [26]. Song et. al. presented the limit of predictability in human mobility by asking a fundamental question: "To what degree is human behavior predictable?". Through human mobility, they aim to find the limits that characterize the predictability of human dynamics. They used a 3-month long mobile carriers records, collected for capturing the mobility patterns of 50,000 individuals. From measuring the *entropy* of each individual's trajectory, they found a 93% potential predictability in user mobility. Entropy is the most fundamental quantity capturing the degree of predictability characterizing a time series. It is a natural measurement of the difficulty in predicting the development of the process [4]. Moreover, low entropy means

higher predictability. Indeed, they determined Π^{max} separately for each user in the data set. Where Π^{max} represents the maximum predictability that an appropriate predictive algorithm can predict correctly. So, for a user with entropy S and moves between N locations, the predictability for this user $\Pi \leq \Pi^{max}(S, N)$. For instance, a user with maximum predictability $\Pi^{max} = 0.4$ means that 40% of the time we can predict his whereabouts and 60% of the time the way of chooses of his location appears to be random. Some researchers used the limit of predictability approach with the taxi demand prediction problem. Zhao et. al. [30] analyzed Π^{max} of the taxi demand in a region to select the best predictor. The maximum predictability Π^{max} is a value between 0 and 1. The algorithm is more accurate when Π^{max} is larger. So for the given prediction algorithm α, Π^{max} is the maximum predictability that a predictive algorithm can reach. They used three types of entropy: the random entropy $S^{(i)}_{random}$, the Shannon entropy $S^{(i)}_{Shannon}$ and the real entropy $S^{(i)}_{real}$. So, for an entropy S and clear taxi demand $N^{(i)}$ at clear location i, the maximum predictability Π^{max} computed as follows:

$$S = -\Pi^{max} log_2(\Pi^{max}) - (1 - \Pi^{max}) log_2(1 - \Pi^{max}) + (1 - \Pi^{max}) log_2(N^{(i)} - 1) \tag{3.1}$$

From Equation 3.1 we found Π^{max} by move S to the right side and get $f(\Pi^{max}) = 0$. Where S and $N^{(i)}$ are knowing for each building block i. So, the computation time to find Π^{max} for m building blocks is $O(m)$. Thus, there are three maximum predictability values for three different entropy Π^{random}, $\Pi^{Shannon}$ and Π^{real}, such that $\Pi^{random} < \Pi^{Shannon} < \Pi^{real}$. If we consider Π is the correct predictability of algorithm α, then subject to the Fano's inequality, $\Pi < \Pi^{max}$ [26]. So, they defined the problem of taxi demand

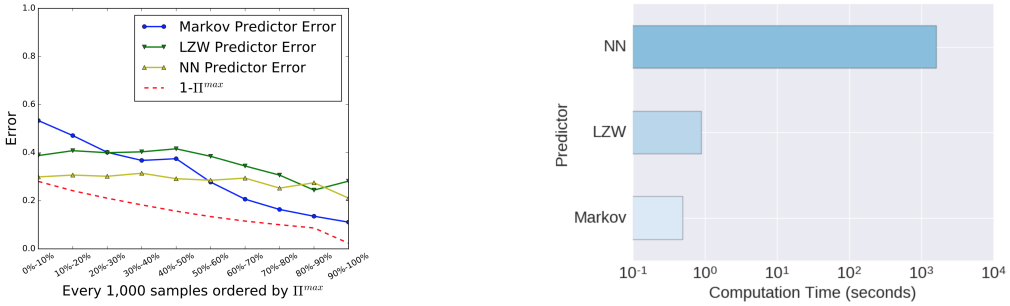
prediction as follows: "Given a predictive algorithm α and a sequence $D_n^{(i)}$ representing the historical taxi demand from time 1 to n at the building block i, considering both the randomness and the temporal correlation of the taxi demand, they wanted to find the maximum predictability (the highest potential accuracy) Π^{max} that a predictive algorithm α can reach". The taxi pick-ups at building lock i and time t represented by $d_t^{(i)}$ ($1 \leq i \leq m$ and $1 \leq t \leq n$). Thus, for a given location i we have

$$D_n^{(i)} = d_1^{(i)}, d_2^{(i)}, ..., d_n^{(i)}$$

For example, $D_n^{(i)} = 3244$ means that for block i, at time step 1 there were 3 pickups, at time step 2 there was 2 pickups, at time step 3 there was 3 pickups and at time step 4 there were 4 pickups [30]. To predict taxi demand they implemented and compared between three models: the Markov model, Lempel-Ziv-Welch (LZW), and the Neural Network . Then they tried to find the best model given the maximum predictability Π^{max}. They used k-order Markov model to predict the next taxi demand after k of taxi demands sequence $(d^i_{n-k+1}, d^i_{n-k+2}, ..., d^i_n)$. $d_t^{(i)}$ is the real taxi demand at the building block i at time t. Also, they defined a random variable $X_t^{(i)}$ to donate the taxi demand at time t and building block i. Thus, for building block i with historical taxi demand sequence $D_n^{(i)} = d_1^{(i)}, d_2^{(i)}, ..., d_n^{(i)}$ and the set of all possible taxi demand $N^{(i)}$, the Markov property is that $P(X_{n+1}^{(i)} = \beta | X_n^{(i)} = D_n^{(i)})$. Also, they defined an algorithm to find the Transition Matrix T, which is the most important tool for Markov prediction. Thus, from T they directly found the probability of that there are β number taxi demand in the location i at time step $n+1$. In addition, they use Lempel-Ziv-Welch (LZW) [8], which is an algorithm used to encode the characters stream. They used LZW to parse

the sequence of the taxi demand $D_n^{(i)}$ to sub-sequences $s_t^{(i)}$. For instance, if we have $D_n^{(i)} = 2213231$ then the sub-sequence will be $1, 2, 3, 22, 21, 31, 32, 23$. They built a tree with empty root and growing while parsing $D_n^{(i)}$. So, each node in the tree presents a sub-sequence from $s_t^{(i)}$. Then, as Markov model, they defined the probability $P(X_{n+1}^{(i)} = \beta | X_n^{(i)} = D_n^{(i)})$ but only considers the sequence in the partition $s_t^{(i)}$. Another approach they used is Neural Network NN. They implemented NN with two layers and 100 neurons for each layer. The input of the NN is the same as the research [17], and the output is β. They used the data of all trips NYC yellow taxi trips in June 2014 (13,813,031 taxi pick-up records), Uber taxi data set in June 2014 (663,845 pick-up records), and NYC Pluto data to map the GPS points for the associated building blocks. For there Data Preprocessing they used: "a 1200+ core cluster running Cloudera Data Hub 5.4 with Apache Spark 1.6. The cluster consisted of 20 high-end nodes, each with 24TB of disk, 256GB of RAM, and 64 AMD cores". They grouped the taxi data every hour and use yellow taxi data to find the entropy S and maximum predictability Π for each building block. From the distribution of $S_{random}^{(i)}$ they found that its peak at 3.6. So, if $S_{random}^{(i)} = log_2 N^{(i)}$, then the building block will have $N^{(i)} = 2^{s_{random}^{(i)}} = 2^{3.6} \approx 12$ taxi demand differences every hour. Based on the distribution of maximum predictability, they found that the average of $\Pi^{max} = 0.83$. It means that the taxi demand can be correctly predicted with an accuracy of 83%. From maximum predictability, they concluded that there is a temporal correlation for taxi demand. This helped them to improve the predictive accuracy in their algorithms. In their experiments, they used three weeks as training data and one week for testing. Also, the taxi data records were grouped every one hour for each location. They used 12-time intervals, Monday, Wednesday, and Sunday at midnight (00:00), morning (06:00), noon

(12:00), and evening (18:00), to reduce the bias. They sorted the building blocks by the maximum predictability, and group them to ten 1000 groups. Thus, they had 60,480 training samples and 1,200 for testing. They used sMAPE to evaluate the predictors and the prediction error bound $1 - \Pi^{max}$. The results are shown in Figure 3.1. The conclusion of their experiments was: NN used with the location with low predictability, because it can capture the multiple features. Markov predictor had the least computation time.



(a) (b)

Figure 3.1: (a) Prediction error and (b) Computation time of the Markov, LZW and NN predictor. This figure reprinted from Zhao et. al. [30]

3.2 Machine learning approaches

Machine learning models widely are used in various fields to build accurate forecasting models. Inspired by this, some machine learning models are applied to predict taxi demand. One of the most important models in machine learning is the Neural Network NN. Therefore, many researchers used it for the taxi demand prediction.

Mukai and Yoden [17] predict taxi demand based on probe data by the neural network. The probe systems are gathering information methods, which can store a large amount of data. The taxi probe system (Such as sensors in a taxi) provides data of taxi. They used the records from February to March

2009 in region (e.g. Tokyo's 23 wards, Mitaka-shi, and Musashino-shi). Each record contains ID, Latitude, Longitude, Month, Day, Time, Region. Besides, they added the day of the Week, and for this feature, there are three input patterns. Pattern 1: One-day, If the day weekend, the input value is 1, otherwise, 0. Pattern 2: Three-days, which are the previous day, the current day, and the after day. Pattern 3: Seven-days, day of the week (i.e 1,2,...,7). They adopt demands in each region, and there are 25 regions. Also, weather information (such as the amount of precipitation and air temperature) they are considering. The data in the February month used for training. And March month data used for validation. The structure of the neural network consists of three layers input layer, hidden layer, and output layer. In the input layer, the number of neurons depends on the day of the Week patterns. Table 3.1 presents 11 different input patterns. Therefore, the input layer has at least 25 neurons, that are needing to input values of demands in each region. The NN model forecast the future demand from the past data. For example, demands during 9:00-13:00 as input values and demands during 13:00-17:00 as output values. Thus, the output layer has 25 neurons. Each represents the value of the future demand for a region. In their experiments, they concluded that a neural network forecasting in a 4-hours time stamp and hidden layer with 50 neurons better than other network scenarios. Moreover, taxi demands vary widely according to the day of the week. About weather information, precipitation amount is ineffective, but rainfall must be considered.

Gustav Zander in his master thesis [29], predicted taxi demand in different geographical zones in the city of Stockholm using artificial neural networks. He tested different choice of parameters: number of neurons (20, 80, 150,

Table 3.1: Input Patterns of Neural Network. Reprinted from [17].

Pattern	One-day	Three-days	Seven-days	Precipitation
PT1	Unused	Unused	Unused	Unused
PT2	used	Unused	Unused	Unused
PT3	Unused	used	Unused	Unused
PT4	Unused	Unused	used	Unused
PT5	used	Unused	used	Unused
PT6	Unused	used	used	Unused
PT7	Unused	Unused	used	used
PT8	used	Unused	Unused	used
PT9	Unused	used	Unused	used
PT10	used	Unused	used	used
PT11	Unused	used	used	used

450), number of the hidden layer (1, 2, 3, 4, 5), sets of features (he tested five sets, which are present in Table 3.2), three optimization algorithms (SGD, RMSProp, Adam), four activation function (ReLU, ReLU6, Sigmoid, Tanh). Also, he combated overfitting when training the network by using different numbers of epochs.

Table 3.2: The feature sets Zander used for testing. Reprinted from Zander report [29].

Feature set number	Features
1 - basic	zone, hour.
2 - basic + week	zone, hour, day-of-week.
3 - basic + week/month	zone, hour, day-of-week, day-of-month, month-of-year.
4 - basic + precipitation	zone, hour, precipitation, temperature.
5 - all	zone, hour, day-of-week, day-of-month, month-of-year, precipitation, temperature.

He used two datasets, set contains the records in the district of Södermalm at the period from January 2010 to February 2016 (4934694 rides). The second one contains the data of the whole city of Stockholm (15463312 rids). For testing the network parameters he used K-fold cross-validation with $K = 10$ and *Loss* and *Accuracy* metrics. In his report, he presented all the differences

and the comparison between the network parameters. The final architecture of the network: fully connected neurons, with 150 hidden neurons per layer, 3 hidden layers, Mean squared error as a loss function, ADAM optimizer as an optimization algorithm, Tanh as an activation function for hidden layers, linear activation functions for the output layer. To prevent overfitting, he stopped training after five epochs. The best results were achieved when he used the complete feature set 5 (described in Table 3.2). Also, his network was able to predict 46% of the records within a 30% or 1 ride boundary with a mean error of 2.72.

Runmin Xu [28] used machine learning methods to predict taxi demand in NYC. He built five phases of ensemble estimator, which makes several single models work together. The most important in any machine learning method is the selection of problem features. He added demography of the place (population density, income) to his features, which correlated obviously to the taxi demand. The demography of a place has one weakness; it is changing slowly compared with the traffic dynamics,. So in some cases, it may not have any valuable effect. Events such as the parade, popular sport game, elections, and big music parties can also change the taxi demand. He collected the events manually from the news. In Public transit, there is a strong relationship between the different transportation modes. He defined this feature as $(p_t, p_{t-1}, \ldots, p_{t-d+1})$, where p_t is demand for public transit at hour t, and day d. Feature class for collaborative modeling: T stands for trend information, S is seasonal information, H for historical data, and E exogenous inputs.

In the methodology, he used five machine learning methods. K Nearest Neighbours KNN regression: for a given data, KNN computes the distance between the point and all the other points in the training set, then select

the closest K training data points and compute the average of the target output values of these K points, which is the final predicted result. The critical challenge in KNN is to choose the suitable value of parameter K, which must be careful. A large K works well for models with lower variance, and small K well with higher variance models. For taxi demand forecasting, KNN finds the nearest in history with the current pattern. The disadvantage of using KNN is the computational inefficient for large datasets. Artificial Neural Network: he used it with the seasonal data. Support Vector Machine: Support vector regression SVR proposed to get better generalization results. The special in SVR is $Kernal$. With kernels, the inputs of SVR mapped to a high dimensional feature space. SVR works well with events feature, which is irregular and may cause overfitting. Decision Tree DT: DT aims to make predictions by learning simple decision rules derived from the data features. DT works well with datasets that have missing features, in taxi demand prediction, there are many missing features such as events and transits data. But the problem with DT can easily tend to be overfitting. This weakness can be solved by using ensemble methods. Random forest is a good example of these methods. He used a grid search and random search for hyper-parameters selection. Under time and computational limitations, the random search is likely to be a suitable approach in some cases.

It is very hard to decide which model works best in all cases. Runnin Xu proposed "a five-phase procedure to build a multi-model ensemble" estimator, to produce better forecasting performances than single models. Phase one: Single model training (the models trained separately). This phase uses historical data. Phase two: Parameters optimization. Grid search method and random search method are considered to choose the appropriate one, with keeping the trade-off between accuracy requirements and computational ef-

ficiency. It uses one-week data. Phase three: Ensemble construction. He optimized single models and the entire set of each model is tested to construct the ensemble. One-week data is has been used. Phase four: Ensemble selection. The best ensemble method is selected among the candidate methods. Also, this phase uses one-week data. Phase five: Forecasting. The selected ensemble model phase four is used for forecast using. this phase forecasts the hourly demand for the next 12 hours. Model training and forecasting were described in detail in his report [28]. Table 3.3 represents the performance comparison for all models, where BS is a very simple method to make predictions.

Table 3.3: Ovarall performance measures of each model. Reprinted from Runmin report [28].

Model	MAE	MAPE(%)	RMSE
BS	44.5	18.5	60.0
KNN	37.8	15.2	51.5
ANN	34.8	14.2	46.2
SVM	34.1	14.3	44.2
DT	33.5	13.9	44.3
RF	30.0	12.7	40.2
ME	25.3	10.6	33.1

Chapter 4

Data and Methods

4.1 Datasets

Data Acquisition

Our research deals with the data of New York City, which is taken from [1]. Kaggle developed this data set for a challenge to build a model that predicts the total ride duration of taxi trips in New York City. But we use this data to build a model that predicts taxi demand prediction in NYC. This dataset is based on the 2016 NYC Yellow Cab trip record data and made available in Big Query on Google Cloud Platform. The data was published by the NYC Taxi and Limousine Commission (TLC). The set contains 1458644 trip records from January 2016 to June 2016. Indeed, Kaggle published two CSV files, a train set with 1458644 trip records, and a test set with 625134 trip records. We work with the train set for train and test. Each record contains 11 fields, we keep the following: "pickup-datetime", "dropoff-datetime", "passenger-count", "pickup-longitude", "pickup-latitude", "dropoff-longitude", "dropoff-latitude", "passengers-count'.

Data Pre-processing

The dataset is clean and has no none-values or invalid values. We use Python in our work for programming. The dataset contains information on locations in Latitude and Longitude format. We extracted some useful feature such as: day of week, day of year, day category (i.e: Saturday, Sunday,...) and weekend (i.e: 1 for Saturday and Sunday, 0 for other days).

In our research, the goal is to predict the number of pickups in a location at a time bin. So we need to do time binning. We predict the number of pickups in each location during 30-minutes (30mins time bin). To do time binning we take inspiration of the idea of Sdaulton project [22], which is a project work on such research. We do 48 number of time bins per day, this means we have 10 minutes per bin. Before that, we split pickup-DateTime to date and time (For example: split 2016-03-14 17:24:55 to date =2016-03-14, time = 17:24:55). After that, we find the time of the start of the bin. To make the time easy usable in machine learning methods, we calculate the floating-point representation of the center of the time bin. Then, to get the number of pickups for each location at time bins, we group data by pickup location and pickup time bin.

Another scenario to pre-prepare data, Using K-means clustering. Figure 4.1, and Figure 4.2 presents 40 clusters for the region by using pickup location coordinates, and 30 by using dropoff location coordinates. Then, we group data by pickup clusters and pickup time bin. Thus, we have the number of pickups in a cluster at a time bin.

To improve our data-set, we added weather data. We use the weather data in New York City for the first six months of 2016. This data set also published on Kaggle. It contains for each day the minimum temperature, maximum temperature, average temperature, precipitation, new snowfall, and current

snow depth. We merge it with our data by the day of the year. So, our features that will be used in the prediction are: time num, day-of-year, day-of-week, weekend, average temperature, precipitation, snowfall, pickup-latitude, pickup-longitude, geohash.

4.2 Data visualization

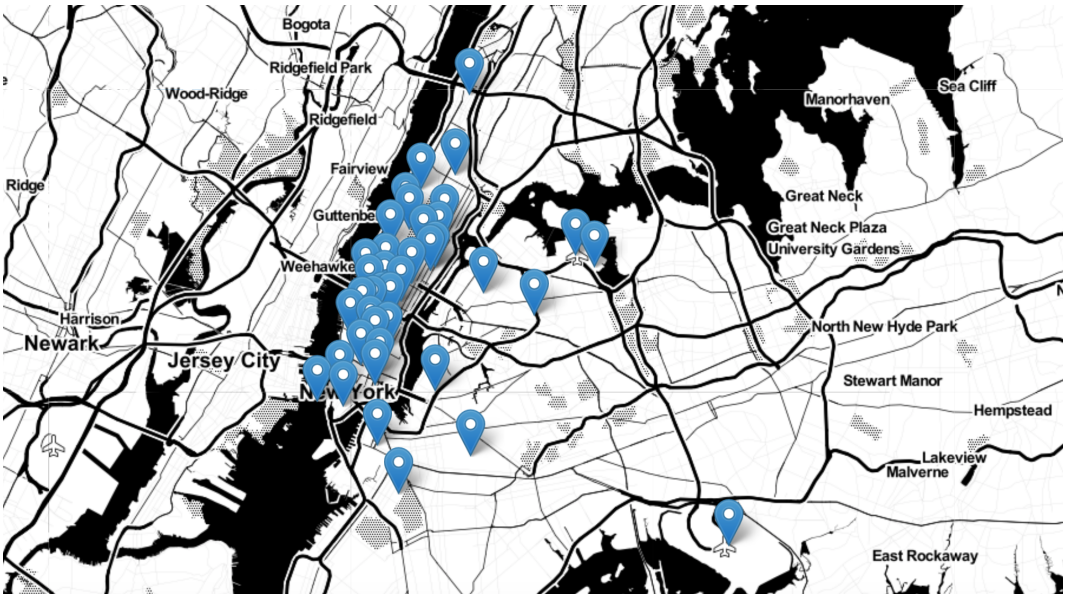
In this section we present some of data visualization as shown in Figure 4.1 and Figure 4.2. To visualize the data, we refer to [25] and [11].

4.2.1 NYC zones

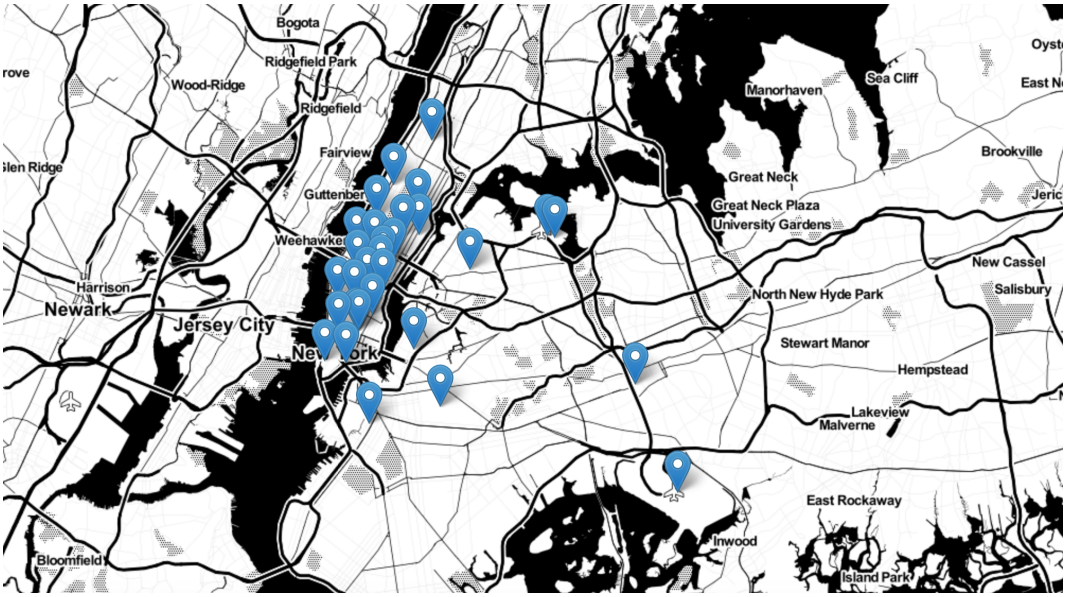
We use the codes line in [11] and split the NYC to 15 zones, which are the most famous neighborhood there. First, we use K-means clustering to get 15 clusters of locations for NYC as shown in Figure 4.3. Then, we use the coordinates to define each location zone. Table 4.1 shows the description of NYC neighborhoods and the important features for each place.

4.3 Creation of the workspace.

The programming language that we use is **Python** 3.7.6 version, which is the most popular language for scientific computing. This language free for all the major operating systems, namely Windows, Mac, and Linux. Also, Python is easy to understand language and comes with a large number of libraries. Some of the libraries are scikit-learn (*sklearn*) using for data mining, analysis, and machine learning; *NumPy*, to deal with multidimensional arrays; and *Matplotlib*, use to plot high-quality graphs using Python [21]. Thus, Python is the best programming tool to use with machine learning.



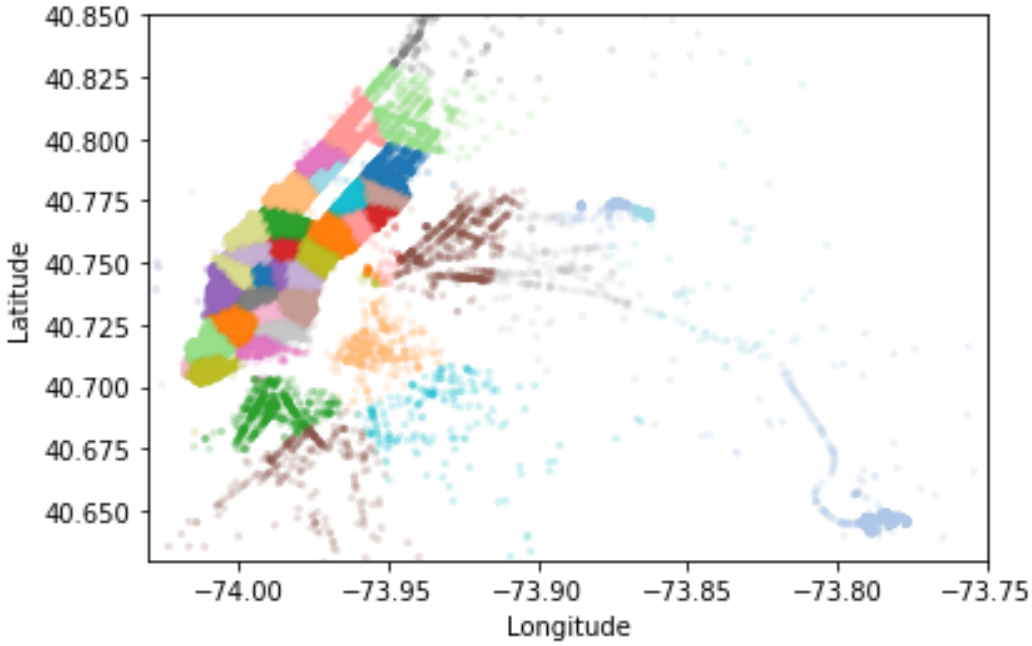
(a) 40-means clusters for the region by using pickup location coordinate



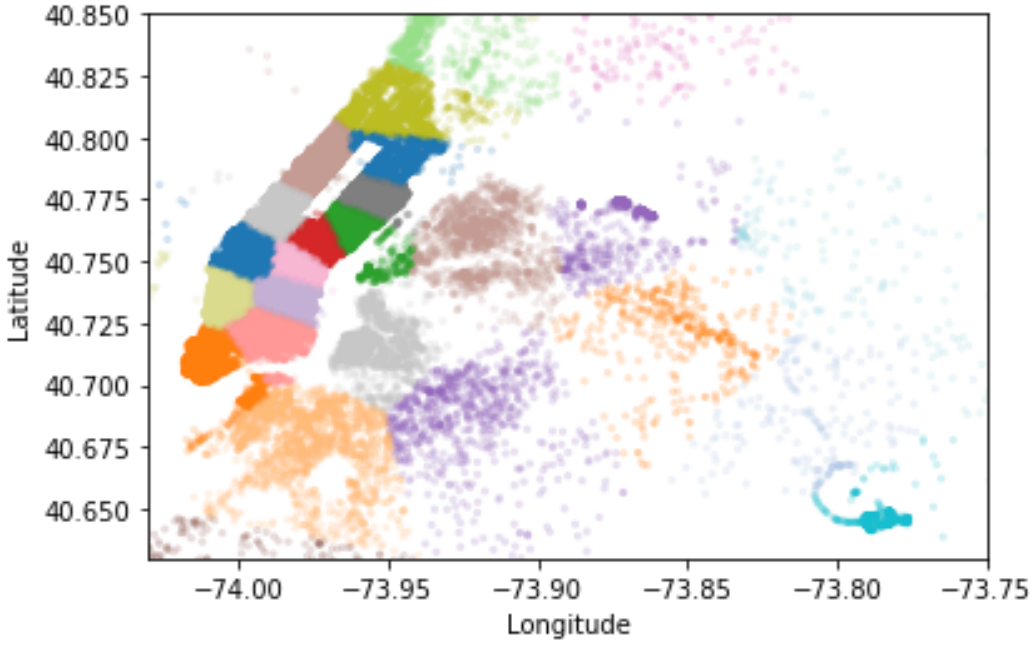
(b) 30-means clusters for the region by using dropoff location coordinate

Figure 4.1: K-means clusters for the region by using location coordinate

We need to create the work-space for our machine learning code and data set. Typically, we need some of Python modules, as we said NumPy, Matplotlib, Sklearn also pandas, and Jupyter. Also, Keras to build the ANN. In this section, we review the libraries and tools that help us in our research.



(a) The pickups clusters Visualisation on the map



(b) The dropoff clusters Visualisation on the map

Figure 4.2: The clusters Visualisation on the map. (a) presents 40 clusters for pickups coordinates and (b) presents 30 clusters for dropoff coordinates.

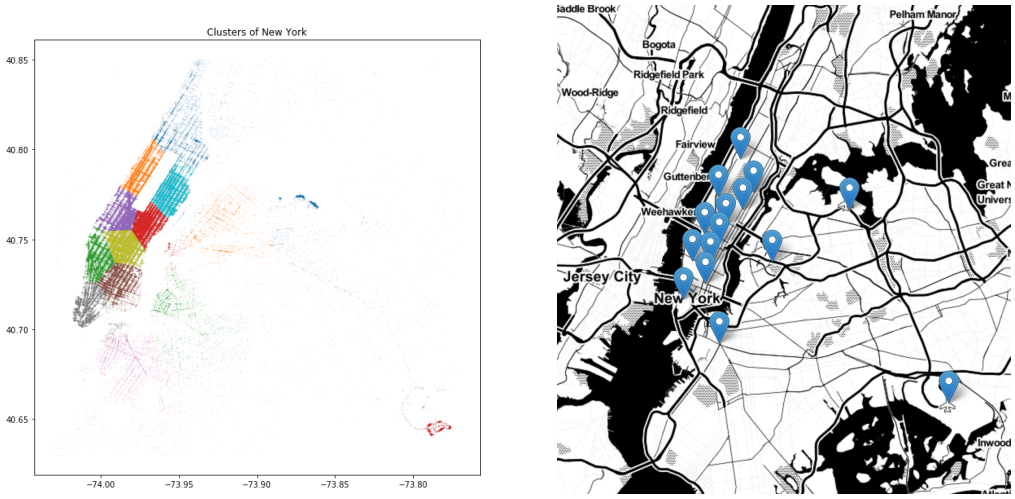


Figure 4.3: The 15 zones clusters Visualisation on the map

4.3.1 Scikit-learn: Machine Learning in Python [20]

Scikit-learn uses to implement many well-known machine learning algorithms, which is maintaining an easy-to-use interface with the Python language. We use it to build an estimator to fit method, accepting as arguments an input data array and an array of labels for supervised problems. We use sklearn to implement some estimators such as random forest regression and SVR. Each estimator has a score method, which is an evaluation of the goodness of fit. So it is easy to evaluate our models by finding all the evaluation matrices we want.

In machine learning procedures, we need to split data set to train, evaluate, and test subsets. In the sklearn, a cross-validation iterator provides pairs of train and test indices to split input data, such as K-fold [23]. Each machine learning algorithm needs to tune its parameters. Luckily, sklearn helps us to tune our estimator's hyper-parameters. Sklearn does this by using cross-validation to evaluate the performance of the estimator and select parameters. This is done by using GridSearchCV and RandomizedSearchCV, where the "CV" stands for Cross-Validated.

Table 4.1: NYC neighbourhoods (zones) Description. Taken from Wikipedia [27].

Zones	**Zone description**
Midtown	Midtown Manhattan is the central portion of the NYC.
Downtown	the central borough for business, culture, and government of the NYC.
Chelsea	houses, luxury high-rises, and modern attractions like the High Line.
Queens	hosts many sports tournament. Contains Queens Museum.
JFK	international airport
Midtown (West, East)	the central portion of the New York City borough of Manhattan.
East Village	Old bars, music venues, performance spaces share.
Brooklyn park slope	residential area with boutiques, laid-back bars and casual restaurants.
LaGuardia	An airport
Queens-Astoria	multicultural neighborhood with ethnic eateries and trendy spots.
Harlem	knowns for jazz clubs, and African-American heritage.
Upper East Side	knowns for its rich denizens, fancy restaurants and designer shops.
Williamsburgt	have a waterfront view for seasonal venues for outdoor concerts.
Upper West Side	hosts performing-arts institutions like the NYC Opera and Ballet.

4.3.2 Anaconda and JupyterLab

Anaconda [2] is a Python data science platform that provides most of the Python tools and libraries. By using the package manager Conda, we can quickly install, run, and update packages. Conda is an open-source package management system that runs on different operating systems. Also, we write, run, and iterate our Python code by using Anaconda's JupyterLab. We use JupyterLab because it is a powerful tool for interactively developing, which based on the Jupyter Notebook. Also, by using Jupyter Notebook, we can

easily export the code to pdf, latex, etc. Jupyter uses the kernel program to runs and introspects the code. Consequently, we use Python 3 as a kernel in Jupyter, since we use Python as a programming language in our thesis.

4.4 General description of the proposed methods

This section describes and defines some of the proposed methods we use in this research. The features of the problem described in Table 4.2. These features are used to predict taxi demand (number of pickups).

Table 4.2: Features Description

Feature	feature description
dayofyear	day of the year, we have half of year, so dayofyear= $1, 2, ..., 182$.
pu-dayofweek	pickup day of week, pu-dayofweek= $1, 2, ..., 7$
time-num	floating-point representation of the center of the time bin
pickup-cluster	the cluster of pickup.
pickup-longitude	the longitude coordinate for pickup
pickup-latitude	the latitude coordinate for pickup
average temperature	the average temperature of the day
precipitation	the precipitation and rain of the day
snowfall	snowfall at the day

4.4.1 Methodology for taxi demand prediction

We use Python sklearn library to build Random forest regression and SVR model and Keras to build ANN. Also, sklean work with us each step with data splitting and evaluation of models.

Train/Test split

To split train-test split data, we use the sklearn model-selection train-test-split with 20% of data is test size.

Weather Data

Weather data for days of the year 2016 is taken from Kaggle.com, which is collected from the National Weather Service. This data contains only the first six months, which are the months that we have in our dataset. The location of weather information is in the Central Park of NYC. For each day, we have minimum temperature, maximum temperature, average temperature, precipitation, new snowfall, and current snow depth. We use some of this information, as shown in Table 4.2. Some of the columns contain the value 'T', which means that the information was registered, but not valuable (not enough for a value). We replace 'T' with zero value, to make data easy to use with ML algorithms.

Taxi trip location

We only consider the pickup location. Location information contains latitude and longitude coordinates. We round the location coordinates to three decimal coordinates. Also, We use sklearn K-Means clustering to group location into 40 clusters, as we showed in data visualization. And we use K-means clustering also to get the main 15 NYC neighbourhoods.

Random Forest Regression

The decision tree has a weakness with big variance data. Ensemble models solve this problem by providing the Random Forest Regression RFR. First, we use sklearn to implement random forest regression with 500 decision tree

as in [13]. The number of estimators in ensemble models (here the estimator is the decision tree) necessary for performance grows with the number of predictors. It is not hard to adjust the number of estimators in our RFR model. The best way, as shown in [13], is to start with a default value, half the default, and twice the default, then pick the best. The features important that we get from the initial RFR is shown in Figure 4.2.

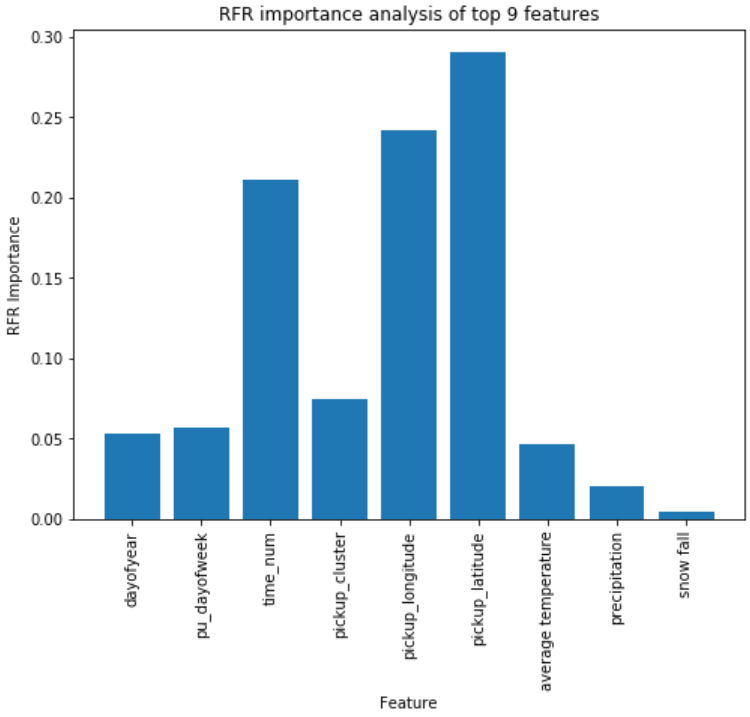


Figure 4.4: Feature importance

SVR

We implement the SVR model to solve our problem to get the benefits of kernel functions. We think that kernel functions have control over the non-linear problems. SVR model has four hyper-parameters (C, gamma, epsilon, and Kernel) that need to adjust. Sklearn provides GridSearchCV (Exhaustive search) and RandomizedSearchC (not all parameter values tried out, a fixed

number are picked randomly from the search space) to tune parameters.

Chapter 5

Experiments and Results

This chapter presents and analyzes the experimental results of applying the machine learning methods.

5.1 Computing Environment

Our experiments, were implemented using Python 3.6.7, under macOs Mojave 10.14.6 with intel core i5 1.30GHz, 256GB RAM.

5.2 Evaluation Metrics

In our research, we use three different evaluation metrics: R square score, MAE and MSE.

R^2 is the coefficient of determination. It shows us the goodness of the model to fit the data. And the formula for R-Squared is

$$R^2 = 1 - \frac{\text{sum square regression (SSR)}}{\text{total sum of squared (SST)}} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2}$$

Where $\hat{y}_i$ is the prediction value, and y_i is the actual value. When R^2=1, this means that the predictions fit actual values perfectly. R^2=0 means that the model not fitted the data at all. The negative R^2 means that the regression line of the model is worse than the horizontal line. So, when R^2 value goes to one, the model fits the data better more. Sometimes, in regression problem the high value of R^2 means that the model is over-fitting model.

Mean Absolute Error MAE, measures how close the predictions value $\hat{y}_i$ to the actual value y_i,

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

Mean Square Error (MSE) is the average of all the errors, and given by

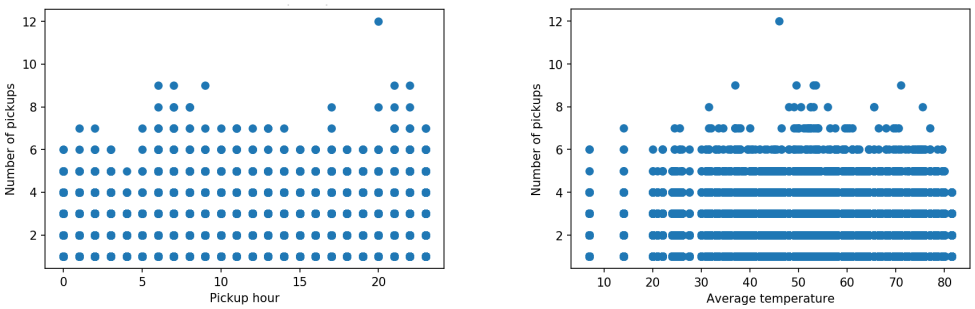
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

5.3 Experiments design

We do our experiment in different ways, depending on location features. We consider the location as a cluster, in which we have 40 different clusters for NYC. In other experiments, we consider the latitude and longitude coordinates of each pickup location rounded to three decimals. With rounded to three decimal coordinates, we get from this scale a neighborhood or street with a length of approximately 111.32 m. The four decimal rounded giving 100 square meters area, but this area is small and does not provide useful information. So, three decimal coordinates do the job. Also, we consider the location as one of NYC neighbourhoods. The details are in the next parts.

5.4 Experiment 1: RFR model

In this experiment, we apply RFR model. First, we use the three decimal coordinates pickups location. And Figure 5.1 presents the scatter plot distribution of the number of poickups at different hours and temperatures.



(a) Pickup hour vs number of pickups

(b) Average temperature vs number of pickups

Figure 5.1: (a) The scatter plot of the pickup hour with the number of pickups (b) The scatter plot of the average temperature with the number of pickups.

RFR model applying on three decimal coordinates location dataset

In this experiment, we use the Sklearn RFR model with 1000 number of estimators (decision trees) and default maximum depth of the tree. Then we apply the same RFR model with a maximum depth of 100. But there are no critical changes in the result when we add the maximum depth of the tree to the model, as shown in Table 5.1.

Table 5.1: Performance measures of RFR model.

Model	R^2	MSE	MAE	max error
RFR(estimators=1000)	0.0115	0.126	0.175	9.86
RFR(estimators=1000, max depth=100)	0.0113	0.126	0.175	9.79

Why the number of estimators equal to 1000? we try a different number of estimators, small values, also large values. Small values (10-500) do not have the best performance, and a large number of estimator takes very long run time with a slow change in performance. So, 1000 estimators do the job. Figure 5.2 shows the importance of the features in this experiment. From the importance of features, we see that the most important features are the time, day of the year, location coordinates, and the temperature.

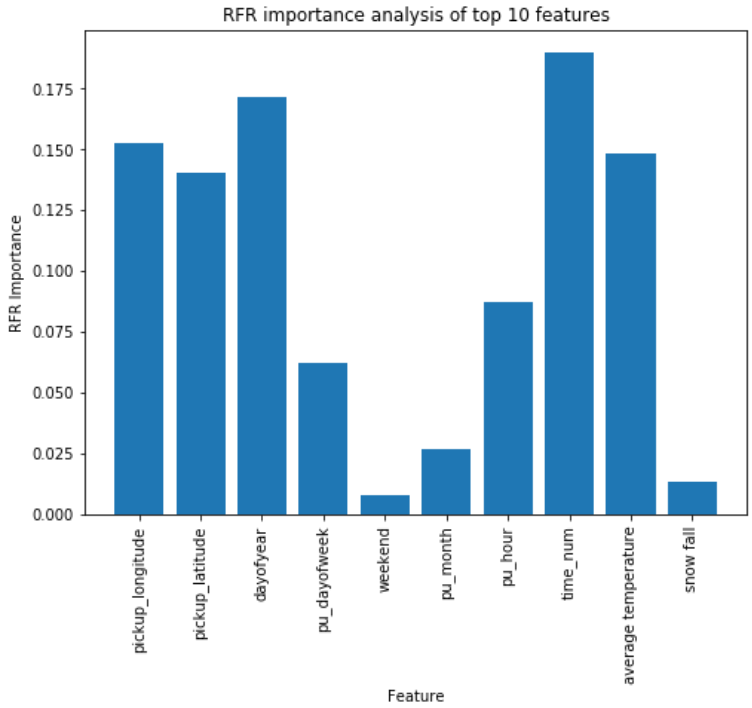


Figure 5.2: Feature importance for RFR with three decimal coordinates feature location

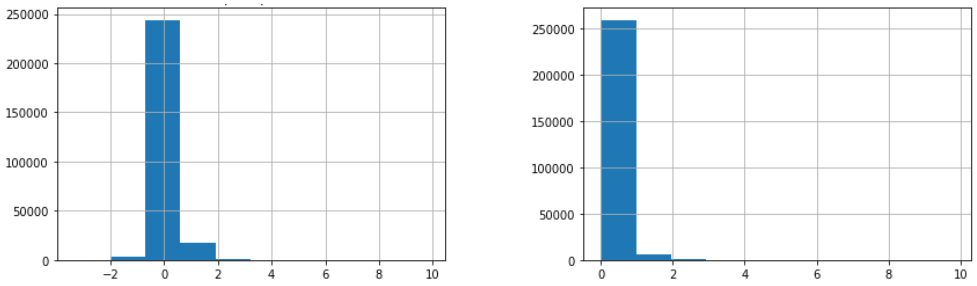
Erorr Analysis for RFR model

Now, to understand the performance of the model we analyze the residual forecast errors. The residuals are the difference between the pickups' actual value and the predicted value. For this analysis, we refer to the machine learning mastery website [7]. We take the absolute value for the difference

between the actual value and the predicted value. Table 5.2 shows the statistical summary for the distribution of the absolute residual errors. It gives a fast description of our residual errors. Figure 5.3 shows the histogram of the residual errors and the absolute residual errors, which give the information about the skews of the residual errors.

Table 5.2: The statistical summary for the distribution of the absolute residual errors

count	265679
mean	0.175487
std	0.309672
min	0.000000
25%	0.021000
50%	0.060000
75%	0.167000
max	9.796000



(a) Error residuals (b) Absolute error residuals

Figure 5.3: (a) The histograms of the residual errors (b) The histograms of the absolute residual errors

From Figure 5.3(a), we see that the most residual errors distributed around zero. Also, this figure shows that the residuals with a value between 0 and 1 are the most frequencies values in an unnormal way. It means that we have a high frequency in our Y valves. That means the dataset needs a different

way to deal with it.

Figure 5.4 shows the scatter plot for the error residuals with the pickup hour. From the figure, we observe that the first hours of the morning and the hours of midnight approximately have the highest and variate error.

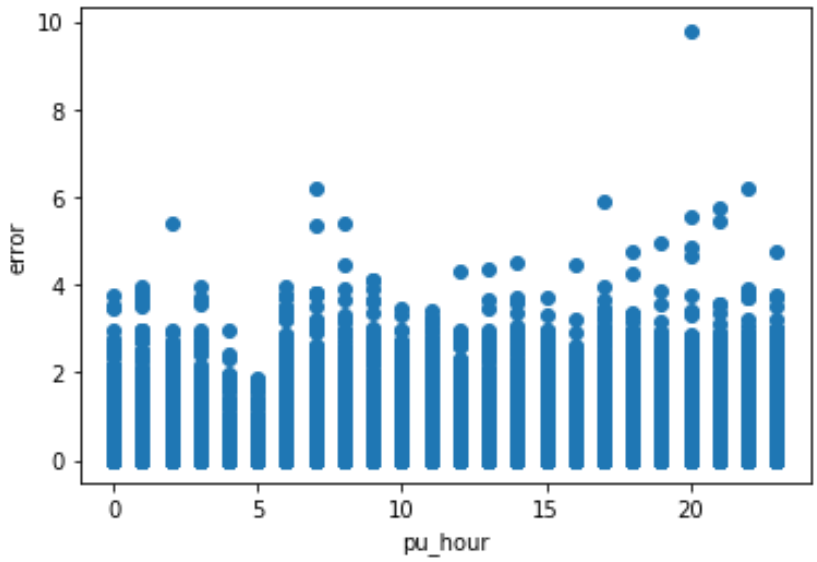


Figure 5.4: Scatter plot for the error residuals vs pickup hour

RFR model applying on cluster location dataset

Now we use RFR model with cluster location. As in the previous, there is no critical difference between RFR with maximum depth consideration or without it. So we compare the results of the different locations (cluster or three decimal coordinates). Table 5.3 presents the compression of performance matrices. Also, the features importance in this case is change as shown in Figure 5.5. Where is the pickup cluster is the most importance feature. In addition, the statistical summary of absolute residual error presented in Table 5.4. Indeed, we can not say which one of the two cases is better. But, from these cases, we see the effects of the location features on the prediction

way and results.

Table 5.3: Performance compression RFR model for different location feature.

RFR location feature	R^2	MSE	MAE	max error
3 decimal coordinates	0.0113	0.126	0.175	9.79
cluster	0.4954	3.2368	1.2978	17.0820

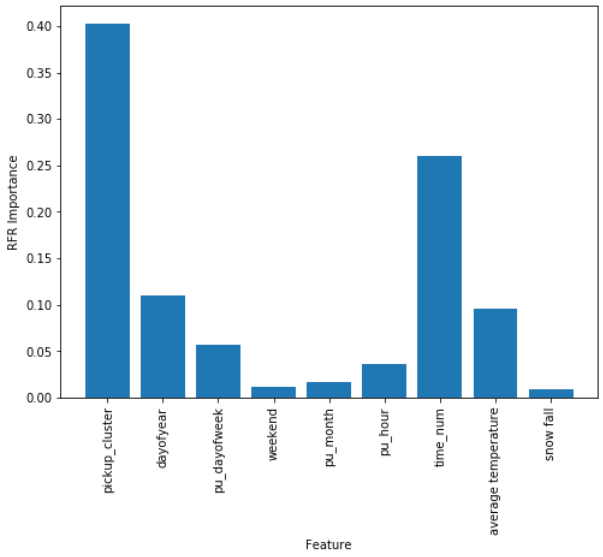
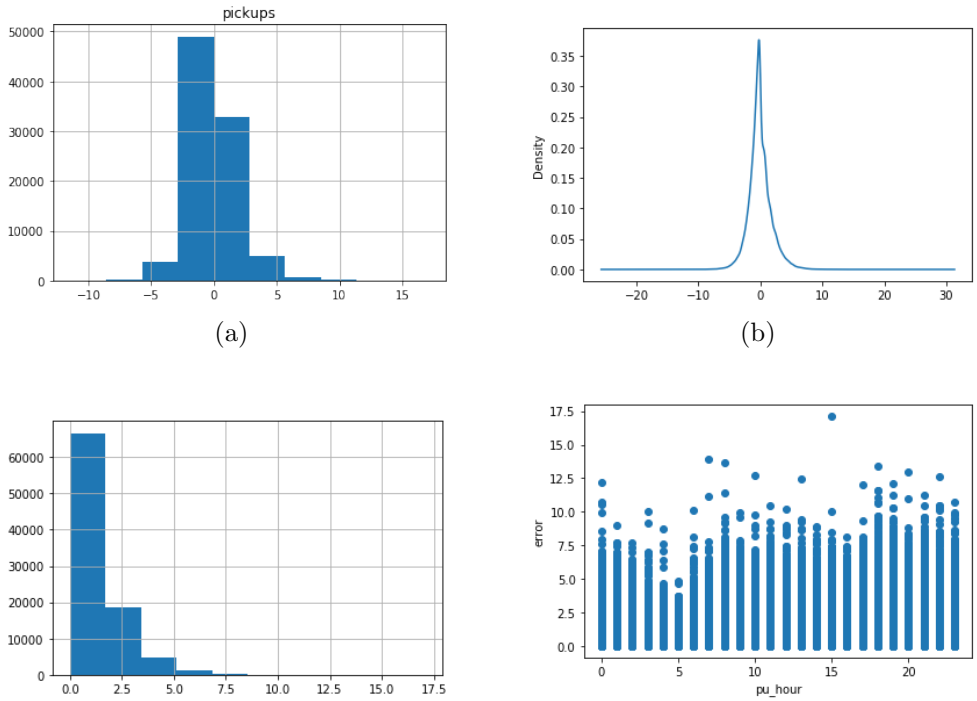


Figure 5.5: Feature importance from RFR with cluster feature location.

Table 5.4: The statistical summary for the distribution of the absolute residual errors

count	91435
mean	1.297767
std	1.246024
min	0.000000
25%	0.405000
50%	0.923000
75%	1.804000
max	17.082000



(a) (b) (c) (d)

Figure 5.6: Some of error virtualization for RFR model with cluster location feature (a) The histograms of the residual errors (b) The density plot of the absolute residual errors (c) luster location feature (a) The histograms of the absolute residual errors (d) luster location feature The scatter plot of the residual errors vs the pickup hour

RFR model applying on NYC neighbourhoods

We apply RFR on the data with NYC neighbourhoods location. First, we try different values of number of estimators and max depth. As shown in Table 5.5, the max depth more than 50 does not has critical change on the performance. The number of estimators effect on the result, as shown in Table 5.5. Consequently, we take the number of estimators equal to 1500, not 2000 or more since it takes long run time with a small change in performance. Figure 5.7 shows the features importance that we get from using NYC neighbourhoods as a location. Figure 5.8 shows the distribution of the error residuals, which is go to the normal distribution around zero more than the previous residuals. Therefore, the error approximately does not skew to

Table 5.5: MSE of using different RFR parameters with NYC neighbourhoods location experiment.

Number of estimators	Max depth	MSE
100	None	43.746
100	30	43.294
1000	10	38.896
1000	20	38.896
1000	50	**10.053**
1000	1000	43.398
1500	50	**8.302**
2000	None	43.393
2500	50	**8.260**
5000	50	**8.278**

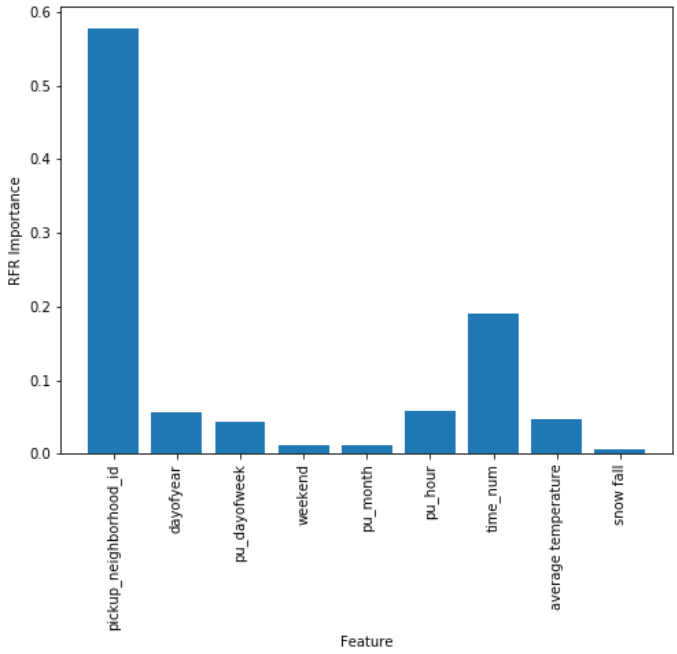
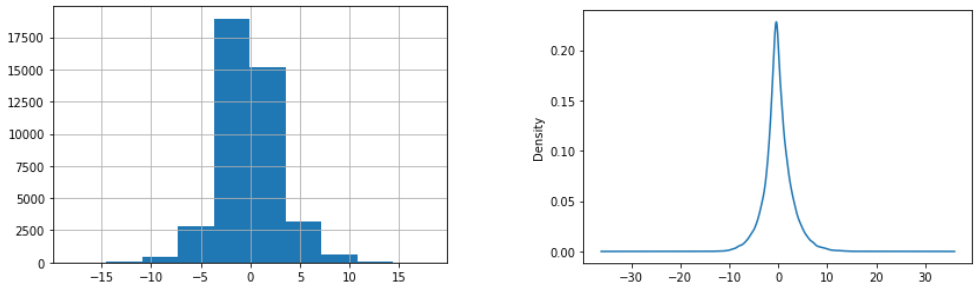


Figure 5.7: Feature importance from RFR with NYC neighbourhoods location.

some values.



(a) (b)

Figure 5.8: Error residuals of RFR with NYC neighbourhoods: (a) The histogram of the error residuals (b) The density plot of the error residuals.

5.5 Experiment 2: SVR model

In this experiment we apply the SVR model on our data with the cases of feature location. We use the three decimal location and try different values of parameter epsilon ϵ. Table 5.6 shows the performance of SVR model with different values of ϵ for the three decimal location feature. Figure 5.9 shows

Table 5.6: The performance of SVR model with three decimal location feature and different values of ϵ .

SVR(kernel='rbf', ϵ)	R^2	MSE	MAE	max error
$\epsilon = 0.1$	-0.0000	0.1288	0.1811	7.9002
$\epsilon = 0.01$	-0.0597	0.1364	0.1059	7.9902
$\epsilon = 0.001$	-0.07	0.14	0.10	11

the visualization of the residual errors. In Figure 5.9(a), when the number of pickups increases, the error also increases. We think this problem comes from the high frequency of the numbers of pickups (such as value 1). In Figure 5.9(c) and (d) we observe that the variance of error increase with the hours of the midnight and with the high temperature.

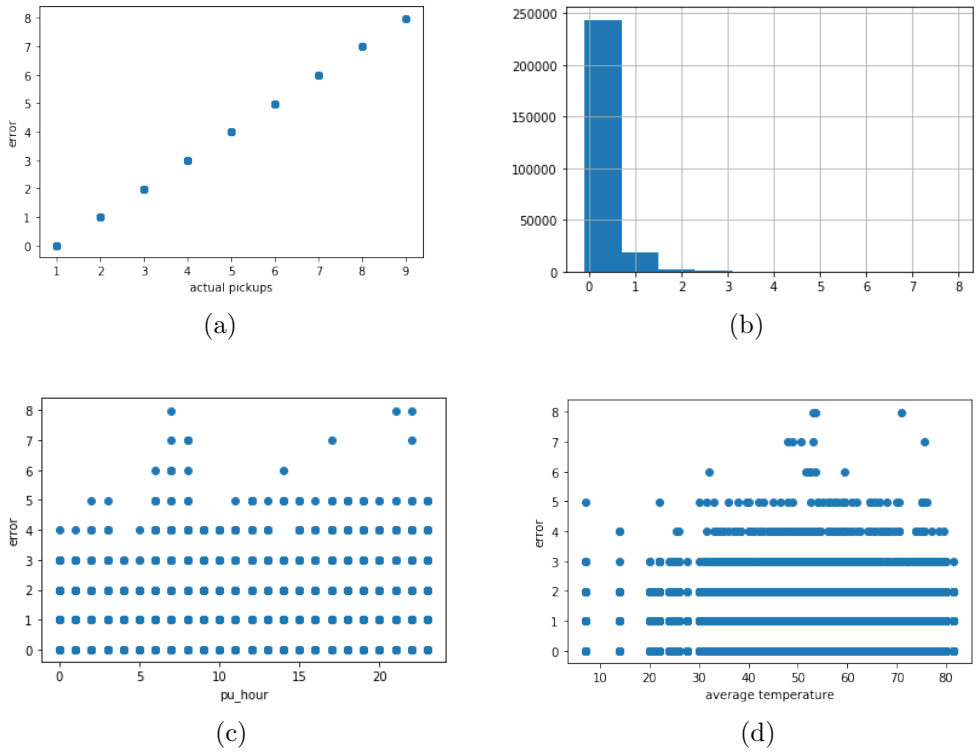


Figure 5.9: Some of the error virtualization for SVR model with three decimal location feature (a) Scatter plot of the actual value of number of pickups with the residuals error. (b) The histogram of the residual errors (c) Scatter plot of the pickup hour with the residuals error. (d) Scatter plot of the average temperature with the residuals error.

5.6 Experiment 3: Artificial Neural Network

In this experiment, we apply ANN. First, we use the location as three decimal coordinates. But, we get unexpected results, in which the network gives approximately the same prediction for all values as shown in Table 5.7. Therefore, NN does not work very well, with data has high skewness. Consequently, we go to apply NN on the data with the NYC neighborhood location. We build a NN with 6 layers as follows:

1. Input layer: Input shape = 9 and activate function = 'Relu',

2. 4 hidden layers: 13 neurons and activate function = 'Relu'

Table 5.7: The predicted values of the taxi pickups by using two different NNs, with 265679 rows.

actual value	predicted value 1	predicted value 2
1	0.701993	-0.506982
1	0.701993	-0.506982
1	0.701993	-0.506982
1	0.701993	-0.506982
1	0.701993	-0.506982
. . .	. . .	. . .
2	0.701993	-0.506982
1	0.701993	-0.506982
1	0.701993	-0.506982
1	0.701993	-0.506982
1	0.701993	-0.506982

3. Output layer.

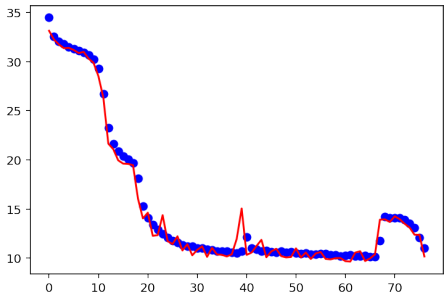
We use Adam optimizer, which is the best one. And mean squared error as a loss function. Indeed, we use NNs with one or two layers, but do not perform accurately. Table 5.8 shows the performance matrices of NN with different values of learning rate and epochs.

Table 5.8: The performance of NN model with NYC neighborhood location feature, different values of learning rate and number of epochs.

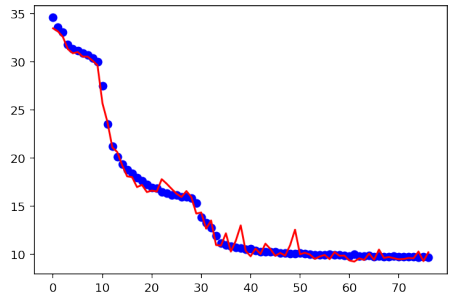
Learning rate	Epochs	R^2	MSE	MAE
0.001	1000	0.724	**10.13**	2.29
0.001	2000	**0.722**	10.20	**2.27**
0.001	3000	0.654	12.55	2.54
0.01	1000	0.082	31.30	4.30
0.01	2000	0.082	33.74	4.63

From the results, we find that the changes in the learning rate effect on the result more than the number of epochs. Figure 5.10 shows the comparison of the validation loss and the training loss with the different learning rate and number of epochs. In Figure 5.10(a) and (b) we have some of over-fitting

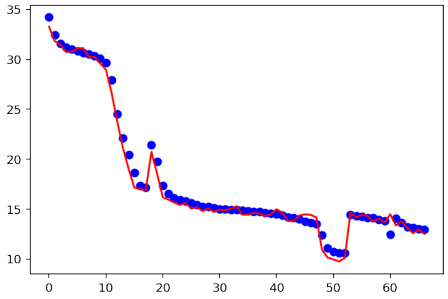
model. In Figure 5.10(d) and (e), we have under-fitting model. Therefore, Figure 5.10(c) has the best model.



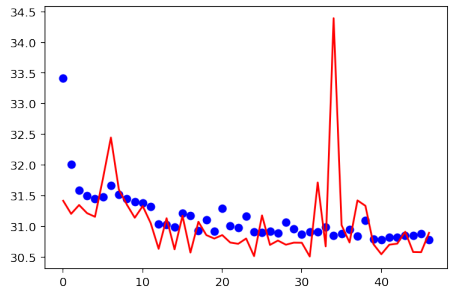
(a) lr=0.001, epochs=1000



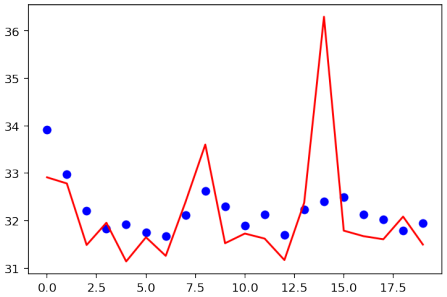
(b) lr=0.001, epochs=2000



(c) lr=0.001, epochs=3000



(d) lr=0.01, epochs=1000



(e) lr=0.01, epochs=2000

Figure 5.10: The plot of validation loss (blue) and training loss (red)

Chapter 6

Conclusion and Future Work

The aim of this research is to predict taxi demand in NYC in a location at a time using the ML approach. We do the preprocessing and visualizing of the dataset to understand the behavior of taxi demand. We apply many ML methods such as RFR, SVR, and ANN. For each model, we test different location features, analyzed as:

1. Pickup location cluster,
2. Pickup location coordinates (3 decimal rounded coordinates),
3. Pickup neighborhood.

For prediction purposes, we used sklearn RFR and SVR, also Keras NN, with different parameters for each. As a result of our experiments we found the following:

- The way of dealing with the features of taxi demand location in NYC has noticeably affected the prediction results. NYC neighborhood was the best as a location feature of the pickup because it has low-frequency values (low skewness), which affects the model performance. Moreover,

NYC is divided into 15 neighborhoods. Each has its own characteristics and features. This makes some variates in taxi demand.

- From the RFR model, we found that the most affected features of our problem are the pickup location and time. When we use the location of the pickups as three decimal coordinates, we noticed the effect of the weather and the pickup day. The best MSE and MAE of RFR was with three decimal coordinates with values equal to 0.126 and 0.175 as shown in Table 5.1. The RFR model has the best R^2 value with cluster location feature, with value equals to 0.494 as shown in Table 5.3.

- We apply SVR model on the three decimal location with different values of ϵ parameters. We get the best MAE value equals to 0.10 with $\epsilon = 0.001$ as shown in Table 5.6. SVR model has the highest executing runtime.

- ANN is the best model, in the training and testing of the dataset. The best performance of NN is with the NYC neighborhoods location feature, with learning rate $lr = 0.001$, $R^2 \approx 72\%$, MSE ≈ 10 and MAE ≈ 2. ANN has a weakness with the other location features, since the high frequency in the target. Consequently, ANN gives the same prediction for all values as shown Table 5.7.

The weather features of NYC do not have critical effect on the taxi demand prediction results. When the temperature is very low, we notice decrease on taxi demand error density as shown in Figure 5.1(b) and 5.9(d). Approximately, we have the highest number of pickups when temperature is moderate. This is a normal situation because most people like to go out to and have outdoors activities. Besides, we found that high temperature made people demand more pickups, too.

The SVR was the hardest model because it takes a long run time to train and tune the parameters. And we could not test it very well. In future work, we will improve the results of SVR experiments by tuning parameters to get the best performance. Therefore, we need to do more different experiments with different parameters and features.

Obviously, time is an important feature in taxi demand prediction problem. Building on our research, we will deal with the pickup time as we did with the location features. From Figure 5.1(a), 5.4, 5.6(d) , and 5.9(c), we learned that the early hours of the morning and the late hours of the midnight have particular characteristic. These time intervals have the highest number of pickups, prediction error and variance. Therefore, we will study time in different ways as we did with location features by dividing the data into time intervals.

Besides, we will try different types of training and testing data split. We want taking some months to be the training data and the next months as testing data as shown in Figure 6.1. The split of data enables us to understand if the taxi demand is a "circular procedure" with no variation over time.

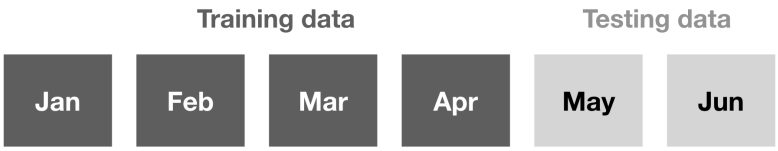


Figure 6.1: Dataset train-test split.

We also can consider the national calendar of social, political and recreation events, as well as important parties or sports events. These activities can be studied separately, especially when we have unscheduled events. Thus, based on the study results, taxis will directly be available for customers in

the right time and location. All these will help the taxi requesting services and applications to offer the best service for passengers by being available on time and faster than other companies. This will reduce the time to find passengers and the cost as well.

Bibliography

[1] New york city taxi trip duration. https://www.kaggle.com/c/nyc-taxi-trip-duration/data. Date accessed: 01.03.2020.

[2] Anaconda, Inc. (previously Continuum Analytics). Anaconda: Python data science platform. https://www.anaconda.com/, Programming language, machine learning, data science. Stable release: 11 March 2020. Written in Python.

[3] Mariette Awad and Rahul Khanna. Support vector regression. In *Efficient Learning Machines*, pages 67–80. Springer, 2015.

[4] A. Brabazon and M. O'Neill. *Natural Computing in Computational Finance*. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2008.

[5] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] Jason Brownlee. How to visualize time series residual forecast errors with python. https://machinelearningmastery.com/visualize-time-series-residual-forecast-errors-with-python/, machine learning mastery, 2017.

[8] Christine Cheng, Ravi Jain, and Eric van den Berg. Location prediction algorithms for mobile wireless systems. In *Wireless internet handbook: technologies, standards, and application*, pages 245–263. 2003.

[9] Chris Piech. Based on a handout by Andrew Ng. Stanford CS221. K means. https://stanford.edu/~cpiech/cs221/handouts/kmeans.html.

[10] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[11] Omri Goldstein. Dynamics of new york city animation, www.kaggle.com, 2017.

[12] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[13] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[14] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[15] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[16] Vinícius Gonçalves Maltarollo, Káthia Maria Honório, and Albérico Borges Ferreira da Silva. Applications of artificial neural networks in chemical problems. *Artificial neural networks-architectures and applications*, pages 203–223, 2013.

[17] Naoto Mukai and Naoto Yoden. Taxi demand forecasting based on taxi probe data by neural network. In *Intelligent Interactive Multimedia: Systems and Services*, pages 589–597. Springer, 2012.

[18] K-R Müller, Alexander J Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. Predicting time series with support vector machines. In *International Conference on Artificial Neural Networks*, pages 999–1004. Springer, 1997.

[19] Andrew Ng. Stanford, machine learning, online course. https://www.coursera.org/learn/machine-learning/. Date accessed: 30.06.2020.

[20] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[21] Willi Richert. *Building machine learning systems with Python*. Packt Publishing Ltd, 2013.

[22] Tijl Kindt Samuel Daulton, Sethu Raman. Nyc taxi data prediction. https://sdaulton.github.io/TaxiPrediction/. Date accessed: 23.10.2019.

[23] Scikit Learn. Cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html. Date accessed: 01.06.2020.

[24] Scikit Learn. Random forests. ensemble methods. https://scikit-learn.org/stable/modules/ensemble.html, Date accessed: 01.06.2020.

[25] Rana singh. Taxi demand prediction in new york city, medium.com, 2019.

[26] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[27] Wikipedia. Neighborhoods in new york city.

[28] Runmin Xu et al. *Machine learning for real-time demand forecasting*. PhD thesis, Massachusetts Institute of Technology, 2015.

[29] Gustav Zander. Predicting taxi passenger demand using artificial neural networks, 2017.

[30] Kai Zhao, Denis Khryashchev, Juliana Freire, Claudio Silva, and Huy Vo. Predicting taxi demand at high spatial resolution: Approaching the limit of predictability. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 833–842. IEEE, 2016.

[31] Jinming Zou, Yi Han, and Sung-Sau So. Overview of artificial neural networks. In *Artificial Neural Networks*, pages 14–22. Springer, 2008.