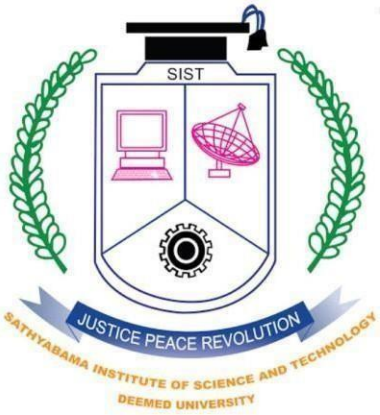
INTERDISCIPLINARY PROJECT REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

Lingampalli Bharath Sai
(40110674)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12 B Status
by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **LINGAMPALLI BHARATH SAI (40110674)** who carried out the project entitled "**Prediction of Cab Fare using Machine Learning**" under my supervision from FEB 2023 to APRIL 2023.

Internal Guide
Dr. J. Jeslin Shanthamalar, M.E., Ph.D.,

Head of the Department
Dr. L. Lakshamnan, M.E., Ph.D.,

Submitted for Viva voice Examination held on _____

Internal Examiner **External Examiner**

DECLARATION

I am **Lingampalli Bharath Sai (40110674)** hereby declare that the Report Entitled as "**Cab Fare Prediction Using Machine Learning**" done by me under the guidance of **Dr. J. Jeslin Shanthamalar, M.E., Ph.D., (Internal)** and **Dr. Jai Prakash Netha (External)** at "**TEAM SMARTBRIDGE**" is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr.L. Lakshmanan, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. J. Jeslin Shanthamalar, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE

ABSTRACT

For predicting the longer-term events predictive analysis use data which is archive. For capturing the trends which are important mathematical model is used from past data. The model then uses present data to predict the longer term or to derive actions to require optical outcomes tones of appreciation in recent time for predictive analytics thanks to development in support technology within areas of massive data in machine learning. Many industries use predictive analytics for making accurate forecast like giving the amount of fare for the ride within the city. These resource planning are enabled by the forecast as an example, cab fare can be predicted more accurately. A lot of factors are taken into consideration for a taxi start-up company. This project tries to know the patterns and use different methods for fare prediction. This project is developed for predicting the cab fare amount within a certain city. The project involves different steps like training, testing by using different variables like pickup, drop-off location for predicting cab fare.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

For prediction using systems machine learning is widely used all over the work. There are different types of app machines using machine learning for prediction some of them are supervised, unsupervised learning. In problems associated with business needs machine learning can also be called as predictive. Machine learning derived into different. Supervised learning is one of the most commonly used learning in machine learning, in order to train our data here, we need some factors like the dataset and we have to use algorithms in order to predict the output of the program which we are doing. If we do not have a data set to train our model then the output which we get is less reliable and the fare will not be accurate. Unsupervised learning is like learning without the help of any dataset or external factor, it automatically selects a suitable algorithm and tries to predict the output which will be accurate most of the time. Because of computing technologies the machine learning which we are looking at today are not at all similar to the machine learning which was used before sometime. This specifically proves that systems can do the tasks which were assure to be done by humans only

Machine learning can also be used in the domain of Artificial intelligence. It can am be used in data mining. We can perform many things in affordable price and also can do many complex things easily with the help of the advance machine learning. It is mainly used to make accurate predictions by using different range of methods and algorithms it provides to the people. So it is widely used in many companies because we use the concept of machine learning daily in our day to day lives and also it will be used in future also as the scope of it is very high.

The main contributions of this project therefore are:

- Data Analysis
- Dataset Preprocessing
- Training the Model
- Testing of Dataset

1.1 PROBLEM STATEMENT

The objective of this project is to predict Cab Fare amount. You are a cab rental start-up company. You have successfully run the pilot project and now want to launch your cab service across the country. You have collected the historical data from your pilot project and now have a requirement to apply analytics for fare prediction. You need to design a system that predicts the fare amount for a cab ride in the city.

CHAPTER 2

AIM AND SCOPE OF THE PRESENTATION INVESTIGATION

2.1 AIM AND SCOPE

A large part of the literature focuses primarily on spatial-temporal analysis and demand & price forecasting of the taxi trips. Popular destinations in New York City were uncovered by detecting popular drop-off locations using density-based clustering (Garcia, Avendano, & Vaca, 2018).

Negative binomial regression model and Spatial association were implemented to analyse the travel patterns in urban areas for effective investment decisions in rapid transit infrastructure and service (Hochmair, 2016) and kernal density analysis was used to detect demand fluctuations (Markou, Rodrigues, & Pereira, 2017). The travel time estimation problem was addressed with a big data-driven approach using a simple baseline for Travel Time Estimation (Wang, Kuo, Kifer, & Li, 2014). With the aid of the Semi-Markov process and approximate dynamic programming (ADP) approach, time of the pricing was determined with supply and demand fluctuations of taxis and resulted in increasing the market revenue by 10% (Qian & Ukkusuri, 2017). Hierarchical density-based spatial clustering (HDBSCAN), Random Swap clustering and sequential pattern mining were implemented to build a system that delivers traffic insights and recommendations to help taxi drivers with useful guidelines (Ibrahim & Shafiq, 2019). A taxi searching algorithm with an accuracy of 97.59% accuracy was built using distributed coordination and clustering to minimize the time of taxi reaching the passengers (Agrawal, Raychoudhury, Saxena, & Kshemkalyani, 2018). The problem of predicting the number of taxis required in zones at various times was solved using STVec (smooth transaction vector error correction model) and multi- outputs support vector regression (MSVR) model (Zhou, Wu, Wu, Chen, & Li, 2015). Visual Exploration is a challenge for big Spatio-Temporal urban taxi data, this was overcome by building a model that uses an adaptive level-of-detail rendering strategy to create a visualization that is clutter-free

for results that are large (Ferreira, Poco, Vo, Freire, & Silva, 2013). The Markov predictor model built for prediction of taxi demand was 89% accurate and better by 11% compared to neural network predictors (Zhao, Khryashchev, Freire, Silva, & Vo, 2016). A platform for distributed spatiotemporal analytics was built to deal with the heterogeneous spatiotemporal dataset (Deva, Raschke, Garzon, & Kupper, 2017) and a simulation combined with effective indexing scheme and parallelization was built to get a scalable approach (Ota, Vo, Silva, & Freire, 2015). Various deep learning models such as Long short-term memory (LSTM) neural networks, adaptive boosting, decision tree regression model was built to detect pattern variation in taxi trips and the LSTM model was proven to be the reliable model (Najafabadi & Allahviranloo, 2018). The implications of social factors on driving behavior of drivers were proved to be dependent on each other based on correlation and social influence theory (Xu, et al., 2017). The yellow taxi demand forecasting was achieved through spatiotemporal autoregressive (STAR) model which outperformed vector autoregressive (VAR) model (Safikhani, Kamga, Mudigonda, Faghih, & Moghimi, 2018). An ensemble of a tree-based model and a

Long Short-Term Memory (LSTM) were implemented to predict taxi demand at

LaGuardia airport and Long Short-Term Memory (LSTM) Recurrent Neural

Network achieved an MAE of 48.1 and tree ensemble model achieved MAE of 56.9 (Coviensky, Katiyal, Agrawal, & Geary, 2017). For optimal assignment and incentive design in the taxi group ride, a Heuristic algorithm was implemented by which 47% of taxi trip mileage was saved (Qian, Zhang, Ukkusuri, & Yang, 2017). To explore the dynamics of human mobility patterns, interactive visual views are were designed through tensor decomposition (Shi, Lv, Seng, Xing, & Chen, 2019).

CHAPTER 3

EXPERIMENTAL MATERIALS OR METHODS AND ALGORITHMS

3.1 SOFTWARE REQUIREMENTS

Operating system	:	Windows 10
Languages used	:	Python
Python version	:	3.5 or 3.6
Notebooks	:	Jupyter Notebook
Emulators	:	No emulators used
Software Libraries	:	Pandas, Matplotlib, NumPy, Seaborn,Math,Sklearn.

3.1.1 OPERATING SYSTEM:

WINDOWS 10:

Windows 10 is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems. It is the successor to Windows 8.1, and was released to manufacturing on July 15, 2015, and broadly released for retail sale on July 29, 2015.Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users, in addition to additional test builds of Windows 10 which are available to Windows Insiders. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

3.1.2 JUPYTER NOTEOOK:

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text.

Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

3.1.3 PANDAS & NUMPY:

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. It offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

3.1.4 MATPLOTLIB & SEABORN:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Seaborn is a Python data visualization library based on matplotlib. It provides high- level interface for drawing attractive and informative statistical graphics.

3.2 HARDWARE REQUIREMENTS

There are big data that require big hardware.
Learning about big machine learning requires big data and big hardware.
Processor – Intel Xeon E2630 v4 – 10 core processor, 2.4 GHz with Turboboost up to 3.1 GHz.
25 MB Cache or Motherboard – ASRock
RAM – 128 GB DDR4 2133 MHz
TB Hard Disk (7200 RPM) + 512 GB SSD
GPU – NVidia TitanX Pascal (12 GB VRAM)

Intel Heatsink to keep temperature under control
Disk space: 2 GB
Operating systems: Windows 10, macOS, and Linux
Python* versions: 3.5.x, 3.6.x
64-bit, x86 desktops or laptops with dedicated Nvidia graphics card

3.3 ALGORITHMS USED

3.3.1 LINEAR REGRESSION

It is a supervised machine learning algorithm. It is mostly used after the step of correlation. If we want to predict the value of an y variable by using the value of another variable then we can use Linear algorithm. Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use cost function.

Cost function- The different values for weights or coefficient of lines (a_0, a_1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line. Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing.

We can use the cost function to find the accuracy of the **mapping function**, which maps the input variable to the output variable. This mapping function is also known as **Hypothesis function**.

For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values.

3.3.2 RANDOM FOREST

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. The reason why we choose Random Forest algorithm over other machine learning algorithms is that Random Forest has the following advantages:

- It runs efficiently on large data bases and is unexcelled in accuracy among current algorithms.
- It can fix well with errors in class population unbalanced data sets.
- It supports an effective method for estimating missing data, which can maintain accuracy even in the case that a large scale of the data is missing.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Random forest belongs to supervised learning algorithm, is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or means prediction of the individual trees. The decision tree is a tree structure (which can be a binary tree or a non-binary tree). Each of its non-leaf nodes corresponds to a test of a feature, each branch representing the output of the feature attribute over a range of values, and each leaf node storing a category. The decision

tree starts with a root node, tests the corresponding feature attributes in the category to be classified, and output branches are selected according to their values until the leaf node is reached, finally the category stored by the leaf node is regards as the decision result. A random forest is a collection of decision trees in which each decision tree is unrelated. Selection metrics we used for splitting attributes in the decision tree is Gini index, and the number of levels in each tree branch depends on the algorithm parameter d.

3.4 SYSTEM ARCHITECTURE

FIG 3.4.1: Architecture diagram

3.4.1 METHODOLOGY

According to industry standards, the process of Data Analysing mainly includes 6 main steps and this process is abbreviated as CRISP DM Process, which is Cross-Industry Process for Data Mining. And the Six main steps of CRISP DM Methodology for developing a model are:

1. Business understanding
2. Data understanding
3. Data preparation/Data Pre-processing
4. Modelling
5. Evaluation
6. Deployment

And in this project, all the above steps are followed to develop the model.

3.4.2 Business Understanding

It is important to understand the idea of business behind the data set. The given data set is asking us to predict fare amount. And it really becomes important for us to predict the fare amount accurately. Else, there might be great loss to the revenue of the firm. Thus, we must concentrate on making the model most efficient.

3.4.3 Data Understanding

To get the best results, to get the most effective model it is important to our data very well. Here, the given train data is a CSV file that consists 7 variables and 16067 Observation. A snapshot of the data provided.

fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.84161	40.712278	1
16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1
5.7	2011-08-18 00:35:00 UTC	-73.982738	40.76127	-73.991242	40.750562	2
7.7	2012-04-21 04:30:42 UTC	-73.98713	40.733143	-73.991567	40.758092	1
5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1
12.1	2011-01-06 09:50:45 UTC	-74.000964	40.73163	-73.972892	40.758233	1

Table 3.4.1: Train data

The different variables of the data are:

fare_amount : fare of the given cab ride.
pickup_datetime : timestamp value explaining the time of ride start.
pickup_longitude : a float value explaining longitude location of the ride start.
pickup_latitude : a float value explaining latitude location of the ride start.
dropoff_longitude : a float value explaining longitude location of the ride end**.**
dropoff_latitude : a float value explaining latitude location of the ride end.
Passenger_cont : an integer indicating number of passengers

Following table explains how the variables are categorized.

Independent Variables
pickup_datetime
pickup_longitude
pickup_latitude
dropoff_longitude
dropoff_latitude
passenger_count
dependent Variables
Fare_amount

Table 3.4.2: Independent variables and dependent variables

From the given train data, it is understood that, we have to predict fare amount, and other variables will help me achieve that, here pickup_latitude/longitude, dropoff_latitude/longitude this data are signifying the location of pick up and drop off. It is explaining starting point and end point of the ride. So, these variables are crucial for us. Passenger_count is another variable, that explains about how many people or passenger boarded the ride, between the pickup and drop off locations. And pick up date time gives information about the time the passenger is picked up and ride has started. But unlike pick up and drop off locations has start and end details both in given data. The time data has only started details and no time value or time related information of end of ride. So, during pre- processing of data we will drop this variable. As it seems the information of time is incomplete.

Also, there is a separate test data given, in the format of CSV file containing

9514 observations and 6 variables. All of them are the independent variables.

An in these data at the end we must predict the fare or the target variable. Following is a snap of the test data provided.

pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
2015-01-27 13:08:24 UTC	-73.97332	40.7638054	-73.9814301	40.7438355	1
2015-01-27 13:08:24 UTC	-73.9868622	40.7193832	-73.9988861	40.7392006	1
2011-10-08 11:53:44 UTC	-73.982524	40.75126	-73.979654	40.746139	1

Table 3.4.3: Test data

3.4.4 Data Preparation

The next step in the CRISP DM Process is, Data pre-processing. It is a data mining process that involves transformation of raw data into a format that helps us execute our model well. As, the data often we get are incomplete, inconsistent, and also may contain many errors. Thus, Data pre-processing is a generic method to deal with such issues and get a data format that is easily understood by

machine and that helps developing our model in best way. In this project also we have followed data pre-processing methods to rectify errors and issues in our data. And this is done by popular data pre-processing techniques, this are following below.

Note: I have removed the variable “Pick up date Time” as it is a timestamp value and it shows only the start time of pick-up time, whereas there is no drop off time, so in this data set it seems, it will have no impact in the target variable, and also it lead to redundancy and model accuracy issues, so I preferred to drop it.

3.4.5 Missing Value Analysis

Missing value is availability of incomplete observations in the dataset. This is found because of reasons like, incomplete submission, wrong input, manual error etc. These Missing values affect the accuracy of model. So, it becomes important to check missing values in our given data.

3.4.5.1 Missing Value Analysis in Given Data:

In the given dataset it is found that there are lot of values which are missing. It is found in the following types:

1. Blank spaces : Which are converted to NA and NaN in R and Python respectively for further operations
2. Zero Values : This is also converted to NA and Nan in R and python respectively prior further operations
3. Repeating Values : there are lots of repeating values in pickup_longitude, pickup_latitude, dropoff_longitude and dropoff_latitude. This will hamper our model, so such data is also removed to improve the performance.

Following the standards of percentage of missing values, we now must decide to accept a variable or drop it for further operations. Industry standards ask to follow following standards:

1. Missing value percentage < 30% : Accept the variable
2. Missing value percentage > 30 % : Drop the variable

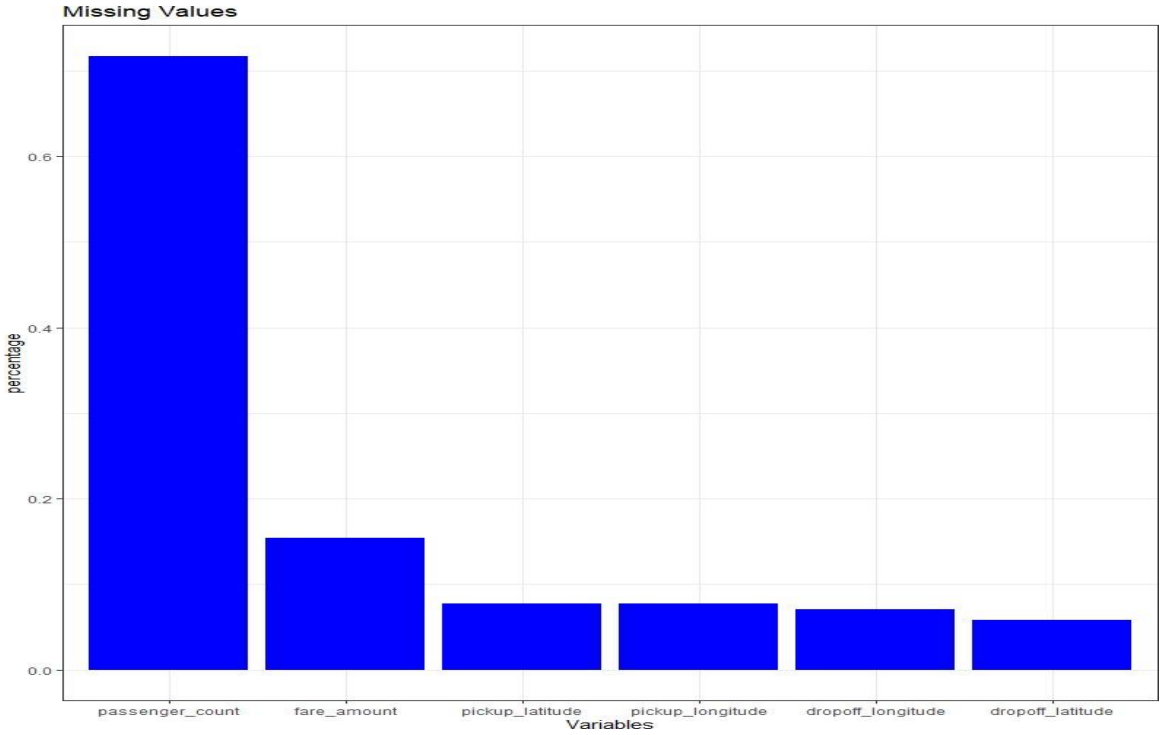


Fig 3.4.2 Missing values

It is found from the above graph plot that the there is no variable exceeding the 30% range so we not need to exclude any of our variable.

Impute the missing value:

After the identification of the missing values the next step is to impute the missing values. And this imputation is normally done by following methods.

1. Central Tendencies: by the help of Mean, Median or Mode
2. Distance based or Data mining method like KNN imputation
3. Prediction Based: It is based on Predictive Machine Learning Algorithm

To use the best method, it is necessary for us to check, which method predicts values close to the original data. And this done by taking a subset of data, taking an example variable and noting down its original value and the replacing that value with NA and then applying available methods. And noting down every value from

the above methods for the example variable we have taken, now we chose the method which gives most close value.

In this project, KNN imputation worked the best. So, I am using KNN method to impute missing Values.

	Variable	Missing Value count	Missing value Percentage
1	fare_amount	0	0
2	pickup_longitude	0	0
3	pickup_latitude	0	0
4	dropoff_longitude	0	0
5	dropoff_latitude	0	0
6	passenger_count	0	0

Table 3.4.4: Missing values after imputation with mean and mode.

3.4.5 Outlier Analysis

Outlier is an abnormal observation that stands or deviates away from other observations. These happens because of manual error; poor quality of data and it is correct but exceptional data. But it can cause an error in predicting the target variables. So, we must check for outliers in our data set and also remove or replace the outliers wherever required.

Outliers in this project:

In this dataset, I have found some irregular data, those are considered as outliers. These are explained below.

Fare_Amount :

I have always seen fare of a cab ride as positive, I have never seen any cab driver, giving me money to take a ride in his cab. But in this dataset, there are many instances where fare amount is negative. Given below are such instances:

fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
-2.9	2010-03-09 23:37:10 UTC	-73.7895	40.6435	-73.7887	40.64195	1
-2.5	2015-03-22 05:14:27 UTC	-74	40.72063	-73.9998	40.72054	1
-3	2013-08-30 08:57:10 UTC	-73.9951	40.74076	-73.9959	40.74136	4

Table 3.4.5: Fare outliers

3.4.5 Passenger_count:

I have always found a cab with 4 seats to maximum of 8 seats. But in this dataset, I have found passenger count more than this, and in some cases a large number of values. This seems irregular data, or a manual error. Thus, these are outliers and needs to be removed. Few instances are following.

Fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
4.9	2010-07-12 09:44:33 UTC	-73.9832	40.73466	-73.9913	40.73892	456
6.1	2011-01-18 23:48:00 UTC	-74.0066	40.73893	-74.0108	40.71791	5334
8.5	2013-06-18 10:27:05 UTC	-73.9921	40.7642	-73.973	40.7627	535
8.1	2009-08-21 19:35:05 UTC	-73.9609	40.76156	-73.9763	40.74836	354

Table 3.4.6: Passenger_count outlier

3.4.6 Location points:

When I checked the data, it is found that most of the longitude points are within the 70 degree and most of the latitude points are within the 40 degree. This symbolizes all the data belongs to a specific location and a specific range. But I also found some data which consists location points too far from the average location point's range of 70 Degree Longitude and 40 Degree latitude. It seems

these far point locations are irregular data. And I consider this as outlier. I have collected the maximum and minimum values of location point as a reference to identify the outliers.

Variable	Minimum Value	Maximum Value
pickup_longitude	-74.43823	40.76613
pickup_latitude	-74.00689	401.08333
dropoff_longitude	-74.22705	40.80244
dropoff_latitude	-74.00638	41.36614

Table 3.4.7: Range of variables

Following are the instances, where the values exceeds far from average location points. And I consider this as Outliers.

Fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
15	40.72913	-74.0069	40.76337	-73.9616	1
52	40.73688	-74.0062	40.73689	-74.0064	6
15.5	40.76442	-73.9929	40.80244	-73.9507	1
6.5	40.74826	-73.9918	40.74037	-73.979	1
3.3	-73.9472	401.0833	-73.9514	40.77893	1

Table 3.4.8: Location Outliers

All these outliers mentioned above happened because of manual error, or interchange of data, or may be correct data but exceptional. But all these outliers can hamper our data model. So, there is a requirement to eliminate or replace such outliers. And impute with proper methods to get better accuracy of the model. In this project, I used mode method to impute the outliers in passenger count and mean for location Points and fare amount.

3.5 FEATURE SELECTION

Sometimes it happens that, all the variables in our data may not be accurate enough to predict the target variable, in such cases we need to analyse our data, understand our data and select the dataset variables that can be most useful for our model. In such cases we follow feature selection. Feature selection helps by reducing time for computation of model and reduces the complexity of the model.

After understanding the data, pre-processing and selecting specific features, there . process to engineer new variables if required to improve the accuracy of the model.

In this project the data contains only the pickup and drop points in longitude and latitude. The fare_amount will mainly depend on the distance covered between these two points. Thus, we must create a new variable prior further processing the data. And in this project the variable I have created is Distance variable (dist), which is a numeric value and explains the distance covered between the pickup and drop of points. After researching I found a formula called the haversine formula, that determines the distance between two points on a sphere based on their given longitudes and latitudes. This formula calculates the shortest distance between two points in a sphere.

The function of haversine function is described, which helped me to engineer our new variable, Distance.

Used in Python :

```
# haversine function
  def haversine(lat1, lon1, lat2, lon2, to_radians=True,
    earth_radius=6371): if to_radians:
      lat1, lon1, lat2, lon2 = np.radians([lat1, lon1, lat2, lon2])
 a = np.sin((lat2-lat1)/2.0)**2 + \
      np.cos(lat1) * np.cos(lat2) * np.sin((lon2-
      lon1)/2.0)**2 return earth_radius * 2*np.arcsin(np.sqrt(a))
```

After executing the haversine function in our project, I got new variable distance and some instances of data are mentioned below.

Fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_ count	Dist
4.5	-73.98172	40.72132	-73.98011	40.71228	1	1.0156499
16.9	-74.01605	40.7113	-73.97927	40.782	1	8.4595997

5.7	-73.98274	40.76127	-73.99124	40.75056	2	1.3910818
7.7	-73.98713	40.73314	-73.99157	40.75809	1	2.8024061
5.3	-73.9681	40.76801	-73.95665	40.78376	1	2.0013963

Table: 3.5.1: Engineering with new variable distance

3.6 Correlation Analysis:

In some cases, it is asked that models require independent variables free from collinearity issues. This can be checked by correlation analysis for the categorical variables and continuous variables. Correlation analysis is a process that is defined to identify the level of relation between two variables.

In this project, our Predictor variable is continuous, so we will plot a correlation table that will predict the correlation strength between independent variables and the ‘fare_amount’ variable

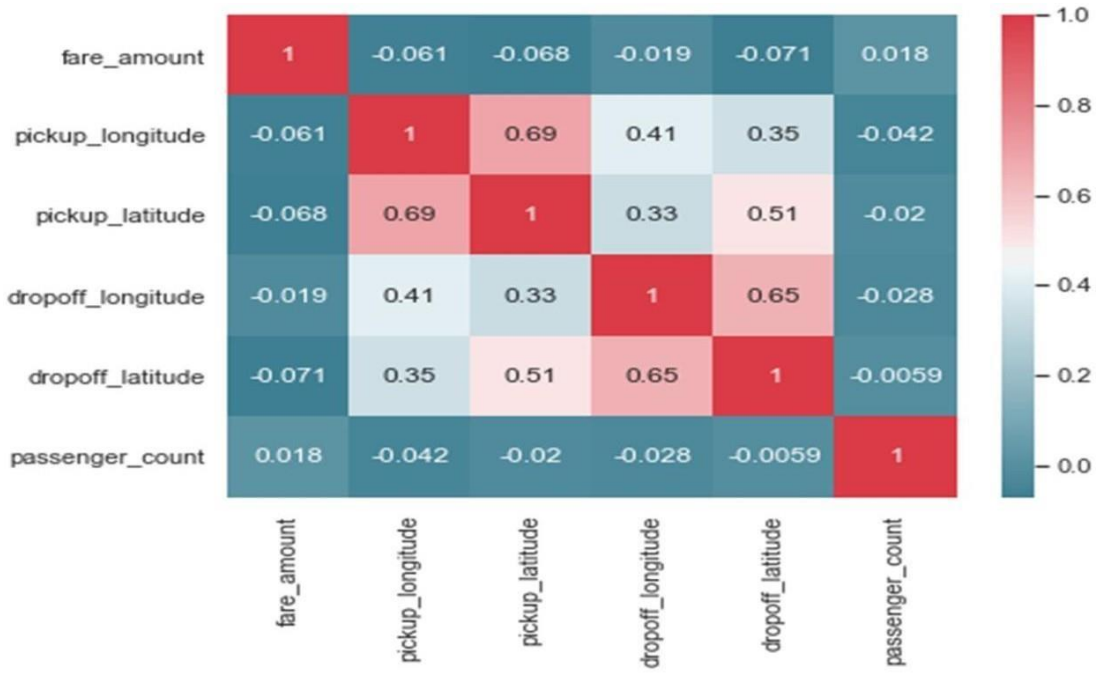


Fig 3.5.1: Correlation matrix

- From the above plot it is found that most of the variables are highly correlated with each other, like fare amount is highly correlated with distance variable.
- All the red charts represent that variables are highly correlated. And as there is some not red and blue charts, which represents negative correlation, it can

be summarized that our dataset has strong or highly positive correlation between the variables.

- Because all the variables are numeric the important features are extracted using the correlation matrix. All the variables are important for predicting the fare_amount since none of the variables have a high correlation factor (considering the threshold as 0.9), so all the variables for model building are kept.

3.7 MODEL DEVELOPMENT

After all the above processes the next step is developing the model based on our prepared data.

In this project we got our target variable as “fare_amount”. The model has to predict a numeric value. Thus, it is identified that this is a Regression problem statement. And to develop a regression model, the various models that can be used are Random Forest and Linear Regression.

3.7.1 RANDOM FOREST

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.* The reason why we choose Random Forest algorithm over other machine learning algorithms is that Random Forest has the following advantages:

- It runs efficiently on large data bases and is unexcelled in accuracy among current algorithms;
- It can fix well with errors in class population unbalanced data sets;
- It supports an effective method for estimating missing data, which can maintain accuracy even in the case that a large scale of the data is missing

- It is capable of handling large datasets with high dimensionality.

Random Forest

```
In [29]: #Random Forest
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.model_selection import train_test_split

In [30]: # Divide the data into train and test
         train1, test1 = train_test_split(Train_Data, test_size=0.2)

In [31]: RF_model = RandomForestRegressor(n_estimators = 100).fit(train1.iloc[:, 1:7], train1.iloc[:,0])

In [32]: RF_model
Out[32]: RandomForestRegressor()

In [33]: RF_Predictions = RF_model.predict(test1.iloc[:, 1:7])
```

Fig 3.7.1 Dividing the data set and applying random forest algorithm

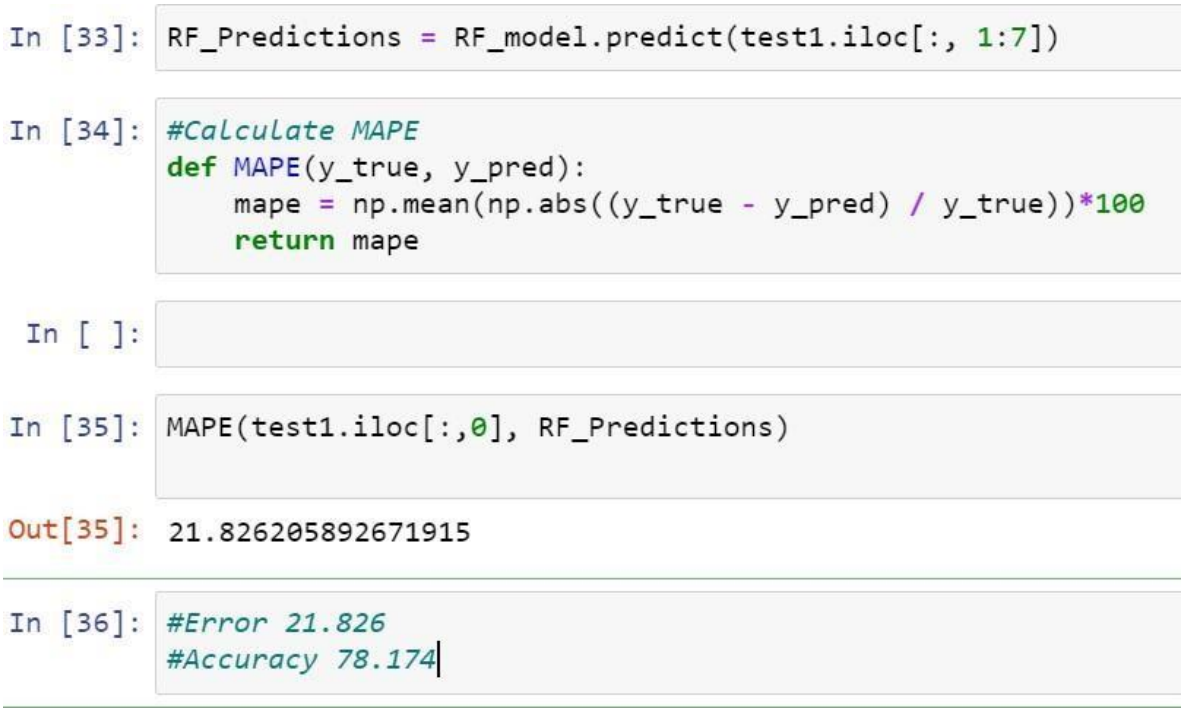


Fig 3.7.2 calculating mean absolute percentage error

3.7.2 Linear Regression

It is a supervised machine learning algorithm. It is mostly used after the step of correlation. If we want to predict the value of an y variable by using the value of

another variable then we can use Linear algorithm. Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

Linear Regression

```
#Combine all the values in one array
values=['fare_amount', 'pickup_longitude','pickup_latitude', 'dropoff_longitude', 'dropoff_latitude', 'dist']
```

```
linear_Data = Train_Data[values]
```

```
#This function is developed to get columns for specific passenger count. The idea is developed from R linear regression fit,
#which explains all the passenger count individualy contributes in the model

cat_names = ['passenger_count']
for i in cat_names:
    temp = pd.get_dummies(Train_Data[i], prefix= i)
    linear_Data = linear_Data.join(temp)
```

```
linear_Data.shape
```

```
(469296, 12)
```

Fig 3.7.3: Applying linear regression

```
linear_Data.head()
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	dist	passenger_count_1	passenger_count_2	passenger_count_3	pa
0	4.5	-73.981709	40.721319	-73.980181	40.712278	1.013527	1	0	0	
1	16.9	-74.016048	40.711303	-73.979268	40.782004	8.450134	1	0	0	
2	5.7	-73.982738	40.761270	-73.991242	40.750562	1.389525	0	1	0	
3	7.7	-73.987130	40.733143	-73.991567	40.758092	2.799270	1	0	0	
4	5.3	-73.968095	40.768008	-73.956655	40.783762	1.999157	1	0	0	

```
#Spliting the newly created data set with passenger count dummies
train1, test1 = train_test_split(linear_Data, test_size=0.2)
```

```
#Import libraries for LR
import statsmodels.api as sm
# Train the model using the training sets
model = sm.OLS(train1.iloc[:, 0].astype(float), train1.iloc[:, 1:12].astype(float)).fit()
```

Fig 3.7.4: Splitting newly created data set

```
# make the predictions by the model
predictions_LR = model.predict(test1.iloc[:,1:12])
```

```
#Calculate MAPE
MAPE(test1.iloc[:,0], predictions_LR)
```

```
28.053010468557822
```

```
#Error 28.174474283418512
#Accuracy 71.8255
```

Fig 3.7.5: Finding mean absolute percentage error For Linear regression

CHAPTER 4

RESULT AND DISCUSSION

So, now we have developed few models for predicting the target variable, now the next step is to identify which one to choose for deployment. To decide these according to industry standards, we follow several criteria. Few among this are, calculating the error rate, and the accuracy. MAE and MAPE is used in our project. MAE or Mean Absolute Error, it is one of the error measures that is used to calculate the predictive performance of the model. In this project we will apply this measure to our models

Method	Mae Error(in Percentage)
Random Forest	20.2135
Linear Regression	28.1744

Table 5.1: Mean absolute error

The second matrix to identify or compare for better model is Accuracy. It is the ratio of number of correct predictions to the total number of predictions made.
Accuracy = number of correct predictions / Total predictions made.
It can also be calculated from MAE as Accuracy = 1- MAPE.

Method	Accuracy (in Percentage)
Random Forest	79.7864
Linear Regression	71.8255

Table 5.2: Accuracy

Prediction on original data set

Prediction on original test data

```
In [48]: pred=(pd.read_csv(r'C:\MAIN PROJECT\test\test.csv', header = 0 )).drop(columns="pickup_datetime")

In [49]: #create Dist variable
         pred['dist'] = \
             haversine( pred['pickup_latitude'], pred['pickup_longitude'],
                        pred['dropoff_latitude'], pred['dropoff_longitude'])

         pred['fare_amount']=0
         pred['passenger_count']=pred['passenger_count'].astype('category')

In [50]: # Build model on the entire Train data
         RF_model = RandomForestRegressor(n_estimators = 10).fit(Train_Data.iloc[:, 1:7], Train_Data.iloc[:,0])

         #predict value
         pred['fare_amount'] = RF_model.predict(pred.iloc[:, 0:6])
```

Fig 5.1 Predicting on test data set

The original test data set does not contain fare amount. By means of random forest algorithm we will calculate fare of the test data set

```
In [51]: pred.head()
```

Out[51]:

	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	dist	fare_amount
0	-73.973320	40.763805	-73.981430	40.743835	1	2.323259	8.24
1	-73.986862	40.719383	-73.998886	40.739201	1	2.425353	8.75
2	-73.982524	40.751260	-73.979654	40.746139	1	0.618628	5.33
3	-73.981160	40.767807	-73.990448	40.751635	1	1.961033	7.74
4	-73.966046	40.789775	-73.988565	40.744427	1	5.387301	16.21

```
In [52]: #write output to csv

         pred.to_csv("C:\MAIN PROJECT\Predicted_Values.csv", index = False)
```

Fig 5.2 Saving the data in csv file

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

CONCLUSION

Prediction of cab fare is one of the essential usages in automobile industry. The purpose study is to predict the cab fare using random forest and linear regression algorithms. Thus, the predicting the cab fare we have achieved our objective of our project. The accuracy for random forest is 79.78%. The accuracy for linear regression is 71.82%

The quality of a regression model depends on the matchup of predictions against actual values. In regression problems, the dependent variable is continuous. In classification problems, the dependent variable is categorical. Random Forest can be used to solve both regression and classification problems. Decision trees are nonlinear; unlike linear regression, there is no equation to express the relationship between independent and dependent variables. Out of the three models left, Random Forest is the best model as it has the lowest RMSE score and highest R-Squared score, which explains the highest variability and tells us how well the model fits in this data.

FUTURE WORK

As is known, with an increase in the number of features; underlying equations become a higher-order polynomial equation, and it leads to overfitting of the data. Generally, it is seen that an overfitted model performs worse on the testing data set, and it is also observed that the overfitted model performs worse on additional new test data set as well. A kind of normalized regression type -Ridge Regression may be further considered.

APPENDIX

SOURCE CODE:

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import os import pandas as pd import
numpy as np import matplotlib.pyplot as
plt from scipy.stats import
chi2_contingency import seaborn as sns
from random import randrange, uniform


# In[2]:


Train_Data=(pd.read_csv(r'C:\MAIN   PROJECT\train.csv\train.csv',nrows=500000,
header = 0 )).drop(columns=["pickup_datetime","key"])


# In[3]:


Train_Data.head()


# In[4]:


Train_Data['fare_amount']=          Train_Data['fare_amount'].apply(pd.to_numeric,
errors='coerce')
Train_Data['fare_amount']= Train_Data['fare_amount'].replace({0:np.nan})
Train_Data['passenger_count']=Train_Data['passenger_count'].fillna(0)
Train_Data['passenger_count']= Train_Data['passenger_count'].astype(int)
Train_Data['passenger_count']=Train_Data['passenger_count'].replace({0:
np.nan})
Train_Data['pickup_longitude']= Train_Data['pickup_longitude'].replace({0:np.nan})
Train_Data['pickup_latitude']= Train_Data['pickup_latitude'].replace({0:np.nan})
Train_Data['dropoff_longitude']=
Train_Data['dropoff_longitude'].replace({0:np.nan})
Train_Data['dropoff_latitude']= Train_Data['dropoff_latitude'].replace({0:np.nan})
```

```
# In[5]:

Train_Data=Train_Data[np.logical_and(Train_Data['pickup_longitude'] !=
Train_Data['dropoff_longitude'],
                                     Train_Data['pickup_latitude']                                    !=
Train_Data['dropoff_latitude'])]


# In[6]:


Train_Data.head()


# In[7]: Train_Data.shape


# In[8]:


sns.set(style='darkgrid',palette='Set1') #


In[9]:


plt.figure(figsize=(20,20)) plt.subplot(321)
_ = sns.distplot(Train_Data['fare_amount'],bins=50) plt.subplot(322)
_ = sns.distplot(Train_Data['pickup_longitude'],bins=50) plt.subplot(323)
_ = sns.distplot(Train_Data['pickup_latitude'],bins=50) plt.subplot(324)
_ = sns.distplot(Train_Data['dropoff_longitude'],bins=50) plt.subplot(325)
_ = sns.distplot(Train_Data['dropoff_latitude'],bins=50) plt.show()


# In[ ]:




# # Missing Value Analysis
# In[10]:


#calculate missing values

missing_val = pd.DataFrame(Train_Data.isnull().sum())
```

```
#print(missing_val)

#Reset index
missing_val = missing_val.reset_index()
#print(missing_val)

#Rename variable
missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'count'})
#print(missing_val)

#Calculate percentage
missing_val['Missing_percentage'] = (missing_val['count']/len(Train_Data)*100)
#print(missing_val)

#sort in descending order
missing_val = missing_val.sort_values('Missing_percentage', ascending =
False).reset_index(drop = True) print(missing_val)



# In[11]:


Train_Data.describe()


# # Missing Value Imputation

# In[12]:


#As it is found Mean is very close to original method we will proceed with
imputation via mean Train_Data['fare_amount'] =
Train_Data['fare_amount'].fillna(Train_Data['fare_amount'].mean())
Train_Data['pickup_longitude']=
Train_Data['pickup_longitude'].fillna(Train_Data['pickup_longitude'].mean())
Train_Data['pickup_latitude']=
Train_Data['pickup_latitude'].fillna(Train_Data['pickup_latitude'].mean())
Train_Data['dropoff_longitude']=
Train_Data['dropoff_longitude'].fillna(Train_Data['dropoff_longitude'].mean())
Train_Data['dropoff_latitude']=
Train_Data['dropoff_latitude'].fillna(Train_Data['dropoff_latitude'].mean())


#And for category variables imputation is done with mode
Train_Data['passenger_count'] =
Train_Data['passenger_count'].fillna(int(Train_Data['passenger_count'].mode()))
```

In[13]:

Train_Data.head() #

In[14]:

Train_Data.isnull() #

In[15]:

#Imputing the NAs in target variables may hamper the model, so it is preferred to remove NA rows of the data Train_Data=Train_Data.dropna() # In[16]:

#conert into proper data type
convert_dic={'fare_amount' : 'float','passenger_count': 'int'}
Train_Data=Train_Data.astype(convert_dic)

In[17]:

Train_Data.shape

Outlier Analysis

In[18]:

#save the data with in another place with different name df

= Train_Data.copy()

Train_Data = Train_Data.copy() #

In[19]:

irregular fare_amount are converted to NA

```
Train_Data.loc[Train_Data['fare_amount']<0 , 'fare_amount']=np.nan
Train_Data.loc[Train_Data['fare_amount'] > 30, 'fare_amount']=np.nan
Train_Data=Train_Data.dropna()


# In[20]:


#irregular passenger counts or those wgich are greater then 8 convertet to NaN
Train_Data.loc[Train_Data['passenger_count'] > 8,'passenger_count'] = np.nan


# In[21]:


#save numeric data names
coutliers = [ 'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
'dropoff_latitude'] for
list in coutliers:
    #Detect and replace with NA
    #Extract quartiles
    q75, q25 = np.percentile(Train_Data[list], [75 ,25])

    #Calculate IQR iqr
    = q75 - q25

    # #Calculate inner and outer fence
    minimum = q25 - (iqr*1.5)
    maximum = q75 + (iqr*1.5)

    # #Replace with NA
    Train_Data.loc[Train_Data[list] < minimum,list] = np.nan
    Train_Data.loc[Train_Data[list] > maximum,list] = np.nan

    # #Calculate missing value
    missing_val = pd.DataFrame(Train_Data.isnull().sum())


# In[22]:
#As Mean is the best method, we impute missing values/ in this case outlier values
with mean

Train_Data['pickup_longitude'] =
Train_Data['pickup_longitude'].fillna(Train_Data['pickup_longitude'].mean())
Train_Data['pickup_latitude'] =
```

```
Train_Data['pickup_latitude'].fillna(Train_Data['pickup_latitude'].mean())
Train_Data['dropoff_longitude'] =
Train_Data['dropoff_longitude'].fillna(Train_Data['dropoff_longitude'].mean())
Train_Data['dropoff_latitude'] =
Train_Data['dropoff_latitude'].fillna(Train_Data['dropoff_latitude'].mean())

#imputed with mode for categorical variables
Train_Data['passenger_count'] =
Train_Data['passenger_count'].fillna(int(Train_Data['passenger_count'].mode()))


# In[23]:


#convert the data type of categorical variable passenger count
Train_Data['passenger_count']=Train_Data['passenger_count'].astype('int')
Train_Data['passenger_count']=Train_Data['passenger_count'].astype('category')


# # Feature Selection

# In[24]:


#haversine function def haversine(lat1, lon1, lat2, lon2, to_radians=True,

earth_radius=6371):

    if to_radians:
        lat1, lon1, lat2, lon2 = np.radians([lat1, lon1, lat2, lon2])

    a = np.sin((lat2-lat1)/2.0)**2 +         np.cos(lat1) * np.cos(lat2) * np.sin((lon2-
lon1)/2.0)**2

    return earth_radius * 2 * np.arcsin(np.sqrt(a)) #


In[25]:


Train_Data['dist'] =     haversine( Train_Data['pickup_latitude'],
Train_Data['pickup_longitude'],
                Train_Data['dropoff_latitude'], Train_Data['dropoff_longitude'])
# In[26]:


##Correlation analysis #Correlation
plot
```

```
numeric=['fare_amount','pickup_longitude','pickup_latitude','dropoff_longitude','dro
poff_latitude', 'dist']
Train_Data_corr = Train_Data.loc[:,numeric] #
```

In[27]:

```
#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix corr
= Train_Data_corr.corr()
print(corr)

#Plotted using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
cmap=sns.diverging_palette(220,      10,      as_cmap=True),
square=True, ax=ax)
```

In[28]:

```
#eliminate all data with the pickup and drop location points as same

Train_Data=Train_Data[np.logical_and(Train_Data['pickup_longitude'] !=
Train_Data['dropoff_longitude'],
                                     Train_Data['pickup_latitude']                                    !=
Train_Data['dropoff_latitude'])]
```

Model Development

Random Forest

In[29]:

```
#Random Forest from sklearn.ensemble import
RandomForestRegressor from sklearn.model_selection
import train_test_split
```

In[30]:

```
# Divide the data into train and test
```

train1, test1 = train_test_split(Train_Data, test_size=0.2) #

In[31]:

RF_model = RandomForestRegressor(n_estimators = 100).fit(train1.iloc[:, 1:7], train1.iloc[:,0])

In[32]:

RF_model

In[33]:

RF_Predictions = RF_model.predict(test1.iloc[:, 1:7]) #

In[34]:

```
#Calculate     MAPE     def
MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100 return
    mape
```

In[ ]:

In[35]:

MAPE(test1.iloc[:,0], RF_Predictions)

In[36]:
#Error 21.826
#Accuracy 78.174

Linear Regression

In[37]:

#Combine all the values in one array
values=['fare_amount', 'pickup_longitude','pickup_latitude', 'dropoff_longitude', 'dropoff_latitude', 'dist']

In[38]:

linear_Data = Train_Data[values] #

In[39]:

#This function is developed to get columns for specific passenger count. The idea is developed from R linear regression fit,
#which explains all the passenger count individualy contributes in the model

cat_names = ['passenger_count'] for i in
cat_names: temp =
pd.get_dummies(Train_Data[i], prefix= i)
linear_Data = linear_Data.join(temp)

In[40]:

linear_Data.shape #

In[41]:

linear_Data.head() #

In[42]:

#Spliting the newly created data set with passenger count dummies train1,
test1 = train_test_split(linear_Data, test_size=0.2)

In[43]:

#Import libraries for LR import
statsmodels.api as sm
# Train the model using the training sets
model = sm.OLS(train1.iloc[:, 0].astype(float), train1.iloc[:, 1:12].astype(float)).fit() #

In[44]:

Print out the statistics model.summary()

In[45]:

# make the predictions by the model
predictions_LR = model.predict(test1.iloc[:,1:12])

In[46]:

#Calculate MAPE
MAPE(test1.iloc[:,0], predictions_LR)

In[47]:

#Error 28.174474283418512
#Accuracy 71.8255

Prediction on original test data

In[48]:

pred=(pd.read_csv(r'C:\MAIN PROJECT\test\test.csv', header = 0)).drop(columns="pickup_datetime")

In[49]:
#create Dist variable

```
pred['dist'] = haversine( pred['pickup_latitude'], pred['pickup_longitude'],
                pred['dropoff_latitude'], pred['dropoff_longitude'])

pred['fare_amount']=0
pred['passenger_count']=pred['passenger_count'].astype('category')


# In[50]:


# Build model on the entire Train data
RF_model = RandomForestRegressor(n_estimators = 10).fit(Train_Data.iloc[:, 1:7],
Train_Data.iloc[:,0])

#predict value
pred['fare_amount'] = RF_model.predict(pred.iloc[:, 0:6])


# In[51]: pred.head() # In[52]:


#write output to csv

pred.to_csv("C:\MAIN PROJECT\Predicted_Values.csv", index = False)


# In[ ]:
```

REFERENCES

[1] Gunjan panda,Supriya p.panda.”Machine learning using exploratory analysis to predict cab fare”. International Journal for Research in Applied Science & Engineering Technology (IJRASET) Aug 2019.

[2] Kelareva, Elena. "Predicting the Future with Google Maps APIs." Web blog post. Geo Developers Blog,https://maps-apis.googleblog.com/2015/11/predicting- future-with-google-maps-apis.html Accessed 15 Dec. 2016.

[3] Van Lint, J. W. C., S. P. Hoogendoorn, and Henk J. van Zuylen. "Accurate freeway travel time prediction with state-space neural networks under missing data." Transportation Research Part C: Emerging Technologies 13.5 (2005): 347- 369.

[4] Vanajakshi, L., S. C. Subramanian, and R. Sivanandan. "Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses." IET intelligent transport systems 3.1 (2009): 1-9.

[5] Weijie Wang and Yanmin Lu, Analysis of the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) in Assessing Rounding Model, ICMEMSCE, IOP Publishing, 324 (2018), doi:10.1088/1757-899X/324/1/012049

[6] Wu, Chun-Hsin, Jan-Ming Ho, and Der-Tsai Lee. "Travel-time prediction with support vector regression." IEEE transactions on intelligent transportation systems 5.4 (2004): 276-281

[7] X. Qian, S. V. Ukkusuri.: Time-of-Day Pricing in Taxi Markets. IEEE Transactions on Intelligent Transportation Systems, Vol. 18 June 2017.

[8] Yildirimoglu, Mehmet, and Nikolas Geroliminis. "Experienced travel time prediction for congested freeways."Transportation Research Part B: Methodological 53 (2013): 45-63.