REPORT

eDWISOR.com

**Project Name**:

**Cab Fare Prediction**

**Submitted By:**

Rishabh Singh

**ABSTRACT**

The objective of this Case is that we are a cab rental start-up company and have successfully run the pilot project and now want to launch our cab service across the country. We have collected the historical data from our pilot project and now have a requirement to apply analytics for fare prediction. We need to design a system that predicts the fare amount for a cab ride in the city. The dataset of our project contains 7 variables each representing different attributes and properties. The total number of observations are 16000+ entries. This project is divided into four major stages.

The first part contains the Exploratory Data Analysis where we check the head of the dataset, how each variable is type defined and use describe method to check the max, min and other values.

The second part contains the Data preprocessing and data cleaning techniques to make it ready for various Machine Learning Algorithms.

The third part contains the training our model on various Machine Learning Algorithms and testing our model on them.

The last and the final part of the project contains the prediction of our sample test data by most appropriate algorithm.

**List of Contents**

|  |  |  |
| --- | --- | --- |
| **SNo.** | **TITLE** | **Page no** |
| 1 | Introduction | 4 |
| 2 | Exploratory Data Analysis | 5 |
| 3 | Data Preprocessing   * Missing value Analysis * Outlier Analysis * Feature Selection * Feature Scaling | 6  6  6  7  9 |
| 4 | Model Development   * Linear Regression * Decision tree * Random forest * KNN | 11  11  12  13  14 |
| 5 | Summary of Models | 16 |
| 6 | Predicting TEST data | 17 |
| 7 | Appendix: Python code | 18 |

**INTRODUCTION**

We are a cab rental start-up company and have successfully run the pilot project and now want to launch our cab service across the country. We have collected the historical data from our pilot project and now have a requirement to apply analytics for fare prediction. We need to design a system that predicts the fare amount for a cab ride in the city.

Number/ Nature of attributes:

* pickup\_datetime – time stamp value indicating when the cab ride

started

* pickup\_longitude - float for longitude coordinate of where the cab

ride started.

* pickup\_latitude - float for latitude coordinate of where the cab ride

started.

* dropoff\_longitude - float for longitude coordinate of where the cab

ride ended.

* dropoff\_latitude - float for latitude coordinate of where the cab ride

ended.

* passenger\_count - an integer indicating the number of passengers

in the cab ride.

Out of the total of 7 variables of the dataset:

Categorical variables are: 0

Continuous variables are: 6

Index/String variable: 1

**Exploratory Data Analysis**

1. Set working directory: To import and export all data to a location
2. Load Data: training data as cabcount\_train
3. Checking the shape of Data: 16067 rows, 7 columns
4. Checking the type of each variable in the dataset:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 16067 entries, 0 to 16066

Data columns (total 7 columns):

fare\_amount 16043 non-null object

pickup\_datetime 16067 non-null object

pickup\_longitude 16067 non-null float64

pickup\_latitude 16067 non-null float64

dropoff\_longitude 16067 non-null float64

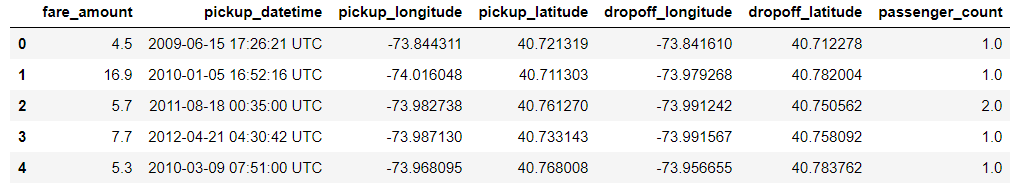
dropoff\_latitude 16067 non-null float64

passenger\_count 16012 non-null float64

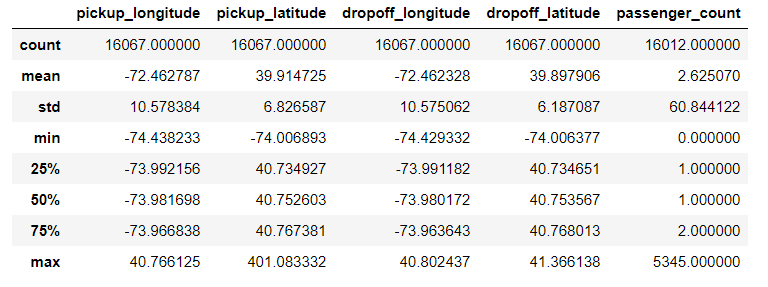
dtypes: float64(5), object(2)

memory usage: 878.7+ KB

1. Check the head of the dataset to know the first five entries:



1. Apply describe () method to calculate the mean, variance, and standard deviation etc of the dataset:



**DATA Pre-processing**

1. **Missing Value Analysis**: Missing value pertains to condition when one or more values are absent from one or more independent variables of the dataset. The reason for missing values could be many. Some commonly included reasons are Human error, refuse to answer during survey, optional box questionnaire etc. Missing values in a dataset may be imputed or that particular variable may be dropped depending upon the percentage of missing values present in the column.

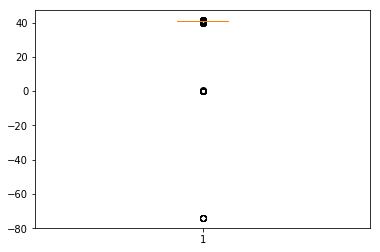
There are various methods to impute missing values, some of which are:

1. Mean
2. Mode
3. Median
4. KNN imputation
5. Prediction method

Checking the missing value in the Dataset: Our Dataset contain missing values and so to gain highest accuracy in our model we impute them with mean.

1. **Outlier Analysis**: Observations inconsistent with the rest of the dataset is called an outlier. Causes of outliers are poor or contaminated data, low quality measurements, malfunctioning equipment, correct but exceptional data.

* Plot boxplot to visualize Outliers.
* Separate Numeric and Categorical variables.



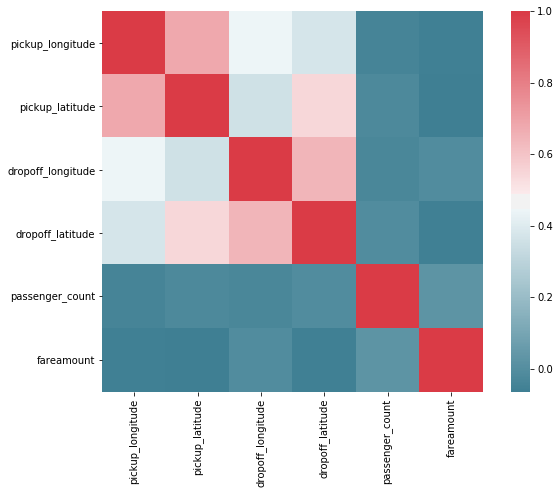
Box plot for dropoff\_latitude

* Calculate outliers and inliers in each numeric variable.
* Calculate inter quartile range.
* Drop those observations which are beyond outliers and inliers.
* Checking again the shape of Dataset after removing outliers.

1. **Feature Selection**: Feature selection refers to selecting a subset of relevant features for use in model development. It refers to a subset of learning algorithms input variable upon which it should focus attention while ignoring the rest. It also reduces the dimensionality of the dataset.

The following set of steps are followed for feature selection:

* Generate Correlation Matrix.
* Plot heatmap using seaborn library to identify correlation between the various continuous variables:



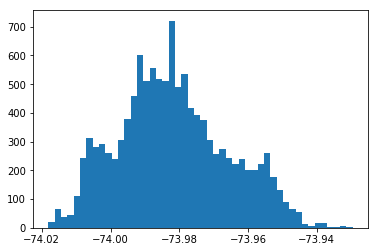
Heatmap for Feature selection

* As can be seen from the above correlation plot that there are no very highly correlated independent variables which may hamper the accuracy of our model.
* Since our Dataset contain categorical variable so there is need to do Chi square test for feature selection.

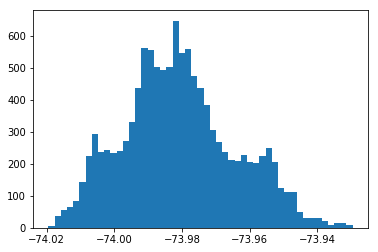
1. **Feature Scaling**: Feature scaling refers to the task of converting the various values of the column within the specified limits. Feature scaling is done to bring all the variable values within the limits so that no single variable has more influence over other variables while developing the model. It is the process of reducing the variations either within or between the variables. It brings all the variables into proper proportions with one another.

Feature scaling commonly uses two methods to compute.

* Normalization
* Standardisation



data distribution for pickup\_longitude



distribution of data in dropoff\_longitude

As can be seen from above histograms our variables are not exactly normally distributed so we have to adopt Normalization method of feature scaling.

**MODEL DEVELOPMENT**

Machine learning is, programming computers to optimize a performance criterion using example data or past experience. During the development of this model we applied various machine learning algorithms.

* Divide data into train and test for model development and to check accuracy and deduce which model is best.

1. **Linear Regression**:

It is a Prediction Model. There are two types of Linear Regression

* Simple linear regression
* Multiple linear regression

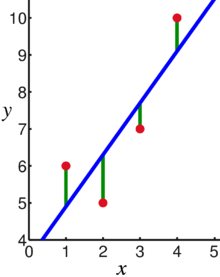
It describes relationship among variables. The one simple case is where a dependent variable may be related to independent or explanatory variable.

* Equation expressing this relation is the line:

Y = b0 + b1x1 + b2x2 +….

Where b0 is the intercept and x1, x2 are independent variables.

* For a given set of values we need to calculate values for b0 and b1.
* Look for a line which minimizes the sum of the residuals.



Linear Regression Line

* Import Libraries for Linear Regression.
* Build and train Linear Regression model using X\_train and y\_train.
* Predict model outcome using test data X\_test.
* Regression Evaluation Metrics:

MAE: 0.14864871305584834

MAPE: 14.864871305584835

MSE: 0.03445213367764688

RMSE: 0.18561285967746652

1. **Decision Tree**:

Decision Tree is a predictive model based on a branching series of Boolean tests. It can be used for classification and regression both type of target variables. There are number of different types

of decision trees that can be used in Machine learning algorithms. Decision tree is a rule. Each branch connects nodes with “and” and multiple branches are connected by “or”. They are extremely easy to understand by the business users. Decision trees build some intuitions about your customer base. E.g. “are customers with different family sizes truly different?”

* Import Libraries.
* Model Development: Build and train model using X\_train and y\_train
* Predict model outcome using test data X\_test
* Evaluation Metrics:

MAE: 0.14181602659490603

MAPE: 14.181602659490602

MSE: 0.03183425036933897

RMSE: 0.1784215524238565

1. **Random Forest**:

Random forest is an ensemble that consists of many decision trees. The term came from random decision forests that was first proposed by Tin Kam Ho of Bell Labs in 1995. This method combines Breiman's "bagging" idea and the random selection of features. It outputs the class that is the mode of the class's output by individual trees. It calculates results by evaluating mean for regression and mode for classification. This algorithm can be used for both classification and regression

* Import Libraries
* Develop and train random forest model on train data using X\_train and y\_train
* Predict new test cases using X\_test
* Evaluation Metrics:

MAE: 0.08484306361784265

MAPE: 8.484306361784265

MSE: 0.012912566667796574

RMSE: 0.11363347511977522

1. **K Nearest Neighbour**:

KNN stands for K-Nearest Neighbour. KNN is simple algorithm that stores all available cases and classifies new cases based on a similarity measure. It is a Supervised Machine Learning Algorithm. It can be used for both classification and regression. It is considered to be a Lazy Learning Algorithm.

Pick a number of neighbours you want to use for classification or regression. Choose a method to measure distances from test point. Keep a data set with records. For every new point, identify the number of nearest neighbours you picked using the method you choose. Let them vote if it is a classification or take a mean/median for regression



KNN Algorithm

* Import Libraries
* Develop and train KNN model on train data using X\_train and y\_train
* Predict new test cases using X\_test
* Evaluation Metrics:

MAE: 0.08543941084437791

MAPE: 8.54394108443779

MSE: 0.013762374467597194

RMSE: 0.1173131470364562

**SUMMARY OF MODELS**

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. Based on MAPE we can deduce that Linear Regression is the best Algorithm to predict our target variable.

* DECISION TREE

MAE for Decision tree = 0.1418

MAPE for Decision tree = 14.18%

MSE for Decision tree = 0.0318

RMSE for Decision tree = 17842

* RANDOM FOREST

MAE for Random Forest = 0.0848

MAPE for Random Forest = 8.484%

MSE for Random Forest = 0.01291

RMSE for Random Forest = 0.1136

* LINEAR REGRESSION

MAE for Linear Regression = 0.1486

MAPE for Linear Regression = 14.864%

MSE for Linear Regression = 0.03445

RMSE for Linear Regression = 0.18561

* KNN K NEAREST NEIGHBOUR

MAE for KNN Regression = 0.08543

MAPE for KNN Regression = 8.543%

MSE for KNN Regression = 0.01376

RMSE for KNN Regression = 0.1173

**PREDICTING TEST DATA**

After forming the machine learning algorithm and tested it for test data we have fetch out from train dataset the most accurate method. Now we are ready to predict any external data given to us. Now since we are not given with any test data we create a sample dataset out of our given train data just to confirm the working of our model using the most appropriate algorithm we have formed in previous section. As we have deduced from MAPE values in our previous section that KNN gives most accurate model so we will use KNN to predict the target value.

Steps followed for predicting target variable in test data:

* Let us import the test dataset to predict our target variable
* Checking the shape and info of the test dataset
* Change the “passenger\_count” variable from int to float type as it was in our train dataset so that it may not hamper our model in any way.
* Drop the "pickup\_datetime” column from the Dataset as our model is not trained for it.
* Now we apply our model to KNN to predict our target variable “target\_amount”.
* Our “target\_amt” column gets added to as the last column of the dataset.
* Writing this new dataset with predicted variable as a CSV file.

The CSV file of the target variable is also included.

APPENDIX: Python Code

* import os

import pandas as pd

import numpy as np

from fancyimpute import KNN

import matplotlib.pyplot as plt

from scipy.stats import chi2\_contingency

import seaborn as sns

from random import randrange, uniform

* os.chdir("C:/Users/risha/Desktop/Cab\_project/cab")
* os.getcwd()
* cabcount\_train = pd.read\_csv('train\_cab.csv')
* cabcount\_train.shape
* cabcount\_train.info()
* cabcount\_train.head(5)
* cabcount\_train.describe()
* cabcount\_train["fareamount"] = pd.to\_numeric(cabcount\_train["fare\_amount"], downcast='float',errors="coerce")
* cabcount\_train = cabcount\_train.drop(["fare\_amount","pickup\_datetime"], axis=1)
* missing\_val = pd.DataFrame(cabcount\_train.isnull().sum())
* cabcount\_train["passenger\_count"] = cabcount\_train["passenger\_count"].fillna(cabcount\_train["passenger\_count"].mean())
* cabcount\_train["fareamount"] = cabcount\_train["fareamount"].fillna(cabcount\_train["fareamount"].mean())
* cnames = ["pickup\_longitude","pickup\_latitude","dropoff\_longitude","dropoff\_latitude","passenger\_count","fareamount"]
* %matplotlib inline
* plt.boxplot(cabcount\_train[‘dropoff\_latitude'])
* plt.boxplot(cabcount\_train['casual'])
* for i in cont\_names:

q75, q25 = np.percentile(cabcount\_train.loc[:,i], [75 ,25])

iqr = q75 - q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

cabcount\_train = cabcount\_train.drop(cabcount\_train[cabcount\_train.loc[:,i] < min].index)

cabcount\_train = cabcount\_train.drop(cabcount\_train[cabcount\_train.loc[:,i] > max].index)

* df\_corr = cabcount\_train.loc[:,cont\_names]

f, ax = plt.subplots(figsize=(10, 7))

corr = df\_corr.corr()

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),square=True, ax=ax)

* %matplotlib inline
* plt.hist(cabcount\_train["dropoff\_longitude"], bins='auto')
* for i in cnames:

cabcount\_train[i] = (cabcount\_train[i] - np.min(cabcount\_train[i]))/(np.max(cabcount\_train[i]) - np.min(cabcount\_train[i]))

* from sklearn import tree

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

from sklearn.model\_selection import cross\_val\_score

* X = cabcount\_train.values[:, 0:5]

Y = cabcount\_train.values[:,5]

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, Y, test\_size = 0.2)

* DT\_model = tree.DecisionTreeRegressor(max\_depth=2).fit(X\_train, y\_train)
* predictions\_DT = DT\_model.predict(X\_test)
* print('MAE:', metrics.mean\_absolute\_error(y\_test, predictions\_DT))

MAPE = metrics.mean\_absolute\_error(y\_test, predictions\_DT)

print("MAPE:", MAPE\*100)

print('MSE:', metrics.mean\_squared\_error(y\_test, predictions\_DT))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, predictions\_DT)))

* from sklearn.ensemble import RandomForestRegressor
* RF\_model = RandomForestRegressor(n\_estimators = 10).fit(X\_train, y\_train)
* RF\_Predictions = RF\_model.predict(X\_test)
* print('MAE:', metrics.mean\_absolute\_error(y\_test, RF\_Predictions))

MAPE = metrics.mean\_absolute\_error(y\_test, RF\_Predictions)

print("MAPE:", MAPE\*100)

print('MSE:', metrics.mean\_squared\_error(y\_test, RF\_Predictions))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, RF\_Predictions)))

* from sklearn.linear\_model import LinearRegression
* lm = LinearRegression()

lm.fit(X\_train,y\_train)

* LR\_predictions = lm.predict(X\_test)
* print('MAE:', metrics.mean\_absolute\_error(y\_test, LR\_predictions))

MAPE = metrics.mean\_absolute\_error(y\_test, LR\_predictions)

print("MAPE:", MAPE\*100)

print('MSE:', metrics.mean\_squared\_error(y\_test, LR\_predictions))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, LR\_predictions)))

* from sklearn.neighbors import KNeighborsRegressor
* KNN\_model = KNeighborsRegressor(n\_neighbors = 3).fit(X\_train, y\_train)
* KNN\_Predictions = KNN\_model.predict(X\_test)
* print('MAE:', metrics.mean\_absolute\_error(y\_test, KNN\_Predictions))

MAPE = metrics.mean\_absolute\_error(y\_test, KNN\_Predictions)

print("MAPE:", MAPE\*100)

print('MSE:', metrics.mean\_squared\_error(y\_test, KNN\_Predictions))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, KNN\_Predictions)))

* cab\_test = pd.read\_csv("test.csv")
* cab\_test.shape
* cab\_test.info()
* cab\_test["passenger\_count"] = cab\_test.passenger\_count.astype(float)
* cab\_test.info()
* cont\_names = ["pickup\_longitude","pickup\_latitude","dropoff\_longitude","dropoff\_latitude","passenger\_count"]
* # Nomalisation

for i in cont\_names:

#print(i)

cab\_test[i] = (cab\_test[i] - np.min(cab\_test[i]))/(np.max(cab\_test[i]) - np.min(cab\_test[i]))

* cab\_test = cab\_test.drop(["pickup\_datetime"], axis=1)
* cab\_test['target\_amt'] = KNN\_model.predict(cab\_test)
* cab\_test.to\_csv('cab\_test\_predict.csv',index=False)

**Important Note**

* INSTRUCTION TO RUN AND DEPLOY THE CODE IS WRITTEN WITH THE CODE ITSELF.
* ONLY THE FILE LOCATION SHOULD BE CHANGED IN OS.CHDIR FUNCTION ACCORDING TO THE USER FILE LOCATION.
* USER CAN SIMPLY USE SHIFT + ENTER TO GET ALL COMMAND RUN.
* FILES INCLUDED ARE:

1. PYTHON CODE FILE
2. PROJECT REPORT IN BOTH WORD AND PDF FORMATS
3. R CODE FILE
4. DATASET WITH TARGET VARIABLE IN .CSV FORMAT

