Predicting Cab Fare

Vipin Kumar

25 September 2019

Contents

1 Introduction

1.1 Problem Statement

1.2 Data

2 Methodology

2.1 Pre Processing

2.1.1 Missing Value Analysis

2.1.2 Formatting & Cleaning Data

2.1.3 Outlier Analysis

2.1.4 Normalisation

2.2 Modelling

2.2.1 Multiple Linear Regression

2.2.2 Random Forest Regression

2.2.3 Support Vector Regression

3 Conclusion

3.1 Model Selection

3.2 Conclusion

Chapter 1

Introduction

1.1 Problem Statement

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.

1.2 Data

Our task is to build regression models which will predict the fare of the cab rides based on our historical data from a pilot project. Given below is a sample of the data set that we are using to predict the fare:

Table 1.1: Train Data Sample

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

Table 1.2: Test Data Sample

pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
2015-01-27 13:08:24 UTC	-73.97332001	40.76380539	-73.98143005	40.74383545	1
2015-01-27 13:08:24 UTC	-73.98686218	40.71938324	-73.99888611	40.73920059	1

2011-10-08 11:53:44 UTC	-73.982524	40.75126	-73.979654	40.746139	1
2012-12-01 21:12:12 UTC	-73.98116	40.767807	-73.990448	40.751635	1
2012-12-01 21:12:12 UTC	-73.966046	40.789775	-73.988565	40.744427	1

As you can see we have 6 independent and 1 dependent variable (which is fare_amount) in our train data. Our target variable is fare_amount and other variables are described as follows :

- datetime - timestamp 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.

Chapter 2

Methodology

2.1 Pre Processing

Any predictive modelling requires that we look at the data before we start modelling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**.
As given in the problem that the data have missing values we'll first start with Missing value analysis.

2.1.1 Missing Value Analysis

It includes analysing if the given data contains any missing. Missing values can make our predictive modelling difficult at a later point of time. So, it's better to remove or impute the missing values before hand.
Our check for the missing values on the train data gave us missing values in fare_amount variable and passenger_count variable.

```
fare_amount        24
pickup_datetime     0
pickup_longitude    0
pickup_latitude     0
dropoff_longitude   0
dropoff_latitude    0
passenger_count    55
```

As the number of missing values are very less compared to the size of the data [(16067, 7)] ,we can remov e the missing values and assume that this will not cost much to our model.
Hence, deleting the rows with missing values.

```
train_data=train_data.dropna()
```

2.1.2 Cleaning Data

Cleaning of data includes converting data into proper format and removing the data which are invalid according to the business need. We'll start with checking the datatypes of the variables.

```
fare_amount          object
pickup_datetime      object
pickup_longitude    float64
pickup_latitude     float64
dropoff_longitude   float64
dropoff_latitude    float64
passenger_count     float64
```

We can see from our output the fare_amount variable is an object type variable which should be ideally numeric. Variable pickup_datetime should be a datetime variable and passenger_count should be an int type.

While converting the variables into their ideal type we encountered some invalid values present in our data such as fare_amount variable have ‘430-‘ as an entry and pickup_datetime variable have ‘43’ as an entry. After removing these invalid entries we’ll covert our variables to their ideal format.

```
train_data['pickup_datetime']=pd.to_datetime(train_data['pickup_datetime']
,infer_datetime_format=True)

train_data['fare_amount'] = train_data['fare_amount'].astype('float64')

train_data['passenger_count']=train_data['passenger_count'].astype('int64'
)
```

Now, we’ll check our data using describe() function which gives us a blueprint of the data providing some statistical insights of data.

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	15986.000000	15986.000000	15986.000000	15986.000000	15986.000000	15986.000000
mean	15.030453	-72.464352	39.915577	-72.463909	39.898671	2.623233
std	431.213944	10.573594	6.829028	10.570256	6.186375	60.892032
min	-3.000000	-74.438233	-74.006893	-74.429332	-74.006377	0.000000
25%	6.000000	-73.992144	40.734935	-73.991182	40.734647	1.000000
50%	8.500000	-73.981691	40.752603	-73.980168	40.753549	1.000000
75%	12.500000	-73.966817	40.767353	-73.963644	40.768005	2.000000
max	54343.000000	40.766125	401.083332	40.802437	41.366138	5345.000000

We can see from the above table that the minimum value of fare_amount is negative which is obviously not possible. Fare_amount should be greater than zero always. Hence, we’ll remove the values have fare_amount as zero or below. And we can also see the maximum passenger_count is 5345 which again is not possible. The maximum count of passengers allowed are 6 in xl cabs. Hence, we’ll remove the entries with passenger_count more than 6.

```
train_data=train_data[train_data.fare_amount > 0]
train_data=train_data[train_data.passenger_count <=6]
```

We have cleaned our data from all invalid entries. All we have is valid relevant data.
While comparing our cleaned train data to our test data we found that our train data have very high variance in comparison to our test data.
This can be explained by the presence of outliers in our data and hence we’ll next move to outlier analysis.

2.1.3 Outlier Analysis

We have various methods to analyse outliers in our data, the most common and widely excepted from them is through Box plot. The plot represents the data lying above 75 percentile plus 1.5*IQR (inter quantile range) and data lying below 25 percentile minus 1.5 times of IQR as outliers in the data.
Where, IQR= (75 percentile data point -25 percentile data point)

[NOTE: We are not including passenger_count variable in the outlier analysis as it’s stats match quite well with that of test data.]
From the box plot shown in Fig 2.1 we can clearly see that all the variables have outliers which can deviate our model. Therefore, the rows containing outliers will be dropped from the data.

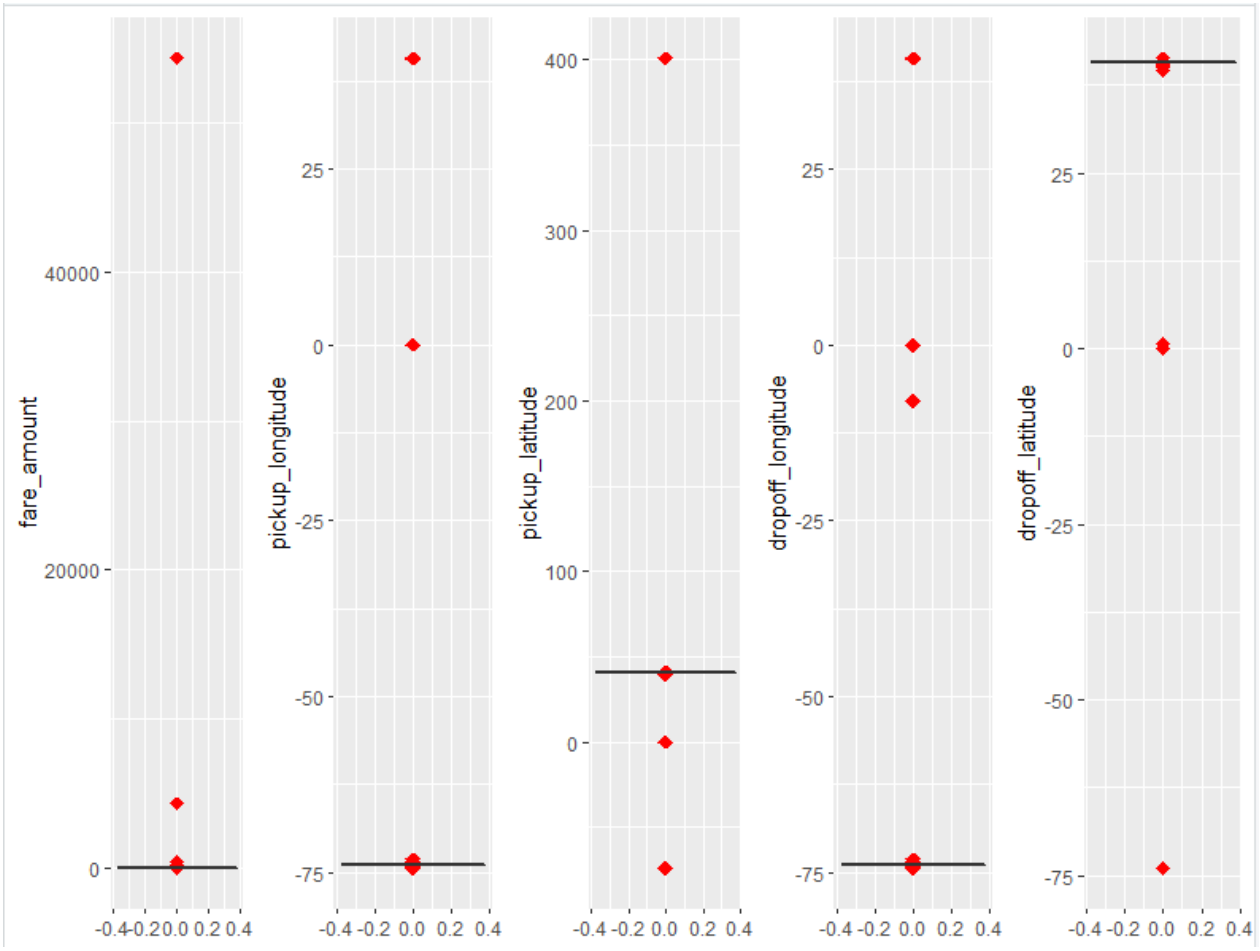

Fig. 2.1 BOX PLOT

2.1.4 Feature Engineering and Exploratory Analysis

Feature engineering is done to create interaction terms. This is done when your available attributes are not directly contributing that much in the prediction and the variables derived from them can contribute better to our model.
For this problem of detecting cab fare, distance travelled cab be the most crucial attribute.
But the distance is not available. So, we'll derive the distance from our coordinates given for each ride.
Next most important aspect to decide the fare amount can be the traffic of cab request that means demand of cab is high or low. While we cannot directly analyse the live demand here, we can take include some important aspects that decides demand. For the very same reason we have derived weekday and hour from our datetime variable.
After extracting relevant features from the given features some given attribute such as the datetime variable hav not much to offer. Hence, there features are excluded while modelling.

Now, some exploratory analysis is done to analyse the distrubution of our variables.

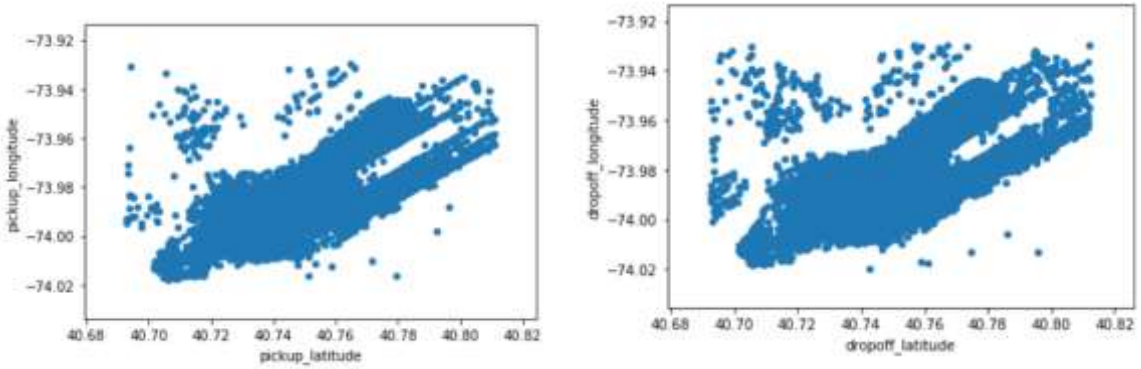

Fig 2.2 : Pickup and drop coordinates

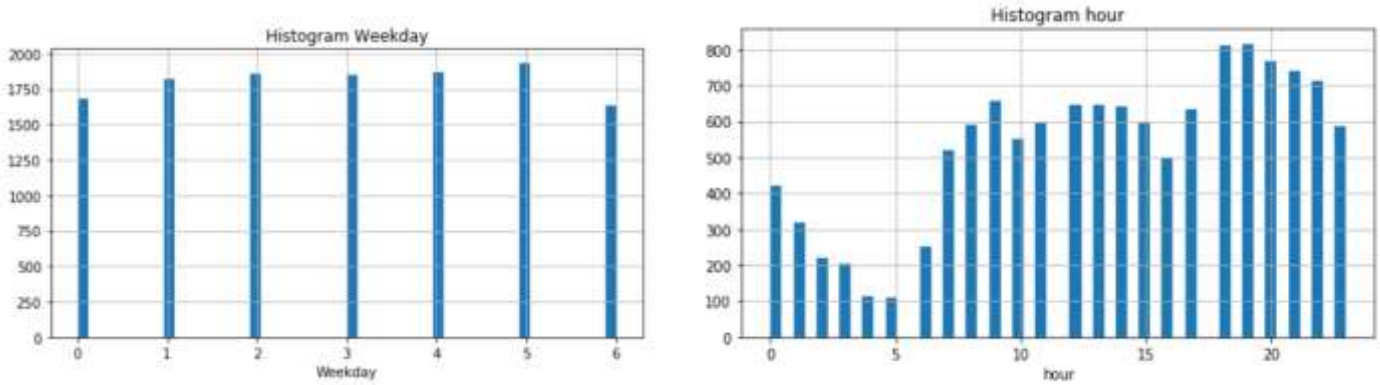

Fig 2.3 : Probability distribution of Weekday and Hour

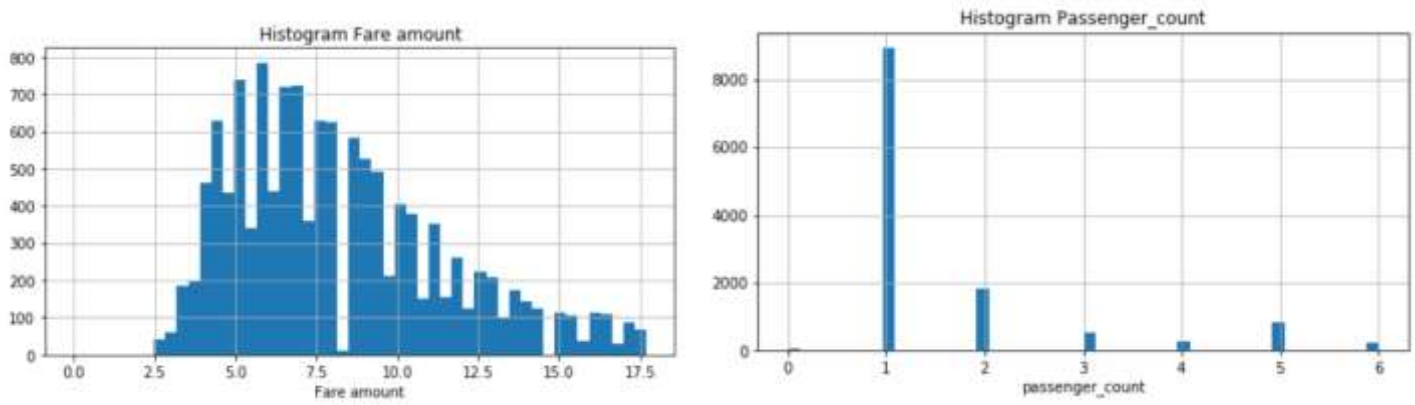

Fig 2.4 : Probability distribution of Fare Amount and passenger count

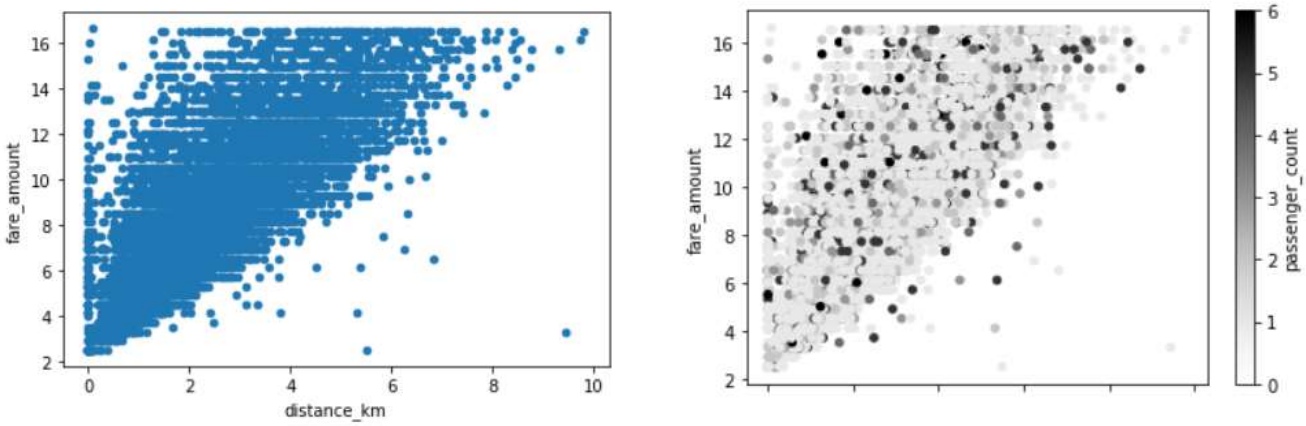

Fig 2.5 : Probability distribution of Fare Amount against distance and passenger count

- Fig 2.2 shows the distribution of pickup and drop points. The less denser areas in the graphs have less frequency of pickup and drops that may affect the availability and traffic in those regions and thus affect cab fare.
- Fig 2.3 shows the frequency of rides in accordance with day of weeks and hours of a day. We can clearly see that frequency is less on weekend (0 and 6) is less than that on weekdays and it is further clear from the hour graph that the rides are most frequent during evening 6pm to 11pm.
- Fig 2.4 shows the distribution of fare and number of passenger.
- Fig 2.5 shows the scatter plot between distance and fare which clearly shows that fare is dependent on distance. The other plot also includes passenger count which looks randomly distributed. So, there is not much effect of number of passengers on fare amount.

2.1.5 Normalisation

Normalisation is a part of data scaling. Data scaling is used to minimize the biasing created in the model by attributes of high magnitude over the low magnitude attributes. Standardization can also be used as a scaling method but it requires data to be in normal form. And we have seen from our distribution plots that the attributes we have are somewhat skewed in nature.
We have normalize all the numeric features which are pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude and distance.

2.2 Modelling

After completing all the pre processing, we have a better cleaner data which will produce less noisy and better predicting models. As we can clearly see that this a regression problem, we'll start with the simplest algorithm for regression modelling that is multiple linear regression and then we'll try to migrate to bit complex models to try and improve the accuracy of model.
Data is first split into train data (Used to train the model) and test data (used to test the model with unseen data) .
In this case we are dividing the data in the ratio of 80% (train) and 20% (test), however we can use different ratios to make this split.

2.2.1 Multiple linear regression

```
lr = LinearRegression()
model1=lr.fit(X_train, y_train)
```

PREDICTION AND ERROR

We'll predict the fare_amount for our test dataset which is new for the model to check error in the model.

```
predictions_LR = model1.predict(X_test)
```

We generally use MAPE (Mean Absolute Percentage Error) or RMSE (Root Mean Square Error) to calculate error in our prediction.
The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a loss function for regression problems in machine learning.
Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are.
We'll use both the methods to calculate and fix the one which suits better for this problem.

```
def rmse(predictions, targets):
    return np.sqrt(((predictions - targets) ** 2).mean())

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

rmse(predictions_LR,y_test)
MAPE(y_test,predictions_LR)
```

#RMSE: 2.1001261816830237
#MAPE: 19.919308961188957

As we can see RMSE looks very low approximating at 2 but the MAPE is very large in comparison with 19.9% , this is happening because our data consists of small trips with small fare amount. Thus, rmse is giving us a false image of highly accurate model.
Therefore, we'll use MAPE to calculate error, going further in modelling.

2.2.2 Random Forest Regression

Random forest is also used for regression purpose and can give good results as it uses bootstrap aggregation which is generally called the bagging technique.

Hyperparameters which decide the depth of a particular tree and decide how much such trees should be made to optimally find the best model, are tuned using GridSearchCV function. The outcomes suggests us to take max_depth as 6 and number of trees as 1000 for best results.

The modelling is done using the best fit parameters and errors are calculated.

```
rfr = RandomForestRegressor(max_depth=6, n_estimators= 1000,
random_state=False, verbose=False)
model2=rfr.fit(X_train, y_train)
```

PREDICTION AND ERROR

```
predictions_RF = model2.predict(X_test)
MAPE(y_test,predictions_RF)
```

#MAPE:18.766974136672264

2.2.3 Support Vector Regression

The epsilon regression technique is used to model our predictor with different kernels ,linear kernel gave best results for the given data and hence, fixed as the kernel for this model.

Hyperparameters are tuned using GridSearchCV fuction.

C = 1 and gamma = 0.001, as our best fit for this model.
Modelling is done with our best fit parameters.

```
svrepslintuned = SVR(kernel = 'linear',C=1,gamma=0.001)
modeltuned=svrepslintuned.fit(X_train, y_train)
```

PREDICTION AND ERROR

```
predictfinal=modeltuned.predict(X_test)
MAPE(y_test, predictfinal)
```

#MAPE: 17.653845736605504

Chapter 3

Conclusion

3.1 Model Evaluation

Broadly, three methods are used in this project to model our predictor. After a fine preprocessing of data all of them gave an error percentage of 20 or less. MAPE of these algorithms are stated below:

Multiple Linear Regression : 19.9 %
Random Forest :18.7%
SVR (eps) : 17.6 %

Clearly, SVR serves as the best method with least error rate of 17.6%
SVR is already tuned for cost parameter and gamma parameter and gives the same result stated above.

3.2 Model Selection and Prediction

We conclude that SVR eps-regression with linear kernel and other default parameters [cost: 1, gamma: 0.0001] is our best model with an error rate of 17.6% and accuracy of 82.4%
.

Now, we can predict the fare_amount of test data using this model after preprocessing the data in same way as done for train.
The result file, R code and python code file is attached with this report for reference.