**Cab Fare Prediction**

Abstraction:

In this project we need to predict the fare\_amount of the cab using attributes like drop -off latitude, drop-off longitude, pickup date-time, passenger\_count, pickup longitude, pickup latitude. Our project helps the organisation to take the right decisions, so that the organisation may run successfully without any losses.

Here we used R studio and Jupyter Notebook as platforms to work on the project. At first, we need to clean data using Exploratory Data Analysis like check for Missing Values, Outliners, then Feature Extraction, Feature Selection, Feature Scaling. Achieving the goal was quite challenging. Using the help of Machine Learning Algorithms like Multi Linear Regression, Random Forest, Decision Tree, enhance the probability of reaching goal early.

**Project Index**

|  |  |
| --- | --- |
|  |  |
| TOPIC | **Page number** |
| 1. Introduction | **3** |
|  |  |
| 2. Problem Statement | **3 - 4** |
|  |  |
| 3. Loading Data | **4** |
|  |  |
| 4. Data Visualisation & Processing | **5 - 7** |
|  |  |
| 5. Missing Value Analysis | **8 - 10** |
|  |  |
| 6. Outliner Analysis | **11 - 12** |
|  |  |
| 7. Feature Extraction | **12 – 16** |
|  |  |
| 8. Feature Selection | **16 - 21** |
|  |  |
| 9. Feature Scaling | **22 - 23** |
|  |  |
| 10. Splitting train and test data | **24** |
|  |  |
| 11. Model Development & Testing | **25 - 26** |
|  |  |
| 12. Conclusion | **27** |

**1. INTRODUCTION:**

Cab rental system is becoming a new frontier in business, particularly in major cities all over the world. They are many cab rental organisations who are competing with one another in race to achieve profit and fame.

In this project, our objective is to predict the cab fare-amount. The test data contains 16067 observations and 7 variables i.e fare-amount which is our target variable, pickup latitude, pickup longitude, drop-off latitude, drop-off longitude, passenger-count, pickup date-time.

Our aim is to develop a model that helps in predicting the fare amount, for future references using the past data.

**2. 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.

**Number of attributes:**

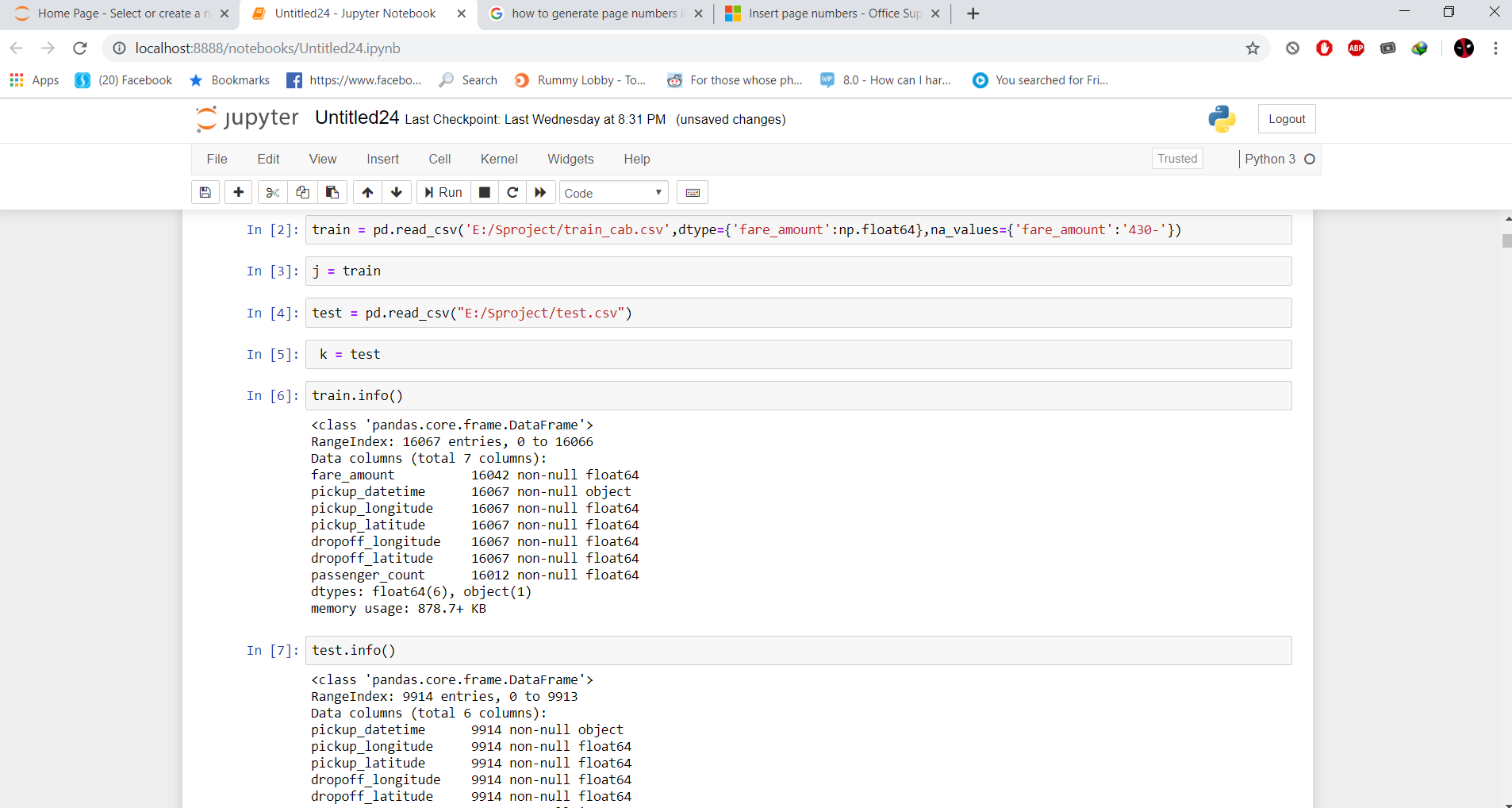
* Pickup-datetime - timestamp value indicating when the cab ride started.
* Pickup-longitude - float for longitude coordinate of where the cab ride started.
* Pickup-latitude - float for latitude coordinate of where the cab ride started.
* Dropoff-longitude - float for longitude coordinate of where the 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.

**3. Loading Data in R:**

After installing important packages.

Here we are using R studio and we are loading data into R- environment. Using following command:

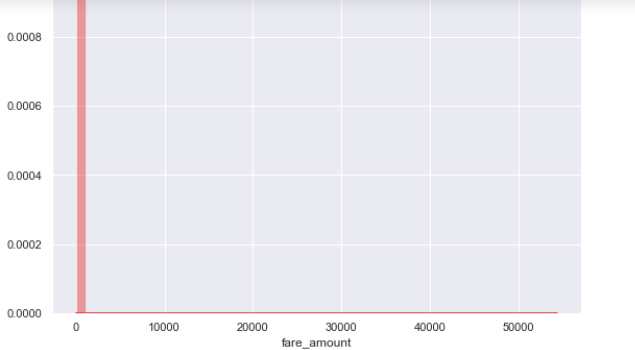
* Orginal\_train=pd.read\_csv("E:/Sproject/train\_cab.csv",header=T)
* Orginal\_test =pd.read\_csv("E:/Sproject/test.csv",header=T)



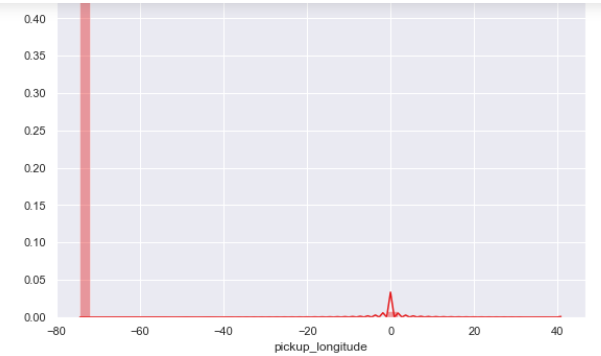
**4. Data Visualisation & Processing:**

Some histogram plots on different variables of train data.

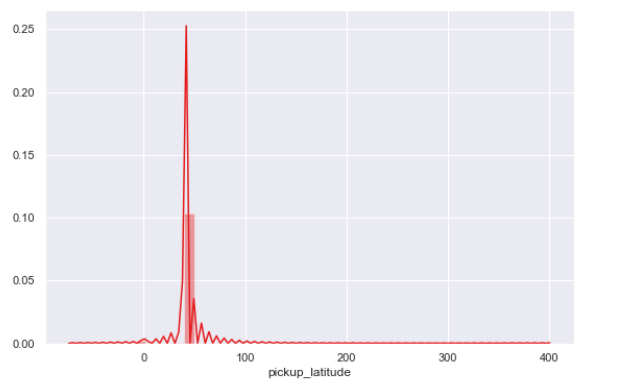
**Histogram plot on fare-amount:**

****

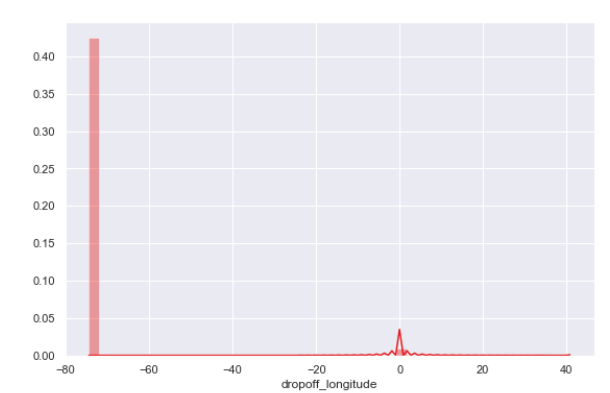
**Histogram plot on pickup-Longitude:**

****

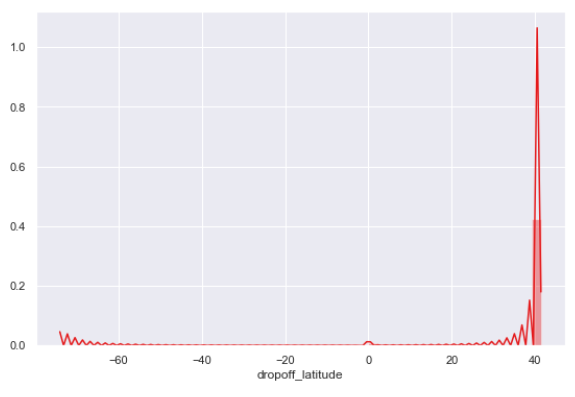
**Histogram plot on pickup-latitude:**

****

**Histogram plot on drop off-longitude:**

****

**Histogram plot on drop off latitude:**

****

**4. Data Processing / Exploratory Data Analysis:**

1. Fare amount has some negative values. it is also having 0 values, so we need to remove these fields. We used following code to overcome this:

sum(j['fare\_amount']<1)

here j is original train data. There are 5 negative values which are less than 1. We need to remove them.

1. The passenger count in a cab should be below 6, but there are values more than 6 values.

j[j['passenger\_count']>6]

There are 20 values which are greater than 6.

1. We need to check for negative values in passenger-count variable using following command:

j[j['passenger\_count']<1]

There are 58 negative values.

Now we are removing the negative values from fare-amount, passenger-count and the values greater than 6 from passenger-count variables, using following commands:

* j = j.drop(j[j['fare\_amount']<1].index, axis=0)
* j = j.drop(j[j['passenger\_count']>6].index, axis=0)
* j = j.drop(j[j['passenger\_count']<1].index, axis=0)

**5. Missing value Analysis:**

Here we are check for missing values in the dataset like empty rows which was filled with Na. we found some missing values in our dataset. Now missing values can be found by using different techniques like mean, median and Knn imputation.

First, we are selecting 1000 rows randomly and performing mean, median, Knn imputation, so that we can choose any one by comparing actual value and predicted value.

There are 3 types of methods to predict the missing values.

1. MEAN method

2.MEDIAN method

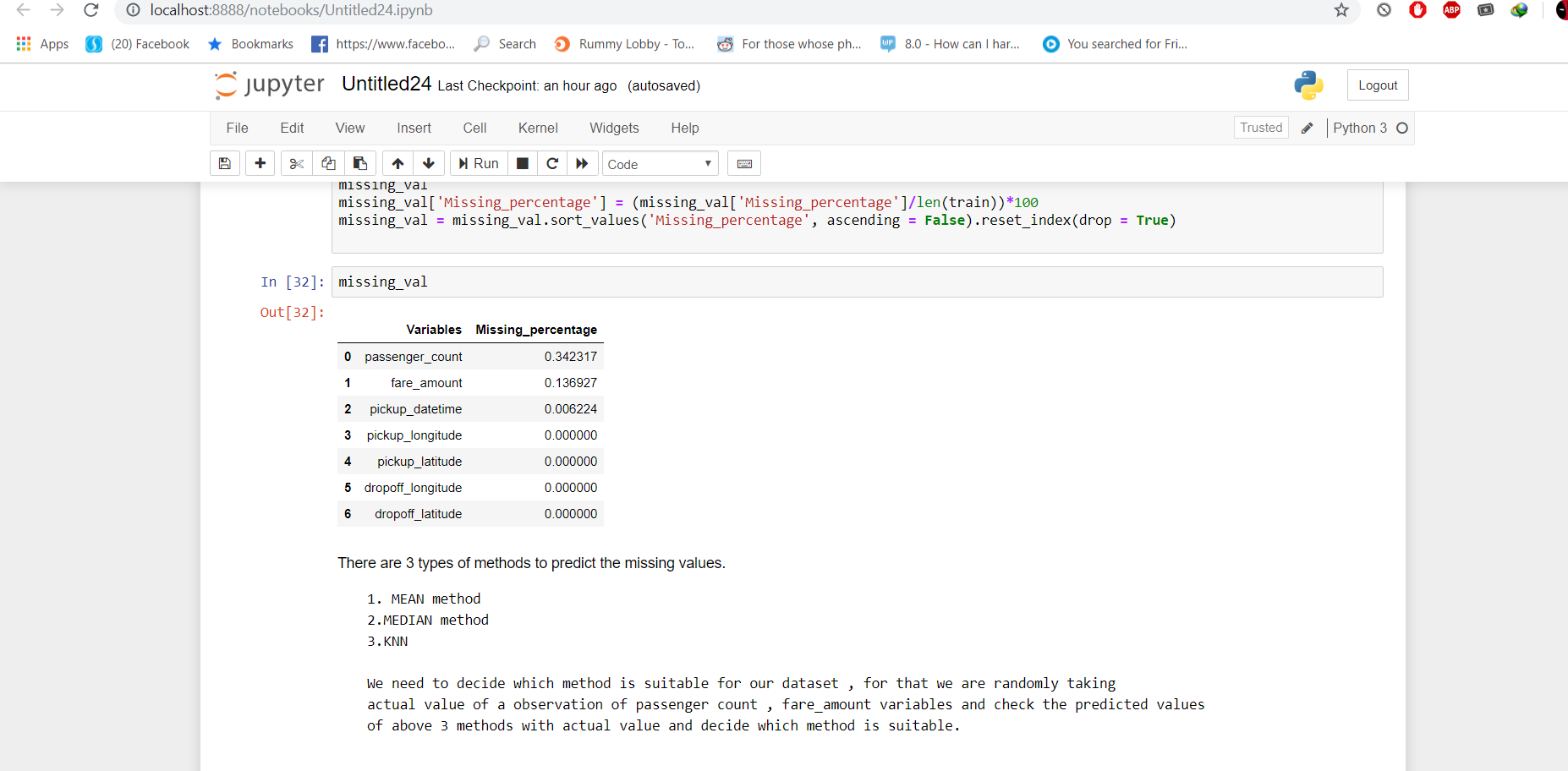
3.KNN

We need to decide which method is suitable for our dataset, for that we are randomly taking actual value of observation of passenger count , fare\_amount variables and check the predicted values of above 3 methods with actual value and decide which method is suitable.

1. For passenger-count variable the location, of 121 observation has actual value =1, here for this variable we can only use knn imputation as, it is not suitable for mean and median method.
2. For fare-amount variable of location 121, the actual value is 4, the mean value is 15, the median value is 8, but knn value is 3 which is nearer. So, we use knn method.

From above experience we decide to use knn method.

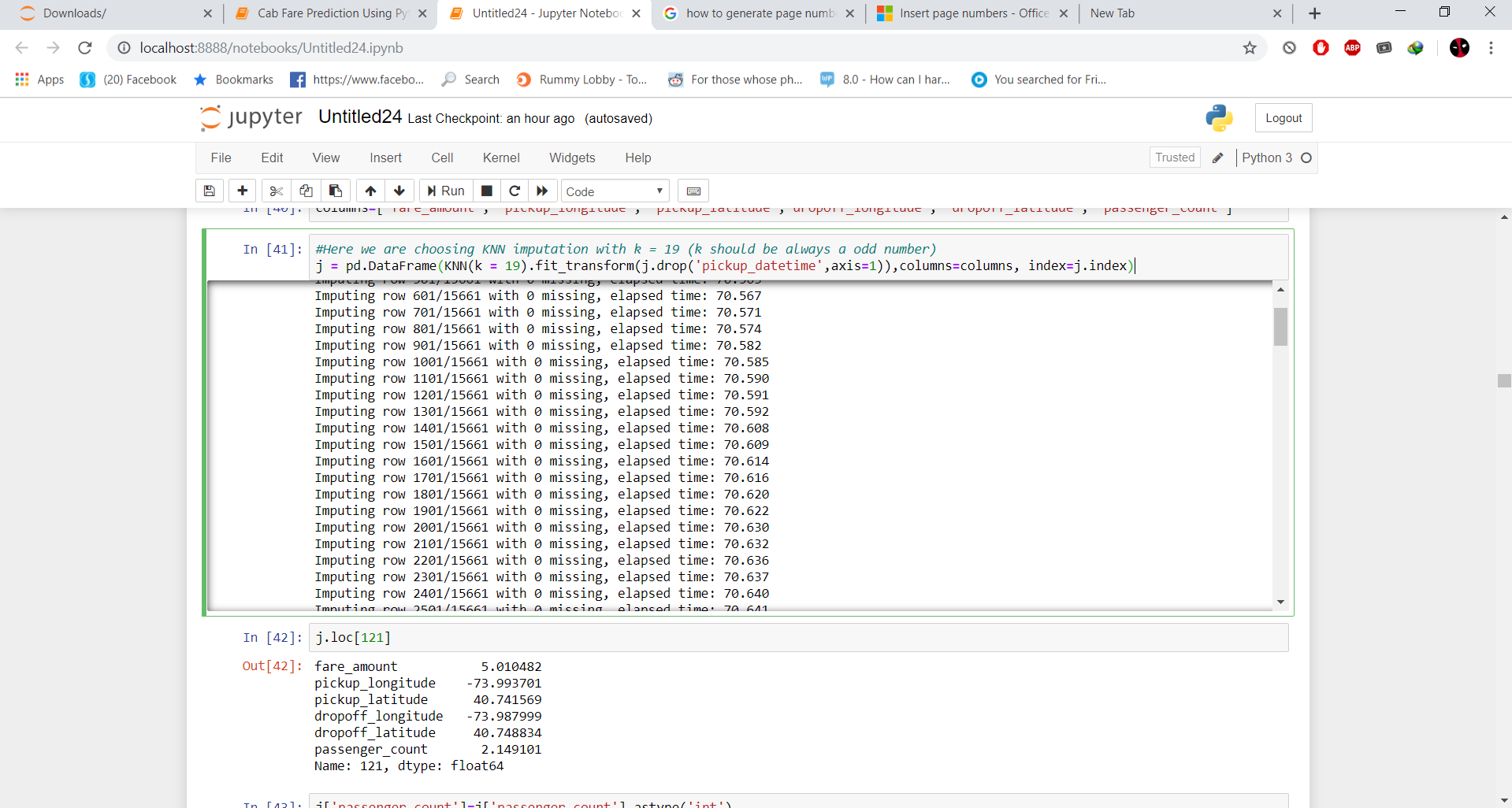
Here are the missing value percentage for variables of dataset.



As passenger-count variable has 34% of missing values & fare-amount has 14% of missing values.

We are now using knn imputation to replace missing values, using following command:

j=pd.DataFrame(KNN(k=19).fit\_transform(j.drop('pickup\_datetime',axis=1)),columns=columns, index=j.index)



**6. Outliner Analysis:**

Here we are performing **Outliner Analysis** only on fare-amount which is our dependent variable.

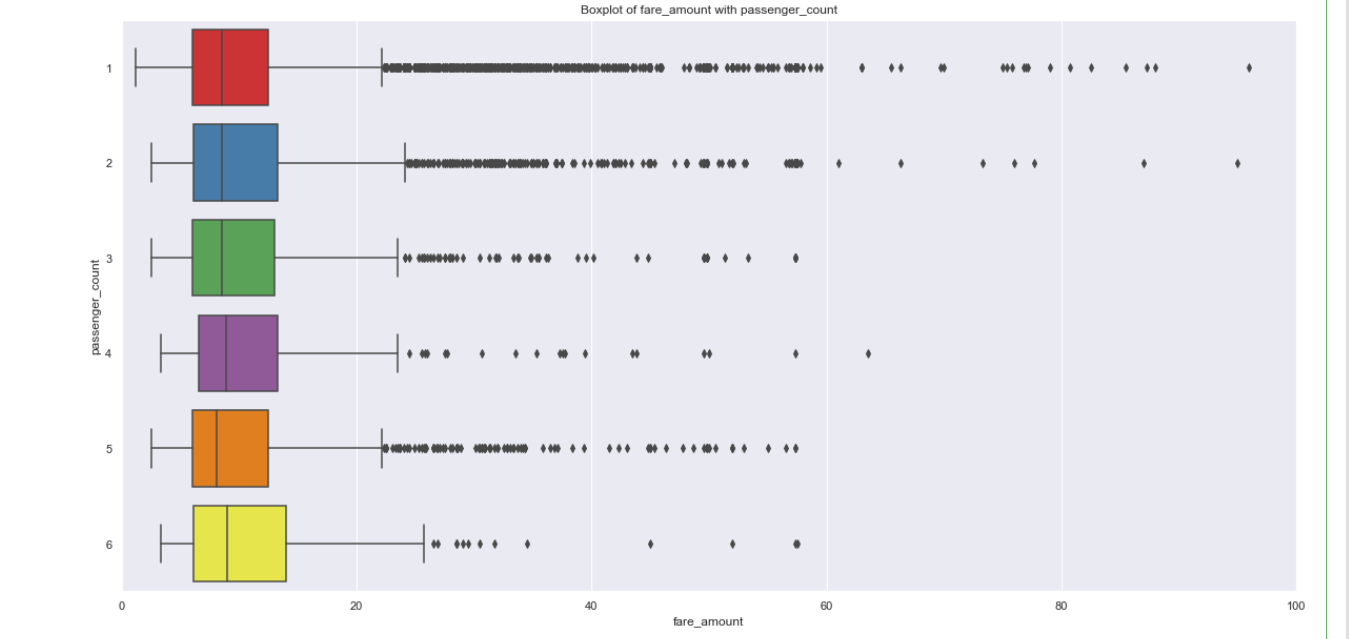
Using the below code, we are representing the outliner graphically.

plt.figure(figsize=(20,10))

plt.xlim(0,100)\_ = sns.boxplot(x=j['fare\_amount'],y=j['passenger\_count'],data=j,orient='h')

plt.title('Boxplot of fare\_amount with passenger\_count')

plt.show()

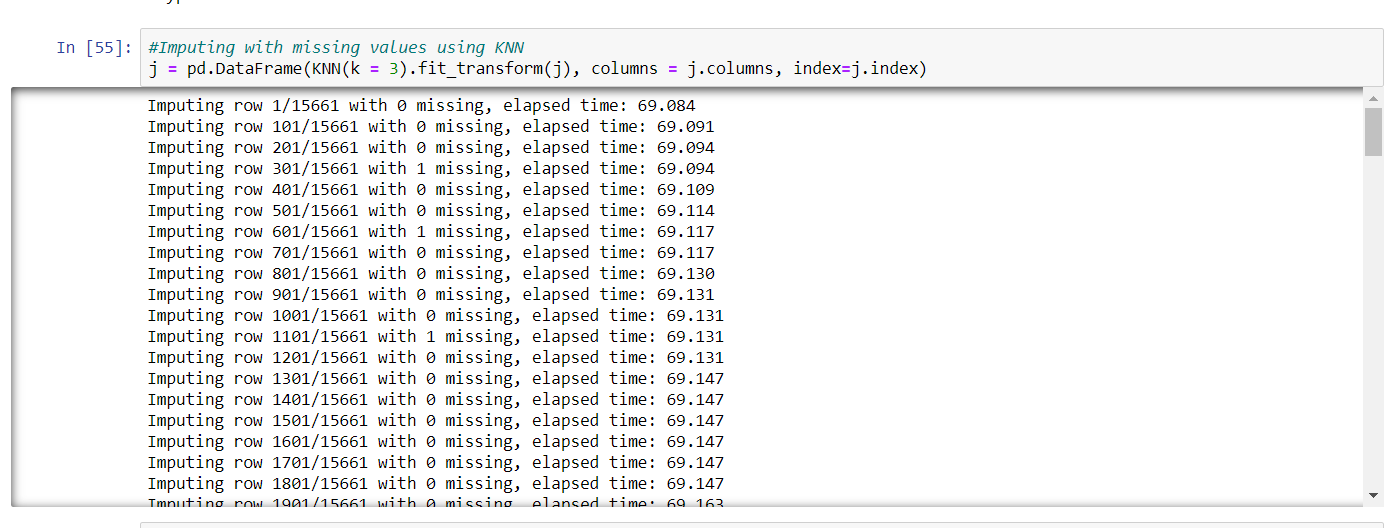


We have found 1358 outliners in fare-amount variable.

We have successfully replaced them with NA and removed them using KNN imputation.

Imputing with missing values using KNN, using following command:

j = pd.DataFrame(KNN(k = 3).fit\_transform(j), columns = j.columns, index=j.index)



**7. Feature Extraction:**

As we know that pickup date-time is a categorical variable of type time-series. We can extract year, month, day from pickup date-time. We can use following command to extract above variables:

data = [j,k]

for i in data:

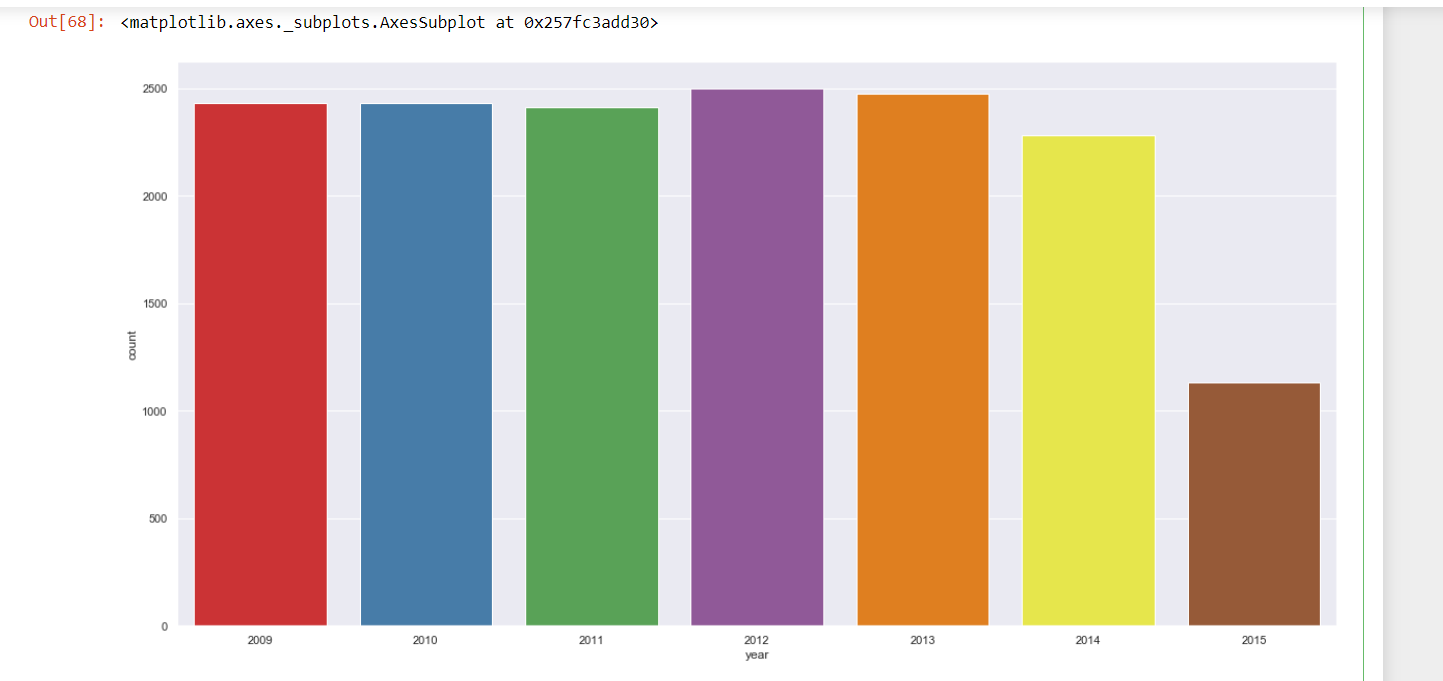
i["year"] = i["pickup\_datetime"].apply(lambda row: row.year)

i["month"] = i["pickup\_datetime"].apply(lambda row: row.month)

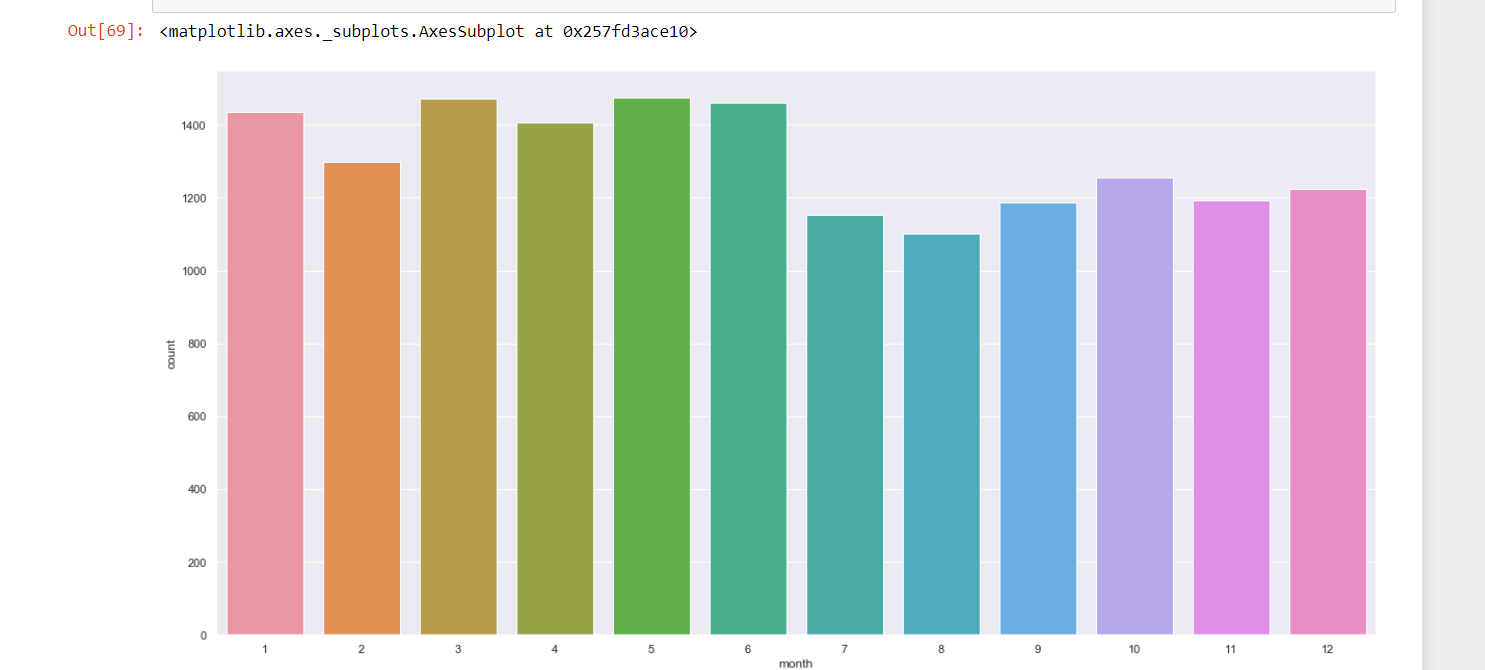
i["day\_of\_week"] = i["pickup\_datetime"].apply(lambda row: row.dayofweek)

i["hour"] = i["pickup\_datetime"].apply(lambda row: row.hour)

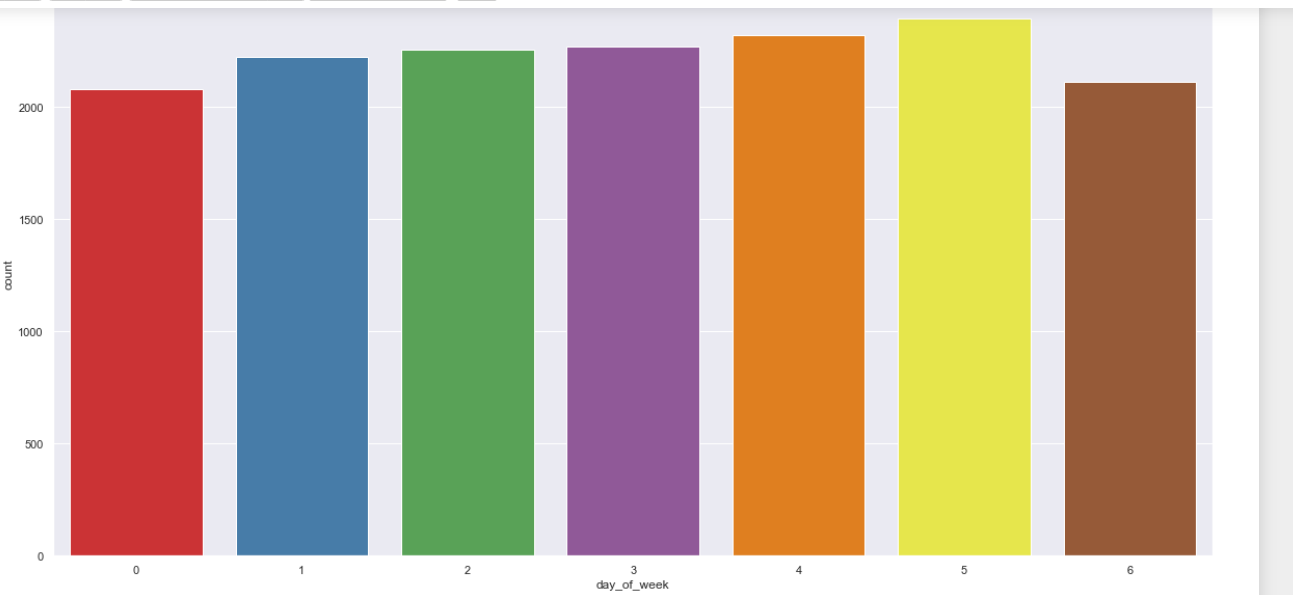
**Plots for year:**



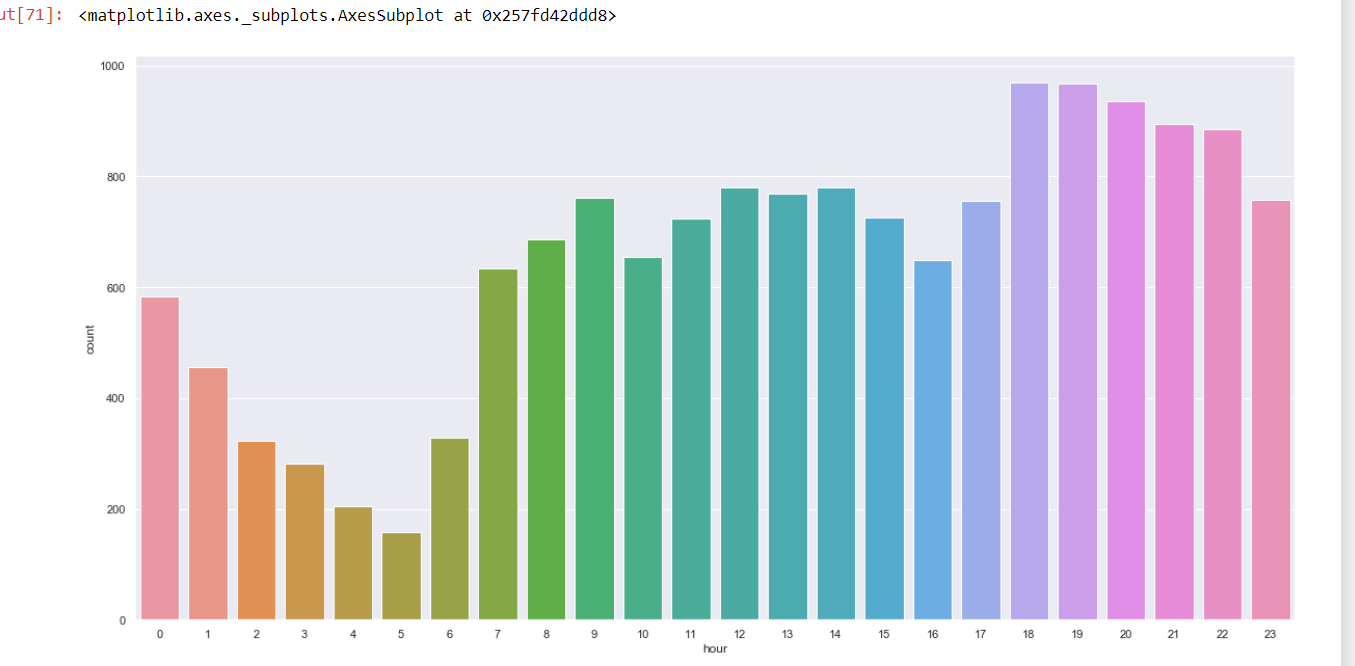
**Plots for month:**

****

**Plots for day:**

****

**Plots for hour:**

****

**We need to calculate distance using longitudes and latitudes:**

Here we calculate the distance from drop-off longitude and latitude, pickup longitude and pickup latitude. Using following command:

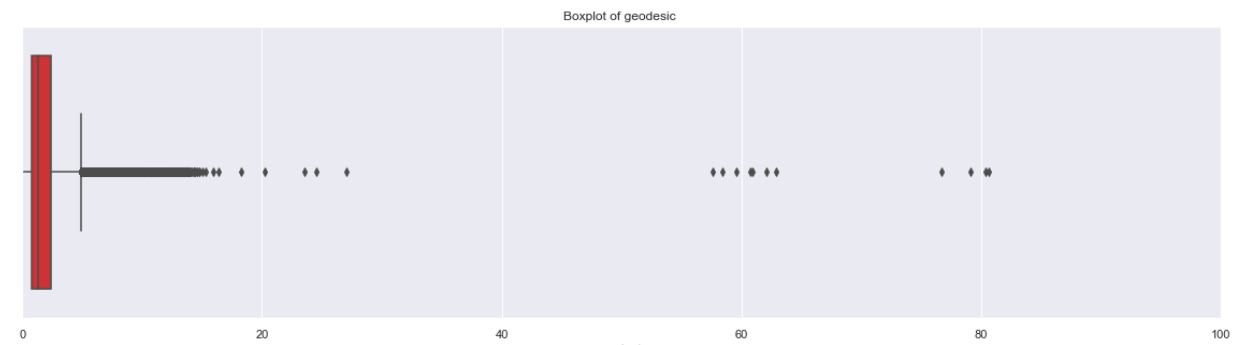
data = [j,k]

for i in data:

i['great\_circle']=i.apply(lambda x: great\_circle((x['pickup\_latitude'],x['pickup\_longitude']), (x['dropoff\_latitude'], x['dropoff\_longitude'])).miles, axis=1)

i['geodesic']=i.apply(lambda x: geodesic((x['pickup\_latitude'],x['pickup\_longitude']), (x['dropoff\_latitude'], x['dropoff\_longitude'])).miles, axis=1)

We create a new variable called distance from above longitudes and latitudes, here is the box plot for distance.



Now we are dropping unwanted variables like pickup longitude, pickup latitude, drop off longitude, drop off latitude. Using following commands:

j=j.drop(['pickup\_datetime','pickup\_longitude', 'pickup\_latitude',

'dropoff\_longitude', 'dropoff\_latitude', 'passenger\_count', 'year',

'month', 'day\_of\_week', 'hour', 'session', 'seasons', 'week','great\_circle'],axis=1)

**8. Feature Selection:**

In this stage we select the variables which are used for target variable prediction. The variables which are not relevant can be removed, by doing so, we can create model which hold high accuracy of data prediction.

As our dataset contains both categorical and numerical variables. We use **Correlation** for numeric data and **Anova & Chi-square test** for categorical data.

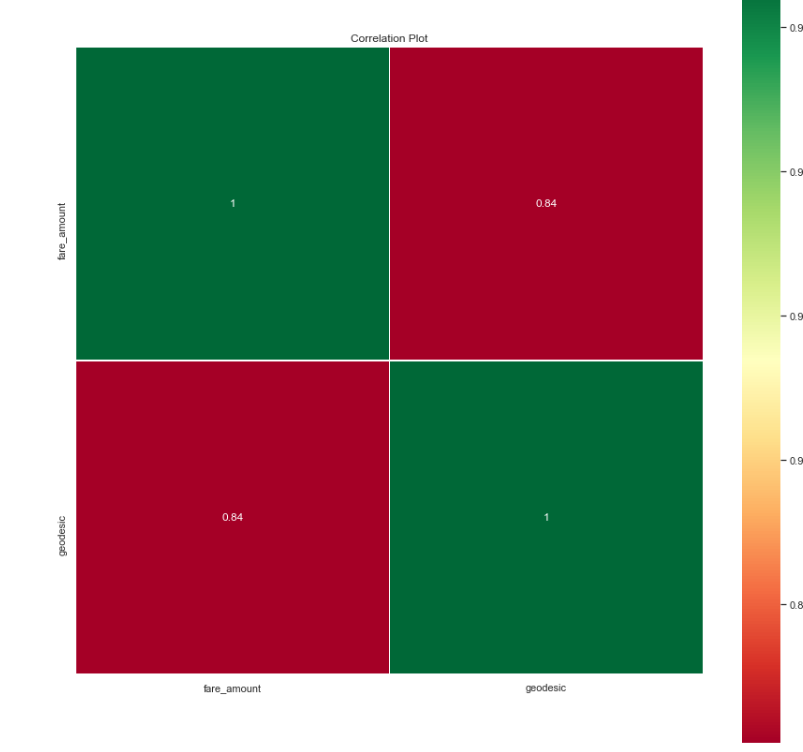
We conduct correlation on numerical data i.e fare-amount and passenger-count.

**Correlation Analysis:** It helps to find the correlation between two independent variables. The correlation between independent variable and dependent variable must be high. If two independent variables are correlated to each other, then we need to choose any one of it.

We correlate numeric variables using following command: plt.figure(figsize=(15,15))

\_ = sns.heatmap(j[num\_var].corr(), square=True, cmap='RdYlGn',linewidths=0.5,linecolor='w',annot=True)

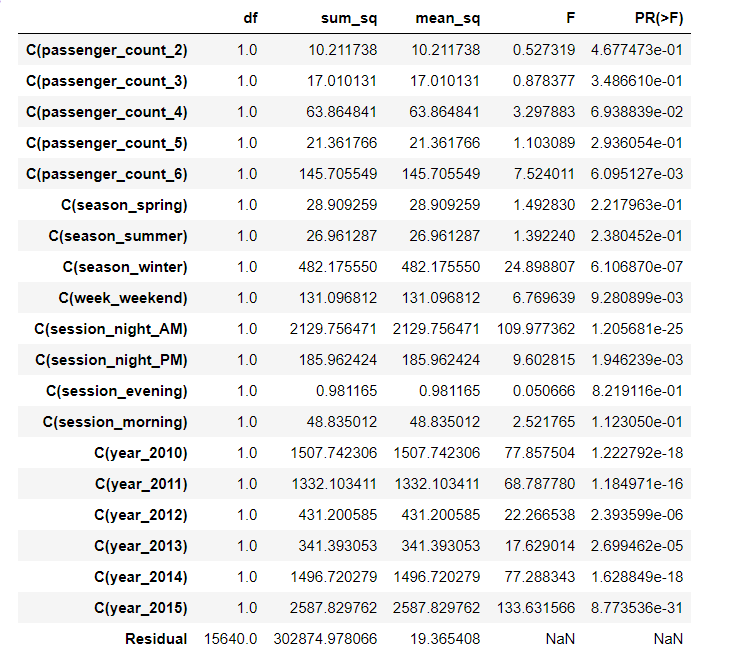
plt.title('Correlation Plot')

plt.show()**Anova:** Here for categorical variables, we are using anova. If p-value is greater than 0.05 then we accept our Null hypothesis saying that two variables are independent. If p-values is less than 0.05, we reject the Null hypothesis saying that two variables are dependent.

We using following command to conduct Anova test on categorical variables

model = ols('fare\_amount ~ C(passenger\_count\_2)+C(passenger\_count\_3)+C(passenger\_count\_4)+C(passenger\_count\_5)+C(passenger\_count\_6)+C(season\_spring)+C(season\_summer)+C(season\_winter)+C(week\_weekend)+C(session\_night\_AM)+C(session\_night\_PM)+C(session\_evening)+C(session\_morning)+C(year\_2010)+C(year\_2011)+C(year\_2012)+C(year\_2013)+C(year\_2014)+C(year\_2015)',data=j).fit()

aov\_table = sm.stats.anova\_lm(model)



1. It is carried out to compare between each group in a categorical variable.
2. ANOVA only lets us know the means for different groups are same or not. It doesn’t help us identify which mean is different.

Null Hypothesis: mean of all categories in a variable are same.

Alternate Hypothesis: mean of at least one category in a variable is different.

If p-value is less than 0.05 then we reject the null hypothesis. And if p-value is greater than 0.05 then we accept the null hypothesis.

**Checking for multi-collinearity:** It is the process of having collinearity that means two or more independent variables are co-related with each other.We use following command to check multi-collinearity between the variables.

outcome, predictors = dmatrices('fare\_amount ~ geodesic+passenger\_count\_2+passenger\_count\_3+passenger\_count\_4+passenger\_count\_5+passenger\_count\_6+season\_spring+season\_summer+season\_winter+week\_weekend+session\_night\_AM+session\_night\_PM+session\_evening+session\_morning+year\_2010+year\_2011+year\_2012+year\_2013+year\_2014+year\_2015',j, return\_type='dataframe')

# calculating VIF for each individual Predictors

vif = pd.DataFrame()

vif["VIF"] = [variance\_inflation\_factor(predictors.values, i) for i in range(predictors.shape[1])]

vif["features"] = predictors.columns

vif

VIF is always greater or equal to 1.

1. if VIF is 1, then Not correlated to any of the variables.
2. if VIF is between 1 to 5, then Moderately correlated.
3. if VIF is above 5, then Highly correlated. If there are multiple variables with VIF greater than 5, only remove the variable with the highest VIF.



**9. Feature Scaling:**

It helps in scaling/measuring data on same units. As, we are aware that different dataset contains different observations of different units, in order to scale them on same units, we use scaling.

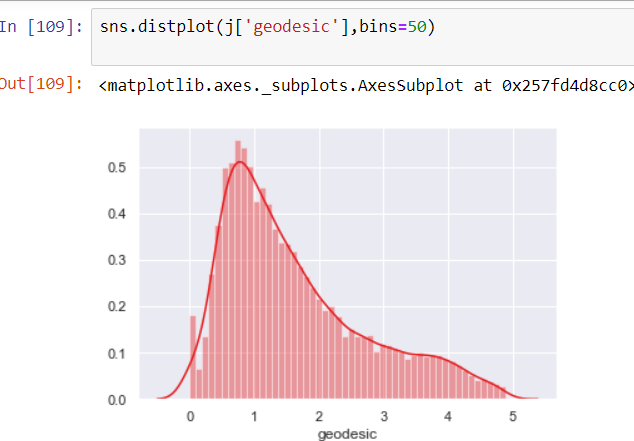
**Normalisation**: It is calculated by dividing the data by its length. It ranges from 0 to 1. It can be checked by using following command:

for i in names:

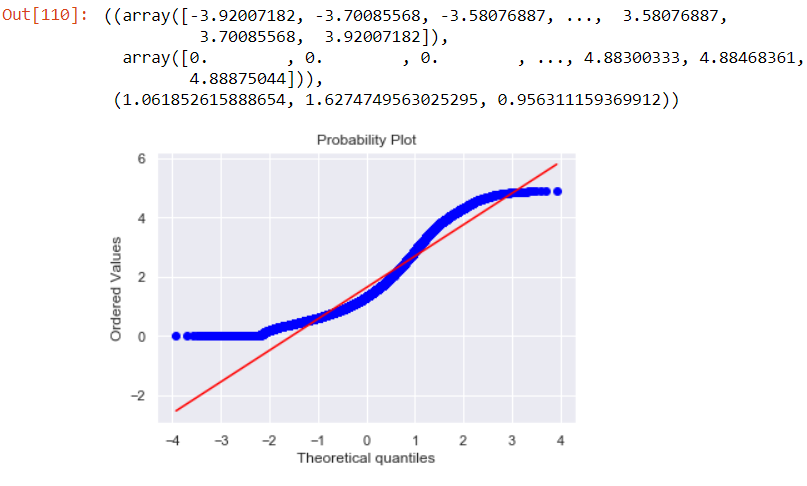
print(i)

j[i] = (j[i] - j[i].min())/(j[i].max() - j[i].min())

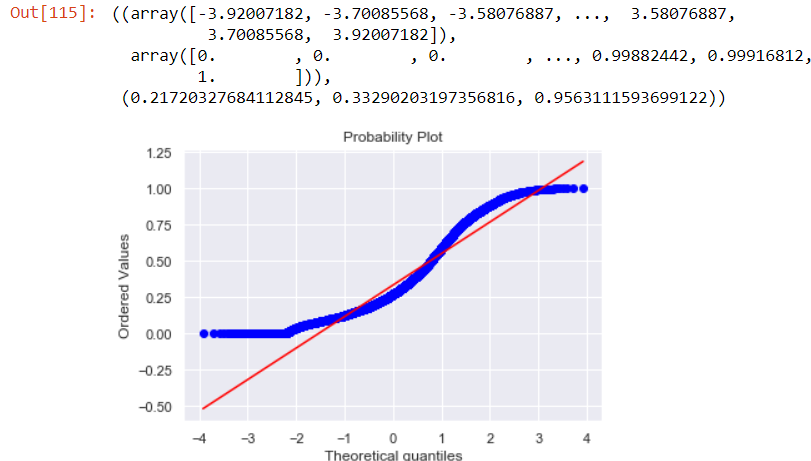
geodisc before Normalisation:



qq probability before Normalisation:



qq probability after Normalisation:



**10. Splitting data into train and test:**

Now before creating model, we need to split the data into train and test data. Here train data has 75% of original train data and test data has 25% of original train data. Using following command:

#Divide data into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state=42)

**11. Model Development:**

Here as we need to predict the data, which has continuous type of values, we are using Regression. We are using 3 types of Regression to predict the values.

1. Multiple Linear Regression
2. Decision Tree Regression
3. Random Forest Regression

The error metrices we are using to evaluate our model is RMSE not MAPE because as our data contains Time-series type of variable called pickup date-time. The RMSE value should be as low as possible for a good model. In the same time, we are also trying to generate values for other metrices like MAE, MSE, RMSE, MAPE, r square, adjusted r square.

**Using Multiple Linear Regression:**

When we try to build a model using Multiple Linear Regression, here are the results:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Error Metrices | r square | Adjusted r square | MAPE | MSE | RMSE | RMSLE |
| Train data | 0.7343 | 0.7339 | 18.7287 | 5.2833 | 2.2985 | 0.2165 |

**Using Decision Tree for Regression:**

When we try to build a model, using Decision Tree Regression. Here are the results:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Error Metrices | r square | Adjusted r square | MAPE | MSE | RMSE | RMSLE |
| Train data | 0.6738 | 0.6732 | 22.6318 | 6.487 | 2.5470 | 0.2427 |

**Using Random Forest Regression:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Error Metrices | r square | Adjusted r square | MAPE | MSE | RMSE | RMSLE |
| Train data | 0.8952 | 0.8950 | 10.506 | 2.0841 | 1.4436 | 0.135 |

The RMSE for above models is less for Random Forest model. Therefore, we can predict the data using Random Forest which has less RMSE.

**12. Conclusion:**

The overall Project is quite challenging, as it takes so much time to sort out the variables and extract the date (year, month, hour, day) from pickup date-time and pickup, dropoff -longitudes and latitudes which helps in calculating distance.

The model we developed can be used for future purpose to predict the cab fare-amount. All we need is to enter the valid data belonging to respected variable.