**Codes in Python**

#importing libraries

import os

os.chdir("C:/Python27/Lib")

os.getwd()

impot pandas as pd

import numpy as np

import matplotlib as mtl

from Sklear.ensemble import RandomForestClassifier

from Sklearn import tree

from Sklearn.metrices import accuracy\_Score

import Sklear.cross\_validation import train\_test\_split

from Sklearn.tree import DecissiontreeRegressor

#loading data

cab = pd.read\_csv(“cab.csv” , sep = “,”)

#removing the 1 variable: “pickup\_datetime”

del pickup\_datetime

#creating dataframe only for missing values

Missing – pd.Date.frame(cab.insull().sum())

Miss = Miss.reset\_index()miss

Miss = Miss.rename(columns = {index “ variable”, ‘0’ : “percentage”})

Miss[‘Precent’] = (Miss[‘percent’]/un(cab))\*100

Miss = Miss.sort\_values(‘missperc’, ascending = F) . reset\_index(drop = T)

#dropping the “passenger\_count”

del passenger\_count

#imputing Missing value

#creating the missing value

Mark[‘fare\_amount’].loc[6] = np.nan

#Actual value = 26

#mean value = 265.26

#knn value = 323.26

# mean method

cab[‘fare\_amount’] = miss[‘fare\_amount’].fillna(cab[‘fare\_amount’].mean())

#knn- imputation

Cab = pd.Dateframe(KNN(K = 3). Fit\_transform(cab), columns = cab.columns)

#Feature selection

Cab\_corr = cab.loc[:, cnames]

Cab\_corr.shape

Cab = cab.drop([“pickup\_longitude”, “dropoff\_latitude”], axis = 1)

#feature scaling

%matplotlib.inline

Plt.hist(cab [‘fare\_amount”] , bins = ‘auto’)

cnames = ["fare\_amount", "pickup\_latitude", "dropoff\_longitude"]

for i in cnames

cab [,i] = (cab[,i] - min(cab[,i]))/ (max(cab[,i] - min(cab[,i])))

#sample

train\_data = cab

test\_data = cab\_pre

#decision Tree

Fit = DecisionTreeRegressor(max\_depth = 2).fit

(train.iloc[:, 0] , train.iloc[: , 1:2])

Prediction = Fit.predict(test.iloc)

#calculating the MAPE

def MAPE (y\_true, y\_ped);

mape = np.mean(np.abs((y\_true – y\_ped)/ y\_true)

return mape

mape(test.iloc[:0], prediction)

# linear regression

Import libraries for LR

Import Statsmodels.api as SM

model = SM.OLS(train.oloc[:, 0], train.iloc[: , 1:2].fit()

model.summary()

Prediction = model.predict(test.iloc)

mape(test.iloc[:0], prediction)