**Practical 1: Uber Price Prediction**

**pip install numpy**

**pip install pandas**

**pip install matplotlib**

**pip install matplotlib.pyplot as plt**

**pip install sklearn**

**pip install sklearn.preprocessing**

**pip install StandardScaler**

**pip install tabulate**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**data=pd.read\_csv('uber.csv')**

**data.info**

**data.shape**

**data.head()**

**data.info()**

**data.isnull().sum()**

**data = data .drop(['Unnamed: 0','key'],axis=1)**

**data .dropna(axis=0,inplace=True)**

**data .isnull().sum()**

**def haversine (lon\_1, lon\_2, lat\_1, lat\_2):**

**lon\_1, lon\_2, lat\_1, lat\_2 = map(np.radians, [lon\_1, lon\_2, lat\_1, lat\_2]) #Degrees to Radians**

**diff\_lon = lon\_2 - lon\_1**

**diff\_lat = lat\_2 - lat\_1**

**km = 2 \* 6371 \* np.arcsin(np.sqrt(np.sin(diff\_lat/2.0)\*\*2 +**

**np.cos(lat\_1) \* np.cos(lat\_2) \* np.sin(diff\_lon/2.0)\*\*2))**

**return km**

**data['Distance']= haversine(data['pickup\_longitude'],data['dropoff\_longitude'],**

**data['pickup\_latitude'],data['dropoff\_latitude'])**

**data['Distance'] = data['Distance'].astype(float).round(2)**

**data.head()**

**plt.scatter(data['Distance'],data['fare\_amount'])**

**plt.xlabel("Distance")**

**plt.ylabel("fare\_amount")**

**data.drop(data[data['Distance'] > 60].index, inplace = True)**

**data.drop(data[data['Distance'] == 0].index, inplace = True)**

**data.drop(data[data['fare\_amount'] == 0].index, inplace = True)**

**data.drop(data[data['fare\_amount'] < 0].index, inplace = True)**

**data.drop(data[(data['fare\_amount']>100) & (data['Distance']<1)].index, inplace = True )**

**data.drop(data[(data['fare\_amount']<100) & (data['Distance']>100)].index, inplace = True )**

**data.info()**

**data['pickup\_datetime'] = pd.to\_datetime(data['pickup\_datetime'])**

**data['Year'] =data['pickup\_datetime'].apply(lambda time: time.year)**

**data['Month'] = data['pickup\_datetime'].apply(lambda time: time.month)**

**data['Day'] = data['pickup\_datetime'].apply(lambda time: time.day)**

**data['Day of Week'] = data['pickup\_datetime'].apply(lambda time: time.dayofweek)**

**data['Day of Week\_num'] = data['pickup\_datetime'].apply(lambda time: time.dayofweek)**

**data['Hour'] = data['pickup\_datetime'].apply(lambda time: time.hour)**

**day\_map = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}**

**data['Day of Week'] = data['Day of Week'].map(day\_map)**

**data['counter'] = 1**

**data['pickup'] = data['pickup\_latitude'].astype(str) + "," + data['pickup\_longitude'].astype(str)**

**data['drop off'] = data['dropoff\_latitude'].astype(str) + "," + data['dropoff\_longitude'].astype(str)**

**data.head()**

**import statistics as st**

**print("Mean of fare prices is % s "**

**% (st.mean(data['fare\_amount'])))**

**print("Median of fare prices is % s "**

**% (st.median(data['fare\_amount'])))**

**print("Standard Deviation of Fare Prices is % s "**

**% (st.stdev(data['fare\_amount'])))**

**print("Mean of Distance is % s "**

**% (st.mean(data['Distance'])))**

**print("Median of Distance is % s "**

**% (st.median(data['Distance'])))**

**print("Standard Deviation of Distance is % s "**

**% (st.stdev(data['Distance'])))**

**X = data['Distance'].values.reshape(-1, 1)**

**y = data['fare\_amount'].values.reshape(-1, 1)**

**from sklearn.preprocessing import StandardScaler**

**std = StandardScaler()**

**y\_std = std.fit\_transform(y)**

**print(y\_std)**

**x\_std = std.fit\_transform(X)**

**print(x\_std)**

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

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(x\_std, y\_std, test\_size=0.2, random\_state=0)**

**from sklearn.linear\_model import LinearRegression**

**l\_reg = LinearRegression()**

**l\_reg.fit(X\_train, y\_train)**

**print("Training set score: {:.2f}".format(l\_reg.score(X\_train, y\_train)))**

**print("Test set score: {:.7f}".format(l\_reg.score(X\_test, y\_test)))**

**y\_pred = l\_reg.predict(X\_test)**

**df = {'Actual': y\_test, 'Predicted': y\_pred}**

**from tabulate import tabulate**

**print(tabulate(df, headers = 'keys', tablefmt = 'psql')**

**from sklearn import metrics**

**print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))**

**print('Mean Absolute % Error:', metrics.mean\_absolute\_percentage\_error(y\_test, y\_pred),"%")**

**print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))**

**print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))**

**print(l\_reg.intercept\_)**

**print(l\_reg.coef\_)**