**1.Uber Ride**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

df=pd.read\_csv('C:/Users/Dell/Desktop/archive (1)/uber.csv')

df

df.head()

df.info()

df.columns

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

df.head()

df.shape

df.dtypes

df.info()

df.describe()

df.isnull().sum()

df['dropoff\_latitude'].fillna(value=df['dropoff\_latitude'].mean(),inplace = True)

df['dropoff\_longitude'].fillna(value=df['dropoff\_longitude'].median(),inplace = True)

df.isnull().sum()

df.pickup\_datetime = pd.to\_datetime(df.pickup\_datetime, errors='coerce')

df.dtypes

df = df.drop('pickup\_datetime',axis=1)

df.head()

df.plot(kind = "box",subplots = True,layout = (7,2),figsize = (15,20))

def remove\_outlier(df1 , col):

Q1 = df1[col].quantile(0.25)

Q3 = df1[col].quantile(0.75)

IQR = Q3 - Q1

lower\_whisker = Q1-1.5\*IQR

upper\_whisker = Q3+1.5\*IQR

df[col] = np.clip(df1[col] , lower\_whisker , upper\_whisker)

return df1

def treat\_outliers\_all(df1 , col\_list):

for c in col\_list:

df1 = remove\_outlier(df , c)

return df1

df = treat\_outliers\_all(df , df.iloc[: , 0::])

df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20))

from math import \*

def distance\_formula(longitude1, latitude1, longitude2, latitude2):

travel\_dist = []

for pos in range (len(longitude1)):

lon1, lan1, lon2, lan2 = map(radians, [longitude1[pos], latitude1[pos], longitude2[pos], latitude2[pos]])

dist\_lon = lon2 - lon1

dist\_lan = lan2 - lan1

a = sin(dist\_lan/2)\*\*2 + cos(lan1) \* cos(lan2) \* sin(dist\_lon/2)\*\*2

#radius of earth = 6371

c = 2 \* asin(sqrt(a)) \* 6371

travel\_dist.append(c)

return travel\_dist

df['dist\_travel\_km'] = distance\_formula(df.pickup\_longitude.to\_numpy(), df.pickup\_latitude.to\_numpy(), df.dropoff\_longitude.to\_numpy(), df.dropoff\_latitude.to\_numpy())

df.head()

df= df.loc[(df.dist\_travel\_km >= 1) | (df.dist\_travel\_km <= 130)]

print("Remaining observastions in the dataset:", df.shape)

df.head()

df.drop(incorrect\_coordinates, inplace = True, errors = 'ignore')

df.head

df.isnull().sum()

sns.heatmap(df.isnull())

corr = df.corr()

corr

fig,axis = plt.subplots(figsize = (10,6))

sns.heatmap(df.corr(),annot = True)

x = df[['pickup\_longitude','pickup\_latitude','dropoff\_longitude','dropoff\_latitude','passenger\_count','dist\_travel\_km']]

y = df['fare\_amount']

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

X\_train,X\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.33)

from sklearn.linear\_model import LinearRegression

regression = LinearRegression()

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

regression.intercept\_

regression.coef\_

prediction = regression.predict(X\_test)

print(prediction)

y\_test

from sklearn.metrics import r2\_score

r2\_score(y\_test,prediction)

from sklearn.metrics import mean\_squared\_error

MSE = mean\_squared\_error(y\_test,prediction)

MSE

RMSE = np.sqrt(MSE)

RMSE

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n\_estimators=100)

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

y\_pred = rf.predict(X\_test)

y\_pred

R2\_Random = r2\_score(y\_test,y\_pred)

R2\_Random

MSE\_Random = mean\_squared\_error(y\_test,y\_pred)

MSE\_Random

RMSE\_Random = np.sqrt(MSE\_Random)

RMSE\_Random