**1:I WILL EXPLAIN:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

plt.rcParams['figure.figsize'] = (12,6)

data = pd.read\_excel('Data\_Train.xlsx')

data.head(5)

data.info()

sns.distplot(data['Price'],kde = False,bins = 30)

sns.barplot(x = 'Price',y = 'Airline',data = data)

sns.countplot(y = 'Airline',data = data)

sns.barplot(x = 'Price',y = 'Source',data = data)

sns.barplot(x = 'Price',y = 'Destination',data = data)

sns.countplot(y = 'Source',data = data)

sns.countplot(y = 'Destination',data = data)

plt.figure(figsize = (14,8))

sns.boxplot(x = 'Price', y = 'Airline', data = data)

sns.countplot(y = 'Total\_Stops',data = data)

data[data['Total\_Stops'] == '4 stops']

sns.boxplot(x = 'Price',y = 'Total\_Stops', data = data)

test = pd.read\_excel('Test\_set.xlsx')

test.head()

sum(data['Airline'].isnull())

sum(data['Total\_Stops'].isnull())

data[data['Total\_Stops'].isnull()]

data[(data['Source'] == 'Delhi') & (data['Destination'] == 'Cochin') & (data['Airline'] == 'Air India') & (data['Duration'] == '23h 40m') & (data['Price'] == 7480)]

data = data.drop(data.index[9039],axis = 0)

data.iloc[9039]

sum(data['Route'].isnull())

sum(data['Arrival\_Time'].isnull())

sum(data['Price'].isnull())

X = data.drop('Price',axis = 1)

y = data['Price'].astype('float32')

**2:DHRUVA WILL EXPLAIN:**

X['Additional\_Info'] = X['Additional\_Info'].replace({'No Info':'No info'})

test['Additional\_Info'] = test['Additional\_Info'].replace({'No Info':'No info'})

X['Total\_Stops'] = X['Total\_Stops'].apply(lambda x: int(x[0]) if x[0]!='n' else 0)

test['Total\_Stops'] = test['Total\_Stops'].apply(lambda x: int(x[0]) if x[0]!='n' else 0)

X['Date'] = X['Date\_of\_Journey'].apply(lambda x: int(x.split('/')[0]))

X['Month'] = X['Date\_of\_Journey'].apply(lambda x: int(x.split('/')[1]))

test['Date'] = test['Date\_of\_Journey'].apply(lambda x: int(x.split('/')[0]))

test['Month'] = test['Date\_of\_Journey'].apply(lambda x: int(x.split('/')[1]))

X['Overnight\_flight'] = X['Arrival\_Time'].apply(lambda x: 1 if len(x) > 5 else 0)

test['Overnight\_flight'] = test['Arrival\_Time'].apply(lambda x: 1 if len(x) > 5 else 0)

X['Date\_of\_Journey'] = X['Date\_of\_Journey'].apply(lambda x: x[-4:]+'-'+x[-7:-5]+'-'+x[-10:-8])

X['Day\_of\_Week'] = pd.to\_datetime(X['Date\_of\_Journey']).dt.dayofweek

test['Date\_of\_Journey'] = test['Date\_of\_Journey'].apply(lambda x: x[-4:]+'-'+x[-7:-5]+'-'+x[-10:-8])

test['Day\_of\_Week'] = pd.to\_datetime(test['Date\_of\_Journey']).dt.dayofweek

X['Airline'] = X['Airline'].replace({'Vistara Premium economy':'Trujet', 'Jet Airways Business':'Trujet','Multiple carriers Premium economy':'Trujet'})

test['Airline'] = test['Airline'].replace({'Vistara Premium economy':'Trujet', 'Jet Airways Business':'Trujet','Multiple carriers Premium economy':'Trujet'})

**3:BVM WILL EXPLAIN:**

plt.scatter(y = X['Day\_of\_Week'],x = y)

# As number of counts for stops are very less we will merge 3 stops and 4 stops

X['Total\_Stops'] = X['Total\_Stops'].replace({4:3})

test['Total\_Stops'] = test['Total\_Stops'].replace({4:3})

X = X.drop(['Date\_of\_Journey','Route','Additional\_Info'],axis = 1)

test = test.drop(['Date\_of\_Journey','Route','Additional\_Info'],axis = 1)

X.head()

cat\_col = ['Airline', 'Source', 'Destination']

from sklearn.preprocessing import LabelEncoder

for cat in cat\_col:

encoder = LabelEncoder()

X[cat] = encoder.fit\_transform(X[cat])

test[cat] = encoder.transform(test[cat])

# Convert Time to hour.

def time\_to\_hour(time):

if 'h' in time:

index\_h = time.index('h')

hour = int(time[:index\_h])

minute = 0

else:

hour = 0

minute = int(time[:-1])

if len(time) > 4:

minute = int(time[-3:-1])

return hour + minute / 60

X['Duration'] = X['Duration'].apply(time\_to\_hour)

test['Duration'] = test['Duration'].apply(time\_to\_hour)

X['Dep\_Time'] = X['Dep\_Time'].apply(lambda x: int(x[:2])+int(x[3:])/60)

test['Dep\_Time'] = test['Dep\_Time'].apply(lambda x: int(x[:2])+int(x[3:])/60)

test['Arrival\_Time'] = test['Arrival\_Time'].apply(lambda x: int(x[:2])+int(x[3:5])/60)

X['Arrival\_Time'] = X['Arrival\_Time'].apply(lambda x: int(x[:2])+int(x[3:5])/60)

X.head()

**4:DEEPAK WILL EXPLAIN:**

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

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size = 0.3,random\_state = 101)

from sklearn.tree import DecisionTreeRegressor

dtree = DecisionTreeRegressor(min\_samples\_split=10)

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

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

from sklearn.metrics import mean\_squared\_log\_error

print("Decision Tree Regressor 1-RMSLE:",1 - mean\_squared\_log\_error(y\_test,y\_pred)\*\*0.5)

from sklearn.svm import SVR

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

y\_pred2 = svr.predict(X\_test)

print("SVR 1-RMSLE:",1 - mean\_squared\_log\_error(y\_test,y\_pred2)\*\*0.5)

from sklearn.ensemble import RandomForestRegressor

forest = RandomForestRegressor(n\_estimators=100,min\_samples\_split=12)

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

y\_pred3 = forest.predict(X\_test)

y\_pred3 = forest.predict(X\_test)

print("Random Forest Regressor 1-RMSLE:",1 - mean\_squared\_log\_error(y\_test,y\_pred3)\*\*0.5)

from sklearn.ensemble import GradientBoostingRegressor

Grad = GradientBoostingRegressor(learning\_rate=0.1,n\_estimators=100)

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

y\_pred4 = Grad.predict(X\_test)

print("Gradient Boosting Regressor 1-RMSLE:",1 - mean\_squared\_log\_error(y\_test,y\_pred4)\*\*0.5)

forest.fit(X\_test,y\_test)

y\_predict = forest.predict(test)

y\_predict

df = pd.DataFrame(y\_predict,columns = ['Price'])

df.to\_excel('answers.xlsx',index = False)

df

y\_pred4

y\_predict

y\_pred3

y\_pred

print(X\_train,y\_train)