Optimizing Flight Booking Decisions through Machine Learning Price Predictions

OVERVIEW:

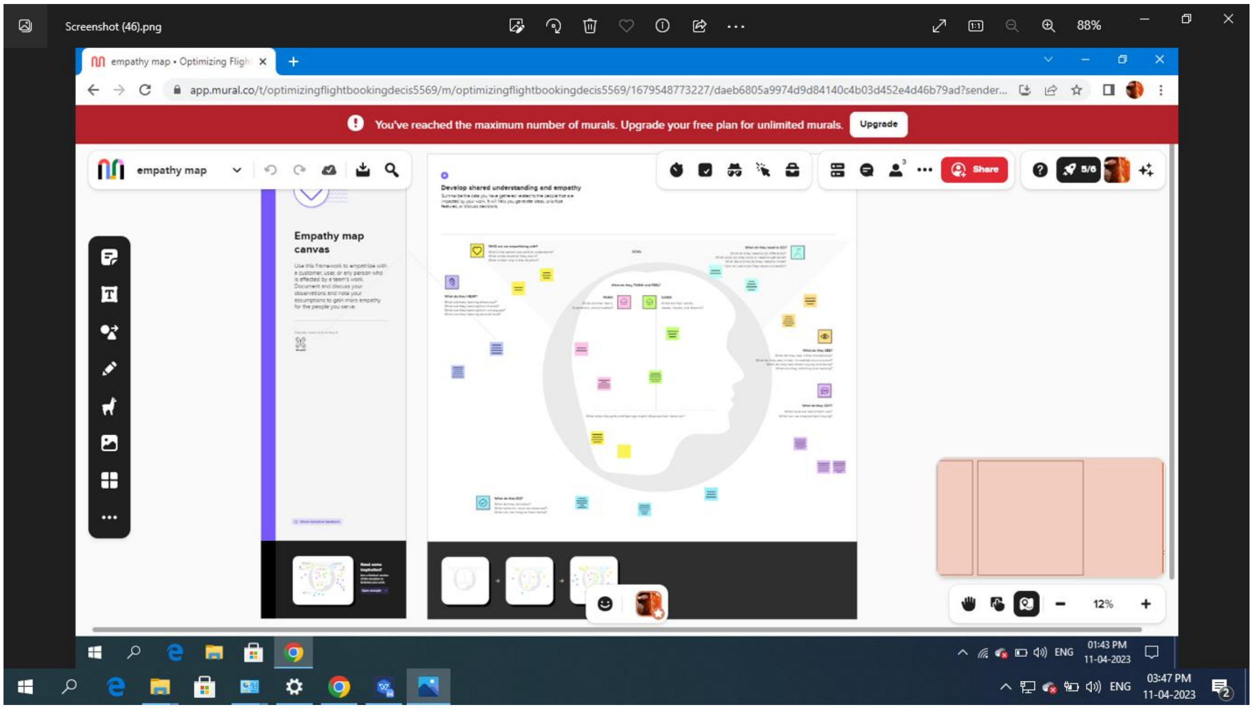
In this article, we will be **analyzing the flight fare prediction using Machine Learning dataset** using essential exploratory data analysis techniques then will **draw some predictions about the price of the flight based on some features** such as what type of airline it is, what is the arrival time, what is the departure time, what is the duration of the flight, source, destination and more.

PURPOSE:

Optimizing Flight Booking Decisions through Machine Learning Price Predictions to provide convenient to passenger.

Problem Definition & Design Thinking

Empathy Map:



Screenshot (46).png
88%
empathy map • Optimizing Fligh
app.mural.co/t/optimizingflightbookingdecis5569/m/optimizingflightbookingdecis5569/1679548773227/daeb6805a9974d9d84140c4b03d452e4d46b79ad?sender...

You've reached the maximum number of murals. Upgrade your free plan for unlimited murals.

Upgrade
empathy map
Share
5/6

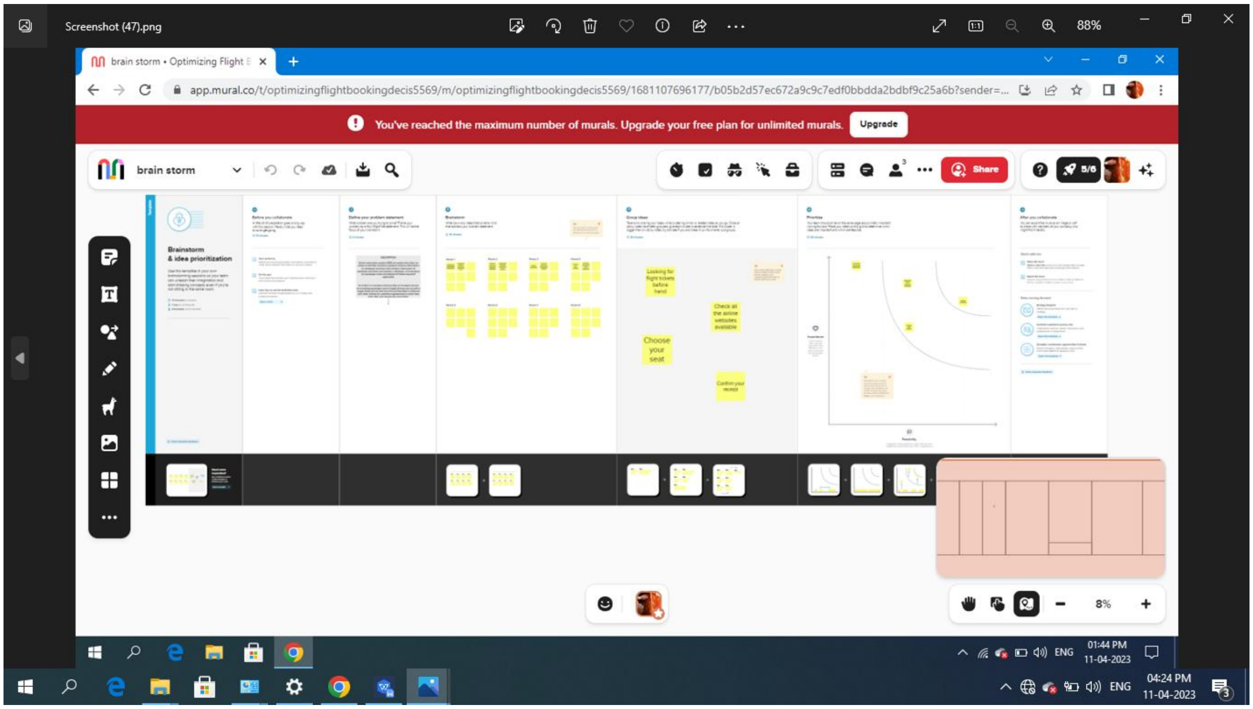
Empathy map canvas

Use this framework to empathize with a customer, user, or any person who is affected by a team's work. Document and discuss your observations and note your assumptions to gain more empathy for the people you serve.

Develop shared understanding and empathy

Summarize the data you have gathered related to the people that are impacted by your work. It will help you generate ideas, prioritize features, or discuss decisions.

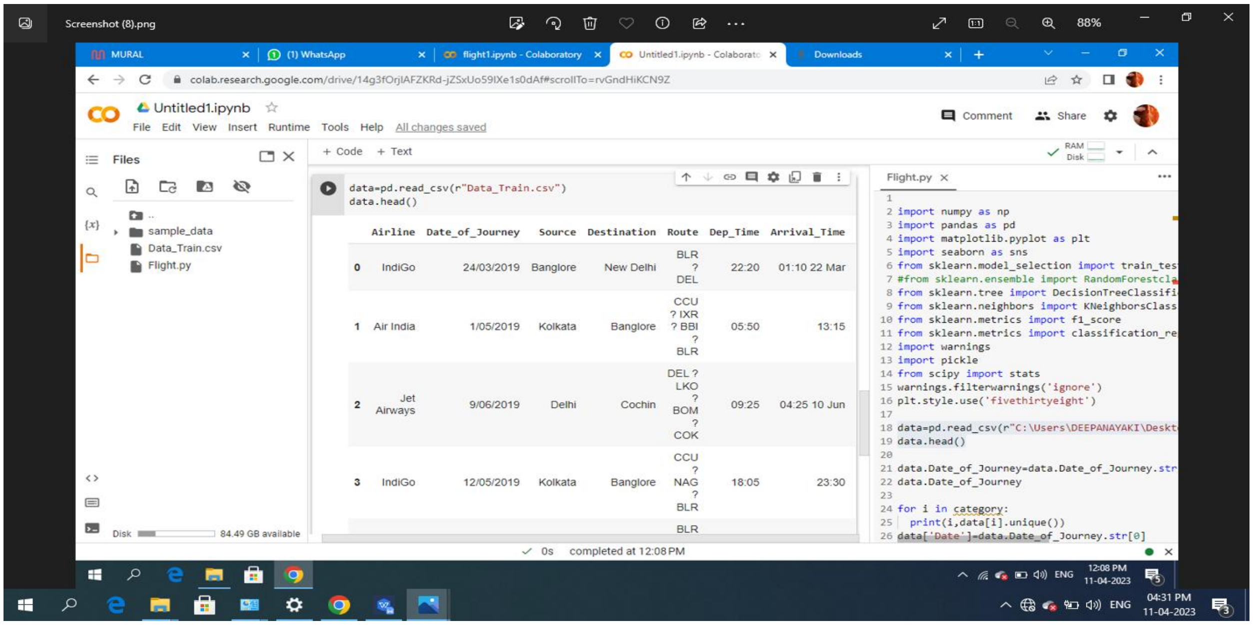
Ideation & Brainstorming Map:



You've reached the maximum number of murals. Upgrade your free plan for unlimited murals.

Screenshot (47).png
88%
brain storm • Optimizing Flight
app.mural.co/t/optimizingflightbookingdecis5569/m/optimizingflightbookingdecis5569/1681107696177/b05b2d57ec672a9c9c7edf0bbdda2bdbf9c25a6b?sender=...
Upgrade
brain storm
Share
5/6
Brainstorm & idea prioritization
Looking for flight tickets before hand
Check all the airline websites available
Choose your seat
Confirm your receipt
8%
ENG
01:44 PM
11-04-2023
04:24 PM
11-04-2023

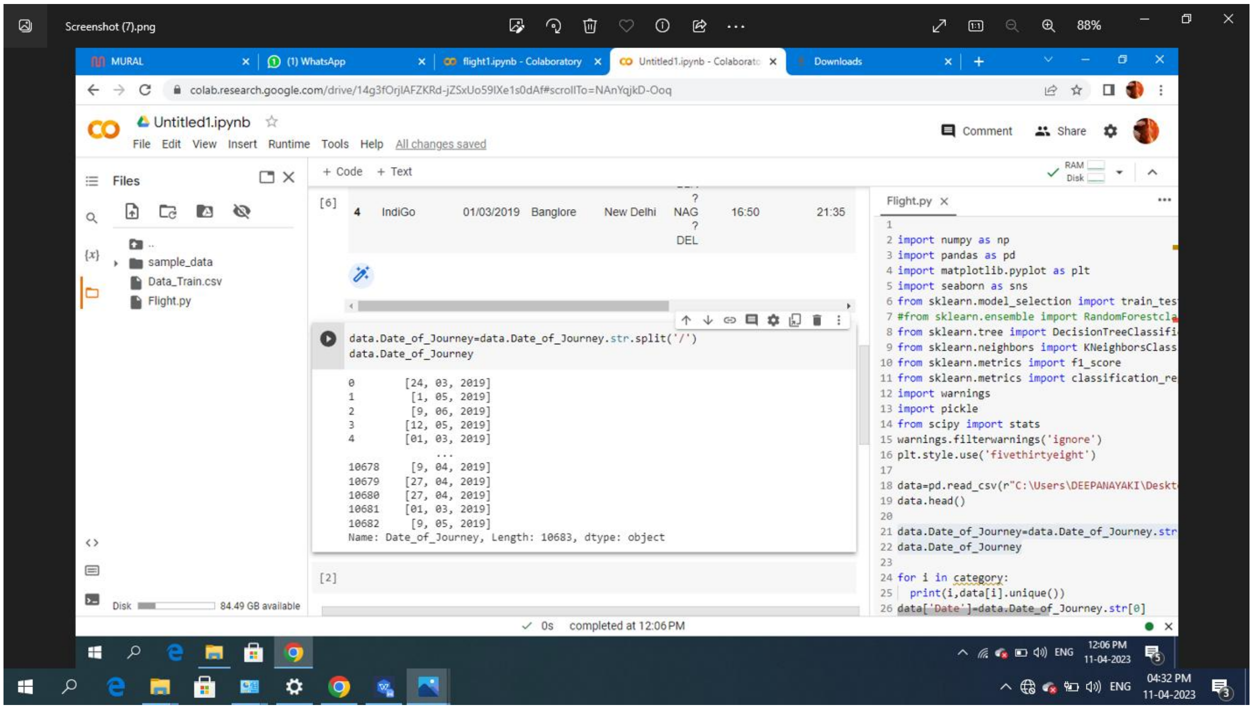
RESULT:


Screenshot (8).png
88%
MURAL
(1) WhatsApp
flight1.ipynb - Colaboratory
Untitled1.ipynb - Colaborato
Downloads
colab.research.google.com/drive/14g3fOrjIAFZKRd-jZSxUo59IXe1s0dAf#scrollTo=rvGndHiKCN9Z
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
sample_data
Data_Train.csv
Flight.py
Disk 84.49 GB available
Flight.py
✓ 0s completed at 12:08 PM
ENG
12:08 PM
11-04-2023
04:31 PM
11-04-2023

```
data=pd.read_csv(r"Data_Train.csv")
data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30
					BLR		

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_tes
#from sklearn.ensemble import RandomForestcla
from sklearn.tree import DecisionTreeClassifi
from sklearn.neighbors import KNeighborsClass
from sklearn.metrics import f1_score
from sklearn.metrics import classification_re
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')

data=pd.read_csv(r"C:\Users\DEEPANAYAKI\Deskt
data.head()

data.Date_of_Journey=data.Date_of_Journey.str
data.Date_of_Journey

for i in category:
  print(i,data[i].unique())
data['Date']=data.Date_of_Journey.str[0]
```


Screenshot (7).png
88%
MURAL
(1) WhatsApp
flight1.ipynb - Colaboratory
Untitled1.ipynb - Colaborato
Downloads
colab.research.google.com/drive/14g3fOrjIAFZKRd-jZSxUo59IXe1s0dAf#scrollTo=NAnYqjkD-Ooq
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
sample_data
Data_Train.csv
Flight.py
Disk 84.49 GB available
Flight.py
✓ 0s completed at 12:06 PM
ENG
12:06 PM
11-04-2023
04:32 PM
11-04-2023

[6]

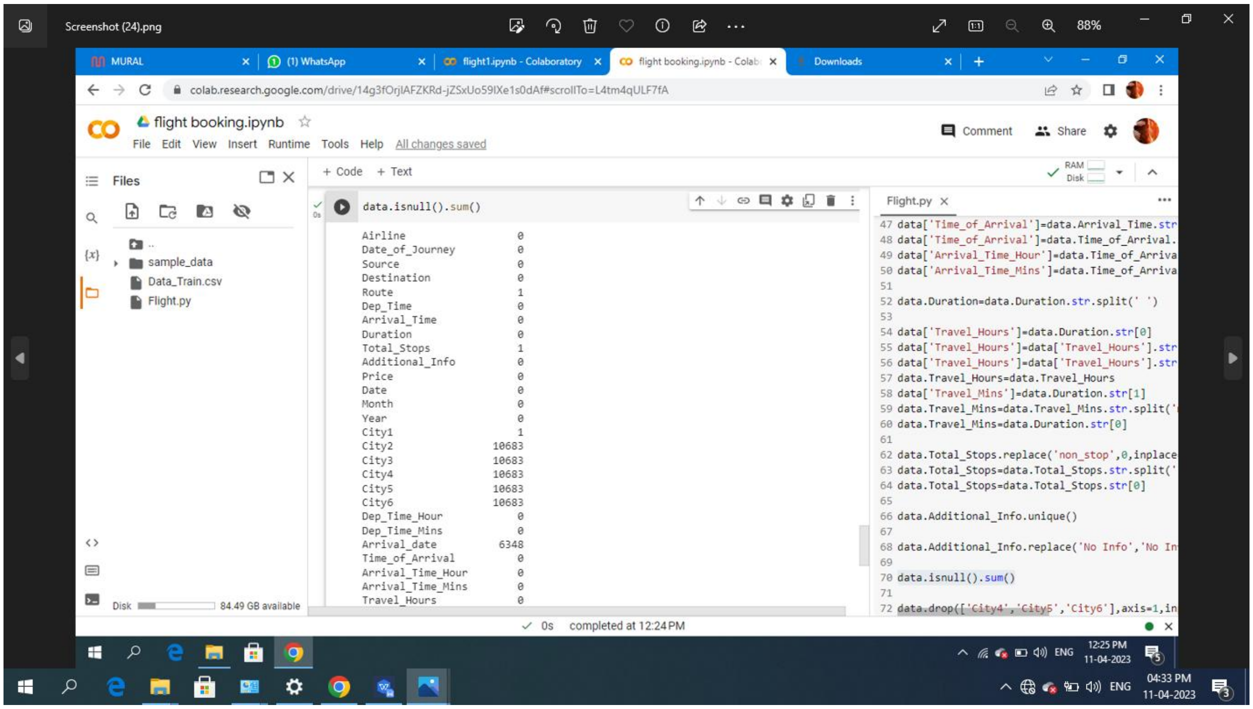
4	IndiGo	01/03/2019	Banglore	New Delhi	? NAG ? DEL	16:50	21:35

```
data.Date_of_Journey=data.Date_of_Journey.str.split('/')
data.Date_of_Journey
```

```
0        [24, 03, 2019]
1         [1, 05, 2019]
2         [9, 06, 2019]
3        [12, 05, 2019]
4        [01, 03, 2019]
              ...
10678     [9, 04, 2019]
10679    [27, 04, 2019]
10680    [27, 04, 2019]
10681    [01, 03, 2019]
10682     [9, 05, 2019]
Name: Date_of_Journey, Length: 10683, dtype: object
```

[2]

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_tes
#from sklearn.ensemble import RandomForestcla
from sklearn.tree import DecisionTreeClassifi
from sklearn.neighbors import KNeighborsClass
from sklearn.metrics import f1_score
from sklearn.metrics import classification_re
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')

data=pd.read_csv(r"C:\Users\DEEPANAYAKI\Deskt
data.head()

data.Date_of_Journey=data.Date_of_Journey.str
data.Date_of_Journey

for i in category:
  print(i,data[i].unique())
data['Date']=data.Date_of_Journey.str[0]
```

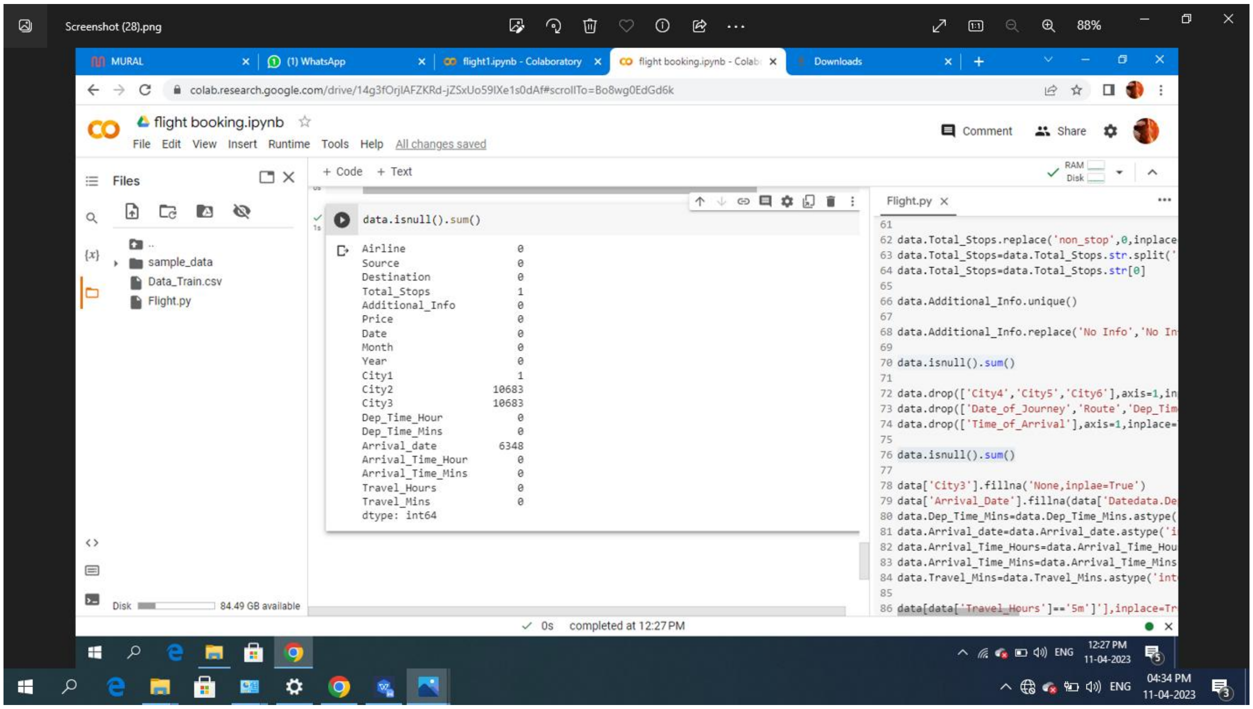


```
data.isnull().sum()
```

```
Airline                0
Date_of_Journey        0
Source                 0
Destination            0
Route                  1
Dep_Time               0
Arrival_Time           0
Duration               0
Total_Stops            1
Additional_Info        0
Price                  0
Date                   0
Month                  0
Year                   0
City1                  1
City2              10683
City3              10683
City4              10683
City5              10683
City6              10683
Dep_Time_Hour          0
Dep_Time_Mins          0
Arrival_date        6348
Time_of_Arrival        0
Arrival_Time_Hour      0
Arrival_Time_Mins      0
Travel_Hours           0
```

Flight.py

```
data['Time_of_Arrival']=data.Arrival_Time.str
data['Time_of_Arrival']=data.Time_of_Arrival.
data['Arrival_Time_Hour']=data.Time_of_Arriva
data['Arrival_Time_Mins']=data.Time_of_Arriva

data.Duration=data.Duration.str.split(' ')

data['Travel_Hours']=data.Duration.str[0]
data['Travel_Hours']=data['Travel_Hours'].str
data['Travel_Hours']=data['Travel_Hours'].str
data.Travel_Hours=data.Travel_Hours
data['Travel_Mins']=data.Duration.str[1]
data.Travel_Mins=data.Travel_Mins.str.split('
data.Travel_Mins=data.Duration.str[0]

data.Total_Stops.replace('non_stop',0,inplace
data.Total_Stops=data.Total_Stops.str.split('
data.Total_Stops=data.Total_Stops.str[0]

data.Additional_Info.unique()

data.Additional_Info.replace('No Info','No In

data.isnull().sum()

data.drop(['City4','City5','City6'],axis=1,in
```

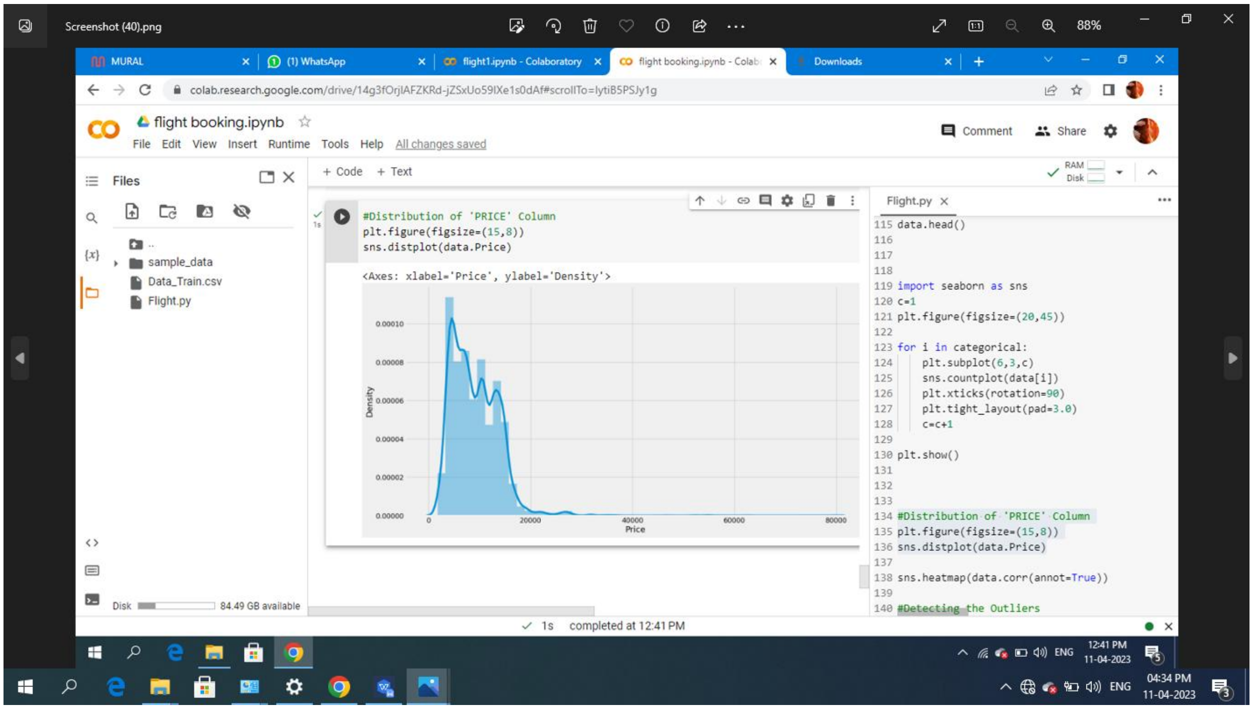


```
data.isnull().sum()
```

```
Airline                0
Source                 0
Destination            0
Total_Stops            1
Additional_Info        0
Price                  0
Date                   0
Month                  0
Year                   0
City1                  1
City2              10683
City3              10683
Dep_Time_Hour          0
Dep_Time_Mins          0
Arrival_date        6348
Arrival_Time_Hour      0
Arrival_Time_Mins      0
Travel_Hours           0
Travel_Mins            0
dtype: int64
```

Flight.py

```

data.Total_Stops.replace('non_stop',0,inplace
data.Total_Stops=data.Total_Stops.str.split('
data.Total_Stops=data.Total_Stops.str[0]

data.Additional_Info.unique()

data.Additional_Info.replace('No Info','No In

data.isnull().sum()

data.drop(['City4','City5','City6'],axis=1,in
data.drop(['Date_of_Journey','Route','Dep_Tim
data.drop(['Time_of_Arrival'],axis=1,inplace=

data.isnull().sum()

data['City3'].fillna('None,inplae=True')
data['Arrival_Date'].fillna(data['Datedata.De
data.Dep_Time_Mins=data.Dep_Time_Mins.astype(
data.Arrival_date=data.Arrival_date.astype('i
data.Arrival_Time_Hours=data.Arrival_Time_Hou
data.Arrival_Time_Mins=data.Arrival_Time_Mins
data.Travel_Mins=data.Travel_Mins.astype('int

data[data['Travel_Hours']=='5m']'],inplace=Tr
```

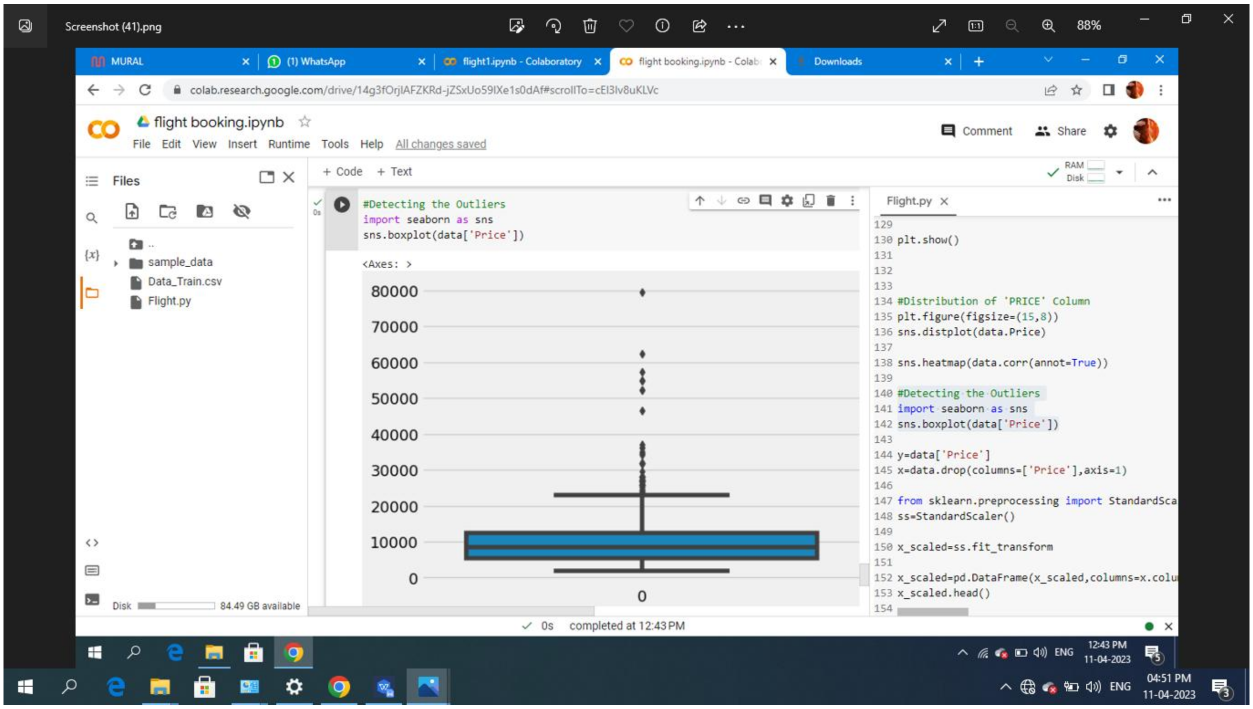


```
#Distribution of 'PRICE' Column
plt.figure(figsize=(15,8))
sns.distplot(data.Price)
```

```
<Axes: xlabel='Price', ylabel='Density'>
```

Flight.py

```
data.head()



import seaborn as sns
c=1
plt.figure(figsize=(20,45))

for i in categorical:
    plt.subplot(6,3,c)
    sns.countplot(data[i])
    plt.xticks(rotation=90)
    plt.tight_layout(pad=3.0)
    c=c+1

plt.show()



#Distribution of 'PRICE' Column
plt.figure(figsize=(15,8))
sns.distplot(data.Price)

sns.heatmap(data.corr(annot=True))

#Detecting the Outliers
```

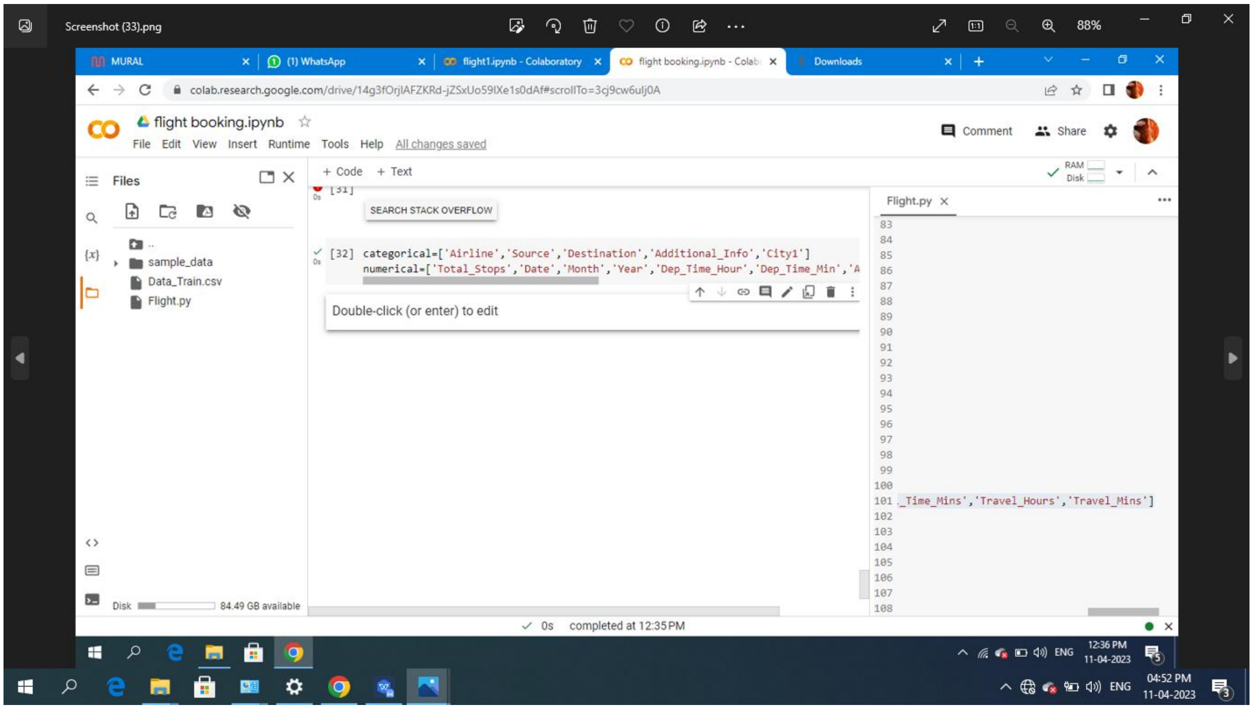


```
#Detecting the Outliers
import seaborn as sns
sns.boxplot(data['Price'])
```

```
<Axes: >
```

Flight.py

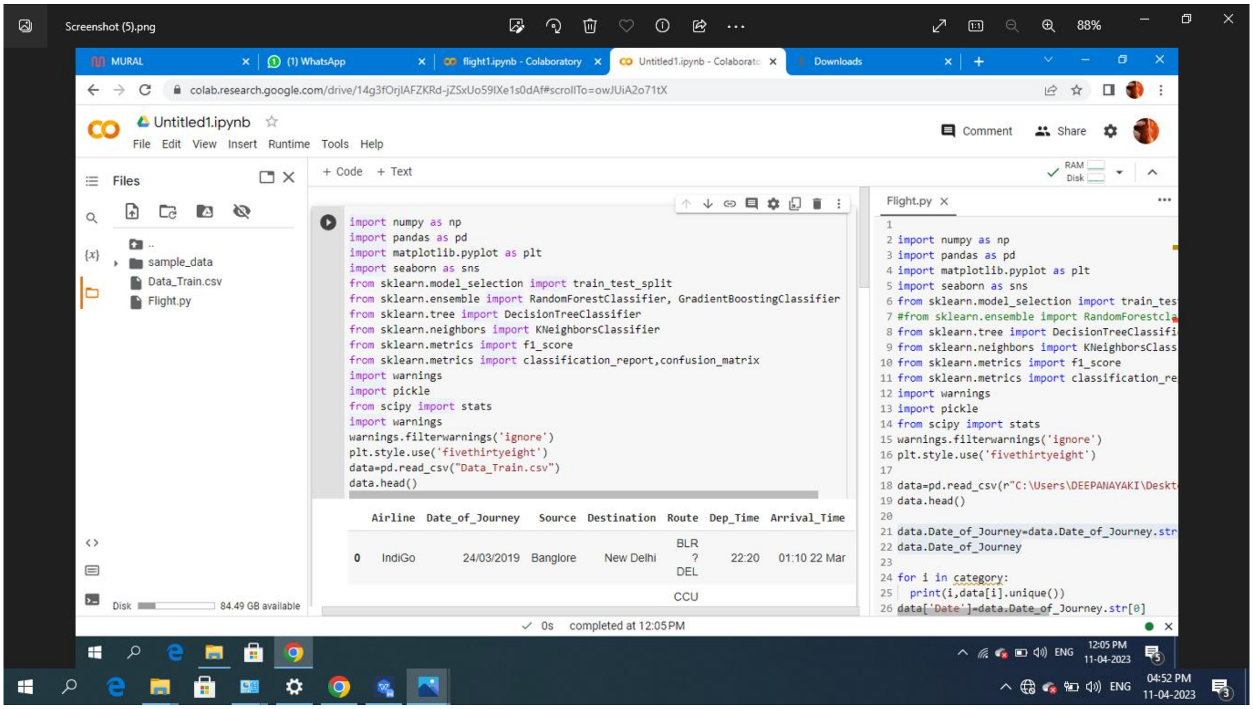
```

plt.show()



#Distribution of 'PRICE' Column
plt.figure(figsize=(15,8))
sns.distplot(data.Price)

sns.heatmap(data.corr(annot=True))

#Detecting the Outliers
import seaborn as sns
sns.boxplot(data['Price'])

y=data['Price']
x=data.drop(columns=['Price'],axis=1)

from sklearn.preprocessing import StandardSca
ss=StandardScaler()

x_scaled=ss.fit_transform

x_scaled=pd.DataFrame(x_scaled,columns=x.colu
x_scaled.head()

```



```
categorical=['Airline','Source','Destination','Additional_Info','City1']
numerical=['Total_Stops','Date','Month','Year','Dep_Time_Hour','Dep_Time_Min','A
```

Flight.py:

```
_Time_Mins','Travel_Hours','Travel_Mins']
```



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report,confusion_matrix
import warnings
import pickle
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
data=pd.read_csv("Data_Train.csv")
data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar
					CCU		

Flight.py:

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_tes
#from sklearn.ensemble import RandomForestcla
from sklearn.tree import DecisionTreeClassifi
from sklearn.neighbors import KNeighborsClass
from sklearn.metrics import f1_score
from sklearn.metrics import classification_re
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')

data=pd.read_csv(r"C:\Users\DEEPANAYAKI\Deskt
data.head()

data.Date_of_Journey=data.Date_of_Journey.str
data.Date_of_Journey

for i in category:
  print(i,data[i].unique())
data['Date']=data.Date_of_Journey.str[0]
```

ADVANTAGES :

Booking airline flights early can give passengers cheaper deals for travel because it gives them time to find and track the best price.

It can also allow them to begin planning their trip early.

While this may not appear to be a huge benefit for airports, cheaper airline tickets can lead to increased flight travel.

DISADVANTAGES:

You need internet access. Reliable internet access is required to check reservations and add bookings that are made over the phone.

You need to be ready for an influx of new customers.

Not all online booking systems are created equal.

APPLICATION:

Flight booking applications help the airline industry automate the booking process. Users worldwide can book flights on the go using the simple apps, which include features such as quick flight search, download tickets, check and modify booking details, one-tap check-in, and many more

CONCLUSION:

Data visualization as well so after these steps one can go for the prediction using machine learning model making steps.

FUTURE SCOPE:

Airline reservation system make the life of passengers very easy as they don’t need to stand in queues for getting their seats reserved and they can easily make reservations on any airline just from a single system.

APPENDIX:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
#from sklearn.ensemble import RandomForestclassifier,GradientBoosting
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report,confusion_matrix
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')

data=pd.read_csv(r"C:\Users\DEEPANAYAKI\Desktop\New folder\Data_Train.csv")
data.head()

data.Date_of_Journey=data.Date_of_Journey.str.split('/')
data.Date_of_Journey

for i in category:
  print(i,data[i].unique())
data['Date']=data.Date_of_Journey.str[0]
```

```
data['Month']=data.Date_of_Journey.str[1]
data['Year']=data.Date_of_Journey.str[2]
data.Total_Stops.unique()
data.Route=data.Route.str.split('->')
data.Route

data['City1']=data.Route.str[0]
data['City2']=data.Route.str[1]
data['City3']=data.Route.str[2]
data['City4']=data.Route.str[3]
data['City5']=data.Route.str[4]
data['City6']=data.Route.str[5]

data.Dep_Time=data.Dep_Time.str.split(':')
data['Dep_Time_Hour']=data.Dep_Time.str[0]
data['Dep_Time_Mins']=data.Dep_Time.str[1]

data.Arrival_Time=data.Arrival_Time.str.split(' ')

data['Arrival_date']=data.Arrival_Time.str[1]
data['Time_of_Arrival']=data.Arrival_Time.str[0]
data['Time_of_Arrival']=data.Time_of_Arrival.str.split(':')
data['Arrival_Time_Hour']=data.Time_of_Arrival.str[0]
data['Arrival_Time_Mins']=data.Time_of_Arrival.str[1]

data.Duration=data.Duration.str.split(' ')
```

```
data['Travel_Hours']=data.Duration.str[0]

data['Travel_Hours']=data['Travel_Hours'].str.split('h')

data['Travel_Hours']=data['Travel_Hours'].str[0]

data.Travel_Hours=data.Travel_Hours

data['Travel_Mins']=data.Duration.str[1]

data.Travel_Mins=data.Travel_Mins.str.split('m')

data.Travel_Mins=data.Duration.str[0]


data.Total_Stops.replace('non_stop',0,inplace=True)

data.Total_Stops=data.Total_Stops.str.split(' ')

data.Total_Stops=data.Total_Stops.str[0]


data.Additional_Info.unique()


data.Additional_Info.replace('No Info','No Info',inplace=True)


data.isnull().sum()


data.drop(['City4','City5','City6'],axis=1,inplace=True)

data.drop(['Date_of_Journey','Route','Dep_Time','Arrival_Time','Duration'],axis=1,
inplace=True)

data.drop(['Time_of_Arrival'],axis=1,inplace=True)


data.isnull().sum()


data['City3'].fillna('None,inplae=True')
```

```
data['Arrival_Date'].fillna(data['Datedata.Dep_Time_Hur=data.Dep_Time_Hour.astype('int64')

data.Dep_Time_Mins=data.Dep_Time_Mins.astype('int64')

data.Arrival_date=data.Arrival_date.astype('int64')

data.Arrival_Time_Hours=data.Arrival_Time_Hours.astype('int64')

data.Arrival_Time_Mins=data.Arrival_Time_Mins.astype('int64')

data.Travel_Mins=data.Travel_Mins.astype('int64')


data[data['Travel_Hours']=='5m']'],inplace=True)

data['Travel_Mins'].fillna(0,inplace=True)


data.info()


#data.Date_of_Journey=data.Date_of_Journey.astype('int64')

#data.Month=odata.Month.astype('int64')

#data.Year=data.Year.astype('int64')


#data.drop(index=6474,inplace=True,axis=0)


data.Travel_Hours=data.Travel_Hours.astype('int64')


categorical=['Airline','Source','Destination','Additional_Info','City1']

numerical=['Total_Stops','Date','Month','Year','Dep_Time_Hour','Dep_Time_Min','Arrival_date','Arrival_Time_Hour','Arrival_Time_Mins','Travel_Hours','Travel_Mins']


from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()
```

```
data.Airline=le.fit_transform(data.Airline)
data.Source=le.fit_transform(data.Source)
data.Destination=le.fit_transform(data.Destination)
data.Total_Stops=le.fit_transform(data.Total_Stops)
data.City1=le.fit_transform(data.City1)
data.City2=le.fit_transform(data.City2)
data.City3=le.fit_transform(data.City3)
data.Adddditional_Info=le.fit_transform(data.Additional_Info)
data.head()

import seaborn as sns
c=1
plt.figure(figsize=(20,45))

for i in categorical:
    plt.subplot(6,3,c)
    sns.countplot(data[i])
    plt.xticks(rotation=90)
    plt.tight_layout(pad=3.0)
    c=c+1

plt.show()

#Distribution of 'PRICE' Column
```

```
plt.figure(figsize=(15,8))
sns.distplot(data.Price)

sns.heatmap(data.corr(annot=True))

#Detecting the Outliers
import seaborn as sns
sns.boxplot(data['Price'])

y=data['Price']
x=data.drop(columns=['Price'],axis=1)

from sklearn.preprocessing import StandardScaler
ss=StandardScaler()

x_scaled=ss.fit_transform

x_scaled=pd.DataFrame(x_scaled,columns=x.columns)
x_scaled.head()

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)

x_train.head()

from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor
```

```
rfr=RandomForestsRegressor()
gb=GradientBoostingRegressor()
ad=AdaBoostRegressor()


from sklearn.metrics import r2_Score,mean_absolute_error,mean_squared_error


for i in [rfr,gb,ad]:
    i.fit(x_train,y_train)
    y_pred=i.predict(x_test)
    test_score=r2_score(y_test,y_pred)
    train_score=r2_score(y_train,i.predict(x_train))
    if abs(train_score-test_score)<=0.2:
       print(i)

       print("R2 score is",r2_score(y_test,y_pred))
       print("R2 for train data",r2_score(y_train,i.predict(x_train))
       print("Mean Absolue Error is",mean_absolute_error(y_pred,y_test))
       print("Mean Squared Error is",mean_squared_error(y_pred,y_test))
       print("Root Mean Sqaured Error is",(mean_Squared_error(y_pred,y_test,squared-False)))
```

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error


knn=KNeighborsRegressor()

svr=SVR()

dt=DecisionTreeRegressor()


for i in [knn,svr,dt]:

    i.fit(x_train,y_train)

    y_pred-i.predict(x_test)

    test_score=r2_score(y_test,y_pred)

    train_score=r2score(y_train,i.predict(x_train))

    if abs(train_score-test_score)<=0.1:

       print(i)

print('R2 Score is',r2_score(y_test,y_pred))

print('R2 Score for train data',r2_score(y_train,i,predict(x_train)))

print('Mean Absolute Error is',mean_absolute_error(y_test,y_pred))

print('Mean squared Error is',mean_squared_error(y_test,y_pred))

print('Root Mean Squared Error is',(mean_squared_error(y_test,y_pred,squared=False)))




from sklearn.model_selection import cross_val_score

for i in range(2,5):

    cv=cross_val_score(rfr,x,y,cv=i)

    print(rfr,cv.mean())
```

```
rfr=RandomForestRegressor(n_estimators=10,max_features='sqrt',max_depth=None)

rfr.fit(x_train,y_train)

y_train_pred=rfr.predict(x_train)

y_test_pred=rfr.predict(x_test)

print("train accurancy",r2_score(y_train_pred,y_train))

print("test accurancy",r2_score(y_test_pred,y_test))


knn=KNeighboursRegressor(n_neighbors=2,algorithm='auto',metric_params=None,n_jobs=-1)

Knn.fit(x_train,y_train)

y_train_pred=knn.predict (x_train)

y_test_pred=Knn.predict(x_test)

print("train accurancy",r2_score(y_train_pred,y_train))

print("test accurancy",r2_score(y_test_pred,y_test))


import pickle

pickle.dump(rfr,open('model1.pkl','wb'))


from Flask import Flask render_template,'request'

import numpy as np

import pickle


model=pickle.load(open(r"mode[].prl",'rb'))
```

```
@app,route("/home")
def home():
    return render_template('home.html')

@app.route("/predict")
def home1():
    return render_template('predict.html')

@app.route("/pred",methods=['POST','GET'])
def predict():
    x=[[int(x) for xin request.form.values()]]
    print(x)

    x=np.array(x)
    print(x.shape)

    print(x)
    pred=model.predict(x)
    print(pred)
    return render_template('submit.html',predicition_text=pred)

if __name__ == "__main__":
    app.run(debug=False)

from tensortflow.kers.models import Sequential
from tensortflow.keras.layers import Dense
```

```
#Fitting the model to the training sets
model = Sequential()
```

```
x_train.shape
```

```
 (4457, 7163)
```

```
model.add(Dense(units =x_train_res.shape[1],activation="relu",kernel_initializer="random_uniform"))
```

```
model.add(Dense(units=100,activation="relu",kernel_initializer="random_uniform"))
```

```
model.add(Dense(units=100,activation="relu",kernel_initializer="random_uniform"))
```

```
model.add(Dense(units*1,activation="sigmoid"))
```

```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
```

```
generator=model.fit(x_train_res,y_train_res,epochs=10,steps_per_epoch=len(x_train_res)//64)
```

```
from sklearn.naive_bayes import MultinomialNB
model=multinomialNB()
```

```
#Fitting the model to the training sets
model.fit(x_train_res,y_train_res)
```