23-04-2023

OPTIMIZING FLIGHT BOOKING DECISIONS THROUGH MACHINE LEARNING PRICE PREDICTIONS

DONE BY

K.SOWMIYA

G.NIVETHA GNANAPOO

S.MAHESWARI

G.POOPATHI

INTRODUCTION

1.1 Overviews:

The Flight ticket prices increase or decrease every now and then depending on various factors like timing of the flights, destination, duration of flights.

In the proposed system a predictive model will be created by applying machine learning algorithms to the collected historical data of flights.

The process of flight booking price prediction using machine learning typically involves the following the steps:

1. Data Collection: Historical data on flight prices and other relevant factors, such as the time of year, destination, and airline, are collected from various sources.

2. Data Preprocessing: The collected data is cleaned and processed to remove any missing values, outliers, or errors that may affect the accuracy of the model.

3. Feature Engineering: Relevant features are identified and extracted from the data to be used as inputs to the machine learning algorithm. For example, features such as the departure and arrival airports, flight date and time, and airline can be used to predict flight prices.

4. Model Training: The preprocessed data is used to train the machine learning algorithm. Various algorithms can be used for this purpose, such as regression models, decision trees, and neural networks.

5. Model Evaluation: The trained model is tested on a separate dataset to evaluate its accuracy and performance. Different metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared can be used to evaluate the model's performance.

6. Model Deployment: Once the model is trained and evaluated, it can be deployed to predict the prices of future flights. The model can be integrated into travel booking websites or travel agencies to assist customers in finding the best deals on flights.

Flight booking price prediction using machine learning can help customers save money by predicting the optimal time to book a flight, and it can also help airlines optimize their pricing strategies to increase revenue.

A Brief Description about Project:

People who work frequently travel through flight will have better knowledge on best discount and right time to buy the ticket.

For the business purpose many airline companies change prices according to the seasons or time duration. They will increase the price when people travel more. Estimating the highest prices of the airlines data for the route is collected with features such as Duration, Source, Destination, Arrival and Departure.

Features are taken from chosen dataset and in the price wherein the airline price ticket costs vary overtime. we have implemented flight price prediction for users by using KNN, decision tree and random forest algorithms. Random Forest shows the best accuracy of 80% for predicting the flight price. also, we have done correlation tests and metrics for the statistical analysis.

1.2 Purpose:

The purpose of building a flight price prediction model using machine learning is help users make informed decisions about their travel plans by providing them with accurate and timely predictions of flight prices.

This can benefits users in various ways, such as saving money, by predicting flight prices in advance, users can make more informed decisions about when to book their flights, potentially saving money on their travel expenses.

Planning Travel: Users can plan their travel in advance, knowing when and where they can get the best deals on flights, making their travel planning more convenient.

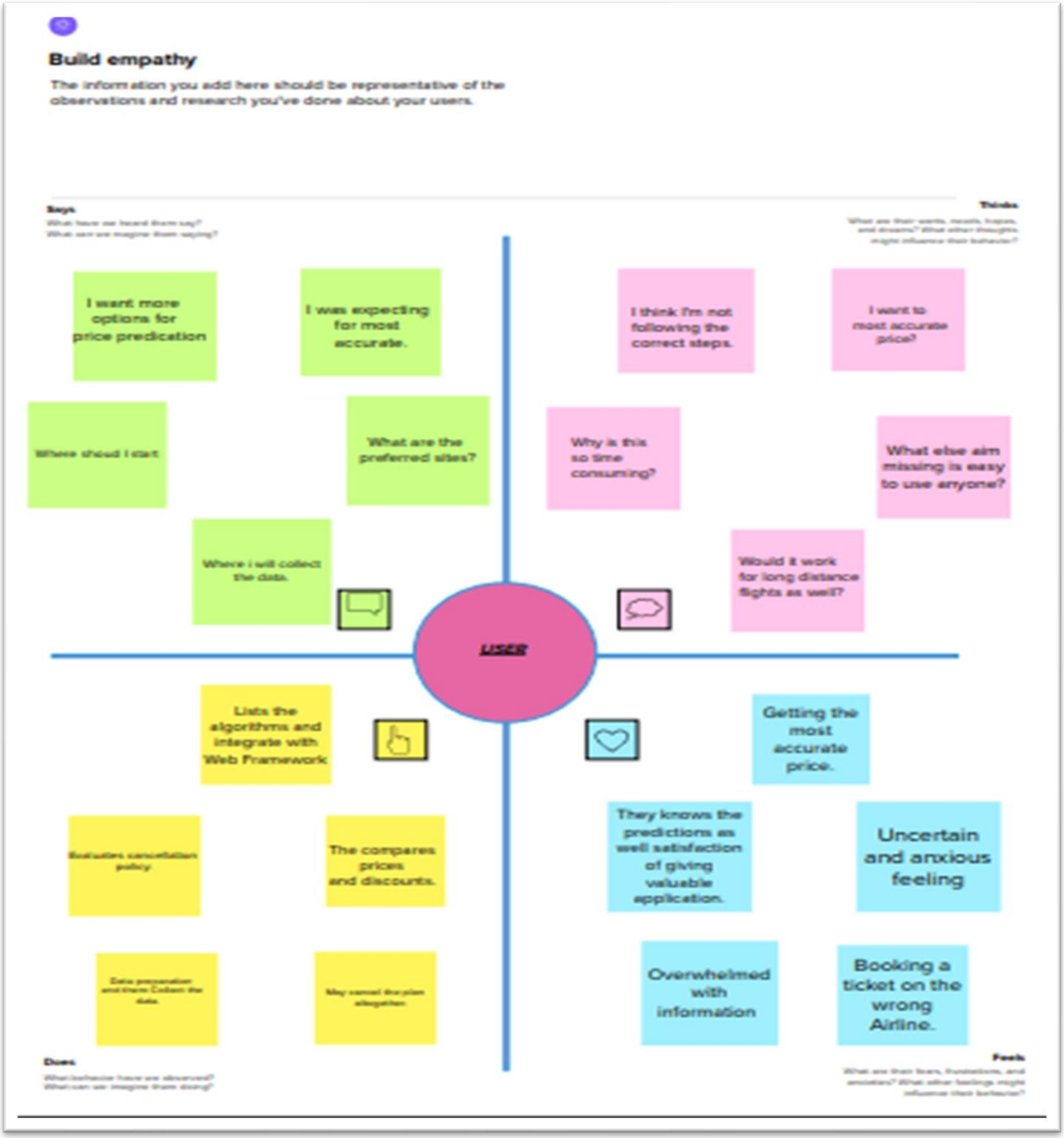
Avoiding Last-Minute Price Hikes: With the help of the flight price prediction model, users can avoid last-minute price hikes that may occur due to increased demand or other factors.

Business Travel: Business travelers can use the model to optimize their travel budget and make informed decisions about when to book their flights.

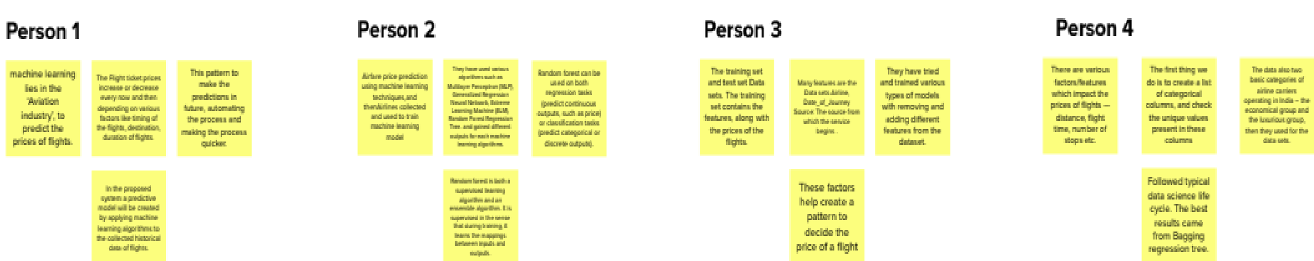
Overall, building a flight price prediction model using machine learning can provide valuable insights to users, help them save money, and make their travel planning more convenient and efficient.

Problem Definition & Design Thinking

2.1 Empathy Map



2.2 Ideation & Brainstorming Map

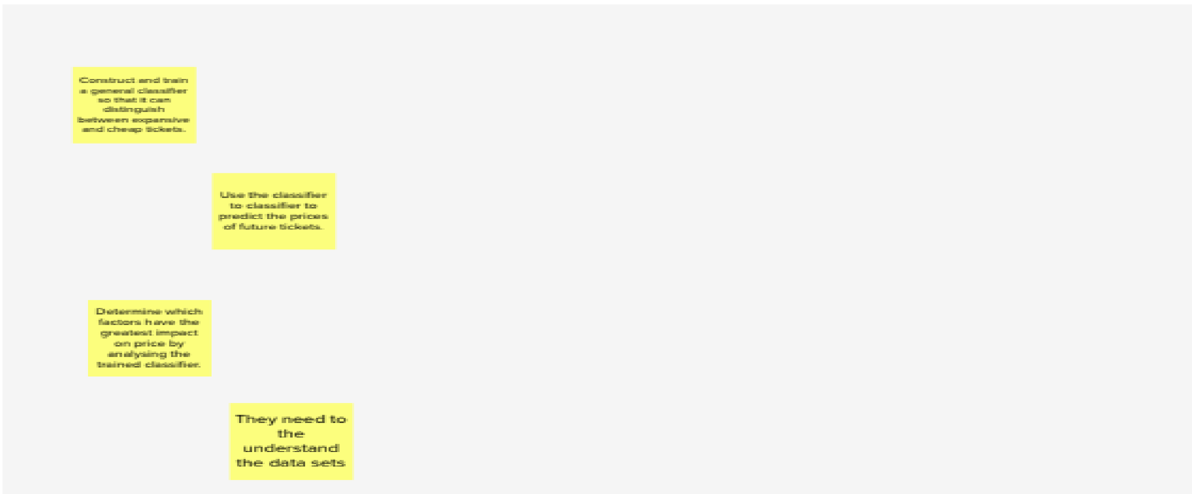


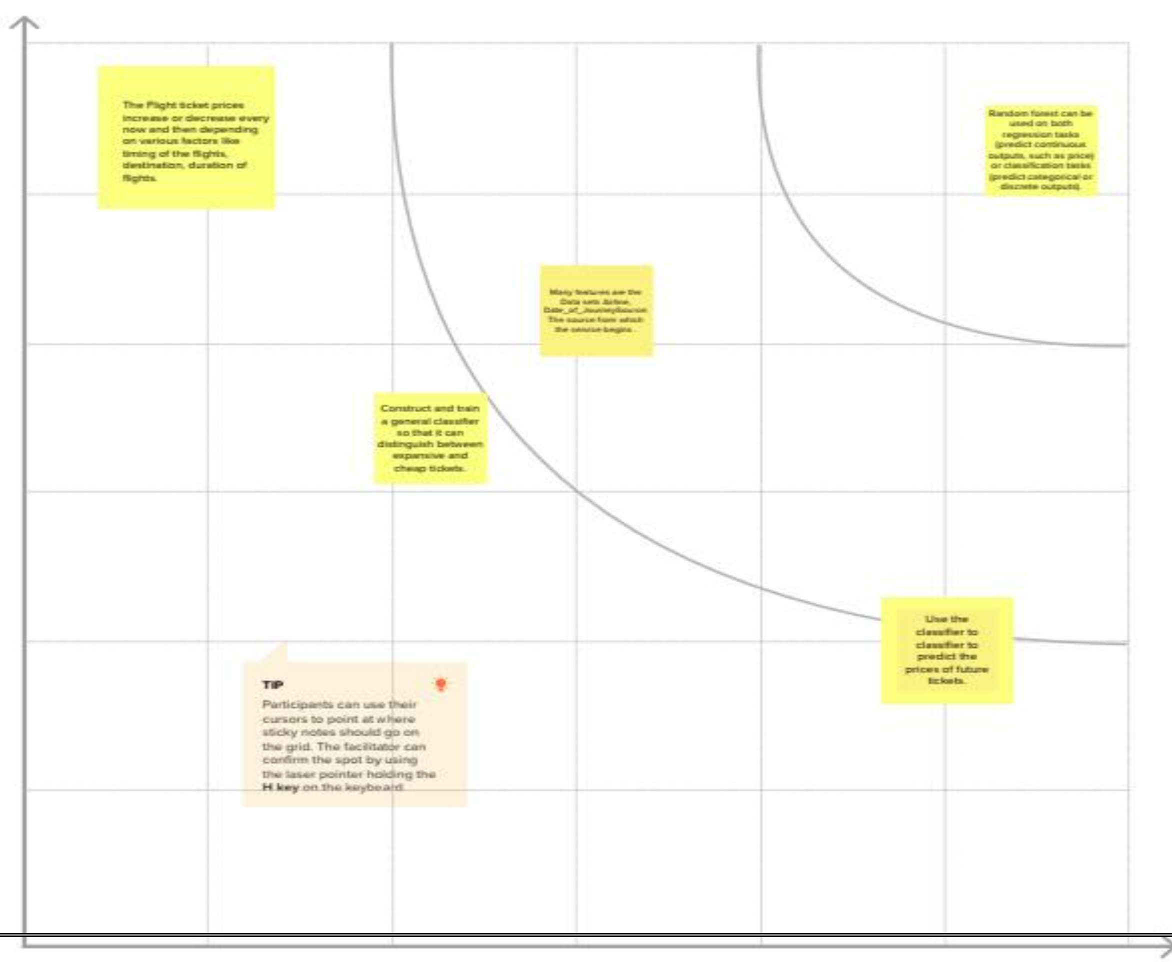
3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes





Results:

Importing required libraries:

```
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.ensemble import RandomForestRegressor, GradientBoostingRegressor, AdaBoostRegressor
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 imblearn
from imblearn.over_sampling import SMOTE
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
from google.colab import drive
drive.mount("/content/gdrive")
```

Read the excel file:

```
import io
import pandas as pd
data=pd.read_excel('Data_Train.xlsx')
data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Handling the category:

```
category=['Airline','Source','Destination','Additional_Info']
for i in category:
  print(i, data[i].unique())
```

```
Airline ['IndiGo' 'Air India' 'Jet Airways' 'SpiceJet' 'Multiple carriers' 'GoAir'
 'Vistara' 'Air Asia' 'Vistara Premium economy' 'Jet Airways Business'
 'Multiple carriers Premium economy' 'Trujet']
Source ['Banglore' 'Kolkata' 'Delhi' 'Chennai' 'Mumbai']
Destination ['New Delhi' 'Banglore' 'Cochin' 'Kolkata' 'Delhi' 'Hyderabad']
Additional_Info ['No info' 'In-flight meal not included' 'No check-in baggage included'
 '1 Short layover' 'No Info' '1 Long layover' 'Change airports'
 'Business class' 'Red-eye flight' '2 Long layover']
```

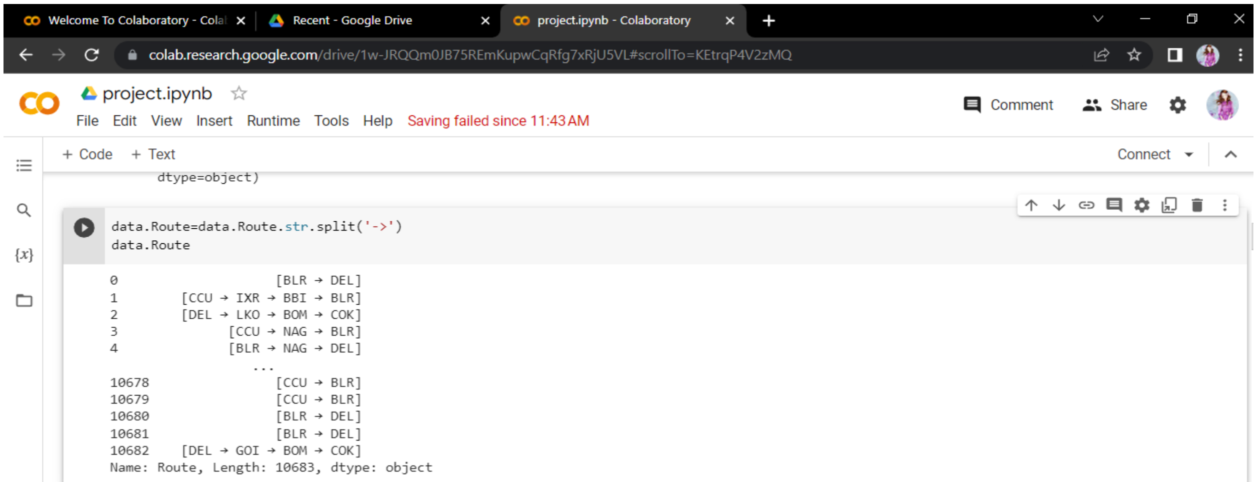
Then split the dataframe:

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

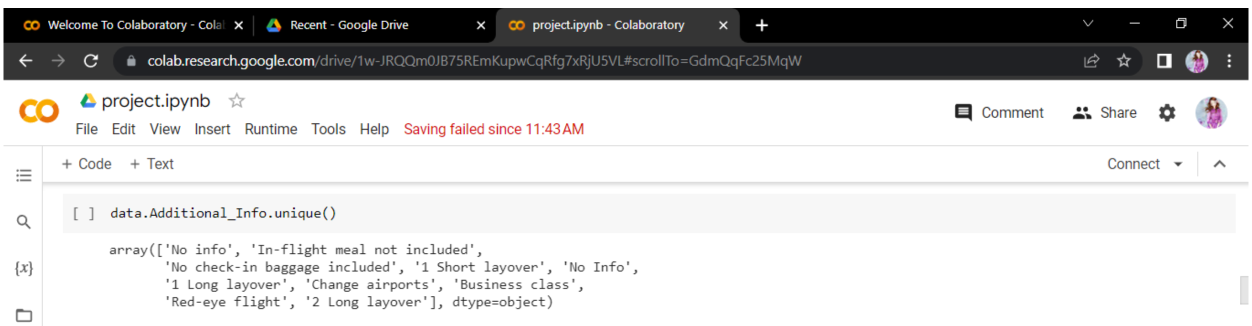
```
0        [24, 03, 2019]
1         [1, 05, 2019]
2         [9, 06, 2019]
3        [12, 05, 2019]
4        [01, 03, 2019]
              ...
10678     [9, 04, 2019]
10679    [27, 04, 2019]
10680    [27, 04, 2019]
10681    [01, 03, 2019]
10682     [9, 05, 2019]
Name: Date_of_Journey, Length: 10683, dtype: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

```
data.Total_Stops.unique()
```

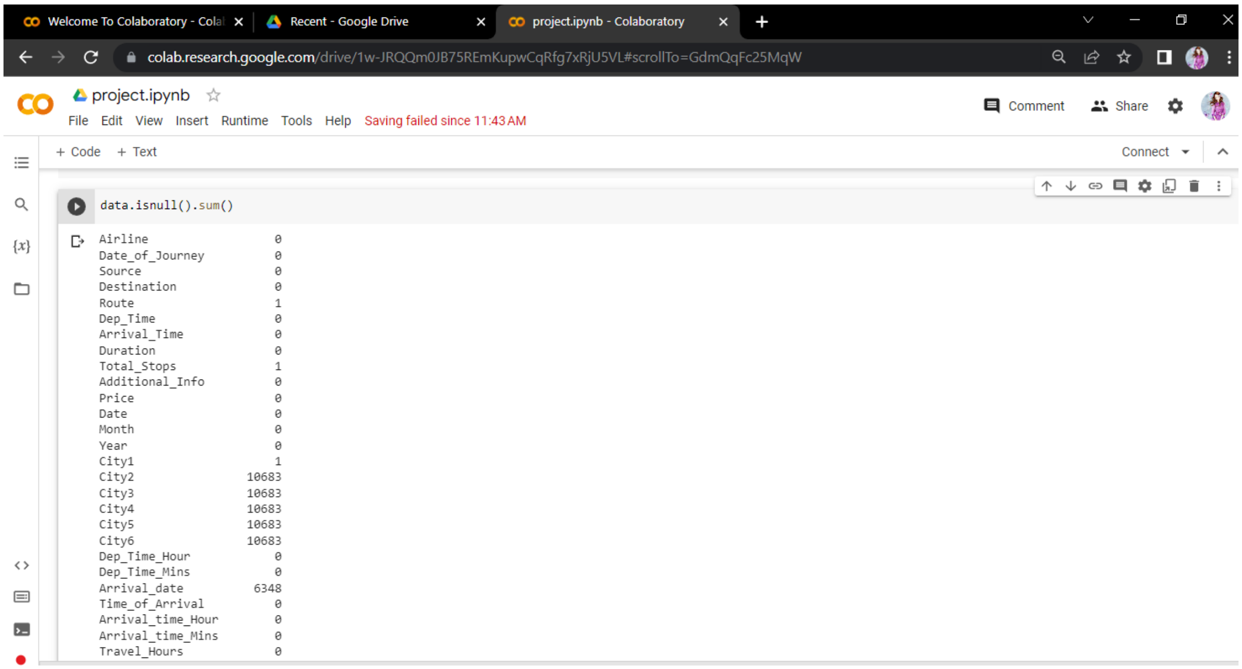
```
array(['non-stop', '2 stops', '1 stop', '3 stops', nan, '4 stops'],
      dtype=object)
```

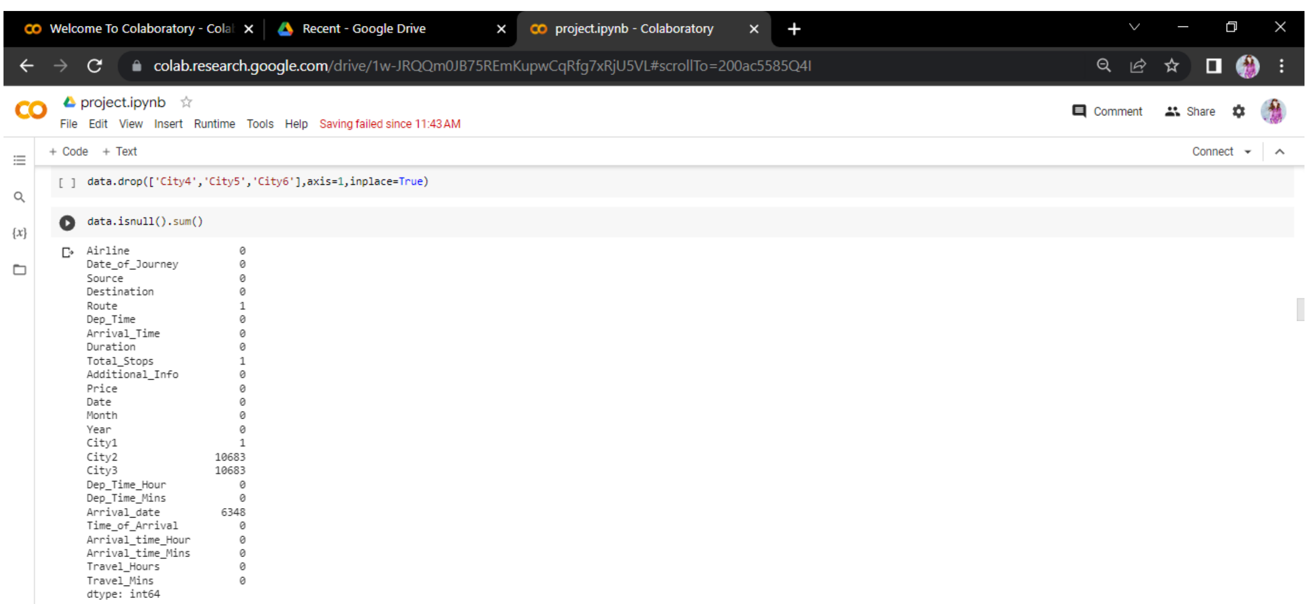


Getting the information using info():

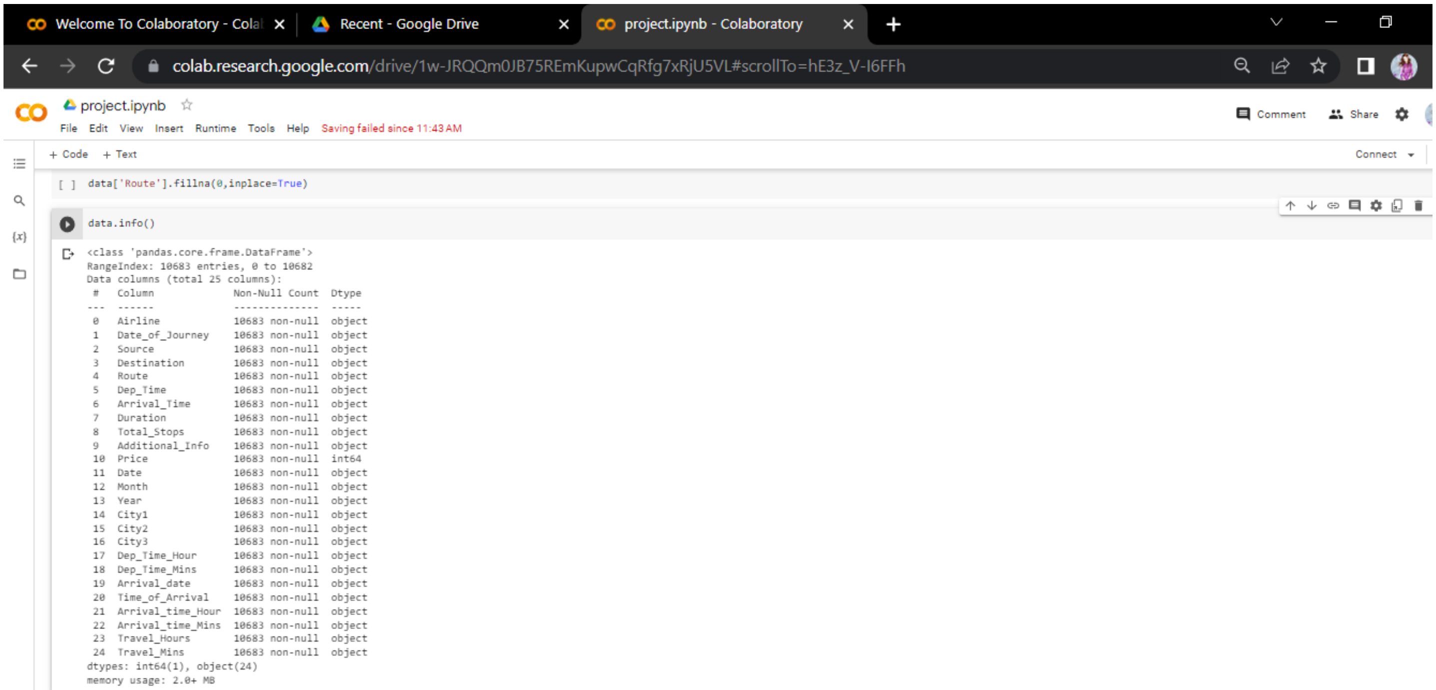


Finding sum of Null Values isnull().sum() function:





Data .info():

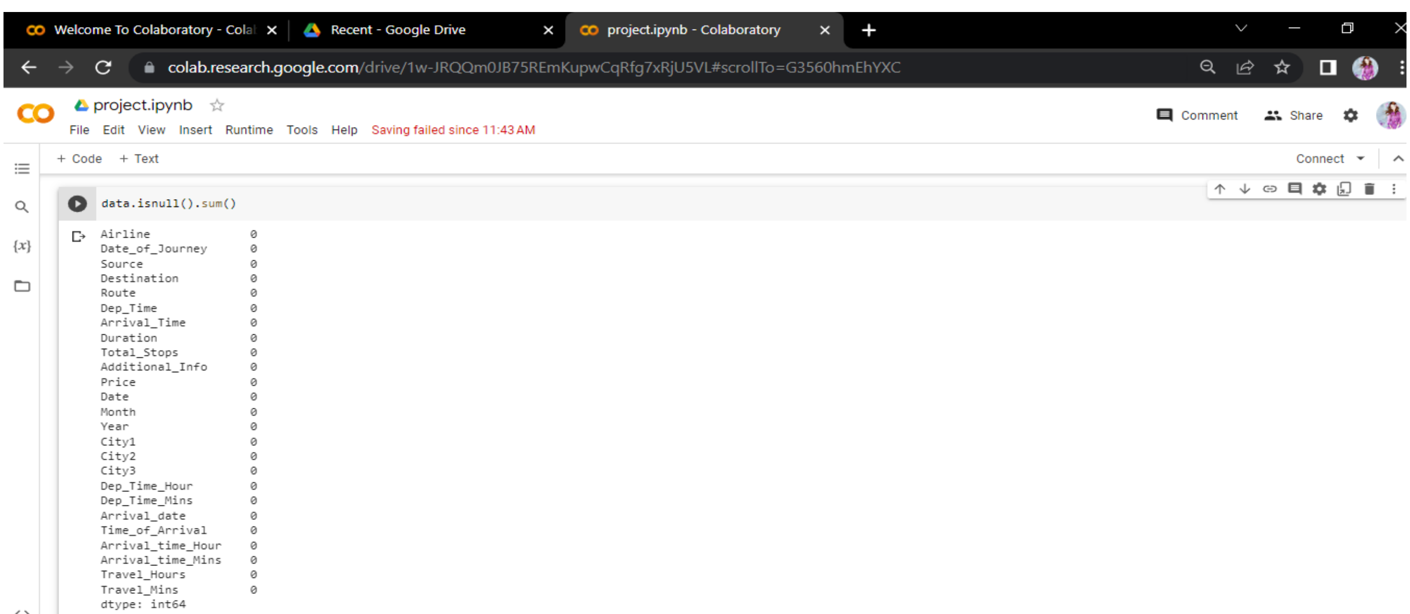


```
data['Route'].fillna(0,inplace=True)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 25 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Airline            10683 non-null  object
 1   Date_of_Journey    10683 non-null  object
 2   Source             10683 non-null  object
 3   Destination        10683 non-null  object
 4   Route              10683 non-null  object
 5   Dep_Time           10683 non-null  object
 6   Arrival_Time       10683 non-null  object
 7   Duration           10683 non-null  object
 8   Total_Stops        10683 non-null  object
 9   Additional_Info    10683 non-null  object
 10  Price              10683 non-null  int64
 11  Date               10683 non-null  object
 12  Month              10683 non-null  object
 13  Year               10683 non-null  object
 14  City1              10683 non-null  object
 15  City2              10683 non-null  object
 16  City3              10683 non-null  object
 17  Dep_Time_Hour      10683 non-null  object
 18  Dep_Time_Mins      10683 non-null  object
 19  Arrival_date       10683 non-null  object
 20  Time_of_Arrival    10683 non-null  object
 21  Arrival_time_Hour  10683 non-null  object
 22  Arrival_time_Mins  10683 non-null  object
 23  Travel_Hours       10683 non-null  object
 24  Travel_Mins        10683 non-null  object
dtypes: int64(1), object(24)
memory usage: 2.0+ MB
```

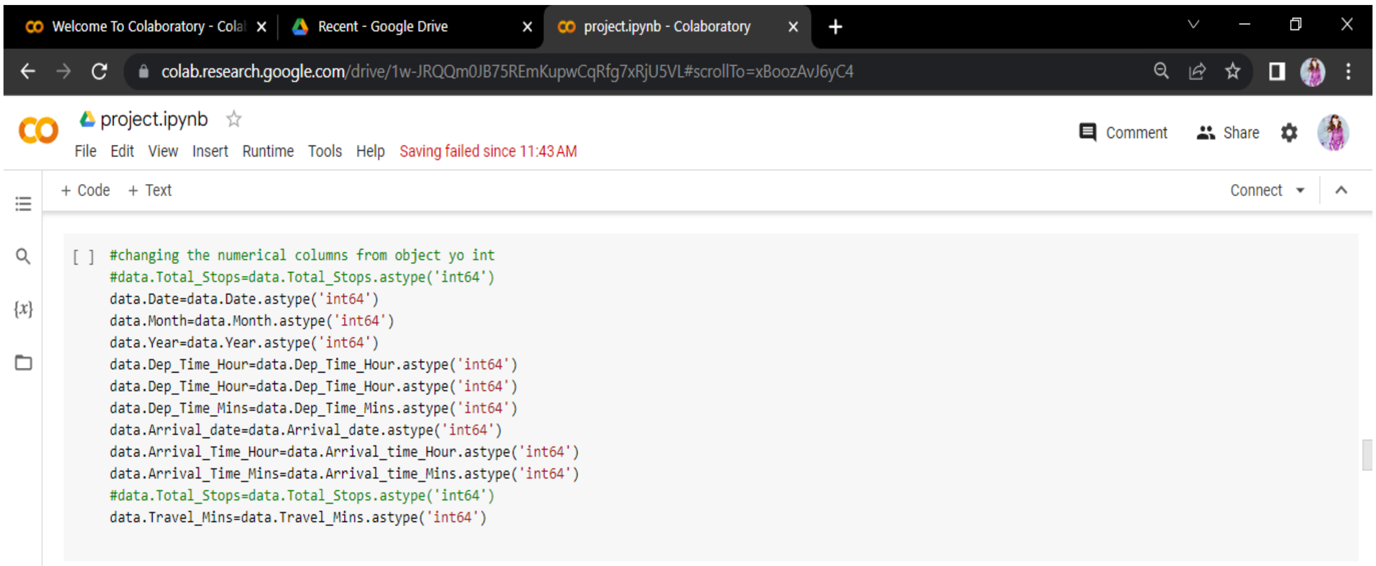
Data.isnull().sum():

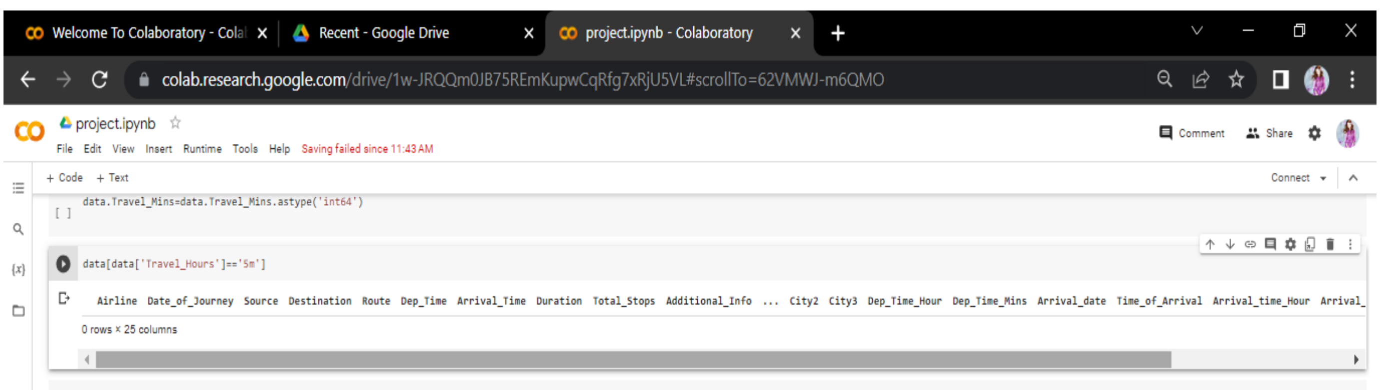


```
data.isnull().sum()
```

```
Airline              0
Date_of_Journey      0
Source               0
Destination          0
Route                0
Dep_Time             0
Arrival_Time         0
Duration             0
Total_Stops          0
Additional_Info      0
Price                0
Date                 0
Month                0
Year                 0
City1                0
City2                0
City3                0
Dep_Time_Hour        0
Dep_Time_Mins        0
Arrival_date         0
Time_of_Arrival      0
Arrival_time_Hour    0
Arrival_time_Mins    0
Travel_Hours         0
Travel_Mins          0
dtype: int64
```

Change the data type of the required columns:





Label Encoding:

```python
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.City1= le.fit_transform(data.City1)
data.City2= le.fit_transform(data.City2)
data.City3= le.fit_transform(data.City3)
data.Additional_Info= le.fit_transform(data.Additional_Info)
```

```python
data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	...	City2	City3	Dep_Time_Hour	Dep_Time_Mins	Arrival_date	Time_of_Arrival	Arrival_time_Hour	Arrival_time_Mins	Travel_Hours	Travel_Mins
0	3	[24, 03, 2019]	0	5	[BLR → DEL]	[22, 20]	[01:10, 22, Mar]	[, 2, h, , 5, 0, m,]	0	7	...	0	0	22	20	22	[01, 10]	01	10		2
1	1	[1, 05, 2019]	3	0	[CCU → IXR → BBI → BLR]	[05, 50]	[13:15]	[, 7, h, , 2, 5, m,]	0	7	...	0	0	5	50	1	[13, 15]	13	15		7
2	4	[9, 06, 2019]	2	1	[DEL → LKO → BOM → COK]	[09, 25]	[04:25, 10, Jun]	[, 1, 9, h,]	0	7	...	0	0	9	25	10	[04, 25]	04	25		1
3	3	[12, 05, 2019]	3	0	[CCU → NAG → BLR]	[18, 05]	[23:30]	[, 5, h, , 2, 5, m,]	0	7	...	0	0	18	5	12	[23, 30]	23	30		5
4	3	[01, 03, 2019]	0	5	[BLR → NAG → DEL]	[16, 50]	[21:35]	[, 4, h, , 4, 5, m,]	0	7	...	0	0	16	50	1	[21, 35]	21	35		4

5 rows × 25 columns

```python
data = data[['Airline','Source', 'Destination','Date','Month','Year','Dep_Time_Mins','Arrival_date','Price']]
data.head()
```

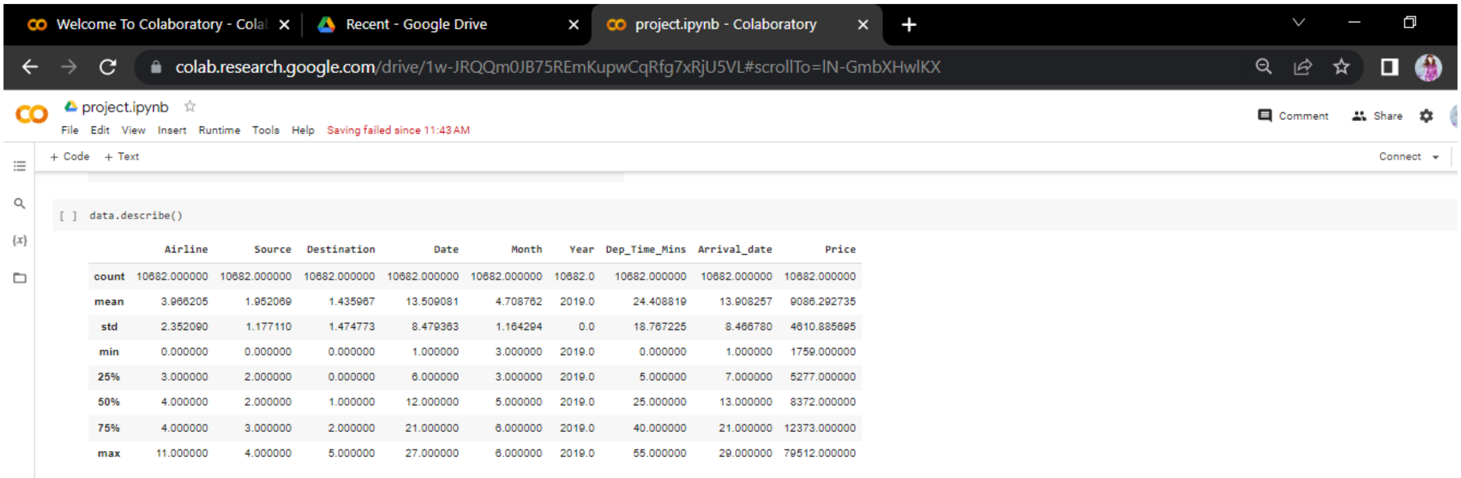
	Airline	Source	Destination	Date	Month	Year	Dep_Time_Mins	Arrival_date	Price
0	3	0	5	24	3	2019	20	22	3897
1	1	3	0	1	5	2019	50	1	7662
2	4	2	1	9	6	2019	25	10	13882
3	3	3	0	12	5	2019	5	12	6218
4	3	0	5	1	3	2019	50	1	13302

```python
data.describe()
```

	Airline	Source	Destination	Date	Month	Year	Dep_Time_Mins	Arrival_date	Price
count	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.0	10682.000000	10682.000000	10682.000000
mean	3.966205	1.952069	1.435967	13.509081	4.708762	2019.0	24.408819	13.908257	9086.292735
std	2.352090	1.177110	1.474773	8.479363	1.164294	0.0	18.767225	8.466780	4610.885695
min	0.000000	0.000000	0.000000	1.000000	3.000000	2019.0	0.000000	1.000000	1759.000000
25%	3.000000	2.000000	0.000000	6.000000	3.000000	2019.0	5.000000	7.000000	5277.000000
50%	4.000000	2.000000	1.000000	12.000000	5.000000	2019.0	25.000000	13.000000	8372.000000
75%	4.000000	3.000000	2.000000	21.000000	6.000000	2019.0	40.000000	21.000000	12373.000000
max	11.000000	4.000000	5.000000	27.000000	6.000000	2019.0	55.000000	29.000000	79512.000000

DATA ANALYSIS

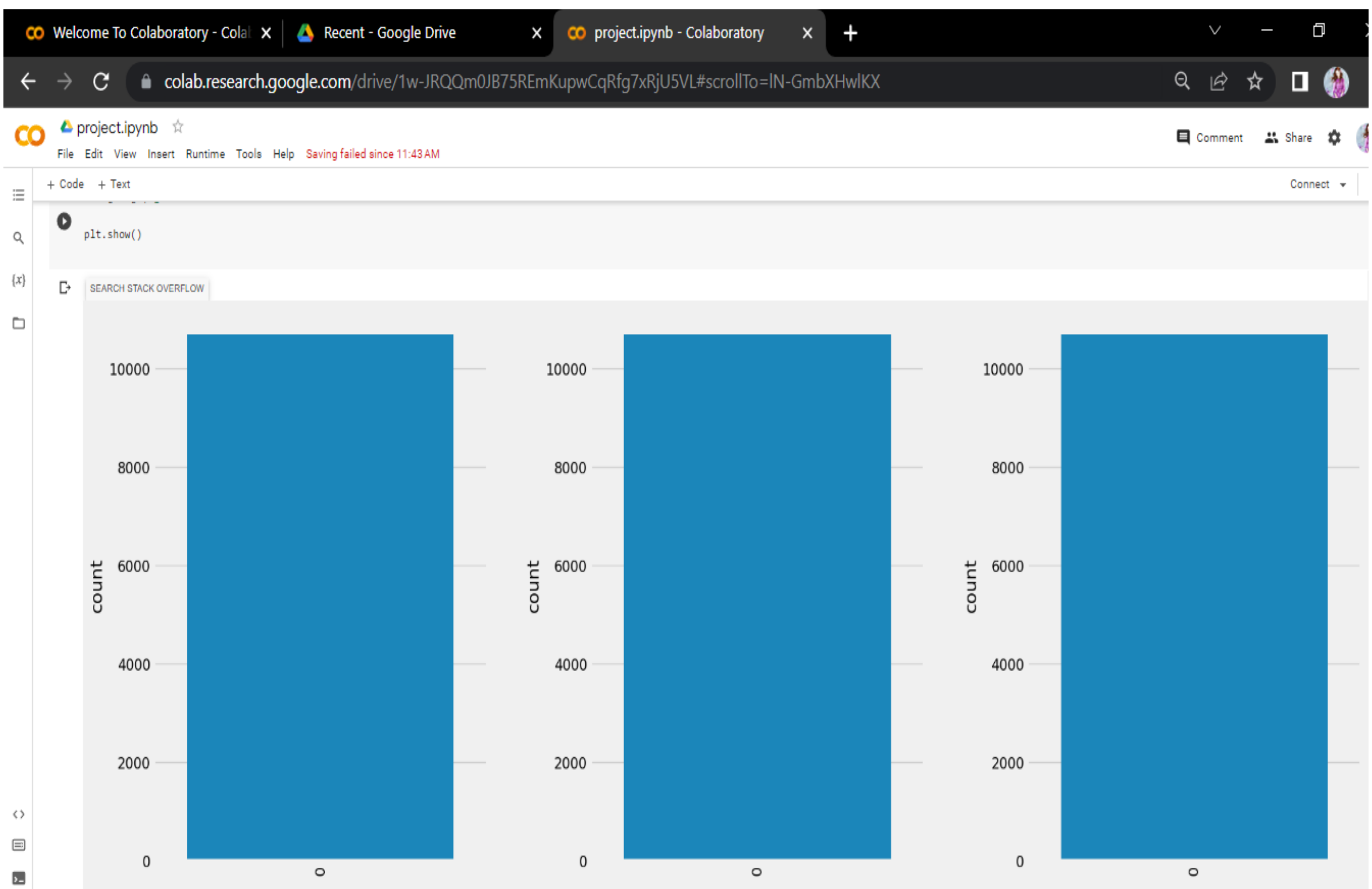
○ Descriptive Analysis:



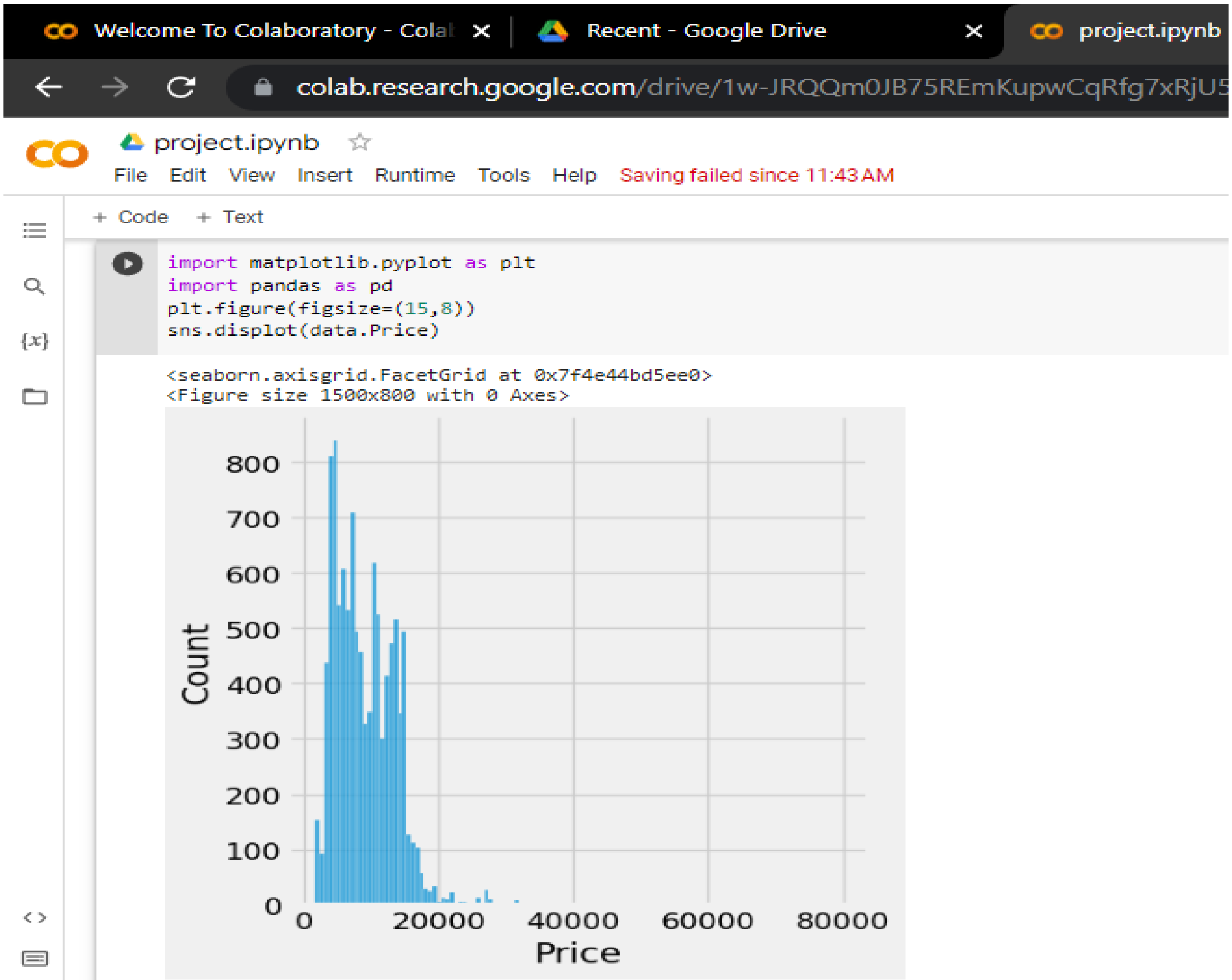
```
data.describe()
```

	Airline	Source	Destination	Date	Month	Year	Dep_Time_Mins	Arrival_date	Price
count	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.0	10682.000000	10682.000000	10682.000000
mean	3.966205	1.952069	1.435967	13.509081	4.708762	2019.0	24.408819	13.908257	9086.292735
std	2.352090	1.177110	1.474773	8.479363	1.164294	0.0	18.767225	8.466780	4610.885695
min	0.000000	0.000000	0.000000	1.000000	3.000000	2019.0	0.000000	1.000000	1759.000000
25%	3.000000	2.000000	0.000000	6.000000	3.000000	2019.0	5.000000	7.000000	5277.000000
50%	4.000000	2.000000	1.000000	12.000000	5.000000	2019.0	25.000000	13.000000	8372.000000
75%	4.000000	3.000000	2.000000	21.000000	6.000000	2019.0	40.000000	21.000000	12373.000000
max	11.000000	4.000000	5.000000	27.000000	6.000000	2019.0	55.000000	29.000000	79512.000000

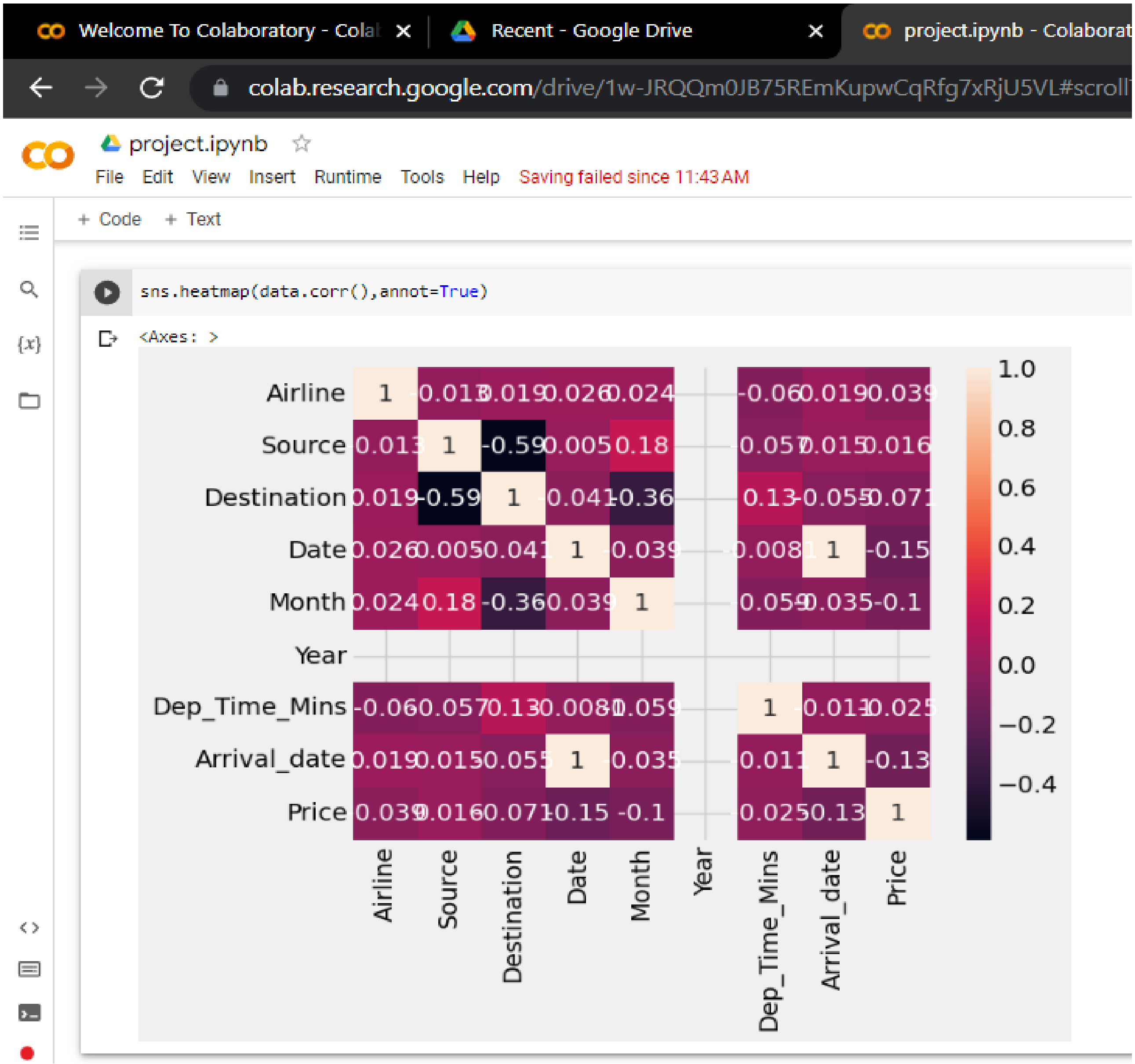
○ Visual Analysis:



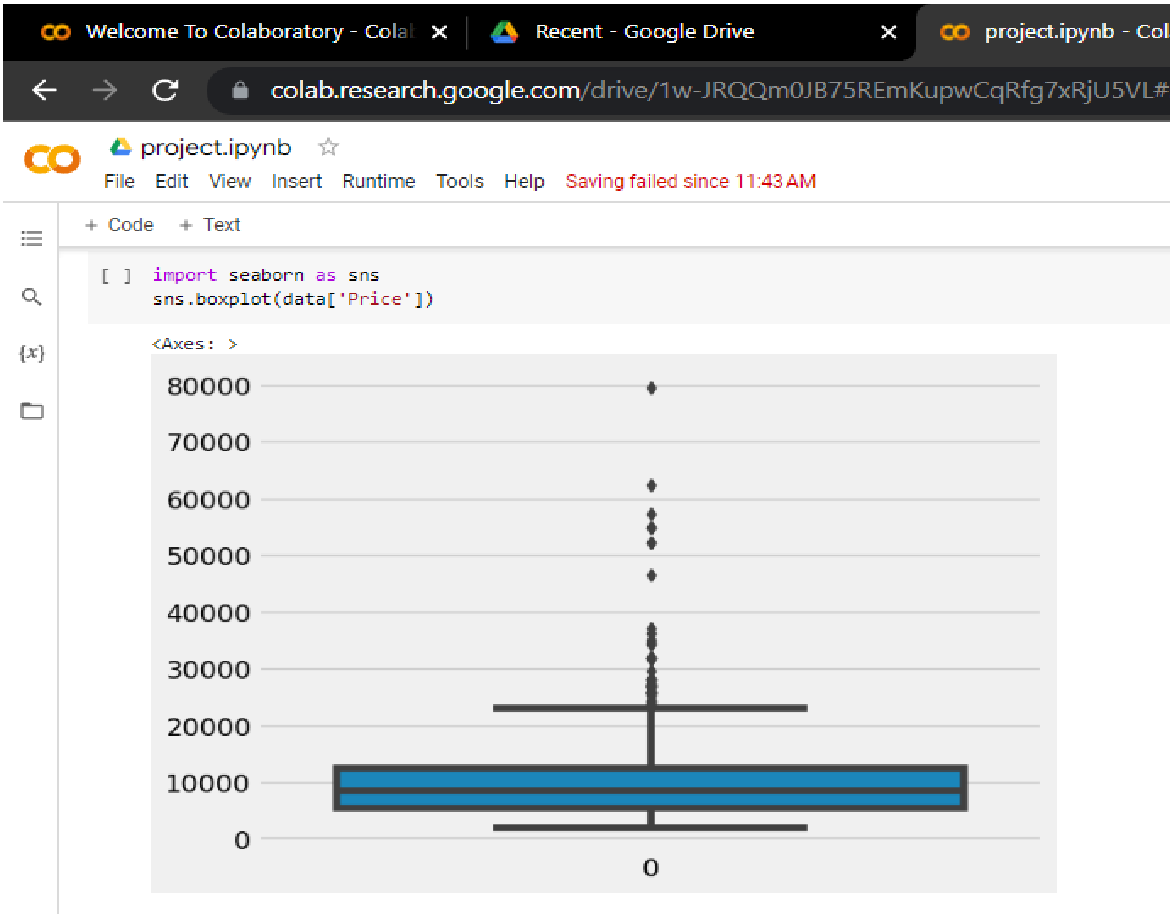
○ Univariate analysis



○ Using Heat Map:



Finding outliers:



Scale the Data:

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

```
x_scaled = ss.fit_transform(x)
x_scaled=pd.DataFrame(x_scaled,columns=x.columns)
x_scaled.head()
```

	Airline	Source	Destination	Date	Month	Year	Dep_Time_Mins	Arrival_date
0	-0.410805	-1.658435	2.416778	1.237288	-1.467707	0.0	-0.234932	0.955750
1	-1.261152	0.890299	-0.973732	-1.475307	0.250153	0.0	1.363674	-1.524648
2	0.014369	0.040721	-0.295630	-0.531796	1.109082	0.0	0.031502	-0.461621
3	-0.410805	0.890299	-0.973732	-0.177979	0.250153	0.0	-1.034235	-0.225392
4	-0.410805	-1.658435	2.416778	-1.475307	-1.467707	0.0	1.363674	-1.524648

Splitting data into train and test:

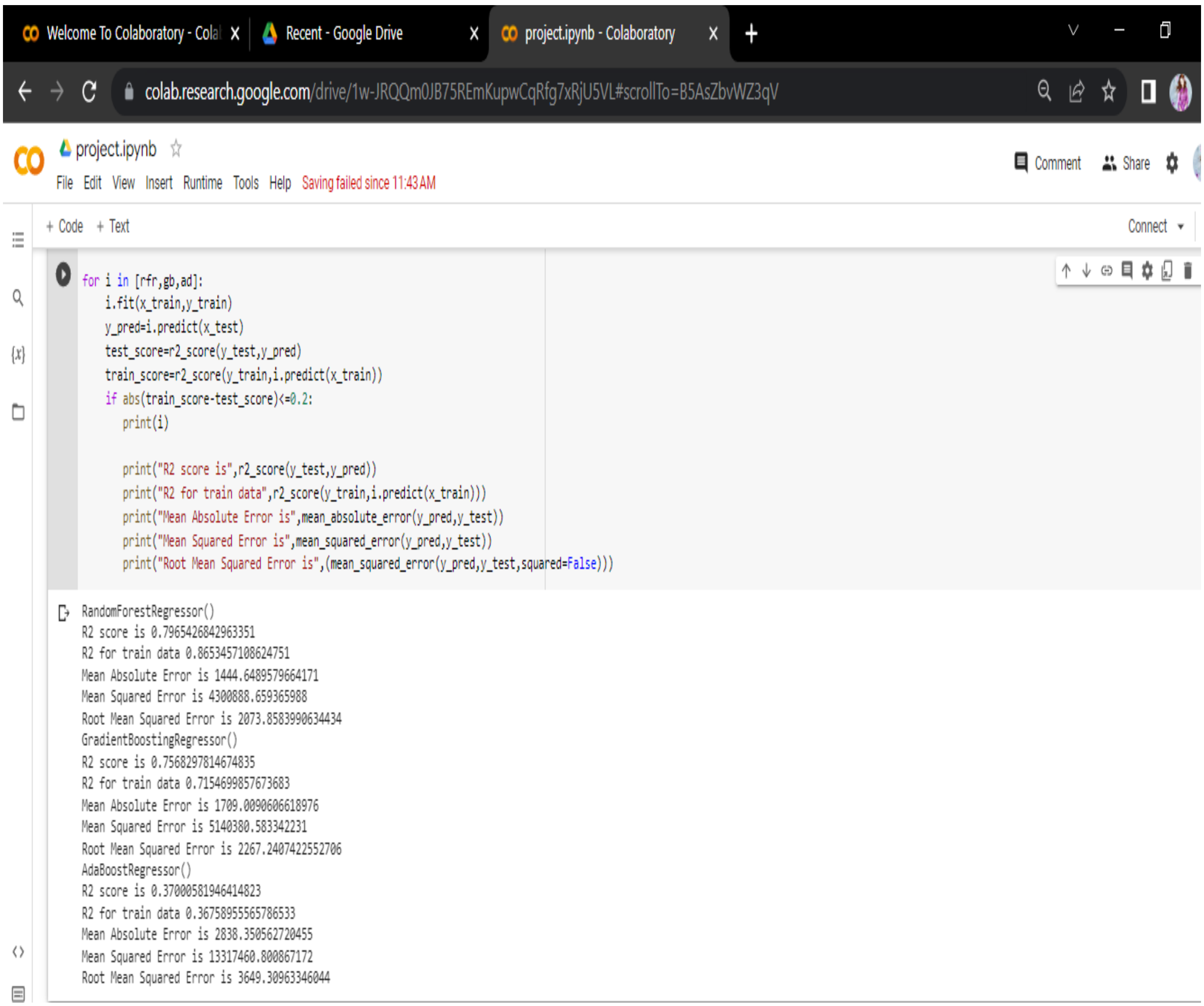
```
import pandas as pd

# Convert numpy array to pandas DataFrame
x_train_df = pd.DataFrame(x_train)

# Use head() method to display the first few rows
x_train.head()
```

	Airline	Source	Destination	Date	Month	Year	Dep_Time_Mins	Arrival_date
10005	6	2	1	27	5	2019	30	27
3684	4	2	1	9	5	2019	30	10
1034	8	2	1	24	4	2019	45	24
3909	6	2	1	21	3	2019	50	22
3088	1	2	1	24	6	2019	15	25

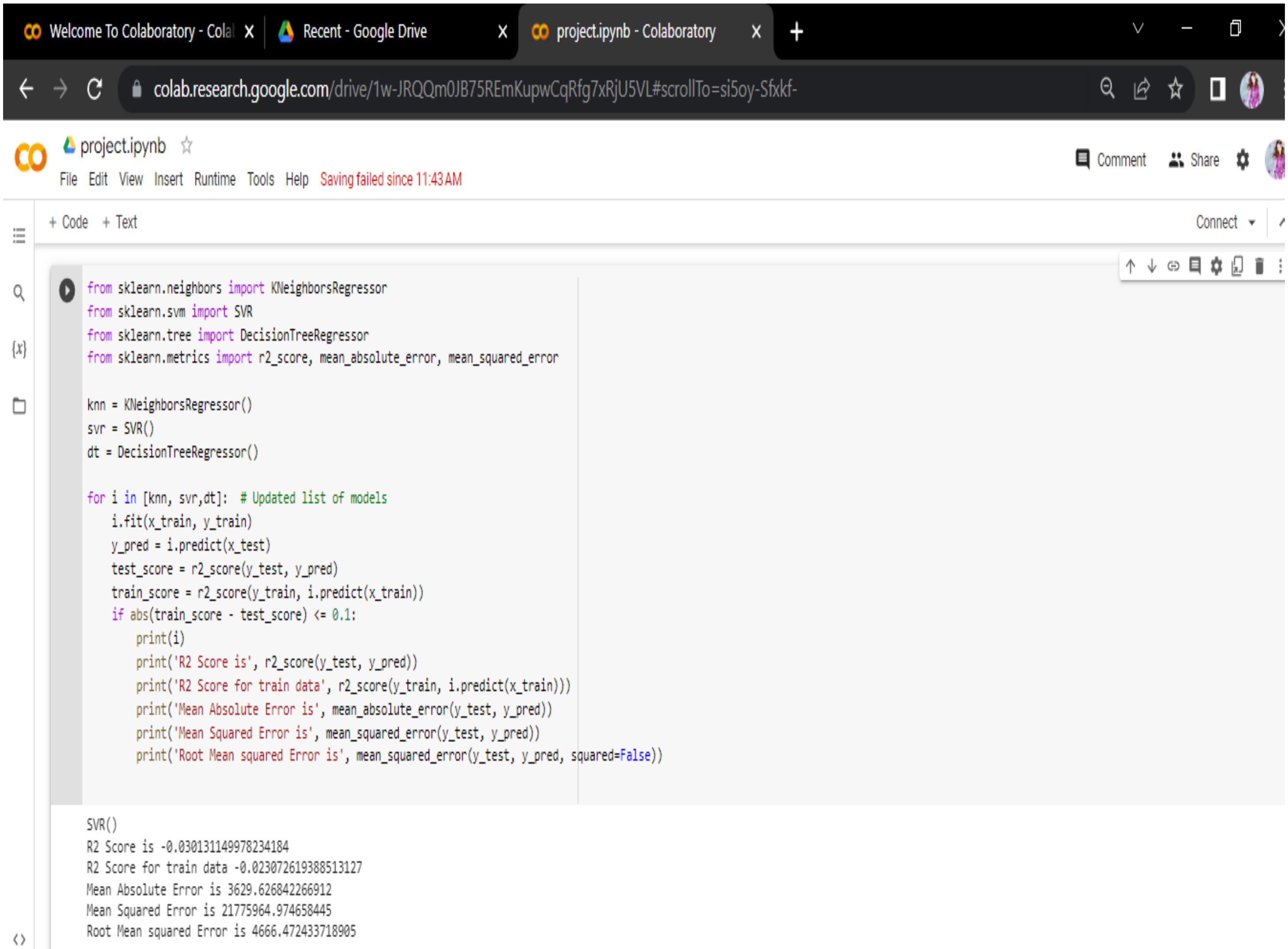
MODEL BUILDING

Applying algorithms

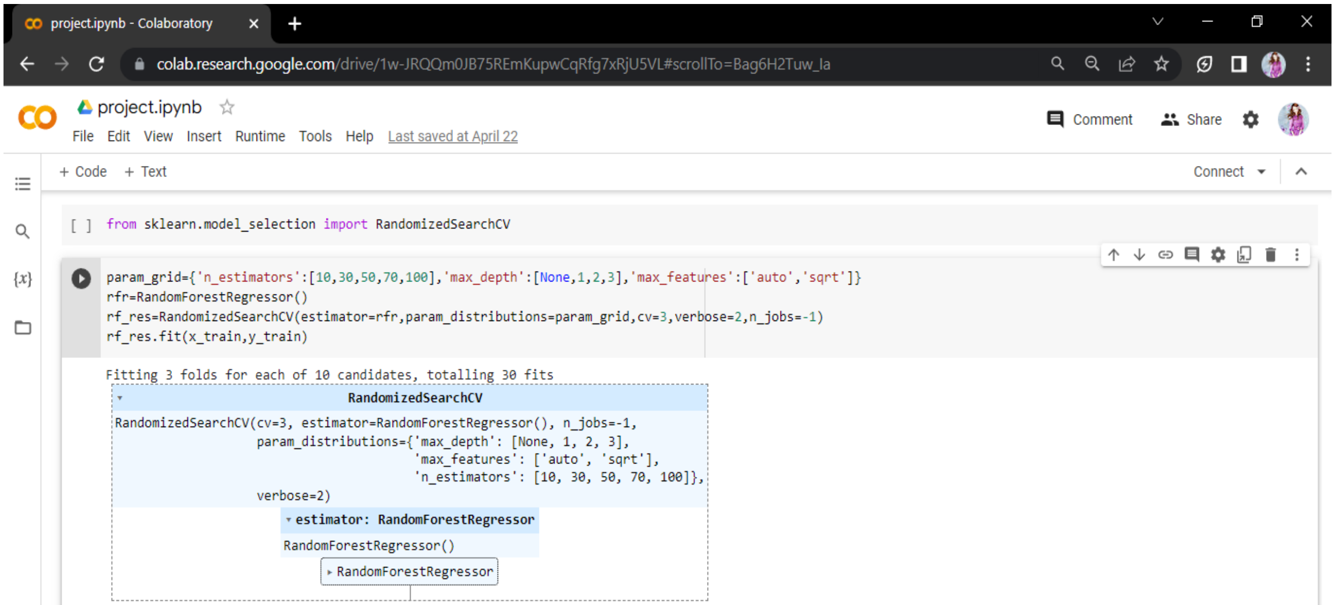
❖ Random Forest Regressor:



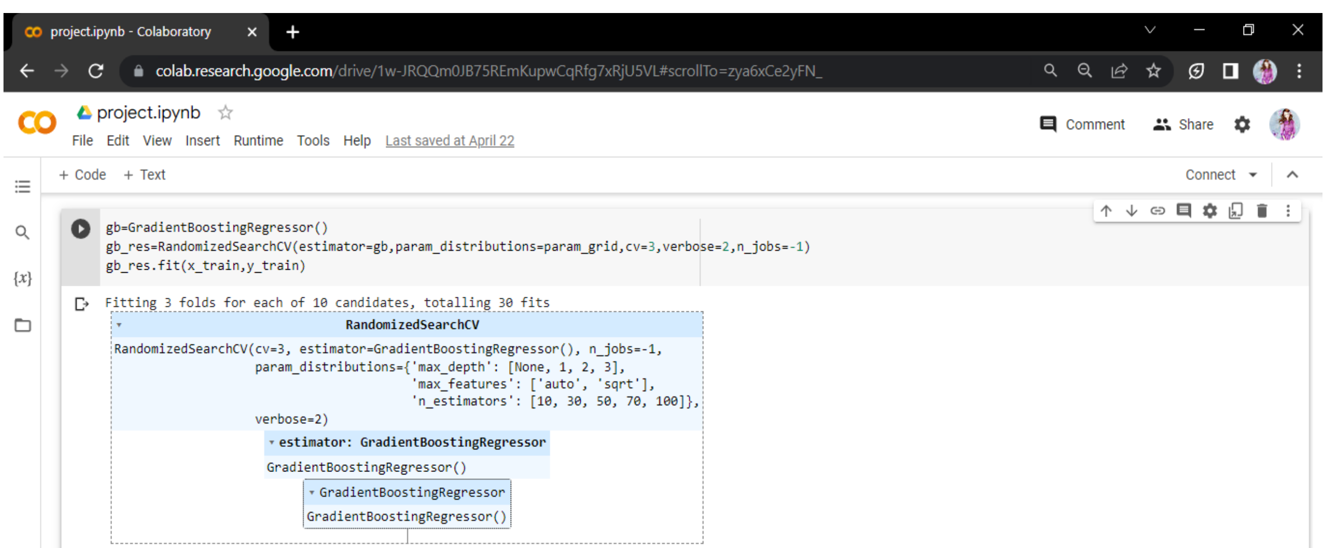
❖ K Neighbors Regressor



Checking Cross Validation for Random Forest Regressor



❖ Gradient Boosting Regressor



Accuracy

```
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 accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
train accuracy 0.8253620364268272
test accuracy 0.7317496985386794
```

```
knn=KNeighborsRegressor(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 accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
train accuracy 0.7356468687366546
test accuracy 0.5348156266902466
```

```
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 accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
train accuracy 0.8233947632452517
test accuracy 0.722207952312962 5
```

```
Price_list = pd.DataFrame({'Price':data['Price']})
Price_list
```

	Price
0	3897
1	7662
2	13882
3	6218
4	13302
...	...
10678	4107
10679	4145
10680	7229
10681	12648
10682	11753

10682 rows × 1 columns

Advantages & Disadvantages

Advantages:

Cost Savings: One of the primary advantages of flight price prediction is that it can help travelers save money. By predicting when prices are likely to be at their lowest, travelers can book their flights at the optimal time and avoid overpaying for their tickets.
predicting flight prices in advance, users can make more informed decisions about when to book their flights, potentially saving money on their travel expenses.

Time savings: Flight price prediction can save travelers a significant amount of time by helping them quickly identify the best times to book their flights. This can be particularly useful for people who need to travel frequently for work or other reasons.

Accuracy: Machine learning algorithms can process large amounts of historical data and identify patterns that humans may not be able to see, resulting in more accurate predictions.

Efficiency: Predicting flight prices using machine learning can be much faster and more efficient than manually analyzing data.

Convenience: Flight price prediction tools are often easy to use and can be accessed from a variety of devices, including smartphones and tablets. This makes it convenient for travelers to check flight prices and make bookings on the go.

Increased accessibility: Flight price prediction makes air travel more accessible to a wider range of people by helping them find affordable flights. By predicting price changes and providing cost-saving opportunities, more people can afford to fly, opening up new opportunities for travel and business.

Enhanced customer experience: By providing travelers with valuable information about pricing trends and helping them find the best deals, flight price prediction can improve the overall customer experience. This can lead to increased customer satisfaction and loyalty, as well as positive word-of-mouth recommendations.

Improved revenue management: Flight price prediction can benefit airlines by enabling them to optimize their revenue management strategies. By accurately forecasting demand and pricing trends, airlines can adjust their prices in real-time to maximize revenue and profitability.

Disadvantages

Inaccuracy: Flight booking price prediction is not Always accurate, and travelers may end up paying more than predicted if prices increase unexpectedly.

Data Availability: The accuracy of the predictions depends on the availability and quality of historical flight data. In some cases, data may not be readily available or may be incomplete, which can impact the accuracy of the predictions.

Model Complexity: Building a machine learning model for flight price prediction can be complex, requiring advanced programming skills and a good understanding of machine learning algorithms.

Limited Scope: The model may not be able to account for all factors that can impact flight prices, such as sudden changes in fuel prices or political events that can affect air travel.

Unforeseen Factors: The model may not be able to predict unforeseen events, such as natural disasters, which can impact flight prices and availability.

Applications

Historical Data: The application can use historical flight data to predict future prices. This can include factors such as seasonality, airline demand, and holiday travel trends.

Price Alerts: Users can set up price alerts for specific flights, which notify them when the price drops below a certain threshold.

Price Trends: Users can view price trends over time for a particular route, and the application can provide recommendations on when to book to get the best price.

Fare Calendar: The application can display a fare calendar, which shows the lowest prices for a particular route over a range of dates.

Travel Budgeting: The application can help users plan their travel budget by providing an estimate of the total cost of a trip, including airfare, accommodations, and other expenses.

Travel Planning: The application can suggest alternative routes or travel dates to help users find the best deal.

Travel booking websites: Online travel agencies and booking websites can use machine learning algorithms to predict flight prices, and offer personalized recommendations to customers.

Airlines: Airlines can use machine learning algorithms to predict demand for flights, adjust prices accordingly, and optimize revenue management.

Travel agencies: Travel agencies can use machine learning algorithms to predict the prices of flights, and offer customized travel packages to customers.

Corporate travel management: Companies can use machine learning algorithms to predict flight prices and optimize travel expenses.

Price comparison websites: Price comparison websites can use machine learning algorithms to provide real-time flight price comparisons across multiple airlines.

Travel apps: Travel apps can use machine learning algorithms to provide users with real-time flight prices and alerts when prices change.

conclusion

Flight price prediction using machine learning is an innovative solution that can benefit the travel industry in numerous ways.

With the help of machine learning algorithms, airlines, travel agencies, booking websites, and consumers can gain more accurate insights into flight prices, enabling them to make better decisions about travel plans and expenses.

By leveraging historical data, machine learning models can identify patterns and predict future prices, optimizing revenue management and enhancing the customer experience.

As machine learning technology continues to evolve, we can expect even more sophisticated flight price prediction models to emerge, providing even greater value to the travel industry.

Overall, flight price prediction using machine learning has the potential to revolutionize the way we plan and book travel, making it more efficient, affordable, and enjoyable for everyone involved.

Future Scope

The scope for flight price prediction using machine learning algorithms is quite promising, and there are several areas where it could be expanded in the future:

Improved accuracy: As more data becomes available and machine learning algorithms become more sophisticated, the accuracy of flight price prediction models is likely to improve.

Personalized pricing: Machine learning algorithms could be used to create personalized pricing models for individual travelers based on their past travel patterns, preferences, and behavior.

Real-time prediction: Real-time prediction models could be developed that use data such as weather conditions, flight delays, and passenger volume to adjust prices in real-time and provide travelers with the most up-to-date information on flight prices.

Integration with other travel-related services: Flight price prediction models could be integrated with other travel-related services such as hotel bookings and car rentals to provide travelers with a more comprehensive and seamless travel experience.

Expansion to other modes of transport: The same machine learning techniques used for flight price prediction could be applied to other modes of transport such as trains and buses, providing travelers with more options and flexibility when planning their journeys.

APPENDIX

CODING

Source code in colab (.ipynb):

```
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,Gra
dientBoostingClassifier
from sklearn.ensemble import RandomForestRegressor, Gra
dientBoostingRegressor, AdaBoostRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report,confu
sion_matrix
import warnings
import pickle
from scipy import stats
import imblearn
from imblearn.over_sampling import SMOTE
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
from google.colab import drive
drive.mount("/content/gdrive")
```

```
from google.colab import files
uploaded = files.upload()
```

```
import io
import pandas as pd
data=pd.read_excel('Data_Train.xlsx')
data.head()


category=['Airline','Source','Destination','Additional]
for i in category:
  print(i, data[i].unique())



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


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['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.Travel_Mins.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.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.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',inplac
e=True)

data.isnull().sum()

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

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

data['City1'].fillna('None',inplace=True)
data['City2'].fillna('None',inplace=True)
data['City3'].fillna('None',inplace=True)

data['Arrival_date'].fillna(data['Date'],inplace=True)

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

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

data.info()

data.isnull().sum()

#changing the numerical columns from object yo int
#data.Total_Stops=data.Total_Stops.astype('int64')
data.Date=data.Date.astype('int64')
data.Month=data.Month.astype('int64')
data.Year=data.Year.astype('int64')
data.Dep_Time_Hour=data.Dep_Time_Hour.astype('int64')
data.Dep_Time_Hour=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_Hour=data.Arrival_time_Hour.astype('i
nt64')
#data.Total_Stops=data.Total_Stops.astype('int64')
data.Travel_Mins=data.Travel_Mins.astype('int64')


data[data['Travel_Hours']=='5m']

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


categorical=['Airline','Source','Destination','Addition
al_Info']
```

```python
numerical=['Total_Stops','Date','Month','Year','Dep_Tim
e_Hour','Dep_Time_Mins','Arrival_date','Arrival_Time_Ho
ur','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.City1= le.fit_transform(data.City1)
data.City2= le.fit_transform(data.City2)
data.City3= le.fit_transform(data.City3)
data.Additional_Info= le.fit_transform(data.Additional_
Info)


data.head()


data = data[['Airline','Source', 'Destination','Date','
Month','Year','Dep_Time_Mins','Arrival_date','Price']]
data.head()


data.describe()

import seaborn as sns
import matplotlib.pyplot as plt
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()
```

```
import matplotlib.pyplot as plt
import pandas as pd
plt.figure(figsize=(15,8))
sns.displot(data.Price)

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


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)
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)

import pandas as pd

# Convert numpy array to pandas DataFrame
x_train_df = pd.DataFrame(x_train)

# Use head() method to display the first few rows
x_train.head()


from sklearn.ensemble import RandomForestRegressor, Gra
dientBoostingRegressor, AdaBoostRegressor
rfr=RandomForestRegressor()
gb=GradientBoostingRegressor()
ad=AdaBoostRegressor()
```

```
from sklearn.metrics import r2_score,mean_absolute_erro
r,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.pre
dict(x_train)))
       print("Mean Absolute Error is",mean_absolute_err
or(y_pred,y_test))
       print("Mean Squared Error is",mean_squared_error
(y_pred,y_test))
       print("Root Mean Squared 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_err
or, mean_squared_error

knn = KNeighborsRegressor()
svr = SVR()
dt = DecisionTreeRegressor()

for i in [knn, svr,dt]:  # Updated list of models
    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) <= 0.1:
        print(i)
        print('R2 Score is', r2_score(y_test, y_pred))
```

```
        print('R2 Score for train data', r2_score(y_tra
in, i.predict(x_train)))
        print('Mean Absolute Error is', mean_absolute_e
rror(y_test, y_pred))
        print('Mean Squared Error is', mean_squared_err
or(y_test, y_pred))
        print('Root Mean squared Error is', mean_square
d_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_train,y_train,cv=i)
    print(rfr,cv.mean())

from sklearn.model_selection import RandomizedSearchCV


param_grid={'n_estimators':[10,30,50,70,100],'max_depth
':[None,1,2,3],'max_features':['auto','sqrt']}
rfr=RandomForestRegressor()
rf_res=RandomizedSearchCV(estimator=rfr,param_distribut
ions=param_grid,cv=3,verbose=2,n_jobs=-1)
rf_res.fit(x_train,y_train)



gb=GradientBoostingRegressor()
gb_res=RandomizedSearchCV(estimator=gb,param_distributi
ons=param_grid,cv=3,verbose=2,n_jobs=-1)
gb_res.fit(x_train,y_train)


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 accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
knn=KNeighborsRegressor(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 accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))



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 accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))


Price_list = pd.DataFrame({'Price':data['Price']})
Price_list

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

Creating Templates

1.Home.html

```
<html>

<head>
<title>Flight Price Prediction</title>

<style type="text/css">

@font-face {
font-family: myFirstFont;
src: url(font/Roboto-Regular.ttf);
/*font-weight: bold;*/
}

body
{
font-family: myFirstFont;
}
.sub_btn
{
background: green;
padding: 10px;
border-radius: 4px;
border: none;
margin-top: 30px;
width: 100%;
color: white;
font-size: 16px;
}
.main_section
```

```
{
width: 100%;
margin: auto;
text-align: center;
}

body
{
background-image: url("img.jpg");
/*height: 100%;*/

/*background: linear-gradient(rgb(193 196 225 / 80%), rgb(237 158 37 /
80%)), url(img.jpg);*/
background-position: center;
background-repeat: no-repeat;
background-size: cover;
}
#price_result
{
width: 60%;
margin: auto;
letter-spacing: 0.8px;
line-height: 35px;
font-size: 17px;
}
.navbar
{
width: 100%;
height: 60px;
}
.navbar_ul
{
float: right;
```

```
}
.navbar_li
{
float: left;
background-color: royalblue;
padding: 10px;
margin-right: 10px;
list-style: none;
}
.nav-icon
{
color: white;
padding: 10px;
text-decoration: none;
}
</style>

</head>
<body>
<div class="main_section">
<div class="navbar">
<ul class="navbar_ul">
<li class="navbar_li"><a href="./home.html" class="nav-icon">Home</a></li>
<li class="navbar_li"><a href="./prediction.html" class="nav-icon">Predict</a></li>
</ul>
</div>
<h1 >Flight Price Prediction</h1>

<div class="form_section">
<p id="price_result">
```

The objective of this article is to predict flight prices given the various parameters. This will be regression problem sice the target or dependent variable is the price (continous numerice value). Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can help customers to predict future flight prices and plan their journey accordingly.

</p>

</div>

</div>

</body>

</html>

2.**Predict.html**

```
<html>

<head>
<title>Flight Price Prediction</title>

<style type="text/css">

@font-face {
font-family: myFirstFont;
src: url(font/Roboto-Regular.ttf);
/*font-weight: bold;*/
}

body
{
font-family: myFirstFont;
}
.sub_btn
{
background: green;
padding: 10px;
border-radius: 4px;
border: none;
margin-top: 30px;
width: 100%;
color: white;
font-size: 16px;
}
.main_section
{
width: 100%;
```

```
margin: auto;
text-align: center;
}
.form_section
{
width: 50%;
margin: auto;
}
.ticket_table
{
width: 100%;
}
.ticket_table tr
{
line-height: 40px;
font-size: 18px;
}

.ticket_table td input
{
padding: 7px;
width: 100%;
}
.ticket_table td select
{
width: 100%;
padding: 7px;
}
body
{
background-image: url("img.jpg");
/*height: 100%;*/
```

```
/*background: linear-gradient(rgb(193 196 225 / 80%), rgb(237 158 37 /
80%)), url(img.jpg);*/
background-position: center;
background-repeat: no-repeat;
background-size: cover;
}
.navbar
{
width: 100%;
height: 60px;
}
.navbar_ul
{
float: right;
}
.navbar_li
{
float: left;
background-color: royalblue;
padding: 10px;
margin-right: 10px;
list-style: none;
}
.nav-icon
{
color: white;
padding: 10px;
text-decoration: none;
}
</style>

</head>
```

```
<body>
<div class="main_section">
<div class="navbar">
<ul class="navbar_ul">
<li class="navbar_li"><a href="./home.html" class="nav-icon">Home</a></li>
<li class="navbar_li"><a href="./prediction.html" class="nav-icon">Predict</a></li>
</ul>
</div>
<h1 >Flight Price Prediction</h1>

<div class="form_section">
<form action="./flight_price_result.html">

<table class="ticket_table">
<tr>
<td>Airline</td>
<td>
<select name="airline">
<option value="">Select</option>
<option value="airindia">Air India</option>
<option value="airasia">Air Asia</option>
</select>
</td>
</tr>
<tr>
<td>Source</td>
<td>
<select name="airline">
<option value="">Select</option>
<option value="Banglore">Banglore</option>
<option value="Chennai">Chennai</option>
```

```
</select>
</td>
</tr>
<tr>
<td>Destination</td>
<td>
<select name="airline">
<option value="">Select</option>
<option value="Banglore">Banglore</option>
<option value="Chennai">Chennai</option>
</select>
</td>
</tr>
<tr>
<td>Dep Date</td>
<td>
<input type="text" name="dep_date">
</td>
</tr>
<tr>
<td>Dep Month</td>
<td>
<input type="text" name="dep_month">
</td>
</tr>
<tr>
<td>Dep Year</td>
<td>
<input type="text" name="dep_year">
</td>
</tr>
<tr>
```

```
<td>Dep Time in Hour</td>
<td>
<input type="text" name="dep_time_hour">
</td>
</tr>
<tr>
<td>Dep Time in mins</td>
<td>
<input type="text" name="dep_time_mins">
</td>
</tr>
<tr>
<td>Arrival Time</td>
<td>
<input type="text" name="arr_time">
</td>
</tr>

<tr>
<td>Arrival hour</td>
<td>
<input type="text" name="arr_hour">
</td>
</tr>
<tr>
<td>Arrival time in mins</td>
<td>
<input type="text" name="arr_mins">
</td>
</tr>
</table>
<button type="submit" class="sub_btn"> Submit </button>
```

```
</form>
</div>
</div>

</body>

</html>
```

3.Submit.html

```
<html>

<head>
<title>Flight Price Prediction</title>

<style type="text/css">

@font-face {
font-family: myFirstFont;
src: url(font/Roboto-Regular.ttf);
/*font-weight: bold;*/
}

body
{
font-family: myFirstFont;
}


.main_section
{
width: 100%;
margin: auto;
text-align: center;
}
.form_section
{
width: 50%;
margin: auto;
```

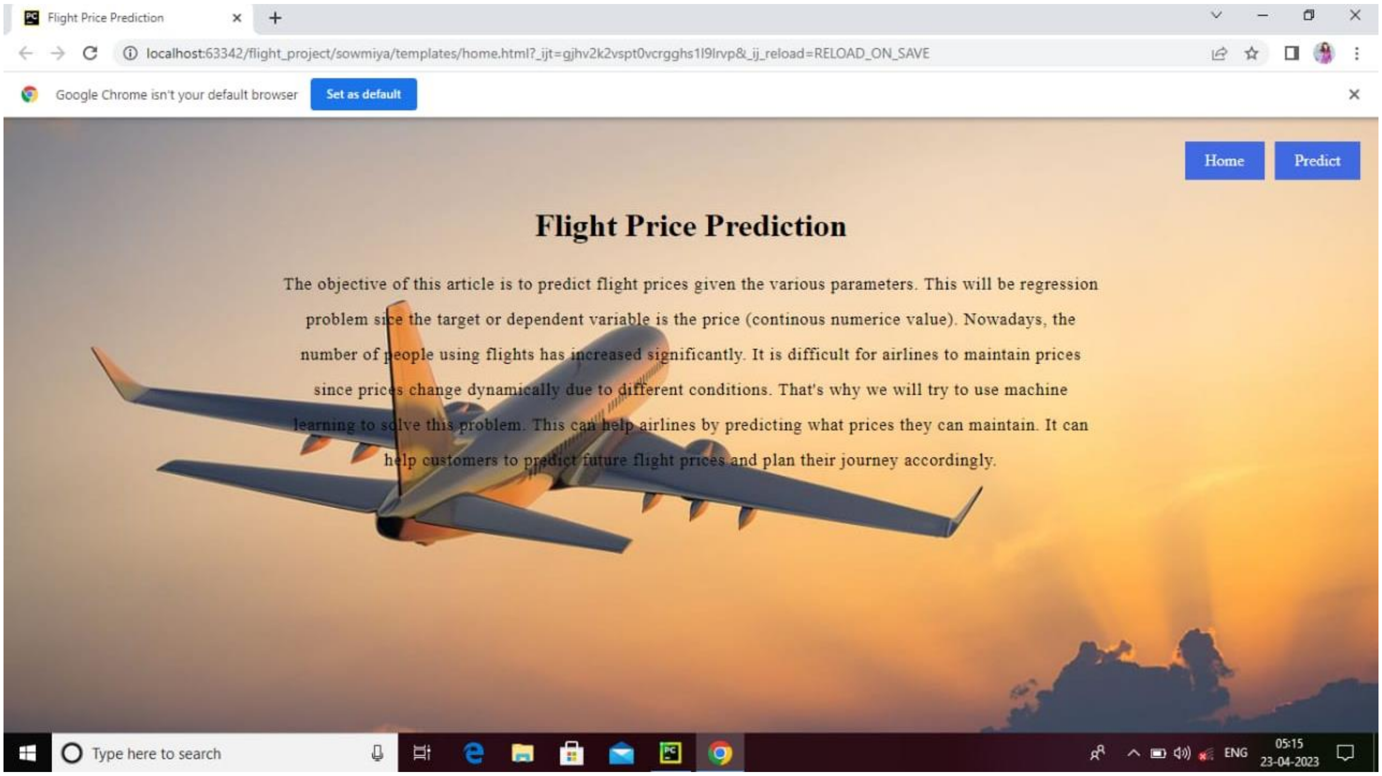
```
}

body
{
background-image: url("img.jpg");
/*height: 100%;*/

/*background: linear-gradient(rgb(193 196 225 / 80%), rgb(237 158 37 /
80%)), url(img.jpg);*/
background-position: center;
background-repeat: no-repeat;
background-size: cover;
}
.navbar
{
width: 100%;
height: 60px;
}
.navbar_ul
{
float: right;
}
.navbar_li
{
float: left;
background-color: royalblue;
padding: 10px;
margin-right: 10px;
list-style: none;
}
.nav-icon
{
color: white;
```

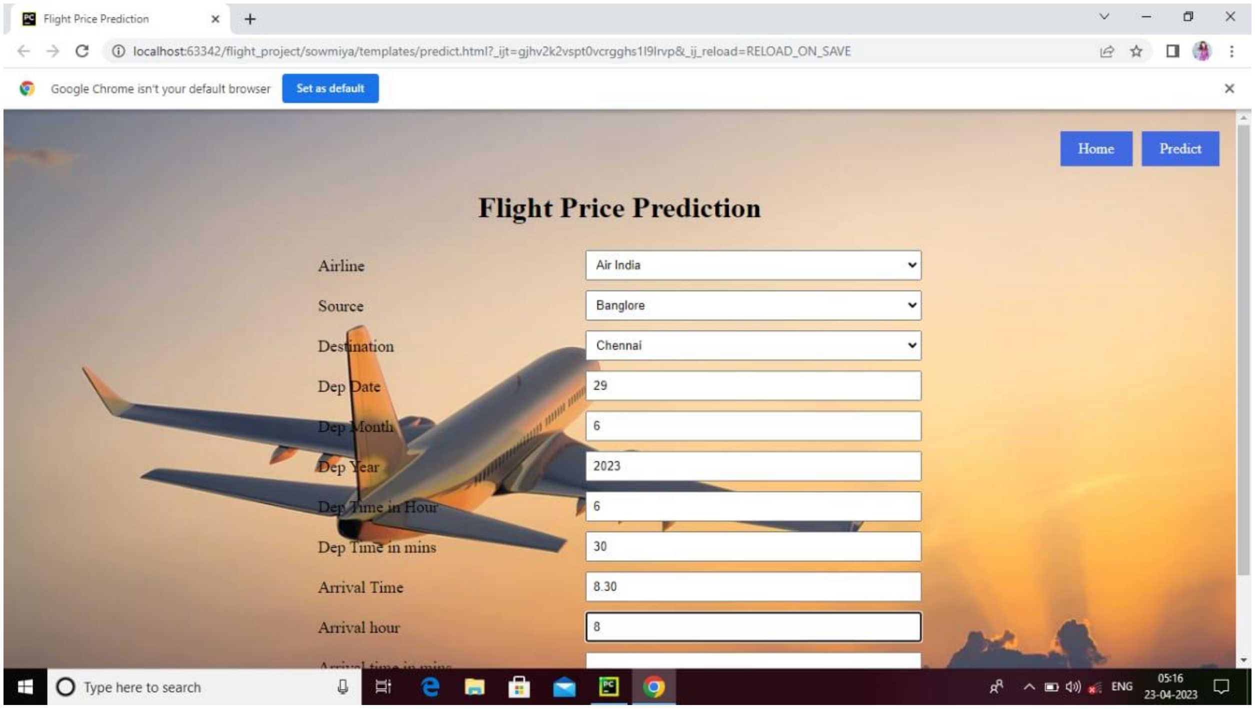
```
padding: 10px;
text-decoration: none;
}
</style>

</head>
<body>
<div class="main_section">
<div class="navbar">
<ul class="navbar_ul">
<li class="navbar_li"><a href="./home.html" class="nav-
icon">Home</a></li>
<li class="navbar_li"><a href="./prediction.html" class="nav-
icon">Predict</a></li>
</ul>
</div>

<h1 >Flight Price Prediction</h1>

<div class="form_section">
<p id="price_result">
Based on the given input, we can get the flight Price as 4824.76 INR.
</p>
</div>
</div>

</body>

</html>
```

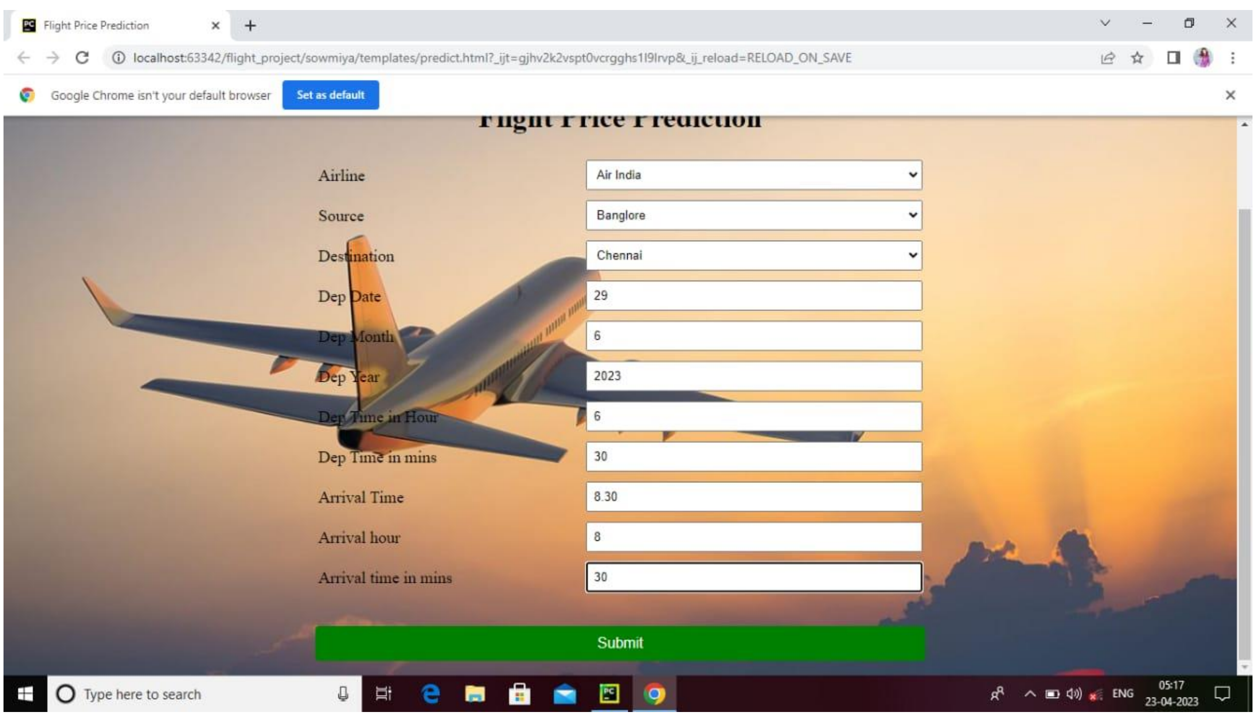
4.App.py

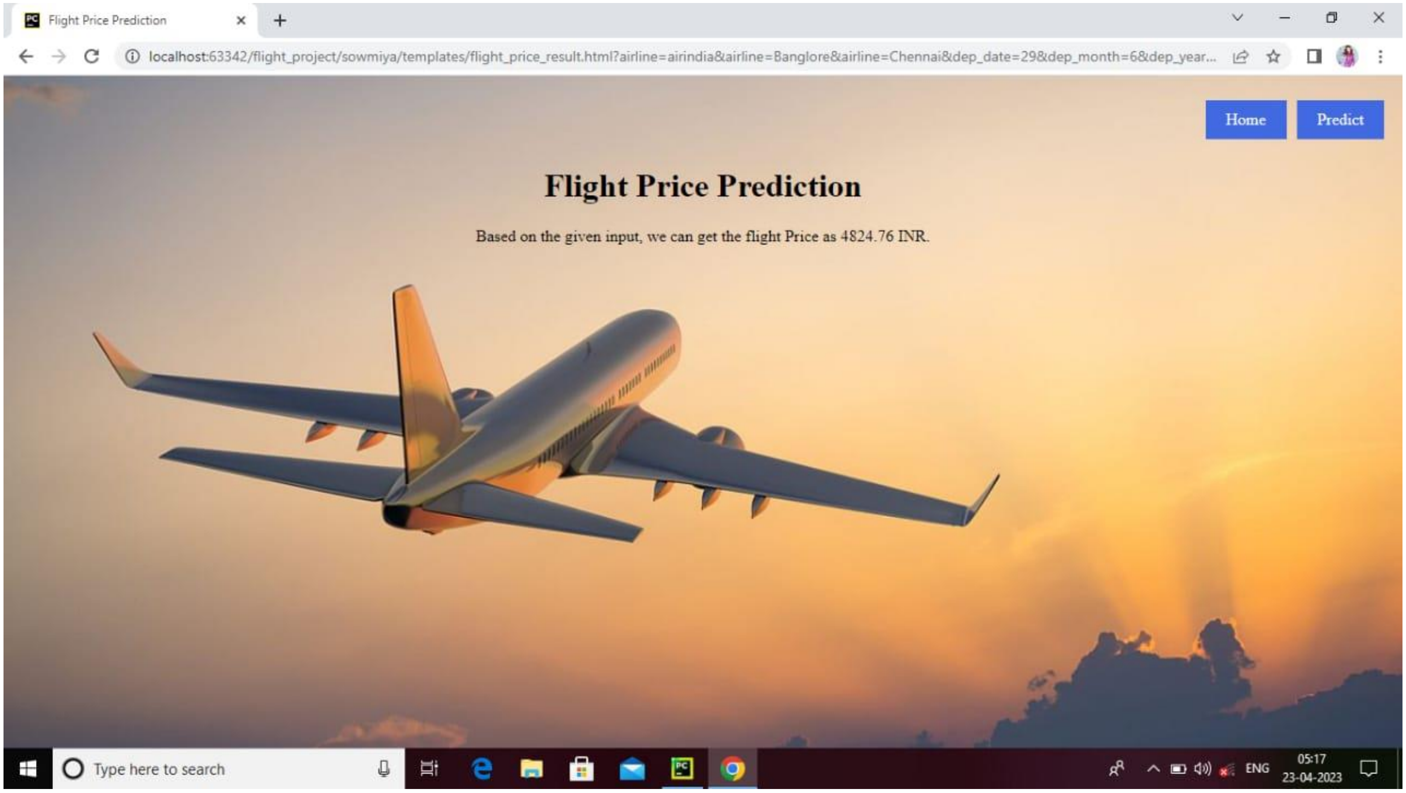
```
from flask import Flask,render_template,request
import numpy as np
import pickle
model=pickle.load(open(r”model1.pkl”,’rb’))
@app.route(“/home)
Def home():
   return render_template(‘home.html’)
@app.route(“/predict”)
Def home():
Return render_template (‘predict.html’)

@app.route(“/pred”,methods=[‘POST’,’GET’])
Def predict():
X=[[int(x) for x in request.form.value()]]
Print(x)
 X=np.array(x)
Print(x.shape)
 Print(x)
Pred=model.predict(x)
Print(pred)
Return render_template
(‘submit.html,prediction_text=pred)

If __name__==”__main__”:
App.run(debug=False)
```

Run the web application




Flight Price Prediction
localhost:63342/flight_project/sowmiya/templates/predict.html?_ijt=gjhv2k2vspt0vcrgghs1l9lrvp&_ij_reload=RELOAD_ON_SAVE
Google Chrome isn't your default browser
Set as default
Home
Predict
Flight Price Prediction
Airline
Air India
Source
Banglore
Destination
Chennai
Dep Date
29
Dep Month
6
Dep Year
2023
Dep Time in Hour
6
Dep Time in mins
30
Arrival Time
8.30
Arrival hour
8
Type here to search
ENG
05:16
23-04-2023


Flight Price Prediction
localhost:63342/flight_project/sowmiya/templates/predict.html?_ijt=gjhv2k2vspt0vcrgghs1l9lrvp&_ij_reload=RELOAD_ON_SAVE
Google Chrome isn't your default browser
Set as default
Airline
Air India
Source
Banglore
Destination
Chennai
Dep Date
29
Dep Month
6
Dep Year
2023
Dep Time in Hour
6
Dep Time in mins
30
Arrival Time
8.30
Arrival hour
8
Arrival time in mins
30
Submit
Type here to search
ENG
05:17
23-04-2023


Flight Price Prediction
localhost:63342/flight_project/sowmiya/templates/flight_price_result.html?airline=airindia&airline=Banglore&airline=Chennai&dep_date=29&dep_month=6&dep_year...
Home
Predict
Flight Price Prediction
Based on the given input, we can get the flight Price as 4824.76 INR.
Type here to search
ENG
05:17
23-04-2023