Optimizing Flight Booking Decisions Through Machine Learning price predictions

TEAM SIZE: 4

TEAM LEADER: Lokeshwari.M – Reg no:20UCA1134

TEAM MEMBERS:

1)Saranya.M – Reg no:20UCA1138

2)Mohanapriya.R – Reg no:20UCA1136

3)Sathiyashalani.S- Reg no:20UCA1139

1. INTRODUCTION

Flight ticket booking is the process of selecting and purchasing a ticket to travel on an airplane. The decision to book a flight ticket typically involves several factors, such as the purpose of travel, destination, budget, travel dates, and airline preferences. To make a flight ticket booking decision, individuals typically begin by researching flight options and comparing prices from different airlines. They may also consider factors such as flight duration, layover times, and in-flight amenities. Additionally, individuals may look for discounts or promotions that can help them save money on their ticket. Once a suitable flight option has been identified, individuals may proceed to book their ticket by providing their personal and payment information. It is important to carefully review the booking details before submitting the reservation to ensure that all information is correct. Overall, the decision to book a flight ticket requires careful consideration of various factors.

1.2 PURPOSE - THE USE OF THIS PROJECT

1.Convenience:

A flight ticket booking system allows travelers to search and book flights from thecomfort of their own home or office, saving them time and effort.

2.Cost savings:

By comparing prices and flight options from different airlines, travelers can find the best deals and save money on their travel expenses.

3.Flexibility:

Flight ticket booking systems often allow travelers to modify their bookings such as changing their travel dates or adding extra services, providing more flexibility in their travel plans.

4.Security:

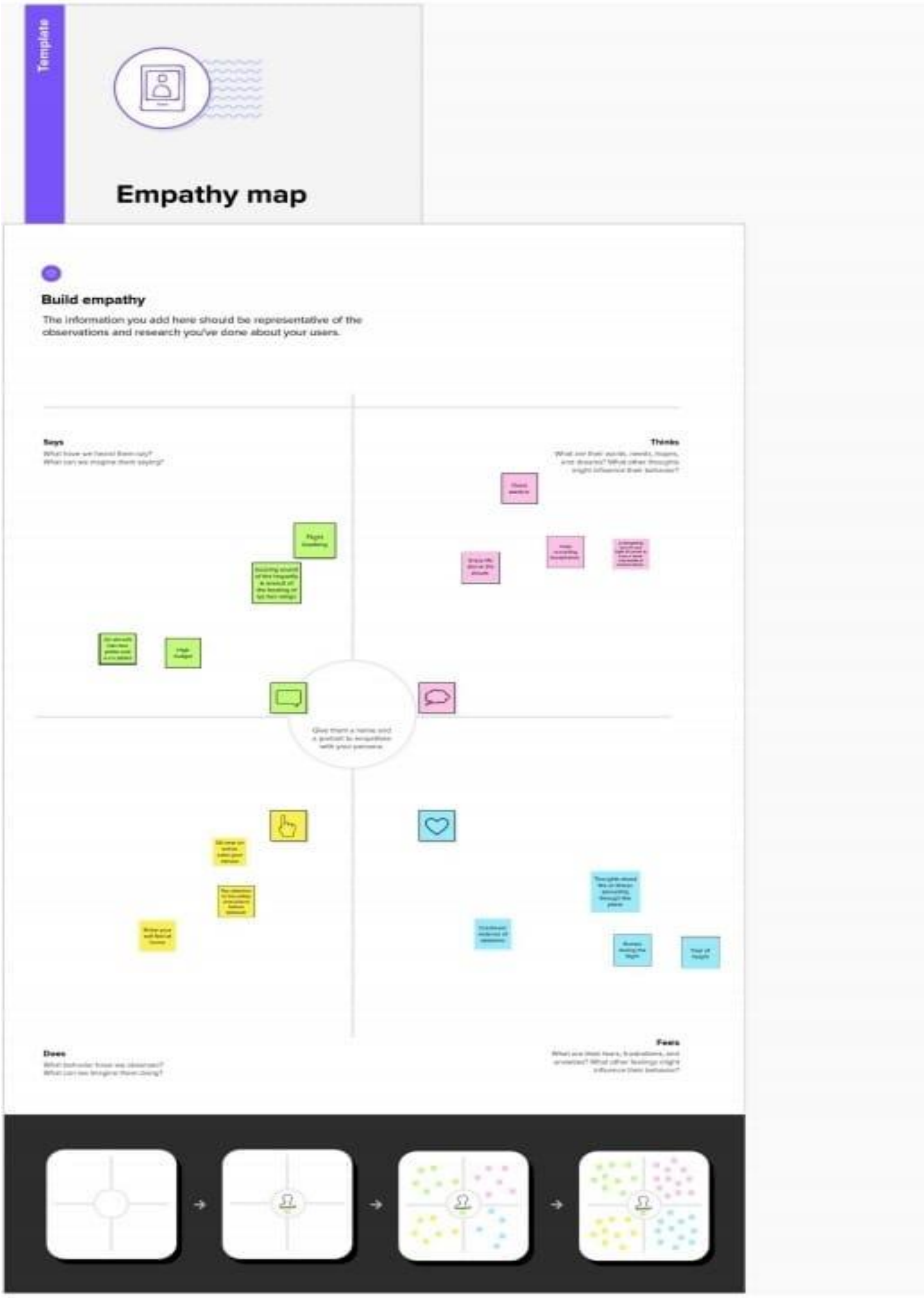
Reputable flight ticket booking systems provide secure payment options, protecting travelers' personal and financial information.

5.Accessibility:

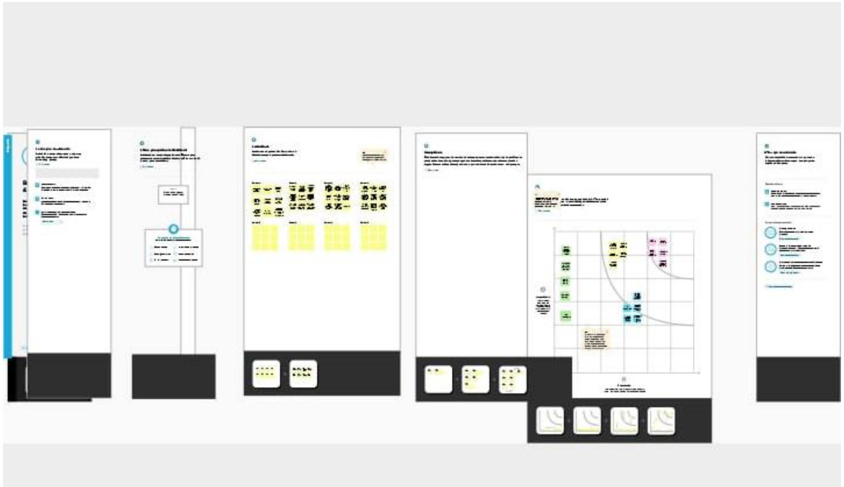
Flight ticket booking systems provide travelers with information about flight schedules, destinations, and airline policies, helping them make informed decisions about their travel plans.

Problem Definition & Design Thinking

2.1 Empathy map



2.2 ideation & brainstorming map screenshot



RESULT:

FINAL FINDINGS(OUTPUT) OF THE PROJECT ALONG WITH SCREENSHOTS

ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

Convenience:

Booking flight tickets online is quick and easy, allowing travelers to book their tickets from anywhere at any time.

Cost-effective:

Online flight booking sites offer competitive prices, enabling travelers to compare prices from different airlines and choose the best option that suits their budget.

Time-saving:

Booking flight tickets online eliminates the need for travelers to visit travel agencies or airline offices, saving time and effort.

24/7 availability:

Online flight booking sites are available 24/7, allowing travelers to book their tickets at any time.

Access to information:

Online flight booking sites provide travelers with information about flight schedules, airline policies, and other travel-related information, helping them make informed decisions about their travel plans.

Variety:

Online flight booking systems provide access to a wide range of airlines, flights, and destinations, giving travelers more options to choose from.

User-friendly interface:

Flight ticket booking systems are designed to be user-friendly, making it easy for travelers to search for flights, compare prices, and make bookings.

Personalization:

A flight ticket booking decision system can use your past booking history and preferences to suggest flights that best suit your needs. For example, if you prefer direct flights or have a preferred airline, the system can suggest flights that match your preferences.

Easy cancellation and rescheduling:

A flight ticket booking decision system can make it easy to cancel or reschedule your flight if your plans change. The system can handle the entire process for you, which saves you time and effort.

Enhanced customer experience:

A flight ticket booking decision system can provide a seamless and hassle-free experience for customers, from the time they search for flights to the time they board the plane. This can help improve customer satisfaction and loyalty.

Disadvantages:

Technical issues:

Online flight booking sites can sometimes experience technical problems, such as website crashes, which can disrupt the booking process.

Hidden fees:

Some online flight booking sites may not disclose all fees upfront, leading to unexpected charges and additional expenses for travelers.

Lack of personalized service:

Online flight booking sites may not provide the personalized service and attention to detail that travelers may receive when booking flights through a travel agent.

Inflexibility:

Once flight tickets are booked online, they may be subject to strict cancellation and refund policies, limiting travelers' flexibility in changing their travel plans.

limited flexibility:

A flight ticket booking decision system may not offer as much flexibility as booking through a travel agent. For example, the system may not be able to accommodate special requests or changes to your itinerary as easily as a human travel agent can.

Information overload:

A flight ticket booking decision system can provide a lot of information about flights, prices, and routes, which can be overwhelming for some customers. This can make it difficult to make a decision and may lead to confusion or frustration.

Security concerns:

A flight ticket booking decision system may require customers to provide personal information such as credit card details, which can pose a security risk if the system is not properly secured. Customers may also be at risk of fraud or identity theft if their information is stolen.

Lack of customer support:

While a flight ticket booking decision system may offer 24/7 availability, it may not provide the same level of customer support as a human travel agent. This can make it difficult to get help if you have a problem or question that cannot be resolved through the system.

APPLICATIONS:

Personalized recommendations:

AI can be used to analyze the user's travel history, preferences, and behavior to suggest flight options that match their interests and needs.

Price prediction:

AI can be used to analyze historical flight pricing data and current market conditions to predict the best time to buy a ticket to get the best deal.

Flight delay prediction:

AI can be used to analyze weather data, flight schedules, and other factors to predict the likelihood of a flight being delayed.

Seat selection:

AI can be used to suggest the best seats based on the user's preferences, such as window or aisle seats, proximity to the restroom, etc.

Virtual travel assistant:

AI can be used to provide personalized travel advice, suggest attractions and activities at the destination, and answer any travel-related questions.

Fraud detection:

AI can be used to detect fraudulent activities in ticket booking, such as credit card fraud, fake bookings, etc.

Language translation:

AI can be used to provide language translation services to users who speak different languages, making it easier for them to book flights.

CONCLUSION:

Flight Booking Considerations. Flight Booking Conclusion summarizing of flight ticket booking decision In summary, booking a flight ticket requires careful consideration of various factors such as travel dates, destination, airline, price, and travel restrictions. It is essential to research and compare multiple options to find the best deal and ensure a smooth and comfortable travel experience. Additionally, it is important to review the airline's policies regarding cancellations, changes, and refunds in case of any unforeseen circumstances. By taking these factors into account, one can make an informed decision and book a flight ticket that meets their needs and budget.More Conclusion summariszing In conclusion, booking a flight ticket is a crucial part of travel planning that can have a significant impact on the overall travel experience. It involves considering several factors, such as travel dates, destination, airline, price, and policies, to find the best option that fits. the traveler's needs and budget. It is essential to research and compare multiple options to make an informed decision and ensure a comfortable and hassle-free journey. Additionally, travelers should be aware of any travel restrictions or entry requirements that may affect their travel plans. By taking these factors into account and making a well-informed decision, travelers can book a flight ticket with confidence and enjoy a successful trip.

FUTURE SCOPE:

Artificial intelligence (AI) and machine learning (ML):

AI and ML can be used to analyze data and provide personalized recommendations to travelers, such as suggesting the best time to book flights, the cheapest routes, and preferred airlines.

Virtual and augmented reality (VR/AR):

VR and AR technologies can be used to provide travelers with immersive experiences of their travel destinations, helping them make informed decisions about their travel plans.

Mobile apps:

Mobile apps can provide travelers with real-time updates about their flights, such as gate changes, delays, and cancellations, providing more flexibility and convenience.

Blockchain technology:

Blockchain technology can be used to enhance the security and transparency of flight ticket bookings, providing travelers with more confidence in the booking process.

Sustainability initiatives:

Flight ticket booking systems can integrate sustainability initiatives, such as carbon offset programs, to provide travelers with more environmentally-friendly travel options.

Overall, the future scope of flight ticket booking decision system is exciting, with several potential developments that can improve the booking process for travelers and provide a more seamless travel experience.

SOURCE CODE:

```
import pandas as pd

import numpy as np

from pandas._libs import sparse

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

from scipy import stats

warnings.filterwarnings('ignore')

plt.style.use('fivethirtyeight')

from sklearn.datasets import load_breast_cancer

iris=lo
ad_iri
s()
x=iris.
data
```

```
y=iris.target

from sklearn.neighbors import

KNeighborsClassifierknn =

KNeighborsClassifier(n_neighb

ors = 1)
```

```
from

google.colab

import files

ubloaded=files.

upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to

enable.

Saving Data Train csv to Data Train (1) csv

```
data=pd.read_csv("

Data_Train.csv")

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 → 22:20 01:10 22 Mar 2h 50m

DEL non-stop No info 3897

CCU →

1 Air India 1/05/2019 IXR 05:50 13:15 7h 25m 2 stops No info

Kolkata Banglore → 7662

BBI

→

BL

R

Jet DEL

→

LKO

→

```
category =

['Airline','Source','Price','Destination','Additional_Info','Dep_

Time']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 conomy' 'Trujet'Source ['Banglore' 'Kolkata' 'Delhi' 'Chennai'
     'Mumbai']Price [ 3897 7662 13882 ... 9790 12352 12648]
    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']
    Dep_Time ['22:20' '05:50' '09:25' '18:05' '16:50' '09:00' '18:55' '08:00' '08:55'
     '11:25' '09:45' '20:20' '11:40' '21:10' '17:15' '16:40' '08:45' '14:00'
     '20:15' '16:00' '14:10' '22:00' '04:00' '21:25' '21:50' '07:00' '07:05'
     '09:50' '14:35' '10:35' '15:05' '14:15' '06:45' '20:55' '11:10' '05:45'
     '19:00' '23:05' '11:00' '09:35' '21:15' '23:55' '19:45' '08:50' '15:40'
     '06:05' '15:00' '13:55' '05:55' '13:20' '05:05' '06:25' '17:30' '08:20'
     '19:55' '06:30' '14:05' '02:00' '09:40' '08:25' '20:25' '13:15' '02:15'
     '16:55' '20:45' '05:15' '19:50' '20:00' '06:10' '19:30' '04:45' '12:55'
     '18:15' '17:20' '15:25' '23:00' '12:00' '14:45' '11:50' '11:30' '14:40'
     '19:10' '06:00' '23:30' '07:35' '13:05' '12:30' '15:10' '12:50' '18:25'
     '16:30' '00:40' '06:50' '13:00' '19:15' '01:30' '17:00' '10:00' '19:35'
     '15:30' '12:10' '16:10' '20:35' '22:25' '21:05' '05:35' '05:10' '06:40'
     '15:15' '00:30' '08:30' '07:10' '05:30' '14:25' '05:25' '10:20' '17:45'
     '13:10' '22:10' '04:55' '17:50' '21:20' '06:20' '15:55' '20:30' '17:25'
```

```
'09:30' '07:30' '02:35' '10:55' '17:10' '09:10' '18:45' '15:20' '22:50'
'14:55' '14:20' '13:25' '22:15' '11:05' '16:15' '20:10' '06:55' '19:05'
'07:55' '07:45' '10:10' '08:15' '11:35' '21:00' '17:55' '16:45' '18:20'
'03:50' '08:35' '19:20' '20:05' '17:40' '04:40' '17:35' '09:55' '05:00'
'18:00' '02:55' '20:40' '22:55' '22:40' '21:30' '08:10' '17:05' '07:25'
'15:45' '09:15' '15:50' '11:45' '22:05' '18:35' '00:25' '19:40' '20:50'
'22:45' '10:30' '23:25' '11:55' '10:45' '11:15' '12:20' '14:30' '07:15'
'01:35' '18:40' '09:20' '21:55' '13:50' '01:40' '00:20' '04:15' '13:45'
'18:30' '06:15' '02:05' '12:15' '13:30' '06:35' '10:05' '08:40' '03:05'
'21:35' '16:35' '02:30' '16:25' '05:40' '15:35' '13:40' '07:20' '04:50'
'12:45' '10:25' '12:05' '11:20' '21:40' '03:00']
```

```
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]
```

```
                ...
    106      [9, 04,
    78        2019]
    106      [27, 04,
    79          2019]
    106      [27, 04,
    80          2019]
    106      [01, 03,
    81          2019]
    106        [9, 05,
    82          2019]
    Name: Date_of_Journey, Length: 10683, dtype: object
```

```
data['Date']=data.Date_of_Jo
urney.str[0]
data['Month']=data.Date_of_J
ourney.str[1]
data['Year']=data.Date_of_Jo
urney.str[2]
```

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

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

```
data.Route.str.spl
it('@')data.Route
```

```
0                    BLR → DEL
1         CCU → IXR → BBI → BLR
2         DEL → LKO → BOM → COK
3              CCU → NAG → BLR
4              BLR → NAG → DEL
                 ...
106       CCU → BLR
78
106       CCU → BLR
79
106        BLR → DEL
80
106        BLR → DEL
81
106    DEL → GOI →
82        BOM → COK
Name: Route, Length: 10683, dtype: object
```

```
data['city1']=data.Rte.
str[0]
data['city2']=datastr[1
]
data['city3']=data.Rou
te.str[2]
data['city4']=data.Rou
te.str[3]
data['city5']=data.Rou
te.str[4]
data['city6']=data.Rou
te.str[5]
```

```
data["Price"]


0       389
          7
1       766
          2
2       138
         82
3       621
          8
4       133
         02
        ...
106      41
78       07
106     414
79        5
106     722
80        9
106     126
81       48
106     117
82       53
Name: Price, Length: 10683, dtype: int64


my_data={'Dep_Time'}



data["Dep_Time"]


0       22:20
```

```
1        05:50
2        09:25
3        18:05
4        16:50
         ...
1067819:55
1067920:45
1068008:20
1068111:30
1068210:55
Name: Dep_Time, Length: 10683, dtype: object
```

```
data.Dep_Time=data.Dep_Time.str.split(':')
```

```
data['Dep_Time_Hour'] =
data.Dep_Time.str[0]
data['Dep_Time_Hour'] =
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_Min
s.str.split('m')
data.Travel_Mins=data.Travel_Min
s.str[0]


data.Total_Stops.replace('non_stop',
0,inplace=True)
data.Total_Stops=data.Total_Stops.s
tr.split(':')
data.Total_Stops=data.Total_Stops.str[0]
```

```
data.Total_Stops.replace('non_stop',
0,inplace=True)
data.Total_Stops=data.Total_Stops.s
tr.split(' ')
data.Total_Stops=data.Total_Stops.str[0]



data.Additional_Info.unique()


    array(['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'], dtype=object)



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

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	1
city3	1
city4	1
city5	1
city6	1
Dep_Time_Hour	0
Arrival_date	10683
Time_of_Arrival	0

```
Arrival_Time_Hour      0
Arrival_Time_Mins      0
Travel_Hours           0
Travel_Mins         1032
dtype: int64
```

```
data.drop(['city4','city5','city6'], axis=1, inplace=True)
```

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

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

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

```
Airline            0
Source             0
Destination        0
Arrival_Time       0
Total_Stops        1
Additional_Info    0
Price              0
```

Date	0
Month	0
Year	0
city1	1
city2	1
city3	1
Dep_Time_Hour	0
Arrival_date	10683
Arrival_Time_Hour	0
Arrival_Time_Mins	0
Travel_Hours	0
Travel_Mins	103
dtype: int64	2

```
data['city3'].fillna('None',inplace=True)
```

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

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

```
data.info()
```

<class

'pandas.core.frame.DataFrame'>RangeIndex: 10683 entries, 0 to 10682 Data columns (total 19 columns):

#	Column	Non-Null Count	Dtype
0	Airline	10683 non-null	object
1	Source	10683 non-null	object
2	Destination	10683 non-null	object
3	Arrival_Time	10683 non-null	object
4	Total_Stops	10682 non-null	object
5	Additional_Info	10683 non-null	object
6	Price	10683 non-null	int64
7	Date	10683 non-null	object

8	Month	10683	obje
		non-null	ct
9	Year	10683	obje
		non-null	ct
1	city1	10682	obje
0		non-null	ct
1	city2	10682	obje
1		non-null	ct
1	city3	10683	obje
2		non-null	ct
1	Dep_Time_H	10683	obje
3	our	non-null	ct
1	Arrival_date	0 non-null	float
4			64
1	Arrival_Time	10683	obje
5	_Hour	non-null	ct
1	Arrival_Time	10683	obje
6	_Mins	non-null	ct
1	Travel_Hour	10683	obje
7	s	non-null	ct
1	Travel_Mins	10683	obje
8		non-null	ct

dtypes: float64(1), int64(1),

object(17)memory usage:

1.5+ MB

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



data.Date=data.Date.astype('int64')
data.Month=data.Month.a
stype('int64')
data.Year=data.Year.astyp
e('int64')
data.Dep_Time_Hour=data.Dep_Time
_Hour.astype('int64')
data.Dep_Time_Hour=data.Dep_Time
_Hour.astype('int64')
data.Arrival_Time_Hour=data.Arrival_Time_
Hour.astype('int64')
data.Arrival_Time_Mins=data.Arrival_Time_
Mins.astype('int64')
```

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

	Airline	Source	Destination	Arrival_Time	Total_Stops	Additional_Info	Price	Date
	Month	Year	city1	city2	city3	Dep_T		
6474	Air India	Mumbai	Hyderabad	[16:55]	2 stops	No info	17327	
	6	3	2019 B	O	M			

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

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

```
categorical=['Airline','Source','Destination','Additional_Info','City1','Price']
numerical=['Total_stops','Date','Month','Year','Dep_Time_Hour','Dep_Time_Mins','Arrival_date','Arriv
al_Time_Hour','Arrival_Time_Mins','Tr
```

```
from sklearn.preprocessing
import LabelEncoder
le=LabelEncoder()
```

```
data.Airline=le.fit_transform(d
ata.Airline)
data.source=le.fit_transform(d
ata.Source)
data.Destination=le.fit_transform(da
```

```
ta.Destination)
data.Total_Stops=le.fit_transform(da
ta.Total_Stops)
data.cityt1=le.fit_transform(data.city
1)
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()
```

	Airline	Source	Destination	Arrival_Time	Total_Stops	Additional_Info	Price	Date	Month	Year	city1	city2	city3	Dep_Tim
0	3	Banglore	5	[01:10 22 Mar]	4	7	3897	24	3	2019	B	3	4	
1	1	Kolkata	0	[13:15]	1	7	7662	1	5	2019	C	1	5	
2	4	Delhi	1	[04:25 10 Jun]	1	7	13882	9	6	2019	D	2	1	
3	3	Kolkata	0	[23:30]	0	7	6218	12	5	2019	C	1	5	
4	3	Banglore	5	[21:35]	0	7	13302	1	3	2019	B	3	4	

```
data.head()
```

	Airline	Source	Destination	Arrival_Time	Total_Stops	Additional_Info	Price	Date	Month	Year	city1	city2	city3	Dep_Tim
0	3	Banglore	5	[01:10 22 Mar]	4	7	3897	24	3	2019	B	3	4	
1	1	Kolkata	0	[13:15]	1	7	7662	1	5	2019	C	1	5	
2	4	Delhi	1	[04:25 10 Jun]	1	7	13882	9	6	2019	D	2	1	
3	3	Kolkata	0	[23:30]	0	7	6218	12	5	2019	C	1	5	
4	3	Banglore	5	[21:35]	0	7	13302	1	3	2019	B	3	4	

```
data=data[['Airline','Source','Destination','Date','Month','Year','Dep_Time_Hour','Arrival_Time_Mins',
'Arrival_Time']]
```

```
data.head()
```

	Airline	Source	Destination	Date	Month	Year	Dep_Time_Hour	Arrival_Time_Mins	Arrival_Time
0	3	Banglore	5	24	3	2019	20	10 22 Mar	[01:10 22 Mar]
1	1	Kolkata	0	1	5	2019	50	15	[13:15]
2	4	Delhi	1	9	6	2019	25	25 10 Jun	[04:25 10 Jun]
3	3	Kolkata	0	12	5	2019	5	30	[23:30]
4	3	Banglore	5	1	3	2019	50	35	[21:35]

```
data.describe()
```

Airline	Destination	Date	Month	Year	Dep_Time_Hour
count	10682.0	10682.0	10682.0	1068	10682.00

	10682.000000	00000	00000	00000	2.0	0000
mean	3.966205	1.435967	13.509081	4.708762	2019.0	24.408819
std	2.352090	1.474773	8.479363	1.164294	0.0	18.767225
min	0.000000	0.000000	1.000000	3.000000	2019.0	0.000000
25%	3.000000	0.000000	6.000000	3.000000	2019.0	5.000000
50%	4.000000	1.000000	12.000000	5.000000	2019.0	25.000000
75%	4.000000	2.000000	21.000000	6.000000	2019.0	40.000000
max	11.000000	5.000000	27.000000	6.000000	2019.0	55.000000

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

```
<Figure size 2000x4500 with 0 Axes>
<Figure size 2000x4500 with 0 Axes>
```

```
for i in
categorical:
plt.subplot(6,3,
c)
```

```
sns.countplot(d
ata[i])
plt.xticks(rotatio
n=90)
plt.tight_layout(p
ad=3.0)c=c+1

plt.show()
```

```
from sklearn.datasets import load_iris

iris=load_iris()

df=pd.DataFrame(iris.data,columns=iris.feature_names)

price_list=pd.DataFrame({'price:prices'})

price_list
```

	0
0	price:prices

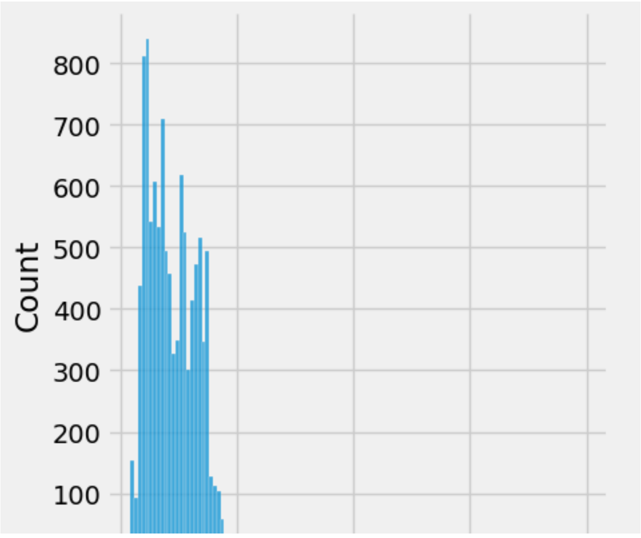
Price

sns.displot(data.Price)

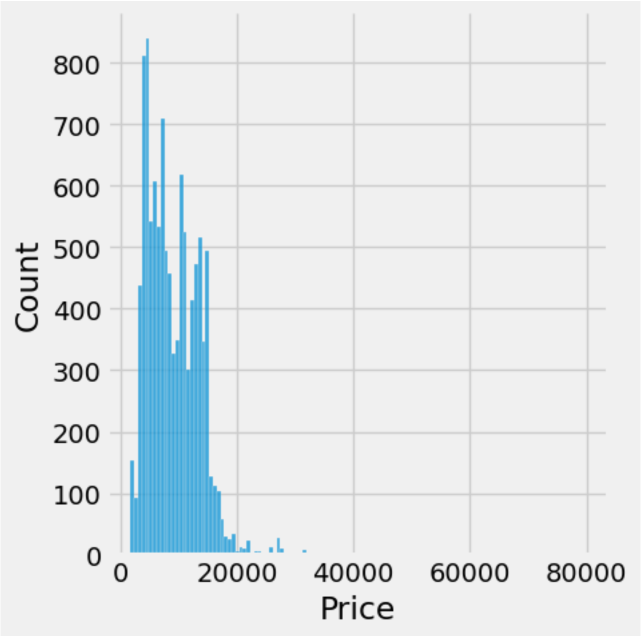
Final output:

Webframe work

<seaborn.axisgrid.FacetGrid at 0x7fd58ceddc10>

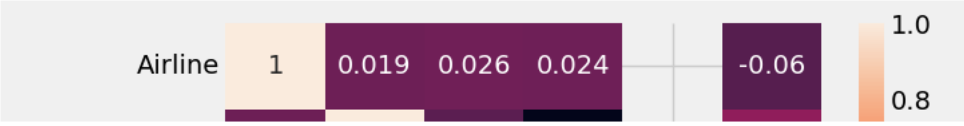


plt.figure(figsize=(
15,8))
sns.displot(data.Pr
ice)



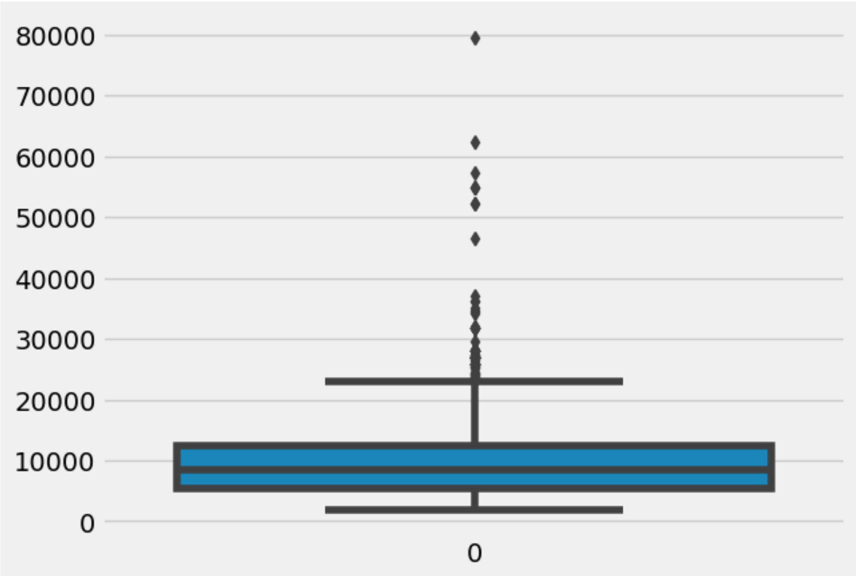
```
sns.heatmap(data.corr(),annot=True)
```

<Axes: >



import seaborn as sns

sns.boxplot(data['Price'])



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

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()

knn.fit(x,y)

▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)

print(x_scaled)

```
[[-9.00681170e-01     1.01900435e+00     -
  1.34022653e+00    -1.31544430e+00]    [-
  1.14301691e+00    -1.31979479e-01     -
  1.34022653e+00    -1.31544430e+00]    [-
  1.38535265e+00     3.28414053e-01     -
  1.39706395e+00    -1.31544430e+00]    [-
  1.50652052e+00     9.82172869e-02     -
  1.28338910e+00    -1.31544430e+00]    [-
  1.02184904e+00     1.24920112e+00     -
  1.34022653e+00    -1.31544430e+00]    [-
  5.37177559e-01     1.93979142e+00     -
  1.16971425e+00    -1.05217993e+00]    [-
  1.50652052e+00     7.88807586e-01     -
```

1.34022653e+00 -1.18381211e+00] [-
1.02184904e+00 7.88807586e-01 -
1.28338910e+00 -1.31544430e+00] [-
1.74885626e+00 -3.62176246e-01 -
1.34022653e+00 -1.31544430e+00] [-
1.14301691e+00 9.82172869e-02 -
1.28338910e+00 -1.44707648e+00] [-
5.37177559e-01 1.47939788e+00 -
1.28338910e+00 -1.31544430e+00] [-
1.26418478e+00 7.88807586e-01 -
1.22655167e+00 -1.31544430e+00] [-
1.26418478e+00 -1.31979479e-01 -
1.34022653e+00 -1.44707648e+00] [-
1.87002413e+00 -1.31979479e-01 -
1.51073881e+00 -1.44707648e+00] [-
5.25060772e-02 2.16998818e+00 -
1.45390138e+00 -1.31544430e+00] [-
1.73673948e-01 3.09077525e+00 -
1.28338910e+00 -1.05217993e+00] [-
5.37177559e-01 1.93979142e+00 -
1.39706395e+00 -1.05217993e+00] [-
9.00681170e-01 1.01900435e+00 -
1.34022653e+00 -1.18381211e+00] [-
1.73673948e-01 1.70959465e+00 -
1.16971425e+00 -1.18381211e+00] [-
9.00681170e-01 1.70959465e+00 -
1.28338910e+00 -1.18381211e+00] [-
5.37177559e-01 7.88807586e-01 -
1.16971425e+00 -1.31544430e+00] [-
9.00681170e-01 1.47939788e+00 -
1.28338910e+00 -1.05217993e+00] [-
1.50652052e+00 1.24920112e+00 -
1.56757623e+00 -1.31544430e+00] [-
9.00681170e-01 5.58610819e-01 -
1.16971425e+00 -9.20547742e-01] [-
1.26418478e+00 7.88807586e-01 -
1.05603939e+00 -1.31544430e+00] [-
1.02184904e+00 -1.31979479e-01 -
1.22655167e+00 -1.31544430e+00] [-
1.02184904e+00 7.88807586e-01 -
1.22655167e+00 -1.05217993e+00] [-
7.79513300e-01 1.01900435e+00 -
1.28338910e+00 -1.31544430e+00] [-
7.79513300e-01 7.88807586e-01 -
1.34022653e+00 -1.31544430e+00] [-
1.38535265e+00 3.28414053e-01 -
1.22655167e+00 -1.31544430e+00] [-
1.26418478e+00 9.82172869e-02 -
1.22655167e+00 -1.31544430e+00] [-
5.37177559e-01 7.88807586e-01 -
1.28338910e+00 -1.05217993e+00]

```
[-7.79513300e-01  2.40018495e+00  -
1.28338910e+00  -1.44707648e+00]  [-
4.16009689e-01  2.63038172e+00  -
1.34022653e+00  -1.31544430e+00]  [-
1.14301691e+00  9.82172869e-02  -
1.28338910e+00  -1.31544430e+00]  [-
1.02184904e+00  3.28414053e-01  -
1.45390138e+00  -1.31544430e+00]  [-
4.16009689e-01  1.01900435e+00  -
1.39706395e+00  -1.31544430e+00]  [-
1.14301691e+00  1.24920112e+00  -
1.34022653e+00  -1.44707648e+00]  [-
1.74885626e+00  -1.31979479e-01  -
1.39706395e+00  -1.31544430e+00]  [-
9.00681170e-01  7.88807586e-01  -
1.28338910e+00  -1.31544430e+00]  [-
1.02184904e+00  1.01900435e+00  -
1.39706395e+00  -1.18381211e+00]  [-
1.62768839e+00  -1.74335684e+00  -
1.39706395e+00  -1.18381211e+00]  [-
1.74885626e+00  3.28414053e-01  -
1.39706395e+00  -1.31544430e+00]  [-
1.02184904e+00  1.01900435e+00  -
1.22655167e+00  -7.88915558e-01]  [-
9.00681170e-01  1.70959465e+00  -
1.05603939e+00  -1.05217993e+00]  [-
1.26418478e+00  -1.31979479e-01  -
1.34022653e+00  -1.18381211e+00]  [-
9.00681170e-01  1.70959465e+00  -
1.22655167e+00  -1.31544430e+00]  [-
1.50652052e+00  3.28414053e-01  -
1.34022653e+00  -1.31544430e+00]  [-
6.58345429e-01  1.47939788e+00  -
1.28338910e+00  -1.31544430e+00]  [-
1.02184904e+00  5.58610819e-01  -
1.34022653e+00  -1.31544430e+00]  [
1.40150837e+00  3.28414053e-01
5.35408562e-01  2.64141916e-01]  [
6.74501145e-01  3.28414053e-01
4.21733708e-01  3.95774101e-01]  [
1.28034050e+00  9.82172869e-02
6.49083415e-01  3.95774101e-01]  [-
4.16009689e-01  -1.74335684e+00
1.37546573e-01  1.32509732e-01]  [
7.95669016e-01  -5.92373012e-01
4.78571135e-01  3.95774101e-01]  [-
1.73673948e-01  -5.92373012e-01
4.21733708e-01  1.32509732e-01]  [
5.53333275e-01  5.58610819e-01
5.35408562e-01  5.27406285e-01]  [-
1.14301691e+00  -1.51316008e+00  -
2.60315415e-01 -2.62386821e-01]
x_scaled = scaler.fit_transform(x)

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

```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

```
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()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
8990	Jet Airways	12/03/2019	Mumbai	Hyderabad	BOM → VNS → DEL → HYD	06:30	16:35	10h 5m	2 stops	No info
3684	Jet Airways	9/05/2019	Delhi	Cochin	DEL → BOM → COK	11:30	12:35 10 May	25h 5m	1 stop	In-flight meal not included
1034	SpiceJet	24/04/2019	Delhi	Cochin	DEL → MAA →	15:45	22:05	6h 20m	1 stop	No info

```
from sklearn.ensemble import AdaBoostRegressor

rfr = RandomForestRegressor()

gb = GradientBoostingRegressor()

ad = AdaBoostRegressor()

from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor, AdaBoostRegressor
rfr=RandomForestRegressor()
gb=
Gra
dien
tBoo
sting
Regr
esso
r()
ad=
Ada
Boos
tReg
ress
or()

from sklearn.metrics import
r2_score,mean_absolute_error,mean
_squared_errorfor 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_sc
  ore(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 Absolute Error
is",mean_absolute_error(y_pre
d,y_test))print("Mean Squred
Error
is",mean_squared_error(y_pre
d,y_test))
print("Root Mean Squared Error is",
(mean_squared_error(y_pred,y_test,squared=False)))
```

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

```
k
n
n
=
K
N
e
i
g
h
b
o
r
s
R
e
g
r
e
s
s
o
r
```

```
()
s
v
r
=
S
V
R
()
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=r2_sc
  ore(y_train,i.predi
  ct(x_train))if
  abs(train_score-
  test_score)<=0.1:
    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 Absolute Error
    is',mean_absolute_error(y_pre
    d,y_test))print('Mean Squred
    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.model_selec
tion import
cross_val_scorefor i
in range(2,5):
  cv=cr
  oss_v
  al_sco
  re(rfr,
```

```
x,y,cv
=i)
print(
rfr,cv.
mean(
))
```

```
RandomForestRegressor(max_features='sqrt', n_estimators=10) -
2.0431999999999997
RandomForestRegressor(max_features='sqrt', n_estimators=10) 0.0
RandomForestRegressor(max_features='sqrt', n_estimators=10)
0.38848557692307695
```

```
from sklearn.model_selection import RandomizedSearchCV

param_grid={'n_estimators':[10,3
            0,50,70,100],'max_depth
            ':[None,1,2,3],
            'max_features':['auto','sq
            rt']}
rfr=RandomForestRegressor()
rf_res=RandomizedSearchCV(estimator=rfr,param_d

istributions=param_grid,cv=3,verbose=2,n_jobs=-1)

rf_res.fit(x_train,y_train)

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

rfr=RandomForestRegressor(n_estimat
ors=10,max_features='sqrt',max_dept
h=None)rfr.fit(x_train,y_train)
y_train
_pred=
rfr.pre
dict(x_
```

```
train)
y_test_
pred=r
fr.pred
ict(x_te
st)
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:prices'})
```

```
price_list
```

	0
0	price:prices

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

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