FLIGHT PRICE PREDICTION

A PROJECT REPORT

for

DATA MINING TECHNIQUES (ITE2006)

in

B.Tech – Information Technology and Engineering

by

PRIYANSHI SINGH(19BIT0111)

BHAVAY ARORA (19BIT0355)

ROSHNI ROY (19BIT0391)

Under the Guidance of

Dr. RANICHANDRA C

Associate Professor, SITE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

June, 2021

DECLARATION BY THE CANDIDATE

We hereby declare that the project report entitled **"FLIGHT PRICE PREDICTION"** submitted by us to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course **Data Mining Techniques (ITE2006)** is a record of bonafide project work carried out by us under the guidance of **Dr. Ranichandra C.** We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course.

Place : Vellore

Signature

Date : 8th june

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology & Engineering [SITE]

CERTIFICATE

This is to certify that the project report entitled **"FLIGHT PRICE PREDICTION "** submitted by **PRIYANSHI SINGH(19BIT0111), BHAVAY ARORA(19BIT0355) and ROSHNI ROY(19BIT0391)** to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course **Data Mining Techniques (ITE2006)** is a record of bonafide work carried out by them under my guidance.

Dr. RANICHANDRA C

GUIDE

Associate Professor, SITE

FLIGHT PRICE PREDICTION

Abstract

Flight Price Prediction is the system which predicts the Flight price in earlier. airline ticket purchasing from the consumer's perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we simulate various models for computing expected future prices. Airlines use using sophisticated tactics which they call **"Revenue management"** or **"Yield management"**. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on –

1. Time of purchase patterns (making sure last-minute purchases are expensive)
2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

Keywords- NIL

I. INTRODUCTION

As someone who purchases flights frequently, I would like to be able to predict when the best time is to buy in order to get the best deal. I have heard some people claiming that certain days of the week are when prices are lowest, and it's likely that that demand for flights is higher at certain hours, which could indicate higher prices. It's possible this varies depending on various other aspects, like the length of the flight, the date and time of the flight, and how much time there is until the flight. In this project, I chose to focus on aspects that are visible on the consumer side and predict only the binary class of whether the price will increase or not, which is essentially whether one should buy now or wait. The input is the time of the day of the request, the time of the week of the request, the time of the day of the flight, the time of the week of the flight, the number of stops, the duration of the flight, the number of hours between the request and the flight, and the current price of the flight. The output is the previously described binary class with 1 indicating "should wait" and 0 indicating "should buy."

II. BACKGROUND

We've tried to create a machine learning algorithm in python which makes flight price predictions. We've used the linear regression model in order to find the best fit line as far as our predictions and the actual prices are concerned.

III. Literature Survey

[1] The algorithm used in this paper Machine learning, adaboost, Decision tree, SVM, naive Bayers, Desired data, Flight numbers, departure date are datasets identified. Explicitly consider holidays and festivals in the Indian market to see how it affect the tickets pricing. The considering of lowest prices, they are letting go of other airlines prices for the day/route.

[2] Feature extracted from external factors such as social media data are not considered.To increase revenue of airline companies. PLS regression, decision tree SVR Ridge regression are few techniques involved. Flight number, numbers of hours ,airline route, current price some of them estimated. Data collected of accuracy of 60% prior to departure date.

[3] Artificial Neural Networks (ANN), Genetic Algorithm(GA), Supervised learning are the algorithms used. The problem of market segments level airfare price prediction by using public available datasets and a novel machine learning framework to predict market segments level airfare price.The framework to include ticket transaction information,which can provide more details about a specific itinerary.

[4] In this paper the Modern airlines use sophisticated pricing models to maximize their revenue, which results in highly volatile airfares. We propose a modification of the original algorithm to make a regressor that is capable of learning incrementally from streaming price series, with an extra ability of multi-step ahead forecasting. We further evaluate the forecast model on realworld price data collected from diverse routes and discuss its performance with respect to short- 4 term and long-term prediction.

[5] This paper deals with the problem of airfare prices prediction. For this purpose a set of features characterizing a typical flight is decided, supposing that these features affect the price of an air ticket. The features are applied to eight state of the art machine learning (ML) models, used to predict the air tickets prices, and the performance of the models is compared to each other. Along with the prediction accuracy of each model, this paper studies the dependency of the accuracy on the feature set used to represent an airfare.

[6] In this paper, we present a review of customer side and airlines side Our review analysis shows that models on both sides rely on limited set of features such as historical ticket price data, ticket purchase date and departure date. Features extracted from external factors such as

social media data and search engine query are not considered.. Our review analysis shows that models on both sides rely on limited set of features such as historical ticket price data, ticket purchase date and departure date. Features extracted from external factors such as social media data and search engine query are not considered. Therefore, we introduce and discuss the concept of using social media data for ticket/demand prediction.

[7] This research proposes four statistical regression models for airline ticket prices and compares the goodness of fit. With this prediction model passengers can make a more informed decision whether to buy the ticket or wait a little longer. We used a data set containing 116,411observations of ticket prices of 1,171 different flights from San Francisco Airport to John F. Kennedy Airport, these observations have been made on a daily basis by Infare

[8] the paper says that data were collected from low-cost airline passengers traveling from Taiwan to Singapore. The results of the continuous multinomial logit model indicate that lower fares increase the number of bookings and heterogeneous preferences in booking timing are present. Some travelers tend to book flights earlier than the other groups: these are the price-sensitive customers.

[9] This paper identifies a source of price discrimination utilized by airlines – price discrimination based on the day-of-the-week that a ticket is purchased. Using unique transaction data, we compare tickets on the same airline and route that are purchased on different days of the week, after controlling for the day of week of travel, the ticket restrictions, the demand characteristics of the flights, and the number of days in advance that the ticket is purchased.

[10] This paper aims to investigate whether price discrimination practices are applied by airlines in online ticket sales The two tested full-service carriers showed typical pricing patterns, whereas the two low-cost carriers did not show many fluctuations. Most changes occurred in the morning (56,5%) and the majority (51,1%) of fluctuations were minor price changes of 0%-5%. Therefore in conclusion, the appliance of customer profiling as pricing practice is not confirmed based on this particular research.

[11] The paper used for determining the minimum price or the best time to buy a ticket is the key issue. The concept of “tickets bought in advance are cheaper” is no longer working. . The ticket price may be affected by several factors thus may change continuously. To address this,

various studies were conducted to support the customer in determining an optimal ticket purchase time and ticket price prediction

[12] This paper introduces an algorithm that optimizes purchase timing on behalf of customers and provides performance estimates of its computed action policy. Given a desired flight route and travel date, the algorithm uses machine-learning methods on recent ticket price quotes from many competing airlines to predict the future expected minimum price of all available flights.

[13] This paper presents general patterns in airline pricing behavior and a methodology for analyzing different routes and/or carriers. The purpose is to provide customers with the relevant information they need to decide the best time to purchase a ticket, striking a balance between the desire to save money and any time restraints the buyer may have.

[14] This paper results that the price level is highest during the holiday period to the lowest-ranked tourist destinations using prediction method and that the price dispersion is highest during the holiday period but to the top-ranked tourist destinations. the difference between tourist and business travel is based on consumer characteristics; however, the main contribution of this paper is to examine this difference with respect to the characteristics of the destination.

[15] In this paper, we address the problem of airfare forecast and present a systematic approach that covers the most important aspects of building a forecast service, including data modelling, forecast algorithm and long-term prediction strategies. the prediction task, we specifically investigate Learn++.NSE, an incremental ensemble classifier designed for learning in nonstationary environments.

IV. DATASET DESCRIPTION & SAMPLE DATA

Historical air flight prices are not readily available on the internet.
The only option that we have is to use some resources and collect data over a period of time. There are many **APIs** made available by companies like Amadeus, Sky Scanner.

Therefore, we decided to collect data from a web spider that extracts the required values from a website and stores it as a CSV file. We collected the scraped data from the Kaggle website using a manual spider made in Python.

Attributes:

- Airline Name
- Flight Duration
- Flight Fly score
- AirlineRating
- Date
- Departure Airport
- Arrival Airport

We got 11.7k observation from scraped data. From that, we took 2683 observation for this system.

Sample data:

Column1	**Column 2**	**Column 3**	**Column 4**	**Column5**	**Column 6**	**Column 7**	**Column8**	**Colu mn9**	**C**
airline_name	legroom	seat_comfort	entertainment	customer_service	value_money	cleanliness	checkin_boarding	food_beverage	
INDIGO	4.0 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	4.5 of 5 bubbles	4.0 of 5 bubbles	3.5 of 5 bubbles	4.0 of 5 bubbles	
AIR ASIA	4.0 of 5 bubbles	4.0 of 5 bubbles	4.5 of 5 bubbles	4.0 of 5 bubbles	4.5 of 5 bubbles	4.5 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	
Air India	3.5 of 5 bubbles	3.5 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	3.5 of 5 bubbles	3.0 of 5 bubbles	
Jet Airways	3.0 of 5 bubbles	3.0 of 5 bubbles	3.5 of 5 bubbles	3.0 of 5 bubbles	3.5 of 5 bubbles	3.5 of 5 bubbles	3.0 of 5 bubbles	3.0 of 5 bubbles	
AIR ASIA	3.5 of 5 bubbles	3.5 of 5 bubbles	4.0 of 5 bubbles	3.5 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	3.5 of 5 bubbles	3.5 of 5 bubbles	
Kingfish Airlines	3.0 of 5 bubbles	3.0 of 5 bubbles	3.5 of 5 bubbles	3.0 of 5 bubbles	3.5 of 5 bubbles	3.5 of 5 bubbles	3.0 of 5 bubbles	3.0 of 5 bubbles	
AIR INDIA	4.0 of 5 bubbles	4.0 of 5 bubbles	4.5 of 5 bubbles	4.0 of 5 bubbles	4.5 of 5 bubbles	4.5 of 5 bubbles	4.0 of 5 bubbles	3.5 of 5 bubbles	

JetBlue	4.0 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	4.5 of 5 bubbles	4.5 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles
Multiple carries	4.0 of 5 bubbles	3.5 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	4.0 of 5 bubbles	3.5 of 5 bubbles	3.5 of 5 bubbles
Jet airways	2.5 of 5 bubbles	2.5 of 5 bubbles	3.0 of 5 bubbles	3.0 of 5 bubbles	3.5 of 5 bubbles	3.0 of 5 bubbles	2.0 of 5 bubbles	1.5 of 5 bubbles
Airlines	2.5 of 5 bubbles	2.5 of 5 bubbles	2.5 of 5 bubbles	3.0 of 5 bubbles	3.5 of 5 bubbles	3.0 of 5 bubbles	2.0 of 5 bubbles	1.5 of 5 bubbles

V. PROPOSED ALGORITHM WITH FLOWCHART

- A wide range of modeling techniques have been applied for ticket price/demand prediction.
- To summarize, we can classify the techniques employed so far into three categories. The first and the most commonly implemented set of techniques include simple hypothesis testing and regression techniques. Regression techniques perform well for relatively small size and homogenous dataset.
- However, they are not generally good enough to tackle complex models that make use of big and heterogeneous dataset.
- The second class of techniques used in previous research comprise of various kinds of data mining and machine learning techniques.
    - Even though these approaches have the capability to handle heterogeneous data, acceptable accuracy levels were not achieved by using just a single data mining technique.
- The third and latest approach applied before is ensemble learning.
- Ensemble learning based models have shown better accuracy than other methods.
- It is noteworthy to observe that this computational overhead comes with relatively small datasets as mentioned earlier.
- Moreover, previous models mainly utilize features extracted from static and internal sources. However, as explained earlier, features extracted from external sources such as social media and websites are also a good source of information for better ticket/demand prediction.
- Therefore, future prediction models should incorporate features from both internal and external dynamic sources.

Linear Regression

Linear regression is a statistical algorithm used to predict a Y value, given X features. Using machine learning and ensemble mining, the data sets are

examined to show a relationship. The relationships are then placed along the X/Y axis, with a straight line running through them to predict further relationships.
Linear regression calculates how the X input (factors) relates to the Y output (price).

Linear regression consists of 3 stages majorly:

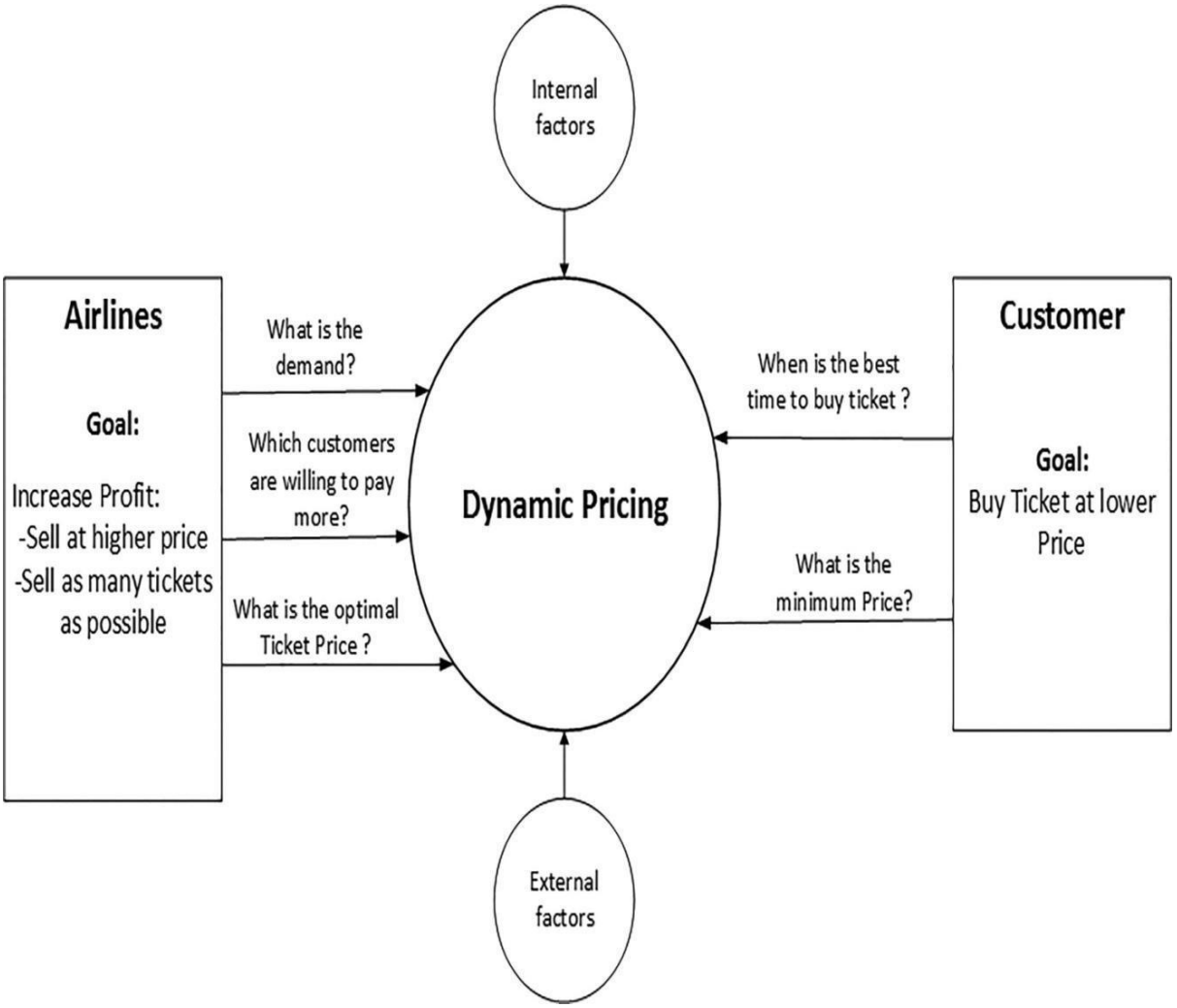
Analysing the correlation and directionality of the data
A scatter plot is drawn to analyse the data and check for directionality and correlation of data.

The first scatter plot drawn indicates a positive relationship between the two variables. The data is fit to run a regression analysis.

The second scatter plot is drawn to have an inverse U-shape; this indicates that a regression line might not be the best way to explain the data, even if a correlation analysis establishes a positive link between the two variables.

The second step of regression analysis is to fit the regression line.

Most often data contains quite a large amount of variability in these cases it is up for decision how to best proceed with the data. This is analysed here in the last step of linear regression.



VI. EXPERIMENTS RESULTS

PROCESS

DATA IMPORT

↓

DATA ANALYSE

↓

DATA CLEANING

↓

DATA SPLITTING

↓

MACHINE LEARNING MODEL

Here firstly we need to check out the essential packages or files inbuilt in the language we are using for coding.

```
# This Python 3 environment comes with many helpful analytics libraries installed
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
```

USE OF LIBRARIES

Numpy is used for doing mathematical operations in the code.

Seaborn is used for plotting the graph with respect to axis

Pandas is built on top of another package named Numpy, which provides support for multi-dimensional arrays.

Then,after installing all the required libraries we can move forward for our dataset files.

```
train_data=pd.read_excel('Data_Train.xlsx')
test_data=pd.read_excel('Test_set.xlsx')
```

Pd.read will read or access our files for the analysis.

dataset

```
[ ] train_data.shape,test_data.shape

    ((10683, 11), (2671, 10))
```

```
train_data.head()
```

.shape will tell us the total no. of rows and columns.

.head will display the top five rows and columns of our data.

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	San francisco	New York	SFO → JFK	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	San francisco	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	San francisco	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	San Francisco	New York	SFO→ NAG → JFK	16:50	21:35	4h 45m	1 stop	No info	13302

```
[ ] train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 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            10682 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      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

Train_data.info() will give us all information about our data,and attributes types or non-null entry.

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          2671 non-null   object
 1   Date_of_Journey  2671 non-null   object
 2   Source           2671 non-null   object
 3   Destination      2671 non-null   object
 4   Route            2671 non-null   object
 5   Dep_Time         2671 non-null   object
 6   Arrival_Time     2671 non-null   object
 7   Duration         2671 non-null   object
 8   Total_Stops      2671 non-null   object
 9   Additional_Info  2671 non-null   object
dtypes: object(10)
memory usage: 208.8+ KB
```

```
[ ] # we can find  missing value in dataset
    train_data.isnull().values.any(),test_data.isnull().values.any()
```

```
(True, False)
```

train_data have missing value.which is-

```
[ ] train_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
dtype: int64
```

Now, we go for data cleaning the code as there can be noisy or outliers which needs to be removed.

Data Cleaning

```
[ ] # Checking for any Duplicate values
    train_data[train_data.duplicated()]
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
683	Jet Airways	1/06/2019	Delhi	Cochin	DEL → NAG → BOM → COK	14:35	04:25 02 Jun	13h 50m	2 stops	No info	13376
1061	Air India	21/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	22:00	19:15 22 May	21h 15m	2 stops	No info	10231
1348	Air India	18/05/2019	Delhi	Cochin	DEL → HYD → BOM → COK	17:15	19:15 19 May	26h	2 stops	No info	12392
1418	Jet Airways	6/06/2019	Delhi	Cochin	DEL → JAI → BOM → COK	05:30	04:25 07 Jun	22h 55m	2 stops	In-flight meal not included	10368
1674	IndiGo	24/03/2019	San francisco	New York	SFO → JFK	18:25	21:20	2h 55m	non-stop	No info	7303
...	...	...	...	...	...	...	...	...	...	...	...
10594	Jet Airways	27/06/2019	Delhi	Cochin	DEL → AMD → BOM → COK	23:05	12:35 28 Jun	13h 30m	2 stops	No info	12819
10616	Jet Airways	1/06/2019	Delhi	Cochin	DEL → JAI → BOM → COK	09:40	12:35 02 Jun	26h 55m	2 stops	No info	13014
10634	Jet Airways	6/06/2019	Delhi	Cochin	DEL → JAI → BOM → COK	09:40	12:35 07 Jun	26h 55m	2 stops	In-flight meal not included	11733
10672	Jet Airways	27/06/2019	Delhi	Cochin	DEL → AMD → BOM → COK	23:05	19:00 28 Jun	19h 55m	2 stops	In-flight meal not included	11150
10673	Jet Airways	27/05/2019	Delhi	Cochin	DEL → AMD → BOM → COK	13:25	04:25 28 May	15h	2 stops	No info	16704

220 rows × 11 columns

```
[ ] # Drop duplicates value
    train_data.drop_duplicates(keep='first',inplace=True)
```

.duplicated will display the duplicates or repeated value in the data.

File Edit View Insert Runtime Tools Help All changes saved

Comment

+ Code + Text

Connect

```
[ ] # Drop duplicates value
    train_data.drop_duplicates(keep='first',inplace=True)
```

```
[ ] train_data["Additional_Info"].value_counts()
```

```
No info                         8182
In-flight meal not included     1926
No check-in baggage included     318
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
Red-eye flight                     1
1 Short layover                    1
2 Long layover                     1
Name: Additional_Info, dtype: int64
```

```
[ ]
```

Convert No Info in No info because both are same

+ Code + Text

```
[ ] train_data["Additional_Info"] = train_data["Additional_Info"].replace({'No Info': 'No info'})
```

Drop_duplicate values will drop those rows or columns that are repeated.

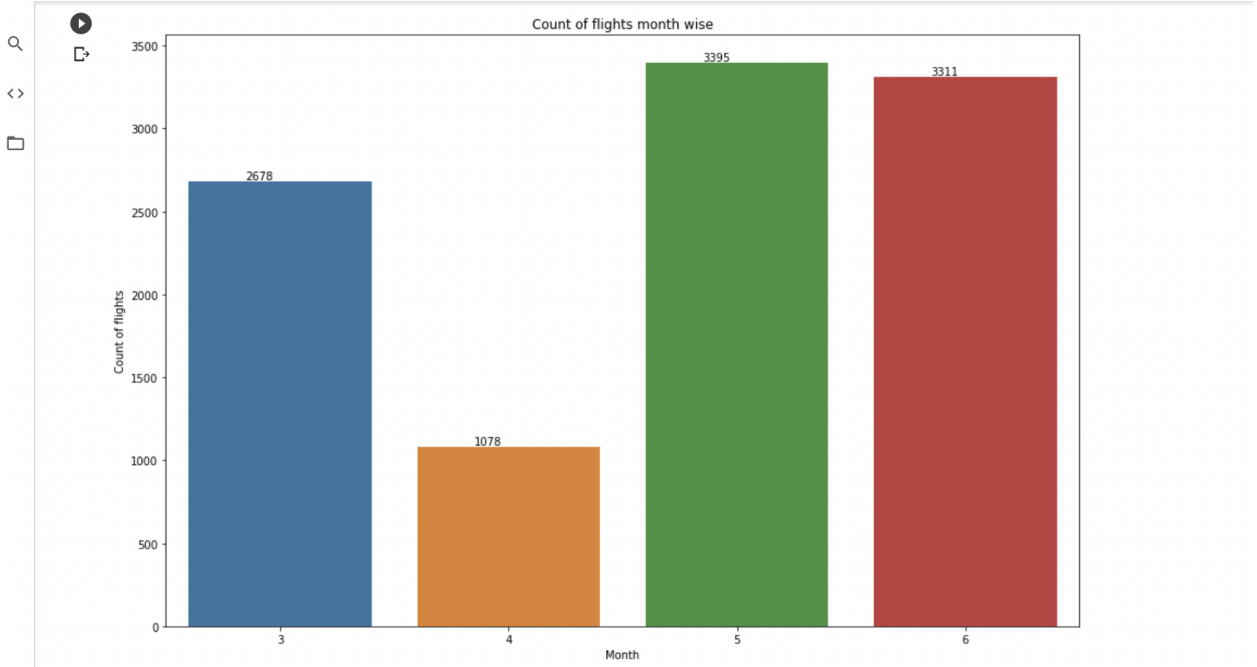
```
[ ] print(test_data.head())
```

```
              Airline         Source  ... Arrival_hour Arrival_min
0        Jet Airways          Delhi  ...            4          25
1             IndiGo        Kolkata  ...           10          20
2        Jet Airways          Delhi  ...           19           0
3  Multiple carriers          Delhi  ...           21           0
4           Air Asia  San Francisco  ...            2          45

[5 rows x 13 columns]
```

```
plt.figure(figsize = (15, 10))
plt.title('Count of flights month wise')
ax=sns.countplot(x = 'Journey_month', data = train_data)
plt.xlabel('Month')
plt.ylabel('Count of flights')
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='bottom',
                color= 'black')
```

Plt.figure is used to plot the figure of a graph.

Plt.title gives the title for the graph.

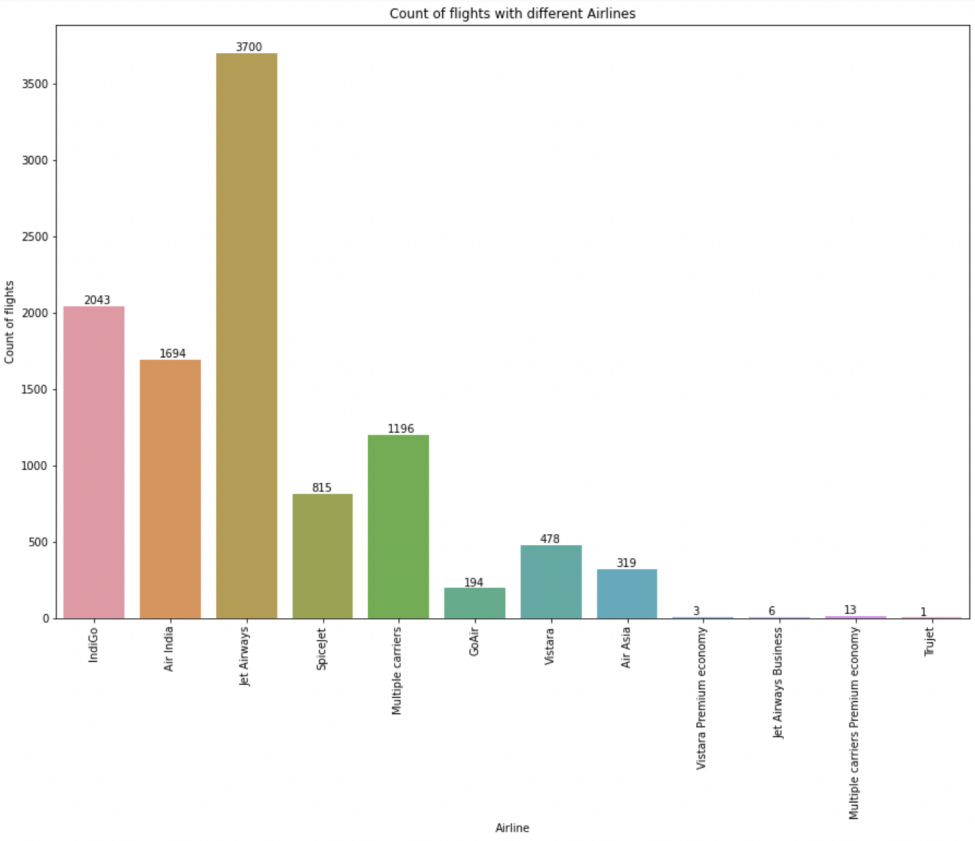



```
[ ] # Total_Stops
    train_data['Total_Stops'].replace(['1 stop', 'non-stop', '2 stops', '3 stops', '4 stops'], [1, 0, 2, 3, 4], inplace=True)
    test_data['Total_Stops'].replace(['1 stop', 'non-stop', '2 stops', '3 stops', '4 stops'], [1, 0, 2, 3, 4], inplace=True)
```

```
[ ] train_data["Airline"].value_counts()
```

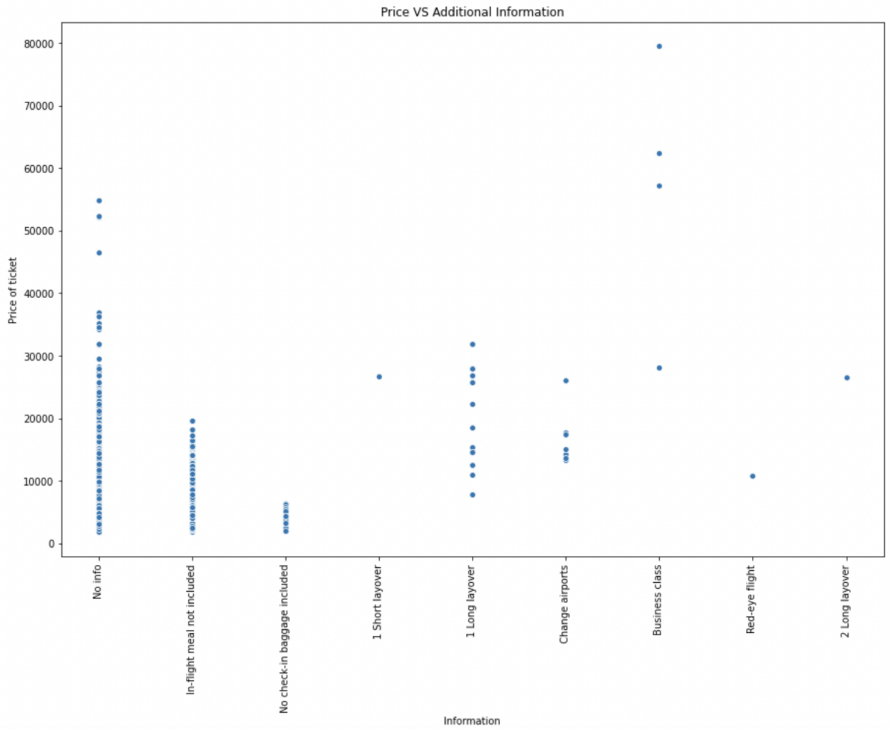
```
Jet Airways                          3700
IndiGo                               2043
Air India                            1694
Multiple carriers                    1196
SpiceJet                              815
Vistara                               478
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

```
plt.figure(figsize = (15, 10))
plt.title('Count of flights with different Airlines')
ax=sns.countplot(x = 'Airline', data =train_data)
plt.xlabel('Airline')
plt.ylabel('Count of flights')
plt.xticks(rotation = 90)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='bottom',
                    color= 'black')
```




```
plt.figure(figsize = (15, 10))
plt.title('Price VS Additional Information')
sns.scatterplot(train_data['Additional_Info'], train_data['Price'],data=train_data)
plt.xticks(rotation = 90)
plt.xlabel('Information')
plt.ylabel('Price of ticket')
```

```
Text(0, 0.5, 'Price of ticket')
```



```
[ ] train_data.head()
```

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
0	IndiGo	San francisco	New York	SFO → JFK	170	0	No info	3897	24	3	22	20	1	10
1	Air India	Kolkata	San francisco	CCU → IXR → BBI → BLR	445	2	No info	7662	1	5	5	50	13	15
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	1140	2	No info	13882	9	6	9	25	4	25
3	IndiGo	Kolkata	San francisco	CCU → NAG → BLR	325	1	No info	6218	12	5	18	5	23	30
4	IndiGo	San Francisco	New York	SFO→ NAG → JFK	285	1	No info	13302	1	3	16	50	21	35

▾ Convert categorical data into numerical

```
[ ] data = train_data.drop(["Price"], axis=1)

    train_categorical_data = data.select_dtypes(exclude=['int64', 'float','int32'])
    train_numerical_data = data.select_dtypes(include=['int64', 'float','int32'])

    test_categorical_data = test_data.select_dtypes(exclude=['int64', 'float','int32','int32'])
    test_numerical_data  = test_data.select_dtypes(include=['int64', 'float','int32'])

[ ] train_categorical_data.head()

[ ] #Label encode Function
    from sklearn.preprocessing import LabelEncoder
    le = LabelEncoder()
    train_categorical_data = train_categorical_data.apply(LabelEncoder().fit_transform)
    test_categorical_data = test_categorical_data.apply(LabelEncoder().fit_transform)

[ ] train_categorical_data.head()
```

Now, joining both categorical and numerical data by using pd.concat

▾ **Concatenate both catagorical and numerical data(JOINING BASICALLY)**

```
[ ] X = pd.concat([train_categorical_data, train_numerical_data], axis=1)
    y=train_data['Price']
    test_set = pd.concat([test_categorical_data, test_numerical_data], axis=1)

[ ] X.head()
```

	Airline	Source	Destination	Route	Additional_Info	Duration	Total_Stops	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
0	3	5	4	127	3	170	0	24	3	22	20	1	10
1	1	2	5	83	3	445	2	1	5	5	50	13	15
2	4	1	0	117	3	1140	2	9	6	9	25	4	25
3	3	2	5	90	3	325	1	12	5	18	5	23	30
4	3	4	4	129	3	285	1	1	3	16	50	21	35

```
[ ] y.head()

0     3897
1     7662
2    13882
3     6218
4    13302
Name: Price, dtype: int64
```

So dataset is ready now!

BUILDING MACHINE LEARNING MODEL FOR REGRESSION

Building Machine Learning Models

```
[ ] from sklearn.preprocessing import StandardScaler
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import mean_squared_error as mse
    from sklearn.metrics import r2_score

    from math import sqrt

    from sklearn.model_selection import GridSearchCV
    from sklearn.model_selection import RandomizedSearchCV


    from sklearn.linear_model import Ridge


    def mean_absolute_percentage_error(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

```
# training testing and splitting the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

```
[ ] print("The size of training input is", X_train.shape)
    print("The size of training output is", y_train.shape)
    print(50 *'*')
    print("The size of testing input is", X_test.shape)
    print("The size of testing output is", y_test.shape)
```

```
The size of training input is (7323, 13)
The size of training output is (7323,)
**************************************************
The size of testing input is (3139, 13)
The size of testing output is (3139,)
```

LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression
lnr = LinearRegression()
lnr.fit(X_train, y_train)
y_pred = lnr.predict(X_test)
LinReg = round(lnr.score(X_test,y_test),2)

print("Train Results for Linear Regression Model:")
print(50 * '-')
print("Root mean squared error: ", sqrt(mse(y_train.values, y_train_pred)))
print("Mean absolute % error: ", round(mean_absolute_percentage_error(y_train.values, y_train_pred)))
print("R-squared: ", r2_score(y_train.values, y_train_pred))
```

```
Train Results for Linear Regression Model:
--------------------------------------------------
Root mean squared error:  3476.3569103824834
Mean absolute % error:  33.0
R-squared:  0.4417992674711877
```

test results

```
[ ] print("Test Results for Linear Regression Model:")
    print(50 * '-')
    print("Root mean squared error: ", sqrt(mse(y_test, y_test_pred)))
    print("Mean absolute % error: ", round(mean_absolute_percentage_error(y_test, y_test_pred)))
    print("R-squared: ", r2_score(y_test, y_test_pred))
```

```
Test Results for Linear Regression Model:
--------------------------------------------------
Root mean squared error:  3371.3131713982816
Mean absolute % error:  33.0
```

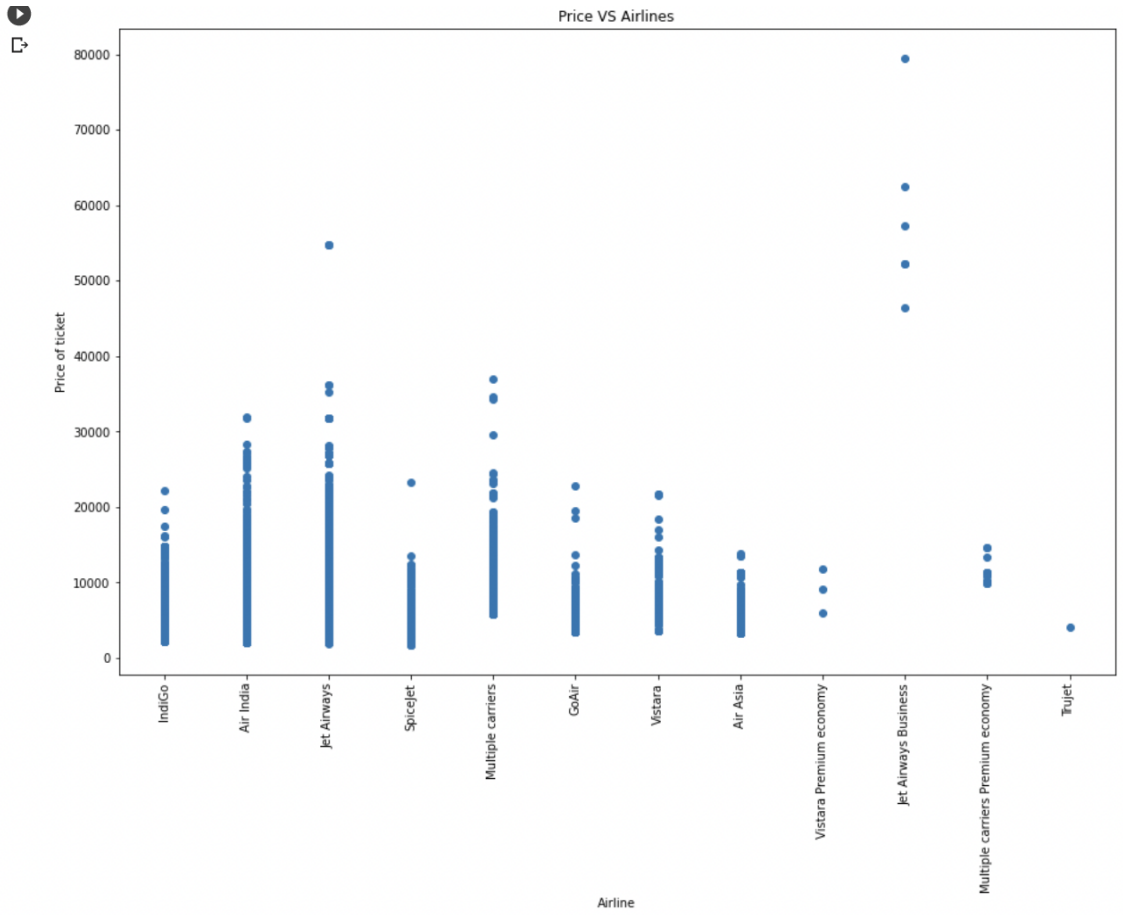
```
import statsmodels.api as sm
ols_model = sm.OLS(y_train,X_train)
ols_results = ols_model.fit()

#Use the fitted model to make predictions on the training and testing data sets:
y_pred_train = ols_results.predict(X_train)
y_pred_test = ols_results.predict(X_test)

#Use the fitted model to make predictions on the training and testing data sets:
y_pred_train = ols_results.predict(X_train)
y_pred_test = ols_results.predict(X_test)
print(ols_results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared (uncentered):                   0.878
Model:                            OLS   Adj. R-squared (uncentered):              0.877
Method:                 Least Squares   F-statistic:                              4030.
Date:                Tue, 01 Jun 2021   Prob (F-statistic):                        0.00
Time:                        13:13:52   Log-Likelihood:                         -70279.
No. Observations:                7323   AIC:                                  1.406e+05
Df Residuals:                    7310   BIC:                                  1.407e+05
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Airline          489.8659     21.129     23.185      0.000     448.448     531.284
Source           444.2560     28.985     15.327      0.000     387.438     501.074
Destination       -4.4436     24.774     -0.179      0.858     -53.007      44.120
Route             -4.8846      1.599     -3.056      0.002      -8.018      -1.751
Additional_Info  844.4831     49.899     16.924      0.000     746.668     942.299
Duration           1.4319      0.125     11.445      0.000       1.187       1.677
Total_Stops     4390.3730     98.521     44.563      0.000    4197.244    4583.502
Journey_day      -55.7229      4.865    -11.454      0.000     -65.260     -46.186
Journey_month   -107.1407     33.727     -3.177      0.001    -173.255     -41.027
Dep_hour          77.5836      7.223     10.741      0.000      63.424      91.743
Dep_min            0.5996      2.192      0.273      0.784      -3.698       4.897
Arrival_hour      19.6980      6.101      3.229      0.001       7.738      31.658
Arrival_min       -0.0811      2.581     -0.031      0.975      -5.142       4.979
==============================================================================
Omnibus:                     5243.847   Durbin-Watson:                   2.008
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           305488.930
Skew:                           2.832   Prob(JB):                         0.00
Kurtosis:                      34.131   Cond. No.                     1.93e+03
```

VII. COMPARATIVE STUDY / RESULTS AND DISCUSSION

The novelty of this project includes the use of linear regression on airline tickets. We focus on predicting the prices of future flights. The predictors we introduced are good predictors for these flights from different Airlines. With this model more information, such as F-stats and R-squared error can be given than just a binary buy/wait advice that HAMLETT gives.

In,PRICE VS AIRLINES graph, our model predicted that on comparing the statistics and analysing the factors by ensemble learning, JET AIRWAYS BUSINESS ticket fare rate is very high in price factor,so that's our results by linear regression.



The ticket rate for Jet Airways Business airline is high.

VIII. CONCLUSION AND FUTURE WORK

In this project, we first presented an overview of dynamic pricing in the airline industry which involves dynamic adjustment of ticket prices based on several internal and external factors. We explained the interaction between customers and airlines in deciding ticket prices dynamically. We then discussed two main previous research areas. Prediction models that are proposed to save money for the customer and those that are designed to increase the revenue of airline companies. Therefore, we classified existing models into customer side and airline side models based on their designed goals. We then summarized and discussed the strengths and weaknesses of existing work. Our analysis showed that this research area has not been greatly explored and that there exist several aspects which need to be properly and thoroughly investigated including: performance issues, dataset issues, usage of dynamic external features such as social media data and search engine query. Therefore, we suggested and discussed a deep learning and social media data-based prediction model as the one of the most promising avenues of research going forward.

IX. REFERENCES

1. Flyr - travel ahead. http://getflyr.com/, August 2014.

2. Infare solutions. http://www.infare.com/about-us/introduction. aspx, August 2014.

3. Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

4. Jan K. Brueckner, Nichola J. Dyer, and Pablo T. Spiller. Fare determination in airline hub-and-spoke networks. The RAND Journal of Economics, 23(3):pp. 309–333, 1992.

5. Oren Etzioni, Rattapoom Tuchinda, Craig A Knoblock, and Alexander Yates. To buy or not to buy: mining airfare data to minimize ticket purchase price. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 119–128. ACM, 2003.

6. Marco Geraci. Linear quantile mixed models: the lqmm package for laplace quantile regression. J Statist Soft, 2014.

7. Marco Geraci and Matteo Bottai. Linear quantile mixed models. Statistics and computing, 24(3):461–479, 2014.

8. William Groves and Maria Gini. A regression model for predicting optimal purchase timing for airline tickets. Technical report, Technical Report 11-025, University of Minnesota, Minneapolis, MN, 2011.

9. Martin Jacobsen. Point process theory and applications. Springer, 2006.

10. Sheryl E. Kimes. The basics of yield management. The Cornell Hotel and Restaurant Administration Quarterly, 30(3):14 – 19, 1989. [11] Sheryl E. Kimes. Yield management: A tool for capacity-considered service firms. Journal of Operations Management, 8(4):348 – 363, 1989. [12] T. Wohlfarth, S. Clemencon, F. Roueff, and X. Casellato. A datamining approach to travel price forecasting. In Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on, volume 1, pages 84–89, Dec 2011.

11. S.B. Kotsiantis, ―Decision trees: a recent overview,‖ Artificial Intelligence Review, vol. 39, no. 4, pp. 261-283, 2013. [6] L. Breiman, ―Random forests,‖ Machine Learning, vol. 45, pp. 5- 32 , 2001.

12. C. Koopmans and R. Lieshout, "Airline cost changes: To what extent are they passed through to the passenger?" Journal of Air Transport Management, vol. 53, pp. 1–11, 2016

13. H. Baik, A. A. Trani, N. Hinze, H. Swingle, S. Ashiabor, and A. Seshadri, "Forecasting model for air taxi, commercial airline, and automobile demand in the united states," Transportation Research Record, vol. 2052, no. 1, pp. 9–20, 2008

14. Dominguez-Menchero, J.Santo, Reviera, ‖optimal purchase timing in airline markets‖ ,2014

15. K. Tziridis, Th. Kalampokas, G.A. Papakotas and K.I. Diamantaras,‖Airfare Prices Prediction Using Machine Learning Techniques‖, EUSIPCO 2017.

Appendix

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
train_data=pd.read_excel('Data_Train.xlsx')
test_data=pd.read_excel('Test_set.xlsx')
train_data.shape,test_data.shape
train_data.head()
train_data.info()
test_data.info()
train_data.isnull().sum()
train_data.dropna(inplace=True)
# Checking for any Duplicate values
train_data[train_data.duplicated()]
# Drop duplicates value
train_data.drop_duplicates(keep='first',inplace=True)
train_data["Additional_Info"].value_counts()
train_data["Additional_Info"] = train_data["Additional_Info"].replace({'No Info': 'No info'})
# Duration convert hours in min.
train_data['Duration']=  train_data['Duration'].str.replace("h", '*60').str.replace(' ','+').str.replace('m','*1').apply(eval)
test_data['Duration']=  test_data['Duration'].str.replace("h", '*60').str.replace(' ','+').str.replace('m','*1').apply(eval)
# Date_of_Journey
```

```
train_data["Journey_day"] = train_data['Date_of_Journey'].str.split('/').str[0].astype(int)
train_data["Journey_month"] = train_data['Date_of_Journey'].str.split('/').str[1].astype(int)
train_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

# Dep_Time
train_data["Dep_hour"] = pd.to_datetime(train_data["Dep_Time"]).dt.hour
train_data["Dep_min"] = pd.to_datetime(train_data["Dep_Time"]).dt.minute
train_data.drop(["Dep_Time"], axis = 1, inplace = True)

# Arrival_Time
train_data["Arrival_hour"] = pd.to_datetime(train_data.Arrival_Time).dt.hour
train_data["Arrival_min"] = pd.to_datetime(train_data.Arrival_Time).dt.minute
train_data.drop(["Arrival_Time"], axis = 1, inplace = True)
# Date_of_Journey
test_data["Journey_day"] = test_data['Date_of_Journey'].str.split('/').str[0].astype(int)
test_data["Journey_month"] = test_data['Date_of_Journey'].str.split('/').str[1].astype(int)
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

# Dep_Time
test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour
test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute
test_data.drop(["Dep_Time"], axis = 1, inplace = True)

# Arrival_Time
test_data["Arrival_hour"] = pd.to_datetime(test_data.Arrival_Time).dt.hour
test_data["Arrival_min"] = pd.to_datetime(test_data.Arrival_Time).dt.minute
```

```
test_data.drop(["Arrival_Time"], axis = 1, inplace = True)


print(test_data.head())

plt.figure(figsize = (15, 10))

plt.title('Count of flights month wise')

ax=sns.countplot(x = 'Journey_month', data = train_data)

plt.xlabel('Month')

plt.ylabel('Count of flights')

for p in ax.patches:

  ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='bottom',

            color= 'black')

# Total_Stops

train_data['Total_Stops'].replace(['1 stop', 'non-stop', '2 stops', '3 stops', '4 stops'], [1, 0, 2, 3, 4], inplace=True)

test_data['Total_Stops'].replace(['1 stop', 'non-stop', '2 stops', '3 stops', '4 stops'], [1, 0, 2, 3, 4], inplace=True)

train_data["Airline"].value_counts()

plt.figure(figsize = (15, 10))

plt.title('Count of flights with different Airlines')

ax=sns.countplot(x = 'Airline', data =train_data)

plt.xlabel('Airline')

plt.ylabel('Count of flights')

plt.xticks(rotation = 90)

for p in ax.patches:

  ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va='bottom',

            color= 'black')

plt.figure(figsize = (15, 10))
```

```
plt.title('Price VS Airlines')
plt.scatter(train_data['Airline'], train_data['Price'])
plt.xticks(rotation = 90)
plt.xlabel('Airline')
plt.ylabel('Price of ticket')
plt.xticks(rotation = 90)

train_data["Airline"].replace({'Multiple carriers Premium economy':'Other',
                               'Jet Airways Business':'Other',
                               'Vistara Premium economy':'Other',
                               'Trujet':'Other'
                              },
                              inplace=True)

test_data["Airline"].replace({'Multiple carriers Premium economy':'Other',
                               'Jet Airways Business':'Other',
                               'Vistara Premium economy':'Other',
                               'Trujet':'Other'
                              },
                              inplace=True)

plt.figure(figsize = (15, 10))
plt.title('Price VS Additional Information')
sns.scatterplot(train_data['Additional_Info'], train_data['Price'],data=train_data)
plt.xticks(rotation = 90)
plt.xlabel('Information')
```

```
plt.ylabel('Price of ticket')
train_data["Additional_Info"].value_counts()
# Additional_Info
train_data["Additional_Info"].replace({'Change airports':'Other',
                                       'Business class':'Other',
                                       '1 Short layover':'Other',
                                       'Red-eye flight':'Other',
                                       '2 Long layover':'Other',
                                      },
                                inplace=True)
test_data["Additional_Info"].replace({'Change airports':'Other',
                                       'Business class':'Other',
                                       '1 Short layover':'Other',
                                       'Red-eye flight':'Other',
                                       '2 Long layover':'Other',
                                      },
                                inplace=True)
train_data.head()
data = train_data.drop(["Price"], axis=1)
train_categorical_data = data.select_dtypes(exclude=['int64', 'float','int32'])
train_numerical_data = data.select_dtypes(include=['int64', 'float','int32'])

test_categorical_data = test_data.select_dtypes(exclude=['int64', 'float','int32','int32'])
test_numerical_data  = test_data.select_dtypes(include=['int64', 'float','int32'])
train_categorical_data.head()
#Label encode Function
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train_categorical_data = train_categorical_data.apply(LabelEncoder().fit_transform)
test_categorical_data = test_categorical_data.apply(LabelEncoder().fit_transform)
train_categorical_data.head()
X = pd.concat([train_categorical_data, train_numerical_data], axis=1)
y=train_data['Price']
test_set = pd.concat([test_categorical_data, test_numerical_data], axis=1)
X.head()
y.head()
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score

from math import sqrt

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV



from sklearn.linear_model import Ridge



def mean_absolute_percentage_error(y_true, y_pred):
   y_true, y_pred = np.array(y_true), np.array(y_pred)
```

```
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
# training testing and splitting the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
print("The size of training input is", X_train.shape)
print("The size of training output is", y_train.shape)
print(50 *'*')
print("The size of testing input is", X_test.shape)
print("The size of testing output is", y_test.shape)
from sklearn.linear_model import LinearRegression
lnr = LinearRegression()
lnr.fit(X_train, y_train)
y_pred = lnr.predict(X_test)
LinReg = round(lnr.score(X_test,y_test),2)

print("Train Results for Linear Regression Model:")
print(50 * '-')
print("Root mean squared error: ", sqrt(mse(y_train.values, y_train_pred)))
print("Mean absolute % error: ", round(mean_absolute_percentage_error(y_train.values,
y_train_pred)))
print("R-squared: ", r2_score(y_train.values, y_train_pred))
print("Test Results for Linear Regression Model:")
print(50 * '-')
print("Root mean squared error: ", sqrt(mse(y_test, y_test_pred)))
print("Mean absolute % error: ", round(mean_absolute_percentage_error(y_test,
y_test_pred)))
print("R-squared: ", r2_score(y_test, y_test_pred))
```

```
import statsmodels.api as sm
ols_model = sm.OLS(y_train,X_train)
ols_results = ols_model.fit()

#Use the fitted model to make predictions on the training and testing data sets:
y_pred_train = ols_results.predict(X_train)
y_pred_test = ols_results.predict(X_test)

#Use the fitted model to make predictions on the training and testing data sets:
y_pred_train = ols_results.predict(X_train)
y_pred_test = ols_results.predict(X_test)
print(ols_results.summary())
```