A Report

On

FLIGHT FARE PREDICDTION USING MULTIPLE LINEAR REGRESSION MODEL

Research and Business Analytics
Batch 2020- 2022

FACULTY MENTOR:

Prof. Sareeta Mugde

SUBMITTED BY

GROUP 10

Jidnyasa Heda (20)

Ruchika Dhamane (41)

Shivani Bihade (49)

Vishal Malpure (58)

Contents

PROJECT INTRODUCTION

Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travelers saying that flight ticket prices are so unpredictable. Hence, we have built a model to predict the prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities. Size of training set: 10,683 records.

FEATURES OF THE DATASET

Independent Variables in the dataset:

1. ID: Contiguous sample number

2. Airline: The name of the airline

3. Date_of_Journey: The date of the journey

4. Source: The source from which the service begins.

5. Destination: The destination where the service ends

6. Dep_Time: The time when the journey starts from the source.

7. Arrival_Time: Time of arrival at the destination.

8. Duration: Total duration of the flight.

9. Total Stops: Total stops between the source and destination.

10. Additional Info: Additional information about the flight

Target Dependent variable:

Price: The price of the ticket

OBJECTIVE

The flight ticket price in India is based on demand and supply model with few restrictions on pricing from regulatory bodies. It is often perceived as unpredictable and, recent dynamic pricing scheme added to the confusion. The objective is to create a machine learning model for predicting the flight price, based on historical data, which can be used for reference price for customers as well as airline service providers.

EXPLORATORY DATA ANALYSIS

Preprocessing of the data has been done on Python.

Flight Price Prediction

In [408]:
```
#Import packages like pandas,numpy,matplotlib,seaborn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [409]:
```
#Importing training data from excel stored in the local machine
df_train= pd.read_excel("Data_Train.xlsx")
```

In [410]:
```
#Displaying first 5 rows of training data
df_train.head()
```

Out[410]:

	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

In [411]:
```
#Print info
#Training Data consists of 10 columns of object/string type and 1 column i.e price of integer type
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
Airline            10683 non-null object
Date_of_Journey    10683 non-null object
Source             10683 non-null object
Destination        10683 non-null object
```

In [412]:
```
#As there is only one numerical feature we got statistical information only for price column
df_train.describe()
```

Out[412]:

	Price
count	10683.000000
mean	9087.064121
std	4611.359167
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

In [413]:
```
#shape of data
#10683 rows and 11 columns
df_train.shape
```

Out[413]: (10683, 11)

In [414]:
```
#number of rows in the training data
len(df_train.index)
```

Out[414]: 10683

In [415]:
```
#Finding sum of null values in all columns
df_train.isnull().sum()
```

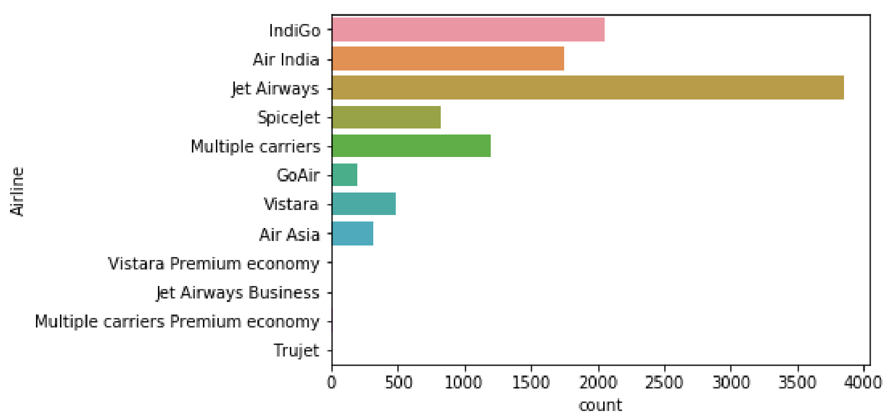
Out[415]:
```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              1
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        1
Additional_Info    0
Price              0
dtype: int64
```

In [416]:
```
#Check if all the entries within a row are null so that we can delete the entire row
df_train.isnull().all(axis=1).sum()
```

Out[416]:
```
0
```

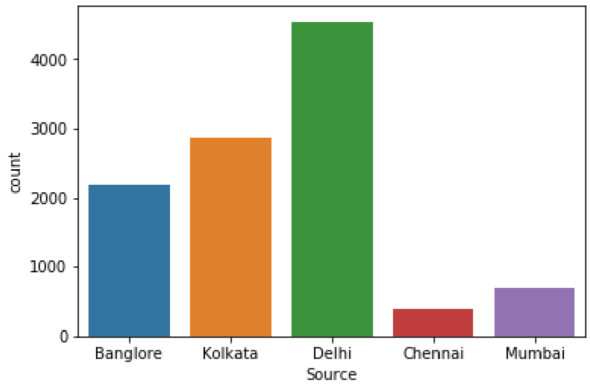
In [417]:
```
# Bar Chart for Flight Categories
sns.countplot(y=df_train["Airline"])
```

Out[417]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1de47f0d828>
```



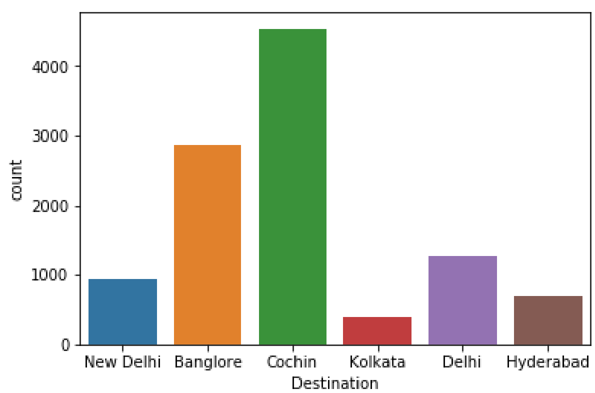
In [418]:
```
#Bar Chart for Source
sns.countplot(x=df_train["Source"])
```

Out[418]: <matplotlib.axes._subplots.AxesSubplot at 0x1de59d11b38>



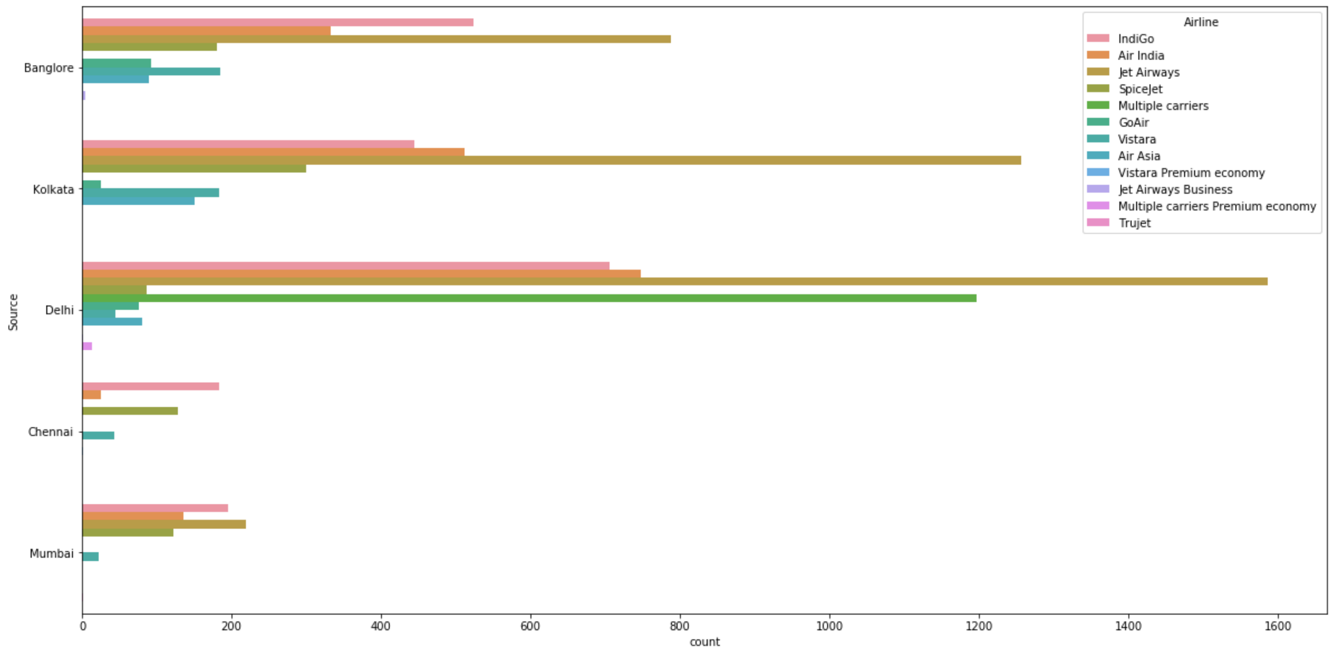
In [419]:
```
#Bar Chart for Destination
sns.countplot(x=df_train["Destination"])
```

Out[419]: <matplotlib.axes._subplots.AxesSubplot at 0x1de5759a208>



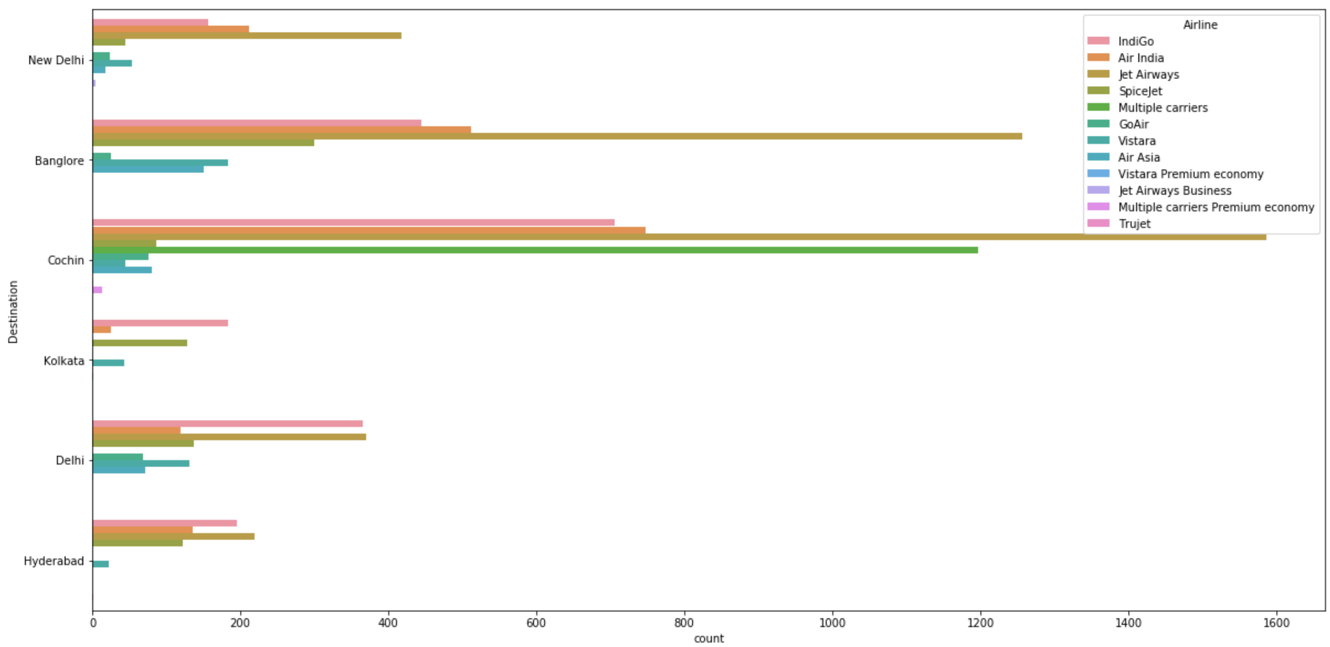
In [420]:
```
#Source wise Flight Categories
fig, ax = plt.subplots(figsize=(20,10))
sns.countplot(y=df_train["Source"],hue=df_train["Airline"],ax=ax)
```

Out[420]: <matplotlib.axes._subplots.AxesSubplot at 0x1de5ba7ba90>



In [421]:
```
#Destination wise Flight Categories
fig, ax = plt.subplots(figsize=(20,10))
sns.countplot(y=df_train["Destination"],hue=df_train["Airline"],ax=ax)
```

Out[421]: <matplotlib.axes._subplots.AxesSubplot at 0x1de59ffc1d0>



```
In [422]: #Value Count of each Flight Category
          df_train["Airline"].value_counts()

Out[422]: Jet Airways                          3849
          IndiGo                               2053
          Air India                            1752
          Multiple carriers                    1196
          SpiceJet                              818
          Vistara                               479
          Air Asia                              319
          GoAir                                 194
          Multiple carriers Premium economy      13
          Jet Airways Business                    6
          Vistara Premium economy                 3
          Trujet                                  1
          Name: Airline, dtype: int64
```

```
In [423]: #Value Count of number of Flights from each Source
          df_train["Source"].value_counts()

Out[423]: Delhi       4537
          Kolkata     2871
          Banglore    2197
          Mumbai       697
          Chennai      381
          Name: Source, dtype: int64
```

```
In [424]: #Value Count of number of Flights to each Destination
          df_train["Destination"].value_counts()

Out[424]: Cochin       4537
          Banglore     2871
          Delhi        1265
          New Delhi     932
          Hyderabad     697
          Kolkata       381
          Name: Destination, dtype: int64
```

```
In [426]: #Value Count of Total Stops
          #There are 5625 flights having 1 stop in between while reaching from source to destination
          df_train["Total_Stops"].value_counts()

Out[426]: 1 stop      5625
          non-stop    3491
          2 stops     1520
          3 stops       45
          4 stops        1
          Name: Total_Stops, dtype: int64
```

```
In [427]: #Value Count of Additional_Info Column
          df_train["Additional_Info"].value_counts()

Out[427]: No info                         8345
          In-flight meal not included     1982
          No check-in baggage included     320
          1 Long layover                    19
          Change airports                    7
          Business class                     4
          No Info                            3
          1 Short layover                    1
          2 Long layover                     1
          Red-eye flight                     1
          Name: Additional_Info, dtype: int64
```

As we can see that 8345/10683 = 78.11% data has no information,hence we can drop this Additional_Info Column

In [428]:
```
#Dropping Additional_Info
df_train.drop(["Additional_Info"],axis=1,inplace=True)
```

In [429]:
```
#Displaying first 5 rows of the dataframe
df_train.head()
```

Out[429]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	13302

Data Preprocessing

In [430]:
```
#From the Date_of Journey Column, create a new column Journey_Day which is an integer
df_train["Journey_Day"]=pd.to_datetime(df_train["Date_of_Journey"],format="%d/%m/%Y").dt.day
```

In [431]:
```
#From the Date_of Journey Column, create a new column Journey_Montg which is an integer
df_train["Journey_Month"]=pd.to_datetime(df_train["Date_of_Journey"],format="%d/%m/%Y").dt.month
```

We will not extract the year from Date_of_Journey Column because all the 10683 rows belong to the year 2019. Now we can drop the Date_of_journey column

In [433]:
```
#Dropping Date_of_Journey Column
df_train.drop(["Date_of_Journey"],axis=1,inplace=True)
```

In [434]:
```
#From the Dep_Time Column, create a new column Dep_Hour which is an integer
df_train["Dep_Hour"]=pd.to_datetime(df_train["Dep_Time"]).dt.hour
```

In [435]:
```
#From the Dep_Time Column, create a new column Dep_Minute which is an integer
df_train["Dep_Minute"]=pd.to_datetime(df_train["Dep_Time"]).dt.minute
```

In [436]:
```
#Dropping Dep_Time Column
df_train.drop(["Dep_Time"],axis=1,inplace=True)
```

In [437]:
```
# As we are considering Duration for the processing hence we can drop Arrival_time column because Duration=Arrival_Time-Dep_Time
df_train.drop(["Arrival_Time"],axis=1,inplace=True)
```

In [438]:
```
#As we have Total number of stops,hence we can remove the redundant information about the route
df_train.drop(["Route"],axis=1,inplace=True)
```

In [439]:
```
# Assigning and converting Duration column into list
duration = list(df_train["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))   # Extracts only minutes from duration
```

In [440]:
```
# Adding duration_hours and duration_mins list to train_data dataframe
df_train["Duration_hours"] = duration_hours
df_train["Duration_mins"] = duration_mins
```

In [441]:
```
#Dropping Duration Column
df_train.drop(["Duration"],axis=1,inplace=True)
```

Data Processing for Categorical Columns

As Airline Column consists of Nominal Data hence performing One hot Encoding for the same ¶

In [442]:
```
#Creating Dummy variables for Airline Column
airline=pd.get_dummies(df_train["Airline"],drop_first=True)
```

In [443]:
```
#Concatinating the inital dataframe with the airline dummy variables
df_train=pd.concat([df_train,airline],axis=1)
```

In [444]:
```
df_train.head()
```

Out[444]:

	Airline	Source	Destination	Total_Stops	Price	Journey_Day	Journey_Month	Dep_Hour	Dep_Minute	Duration_hours	...	GoAir	IndiGo	Jet Airways	Air Busi
0	IndiGo	Banglore	New Delhi	non-stop	3897	24	3	22	20	2	...	0	1	0	
1	Air India	Kolkata	Banglore	2 stops	7662	1	5	5	50	7	...	0	0	0	
2	Jet Airways	Delhi	Cochin	2 stops	13882	9	6	9	25	19	...	0	0	1	
3	IndiGo	Kolkata	Banglore	1 stop	6218	12	5	18	5	5	...	0	1	0	
4	IndiGo	Banglore	New Delhi	1 stop	13302	1	3	16	50	4	...	0	1	0	

5 rows × 22 columns

In [445]:
```
#Creating Dummy variables for Source Column
source_dummies=pd.get_dummies(df_train["Source"],prefix='Source',drop_first=True)
```

In [446]:
```
#Concatinating the inital dataframe with the source dummy variables
df_train=pd.concat([df_train,source_dummies],axis=1)
```

In [447]:
```
#Creating Dummy variables for Destination Column and concatinating the inital dataframe with the destination dummy variables
destination_dummies=pd.get_dummies(df_train["Destination"],prefix='Destination',drop_first=True)
df_train=pd.concat([df_train,destination_dummies],axis=1)
```

In [448]:
```
#As we have created the dummy varaibles for Airline,Source & Destination so we can drop the actual Airline,Source & Destination C
df_train.drop(["Airline","Source","Destination"],axis=1,inplace=True)
```

In [449]:
```
#Value Count of Total_Stops Column
df_train["Total_Stops"].value_counts()
```

Out[449]:
```
1 stop       5625
non-stop     3491
2 stops      1520
3 stops        45
4 stops         1
Name: Total_Stops, dtype: int64
```

As Total_Stops Column consists of ordinal data hence we are performing Label Encoding for the Total_Stops Column

In [450]:
```
#Replacing this Object values with the string values
df_train.replace(to_replace=['1 stop','non-stop','2 stops','3 stops','4 stops'], value=["1","0","2","3","4"],inplace=True)
```

In [451]:
```
df_train.head()
```

Out[451]:

	Total_Stops	Price	Journey_Day	Journey_Month	Dep_Hour	Dep_Minute	Duration_hours	Duration_mins	Air India	GoAir	...	Vistara Premium economy	Source_Chennai	Sou
0	0	3897	24	3	22	20	2	50	0	0	...	0	0	
1	2	7662	1	5	5	50	7	25	1	0	...	0	0	
2	2	13882	9	6	9	25	19	0	0	0	...	0	0	
3	1	6218	12	5	18	5	5	25	0	0	...	0	0	
4	1	13302	1	3	16	50	4	45	0	0	...	0	0	

5 rows × 28 columns

In [452]:
```
# As there is one NA value in the Total_Stops column replacing it with the mode of the column i.e 1
df_train['Total_Stops'].fillna(1,inplace=True)
```

In [453]:
```
#Converting the object type Total_Stops column to integer type Total_Stops Column
df_train['Total_Stops']=df_train['Total_Stops'].astype(int)
```

In [454]:
```
#All the columns are of the integer type
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 28 columns):
Total_Stops                          10683 non-null int32
Price                                10683 non-null int64
Journey_Day                          10683 non-null int64
Journey_Month                        10683 non-null int64
Dep_Hour                             10683 non-null int64
Dep_Minute                           10683 non-null int64
Duration_hours                       10683 non-null int64
Duration_mins                        10683 non-null int64
Air India                            10683 non-null uint8
GoAir                                10683 non-null uint8
IndiGo                               10683 non-null uint8
Jet Airways                          10683 non-null uint8
Jet Airways Business                 10683 non-null uint8
Multiple carriers                    10683 non-null uint8
Multiple carriers Premium economy    10683 non-null uint8
SpiceJet                             10683 non-null uint8
Trujet                               10683 non-null uint8
Vistara                              10683 non-null uint8
Vistara Premium economy              10683 non-null uint8
Source_Chennai                       10683 non-null uint8
Source_Delhi                         10683 non-null uint8
Source_Kolkata                       10683 non-null uint8
Source_Mumbai                        10683 non-null uint8
Destination_Cochin                   10683 non-null uint8
Destination_Delhi                    10683 non-null uint8
Destination_Hyderabad                10683 non-null uint8
Destination_Kolkata                  10683 non-null uint8
Destination_New Delhi                10683 non-null uint8
dtypes: int32(1), int64(7), uint8(20)
memory usage: 834.7 KB
```

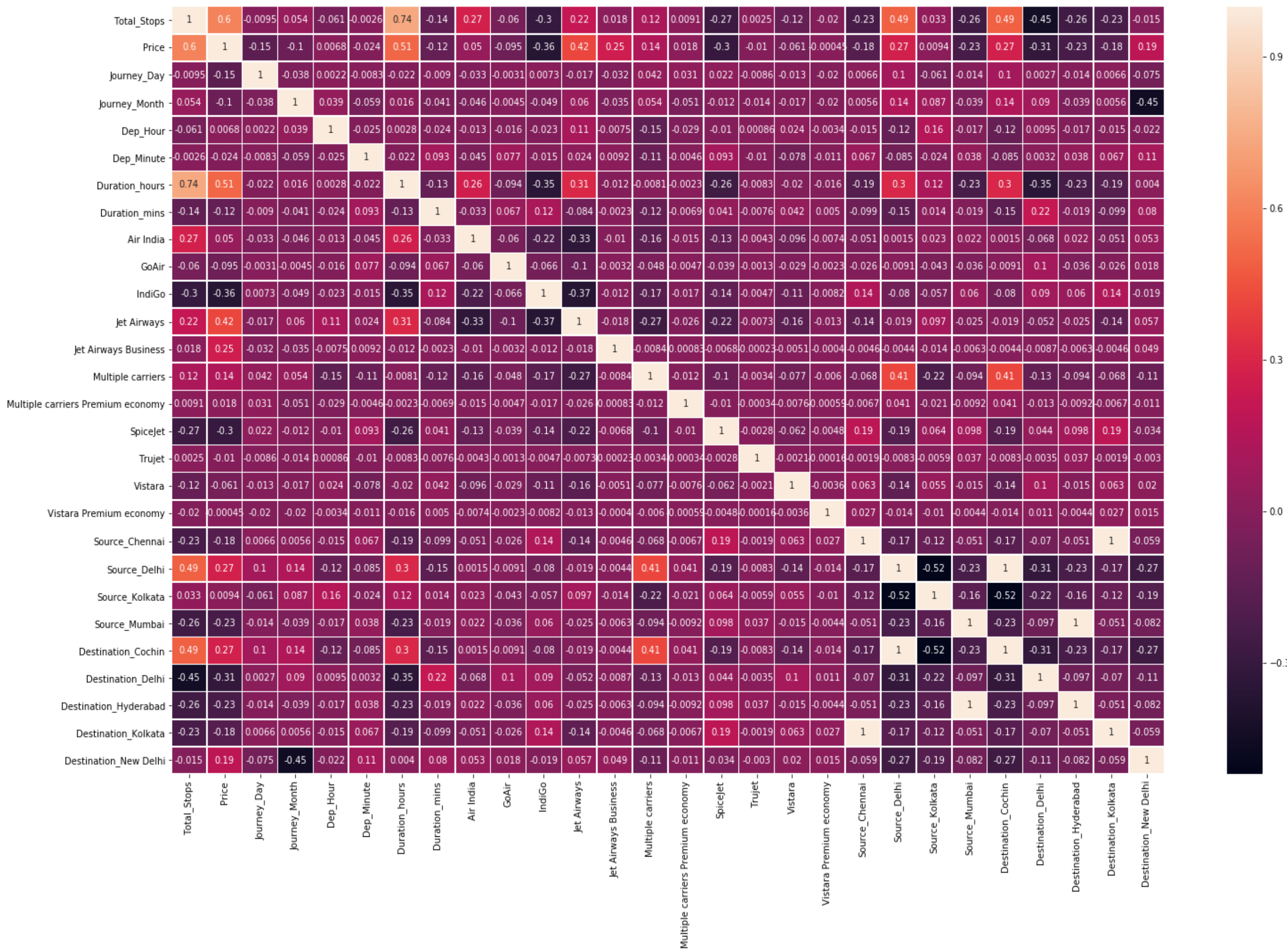
In [455]:
```
# Now the processed data consists of 28 columns
df_train.shape
```

Out[455]: (10683, 28)

In [456]:
```
#Correlation Matrix in the form of heatmap
fig,ax = plt.subplots(figsize=(25,15))
corr=df_train.corr()
sns.heatmap(corr,ax=ax,annot=True,linewidth=.5)
```

Out[456]: <matplotlib.axes._subplots.AxesSubplot at 0x1de623d47b8>

Correlation Matrix:



In [459]:
```
#Displaying the first five rows of the processed data
df_train.head()
```

Out[459]:

	Air India	GoAir	IndiGo	Jet Airways	Jet Airways Business	Multiple carriers	Multiple carriers Premium economy	SpiceJet	Trujet	Vistara	...	Destination_Kolkata	Destination_New Delhi	Journey_Day	Journey_Mo
0	0	0	1	0	0	0	0	0	0	0	...	0	1	24	
1	1	0	0	0	0	0	0	0	0	0	...	0	0	1	
2	0	0	0	1	0	0	0	0	0	0	...	0	0	9	
3	0	0	1	0	0	0	0	0	0	0	...	0	0	12	
4	0	0	1	0	0	0	0	0	0	0	...	0	1	1	

5 rows × 28 columns

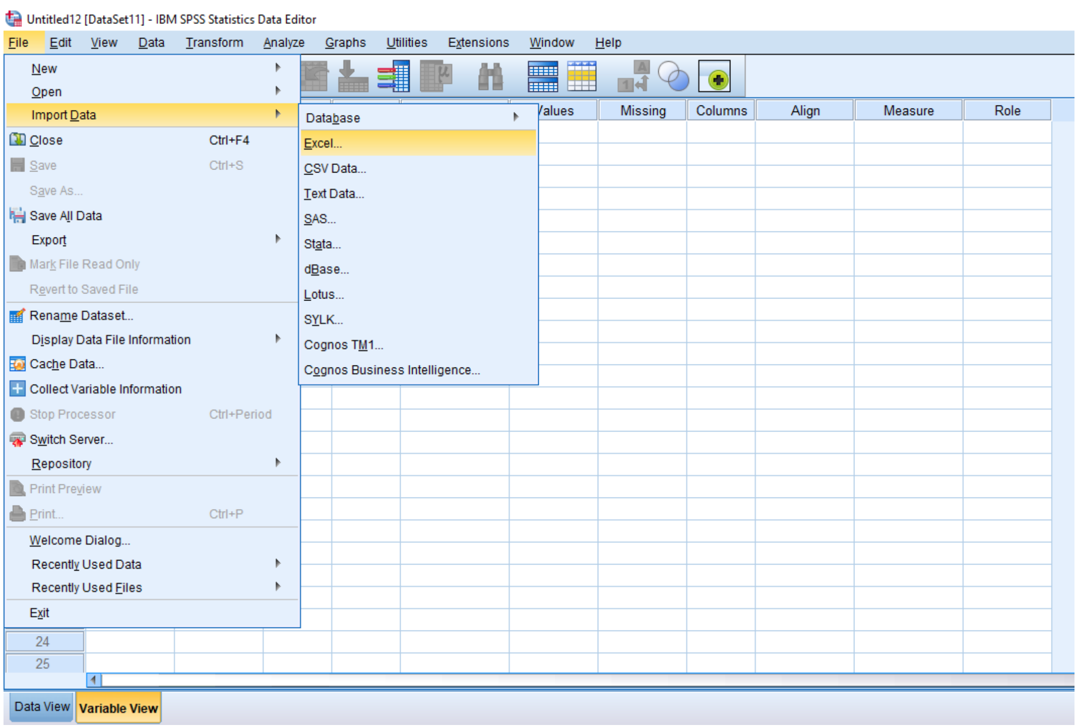
In [460]:
```
# Exporting this processed data to a csv file using to_csv
df_train.to_csv("Final_Data.csv",index=False)
```

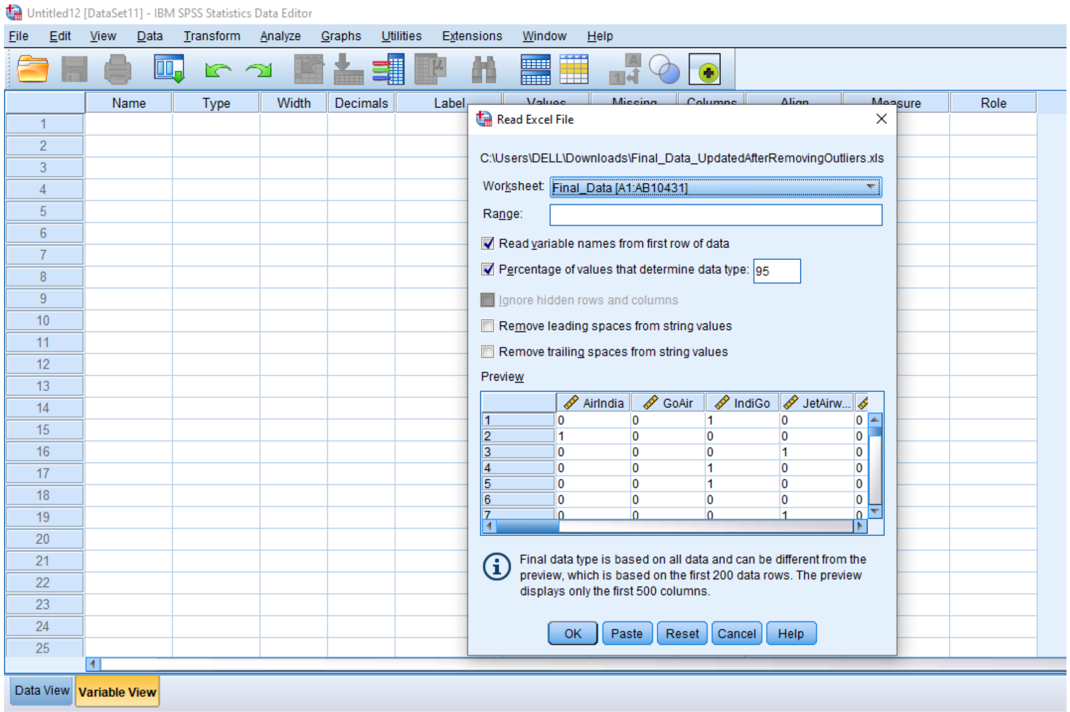
```
In [457]: #List of 28 columns available
          df_train.columns

Out[457]: Index(['Total_Stops', 'Price', 'Journey_Day', 'Journey_Month', 'Dep_Hour',
                 'Dep_Minute', 'Duration_hours', 'Duration_mins', 'Air India', 'GoAir',
                 'IndiGo', 'Jet Airways', 'Jet Airways Business', 'Multiple carriers',
                 'Multiple carriers Premium economy', 'SpiceJet', 'Trujet', 'Vistara',
                 'Vistara Premium economy', 'Source_Chennai', 'Source_Delhi',
                 'Source_Kolkata', 'Source_Mumbai', 'Destination_Cochin',
                 'Destination_Delhi', 'Destination_Hyderabad', 'Destination_Kolkata',
                 'Destination_New Delhi'],
                dtype='object')
```

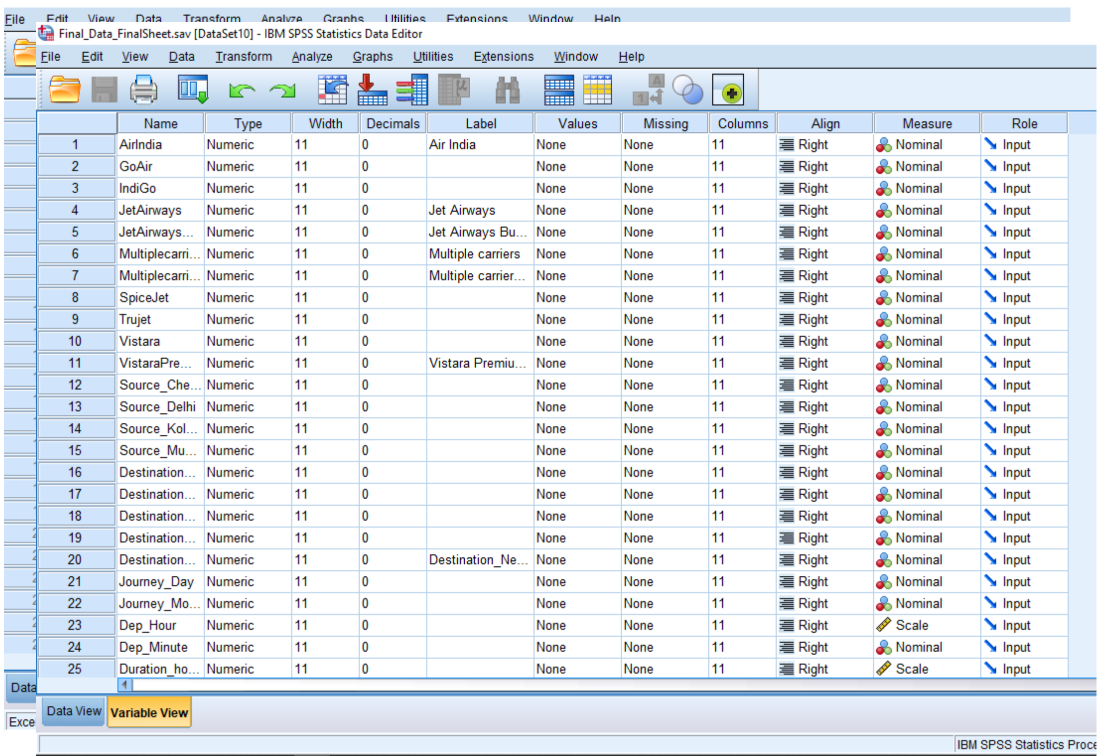
```
In [458]: #Rearranging the features as per Independent and Dependent Features
          df_train=df_train[['Air India', 'GoAir',
                 'IndiGo', 'Jet Airways', 'Jet Airways Business', 'Multiple carriers',
                 'Multiple carriers Premium economy', 'SpiceJet', 'Trujet', 'Vistara',
                 'Vistara Premium economy', 'Source_Chennai', 'Source_Delhi',
                 'Source_Kolkata', 'Source_Mumbai', 'Destination_Cochin',
                 'Destination_Delhi', 'Destination_Hyderabad', 'Destination_Kolkata',
                 'Destination_New Delhi','Journey_Day', 'Journey_Month', 'Dep_Hour',
                 'Dep_Minute', 'Duration_hours', 'Duration_mins','Total_Stops','Price']]
```

MODEL BUILDING AND RESULTS

Step 1: Import the excel file.

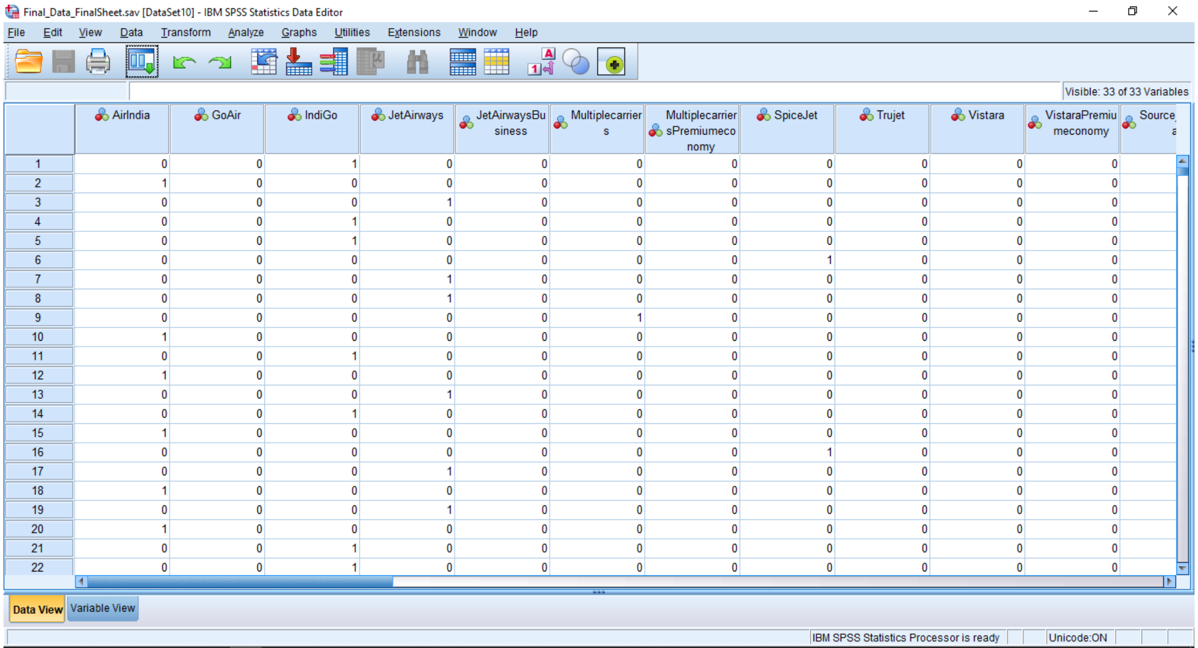




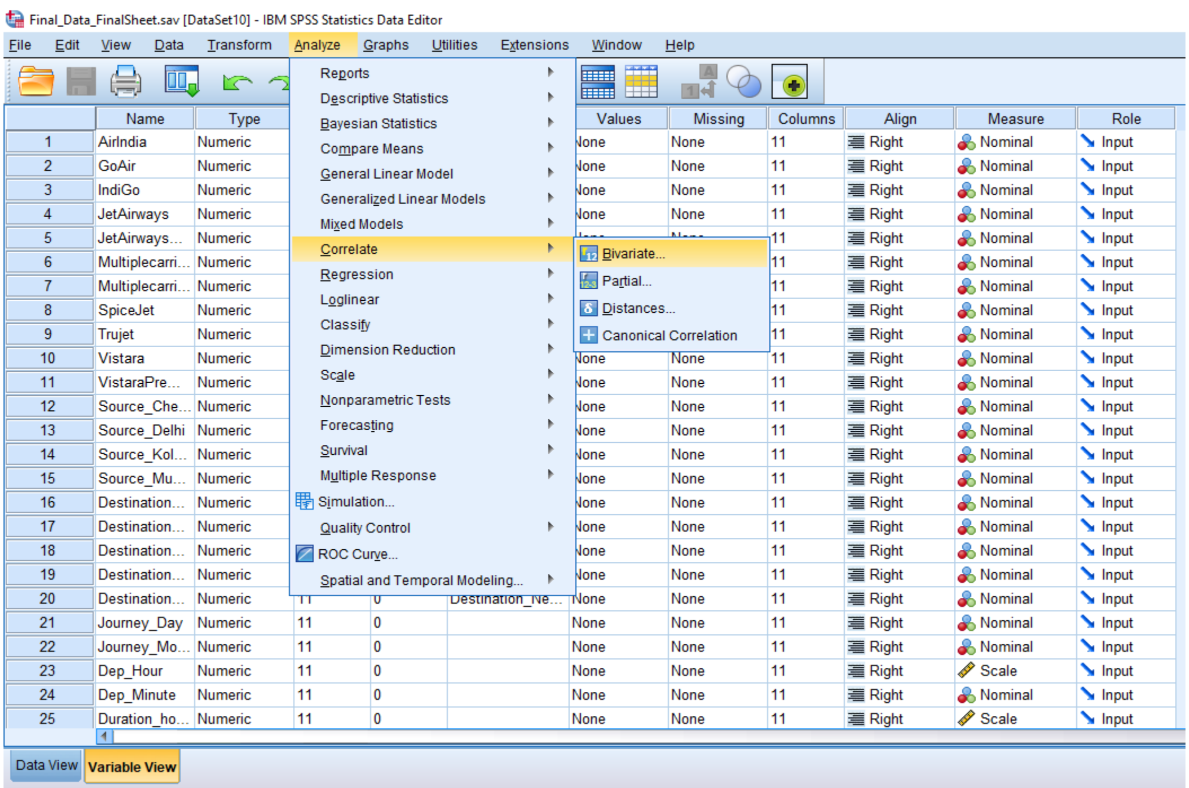


	Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure	Role
1	AirIndia	Numeric	11	0	Air India	None	None	11	Right	Nominal	Input
2	GoAir	Numeric	11	0		None	None	11	Right	Nominal	Input
3	IndiGo	Numeric	11	0		None	None	11	Right	Nominal	Input
4	JetAirways	Numeric	11	0	Jet Airways	None	None	11	Right	Nominal	Input
5	JetAirways...	Numeric	11	0	Jet Airways Bu...	None	None	11	Right	Nominal	Input
6	Multiplecarri...	Numeric	11	0	Multiple carriers	None	None	11	Right	Nominal	Input
7	Multiplecarri...	Numeric	11	0	Multiple carrier...	None	None	11	Right	Nominal	Input
8	SpiceJet	Numeric	11	0		None	None	11	Right	Nominal	Input
9	Trujet	Numeric	11	0		None	None	11	Right	Nominal	Input
10	Vistara	Numeric	11	0		None	None	11	Right	Nominal	Input
11	VistaraPre...	Numeric	11	0	Vistara Premiu...	None	None	11	Right	Nominal	Input
12	Source_Che...	Numeric	11	0		None	None	11	Right	Nominal	Input
13	Source_Delhi	Numeric	11	0		None	None	11	Right	Nominal	Input
14	Source_Kol...	Numeric	11	0		None	None	11	Right	Nominal	Input
15	Source_Mu...	Numeric	11	0		None	None	11	Right	Nominal	Input
16	Destination...	Numeric	11	0		None	None	11	Right	Nominal	Input
17	Destination...	Numeric	11	0		None	None	11	Right	Nominal	Input
18	Destination...	Numeric	11	0		None	None	11	Right	Nominal	Input
19	Destination...	Numeric	11	0		None	None	11	Right	Nominal	Input
20	Destination...	Numeric	11	0	Destination_Ne...	None	None	11	Right	Nominal	Input
21	Journey_Day	Numeric	11	0		None	None	11	Right	Nominal	Input
22	Journey_Mo...	Numeric	11	0		None	None	11	Right	Nominal	Input
23	Dep_Hour	Numeric	11	0		None	None	11	Right	Scale	Input
24	Dep_Minute	Numeric	11	0		None	None	11	Right	Nominal	Input
25	Duration_ho...	Numeric	11	0		None	None	11	Right	Scale	Input

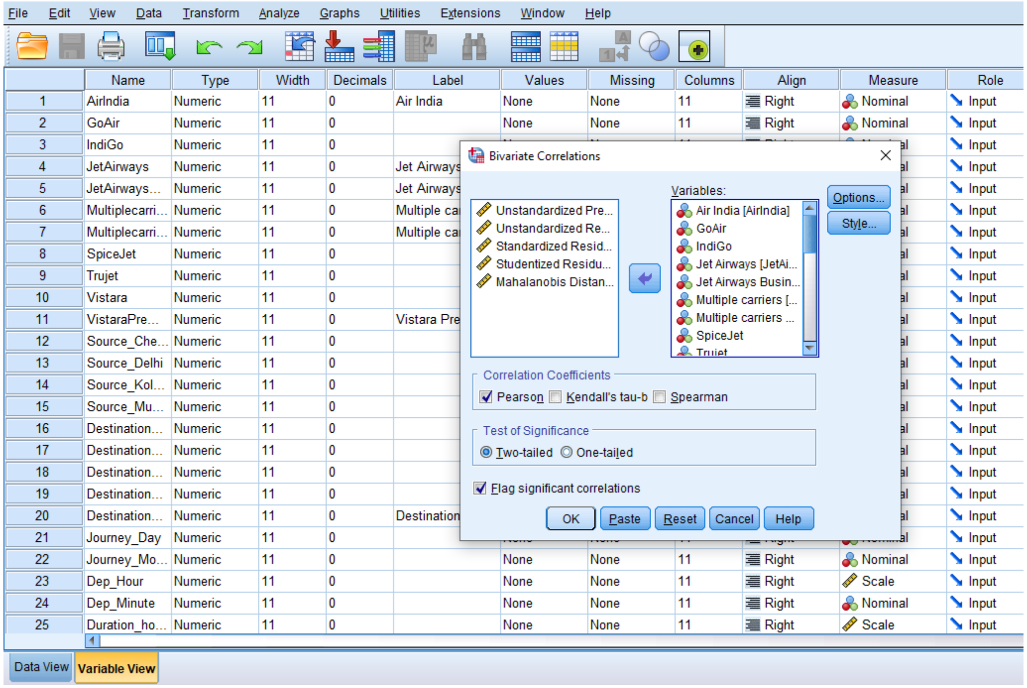
Variable View



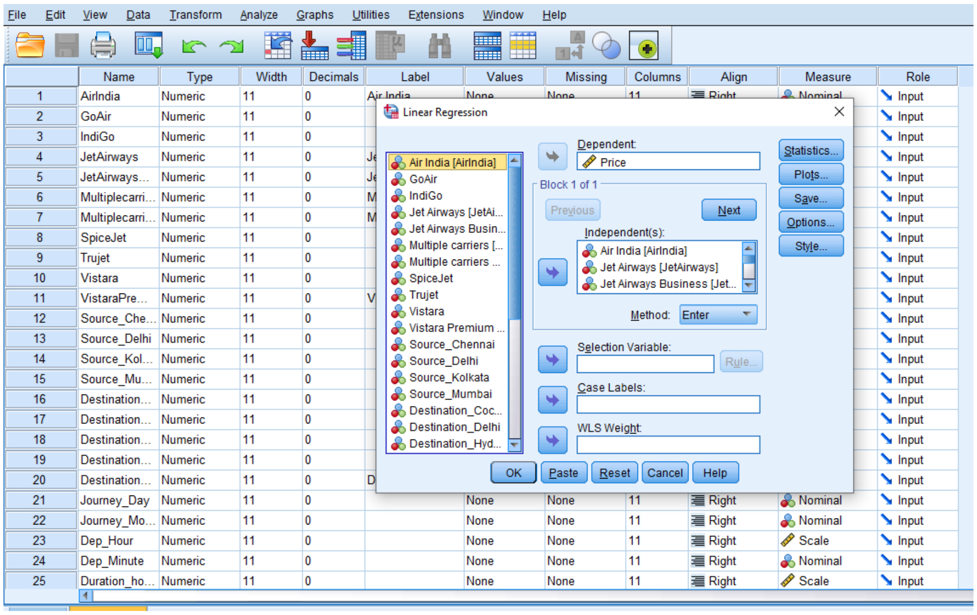
Data View



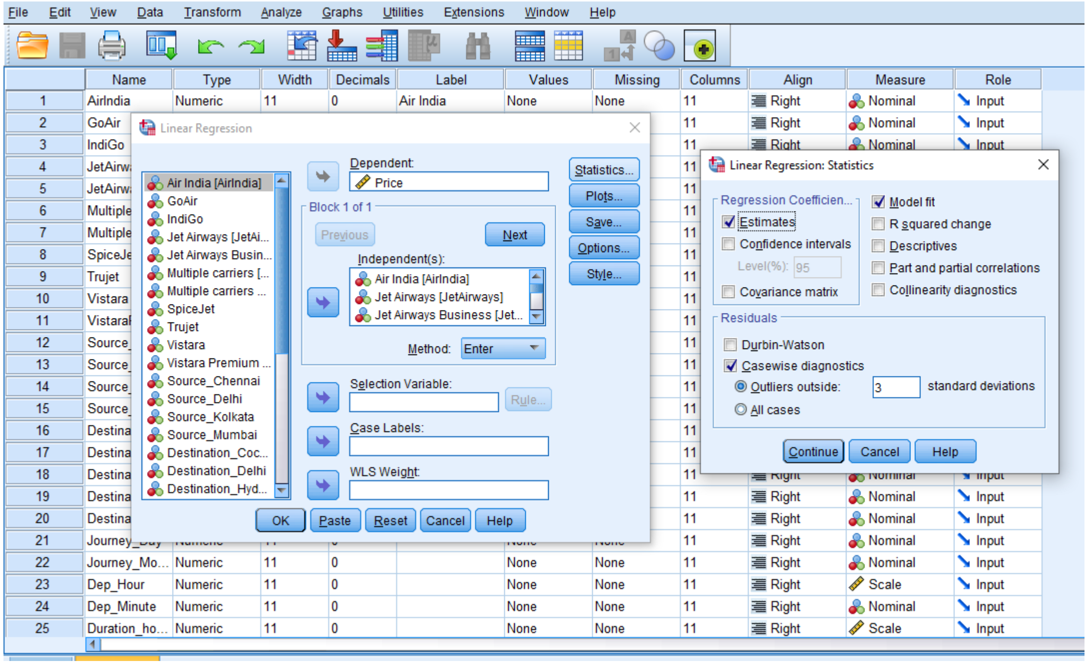
Analyze→Correlate→Bivariate



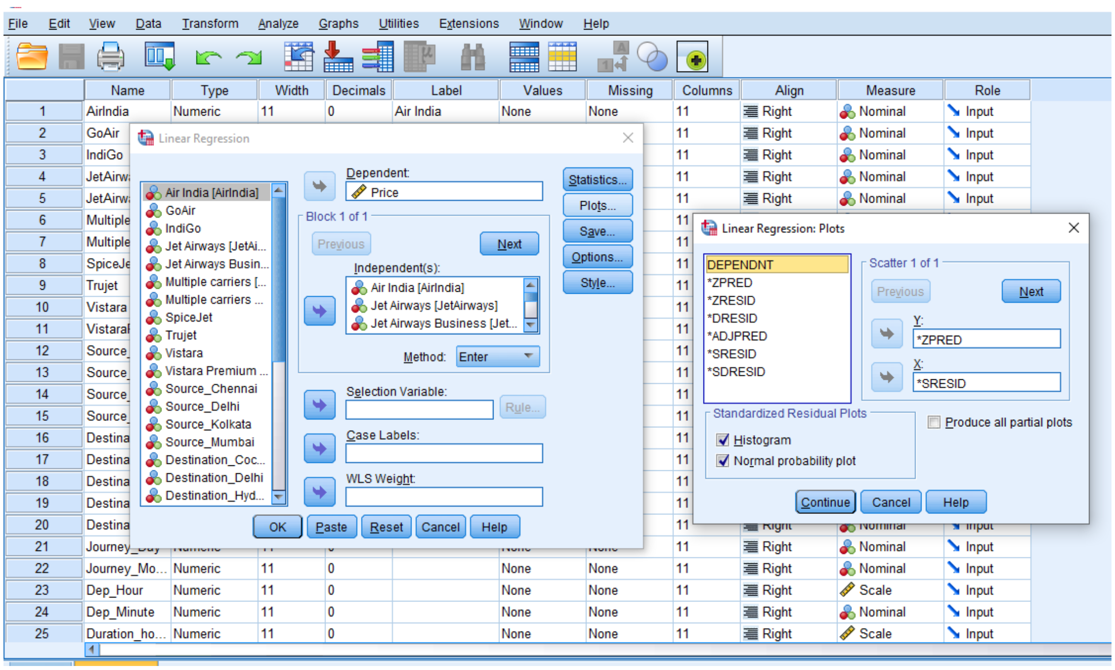
Bivariate Correlations



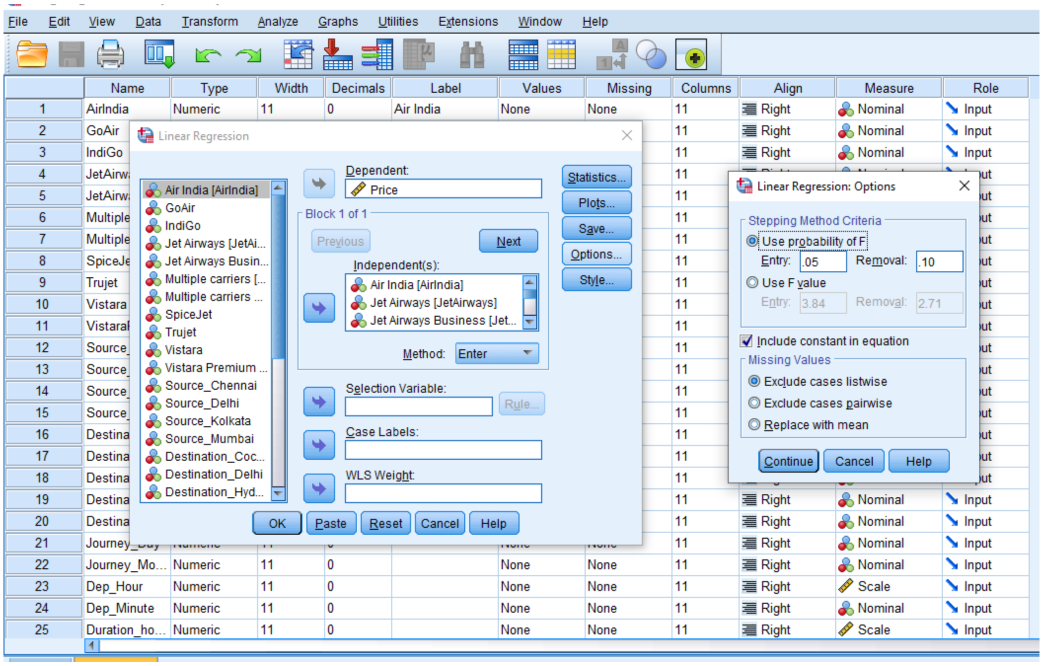
Linear Regression



Linear Regression Statistics



Linear Regression Plots



Linear Regression_Options

Step 2: Output report of linear regression model (Before removing outliers)

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.789[a]	.623	.622	2833.855

a. Predictors: (Constant), Total_Stops, Trujet, Multiple carriers Premium economy, Dep_Minute, Jet Airways Business, Vistara Premium economy, Source_Kolkata, Journey_Day, GoAir, Journey_Month, Vistara, Dep_Hour, Duration_mins, Air India, Destination_Kolkata, SpiceJet, Destination_Hyderabad, Multiple carriers, IndiGo, Destination_New Delhi, Destination_Delhi, Duration_hours, Jet Airways

b. Dependent Variable: Price

From the model summary table, we can see that the R square value also known as coefficient of determination is **0.623.** It indicates that 62.3 percent of the variance in the dependent variable i.e., price depends upon the variables used as predictors (independent variables).

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	1.415E+11	23	6154313241	766.345	.000[b]
	Residual	8.560E+10	10659	8030735.440		
	Total	2.271E+11	10682			

a. Dependent Variable: Price

b. Predictors: (Constant), Total_Stops, Trujet, Multiple carriers Premium economy, Dep_Minute, Jet Airways Business, Vistara Premium economy, Source_Kolkata, Journey_Day, GoAir, Journey_Month, Vistara, Dep_Hour, Duration_mins, Air India, Destination_Kolkata, SpiceJet, Destination_Hyderabad, Multiple carriers, IndiGo, Destination_New Delhi, Destination_Delhi, Duration_hours, Jet Airways

The analysis of variance is used to test the statistical significance of the R-square value in the Model Summary table. Here, the ANOVA results indicate statistical significance (as the significance value is less than 0.05) suggesting that the population R-square is significantly greater than zero.

Coefficients[a]

Model		Unstandardized Coefficients B	Unstandardized Coefficients Std. Error	Standardized Coefficients Beta	t	Sig.
1	(Constant)	7296.968	249.291		29.271	.000
	Air India	1588.002	183.776	.128	8.641	.000
	Go Air	24.569	259.587	.001	.095	**.925**
	IndiGo	223.702	174.620	.019	1.281	**.200**
	Jet Airways	4341.849	172.936	.452	25.107	.000
	Jet Airways Business	47513.870	1171.960	.244	40.542	.000
	Multiple carriers	3654.010	190.139	.250	19.218	.000
	Multiple carriers Premium economy	4073.605	806.106	.031	5.053	.000
	SpiceJet	-273.222	190.833	-.016	-1.432	**.152**
	Trujet	-2713.178	2842.847	-.006	-.954	**.340**
	Vistara	2112.649	211.179	.095	10.004	.000
	Vistara Premium economy	2864.120	1646.483	.010	1.740	**.082**
	Source_Kolkata	-110.784	77.418	-.011	-1.431	**.152**
	Destination_Delhi	-951.478	120.418	-.067	-7.901	.000
	Destination_Hyderabad	-1766.336	136.636	-.095	-12.927	.000
	Destination_Kolkata	-67.851	172.544	-.003	-.393	**.694**
	Destination_New Delhi	1694.997	122.589	.104	13.827	.000
	Journey_Day	-75.147	3.277	-.138	-22.932	.000
	Journey_Month	-412.502	26.900	-.104	-15.335	.000
	Dep_Hour	21.631	4.913	.027	4.402	.000
	Dep_Minute	-1.986	1.532	-.008	-1.296	**.195**
	Duration_hours	.956	5.199	.002	.184	**.854**
	Duration_mins	-.671	1.713	-.002	-.392	**.695**
	Total_Stops	2746.862	73.264	.402	37.493	.000

a. Dependent Variable: Price

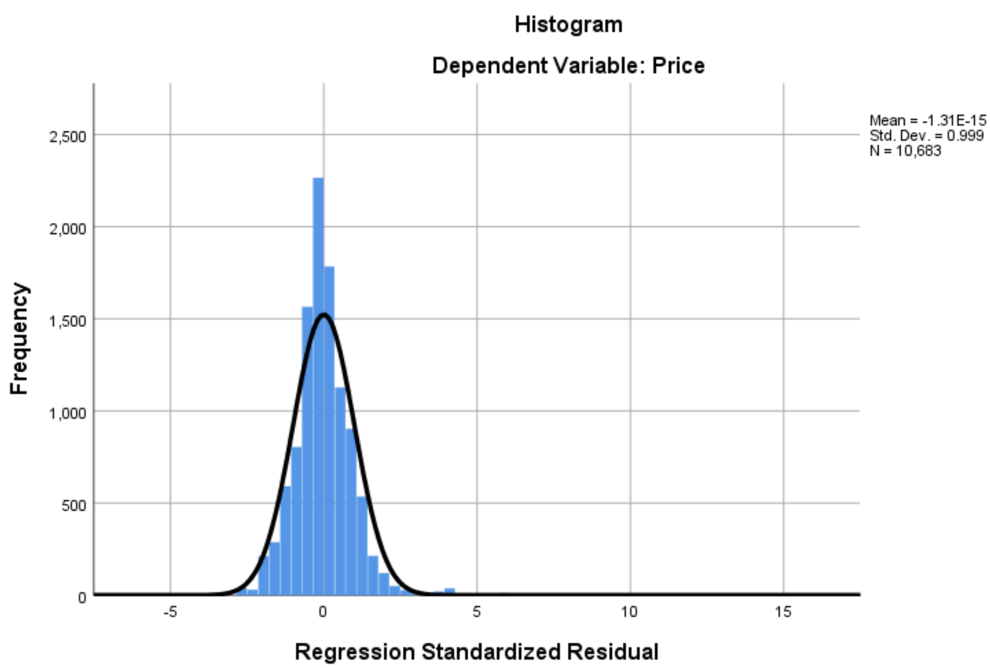
The variables indicated by red arrow have significance value greater than 0.05. Therefore, cannot be taken into consideration for further analysis.

Residuals Statistics[a]

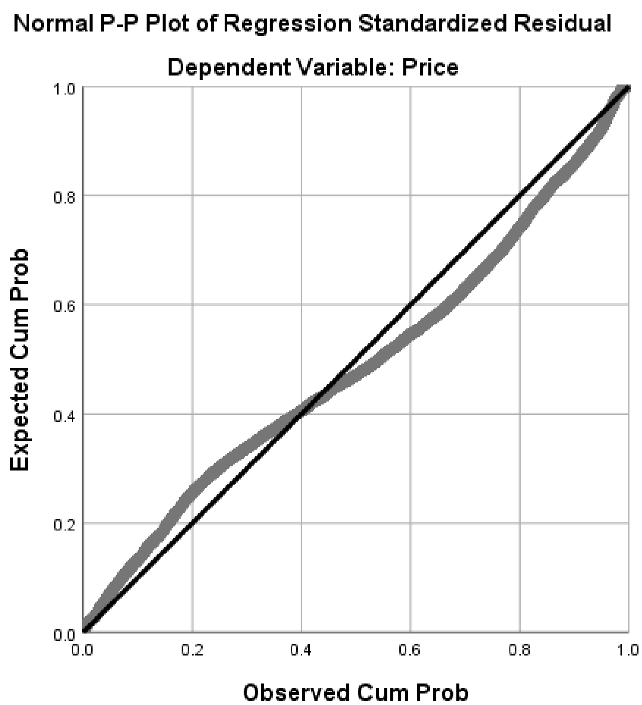
	Minimum	Maximum	Mean	Std. Deviation	N
Predicted Value	753.04	59258.52	9087.06	3640.218	10683
Std. Predicted Value	-2.289	13.783	.000	1.000	10683
Standard Error of Predicted Value	67.993	2833.855	121.381	57.519	10683
Adjusted Predicted Value	749.11	61836.35	9087.52	3640.597	10682
Residual	-12768.524	41023.180	.000	2830.803	10683
Std. Residual	-4.506	14.476	.000	.999	10683
Stud. Residual	-4.940	14.489	.000	1.001	10683
Deleted Residual	-15346.346	41097.641	.006	2841.653	10682
Stud. Deleted Residual	-4.945	14.633	.000	1.002	10682
Mahal. Distance	5.149	10681.000	22.998	129.739	10683
Cook's Distance	.000	.582	.000	.006	10682
Centered Leverage Value	.000	1.000	.002	.012	10683

a. Dependent Variable: Price

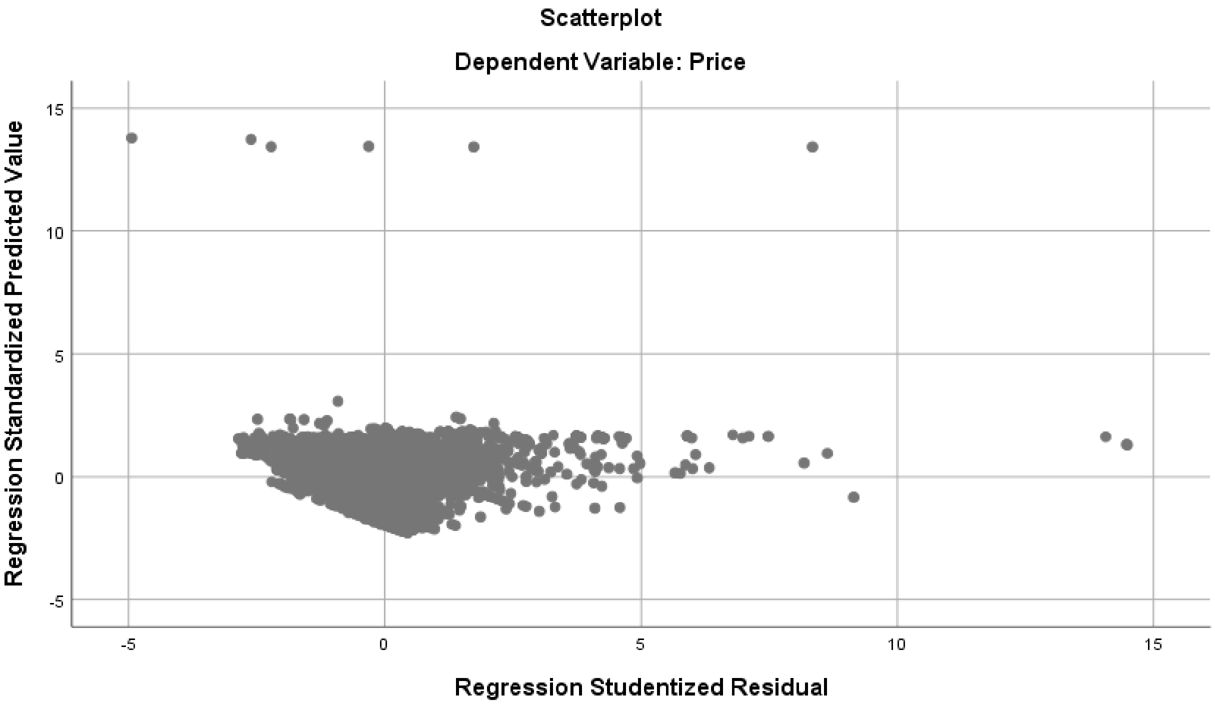
The above table is useful for judging the minimum and maximum values to screen for potential outliers. According to Pituch & Stevens, cases with values falling below -3 or above +3 need to be investigated further as possible candidate outliers. Here the minimum and maximum values in the dataset for the studentized residuals are -4.940 and 14.489 indicating for further investigation.



One of the assumptions of linear regression model is that the residuals are normally distributed. From the above histogram of the standardized residuals, a bell-shaped curve can be seen. Hence, it can be interpreted that the data is normally distributed to a greater extent.



The normal P-P plot can also be used to assess normality of the standardized residuals. This plot shows the relationship between the observed residuals against those expected under the condition of normality. The closer the observed residuals fall in relation to the regression line, the more evidence of normality. From this figure it can be said that this plot provides good support for evidence of normally distributed residuals as the residuals lie very closely to the regression line.



The above figure contains a plot of the studentized residuals against the standardized predicted values. In general, we are looking for the residuals to be randomly and evenly distributed around zero & falling between roughly -3 and +3 units. But here the residual points lie in the range of -5 to +15 units which is beyond the expected range which means that there are many outliers in the dataset and it needs further investigation.

Now, we have repeated these steps multiple times to remove possible outliers in the dataset.

After successful removal of the outliers, we have trained this model on the clean and well processed dataset and got the following results.

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.830[a]	.689	.689	2237.889

a. Predictors: (Constant), Total_Stops, Journey_Day, Jet Airways Business, Multiple carriers Premium economy, Destination_New Delhi, Dep_Hour, Vistara, Multiple carriers, Destination_Hyderabad, SpiceJet, Air India, Journey_Month, Destination_Delhi, Jet Airways

b. Dependent Variable: Price

From the Model Summary[b], it can be seen that the R-square value has increased to 68.9% in comparison with the earlier Model Summary[a] (R-square = 62.3%). Hence, we can say that the "68.9%" of the explained variance in the dependent variable i.e. price is determined by the independent variable.

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	115614558691.740	14	8258182763.696	1648.950	.000[b]
	Residual	52159857576.766	10415	5008147.631		
	Total	167774416268.506	10429			

a. Dependent Variable: Price

b. Predictors: (Constant), Total_Stops, Journey_Day, Jet Airways Business, Multiple carriers Premium economy, Destination_New Delhi, Dep_Hour, Vistara, Multiple carriers, Destination_Hyderabad, SpiceJet, Air India, Journey_Month, Destination_Delhi, Jet Airways

Here, ANOVA table indicates the statistical significance of the model as the Significance-value is less than 0.05.

Coefficients[a]

Model		Unstandardized Coefficients B	Unstandardized Coefficients Std. Error	Standardized Coefficients Beta	t	Sig.
1	(Constant)	6733.047	133.952		50.265	.000
	Air India	1373.347	77.773	.127	17.658	.000
	Jet Airways	3984.485	62.742	.476	63.506	.000
	Jet Airways Business	39866.317	1294.690	.169	30.792	.000
	Multiple carriers	3297.287	83.272	.259	39.597	.000
	Multiple carriers Premium economy	3890.481	624.278	.034	6.232	.000
	SpiceJet	-516.760	91.068	-.034	-5.674	.000
	Vistara	1927.851	112.548	.100	17.129	.000
	Destination_Delhi	-1032.892	80.750	-.084	-12.791	.000
	Destination_Hyderabad	-2157.769	99.512	-.131	-21.684	.000
	Destination_New Delhi	409.779	93.261	.028	4.394	.000
	Journey_Day	-51.410	2.622	-.108	-19.609	.000
	Journey_Month	-304.489	21.559	-.087	-14.123	.000
	Dep_Hour	17.663	3.909	.025	4.519	.000
	Total_Stops	2651.158	45.934	.447	57.717	.000

a. Dependent Variable: Price

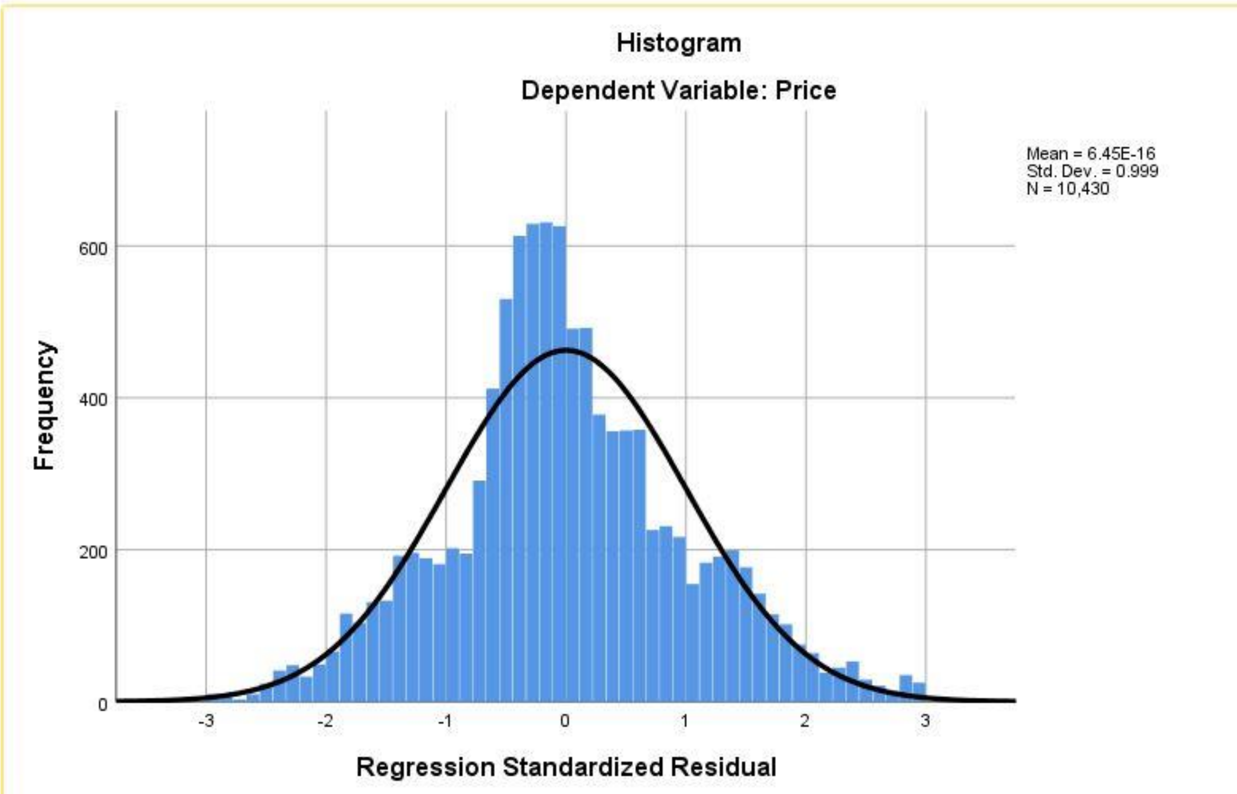
The coefficient table indicates the individual significance-values of all the independent variables. For all the independent variables the significance-value is less than 0.05 (i.e. 0.000). Hence, all the above-mentioned independent variables are statistically significant and can be considered for model building.

Residuals Statistics[a]

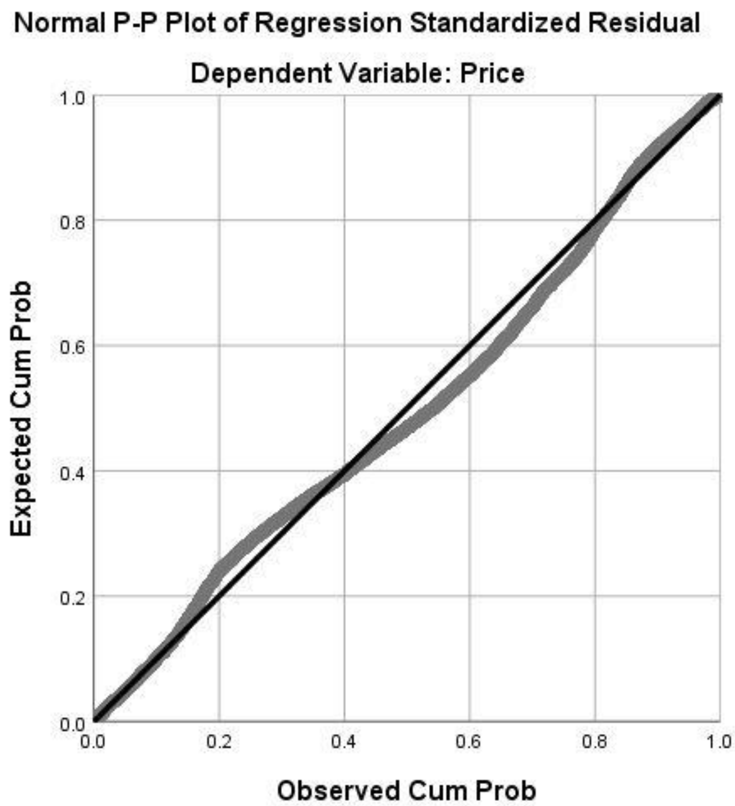
	Minimum	Maximum	Mean	Std. Deviation	N
Predicted Value	931.84	51187.25	8820.63	3329.545	10430
Std. Predicted Value	-2.369	12.724	.000	1.000	10430
Standard Error of Predicted Value	39.819	1294.331	77.561	34.451	10430
Adjusted Predicted Value	928.47	53539.00	8820.65	3329.882	10430
Residual	-6700.604	6615.127	.000	2236.387	10430
Std. Residual	-2.994	2.956	.000	.999	10430
Stud. Residual	-2.996	2.958	.000	1.000	10430
Deleted Residual	-7049.005	6623.639	-.021	2240.154	10430
Stud. Deleted Residual	-2.997	2.959	.000	1.000	10430
Mahal. Distance	2.302	3487.639	13.999	65.447	10430
Cook's Distance	.000	.221	.000	.002	10430
Centered Leverage Value	.000	.334	.001	.006	10430

a. Dependent Variable: Price

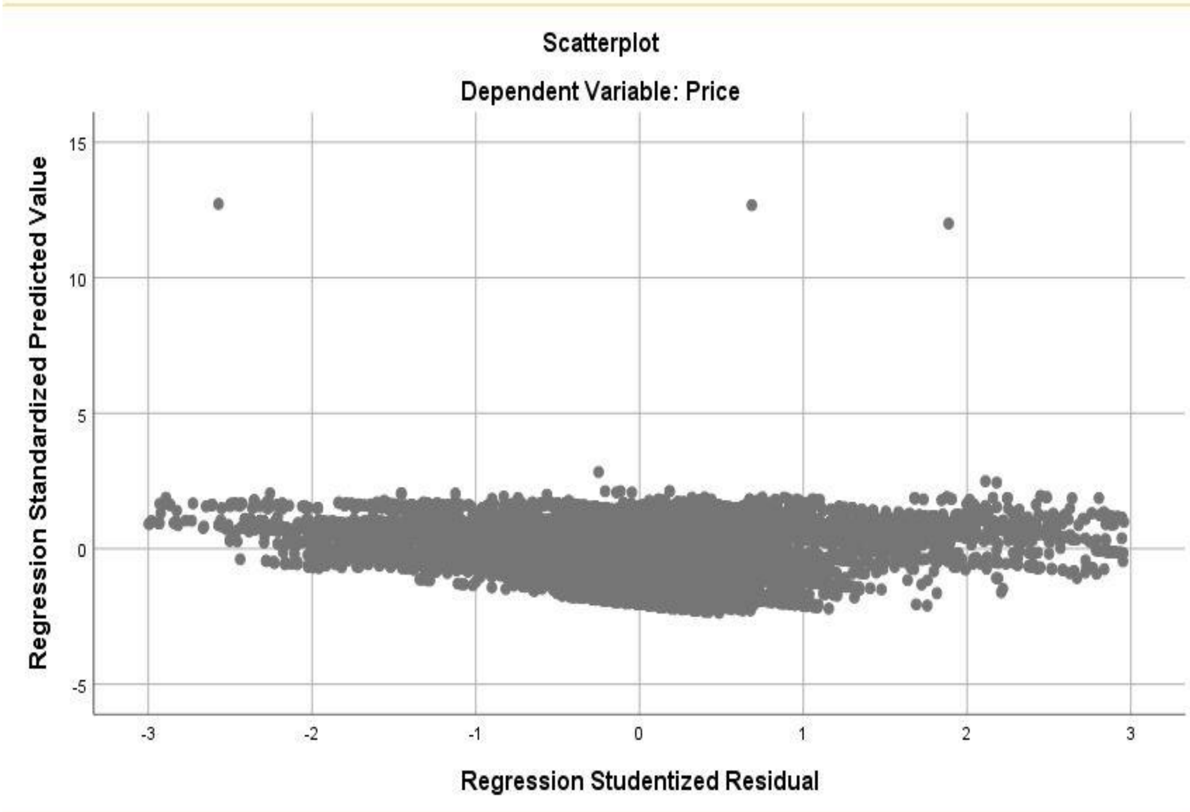
In the above table the studentized residuals lie between -2.996 and +2.958. According to Pituch & Stevens, cases with values lying between -3 to +3 are acceptable and do not contain outliers and are said to possess homoscedasticity.



We can see a significant change in the shape of histogram of standardized residuals after processing of the data. It appears to have grown wider than before. Hence, we can say that the assumption of linear regression that the residuals are normally distributed is satisfied.



From this figure it can be seen that residuals have moved even closer to the regression line than before. Hence, we can say that this p-p plot provides a good evidence for normally distributed residuals.



The above scatter plot shows even distribution of studentized residuals against the standardized predicted values. As no systematic pattern like fan shaped or bow shaped is observed in the spread of the residuals. Hence, it can be said that the data follows the principle of homoscedasticity.

CONCLUSION:

- Final model after removing the outliers showed an improvement in R square to 68.9 %.
- Homoscedasticity is achieved
- P-P plot showed improvement
- Residual normality is achieved

FUTURE SCOPE

- We can deploy this model on AWS or GCP and build a web application wherein the users can enter the basic information (Journey Date, Source, Destination, number of stops) and get the flight price on button click.

- Retraining approach: Generating pickle file after every month to increase the accuracy and R-Square.