FLIP ROBO

Flight Price Prediction

Submitted by:

Bhumik P Shah

ACKNOWLEDGMENT

While preparing this case study I had gone through a couple of videos on You tube for reference purpose to understand in depth about the flight prices fluctuating based on demand and supply.

INTRODUCTION

- ## Business Problem Framing

  Airlines companies use complex algorithms to calculate flight prices given various conditions present at that particular time. These methods take financial, marketing and various social factors into account to predict flight prices. Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That’s why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.

  ## Conceptual Background of the Domain Problem

  Anyone who has booked a flight ticket knows how unexpectedly the prices vary. 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 flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases).

- ## Review of Literature

  Because I truly think that sharing sources and knowledges allow to help others but also ourselves, the sources of the project are available at the following link:
  https://www.researchgate.net/publication/337821411_Predicting_Flight_Prices_in_India

  ## Motivation for the Problem Undertaken

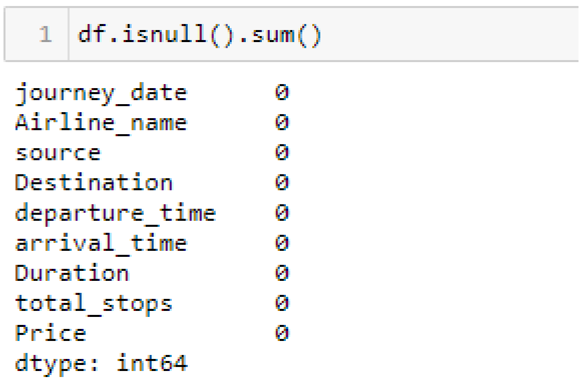
  We could inform the travellers with the optimal time to buy their flight tickets based on the historic data and also show them various trends in the airline industry we could help them save money on their travels.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

  Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a dataset, including its size, initial patterns in the data and other attributes.

  There are no null values present in any of the features in this dataset.



```
df.isnull().sum()
```

```
journey_date       0
Airline_name       0
source             0
Destination        0
departure_time     0
arrival_time       0
Duration           0
total_stops        0
Price              0
dtype: int64
```

There are outliers present in independent features named as duration, total_stops, as confirmed by boxplot. Further, I am not removing outliers as total_stops can be 1 or more depending on the flight connectivity between source and destination i.e whether it is a non stop flight between source and destination or 1 stop flight or 2 stop flight The similar thing is for feature duration i.e time taken to reach destination can be more than 24 hours depending on the total_stops between source and destination and the layover time. The independent features Duration and total_stops are having skewness greater than 0.5 which is treated using power transform method.

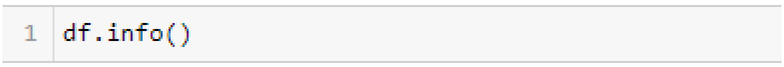
The correlation of every numerical feature with label/target is checked and one of the feature journey_month is having no correlation with label/target which is price in your case so that feature is dropped from the dataset.

Data Sources and their formats

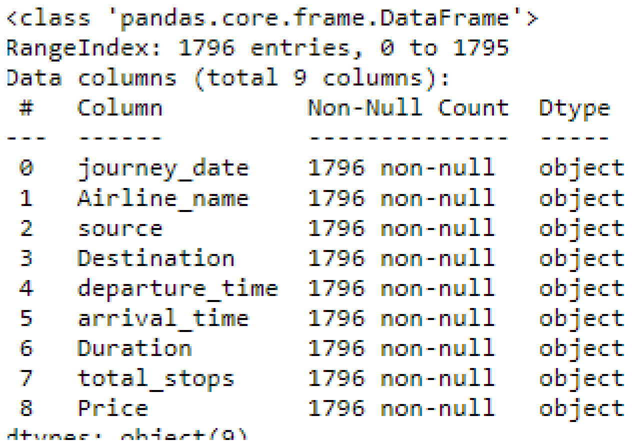
The data is collected from easemytrip.com for the duration 05.10.2021 to 13.10.2021 for the flights available between Delhi as a source location and Mumbai as a destination location using web scrapping tool selenium.

The dataset contains 1796 records (rows) and 9 features (columns) with price feature being the target attribute.

The details of 9 features(columns) are attached below:



```
df.info()
```

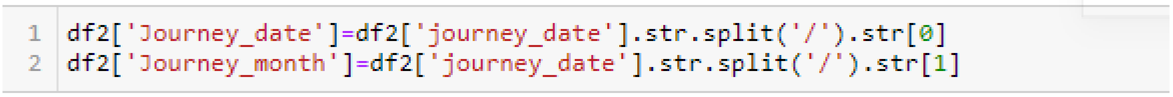


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1796 entries, 0 to 1795
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   journey_date    1796 non-null   object
 1   Airline_name    1796 non-null   object
 2   source          1796 non-null   object
 3   Destination     1796 non-null   object
 4   departure_time  1796 non-null   object
 5   arrival_time    1796 non-null   object
 6   Duration        1796 non-null   object
 7   total_stops     1796 non-null   object
 8   Price           1796 non-null   object
dtypes: object(9)
```

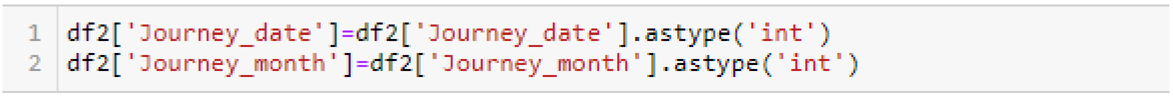
All the features in this dataset are of object datatype.

- ## Data Pre-processing Done

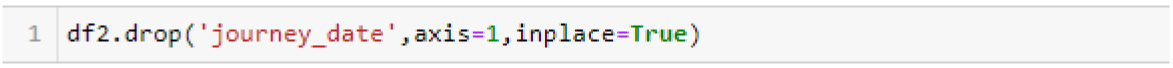
Extracting out new features named as Journey_date and Journey_month from the existing feature journey_date and finally dropping the feature journey_date as all the useful information is extracted.



```
df2['Journey_date']=df2['journey_date'].str.split('/').str[0]
df2['Journey_month']=df2['journey_date'].str.split('/').str[1]
```

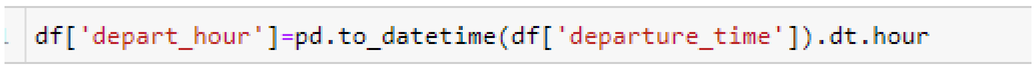


```
df2['Journey_date']=df2['Journey_date'].astype('int')
df2['Journey_month']=df2['Journey_month'].astype('int')
```

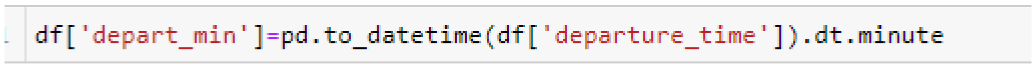


```
df2.drop('journey_date',axis=1,inplace=True)
```

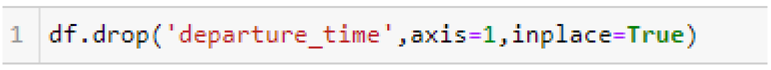
On the similar lines new features depart_hour and depart_min are extracted from the existing feature departure_time and finally dropping the feature departure_time as all useful information is extracted.



```
df['depart_hour']=pd.to_datetime(df['departure_time']).dt.hour
```

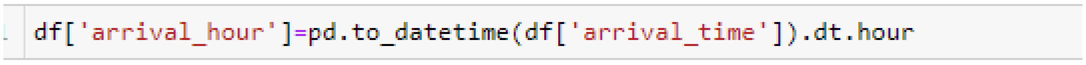


```
df['depart_min']=pd.to_datetime(df['departure_time']).dt.minute
```

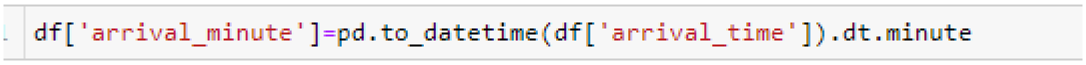


```
df.drop('departure_time',axis=1,inplace=True)
```

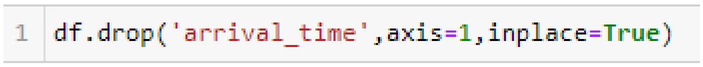
Similarly new features named as arrival_hour and arrival_minute are extracted from the existing feature arrival_time and finally dropping the feature arrival_time as all useful information is extracted.



```
df['arrival_hour']=pd.to_datetime(df['arrival_time']).dt.hour
```

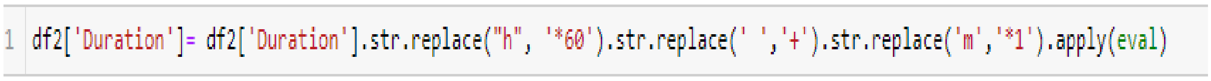


```
df['arrival_minute']=pd.to_datetime(df['arrival_time']).dt.minute
```



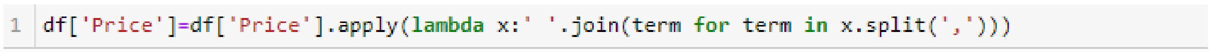
```
df.drop('arrival_time',axis=1,inplace=True)
```

The feature duration i.e time taken to reach from source to destination is having hours and minutes in records so the same is converted entirely into minutes from hours say for example if it is 2 hours and 15 minutes it is converted into 135 minutes for better visualization purpose.

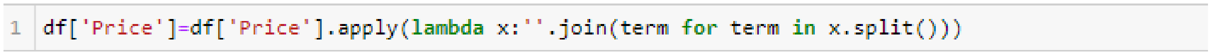


```
df2['Duration']= df2['Duration'].str.replace("h", '*60').str.replace(' ','+').str.replace('m','*1').apply(eval)
```

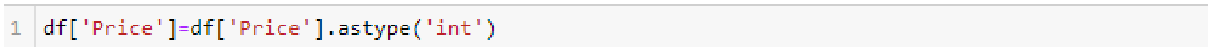
The feature price which is the target attribute in this dataset is having object datatype which is converted into integer datatype as price is always a numerical value.



```
df['Price']=df['Price'].apply(lambda x:' '.join(term for term in x.split(',')))
```



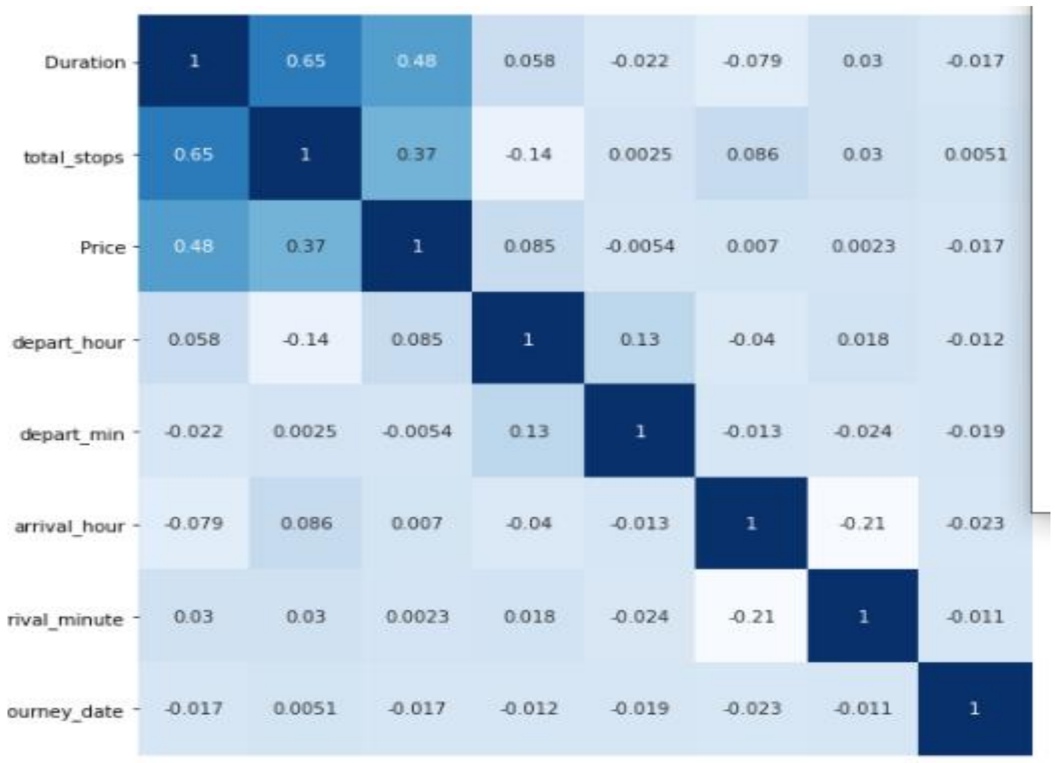
```
df['Price']=df['Price'].apply(lambda x:''.join(term for term in x.split()))
```



```
df['Price']=df['Price'].astype('int')
```

- Data Inputs- Logic- Output Relationships

As per the correlation details attached below the target attribute price i s having positive linear correlation with features Duration and total_st ops The feature journey_month is dropped from the dataset as the ent ire data is extracted for the month of October so there is no variation in the data resulting into no correlation with target attribute price.

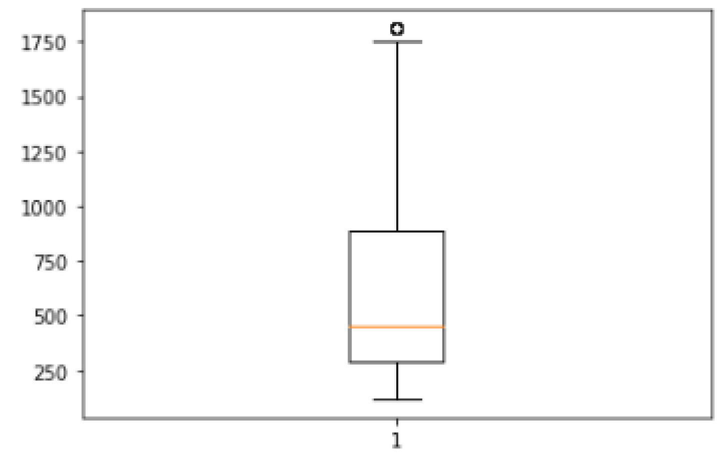


- Hardware and Software Requirements and Tools Used
  I have used Jupyter Notebook for writing python codes. Further, libraries such as Pandas, Seaborn, Matplotlib, SKlearn, NumPy are used.

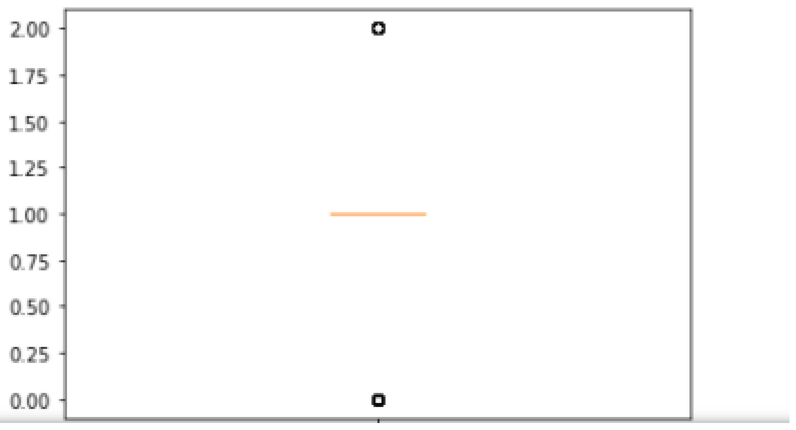
Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  The shape of the dataset, datatypes, null values present in the dataset or not are amongst few things that are first to be known. There are no null values present in this dataset. Further, next step is to go for outlier detection if any using box plot and distribution plot. The features Duration and total_stops are having outliers as confirmed from the boxplot.

Duration



total_stops



Further, the outliers are not treated as the total_stops can be 1 or more and the same is for duration feature as the duration to reach from source to destination can be more than 24 hours.

After treating outliers next step is to go for skewness check .The features duration and total_stops are having skewness greater than 0.5 which is treated using power transform method.

```
df2.skew()
```

```
Duration          0.822944
total_stops      -1.206754
Price             1.483523
depart_hour       0.070631
depart_min       -0.040267
arrival_hour     -0.487459
arrival_minute    0.138137
Journey_date     -0.093633
Journey_month     0.000000
```

```python
df3['Duration']=power_transform(df3['Duration'].values.reshape(-1,1))
df3['total_stops']=power_transform(df3['total_stops'].values.reshape(-1,1))
```

```python
df3.skew()
```

```
Duration         -0.046080
total_stops       0.280533
Price             1.483523
depart_hour       0.070631
depart_min       -0.040267
arrival_hour     -0.487459
arrival_minute    0.138137
Journey_date     -0.093633
Journey_month     0.000000
dtype: float64
```

The categorical features are converted into numeric type using label encoder as machine learning models only accepts numerical values.

```python
categorical_columns=df3.select_dtypes(include=[np.object])
```

```python
categorical_columns
```

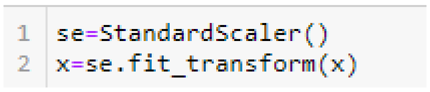
	Airline_name	source	Destination
0	Indigo	Delhi	Mumbai
1	GO FIRST	Delhi	Mumbai
2	GO FIRST	Delhi	Mumbai
3	GO FIRST	Delhi	Mumbai
4	SpiceJet	Delhi	Mumbai
5	Indigo	Delhi	Mumbai
6	Indigo	Delhi	Mumbai
7	Indigo	Delhi	Mumbai
8	Indigo	Delhi	Mumbai
9	Indigo	Delhi	Mumbai
10	Indigo	Delhi	Mumbai
11	Indigo	Delhi	Mumbai

```python
from sklearn.preprocessing import LabelEncoder
for j in categorical_columns.columns:
    le=LabelEncoder()
    df3[j]=le.fit_transform(df3[j])
```

After completing all the above mentioned steps dataset is splitted into two variables x and y.

```python
x=df3.drop('Price',axis=1)
y=df3['Price']
```

Standard Scaling is to be applied first before splitting the data into train_test_split.



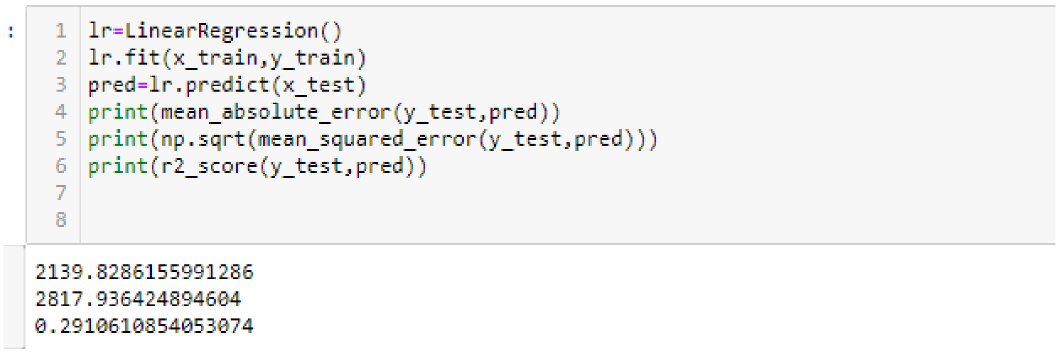
```python
se=StandardScaler()
x=se.fit_transform(x)
```

- ## Testing of Identified Approaches (Algorithms)

  The various algorithms like Linear Regression, Decision Tree Regressor, SVR, KNeighborsRegressor, Ada Boost Regressor, Gradient Boosting Regressor, Random Forest Regressor, Lasso , Ridge are used. First, approach is to find r2 score using all mentioned algorithms. Secondly, cross validation score of all the above mentioned algorithms are checked .The difference between r2 score and cross validation score is minimum for KNeighborsRegressor. KNeighborsRegressor is my best model and to obtain better r2_score hypertuning is carried out.

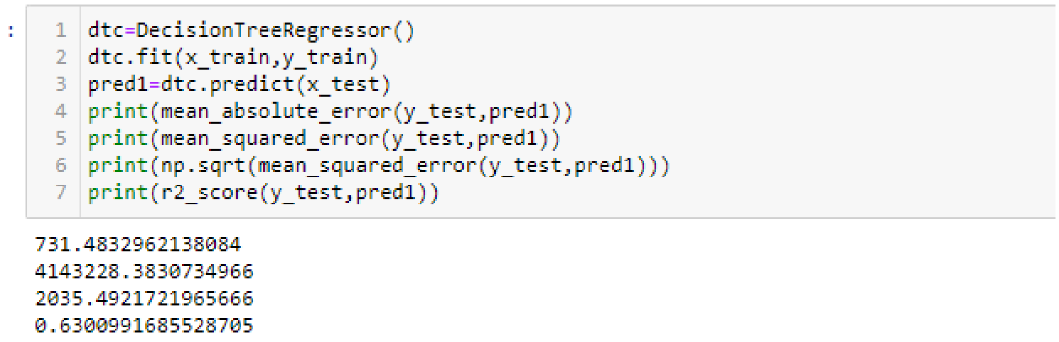
- ## Run and Evaluate selected models

  Kindly find attached below the snapshots of various algorithms used



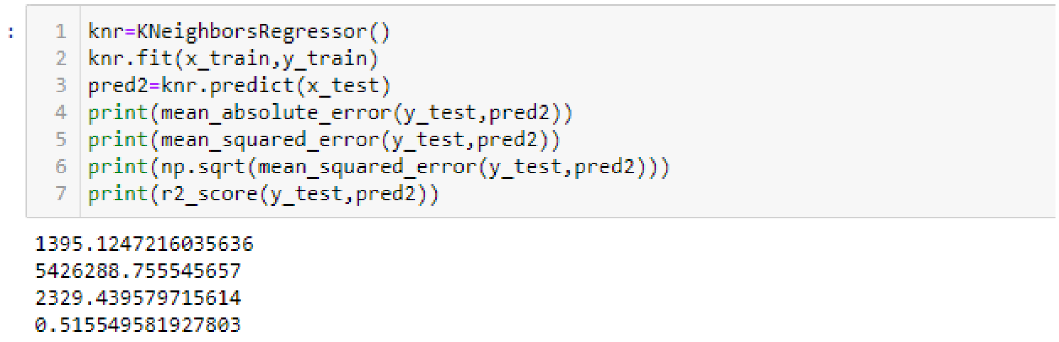
```python
lr=LinearRegression()
lr.fit(x_train,y_train)
pred=lr.predict(x_test)
print(mean_absolute_error(y_test,pred))
print(np.sqrt(mean_squared_error(y_test,pred)))
print(r2_score(y_test,pred))
```

```
2139.8286155991286
2817.936424894604
0.2910610854053074
```



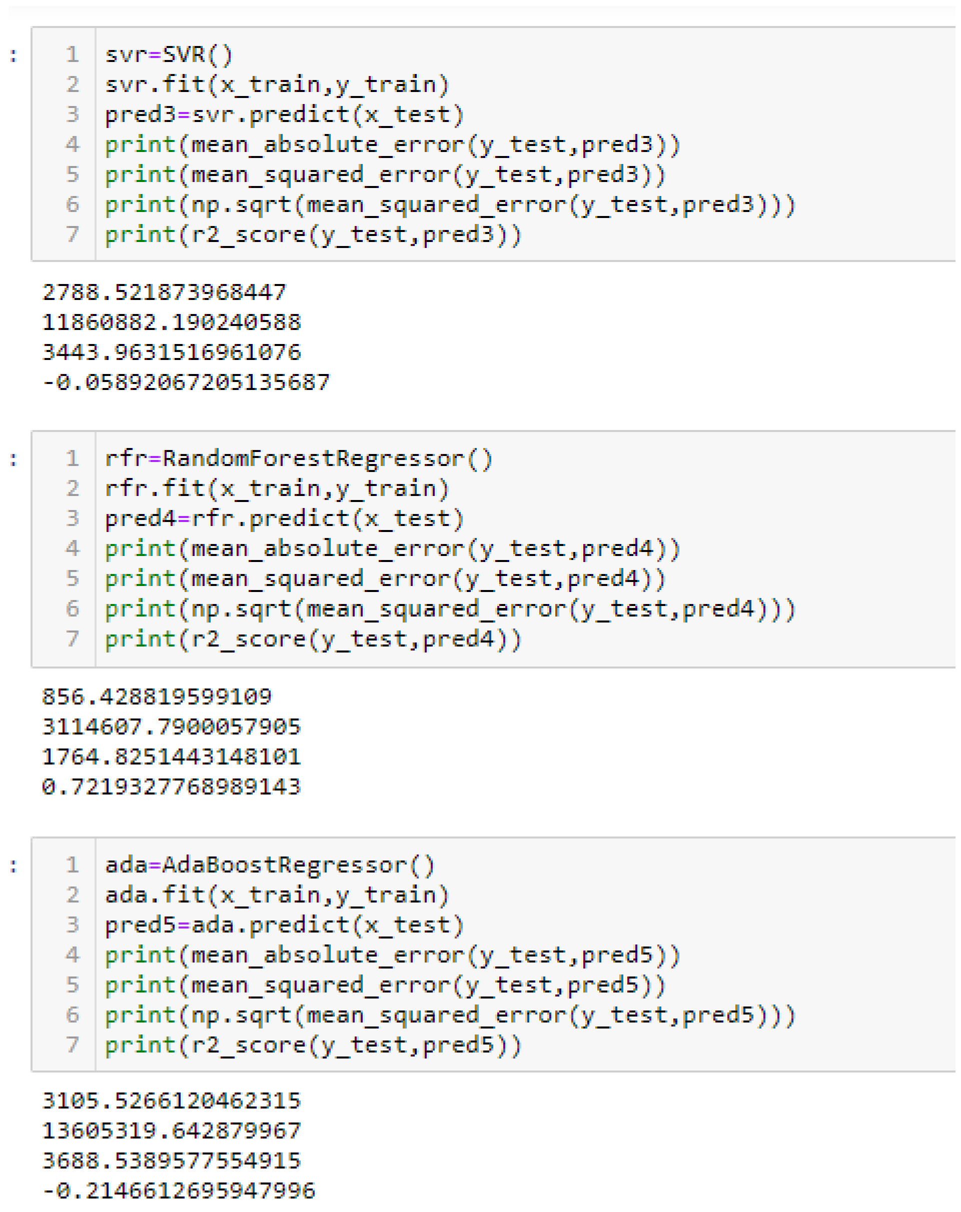
```python
dtc=DecisionTreeRegressor()
dtc.fit(x_train,y_train)
pred1=dtc.predict(x_test)
print(mean_absolute_error(y_test,pred1))
print(mean_squared_error(y_test,pred1))
print(np.sqrt(mean_squared_error(y_test,pred1)))
print(r2_score(y_test,pred1))
```

```
731.4832962138084
4143228.3830734966
2035.4921721965666
0.6300991685528705
```



```python
knr=KNeighborsRegressor()
knr.fit(x_train,y_train)
pred2=knr.predict(x_test)
print(mean_absolute_error(y_test,pred2))
print(mean_squared_error(y_test,pred2))
print(np.sqrt(mean_squared_error(y_test,pred2)))
print(r2_score(y_test,pred2))
```

```
1395.1247216035636
5426288.755545657
2329.439579715614
0.515549581927803
```



```
svr=SVR()
svr.fit(x_train,y_train)
pred3=svr.predict(x_test)
print(mean_absolute_error(y_test,pred3))
print(mean_squared_error(y_test,pred3))
print(np.sqrt(mean_squared_error(y_test,pred3)))
print(r2_score(y_test,pred3))
```

```
2788.521873968447
11860882.190240588
3443.9631516961076
-0.05892067205135687
```

```
rfr=RandomForestRegressor()
rfr.fit(x_train,y_train)
pred4=rfr.predict(x_test)
print(mean_absolute_error(y_test,pred4))
print(mean_squared_error(y_test,pred4))
print(np.sqrt(mean_squared_error(y_test,pred4)))
print(r2_score(y_test,pred4))
```

```
856.428819599109
3114607.7900057905
1764.8251443148101
0.7219327768989143
```

```
ada=AdaBoostRegressor()
ada.fit(x_train,y_train)
pred5=ada.predict(x_test)
print(mean_absolute_error(y_test,pred5))
print(mean_squared_error(y_test,pred5))
print(np.sqrt(mean_squared_error(y_test,pred5)))
print(r2_score(y_test,pred5))
```

```
3105.5266120462315
13605319.642879967
3688.5389577554915
-0.2146612695947996
```

```python
gr=GradientBoostingRegressor()
gr.fit(x_train,y_train)
pred6=ada.predict(x_test)
print(mean_absolute_error(y_test,pred6))
print(mean_squared_error(y_test,pred6))
print(np.sqrt(mean_squared_error(y_test,pred6)))
print(r2_score(y_test,pred6))
```

```
3105.5266120462315
13605319.642879967
3688.5389577554915
-0.2146612695947996
```

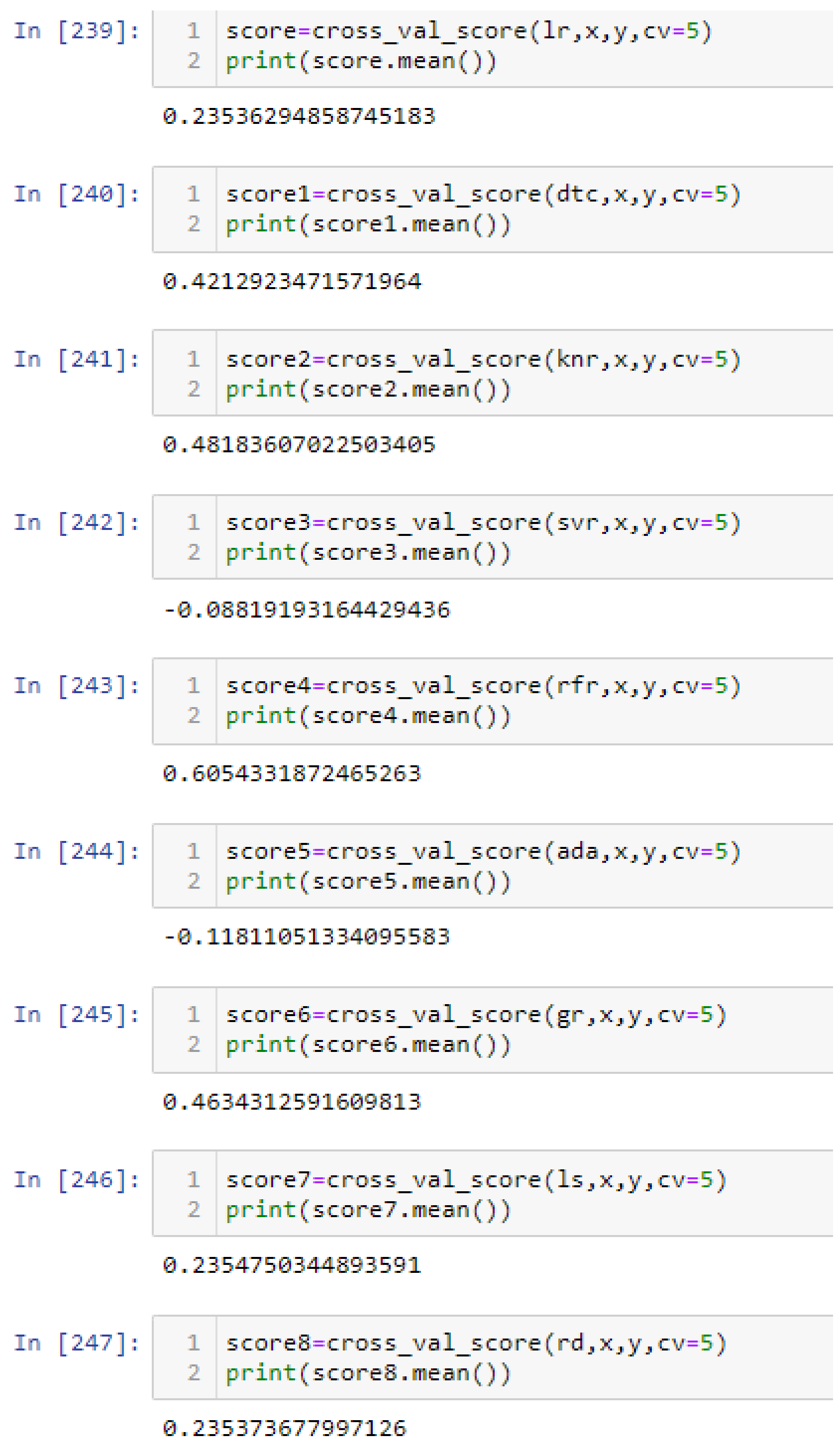
```python
from sklearn.linear_model import Lasso,Ridge
```

```python
ls=Lasso()
ls.fit(x_train,y_train)
pred7=ls.predict(x_test)
print(mean_absolute_error(y_test,pred7))
print(mean_squared_error(y_test,pred7))
print(np.sqrt(mean_squared_error(y_test,pred7)))
print(r2_score(y_test,pred7))
```

```
2139.7172378661453
7939803.55564747
2817.7657027594523
0.2911469836518602
```

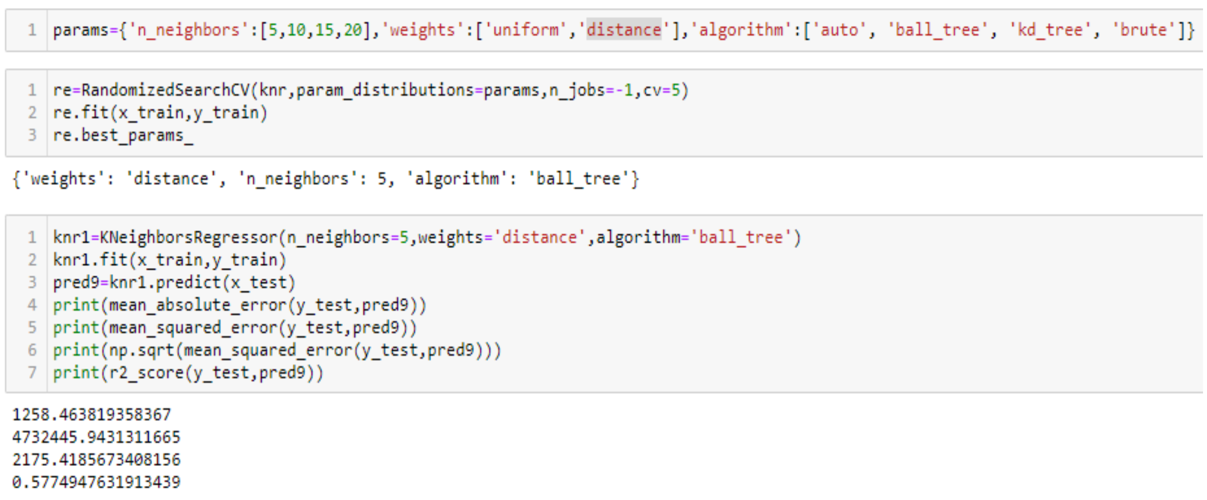
```python
rd=Ridge()
rd.fit(x_train,y_train)
pred8=rd.predict(x_test)
print(mean_absolute_error(y_test,pred8))
print(mean_squared_error(y_test,pred8))
print(np.sqrt(mean_squared_error(y_test,pred8)))
print(r2_score(y_test,pred8))
```

```
2139.8498302838893
7940801.373629819
2817.942755562969
0.2910579000515423
```



```
In [239]:
score=cross_val_score(lr,x,y,cv=5)
print(score.mean())

0.23536294858745183

In [240]:
score1=cross_val_score(dtc,x,y,cv=5)
print(score1.mean())

0.4212923471571964

In [241]:
score2=cross_val_score(knr,x,y,cv=5)
print(score2.mean())

0.48183607022503405

In [242]:
score3=cross_val_score(svr,x,y,cv=5)
print(score3.mean())

-0.08819193164429436

In [243]:
score4=cross_val_score(rfr,x,y,cv=5)
print(score4.mean())

0.6054331872465263

In [244]:
score5=cross_val_score(ada,x,y,cv=5)
print(score5.mean())

-0.11811051334095583

In [245]:
score6=cross_val_score(gr,x,y,cv=5)
print(score6.mean())

0.4634312591609813

In [246]:
score7=cross_val_score(ls,x,y,cv=5)
print(score7.mean())

0.2354750344893591

In [247]:
score8=cross_val_score(rd,x,y,cv=5)
print(score8.mean())

0.235373677997126
```

Hyper tuning of KNeighborsRegressor, is carried out to find out if the r2_score can be improved or not.



```
params={'n_neighbors':[5,10,15,20],'weights':['uniform','distance'],'algorithm':['auto', 'ball_tree', 'kd_tree', 'brute']}
```

```
re=RandomizedSearchCV(knr,param_distributions=params,n_jobs=-1,cv=5)
re.fit(x_train,y_train)
re.best_params_
```

```
{'weights': 'distance', 'n_neighbors': 5, 'algorithm': 'ball_tree'}
```

```
knr1=KNeighborsRegressor(n_neighbors=5,weights='distance',algorithm='ball_tree')
knr1.fit(x_train,y_train)
pred9=knr1.predict(x_test)
print(mean_absolute_error(y_test,pred9))
print(mean_squared_error(y_test,pred9))
print(np.sqrt(mean_squared_error(y_test,pred9)))
print(r2_score(y_test,pred9))
```

```
1258.463819358367
4732445.9431311665
2175.4185673408156
0.5774947631913439
```

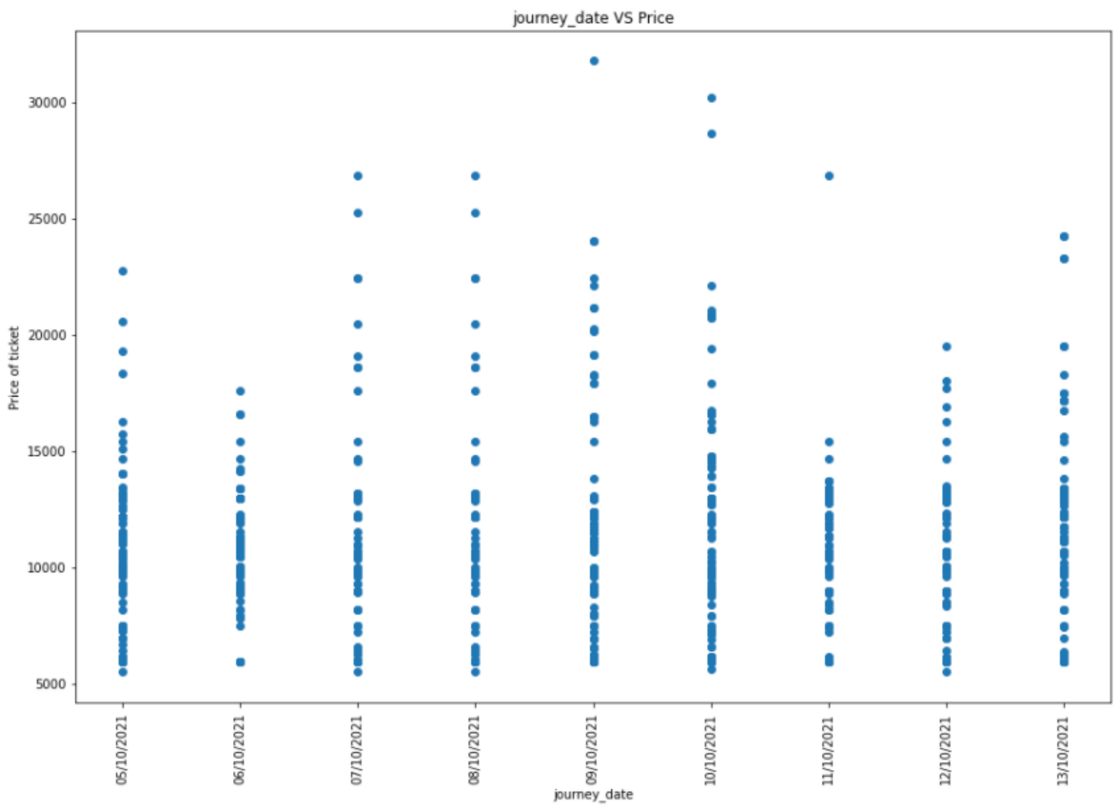
KNeighborsRegressor is my best model with r2_Score of 58%.

Key Metrics for success in solving problem under consideration

The feature duration is converted into minutes from hours so that better visualization can be carried out.

- Visualizations

  Data Visualization is the graphical representation of the information and data. By using visual elements like charts, graphs and maps data visualization tools provide an accessible way to see and understand trends, outliers and patterns in data.



```
pd.crosstab(df['journey_date'],df['Price'],margins=True)
```

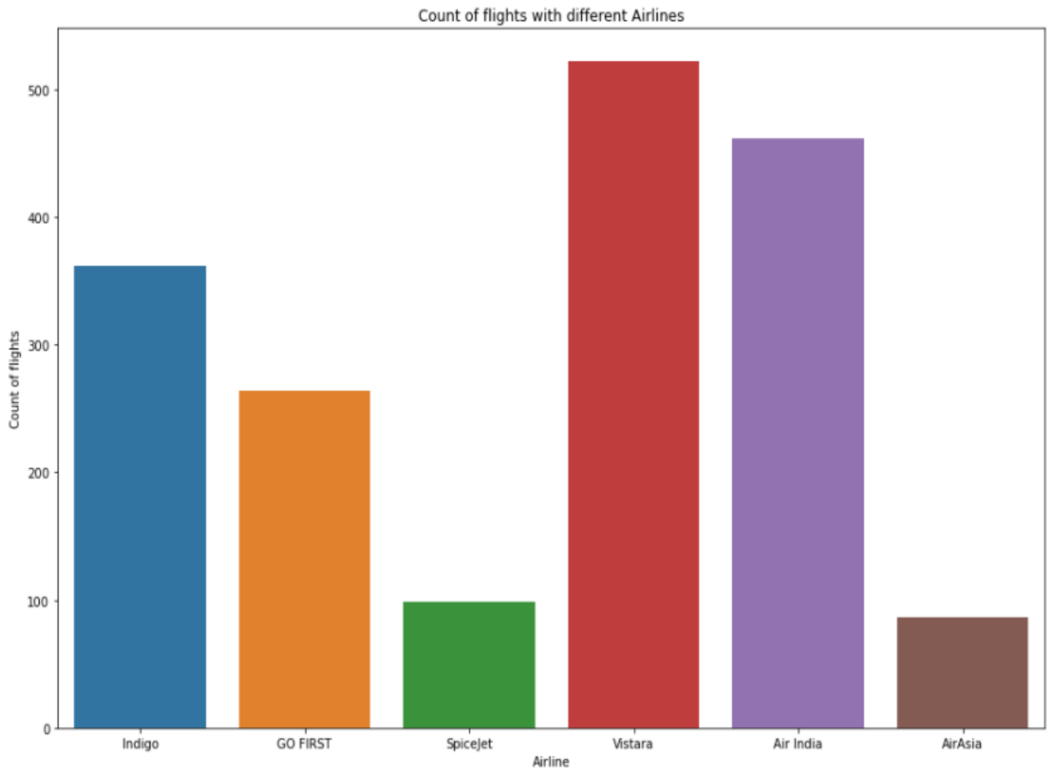
Price journey_date	5496	5607	5953	5954	5955	5956	6060	6165	6166	6271	...	22755	23281	24015	24225	25275	26850	28635	30210	31785	All
05/10/2021	1	0	6	14	42	0	1	3	0	0	...	1	0	0	0	0	0	0	0	0	171
06/10/2021	0	0	5	18	55	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	190
07/10/2021	1	0	8	18	58	0	1	0	0	1	...	0	0	0	0	1	1	0	0	0	188
08/10/2021	1	0	8	18	58	0	1	0	0	1	...	0	0	0	0	1	1	0	0	0	188
09/10/2021	0	0	4	25	58	0	0	0	1	1	...	0	0	2	0	0	0	0	0	1	202
10/10/2021	0	1	6	23	46	1	0	5	0	0	...	0	0	0	0	0	0	1	1	0	204
11/10/2021	0	0	10	32	63	1	0	0	1	0	...	0	0	0	0	0	1	0	0	0	217
12/10/2021	1	0	11	33	61	1	0	0	1	0	...	0	0	0	0	0	0	0	0	0	218
13/10/2021	0	0	10	32	57	1	0	0	1	1	...	0	2	0	2	0	0	0	0	0	218
All	4	1	68	213	498	4	3	8	4	4	...	1	2	2	2	2	3	1	1	1	1796

10 rows × 179 columns

The flight fare of one the airlines is highest i.e 31785 Rs on dated 09.10.2021 and least flight fare is 5496 on 05.10.2021,07.10.2021,08.10.2021 and 12.10.2021.

```
plt.figure(figsize = (15, 10))
plt.title('Count of flights with different Airlines')
ax=sns.countplot(x = 'Airline_name', data =df)
plt.xlabel('Airline')
plt.ylabel('Count of flights')
```

```
Text(0, 0.5, 'Count of flights')
```



Vistara is having better connectivity of flights from Delhi to Mumbai as compared to other Airlines.

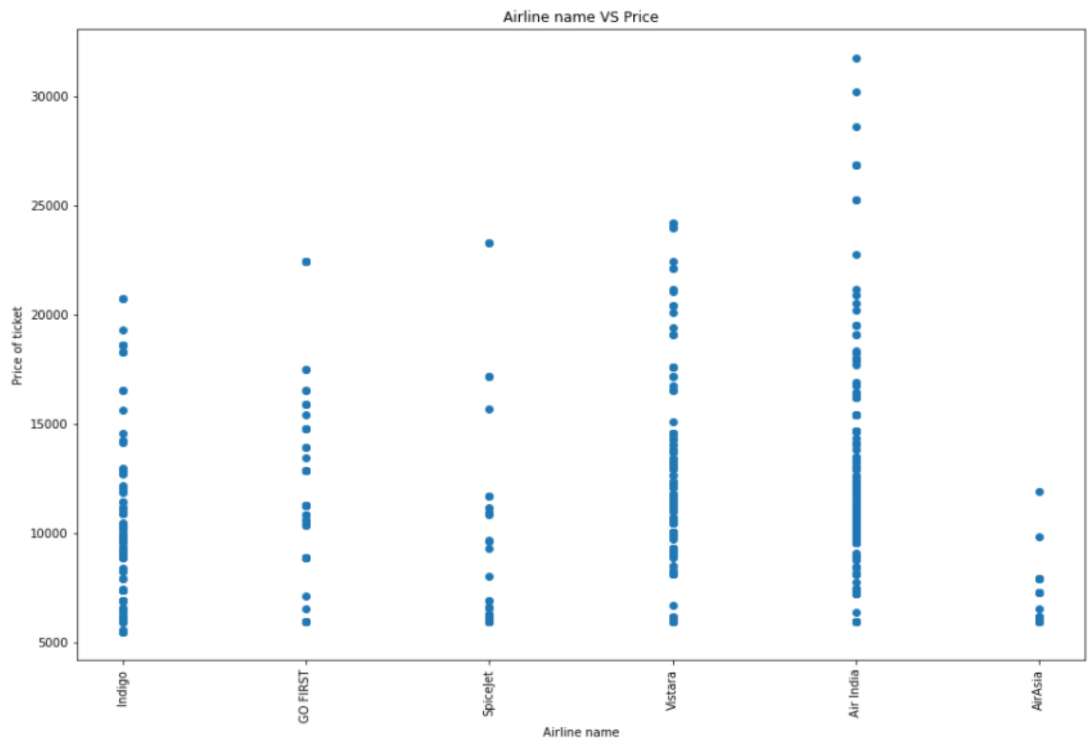
```
df['Airline_name'].value_counts()
```

```
Vistara      522
Air India    462
Indigo       362
GO FIRST     264
SpiceJet      99
AirAsia       87
Name: Airline_name, dtype: int64
```

The maximum flights i.e 522 from Delhi to Mumbai are of Vistara and the least flights are of AirAsia.

```python
plt.figure(figsize = (15, 10))
plt.title('Airline name VS Price')
plt.scatter(df['Airline_name'], df['Price'])
plt.xticks(rotation = 90)
plt.xlabel('Airline name')
plt.ylabel('Price of ticket')
```

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



The highest flight fare of Rs 31785 is of Airindia.

```python
pd.crosstab(df['Airline_name'],df['Price'],margins=True)
```

Price	5496	5607	5953	5954	5955	5956	6060	6165	6166	6271	...	22755	23281	24015	24225	25275	26850	28635	30210	31785	All
Airline_name																					
Air India	0	0	0	0	63	0	0	0	0	0	...	1	0	0	0	2	3	1	1	1	462
AirAsia	0	0	68	0	0	4	0	0	4	0	...	0	0	0	0	0	0	0	0	0	87
GO FIRST	0	0	0	213	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	264
Indigo	4	1	0	0	260	0	0	5	0	0	...	0	0	0	0	0	0	0	0	0	362
SpiceJet	0	0	0	0	72	0	3	0	0	4	...	0	2	0	0	0	0	0	0	0	99
Vistara	0	0	0	0	103	0	0	3	0	0	...	0	0	2	2	0	0	0	0	0	522
All	4	1	68	213	498	4	3	8	4	4	...	1	2	2	2	2	3	1	1	1	1796

7 rows × 179 columns

The least fare of Rs 5496 is of indigo airlines.

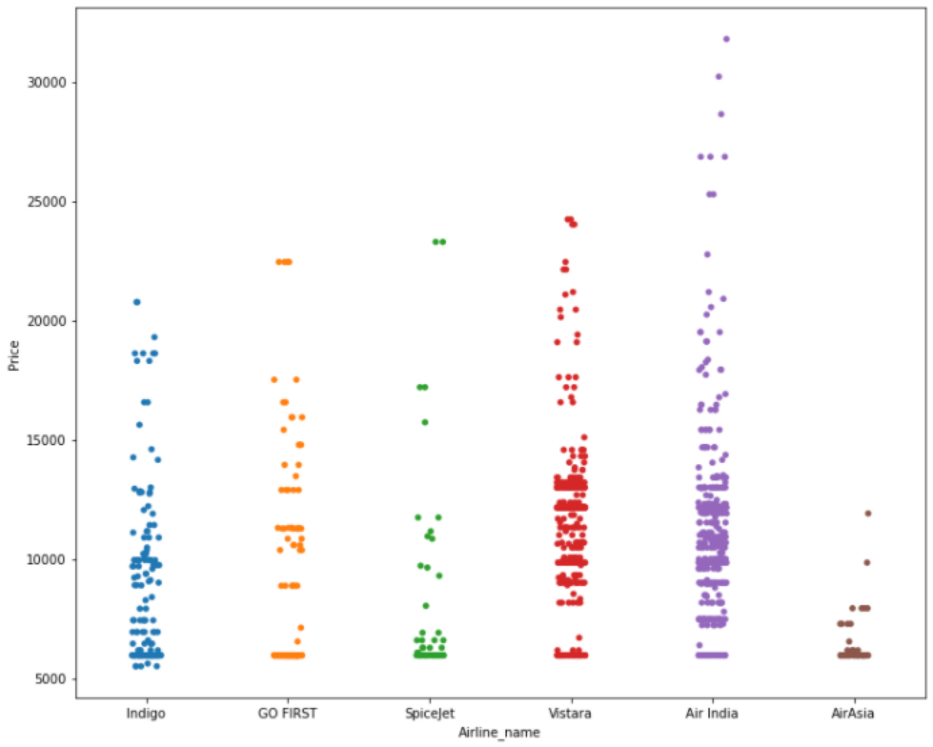
The flight fare is Rs 5955 for 72% of the records(260 out of 362)pertaining to indigo airlines.

The flight fare is Rs 5954 for 81% of the records (213 out of 264)pertaining to GO first airlines.

The flight fare is Rs 5953 for 78% of the records (68 out of 87)pertaining to AirAsia.

As most of the records(78%) for Airasia is having flight price of Rs 5953 so it can be said that airasia is cheapest as compared to others.

```
plt.figure(figsize=[12,10])
sns.stripplot(x='Airline_name',y='Price',data=df)
plt.show()
```

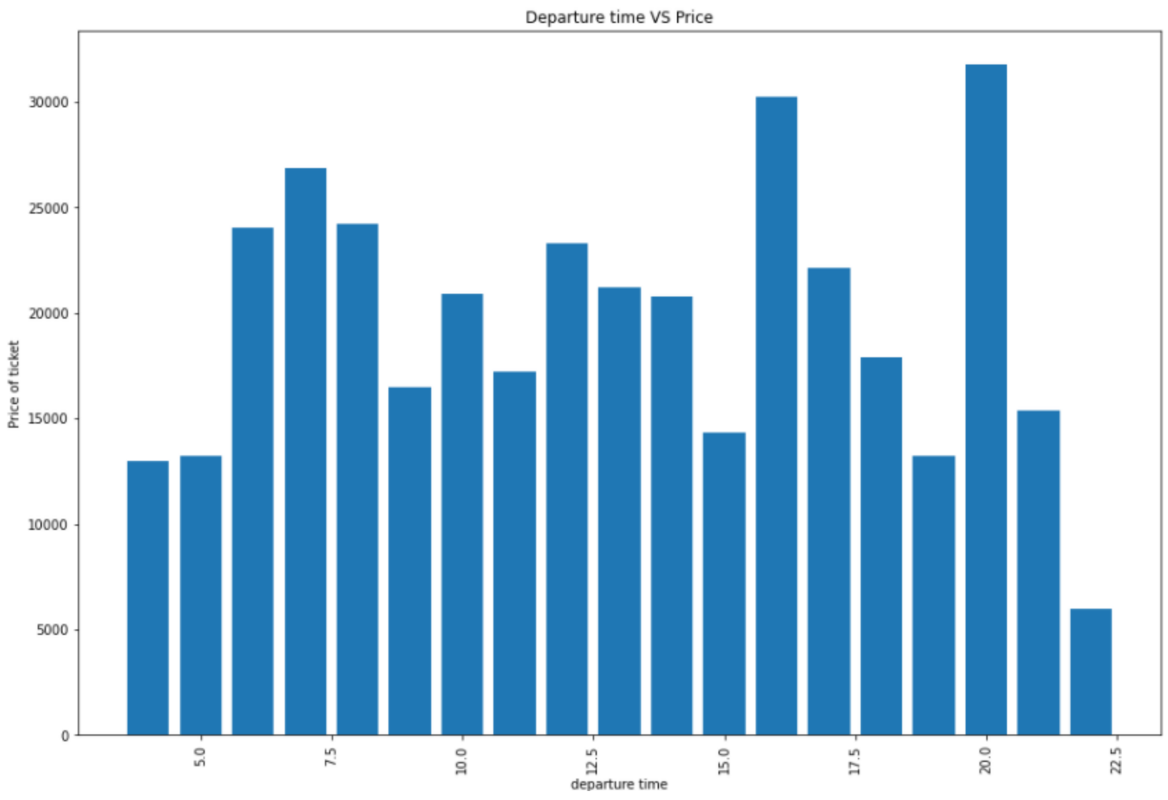


```
pd.crosstab(df['Airline_name'],df['Price']>20000,margins=True)
```

Price	False	True	All
Airline_name			
Air India	449	13	462
AirAsia	87	0	87
GO FIRST	260	4	264
Indigo	360	2	362
SpiceJet	97	2	99
Vistara	510	12	522
All	1763	33	1796

```python
plt.figure(figsize = (15, 10))
plt.title('Departure time VS Price')
plt.bar(df['depart_hour'], df['Price'])
plt.xticks(rotation = 90)
plt.xlabel('departure time')
plt.ylabel('Price of ticket')
```

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



```python
pd.crosstab(df['depart_hour'],df['Price'],margins=True)
```

Price	5496	5607	5953	5954	5955	5956	6060	6165	6166	6271	...	22755	23281	24015	24225	25275	26850	28635	30210	31785	All
depart_hour																					
4	0	0	3	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	10
5	0	0	11	12	4	0	0	0	4	0	...	0	0	0	0	0	0	0	0	0	66
6	4	1	0	13	38	0	1	0	0	0	...	0	0	1	0	0	0	0	0	0	107
7	0	0	0	5	25	0	0	0	0	0	...	0	0	1	0	0	1	0	0	0	90
8	0	0	13	40	44	0	0	0	0	0	...	0	0	0	2	0	0	0	0	0	188
9	0	0	4	18	54	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	134
10	0	0	0	20	41	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	100
11	0	0	8	9	17	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	50
12	0	0	8	13	32	0	0	0	0	2	...	0	2	0	0	0	0	0	0	0	121
13	0	0	0	5	41	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	133
14	0	0	0	6	19	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	94
15	0	0	0	6	21	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	39
16	0	0	9	10	23	0	0	0	0	1	...	1	0	0	0	0	2	0	1	0	115
17	0	0	0	6	25	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	134
18	0	0	8	9	42	0	2	0	0	1	...	0	0	0	0	0	0	0	0	0	84
19	0	0	0	13	26	4	0	3	0	0	...	0	0	0	0	0	0	0	0	0	93
20	0	0	0	11	34	0	0	2	0	0	...	0	0	0	0	2	0	1	0	1	160
21	0	0	4	12	12	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	73
22	0	0	0	5	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	5
All	4	1	68	213	498	4	3	8	4	4	...	1	2	2	2	2	3	1	1	1	1796

20 rows × 179 columns

5 flights are available with departure time of 10 pm and all of them are having same flight fare of 5954 Rs.

The flights with departure time i.e red eye flights early in the morning from 5 am to 6am are comparatively cheaper. Further, the flight fare is below 6000 Rs for 52%(56 out of 107) of the records having departure time 6 am.

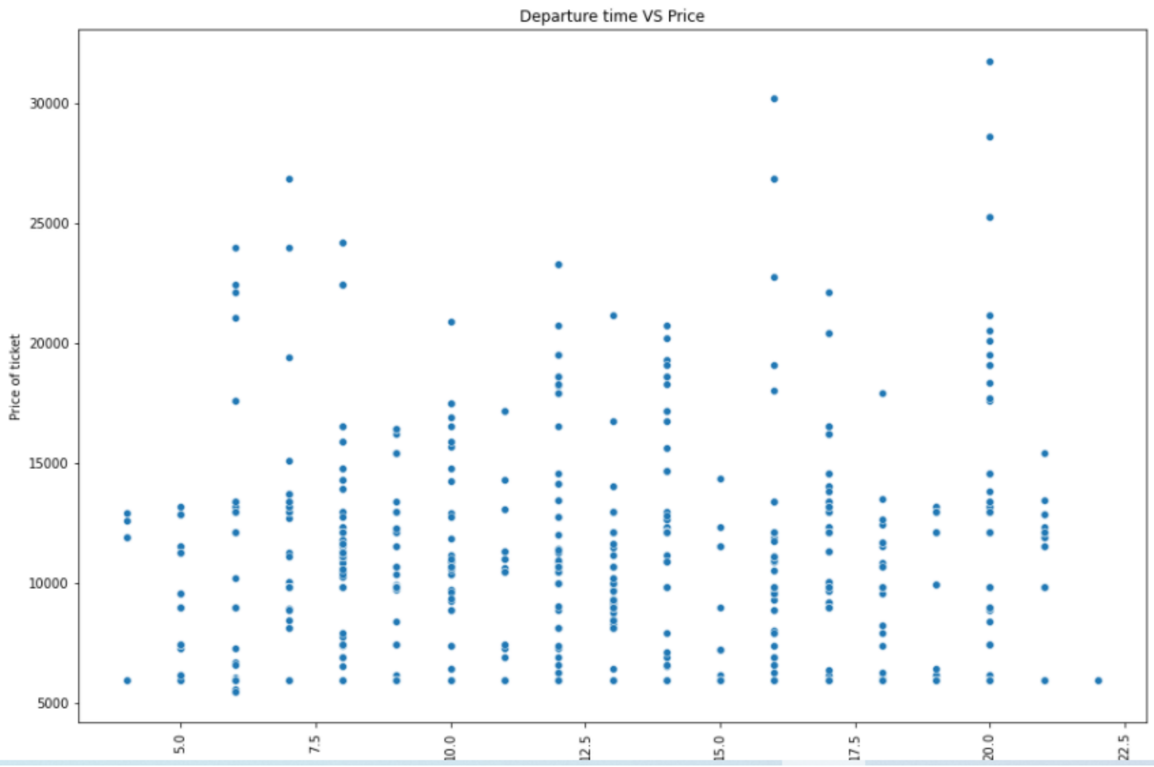
188 flights are there with departure time of 8 am from source i.e Delhi. Further, as number of flights are more it can be said it is a peak time for travelling.

One of the flight having departure time of 8 pm is costliest where the fare is 31785 Rs.

```python
plt.figure(figsize = (15, 10))
plt.title('Departure time VS Price')
sns.scatterplot(df['depart_hour'], df['Price'])
plt.xticks(rotation = 90)
plt.xlabel('departure time')
plt.ylabel('Price of ticket')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyw
gs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
keyword will result in an error or misinterpretation.
  warnings.warn(
```

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

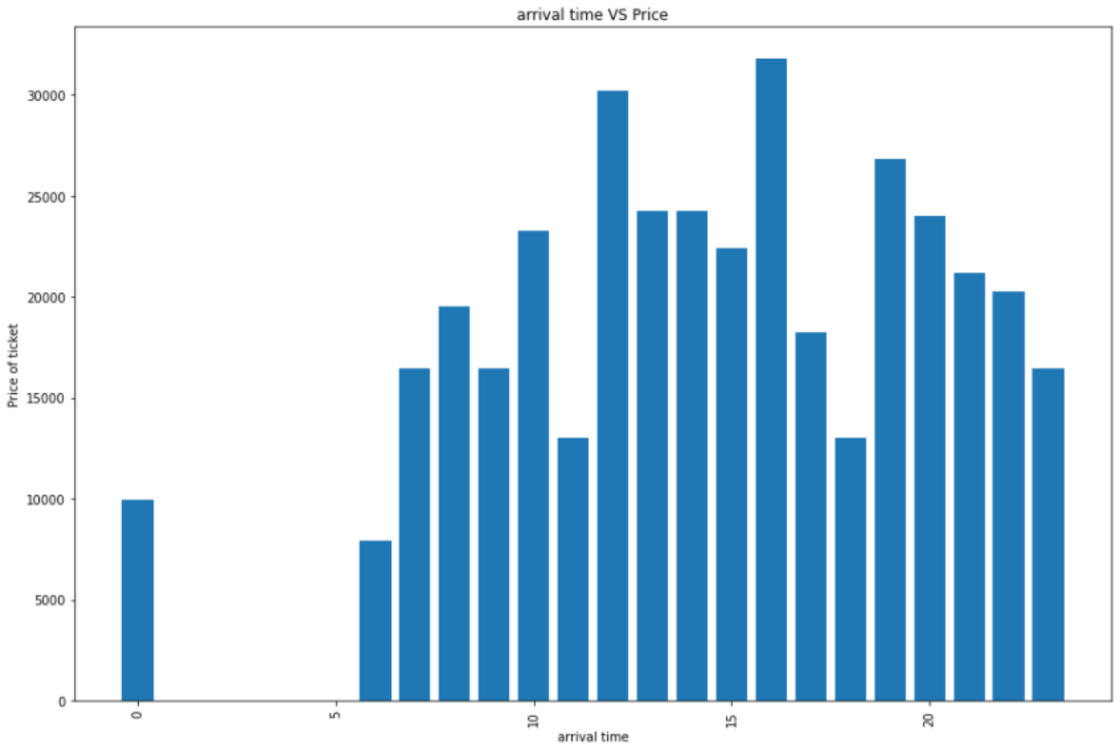


There are no flights available after 11 pm from source location.

The frequency of flights reduces drastically from source location after 9 pm indicating non peak hours for travelling.

```python
plt.figure(figsize = (15, 10))
plt.title('arrival time VS Price')
plt.bar(df['arrival_hour'], df['Price'])
plt.xticks(rotation = 90)
plt.xlabel('arrival time')
plt.ylabel('Price of ticket')
```

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



There are no flights arriving at destination i.e Mumbai in your case between 12:30 pm to 5 am.

```python
pd.crosstab(df['arrival_hour'],df['Price'],margins=True)
```

Price arrival_hour	5496	5607	5953	5954	5955	5956	6060	6165	6166	6271	...	22755	23281	24015	24225	25275	26850	28635	30210	31785	All
0	0	0	0	5	13	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	28
6	0	0	14	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	17
7	0	0	0	12	4	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	45
8	0	0	0	11	17	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	89
9	0	0	0	6	23	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	84
10	0	0	0	1	24	0	0	0	0	3	...	0	1	0	0	0	0	0	0	0	91
11	0	0	4	9	34	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	82
12	4	1	0	0	27	0	0	0	0	0	...	1	0	0	0	2	2	1	1	0	67
13	0	0	0	11	27	0	0	0	0	0	...	0	1	0	1	0	0	0	0	0	109
14	0	0	7	5	45	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	93
15	0	0	0	20	14	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	111
16	0	0	12	8	17	0	0	0	0	1	...	0	0	0	0	0	0	0	0	1	106
17	0	0	0	14	58	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	133
18	0	0	0	37	24	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	94
19	0	0	0	17	29	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	86
20	0	0	23	9	28	0	2	1	0	0	...	0	0	2	0	0	0	0	0	0	141
21	0	0	0	9	42	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	135
22	0	0	8	28	49	4	0	2	4	0	...	0	0	0	0	0	0	0	0	0	123
23	0	0	0	11	23	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	162
All	4	1	68	213	498	4	3	8	4	4	...	1	2	2	2	2	3	1	1	1	1796

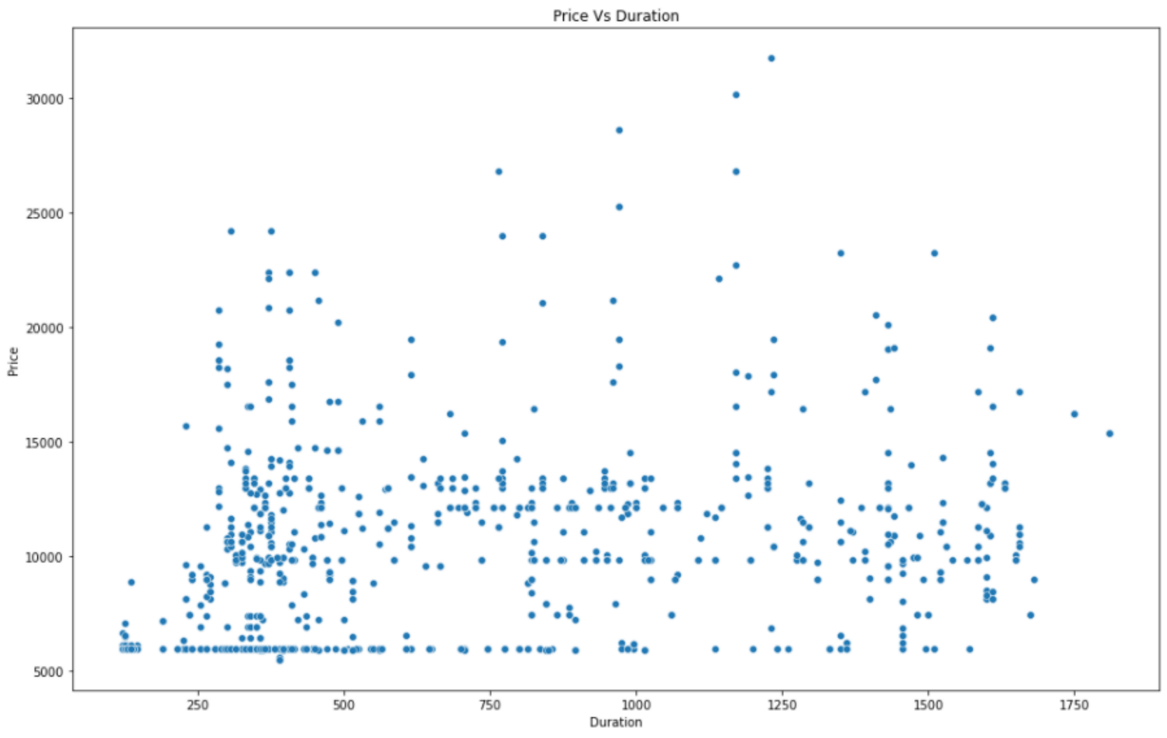
20 rows × 179 columns

162 flights are having arrival time of 4pm at destination.

17 flights are having arrival time of 6 am at destination which points out to the fact that the frequency of the flights reduces drastically from source after 8 pm.

The flight fare is below 6000 Rs for 82%(14 out of 17) of the records having arrival time 6 am.

```
plt.figure(figsize=[16,10])
plt.title("Price Vs Duration")
sns.scatterplot(x='Duration',y='Price',data=df2)
plt.show()
```



As such no relationship can be established between the independent variable (Duration)amd dependent variable (Price).

Time taken by flights to reach to the destination ranges from 120 minutes to 1810 minutes.

```
df2['total_stops'].value_counts()
```

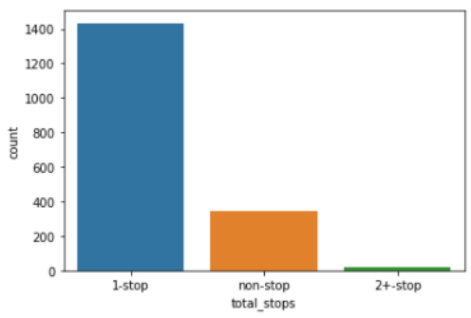
```
1-stop      1433
non-stop     345
2+-stop       18
Name: total_stops, dtype: int64
```

Most of the flights i.e are having 1 stop during the journey between source and destination amd the least records i.e 0.01%(18 out of 1796) are for the flights having more than 2 stops.

```
sns.countplot(df2['total_stops'])
```

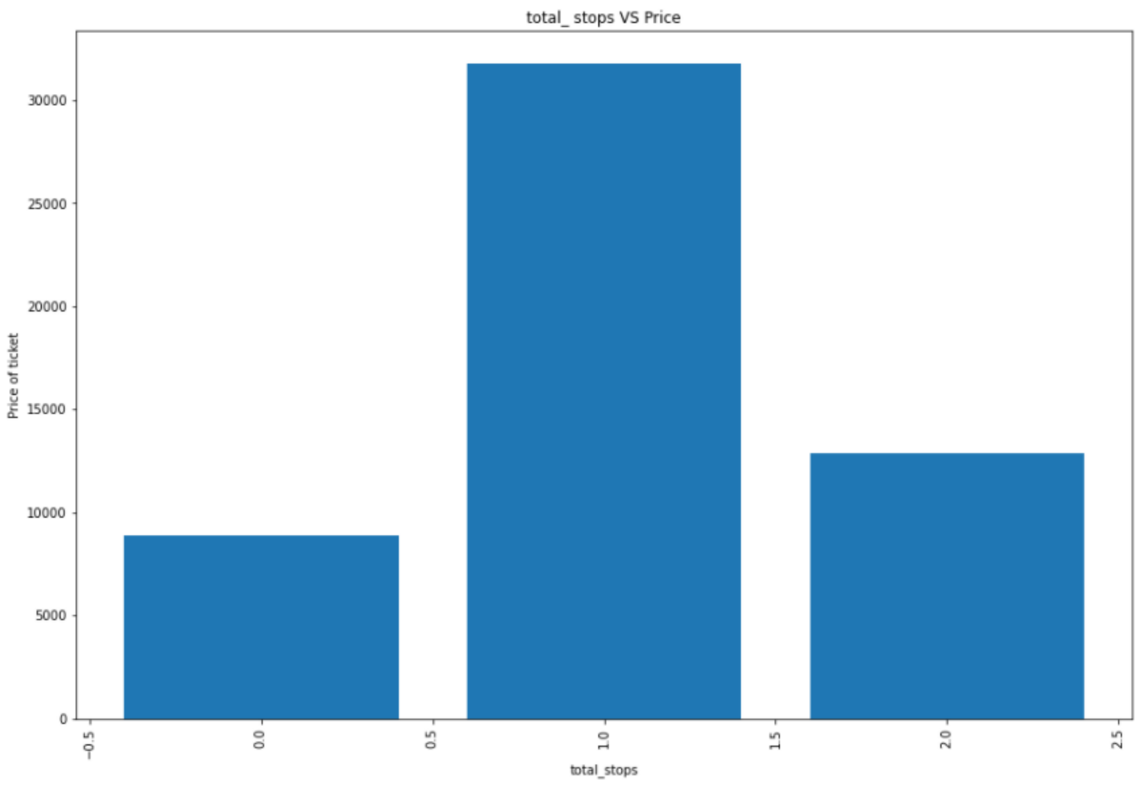
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='total_stops', ylabel='count'>
```



```
plt.figure(figsize = (15, 10))
plt.title('total_ stops VS Price')
plt.bar(df2['total_stops'], df2['Price'])
plt.xticks(rotation = 90)
plt.xlabel('total_stops')
plt.ylabel('Price of ticket')
```

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



Non stops flights are cheaper as compared to the flights having one stop.

```
pd.crosstab(df2['total_stops'],df2['Price'],margins=True)
```

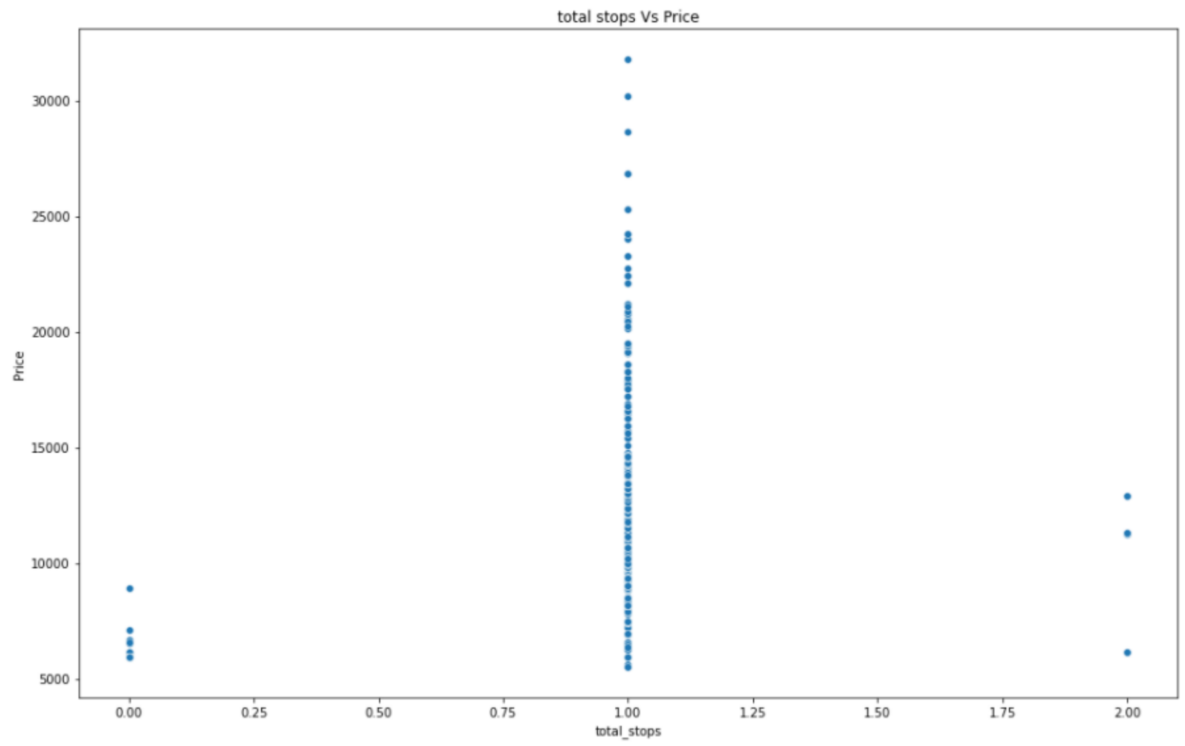
Price	5496	5607	5953	5954	5955	5956	6060	6165	6166	6271	...	22755	23281	24015	24225	25275	26850	28635	30210	31785	All
total_stops																					
0	0	0	0	58	267	4	3	8	0	0	...	0	0	0	0	0	0	0	0	0	345
1	4	1	68	155	231	0	0	0	0	4	...	1	2	2	2	2	3	1	1	1	1433
2	0	0	0	0	0	0	0	0	4	0	...	0	0	0	0	0	0	0	0	0	18
All	4	1	68	213	498	4	3	8	4	4	...	1	2	2	2	2	3	1	1	1	1796

4 rows × 179 columns

There are 4 flights with one stop having minimum fare of 5496 Rs.

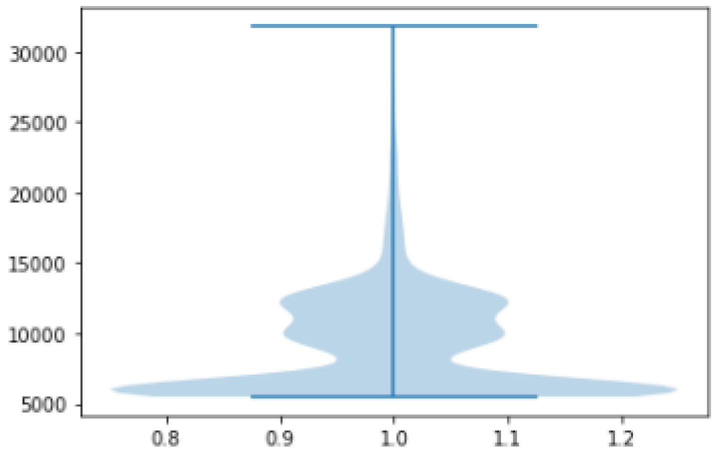
There is a single flight with one stop having maximum fare of 31785 Rs.

```python
plt.figure(figsize=[16,10])
plt.title("total stops Vs Price")
sns.scatterplot(x='total_stops',y='Price',data=df2)
plt.show()
```

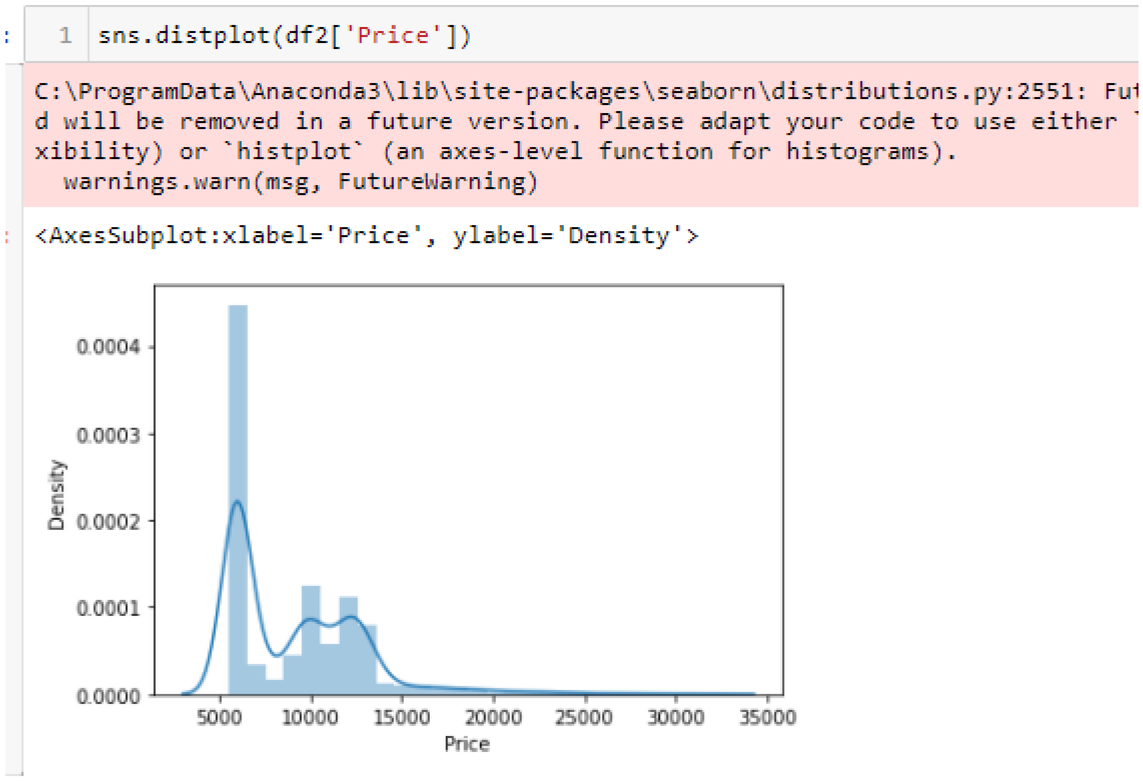


```python
plt.violinplot(df2['Price'])
```

```
{'bodies': [<matplotlib.collections.PolyCollection at 0x24aea788eb0>],
 'cmaxes': <matplotlib.collections.LineCollection at 0x24aeb2de9d0>,
 'cmins': <matplotlib.collections.LineCollection at 0x24aeb2a3820>,
 'cbars': <matplotlib.collections.LineCollection at 0x24aea7fae20>}
```



The feature price is not uniformly distributed as per the above plot.



```
1  sns.distplot(df2['Price'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fut
d will be removed in a future version. Please adapt your code to use either `
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

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

The dependent feature price whcih is the target variable is having right tailed skewness as mean is greater than median but as it is target variable there is no point in removing skewness from it.

- Interpretation of the Results

  From visualization the key findings like skewness, outliers present in the dataset were known. Further, the heat map was drawn showing the correlation with target feature and based on it the one's having less/no correlation with the target feature can be known and if the need arises can be dropped.

  Vistara is having better connectivity of flights(522) from Delhi to Mumbai as compared to other Airlines and the highest flight fare of Rs 31785 is of AirIndia.

  As most of the records(78%) for Airasia is having fare of Rs 5953 so it can be said that airasia is cheapest as compared to others.

The least ticket fare is of Rs 5496 for one of the flight of indigo airlines. Further, there are in total 8 records for Air india having fare of Rs 25000 and above.

The flights with departure time i.e red eye flights early in the morning from 5 am to 6am are comparatively cheaper. Further, the flight fare is below 6000 Rs for 52%(56 out of 107) of the records having departure time 6 am.

There are no flights available after 11pm from source location and the frequency of flights reduces drastically from source location after 9 pm.

The flight fare is below 6000 Rs for 82%(14 out of 17) of the records having arrival time 6 am.

The non stop flights are cheaper as compared to one stop flights.

CONCLUSION

- Key Findings and Conclusions of the Study

  KNeighborsClassifier is my best model with r2 score of 58%

  .

- Learning Outcomes of the Study in respect of Data Science

  This case helped me how to hyper tune the parameters using RandomizedSeachCV to get the best accuracy from the model.

- Limitations of this work and Scope for Future Work

  More routes can be added and the same analysis can be expanded to major airports and travel routes in India.

  The analysis can be done by increasing the data points and increasing the historical data used . This will train the model better giving better accuracies and more savings.

More rules can be added in the Rule based learning based on our understanding of the industry, also incorporating the offer periods given by the airline.

Develop a more user friendly interface for various routes giving more flexibility to the users.