Flight Price Prediction

Data Source: https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction

importing the primary libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

(1) **Data Loading**

```
flight_data=pd.read_csv('/content/drive/MyDrive/Clean_Dataset.csv')
```

```
# reading the 1st 3 rows of the dataset
flight_data.head(3)
```

	Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	desti
0	0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	

As the column Unnmed: 0 is not needed, it is dropped

```
flight_data=flight_data.drop(columns=['Unnamed: 0'])
```

Reading the dataset

```
# reading the 1st 3 rows of the dataset
flight_data.head(3)
```

	airline	flight	source_city	departure_time	stops	arrival_time	destination_cit
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumba
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumba

```
# reading the last 3 rows of the dataset
flight_data.tail(3)
```

	airline	flight	source_city	departure_time	stops	arrival_time	destinatio
300150	Vistara	UK-832	Chennai	Early_Morning	one	Night	Hyc
300151	Vistara	UK-828	Chennai	Early_Morning	one	Evening	Hyc

(2) **Data Preprocessing**

Dimensions of the dataset

```
flight_data.shape
```

```
(300153, 11)
```

Checking the data types for each column

```
flight_data.dtypes
```

```
airline           object
flight            object
source_city       object
departure_time    object
stops             object
```

```
arrival_time        object
destination_city    object
class               object
duration           float64
days_left            int64
price                int64
dtype: object
```

Checking for null, missing or duplicate values in the dataset.

```
print('Null values:',flight_data.isnull().any().sum())
print('NaN values:', flight_data.isna().any().sum())
print('duplicates:',flight_data.duplicated().any().sum())
```

```
Null values: 0
NaN values: 0
duplicates: 0
```

(3) **Exploratory Data Analysis**

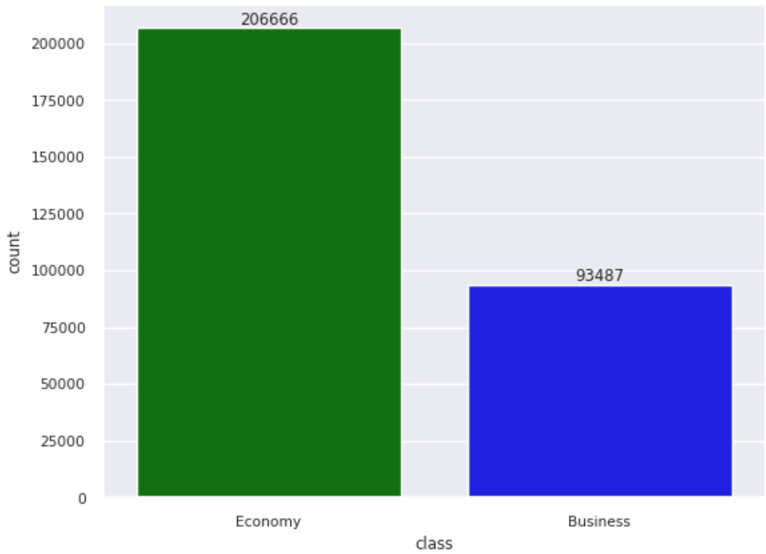
a. Checking for no.of distinct values in each column in the dataset

```
flight_data.nunique()
```

```
airline                 6
flight               1561
source_city             6
departure_time          6
stops                   3
arrival_time            6
destination_city        6
class                   2
duration              476
days_left              49
price               12157
dtype: int64
```

b. No.of flights per class - Economy and Business

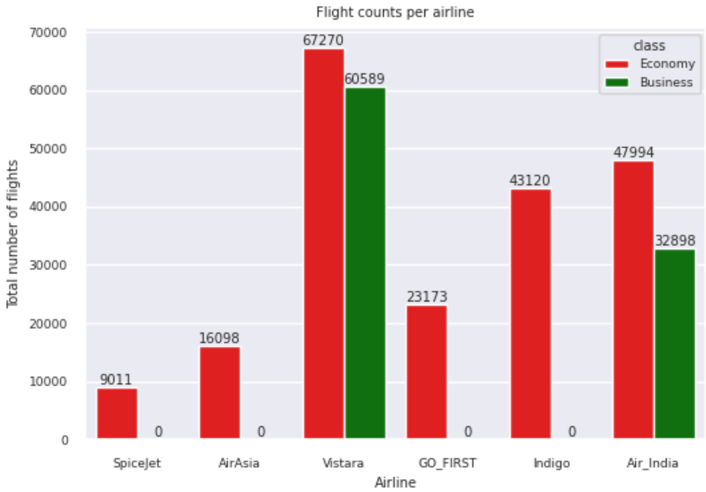
```
sns.set(font_scale=0.7)
cl={'Economy':'green','Business':'blue'}
c=sns.countplot(data=flight_data,x='class',palette=cl)
for label in c.containers:
  c.bar_label(label)
```



c. Total number of flights under each Airline and class

```
sns.set(font_scale=0.6)
plt.figure(figsize=(6,4))
col={'Economy':'red','Business':'green'}
a=sns.countplot(data=flight_data,x='airline',hue='class',palette=col)
for l in a.containers:
  a.bar_label(l)
plt.title('Flight counts per airline')
plt.xlabel('Airline')
plt.ylabel('Total number of flights')
```

```
Text(0, 0.5, 'Total number of flights')
```



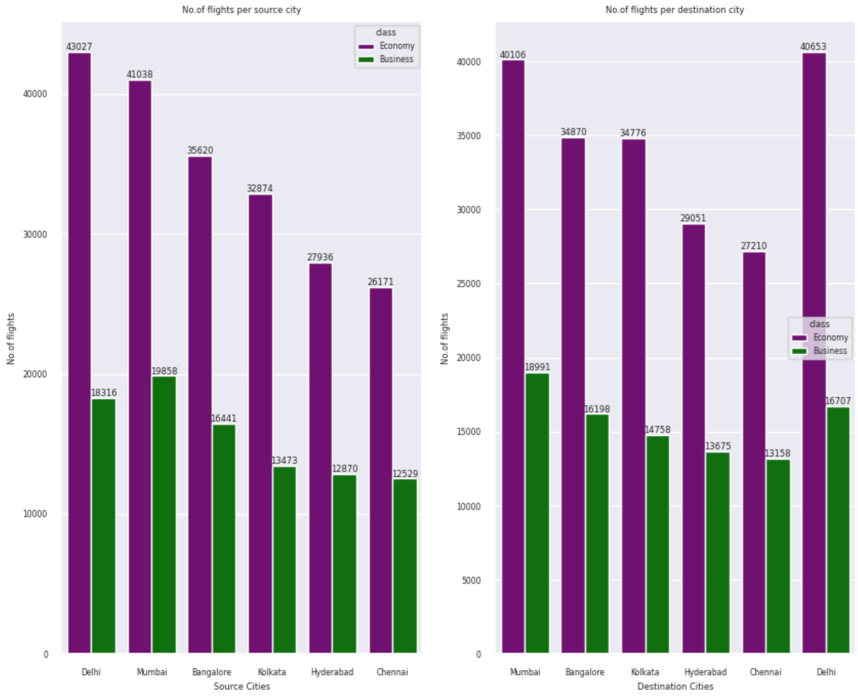
1. Among the six airlines, only Vistara and Air India have both classes Economy and Business
2. And the airline Vistara has the highest no.of flights from both classes
3. Spicejet is the airline which has lowest no.of flights

d. Plotting No.of flights per cities and class category

```
sns.set(font_scale=0.5) # setting the font scale
plt.figure(figsize=(10,8)) # setting the chart size

plt.subplot(1,2,1) # 1st plot in the subplot
col={'Economy':'purple','Business':'green'}
ax=sns.countplot(data=flight_data,x='source_city',hue='class',palette=col)
plt.title('No.of flights per source city')
plt.xlabel('Source Cities')
plt.ylabel('No.of flights')
for label in ax.containers:
    ax.bar_label(label) # adding label to the bars

plt.subplot(1,2,2) # 2nd plot in the sub plot
col={'Economy':'purple','Business':'green'}
bx=sns.countplot(data=flight_data,x='destination_city',hue='class',palette=col)
sns.move_legend(bx,"right")
plt.title('No.of flights per destination city')
plt.xlabel('Destination Cities')
plt.ylabel('No.of flights')
for c in bx.containers:
  bx.bar_label(c)
plt.show()
```



From both charts,

- Economy class:- Delhi has the highest number, and
- Business class:- Mumbai is the city with highest no.of flights

e. Statistical info of the dataset

```
flight_data.describe()
```

	duration	days_left	price
count	300153.000000	300153.000000	300153.000000
mean	12.221021	26.004751	20889.660523
std	7.191997	13.561004	22697.767366
min	0.830000	1.000000	1105.000000
25%	6.830000	15.000000	4783.000000
50%	11.250000	26.000000	7425.000000
75%	16.170000	38.000000	42521.000000
max	49.830000	49.000000	123071.000000

f. Viewing ticket price by each airline and class

```
flight_data[['airline','price','class']].sort_values(by='price',ascending=False)
```

	airline	price	class
261377	Vistara	123071	Business
216096	Vistara	117307	Business
215859	Vistara	116562	Business
277345	Vistara	115211	Business
270999	Vistara	114705	Business
...	...	...	...
204375	AirAsia	1105	Economy
204376	GO_FIRST	1105	Economy
206598	Indigo	1105	Economy
206599	Indigo	1105	Economy
205024	Indigo	1105	Economy

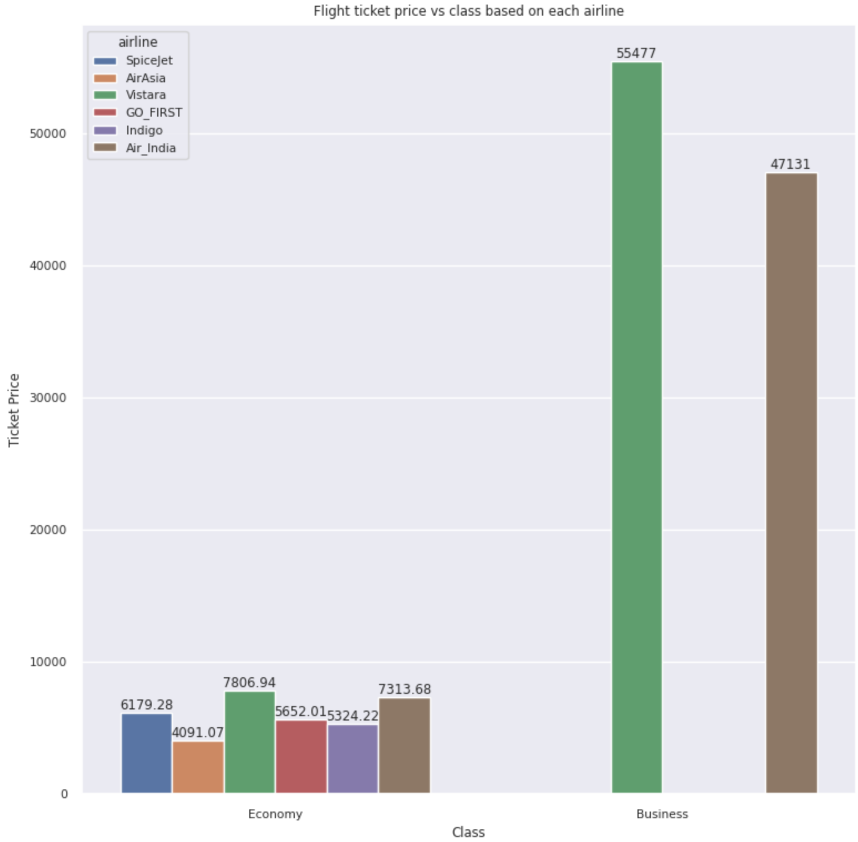
300153 rows × 3 columns

Among the various airlines, Vistara charges highest price under the business class.

g. Ticket price vs class based on different airlines

```
sns.set(font_scale=0.7)
plt.figure(figsize=(9,9))
x=sns.barplot(data=flight_data,x='class',y='price',hue='airline',errorbar=None)
for i in x.containers:
  x.bar_label(i)
plt.xlabel('Class')
plt.ylabel('Ticket Price')
plt.title('Flight ticket price vs class based on each airline')
```

```
Text(0.5, 1.0, 'Flight ticket price vs class based on each airline')
```



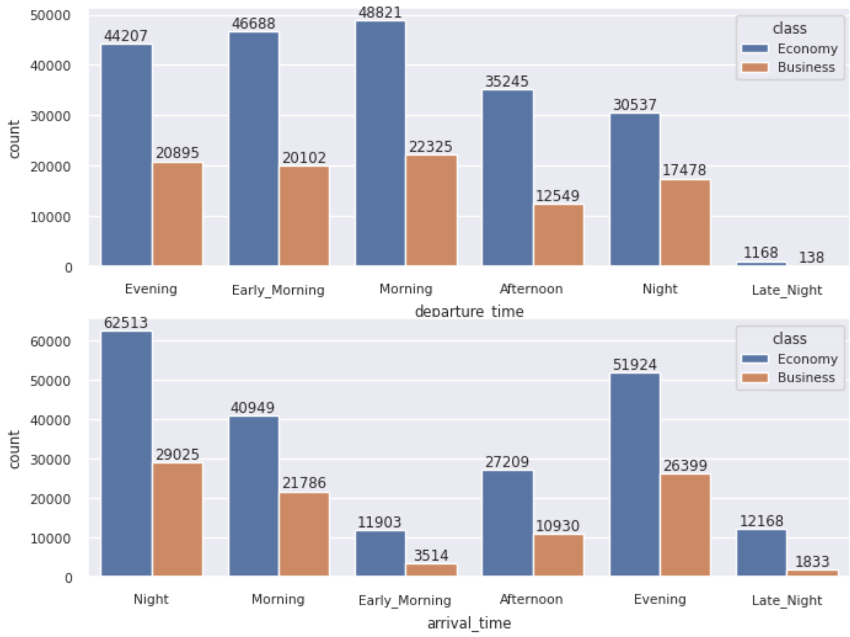
The ticket price charged by Vistara is the highest under both classes, and AirAsia offers the lowest under Economy class.

h. Plotting No.of flights per class under different departure and arrival time.

```
sns.set(font_scale=0.7)
plt.figure(figsize=(8,6))

plt.subplot(2,1,1)
cl=sns.countplot(data=flight_data,x='departure_time',hue='class')
for l in cl.containers:
  cl.bar_label(l)

plt.subplot(2,1,2)
cl=sns.countplot(data=flight_data,x='arrival_time',hue='class')
for l in cl.containers:
  cl.bar_label(l)
```



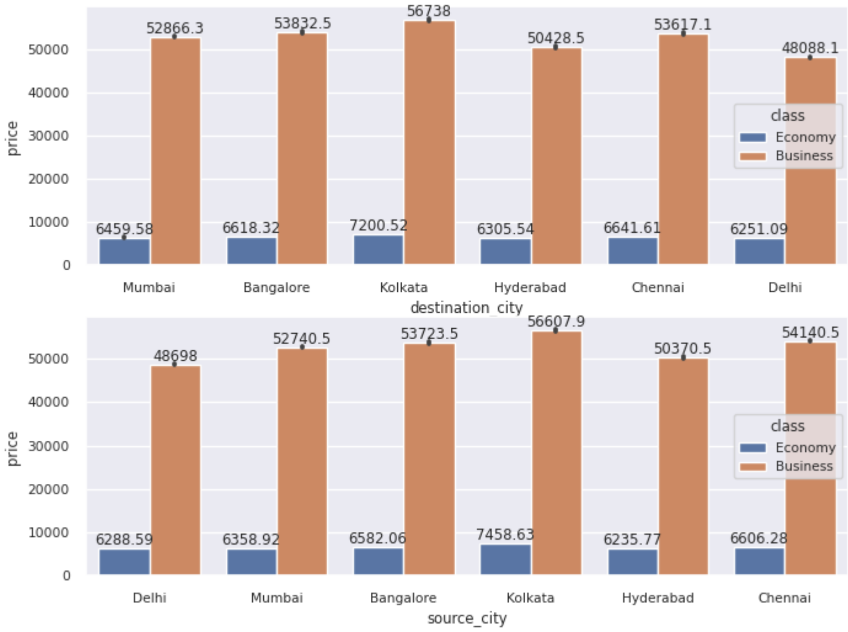
This graph shows that, more morning flights are departed as well as more night flights arrive at the airport.

i. Analysing ticket price vs destination and source cities base on each class

```
sns.set(font_scale=0.7)
plt.figure(figsize=(8,6))

plt.subplot(2,1,1)
cl=sns.barplot(data=flight_data,x='destination_city',y='price',hue='class')
for l in cl.containers:
  cl.bar_label(l)

plt.subplot(2,1,2)
cl=sns.barplot(data=flight_data,x='source_city',y='price',hue='class')
for l in cl.containers:
  cl.bar_label(l)
```



Kolkata's flight is the costliest

j. Analysing duration of flights

```
flight_data['duration'].describe()
```

```
count    300153.000000
mean         12.221021
std           7.191997
min           0.830000
25%           6.830000
50%          11.250000
75%          16.170000
max          49.830000
Name: duration, dtype: float64
```

```
# Row numbers of flights with minimum duration
flight_data[flight_data['duration']== 49.830000].index
```

```
Int64Index([193889, 194359], dtype='int64')
```

```
# Row numbers of flights with maximum duration
flight_data[flight_data['duration']== 0.830000].index
```

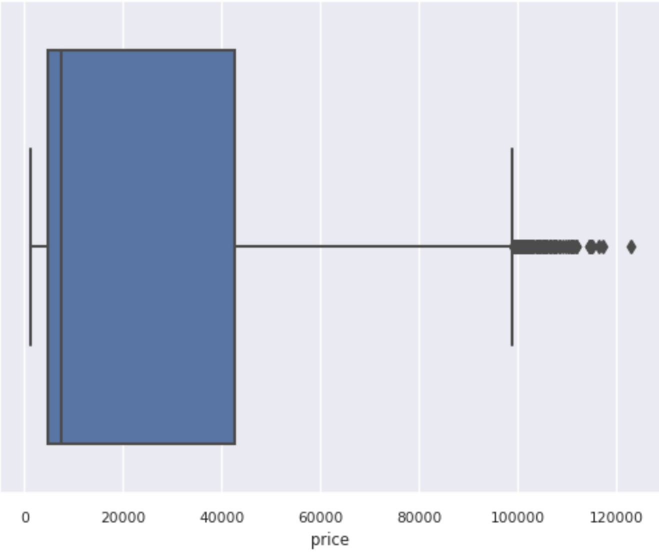
```
Int64Index([115869, 115943, 116010, 116081, 116163, 116236, 116322, 116411,
            116496, 116656, 116835, 116924, 117019, 117101, 117190, 117274,
            117366, 117461, 117547, 117643, 117728, 117817, 117900, 117995,
            118086, 118173, 118269, 118355, 118445, 118528, 118622, 118712,
            118799, 118896, 118982, 119072, 119155, 197354, 197355, 197356,
            197445, 197446, 197447, 197537, 197538, 197539, 197626, 197627,
            197628, 197712, 197713, 197724],
           dtype='int64')
```

(4) Feature Engineering

1. Checking for outliers in price column

```
sns.boxplot(data=flight_data,x='price')
```

```
<Axes: xlabel='price'>
```

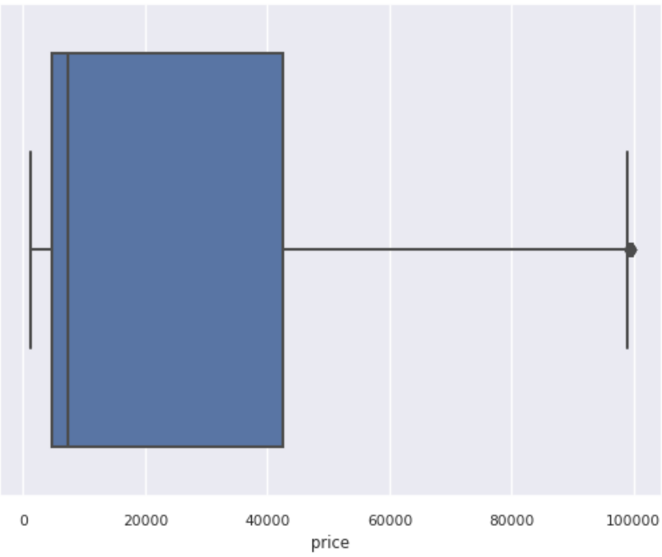


From the boxplot, we can infer that, the flight ticket price falls in the range of 0 to 100000 only, whereas there are few outliers that is beyond the value of 120000. Since, the dataset is large enough, the outliers are removed from the data in order to develop a proper model for the prediction.

```
f_out=flight_data[flight_data['price']>=100000].index
flight_data=flight_data.drop(index=f_out)
```

```
sns.boxplot(x=flight_data['price'])
```

```
<Axes: xlabel='price'>
```




```
flight_data.shape
```

```
(300045, 11)
```

```
flight_data[['destination_city','price']].groupby('destination_city').max()
```

	price
destination_city	
Bangalore	99403
Chennai	99577
Delhi	98543
Hyderabad	99677
Kolkata	99129
Mumbai	99680

```
flight_data[flight_data['price']==99680]
```

	airline	flight	source_city	departure_time	stops	arrival_time	destinatio
248968	Vistara	UK-	Bangalore	Night	one	Morning	

```
flight_data.head(2)
```

	airline	flight	source_city	departure_time	stops	arrival_time	destination_cit
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumba

Vistara offers Business Class at the highest ticket price to the city Mumbai flies from Bangalore with duration of 14.42 at Rs 99680.

2. Removing unnecessary columns

```
flight_data=flight_data.drop(columns='flight')
```

3. Encoded multi columns containing categorical varibles at once

```
from sklearn.preprocessing import LabelEncoder
```

```
df=flight_data.iloc[:,:7] # poisition of columns that have categorical variables

# Encoding:
enc_all_cols=df.apply(LabelEncoder().fit_transform)


#Concating with the remaining columns of the dataset
df_enc=pd.concat([enc_all_cols,flight_data.iloc[:,-3:]],axis=1)

# reading the first 2 rows of the dataframe which now has encoded data and ready for train test split
df_enc.head(2)
```

	airline	source_city	departure_time	stops	arrival_time	destination_city	class
0	4	2	2	2	5	5	1
1	4	2	1	2	4	5	1

(5) Model Building

Train test split

```
from sklearn.model_selection import train_test_split

X = df_enc.drop(columns='price') # feature
y=df_enc['price']  # target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
print('X_train size: {}, X_test size: {}'.format(X_train.shape, X_test.shape))
print('y_train size: {}, y_test size: {}'.format(y_train.shape, y_test.shape))
```

```
X_train size: (240036, 9), X_test size: (60009, 9)
y_train size: (240036,), y_test size: (60009,)
```

Finding the best model with the help of GridSearchCV

```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

model_params={
    'LR':{
        'model':LinearRegression(),
        'params':{

        }
    },
    'KNR':{
        'model':KNeighborsRegressor(),
        'params':{
            'n_neighbors':[2,5,10]
        }
    },
    'RFR':{
        'model':RandomForestRegressor(),
        'params':{
            'n_estimators':[5,10,20]
        }
    }
}
```

```
from sklearn.model_selection import ShuffleSplit
scores=[]
cv = ShuffleSplit(n_splits=5, test_size=0.20, random_state=0)
for model,mp in model_params.items():
  clf=GridSearchCV(mp['model'],mp['params'],cv=cv,return_train_score=False)
  clf.fit(X,y)
  scores.append({
      'model':model,
      'best score':clf.best_score_,
      'best params':clf.best_params_
  })
```

```
dd=pd.DataFrame(scores,columns=['model','best score','best params'])
dd
```

	model	best score	best params
0	LR	0.906194	{}
1	KNR	0.710043	{'n_neighbors': 5}
2	RFR	0.985239	{'n_estimators': 20}

Among the 3 models used, Random Forest Regressor gives the highest score.

Hence, a model with the Random Forest Regression is built and evaluated.

```
from sklearn.model_selection import cross_val_score
cv=ShuffleSplit(n_splits=5,test_size=0.2)
s=cross_val_score(RandomForestRegressor(n_estimators=20),X,y,cv=cv)
print('Average Accuracy : {}%'.format(round(sum(s)*100/len(s)), 3))
```

```
Average Accuracy : 99%
```

As per the model evaluation, the prediction is around 99% accurate. Therefore, for flight prediction, 'rf' the model is chosen.

```
rf=RandomForestRegressor(n_estimators=20)
```

```
rf.fit(X_train,y_train)
```

```
▾        RandomForestRegressor
RandomForestRegressor(n_estimators=20)
```

```
r_pred=rf.predict(X_test)
```

evaluating the model

```
from sklearn import metrics
```

```
metrics.r2_score(r_pred,y_test)
```

```
0.9850582127303147
```

Looking for the labels of the categorical columns- For reference (Since the columns are encoded)

```
q=pd.DataFrame(data=['Vistara','Air_India','Indigo','GO_FIRST','AirAsia','SpiceJet'],columns=['Airline Name'])
q['Code']=[5,1,3,2,0,4]
q['Source City']=['Delhi','Mumbai','Bangalore','Kolkata','Hyderabad','Chennai']
q['Code_S']=[2,5,0,4,3,1]
q['Destination City']=['Mumbai','Delhi','Bangalore','Kolkata','Hyderabad','Chennai']
q['Code_D']=[5,2,0,4,3,1]
print(q)
```

```
  Airline Name  Code Source City  Code_S Destination City  Code_D
0      Vistara     5       Delhi       2           Mumbai       5
1    Air_India     1      Mumbai       5            Delhi       2
2       Indigo     3   Bangalore       0        Bangalore       0
3     GO_FIRST     2     Kolkata       4          Kolkata       4
4      AirAsia     0   Hyderabad       3        Hyderabad       3
5     SpiceJet     4     Chennai       1          Chennai       1
```