PROJECT REPORT

TITLE:-

FLIGHT PRICE PREDICTION

TEAM MEMBERS:-

20BCE7272 – Batchu Priyanka

priyanka.20bce7272@vitap.ac.in

20BCI7149 – Veeranki Yasaswini

yasaswini.20bci7149@vitap.ac.in

20BCE7345 – Chaparala Sri Manasa

manasa.20bce7345@vitap.ac.in

20BCI7295 – Bollineni Sri Tonya

sritonya.20bci7295@vitap.ac.in

TEAM NUMBER:- 388

INDEX

1 INTRODUCTION

1.1 Overview:

People who often travel by plane will be more knowledgeable about the greatest discounts and the ideal time to purchase a ticket. Many airline companies adjust their costs based on the season or the duration of the flight. When individuals travel more, the price will rise. Estimating the highest costs of airline data for the route with parameters such as Duration, Source, Destination, Arrival, and Departure. Features are drawn from a selected dataset, and the price of plane tickets varies over time. We used Linear Regression, Decision Trees, Random Forest, Gradient Boost Regressor and Support Vector Regression algorithms to estimate flight prices for consumers. Random Forest has the highest accuracy of 83.8% in forecasting flight prices. We have also done the metrics for the statistical analysis.

1.2 Purpose:

Both travellers and the travel industry can profit from the flight price prediction initiative. It assists consumers in making educated decisions about when to book flights by analysing previous data and taking into account aspects such as seasonality and demand. By selecting the most cost-effective times to purchase tickets, travellers may save significantly. Furthermore, it helps customers to more efficiently plan their journeys, securing convenient schedules and optimising their travel budget. Insights from the study can also help the travel sector understand customer behaviour, market trends, and price dynamics, resulting in improved business strategies and marketing campaigns. Furthermore, including flight price prediction into travel platforms can improve customer experience by providing personalised ticket suggestions. Overall, the flight price prediction project provides significant information and tools for travellers, businesses, and analysts, allowing them to make better decisions, save money, and acquire insights into the travel industry's dynamics..

2 LITERATURE SURVEY

2.1 Existing problem:

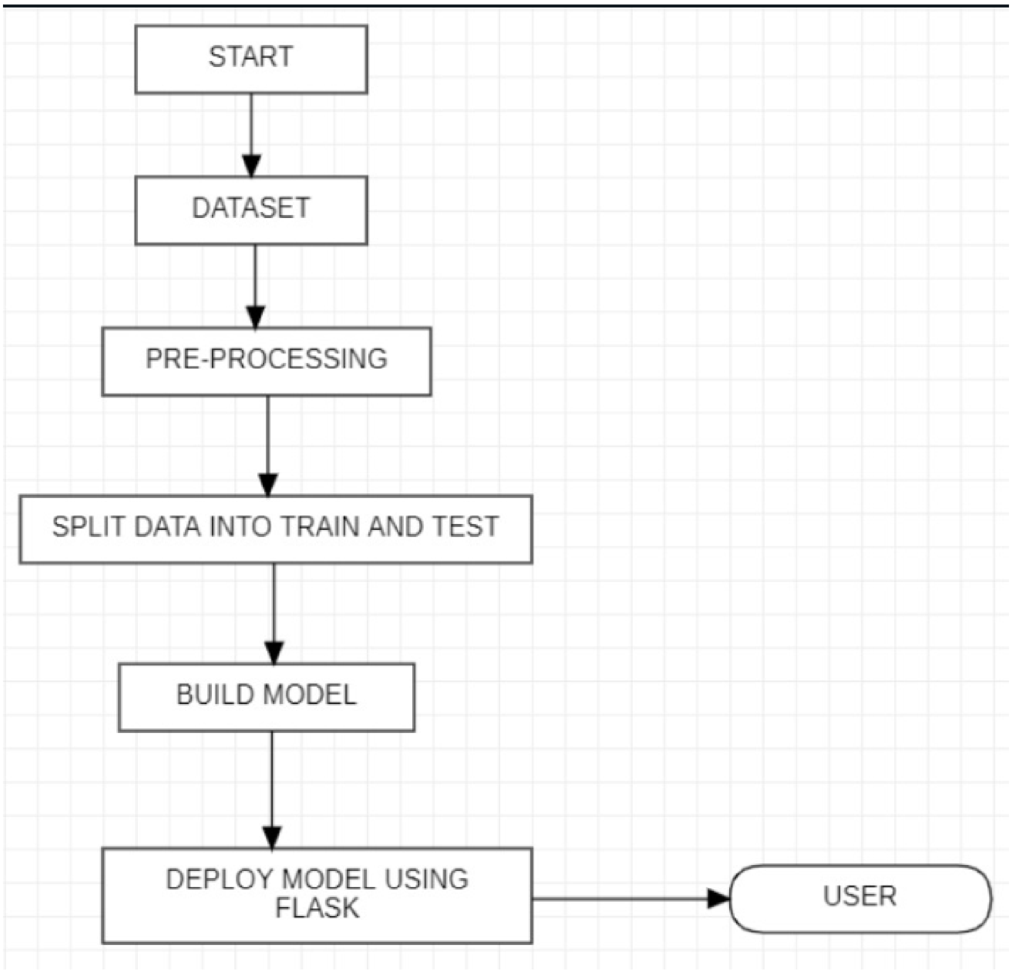
Several existing approaches and methods are employed to solve flight price prediction. These include statistical analysis, machine learning algorithms, and data mining techniques. Historical pricing data, along with various factors such as seasonality, time of booking, departure date, airline, route, and economic indicators, are used as input features. Regression models, time series analysis, and ensemble techniques like random forest or gradient boosting are commonly applied to build predictive models. Feature engineering, including lag variables, moving averages, and categorical encoding, is often performed to enhance the accuracy of predictions. The models are trained on historical data and evaluated using performance metrics such as mean absolute error or root mean square error to measure their predictive capabilities.

2.2 Proposed solution:

To approach the problem of flight price prediction, we began by compiling a large dataset of previous flight costs and pertinent attributes. Pre-process the data by dealing with missing values, outliers, and normalisation. Following that, handle category characteristics such as time-related indications, flight length, Source and Destination. Divide the dataset into training and testing sets and choose a machine learning algorithm such as Linear Regression, Decision Tree, or Random Forest. Train the model using training data, optimising its parameters, and then evaluate its performance using measures such as mean absolute error or root mean square error, r2 score. Using approaches such as random search, fine-tune the model's hyperparameters. Use a different validation dataset or cross-validation to validate the model's performance. Deploy the trained model and continually monitor and update its performance.This iterative technique, which combines data processing, feature engineering, model selection, training, assessment, deployment, and monitoring, aids in the prediction of flight prices.

3 THEORITICAL ANALYSIS

3.1 Block diagram:



3.2 Hard ware/Software Designing:

Hardware Requirements:

1.Memory (RAM): Enough RAM to accommodate the dataset size as well as the memory needs of the machine learning algorithms of choice.

2. Storage: Enough space to save the dataset, any pre-processed data, and trained models.

Software Requirements:

1. Programming Language: A programming language, such as Python, that is ideal for data analysis and machine learning.

2. Manipulation and Analysis of Data Python packages like as Pandas, NumPy, and Scikit-learn may be used to handle and analyse data.

3. Data Visualisation: To visualise data, model performance, and insights, use packages such as Matplotlib, Seaborn, or ggplot2.

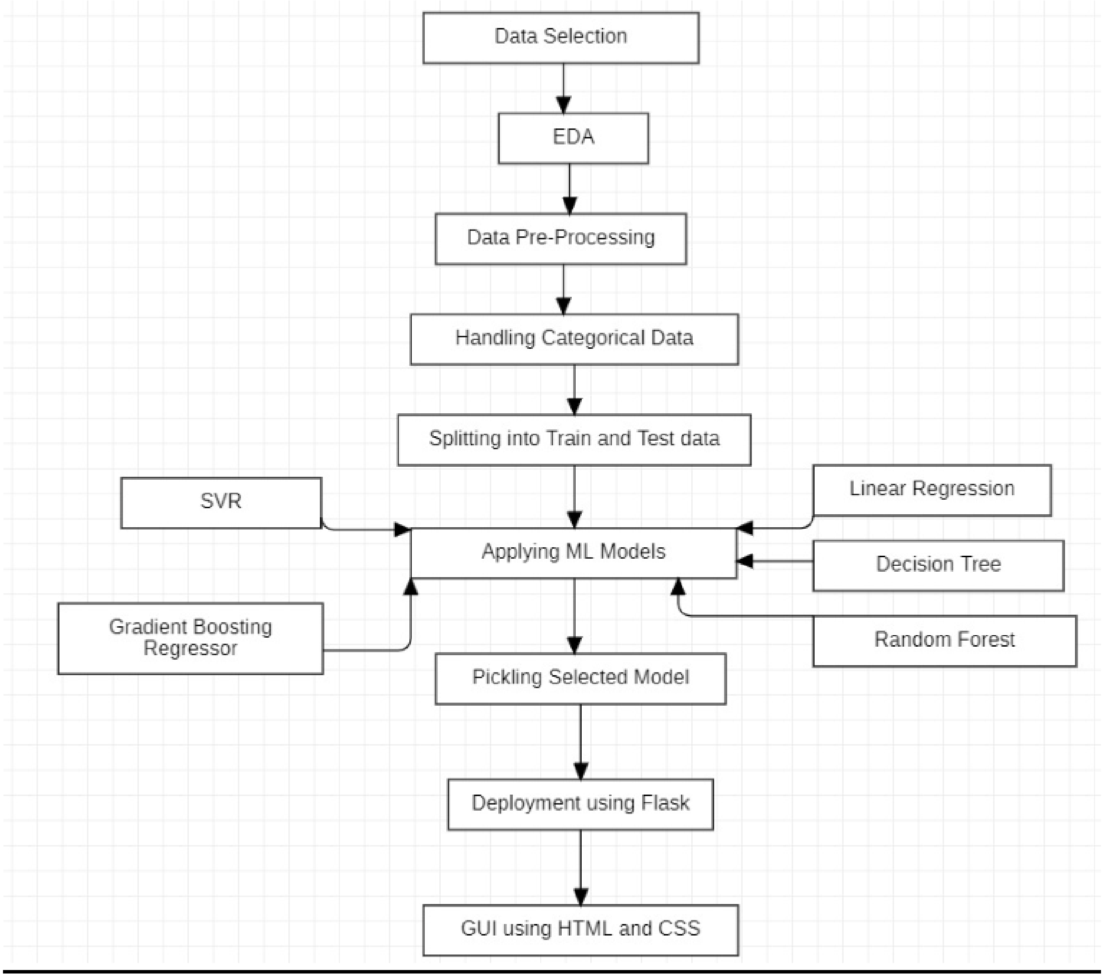
4. Development Setting: Set up an integrated development environment (IDE) for efficient coding and prototyping, such as Jupyter Notebook, PyCharm, or RStudio.

5. Deployment: Install Spyder into the IDE that was chosen.

4 EXPERIMENTAL INVESTIGATIONS

Several analysis and investigations are frequently carried out while working on flight price prediction. This involves exploratory data analysis (EDA) to better understand the dataset's properties, such as flight price distribution and patterns in past data. Feature significance analysis aids in the identification of influential characteristics for prediction. Time series analysis is used to spot patterns and trends in flight fares over time. Model performance evaluation utilising measures such as mean absolute error (MAE) or root mean square error (RMSE) aids in determining forecast accuracy. To optimise model parameters, hyperparameter adjustment is undertaken. Error analysis assists in identifying and comprehending prediction mistakes. These analysis and research help to improve the forecasting models' accuracy and acquire insight into the elements impacting flight pricing.

5 FLOWCHART



6 RESULT

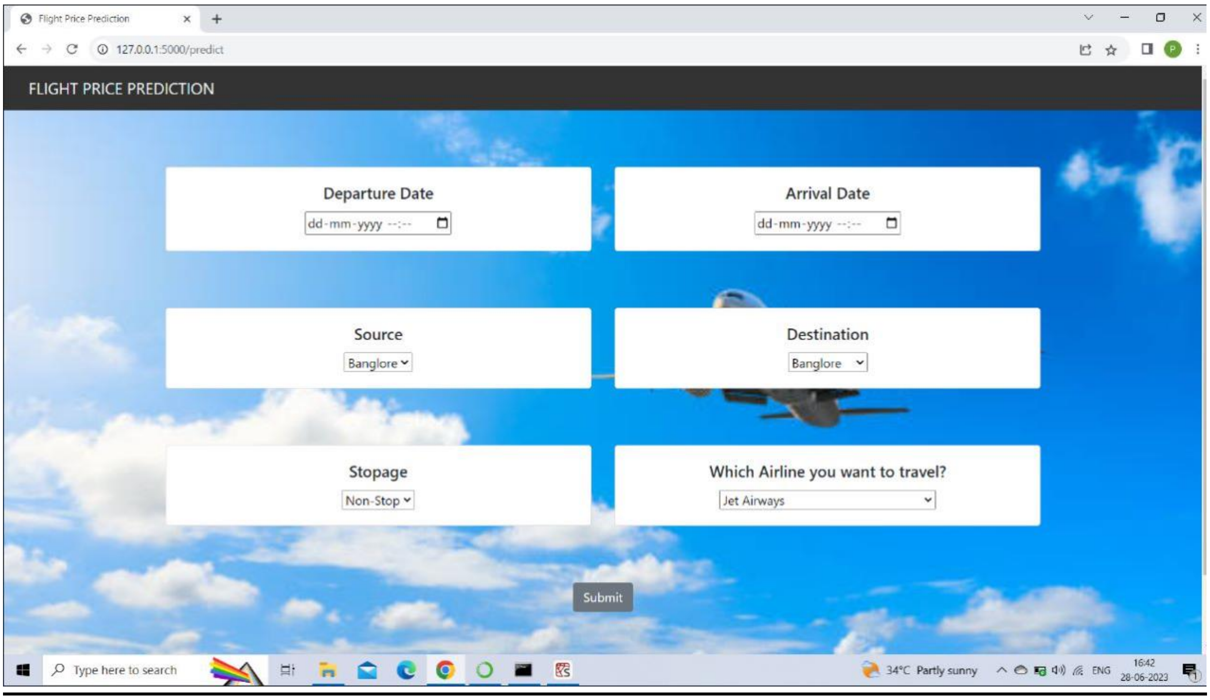
```
[ ] # Testing accuracy
    accuracy=r2_score(y_test,prediction)
    accuracy

    0.8375068811018169
```

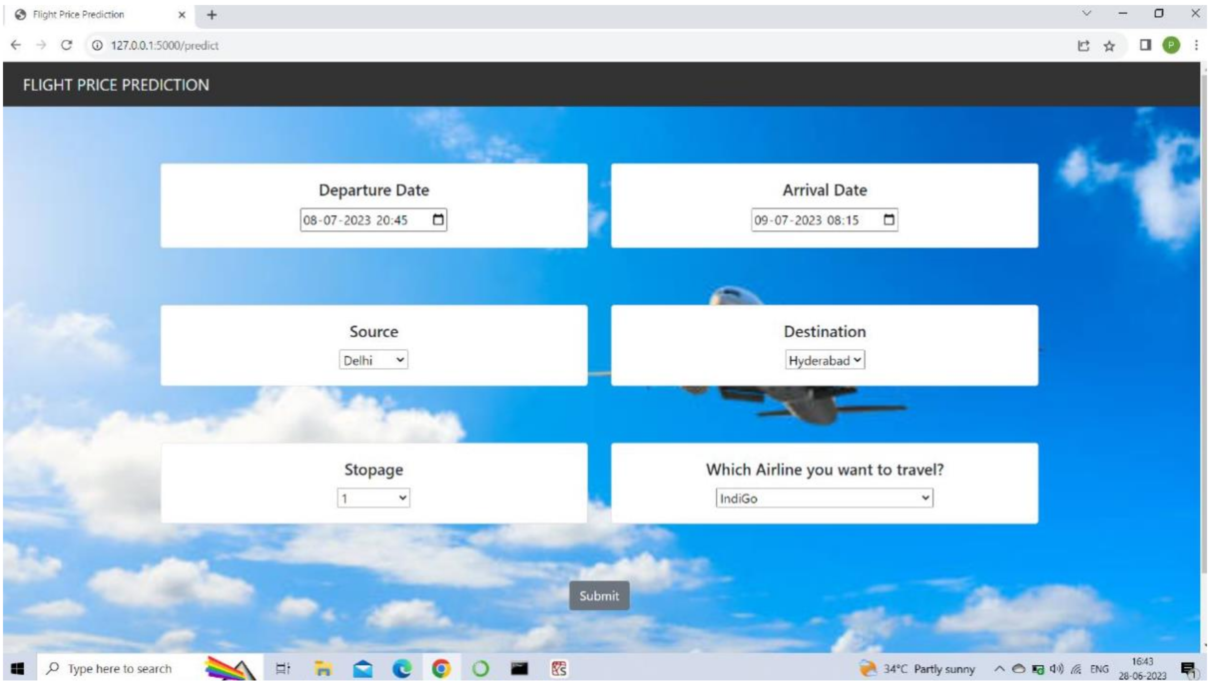
```
[ ] #Training
    r2_score(y_train,predict_train)

    0.888721744522617
```

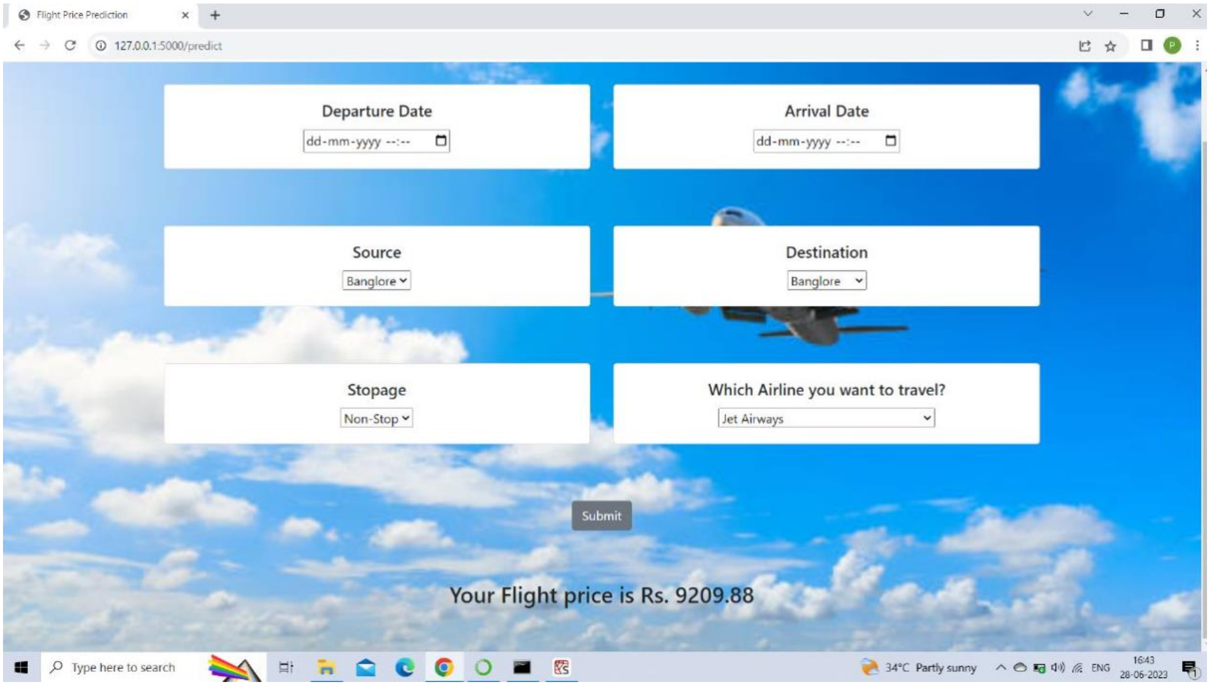
AFTER DEPLOYMENT:-



INPUT:



OUTPUT:-



7 ADVANTAGES & DISADVANTAGES

Advantages:

1.Cost Savings: Predicting flight costs allows travellers to make educated decisions by determining the optimal time to book tickets at cheaper prices. Individuals, families, and organisations may all benefit from huge cost reductions as a consequence of this.

2.Planning and Budgeting: Knowing flight fares ahead of time assists travellers to plan their travels more efficiently, manage their spending, and make necessary reservations. It gives predictability and aids in the management of travel expenditures.

3.Improved Travel Experience: Accurate airline price projections allow travellers to acquire better discounts, select more convenient travel dates, and optimise their itineraries. This results in a better overall travel experience and less stress.

Disadvantages:

1.Uncertainty and Fluctuations: Many factors impact flight pricing, including market circumstances, fuel costs, competition, and unforeseeable occurrences.

Predicting these variables effectively is difficult, and price changes can occur despite prediction algorithms.

2.Data Restrictions: Predicting flight prices is based on past data, which may not always reflect real-time market dynamics or account for unanticipated developments. Prediction accuracy and dependability can be impacted by data availability or quality.

3.Overdependence on forecasts: Relying only on flight price forecasts might result in missed chances or disappointment if the projections do not match the real pricing. It is critical to see forecasts as guidelines rather than absolute promises.

8 APPLICATIONS

1.Travel and Tourism Industry: Flight price prediction can help travel companies, online travel platforms, and tour operators provide their consumers with accurate and real-time information. It assists travellers in making educated decisions on flight booking, selecting the ideal time to travel, and locating the most cost-effective options.

2.Airline Revenue Management: To optimise their revenue management techniques, airlines might employ flight price prediction models. Airlines can dynamically alter ticket price to maximise revenue while taking into account factors like as seat occupancy, demand patterns, and competition by analysing historical data, market trends, and other pertinent factors.

3.Travel Aggregators and Online Booking Platforms: Flight price prediction can help travel aggregators and online booking platforms improve their offerings. These systems can attract more consumers, enhance user engagement, and improve customer happiness by assisting users in finding the best offers by providing pricing projections and predictions.

4.Travel Analytics and Insights: Flight price prediction may give useful insights and analytics for the aviation industry's market research, forecasting, and decision-making. This data may be used by airlines, airports, and industry experts to better understand consumer behaviour, follow market trends, and optimise their operations and pricing strategies.

9 CONCLUSION

Flight price prediction will change the way we approach air travel, delivering useful information to both passengers and carriers. These models provide precise forecasts using modern data analytics approaches, allowing travellers to make educated decisions and airlines to optimise revenue management. Improved price accuracy, personalised offers, and interaction with travel platforms have all come from the hyper tuning process. We were able to create a model with an accuracy of 83.8% and successfully deploy it. While obstacles persist, the future of airline price prediction offers enormous potential for additional breakthroughs, which will eventually help the industry by improving the travel experience and driving pricing strategy optimisation.

10 FUTURE SCOPE

1.Personalized Pricing and Offers:

Flight price prediction may be used to give individual travellers with personalised pricing and offers. Airlines and travel platforms may provide customised price alternatives, discounts, and promotions suited to particular travellers by analysing historical data and user preferences, increasing consumer satisfaction and loyalty.

2.Real-Time Dynamic Pricing:

Real-time dynamic pricing has a lot of possibilities in the future. Airlines may dynamically modify ticket rates in real-time by integrating live data streams that include parameters like as seat availability, demand-supply dynamics, rival pricing, and market trends. This strategy can improve revenue management, increase seat occupancy, and give consumers with the most current price alternatives.

3.Integration with trip Planning Platforms:

To provide complete trip planning, flight price prediction may be combined with travel planning platforms and mobile applications.

11 BIBILOGRAPHY

https://www.kaggle.com/code/anshigupta01/flight-price-prediction/notebook

https://www.ijraset.com/research-paper/aircraft-ticket-price-prediction-using-machine-learning

https://medium.com/geekculture/flight-fare-prediction-93da3958eb95

https://ieeexplore.ieee.org/abstract/document/8081365

A.SOURCE CODE:

```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the Dataset
df=pd.read_csv('Data_Train.csv')

df

df.head()

df.tail()

df.describe()

Understanding through Vizualization
```

```
# Uni-variate Analysis
# As there is only one numerical data column
plt.hist(df['Price'])

total_stops=df['Total_Stops'].value_counts()
total_stops

plt.pie(total_stops,autopct='%.2f',labels=['1 stop','non-stop','2 stops','3 stops','4 stops'])

# Bi-Variate Analysis
sns.scatterplot(data=df, x='Price', y='Total_Stops')
```

Descriptive Statistics

```
print(df.describe())

print('median')
print(df.median())

print('mode')
print(df.mode())

print(df.kurt())

print('printing quartile')
quantile=df.quantile(q=[0.75,0.25])
print(quantile)
print(quantile.iloc[0])
print(df.quantile(0.5))
```

```
print(quantile.iloc[1])
```

Checking and Handing the NULL values

```
df.isna()

df.isna().sum()

# Dropping the null values, as they are very less that is 2
df.dropna(inplace=True)

# Checking again for null values
df.isna().sum()
```

Checking for Outliers

```
sns.boxplot(df.Price)

perc99=df.Price.quantile(0.99)
perc99

df=df[df.Price<=perc99]
sns.boxplot(df.Price)
```

Handling Categorial Data

```
# Handling date column
df['Journey_Day']= pd.to_datetime(df.Date_of_Journey, format='%d/%m/%Y').dt.day
df['Journey_Month']=pd.to_datetime(df.Date_of_Journey, format='%d/%m/%Y').dt.month
```

```
df.head()

df.drop(['Date_of_Journey'], axis=1, inplace=True)
df.head()

# Handling Dep_Time
df['Dep_hour']=pd.to_datetime(df['Dep_Time']).dt.hour
df['Dep_min']=pd.to_datetime(df['Dep_Time']).dt.minute

df.drop(['Dep_Time'], axis=1, inplace=True)
df.head()

# Handling Arrival_Time
df['Arrival_hr']=pd.to_datetime(df['Arrival_Time']).dt.hour
df['Arrival_min']=pd.to_datetime(df['Arrival_Time']).dt.minute

df.drop(['Arrival_Time'], axis=1, inplace=True)
df.head()

# Converting the duration into two different columns i.e, to numerical values
duration=list(df['Duration'])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]
```

```
duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))
df['Duration_hours']=duration_hours
df['Duration_mins']=duration_mins
df.head()

df.drop(['Duration'], axis=1, inplace=True)
df.head()
```

Handling Categorial data

```
# Dropping the unimportant column
df.drop(['Route'], axis=1, inplace=True)
df.drop(['Additional_Info'], axis=1, inplace=True)
df.head()

# Using one-hot encoding
df_main=pd.get_dummies(df,columns=['Airline','Source','Destination'])
df_main.head()

df_main.replace({'non-stop':0, '1 stop':1, '2 stops':2, '3 stops':3, '4 stops':4}, inplace=True)
df_main.head()

#Multivariate analysis
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True)
```

Split data into dependent and independent variable

```
# Dependent variable is price
y=df_main['Price']

y.head()

# Reamining are independent variables
X=df_main.drop(columns=['Price'],axis=1)
X.head()

name=X.columns
name

from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
X_scaled=scale.fit_transform(X)
X_scaled

X=pd.DataFrame(X_scaled,columns=name)
X
```

Splitting data into traning and testing

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)

X_train.head()
```

```
X_test.head()

y_train

y_test
```

Building the Model

```
# Linear Regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

model = LinearRegression().fit(X_train, y_train)


y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)


accuracy_train = model.score(X_train, y_train)
mse_train = mean_squared_error(y_train, y_pred_train)
r2_train = r2_score(y_train, y_pred_train)
accuracy_train


accuracy_test = model.score(X_test, y_test)
mse_test = mean_squared_error(y_test, y_pred_test)
r2_test = r2_score(y_test, y_pred_test)
```

```
print("Accuracy - Train: {:} Test: {:}".format(accuracy_train, accuracy_test))
print("MSE - Train: {:} Test: {:}".format(mse_train, mse_test))
print("R2 - Train: {:} Test: {:}".format(r2_train, r2_test))

# Decision Tree
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score

dt = DecisionTreeRegressor(random_state=42)

dt.fit(X_train, y_train)

# Predict the values for the training and testing sets
y_pred_train = dt.predict(X_train)
y_pred_test = dt.predict(X_test)

# Compute the accuracy, MSE, and R2 for the training set
accuracy_train = dt.score(X_train, y_train)
mse_train = mean_squared_error(y_train, y_pred_train)
r2_train = r2_score(y_train, y_pred_train)

# Compute the accuracy, MSE, and R2 for the testing set
accuracy_test = dt.score(X_test, y_test)
mse_test = mean_squared_error(y_test, y_pred_test)
r2_test = r2_score(y_test, y_pred_test)
print("Accuracy - Train: {:} Test: {:}".format(accuracy_train, accuracy_test))
print("MSE - Train: {:} Test: {:}".format(mse_train, mse_test))
print("R2 - Train: {:} Test: {:}".format(r2_train, r2_test))
```

```
# Random Forest
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

rf = RandomForestRegressor(n_estimators=10, random_state=42)

rf.fit(X_train, y_train)

# Predict the values for the training and testing sets
y_pred_train = rf.predict(X_train)
y_pred_test = rf.predict(X_test)

# Compute the accuracy, MSE, and R2 for the training set
accuracy_train = rf.score(X_train, y_train)
mse_train = mean_squared_error(y_train, y_pred_train)
r2_train = r2_score(y_train, y_pred_train)

# Compute the accuracy, MSE, and R2 for the testing set
accuracy_test = rf.score(X_test, y_test)
mse_test = mean_squared_error(y_test, y_pred_test)
r2_test = r2_score(y_test, y_pred_test)

print("Accuracy - Train: {:} Test: {:}".format(accuracy_train, accuracy_test))
print("MSE - Train: {:} Test: {:}".format(mse_train, mse_test))
print("R2 - Train: {:} Test: {:}".format(r2_train, r2_test))
```

So, here we can understand that the model is overfitting

```
#Support vector Regressor
from sklearn.svm import SVR
svr=SVR().fit(X_train, y_train)
# Predict the values for the training and testing sets
y_pred_train = svr.predict(X_train)
y_pred_test = svr.predict(X_test)

# Compute the accuracy, MSE, and R2 for the training set
accuracy_train = svr.score(X_train, y_train)
mse_train = mean_squared_error(y_train, y_pred_train)
r2_train = r2_score(y_train, y_pred_train)

# Compute the accuracy, MSE, and R2 for the testing set
accuracy_test = svr.score(X_test, y_test)
mse_test = mean_squared_error(y_test, y_pred_test)
r2_test = r2_score(y_test, y_pred_test)
print("Accuracy - Train: {:} Test: {:}".format(accuracy_train, accuracy_test))
print("MSE - Train: {:} Test: {:}".format(mse_train, mse_test))
print("R2 - Train: {:} Test: {:}".format(r2_train, r2_test))

#GradientBossting Regressor
from sklearn.ensemble import GradientBoostingRegressor
gbr=GradientBoostingRegressor().fit(X_train,y_train)
# Predict the values for the training and testing sets
y_pred_train = gbr.predict(X_train)
y_pred_test = gbr.predict(X_test)

# Compute the accuracy, MSE, and R2 for the training set
accuracy_train = gbr.score(X_train, y_train)
```

```
mse_train = mean_squared_error(y_train, y_pred_train)
r2_train = r2_score(y_train, y_pred_train)

# Compute the accuracy, MSE, and R2 for the testing set
accuracy_test = gbr.score(X_test, y_test)
mse_test = mean_squared_error(y_test, y_pred_test)
r2_test = r2_score(y_test, y_pred_test)
print("Accuracy - Train: {:} Test: {:}".format(accuracy_train, accuracy_test))
print("MSE - Train: {:} Test: {:}".format(mse_train, mse_test))
print("R2 - Train: {:} Test: {:}".format(r2_train, r2_test))

#Hypertuning
from sklearn.model_selection import RandomizedSearchCV

random_grid = {
  'n_estimators' : [100, 120, 150, 180, 200,220],
  'max_features':['auto','sqrt'],
  'max_depth':[5,10,15,20],
  }

rf=RandomForestRegressor()
rf_random=RandomizedSearchCV(estimator=rf,param_distributions=random_grid,cv=3,verbose=2,n_jobs=-1,)

rf_random.fit(X_train,y_train)

# best parameter
rf_random.best_params_

# best parameter
```

```
rf_random.best_params_

#predicting the values
prediction = rf_random.predict(X_test)
predict_train=rf_random.predict(X_train)

#distribution plot between actual value and predicted value
sns.displot(y_test-prediction)

accuracy=r2_score(y_test,prediction)
accuracy

r2_score(y_train,predict_train)

mse=mean_squared_error(y_test, prediction)
mse
```

By hypertuning the model, the accuracy increases. The model accuracy is 83.8%

```
import pickle
pickle.dump(rf_random,open("model.pkl","wb"))
```