CHIKKANNA GOVERNMENT ARTS COLLEGE

DEPARTMENT OF BACHELOR OF COMPUTER APPLICATION

TIRUPUR-641602

(AFFILIATED TO BHARATHIAR UNIVERSITY)

TEAM MEMBERS NAME :

GOWTHAM A (2022J0031)

BALAJI D (2022J0021)

DHANUSH S (2022J0025)

PRAVEEN P (2022J0050)

Optimizing Flight Booking Decisions through Machine Learning Price Predictions

1.INTRODUCTION

1.1 OVERVIEW :

People who work frequently travel through flight will have better knowledge on best discount and right time to buy the ticket. For the business purpose many airline companies change prices according to the seasons or time duration. They will increase the price when people travel more. Estimating the highest prices of the airlines data for the route is collected with features such as Duration, Source, Destination, Arrival and Departure. Features are taken from chosen dataset and in the price wherein the airline price ticket costs vary overtime. we have implemented flight price prediction for users by using KNN, decision tree and random forest algorithms. Random Forest shows the best accuracy of 80% for predicting the flight price. also, we have done correlation tests and metrics for the statistical analysis.

The use of machine learning algorithms for flight price prediction can have significant benefits for both travelers and travel booking platforms. Travelers can save time and money by receiving personalized recommendations, while travel booking platforms can increase customer satisfaction by providing accurate and relevant information. Overall, this project aims to improve the customer experience in the travel industry through the use of data-driven insights and recommendations.

1.2. PURPOSE :

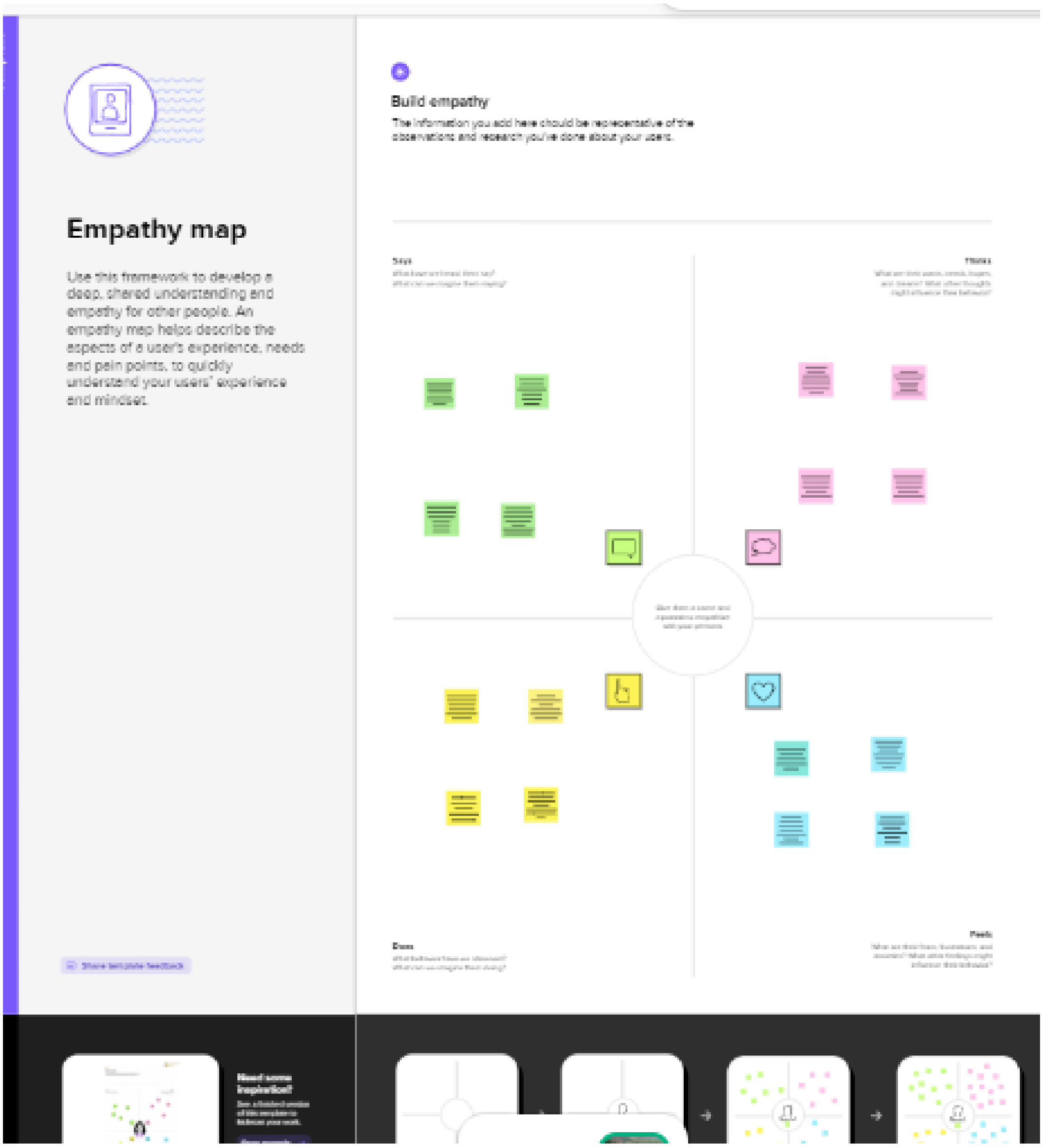
The purpose of optimizing flight booking decisions through machine learning price predictions is to help travelers make informed booking decisions and save money while improving the overall customer experience for travel booking platforms. By leveraging historical flight data and relevant factors, machine learning algorithms can accurately predict flight prices and provide personalized recommendations to travelers.

The use of machine learning algorithms for flight price prediction can have significant benefits for both travelers and travel booking platforms. For travelers, it can save time and money by providing accurate and relevant information on flight prices and optimal booking times. For travel booking platforms, it can increase customer satisfaction by providing a better customer experience and increasing customer loyalty.

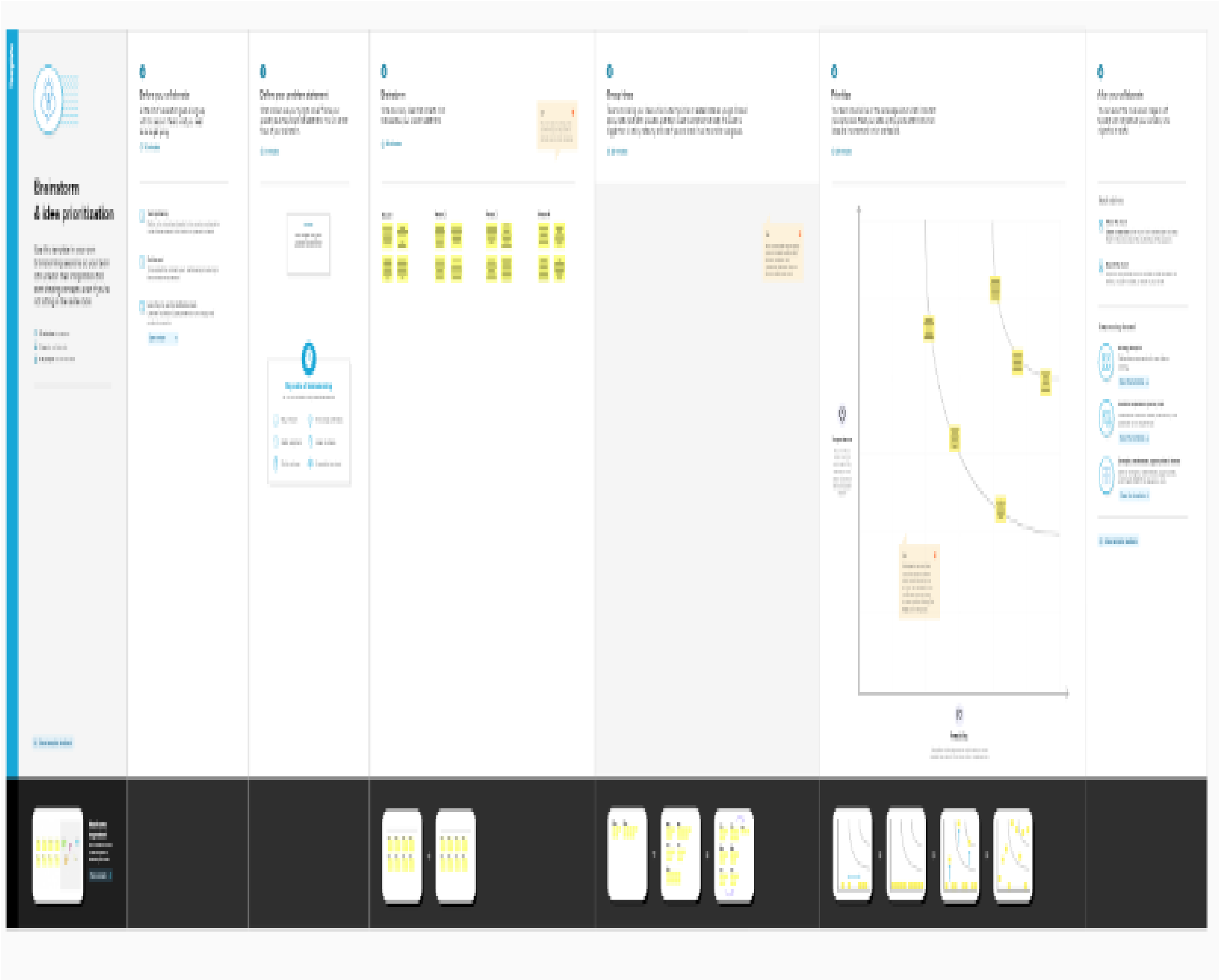
Overall, the purpose of this project is to improve the customer experience in the travel industry through the use of data-driven insights and recommendations. By optimizing flight booking decisions through machine learning price predictions, we can help travelers make better-informed decisions and ultimately make air travel more accessible and affordable.

2. PROBLEM DEFINITION & DESIGN THINKING

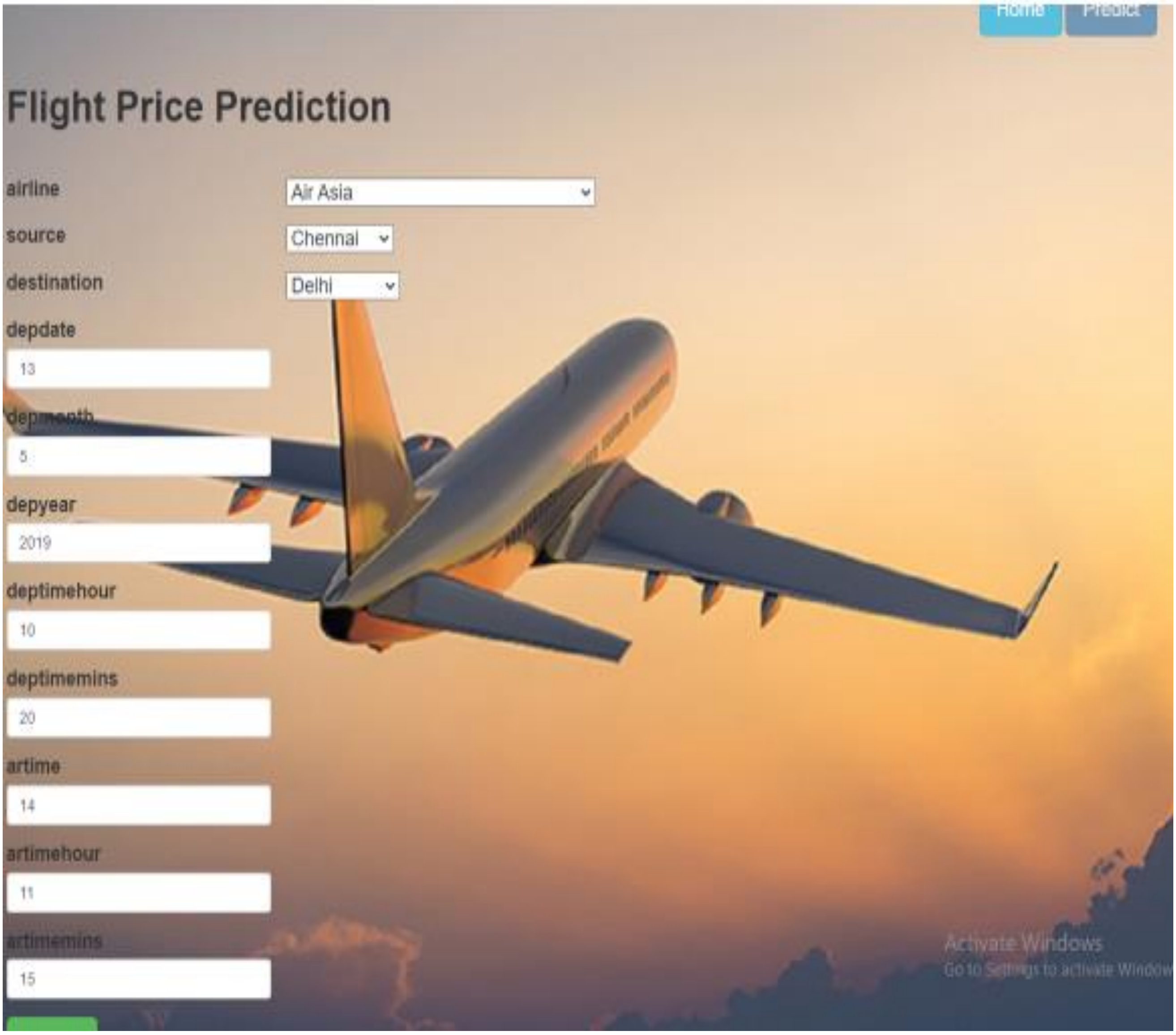
2.1. EMPATHY MAP



2.2 IDEATION & BRAINSTORMINGS MAP



3. RESULT


Home
Predict
Flight Price Prediction
airline
Air Asia
source
Chennai
destination
Delhi
depdate
13
depmonth
5
depyear
2019
deptimehour
10
deptimemins
20
artime
14
artimehour
11
artimemins
15
Activate Windows

4.ADVANTAGES & DISADVANTAGES

ADVANTAGES :

Cost savings:

By analyzing historical data and current trends, machine learning algorithms can predict the optimal time to book a flight, allowing you to save money on your travel expenses.

Increased accuracy:

Machine learning algorithms can analyze large amounts of data quickly and accurately, providing more reliable price predictions than traditional methods.

Time savings:

By quickly identifying the best flight options for your travel needs, machine learning price prediction can save you time and frustration.

Personalization:

Machine learning algorithms can provide personalized recommendations based on your travel history and preferences, helping you find the best flights and routes for your specific needs.

Flexibility:

Machine learning algorithms can adapt to changing market conditions and fluctuations in flight prices, providing up-to-date predictions for when to book your flights.

DISADVANTAGES :

Limited data:

Machine learning algorithms require a significant amount of data to make accurate predictions. If there is not enough historical data available, the predictions may not be reliable.

Lack of transparency:

Machine learning algorithms can be complex, making it difficult to understand how they make their predictions. This lack of transparency can make it challenging to trust the predictions.

Unforeseen events:

Machine learning algorithms cannot account for unforeseen events such as natural disasters or political unrest, which can impact flight prices and availability.

Overreliance on predictions:

Relying too heavily on machine learning price predictions can lead to missed opportunities or overpaying for flights if the predictions are incorrect.

5. APPLICATION

There are several applications for using machine learning price prediction to optimize flight booking decisions, including:

Online travel agencies (OTAs):

OTAs can use machine learning algorithms to analyze large amounts of data and provide personalized recommendations to their users on when to book flights.

Airlines:

Airlines can use machine learning price prediction to optimize their revenue management strategies by predicting demand and setting prices accordingly.

Travel management companies:

Travel management companies can use machine learning price prediction to help their clients save money on business travel expenses by identifying the best times to book flights.

Travel search engines:

Travel search engines can use machine learning algorithms to analyze flight prices and provide users with real-time price predictions for different travel dates and destinations.

Loyalty programs:

Airlines and OTAs can use machine learning price prediction to provide personalized offers and rewards to their loyalty program members, such as discounts on flights or free upgrades.

6. CONCLUSION

Evaluating the algorithmic rule, a dataset is collected, pre-processed, performed data modelling and studied a value difference for the number of restricted days by the passengers for travelling. Machine Learning algorithms with square measure for forecasting the accurate fare of airlines and it gives accurate value of plane price ticket at limited and highest value. Information is collected from Kaggle websites that sell the flight tickets therefore restricting data which are often accessed. The results obtained by the random forest and decision tree algorithm has better accuracy, but best accuracy is predicted by decision tree algorithm as shown is the above analysis. Accuracy of the model is also forecasted by the R-squared value.

In Upcoming days when huge amount of information is accessed as in detailed information in the dataset, the expected results in future are highly correct. For further research anyone desire to expand upon it ought to request different sources of historical data or be a lot of organized in collection knowledge manually over amount of your time to boot, a lot of different combination of plane are going to be traversed. There is whole possibility that planes differ their execution ideas consisting characteristics of the plane. At last, it is curious to match our model accuracy with that of the business models accuracy offered nowadays.

7. FUTURE SCOPE

The future scope of optimizing flight booking decisions through machine learning price prediction is promising, as the technology and applications continue to evolve.

Some potential areas for future development and expansion include:

Increased personalization:

Machine learning algorithms can analyze individual traveler data to provide more personalized recommendations for when to book flights, taking into account travel history, preferences, and budget.

Integration with other travel services:

Machine learning price prediction can be integrated with other travel services, such as hotel and rental car bookings, to provide a more comprehensive and seamless travel experience.

Real-time pricing and availability updates:

Machine learning algorithms can be used to provide real-time updates on flight prices and availability, allowing travelers to make informed booking decisions.

Enhanced accuracy and reliability:

As machine learning algorithms continue to be refined and trained on larger data sets, their accuracy and reliability in predicting flight prices will likely improve.

Use of alternative data sources:

Machine learning algorithms can be trained on alternative data sources, such as social media and weather data, to provide more accurate predictions on flight prices and availability.

Integration with blockchain technology:

Blockchain technology can help improve the transparency and security of flight bookings and payment processing, which can further enhance the reliability and efficiency of machine learning price prediction.

8. APPENDIX

A.SOURCE CODE :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import metrics
import seaborn as sns
sns.set()
```

```
# Mount Google Drive - applicable, if working on Goog
le Drive
from google.colab import drive
drive.mount('/content/drive')



# Set Working Directory - if working on Google Drive
%cd /content/drive/MyDrive/Project10_FlightPricePredi
ction

# # Set Working Directory - if working on Local Machi
ne
# import os
# os.chdir('/Users//replace_me')



# Load dataset from Project folder
dataset = pd.read_excel("/content/Data_Train.xlsx")

# To stretch head function output to the notebook wid
th
pd.set_option('display.max_columns', None)

dataset.head()
```

```
dataset.info()
```

```
dataset.isnull().sum()
```

```
dataset.dropna(inplace = True)
```

```
dataset.isnull().sum()
```

```
dataset.head()
```

```
# Date_of_Journey is the day when plane departs.
dataset["journey_day"] = pd.to_datetime(dataset.Date_
of_Journey, format="%d/%m/%Y").dt.day
dataset["journey_month"] = pd.to_datetime(dataset["Da
te_of_Journey"], format = "%d/%m/%Y").dt.month
dataset.head()
```

```
# Since we have converted Date_of_Journey column into
 integers, Now we can drop as it is of no use.
dataset.drop(["Date_of_Journey"], axis = 1, inplace =
 True)
```

```
# Departure time is when a plane leaves the gate.
# Similar to Date_of_Journey we can extract values fr
om Dep_Time
# Extracting Hours
dataset["dep_hour"] = pd.to_datetime(dataset["Dep_Tim
e"]).dt.hour
# Extracting Minutes
dataset["dep_min"] = pd.to_datetime(dataset["Dep_Time
"]).dt.minute
# Now we drop Dep_Time as it is of no use
```

```
dataset.drop(["Dep_Time"], axis = 1, inplace = True)
```

```
# Arrival time is when the plane pulls up to the gate
.
# Similar to Date_of_Journey we can extract values fr
om Arrival_Time

# Extracting Hours
dataset["arrival_hour"] = pd.to_datetime(dataset["Arr
ival_Time"]).dt.hour
# Extracting Minutes
dataset["arrival_min"] = pd.to_datetime(dataset["Arri
val_Time"]).dt.minute
# Now we can drop Arrival_Time as it is of no use
dataset.drop(["Arrival_Time"], axis = 1, inplace = Tr
ue)
```

```
dataset.head()
```

```
# Duration is the time taken by plane to reach destin
ation
# It is the difference betwen Arrival Time and Depart
ure time
# Assigning and converting Duration column into list,
 for looping through
duration = list(dataset["Duration"])
# In table above, Row Index=2, we have Duration = 19h
 (missing minutes)
# Looping through all duration values, to ensure it h
as both hours & mins: 'xh ym'
for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if d
uration 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
# Prepare separate duration_hours and duration_mins l
ists
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

# Add duration_hours and duration_mins list to our da
taset df
dataset["Duration_hours"] = duration_hours
dataset["Duration_mins"] = duration_mins
# Drop Duration column from the dataset
dataset.drop(["Duration"], axis = 1, inplace = True)

dataset.head()
```

```
# Feature engineering on: Airline
dataset["Airline"].value_counts()
```

```
# As Airline is Nominal Categorical data we will perf
orm OneHotEncoding
Airline = dataset[["Airline"]]
Current_Airline_List = Airline['Airline']
New_Airline_List = []

for carrier in Current_Airline_List:
  if carrier in ['Jet Airways', 'IndiGo', 'Air India'
, 'SpiceJet',
```

```
        'Multiple carriers', 'GoAir', 'Vistara', 'Air
Asia']:
    New_Airline_List.append(carrier)
  else:
    New_Airline_List.append('Other')

Airline['Airline'] = pd.DataFrame(New_Airline_List)
Airline['Airline'].value_counts()


Airline = pd.get_dummies(Airline, drop_first= True)
Airline.head()
```

```
# Feature engineering on: Source
dataset["Source"].value_counts()
```

```
# As Source is Nominal Categorical data we will perfo
rm OneHotEncoding
Source = dataset[["Source"]]
Source = pd.get_dummies(Source, drop_first= True)
# drop_first= True means we drop the first column to
prevent multicollinearity
Source.head()
```

```
# Feature engineering on: Destination
dataset["Destination"].value_counts()
```

```
# Renaming destination 'New Delhi' to 'Delhi' - to ma
tch with Source
Destination = dataset[["Destination"]]
```

```
Current_Destination_List = Destination['Destination']
New_Destination_List = []

for value in Current_Destination_List:
  if value in ['New Delhi']:
    New_Destination_List.append('Delhi')
  else:
    New_Destination_List.append(value)

Destination['Destination'] = pd.DataFrame(New_Destina
tion_List)

# As Destination is Nominal Categorical data we will
perform OneHotEncoding
Destination = pd.get_dummies(Destination, drop_first
= True)
Destination.head()
```

```
# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other
dataset.drop(["Route", "Additional_Info"], axis = 1,
inplace = True)
```

```
# Feature engineering on: Total_Stops
dataset["Total_Stops"].value_counts()
```

```
# As this is case of Ordinal Categorical type we perf
orm LabelEncoder
# Here Values are assigned with corresponding keys
dataset.replace({"non-
stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4
 stops": 4}, inplace = True)
dataset.head()
```

```
# Concatenate dataframe -->
> train_data + Airline + Source + Destination
data_train = pd.concat([dataset, Airline, Source, Destination], axis = 1) # axis = 1 signifies column
data_train.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)


data_train.head()


data_train.shape


data_train.columns




X = data_train.loc[:, ['Total_Stops', 'journey_day', 'journey_month', 'dep_hour',
       'dep_min', 'arrival_hour', 'arrival_min', 'Duration_hours',
       'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
       'Airline_Jet Airways', 'Airline_Multiple carriers', 'Airline_Other',
       'Airline_SpiceJet', 'Airline_Vistara', 'Source_Chennai', 'Source_Delhi',
       'Source_Kolkata', 'Source_Mumbai', 'Destination_Cochin',
       'Destination_Delhi', 'Destination_Hyderabad', 'Destination_Kolkata']]
y = data_train.iloc[:, 1]




print(X.shape, y.shape)
```

```
# Important feature using ExtraTreesRegressor
from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(X, y)

print(selection.feature_importances_)
```

```
#plot graph of feature importances for better visuali
zation
plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_import
ances_, index=X.columns)
feat_importances.nlargest(25).plot(kind='barh')
plt.show()
```

```
# Checking for Multicollinearity
from statsmodels.stats.outliers_influence import vari
ance_inflation_factor
def calc_vif(z):
    # Calculating Variable Inflation Factor (VIF)
    vif = pd.DataFrame()
    vif["variables"] = z.columns
    vif["VIF"] = [variance_inflation_factor(z.values,
 i) for i in range(z.shape[1])]
    return(vif)
```

```
# Compute VIF on X
calc_vif(X)
```

```
# Drop 'Source_Delhi'
```

```
X = data_train.loc[:, ['Total_Stops', 'journey_day',
'journey_month', 'dep_hour',
       'dep_min', 'arrival_hour', 'arrival_min', 'Dur
ation_hours',
       'Duration_mins', 'Airline_Air India', 'Airline
_GoAir', 'Airline_IndiGo',
       'Airline_Jet Airways', 'Airline_Multiple carri
ers', 'Airline_Other',
       'Airline_SpiceJet', 'Airline_Vistara', 'Source
_Chennai',
       'Source_Kolkata', 'Source_Mumbai', 'Destinatio
n_Cochin',
       'Destination_Delhi', 'Destination_Hyderabad',
'Destination_Kolkata']]
X.head()
```

```
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 = 42)
```

```
from sklearn.ensemble import RandomForestRegressor
rf_reg = RandomForestRegressor()
rf_reg.fit(X_train, y_train)
```

```
print('Model Performance on Training Set:', round(rf_
reg.score(X_train, y_train)*100,2))
print('Model Performance on Test Set:', round(rf_reg.
score(X_test, y_test)*100,2))
```

```
# Plot performance graph
y_pred = rf_reg.predict(X_test)
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

```
# Model Error Values
print('MAE:', metrics.mean_absolute_error(y_test, y_p
red))
print('MSE:', metrics.mean_squared_error(y_test, y_pr
ed))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_t
est, y_pred)))
#RMSE = sqrt((PV-OV)^2/n)
```

```
# RMSE/(max(DV)-min(DV))
print('Normalized RMSE ', round(np.sqrt(metrics.mean_
squared_error(y_test, y_pred))/(max(y_test)-
min(y_test)),2))
print('Max Value: ', max(y), '\nMin Value: ', min(y))
```

```
import pickle
# open a file, where you ant to store the data
file = open('c2_flight_rf.pkl', 'wb')
# dump information to that file
pickle.dump(rf_reg, file)
```

```
import pickle
path = 'c1_flight_rf.pkl'
model = open(path,'rb')
rf_model = pickle.load(model)
```

```
unseen_dataset = pd.read_excel("./a2_Unseen_Dataset.x
lsx")
unseen_dataset.head()
```

```
# Perform feature engineering on object dt variables
# Feature Engineering on: 'Date_of_Journey'
unseen_dataset["journey_day"] = pd.to_datetime(unseen
_dataset.Date_of_Journey, format="%d/%m/%Y").dt.day
unseen_dataset["journey_month"] = pd.to_datetime(unse
en_dataset["Date_of_Journey"], format = "%d/%m/%Y").d
t.month
unseen_dataset.drop(["Date_of_Journey"], axis = 1, in
place = True)

# Feature Engineering on: 'Dep_Time'
unseen_dataset["dep_hour"] = pd.to_datetime(unseen_da
taset["Dep_Time"]).dt.hour
unseen_dataset["dep_min"] = pd.to_datetime(unseen_dat
aset["Dep_Time"]).dt.minute
unseen_dataset.drop(["Dep_Time"], axis = 1, inplace =
 True)

# Feature Engineering on: 'Arrival_Time'
unseen_dataset["arrival_hour"] = pd.to_datetime(unsee
n_dataset["Arrival_Time"]).dt.hour
unseen_dataset["arrival_min"] = pd.to_datetime(unseen
_dataset["Arrival_Time"]).dt.minute
unseen_dataset.drop(["Arrival_Time"], axis = 1, inpla
ce = True)

# Feature Engineering on: 'Duration'
duration = list(unseen_dataset["Duration"])
for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if d
uration 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
unseen_dataset["Duration_hours"] = duration_hours
unseen_dataset["Duration_mins"] = duration_mins
unseen_dataset.drop(["Duration"], axis = 1, inplace =
 True)


# Perform feature engineering on Categorical dt varia
bles
# Feature Engineering on: 'Airline'
Airline = unseen_dataset[["Airline"]]
New_Airline_List = []
Current_Airline_List = Airline['Airline']
for carrier in Current_Airline_List:
  if carrier in ['IndiGo', 'Air India', 'Jet Airways'
, 'SpiceJet',
       'Multiple carriers', 'GoAir', 'Vistara', 'Air
Asia']:
    New_Airline_List.append(carrier)
  else:
    New_Airline_List.append('Other')
Airline['Airline'] = pd.DataFrame(New_Airline_List)
Airline = pd.get_dummies(Airline, drop_first= True)

# Feature Engineering on: 'Source'
Source = unseen_dataset[["Source"]]
Source = pd.get_dummies(Source, drop_first= True)
Source.head()

# Feature Engineering on: 'Destination'
Destination = unseen_dataset[["Destination"]]
Current_Destination_List = Destination['Destination']
New_Destination_List = []
for value in Current_Destination_List:
  if value in ['New Delhi']:
    New_Destination_List.append('Delhi')
  else:
    New_Destination_List.append(value)
Destination['Destination'] = pd.DataFrame(New_Destina
tion_List)
Destination['Destination'].value_counts()
```

```
Destination = pd.get_dummies(Destination, drop_first
= True)
Destination.head()

# Feature Engineering on: 'Route', 'Additional_Info
unseen_dataset.drop(["Route", "Additional_Info"], axi
s = 1, inplace = True)

# Feature Engineering on: 'Total_Stops'
unseen_dataset.replace({"non-
stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4
 stops": 4}, inplace = True)

# Concatenate dataframe --
> train_data + Airline + Source + Destination
data_test = pd.concat([unseen_dataset, Airline, Sourc
e, Destination], axis = 1)
data_test.drop(["Airline", "Source", "Destination"],
axis = 1, inplace = True)

# See how the test dataset looks
data_test.head()




# Drop 'Source_Delhi'
X_unseen = data_test.loc[:, ['Total_Stops', 'journey_
day', 'journey_month', 'dep_hour',
       'dep_min', 'arrival_hour', 'arrival_min', 'Dur
ation_hours',
       'Duration_mins', 'Airline_Air India', 'Airline
_GoAir', 'Airline_IndiGo',
       'Airline_Jet Airways', 'Airline_Multiple carri
ers', 'Airline_Other',
       'Airline_SpiceJet', 'Airline_Vistara', 'Source
_Chennai',
       'Source_Kolkata', 'Source_Mumbai', 'Destinatio
n_Cochin',
       'Destination_Delhi', 'Destination_Hyderabad',
'Destination_Kolkata']]
y_unseen = data_test.iloc[:, 1]
```

```
y_pred = rf_model.predict(X_unseen)
```

```
print('R2 value: ', round(metrics.r2_score(y_unseen,
y_pred),2))
print('Normalized RMSE: ', round(np.sqrt(metrics.mean
_squared_error(y_unseen, y_pred))/(max(y_unseen)-
min(y_unseen)),2))
print('Max Value: ', max(y_unseen), '\nMin Value: ',
min(y_unseen))
```

```
# writing model output file
df_y_pred = pd.DataFrame(y_pred,columns= ['Predicted
Price'])
original_dataset = pd.read_excel("./a2_Unseen_Dataset
.xlsx")
dfx = pd.concat([original_dataset, df_y_pred], axis=1
)
dfx.to_excel("c2_ModelOutput.xlsx")
dfx.head()
```

```
from sklearn.model_selection import RandomizedSearchC
V
```

```
#Randomized Search CV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 1
00, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num =
 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
```

```
# Minimum number of samples required at each leaf nod
e
min_samples_leaf = [1, 2, 5, 10]



# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split
,
               'min_samples_leaf': min_samples_leaf}



# Random search of parameters, using 5 fold cross val
idation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = rf_reg, pa
ram_distributions = random_grid,
                               scoring='neg_mean_squa
red_error', n_iter = 10, cv = 5,
                               verbose=2, random_stat
e=42, n_jobs = 1)



# Model Training with Hyperparameter Tuning
rf_random.fit(X_train,y_train)


rf_random.best_params_


# Plot Performance Chart
prediction = rf_random.predict(X_test)
plt.figure(figsize = (8,8))
```

```
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

```
# RMSE/(max(DV)-min(DV))
print('R2 value: ', round(metrics.r2_score(y_test, pr
ediction),2))
print('RMSE: ', round(np.sqrt(metrics.mean_squared_er
ror(y_test, prediction)),2))
print('Normalized RMSE: ', round(np.sqrt(metrics.mean
_squared_error(y_test, prediction))/(max(y_test)-
min(y_test)),2))
print('Max Value: ', max(y_test), '\nMin Value: ', mi
n(y_test))
```