FLIGHT *Price* PREDICTION

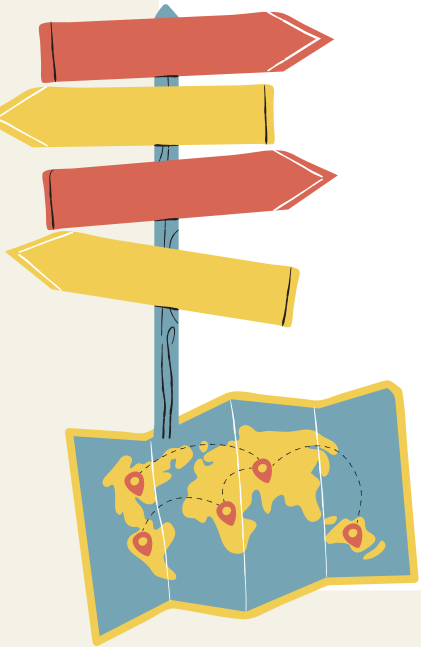
VIJAYARAGHAVAN S

Objective

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 a flight that is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases).

So, you have to work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.



Steps

Data Collections:

Need to collect at least 1500 flight schedules.

Used a Web-Scrapping method to collect the data from www.paytm.com/flights with the following columns and data.

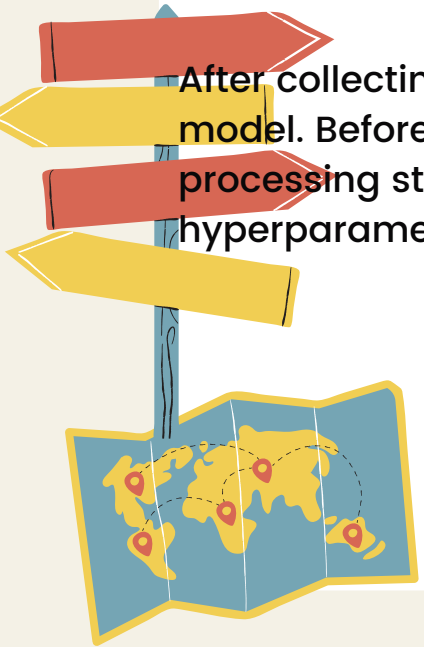
1. Airline name
2. Destination
3. Price.
4. The number of stops.
5. Departure time.
6. Arrival time.

Data Analysis:

Cleaning the data and analyzing the data and the best time to buy the ticket.

Model Building:

After collecting the data, you need to build a machine learning model. Before the model, the building does all data pre-processing steps. Try different models with different hyperparameters and select the best models.



Data Source

Date of Travel: 16th May 2022

Departure City: Delhi

Data Scrapped:

Paytm.com/Flights

Number of Passenger(s): 1

Class: Economy

Schedule

Cities	Cities
Mumbai	Kannur
Goa	Vijayawada
Chennai	Visakapatnam
Hyderabad	Tirupati
Port Blair	Tuticorin
Thiruvanathapuram	Coimbatore
Tiruchirrapalli	Patna
Kochi	Bhubaneshwar

Model Building

After Scrapping the data from the paytm.com/flights website for a sample of 1500 to build a model loaded the data and other necessary things to build a model.

Imported the dataset.

```
df = pd.read_csv("myDataFrame.csv")
df
```

	Unnamed: 0	name	departue_time	arrival_time	departure_city	arrival_city	journey_time	number_of_stops	price
0	0	Air Asia	13:00	14:50	Delhi	Mumbai	1h 50m	Non Stop	4,799
1	1	IndiGo	11:55	13:55	Delhi	Mumbai	2h	Non Stop	4,799
2	2	IndiGo	10:45	12:50	Delhi	Mumbai	2h 5m	Non Stop	4,799
3	3	IndiGo	06:30	08:35	Delhi	Mumbai	2h 5m	Non Stop	4,799
4	4	IndiGo	23:35	01:40	Delhi	Mumbai	2h 5m	Non Stop	4,799
...	...	...	...	...	...	...	...	...	...
1551	72	Air India	21:15	14:25	Delhi	BBI	17h 10m	2	14,253
1552	73	Air India	21:15	14:25	Delhi	BBI	17h 10m	2	14,253
1553	74	Air India	06:10	14:25	Delhi	BBI	32h 15m	2	14,253
1554	75	Air India	06:10	14:25	Delhi	BBI	32h 15m	2	14,253
1555	76	Air India	06:10	14:25	Delhi	BBI	32h 15m	2	14,253

1556 rows × 9 columns

Model Building

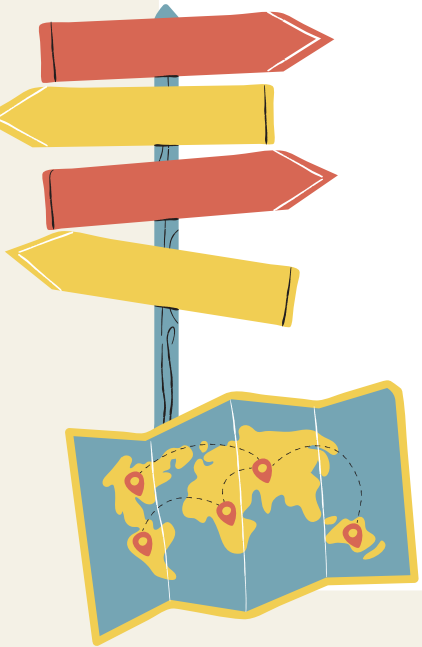
Pre-Processing and cleaning the data for better building a model.

Data Preprocessing

```
#departue_time (converted the departure time)

df["Dep_Hour"] = pd.to_datetime(df.departue_time, format="%H:%M").dt.hour
df["Dep_Min"] = pd.to_datetime(df.departue_time, format="%H:%M").dt.minute
df["departure_time"] = df['Dep_Hour'] + df['Dep_Min'] / 60

#arrival_time (converted the arrival time)

df["Arr_Hour"] = pd.to_datetime(df.arrival_time, format="%H:%M").dt.hour
df["Arr_Min"] = pd.to_datetime(df.arrival_time, format="%H:%M").dt.minute
df["arrival_time"] = df['Arr_Hour'] + df['Arr_Min'] / 60

#journey_time (replaced the journey time)

df['journey_time'] = df['journey_time'].str.replace('h ','.')
df['journey_time'] = df['journey_time'].str.replace('m','')
df['journey_time'] = df['journey_time'].str.replace('h','')
df['journey_time'] = df['journey_time'].astype('float')

#Price (#removed the "," in price column)

df['price'] = df['price'].str.replace(',','')
df['price'] = df['price'].astype('float')

#Number of Stops (Replaced the Non-Stop to "0" for uniform since while scrapping itself limited the data to 1 stop or 2 stops in

df.number_of_stops.replace({"Non Stop": 0}, inplace = True)
df['number_of_stops'] = df['number_of_stops'].astype('float')

df
```

```
df.drop('Unnamed: 0', axis=1, inplace=True) (#Unncessary columns since it is a Serial number column)
df.drop(columns = ['Dep_Hour','Dep_Min'], inplace=True) (#columns not required after conversion)
df.drop(columns = ['Arr_Hour','Arr_Min'], inplace=True) (#columns not required after conversion)
```

```
df.drop('departue_time', axis=1, inplace=True) #since already replaced dropping the main column.
```



Model Building

Explanatory Data Analysis

Exploratory Data Analysis (EDA)

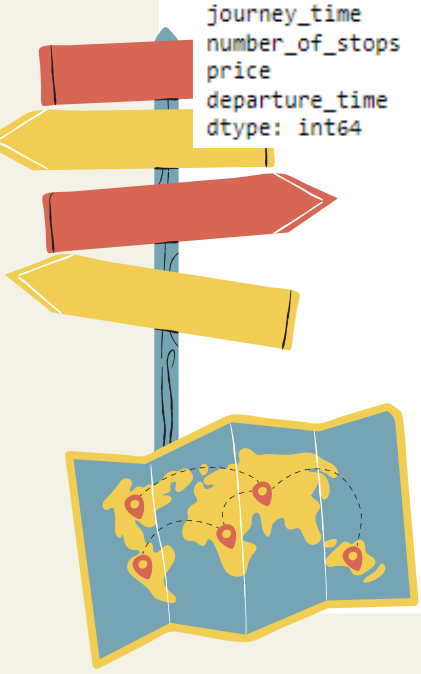
```
print("We have {} Rows and {} Columns in our dataframe".format(df.shape[0], df.shape[1]))
df.head(10)
```

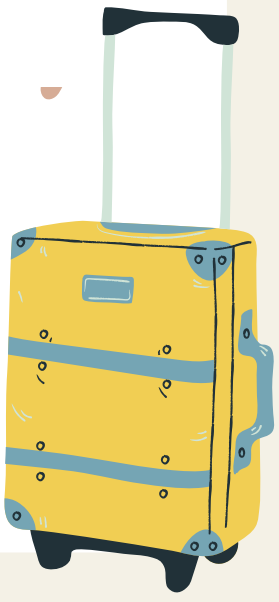
```
We have 1556 Rows and 8 Columns in our dataframe
```

	name	arrival_time	departure_city	arrival_city	journey_time	number_of_stops	price	departure_time
0	Air Asia	14.833333	Delhi	Mumbai	1.5	0.0	4799.0	13.000000
1	IndiGo	13.916667	Delhi	Mumbai	2.0	0.0	4799.0	11.916667
2	IndiGo	12.833333	Delhi	Mumbai	2.5	0.0	4799.0	10.750000
3	IndiGo	8.583333	Delhi	Mumbai	2.5	0.0	4799.0	6.500000
4	IndiGo	1.666667	Delhi	Mumbai	2.5	0.0	4799.0	23.583333
5	IndiGo	19.083333	Delhi	Mumbai	2.5	0.0	4799.0	17.000000
6	IndiGo	9.416667	Delhi	Mumbai	2.5	0.0	4799.0	7.333333
7	IndiGo	4.166667	Delhi	Mumbai	2.1	0.0	4799.0	2.000000
8	Air Asia	6.583333	Delhi	Mumbai	2.1	0.0	4799.0	4.416667
9	IndiGo	7.666667	Delhi	Mumbai	2.1	0.0	4799.0	5.500000

```
df.isna().sum() # checking for missing values and it shows that there is no missing values.
```

```
name              0
arrival_time      0
departure_city    0
arrival_city      0
journey_time      0
number_of_stops   0
price             0
departure_time    0
dtype: int64
```





Model Building

Explanatory Data Analysis

```
df.info()

#the below result shows that non null count and col
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1556 entries, 0 to 1555
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             1556 non-null   object 
 1   arrival_time     1556 non-null   float64
 2   departure_city   1556 non-null   object 
 3   arrival_city     1556 non-null   object 
 4   journey_time     1556 non-null   float64
 5   number_of_stops  1556 non-null   float64
 6   price            1556 non-null   float64
 7   departure_time   1556 non-null   float64
dtypes: float64(5), object(3)
memory usage: 97.4+ KB
```

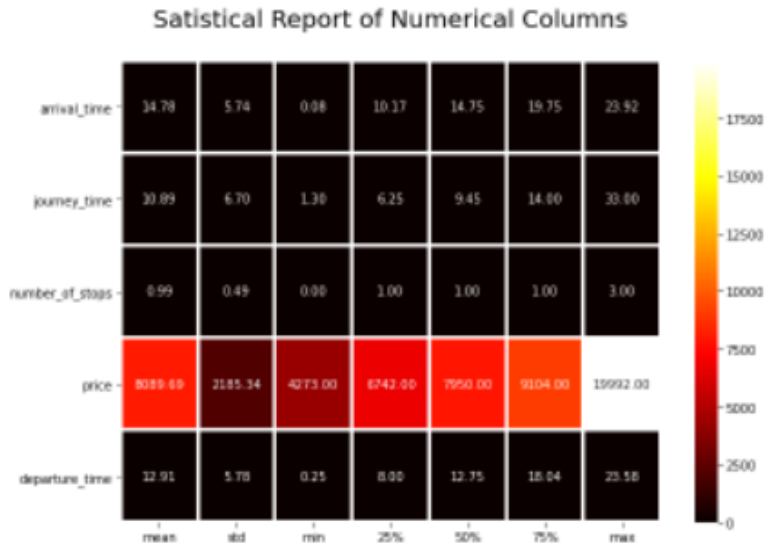
```
df.nunique().sort_values().to_frame("Unique Values"

#checking for unique values in Dataset.
```

	Unique Values
departure_city	1
number_of_stops	4
name	6
arrival_city	16
departure_time	176
arrival_time	226
price	253
journey_time	273

Model Building

Visualizing the Statistical Data

```
# Visualizing the statistical description of numeric datatype columns

plt.figure(figsize = (10,7))
sns.heatmap(round(df.describe()[1:].transpose(),2), linewidth = 2, annot= True, fmt = ".2f", cmap="hot")
plt.title("Satistical Report of Numerical Columns\n", fontsize = 20)
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)
plt.show()

#visualizing the statistical data.
```

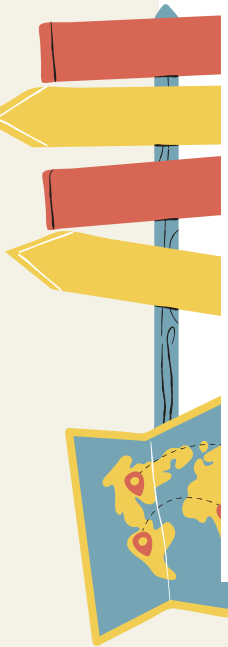


Model Building

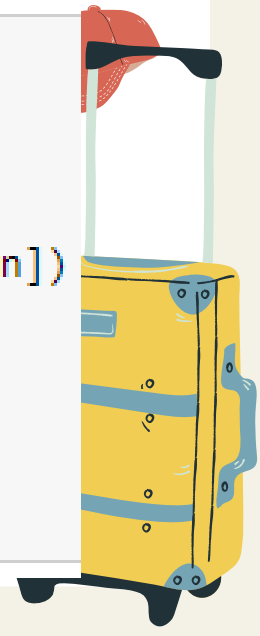
Data Inputs- Logic- Output Relationships:

The input data were initially all object datatype so had to clean the data by removing unwanted information like “h” and “m” from Flight_Duration column and ensuring the numeric data are converted accordingly. I then used Ordinal Encoding method to convert all the categorical feature columns to numeric format.

```
# Ordinal Encoder

oe = OrdinalEncoder()
def ordinal_encode(df, column):
    df[column] = oe.fit_transform(df[column])
    return df

column=["name", "arrival_city"]
df=ordinal_encode(df, column)
df

#encoding the categorical data for better build.
```



```
# Ordinal Encoder

oe = OrdinalEncoder()
def ordinal_encode(df, column):
    df[column] = oe.fit_transform(df[column])
    return df

column=["departure_city"]
df=ordinal_encode(df, column)
df
```



Model Building

Since we had mostly categorical feature data we did not have to worry about outliers and skewness concerns.

After completing the basic cleaning and visualizing create a model.

Splitting the dataset into 2 variables namely 'X' and 'Y' for feature and label

```
]: X = df.drop('price', axis=1)
   Y = df['price']
```

Finding the best random state for building Regression Models

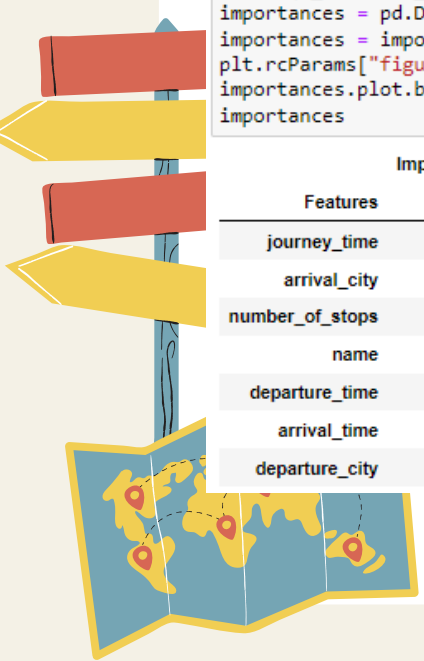
```
]: maxAccu=0
   maxRS=0

   for i in range(1, 1000):
       X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=i)
       lr=LinearRegression()
       lr.fit(X_train, Y_train)
       pred = lr.predict(X_test)
       r2 = r2_score(Y_test, pred)

       if r2>maxAccu:
           maxAccu=r2
           maxRS=i

   print("Best R2 score is", maxAccu*100,"on Random State", maxRS)
```

```
Best R2 score is 56.65620517060137 on Random State 68
```

Feature importance bar graph

```
rf=RandomForestRegressor()
rf.fit(X_train, Y_train)
importances = pd.DataFrame({'Features':X.columns, 'Importance':np.round(rf.feature_importances_,3)})
importances = importances.sort_values('Importance', ascending=False).set_index('Features')
plt.rcParams["figure.figsize"] = (15,8)
importances.plot.bar(color='teal')
importances
```

Features	Importance
journey_time	0.372
arrival_city	0.164
number_of_stops	0.154
name	0.135
departure_time	0.093
arrival_time	0.082
departure_city	0.000

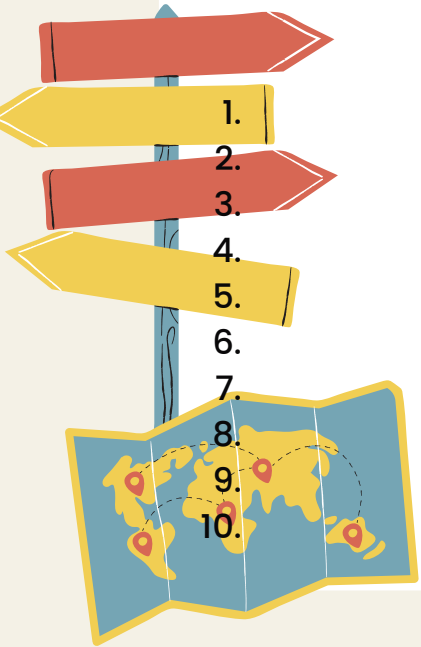


Model Building

Since we had mostly categorical feature data we did not have to worry about outliers and skewness concerns.

After completing the basic cleaning and visualizing create a model.

```
# Regression Model Function

def reg(model, X, Y):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=638)

    # Training the model
    model.fit(X_train, Y_train)

    # Predicting Y_test
    pred = model.predict(X_test)

    # RMSE - a lower RMSE score is better than a higher one
    rmse = mean_squared_error(Y_test, pred, squared=False)
    print("RMSE Score is:", rmse)

    # R2 score
    r2 = r2_score(Y_test, pred, multioutput='variance_weighted')*100
    print("R2 Score is:", r2)

    # Cross Validation Score
    cv_score = (cross_val_score(model, X, Y, cv=5).mean())*100
    print("Cross Validation Score:", cv_score)

    # Result of r2 score minus cv score
    result = r2 - cv_score
    print("R2 Score - Cross Validation Score is", result)
```



Building the below models.

1. Linear Regression Model
2. Ridge Regularization
3. Lasso Regularization
4. Support Vector Regression
5. Decision Tree Regressor
6. Random Forest Regressor
7. K Neighbors Regressor
8. Gradient Boosting Regressor
9. Ada Boost Regressor
10. Extra Trees Regressor

Model Building

Since we had mostly categorical feature data we did not have to worry about outliers and skewness concerns.

After completing the basic cleaning and visualizing create a model.

```
# Linear Regression Model

model=LinearRegression()
reg(model, X, Y)
```

```
RMSE Score is: 1623.4968691281247
R2 Score is: 40.23785866235098
Cross Validation Score: 42.06098514157602
R2 Score - Cross Validation Score is -1.8231264792250386
```

```
# Ridge Regularization

model=Ridge(alpha=1e-2, normalize=True)
reg(model, X, Y)
```

```
RMSE Score is: 1622.0443573184218
R2 Score is: 40.34474693078812
Cross Validation Score: 42.11252697017341
R2 Score - Cross Validation Score is -1.7677800393852934
```

```
# Lasso Regularization

model=Lasso(alpha=1e-2, normalize=True, max_iter=1e5)
reg(model, X, Y)
```

```
RMSE Score is: 1623.3883343473824
R2 Score is: 40.24584888914906
Cross Validation Score: 42.06370909192742
R2 Score - Cross Validation Score is -1.8178602027783626
```

Model Building

Since we had mostly categorical feature data we did not have to worry about outliers and skewness concerns.

After completing the basic cleaning and visualizing create a model.

```
# Support Vector Regression

model=SVR(C=1.0, epsilon=0.2, kernel='poly', gamma='auto')
reg(model, X, Y)
```

```
RMSE Score is: 1614.658457733153
R2 Score is: 40.886784593447295
Cross Validation Score: 31.607360775956124
R2 Score - Cross Validation Score is 9.279423817491171
```

```
# Decision Tree Regressor

model=DecisionTreeRegressor(criterion="poisson", random_state=111)
reg(model, X, Y)
```

```
RMSE Score is: 1813.6628265278662
R2 Score is: 25.417602024776585
Cross Validation Score: -13.970231192678135
R2 Score - Cross Validation Score is 39.38783321745472
```

```
# Random Forest Regressor

model=RandomForestRegressor(max_depth=2, max_features="sqrt")
reg(model, X, Y)
```

```
RMSE Score is: 1631.1959601654867
R2 Score is: 39.6696959850477
Cross Validation Score: 24.177630407370145
R2 Score - Cross Validation Score is 15.492065577677558
```

Model Building

Since we had mostly categorical feature data we did not have to worry about outliers and skewness concerns.

After completing the basic cleaning and visualizing create a model.

```
# K Neighbors Regressor

KNeighborsRegressor(n_neighbors=2, algorithm='kd_tree')
reg(model, X, Y)
```

```
RMSE Score is: 1648.7782698326782
R2 Score is: 38.362111991128685
Cross Validation Score: 23.338768219269863
R2 Score - Cross Validation Score is 15.023343771858823
```

```
# Gradient Boosting Regressor

model=GradientBoostingRegressor(loss='quantile', n_estimators=200, max_depth=5)
reg(model, X, Y)
```

```
RMSE Score is: 1714.7493061602734
R2 Score is: 33.3309124927287
Cross Validation Score: 1.5310021155481834
R2 Score - Cross Validation Score is 31.799910377180517
```

```
# Ada Boost Regressor

model=AdaBoostRegressor(n_estimators=300, learning_rate=1.05, random_state=42)
reg(model, X, Y)
```

```
RMSE Score is: 1602.5989178804589
R2 Score is: 41.766495105646925
Cross Validation Score: 17.422471240544017
R2 Score - Cross Validation Score is 24.34402386510291
```

Model Building

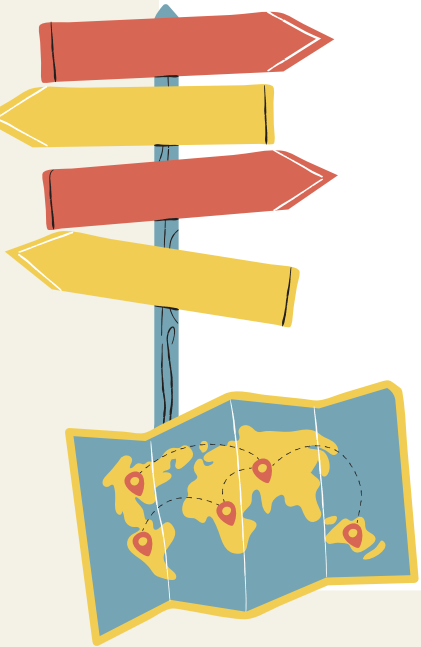
Since we had mostly categorical feature data we did not have to worry about outliers and skewness concerns.

After completing the basic cleaning and visualizing create a model.

```
# Extra Trees Regressor

model=ExtraTreesRegressor(n_estimators=200, max_features='sqrt', n_jobs=6)
reg(model, X, Y)
```

```
RMSE Score is: 1143.8981060279552
R2 Score is: 70.33134070775829
Cross Validation Score: 44.292400658999185
R2 Score - Cross Validation Score is 26.038940048759102
```

With the above models have chosen an Extra Trees Regressor model to hyper tuning and do a girdsearch for betteer result.



Model Building

Hypertuning and Grid-Search CV.

Hyper parameter tuning

```
# Choosing Extra Trees Regressor

fmod_param = {'n_estimators' : [100, 200, 300],
              'criterion' : ['squared_error', 'mse', 'absolute_error', 'mae'],
              'n_jobs' : [-2, -1, 1],
              'random_state' : [42, 251, 340]
             }

GSCV = GridSearchCV(ExtraTreesRegressor(), fmod_param, cv=5)
GSCV.fit(X_train,Y_train)
```

```
GridSearchCV(cv=5, estimator=ExtraTreesRegressor(),
             param_grid={'criterion': ['squared_error', 'mse', 'absolute_error',
                                       'mae'],
                         'n_estimators': [100, 200, 300], 'n_jobs': [-2, -1, 1],
                         'random_state': [42, 251, 340]})
```

```
GSCV.best_params_
```

```
{'criterion': 'mae', 'n_estimators': 200, 'n_jobs': -2, 'random_state': 251}
```

```
Final_Model = ExtraTreesRegressor(criterion='mae', n_estimators=300, n_jobs=-2, random_state=251)
Model_Training = Final_Model.fit(X_train, Y_train)
fmod_pred = Final_Model.predict(X_test)
fmod_r2 = r2_score(Y_test, fmod_pred, multioutput='variance_weighted')*100
print("R2 score for the Best Model is:", fmod_r2)
```

```
R2 score for the Best Model is: 71.81802743705863
```

After fine tuning it Extra Trees Regressor the updated result is 71.818.

Model Building

Predicting the price with the samples after the Model Built.

Prediction of Flight Prices

```
Predicted_Price = Final_Model.predict(X)
# Checking the predicted price details in dataframe format
predicted_output = pd.DataFrame()
predicted_output['Flight Price Predicted'] = Predicted_Price
predicted_output['Flight Price Actual'] = df["price"]
predicted_output
```

	Flight Price Predicted	Flight Price Actual
0	4799.000000	4799.0
1	4799.000000	4799.0
2	4800.753333	4799.0
3	4800.753333	4799.0
4	4799.000000	4799.0
...	...	...
1551	14253.000000	14253.0
1552	14253.000000	14253.0
1553	14253.000000	14253.0
1554	14253.000000	14253.0
1555	14253.000000	14253.0

1556 rows × 2 columns

Model Building

Saving the best model using joblib.

Saving the best model

```
filename = "FlightPricePrediction.pkl"
joblib.dump(Final_Model, filename)
```

```
['FlightPricePrediction.pkl']
```

Conclusions:

From the above EDA we can easily understand the relationship between features and we can even see which things are affecting the price of flights. 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.

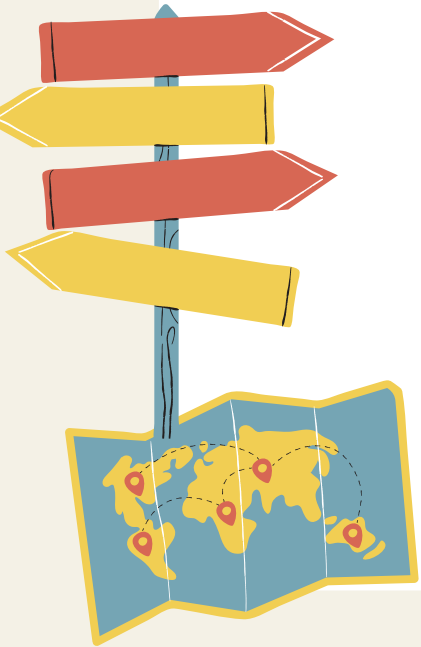
This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.

Model Building

Key Findings and Conclusions of the Study:

Features like flight duration, number of stops during the journey, date of travel, and departure time plays a major role in the flight prices.

As we have seen, the prediction is showing a similar relationship with the actual price from the scrapped data set. This means the model is predicted correctly and it could help airlines by predicting what prices they can maintain. It could also help customers to predict future flight prices and plan the journey accordingly because it is difficult for airlines to maintain prices since it changes dynamically due to different conditions. Hence by using Machine Learning techniques we can solve this problem. The above research will help our client to study the latest flight price market and with the help of the model built he can easily predict the price ranges of the flight, and also will help him to understand Based on what factors the fight price is decided.



Model Building

Learning Outcomes of the Study in respect of Data Science :

The visualization part helped me to understand the data as it provides a graphical representation of huge data. It assisted me to understand the feature importance, outliers/skewness detection, and comparing the independent-dependent features. Data cleaning is the most important part of model building and therefore before model building, I made sure the data is cleaned.

Have generated multiple regression machine learning models to get the best model wherein I found Extra Trees Regressor Model being the best based on the metrics I have used.

Acknowledgment:

Have used various sources from google and other external websites for a reference and to build a better model as well as to understand the dataset and aviation industry in a fair usage without impacting the service provider.

Model Building

Learning Outcomes of the Study in respect of Data Science :

The visualization part helped me to understand the data as it provides a graphical representation of huge data. It assisted me to understand the feature importance, outliers/skewness detection, and comparing the independent-dependent features. Data cleaning is the most important part of model building and therefore before model building, I made sure the data is cleaned.

Have generated multiple regression machine learning models to get the best model wherein I found Extra Trees Regressor Model being the best based on the metrics I have used.

Acknowledgment:

Have used various sources from google and other external websites for a reference and to build a better model as well as to understand the dataset and aviation industry in a fair usage without impacting the service provider.

Thank You

"Do more of what you love."