PROJECT REPORT

1.INTRODUCTION

A person who already has reserved a ticket for a flight realizes how powerfully the price of the ticketswitches . Airline utilizes progressed techniques considered Revenue Management to accomplish acharacteristic esteeming technique . The most affordable ticket available changes over a course of time.The expense of the booking may be far and wide.

This esteeming technique normally alters the costaccording to the different times in a day namely forenoon, evening, or night. Expenses for the flight maysimilarly alter according to the different seasons in a year like summers, rainy and winters, also during theperiod of festivals. The buyers would be looking for the cheapest ticket while the outrageous objective ofthe transporter would be generating more and more revenue.

Travelers for the most part attempt to buythe ticket ahead of their departure day. The reason would be their belief that the prices might be the highestwhen they would make a booking much nearer to the day of their flight but conventionally this isn'tverifiable. The buyer might wrap up paying more than they should for a comparable seat.

Considering thechallenges faced by the travellers for getting an affordable seat, various strategies are utilized which willextract a particular day on which the fare will be the least. For this purpose, Machine Learning comes intothe picture. Gini and Groves developed a model using PLSR, to predict the appropriate time to book theseats .

OVERVIEW

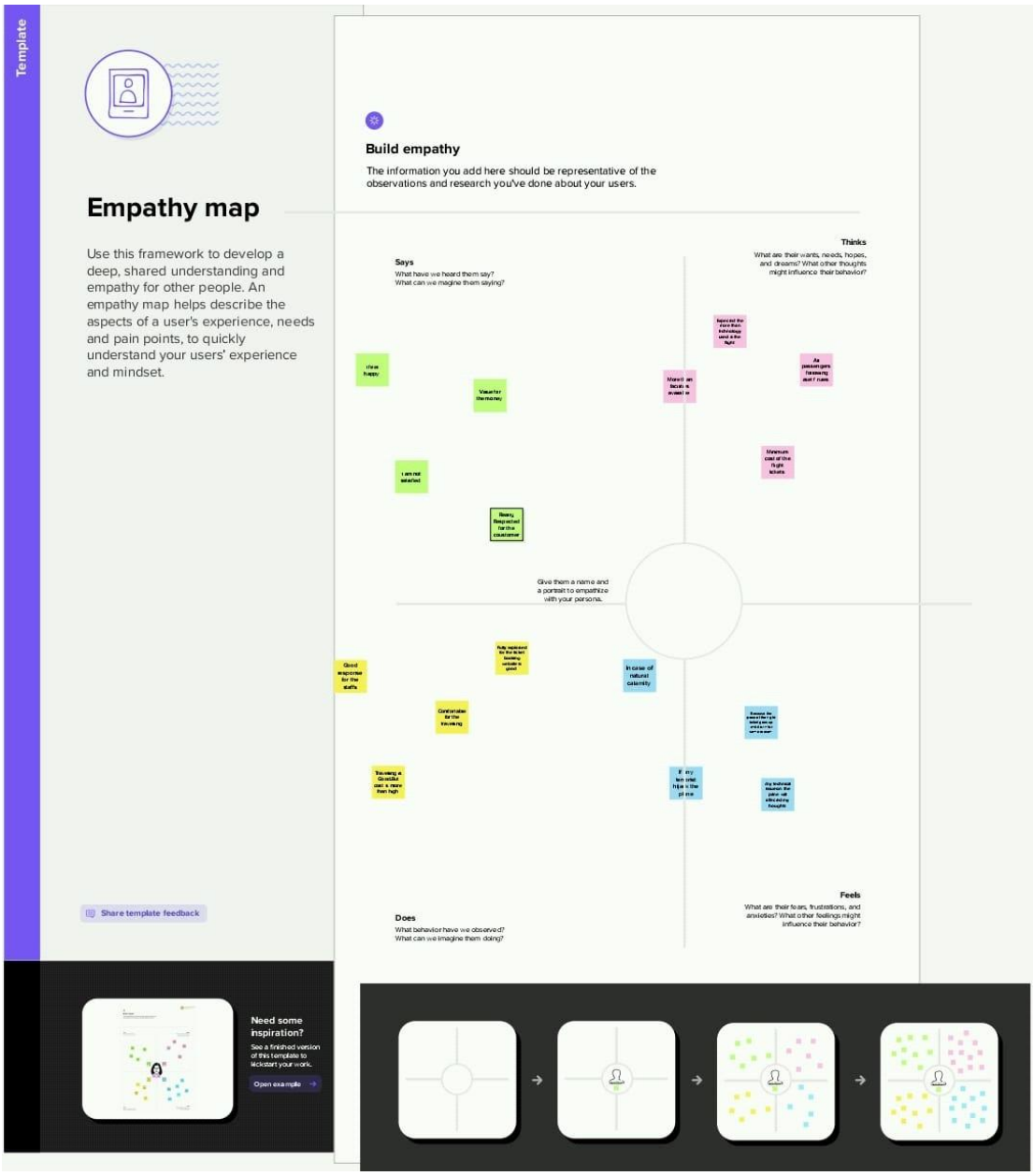
In this article, we will be analyzing the flight fare prediction using Machine Learning dataset using essential exploratory data analysis techniques then will draw some predictions about the price of the flight based on some features such as what type of airline it is, what is the arrival time, what is the departure time, what is the duration of the flight, source, destination and more.

PURPOSE

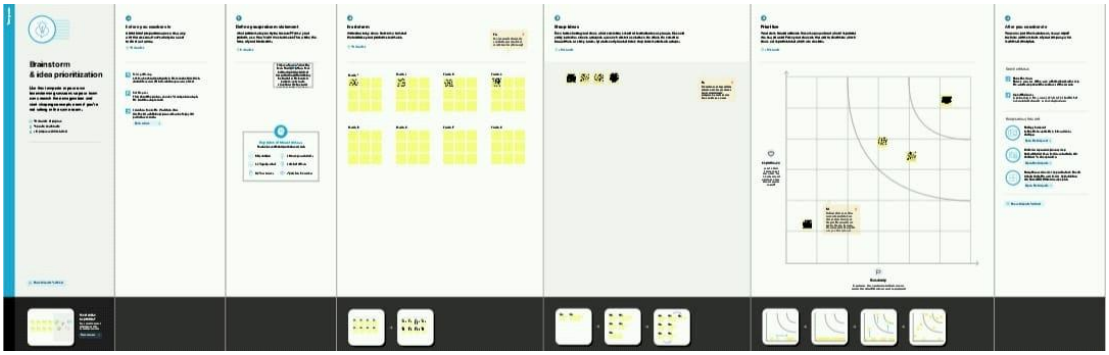
The purpose of this article is to predict flight prices given the various parameters. Data used in this article is publicly available at Kaggle. This will be a regression problem since the target or dependent variable is the price (continuous numeric value).

PROBLEM DEFINITION & DESIGN THINKING

EMPATHY MAP



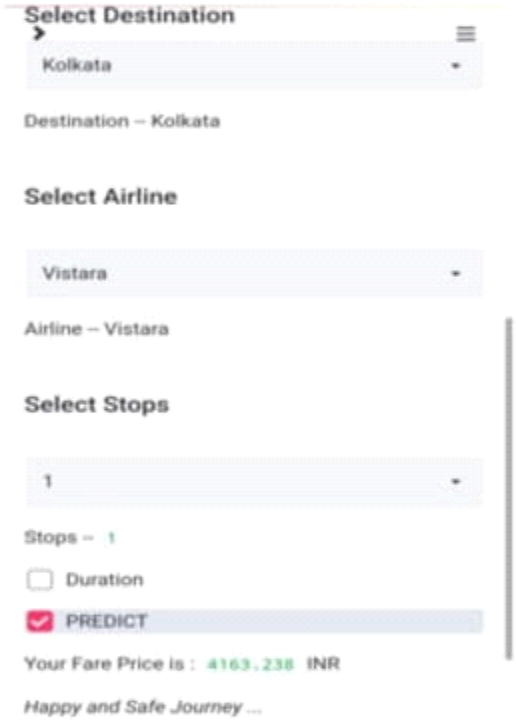
IDEATION & BRAINSTROMING MAP



RESULT

TABLE I. Results

Algorithm	Training Accuracy	Testing accuracy
XGBoost	0.92	0.77
Random Forest	0.95	0.78
Decision Tree	0.97	0.67



ADVANTAGES

The objective of this Model that we are building is to predict flight prices based on given parameters. Data employed in this is publicly available at Kaggle. This can be a regression problem since the target or label is the price (continuous numeric value). Airline companies use complex algorithms to calculate flight prices given various conditions present at that individual time.

These methods take financial, marketing, and various social factors into consideration to predict flight prices. Nowadays, the amount of individuals using flights has increased significantly. It's difficult for airlines to keep up with prices since prices change dynamically because of different conditions.

That’s why we are going to attempt to use machine learning to unravel this problem. This will help airlines by predicting what prices they will maintain. It may also help customers to predict future flight prices and plan their journey accordingly.

We will analyze the flight fare prediction using Machine Learning, in the dataset, we will be using a few necessary features to draw some predictions about the price of the flight like what type of airline it is, what is the arrival time, what is departure time, what is the duration of the flight, source, destination, and more.

DISADVANTAGES

It is critical for airlines to be capable of predicting airfare trends at the market segment level in order to alter strategies and resources for a given route. Scientific literatures on business segment price prediction, on the other hand.

use biased conventional predictive methods, including such linear regression , and thus are founded on the supposition that the selected variables have a linear relationship, that might not be true in most cases.

Proposed research Prediction of airfare prices utilizing machine learning approach, A dataset of 1814 Aegean Airways data flights was gathered and utilized to develop the machine learning technique for the study effort.

Various figures of variables have been used to train the classifiers to demonstrate how feature extraction might affect validity of the model.

APPLICATION

One key observation that might not be immediately apparent when reviewing the most important variables in the model is that 7 out of the top 10 predictive variables are available at the time of booking, before the customer has purchased the flight. This means that when a passenger is engaged with the airline, the supplier can take real-time decisions to adapt the product offering and possibly price, according to the purpose of travel.

This is a very powerful result from a sales perspective as our analysis, previous research and natural intuition all support the idea that the purchasing behaviour is influenced by the purpose of travel. Although the airline industry recognizes the need to personalize, presently only very few companies are actively personalizing offers at the booking stage. The primary reason for this is the lack of ways that customers can be segmented with limited information and gaps in understanding customer purchasing behaviour. The methodology, results and discussions presented in this paper all contribute to reduce this problem.

Furthermore, the purpose of trip categorization along with a flexible e-commerce platform opens several other experimentation possibilities that can be conducted during the selling process. Using the purpose of travel prediction, an airline can study how different groups react to different offers to continue building an understanding of the customer behaviour. With the ever-changing landscape of the industry, this process of experimentation should be a continuous process embedded in the e-commerce practices of the airlines.

Historically, the airline business was at the forefront of this type of analytical thinking. For instance, the Saturday night stay variable was often embedded in the ticketing pricing structure to provide a discount for passengers staying overnight on a Saturday. However, the disruption that the internet brought about in the form of a new sales distribution technology, further coupled with the disruption from low-cost airlines, has changed the business landscape and increased pressure on airlines, sometimes putting them in a reactive position as opposed to an innovative position.

In reaction to this, many airlines have simplified their pricing structures to one-way prices as opposed to round trip prices. This clearly benefits the consumer and simplifies backend processes for the airline's side; however, it may have diluted revenue in the process of doing so. By adapting to and adopting new technologies, airlines can now retrain the simplicity of one-way pricing and improve revenue with targeted and useful product offerings.

CONCLUSION

Three machine learning models were examined in this case study to forecast the average flight price at the business segment level. We used training data to train the training data and test data to test it. These records were used to extract a number of characteristics. Our

suggested model can estimate the quarterly average flight price using attribute selection strategies.

To the highest possible standard, much prior studies into flight price prediction using the large dataset depended on standard statistical approaches, which have their own limitations in terms of underlying issue estimates and hypotheses. To our knowledge, no other research have included statistics from holidays, celebrations, stock market price fluctuations, depression, fuel price, and socioeconomic information to estimate the air transport market sector; nonetheless, there are numerous restrictions.As example, neither of the databases provide precise information about ticket revenue, including such departing and arrival times and days of the week.

This framework may be expanded in the future to also include airline tickets payment details, that can offer more detail about each area, such as timestamp of entry and exit, seat placement, covered auxiliary items, and so on. By merging such data, it is feasible to create a more robust and complete daily and even daily flight price forecast model. Furthermore, a huge surge of big commuters triggered by some unique events might alter flight costs in a market sector.

Thus, incident data will be gathered from a variety of sources, including social media sites and media organizations, to supplement our forecasting models. We will also examine specific technological Models, such as Deeper Learning methods, meanwhile striving to enhance existing models by modifying their hyper-parameters to get the optimum design for airline price prediction.

FUTURE SCOPE

One key observation that might not be immediately apparent when reviewing the most important variables in the model is that 7 out of the top 10 predictive variables are available at the time of booking, before the customer has purchased the flight. This means that when a passenger is engaged with the airline, the supplier can take real-time decisions to adapt the product offering and possibly price, according to the purpose of travel. This is a very powerful result from a sales perspective as our analysis, previous research and natural intuition all support the idea that the purchasing behaviour is influenced by the purpose of travel.

Although the airline industry recognizes the need to personalize, presently only very few companies are actively personalizing offers at the booking stage. The primary reason for this is the lack of ways that customers can be segmented with limited information and gaps in understanding customer purchasing behaviour. The methodology, results and discussions presented in this paper all contribute to reduce this problem.

Furthermore, the purpose of trip categorization along with a flexible e-commerce platform opens several other experimentation possibilities that can be conducted during the selling process. Using the purpose of travel prediction, an airline can study how different groups react to different offers to continue building an understanding of the customer behaviour. With the ever-changing landscape of the industry, this process of experimentation should be a continuous process embedded in the e-commerce practices of the airlines.

Historically, the airline business was at the forefront of this type of analytical thinking. For instance, the Saturday night stay variable was often embedded in the ticketing pricing structure to provide a discount for passengers staying overnight on a Saturday. However, the disruption that the internet brought about in the form of a new sales distribution technology, further coupled with the disruption from low-cost airlines, has changed the business landscape and increased pressure on airlines, sometimes putting them in a reactive position as opposed to an innovative position.

In reaction to this, many airlines have simplified their pricing structures to one-way prices as opposed to round trip prices. This clearly benefits the consumer and simplifies backend processes for the airline's side; however, it may have diluted revenue in the process of doing so. By adapting to and adopting new technologies, airlines can now retrain the simplicity of one-way pricing and improve revenue with targeted and useful product offerings.

Throughout the experiment, the inclusion and exclusion of features had a significantly higher impact than the type of classifier used once the best classifier was identified. Table 4 shows an ordered list of the top 10 features including their contribution to the predictor computed according to Breiman et al. (1984). The list is a representative list of the features that were commonly observed as having the highest contribution to the results throughout all experiments.

Two of the most important calculated features were the length of the journey (SINGLE JOURNEY TIME) and the indication that the passenger stayed overnight on a Saturday (SATURDAY NIGHT STAY). This feature has the intuitive understanding that a passenger travelling on business in a Monday–Friday business environment, does not need to overnight on a Saturday. Some of the other top features also easily relate to preferences of business passengers; higher price paid (SINGLE JOURNEY REVENUE), solo travel (NUM PASSENGERS IN BOOKING = 1) and booking closer to the departure date (NUMBER OF DAYS OUT).

The other features in the top 10 list are the more exploratory features employed for this task. These features either are more challenging to compute, such as the historic passenger profile features, which requires a separate entity resolution procedure on the complete passenger dataset, or constitute data elements that historically were not commonly available.

The importance of the historic customer travel profile can be observed in the MEDIAN JOURNEY TIME and CUSTOMER AVERAGE FARE features. Interestingly, these two features are representations of the historic purpose of travel of the same customer. Intuitively, this means that passengers with a historic business profile are more likely to be travelling on business in the current journey.

Table 4. Top 10 features.

Name of feature	Weight
SINGLE JOURNEY TRIP	0.10
NUMBER OF PASSENGERS IN BOOKING	0.10
SATURDAY NIGHT STAY	0.09
MINIMUM AGE	0.05
NUMBER OF DAYS OUT	0.05
*MEDIAN JOURNEY TIME	0.04
*CUSTOMER AVERAGE FARE	0.04
SINGLE JOURNEY MILEAGE	0.04
COMMON EMAIL DOMAIN	0.04
SINGLE JOURNEY REVENUE	0.04

**Historic customer profile features.

A separate experiment was conducted to test this hypothesis. The single journey classifier was used to predict all the prior customer trips. The output of this classifier was aggregated into the proportion of historic business and leisure trips, normalized by volume. This feature was then used in a second classifier with the single journey details.

The result of this classifier improved the performance over the single journey only classifier and had almost identical results to the classifier including the historic customer profile features. This finding suggests that if historic customer features are not available, the first model can be used to populate the missing values of the customer history.

The MINIMUM AGE feature indicates that groups of passengers travelling with children or young adults are more likely to be travelling for leisure purposes. Finally, the importance of the COMMON EMAIL DOMAIN feature can be explained by the notion that business passengers tend to use their work email when travelling on business, as opposed to their personal email address, which is more likely to be with a common email service provider.

There were a number of other derived features that were not as significant as we expected them to be. For instance, a feature was derived to infer the case when a family was travelling together. This feature was intended to cover the hypothesis that families travelling together typically do so for leisure purposes. In this case, the feature was superseded in importance by other overlapping features. In general, it is more important to try to identify features that correctly

identify the smallest class, in this case business travellers than the other way round.

Table 5 shows the results of the random forest classifier comparing the 3 different sets of feature categories available. Both the results in this table and the results of the feature importance in Table 4 shows that the current undertaken journey features are the most important ones. The combined single journey data and historic pattern data performs had better than the single journey details alone. These features also feature in the top 10 features from the total number of 161 features.

This illustrates the importance of uniquely resolving the customer identity and applying the result to help the prediction of other machine learning tasks. Finally, we observed that the passenger survey details do not contribute to the improvement of the model and that customer satisfaction ratings are not correlated with the purpose of travel. This finding was further analysed and confirmed in a separate analysis.

Table 5. Feature results.

Class	Single Journey			+ Historic			+ Historic + Survey		
Empty Cell	Pre	Rec	F-m	Pre	Rec	F-m	Pre	Rec	F-m
Business	0.7889	0.5449	0.6443	0.8169	0.5594	0.6639	0.8195	0.5642	0.6682
Leisure	0.9160	0.9715	0.9429	0.9187	0.9755	0.9462	0.9196	0.9757	0.9468
Overall	0.8953	0.9017	0.8941	0.9021	0.9073	0.9000	0.9032	0.9083	0.9012

The results of the prediction model assent to previous literature (Dresner, 2006, Martínez-Garcia et al., 2012) that business and leisure passengers sharing the same cabin share similar preferences and behaviours. The division between groups also corroborates previous hypothesis that business passengers travelling on low-cost airlines are still cost conscious and still somewhat price elastic, as the average fare figures amongst the different customer segments (in Table 5) suggest.

This increased homogeneity between passenger behaviours makes the prediction task more challenging. The biggest challenge in the prediction task is to identify the segment of business passengers that behaves like leisure passengers, i.e., the remaining 45% of passengers that are not recalled by the model. From a practical commercial perspective, these customers might present an opportunity to upsell, however the effort required in identifying and targeting such passengers might not be economically feasible.

The single journey model derived from our primary

dataset was also applied to a secondary dataset of a different low-cost airline. The overall f1 score on this dataset was 0.86. Upon further investigation, we determined that the reason for this was missing data in the journey length and Saturday night stay features. Over 37% of passengers in the second airline (as opposed to 2% in the first) purchased one-way tickets.

This behaviour significantly reduced the number of records where the journey length and Saturday night stay features could be computed. A model trained without these features on the primary dataset resulted in a similar performance degradation. This highlights the importance of these features and the importance of considering the characteristics of each dataset before generalizing models.

APPENDIX

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoosti
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matri
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
```

```
data=pd.read_csv("Data_Train.csv")
```

```
data.head()
```

```
for i in category:
    print(i, data[i].unique())
```

```
#We now split the Date column to extract the 'Date', 'Month' and 'Year' values, and store them in new

data.Date_of_Journey=data.Date_of_Journey.str.split('/')
```

```
data.Date_of_Journey
```

```
#Treating the data_column


data['Date']=data.Date_of_Journey.str[0]
data['Month']=data.Date_of_Journey.str[1]
data['Year']=data.Date_of_Journey.str[2]
```

```
data.Total_Stops.unique()
```

```
array(['non-stop', '2 stops', '1 stop', '3 stops', '4 stops'],
      dtype=object)
```

```
data.Route=data.Route.str.split('→')
data.Route
```

```
data['City1']=data.Route.str[0]
data['City2']=data.Route.str[1]
data['City3']=data.Route.str[2]
data['City4']=data.Route.str[3]
data['City5']=data.Route.str[4]
data['City6']=data.Route.str[5]
```

```
data.Dep_Time=data.Dep_Time.str.split(':')
```

```
data['Dep_Time_Hour']=data.Dep_Time.str[0]
data['Dep_Time_Mins']=data.Dep_Time.str[1]
```

```
data.Arrival_Time=data.Arrival_Time.str.split(' ')
```

```
data['Arrival_date']=data.Arrival_Time.str[1]
data['Time_of_Arrival']=data.Arrival_Time.str[0]
```

```
data['Time_of_Arrival']=data.Time_of_Arrival.str.split(':')
```

```
data['Arrival_Time_Hour']=data.Time_of_Arrival.str[0]
data['Arrival_Time_Mins']=data.Time_of_Arrival.str[1]
```

```
data.Duration=data.Duration.str.split(' ')
```

```
data['Travel_Hours']=data.Duration.str[0]
data['Travel_Hours']=data['Travel_Hours'].str.split('h')
data['Travel_Hours']=data['Travel_Hours'].str[0]
data.Travel_Hours=data.Travel_Hours
data['Travel_Mins']=data.Duration.str[1]

data.Travel_Mins=data.Travel_Mins.str.split('m')
data.Travel_Mins=data.Travel_Mins.str[0]
```

```
data.Total_Stops.replace('non_stop',0,inplace=True)
data.Total_Stops=data.Total_Stops.str.split(' ')
data.Total_Stops=data.Total_Stops.str[0]
```


```
data.Total_Stops.replace('non_stop',0,inplace=True)
data.Total_Stops=data.Total_Stops.str.split(' ')
data.Total_Stops=data.Total_Stops.str[0]
```


```
data.Additional_Info.unique()
```

```
array(['No info', 'In-flight meal not included',
       'No check-in baggage included', '1 Short layover', 'No Info',
       '1 Long layover', 'Change airports', 'Business class',
       'Red-eye flight', '2 Long layover'], dtype=object)
```

```
data.Additional_Info.replace('No Info','No info',inplace=True)
```

```
data.isnull().sum()
```

```
data.drop(['City4','City5','City6'],axis=1,inplace=True)
```

```
data.drop(['Date_of_Journey','Route','Dep_Time','Arrival_Time','Duration'],axis=1, inplace=True)
data.drop(['Time_of_Arrival'],axis=1,inplace=True)
```

```
#filling City3 as None, the missing values are less
data['City3'].fillna('None',inplace=True)
```

```
#filling Arrival_Date as Departure_Date
data['Arrival_date'].fillna(data['Date'],inplace=True)
```

```
#filling Travel_Mins as Zero(0)
data['Travel_Mins'].fillna(0,inplace=True)
```

```
data.info()
```

```
#chcnging the numerical columns from object to int
#data.Total_Stops=data.Total_Stops.astype('int64')
data.Date=data.Date.astype('int64')
data.Month=data.Month.astype('int64')
data.Year=data.Year.astype('int64')
data.Dep_Time_Hour=data.Dep_Time_Hour.astype('int64')
data.Dep_Time_Hour=data.Dep_Time_Hour.astype('int64')
data.Dep_Time_Mins=data.Dep_Time_Mins.astype('int64')
data.Arrival_date=data.Arrival_date.astype("int64")
data.Arrival_Time_Hour=data.Arrival_Time_Hour.astype('int64')
data.Arrival_Time_Mins=data.Arrival_Time_Mins.astype('int64')
#data.Travel_Hours=data.Travel_Hours.astype('int64')
data.Travel_Mins=data.Travel_Mins.astype('int64')
```

```
data[data['Travel_Hours']=='5m']
```

```
data.drop(index=6474,inplace=True,axis=0)
```

```
data.Travel_Hours=data.Travel_Hours.astype('int64')
```

```
#Creating list of Different types of columns
categorical=['Airline','Source','Destination','Additional_Info','City1']
numerical=['Total_Stops','Date','Month','Year','Dep_Time_Hour','Dep_Time_Mins','Arrival_date','Arrival_Time_Hour',
           'Arrival_Time_Mins','Travel_Hours','Travel_Mins']
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
data.Airline=le.fit_transform(data.Airline)
data.Source=le.fit_transform(data.Source)
data.Destination=le.fit_transform(data.Destination)
data.Total_Stops=le.fit_transform(data.Total_Stops)
data.City1=le.fit_transform(data.City1)
data.City2=le.fit_transform(data.City2)
data.City3=le.fit_transform(data.City3)
data.Additional_Info=le.fit_transform(data.Additional_Info)
data.head()
```

```
data.head()
```

```
data.describe()
```

Price

```
import seaborn as sns
c=1
plt.figure(figsize=(20,45))

for i in categorical:
    plt.subplot(6,3,c)
    sns.countplot(data[i])
    plt.xticks(rotation=90)
    plt.tight_layout(pad=3.0)
    c=c+1

plt.show()
```

```
plt.figure(figsize=(15,8))
sns.distplot(data.Price)
```

```
sns.heatmap(data.corr(),annot=True)
```

```
import seaborn as sns
sns.boxplot(data['Price'])
```

```
y = data['Price']
x = data.drop(columns=['Price'],axis=1)
```

```
### Scaling the Data
```

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

```
x_scaled = ss.fit_transform(x)
```

```
x_scaled = pd.DataFrame(x_scaled,columns=x.columns)
x_scaled.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)
```

```
x_train.head()
```

```
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, AdaBoostRegressor
rfr=RandomForestRegressor()
gb=GradientBoostingRegressor()
ad=AdaBoostRegressor()
```

```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

for i in [rfr,gb,ad]:
    i.fit(x_train,y_train)
    y_pred=i.predict(x_test)
    test_score=r2_score(y_test,y_pred)
    train_score=r2_score(y_train, i.predict(x_train))
    if abs(train_score-test_score)<=0.2:
        print(i)

        print("R2 score is",r2_score(y_test,y_pred))
        print("R2 for train data",r2_score(y_train, i.predict(x_train)))
        print("Mean Absolute Error is",mean_absolute_error(y_pred,y_test))
        print("Mean Squared Error is",mean_squared_error(y_pred,y_test))
        print("Root Mean Sqaured Error is", (mean_squared_error(y_pred,y_test,squared=False)))
```

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

knn=KNeighborsRegressor()
svr=SVR()
dt=DecisionTreeRegressor()

for i in [knn,svr,dt]:
    i.fit(x_train,y_train)
    y_pred=i.predict(x_test)
    test_score=r2_score(y_test,y_pred)
    train_score=r2_score(y_train,i.predict(x_train))
    if abs(train_score-test_score)<=0.1:
        print(i)
        print('R2 Score is',r2_score(y_test,y_pred))
        print('R2 Score for train data',r2_score(y_train,i.predict(x_train)))
        print('Mean Absolute Error is',mean_absolute_error(y_test,y_pred))
        print('Mean Squared Error is',mean_squared_error(y_test,y_pred))
        print('Root Mean Squared Error is',(mean_squared_error(y_test,y_pred,squared=False)))
```

```
from sklearn.model_selection import cross_val_score
for i in range(2,5):
    cv=cross_val_score(rfr,x,y,cv=i)
    print(rfr,cv.mean())
```

```
from sklearn.model_selection import RandomizedSearchCV
```

```
param_grid={'n_estimators':[10,30,50,70,100],'max_depth':[None,1,2,3],
            'max_features':['auto','sqrt']}
rfr=RandomForestRegressor()
rf_res=RandomizedSearchCV(estimator=rfr,param_distributions=param_grid,cv=3,verbose=2,n_jobs=-1)

rf_res.fit(x_train,y_train)
```

```
rfr=RandomForestRegressor(n_estimators=10,max_features='sqrt',max_depth=None)
rfr.fit(x_train,y_train)
y_train_pred=rfr.predict(x_train)
y_test_pred=rfr.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
knn=KNeighborsRegressor(n_neighbors=2,algorithm='auto',metric_params=None,n_jobs=-1)
knn.fit(x_train,y_train)
y_train_pred=knn.predict(x_train)
y_test_pred=knn.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
rfr=RandomForestRegressor(n_estimators=10,max_features='sqrt',max_depth=None)
rfr.fit(x_train,y_train)
y_train_pred=rfr.predict(x_train)
y_test_pred=rfr.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
price_list=pd.DataFrame({'Price':prices})
```

```
price_list
```

```
import pickle
pickle.dump(rfr,open('model1.pkl','wb'))
```


```
import pickle
pickle.dump(rfr,open('model1.pkl','wb'))
```


```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

```
model = pickle.load(open(r"model1.pkl",'rb'))
```

```
@app.route("/home")
def home():
    return render_template('home.html')
```

```python
@app.route("/predict")
def home1():
    return render_template('predict.html')



@app.route("/pred", methods=['POST','GET'])
def predict():
    x = [[int(x) for x in request.form.values()]]
    print(x)

    x = np.array(x)
    print(x.shape)


    print(x)
    pred = model.predict(x)
    print(pred)
    return render_template('submit.html', prediction_text=pred)
```

```python
if __name__ == "__main__":
    app.run(debug=False)
```

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

r