

# Flight Fair Prediction

## CHAPTER 1

### INTRODUCTION

Someone who purchase flight tickets frequently would be able to predict the right time to procure a ticket to obtain the best deal. Many airlines change ticket prices for their revenue management. The airline may increase the prices when the demand is to be expected to increase the capacity. To estimate the minimum airfare, data for a specific air route has been collected including the features like departure time, arrival time and airways over a specific period. Features are extracted from the collected data to apply Machine Learning (ML) models.

#### Flight Price Prediction:

The flight ticket buying system is to purchase a ticket many days prior to flight takeoff so as to stay away from the effect of the most extreme charge. Mostly, aviation routes don't agree this procedure. Plane organizations may diminish the cost at the time, they need to build the market and at the time when the tickets are less accessible. They may maximize the costs. So the cost may rely upon different factors. To foresee the costs this venture uses AI to exhibit the ways of flight tickets after some time. All organizations have the privilege and opportunity to change it's ticket costs at anytime. Explorer can set aside cash by booking a ticket at the least costs. People who had travelled by flight frequently are aware of price fluctuations. The airlines use complex policies of Revenue Management for execution of distinctive evaluating systems [1]. The evaluating system as a result changes the charge depending on time, season, and festive days to change the header or footer on successive pages. The ultimate aim of the airways is to earn profit whereas the customer searches for the minimum rate. Customers usually try to buy the ticket well in advance of departure date so as to avoid hike in airfare as date comes closer. But actually this is not the fact. The customer may wind up by giving more than they ought to for the same seat.

## 1.2 Objectives:

The main objective of the project is about machine learning algorithm to predict the species of the flowers according to the characteristics of the Iris dataset. This model is trained to learn patterns from the data set based on those features.

1. It shows the information of the flowers about shape, color, width, length in the form of real values like int, float or binary and measure the image by merging different feature descriptors which identify the image more efficiently.
2. After extracting the features and labels from Iris dataset, we need to train the system with the help of scikit-learn. We create machine models, which classify the Iris flower into sub species.
3. Iris dataset has four attributes, they are: sepal\_length, sepal\_width, petal\_length and petal\_width.
4. It shows the relationship graph of the sepal and petal length.
5. We use some algorithms that predict the correct accuracy outcomes of the Iris flower.
6. We can also use score method for the object which we will compute.

## 1.3 Methodology to be followed

In this project we use python and supervised machine learning techniques. It specifies existing Iris flower dataset is preloaded and is used for clustering into different species.

In this we create a machine model that can learn from the measurements of these Irises whose species are already known, so that we can predict the species for new Irises that have been found.

In this project, we will predict the species of Iris, having possible outputs are called classes. Every Iris in the dataset belongs to one of three classes considered in the model.

The desired output for a single data is the species of the flower considering its features.

## 1.4 Expected outcomes

- The shape of the data array is the number of samples multiplied by the number of features.
- The target contains a Price of a Flight based on
  - Date of journey
    - Time of Departure
    - Place of Departure
    - Time of Arrival
    - Place of Destination/Arrival
    - Airway company
- We will the model's performance, we show it new data for which we have labels and this is usually done by splitting the data we collected.
- We will make the predictions using the model on any new data for which we might not know the corrected labels. We will calculate by the shape and number of samples multiplied by the number of features.
- Model predicts the Price of The Flight
- We can also use score method for the object, which will compute the test set accuracy.
- We can expect the model to be correct 78% of the time for predicting the species of new Irises.

## CHAPTER 2

### REQUIREMENT SPECIFICATION

#### 2.1 Hardware Requirements:

- Processor: Any processor above 500MHz.
- RAM: 4 GB+ RAM or more.
- Space: 500 GB

#### 2.2 Software Requirements:

- Operation system: Windows 7 / Windows 8 / Windows 10
- User interface: Python, scikit-learn, pandas, numpy, seaborn
- Language: Python

## CHAPTER 3

# PYTHON

## Python

Python is a programming language created by Guido van Rossum in 1991.

Python is interpreted, object-oriented, dynamic data type of high-level programming language.

### 3.1 Python built-in data types:

1. Numeric
2. Sequence type
3. Boolean
4. Set
5. Dictionary

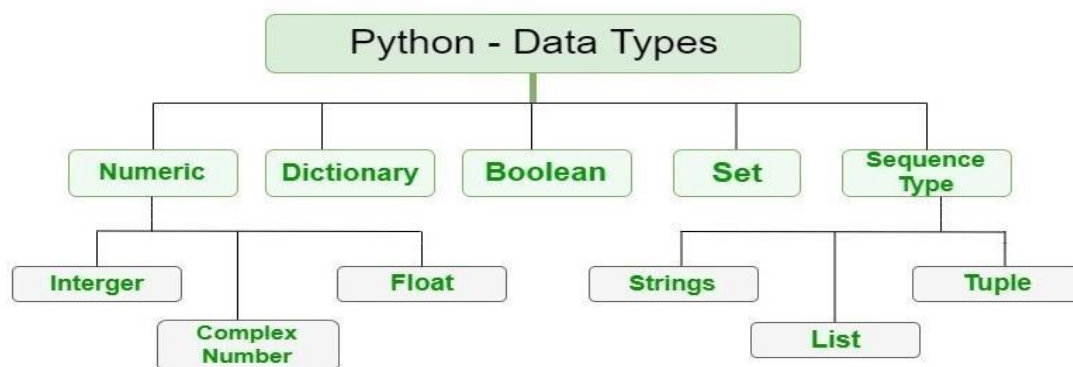


Figure: 3.1

In Python, numeric data type represents the data which has numeric value. Numeric value can be integer, floating number or even complex numbers.

- **Integers:** This value is represented by int class. It contains positive or negative whole numbers. In python there is no limit to how long an integer value can be.
- **Float:** This value is represented by float class. It is a real number with floating point representation. It is specified by a decimal point.
- **Complex Numbers:** It is represented by complex class and is specified as real part and imaginary part.

Example program: `a=5 print("type`

`of a:", type(a)) b=5.0 print("\n type`

`of b:", type(b)) c=2 +4j print("\n`

`type of c:",type(c))` Output: type of

a : <class 'int'>                      type of

b:<class 'float'>                      type of

c:<class 'complex'>

In python, sequence is the ordered collection of similar or different data types. Sequences allow storing multiple values in an organized and efficient fashion.

- 1) **String:** these are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double- quote or triple quote.

In python, there is no character type, a character is a string of length one. It is represented by str class.

2) List: they are just like arrays, declared in other languages which are an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

3) Tuple: It is an ordered collection of python objects. The only difference between type and list is that tuples are immutable i.e., tuples cannot be modified after it is created. It is represented by tuple class.

Boolean: Data type with one of the two built-in values, true or false. Boolean objects that are equal to true are true and those equal to false are false. But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.

Set: It is an unordered collection of data type that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

Dictionary: It is an unordered collection of data values, used to store data values like a map, which unlike other data types that hold only single value as an element, dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a dictionary is separated by a colon : , whereas each key is separated by 'comma'.

Example:

```
Dict={1:'welcome', 2:'to', 3:'python'} print(Dict)
```

Output: {1:'welcome',2:'to', 3:'python'}

### 3.2 Pandas

It is a high-level data manipulation tool developed by Wes McKinney. It is the most popular library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in c or python.

It is built on the numpy package and its key data structure is called “DataFrame”. DataFrame allows us to store and manipulate tabular data in rows of observations and columns of variables.

We can analyze data in pandas with series and DataFrames.

#### □ Import panda library:

Import pandas as pd

	<i>Name</i>	<i>Team</i>	<i>Number</i>	<i>Position</i>	<i>Age</i>
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Figure: 3.2



### 3.3 NumPy

It was created in 2005 by Travis Olipant. It is an open source project and it stands for Numerical Python. It is a python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform and matrices.

It provides a high-performance multidimensional array objects and tools for working with these arrays. It can be utilized to perform a number of mathematical operations on arrays such as trigonometric, statistical and algebraic routines. Pandas objects rely heavily on NumPy objects.

- Installing NumPy via pip command:

Pip install numpy

- **Import NumPy library:** import numpy as np Example:

```
import numpy as np          array_2d=numpy.array([[1,4], [9,
16]], dtype=numpy.float)      print(array_2d)

array_2d_sqrt              =      numpy.sqrt(array_2d)

print(array_2d_sqrt)
```

Output: [[ 1.4.]

[ 9.16.]]

[[1.2.]

[3.4.]]

### 3.4 Seaborn

It is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps us explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally performs the necessary semantic mapping and statistical aggregation to produce informative plots.

Example: `# Import seaborn`

Import seaborn as sns

`sns.set_theme()` #Apply the default theme

`tips = sns.load_dataset("tips")` # Load an example dataset

`sns.replot( data=tips, x="total_bill", y="tip", col="time", hue="smoker",`

`style="smoker", size="size",)` # Creating a visualization

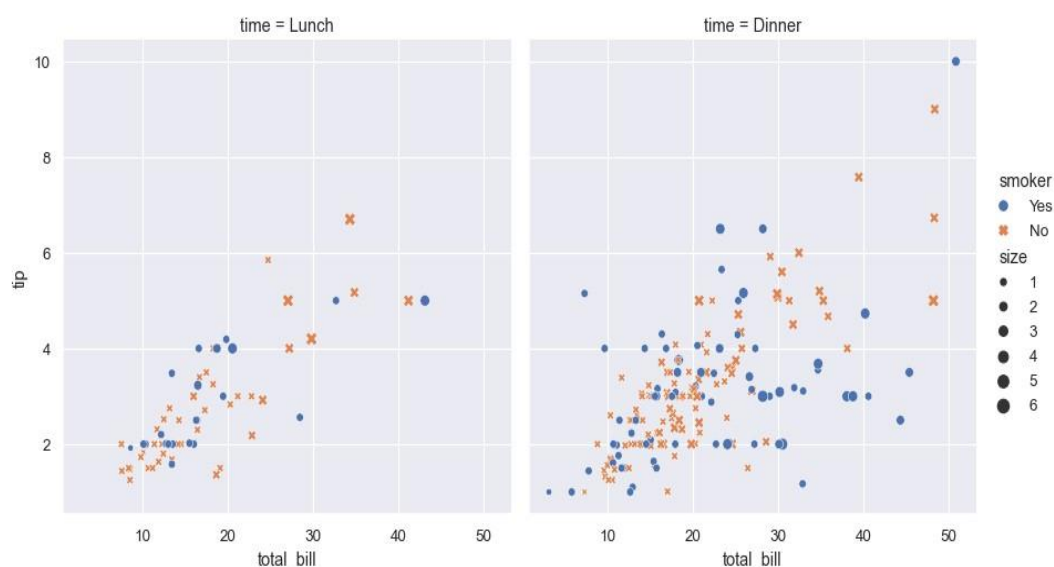


Figure: 3.4

### 3.5 Scikit-learn

It is a free software machine learning library for the python programming language. It provides a selection of tools for machine learning and classification, regression and clustering algorithms including support vector machines. This library is largely written in Python is built upon NumPy, SciPy and Matplotlib.

#### □ Install using pip command or conda:

Pip install -U scikit-learn / conda install scikit-learn

#### Features:

- Supervised Learning Algorithms
- Unsupervised Learning Algorithms
- Clustering
- Datasets
- Cross validation
- Dimensionality Reduction
- Ensemble Methods
- Feature selection
- Feature Extraction
- Open Source

### 3.6 Supervised Learning Algorithms

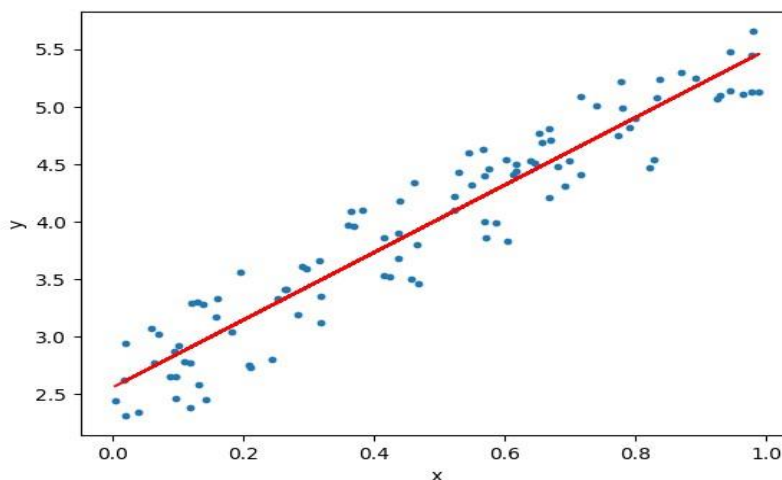
Supervised learning is the machine learning task of learning a function that maps an input to output based on example input-output pairs. It provides us with a powerful tool to classify and process data using machine language. With supervised learning we use labeled data, which is a data set that has been classified to infer a learning algorithm. The

data set is used as the basis for predicting the classification of other unlabeled data through the use of machine learning algorithms.

We have two important techniques in supervised learning:

- Linear Regression
- Classification Techniques

**Linear Regression:** It is a machine learning algorithm based on supervised learning. It performs regression task and it is a prevised learning technique typically used in predicting, forecasting and finding relationships between quantitative data. It is one of the earliest learning techniques, which is still widely used.



**Figure: 3.6**

Linear regression performs the task to predict a dependent variable value(y) based on a independent variable(x). So, this regression technique finds out a linear relationship between x and y. Hence, it is linear regression. Hypothesis function for linear regression:  
 $y = \theta_1 + \theta_2 \cdot x$  While training the model we are given: x: Input training data, y: labels to data  $\theta_1$ : Intercept,  $\theta_2$ : Coefficient of x

**Classification Techniques:** These are focused on predicting a qualitative response by analyzing data and recognizing patterns. For example, this type of technique is used to classify whether or not a credit card transaction is fraudulent. There are many different classification techniques or classifiers, but some of the widely used are include:

- Logistic regression
- Support Vector Machines
- Neural Networks
- Linear discriminant analysis
- Trees
- K- nearest neighbors

### 3.7 Logistic Regression

Logistic Regression is one of the most popular machine learning algorithms, which comes under the supervised learning techniques. It is used for predicting the categorized dependent variable using a given set of independent variables.

It predicts the output of a categorized dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either yes or no, 0 or 1, true or false, etc. But instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic regression is used for solving classification problems whereas linear regression is used for solving regression problems.

Logistic regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Logistic regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image showing the logistic function:

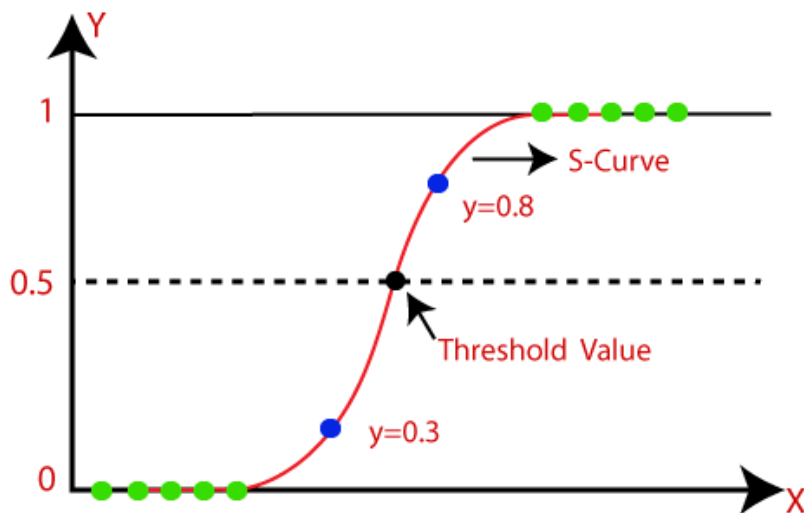


Figure: 3.7

### 3.8 Support Vector Machine

Support Vector Machine or SVM is one of the most popular supervised learning algorithms, which is used for classification as well as regression problems. However, first it is used for classification problems in Machine learning.

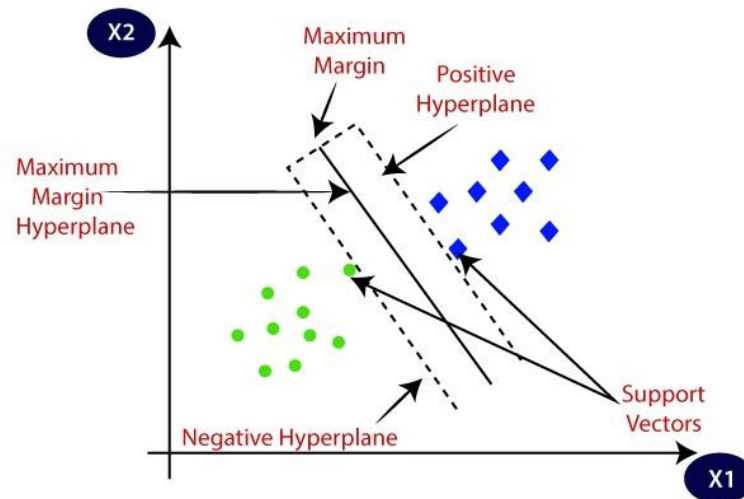
The goal of SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme vectors that help in creating the hyperplane. These extremes cases are called as support vectors and hence algorithm is termed as Support Vector Machine.

SVM algorithm can be used for face detection, image classification, text categorization etc.

In Python, scikit-learn is widely used library for implementing machine learning algorithms. SVM is also available in the scikit-learn library and we follow the same structure using for object-creation, fitting model and prediction.

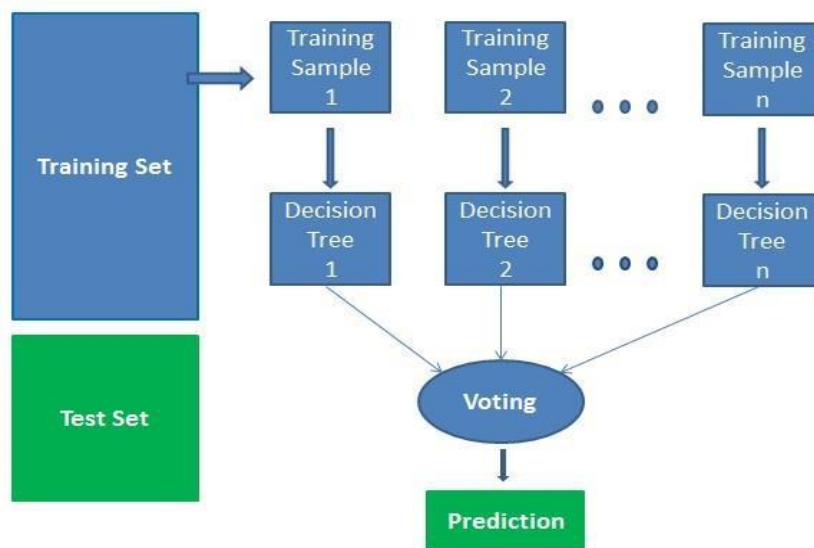
The below diagram in which two different categories that is classified using a decision boundary or hyperplane:



### 3.9 Random Forest Classifier

A random forest is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use the algorithm. Forests are comprised of trees and have robust forest. It creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forest has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal local applicants, identify fraudulent activity and predict diseases. It lies at the base of the brute algorithm Selects important features in dataset.

**Figure: 3.9**



## CHAPTER 4

### DESIGN

#### 4.1. DESIGN GOALS

- Random forest is an ensemble classifier(methods that generate many classifiers and aggregate their results) that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.
- The term came from random decision forests that was first proposed by Tin Kam Ho of Bell Labs in 1995.
- The method combines Breiman's "**bagging**" (randomly draw datasets with replacement from the training data, each sample the same size as the original training set) idea and the random selection of features
- It is very user friendly as it has only two parameters i.e number of variables and number of trees.

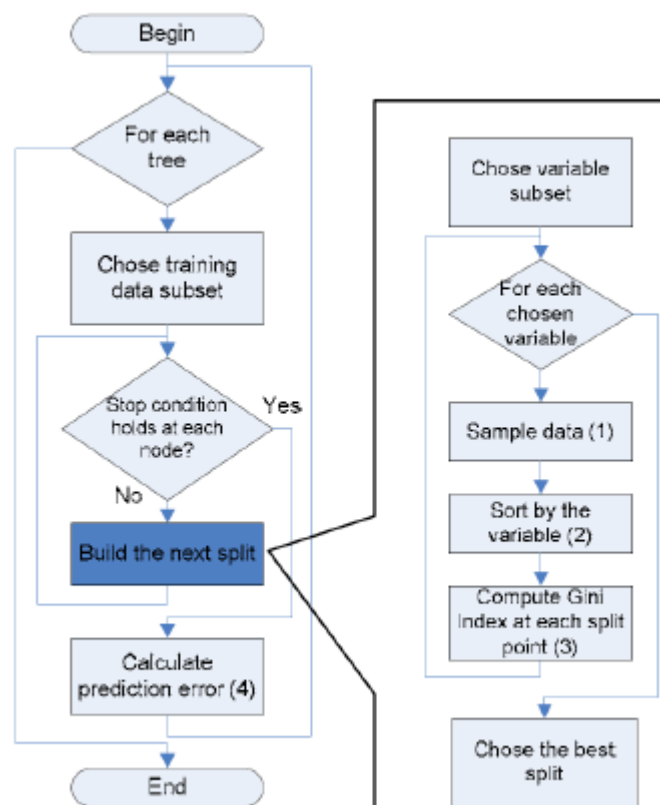
#### ALGORITHM:

Each tree is constructed using the following algorithm:

- Let the number of training cases be  $N$ , and the number of variables in the classifier be  $M$ .
- We are told the number  $m$  of input variables to be used to determine the decision at a node of the tree;  $m$  should be much less than  $M$ .
- Choose a training set for this tree by choosing  $n$  times with replacement from all  $N$  available training cases (i.e. take a bootstrap sample).

- For each node of the tree, randomly choose  $m$  variables on which to base the decision at that node. Calculate the best split based on these  $m$  variables in the training set.
- Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).  
For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction

### FLOW CHART:



## CHAPTER 5

### IMPLEMENTATION

# Importing Important Libraries:

```
from flask import Flask, request, render_template
```

```
from flask_cors import cross_origin
```

```
import pickle
```

```
import sklearn
```

```
import numpy as np
```

```
import pandas as pd
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
app = Flask(__name__)
```

```
model = pickle.load(open('flight_fa.pkl', 'rb'))
```

```
@app.route('/')
```

```
@cross_origin()
```

```
def home():
```

```
    return render_template('home.html')
```

```

@app.route('/predict', methods=['POST', 'GET'])

@cross_origin()

def predict():

    if request.method == 'POST':

        # Date Of Journey:

        date_dep = request.form['Dep_Time']

        Journey_day = int(pd.to_datetime(date_dep, format='%Y-%m-%dT%H:%M').day)

        Journey_month = int(pd.to_datetime(date_dep, format='%Y-%m-%dT%H:%M').month)


        # Departure :

        Dep_hour = int(pd.to_datetime(date_dep, format='%Y-%m-%dT%H:%M').hour)

        Dep_min = int(pd.to_datetime(date_dep, format='%Y-%m-%dT%H:%M').minute)


        # Arrival :

        date_arr = request.form['Arrival_Time']

        Arr_Hour = int(pd.to_datetime(date_arr, format='%Y-%m-%dT%H:%M').hour)

        Arr_min = int(pd.to_datetime(date_arr, format='%Y-%m-%dT%H:%M').minute)


        # Duration :

        dur_hour = abs(Arr_Hour - Dep_hour)

```

```
dur_min = abs(Arr_min - Dep_min)
```

```
# Total_Stops :
```

```
total_Stops = int(request.form['stops'])
```

```
# Airline Manual Label Encoding :
```

```
airline = request.form['Airline']
```

```
if (airline == 'Jet Airways'):
```

```
    Jet_Airways = 1
```

```
    IndiGo = 0
```

```
    Air_India = 0
```

```
    Multiple_carriers = 0
```

```
    SpiceJet = 0
```

```
    Vistara = 0
```

```
    GoAir = 0
```

```
    Multiple_carriers_Premium_economy = 0
```

```
    Jet_Airways_Business = 0
```

```
    Vistara_Premium_economy = 0
```

```
    Trujet = 0
```

```
elif (airline == 'IndiGo'):
```

```
    Jet_Airways = 0
```

```
IndiGo = 1
Air_India = 0
Multiple_carriers = 0
SpiceJet = 0
Vistara = 0
GoAir = 0
Multiple_carriers_Premium_economy = 0
Jet_Airways_Business = 0
Vistara_Premium_economy = 0
Trujet = 0
elif (airline == 'Air India'):
    Jet_Airways = 0
    IndiGo = 0
    Air_India = 1
    Multiple_carriers = 0
    SpiceJet = 0
    Vistara = 0
    GoAir = 0
    Multiple_carriers_Premium_economy = 0
    Jet_Airways_Business = 0
    Vistara_Premium_economy = 0
    Trujet = 0
```

```
elif (airline == 'Multiple carriers'):
```

```
    Jet_Airways = 0
```

```
    IndiGo = 0
```

```
    Air_India = 0
```

```
    Multiple_carriers = 1
```

```
    SpiceJet = 0
```

```
    Vistara = 0
```

```
    GoAir = 0
```

```
    Multiple_carriers_Premium_economy = 0
```

```
    Jet_Airways_Business = 0
```

```
    Vistara_Premium_economy = 0
```

```
    Trujet = 0
```

```
elif (airline == 'SpiceJet'):
```

```
    Jet_Airways = 0
```

```
    IndiGo = 0
```

```
    Air_India = 0
```

```
    Multiple_carriers = 0
```

```
    SpiceJet = 1
```

```
    Vistara = 0
```

```
    GoAir = 0
```

```
    Multiple_carriers_Premium_economy = 0
```

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline == 'Vistara'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 1

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline == 'GoAir'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0



```
SpiceJet = 0

Vistara = 0

GoAir = 1

Multiple_carriers_Premium_economy = 0

Jet_Airways_Business = 0

Vistara_Premium_economy = 0

Trujet = 0


elif (airline == 'Multiple carriers Premium economy'):

    Jet_Airways = 0

    IndiGo = 0

    Air_India = 0

    Multiple_carriers = 0

    SpiceJet = 0

    Vistara = 0

    GoAir = 0

    Multiple_carriers_Premium_economy = 1

    Jet_Airways_Business = 0

    Vistara_Premium_economy = 0

    Trujet = 0


elif (airline == 'Jet Airways Business'):

    Jet_Airways = 0
```

```
IndiGo = 0  
Air_India = 0  
Multiple_carriers = 0  
SpiceJet = 0  
Vistara = 0  
GoAir = 0  
Multiple_carriers_Premium_economy = 0  
Jet_Airways_Business = 1  
Vistara_Premium_economy = 0  
Trujet = 0
```

```
elif (airline == 'Vistara Premium economy'):
```

```
Jet_Airways = 0  
IndiGo = 0  
Air_India = 0  
Multiple_carriers = 0  
SpiceJet = 0  
Vistara = 0  
GoAir = 0  
Multiple_carriers_Premium_economy = 0  
Jet_Airways_Business = 0  
Vistara_Premium_economy = 1  
Trujet = 0
```

```
elif (airline == 'Trujet'):
```

```
    Jet_Airways = 0
```

```
    IndiGo = 0
```

```
    Air_India = 0
```

```
    Multiple_carriers = 0
```

```
    SpiceJet = 0
```

```
    Vistara = 0
```

```
    GoAir = 0
```

```
    Multiple_carriers_Premium_economy = 0
```

```
    Jet_Airways_Business = 0
```

```
    Vistara_Premium_economy = 0
```

```
    Trujet = 1
```

```
else:
```

```
    Jet_Airways = 0
```

```
    IndiGo = 0
```

```
    Air_India = 0
```

```
    Multiple_carriers = 0
```

```
    SpiceJet = 0
```

```
    Vistara = 0
```

```
GoAir = 0

Multiple_carriers_Premium_economy = 0

Jet_Airways_Business = 0

Vistara_Premium_economy = 0

Trujet = 0
```

```
# Source :
```

```
source = request.form['Source']
```

```
if (source == 'Delhi'):
```

```
    s_Delhi = 1

    s_Kolkata = 0

    s_Mumbai = 0

    s_Chennai = 0
```

```
elif (source == 'Kolkata'):
```

```
    s_Delhi = 0

    s_Kolkata = 1

    s_Mumbai = 0

    s_Chennai = 0
```

```
elif (source == 'Mumbai'):
```

```
    s_Delhi = 0

    s_Kolkata = 0
```

```
s_Mumbai = 1  
s_Chennai = 0  
  
elif (source == 'Chennai'):  
    s_Delhi = 0  
    s_Kolkata = 0  
    s_Mumbai = 0  
    s_Chennai = 1  
  
else:  
    s_Delhi = 0  
    s_Kolkata = 0  
    s_Mumbai = 0  
    s_Chennai = 0  
  
# Destination :  
Destination = request.form['Destination']  
if (Destination == 'Cochin'):  
    d_Cochin = 1  
    d_Delhi = 0  
    d_New_Delhi = 0  
    d_Hyderabad = 0  
    d_Kolkata = 0
```

```
elif (Destination == 'Delhi'):
```

```
    d_Cochin = 0
```

```
    d_Delhi = 1
```

```
    d_New_Delhi = 0
```

```
    d_Hyderabad = 0
```

```
    d_Kolkata = 0
```

```
elif (Destination == 'New_Delhi'):
```

```
    d_Cochin = 0
```

```
    d_Delhi = 0
```

```
    d_New_Delhi = 1
```

```
    d_Hyderabad = 0
```

```
    d_Kolkata = 0
```

```
elif (Destination == 'Hyderabad'):
```

```
    d_Cochin = 0
```

```
    d_Delhi = 0
```

```
    d_New_Delhi = 0
```

```
    d_Hyderabad = 1
```

```
    d_Kolkata = 0
```

```
elif (Destination == 'Kolkata'):
```

```
d_Cochin = 0  
d_Delhi = 0  
d_New_Delhi = 0  
d_Hyderabad = 0  
d_Kolkata = 1
```

```
else:
```

```
    d_Cochin = 0  
    d_Delhi = 0  
    d_New_Delhi = 0  
    d_Hyderabad = 0  
    d_Kolkata = 0
```

# Model Prediction Arguments Sequence Should Be According To The Final Processed Dataframe :

```
prediction = model.predict([[  
    total_Stops,  
    Journey_day,  
    Journey_month,  
    Dep_hour,  
    Dep_min,  
    Arr_Hour,  
    Arr_min,  
    dur_hour,
```

```
dur_min,  
Jet_Airways,  
IndiGo,  
Air_India,  
Multiple_carriers,  
SpiceJet,  
Vistara,  
GoAir,  
Multiple_carriers_Premium_economy,  
Jet_Airways_Business,  
Vistara_Premium_economy,  
Trujet,  
s_Chennai,  
s_Delhi,  
s_Kolkata,  
s_Mumbai,  
d_Cochin,  
d_Delhi,  
d_Hyderabad,  
d_Kolkata,  
d_New_Delhi,  
]])
```



```
output = round(prediction[0], 2)
```

```
    return render_template('home.html', prediction_text="Your Flight's Estimated Fare  
Is Rs. {}".format(output))
```

```
    return render_template('home.html')
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

## CHAPTER 6

## RESULT

FLIGHT FARE PREDICTION [🔗](#) [in](#) [@](#) Portfolio

<b>Departure Date</b> dd-mm-yyyy --:-- 📅	<b>Arrival Date</b> dd-mm-yyyy --:-- 📅
<b>Source</b> Delhi ▼	<b>Destination</b> Cochin ▼
<b>Stops</b> Non-Stop ▼	<b>Airline Name</b> Jet Airways ▼

Submit




FLIGHT FARE PREDICTION [🔗](#) [in](#) [@](#) Portfolio


<b>Departure Date</b> 01-12-2020 14:04 📅	<b>Arrival Date</b> 30-11-2020 14:04 📅
<b>Source</b> Delhi ▼	<b>Destination</b> Cochin ▼
<b>Stops</b> Non-Stop ▼	<b>Airline Name</b> Jet Airways ▼


Submit


**Your Flight's Estimated Fare Is Rs. 8243.97**


## FLIGHT FAIR PREDICTION


FLIGHT FARE PREDICTION    Portfolio


Departure Date  
17-12-2020 14:07 

Arrival Date  
16-12-2020 14:07 

Source  
Delhi 




Destination  
Cochin 


Stops  
2 Stop 


Airline Name  
Air Asia 


Submit


Your Flight's Estimated Fare Is Rs. 7174.8


FLIGHT FARE PREDICTION    Portfolio


Departure Date  
26-12-2020 14:09 

Arrival Date  
26-12-2020 14:09 

Source  
Delhi 

Destination  
Cochin 

Stops  
Non-Stop 

Airline Name  
Vistara 

Submit

Your Flight's Estimated Fare Is Rs. 4318.97

## CHAPTER 7:

### CONCLUSION

- The trend of flight prices vary over various months and across the holiday
- There are two groups of airlines: the economical group and the luxurious group. Spicejet, AirAsia, IndiGo, Go Air are in the economical class, whereas Jet Airways and Air India in the other. Vistara has a more spread out trend.
- The airfare varies depending on the time of departure, making timeslot used in analysis an important parameter.
- The airfare increases during a holiday season. In our time period, during Diwali the fare remained high for all the values of days to departure. We haven't considered holiday season as a parameter now, since we are looking at data for a few months.
- Airfare varies according to the day of the week of travel. It is higher for weekends and Monday and slightly lower for the other days.
- There are a few times when an offer is run by an airline because of which the prices drop suddenly. These are difficult to incorporate in our mathematical models, and hence lead to error.
- Along the Mumbai-Delhi route, we find that the price of flights increases or remains constant as the days to departure decreases. This is because of the high frequency of the flights, high demand and also could be due to heavy competition.
- Only about 8-10% of the times, a person should wait according to the data collected across the Mumbai-Delhi route, compared to 30-40% in Delhi-Guwahati route

## 7.1 Future scope

- More routes can be added and the same analysis can be expanded to major airports and travel routes in India.
- The analysis can be done by increasing the data points and increasing the historical data used. That will train the model better giving better accuracies and more savings.
- More rules can be added in the Rule based learning based on our understanding of the industry, also incorporating the offer periods given by the airlines.
- Developing a more user friendly interface for various routes giving more flexibility to the users.

## 7.2 Limitations

- Improper data will result in incorrect fare predictions.

## CHAPTER 8

### REFERENCES

1. O. Etzioni, R. Tuchinda, C. A. Knoblock, and A. Yates. To buy or not to buy: mining airfare data to minimize ticket purchase price.
2. Manolis Papadakis. Predicting Airfare Prices.
3. Groves and Gini, 2011. A Regression Model For Predicting Optimal Purchase Timing For Airline Tickets.
4. Modeling of United States Airline Fares – Using the Official Airline Guide (OAG) and Airline Origin and Destination Survey (DB1B), Krishna Rama-Murthy, 2006.
5. B. S. Everitt: The Cambridge Dictionary of Statistics, Cambridge University Press, Cambridge (3rd edition, 2006). ISBN 0-521-69027-7.
6. Bishop: Pattern Recognition and Machine Learning, Springer, ISBN 0-387-31073-8.
7. E. Bachis and C. A. Piga. Low-cost airlines and online price dispersion. International Journal of Industrial Organization, In Press, Corrected Proof, 2011.
8. P. P. Belobaba. Airline yield management. an overview of seat inventory control. Transportation Science, 21(2):63, 1987.
9. Y. Levin, J. McGill, and M. Nediak. Dynamic pricing in the presence of strategic consumers and oligopolistic competition. Management Science, 55(1):32–46, 2009