A PROJECT REPORT ON

FLIGHT PRICE PREDICTION

SUBMITTED IN PARTIAL FULFILLMENT OF

THE DEGREE M.Sc.IT (PART-II)

SUBMITTED BY

UJWAL KUMAR
KSMSCIT011

2021-2022

UNDER THE GUIDANCE OF

DR. RAKHI GUPTA

&

CO-GUIDANCE OF

Ms. AAFREEN SHAIKH

HSNC UNIVERSITY, MUMBAI

KISHINCHAND CHELLARAM COLLEGE

D.W. ROAD, CHURCHGATE, MUMBAI: 400020

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Acknowledgement

I would like to express my deep sense of gratitude to all those who are associated with my project **"Flight Price Prediction"** a Machine learning-based project, fulfilment of the course of M.Sc. Information Technology affiliated by the HSNC University.

I would like to acknowledge the support with blessing by our Dear Principal **Dr. Hemlata Bagla,** and our Coordinator of M.Sc. IT **Dr. Rakhi Gupta** and **Ms. Nashrah Gowalker** without whom an opportunity to study and pursue a career in M.Sc. IT as a result to work on a project would not have been possible.

I would like to extend my hearty thanks to my internal guide. Her guidance helped me to take right decision with regards to my project. I thank her for sharing her immaculate knowledge and experience which helped me to achieve my goals.

I would even like to thanks our college management and staff members **Mr. Swapnil D. Naidu** Sir for their support co-operation and other facilities. And at last, but not least, I would like to acknowledge the great patience and support given by my friends and family.

TABLE OF CONTENTS

Introduction

Abstract

Optimal timing for airline ticket purchasing from the consumer's perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we majorly targeted to uncover underlying trends of flight prices in India using historical data and also to suggest the approximate price of a flight ticket.

The project implements the validations or contradictions towards myths regarding the airline industry, a comparison study among various models in predicting the optimal time to buy the flight ticket and the amount that can be saved if done so. A customized model which included a combination of ensemble and statistical models have been implemented with a best accuracy of above 90% for a few routes, mostly from metro to metro cities. These models have led to significant savings and produced average positive savings on each transaction.

Remarkably, the trends of the prices are highly sensitive to the route, month of departure, day of departure, time of departure, whether the day of departure is a holiday and airline carrier. Highly competitive routes like most business routes (tier 1 to tier 1 cities like Mumbai-Delhi) had a non-decreasing trend where prices increased as days to departure decreased.

With a high probability (about 20-25%) that a person has to wait to buy a ticket, the scope of the project can be extensively extended across the various routes to make significant savings on the purchase of flight prices across the Indian Domestic Airline market.

Problem Definition

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. Airlines use using sophisticated quasi-academic tactics known as "revenue management" or "yield management". The cheapest available ticket for a given date gets more or less expensive over time. This usually happens as an attempt to maximize revenue based on –

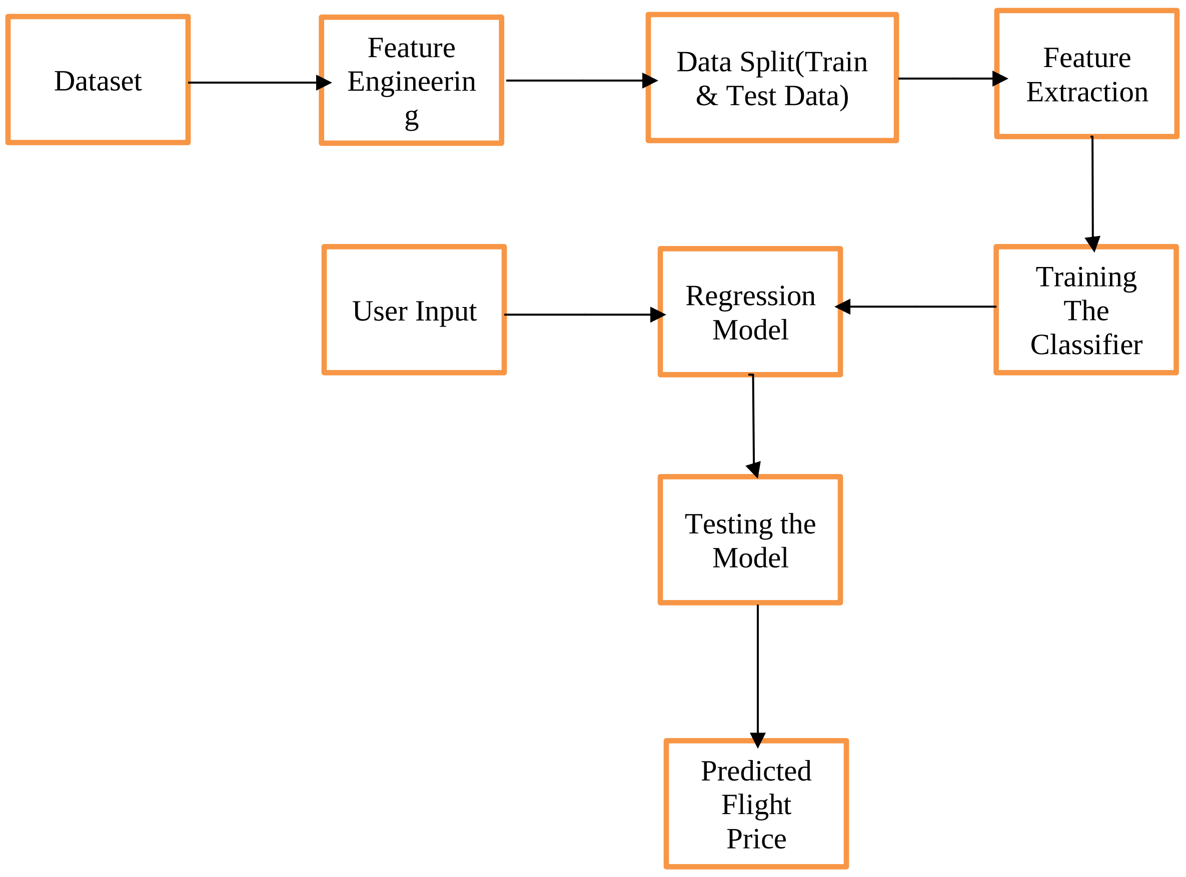
1. Time of purchase patterns (making sure last-minute purchases are expensive)
2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases).

So, if we could inform the travelers with the optimal time to buy their flight tickets based on the historic data and also show them various trends in the airline industry, we could help them save money on their travels. This would be a practical implementation of a data analysis, statistics and machine learning techniques to solve a daily problem faced by travelers.

1. Flight Trends Do airfares change frequently?
2. Do they move in small increments or in large jumps?
3. Do they tend to go up or down over time?
4. Best Time to Buy: What is the best time to buy so that the consumer can save the most by taking the least risk? So should a passenger wait to buy his ticket, or should he buy as early as possible?
5. Verifying myths does price increase as we get near to departure date? Is Indigo cheaper than Jet Airways? Are morning flights expensive?

Proposed Methodology

- End users will be given a link to access the site.
- The link will be an extension to Heroku.
- Users need to fill all the necessary details like from, to, date of journey, type of journey etc.
- Once all the journey details are filled, users can click on predict to see the precited price of that specific journey.
- Below block diagram gives an insight about the project.



Review of Literature

In the preceding work on improving prediction models for airline prices by using Machine Learning (ML) techniques, the different exploration team has concentrated on various attributes and have trained the models on various kinds of Airlines. Specific trend is that they are trying to predict the price. Specifically, categorizing flight price with two divisions of elements helps the studied impact on mean price of the plane. Authors have examined the airline profit by applying pricing modes and have found that after a time duration of 70 days, categorical cases for a flight are observed as flight departure and the discount opportunities also tend to increase over time. Through the analysis we identify equal pricing techniques applied by the airline companies to positively manage the airline offers and demand to increase their business profit. Results shows airlines worry about the price changes according to the season in websites.

The point should be noted down that the importance between the online pricing, and the realized price dispersion on a flight. At the end using prices from actual transactions, the authors have found that online price division is more highly present in lower business airline competition.

Predicting the plane ticket prices and limiting the price for passengers. Price models with reliability can assist passengers to determine the scope of future prices for the airline companies. Present business airline companies will not give passengers to estimate the future reliable costs of any departure requirements. For the usually travelling passengers in this model was developed with price attribute from their own history. First, the high price duration will not be the minimum price accessible for a plane, the passenger data is specifically scheduled that may need to be changed for the price. The attribute from the dataset with subject information also as mentioning high understanding is not required. The Final model can be inspected for subject understanding. In the final work there are extra price limitations that pull out to have outcomes nearer to the accurate solution.

Price attribute allows grouping into clusters, also on the similarity of their behavior. By this representation, the clustering stage follows statistical analysis in group thus formed. The probability distribution for observed trajectories has been estimated for each cluster. From previous data, we learn a decision tree method, by visualizing through different attributes to assign new flight. A variant, considering the first points of the trajectory in the list of predictor variables, has also been considered, to obtain more accurate predictions. In future, the data should be improved more by including with stops, prices found on online flight booking websites, and the increase in prices are collected from 28 days to 90 days.

Improving the ML structure to predict the mean plane price for the business purpose. For predicting the mean plane price with modification of R squared score, feature selection techniques were proposed in our model. Comparing the production of various ML classifiers which tells the greater plane price prediction task. Facts gathered from website that sells the planes ticket through internet apps. Authors have reported that there is limited public information access which will miss the main target attribute. Final accountable prediction model is improved by two unrelated prediction models such as Random Forest and Multilayer Perceptron. Weights for drifting with R-square value and the main estimation of the metric was used.

Survey of Technology

Domain: Artificial Intelligence

Artificial intelligence (AI) makes it possible for machines to learn from experience, adjust to new inputs and perform human-like tasks. Most AI examples that you hear about today – from chess-playing computers to self-driving cars – rely heavily on deep learningand natural language processing. Using these technologies, computers can be trained to accomplish specific tasks by processing large amounts of data and recognizing patterns in thedata.

AI automates repetitive learning and discovery through data. It adds intelligence to existing products. It adapts through progressive learning algorithms to let the data do the programming. It analyses more and deeper data using neural networks that have many hidden layers. It achieves incredible accuracy through deep neural networks – which was previously impossible. AI gets the most out of data.

Every industry has a high demand for AI capabilities – especially question answering systems that can be used for legal assistance, patent searches, risk notification and medical research.

AI applications can provide personalized medicine and X-ray readings. Personal health care assistants can act as life coaches, reminding you to take your pills, exercise or eat healthier. AI provides virtual shopping capabilities that offer personalized recommendations and discuss purchase options with the consumer. Stock management and site layout technologieswill also be improved with AI. AI can analyze factory IoT data as it streams from connected equipment to forecast expected load and demand using recurrent networks, a specific type ofdeep learning network used with sequence data.

Artificial Intelligence enhances the speed, precision and effectiveness of human efforts. In financial institutions, AI techniques can be used to identify which transactions arelikely to be fraudulent, adopt fast and accurate credit scoring, as well as automate manually intense data management tasks.

Domain: Machine Learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

Hardware Requirements

1. Intel Pentium 4 processor or equivalent, 1GHz or faster (Qualcomm Snapdragon and any other ARM processors are not supported)
2. 4 GB RAM
3. 1GB available disk space
4. 64-bit OS

Software Requirements

Software used: Python 3.9.2.

Front End: Python

1. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
2. Its high-level built-in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.
3. The edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception.
4. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Back End: Machine Learning

1. Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.
2. Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics.
3. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

System Design Requirements and System Analysis

Feasibility Study

The very first phase in any system developing life cycle is preliminary investigation. The feasibility study is a major part of this phase. A measure of how beneficial or practical the development of any information system would be to the organization is the feasibility study.

The feasibility of the development software can be studied in terms of the following aspects:

1. **Operational Feasibility**: The site will reduce the time consumed to maintain manual records and is not tiresome and cumbersome to maintain the records. Hence operational feasibility isassured.

2. **Technical Feasibility**:
   - At least 166 MHz Pentium Processor or Intel compatible processor.
   - At least 512 MB RAM.
   - 14.4 kbps or higher modem.
   - A mouse or other pointing device.
   - At least 16 GB free hard disk space.
   - Microsoft Internet Explorer 4.0 or higher.

3. **Economic Feasibility**: Once the hardware and software requirements get fulfilled, there is no need for the user of our system to spend for any additional overhead. For the user, the web site will be economically feasible in the following aspects:
   - The web site will reduce a lot of project work. Hence the cost will be reduced.
   - Our web site will reduce the time that is wasted in manual processes.
   - The storage and handling problems of the registers will be solved.

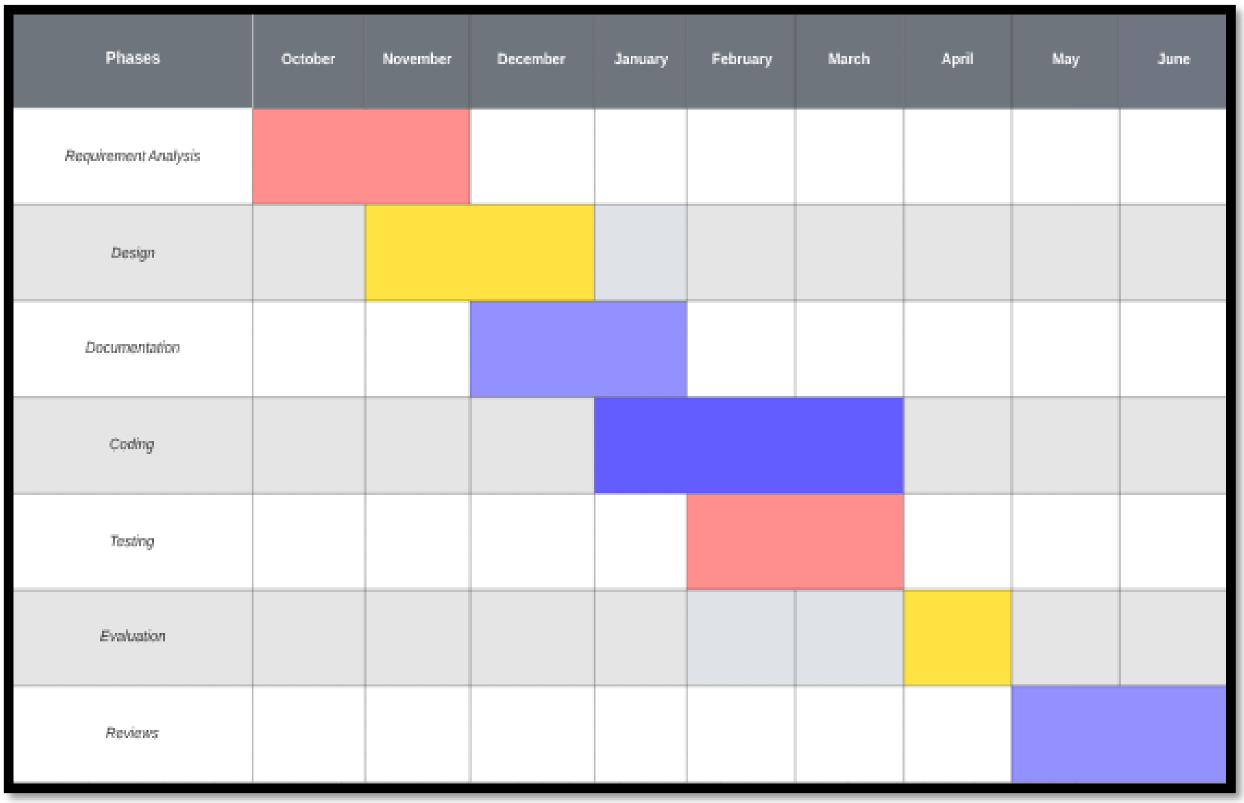
4. **Legal Feasibility**: The licensed copy of the required software is quite cheap and easy to get. Therefore, from legal point of view the proposed system is legally feasible.

Project Scheduling

Sr. No.	Contents	Proposed Date	Submission Date	Teacher's Signature	Remark
1	**Investigation**	14/10/2021			
	Project Fixing	15/10/2021	15/10/2021		
	Synopsis	16/10/2021	16/10/2021		
2	**Analysis**				
	Project History	19/10/2021	18/10/2021		
	Requirement Gathering		18/10/2021		
	Objective and scope of the project		18/10/2021		
	Problems with existing system		19/10/2021		
	Advantages of proposed system		19/10/2021		
	Feasibility study	24/10/2021	20/10/2021		
	Cost benefits analysis		20/10/2021		
	Requirement Specification		23/10/2021		
	Tools and Technology		24/10/2021		
3	**Design Phase**				
	Detailed Life Cycle of the project (Logical design)	03/11/2021	27/10/2021		
	E R Diagram		27/10/2021		
	Use Case Diagram		27/10/2021		
	Activity Diagram		27/10/2021		
	System Flowchart		01/11/2021		
	Sequence Diagram		01/11/2021		
	Data Flow Diagram		03/11/2021		
4	**Coding Phase**				
	Forms		05/01/2022		
	Modules Design		12/01/2022		
	Validating Forms/ Application		10/02/2022		
5	**Testing Phase**				
	Module Testing/Unit Testing		13/02/2022		
	Integration Testing		16/02/2022		
	System Testing		04/03/2022		
	Acceptance Testing		19/03/2022		

6	**Evaluation and Enhancement**				
	System maintenance and future	**15/04/2022**	01/04/2022		
	Enhancement		05/04/2022		
	User Manual		18/04/2022		
7	**Review**		20/04/2022		
8	**Project/Black-Book and Back-up Softcopy Submission**	**27/06/2022**	27/06/2022		

Gantt chart

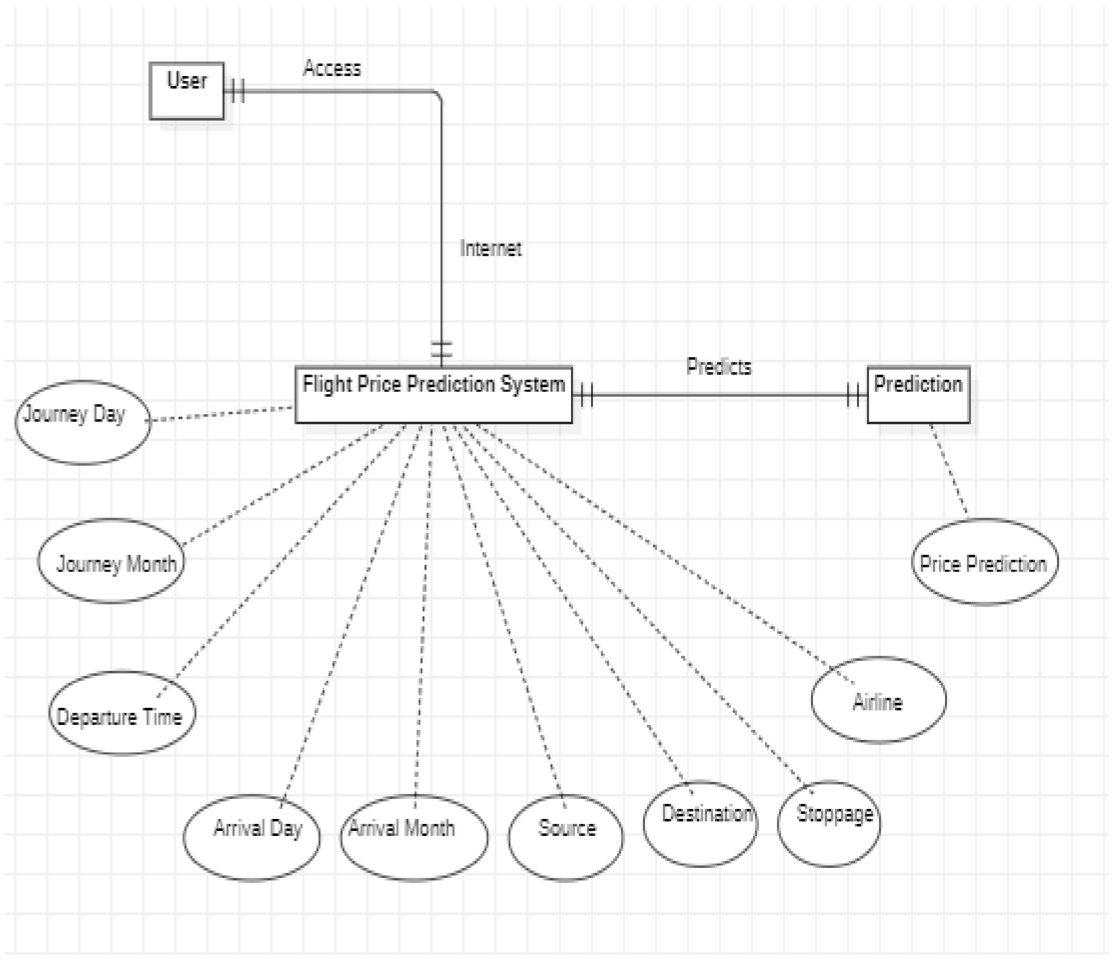


Phases	October	November	December	January	February	March	April	May	June
Requirement Analysis	🟥	🟥							
Design		🟨	🟨						
Documentation			🟪	🟪					
Coding				🟦	🟦	🟦			
Testing					🟥	🟥			
Evaluation							🟨		
Reviews								🟪	🟪

System Design

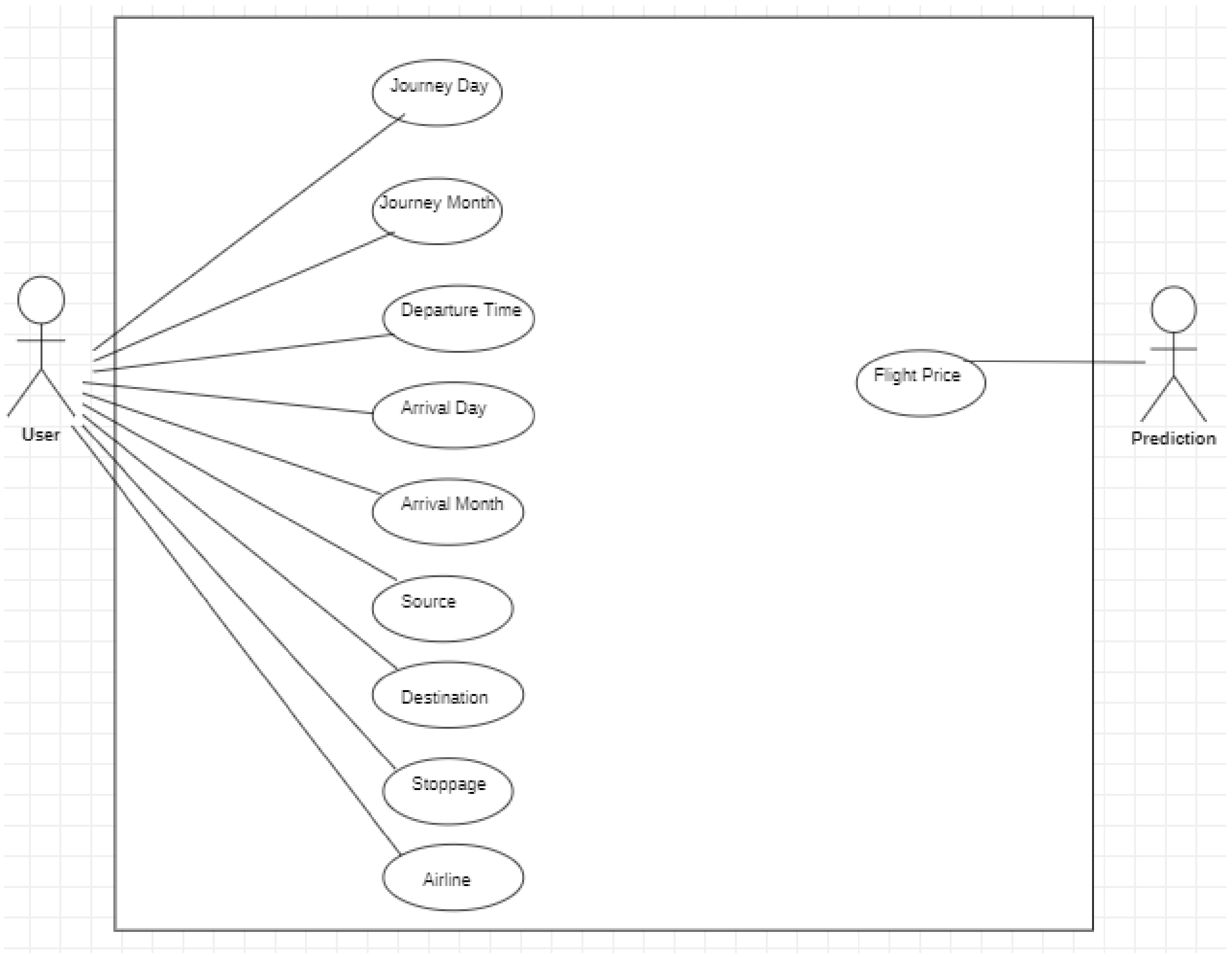
ER Diagram

Entity–Relationship Model (ER model) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERDiagram is a conceptual and representational model of data used to represent the entity framework infrastructure.



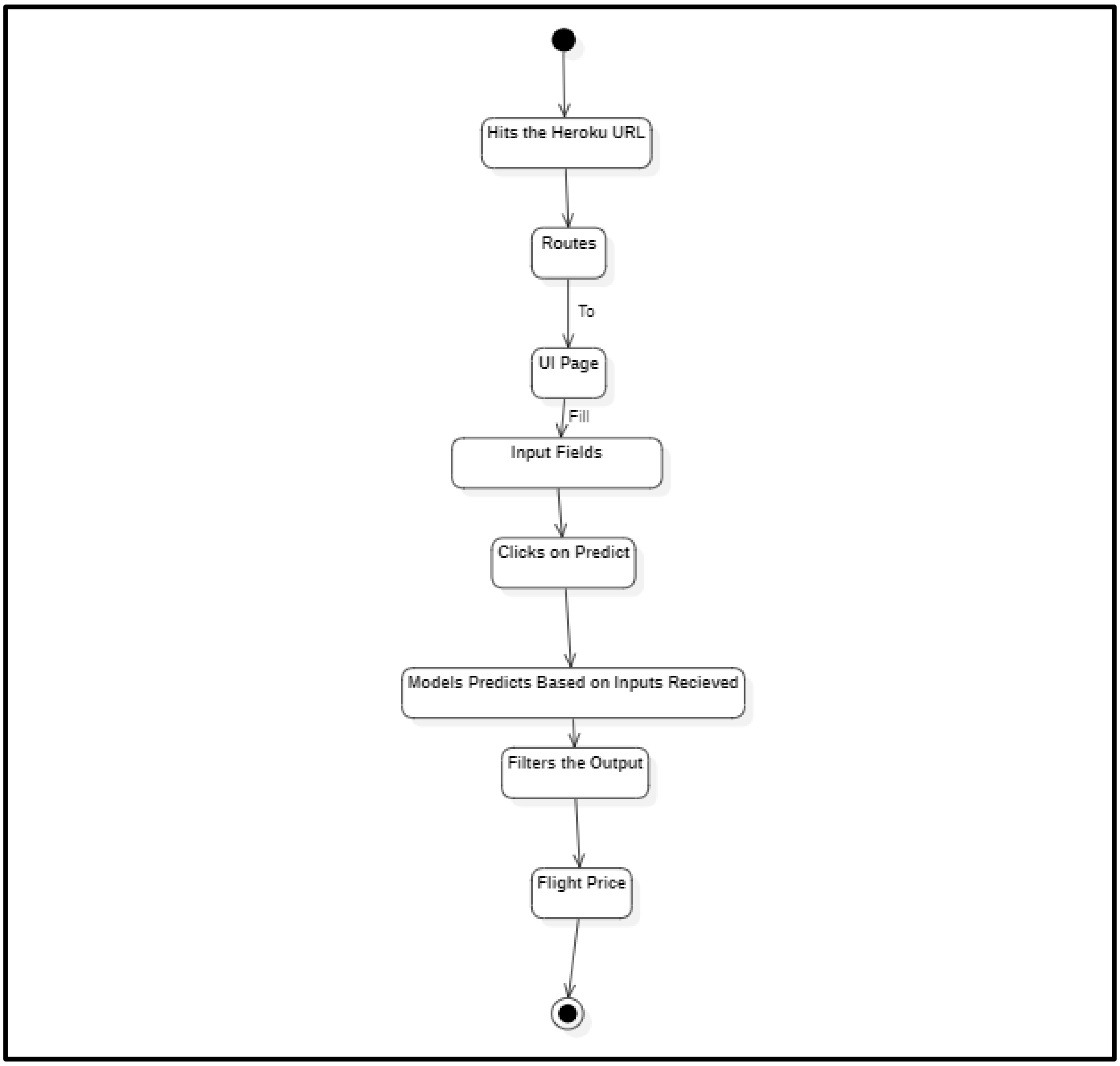
Use Case Diagram

Use case diagrams are usually referred to as behavior diagrams used to describe a setof actions (use cases) that some system or systems (subject) should or can perform incollaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.



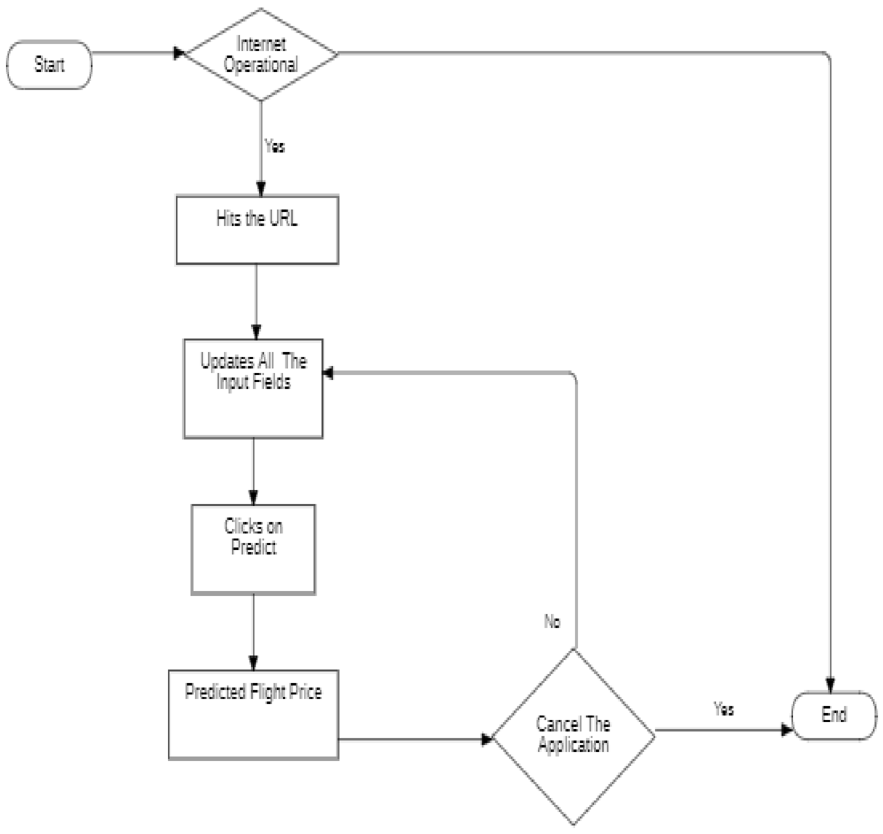
Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.



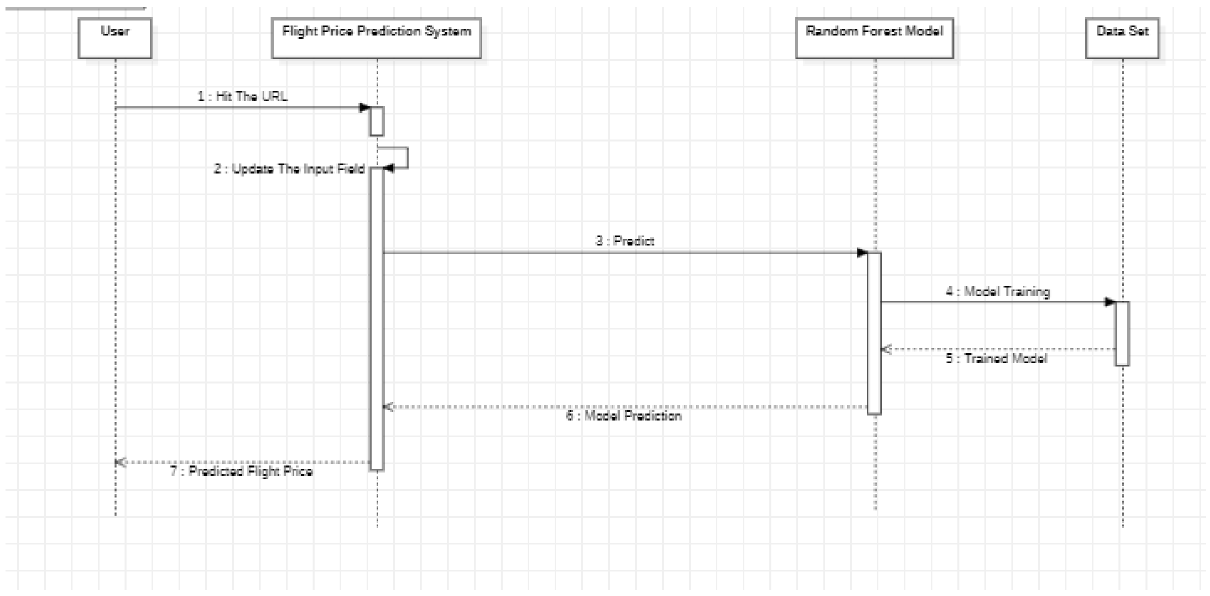
System Flowchart

A flowchart is a graphical representation of steps. It was originated from computer science as a tool for representing algorithms and programming logic but had extended to usein all other kinds of processes. Nowadays, flowcharts play an extremely important role in displaying information and assisting reasoning. They help us visualize complex processes, or make explicit the structure of problems and tasks. A flowchart can also be used to define a process or project to be implemented.



Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of message over time. A sequence diagram is way to visualize and validate various run time scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in process of modelling a new system.



System Implementation

Code Listing:

app.py

```
from flask import Flask, request, render_template
from flask_cors import cross_origin
import sklearn
import pickle
import pandas as pd

app = Flask(__name__)
model = pickle.load(open("flight_rf.pkl", "rb"))



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




@app.route("/predict", methods = ["GET", "POST"])
@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)
        # print("Journey Date : ",Journey_day, Journey_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)
        # print("Departure : ",Dep_hour, Dep_min)

        # Arrival
        date_arr = request.form["Arrival_Time"]
        Arrival_hour = int(pd.to_datetime(date_arr, format ="%Y-%m-%dT%H:%M").hour)
        Arrival_min = int(pd.to_datetime(date_arr, format ="%Y-%m-%dT%H:%M").minute)
        # print("Arrival : ", Arrival_hour, Arrival_min)

        # Duration
        dur_hour = abs(Arrival_hour - Dep_hour)
        dur_min = abs(Arrival_min - Dep_min)
        # print("Duration : ", dur_hour, dur_min)
        # Total Stops
        Total_stops = int(request.form["stops"])
        # print(Total_stops)

        # Airline
        # AIR ASIA = 0 (not in column)
        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

# print(Jet_Airways,
#     IndiGo,
#     Air_India,
#     Multiple_carriers,
#     SpiceJet,
#     Vistara,
#     GoAir,
#     Multiple_carriers_Premium_economy,
#     Jet_Airways_Business,
#     Vistara_Premium_economy,
#     Trujet)

# Source
# Banglore = 0 (not in column)
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
```

```
# print(s_Delhi,
#     s_Kolkata,
#     s_Mumbai,
#     s_Chennai)

# Destination
# Banglore = 0 (not in column)
Source = request.form["Destination"]
if (Source == 'Cochin'):
    d_Cochin = 1
    d_Delhi = 0
    d_New_Delhi = 0
    d_Hyderabad = 0
    d_Kolkata = 0

elif (Source == 'Delhi'):
    d_Cochin = 0
    d_Delhi = 1
    d_New_Delhi = 0
    d_Hyderabad = 0
    d_Kolkata = 0

elif (Source == 'New_Delhi'):
    d_Cochin = 0
    d_Delhi = 0
    d_New_Delhi = 1
    d_Hyderabad = 0
    d_Kolkata = 0

elif (Source == 'Hyderabad'):
    d_Cochin = 0
    d_Delhi = 0
    d_New_Delhi = 0
    d_Hyderabad = 1
    d_Kolkata = 0

elif (Source == '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

# print(
    #     d_Cochin,
    #     d_Delhi,
    #     d_New_Delhi,
    #     d_Hyderabad,
    #     d_Kolkata
    # )


#     ['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_Jet Airways Business',
#    'Airline_Multiple carriers',
#    'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
#    'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
#    'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
#    'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
#    'Destination_Kolkata', 'Destination_New Delhi']

    prediction=model.predict([[
        Total_stops,
        Journey_day,
        Journey_month,
        Dep_hour,
        Dep_min,
        Arrival_hour,
        Arrival_min,
        dur_hour,
        dur_min,
        Air_India,
        GoAir,
        IndiGo,
        Jet_Airways,
        Jet_Airways_Business,
        Multiple_carriers,
        Multiple_carriers_Premium_economy,
        SpiceJet,
        Trujet,
        Vistara,
        Vistara_Premium_economy,
        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 price is Rs. {}".format(output))


    return render_template("home.html")




if __name__ == "__main__":
    app.run(debug=True Flight_Price.ipynb
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
train_data = pd.read_excel(r"E:\MachineLearning\EDA\Flight_Price\Data_Train.xlsx")
pd.set_option('display.max_columns', None)
train_data.head()
train_data.info()
train_data["Duration"].value_counts()
train_data.dropna(inplace = True)
train_data.isnull().sum()
```

EDA
From description we can see that Date_of_Journey is a object data type,\ Therefore, we have to convert this datatype into timestamp so as to use this column properly for prediction

For this we require pandas to_datetime to convert object data type to datetime dtype.

```
train_data["Journey_day"] = pd.to_datetime(train_data.Date_of_Journey,
format="%d/%m/%Y").dt.day
train_data["Journey_month"] = pd.to_datetime(train_data["Date_of_Journey"], format =
"%d/%m/%Y").dt.month

train_data.head()

# Since we have converted Date_of_Journey column into integers, Now we can drop as it is of no use.

train_data.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 from Dep_Time

# Extracting Hours
train_data["Dep_hour"] = pd.to_datetime(train_data["Dep_Time"]).dt.hour

# Extracting Minutes
train_data["Dep_min"] = pd.to_datetime(train_data["Dep_Time"]).dt.minute

# Now we can drop Dep_Time as it is of no use
```

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

train_data.head()

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

# Extracting Hours
train_data["Arrival_hour"] = pd.to_datetime(train_data.Arrival_Time).dt.hour

# Extracting Minutes
train_data["Arrival_min"] = pd.to_datetime(train_data.Arrival_Time).dt.minute

# Now we can drop Arrival_Time as it is of no use
train_data.drop(["Arrival_Time"], axis = 1, inplace = True)
train_data.head()

# Time taken by plane to reach destination is called Duration
# It is the differnce betwwen Departure Time and Arrival time


# Assigning and converting Duration column into list
duration = list(train_data["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration 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

# Adding duration_hours and duration_mins list to train_data dataframe

train_data["Duration_hours"] = duration_hours
train_data["Duration_mins"] = duration_mins

train_data.drop(["Duration"], axis = 1, inplace = True)
train_data.head()
```

Handling Categorical Data
One can find many ways to handle categorical data. Some of them categorical data are,

Nominal data --> data are not in any order --> **OneHotEncoder** is used in this case
Ordinal data --> data are in order --> **LabelEncoder** is used in this case

```
train_data["Airline"].value_counts()
# From graph we can see that Jet Airways Business have the highest Price.
# Apart from the first Airline almost all are having similar median

# Airline vs Price
sns.catplot(y = "Price", x = "Airline", data = train_data.sort_values("Price", ascending = False),
kind="boxen", height = 6, aspect = 3)
plt.show()

# As Airline is Nominal Categorical data we will perform OneHotEncoding

Airline = train_data[["Airline"]]

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

Airline.head()


train_data["Source"].value_counts()
# Source vs Price

sns.catplot(y = "Price", x = "Source", data = train_data.sort_values("Price", ascending = False),
kind="boxen", height = 4, aspect = 3)
plt.show()

# As Source is Nominal Categorical data we will perform OneHotEncoding

Source = train_data[["Source"]]

Source = pd.get_dummies(Source, drop_first= True)

Source.head()

train_data["Destination"].value_counts()

# As Destination is Nominal Categorical data we will perform OneHotEncoding

Destination = train_data[["Destination"]]

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

Destination.head()

train_data["Route"]

# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other

train_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)


train_data["Total_Stops"].value_counts()
```

```
# As this is case of Ordinal Categorical type we perform LabelEncoder
# Here Values are assigned with corresponding keys

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


train_data.head()


# Concatenate dataframe --> train_data + Airline + Source + Destination

data_train = pd.concat([train_data, Airline, Source, Destination], axis = 1)


data_train.head()


data_train.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)

data_train.head()

data_train.shape
```

Test set

```
test_data = pd.read_excel(r"E:\MachineLearning\EDA\Flight_Price\Test_set.xlsx")

test_data.head()


# Preprocessing

print("Test data Info")
print("-"*75)
print(test_data.info())

print()
print()

print("Null values :")
print("-"*75)
test_data.dropna(inplace = True)
print(test_data.isnull().sum())

# EDA

# Date_of_Journey
test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey, format="%d/%m/%Y").dt.day
test_data["Journey_month"] = pd.to_datetime(test_data["Date_of_Journey"], format =
"%d/%m/%Y").dt.month
```

```
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

# Dep_Time
test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour
test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute
test_data.drop(["Dep_Time"], axis = 1, inplace = True)


# Arrival_Time
test_data["Arrival_hour"] = pd.to_datetime(test_data.Arrival_Time).dt.hour
test_data["Arrival_min"] = pd.to_datetime(test_data.Arrival_Time).dt.minute
test_data.drop(["Arrival_Time"], axis = 1, inplace = True)

# Duration
duration = list(test_data["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration 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

# Adding Duration column to test set
test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
test_data.drop(["Duration"], axis = 1, inplace = True)


# Categorical data

print("Airline")
print("-"*75)
print(test_data["Airline"].value_counts())
Airline = pd.get_dummies(test_data["Airline"], drop_first= True)

print()

print("Source")
print("-"*75)
print(test_data["Source"].value_counts())
Source = pd.get_dummies(test_data["Source"], drop_first= True)

print()
print("Destination")
print("-"*75)
print(test_data["Destination"].value_counts())
Destination = pd.get_dummies(test_data["Destination"], drop_first = True)

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

# Replacing Total_Stops
test_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4}, inplace = True)

# Concatenate dataframe --> test_data + Airline + Source + Destination
data_test = pd.concat([test_data, Airline, Source, Destination], axis = 1)

data_test.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)

print()
print()

print("Shape of test data : ", data_test.shape)

data_test.head()
```

Feature Selection
Finding out the best feature which will contribute and have good relation with target variable.
Following are some of the feature selection methods,
heatmap
feature_importance_
SelectKBest

```
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_Jet Airways Business',
       'Airline_Multiple carriers',
       'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
       'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi']]
X.head()

y = data_train.iloc[:, 1]
y.head()

# Finds correlation between Independent and dependent attributes
```

```
plt.figure(figsize = (18,18))
sns.heatmap(train_data.corr(), annot = True, cmap = "RdYlGn")

plt.show()

# 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 visualization

plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```

Fitting model using Random Forest
Split dataset into train and test set in order to prediction w.r.t X_test
If needed do scaling of data
Scaling is not done in Random forest
Import model
Fit the data
Predict w.r.t X_test
In regression check **RSME** Score
Plot graph

```
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
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train, y_train)
y_pred = reg_rf.predict(X_test)
reg_rf.score(X_train, y_train)
reg_rf.score(X_test, y_test)
sns.distplot(y_test-y_pred)
plt.show()
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
# RMSE/(max(DV)-min(DV))

2090.5509/(max(y)-min(y))
metrics.r2_score(y_test, y_pred)
```

Hyperparameter Tuning
Choose following method for hyperparameter tuning
RandomizedSearchCV --> Fast
GridSearchCV
Assign hyperparameters in form of dictionery
Fit the model
Check best paramters and best score

```
from sklearn.model_selection import RandomizedSearchCV
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, 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 node
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 validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions =
random_grid,scoring='neg_mean_squared_error', n_iter = 10, cv = 5, verbose=2, random_state=4)
rf_random.fit(X_train,y_train)
rf_random.best_params_
prediction = rf_random.predict(X_test)
plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()
plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

Save the model to reuse it again

```
import pickle
# open a file, where you ant to store the data
```

```
file = open('flight_rf.pkl', 'wb')

# dump information to that file
pickle.dump(reg_rf, file)
model = open('flight_price_rf.pkl','rb')
forest = pickle.load(model)
y_prediction = forest.predict(X_test)
metrics.r2_score(y_test, y_prediction)
```

home.html

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flight Price Prediction</title>


    <!-- BootStrap -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">


    <!-- css -->
    <link rel="stylesheet" href="static/css/styles.css">


</head>

<body>

    <!-- As a heading -->
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container-fluid">
            <div class="navbar-header">
                <a class="navbar-brand" href="/">FLIGHT PRICE PREDICTION</a>
            </div>
        </div>
    </nav>

    <br><br><br>


    <div class="container">
    <form action="\predict" method="post">


      <div class="row">
        <div class="col-sm-6">
          <div class="card">
            <div class="card-body">
              <h5 class="card-title">Departure Date</h5>
              <!-- Departure -->
              <input type="datetime-local" name="Dep_Time" id="Dep_Time"
required="required">
            </div>
          </div>
        </div>
        <br>
        <br>
        <br>
        <div class="col-sm-6">
          <div class="card">
            <div class="card-body">
              <h5 class="card-title">Arrival Date</h5>
              <!-- Arrival -->
              <input type="date " name="Arrival_Time" id="Arrival_Time" required="required">
            </div>
          </div>
        </div>
      </div>

      <br>
      <br>
      <br>

      <div class="row">
        <div class="col-sm-6">
          <div class="card">
            <div class="card-body">
              <!-- Source -->
              <h5 class="card-title">Source</h5>
              <select name="Source" id="Source" required="required">
                <option value="Delhi">Delhi</option>
                <option value="Kolkata">Kolkata</option>
                <option value="Mumbai">Mumbai</option>
                <option value="Chennai">Chennai</option>
              </select>
            </div>
          </div>
        </div>
        <div class="col-sm-6">
          <div class="card">
            <div class="card-body">
```

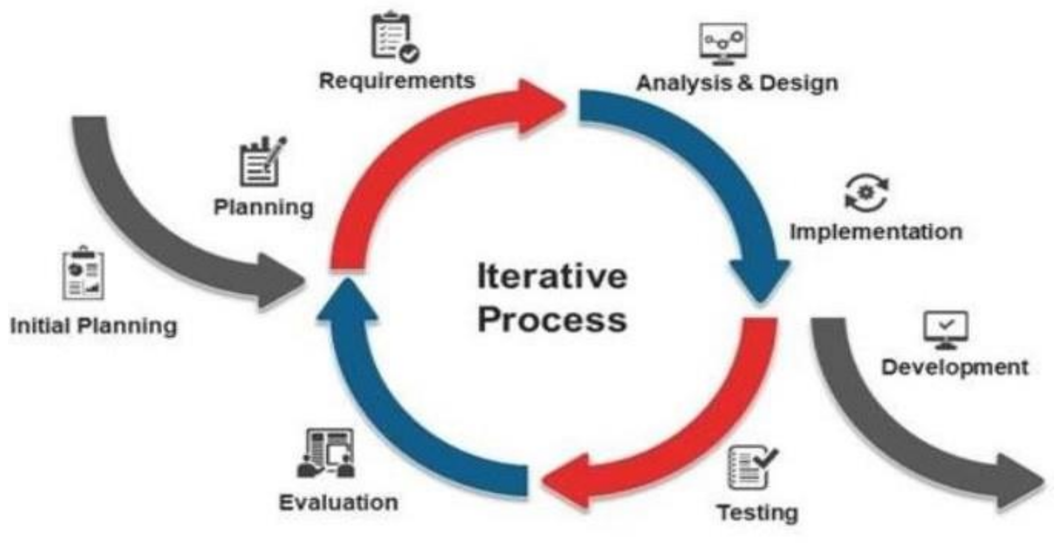
```
                <h5 class="card-title">Destination</h5>
                <!-- Destination -->
                <select name="Destination" id="Destination" required="required">
                    <option value="Cochin">Cochin</option>
                    <option value="Delhi">Delhi</option>
                    <option value="New Delhi">New Delhi</option>
                    <option value="Hyderabad">Hyderabad</option>
                    <option value="Kolkata">Kolkata</option>
                </select>
            </div>
        </div>
    </div>
</div>



<br>
<br>
<br>

<div class="row">
    <div class="col-sm-6">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Stopage</h5>
                <!-- Total Stops -->
                <select name="stops" required="required">
                    <option value="0">Non-Stop</option>
                    <option value="1">1</option>
                    <option value="2">2</option>
                    <option value="3">3</option>
                    <option value="4">4</option>
                </select>
            </div>
        </div>
    </div>
    <div class="col-sm-6">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Which Airline you want to travel?</h5>
                <!-- Airline -->
                <select name="airline" id="airline" required="required">
                    <option value="Air India">Air India</option>
                    <option value="IndiGo">IndiGo</option>
                    <option value="Jet Airways">Jet Airways</option>
                    <option value="Multiple carriers">Multiple carriers</option>
                    <option value="SpiceJet">SpiceJet</option>
                    <option value="Vistara">Vistara</option>
                    <option value="Air Asia">Air Asia</option>
                    <option value="GoAir">GoAir</option>
                    <option value="Multiple carriers Premium economy">Multiple carriers Premium
economy
```

```
                        </option>
                        <option value="Jet Airways Business">Jet Airways Business</option>
                        <option value="Vistara Premium economy">Vistara Premium economy</option>
                        <option value="Trujet">Trujet</option>
                    </select>
                </div>
            </div>
        </div>
    </div>



    <br>
    <br>
    <br>
    <!-- Submit -->
    <input type="submit" value="Submit" class="btn btn-secondary">
  </form>

  <br>
  <br>
  <h3>{{ prediction_text }}</h3>

  <br>
  <br>
  <p>© UJWAL KUMAR</p>
  <p>We hope that our system was able to help you! Happy Journey!!</p>

</div>




<!-- JavaScript -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
    integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
    crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
    integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
    crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
    integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
    crossorigin="anonymous"></script>

</body>

</html>
```

Methodology Adopted

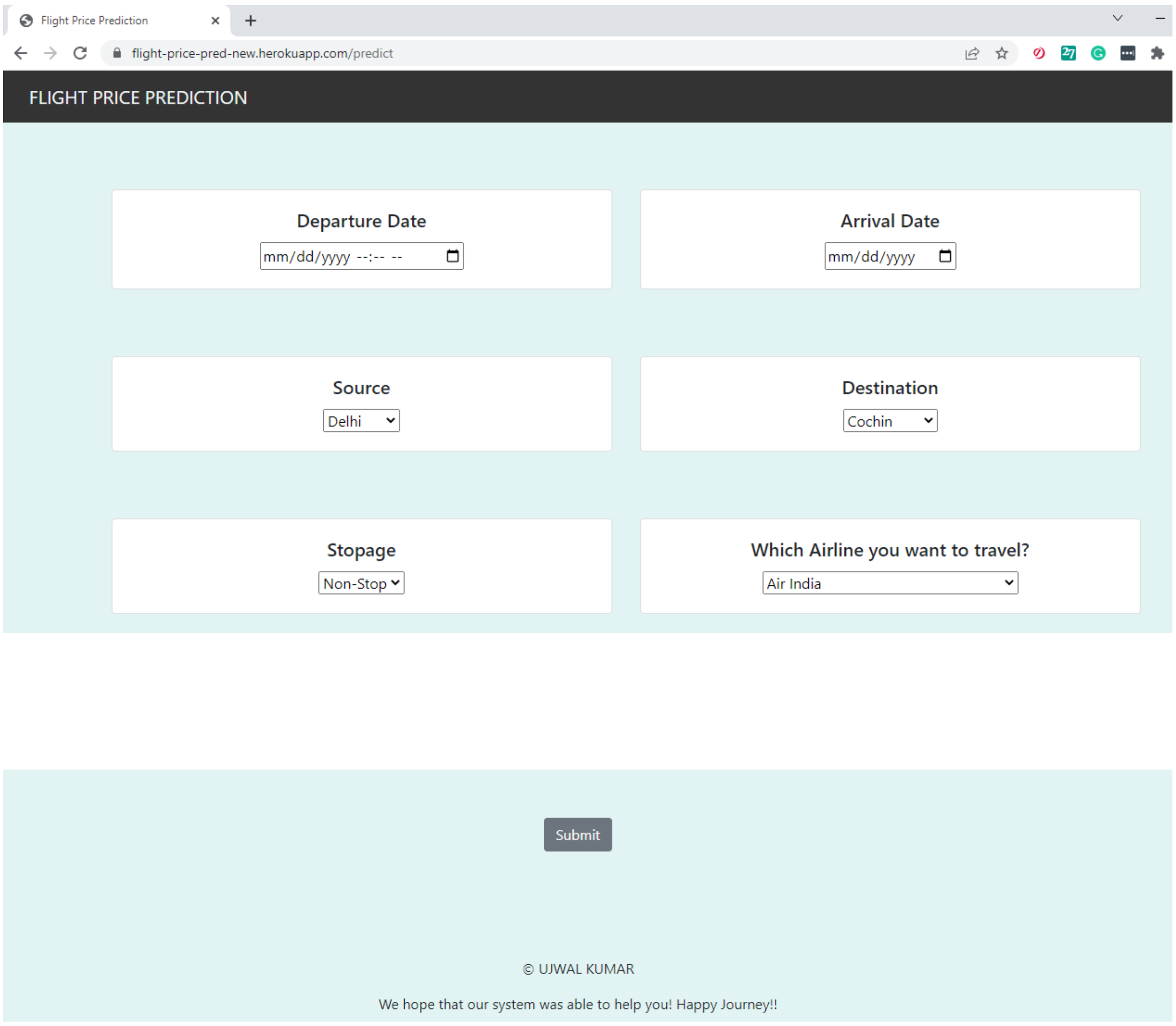
Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

Iterative and Incremental development is a combination of both iterative design oriterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time."This process may be described as an "evolutionary acquisition" or "incremental build" approach." The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model.



User Manual

Screenshots


Flight Price Prediction

flight-price-pred-new.herokuapp.com/predict

FLIGHT PRICE PREDICTION

Departure Date: mm/dd/yyyy --:-- --

Arrival Date: mm/dd/yyyy

Source: Delhi

Destination: Cochin

Stopage: Non-Stop

Which Airline you want to travel?: Air India

Submit

© UJWAL KUMAR

We hope that our system was able to help you! Happy Journey!!

System Testing

Testing Methodology

Testing is a process of executing a program with the intent of finding errors. A good test case is one that has a high probability of finding an as yet undiscovered error. A successful testis one that uncovers an as yet undiscovered error.

If testing is conducted successfully, it will uncover error in the software and testing demonstrates that software functions appear to be working according to specification, that behavior and performance requirements appear to have been met. In addition, data collected as testing provide a good indication of software reliability and some indication of software quality as a whole. However, testing cannot show the absence of errors and defects, it can only showthat errors and defects are present.

Testing Methodology to be adopted:

All testing should be traceable to customer requirements:

- Test should be planned long before testing begins.
- Testing should begin in small scale and progress towards large scale
- Exhaustive testing is not possible.
- To be most effective testing should be conducted by independent third party.

Testing Methods:

Test must be designed with the highest likelihood to find possible errors in the system to avoid major problems before the system goes live. There are two methods to design the test cases.

Verification and validation:

Verification refers to the set of activities that ensure, software correctly implements a specific function. Validation refers to different set of activities that ensures the software that has been built is traceable to customer requirements.

Types of testing

Unit Testing

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output.

Unit Testing for UI Page:

Sr. No.	Unit	Valid Input	Invalid Input	Sample Input
1	Departure Date	Date Time	Special character	06/15/2022 05:37PM
2	Arrival Date	Date	Special character, Number	06/15/2022
3	Source	Character	Special character, Number	Delhi
4	Destination	Character	Special character, Number, Character	Chennai
5	Stoppage	Character, Number	Special character, Number	1
6	Airline	Character	Special character, Character	Air India

Integration Testing

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Modules	Why	Desired output	Observed output
User Access	Check whether the URL is accessible or not.	To be directed to the homepage.	Directed to the Home Page

System Testing

This testing method is performed for all the data in the system. The data are completely validated according to the companies requested and requirement.

In this testing, software is completely assembled as package, interfacing errors have been uncovered and correction testing begins after each one of the two possible conditions exists. The function or performance characteristic's is confirmed specification and are accepted. A deviation from the specification is uncovered and efficiency list is created.

Black Box Testing

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here themodule will be treated as a black box that will take some input generate output. Output for a given set of input combinations are forwarded to another module.

The main purpose of the Black Box is to check whether the software is working as per expected in requirement document & whether it is meeting the user expectations or not.

Equivalence partitioning:

Equivalence partitioning or equivalence class partitioning (ECP) is a software testing technique that divides the input data of a software unit into partitions of equivalent datafrom which test cases can be derived. In principle, test cases are designed to cover each partition at least once.

For Departure Date:

Condition	Valid	Invalid
Type of input	Number, Character	Special character
Number of input	1	i/p<1 || i/p>1

For Source:

Condition	Valid	Invalid
Type of input	Character	Number, Special character
Number of input	1	i/p<1 || i/p>1
Range of input	10	i/p<10, i/p>10

Boundary value analysis:

Boundary value analysis is a black box test design technique and it is used to find the errors at boundaries of input domain rather than finding those errors in the center of input. Equivalence Partitioning and Boundary value analysis are linked to each other and can be used together at all levels of testing.

For Departure Date Field:

Input No.	Input value
1	06/10/2022 11:25 AM
2	29th July
3	10-2022-06
4	Twenty-night June
5	(Null)

For Source Field:

Input No.	Input value
1	Kolkata
2	123
3	@77896136
4	845tcgb4
5	(Null)

Test case for Departure Date Field:

Test Case ID	Input value	Output	Remark
1	06/10/2022 11:25 AM	Successfully Entered	Valid
2	29th July	Enter appropriate name	Invalid
3	10-2022-06	Enter appropriate name	Invalid
4	Twenty-night June	Enter appropriate name	Invalid
5	(Null)	Enter appropriate name	Invalid

Test case for Source Field:

Test Case ID	Kolkata	Output	Remark
1	Kolkata	Successfully Entered	Valid
2	123	Enter appropriate number	Invalid
3	@77896136	Enter appropriate number	Invalid
4	845tcgb4	Enter appropriate number	Invalid
5	(Null)	Enter appropriate number	Invalid

White Box Testing

Also known as, glass box, structural, clear box and open box testing. A software testing technique whereby explicit knowledge of the internal workings of the item being testedare used to select the test data. Unlike black box, testing, white box testing uses specific knowledge of programming code to examine outputs.

- **Path Coverage:**

  In this, the test case is executed in such a way that every path is executed at least once.

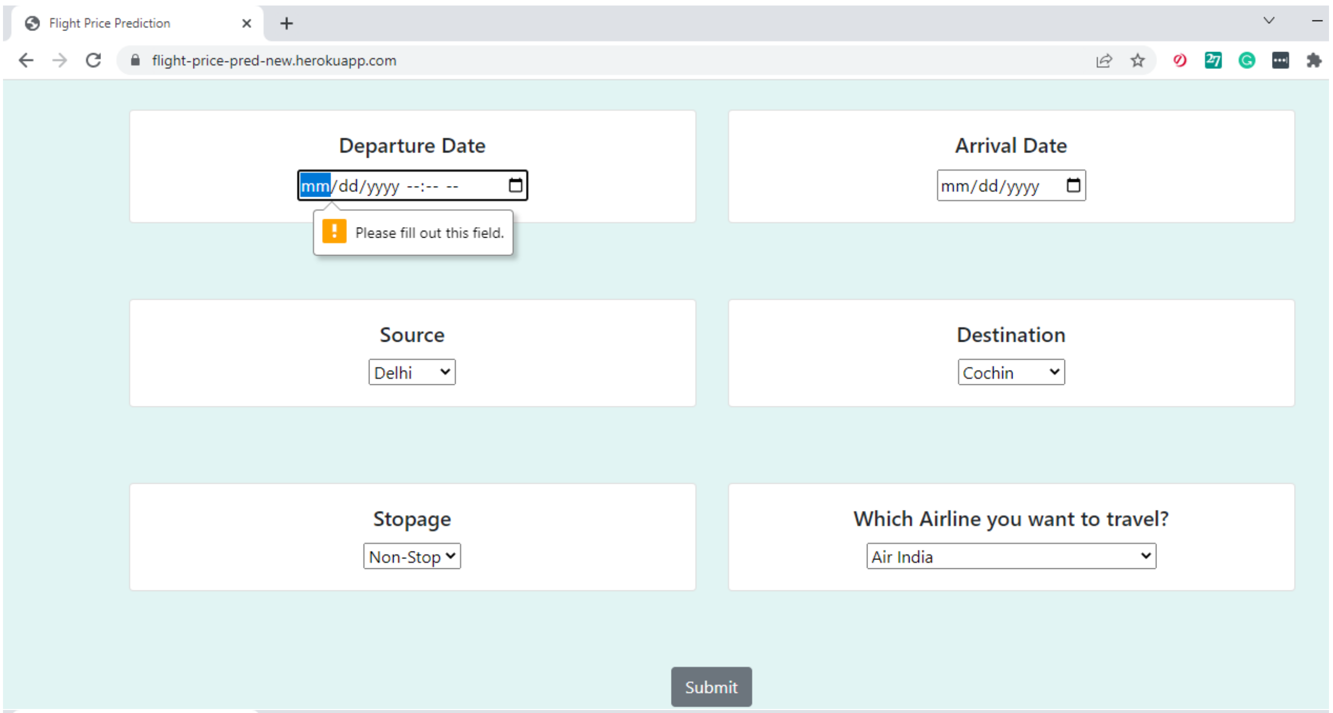
- **Statement Coverage:**

  In this, the test case is executed in such a way that every statement of the code is executed at least once.

- **Data Flow:**

  Data flow testing is a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of variables or data objects. Dataflow Testing focuses on the points at which variables receive values and the points at which these values are used.

Testing Screenshots:



Flight Price Prediction

flight-price-pred-new.herokuapp.com

Departure Date: mm/dd/yyyy --:-- --

Please fill out this field.

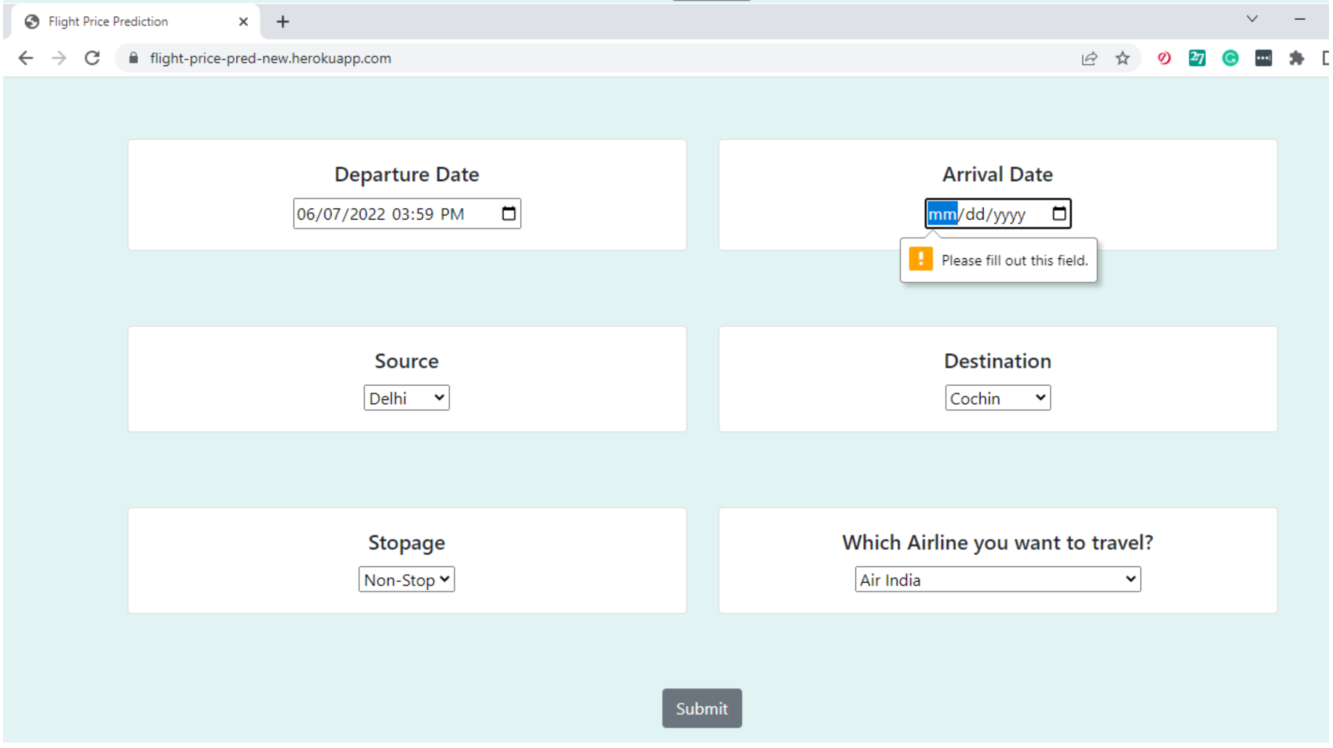
Arrival Date: mm/dd/yyyy

Source: Delhi

Destination: Cochin

Stopage: Non-Stop

Which Airline you want to travel?: Air India



Flight Price Prediction

flight-price-pred-new.herokuapp.com

Departure Date: 06/07/2022 03:59 PM

Arrival Date: mm/dd/yyyy

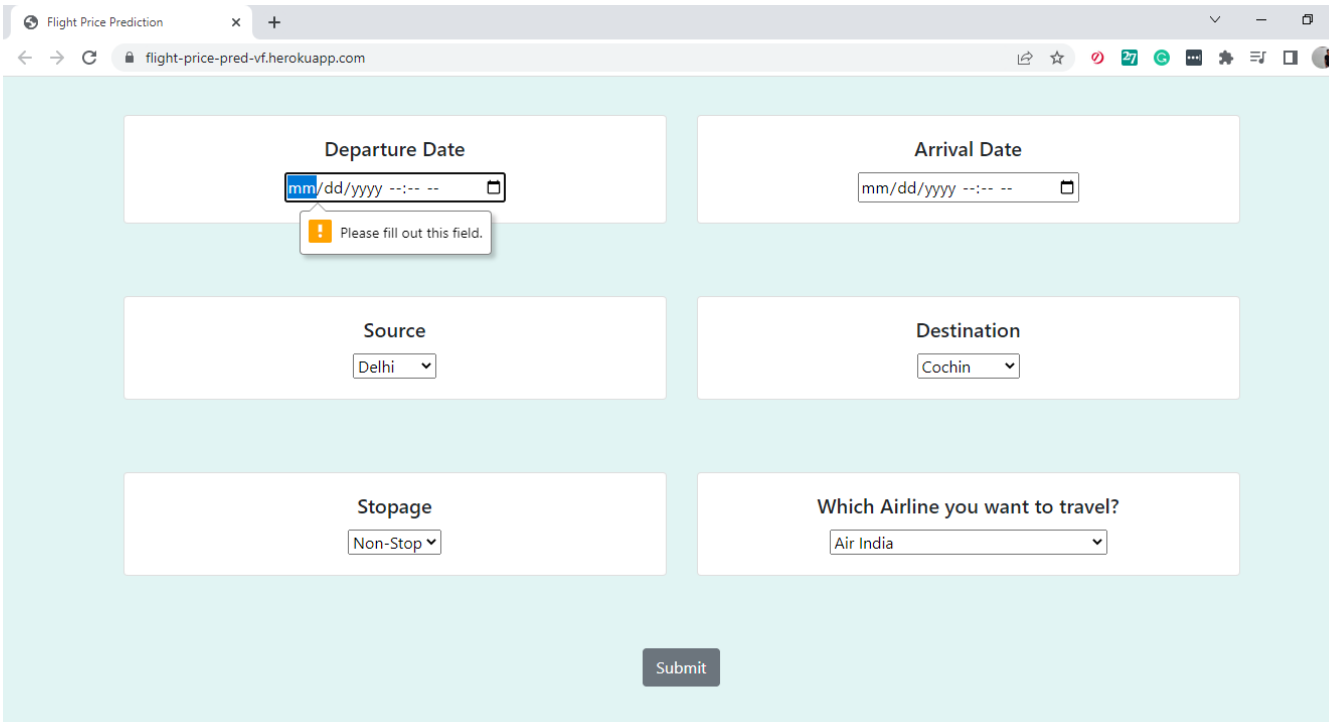
Please fill out this field.

Source: Delhi

Destination: Cochin

Stopage: Non-Stop

Which Airline you want to travel?: Air India



Flight Price Prediction

flight-price-pred-vf.herokuapp.com

Departure Date

mm/dd/yyyy --:-- --

Please fill out this field.

Arrival Date

mm/dd/yyyy --:-- --

Source

Delhi

Destination

Cochin

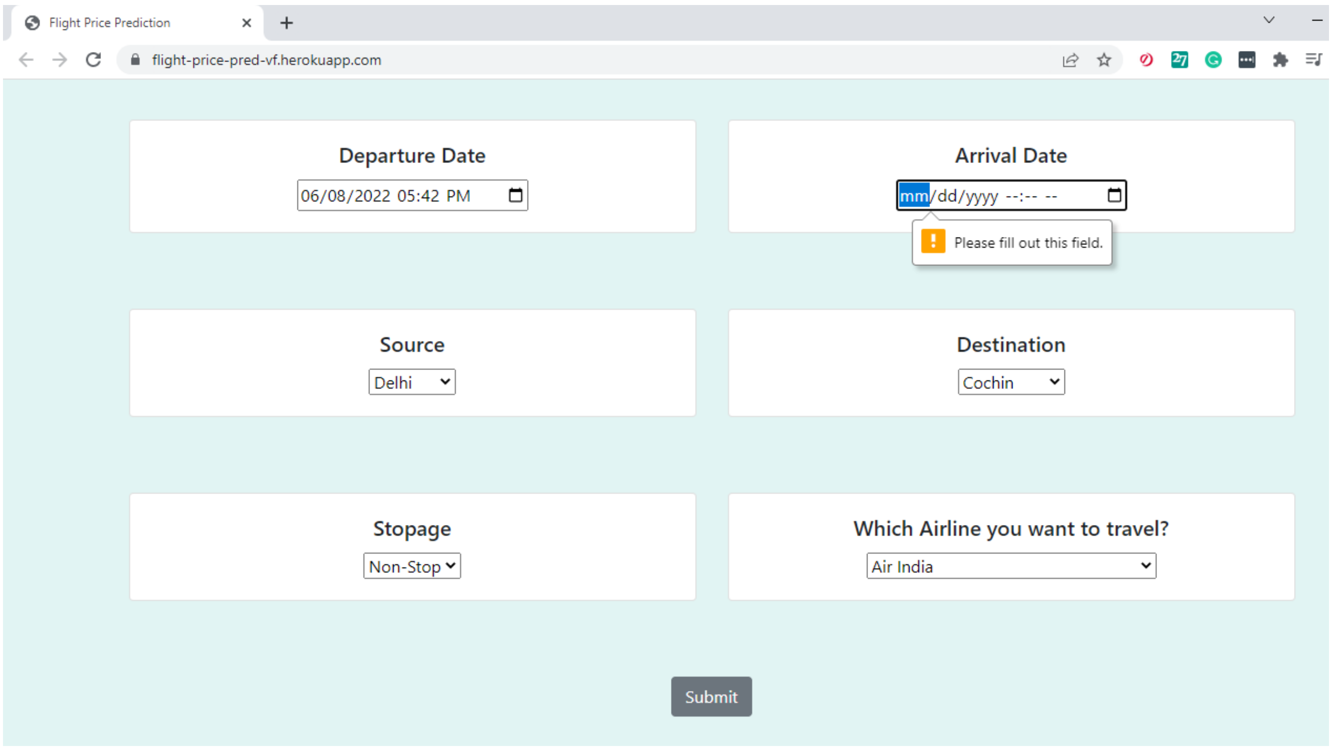
Stopage

Non-Stop

Which Airline you want to travel?

Air India

Submit



Flight Price Prediction

flight-price-pred-vf.herokuapp.com

Departure Date

06/08/2022 05:42 PM

Arrival Date

mm/dd/yyyy --:-- --

Please fill out this field.

Source

Delhi

Destination

Cochin

Stopage

Non-Stop

Which Airline you want to travel?

Air India

Submit

Cost and Benefit Analysis

Developing a cost benefit analysis

Investigations of software development practices, processes & techniques frequently report separately on the costs & benefits of a phenomenon under study, but rarely adequately address the combined bottom-line implications.

In particular, tensions between the quality & productivity effects are hard to reconcile, making objective, high-level insights elusive. The organization will obviously be able to gain benefits such a saving in operating cost, reduction in paperwork, better utilization of human resources & more image increasing goodwill.

Initial cost:

The initial cost of setting up the system will include the cost of hardware software (OS, add-on software, utilities) & labor (setup & maintenance). The same has to bear by theorganization.

Running cost:

Besides, the initial cost the long-term cost will include the running cost for the system including the cost for human resources, cost of update/renewal of various related software.

Need for training:

The users and administrator need to be trained at the time of implementation of the system for smooth running of the system. The client will provide the training site.

We spoke to the management, people who were managing financial issues of the center, the staff who were keeping the records in lots of registers, etc. Then we did the system study of the entire system based on their requirements & the additional features they wanted to incorporate in this system. Reliable, accurate & secure data was also considered too complex task without this proposed system.

Cost evaluation

The primary reason of cost & schedule estimation is to enable the client or developer to perform a cost benefit analysis & for project monitoring & control. The cost of a project is a function of many parameters. Foremost among them is the size of the project. Other factors that affect the cost are programmer ability, experience of the developers in the area, complexity of the project, & reliability requirements. It is also due to the requirements of software, hardware & human resources. Cost required for the project is to install the software & hardware requirements.

Total Implementation Cost:

Total implementation costs are the present value costs calculated over the length of the project. Because there might be overlapping in costs of implementing architectural & policy recommendation when some recommendations may be necessary to mitigate multiple categories of threats, total implementation cost are the sum of all present value costs of implementation minus any overlapping costs.

Net Project Value:

Net project value is the present value of saving form the total benefits of implementation recommendations minus total cost of implementing recommendations minus total cost of implementing recommendations. The higher the net, project value is the better.

Formula:

Net Project Value (NPV) = Total Benefits - Total Implementations Cost

Total system value:

Total system value is the present value of net project value minus the present value of expected loss from unmitigated threats. It takes unmitigated still cost companies some amount ofmoney in risks. If a category of threat is mitigated, then its residual costs are used, otherwise its baseline cost is used. Total system value accounts for scenarios where the net project value is high while theoverall value of the system is low because the solution did not address costly threats.

Formula:

Total System Value (TV) =NPV - Costs of Unmitigated Risk

Costing:

The Total Costing for this project has been:

TOTAL COST = TOTAL MANHOURS * COST PER HOUR

= 400 * 150

= 60,000

Conclusion and Future Enhancements

Conclusion

The proposal was created with the problem of regular commuters in mind. The system is centralised and may be accessed from any location. The technology will provide information about likely flight fares at a given moment, allowing customers to better manage their spending.In a world of increasing technology, automating the manual tasks will definitely help in saving time and getting results that are more reliable.

Limitation

- We have taken data concerned with India Metropolitan cities.
- The systems are susceptible since it is dependent on Heroku, and Heroku limits users to 75 calls to Heroku Git repositories per hour, per app, and per user.
- We can have outdated data in our system.

Future Enhancement

- We may also incorporate tier 2 and tier 3 flight data into our trained model. Scalability is possible.
- Create a live system which is not dependent on any external platforms.
- Can add cloud based database.

References

Bibliography:

We have gathered information required to understand the project concept and the ideas used from various online resources.

- https://www.javatpoint.com/python-data-analytics
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- https://medium.com/analytics-vidhya/regression
- https://www.geeksforgeeks.org/python-convert-string-to-datetime-and-vice-versa/

Websites Used:

https://www.google.co.in/
https://www.github.com
https://www.youtube.com
https://www.guru99.com/
https://en.wikipedia.org/
https://www.medicinenet.com/
https://www.drugs.com/
https://pypi.org/
https://stackoverflow.com