AIRBUS FARE PREDICTION WEB FRAMEWORK WITH THE USE OF MACHINE LEARNING MODEL

A PROJECT REPORT

Submitted by

VEMUNURI GNANESHWAR REDDY(18BCS3818)

SURYA SNEHAVI SANAPALA(18BCS3854)

POLEMONI USHASWINI(18BCS3860)

RAMSHA AHMED(18BCS3845)

Under the Supervision of:

Ms. Jothi Pruthi

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

BIG DATA AND ANALYTICS

CHANDIGARH UNIVERSITY, GHARUAN,
MOHALI - 140413, PUNJAB

MAY & 2022

BONAFIDE CERTIFICATE

Certified that this project report **“ AIRBUS FARE PREDICTION WEB FRAMEWORK WITH THE USE OF MACHINE LEARNING MODEL ”** is the bonafide work of **“ VEMUNURI GNANESHWAR REDDY, SURYA SNEHAVI SANAPALA, POLEMONI USHASWINI, RAMSHA AHMED ”** who carried out the project work under my/our supervision.

<<Signature of the Supervisor>>

SIGNATURE

<<Name>>

SUPERVISOR

<<Academic Designation>>

<<Department>>

<<Signature of the Head of the Department>>

SIGNATURE

<<Name>>

HEAD OF THE DEPARTMENT

<<Department>>

Submitted for the project viva-voce examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

I, "**VEMUNURI GNANESHWAR REDDY, SURYA SNEHAVI SANAPALA, POLEMONI USHASWINI, RAMSHA AHMED "** , student of 'Bachelor of Engineering in computer Science', session: 2018 – 2022 , Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled '**AIRBUS FARE PREDICTION WEB FRAMEWORK WITH THE USE OF MACHINE LEARNING MODEL**' is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

(VEMUNURI GNANESHWAR REDDY)

Candidate UID: 18BCS3818

(SURYA SNEHAVI SANAPALA)

Candidate UID: 18BCS3854

(POLEMONI USHASWINI)

Candidate UID: 18BCS3860

(RAMSHA AHMED)

Candidate UID: 18BCS3845

Date: 19-04-2022

Place: Chandigarh.

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to Ms. Jyoti Pruthi, our project supervisor. We are highly indebted to Apex Institute of Technology, Chandigarh University for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. We would like to express our gratitude towards my parents & amp; members of AIT for their kind cooperation and encouragement which helped me in completion of this project. We would like to express our special gratitude and thanks to industry persons for giving us such attention and time. Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Title **PageNo**

LIST OF TABLES

Table Title **Page Number**

Abstract

People who travel by plane regularly will have a better understanding of the greatest deals and the ideal times to buy tickets. Many airline businesses adjust their pricing based on the seasons or the length of time they are in operation. When individuals travel more, they will raise the price. Estimating the highest airline rates for the trip using features like[1] Duration, Source, Destination, Arrival, Departure.Features are extracted from a dataset, and in this study, we apply machine learning techniques and regression methodologies to forecast the price of airline tickets, which fluctuate over time. We used random forest techniques to provide flight price prediction for consumers**.**

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 best time to buy a flight ticket.

For this project, we have collected data from 18 routes across India while the data of 4 routes were extensively used for the analysis due to the sheer volume of data collected over 4 months resulting in 5.28 lakh data points each across the Mumbai-Delhi and Delhi-Mumbai route and 1.05 lakh data points each across the Delhi-Guwahati and Guwahati-Delhi route. 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 Tier 2 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, however other routes (tier 1 to tier 2 cities like Delhi - Guwahati) had a specific time frame where the prices are minimum. Moreover, the data also uncovered two basic categories of airline carriers operating in India – the economical group and the luxurious group, and in most cases, the minimum priced flight was a member of the economical group. The data also validated the fact that, there are certain time-periods of the day where the prices are expected to be maximum.

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.

Keywords— Machine learning, Airline rates, Features, Prediction model, Random Forest

I. INTRODUCTION

Passengers have little knowledge about future business price rates, so finding the best moment to buy an airline ticket is tough.Different methods predict future business plans prices and classify the ideal moment to buy a plane ticket. Airlines utilize a variety of pricing systems for their tickets, deciding on a price later because the order displays a better value for the approximation models.Each of the factors that contribute to the complicated system are listed below.Because planes have a finite amount of seats, [2] airlines must manage demand.When demand is predicted to grow, the airline may raise prices in order to slow the rate at which seats are filled.Passengers or consumers should prepare ahead of time to take advantage of the greatest deals offered by various airlines and fly for a lower cost.The price of plane tickets fluctuates over time, removing the factors that contribute to the disparity.Reporting on the correlations and models that are used to price plane tickets.Then, based on that data, creating a model that assists travelers in selecting and purchasing tickets, as well as forecasting future air ticket costs. Duration,Arrivaltime,[3]Price, Source, Destination and much more are the attributes used for flight price prediction.
Since the deregulation of the airline industry, airfare pricing strategy has developed into a complex structure of sophisticated rules and mathematical models that drive the pricing strategies of airfare. Although still largely held in secret, studies have found that these rules are widely known to be affected by a variety of factors.

Traditional variables such as distance, although still playing a significant role, are no longer the sole factor that dictate the pricing strategy. Elements related to economic, marketing and societal trends have played increasing roles in dictating the airfare prices.

Most studies on airfare price prediction have focused on either the national level or a specific market. Research at the market segment level, however, is still very limited. We define the term market segment as the market/airport pair between the flight origin and the destination. Being able to predict the airfare trend at the specific market segment level is crucial for airlines to adjust strategy and resources for a specific route. However, existing studies on market segment price prediction use heuristic-based conventional statistical models, such as linear regression , and are based on the assumption that there exists a linear relationship between the dependent and independent variables, which in many cases, may not be true.

Recent advances in Artificial Intelligence (AI) and Machine Learning (ML) make it possible to infer rules and model variations on airfare price based on a large number of features, often uncovering hidden relationships amongst the features automatically. To the best of our knowledge, all existing work leveraging machine learning approaches for airfare price prediction are based on: 1) proprietary datasets that are not publicly available and 2) transaction records data crawled from online travel booking sites like Kayak.com . The problem of the former lies in the difficulty of gaining access to the data, making reproducing the results and extending the work nearly impossible. The issue with the later is that the transaction records from each online booking site are a small fraction of the total ticket sales from the entire market, making the acquired data likely to be skewed, and thus, not representing the true nature of the entire market.

In this paper, we address the problem of market segment level airfare price prediction by using publicly available datasets and a novel machine learning framework to predict market segment level airfare price. More specifically, our proposed framework extracts information from two specific public datasets, the DB1B and the T-100 datasets that are collected and maintained by the Office of Airline Information within the United States Bureau of Transportation Statistics (BTS). The DB1B dataset has been utilized in various studies that assess the determinants of aircraft characteristics and frequency of

flights , analyses for the structure and dynamics of O-D for the core of the air travel market , and demand-prediction. The T-100 dataset includes air passenger volumes for U.S. domestic and international markets and covers large certified carriers that hold Certificates of Public Convenience and Necessity. The goal of our proposed framework is to draw a comprehensive profile of each market and uses machine learning techniques to predict the average airfare on market segment level.

The remainder of this paper is organized as follows. Section II reviews existing work that utilized either conventional statistical or machine learning algorithms for airfare price prediction. Section III provides a detailed description of the two datasets and the proposed framework. Section IV describes the experimental setup and presents the results of applying our proposed framework, as well as a comparison with several baseline methods. In section V, we conclude the paper with a discussion of our contribution and several potential directions for future work

Machine learning:

Machine Learning tutorial provides basic and advanced concepts of machine learning. Our machine learning tutorial is designed for students and working professionals.

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more.

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machines build the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes machine learning to make things easy for us.

We can train machine learning algorithms by providing them with a huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its use cases. Currently, machine learning is used in self-driving cars, cyber fraud detection, face recognition, and friend suggestions by Facebook, etc. Various top companies such as Netflix and Amazon have built machine learning models that are using a vast amount of data to analyze the user interest and recommend products accordingly. With the help of sample historical data, which is known as training data, machine learning algorithms build a mathematical model that helps in making predictions or decisions without

being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance

Supervised learning:

Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output.

The system creates a model using labeled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.

The goal of supervised learning is to map input data with the output data. Supervised learning is based on supervision, and it is the same as when a student learns things under the supervision of the teacher.

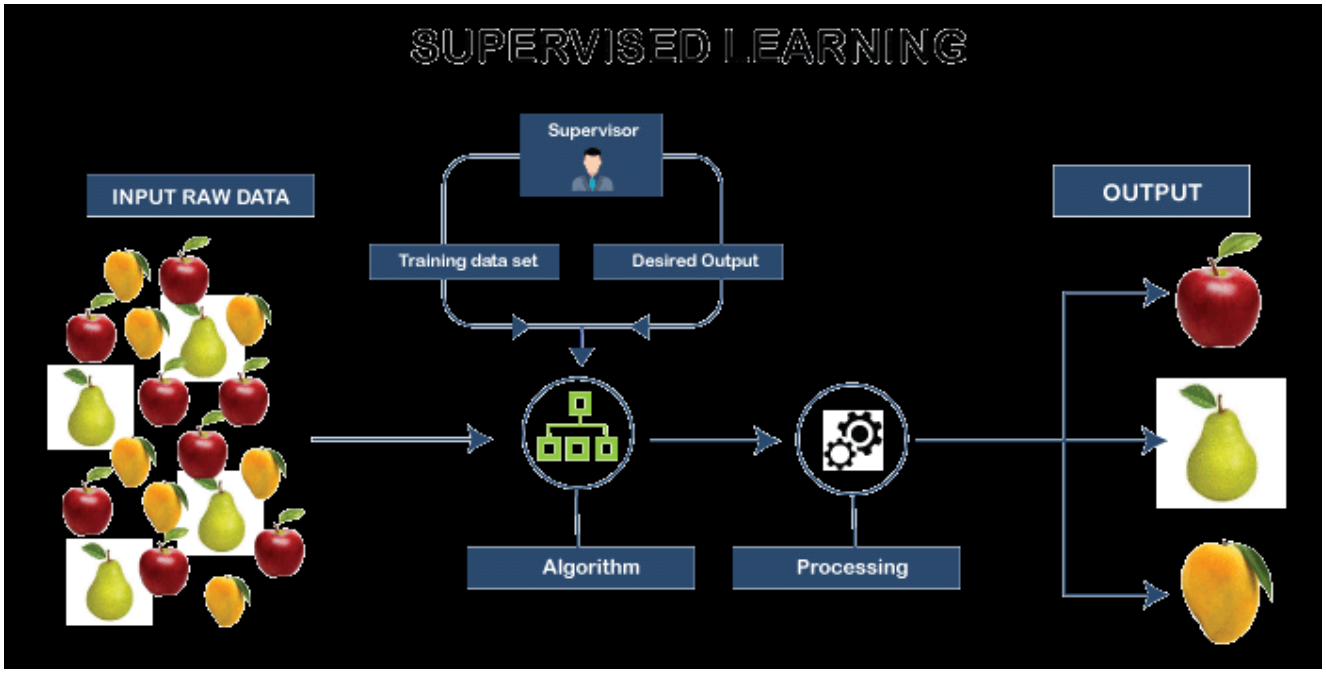

Fig 1:Cycle of Supervised learning

Unsupervised learning:

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

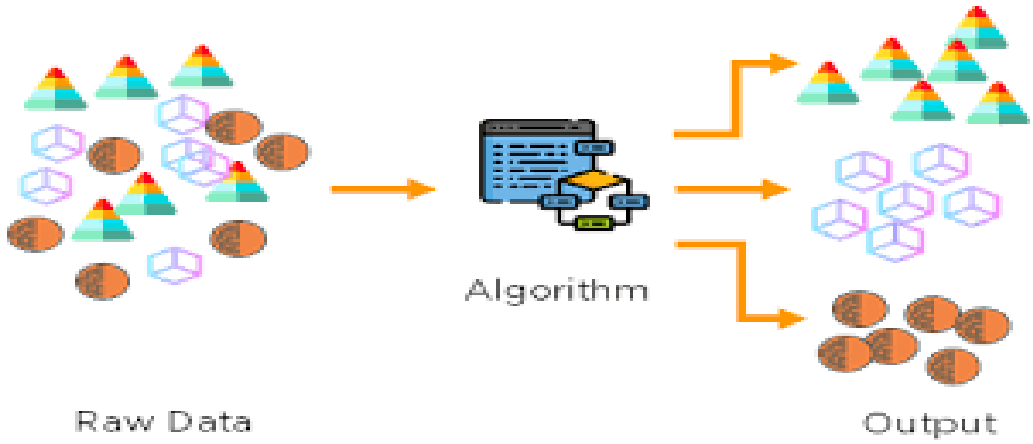

Fig 2: Cycle of Unsupervised Learning

Reinforcement learning:

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

Before some years (about 40-50 years), machine learning was science fiction, but today it is a part of our daily life. Machine learning is making our day to day life easy from self-driving cars to Amazon virtual assistant "Alexa". However, the idea behind machine learning is so old and has a long history. Below some milestones are given which have occurred in the history of machine learning

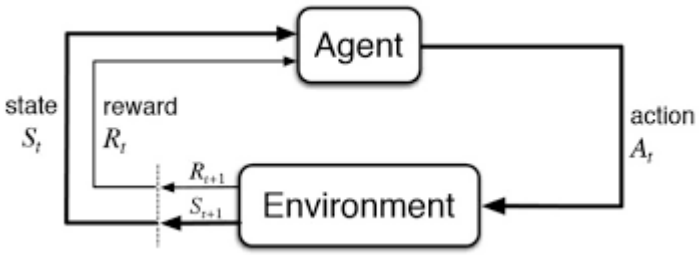

Fig 3: Cycle of Reinforcement learning

Python Libraries:

Fig 4: All libraries of Python

Normally, a library is a collection of books or is a room or place where many books are stored to be used later. Similarly, in the programming world, a library is a collection of precompiled codes that can be used later on in a program for some specific well-defined operations. Other than pre-compiled codes, a library may contain documentation, configuration data, message templates, classes, and values, etc.

A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer. As we don't need to write the same code again and again for different programs. Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, etc.

Working of Python Library

As is stated above, a Python library is simply a collection of codes or modules of codes that we can use in a program for specific operations. We use libraries so that we don't need to write the code again in our program that is already available. But how it works. Actually, in the MS Windows environment, the library files have a DLL extension (Dynamic Load Libraries). When we link a library with our program and run that program, the linker automatically searches for that library. It extracts the functionalities of that library and interprets the program accordingly. That's how we use the methods of a library in our program. We will see further, how we bring in the libraries in our Python programs.

Python standard library

The Python Standard Library contains the exact syntax, semantics, and tokens of Python. It contains built-in modules that provide access to basic system functionality like I/O and some other core modules. Most of the Python Libraries are written in the C programming language. The Python standard library consists of more than 200 core modules.

Numpy is the most popular library in Python. This library is used for processing large multi-dimensional array and matrix formation by using a large collection of high-level mathematical functions and formulas. It is mainly used for the computation of fundamental science in machine learning. It is widely used for linear algebra, Fourier transformation, and random number capabilities. There are other High-end libraries such as TensorFlow, which use NumPy as internal functions for manipulation of tensors.

Fig 5:Numpy logo

Scipy is a popular library among Machine Learning developers as it contains numerous modules for performing optimization, linear algebra, integration, and statistics. SciPy library is different from SciPy stack, as SciPy library is one of the core packages which made up the SciPy stack. SciPy library is Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. used for image manipulation tasks.

Fig 6: Scipy logo

Scikit-learn is a Python library which is used for classical machine learning algorithms. It is built on the top of two basic libraries of Python, that is NumPy and SciPy. Scikit-learn is popular in Machine learning developers as it supports supervised and unsupervised learning algorithms. This library can alsScikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.o be used for data-analysis, and data-mining processes.

Fig 7: Scikit learn logo

Theano is a famous library of Python, which is used for defining, evaluating, and optimizing mathematical expressions, which also efficiently involves multi-dimensional arrays.It is achieved by optimizing the utilization of CPU and GPU. As machine learning is all about mathematics and statistics, Theano makes it easy for the users to perform mathematical operations. It is extensively used for unit-testing and self-verification for detecting and diagnosing different types of errors. Theano is a powerful library that can be used on a large scale computationally intensive scientific project. It is a simple and approachable library, which an individual can use for their projects.

Fig 8: Theano logo

Tensorflow is an open-source library of Python which is used for high performance of numerical computation. It is a popular library, which was developed by the Brain team in Google. TensorFlow is a framework that involves defining and running computations involving tensors. TensorFlow can be used for training and running deep neural networks, which can be used for developing several Artificial Intelligence applications.

Fig 9:Tensorflow logo

Keras is a high-level neural networking API, which is capable of running on top of TensorFlow, CNTK and Theano libraries. It is a very famous library of Python among Machine learning developers. It can run without a glitch on both CPU and GPU. IT makes it really easy and simple for Machine Learning beginners and for designing a Neural Network. It is also used for fast prototyping.

Fig 10: Keras logo

Pytorch is also an open-source Python library for Machine Learning based on Torch, which is implemented in C language and used for Machine learning. It has numerous tools and libraries supported on the computer version, Natural Language Processing (NLP) and many other Machine Learning programs. This library also allows users to perform computational tasks on Tensor with GPU acceleration

Fig 11: Pytorch logo

Pandas is a Python library that is mainly used for data analysis. The users have to prepare the dataset before using it for training the machine learning. Pandas make it easy for the developers as it is developed specifically for data extraction. It has a wide variety of tools for analyzing data in detail, providing high-level data structures.

Fig12: Pandas logo

Matplotlib is a Python library that is used for data visualization. It is used by developers when they want to visualize the data and its patterns. It is a 2-D plotting library that is used to create 2-D graphs and plots.

Fig 13: Matplotlib logo

It has a module pyplot which is used for plotting graphs, and it provides different features for control line styles, font properties, formatting axes and many more. Matplotlib provides different types of graphs and plots such as histograms, error charts, bar charts and many more.

Motivation

What to do when you are at home due to this pandemic situation? I started to learn Machine Learning to get the most out of it. I came to know the mathematics behind all supervised models. Finally it is important to work on applications (real world applications) to actually make a difference.

Objective

As a project, I intended to make it as instructional as possible by tackling each stage of the machine learning process and attempting to comprehend it well. I picked Flight rates as a method, which is known as a "toy issue," identifying problems that are not of immediate scientific relevance but are helpful to demonstrate and practice. The objective was to forecast the price of a specific apartment based on market pricing while accounting for various "features" that would be established in the following sections.

II. Literature review

Proposed study[4] Airfare price prediction using machine learning techniques, For the research work a dataset consisting of 1814 data flights of the Aegean Airlines was collected and used to train a machine learning model. Different numbers of features were used to train models to showcase how selection of features can change the accuracy of the model.

K. Tziridis, Th. Kalampokas, et.al in [5] have developed an airfare price prediction system, the paper they did is divided into 4 phases Feature selection,data collection, ML model selection and Evaluation , they collected the data manually for months between December to July from net.They used eight machine learning models to predict such as , Multilayer Perceptron (MLP),Generalized Regression Neural Network,ExtremeLearning Machine (ELM),Random Forest regression Tree,Regression Tree,Bagging Regression Tree, Regression SVM (Polynomial and Linear),Linear Regression (LR). They compared the accuracy of all these models and concluded as Bagging Regression Tree has the highest accuracy of 87.42%

Tao Liu, Jian Cao, et. al in [6] introduces an ACER(Adaptive Context- Aware Ensemble Regression)framework for airfare price prediction, it predicts the lowest price of flight before the departure date. In this paper they used three steps in model Feature selection and Extraction, single and multiple forecast algorithm, they took the data set from OATS in china. They used Bayesion Regression and on implementation they got RMSE scores between 3.7%-6%.

Supriya Rajankar, Neha Sakharkar, Omprakash Rajankar have developed the [7]Predicting The Price Of A Flight Ticket With The Use Of Machine Learning Algorithms, they divided it into two parts data collection and models in data collection they did collecting the data, cleaning and preparing the data and Analyzing the data . In models they used Decision tree,Random forest,K-NN,Linear Regression, they have taken the data set from [8]Makemytrip.com.

Tianyi Wang, Samira Pouyanfar, et. al in [9] states the problem of market segment level airfare price prediction and propose a novel application for the same using a Machine learning approach, they have taken two datasets DBIB and T-100 which contains minimal features.They have performed data preprocessing, feature extraction and selecting, models. They used models such as LR, SVM, Multilayer Perceptrons (MLPs), XGBoost Tree, and Random forest out of all they used Random Forest as it generated RMSE of 62.753 and it is low as compared to other models.

In case study[10] by William groves an agent is introduced which is able to optimize purchase timing on behalf of customers. Partial least square regression technique is used to build a model.

Research done by Santos[11] analysis is done on air fare routes from Madrid to London, Frankfurt, New York and Paris over course of few months. The model provides the accepted number of days before buying the flight ticket

In[12] the research a desired model is implemented using the Linear Quantile Blended Regression methodology for San Francisco–New York course where each day airfares are given by online website. Two features such as number of days for departure and whether departure is on weekend or weekday are considered to develop the model.

Juhar Ahmed Abdella, Nazar Zaki, et, al. in [13] present a review of deep learning and social media data-based Airline ticket price prediction model. The authors introduce the current airline ticket pricing situation with the factors that affect ticket prices. They also touch upon the strategies which airlines induce to increase their revenue and maximize profits. This model helps its users by advising them whether to buy tickets or wait for a suitable time to get the optimal deal. It uses data mining

techniques like Rule Learning, Reinforcement Learning, time-series methods, and their combinations to achieve greater accuracy in predicting the fare of flights. Features considered for the study include flight number, hours till departure, the current price of a ticket, airline, and its route. The model attained maximum accuracy of 61.9% when a combination of the above-mentioned techniques was used.

Groves and Gini [14] applied the PLS regression model to optimize airline ticket purchasing, with 75.3% accuracy (acc.)

Papadakis [15] predicted if the price of the ticket will drop in the future, by handling the problem as a

classification task using Ripple Down Rule Learner (74.5% acc.), Logistic Regression (69.9% acc.) and Linear SVM (69.4% acc.) ML models.

Ren, Yang and Yuan [16], studied the performance of Linear Regression (77.06% acc.), Naïve Bayes

(73.06% acc.), Softmax Regression (76.84% acc.) and SVM (80.6% acc. for two bins) models in predicting air ticket prices.

Wohlfarth [17] proposed a ticket purchasing time improvement model subject to a significant pre-processing known as macked point processors, data mining frameworks (course of action and grouping) and quantifiable examination system. This framework is proposed to change various added value arrangements into included added value arrangement heading which can support solo gathering estimation. This value heading is packed into get- together reliant on near evaluating conduct. Headway models measure the value change plans. A tree-based analysis used to pick the best planning gathering and a short time later looking at the progression model

An investigation by Dominguez-Menchero [18] suggests the perfect purchase timing reliant on a nonparametric isotonic backslide technique for a specific course, carriers, and time frame. The model provides the most acceptable number of days before buying the flight ticket. The model considers two types of a variable such as the entry and its date of acquisition.

III. METHODOLOGY

We divided the project into three parts:

- Machine learning
- Front-end(HTML,CSS)
- Flask

Machine learning is a field of study that gives computers the capability to learn without being explicitly programmed .It is a subset of artificial intelligence and deep learning is a subset of machine learning . It focuses mainly on designing systems, thereby allowing them to learn and make predictions based on some experience which is data in case of Machines. There are three types of machine learning: supervised learning, unsupervised learning and reinforcement learning . Supervised machine learning with base problem solving two types of problems one is regression and other is classification. In unsupervised machine learning we basically solve two different types of problems, one is clustering and one is dimensionality reduction.

HTML stands for hypertext markup language, HTML is a language of the web. It is used to create website .We use HTML tags to define look and feel of a website html is used for defining layout of a page and it is also called as backbone page structure.CSS is used to add styling to the backbone page 111created by using HTML .JavaScript is used to program logic for the page layout example what happens when a user have on a text when to hide or show elements ect.HTML can be combined with any other like flask,django and angular .

Flask is a very important web application Framework which is written in Python. Flask is mainly used to deploy machine learning models.But flask has some restrictions like it cannot offer dynamic HTML pages but django supports it.

In this project we used Machine learning for model building . We used libraries like pandas,numpy,seaborn,scikit learn and Matplotlib and for frontend we used HTML and css and for backend Flask framework. fig1 shows the methodology block diagram

Fig 14 shows the methodology block diagram

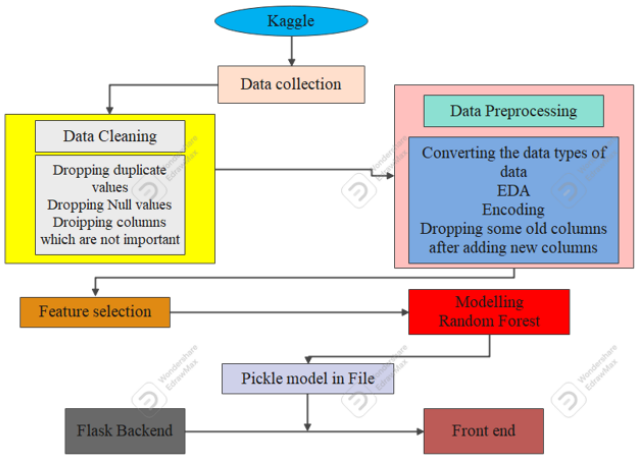

Fig 14:Block diagram of Methodology

A.Data collection

Data collection is the first step in this process. We collected the data from kaggle. It has two types of data sets, one is the Train data set and Other is the Test data set. In train data set we have total of 11 columns and 10683 rows, in test data set we have 10 columns and 2671 rows , both contains the columns like Airline , Date_of_Journey ,Source, Destination,Route, Dep_Time,Arrival_Time , Duration, Total_Stops ,Additional_Info Price, but in test data set we don't have the Price feature but present in Train data set because it is the feature we need to predict.

- **Data**

The data is the most important aspect of a machine learning assignment, to which special attention should be paid. Indeed, the data will heavily affect the findings depending on where we found them, how they are presented, if they are consistent, if there is an outlier, and so on. Many questions must be addressed at this stage to ensure that the learning algorithm is efficient and correct. To obtain, clean, and convert the data, many sub steps are required. I'll go through each one to illustrate how they've been used in my project and why they're helpful for the machine learning section. The first stage is standard data science work, in which we take a data set named 'Flight price'' from Kaggle. We will do significant data cleaning on it to guarantee that it provides reliable predictions throughout prediction. This jupyter notebook, 'Flight_price.ipynb,' is where we do all of our data science work. Because the jupyter notebook is self-explanatory, I'll only touch on the principles that I've implemented briefly. In terms of data cleansing, our dataset needs a significant amount of effort. In fact, 70% of the notebook is dedicated to data cleaning, in which we eliminate empty rows and remove superfluous columns that will not aid in prediction. The process of obtaining valuable and significant information from a dataset that will contribute the most to a successful prediction is the next stage. The final stage is to deal with outliers. Outliers are abnormalities that do massive damage to data and prediction. There is a lot to comprehend conceptually from the dataset in order to discover and eliminate these outliers.

About the dataset

Feature	Description
Airline	So this column will have all the types of airlines like Indigo, Jet Airways, Air India, and many more.
Date_of_Journey	This column will let us know about the date on which the passenger's journey will start.
Source	This column holds the name of the place from where the passenger's journey will start.
Destination	This column holds the name of the place to where passengers wanted to travel.
Route	Here we can know about what is the route through which passengers have opted to travel from his/her source to their destination.

Dep_Time	Departure time is when the flight starts from the source
Arrival_Time	Arrival time is when the passenger will reach his/her destination
Duration	Duration is the whole period that a flight will take to complete its journey from source to destination.
Total_Stops	This will let us know in how many places flights will stop there for the flight in the whole journey.
Additional_Info	In this column, we will get information about food, kind of food, and other amenities.
Price	Price of the flight for a complete journey including all the expenses before onboarding.

Table I **:**Provides the description of each feature

Steps in data collection

1. Importing libraries
2. Loading data

B. Data Cleaning

In data cleaning we check whether there are any null values in the column and any duplicate values in the column , if there are any we drop the rows and we also check whether the column is useful for the process or not, if the column is not useful we drop that column too, like if any column has same data in all rows and has total null values these type of columns should be drop.

Steps in Data cleaning

1. Feature analysis

C. Data pre processing

In data preprocessing we convert the data types of Date_of_Journey, Dep_Time,Arrival_Time and Duration from object to datetime and we create new columns like Journey_day and Journey_month from Date_of_Journey,Dep_Hr and Dep_min from Dep_Time,Arrival_Hr and Arrival_min from Arrival_Time, Dur_Hr and Dur_Min from Duration. After the conversion and Creating new columns we deleted Date_of_Journey, Dep_Time,Arrival_Time and Duration columns as we created alternate columns for them.Later, we performed the Encoding , for nominal data we use One Hot encoding such columns as Source and Destination, and for Ordinal data we use Label encoder such columns as Total_stops.

Steps in Data Pre processing

1. Exploratory data analysis(EDA)
2. Cleaning the extra data after eda
3. Encoding

D. Feature selection

Feature selection also known as Attribute selection, As we work with randomly generated data or raw data may contain lots of noise data, if we keep these features in model we may not be able to produce accuracy in our models , so we select the important and useful features from all features and use them in the model. fig 2 shows the feature importance in visualization.

E.Modeling

We used Random Forest because it is a combination of decision trees and gives more accuracy than others because it generates multiple models.

F.Pickle model in file

We saved the data in a pickle file to reuse when we needed it.

G. Flask backend

To deploy the project we used the flask web framework to create a web page.

F. Front end

We used Html and CSS to create web pages and using Flask we deployed it.

IV. IMPLEMENTATION

1. Importing libraries:

Fig 15 shows the imported libraries

```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import LabelEncoder
```

Fig 15: Libraries

Pandas -Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

Advantages

- Fast and efficient for manipulating and analyzing data.
- Data from different file objects can be loaded.
- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Data set merging and joining.
- Flexible reshaping and pivoting of data sets
- Provides time-series functionality.
- Powerful group by functionality for performing split-apply-combine operations on data sets.

NumPy -NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional

container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases

Matplotlib - Matplotlib is the most important plotting library. There are many plotting tools but matplotlib.pyplot is one of the most useful tools.This whole plotting module itself is inspired by plotting tools that are available in MATLAB.It contains a wide variety of plots, It allows us the digital visuals of data. It contains many plots as bar,lines,box etc.In our project in the section of eda we give an option of correlation plot using matplotlib

Seaborn - Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for the same variables for better understanding of the dataset.

Different categories of plot in Seaborn

Plots are basically used for visualizing the relationship between variables. Those variables can either be completely numerical or a category like a group, class or division. Seaborn divides plot into the below categories –

- **Relational plots:** This plot is used to understand the relation between two variables.
- **Categorical plots:** This plot deals with categorical variables and how they can be visualized.
- **Distribution plots:** This plot is used for examining univariate and bivariate distributions
- **Regression plots:** The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses.
- **Matrix plots:** A matrix plot is an array of scatterplots.
- **Multi-plot grids:** It is a useful approach to draw multiple instances of the same plot on different subsets of the dataset.

scikit-learn-scikit-learn is an open-source Python library that implements a range of machine learning, pre-processing, cross-validation, and visualization algorithms using a unified interface.

Important features of scikit-learn:

- Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
- Accessible to everybody and reusable in various contexts.
- Built on the top of NumPy, SciPy, and matplotlib.
- Open source, commercially usable – BSD license.

2. Load Dataset

https://www.kaggle.com/nikhilmittal/flight-fare-prediction-mh

Fig 16 - shows how to read excel file using pandas

```
#Readind the csv file and showing 1st 5 rows of data
df =  pd.read_excel(r"Data_Train.xlsx")
df.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Fig 16:Data

read_excel - read_excel is an important pandas function to read Excel files and do operations on them.

head() - To obtain the first n rows, use the head() function. This function retrieves the object's first n rows based on location. It is handy for rapidly determining whether your object contains the correct type of data.

3. Feature Analysis

In this we analyze each feature like which type of data it belongs to , any null values , frequency of repeating data and we understand if there are any changes to be made .

Fig 17 - shows the xull values and value count of Airline column

1.1 Airline

- Name of the airline used for traveling

```
#sum of null values in this columns
df.Airline.isna().sum()
```

```
0
```

```
# relative frequency by dividing all unique values by the sum of  values.
df.Airline.value_counts(normalize=True)
```

```
Jet Airways                          0.353627
IndiGo                               0.195259
Air India                            0.161999
Multiple carriers                    0.114308
SpiceJet                             0.077894
Vistara                              0.045685
Air Asia                             0.030488
GoAir                                0.018542
Multiple carriers Premium economy    0.001242
Jet Airways Business                 0.000573
Vistara Premium economy              0.000287
Trujet                               0.000096
Name: Airline, dtype: float64
```

Observations

1. `Airline` is a Nominal Categorical data
2. It contains 12 unique values with heavy imbalance.

Fig 17:Analysis of Feature

isna() - Missing values are detected using the isna() method.

If the values are NA, return a boolean same-sized object. NA values include None and numpy. True values are mapped to NaN values. False values are assigned to everything else. Unless you specify pandas.options.mode.use inf as na = True, characters like empty strings " or numpy.inf are not considered NA values.

sum() - The sum() function in Python computes the sum of all numerical values in an iterable. sum() may be used with both integers and floating-point values. The sum() function in Python may be used to compute the sum of all values in an iterable object.

value_counts() - Return a Series with unique value counts. The resultant object will be sorted in decreasing order, with the first member being the most common. By default, NA values are excluded.

4. Exploratory Data Analysis (EDA)

In this way I make the needed changes in the data for models, because the data may be in different formats so we need to change them to the needed data types.

Fig 18 - shows the data type conversion and forming new columns from Date_of_Journey

```
#Creating a new column called Journey_day on which day the journey starts
df["Journey_day"] = pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.day
```

```
#Creating a new column called Journey_month on which day the journey starts
df["Journey_month"] = pd.to_datetime(df.Date_of_Journey, format = "%d/%m/%Y").dt.month
```

```
df.head(3)
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	3
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662	1	5
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882	9	6

Fig 18:Data Type conversion and creating new columns

Column `Date_of_Journey` is an object data type. So,we have to convert this data type into a timestamp so as to use this column properly for prediction. We will divide it into two columns `Journey_Month` and `Journey_day`

to_datetime - When a csv file or xsl file is imported and a Data Frame is created, the Date Time objects in the file are read as a string object rather than a Date Time object, making operations like Time difference difficult to do on a string rather than a Date Time object. The to_datetime() function in Pandas allows you to transform a text Date Time object into a Python Date Time object.

dt.day - The datetime like values of the series may be accessed using Series.dt, which returns numerous characteristics. Pandas Series.dt.day returns a numpy array representing the datetime's day in the underlying data of the specified series object.

dt.month - The datetime like values of the series may be accessed using Series.dt, which returns numerous characteristics. The month of the datetime in the underlying data of the supplied series object is returned by the Pandas Series.dt.month property, which returns a numpy array.

Fig 19 - shows how Dep_time data type is converted and forming two columns from it

```
# Creating a new column called Dep_Hr represents at what hour the Flight departure
df["Dep_Hr"] = pd.to_datetime(df.Dep_Time).dt.hour
```

```
# Creating a new column called Dep_Min represents at what Minute the Flight departure
df["Dep_Min"] = pd.to_datetime(df.Dep_Time).dt.minute
```

```
df.head(3)
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_Hr	Dep_Min
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	3	22	20
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662	1	5	5	50
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882	9	6	9	25

Fig 19: Data Type conversion and creating new columns

Column `Dep_Time` and `Arrival_Time` is a object data type , we have to convert this data type into timestamp so as to use this column properly for prediction, We will divide it into two columns `Dep_Hr` , `Dep_Min` and `Arrival_Hr` , `Arrival_Min`

dt.hour - The datetimelike values of the series may be accessed using Series.dt, which returns numerous characteristics. The numpy array returned by Pandas Series.dt.hour attribute contains the hour of the datetime in the underlying data of the supplied series object.

dt.minute - The datetimelike values of the series may be accessed using Series.dt, which returns numerous characteristics. The numpy array returned by Pandas Series.dt.minute attribute contains the minutes of the datetime in the underlying data of the supplied series object.

Fig 20 - shows how Duration data type is converted and forming two columns from it

```
Dur_Hr=[]
Dur_Min=[]
duration = list(df.Duration)
for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]
for i in range(len(duration)):
    Dur_Hr.append(int(duration[i].split(sep = "h")[0]))
    Dur_Min.append(int(duration[i].split(sep = "m")[0].split()[-1]))
```

```
df["Dur_Hr"]=Dur_Hr
df["Dur_Min"]=Dur_Min
```

```
df.head(3)
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_Hr	Dep_Min	Arrival_Hr	Arrival_Min	Dur_Hr
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	3	22	20	1	10	2
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662	1	5	5	50	13	15	7
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM →	09:25	04:25 10 Jun	19h	2 stops	No info	13882	9	6	9	25	4	25	19

Fig 20:Data Type conversion and creating new columns

Column `Duration` is an object data type. We have to convert this data type into timestamp so as to use this column properly for prediction. We will divide it into two columns `Dur_Hr` and `Dur_Min`.

5. Data Cleaning

Fig 21 - shows dropping the columns

```
#Dropping the features because we created alternative columns for them
df.drop('Date_of_Journey',inplace=True,axis=1)

df.drop('Dep_Time',inplace=True,axis=1)

df.drop('Arrival_Time',inplace=True,axis=1)

df.drop('Duration',inplace=True,axis=1)
```

```
df.head(3)
```

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_Hr	Dep_Min	Arrival_Hr	Arrival_Min	Dur_Hr	Dur_Min
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	3897	24	3	22	20	1	10	2	50
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7662	1	5	5	50	13	15	7	25
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	13882	9	6	9	25	4	25	19	0

```
#As per 1.10
df.drop(["Additional_Info"], axis = 1, inplace = True)
```

```
#As we knoe Route and Total_Stops are related we can drop Route
df.drop(["Route"], axis = 1, inplace = True)
```

Fig 21:Deleting features

drop() - The drop() method removes the specified row or column. By specifying the column axis (axis='columns'), the drop() method removes the specified column. By specifying the row axis (axis='index'), the drop() method removes the specified row.

6. Encoding

One can find many ways to handle categorical data. Some of them categorical data are,
1.Nominal data→ data are not in any order -->OneHotEncoder is used in this case

2. Ordinal data --> data are in order -->LabelEncoder is used in this case

Encoding - Encoding is used to make the data into a binary system based on the type of data we perform the required encoding like for nominal data we use one hot encoding and for ordinal categorical data we use label encoding.

One hot Encoding - One hot encoding is basically applied to normal categorical variables. If we have a column of normal categorical variables To apply one hot encoding How many number of categories are there that many number of columns are created And for each column it assigns a value in binary so in every column you will able to see onlysee value one and rest of them will be zero. In this process you may encounter the dummy variable trap which means if we have three values by seeing the first two values we can identify the third value , so in this one hot and coding we delete the s one of the columns to make it easier and make the matrix less columns . A dummy variable trap can be achieved by Pandas and sklearn. If we have many categories like pincode in this case we cannot apply the one hot encoding because it forms a larger Matrix and it becomes harder .

Label encoding - Label encoding is basically applied to ordinal categorical Variables . In label encoding we assign a numerical value to each categorical value because when we fit the data set in the model, the model gets to easily find the numeric rather than string to process .

Fig 22 - shows One hot encoding on a column

```
# As Airline is Nominal Categorical data from 1.1  we will perform OneHotEncoding
Airline = pd.get_dummies(df.Airline, drop_first= True)
Airline.head(3)
```

	Air India	GoAir	IndiGo	Jet Airways	Jet Airways Business	Multiple carriers	Multiple carriers Premium economy	SpiceJet	Trujet	Vistara	Vistara Premium economy
0	0	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0

Fig 22:One hot encoding of Airline

- As Airline is Nominal Categorical data we will perform OneHotEncoding

Fig 23 -shows One hot encoding on a column

```
# As Source is Nominal Categorical data from 1.3  we will perform OneHotEncoding
Source = pd.get_dummies(df.Source, drop_first= True)
Source.head(3)
```

	Chennai	Delhi	Kolkata	Mumbai
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0

Fig 23:OneHot Encoding of Source

- As Source is Nominal Categorical data we will perform OneHotEncoding

Fig 24 - shows One hot encoding on a column

```
# As Destination is Nominal Categorical data  from 1.4 we will perform OneHotEncoding

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

	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
0	0	0	0	0	1
1	0	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	1

Fig 24:One Hot Encoding of Destination

- As Destination is Nominal Categorical data we will perform OneHotEncoding

Fig 25 - shows Label encoding on a column

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

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_Hr	Dep_Min	Arrival_Hr	Arrival_Min	Dur_Hr	Dur_Min
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	10	2	50
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	15	7	25
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	25	19	0

Fig 25:Label Encoding of stops

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

Fig 26 - Concatenating the columns

```
df_train=pd.concat([df, Airline, Source, Destination], axis = 1)
```

```
df_train.head(3)
```

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_Hr	Dep_Min	Arrival_Hr	...	Vistara Premium economy	Chennai	Delhi	Kolkata	Mumbai	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	...	0	0	0	0	0	0	0	0	0	1
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	...	0	0	0	1	0	0	0	0	0	0
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	...	0	0	1	0	0	1	0	0	0	0

Fig 26:Combining columns

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

Fig 27 -showing Dropping columns

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

	Total_Stops	Price	Journey_day	Journey_month	Dep_Hr	Dep_Min	Arrival_Hr	Arrival_Min	Dur_Hr	Dur_Min	...	Vistara Premium economy	Chennai	Delhi	Kolkata	Mumbai	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
0	0	3897	24	3	22	20	1	10	2	50	...	0	0	0	0	0	0	0	0	0	1
1	2	7662	1	5	5	50	13	15	7	25	...	0	0	0	1	0	0	0	0	0	0
2	2	13882	9	6	9	25	4	25	19	0	...	0	0	1	0	0	1	0	0	0	0
3	1	6218	12	5	18	5	23	30	5	25	...	0	0	0	1	0	0	0	0	0	0
4	1	13302	1	3	16	50	21	35	4	45	...	0	0	0	0	0	0	0	0	0	1

5 rows × 30 columns

Fig 27:Dropping Columns

- Dropping Airline,Source,Destination from data set as we created alternate columns for them.

- **NOTE:** We repeat all the same steps for our test data set
- **Test set**
    1. It does not contain the dependent Value i.e; `Price`
    2. We will repeat all the steps

Fig 28 - Reading excel file

```
df_Test = pd.read_excel(r"Test_set.xlsx")
```

```
df_Test.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL → BOM → COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU → MAA → BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL → BOM → COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL → BOM → COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	Banglore	Delhi	BLR → DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info

Fig 28: Data

Preprocessing

Fig 29 - shows how to drop null values and finding duplicate values

```
df_Test.dropna(inplace = True)
(df_Test.duplicated().sum() / len(df)) * 100
```

Fig 29: Analysis of features

- dropping duplicate and null values
- Doest have many duplicate values we can ignore them

Exploratory data analysis(EDA):

Fig 30 - Prep processing of test data

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

#Departure Time
df_Test["Dep_Hr"] = pd.to_datetime(df_Test.Dep_Time).dt.hour
df_Test["Dep_Min"] = pd.to_datetime(df_Test.Dep_Time).dt.minute

#Arrival Time
df_Test["Arrival_Hr"] = pd.to_datetime(df_Test.Arrival_Time).dt.hour
df_Test["Arrival_Min"] = pd.to_datetime(df_Test.Arrival_Time).dt.minute

#Duration
Dur_Hr=[]
Dur_Min=[]
duration = list(df_Test.Duration)
for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]
for i in range(len(duration)):
    Dur_Hr.append(int(duration[i].split(sep = "h")[0]))
    Dur_Min.append(int(duration[i].split(sep = "m")[0].split()[-1]))

df_Test["Dur_Hr"]=Dur_Hr
df_Test["Dur_Min"]=Dur_Min
```

Fig 30:Pre processing Test data

- Repeating the same process as we done in the train set

Fig 31 - Cleaning unnecessary data from test data set

```
df_Test.drop('Date_of_Journey',inplace=True,axis=1)
df_Test.drop('Dep_Time',inplace=True,axis=1)
df_Test.drop('Arrival_Time',inplace=True,axis=1)
df_Test.drop('Duration',inplace=True,axis=1)
df_Test.drop('Route',inplace=True,axis=1)
df_Test.drop('Additional_Info',inplace=True,axis=1)
```

Fig 31:Dropping features

- Deleting the unnecessary columns

7. Feature Selection

Feature selection also known as Attribute selection, As we work with randomly generated data or raw data may contain lots of noise data, if we keep these features in model we may not be able to produce accuracy in our models , so we select the important and useful features from all features and use them in the model.There are three types of methods in feature engineering, they are; Filter method, wrapper method and embedded methods.

- Filter method: In filter method we take an attribute and check how much the attribute is relevant with the target attribute, if the attribute is relevant we call it as relevant attribute Filter method are IG,Chi-square , correlation coefficient.
- Wrapper Method: In wrapper method, if we consider all attributes as sets and we generate multiple models using subsets of all attributes with target attribute constant, and finally which model has high efficiency we select those features, but this method is highly computationally expensive wrapper method are Recursive feature elimination,Genetic algorithm.
- Embedded method:Embedded method is almost same as Wrapper method, but it is not as computationally expensive as Wrapper model and has low possibility of over fitting. Example of an Embedded method is Decision Tree.

Fig 32 - code for create heat map

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

Fig 32: Heatmap code

Fig 33 - correlation heatmap of columns

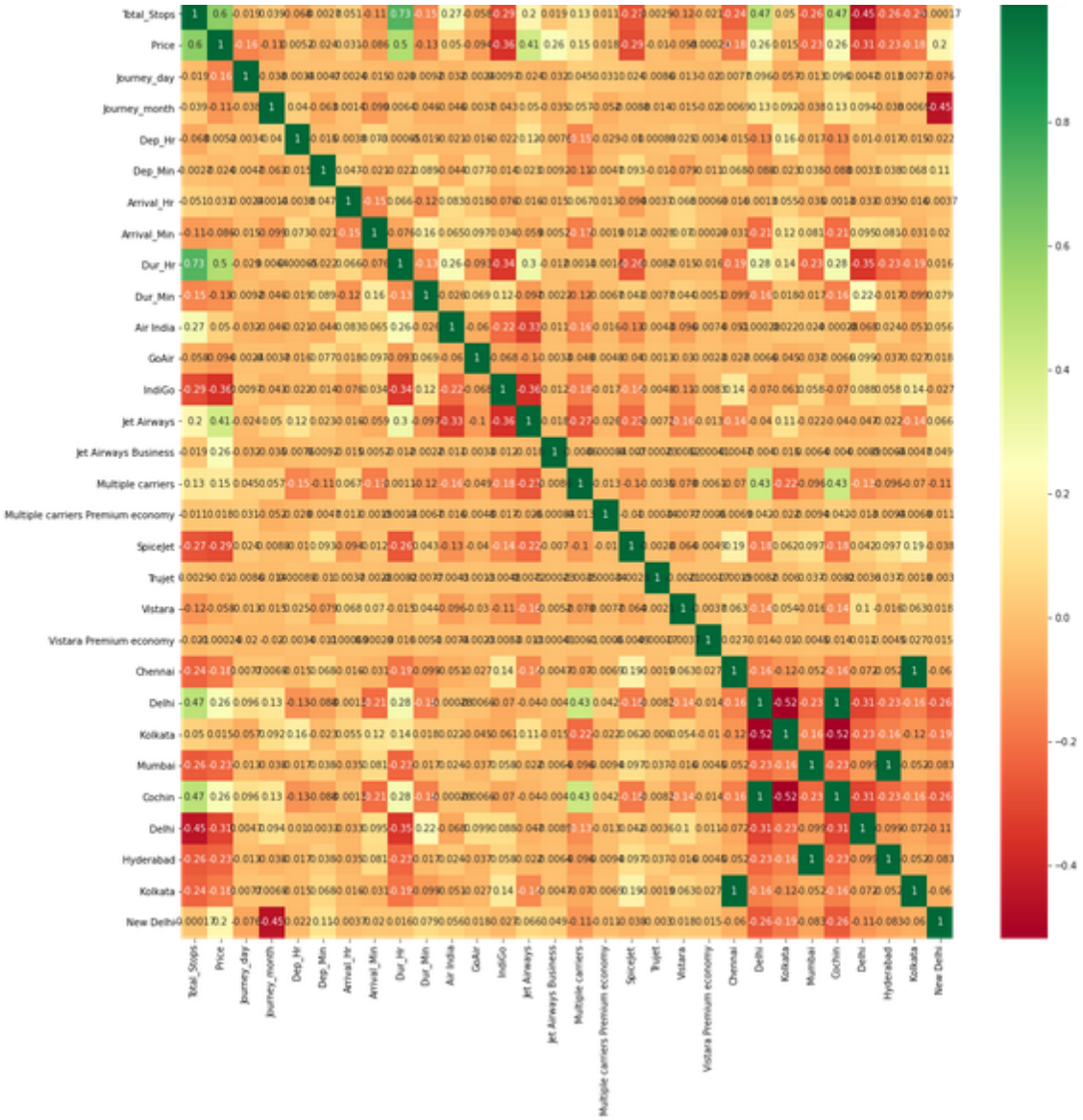

Fig 33: Heatmap

- Finds correlation between Independent and dependent attributes

Correlation:
Variables within a dataset can be related for lots of reasons.

For example:

- One variable could cause or depend on the values of another variable.
- One variable could be lightly associated with another variable.
- Two variables could depend on a third unknown variable.

It can be useful in data analysis and modeling to better understand the relationships between variables. The statistical relationship between two variables is referred to as their correlation.

A correlation could be positive, meaning both variables move in the same direction, or negative,

meaning that when one variable's value increases, the other variables' values decrease. Correlation can also be neutral or zero, meaning that the variables are unrelated.

- **Positive Correlation:** Two features (variables) can be positively correlated with each other. It means that when the value of one variable increases then the value of the other variable(s) also increases.

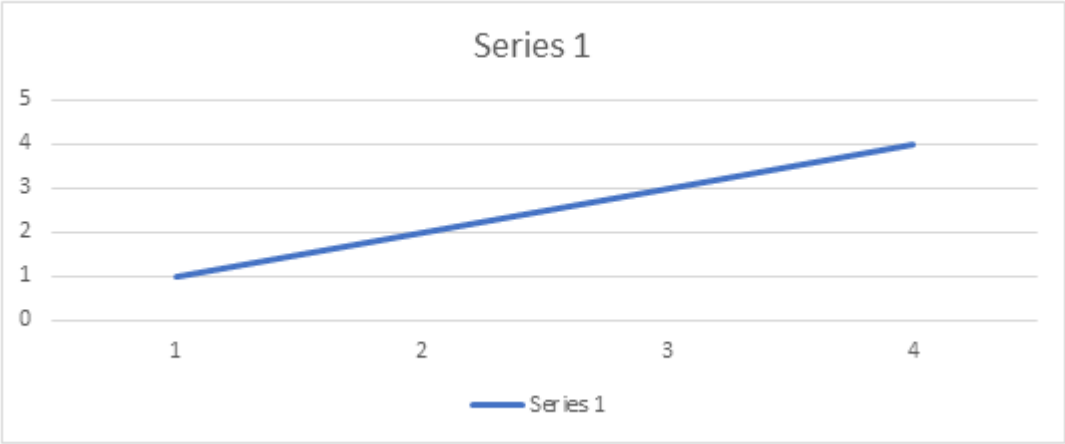

Fig 34:Positive correlation

- **Negative Correlation:** Two features (variables) can be negatively correlated with each other. It means that when the value of one variable increases then the value of the other variable(s) decreases.

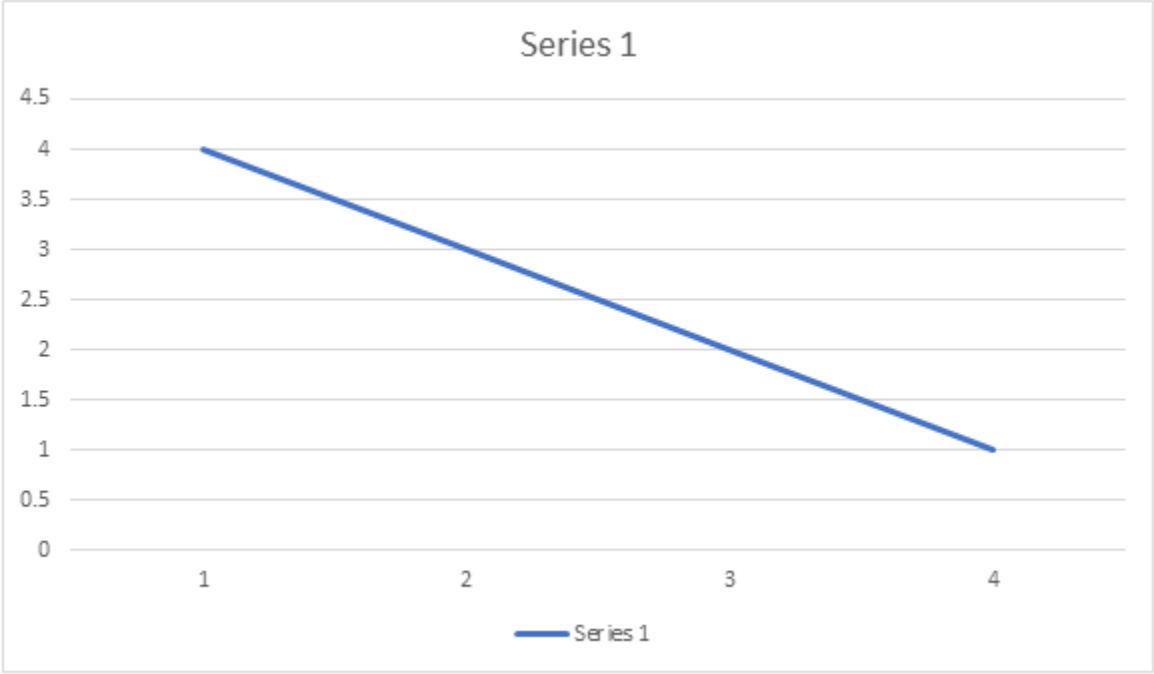

Fig 35:Negative Correlation

- **No Correlation:** Two features (variables) are not correlated with each other. It means that when the value of one variable increases or decreases then the value of the other variable(s) doesn't increase or decrease.

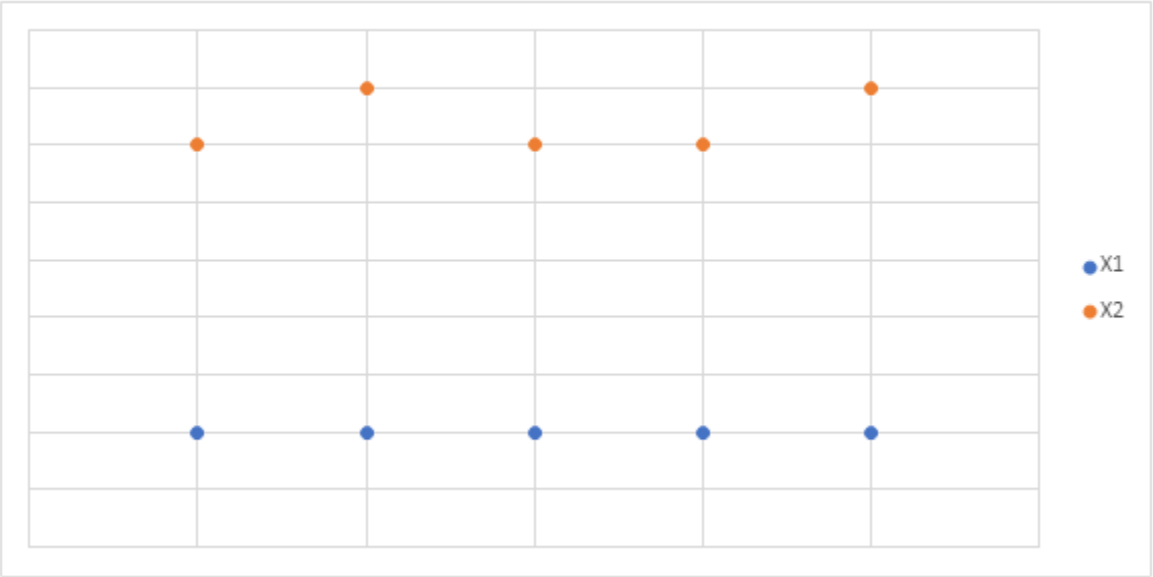

Fig 36:No Correlation

The performance of some algorithms can deteriorate if two or more variables are tightly related, called multicollinearity. An example is linear regression, where one of the offending correlated variables should be removed in order to improve the skill of the model.

We may also be interested in the correlation between input variables with the output variable in order to provide insight into which variables may or may not be relevant as input for developing a model.

The structure of the relationship may be known, e.g. it may be linear, or we may have no idea whether a relationship exists between two variables or what structure it may take. Depending what is known about the relationship and the distribution of the variables, different correlation scores can be calculated.

```
from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(x, y)
```

```
ExtraTreesRegressor()
```

```
print(selection.feature_importances_)
```

```
[2.07412567e-01 1.44624028e-01 5.22874086e-02 2.41395819e-02
 2.14410058e-02 2.77654532e-02 1.94656376e-02 1.47843300e-01
 1.68643304e-02 9.98039105e-03 1.92373062e-03 1.53832890e-02
 1.33838646e-01 6.87065323e-02 2.11828922e-02 8.94173984e-04
 2.98742859e-03 1.02677537e-04 4.77841990e-03 7.86177133e-05
 3.45449681e-04 3.93151163e-03 9.18297775e-03 2.05011990e-03
 3.53953789e-04 7.09280464e-03 4.13684224e-03 6.75982215e-03
 8.75641924e-03 7.86938994e-03 2.08381019e-03 3.80216904e-04
 2.53565713e-02]
```

Fig 37:Feature selection

- Important feature using ExtraTreesRegressor

Fig 38 - Code to create plot for visualization of important features

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

Fig 38:code for visualizing important features

Fig 39 - Plotting graph of feature importances for better visualization

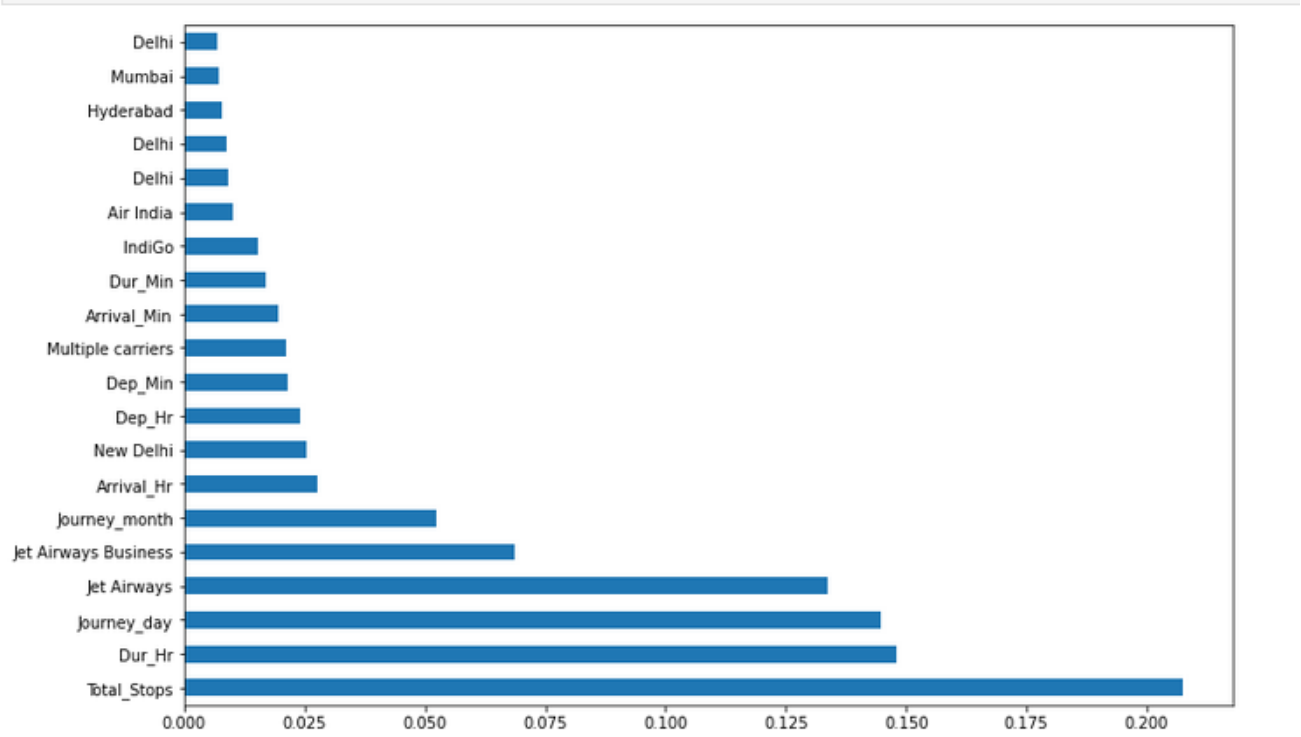

Fig 39: Plot for Feature selection

Feature	**Description**
Departure_hr	At what hour the flight departs from the source
Departure_min	At what Min the flight departs from the source
Arrival_hr	At what hour the flight arrives to the destination
Arrival_min	At what Min the flight arrives to the destination
Journey_month	On what month user wants to travel
Journey_day	On which day of the month the user wants to travel
Source	From where the flight departs
Airlines	Type of airline the user want to travel
Total stops	Number of stops between the Source and destination
Duration_hr	Number of hours user travel in flight
Duration_min	Number of minutes along with Hours user travel

	in flight
Destination	Where the flights arrive

Table II: Provides the description of each feature after encoding is performed

8 . Modeling

Machine learning techniques are used to anticipate flight prices.Support Vector Machine, Linear Regression, K-Nearest Neighbors, Multilayer Perceptron, Gradient Boosting and Random Forest Algorithm, Decision Tree are the algorithms used for predicting. To test the development of such models, I used the Python library and parameters like R-square, MAE, and MSE area unit.

K-Nearest Neighbor

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. The K-Nearest Neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph. There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice. It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).

Multi-layer Perceptron

Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.

A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is depicted below.

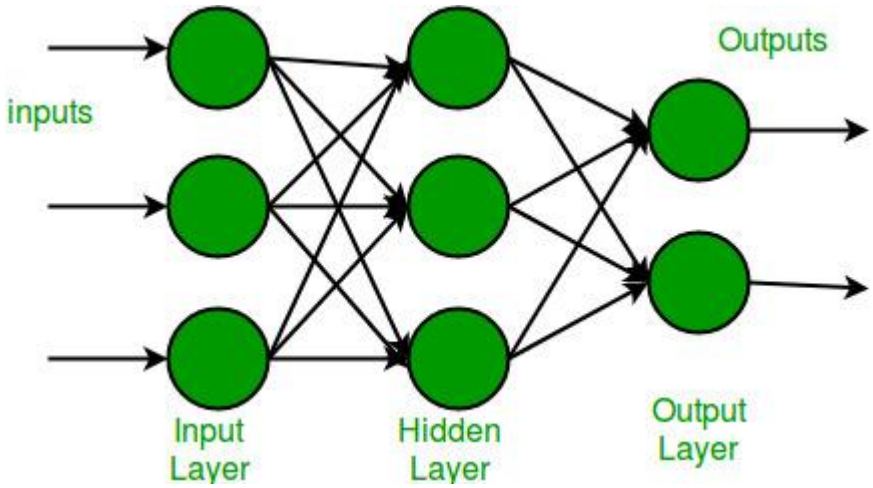

Fig 40: A schematic diagram of a Multi-Layer Perceptron

In the multi-layer perceptron diagram above, we can see that there are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.

Every node in the multilayer perceptron uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula.

Gradient Boosting

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

There is a technique called the **Gradient Boosted Trees** whose base learner is CART (Classification and Regression Trees).

The below diagram explains how gradient boosted trees are trained for regression problems.

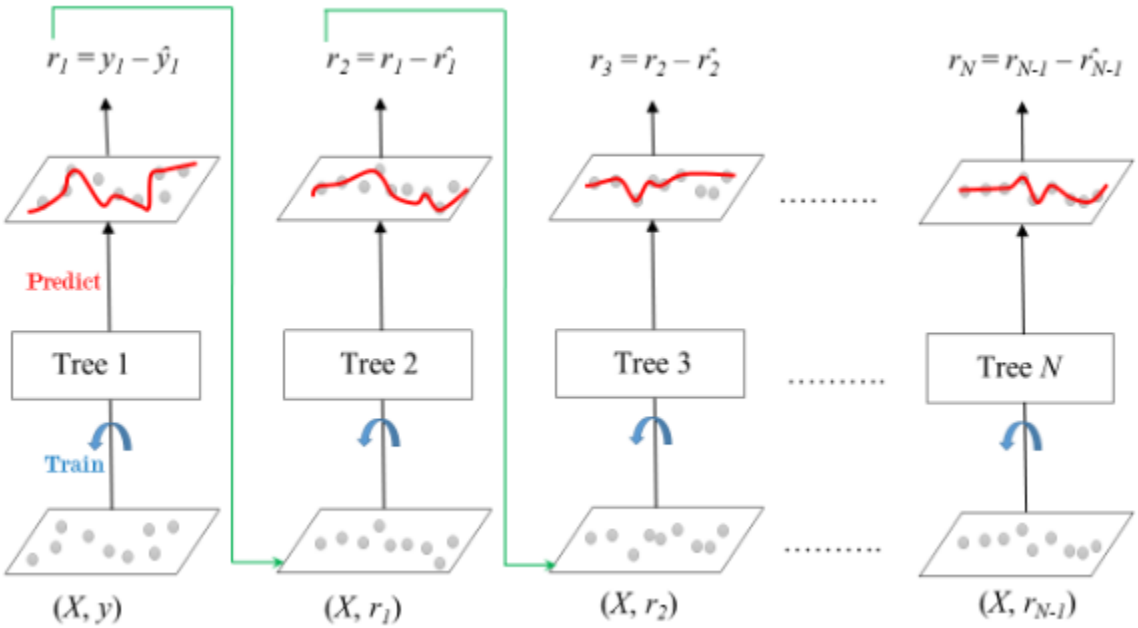

Fig :41 Gradient Boosted Trees for Regression

The ensemble consists of *N* trees. Tree1 is trained using the feature matrix *X* and the labels *y*. The predictions labeled *y1(hat)* are used to determine the training set residual errors *r1*. Tree2 is then trained using the feature matrix *X* and the residual errors *r1* of Tree1 as labels. The predicted results *r1(hat)* are then used to determine the residual *r2*. The process is repeated until all the *N* trees forming the ensemble are trained.

There is an important parameter used in this technique known as **Shrinkage**.

Shrinkage refers to the fact that the prediction of each tree in the ensemble is shrunk after it is multiplied by the learning rate (eta) which ranges between 0 to 1. There is a trade-off between eta and number of estimators, decreasing learning rate needs to be compensated with increasing estimators in order to reach certain model performance. Since all trees are trained now, predictions can be made.

Each tree predicts a label and final prediction is given by the formula,

y(pred) = y1 + (eta * r1) + (eta * r2) + ... + (eta * rN)

Linear Regression

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.

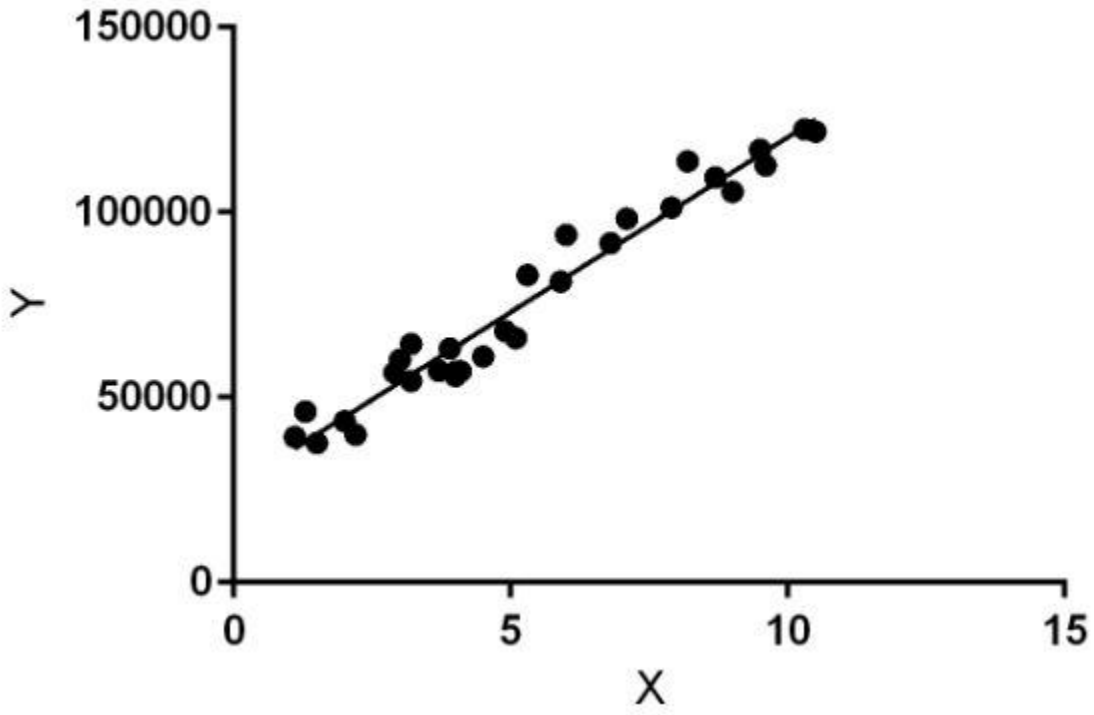

Fig 42:Linear regression illustrated

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2.x$$

Fig 43:Hypothesis for linear regression

While training the model we are given :

x: input training data (univariate – one input variable(parameter))

y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

θ_1: intercept

θ_2: coefficient of x

Once we find the best θ_1 and θ_2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

Cost Function (J):

By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ_1 and θ_2 values, to reach the best value that minimizes the error between predicted y value (pred) and true y value (y).

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

Fig 44:Cost function formula

Cost function(J) of Linear Regression is the **Root Mean Squared Error (RMSE)** between predicted y value (pred) and true y value (y).

Gradient Descent:

To update θ_1 and θ_2 values in order to reduce Cost function (minimizing RMSE value) and achieve the best fit line the model uses Gradient Descent. The idea is to start with random θ_1 and θ_2 values and then iteratively update the values, reaching minimum cost.

Decision Tree Regression

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

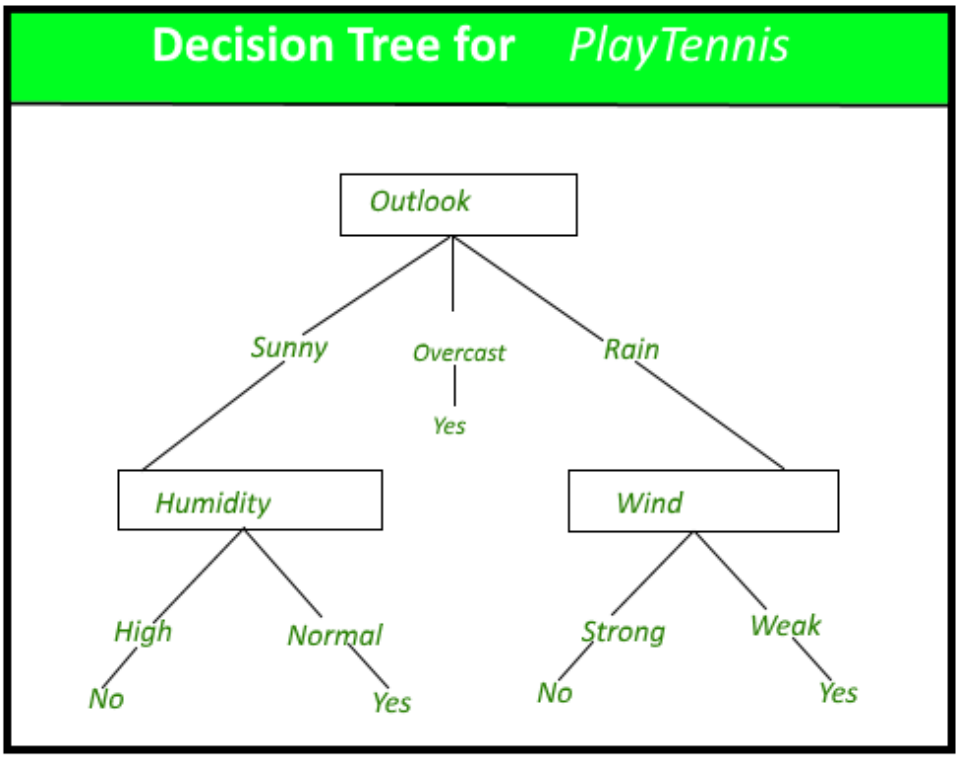

Fig 45:A decision tree for the concept PlayTennis.

Construction of Decision Tree :

A tree can be *"learned"* by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general, the decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Decision Tree Representation :

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

The decision tree in the above figure classifies a particular morning according to whether it is suitable for playing tennis and returning the classification associated with the particular leaf.(in this case Yes or No).

Strengths and Weakness of Decision Tree approach

The strengths of decision tree methods are:

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

The weaknesses of decision tree methods :

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.
- Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

Regression Trees

It supports both continuous and categorical input variables. Regression trees are regarded as research with various machine algorithms for the regression issue, with the Decision Tree approach providing the lowest loss. The R-Squared value for the Decision Tree is 0.998, indicating that it is an excellent model. The Decision Tree was used to complete the web development.

Support Vector Regression

Support Vector Machine (SVM) is a relatively simple **Supervised Machine Learning Algorithm** used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line.

In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data. So by this, you must have understood that inherently, SVM can only perform binary classification (i.e., choose between two classes). However, there are various techniques to use for multi-class problems.

Support Vector Machine for Multi-Class Problems

To perform SVM on multi-class problems, we can create a binary classifier for each class of the data. The two results of each classifier will be :

- The data point belongs to that class OR
- The data point does not belong to that class.

For example, in a class of fruits, to perform multi-class classification, we can create a binary classifier for each fruit. For say, the 'mango' class, there will be a binary classifier to predict if it IS a mango OR it is NOT a mango. The classifier with the highest score is chosen as the output of the SVM.

SVM for complex (Non Linearly Separable)

SVM works very well without any modifications for linearly separable data. **Linearly Separable Data** is any data that can be plotted in a graph and can be separated into classes using a straight line.

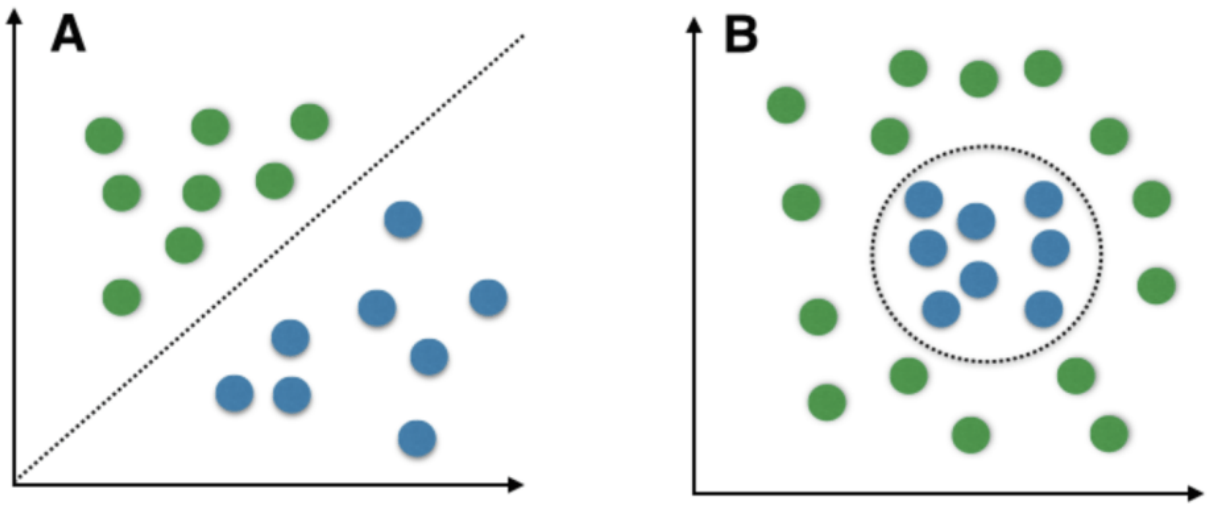

Fig 46: A: Linearly Separable Data B: Non-Linearly Separable Data

We use **Kernelized SVM** for non-linearly separable data. Say, we have some non-linearly separable data in one dimension. We can transform this data into two-dimensions and the data will become linearly separable in two dimensions. This is done by mapping each 1-D data point to a corresponding 2-D ordered pair.

So for any non-linearly separable data in any dimension, we can just map the data to a higher dimension and then make it linearly separable. This is a very powerful and general transformation.

A **kernel** is nothing but a measure of similarity between data points. The **kernel function** in a kernelized SVM tells you, given two data points in the original feature space, what the similarity is between the points in the newly transformed feature space.

There are various kernel functions available, but two of are very popular :

- **Radial Basis Function Kernel (RBF):** The similarity between two points in the transformed feature space is an exponentially decaying function of the distance between the vectors and the original input space as shown below. RBF is the default kernel used in SVM.
-
- **Polynomial Kernel:** The Polynomial kernel takes an additional parameter, 'degree' that controls the model's complexity and computational cost of the transformation

A very interesting fact is that SVM does not actually have to perform this actual transformation on the data points to the new high dimensional feature space. This is called the **kernel trick**.

The Kernel Trick:

Internally, the kernelized SVM can compute these complex transformations just in terms of similarity calculations between pairs of points in the higher dimensional feature space where the transformed feature representation is implicit.

This similarity function, which is mathematically a kind of complex dot product, is actually the kernel of a kernelized SVM. This makes it practical to apply SVM, when the underlying feature space is complex, or even infinite-dimensional. The kernel trick itself is quite complex and is beyond the scope of this article.

Important Parameters in Kernelized SVC (Support Vector Classifier)

1. **The Kernel** : The kernel is selected based on the type of data and also the type of transformation. By default, the kernel is the Radial Basis Function Kernel (RBF).
2. **Gamma** : This parameter decides how far the influence of a single training example reaches during transformation, which in turn affects how tightly the decision boundaries end up surrounding points in the input space. If there is a small value of gamma, points farther apart are considered similar. So more points are grouped together and have smoother decision boundaries (may be less accurate). Larger values of gamma cause points to be closer together (may cause overfitting).
3. **The 'C' parameter** : This parameter controls the amount of regularization applied on the data. Large values of C mean low regularization which in turn causes the training data to fit very well (may cause overfitting). Lower values of C mean higher regularization which causes the model to be more tolerant of errors (may lead to lower accuracy).

Pros of Kernelized SVM:

1. They perform very well on a range of datasets.
2. They are versatile : different kernel functions can be specified, or custom kernels can also be defined for specific data types.
3. They work well for both high and low dimensional data.

Cons of Kernelized SVM:

1. Efficiency (running time and memory usage) decreases as the size of the training set increases.
2. Needs careful normalization of input data and parameter tuning.
3. Does not provide a direct probability estimator.

4. Difficult to interpret why a prediction was made.

Random Forest Regression

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn't depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

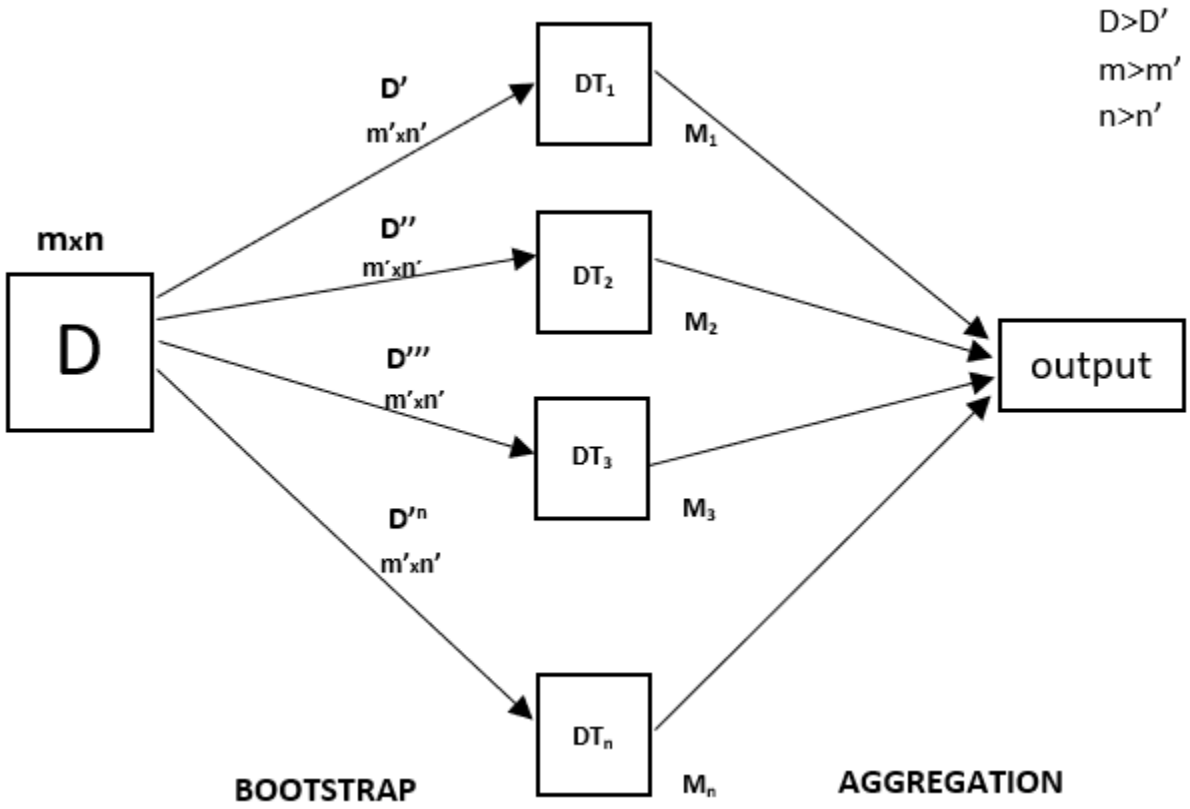

Fig 47: Block diagram for Random Forest

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

We need to approach the Random Forest regression technique like any other machine learning technique

- Design a specific question or data and get the source to determine the required data.

- Make sure the data is in an accessible format else convert it to the required format.
- Specify all noticeable anomalies and missing data points that may be required to achieve the required data.
- Create a machine learning model
- Set the baseline model that you want to achieve
- Train the data machine learning model.
- Provide an insight into the model with test data
- Now compare the performance metrics of both the test data and the predicted data from the model.
- If it doesn't satisfy your expectations, you can try improving your model accordingly or dating your data or use another data modeling technique.
- At this stage you interpret the data you have gained and report accordingly.

Fitting model using Random Forest

1. Split dataset into train and test set in order to prediction w.r.t X_test
2. If needed do scaling of data

    ->Scaling is not done in Random forest

3. Import model
4. Fit the data
5. Predict w.r.t X_test
6. In regression check RSME Score
7. Plot gr

Fig 48 - Splitting the data and fitting regression and predicting the value

```
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)
```

6.2 Import model

```
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
```

6.3 Fit the data

```
reg_rf.fit(X_train, y_train)
```

```
RandomForestRegressor()
```

6.4 Predict with regard to X_test

```
y_pred = reg_rf.predict(X_test)
```

```
reg_rf.score(X_train, y_train)
```

```
0.9536188792120799
```

```
reg_rf.score(X_test, y_test)
```

```
0.8112216503435365
```

Fig 48:Splitting data

Fig 49 - shows it is right skew

```
sns.distplot(y_test-y_pred)
plt.show()
```

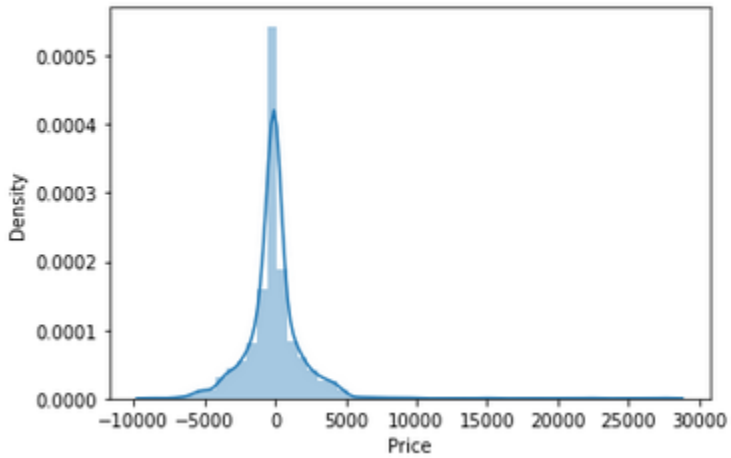

Fig 49:Distplot

Fig 50 - shows there are very less outliners

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

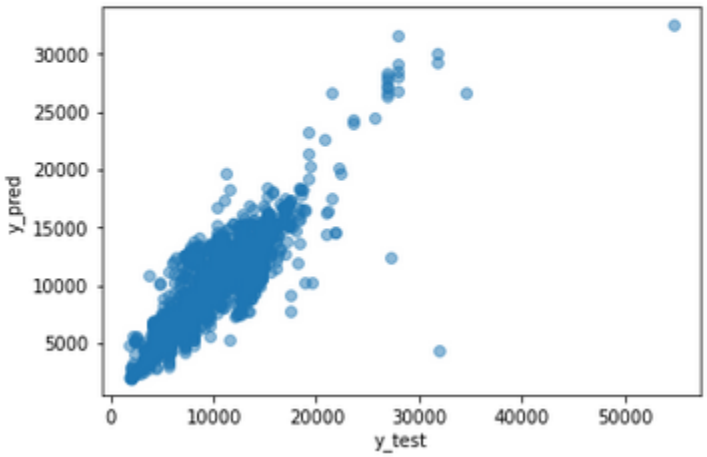

Fig 50: Scatter plot

Hyperparameter Tuning

While building an AI model, you'll be given arrangement decisions for how to decide the plan of your model. We fundamentally have no clue about what the best model designing is for a particular model, thus we should have the choice to investigate various roads in regards to a grouping of decisions. In praiseworthy AI style, we'll demand that the machine embrace this examination and normally pick the best model designing. The limits that conclude the model designing are known as hyperparameters, and the technique associated with noticing the ideal model plan is known as hyperparameter tuning.

Fig 51 - implementing randomized search

```
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]
```

Fig 51: Randomized search

model_selection - Surprise provides various tools to run cross-validation procedures and search the best parameters for a prediction algorithm. The tools presented here are all heavily inspired from the excellent scikit learn library.

Randomized SearchCV - RandomizedSearchCV implements a "fit" and a "score" method. It also implements "score_samples", "predict", "predict_proba", "decision_function", "transform" and "inverse_transform" if they are implemented in the estimator used. The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings. In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n_iter. If all parameters are presented as a list, sampling without replacement is performed. If at least one parameter is given as a distribution, sampling with replacement is used. It is highly recommended to use continuous distributions for continuous parameters.

```
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}
```

Fig 52: Random grid

- Creating Random Grid
- From these results, we should be able to narrow the range of values for each hyperparameter.

```
rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 10, cv = 5, verbose=2, random_state=42
```

```
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   5.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   5.0s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   4.3s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   4.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   5.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   7.6s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   8.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   6.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   7.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   7.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   5.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   4.9s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   5.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   5.0s
```

Fig 53: Fitting model

Random search of parameters, using 5 fold cross validation,

search across 100 different combinations

```
rf_random.best_params_
```

```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

```
prediction = rf_random.predict(X_test)
```

Fig 54: best parameters

best_params_ - best_params_ gives the best combination of tuned hyperparameters.

predict() - The predict() function accepts only a single argument which is usually the data to be tested. It returns the labels of the data passed as arguments based upon the learned or trained data obtained from the model. Thus, the predict() function works on top of the trained model and makes use of the learned label to map and predict the labels for the data to be tested.

Fig 55 - shows it is right skew

```
plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()
```

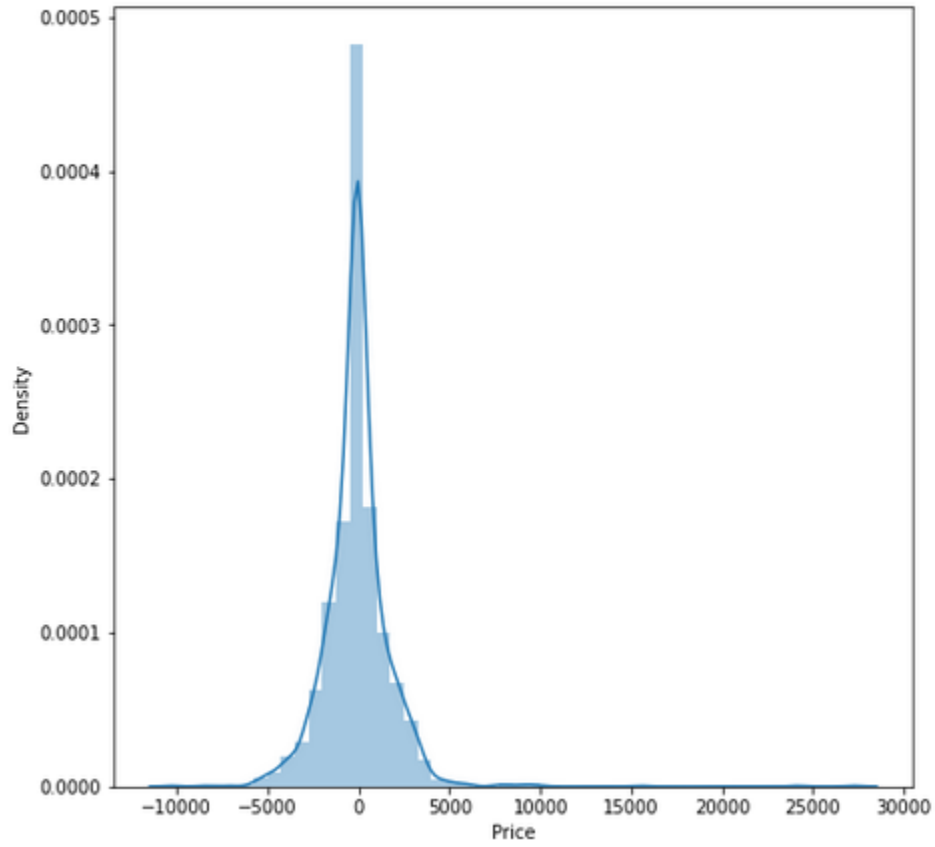

Fig 55: distplot-2

plt.figure() - The purpose of using plt.figure() is to create a figure object. The whole figure is regarded as the figure object. It is necessary to explicitly use plt.figure() when we want to tweak the size of the figure and when we want to add multiple Axes objects in a single figure.

distplot() - A distribution plot, also known as a Distplot, displays the variance in data distribution. The total distribution of continuous data variables is shown by the Seaborn Distplot.

Fig 56 - shows there are very less outliners

```
plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

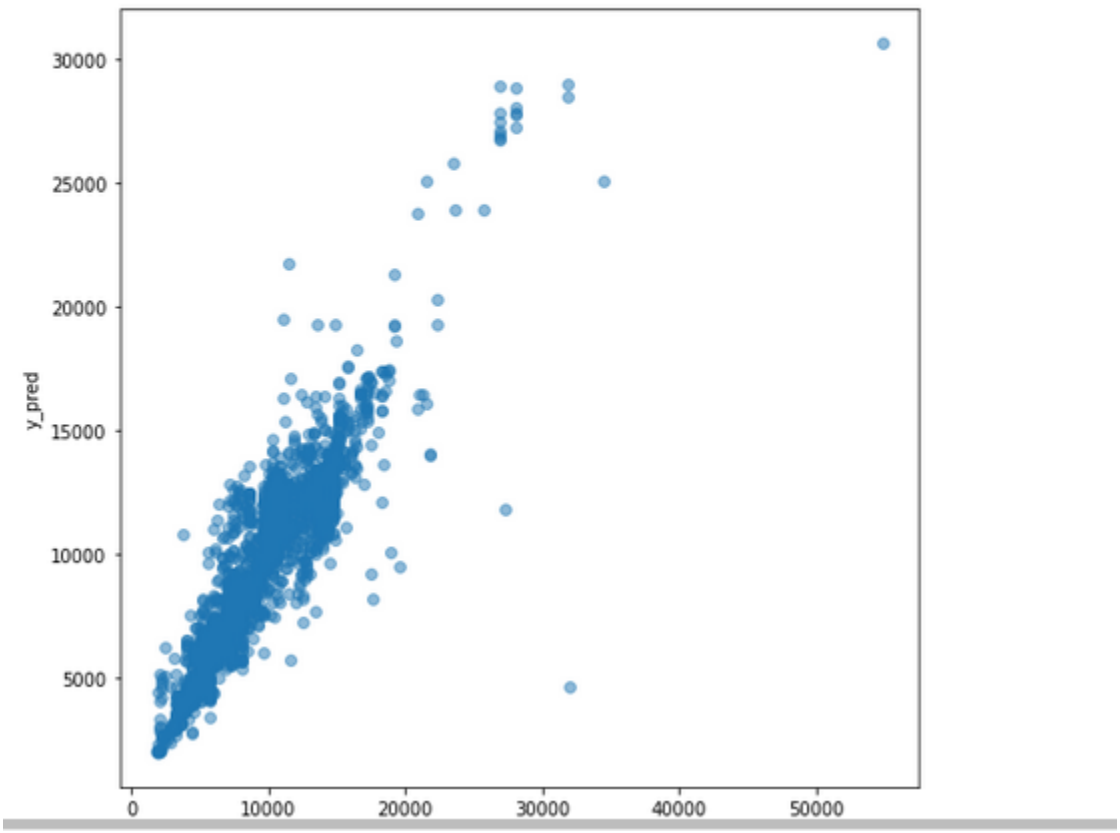

Fig 56: Scatter plot-2

9. Save the model to reuse it again

Fig 57 - Saving model in pickle file

```
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_rf.pkl','rb')
forest = pickle.load(model)
```

```
y_prediction = forest.predict(X_test)
```

```
metrics.r2_score(y_test, y_prediction)
```

```
0.8112216503435365
```

Fig 57: pickle file

The pickle module implements binary protocols for serializing and de-serializing a Python object structure. *"Pickling"* is the process whereby a Python object hierarchy is converted into a byte stream, and *"unpickling"* is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as "serialization", "marshaling," 1 or "flattening"; however, to avoid confusion, the terms used here are "pickling" and "unpickling

10. Frontend

The front-end is built using a combination of technologies such as Hypertext Markup Language (HTML), JavaScript and Cascading Style Sheets (CSS).

Fig 58: Logo of HTML & CSS

Front-end developers design and construct the user experience elements on the web page or app including buttons, menus, pages, links, graphics and more.

Fig 59:Css frameworks

HTML

HTML is a **markup** language for **describing** web documents (web pages). HTML (HyperText Markup Language) is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables.

- The opening tag: This consists of the name of the element (in this case, p), wrapped in opening and closing angle brackets. This states where the element begins or starts to take effect — in this case where the paragraph begins.
- The closing tag: This is the same as the opening tag, except that it includes a *forward slash* before the element name. This states where the element ends — in this case where the paragraph ends. Failing to add a closing tag is one of the standard beginner errors and can lead to strange results.
- The content: This is the content of the element, which in this case, is just text.
- The element: The opening tag, the closing tag, and the content together comprise the element
- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- A markup language is a set of **markup tags**
- HTML documents are described by **HTML tags**
- Each HTML tag **describes** different document content
- <!DOCTYPE html> — doctype. It is a required preamble. In the mists of time, when HTML was young (around 1991/92), doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML, which could mean automatic error checking and other useful things. However these days, they don't do much and are basically just needed to make sure your document behaves correctly. That's all you need to know for now.
- <html></html> — the <html> element. This element wraps all the content on the entire page and is sometimes known as the root element.
- <head></head> — the <head> element. This element acts as a container for all the stuff you want to include on the HTML page that *isn't* the content you are showing to your page's viewers. This includes things like keywords and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more.

- <meta charset="utf-8"> — This element sets the character set your document should use to UTF-8 which includes most characters from the vast majority of written languages. Essentially, it can now handle any textual content you might put on it. There is no reason not to set this and it can help avoid some problems later on.
- <title></title> — the <title> element. This sets the title of your page, which is the title that appears in the browser tab the page is loaded in. It is also used to describe the page when you bookmark/favorite it.
- <body></body> — the <body> element. This contains *all* the content that you want to show to web users when they visit your page, whether that's text, images, videos, games, playable audio tracks, or whatever else.

Html Tags

HTML tags are **keywords** (tag names) surrounded by **angle brackets**:

<tagname>content</tagname>

- HTML tags normally come **in pairs** like and
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- The end tag is written like the start tag, but with a **slash** before the tag name
- The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
- Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
- HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.
- HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[2] A form of HTML, known as HTML5, is used to display video and audio, primarily using the <canvas> element, in collaboration with javascript.

Web Browsers

The purpose of a web browser (Chrome, IE, Firefox, Safari) is to read HTML documents and display them

CSS

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files**
- CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.
- HTML was NEVER intended to contain tags for formatting a web page!
- HTML was created to **describe the content** of a web page, like:
- <h1>This is a heading</h1>
- This is a paragraph.
- When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.
- To solve this problem, the World Wide Web Consortium (W3C) created CSS.
- CSS removed the style formatting from the HTML page!
- The style definitions are normally saved in external .css files.
- With an external stylesheet file, you can change the look of an entire website by changing just one file!

Fig 58- In this figure we are specifying the Doctype of our code and language.

```
<!DOCTYPE html>
<html lang="en">
```

Fig 60: Specify Doctype and Language

<!DOCTYPE html>:

The <!DOCTYPE> Declaration

The <!DOCTYPE> declaration helps the browser to display a web page correctly. There are different document types on the web. To display a document correctly, the browser must know both type and version.

The doctype declaration is not case sensitive. All cases are acceptable:

<!DOCTYPE html>

<!DOCTYPE HTML>

<!doctype html>

<!Doctype Html>

Common Declarations

HTML5

<!DOCTYPE html>

HTML 4.01

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

XHTML 1.0

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

HTML versions:

Version	Year
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

Table III: Versions and year released of HTML

<html lang="en">:

The LANG tag (i.e. the lang="" attribute) is designed to **signal screen readers pronunciation engines** to switch to another language. For this reason and other, tagging Web text as being in a particular language is required in WCAG 2.0.

WCAG 2.0 Guideline 3.1.1—"The default human language of each Web page can be programmatically determined. "

Even more critical is to use language tagging to signal a switch in languages.

WCAG 2.0 Guideline 3.1.2—"The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the

immediately surrounding text. "

These are some codes for languages:

These codes are supported in many screen readers, including JAWS.

Language	Code	Variants
English	en	• American English – Code: **en-US** • British English – Code: **en-GB**
Spanish	es	• Castillian Spanish – Code: **es-ES** • Mexican Spanish – Code: **es-MX** • Other Spanish National Variants
French	fr	• Canadian French – Code: **fr-CA** • Other French National Variants
Italian	it	No Major Variations. See Italian Page for dialect codes
Portuguese	pt	• Brazilian Portuguese – Code: **pt-BR** • European Portuguese – Code: **pt-PT** • See Portuguese page

Table IV: Western European Languages

Language	Code	Variants
Arabic	ar	See Arabic information
Chinese	zh	• Simplified Chinese – Code: **zh-CN**

		• Traditional Chinese – Code: **zh-TW** • Hong Kong – Code: **zh-HK** • Other Chinese variants
Hebrew	he	No Major Variants
Hindi	hi	No Major Variants
Japanese	ja	No Major Variants
Korean	ko	No Major Variants
Swahili	sw	No Major Variants

Table V : Non-Western European Languages

Language	**Code**	**Variants**
Ancient Greek	grc	• Modern Greek: **el**
Latin	la	No Major Variants
Old English	ang	No Major Variants
Middle English	enm	No Major Variants

Table VI: Ancient Languages

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

Fig 61: Starting metadata

Head tag:

The <head> element is a container for metadata (data about data) and is placed between the <html> tag and the <body> tag.

Metadata is data about the HTML document. Metadata is not displayed.Metadata typically define the document title, character set, styles, scripts, and other meta information.

The following elements can go inside the <head> element:

- <title> (required in every HTML document)
- <style>
- <base>
- <link>
- <meta>
- <script>
- <noscript>

Meta tag:

The <meta> tag defines metadata about an HTML document. Metadata is data (information) about data.

<meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.

Metadata will not be displayed on the page, but is machine parsable.

Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.

There is a method to let web designers take control over the viewport (the user's visible area of a web page), through the <meta> tag

Attribute	Value	Description
charset	character_set	Specifies the character encoding for the HTML document
content	text	Specifies the value associated with the http-equiv or name attribute
name	application-name author description generator keywords viewport	Specifies a name for the metadata
http-equiv	content-security-policy content-type default-style refresh	Provides an HTTP header for the information/value of the content attribute

Table VII: Attributes

View port:

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

Charset:

The Unicode Consortium develops the Unicode Standard. Their goal is to replace the existing character sets with its standard Unicode Transformation Format (UTF). The Unicode Standard has become a success and is implemented in HTML, XML, Java, JavaScript, E-mail, ASP, PHP, etc. The Unicode standard is also supported in many operating systems and all modern browsers. The Unicode Consortium cooperates with the leading standards development organizations, like ISO, W3C, and ECMA.

Unicode can be implemented by different character sets. The most commonly used encodings are UTF-8 and UTF-16:

Character-set	Description
UTF-8	A character in UTF8 can be from 1 to 4 bytes long. UTF-8 can represent any character in the Unicode standard. UTF-8 is backwards compatible with ASCII. UTF-8 is the preferred encoding for e-mail and web pages
UTF-16	16-bit Unicode Transformation Format is a variable-length character encoding for Unicode, capable of encoding the entire Unicode repertoire. UTF-16 is used in major operating systems and environments, like Microsoft Windows, Java and .NET.

Table VIII: The Unicode Character Sets

Title tag:

The <title> tag defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.

The <title> tag is required in HTML documents!

The contents of a page title is very important for search engine optimization (SEO)! The page

title is used by search engine algorithms to decide the order when listing pages in search results.

The <title> element:

- defines a title in the browser toolbar
- provides a title for the page when it is added to favorites
- displays a title for the page in search-engine results

Here are some tips for creating good titles:

- Go for a longer, descriptive title (avoid one- or two-word titles)
- Search engines will display about 50-60 characters of the title, so try not to have titles longer than that
- Do not use just a list of words as the title (this may reduce the page's position in search results)

So, try to make the title as accurate and meaningful as possible!

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

Fig 62: Bootstrap

Bootstrap:

Bootstrap is a free front-end framework for faster and easier web development

Bootstrap includes HTML,CSS and design templates for typography,forms,buttons,tables,navigation,models,image carousels and many other, as well as optional javascript plugins

Bootstrap also gives you the ability to easily create responsive designs.

Link tag:

The <link> tag defines the relationship between the current document and an external resource. The <link> tag is most often used to link to external style sheets or to add a favicon to your website. The <link> element is an empty element, it contains attributes only.

Attribute	Value	Description
crossorigin	anonymous use-credentials	Specifies how the element handles cross-origin requests
href	URL	Specifies the location of the linked document

hreflang	language_code	Specifies the language of the text in the linked document
media	media_query	Specifies on what device the linked document will be displayed
refferrerpolicy	no-referrer no-referrer-when-downgrade origin origin-when-cross-origin unsafe-url	Specifies which referrer to use when fetching the resource
rel	alternate author dns-prefetch help icon license next pingback preconnect prefetch preload prerender prev search stylesheet	Required. Specifies the relationship between the current document and the linked document
sizes	HeightxWidth any	Specifies the size of the linked resource. Only for rel="icon"
title		Defines a preferred or an alternate stylesheet
type	media_type	Specifies the media type of the linked document

Table IX: Attributes of link tag and their descriptions

Href attribute:

The href attribute specifies the URL of the page the link goes to. If the href attribute is not present, the <a> tag will not be a hyperlink. You can use href="#top" or href="#" to link to the top of the current page!

Integrity attribute:

The integrity attribute allows a browser to check the fetched script to ensure that the code is never loaded if the source has been manipulated. Subresource Integrity (SRI) is a W3C

specification that allows web developers to ensure that resources hosted on third-party servers have not been altered. Use of SRI is recommended! When using SRI, the webpage holds the hash and the server holds the file (the .js file in this case). The browser downloads the file, then checks it, to make sure that it is a match with the hash in the integrity attribute. If it matches, the file is used, and if not, the file is blocked.

Cross origin:

The crossorigin attribute sets the mode of the request to an HTTP CORS Request. Web pages often make requests to load resources on other servers. Here is where CORS comes in. A cross-origin request is a request for a resource (e.g. style sheets, iframes, images, fonts, or scripts) from another domain. CORS is used to manage cross-origin requests. CORS stands for Cross-Origin Resource Sharing, and is a mechanism that allows resources on a web page to be requested from another domain outside their own domain. It defines a way of how a browser and server can interact to determine whether it is safe to allow the cross-origin request. CORS allows servers to specify who can access the assets on the server, among many other things. The opposite of cross-origin requests is same-origin requests. This means that a web page can only interact with other documents that are also on the same server. This policy enforces that documents that interact with each other must have the same origin (domain).

Rel tag:

The rel attribute specifies the relationship between the current document and the linked document. Only used if the href attribute is present. Search engines can use this attribute to get more information about a link!

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

Fig 63: Connecting css page

Stylesheet : Imports a style sheet

```
<style>

.design {
    color: white;
    font-family: Arial, Helvetica, sans-serif;
    font-style: italic;
    font-weight: bold;
}

body{
    background-color: aqua;
}

</style>
```

Fig 64: Internal Css

Internal Css:

An internal CSS is used to define a style for a single HTML page. An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

Style attribute:

The <style> tag is used to define style information (CSS) for a document. Inside the <style> element you specify how HTML elements should render in a browser.

Color:

There are following three different methods to set colors in your web page −

- **Color names** − You can specify color names directly like green, blue or red.
- **Hex codes** − A six-digit code representing the amount of red, green, and blue that makes up the color.
- **Color decimal or percentage values** − This value is specified using the rgb( ) property.

We used color names in this.

font-family:

The CSS font-family property defines the font to be used.

The font-family property should hold several font names as a "fallback" system, to ensure maximum compatibility between browsers/operating systems. Start with the font you want, and end with a generic family (to let the browser pick a similar font in the generic family, if no other fonts are available). The font names should be separated with commas.

In CSS there are five generic font families:

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.

All the different font names belong to one of the generic font families.

font-style:

The font-style property specifies the font style for a text.

Value	Description
normal	The browser displays a normal font style. This is default
italic	The browser displays an italic font style
oblique	The browser displays an oblique font style
initial	Set this property to its default value.
inherit	Inherits this property from its parent element.

Table X: Properties value of font-style and their description

Font-weight:

The font-weight property sets how thick or thin characters in text should be displayed.

Value	Description
normal	Defines normal characters. This is default
bold	Defines thick characters
bolder	Defines thicker characters
lighter	Defines lighter characters
100 200 300 400 500 600 700 800 900	Defines from thin to thick characters. 400 is the same as normal, and 700 is the same as bold
initial	Set this property to its default value.

inherit	Inherits this property from its parent element.

Table XI: Property values of font-weight and their descriptions

background-color:

The background-color property sets the background color of an element. The background of an element is the total size of the element, including padding and border (but not the margin). Use a background color and a text color that makes the text easy to read.

Value	Description
color	Specifies the background color.
transparent	Specifies that the background color should be transparent. This is default
initial	Set this property to its default value.
inherit	Inherits this property from its parent element.

Table XII: Property values of background-color and their descriptions

```
<body>
    <!-- As a heading -->
                <h1 align="center" style="font-family: algerian; background-color: black; color: white;"> Flight Fare Prediction </h1>
    <br><br><br>
```

Fig 65: Staring of body and Heading

Body tag:

- The <body> tag defines the document's body.
- The <body> element contains all the contents of an HTML document, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- There can only be one <body> element in an HTML document.
- The <body> tag also supports the Global attributes in HTML.
- The <body> tag also supports theEvent attributes in HTML.

Headings (<h1>):

HTML headings are defined with the <h1> to <h6> tags. <h1> defines the most important heading. <h6> defines the least important heading. Browsers automatically add some white space (a margin) before and after a heading. Search engines use the headings to index the structure and content of your web pages. Users often skim a page by its headings. It is important to use headings to show the document structure. <h1> headings should be used for main headings, followed by <h2> headings, then the less important <h3>, and so on. Use HTML

headings for headings only. Don't use headings to make text **BIG** or **bold**. Each HTML heading has a default size. However, you can specify the size for any heading with the style attribute, using the CSS font-size property. HTML headings can also be used with nested elements. Following are different codes to display the way to use heading elements.

**break(
):**

The
 tag inserts a single line break. The
 tag is useful for writing addresses or poems. The
 tag is an empty tag which means that it has no end tag. Use the
 tag to enter line breaks, not to add space between paragraphs.The
 tag also supports the Global attributes in HTML. The
 tag also supports theEvent attributes in HTML.

align tag:

The text-align property specifies the horizontal alignment of text in an element.

Value	Description
left	Aligns the text to the left
right	Aligns the text to the right
center	Centers the text
justify	Stretches the lines so that each line has equal width (like in newspapers and magazines)
initial	Set this property to its default value.
inherit	Inherits this property from its parent element.

Table XIII: Property values of align and their descriptions

```
<div class="container" >


<form action="\predict" method="post">
```

Fig 66: starting form

div tag:

- The tag defines a division or a section in an HTML document.
- The tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript.
- The tag is easily styled by using the class or id attribute.
- Any sort of content can be put inside the tag!

- By default, browsers always place a line break before and after the element.
- The tag also supports the Global attributes in HTML.
- The tag also supports theEvent attributes in HTML.

class attribute:

- The class attribute specifies one or more classnames for an element.
- The class attribute is mostly used to point to a class in a style sheet. However, it can also be used by a JavaScript (via the HTML DOM) to make changes to HTML elements with a specified class.
- The class attribute also supports the Global attributes in HTML.
- The Class attribute tag also supports the Event attributes in HTML.

container:

Containers are used to pad the content inside of them, and there are two container classes available:

1. The .container class provides a responsive **fixed width container**
2. The .container-fluid class provides a **full width container**, spanning the entire width of the viewport

Fixed Container:

Use the **.container** class to create a responsive, fixed-width container.

its width (max-width) will change on different screen sizes

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px
max-width	100%	540px	720px	960px

Table XIV: Max-width on different screen sizes

Fluid Container:

Use the **.container-fluid clas**s to create a full width container, that will always span the entire width of the screen (width is always 100%)

Container Padding:

By default, containers have 15px left and right padding, with no top or bottom padding. Therefore, we often use spacing utilities, such as extra padding and margins to make them look even better. For example, .pt-3 means "add a top padding of 16px"

form tag:
The <form> tag is used to create an HTML form for user input.

The <form> element can contain one or more of the following form elements:

- <input>
- <textarea>
- <button>
- <select>
- <option>
- <optgroup>
- <fieldset>
- <label>
- <output>

The <form> tag also supports the Global attributes in HTML.

The <form> tag also supports theEvent attributes in HTML.

Attribute	Value	Description
accept-charset	character_set	Specifies the character encodings that are to be used for the form submission
action	URL	Specifies where to send the form-data when a form is submitted
autocomplete	on off	Specifies whether a form should have autocomplete on or off
enctype	application/x-www-form-urlencoded multipart/form-data text/plain	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")
method	get post	Specifies the HTTP method to use when sending form-data
name	text	Specifies the name of a form
novalidate	novalidate	Specifies that the form should not be validated when submitted
rel	external help license next	Specifies the relationship between a linked resource and the current document

	nofollow noopener noreferrer opener prev search	
target	_blank _self _parent _top	Specifies where to display the response that is received after submitting the form

Table XV: Attributes of form tag and their descriptions

action:

The action attribute specifies where to send the form-data when a form is submitted.

method:

The method attribute specifies how to send form-data (the form-data is sent to the page specified in the action attribute).

The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").

Post:

- Appends form-data inside the body of the HTTP request (data is not shown in URL)
- Has no size limitations
- Form submissions with POST cannot be bookmarked

```
<div class="row">
    <div class="col-sm-6 ">
        <div class="card text-white bg-secondary mb-3">
            <div class="card-body">
                <h5 class="card-title design" >Departure Date</h5>
```

Fig 67:creating classes for inputting data

row:

Some Bootstrap grid system rules:

- Rows must be placed within a .container (fixed-width) or .container-fluid (full-width) for proper alignment and padding
- Use rows to create horizontal groups of columns

- Content should be placed within columns, and only columns may be immediate children of rows
- Predefined classes like .row and .col-sm-4 are available for quickly making grid layouts
- Columns create gutters (gaps between column content) via padding. That padding is offset in rows for the first and last column via negative margin on .rows
- Grid columns are created by specifying the number of 12 available columns you wish to span. For example, three equal columns would use three .col-sm-4
- Column widths are in percentage, so they are always fluid and sized relative to their parent element

col-sm:

- Bootstrap 4 grid system offers a set of responsive classes to specify on what screens a certain layout works.
- Bootstrap Col-SM (**small**) classes applies a grid column class to an element when the screen is wider than 576px.
- You can add them to your layouts by typing col-sm-*
- Bootstrap **small** grid column classes apply when the screen is wider than 576px and collapses otherwise
- Using the Bootstrap grid system, you can use a set of responsive classes that specify what screens a certain layout works on.
- Bootstrap **small** class applies when the screen is wider than 576px.
- Applying Bootstrap Col-SM class without defining the number of columns creates an auto layout.

cards:

A card in Bootstrap 4 is a bordered box with some padding around its content. It includes options for headers, footers, content, colors, etc.

You can simply use the background and color utility classes to change the appearance of a card.

bg means background and mb means margin bottom in code.

Contextual Cards:

To add a background color the card, use contextual classes (.bg-primary, .bg-success, .bg-info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark and .bg-light

card-body:

Cards support a wide variety of content, including images, text, list groups, links, and more. Below are examples of what's supported.

The building block of a card is the .card-body. Use it whenever you need a padded section within a card.

card-title:

Use .card-title to add card titles to any heading element. The .card-text class is used to remove

bottom margins for a <p> element if it is the last child (or the only one) inside the .card-body. The .card-link class adds a blue color to any link, and a hover effect.

```
                <input type="datetime-local" name="Dep_Time" id="Dep_Time" required="required" class="form-control">
            </div>
        </div>
    </div>
    <br>
    <br>
    <br>
```

Fig 68: Input details for departure time

Input tag:

- The <input> tag specifies an input field where the user can enter data.
- The <input> element is the most important form element.
- The <input> element can be displayed in several ways, depending on the type attribute.
- The <input> tag also supports the Global attributes in HTML

- The <input> tag also supports the Event attributes in HTML.

The different input types are as follows:

- <input type="button">
- <input type="checkbox">
- <input type="color">
- <input type="date">
- <input type="datetime-local">
- <input type="email">
- <input type="file">
- <input type="hidden">
- <input type="image">
- <input type="month">
- <input type="number">
- <input type="password">
- <input type="radio">
- <input type="range">
- <input type="reset">
- <input type="search">
- <input type="submit">
- <input type="tel">
- <input type="text"> (default value)
- <input type="time">
- <input type="url">
- <input type="week">

Always use the <label> tag to define labels for <input type="text">, <input type="checkbox">, <input type="radio">, <input type="file">, and <input type="password">.

datetime-local:

- The <input type="datetime-local"> defines a date picker.
- The resulting value includes the year, month, day, and time.
- Always add the <label> tag for best accessibility practices!

name:

- The name attribute specifies the name of an <input> element.
- The name attribute is used to reference elements in a JavaScript, or to reference form data after a form is submitted.
- Only form elements with a name attribute will have their values passed when submitting a form.

id:

The **id** attribute of an HTML tag can be used to uniquely identify any element within an HTML page. There are two primary reasons that you might want to use an id attribute on an element −

- If an element carries an id attribute as a unique identifier, it is possible to identify just that element and its content.
- If you have two elements of the same name within a Web page (or style sheet), you can use the id attribute to distinguish between elements that have the same name.
- The id attribute also supports the Global attributes in HTML

- The id attribute also supports the Event attributes in HTML.

required:

The Boolean **required** attribute, if present, indicates that the user must specify a value for the input before the owning form can be submitted.The required attribute is supported by text, search, url, tel, email, password, date, month, week, time, datetime-local, number, checkbox, radio, file, <input> types along with the <select> and <textarea> form control elements. If present on any of these input types and elements, the :required pseudo class will match. If the attribute is not included, the :optional pseudo class will match.The attribute is not supported or relevant to range and color, as both have default values. It is also not supported on hidden as it can not be expected that a user to fill out a form that is hidden. Nor is it supported on any of the button types, including image.Note color and range don't support required, but type color defaults to #000000, and range defaults to the midpoint between min and max -- with min and max defaulting to 0 and 100 respectively in most browsers if not declared -- so always has a value.

In the case of a same named group of radio buttons, if a single radio button in the group has the required attribute, a radio button in that group must be checked, although it doesn't have to be the one with the attribute is applied. So to improve code maintenance, it is recommended to either include the required attribute in every same-named radio button in the group, or else in none.

In the case of a same named group of checkbox input types, only the checkboxes with the required attribute are required.

HTML Form Controls

There are different types of form controls that you can use to collect data using HTML form −

- Text Input Controls
- Checkboxes Controls
- Radio Box Controls
- Select Box Controls
- File Select boxes
- Hidden Controls
- Clickable Buttons
- Submit and Reset Button

Text Input Controls

There are three types of text input used on forms −

- **Single-line text input controls** − This control is used for items that require only one line of user input, such as search boxes or names. They are created using the HTML **<input>** tag.
- **Password input controls** − This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tags.
- **Multi-line text input controls** − This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using the HTML **<textarea>** tag.

Single-line text input controls

This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag

Password input controls

This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input>tag but type attribute is set to **password**.

Multiple-Line Text Input Controls

This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using the HTML <textarea> tag.

Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **checkbox.**

Radio Button Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to radio.

Select Box Control

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

File Upload Box

If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to **file**.

Button Controls

There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input>tag by setting its type attribute to **button**.

```
<select name="Source" id="Source" required="required" class="form-control">
    <option value="Delhi">Delhi</option>
    <option value="Kolkata">Kolkata</option>
    <option value="Mumbai">Mumbai</option>
    <option value="Chennai">Chennai</option>
</select>
```

Fig 69: Selecting source

Select tag:

- The <select> element is used to create a drop-down list.
- The <select> element is most often used in a form, to collect user input.
- The name attribute is needed to reference the form data after the form is submitted (if you omit the name attribute, no data from the drop-down list will be submitted).
- The id attribute is needed to associate the drop-down list with a label.
- The <option> tags inside the <select> element define the available options in the drop-down list.
- Always add the <label> tag for best accessibility practices!
- The <select> tag also supports the Global attributes in HTML

- The <select> tag also supports the Event attributes in HTML.

Attribute	Value	Description
autofocus	autofocus	Specifies that the drop-down list should automatically get focus when the page loads
disabled	disabled	Specifies that a drop-down list should be disabled
form	*form_id*	Defines which form the drop-down list belongs to
multiple	multiple	Specifies that multiple options can be selected at once
name	*name*	Defines a name for the drop-down list
required	required	Specifies that the user is required to select a value before submitting the form
size	*number*	Defines the number of visible options in a drop-down list

Table XVI: Attributes its values and their description of select tag

Option tag:

- The <option> tag defines an option in a select list.
- <option> elements go inside a <select> , <optgroup>, or <datalist> element.
- The <option> tag can be used without any attributes, but you usually need the **value** attribute, which indicates what is sent to the server on form submission.
- If you have a long list of options, you can group related options within the <optgroup> tag.
- The <option> tag also supports the Global attributes in HTML
- The <option> tag also supports the Event attributes in HTML.

Attribute	Value	Description
disabled	disabled	Specifies that an option should be disabled
label	*text*	Specifies a shorter label for an option
selected	selected	Specifies that an option should be pre-selected when the page loads
value	*text*	Specifies the value to be sent to a server

Table XVII: Attributes its values and their description of option tag

value:

- The value attribute specifies the value to be sent to a server when a form is submitted.
- The content between the opening <option> and closing </option> tags is what the browsers will display in a drop-down list. However, the value of the value attribute is what will be sent to the server when a form is submitted.
- If the value attribute is not specified, the content will be passed as a value instead.

```
   <center> <input type="submit" value="Submit" class="btn btn-secondary"></center>
</form>
```

Fig 70: submit button

Center tag:

The HTML ** tag** is used to center the text horizontally in the HTML document. Since this tag was removed in HTML5, it is recommended that you use the CSS text-align property to format the text horizontally in the document. This tag is also commonly referred to as the element. Only the Global Attributes apply to the tag. There are no attributes that are specific to the tag.

submit:

- The <input type="submit"> defines a submit button which submits all form values to a form-handler.
- The form-handler is typically a server page with a script for processing the input data.
- The form-handler is specified in the form's action attribute.

Buttons tag:

The .btn classes are designed to be used with the <button> element. However, you can also use these classes on <a> or <input> elements (though some browsers may apply a slightly different rendering).

When using button classes on <a> elements that are used to trigger in-page functionality (like collapsing content), rather than linking to new pages or sections within the current page, these links should be given a role="button" to appropriately convey their purpose to assistive technologies such as screen readers.

Outline buttons

In need of a button, but not the hefty background colors they bring? Replace the default modifier classes with the .btn-outline-* ones to remove all background images and colors on any button.

Sizes

Fancy larger or smaller buttons? Add .btn-lg or .btn-sm for additional sizes.

Active state

Buttons will appear pressed (with a darker background, darker border, and inset shadow) when active. **There's no need to add a class to <button>s as they use a pseudo-class**. However, you can still force the same active appearance with .active (and include the aria-pressed="true" attribute) should you need to replicate the state programmatically.

Disabled state

Make buttons look inactive by adding the disabled boolean attribute to any <button> element.

Disabled buttons using the <a> element behave a bit different:

- <a>s don't support the disabled attribute, so you must add the .disabled class to make it visually appear disabled.
- Some future-friendly styles are included to disable all pointer-events on anchor buttons. In browsers which support that property, you won't see the disabled cursor at all.
- Disabled buttons should include the aria-disabled="true" attribute to indicate the state of the element to assistive technologies.

Button plugin

Do more with buttons. Control button states or create groups of buttons for more components like toolbars.

Toggle states

Add data-toggle="button" to toggle a button's active state. If you're pre-toggling a button, you must manually add the .active class **and** aria-pressed="true" to the <button>.

Checkbox and radio buttons

Bootstrap's .button styles can be applied to other elements, such as <label>s, to provide checkbox or radio style button toggling. Add data-toggle="buttons" to a .btn-group containing those modified buttons to enable their toggling behavior via JavaScript and add .btn-group-toggle to style the <input>s within your buttons. **Note that you can create single input-powered buttons or groups of them.**

The checked state for these buttons is **only updated via click event** on the button. If you use another method to update the input—e.g., with <input type="reset"> or by manually applying the input's checked property—you'll need to toggle .active on the <label> manually.

Note that pre-checked buttons require you to manually add the .active class to the input's <label>.

```
<p>Vemunuri Gnaneshwar Reddy</p> </center>
```

Fig 71: paragraph

P tag:

- The ** ** HTML element represents a paragraph. Paragraphs are usually represented in visual media as blocks of text separated from adjacent blocks by blank lines and/or first-line indentation, but HTML paragraphs can be any structural grouping of related content, such as images or form fields.
- Paragraphs are block-level elements, and notably will automatically close if another block-level element is parsed before the closing tag
- This element only includes the global attributes.

Styling paragraphs

By default, browsers separate paragraphs with a single blank line. Alternate separation methods, such as first-line indentation, can be achieved with CSS

Breaking up content into paragraphs helps make a page more accessible. Screen-readers and other assistive technology provide shortcuts to let their users skip to the next or previous paragraph, letting them skim content like how white space lets visual users skip around.

Using empty elements to add space between paragraphs is problematic for people who navigate with screen-reading technology. Screen readers may announce the paragraph's presence, but not any content contained within it — because there is none. This can confuse and frustrate the person using the screen reader.

```
<!-- 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>
```

Fig 72: Javascript

JavaScript:

JavaScript is a lightweight, interpreted **programming** language. It is designed for creating network-centric applications. It is complementary to and integrated with Java. **JavaScript** is very easy to implement because it is integrated with HTML. It is open and cross-platform.

Javascript is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Javascript:

- Javascript is the most popular **programming language** in the world and that makes it a programmer's great choice. Once you learn Javascript, it helps you develop great front-end as well as back-end softwares using different Javascript based frameworks like jQuery, Node.JS etc.
- Javascript is everywhere, it comes installed on every modern web browser and so to learn Javascript you really do not need any special environment setup. For example Chrome, Mozilla Firefox , Safari and every browser you know as of today, supports Javascript.
- Javascript helps you create really beautiful and crazy fast websites. You can develop your website with a console like look and feel and give your users the best Graphical User Experience.
- JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for you as a Javascript Programmer.
- Due to high demand, there is tons of job growth and high pay for those who know JavaScript. You can navigate over to different job sites to see what having JavaScript skills looks like in the job market.
- Great thing about Javascript is that you will find tons of frameworks and Libraries already developed which can be used directly in your software development to reduce your time to market.

There could be 1000s of good reasons to learn Javascript Programming. But one thing for sure, to learn any **programming language**, not only Javascript, you just need to code, and code and

finally code until you become an expert.

There are many useful **Javascript frameworks** and libraries available:

- Angular
- React
- jQuery
- Vue.js
- Ext.js
- Ember.js
- Meteor
- Mithril
- Node.js
- Polymer
- Aurelia
- Backbone.js

It is really impossible to give a complete list of all the available Javascript frameworks and libraries. The Javascript world is just too large and too much new is happening.

As mentioned before, **Javascript** is one of the most widely used **programming languages** (Front-end as well as Back-end). It has it's presence in almost every area of software development. I'm going to list few of them here:

- **Client side validation** - This is really important to verify any user input before submitting it to the server and Javascript plays an important role in validating those inputs at the front-end itself.
- **Manipulating HTML Pages** - Javascript helps in manipulating HTML pages on the fly. This helps in adding and deleting any HTML tag very easily using javascript and modifying your HTML to change its look and feel based on different devices and requirements.
- **User Notifications** - You can use Javascript to raise dynamic pop-ups on the webpages to give different types of notifications to your website visitors.
- **Back-end Data Loading** - Javascript provides Ajax library which helps in loading back-end data while you are doing some other processing. This really gives an amazing experience to your website visitors.
- **Presentations** - JavaScript also provides the facility of creating presentations which gives the website look and feel. JavaScript provides RevealJS and BespokeJS libraries to build web-based slide presentations.
- **Server Applications** - Node JS is built on Chrome's Javascript runtime for building fast and scalable network applications. This is an event based library which helps in developing very sophisticated server applications including Web Servers.

This list goes on, there are various areas where millions of software developers are happily using Javascript to develop great websites and other softwares.

JS

Many of our components require the use of JavaScript to function. Specifically, they require jQuery, Popper.js, and our own JavaScript plugins. We use jQuery's slim build, but the full version is also supported.

Place **one of the following <script>s** near the end of your pages, right before the closing </body> tag, to enable them. jQuery must come first, then Popper.js, and then our JavaScript plugins.

Bundle

Include everything you need in one script with our bundle. Our bootstrap.bundle.js and bootstrap.bundle.min.js include Popper, but not jQuery. For more information about what's included in Bootstrap, please see our contents section.

Separate

If you decide to go with the separate scripts solution, Popper.js must come first, and then our JavaScript plugins.

Components

Curious which components explicitly require jQuery, our JS, and Popper.js? Click the show components link below. If you're unsure about the page structure, keep reading for an example page template.

Starter template

Be sure to have your pages set up with the latest design and development standards. That means using an HTML5 doctype and including a viewport meta tag for proper responsive behaviors

11. Backend

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

WSGI

Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.

Werkzeug

It is a WSGI toolkit, which implements requests, response objects, and other utility functions.

This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

Jinja2

Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have a built-in abstraction layer for database handling, nor does it have form validation support. Instead, Flask supports the extensions to add such functionality to the application

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

Fig73:Importing libraries for Flask

flask.templating render_template :

render_template is a Flask function from the flask.templating package. render_template is used to generate output from a template file based on the Jinja2 engine that is found in the application's templates folder.

Note that render_template is typically imported directly from the flask package instead of from flask.templating. It is the same function that is imported, but there are less characters to type when you leave off the templating part.

render_template_string is another callable from the flask.templating package with code examples.

These topics are also useful while reading the render_template examples:

- template engines, specifically Jinja2
- Flask and the concepts for web frameworks
- Cascading Style Sheets (CSS) and web design

Flask-Cors:

A Flask extension for handling Cross Origin Resource Sharing (CORS), making cross-origin AJAX possible.

This package has a simple philosophy: when you want to enable CORS, you wish to enable it for all use cases on a domain. This means no mucking around with different allowed headers, methods, etc.

By default, submission of cookies across domains is disabled due to the security implications.

Please see the documentation for how to enable credentialed requests, and please make sure you add some sort of CSRF protection before doing so!

This package exposes a Flask extension which by default enables CORS support on all routes, for all origins and methods. It allows parameterization of all CORS headers on a per-resource level. The package also contains a decorator, for those who prefer this approach.

cross_origin:

Initializes Cross Origin Resource sharing for the application. The arguments are identical to cross_origin(), with the addition of a resources parameter. The resources parameter defines a series of regular expressions for resource paths to match and optionally, the associated options to be applied to the particular resource. These options are identical to the arguments to cross_origin().

The settings for CORS are determined in the following order

1. Resource level settings (e.g when passed as a dictionary)
2. Keyword argument settings
3. App level configuration settings (e.g. CORS_*)
4. Default settings

As it is possible for multiple regular expressions to match a resource path, the regular expressions are first sorted by length, from longest to shortest, in order to attempt to match the most specific regular expression. This allows the definition of a number of specific resource options, with a wildcard fallback for all other resources.

Pickle:

Python pickle module is used for serializing and de-serializing python object structures. The process to convert any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshaling. We can convert the byte stream (generated through pickling) back into python objects by a process called unpickling.

In real world scenarios, the use of pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.

It is advisable not to unpickle data received from an untrusted source as they may pose a security threat. However, the pickle module has no way of knowing or raising alarm while pickling malicious data.Only after importing the pickle module we can do pickling and unpickling.

Below are some of the common exceptions raised while dealing with pickle module −

- Pickle.PicklingError: If the pickle object doesn't support pickling, this exception is raised.
- Pickle.UnpicklingError: In case the file contains bad or corrupted data.
- EOFError: In case the end of file is detected, this exception is raised.

Prons:

- Comes handy to save complicated data.
- Easy to use, lighter and doesn't require several lines of code.
- The pickled file generated is not easily readable and thus provides some security.

Cons:

- Languages other than python may not be able to reconstruct pickled python objects.
- Risk of unpickling data from malicious sources.

```
model = pickle.load((open("Flight_rf.pkl", "rb")))

app = Flask(__name__)
```

Fig 74:Loading Pickle file

Unpickling data:

pickle.load(file, *, fix_imports=True, encoding='ASCII', errors='strict', buffers=None)

Read the pickled representation of an object from the open file object file and return the reconstituted object hierarchy specified therein. This is equivalent to Unpickler(file).load().

The protocol version of the pickle is detected automatically, so no protocol argument is needed. Bytes past the pickled representation of the object are ignored.

Arguments file, fix_imports, encoding, errors, strict and buffers have the same meaning as in the Unpickler constructor.Note the usage of "rb" instead of "r" as we are reading bytes. This is a very basic example, be sure to try more on your own.

__name__:

Python sets the __name__ variable to the module name, so the value of this variable will vary depending on the Python source file in which you use it.

For example, in a module named test.py that is located in the top-level directory of the application, the value of __name__ is test. If the test.py module is located inside a Python package called my_package, then the value of __name__ is my_package.test.

There are two special exceptions with regards to the value of __name__:

- Inside a __init__.py package constructor module, the value of __name__ is the package name, without __init__. For example, in my_package/__init__.py, the value of __name__ is just my_package.
- In the main module of the application (the file you run the Python interpreter on) the value of __name__ has the special value of __main__.

The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central

registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an __init__.py file inside) or a standard module (just a .py file).

For more information about resource loading, see open_resource().

Usually you create a Flask instance in your main module

The idea of the first parameter is to give Flask an idea of what belongs to your application. This name is used to find resources on the filesystem, can be used by extensions to improve debugging information and a lot more.

So it's important what you provide there. If you are using a single module, __name__ is always the correct value. If you however are using a package, it's usually recommended to hardcode the name of your package there.

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

Fig 75: Routing and returning in html

Flask App routing:

App routing is used to map the specific URL with the associated function that is intended to perform some task. It is used to access some particular page like Flask Tutorial in the web application.

In our first application, the URL ('/') is associated with the home function that returns a particular string displayed on the web page.

In other words, we can say that if we visit the particular URL mapped to some particular function, the output of that function is rendered on the browser's screen.

Flask facilitates us to add the variable part to the URL by using the section. We can reuse the variable by adding that as a parameter into the view function.

The converter can also be used in the URL to map the specified variable to the particular data type. For example, we can provide the integers or floats like age or salary respectively.

The following converters are used to convert the default string type to the associated data type.

1. string: default
2. int: used to convert the string to the integer
3. float: used to convert the string to the float.
4. path: It can accept the slashes given in the URL

Returning in form of html:

It is possible to return the output of a function bound to a certain URL in the form of HTML.

However, generating HTML content from Python code is cumbersome, especially when variable data and Python language elements like conditionals or loops need to be put. This would require frequent escaping from HTML.

This is where one can take advantage of **Jinja2** template engine, on which Flask is based. Instead of returning hardcode HTML from the function, a HTML file can be rendered by the **render_template()** function.

Flask will try to find the HTML file in the templates folder, in the same folder in which this script is present.

- Application folder
    - app.py
    - templates
        - home.html

The term **'web templating system'** refers to designing an HTML script in which the variable data can be inserted dynamically. A web template system comprises a template engine, some kind of data source and a template processor.

Flask uses a jinja2 template engine. A web template contains HTML syntax interspersed placeholders for variables and expressions (in this case Python expressions) which are replaced values when the template is rendered.

```
@app.route("/predict", methods = ["GET", "POST"])
@cross_origin()
```

Fig 76: HTTP method and cross origin

Flask HTTP methods:

HTTP is the hypertext transfer protocol which is considered as the foundation of data transfer in the world wide web. All web frameworks including flask need to provide several HTTP methods for data communication.

The methods are given in the following table.

SN	Method	Description
1	GET	It is the most common method which can be used to send data in the unencrypted form to the server.
2	HEAD	It is similar to the GET but used without the response body.
3	POST	It is used to send the form data to the server. The server does not cache the data transmitted using the post method.
4	PUT	It is used to replace all the current representation of the target resource with the uploaded content.
5	DELETE	It is used to delete all the current representation of the target resource specified in the URL.

Table XVIII: HTTP methods and their description

POST Method:

The HTTP POST method sends data to the server. The type of the body of the request is indicated by the Content-Type header.

The difference between PUT and POST is that PUT is idempotent: calling it once or several times successively has the same effect (that is no *side* effect), whereas successive identical POST may have additional effects, like passing an order several times.

A POST request is typically sent via an HTML form and results in a change on the server. In this case, the content type is selected by putting the adequate string in the enctype attribute of the <form> element or the formenctype attribute of the <input> or <button> elements:

- application/x-www-form-urlencoded: the keys and values are encoded in key-value tuples separated by '&', with a '=' between the key and the value. Non-alphanumeric characters in both keys and values are percent encoded: this is the reason why this type is not suitable to use with binary data (use multipart/form-data instead)
- multipart/form-data: each value is sent as a block of data ("body part"), with a user

agent-defined delimiter ("boundary") separating each part. The keys are given in the Content-Disposition header of each part.

- text/plain

When the POST request is sent via a method other than an HTML form — like via an XMLHttpRequest — the body can take any type. As described in the HTTP 1.1 specification, POST is designed to allow a uniform method to cover the following functions:

- Annotation of existing resources
- Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;
- Adding a new user through a signup modal;
- Providing a block of data, such as the result of submitting a form, to a data-handling process;
- Extending a database through an append operation.

GET Method:

Let's consider the same example for the Get method. However, there are some changes in the data retrieval syntax on the server side

Cross_origin:

This package exposes a Flask extension which by default enables CORS support on all routes, for all origins and methods. It allows parameterization of all CORS headers on a per-resource level. The package also contains a decorator, for those who prefer this approach.

If the CORS extension does not satisfy your needs, you may find the decorator useful. It shares options with the extension, and should be simple to use.

flask_cors.cross_origin(**args*, ***kwargs*)

This function is the decorator which is used to wrap a Flask route with. In the simplest case, simply use the default parameters to allow all origins in what is the most permissive configuration. If this method modifies state or performs authentication which may be brute-forced, you should add some degree of protection, such as Cross Site Forgery Request protection.

- **origins** (*list, string or regex*) –
  The origin, or list of origins to allow requests from. The origin(s) may be regular expressions, case-sensitive strings, or else an asterisk

  Default : '*'

```
def predict():
    if request.method == "POST":
        dep_date = request.form.get('Dep_Time')
```

Fig 77: Predicting and request from feature

Prediction API

The prediction API is quite simple. We give it our data, the years of experience, and pass that into our predict method of our model.

To call our APIs we're going to use the [requests](http://docs.python-requests.org/en/master/) package which will make it easier to call APIs than using a built-in module for Python. With requests we can call the post method to indicate we want to send a POST request, and pass in our URL

Flask Request Object

In the client-server architecture, the request object contains all the data that is sent from the client to the server. As we have already discussed in the tutorial, we can retrieve the data at the server side using the HTTP methods.

In this section of the tutorial, we will discuss the Request object and its important attributes that are given in the following table.

SN	Attribute	Description
1	Form	It is the dictionary object which contains the key-value pair of form parameters and their values.
2	args	It is parsed from the URL. It is the part of the URL which is specified in the URL after the question mark (?).
3	Cookies	It is the dictionary object containing cookie names and the values. It is saved at the client-side to track the user session.
4	files	It contains the data related to the uploaded file.
5	method	It is the current request method (get or post).

Table XIX: Request object attributes and their description

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

Fig 78: Date conversions and creating new columns

```
dur_hour = abs(arrival_hour - dep_hour)
dur_min =  abs(arrival_min - dep_min)
```

Fig 79: Values for dur_hour and dur_min

abs():

Python abs() function is used to return the absolute value of a number, i.e., it will remove the negative sign of the number.

The abs() takes only one argument, a number whose absolute value is to be returned. The argument can be an integer, a floating-point number, or a complex number.

- If the argument is an integer or floating-point number, abs() returns the absolute value in integer or float.
- In the case of a complex number, abs() returns only the magnitude part and that can also be a floating-point number.

The abs() function makes any negative number to positive, while positive numbers are unaffected. The absolute value of any number is always positive. For any positive number, the absolute value is the number itself and for any negative number, the absolute value is (-1) multiplied by the negative number.

```
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
```

Fig 80: Label encoding for airlines

- We will perform same encoding for all other airlines

```
Source = request.form["Source"]
if (Source == 'Delhi'):
    s_Delhi = 1
    s_Kolkata = 0
    s_Mumbai = 0
    s_Chennai = 0
```

Fig 81: Label encoding for Source

- We will perform all other encoding for all other sources

```
Source = request.form["Destination"]
if (Source == 'Cochin'):
    d_Cochin = 1
    d_Delhi = 0
    d_New_Delhi = 0
    d_Hyderabad = 0
    d_Kolkata = 0
```

Fig 82: Label encoding for Destination

- We will perform all other encoding for all other Destinations

```
        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=f"Your Flight price is Rs. {output}")
```

Fig 83: Output

We use the /predict route with the GET method as the default method.

In this method, we retrieve the data from the form action in home.html through a GET request. Now, we retrieve the data from each of the input fields in the form, using its name attribute. To retrieve, we use request.args.get().

Using the above command, we retrieve the data from all 10 input fields. Later, it is passed to the predict() method of our Machine Learning model model.py.

To call a method in another file, we specify the filename.methodname(). Here, it is model.classify(). This method returns the variety of the flower as a string data type.

To render the returned value i.e the variety of flowers, we specify the output HTML file along with the arguments to be rendered, using the render_template(filename, arguments) command.

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

Fig 84: Debug mode

Activating the Flask Debug Mode

Consider a Flask Application with **Debug Mode** = False. When you update some code, you need to restart the server for the changes to come upon the web page.

This can be quite repetitive since we keep changing and updating our code. So to make coding easy, Flask gives us the Debug Mode!

Hence, with Debug **Mode** on, all the application code changes will get updated right away in the development stage, eliminating the need to restart the server.

Never enable the Debug Mode or any other built-in debugger in a production environment. The debugger allows executing arbitrary Python code from the browser. Although that requires a protection key, it is still not secure.

V. Results and Discussion

```
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)))
```

```
MAE: 1147.2435095277435
MSE: 3489796.675032163
RMSE: 1868.0997497543226
```

Fig 85: Printing MAE,MSE,RMSE

MAE:
The Mean Absolute Error (MAE) is only slightly different in definition from the MSE, but interestingly provides almost exactly opposite properties! To calculate the MAE, you take the difference between your model's predictions and the ground truth, apply the absolute value to that difference, and then average it out across the whole dataset.

The MAE, like the MSE, will never be negative since in this case we are always taking the absolute value of the errors. The MAE is formally defined by the following equation:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$

Fig 86: MAE formula

Where:

- n = the number of errors,
- Σ = summation symbol (which means "add them all up"),
- $|x_i - x|$ = the absolute errors.

The formula may look a little daunting, but the steps are easy:

1. Find all of your absolute errors, $x_i - x$.
2. Add them all up.
3. Divide by the number of errors. For example, if you had 10 measurements, divide by 10.

We can easily calculate the mean absolute error in Python by using the mean_absolute_error function from Scikit-learn
Advantage: The beauty of the MAE is that its advantage directly covers the MSE disadvantage. Since we are taking the absolute value, all of the errors will be weighted on the same linear scale. Thus, unlike the MSE, we won't be putting too much weight on our outliers and our loss function provides a generic and even measure of how well our model is performing.

Disadvantage: If we do in fact care about the outlier predictions of our model, then the MAE won't be as effective. The large errors coming from the outliers end up being weighted the exact same as lower errors. This might result in our model being great most of the time, but making a few very poor predictions every so-often.

MSE:
The Mean Squared Error (MSE) is perhaps the simplest and most common loss function, often taught in introductory Machine Learning courses. To calculate the MSE, you take the difference between your model's predictions and the ground truth, square it, and average it out across the whole dataset.

The MSE will never be negative, since we are always squaring the errors. The MSE is formally defined by the following equation:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Fig 87: MSE formula

Where:

- *N* is the number of samples we are testing against
- Yi-y2 is the difference between original values and predicted values
- Σ = summation symbol (which means "add them all up"),

In Python, the MSE can be calculated rather easily, especially with the use of lists

Advantage: The MSE is great for ensuring that our trained model has no outlier predictions with huge errors, since the MSE puts larger weight on theses errors due to the squaring part of the function.

Disadvantage: If our model makes a single very bad prediction, the squaring part of the function magnifies the error. Yet in many practical cases we don't care much about these outliers and are aiming for more of a well-rounded model that performs good enough on the majority.

RMSE:

RMSE is an acronym for **Root Mean Square Error**, which is the **square root of value obtained from Mean Square Error** function.
Another popular way to calculate the error in a set of regression predictions is to use the Root Mean Squared Error.

Shortened as RMSE, the metric is sometimes called Mean Squared Error or MSE, dropping the Root part from the calculation and the name.

RMSE is calculated as the square root of the mean of the squared differences between actual outcomes and predictions.

Squaring each error forces the values to be positive, and the square root of the mean squared error returns the error metric back to the original units for comparision.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(S_i - O_i)^2}$$

Where, n = Data set observations

Si = Predicted values

Oi = Observations

Fig 88: RMSE formula

Using RMSE, we can easily plot a difference between the estimated and actual values of a parameter of the model.

By this, we can clearly judge the efficiency of the model.

Usually, a RMSE score of less than 180 is considered a good score for a moderately or well working algorithm. In case, the RMSE value exceeds 180, we need to perform feature selection and hyper parameter tuning on the parameters of the model

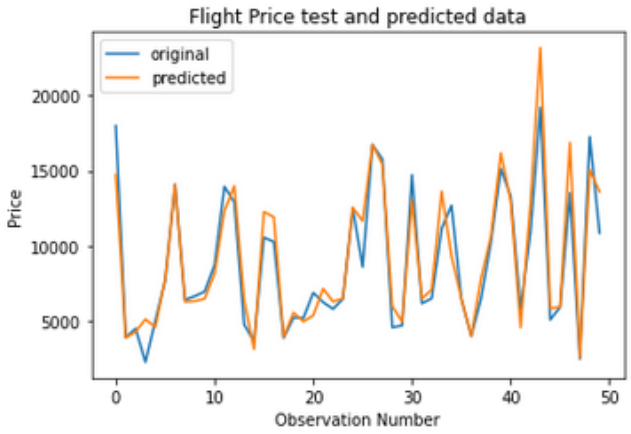

Fig 89: Original price vs predicted price

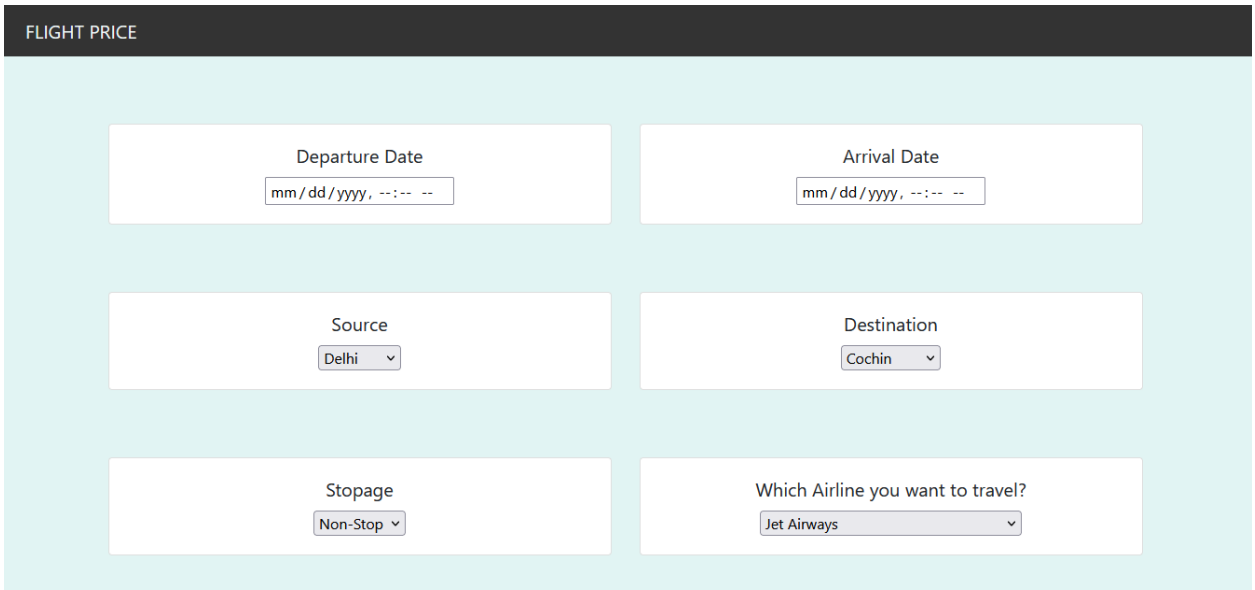

Fig 90: Form to fill details

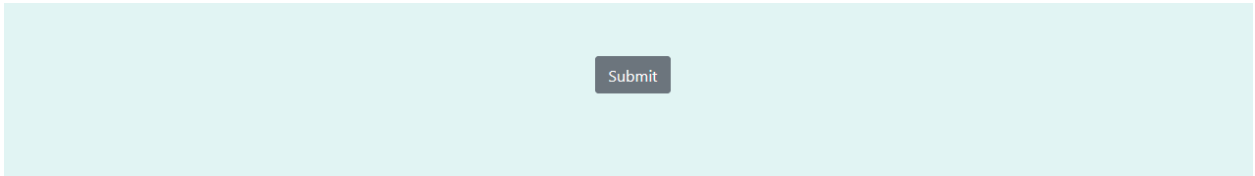

Fig 91: Submit button

VI. CONCLUSION

By using this correctly we can save the money by giving the person information for related people to travel, by this the person can decide to travel on the particular date he wanted to or to change the plans. As we know the accuracy is not good. It has high scope in improving accuracy as data is growing day by day and we can use it as well as we can other types of Machine learning algorithms. We can improve the front end and make it more user friendly. We can get other features data like coupons ,days left until departure,overnight flight (yes or no),holiday day (yes or no) and we optimize the flask app.py, we may use django or any other web frameworks.

VII. REFERENCES

Tao Liu, Jian Cao, Yudong Tan, Quanwu Xiao, “ACER: An Adaptive Context- Aware Ensemble Regression Model for Airfare Price Prediction”, 2017 International Conference on Progress in Informatics and Computing (PIC), December, 2017

Supriya Rajankar, Neha Sakharkar, Omprakash Rajankar, “Predicting the price of a flight ticket with the use of Machine Learning algorithms”, international journal of scientific & technology research volume 8, December, 2019

]www.Makemytrip.com

Tianyi Wang, Samira Pouyanfar, Haiman Tian, Yudong Tao, Miguel Alonso Jr., Steven Luis and Shu-Ching Chen, “A Framework for Airfare Price Prediction: A Machine Learning Approach”, 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), September 9, 2019

William Groves and Maria Gini "An agent for optimizing airline ticket purchasing" in proceedings of the 2013 international conference on autonomous agents and multi-agent systems.

J. Santos Dominguez-Menchero, Javier Rivera and Emilio Torres- Manzanera "Optimal purchase timing in the airline market".

T. Janssen, “A linear quantile mixed regression model for prediction of airline ticket prices,” Bachelor Thesis, Radboud University, 2014.

Juhar Ahmed Abdella, Nazar Zaki and Khaled Shuaib, “Automatic Detection of Airline Ticket Price and Demand: A Review”, 13th International Conference on Innovations in Information technology (IIT), January 10, 2019

W. Groves and M. Gini, “An agent for optimizing airline ticket purchasing,” 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), St. Paul, MN, May 06 - 10, 2013, pp. 1341-1342.

M. Papadakis, “Predicting Airfare Prices,” 2014

[1] Pavithra Maria K and Anitha K L, 2021. “ like Duration, Source, Destination, Arrival, Departure.”. Available at:

<https://iarjset.com/wp-content/uploads/2021/04/IARJSET.2021.8321.pdf>

[2] Pavithra Maria K and Anitha K L, 2021. “airlines must manage demand.When demand is predicted to grow, the airline may raise prices in order to slow the rate at which seats.”. Available at:

<https://iarjset.com/wp-content/uploads/2021/04/IARJSET.2021.8321.pdf>

[3] Pavithra Maria K and Anitha K L, 2021. “ Price, Source, Destination and much more are the , used for flight price prediction..”. Available at:

<https://iarjset.com/wp-content/uploads/2021/04/IARJSET.2021.8321.pdf>

[4] K. Tziridis T. Kalampokas G.Papakostas and K. Diamantaras "Airfare price prediction using machine learning techniques" in European Signal Processing Conference (EUSIPCO), DOI:

10.23919/EUSIPCO .2017.8081365L. Li Y. Chen and Z. Li" Yawning detection for monitoring driver fatigue based on two cameras" Proc. 12th Int. IEEE Conf. Intell. Transp. Syst. pp. 1-6 Oct. 2009.

[5]K. Tziridis, Th. Kalampokas, G. A. Papakostas, "Airfare Prices Prediction Using Machine Learning Techniques", 25th European Signal Processing Conference (EUSIPCO), IEEE, October 26, 2017

[6]Tao Liu, Jian Cao, Yudong Tan, Quanwu Xiao, "ACER: An Adaptive Context- Aware Ensemble Regression Model for Airfare Price Prediction", 2017 International Conference on Progress in Informatics and Computing (PIC), December, 2017

[7]Supriya Rajankar, Neha Sakharkar, Omprakash Rajankar, "Predicting the price of a flight ticket with the use of Machine Learning algorithms", international journal of scientific & technology research volume 8, December, 2019

[8]www.Makemytrip.com

[9]Tianyi Wang, Samira Pouyanfar, Haiman Tian, Yudong Tao, Miguel Alonso Jr., Steven Luis and Shu-Ching Chen, "A Framework for Airfare Price Prediction: A Machine Learning Approach", 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), September 9, 2019

[10] William Groves and Maria Gini "An agent for optimizing airline ticket purchasing" in proceedings of the 2013 international conference on autonomous agents and multi-agent systems.

[11]J. Santos Dominguez-Menchero, Javier Rivera and Emilio Torres- Manzanera "Optimal purchase timing in the airline market".

[12]T. Janssen, "A linear quantile mixed regression model for prediction of airline ticket prices," Bachelor Thesis, Radboud University, 2014.

[13] Juhar Ahmed Abdella, Nazar Zaki and Khaled Shuaib, "Automatic Detection of Airline Ticket Price and Demand: A Review", 13th International Conference on Innovations in Information technology (IIT), January 10, 2019

[14] W. Groves and M. Gini, "An agent for optimizing airline ticket purchasing," 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), St. Paul, MN, May 06 - 10, 2013, pp. 1341-1342.

[15] M. Papadakis, "Predicting Airfare Prices," 2014.

[16] R. Ren, Y. Yang and S. Yuan, "Prediction of airline ticket price," Technical Report, Stanford University, 2015.

[17] Wohlfarth, T. Clemencon, S.Roueff, ―A Dat mining approach to travel price forecasting‖, 10 th international conference on machine learning Honolulu 2011.

[18] Dominguez-Menchero, J.Santo, Riviera, ‖optimal purchase timing in airline markets‖ ,2014

Github link: https://github.com/18bcs3818/Flight_Price_Prediction