AIRLINE TICKETS PRICE PREDICTION USING MACHINE LEARNING AND FLASK

A Mini Project Report

Submitted in partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

AMIREDDY VINUTHNA **(18BT1A0501)**

Under the Guidance of

Ms.Syed Thaisin

Assistant Professor (CSE)

To

VISVESVARAYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Recognized by A.I.C.T.E Affiliated to JNTU, Hyderabad),

(MP Patelguda, Ibrahimpatnam, Telangana, INDIA - 501510)

Visvesvaraya
College of Engineering & Technology
M.P.Patelguda (V), Ibrahimpanam (M), Ranga Reddy Dist- 501510

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that this project report entitled **AIRLINE TICKETS PRICE PREDICTION USING MACHINE LEARNING AND FLASK** submitted by AMIREDDY VINUTHNA (**18BT1A0501**) in partial fulfillment of the requirements for the degree Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University, Hyderabad. During the academic year 2021-22, is a bonafide record of work carried out under our guidance and supervision.

The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

Ms.SYED THAISIN
(Assistant Professor)
(Internal Guide)

Dr.L. KIRAN KUMAR REDDY
(Head of the Department)
(Dept. of CSE)

Internal Examiner

External Examiner

Department of CSE

Visvesvaraya
College of Engineering & Technology
M.P.Patelguda (V), Ibrahimpanam (M), Ranga Reddy Dist- 501510

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:

Name: **AMIREDDY VINUTHNA**

Roll No.: **18BT1A0501**

ACKNOWLEDGEMENT

It gives me a great sense of pleasure to present the report of the project undertaken during B.Tech. I would like to express my special thanks to the **Principal & Professor** (ME) **Dr.D Ramesh** for moral support and **college Management** of Visvesvaraya College of Engineering & Technology, Hyderabad for providing me infrastructure to complete the project.

I owe special debt of gratitude to **Dr. Raji Reddy, Dean Academics** of Visvesvaraya College of Engineering & Technology, Hyderabad for his constant support and guidance throughout the course of our work.

I thank **Dr. L. Kiran Kumar Reddy Head of the Department of Computer Science & Engineering** for his constant support and cooperation.

I take this opportunity to acknowledge the contribution of Project Coordinator **Mr. V. Hirish Reddy,** Assistant Professor, Department of Computer Science & Engineering, for his full support and assistance during the development of the project.

I also convey thanks to my internal project Guide **Ms. Syed Thaisin,** Assistant Professor (CSE) Visvesvaraya College of Engineering & Technology, Hyderabad for her guidance throughout the course of work. It is only her cognizant efforts that my endeavors have seen light of the day.

I also like to express my gratitude towards my **Parents** for their kind co-operation and constant encouragement which helped me in completing this project successfully.

I do not want to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of this project. Last but not the least, I acknowledge my **friends** for their contribution in the completion of the project.

AMIREDDY VINUTHNA

Roll No: **18BT1A0501**

INDEX

LIST OF CONTENTS

LIST OF FIGURES

ABSTRACT

Airline ticket prices can be somewhat hard to guess, as airline prices fluctuate constantly. So purchasing at different times could mean large differences in price. From the customer side, models are used to predict optimal time to buy a ticket and to predict the minimum ticket price.

Someone who purchases flight tickets frequently would be able to predict the right time to procure a ticket to obtain the best deal, as many airlines change ticket prices for their revenue management.

The airline may increase the prices when the demand is to be expected to increase the capacity. To estimate the minimum airfare, data for a specific air route has been collected including the features like departure time, arrival time and airways over a specific period.

Features are extracted from the collected data to apply Machine Learning (ML) models. This project gives the machine learning regression methods to predict the prices at the given time.

We use Linear Regression, KNeighbours Regressor, Random Forest Regressor and Decision Tree Regressor algorithms and check the accuracy of each algorithm and use the best accuracy algorithm to predict the price of the ticket andwe also use Flask to create a web interface where departure time, arrival time, source, destination, no.of stops and airways are choosen by the user and the predicted price is givenby the model.

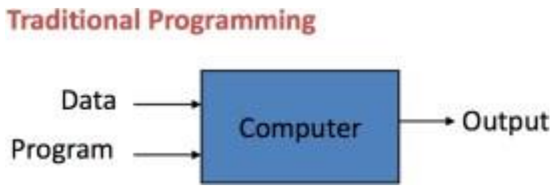
1. INTRODUCTION

***What is Machine Learning?**

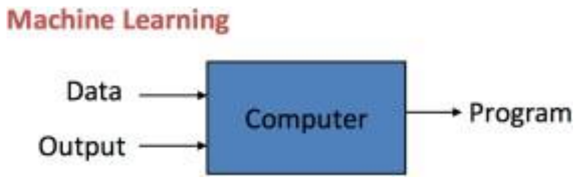
Machine Learning is a subfield of Artificial Intelligence(AI) and Computer Science that provides system the ability to automatically learn and improve from experience without being explicitly programmed. Machine Learning focuses on the development of computer programs that can access data and use it to learn for themselves.

***Traditional Programming vs Machine Learning:**

In Traditional Programming, data and the program are run on the computer to produce the output.



In Machine Learning, data and output are run on the computer to create a program. This program can be used in traditional programming.



The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in future based on examples we provide. The primary goal is to allow computers learn automatically without human intervention and adjust actions accordingly.

*****Every machine learning algorithm has three components:

→**Representation:** Representation of knowledge. Examples include decision trees, sets of rules, instances, graphical models, neural networks, support vector machines, model ensembles and others.

→**Evaluation:** The way to evaluate candidate programs (hypotheses). Examples include accuracy, prediction and recall, squared error, likelihood, posterior probability, cost, margin, entropy k-L divergence and others.

→**Optimization:** The way candidate programs are generated is known as the search process. For example, combinatorial optimization, convex optimization, constrained optimization.

*** Machine Learning Methods:**

Machine Learning algorithms are often categorized as:

→Supervised Learning
→Unsupervised Learning
→Semi-Supervised Learning
→Reinforcement Learning

• Supervised machine learning algorithms can apply what has been learned in the past to new data using labelled examples to predict future events.

• Unsupervised machine learning algorithms are used when the information used to train is neither classified nor labelled.

• Semi-Supervised learning algorithms has only few desired output labels.

• Reinforcement machine learning algorithm is a learning method that interacts with its environment by producing actions and discovers the errors or rewards.

1.1 ABOUT PROJECT :

The Airline Companies are considered as one of the most enlightened industries using complex methods and complex strategies to allocate airline prices in a dynamic fashion. These industries are trying to keep their all inclusive revenue as high as possible and boost their profit.

For that number of strategies are taken under consideration several financial, commercial, marketing and social factors closely connected with the ultimate airline prices. On account of the high intricacy of the pricing models applied by the airlines, it’s very complicated for a customer to buy an air ticket at the least price, the formulation of ticket bought in advance is low priced, it is possible that customers who secure a ticket earlier pay more than those who bought the same ticket later.

The ticket price may be influenced by a number of components, since the worth changes dynamically. Dynamic pricing allows more absolute forecasting of ticket prices on sparkling factors such as change in demand and price discrimination.
However dynamic pricing is challenging as it is highly dominated by various factors including internal and external factors, competition among airlines and strategic customers. The last two decades have seen a regular increase in research targeting both customers and airlines. From the customer point of view, establishing the lowest price or the best time to buy a ticket is the key issue and for the airline side increasing the company's revenue.

The features used for the analysis are: Airline, Date_of_journey, Source, Destination, Route, Departure_time, Arrival_time, Duration, Total_stops, Additional_information, Price.

EXISTING SYSTEM DRAWBACKS :

Existing system which predicts the price of airlines is considered to be a bit bad due to the hardness inovled to collect the data and do the analysis it would take a long duration. And by just performing the analysis by using the maths is a bit complex. Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travelers saying that flight ticket prices are so unpredictable.

Training data is combination of both categorical and numerical also we can see some special character also being used because of which we have to do data Transformation on it before applying it to our model .

***Drawbacks:**

- It is hard to estimate the price of the fight manually because people don't have entire knowledge about the distance cost per Km.
- And even the cost of the fight depends based on the vacations every thing need to be taken under consideration for calculating the price which is quite tricky for a human to calculate.

PROPOSED SYSYTEM WITH FEATURES :

For building the model we used four different machine learning algorithms and see which algorithm gives us more accuracy among all algorithms used. Based on the accuracy we will implement that algorithm for creating the model of Airline price prediction. Here we are provided with prices of flight tickets for various airlines between various cities.

***Features:**

- Helps in predicting the flights rate earlier.
- Accordingly, people can plan based on the tickets.
- Will make easy in calculations.

2. SYSTEM ANALYSIS

SYSTEM REQUIREMENTS

Hardware and Software Requirements:

Hardware Requirements:

The selection of hardware is very important in the existence and proper working of any software. The hardware used for the development of the project is:

- **RAM:** RAM 2GB or above
- **Processor:** Processor i3 or above
- **Speed:** Processor speed 2.4GHz
- **Hard Disk:** 500GB
- **I/O:** Keyboard, Mouse, Monitor.

Software Requirements:

The software used for the development of the project is:

- **Programming Language:** Python 3.7
- **Tools:** Anaconda
- **Operating system:** Windows 7 or above
- **IDE:** Jupyter notebook

*ANACONDA:

Anaconda is a distribution of Python and R programming languages for scientific computing, that aims to simply package management and deployment.

*Installation of anaconda on Windows:

1. Download the Anaconda installer.

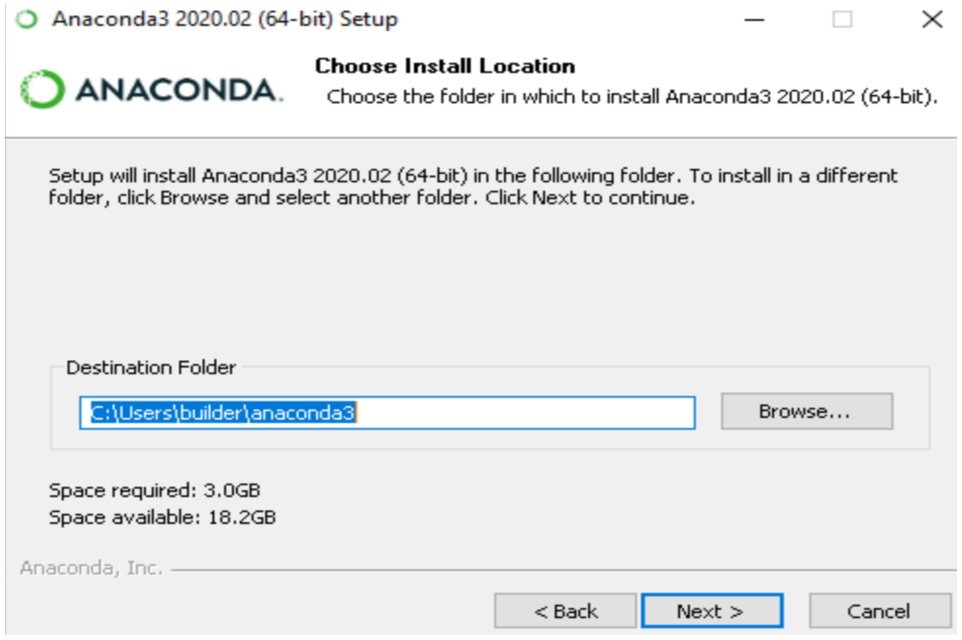
2. Double click the installer to launch.

3. Click Next.

4. Read the licensing terms and click "I Agree".

5. Select an install for "Just Me" unless you're installing for all users (which requires Windows Administrator privileges) and click Next.

6. Select a destination folder to install Anaconda and click the Next button.



Choose Install Location

Choose the folder in which to install Anaconda3 2020.02 (64-bit).

Setup will install Anaconda3 2020.02 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

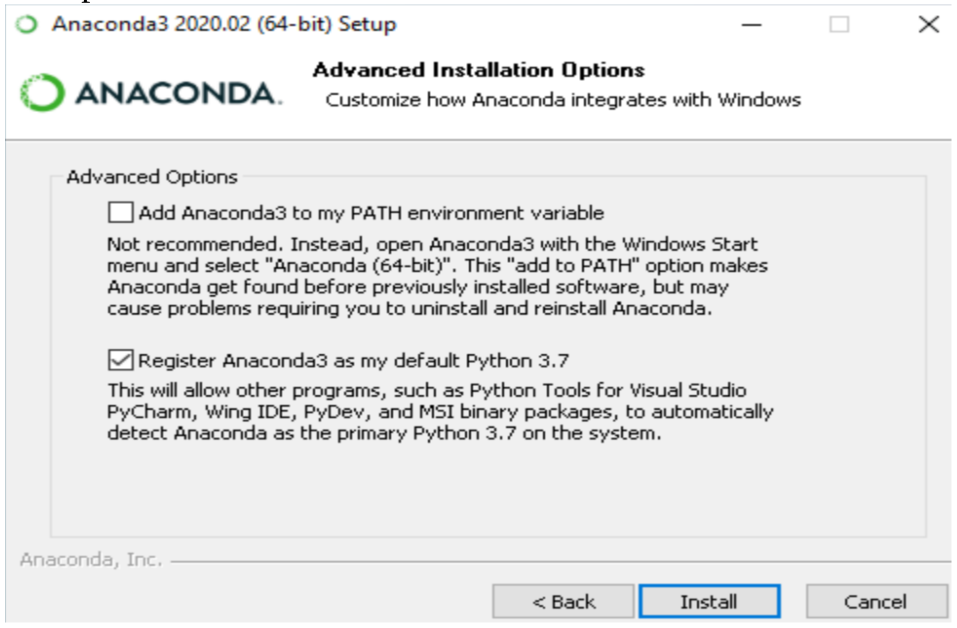
Destination Folder

C:\Users\builder\anaconda3

Space required: 3.0GB
Space available: 18.2GB

Anaconda, Inc.

7. Choose whether to add Anaconda to your PATH environment variable. We recommend not adding Anaconda to the PATH environment variable, since this can interfere with other software. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.



Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☐ Add Anaconda3 to my PATH environment variable

Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda3 as my default Python 3.7

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.7 on the system.

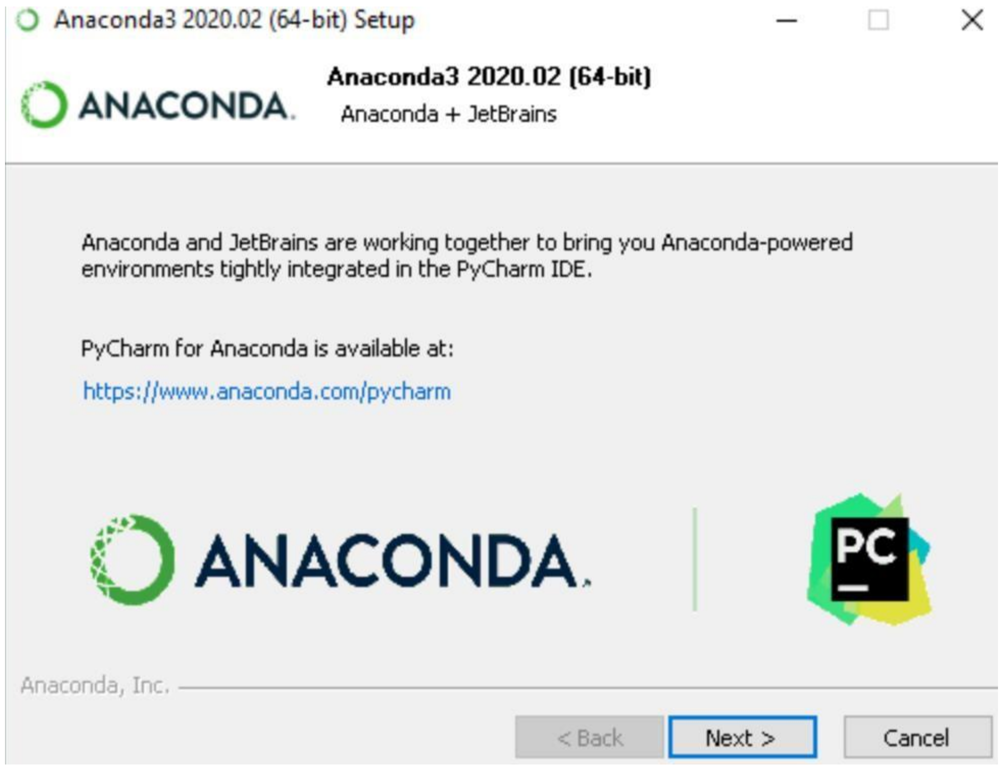
Anaconda, Inc.

8. Choose whether to register Anaconda as your default Python. Unless you plan on installing and running multiple versions of Anaconda or multiple versions of Python, accept the default and leave this box checked.

9. Click the Install button. If you want to watch the packages Anaconda is installing, click Show Details.

10. Click the Next button.

11. Optional: To install PyCharm for Anaconda, click on the link to https://www.anaconda.com/pycharm.


Anaconda3 2020.02 (64-bit) Setup

Anaconda3 2020.02 (64-bit)
Anaconda + JetBrains

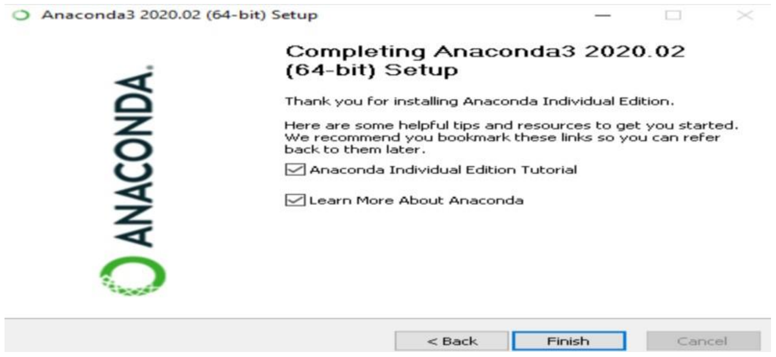
Anaconda and JetBrains are working together to bring you Anaconda-powered environments tightly integrated in the PyCharm IDE.

PyCharm for Anaconda is available at:

https://www.anaconda.com/pycharm

Or to install Anaconda without PyCharm, click the Next button

12. After a successful installation you will see the "Thanks for installing Anaconda" dialog


Anaconda3 2020.02 (64-bit) Setup

Completing Anaconda3 2020.02 (64-bit) Setup

Thank you for installing Anaconda Individual Edition.

Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

☑ Anaconda Individual Edition Tutorial

☑ Learn More About Anaconda

box.

Functional Requirements:

The following are the functional requirements of our project:

- A training dataset has to be created on which training is performed.
- A testing dataset has to be created on which testing is performed.

Non-Functional Requirements:

The Non-Functional Requirements may include:

- **Maintainability:** It is used to make future maintenance easier, meet new requirements.
- **Robustness:** It is the quality of being able to withstand stress, pressures or changes in procedure or circumstance.
- **Size:** Size of application plays a major role, if the size is less then efficiency will be high.
- **Speed:** If speed is high then it is good. Since the no.of lines of code is less, hence speed is high.

MODULE DESCRIPTION:

The project contains the following main modules:

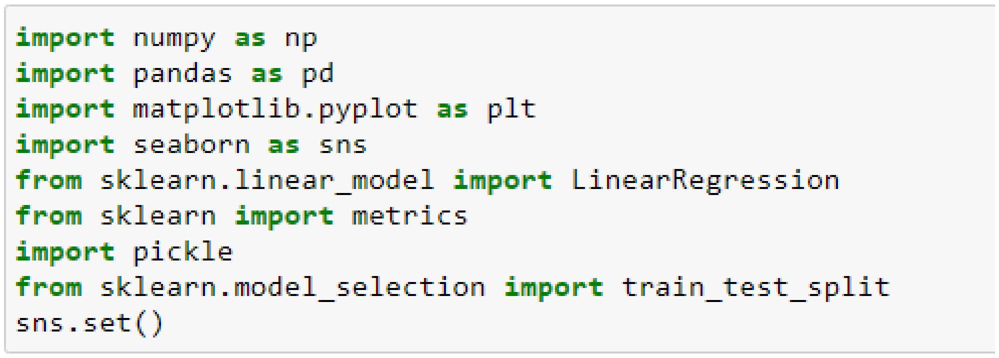
- Data Collection
- Cleaning and preparing data
- Analyzing data
- Applying the algorithms
- Prediction of Output

Data Collection:

The python-script take out the data from the site, and provides output as a CSV record. The document contains the data with features and its details. A significant perspective is to choose the features required for calculation of expected flight price. Output gathered from the site contains number of parameters for each flight: yet not all are required, so just the accompanying components are,

- Date of journey
- Time of Departure
- Place of Departure
- Time of Arrival
- Place of Destination/Arrival
- Airway company
- Total Fare

In this investigation, the attention is just to limit the airfare considering a single route. This information is gathered for perhaps the busiest course in India (BOM to DEL) over a time of a quarter of a year that is from February to April. For each flight information each feature is collected physically.



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import pickle
from sklearn.model_selection import train_test_split
sns.set()
```

We use these Python in-built libraries for Data Pre-Processing, Exploratory Data Analysis(EDA) , Data Visualization and to train our model.

Cleaning and preparing data:

All the gathered information required a great deal of work, so after the accumulation of information, it should have been perfect and be ready as indicated by the model prerequisites.

All the superfluous information is deleted like copies and invalid qualities. In all AI this innovation, is the most significant and time consuming. Different statistical methods and logics in python clean and set up the information. For instance, the cost was character type, not a number

- **Data cleaning** is the process of **preparing data for analysis** by removing or modifying data that is incorrect, incomplete, irrelevant, or duplicated. Having bad quality data can be disastrous to your processes and **analysis**. Poor data quality leads to poorer results; thus, it is important to understand what is **data cleaning.**

Analyzing the Data(EDA):

Preparation of data is trailed by breaking down the information, revealing the concealed patterns and afterward applying different AI models. Likewise, a few features can be determined from the current features. Flight days can be issued by computing the difference of the flight date and the date on which information is collected. This can be observed for 45 days. Additionally, flight date is important, whether it is on festive day or a weekday or weekend. Instinctively the flights planned during weekends cost more than the flights on weekdays. Additionally, time plays important role. So the time is considered in classes as: Morning, evening and night**.**

Applying the Algorithms:

For predicting the flight ticket prices, many algorithms are introduced in machine learning. The algorithms that we are using are Linear Regression, KNeighbors Regressor, Random Forest Regressor and Decision Tree Regressor.

- **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 being used.

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.

The equation for linear regression is y(pred) = b0+b1 ∗ x

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 b0 and b1 values.
b0: intercept
b1: coefficient of x
Once we find the best b0 and b1 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.

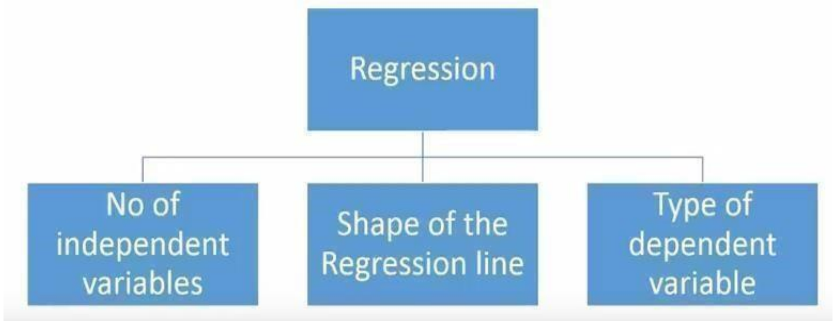

Regression
No of independent variables
Shape of the Regression line
Type of dependent variable

Figure 2.4.1 Regression Structure

➢ **KNeighbors Regressor:**

Regression based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set. KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses ‘**feature similarity**’ to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

Below is a stepwise explanation of the algorithm:

1. First, the distance between the new point and each training point is calculated.

2. The closest k data points are selected (based on the distance).

3. The average of these data points is the final prediction for the new point.

The first step is to calculate the distance between the new point and each training point. There are various methods for calculating this distance, of which the most commonly known methods are – Euclidian, Manhattan (for continuous) and Hamming distance (for categorical).

1. **Euclidean Distance:** Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (y).

Euclidean :

$$d(x, y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city - block :

$$d(x, y) = \sum_{i=1}^{m} |x_i - y_i|$$

2. **Manhattan Distance**: This is the distance between real vectors using the sum of their absolute difference.
3. **Hamming Distance**: It is used for categorical variables. If the value (x) and the value (y) are the same, the distance D will be equal to 0 . Otherwise D=1.

Hamming Distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

Once the distance of a new observation from the points in our training set has been measured, the next step is to pick the closest points. The number of points to be considered is defined by the value of k.

The **second step** is to select the k value. This determines the number of neighbors we look at when we assign a value to any new observation. Lastly, workit on the data set.

- **Random Forest Regressor:**

  A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

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.

➢ **Decision Tree Regressor:**

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

The branches/edges represent the result of the node and the nodes have either:

1.Conditions [Decision Nodes]

2.Result [End Nodes]

The branches/edges represent the truth/falsity of the statement and take makes a decision based on that.

Decision Tree Regression:
Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

Discrete output example: A weather prediction model that predicts whether or not there'll be rain on a particular day.
Continuous output example: A profit prediction model that states the probable profit that can be generated from the sale of a product.
Here, continuous values are predicted with the help of a decision tree regression model.

Prediction of Output:

Output can be predicted by using Machine Learning algorithms. Data is collected and stored in NoSQL / SQL format.

That data is divided into two parts:

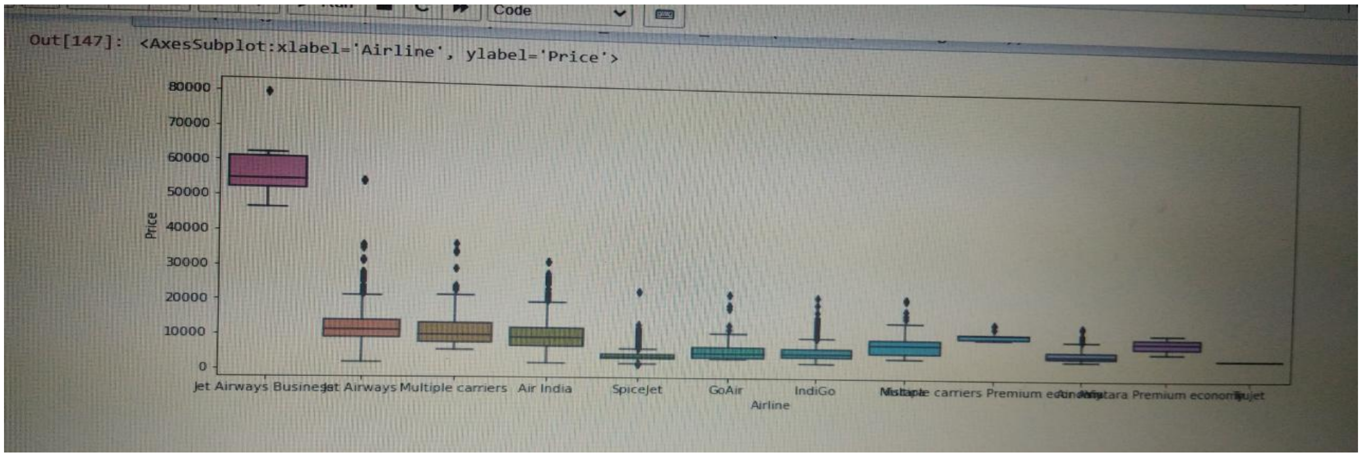
i) Training data

ii) Testing data.

- Training data is used for training the model and then that model is tested using testing data. After this, the trained model is used for predicting price of airlines ticket for given feature set.

Here, as Random Forest Regressor has the best accuracy with low RMSE value when compared with other Algorithms, we used Random Forest Regressor to predict the price of airline tickets.

Data Visualization: Data visualization is the presentation of data or information in a graph, chart or other visual format. It communicates relationships of data with images.



```
Out[147]: <AxesSubplot:xlabel='Airline', ylabel='Price'>
```

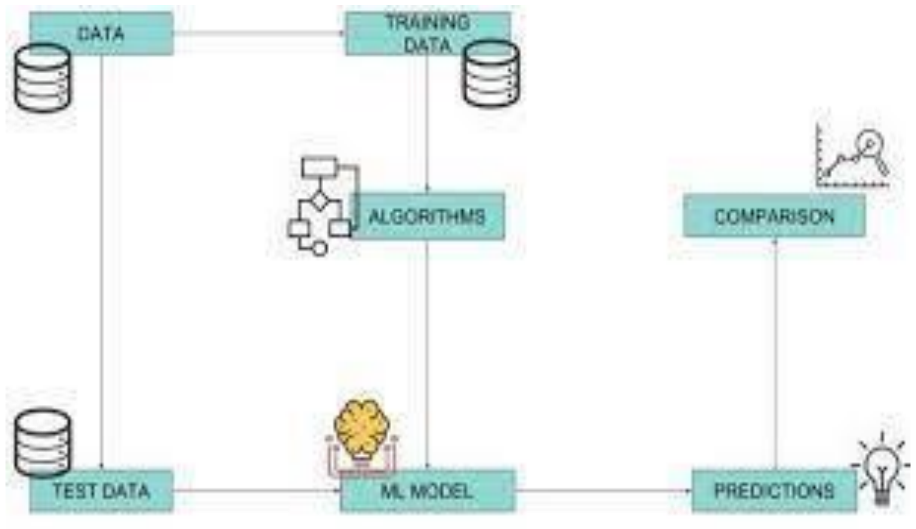
Here, we have visualized the predicted price by our model.

3. DESIGN

System design is the process of defining the architecture, modules, interfaces and data fora system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product management.

Block Diagram:

The block diagram typically used for a high level, less detailed description aimed more at understanding thr overall concepts and less at understanding the details of implementation.



Data Flow Diagrams:

Data flow diagram (DFD) is a graphical representation of "flow" of data through an information system, modelling its process concepts. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFD's can also be used for the visualization of data processing (structured design).

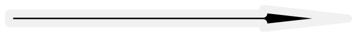
A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It doesn't show information about timing of processes, or information about whether processes will operate in sequence or parallel. A DFD is also called as "bubble chart".

DFD Symbols:

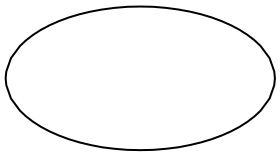
In the DFD, there are four symbols:

- A square defines a source or destination of system data.
- An arrow indicates dataflow. It is the pipeline through which the information flows.
- A circle or a bubble represents transforms dataflow into outgoing dataflow.
- An open rectangle is a store, data at reset or at temporary repository of data.

Dataflow: Data move in a specific direction from an origin to a destination.

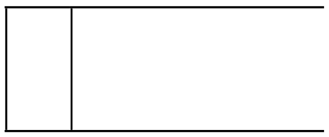


Process: People, procedures or devices that use or produce (Transform) data. The physical component is not identified.



Sources: External sources or destination of data, which may be programs, organizations or other entity.

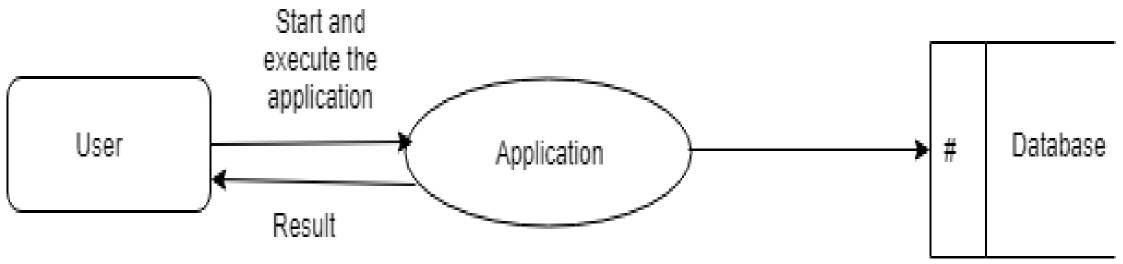
Data store: Here data is stored or referenced by a process in the system's.



In our project, we had built the data flow diagrams at the very beginning of business process modelling in order to model the functions that our project has to carry out and the interaction between those functions together with focusing on dataexchanges betweenprocesses.

<u>Context level DFD</u>:

A Context level Data flow diagram created using select structured systems analysis and design method (SSADM). This level shows the overall contextof the system and its operating environment and shows the whole system as just oneprocess. It does not usually show data stores, unless they are “owned” by external systems, e.g., are accessed by but not maintained by this system, however, these areoften shown as external entities. The Context level DFD is shown in fig.3.2.1

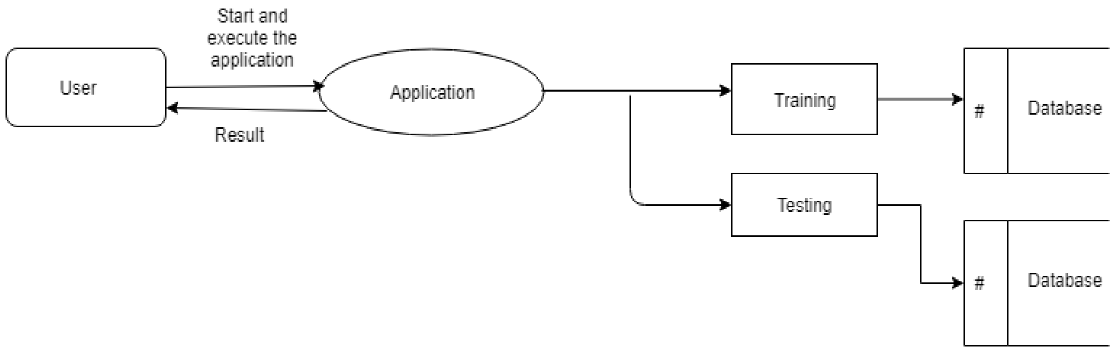


Context Level DFD for Airlines Ticket Price Prediction

The Context Level Data Flow Diagram shows the data flow from the application to the database and to the system.

<u>Top level DFD</u>:

A data flow diagram is that which can be used to indicate the clear progressof a business venture. In the process of coming up with a data flow diagram, the level one provides an overview of the major functional areas of the undertaking. After presenting the values for most Important fields of discussion, it gives room for



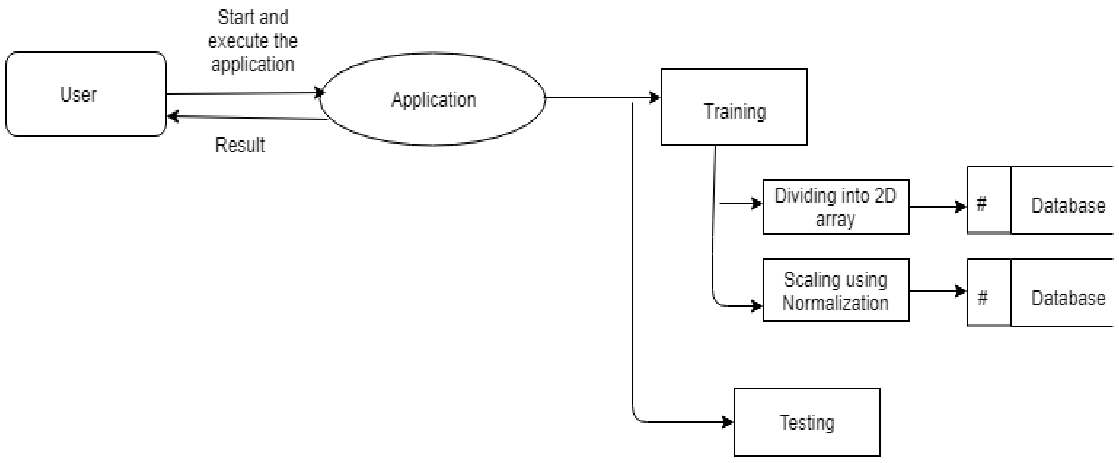
level two to be drawn.

After starting and executing the application, training and testing the datasetcan be done as shown in the above figure.

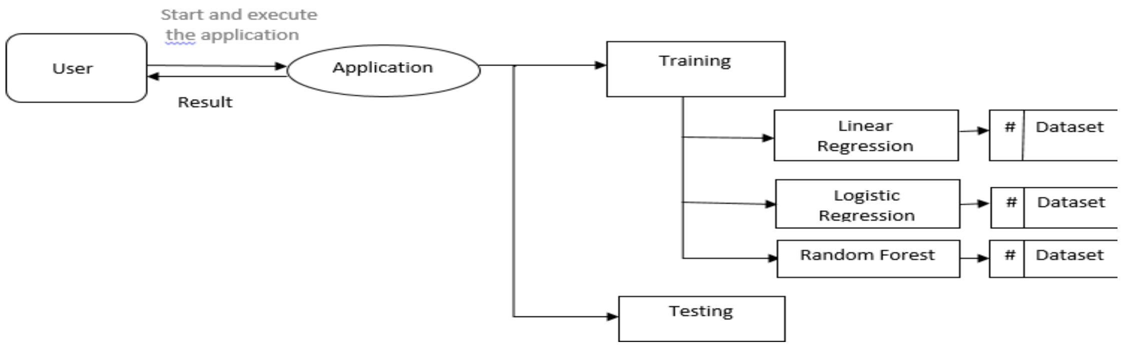
Detailed Level DFD:

This level explains each process of the system in a detailed manner. In firstdetailed level DFD (Generation of individual fields): how data flows through individual process/fields in it are shown.

In second detailed level DFD (generation of detailed process of the individual field show data flows through the system to form a detailed description of the individual processes.



After starting and executing the application, training the dataset is done by using dividing into 2D array and scaling using normalization algorithms, and then testing is done.



After starting and executing the application, training the dataset is done byusing linear regression and then testing is done.

UNIFIED MODELLING LANGUAGE DIAGRAMS:

The Unified Modelling Language (UML) is a Standard language for specifying, visualizing, constructing and documenting the software system and its components. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

- **User Model View**

i. This view represents the system from the user's perspective.
ii. The analysis representation describes a usage scenario from the end-user's perspective.

- **Structural Model View**

i. In this model the data and functionality are arrived from inside the system.
ii. This model view models the static structures.

- **Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- **Implementation model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

- **Environmental Model View**

In this the structural and behavioral aspect of the environment in which the system is to be implemented are represented.

Use Case Diagram:

Use case diagrams are one of the five diagrams in the UML for modelling the dynamic aspects of the systems (activity diagrams, sequence diagram, state chart diagram, collaboration diagram are the four other kinds of diagrams in the UML for modelling the dynamic aspects of systems).Use case diagrams are central to modelling the behavior of the system, a sub-system, or a class. Each one shows a set of use cases and actors and relations.

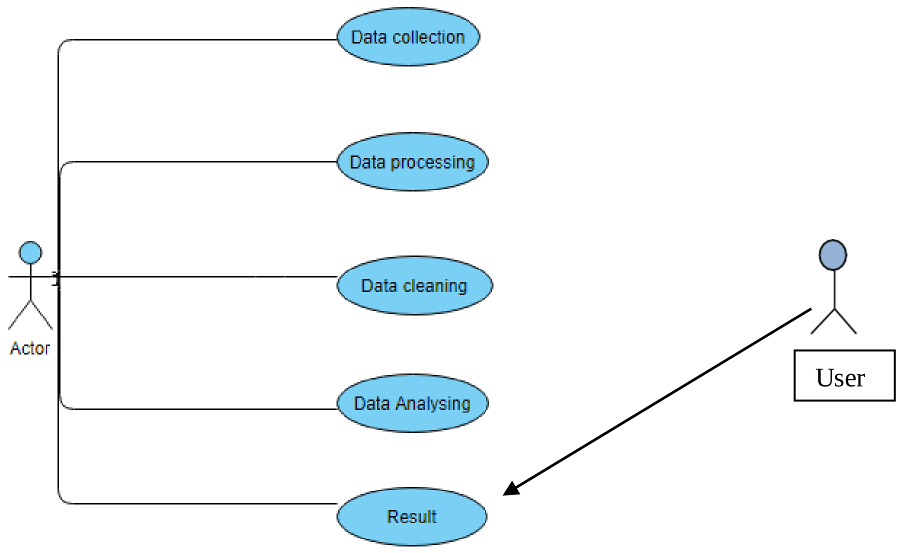


Figure 3.3.1 Use Case Diagram

Purpose of Use case diagram:

Use case diagrams are used to gather the requirements of a system including internal andexternal influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

Contents:

- Functionalities to be represented as use case
- Actors
- Among the use Relationships cases and actors.

Sequence Diagram:

Sequence diagrams are a popular dynamic modelling solution. Dynamic modelling focuses on the interactions occurring within the system. Sequence diagrams specifically focus on the "lifelines" of an object and how they communicate with other objects to perform a function before the lifeline ends.

Purpose of sequence diagram:

Sequence diagram were made for developers, the truth is that a company's business staff could use such diagrams to communicate how exactly the business presently currently works by illustrating how the different business objects interact.

Content:

- Messages
- Lifelines
- Guards
- Actor
- Objects

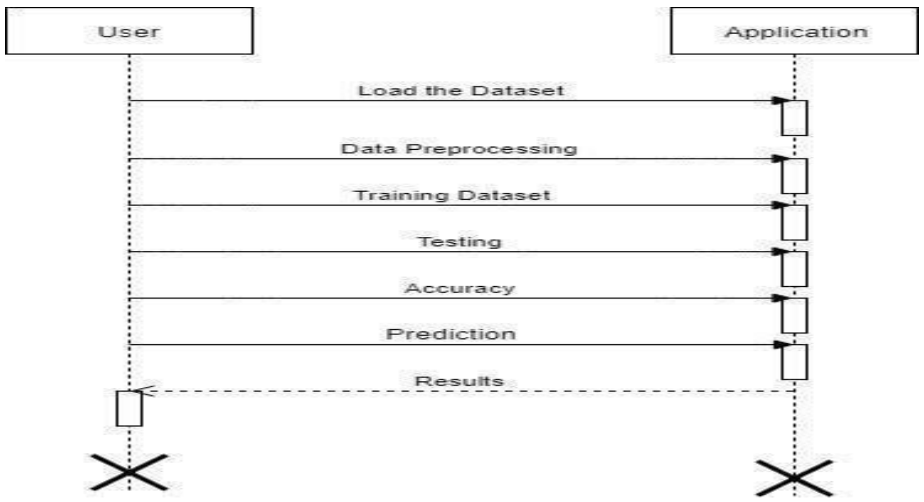


Figure 3.3.2 Sequence Diagram

Collaboration Diagram:

In the collaboration diagram, the method call sequence is indicated by some numbering technique. The number indicates how the methods are called one after another. A collaboration diagram resembles a flow chart that portrays the roles, functionality and behaviour of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencies.

Purpose of collaboration diagram:

Allocate functionality to classes by exploring the behavioral aspects of a system. Model the logic of the implementation of a complex operation, particularly one that interacts with a large number of other objects. Explore the roles that objects take within a system, as well as the different relationships they are involved with when in those roles.

Content:

- Messages
- Link

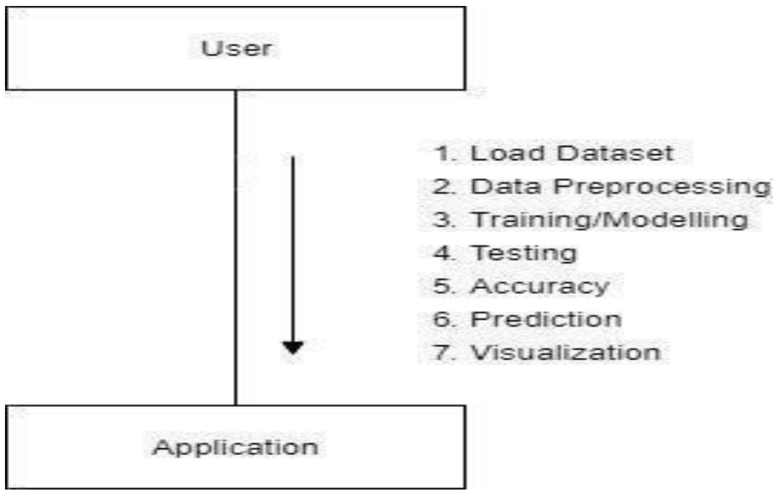


Figure 3.3.3 Collaboration Diagram

Activity Diagram:

An Activity diagram shows the flow from activity to activity within a system it emphasizes the flow of control among objects**.** It is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to representthe flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. Thisflow can be sequential, branched, or concurrent. Activity diagrams deal with all typeof flow control by using different elements such as fork, join, etc.

Purpose of Activity diagram:

The basic purpose of activity diagrams is similar to other diagrams. It captures the dynamic behavior of the system. Other diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Contents:

- Activities
- Association
- Conditions
- Constraints

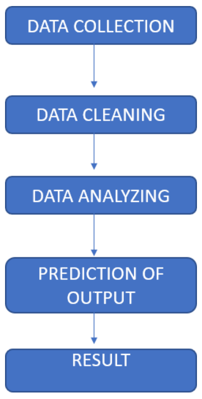


Figure 3.3.4 Activity Diagram

4. IMPLEMENTATION

Implementation is the process of defining how the system should be built, ensuring that the system is operational and used, ensuring that the system meets quality standard. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Technology Used:

Python: Python is a popular programming language. It was created by Guido Van Rossum, and released in 1991.
It is used for: Web development (server-side), Software development, Mathematics, System scripting.

Why Python?

- Python is a scripting language like PHP, Perl, and Ruby.
- No licensing, distribution, or development fees
- It is a Desktop application.
- Linux, windows
- Excellent documentation
- Thriving developer community

What can python do?

1. Python can be used on a server to create web applications.
2. Python can be used alongside to create workflows.
3. Python can connect to database systems. It can also read and modify files.
4. Python can be used to handle big data and perform complex mathematics.
5. Python can be used for rapid prototyping, or for production-ready software development.

Python Modules:

A module allows you to logically organize your python code. Grouping related code into a module makes the code easier to understand and use. A module is a python object with arbitarily named attributes that you can bind and reference.

Libraries Of python:

Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited too many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and unit testing.

As of March 2018, the Python Package Index (PyPI), the official repository for third party Python software, contains over 130,000 packages with a wide range of functionality, including:

- Graphical user interfaces
- Web frameworks
- Multimedia
- Databases
- Networking
- Test frameworks
- Automation
- Web scraping
- Documentation
- System Administration

Pandas:

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive.
It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

Pandas is well suited for many different kinds of data: Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spread sheet. Ordered and unordered (not necessarily fixed-frequency) time series data.
Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels. Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a panda’s data structure.

The two primary data structures of pandas, Series (1-dimensional) and Data Frame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, Data Frame provides everything that R’s data frame provides and much more.

Pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Few of the things that pandas does well:

1. Easy handling of missing data (represented as Nan) in floating point as well as non-floating-point data.
2. Size mutability: columns can be inserted and deleted from Data Frame and higher dimensional objects.
3. Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, Data Frame, etc. automatically align the data for you in computations.
4. Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data.
5. Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into Data Frame objects.
6. Intelligent label-based slicing, fancy indexing, and sub setting of large data sets.
7. Intuitive merging and joining data sets
8. Flexible reshaping and pivoting of data sets.
9. Hierarchical labeling of axes (possible to have multiple labels per tick).

10. Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format.
11. Time series-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.
12. Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. Pandas is the ideal tool for all of these tasks.

13. Pandas is fast. Many of the low-level algorithmic bits have been extensively improved in Python code. However, as with anything else generalization usually sacrifices performance.
So, if you focus on one feature for your application you may be able to create a faster specialized tool.
14. Pandas are a dependency of stats models, making it an important part of the statistical computing ecosystem in Python .
15. Pandas have been used extensively in production in financial applications.

NumPY:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array
objects and a collection of routines for processing those arrays. Using NumPy , mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

NumPy is a Python package. It stands for 'Numerical Python. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. Numeric, the ancestor of NumPy, was developed by Jim Humulin. Another package. NumPy was also developed, having some additional functionality. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numara into Numeric package.

Operations using NumPy :

Using NumPy, a developer can perform the following operations
−Fourier transforms and routines for shape manipulation, Mathematical and logical

operations on arrays, Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Sklearn:

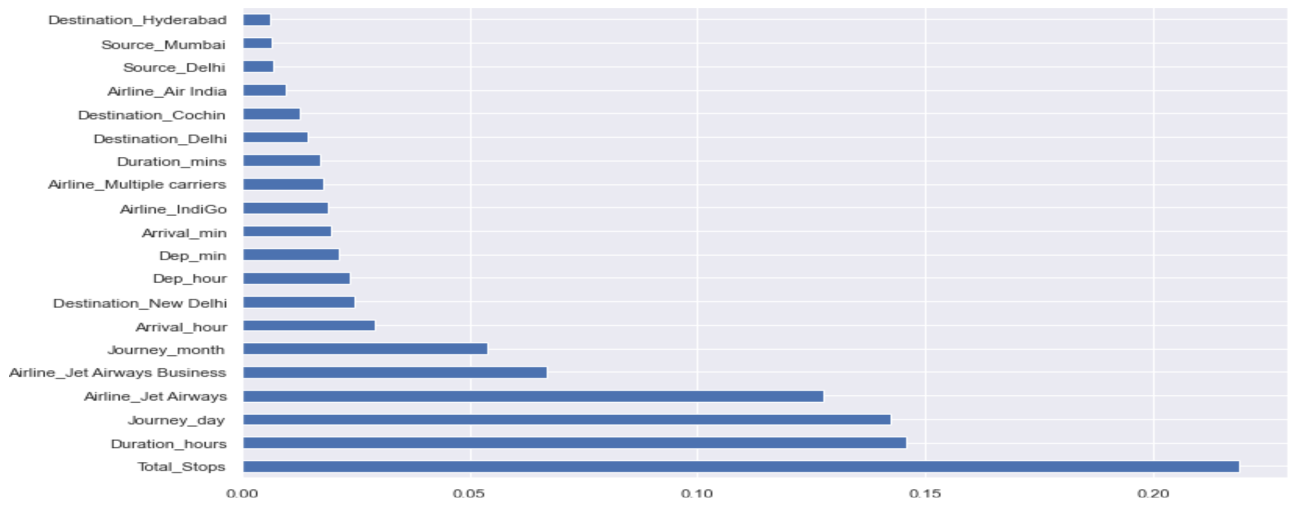
Sklearn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Some popular groups of models provided by sklearn include:

Ensemble methods: For combining the predictions of multiple supervised models.
Feature extraction: For defining attributes in image and text data.
Feature selection: For identifying meaningful attributes from which to create supervised models.
Parameter Tuning: For getting the most out of supervised models.
Manifold Learning: For summarizing and depicting complex multi-dimensional data.
Supervised Models: a vast array not limited to generalize linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.

Matplotlib:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB. Matplotlib was originally written by John D. Hunter in 2003. The current stable version is 2.2.0 released in January 2018.

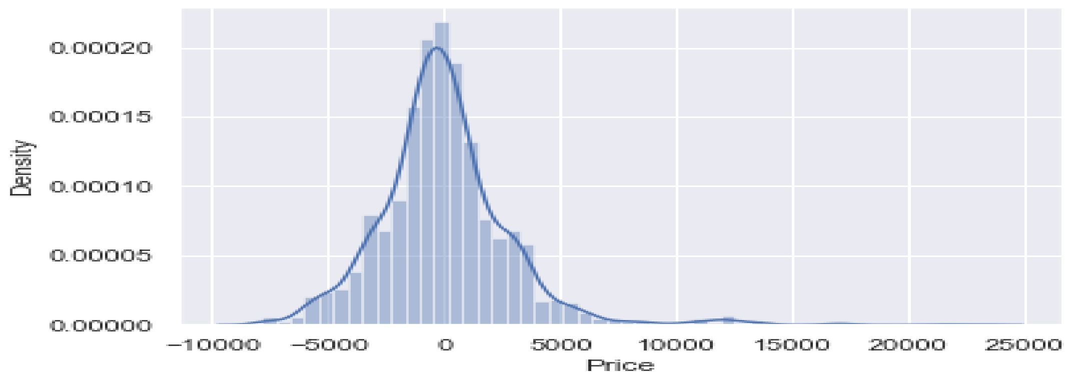


Seaborn:

Seaborn is a library mostly used for statistical plotting in Python. It is built on top of Matplotlib and provides beautiful default styles and color palettes to make statistical plots more attractive.

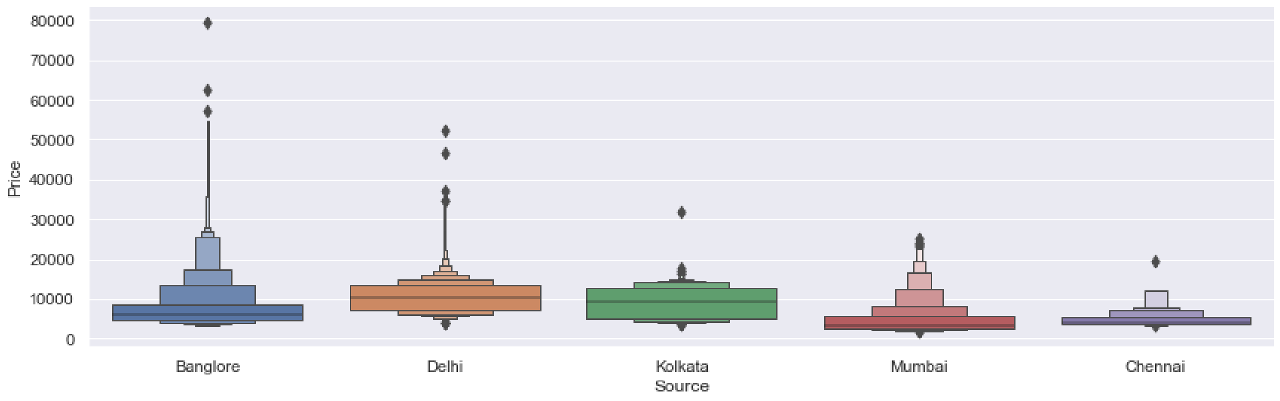
Seaborn is built on top of Python's core visualization library Matplotlib. It is meant to serve as a complement, and not a replacement. However, Seaborn comes with some very important features. Let us see a few of them here. The features help in –

- Built in themes for styling matplotlib graphics
- Visualizing univariate and bivariate data
- Fitting in and visualizing linear regression models
- Plotting statistical time series data
- Seaborn works well with NumPy and Pandas data structures
- It comes with built in themes for styling Matplotlib graphics



Seaborn catplot:

The default representation of the data in catplot() uses a scatterplot. There are actually two different categorical scatter plots in seaborn. They take different approaches to resolving the main challenge in representing categorical data with a scatter plot, which is that all of the points belonging to one category would fall on the same position along the axis corresponding to the categorical variable. The approach used by stripplot(), which is the default “kind” in catplot() is to adjust the positions of points on the categorical axis with a small amount of random “jitter”.The jitter parameter controls the magnitude of jitter or disables it altogether.



Seaborn Heatmap :

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred.

Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the seaborn.heatmap() function.

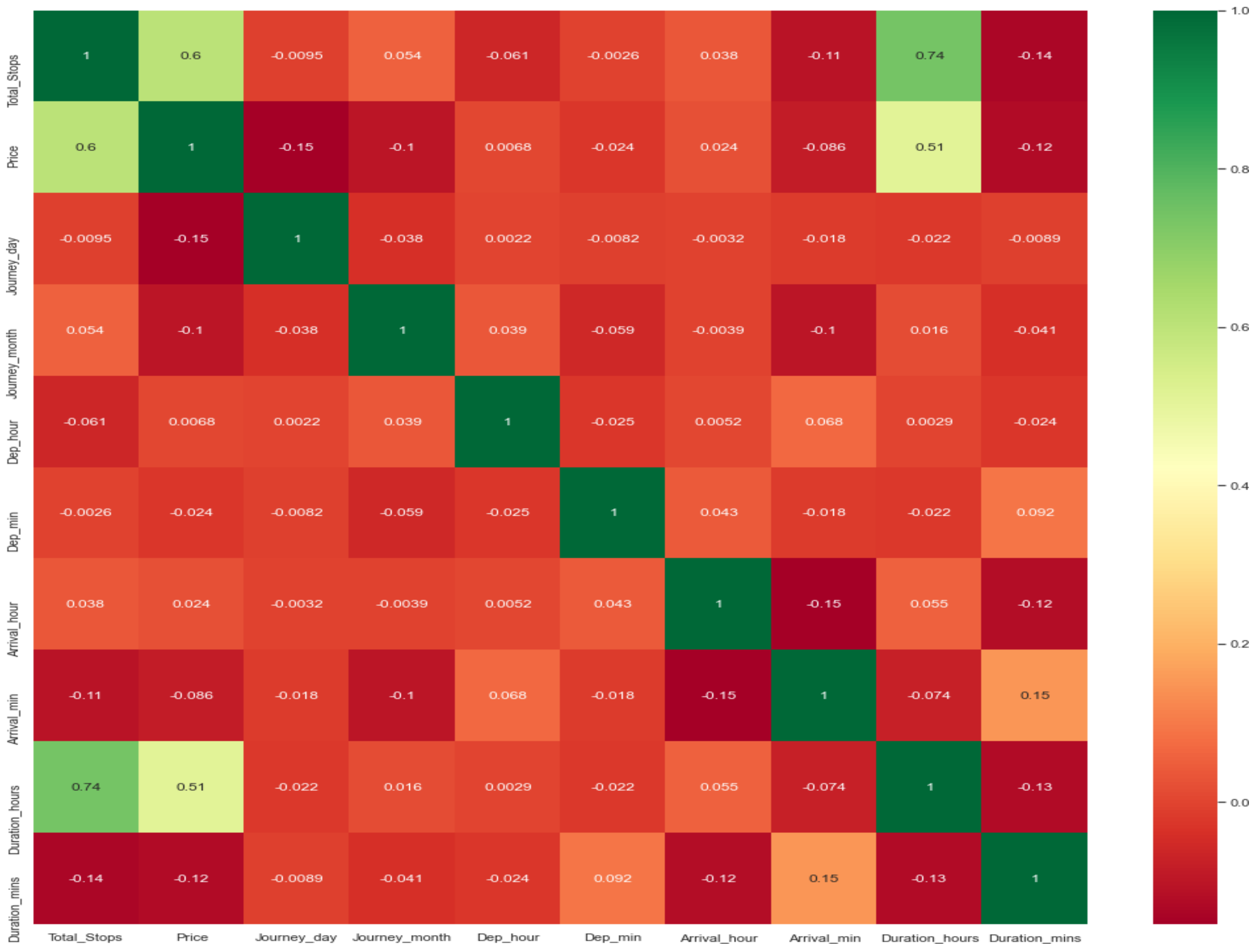
Syntax: seaborn.heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None, annot_kws=None, linewidths=0, linecolor=’white’, cbar=True, **kwargs)

Important Parameters:

- **data:** 2D dataset that can be coerced into an ndarray.
- **vmin**, **vmax:** Values to anchor the colormap, otherwise they are inferred from the data and other keyword arguments.

- **cmap:** The mapping from data values to color space.
- **center:** The value at which to center the colormap when plotting divergent data.
- **annot:** If True, write the data value in each cell.
- **fmt:** String formatting code to use when adding annotations.
- **linewidths:** Width of the lines that will divide each cell.
- **linecolor:** Color of the lines that will divide each cell.
- **cbar:** Whether to draw a colorbar.

All the parameters except data are optional.
Returns: An object of type matplotlib.axes._subplots.AxesSubplot



Deployment Using Flask:

Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier.Lets us focus on what the users are requesting and what sort of response to give back**.** The Flask Framework looks for HTML files in a folder called **templates.** You need to create a templates folder and put all your HTML files in there. Always keep the main.py outside of your templates folder.

home.html contains the HTML code for the drop down lists where the user can select the Arrival_Datetime, destination_Datetime, source, destination,total stops details.

We are importing the Flask module and creating a Flask web server from the Flask module. We are creating an instance of the Flask class and calling it app. we are creating a new web application. Save the file and use your browser to navigate to http://127.0.0.1:5000/ the browser will display the application.

5. CODING

Flight Price Prediction

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import pickle
from sklearn.model_selection import train_test_split
sns.set()
# flask

train_data = pd.read_excel(r"./Data_Train.xlsx")

pd.set_option('display.max_columns', None)

train_data.head()

train_data.info()

train_data["Duration"].value_counts()

train_data.dropna(inplace = True)

train_data.isnull().sum()
```

EDA

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

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

.dt.day method will extract only day of that date
.dt.month method will extract only month of that date

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

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

train_data.head()

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

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

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

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

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

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

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

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

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))   # Extracts only
minutes from duration
```

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

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

Handling Categorical Data

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

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

# Airline vs Price

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

plt.show()

# As Airline is Nominal Categorical data we will perform OneHotEncoding



Airline = train_data[["Airline"]]



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



Airline.head()

train_data["Source"].value_counts()

# Source vs Price



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

```
plt.show()

# As Source is Nominal Categorical data we will perform OneHotEncoding


Source = train_data[["Source"]]


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


Source.head()

train_data["Destination"].value_counts()

Destination = train_data[["Destination"]]


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


Destination.head()

train_data["Route"]

# Additional_Info contains almost 80% no_info

# Route and Total_Stops are related to each other


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

train_data["Total_Stops"].value_counts()

# As this is case of Ordinal Categorical type we perform LabelEncoder

# Here Values are assigned with corresponding keys
```

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

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


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

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

data_train.head()

data_train.shape
```

Test Set

```
test_data = pd.read_excel(r"./Test_set.xlsx")

# Preprocessing


print("Test data Info")

print("-"*75)

print(test_data.info())


print()

print()


print("Null values :")

print("-"*75)

test_data.dropna(inplace = True)
```

```
print(test_data.isnull().sum())


# EDA


# Date_of_Journey

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

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

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


# Dep_Time

test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour

test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute

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


# Arrival_Time

test_data["Arrival_hour"] = pd.to_datetime(test_data.Arrival_Time).dt.hour

test_data["Arrival_min"] = pd.to_datetime(test_data.Arrival_Time).dt.minute

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


# Duration
```

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


for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour


duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))   # Extracts only
minutes from duration


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

```
# Categorical data


print("Airline")

print("-"*75)

print(test_data["Airline"].value_counts())

Airline = pd.get_dummies(test_data["Airline"], drop_first= True)


print()


print("Source")

print("-"*75)

print(test_data["Source"].value_counts())

Source = pd.get_dummies(test_data["Source"], drop_first= True)


print()


print("Destination")

print("-"*75)

print(test_data["Destination"].value_counts())

Destination = pd.get_dummies(test_data["Destination"], drop_first = True)
```

```
# Additional_Info contains almost 80% no_info

# Route and Total_Stops are related to each other

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



# Replacing Total_Stops

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



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

data_test = pd.concat([test_data, Airline, Source, Destination], axis = 1)



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



print()

print()



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

Feature Selection

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

1. **heatmap**

2. **feature_importance_**
3. **SelectKBest**

```
data_train.shape

data_train.columns

X = data_train.loc[:, ['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour',

       'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',

       'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',

       'Airline_Jet Airways', 'Airline_Jet Airways Business',

       'Airline_Multiple carriers',

       'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',

       'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',

       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',

       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',

       'Destination_Kolkata', 'Destination_New Delhi']]

X.head()

y = data_train.iloc[:, 1]

y.head()

# Finds correlation between Independent and dependent attributes


plt.figure(figsize = (18,18))

sns.heatmap(train_data.corr(), annot = True, cmap = "RdYlGn")
```

```
plt.show()

# Important feature using ExtraTreesRegressor


from sklearn.ensemble import ExtraTreesRegressor

selection = ExtraTreesRegressor()

selection.fit(X, y)

print(selection.feature_importances_)

#plot graph of feature importances for better visualization


plt.figure(figsize = (12,8))

feat_importances = pd.Series(selection.feature_importances_, index=X.columns)

feat_importances.nlargest(20).plot(kind='barh')

plt.show()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =
42)

def predict(ml_model,dump):

    model=ml_model.fit(X_train,y_train)

    print('Training score : {}'.format(model.score(X_train,y_train)))

    y_prediction=model.predict(X_test)

    print('predictions are: \n {}'.format(y_prediction))
print('\n')
r2_score=metrics.r2_score(y_test,y_prediction)
print('r2 score: {}'.format(r2_score))
print('MAE:',metrics.mean_absolute_error(y_test,y_prediction))
print('MSE:',metrics.mean_squared_error(y_test,y_prediction))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_prediction)))
sns.distplot(y_test-y_prediction)


if dump==1:
    ##dump your model using pickle so that we will re-use
    file=open('./model.pkl','wb')
    pickle.dump(model,file)
    predict(LinearRegression(),0)
    predict(KNeighborsRegressor().0)
    predict(RandomForestRegressor(),0)
    predict(DecisionTreeRegressor(),0)
```

6. TESTING

It is the process of testing the functionality and it is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as at undiscovered error. A successful test is one that uncovers an as at undiscovered error.

Software testing is usually performed for one of two reasons:

- Defect Detection
- Reliability estimation

BLACK BOX TESTING:

The base of the black box testing strategy lies in the selection of appropriate data as per functionality and testing it against the functional specifications in order to check for normal and abnormal behavior of the system. Now a days, it is becoming to route the testing work to a third party as the developer of the system knows too much of the internal logic and coding of the system, which makes it unfit to test application by the developer. The following are different types of techniques involved in black box testing. They are:

- Decision Table Testing
- All pairs testing
- State transition tables testing
- Equivalence Partitioning

Software testing is used in association with Verification and Validation. Verification is the checking of or testing of items, including software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques as reviews, inspections, walk-through. Validation is the process of checking what has been specified is what the user actually wanted.

- Validation: Are we doing the right job?
- Verification: Are we doing the job right?

In order to achieve consistency in the Testing style, it is imperative to have and follow a set of testing principles. This enhances the efficiency of testing within SQA team members and thus contributes to increased productivity. The purpose of this document is to provide overview of the testing, plus the techniques. Here, after training is done on the training dataset, testing is done.

<u>WHITE BOX TESTING</u>:

White box testing requires access to source code. Though white box testing can be performed any time in the life cycle after the code is developed, it is a good practice to perform white box testing during unit testing phase. In designing of database, the flow of specific inputs through the code, expected output and the functionality of conditional loops are tested.

At SDEI, 3 levels of software testing are done at various SDLC phases:

UNIT TESTING: in which each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented

INTEGRATION TESTING: in which progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a whole.

SYSTEM TESTING: in which the software is integrated to the overall product and tested to show that all requirements are met. A further level of testing is also done, in accordance with requirements:

REGRESSION TESTING: is used to refer the repetition of the earlier successful tests to ensure that changes made in the software have not introduced new bugs/side effects.

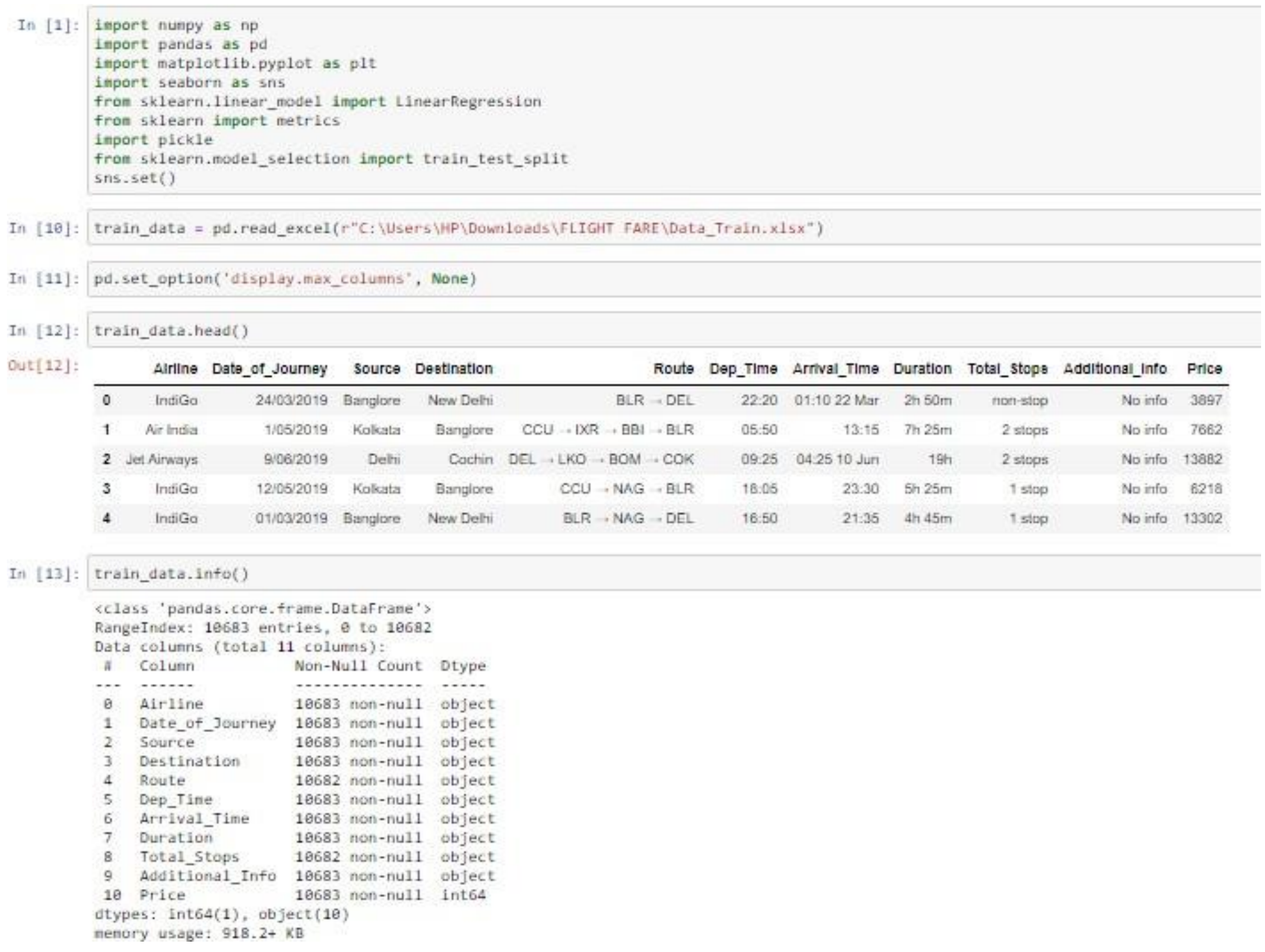
ACCEPTANCE TESTING: Testing to verify a product meets customer specified requirements. The acceptance test suite is run against supplied input data. Then the results obtained are compared with the expected results of the client. A correct match was obtained.

7. RESULT

In machine learning project Result is viewed by module wise.

- Data Collection
- Cleaning and preparing data.
- Data preprocessing
- Applying the algorithms

Data Collection:



```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.linear_model import LinearRegression
        from sklearn import metrics
        import pickle
        from sklearn.model_selection import train_test_split
        sns.set()
```

```
In [10]: train_data = pd.read_excel(r"C:\Users\HP\Downloads\FLIGHT FARE\Data_Train.xlsx")
```

```
In [11]: pd.set_option('display.max_columns', None)
```

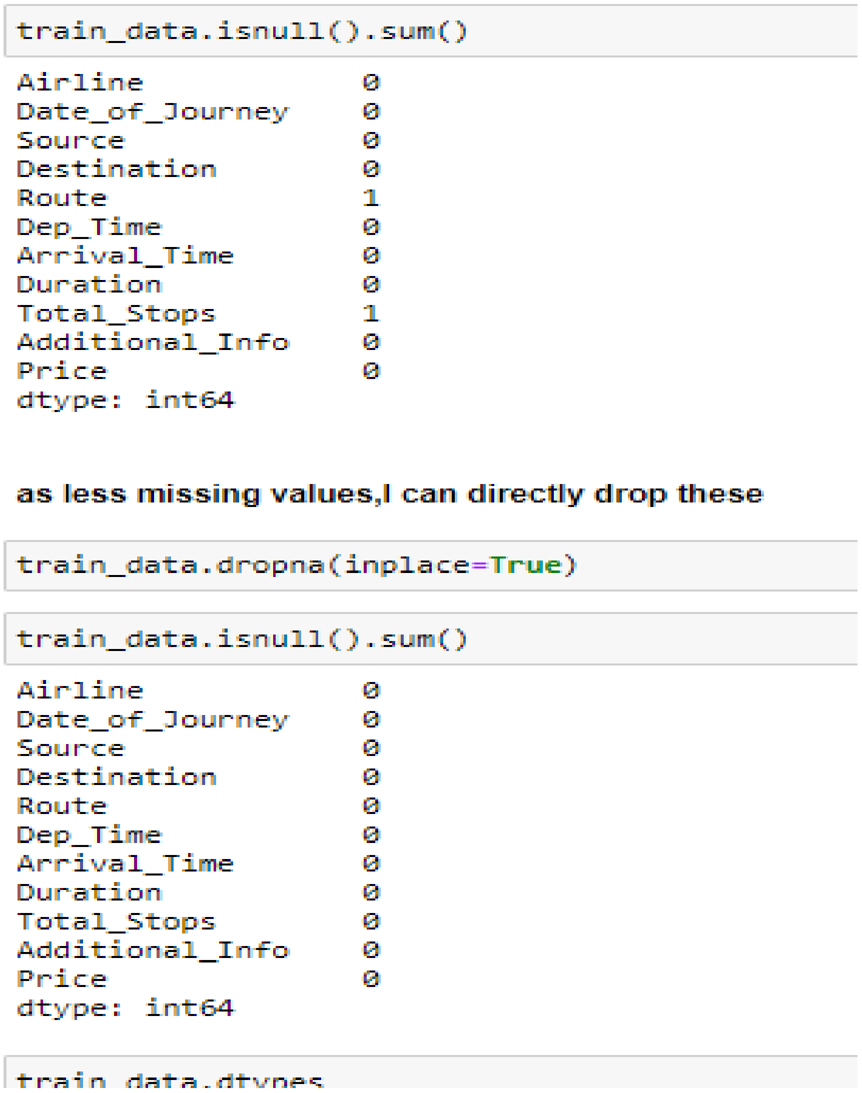
```
In [12]: train_data.head()
```

Out[12]:

	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

```
In [13]: train_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

Cleaning and preparing data:



```
train_data.isnull().sum()
```

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              1
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        1
Additional_Info    0
Price              0
dtype: int64
```

as less missing values,I can directly drop these

```
train_data.dropna(inplace=True)
```

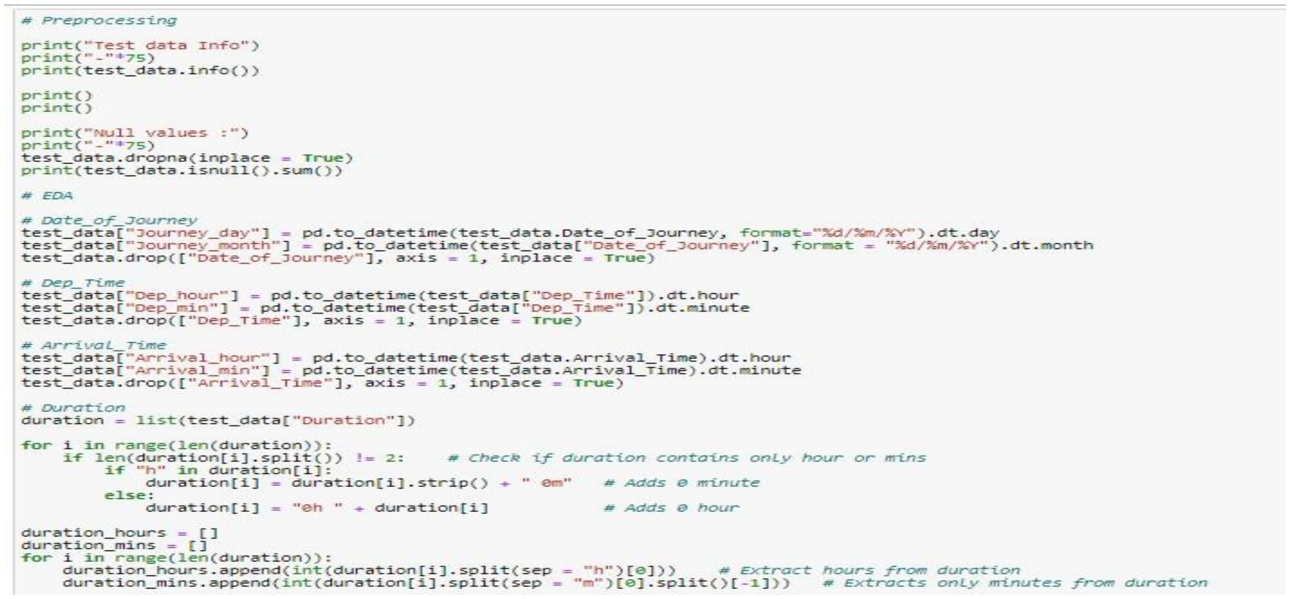
```
train_data.isnull().sum()
```

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
Price              0
dtype: int64
```

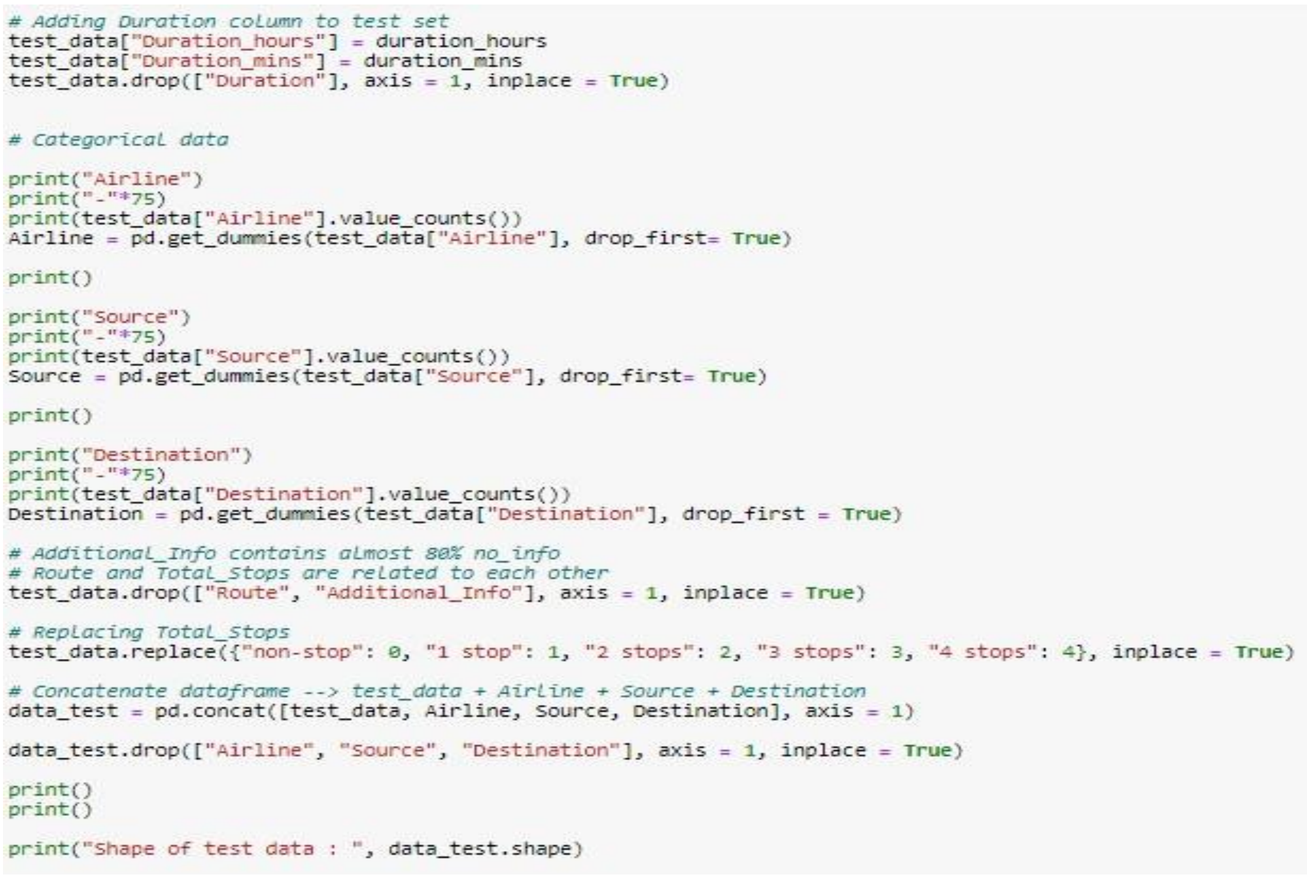
```
train_data.dtypes
```

\

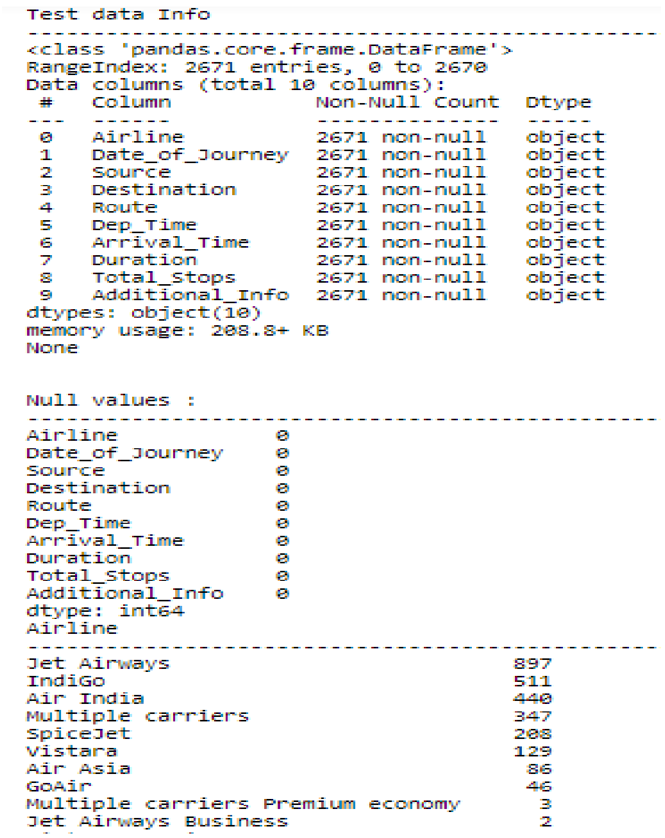
Data preprocessing:



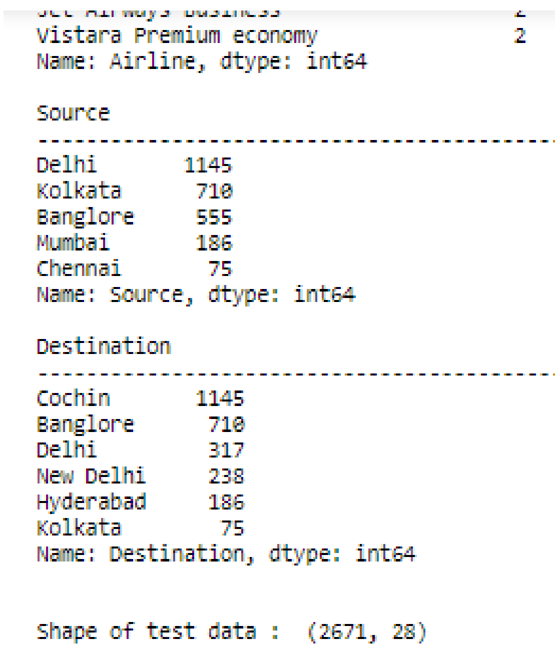
```
# Preprocessing

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

print()
print()

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

# EDA

# Date_of_Journey
test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey, format="%d/%m/%Y").dt.day
test_data["Journey_month"] = pd.to_datetime(test_data["Date_of_Journey"], format = "%d/%m/%Y").dt.month
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

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

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

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

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))   # Extracts only minutes from duration
```



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


# Categorical data

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

print()

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

print()

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

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

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

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

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

print()
print()

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

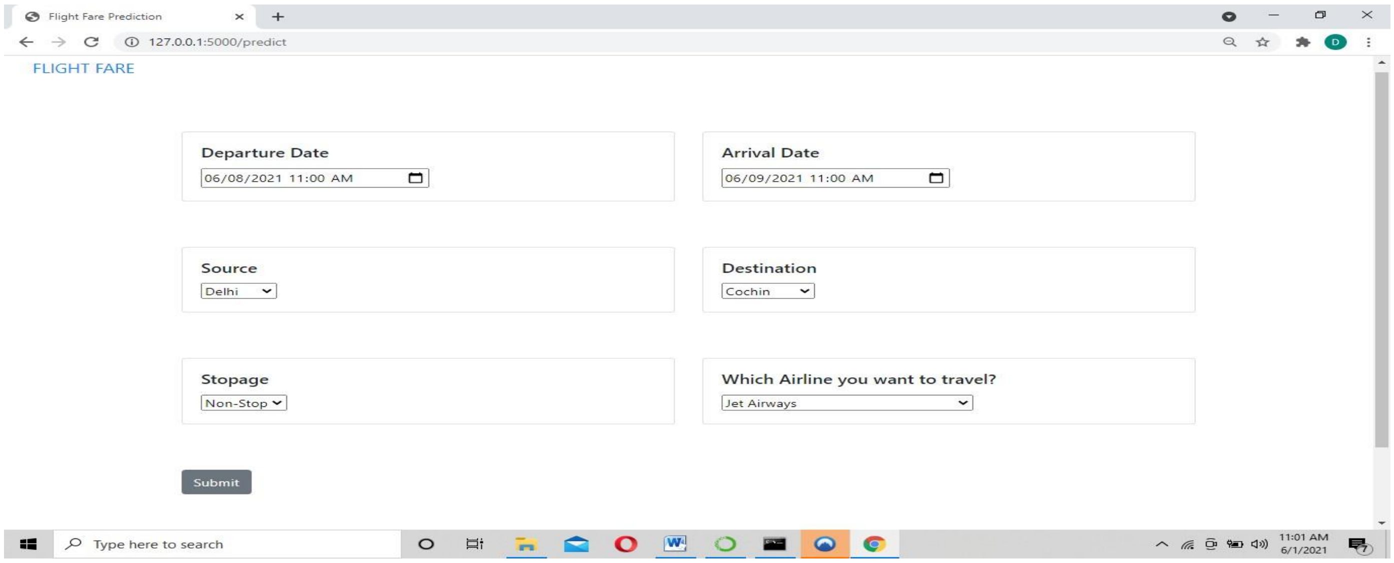


```
Test data Info
-----------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          2671 non-null   object
 1   Date_of_Journey  2671 non-null   object
 2   Source           2671 non-null   object
 3   Destination      2671 non-null   object
 4   Route            2671 non-null   object
 5   Dep_Time         2671 non-null   object
 6   Arrival_Time     2671 non-null   object
 7   Duration         2671 non-null   object
 8   Total_Stops      2671 non-null   object
 9   Additional_Info  2671 non-null   object
dtypes: object(10)
memory usage: 208.8+ KB
None

Null values :
-----------------------------------------------------
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
dtype: int64
Airline
-----------------------------------------------------
Jet Airways                          897
IndiGo                               511
Air India                            440
Multiple carriers                    347
SpiceJet                             208
Vistara                              129
Air Asia                              86
GoAir                                 46
Multiple carriers Premium economy      3
Jet Airways Business                   2
```



```
Vistara Premium economy              2
Name: Airline, dtype: int64

Source
-------------------------------------------
Delhi       1145
Kolkata      710
Banglore     555
Mumbai       186
Chennai       75
Name: Source, dtype: int64

Destination
-------------------------------------------
Cochin       1145
Banglore      710
Delhi         317
New Delhi     238
Hyderabad     186
Kolkata        75
Name: Destination, dtype: int64


Shape of test data :  (2671, 28)
```

Output


Flight Fare Prediction

127.0.0.1:5000/predict

FLIGHT FARE

Departure Date
06/08/2021 11:00 AM

Arrival Date
06/09/2021 11:00 AM

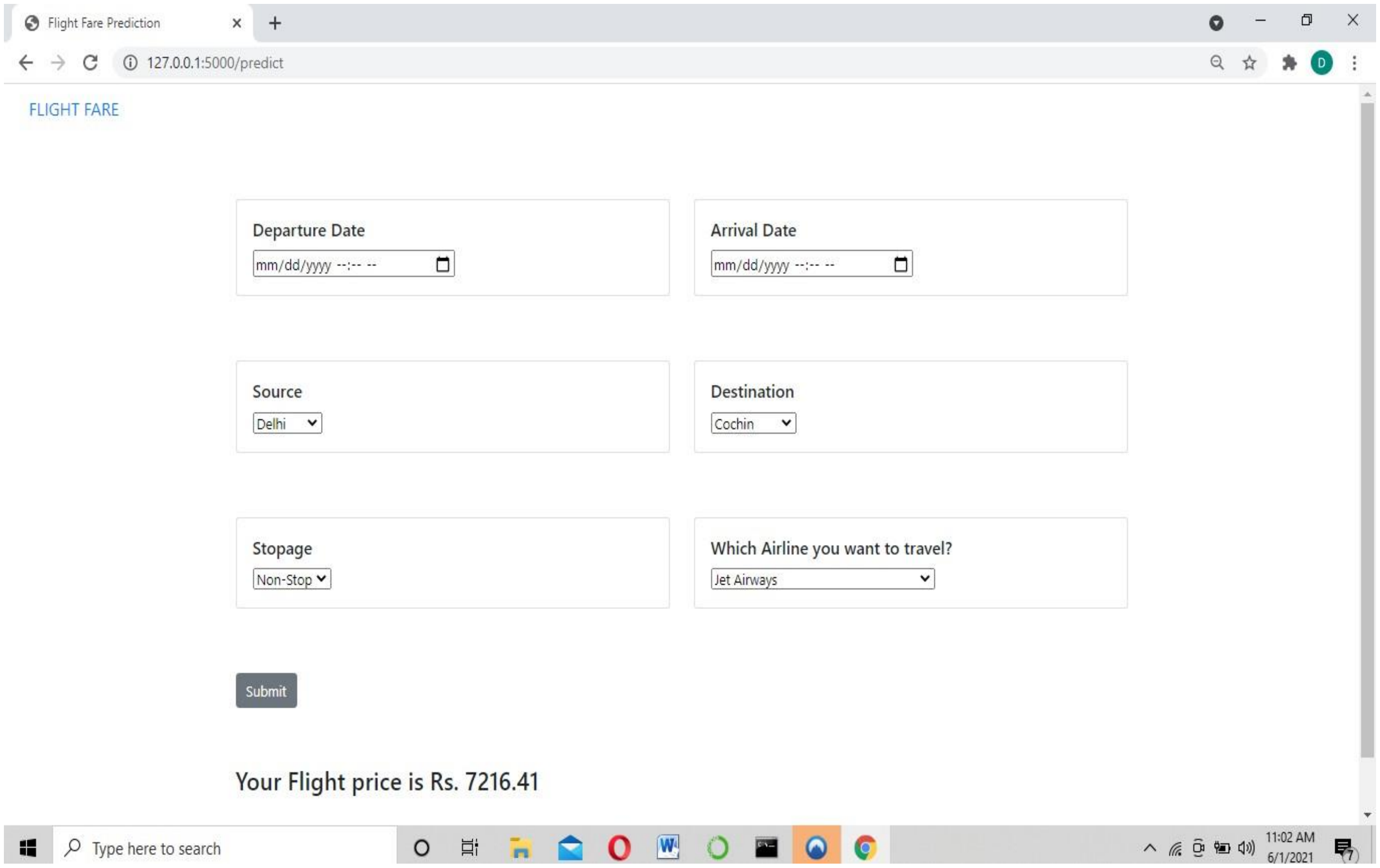
Source
Delhi

Destination
Cochin

Stopage
Non-Stop

Which Airline you want to travel?
Jet Airways

Submit


Flight Fare Prediction

127.0.0.1:5000/predict

FLIGHT FARE

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

Arrival Date
mm/dd/yyyy --:-- --

Source
Delhi

Destination
Cochin

Stopage
Non-Stop

Which Airline you want to travel?
Jet Airways

Submit

Your Flight price is Rs. 7216.41

8. CONCLUSION

To evaluate the conventional algorithms, a dataset is built for different routes across India and studied a trend of price variation for the period of limited days. Machine Learning algorithms are applied on the dataset to predict the dynamic fare of flights. As Random Forest Regressor algorithm has found to have best accuracy with less RMSE value as compared with other algorithmms used, we used RandomForest Regressor Algorithm to predic the values of flight fare to get a flight ticket at minimum cost.

Data is collected from the websites which sell the flight tickets so only limited informationcan be accessed. The values of R-squared obtained from the algorithm give the accuracy ofthe model. In the future, if more data could be accessed such as the current availability of seats, the predicted results will be more accurate.

This project reported on a preliminary study in "airfare prices prediction". We gathered airfare data from a specific Greek airline corporation (Aegean Airlines) from the web and showed that it is feasible to predict prices for flights based on historical fare data. The experimental results show that ML models are a satisfactory tool for predicting airfare prices. Other important factors in airfare prediction are the data collection and feature selection from which we drew some useful conclusions. From the experiments we concluded which features influence the airfare prediction at most. Apart from the features selected, there are other features that could improve the prediction accuracy.

9. FUTURE SCOPE AND ENHANCEMENT

In the future, this work could be extended to predict the airfare prices for the entire flight map of the airline. As the increase in the use of flights this days based on this we can say that in future if we implement or add more columns to this model the accuracy will be much more better and the prediction of this project will be more accurate.

In addition, Recurrent Neural Network(RNN) can be used to improve the accuracy of the model. Additional experiments on larger airfare data sets are essential, but this initial pilot study highlights the potential of Machine Learning models to guide consumers to make an airfare purchase in the best market period.

10. BIBLIOGRAPHY

[1] B. Smith, J. Leimkuhler, R. Darrow, and Samuels,―Yield managementat american airlines,‖Interfaces, vol.22, pp. 8–31, 1992.

[2] 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.

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

[4]Viet Hoang Vu, Quang Tran Minh and Phu H. Phung,‖An Airfare Prediction Model for Developing Markets‖, IEEE paper 2018.

[5] S.B. Kotsiantis, ―Decision trees: a recent overview,‖ Artificial Intelligence Review, vol. 39, no. 4, pp. 261-283, 2013.

[6] L. Breiman, ―Random forests,‖ Machine Learning, vol. 45, pp. 5- 32 , 2001.

[7] S. Haykin, Neural Networks – A Comprehensive Foundation. Prentice Hall, 2nd Edition, 1999.

[8] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola and V. Vapnik, ‖Support vector regression machines,‖ Advances in neural information processing systems, vol. 9, pp. 155-161, 1997.

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

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

[11] K. Tziridis, Th. Kalampokas, G.A. Papakotas and K.I. Diamantaras,‖Airfare Prices Prediction Using Machine Learning Techniques‖, EUSIPCO 2017