FLIGHT DELAY PREDICTION

A Mini project report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

G. Jeshmitha-17071A05J4

U. Bindu Sri Sai-17071A05N2

M. Samanvita-17071A05L0

J. Mahesh-18075A0540

Under the guidance of

Mrs. N. Lakshmi Kalyani

(Assistant Professor, VNR VJIET)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING &TECHNOLOGY

(An Autonomous Institute, NAAC Accredited With ‘A++’ Grade, NBA Accredited, Approved by AICTE, New Delhi, Affiliated to JNTUH)

VALLURUPALLI NAGESHWARA RAO VIGNANA JYOTHI
INSTITUTE OF ENGINEERING &TECHNOLOGY

(An Autonomous Institute)

Hyderabad – 500090

CERTIFICATE

This is to certify that **G. Jeshmitha-17071A05J4, U. Bindu Sri Sai-17071A05N2, M. Samanvita-17071A05L0, J. Mahesh-18075A0540** have successfully completed their project work at Department of CSE, VNR VJIET, Hyderabad entitled **"FLIGHT DELAY PREDICTION"** in partial fulfilment of the requirements for the award of B.Tech degree during the academic year 2019-20.

Mrs. N. Lakshmi Kalyani
Assistant Professor & Internal Guide
Department of Computer Science
Science
VNRVJIET

Mrs. B.V.Kiranmayee
Professor & HOD
Department of Computer
Science
VNRVJIET

DECLARATION

We hereby declare that the project entitled **"FLIGHT DELAY PREDICTION"** submitted in partial fulfilment of the requirements for award of the degree of Bachelor of Technology in Computer Science and Engineering at **VNR Vignana Jyothi Institute of Engineering and Technology,** affiliated to Jawaharlal Nehru Technological University, Hyderabad, is a bonafide report of the work carried out by us under the guidance and supervision of Mrs. N. Lakshmi Kalyani (Assistant Professor), Department of CSE, VNRVJIET. To the best of our knowledge, this report has not been submitted in any form to any University/Institute for award of any degree or diploma.

G. Jeshmitha	**M. Samanvita**	**U. Bindu Sri Sai**	**J. Mahesh**
(17071A05J4)	(17071A05L0)	(17071A05N2)	(18075A0540)
III B. Tech CSE	III B. Tech CSE	III B. Tech CSE	III B. Tech CSE
VNRVJIET	VNRVJIET	VNRVJIET	VNRVJIET

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who activated it, without it would ever never have come into existence. To them we lay the words of gratitude imprinting within us.

We are indebted to our venerable principal **Dr. C. D. Naidu** for this unflinching devotion, which lead us to complete this project. The support, encouragement given by him and his motivation lead us to complete this project. We are very much thankful to our H.O.D., **Mrs. B.V. Kiranmayee** madam for extending her cooperation in doing this project.

We extend our heartfelt thanks to our guide, **Mrs. N. Lakshmi Kalyani** madam, for her enthusiastic guidance throughout the course of our project. We also express our sincere thanks to our Mini project co-ordinators **Mrs. N.V.Sailaja and Mr. A. Brahmananda Reddy** who extended their valuable support in helping us complete the project in a correct way.

Last but not the least, our appreciable obligation also goes to all staff members of Computer Science & Engineering Department and to our fellow classmates who directly or indirectly helped us.

G. Jeshmitha	(17071A05J4)
M. Samanvita	(17071A05L0)
U. Bindu Sri Sai	(17071A05N2)
J. Mahesh	(18075A0540)

ABSTRACT

The prediction of flight delays plays a significantly important role for airlines and travellers because flight delays cause not only tremendous economic loss but also potential security risks. In this work, we aim to integrate multiple data sources to predict the departure delay of a scheduled flight. The primary goal of the model proposed in this paper is to predict airline delays using supervised machine learning algorithms. US domestic flight data and the weather data from July 2019 to December 2019 were extracted and used to train the model. To overcome the effects of imbalanced training data, sampling techniques are applied. XGBoost and linear regression were implemented to build models which can predict delays of individual flights. These models were built by continually tuning the hyper parameters to achieve greater accuracy. Then, each of the algorithms performance metrics were analysed. In the prediction step, flight schedule and weather forecast were gathered and fed into the model. Using this data, the XGBoost trained model performed a binary classification to predicted whether a scheduled flight will be delayed or on-time and the linear regression model predicts the delay time of the flight .

INDEX

LIST OF FIGURES

1. INTRODUCTION

1.1 Introduction

1.1.1 Flight and Flight delay

Air travel has been a big tool in an era of globalisation. Everyday there are long distance international and domestic journeys across the globe. With flights and airlines increasing, Flight delays have become a serious problem for airlines and passengers fly by air. The Federal Aviation Administration, in 2017 (FAA) estimated a flight delay cost of $26.6 per annum. In 2018, the United States Government Accountability Office (GAO) reported that flight delay and cancellation accounted for an average of almost 33 percent of complaints from air travel passengers for selected airlines. Flight delays not only cause tremendous economic costs but also bad psychological impact on air travellers. Flight delays are widely spread in air travel area. In recent years, around a quarter of all commercial flights have been delayed or cancelled

Airlines usually increase the time between gate departure and gate arrival time in an effort to anticipate potential delays due to weather conditions or airport and airspace congestion. Although this operation provides passengers' additional certainty, it also increases passengers waiting time. A more accurate departure delay prediction model can provide passengers the same level of certainty, without arbitrarily increasing waiting time. According to the USA Department of Transportation's data (DOT), the primary cause of flight departure delay was bad weather and traffic control. For example, the report claims that airports can be closed due to severe weather. Additionally, the number of aircraft that can be safely accommodated in a given portion of airspace further affects capacity. Overloaded airspace is likely to lead to delays on the ground or en route

1.1.2 Weather Conditions

It's the component of the wind that's blowing across the runway in use. Planes like to take off into the wind, because it's the only thing in aviation that's free and provides lift.

When air flows over the wings, flight happens, and the wind helps with that during takeoff. Less wind helps to reduce flight delays.

Wet bulb globe temperature (WBGT) is an apparent measurement used to estimate the most accurate level of heat stress in direct sunlight.

Dry-bulb temperature (DBT) is the temperature of air measured by a thermometer freely exposed to the air.

The **dew point** is the temperature to which air must be cooled .The dew point in relation to the temperature gives the pilots information about the humidity, and can affect **visibility.**

Pressurization systems are designed to keep the interior cabin pressure between 12 and 11 psi at cruise altitude. On a typical flight, as the aircraft climbs to 36,000 feet, the interior of the plane climbs to between 6000-8000 feet.

1.1.3 Statistics

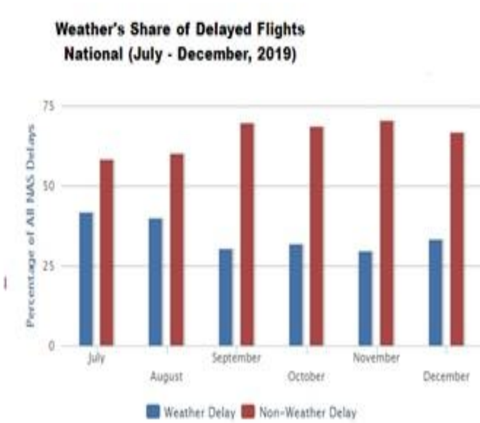


Fig 1.1:Weather share of delayed flights

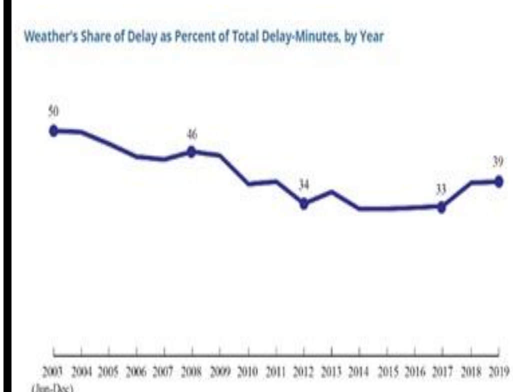


Fig 1.2: Weathers share of delay as percentage

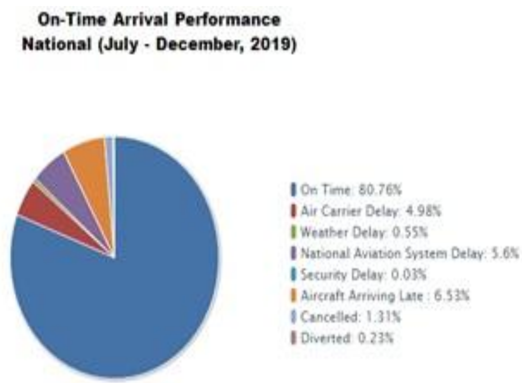


Fig 1.3: On-Time Arrival Performance

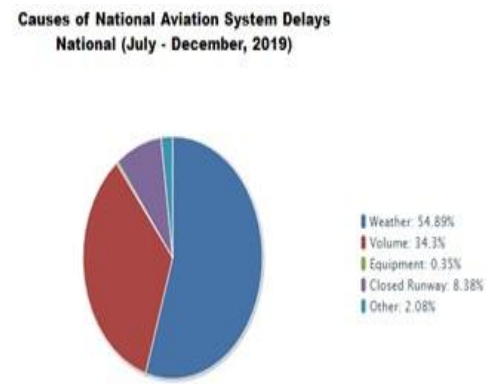


Fig 1.4: Causes of NAS delays

1.2 Existing System

The most common and traditional method used to track weather conditions is the usage of Weather Satellites which include information on storm location, temperature and heat balance in the earth's atmosphere.

1.2.1 Drawbacks of Existing System

The drawbacks of the current system include the following:

- Periodic inspections and remote monitoring of flights is necessary
- Passengers suffer a lot
- Prior Prediction is not possible
- Incapable of handling unexpected delays

1.3 Proposed System

XGB Classifier: XGBoost stands for "Extreme Gradient Boosting". XGBoost is used for supervised learning problems, where we use the training data (with multiple features) x_i to predict a target variable y_i. This algorithm has a classification accuracy of 94.1% in classifying into on-time or delay and regression error of 8 to 10 minutes in predicting delay time. Whereas the algorithms currently being used have an accuracy of 85%. Hence, we are trying to improve the accuracy with which we are predicting.

1.3.1 Advantages of Proposed System

The advantages of the Proposed System include the following:

- Different from previous work, we aim to take advantage of both flight information as well as weather conditions .
- In this system we predict ,not only if the flight is on-time or delayed, but also the approximate delay time in minutes.
- Using our proposed framework, an improvement in accuracy for flight departure delay prediction is obtained.
- Reduce further economic loss for airlines
- Lessen inconvenience occurred to passengers
- Optimize flight operations
- Airlines can determine efficient routes with minimum delay possibility

2. FEASIBILITY STUDY

A feasibility study involves taking a judgment call on whether a project is doable. The two criteria to judge feasibility are **cost required** and **value to be** delivered. A well-designed study should offer a historical background of the business or project, a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, such studies precede technical development and project implementation

A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions.

2.1 Technical Feasibility

Technical feasibility involves evaluation of the hardware and the software requirements of the proposed system.

In this project, the technology involved is Machine Learning. The language that is used to implement the concepts of Machine Learning is Python Programming and the tool that is used to execute the Python code is Jupyter Notebook (IPython notebook).

2.2 Economic Feasibility

Economic Feasibility helps in assessing the viability, cost, and benefits associated with projects before financial resources are allocated. This assessment typically involves a cost/ benefits analysis of the project.

The application is so designed that it requires minimal cost and eliminates costs as there would minimal need for manual work. The technologies used helps in understanding the user without any investment. As the machine will be trained it reduces the cost that is required to deploy the man power and also eliminates the problem of time consumption.

2.3 Legal Feasibility

The proposed system doesn’t conflict with legal requirements like data protection acts or social media laws. It ensures legal data access and gives prominence to data security.

2.4 Operational Feasibility

The application involves design-dependent parameters such as reliability, maintainability, supportability, usability, disposability, sustainability, affordability, and others. It minimizes the drawbacks of the current system by building an application that automatically resolves the user queries and helps to analyses the user data.

2.5 Scheduling Feasibility

The project development took place in timely process by understanding time schedules of the project and maintains good time line for project development.

3. LITERATURE SURVEY

3.1 Machine Learning

"Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed."

Machine learning is an application of artificial intelligence (AI) that provides systems 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 learn for themselves.

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 the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

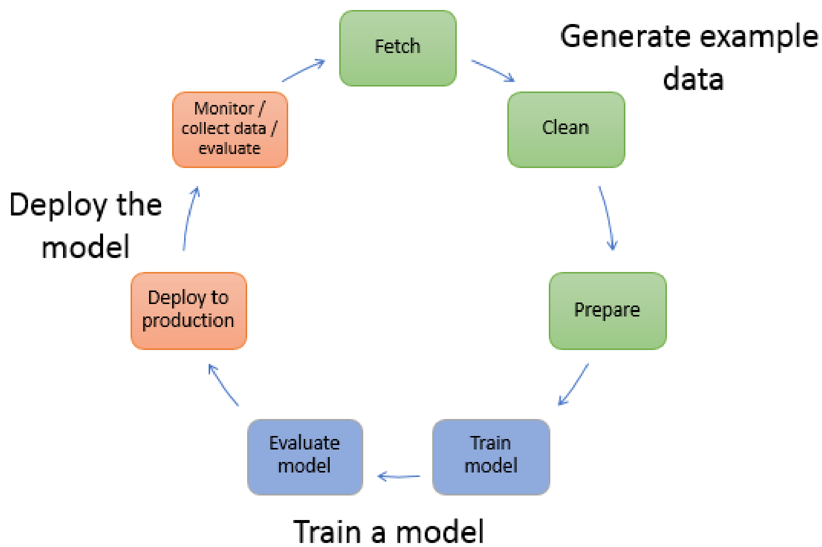


Fig 3.1: Block Diagram of Machine learning algorithm

3.2 Types of Machine Learning Methods

Some machine learning methods:

Machine learning algorithms are often categorized as supervised or unsupervised.

Supervised machine learning:

Supervised machine learning algorithms can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

Unsupervised machine learning:

Unsupervised machine learning algorithms are used when the information used to train is neither classified nor labelled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Semi-supervised machine learning:

Semi-supervised machine learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both labelled and unlabeled data for training typically a small amount of labelled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

Reinforcement machine learning:

Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behaviour within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly.

Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

The main topics of machine learning discussed in this project are:

- Classification
- Regression

Classification

Classification algorithms define which category the objects from the dataset belong to. Thus, categories are usually referred to as classes. By solving classification problems, you can address a variety of questions.

Binary classification problems:

• Is this email spam or not?

• Is this transaction fraudulent or not?

And, multiclass problems :

• Is this apartment in New York, San Francisco, or Boston?

• What is pictured: a cat, a dog, or a bird?

• Which type of product is this customer more likely to buy: a laptop, a desktop, or a Smartphone?

Performance metrics for classification:

- Confusion matrix
- Accuracy score
- Precision
- Recall
- F1score

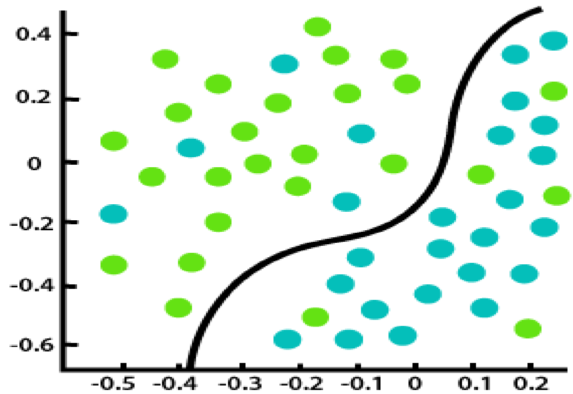


Fig 3.2: Classification

Regression

Regression algorithms define numeric target values instead of classes. By estimating numeric variables, these algorithms are used in predicting product demand, sales figures, marketing returns, etc.

For example:

• How many items of this product will we be able to sell next month?

• What's will the airfare be for this destination?

• What's going to be the rental price for this house?

Performance metrics for Regression:

- Root mean square error
- Mean squared error
- Mean absolute error
- r2 score

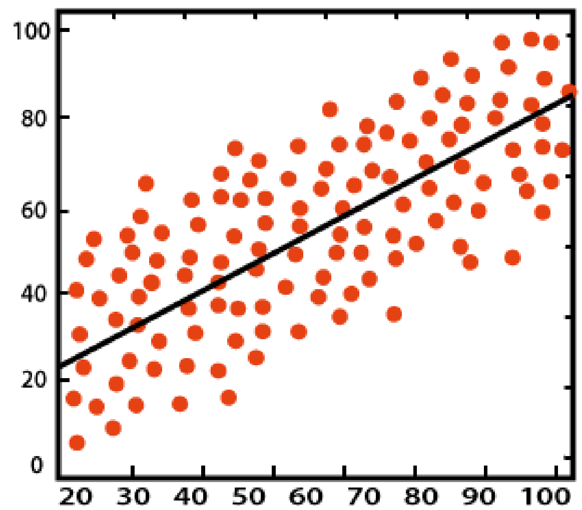


Fig 3.3: Regression

4. ALGORITHM DESCRIPTION

4.1 XGB Boost Algorithm

XGBoost stands for "Extreme Gradient Boosting". XGBoost is used for supervised learning problems, where we use the training data (with multiple features) xi to predict a target variable yi. This algorithm has an accuracy of 94.1%.Whereas the algorithms currently are used have an accuracy of 85% **XGBoost** is an optimized distributed gradient boosting library designed to be highly **efficient, flexible** and **portable.** It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

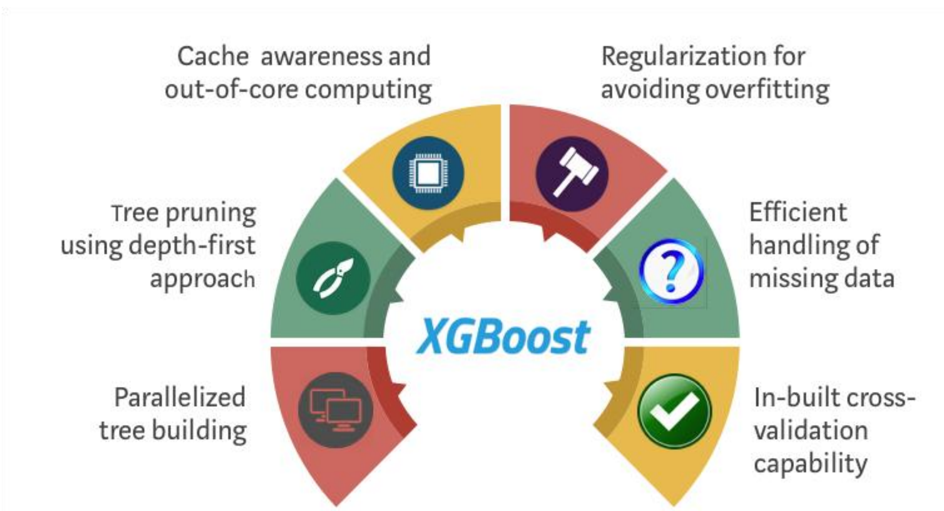


Fig 4.1: XGBoost

Before running XGBoost,

We must set three types of parameters: general parameters, booster parameters and task parameters.

- **General parameters:** relate to which booster we are using to do boosting, commonly tree or linear model
- **Booster parameters:** depend on which booster you have chosen
- **Learning task parameters:** decide on the learning scenario. For example, regression tasks may use different parameters with ranking tasks

4.2 Linear Regression

Linear regression is one of the most popular machine learning algorithms used to predict values given a certain set of values. Linear regression is a linear method to model the relationship between your independent variables and your dependent variables.
Advantages include how simple it is and ease with implementation and disadvantages include how is' lack of practicality and how most problems in our real world aren't "linear".

You can use the least square method to create a line that would best fit the data. Some applications of linear regression can be found in machine learning, economics and in places where estimation is required.

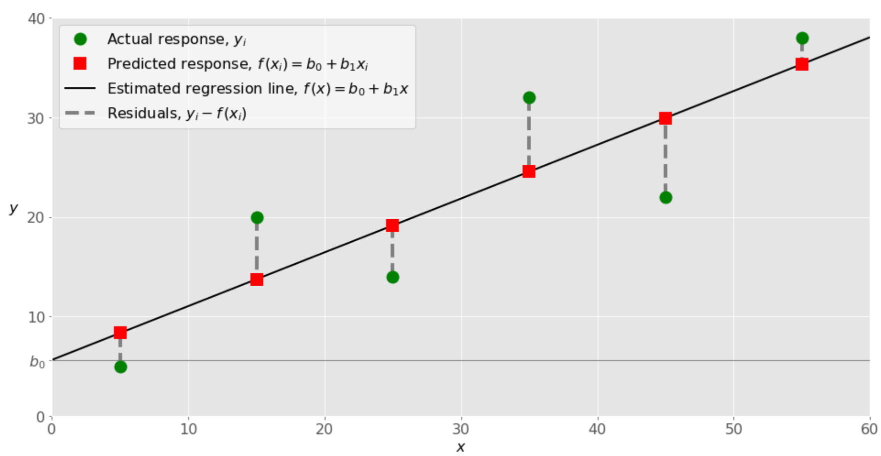


Fig 4.2: Linear Regression

Parameters for a Linear Regression model :

- **fit_intercept** is a Boolean (True by default) that decides whether to calculate the intercept b_0 (True) or consider it equal to zero (False).

- **normalize** is a Boolean (False by default) that decides whether to normalize the input variables (True) or not (False).
- **copy_X** is a Boolean (True by default) that decides whether to copy (True) or overwrite the input variables (False).
- **n_jobs** is an integer or None (default) and represents the number of jobs used in parallel computation. None usually means one job and -1 to use all processors.

5. SYSTEM ANALYSIS

5.1 System Requirements:

5.1.1 Jupyter Notebook

The IPython Notebook is now known as the Jupyter Notebook. It is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media.

The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The IPython notebook combines two components:

A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.

Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

Fig 5.1 Jupyter Notebook Logo

5.2 Python API's & Libraries requirements

5.2.1 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.Matplotlib tries to make easy things easy and hard things possible.You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code.

5.2.2 Sklearn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy.

5.2.3 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation etc

5.2.4 Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labelled" 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.

5.2.5 Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

5.3 Dataset:

5.3.1 Flight Dataset:

The US Bureau of Transport Statistics provides data on all domestic flights, including their scheduled and actual departure and takeoff times, date, origin, destination and carrier.

5.3.2 Weather Dataset:

The Local Climatological Data from NOAA includes the closest available weather data at the departure airport along with the standard flight data. The departure airport weather features include temperature, humidity, air pressure, and precipitation type and amount, if any.

5.3.3 Final Dataset:

Joining the flight and weather datasets to form final dataset:

This presents a significant challenge. The weather observation times are not the same as the flight departures, and the weather is not observed on a strict cycle stations report more or less often depending on how quickly the weather is changing. Ideally, we would then join based on the closest possible weather observation time. Instead, we calculated average of weather observation from each hour at each station , and then performed an inner join on the unique code filed that was created by us ,to uniquely identify flight and weather data at each hour on a particular day at a given place.

i. e., code used to join two datasets was: PLACE.YEAR.MONTH.DAY.HOUR

5.3.4 Features

Features considered in building the model:

- DEP_DEL15 (0(on-time) or 1(delay))
- DEP_DELAY
- DISTANCE
- Altimeter Setting
- Dew Point Temperature
- Dry Bulb Temperature

- Relative Humidity
- Station Pressure
- Visibility
- Wet Bulb Temperature
- Sea Level Pressure

.

6. SYSTEM DESIGN

6.1 UML Diagrams Introduction:

UML is a standard language for specifying, visualizing, constructing, and documenting the artefacts of software systems. UML can be described as a general-purpose visual modelling language to visualize, specify, construct and document software system. Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non-software systems as well like process flow in a manufacturing unit etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. The goal of UML can be defined as a simple modelling mechanism to model all possible practical systems in today’s complex environment.

6.2 Activity Diagram:

6.2.1 Definition:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes. Activity diagrams show the overall flow of control.

6.2.2 Activity Diagram

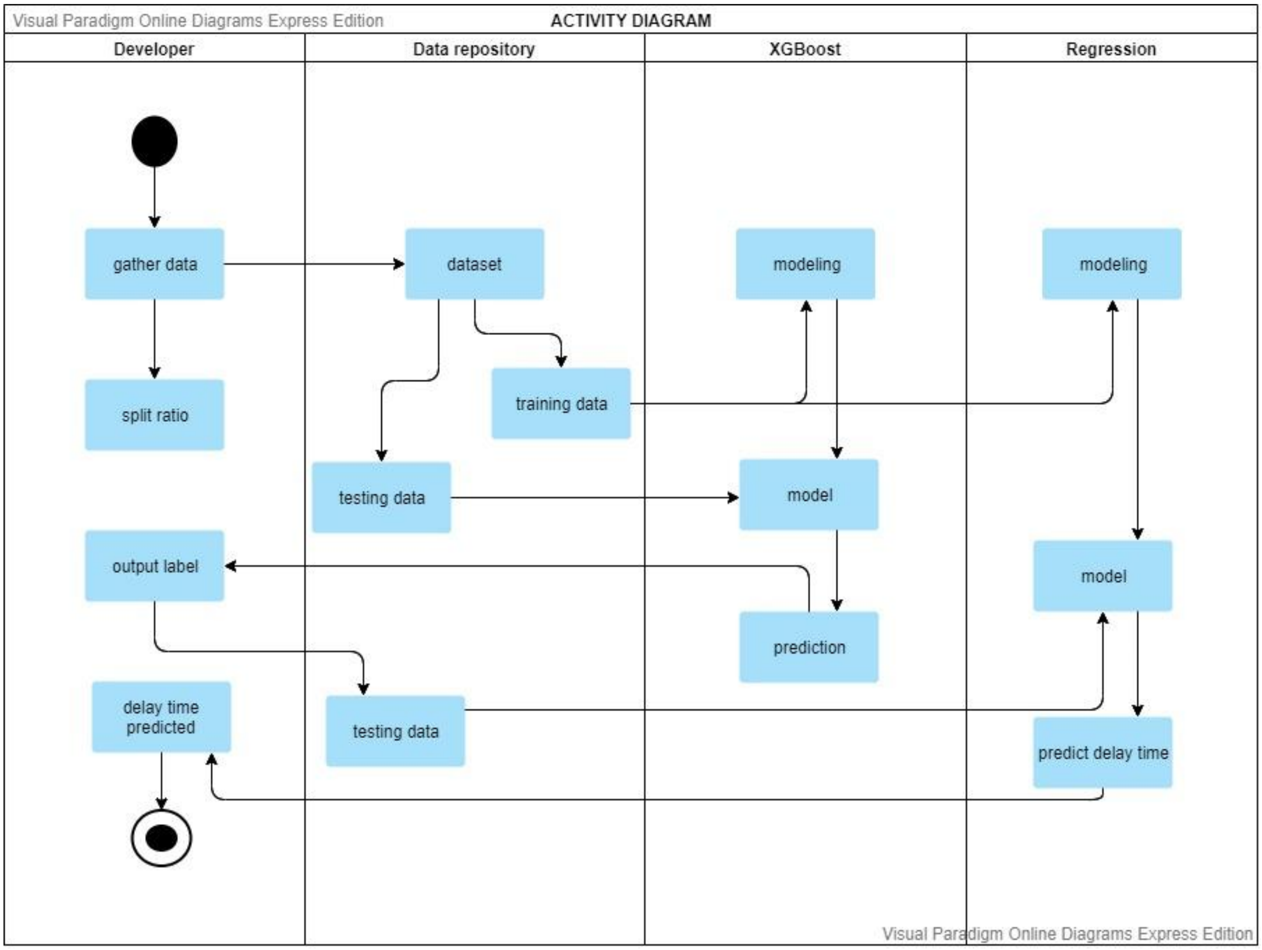


Fig 6.1: Activity Diagram

6.3 Class Diagram:

6.3.1 Definition:

A **class diagram** in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

6.3.2 Class Diagram

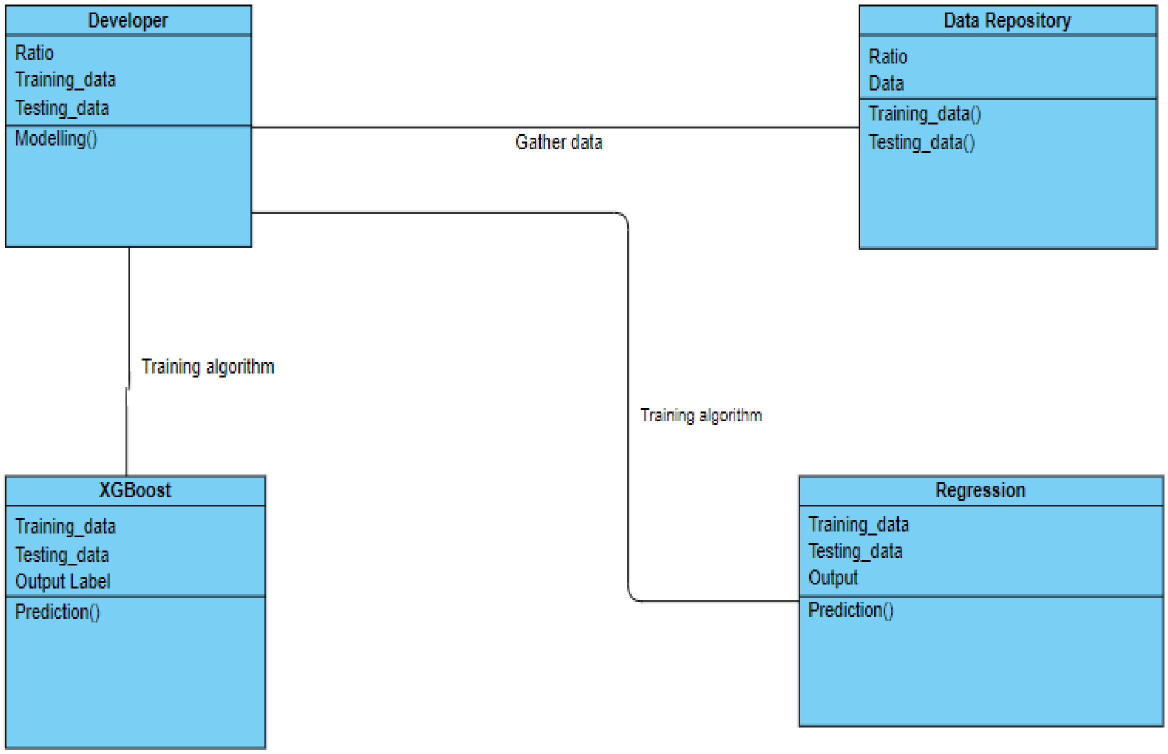


Fig 6.2: Class Diagram

6.4 Use case Diagram:

6.4.1 Definition:

Use case diagrams are a way to capture the system's functionality and requirements in UML diagrams. It captures the dynamic behaviour of a live system. A use case diagram consists of a use case and an actor. A use case represents a distinct functionality of a system, a component, a package, or a class.

6.4.2 Use case Diagram

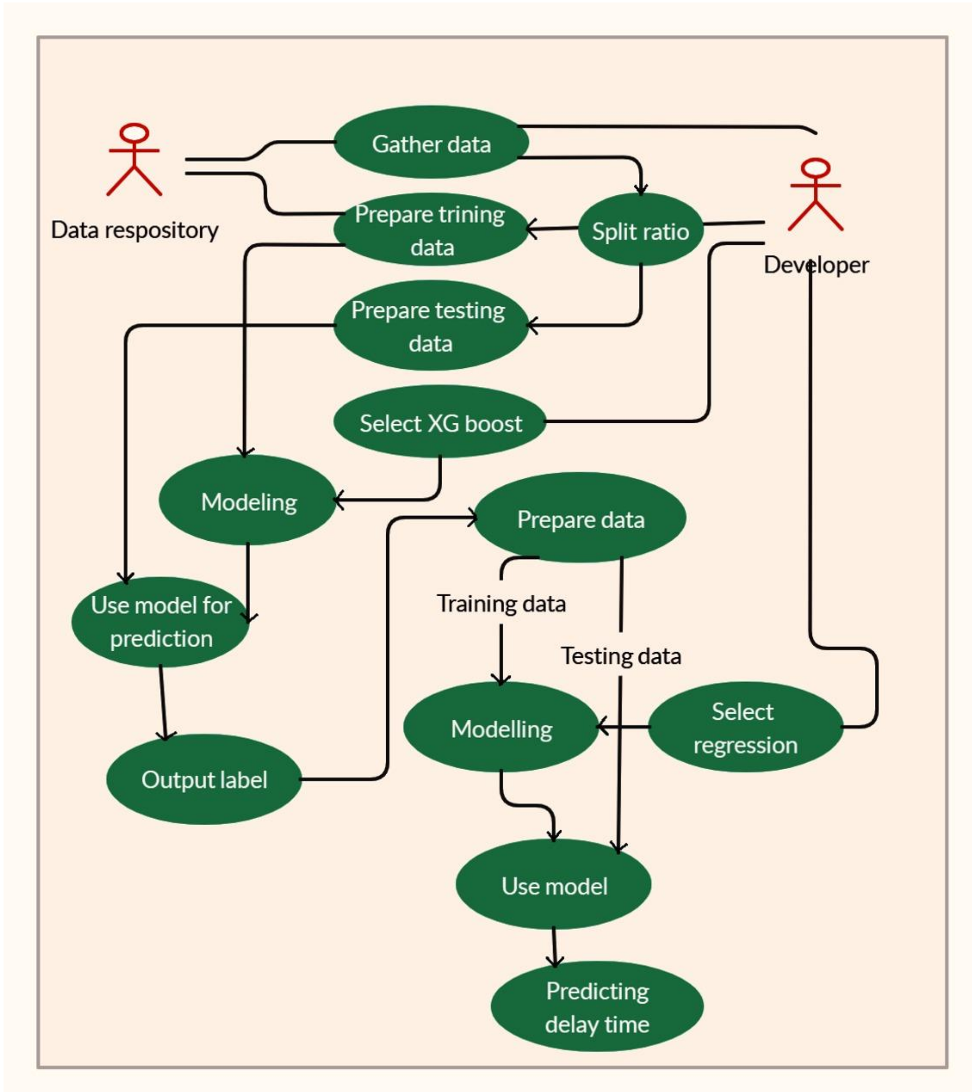


Fig 6.3: Use Case Diagram

6.5 Sequence Diagram:

6.5.1 Definition:

A **sequence diagram** simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.

6.5.2 Sequence Diagram

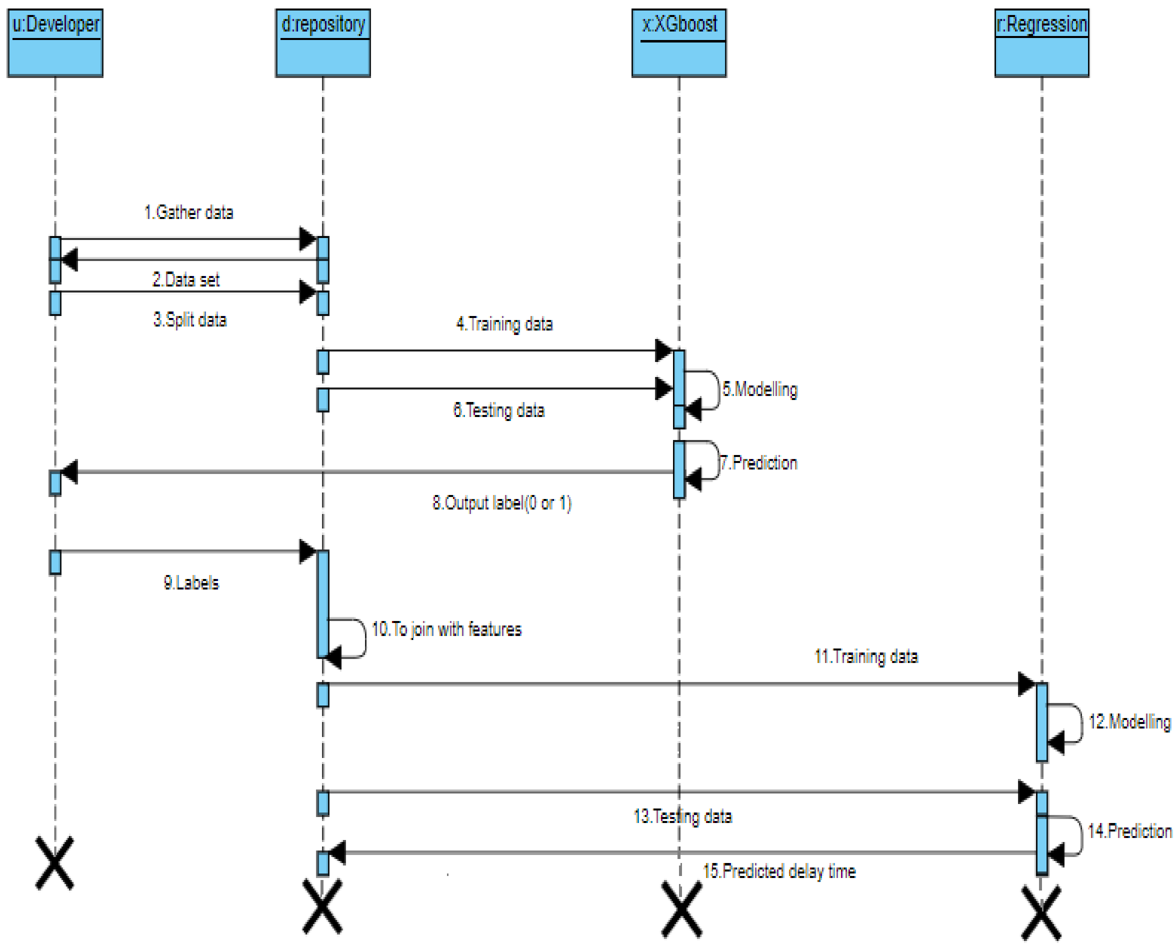


Fig 6.4: Sequence Diagram

7. IMPLEMENTATION

7.1 Implementation Flow Charts

7.1.1 Flow Chart for Data Pre-processing

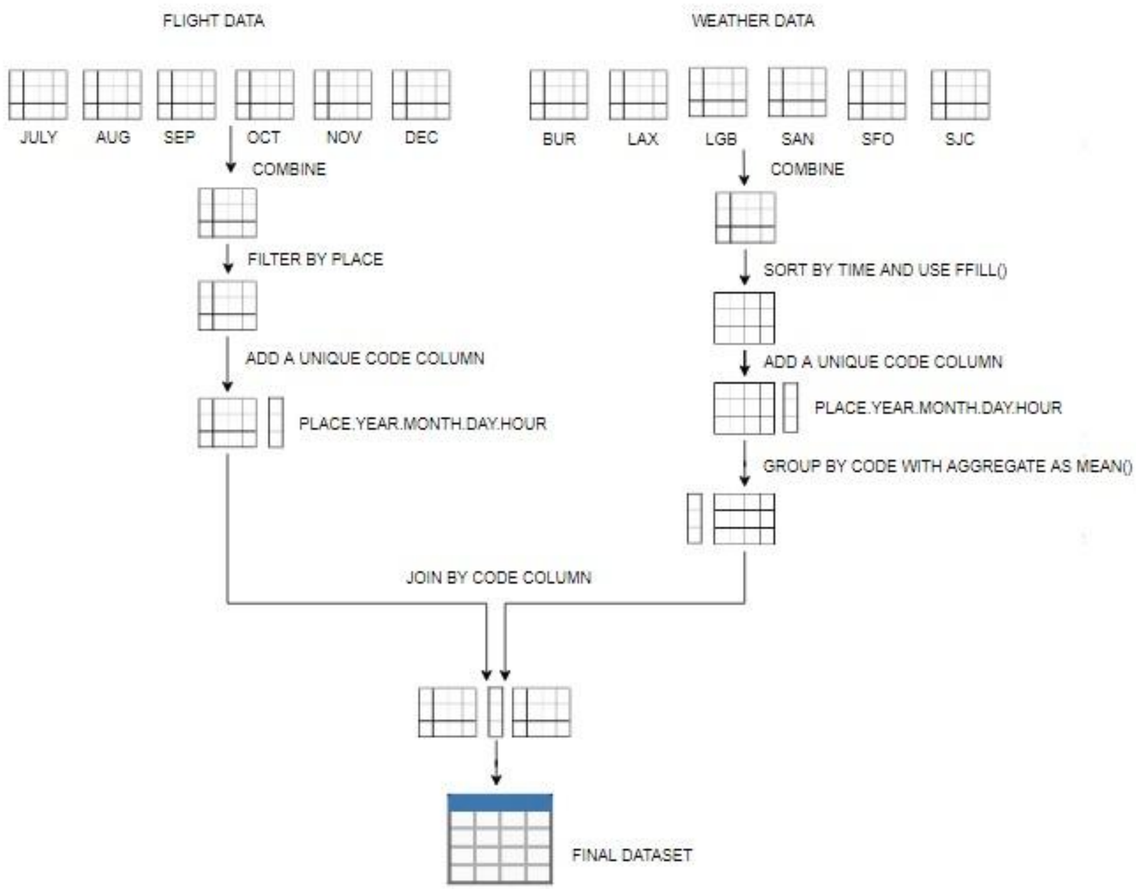


Figure 7.1 Flow Chart for Data Pre-Processing

7.1.2 Flow Chart for Modelling

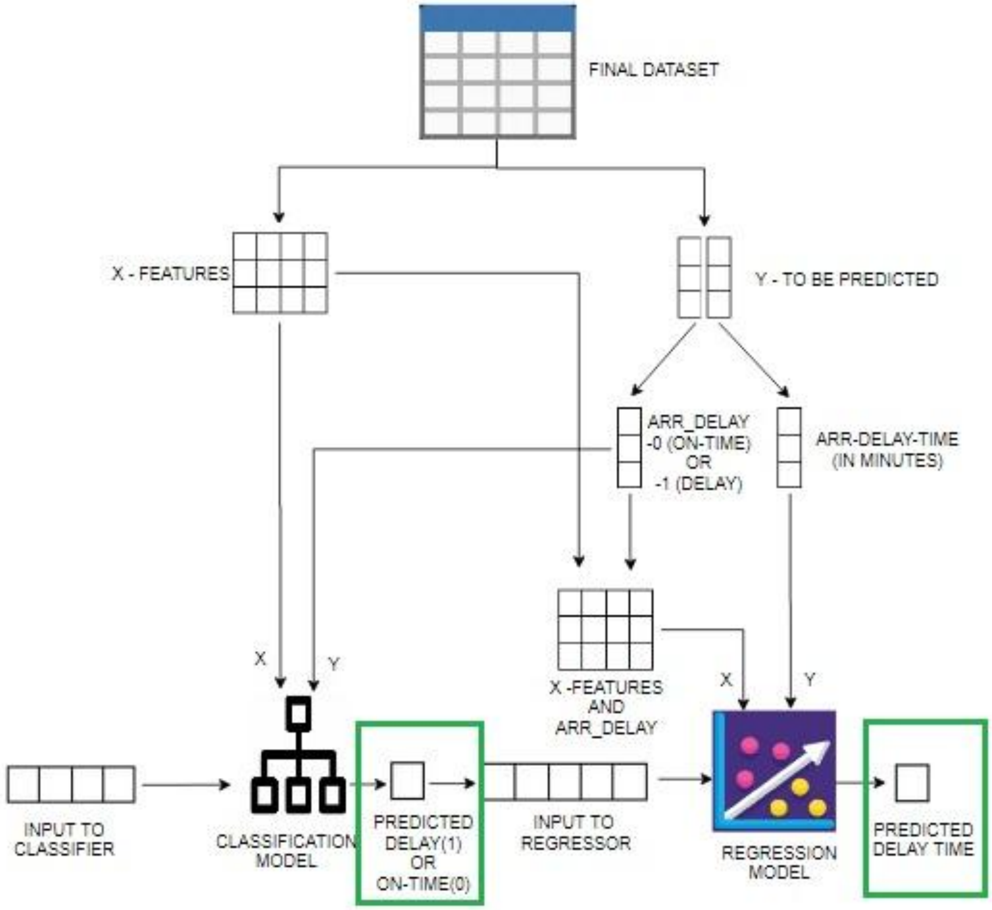


Fig 7.2: Flow Chart for Modelling

We can define the machine learning workflow in 5 stages:

- Gathering data
- Data pre-processing
- Researching the model that will be best for the type of data
- Training and testing the model
- Evaluation

7.2 Code

```
In [ ]: #import necessary python libraries
        import pandas as pd
        import numpy as np

        #collect U.S flight data
        flight_data_july=pd.read_csv("E:\\mini project data\\flight_data_july.csv")
        flight_data_august=pd.read_csv("E:\\mini project data\\flight_data_august.csv")
        flight_data_september=pd.read_csv("E:\\mini project data\\flight_data_september.csv")
        flight_data_october=pd.read_csv("E:\\mini project data\\flight_data_october.csv")
        flight_data_november=pd.read_csv("E:\\mini project data\\flight_data_november.csv")
        flight_data_december=pd.read_csv("E:\\mini project data\\flight_data_december.csv")

        #airports
        codes=['BUR','LAX','LGB','SAN','SFO','SJC']

        #filter flight data based on airports
        flight_data_7=flight_data_july[flight_data_july['ORIGIN'].isin(codes)]
        flight_data_8=flight_data_august[flight_data_august['ORIGIN'].isin(codes)]
        flight_data_9=flight_data_september[flight_data_september['ORIGIN'].isin(codes)]
        flight_data_10=flight_data_october[flight_data_october['ORIGIN'].isin(codes)]
        flight_data_11=flight_data_november[flight_data_november['ORIGIN'].isin(codes)]
        flight_data_12=flight_data_december[flight_data_december['ORIGIN'].isin(codes)]

        #complete flight data
        flight_data=pd.concat([flight_data_7,flight_data_8,flight_data_9,flight_data_10,flight_data_11,flight_(
        flight_data.reset_index(drop=True , inplace=True)
        flight_data.info()
        flight_data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 295611 entries, 0 to 295610
Data columns (total 60 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   YEAR                   295611 non-null  int64
 1   QUARTER                295611 non-null  int64
 2   MONTH                  295611 non-null  int64
 3   DAY_OF_MONTH           295611 non-null  int64
 4   DAY_OF_WEEK            295611 non-null  int64
 5   FL_DATE                295611 non-null  object
 6   OP_UNIQUE_CARRIER      295611 non-null  object
 7   OP_CARRIER_AIRLINE_ID  295611 non-null  int64
 8   OP_CARRIER             295611 non-null  object
 9   TAIL_NUM               295209 non-null  object
 10  OP_CARRIER_FL_NUM      295611 non-null  int64
 11  ORIGIN_AIRPORT_ID      295611 non-null  int64
 12  ORIGIN_AIRPORT_SEQ_ID  295611 non-null  int64
 13  ORIGIN_CITY_MARKET_ID  295611 non-null  int64
 14  ORIGIN                 295611 non-null  object
 15  ORIGIN_CITY_NAME       295611 non-null  object
 16  ORIGIN_STATE_ABR       295611 non-null  object
 17  ORIGIN_STATE_FIPS      295611 non-null  int64
 18  ORIGIN_STATE_NM        295611 non-null  object
 19  ORIGIN_WAC             295611 non-null  int64
 20  DEST_AIRPORT_ID        295611 non-null  int64
 21  DEST_AIRPORT_SEQ_ID    295611 non-null  int64
 22  DEST_CITY_MARKET_ID    295611 non-null  int64
 23  DEST                   295611 non-null  object
 24  DEST_CITY_NAME         295611 non-null  object
```

```
 26  DEST_STATE_FIPS       295611 non-null  int64
 27  DEST_STATE_NM         295611 non-null  object
 28  DEST_WAC              295611 non-null  int64
 29  CRS_DEP_TIME          295611 non-null  int64
 30  DEP_TIME              292081 non-null  float64
 31  DEP_DELAY             292081 non-null  float64
 32  DEP_DELAY_NEW         292081 non-null  float64
 33  DEP_DEL15             292081 non-null  float64
 34  DEP_DELAY_GROUP       292081 non-null  float64
 35  DEP_TIME_BLK          295611 non-null  object
 36  TAXI_OUT              292016 non-null  float64
 37  WHEELS_OFF            292016 non-null  float64
 38  WHEELS_ON             291851 non-null  float64
 39  TAXI_IN               291851 non-null  float64
 40  CRS_ARR_TIME          295611 non-null  int64
 41  ARR_TIME              291852 non-null  float64
 42  ARR_DELAY             291219 non-null  float64
 43  ARR_DELAY_NEW         291219 non-null  float64
 44  ARR_DEL15             291219 non-null  float64
 45  ARR_DELAY_GROUP       291219 non-null  float64
 46  ARR_TIME_BLK          295611 non-null  object
 47  CRS_ELAPSED_TIME      295611 non-null  float64
 48  ACTUAL_ELAPSED_TIME   291219 non-null  float64
 49  AIR_TIME              291219 non-null  float64
 50  FLIGHTS               295611 non-null  float64
 51  DISTANCE              295611 non-null  float64
 52  DISTANCE_GROUP        295611 non-null  int64
 53  CARRIER_DELAY         39529 non-null   float64
 54  WEATHER_DELAY         39529 non-null   float64
 55  NAS_DELAY             39529 non-null   float64
 56  SECURITY_DELAY        39529 non-null   float64
 57  LATE_AIRCRAFT_DELAY   39529 non-null   float64
 58  Unnamed: 53           0 non-null       float64
 59  Unnamed: 58           0 non-null       float64
dtypes: float64(26), int64(20), object(14)
memory usage: 135.3+ MB
```

Out[ ]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_DATE	OP_UNIQUE_CARRIER	OP_CARRIER_AIRLINE_
0	2019	3	7	1	1	01-07-2019	NK	204
1	2019	3	7	2	2	02-07-2019	NK	204
2	2019	3	7	3	3	03-07-2019	NK	204
3	2019	3	7	4	4	04-07-2019	NK	204
4	2019	3	7	5	5	05-07-2019	NK	204
...	...	...	...	...	...	...	...	
295606	2019	4	12	31	2	2019-12-31	B6	204
295607	2019	4	12	31	2	2019-12-31	B6	204
295608	2019	4	12	31	2	2019-12-31	B6	204
295609	2019	4	12	31	2	2019-12-31	B6	204
295610	2019	4	12	31	2	2019-12-31	B6	204

295611 rows × 60 columns

```
In [ ]: # Do required type castings
        flight_data[['CRS_DEP_TIME']]=flight_data[['CRS_DEP_TIME']].astype(str)
        flight_data[['YEAR']]=flight_data[['YEAR']].astype(str)
        flight_data[['MONTH']]=flight_data[['MONTH']].astype(str)
        flight_data[['DAY_OF_MONTH']]=flight_data[['DAY_OF_MONTH']].astype(str)
        flight_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 295611 entries, 0 to 295610
Data columns (total 60 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   YEAR                   295611 non-null  object
 1   QUARTER                295611 non-null  int64
 2   MONTH                  295611 non-null  object
 3   DAY_OF_MONTH           295611 non-null  object
 4   DAY_OF_WEEK            295611 non-null  int64
 5   FL_DATE                295611 non-null  object
 6   OP_UNIQUE_CARRIER      295611 non-null  object
 7   OP_CARRIER_AIRLINE_ID  295611 non-null  int64
 8   OP_CARRIER             295611 non-null  object
 9   TAIL_NUM               295209 non-null  object
 10  OP_CARRIER_FL_NUM      295611 non-null  int64
 11  ORIGIN_AIRPORT_ID      295611 non-null  int64
 12  ORIGIN_AIRPORT_SEQ_ID  295611 non-null  int64
 13  ORIGIN_CITY_MARKET_ID  295611 non-null  int64
 14  ORIGIN                 295611 non-null  object
 15  ORIGIN_CITY_NAME       295611 non-null  object
 16  ORIGIN_STATE_ABR       295611 non-null  object
 17  ORIGIN_STATE_FIPS      295611 non-null  int64
 18  ORIGIN_STATE_NM        295611 non-null  object
 19  ORIGIN_WAC             295611 non-null  int64
 20  DEST_AIRPORT_ID        295611 non-null  int64
 21  DEST_AIRPORT_SEQ_ID    295611 non-null  int64
 22  DEST_CITY_MARKET_ID    295611 non-null  int64
 23  DEST                   295611 non-null  object
 24  DEST_CITY_NAME         295611 non-null  object
 25  DEST_STATE_ABR         295611 non-null  object
 26  DEST_STATE_FIPS        295611 non-null  int64
 27  DEST_STATE_NM          295611 non-null  object
 28  DEST_WAC               295611 non-null  int64
 29  CRS_DEP_TIME           295611 non-null  object
 30  DEP_TIME               292081 non-null  float64
 31  DEP_DELAY              292081 non-null  float64
 32  DEP_DELAY_NEW          292081 non-null  float64
 33  DEP_DEL15              292081 non-null  float64
 34  DEP_DELAY_GROUP        292081 non-null  float64
 35  DEP_TIME_BLK           295611 non-null  object
 36  TAXI_OUT               292016 non-null  float64
 37  WHEELS_OFF             292016 non-null  float64
 38  WHEELS_ON              291851 non-null  float64
 39  TAXI_IN                291851 non-null  float64
 40  CRS_ARR_TIME           295611 non-null  int64
 41  ARR_TIME               291852 non-null  float64
 42  ARR_DELAY              291219 non-null  float64
 43  ARR_DELAY_NEW          291219 non-null  float64
 44  ARR_DEL15              291219 non-null  float64
 45  ARR_DELAY_GROUP        291219 non-null  float64
 46  ARR_TIME_BLK           295611 non-null  object
 47  CRS_ELAPSED_TIME       295611 non-null  float64
 48  ACTUAL_ELAPSED_TIME    291219 non-null  float64
 49  AIR_TIME               291219 non-null  float64
 50  FLIGHTS                295611 non-null  float64
 51  DISTANCE               295611 non-null  float64
 52  DISTANCE_GROUP         295611 non-null  int64
 53  CARRIER_DELAY          39529 non-null   float64
 54  WEATHER_DELAY          39529 non-null   float64
 55  NAS_DELAY              39529 non-null   float64
 56  SECURITY_DELAY         39529 non-null   float64
 57  LATE_AIRCRAFT_DELAY    39529 non-null   float64
 58  Unnamed: 53            0 non-null       float64
 59  Unnamed: 58            0 non-null       float64
dtypes: float64(26), int64(16), object(18)
memory usage: 135.3+ MB
```

```
In [ ]: #create a new column required to join flight n weather datasets
for i in flight_data.index:
    l=len(flight_data.at[i,"CRS_DEP_TIME"])
    flight_data.at[i,"CRS_DEP_TIME"]= "0"*(4-l) + flight_data.at[i,"CRS_DEP_TIME"]
    l=len(flight_data.at[i,"MONTH"])
    flight_data.at[i,"MONTH"]= "0"*(2-l) + flight_data.at[i,"MONTH"]
    l=len(flight_data.at[i,"DAY_OF_MONTH"])
    flight_data.at[i,"DAY_OF_MONTH"]= "0"*(2-l) + flight_data.at[i,"DAY_OF_MONTH"]
    flight_data.at[i,"HOUR"]=flight_data.at[i,"CRS_DEP_TIME"][0:2]
    flight_data.at[i,"PYMDH_code"]=flight_data.at[i,"ORIGIN"] + "." + flight_data.at[i,"YEAR"] + "." +
print(flight_data["PYMDH_code"])
```

```
0         LAX.2019.07.01.10
1         LAX.2019.07.02.10
2         LAX.2019.07.03.10
3         LAX.2019.07.04.10
4         LAX.2019.07.05.10
                ...
295606    LAX.2019.12.31.13
295607    LAX.2019.12.31.10
295608    SJC.2019.12.31.08
295609    LGB.2019.12.31.06
295610    SFO.2019.12.31.20
Name: PYMDH_code, Length: 295611, dtype: object
```

```
In [ ]: flight_data=flight_data[['PYMDH_code','FL_DATE','ORIGIN_AIRPORT_ID','ORIGIN','ORIGIN_CITY_NAME','ORIGI
flight_data
```

Out[4]:

	PYMDH_code	FL_DATE	ORIGIN_AIRPORT_ID	ORIGIN	ORIGIN_CITY_NAME	ORIGIN_STATE_NM	DEST_AIRPORT_I
0	LAX.2019.07.01.10	01-07-2019	12892	LAX	Los Angeles, CA	California	103
1	LAX.2019.07.02.10	02-07-2019	12892	LAX	Los Angeles, CA	California	103
2	LAX.2019.07.03.10	03-07-2019	12892	LAX	Los Angeles, CA	California	103
3	LAX.2019.07.04.10	04-07-2019	12892	LAX	Los Angeles, CA	California	103
4	LAX.2019.07.05.10	05-07-2019	12892	LAX	Los Angeles, CA	California	103
...	...	...	...	...	...	...	
295606	LAX.2019.12.31.13	2019-12-31	12892	LAX	Los Angeles, CA	California	107
295607	LAX.2019.12.31.10	2019-12-31	12892	LAX	Los Angeles, CA	California	116
295608	SJC.2019.12.31.08	2019-12-31	14831	SJC	San Jose, CA	California	129
295609	LGB.2019.12.31.06	2019-12-31	12954	LGB	Long Beach, CA	California	148
295610	SFO.2019.12.31.20	2019-12-	14771	SFO	San Francisco, CA	California	107

```
In [ ]: import pandas as pd
import xlrd

#weather dataset
weather_data=pd.read_excel("E:\\mini project data\\Weather_Data.xlsx","weather_data")

weather_data["STATION"]=weather_data["STATION"].astype(str)

#find station names with given station id's
station_name={'72288023152':'BUR','72290023188':'SAN','72295023174':'LAX','72297023129':'LGB','7249452
weather_data["station_name"]=weather_data["STATION"].map(station_name)
print(weather_data["station_name"])
```

```
0        SAN
1        SAN
2        SAN
3        SAN
4        SAN
        ...
28734    SJC
28735    SJC
28736    SJC
28737    SJC
28738    SJC
Name: station_name, Length: 28739, dtype: object
```

In [ ]:
```
#Create a new column required to join flight and weather datasets
for i in weather_data.index:
    weather_data.at[i,"MINUTE"]=weather_data.at[i,"DATE"][14:16]
    weather_data.at[i,"PYMDH_code"]= weather_data.at[i,"station_name"]+"."+weather_data.at[i,"DATE"][0:
print(weather_data["PYMDH_code"])
```

```
0        SAN.2019.07.01.00
1        SAN.2019.07.01.00
2        SAN.2019.07.01.01
3        SAN.2019.07.01.01
4        SAN.2019.07.01.02
               ...
28734    SJC.2019.12.31.21
28735    SJC.2019.12.31.22
28736    SJC.2019.12.31.23
28737    SJC.2019.12.31.23
28738    SJC.2019.12.31.23
Name: PYMDH_code, Length: 28739, dtype: object
```

In [ ]:
```
import openpyxl

#Handling missing values in weather dataset

weather_data.sort_values(by=['PYMDH_code','MINUTE'],inplace= True)
weather_data.reset_index(inplace=True,drop=True)
weather_data=weather_data[['PYMDH_code','STATION','station_name','DATE','MINUTE','HourlyAltimeterSettin
weather_data.ffill(axis=0,inplace=True)

weather_data=weather_data.groupby('PYMDH_code').agg({'HourlyAltimeterSetting':np.mean,'HourlyDewPointTe
weather_data
```

Out[ ]:

PYMDH_code	HourlyAltimeterSetting	HourlyDewPointTemperature	HourlyDryBulbTemperature	HourlyRelativeHumidity	Hour
BUR.2019.07.01.00	29.96	55.0	66.0	68.0	
BUR.2019.07.01.01	29.95	55.0	65.0	70.0	
BUR.2019.07.01.02	29.95	56.0	65.0	73.0	
BUR.2019.07.01.03	29.95	56.0	63.0	78.0	
BUR.2019.07.01.04	29.96	54.0	64.0	70.0	
...	...	...	...	...	
SJC.2019.12.31.19	30.14	40.0	51.0	66.0	
SJC.2019.12.31.20	30.15	40.0	51.0	66.0	
SJC.2019.12.31.21	30.16	40.0	50.0	68.5	
SJC.2019.12.31.22	30.17	40.0	49.0	71.0	
SJC.2019.12.31.23	30.13	39.0	45.0	80.0	

22397 rows × 8 columns

In [ ]:
```
#Join wethaer and flight datasets
final_data=pd.merge(flight_data,weather_data,on='PYMDH_code')
final_data.to_excel("final.xlsx")
final_data
```

Out[ ]:

	PYMDH_code	FL_DATE	ORIGIN_AIRPORT_ID	ORIGIN	ORIGIN_CITY_NAME	ORIGIN_STATE_NM	DEST_AIRPORT_
0	LAX.2019.07.01.10	01-07-2019	12892	LAX	Los Angeles, CA	California	10
1	LAX.2019.07.01.10	01-07-2019	12892	LAX	Los Angeles, CA	California	12
2	LAX.2019.07.01.10	01-07-2019	12892	LAX	Los Angeles, CA	California	14
3	LAX.2019.07.01.10	01-07-2019	12892	LAX	Los Angeles, CA	California	14
4	LAX.2019.07.01.10	01-07-2019	12892	LAX	Los Angeles, CA	California	11
...	...	...	...	...	...	...	
215729	LGB.2019.12.30.17	2019-12-30	12954	LGB	Long Beach, CA	California	14
215730	LGB.2019.12.31.21	2019-12-31	12954	LGB	Long Beach, CA	California	12
215731	LGB.2019.12.31.21	2019-12-31	12954	LGB	Long Beach, CA	California	10
215732	BUR.2019.12.31.21	2019-12-31	10800	BUR	Burbank, CA	California	12
215733	SJC.2019.12.31.22	2019-12-31	14831	SJC	San Jose, CA	California	10

215734 rows × 30 columns

In [ ]:

```
#handling missing values in final dataset

final_data.dropna(inplace=True)
final_data=final_data.sample(frac=1,axis=0)
final_data.reset_index(drop=True,inplace=True)

final_data.info()
final_data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 212887 entries, 0 to 212886
Data columns (total 30 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   PYMDH_code               212887 non-null  object
 1   FL_DATE                  212887 non-null  object
 2   ORIGIN_AIRPORT_ID        212887 non-null  int64
 3   ORIGIN                   212887 non-null  object
 4   ORIGIN_CITY_NAME         212887 non-null  object
 5   ORIGIN_STATE_NM          212887 non-null  object
 6   DEST_AIRPORT_ID          212887 non-null  int64
 7   DEST                     212887 non-null  object
 8   DEST_CITY_NAME           212887 non-null  object
 9   DEST_STATE_NM            212887 non-null  object
 10  CRS_DEP_TIME             212887 non-null  object
 11  DEP_TIME                 212887 non-null  float64
 12  DEP_DELAY                212887 non-null  float64
 13  DEP_DELAY_NEW            212887 non-null  float64
 14  DEP_DEL15                212887 non-null  float64
 15  CRS_ARR_TIME             212887 non-null  int64
 16  ARR_TIME                 212887 non-null  float64
 17  ARR_DELAY                212887 non-null  float64
 18  ARR_DELAY_NEW            212887 non-null  float64
 19  ARR_DEL15                212887 non-null  float64
 20  FLIGHTS                  212887 non-null  float64
 21  DISTANCE                 212887 non-null  float64
 22  HourlyAltimeterSetting   212887 non-null  float64
 23  HourlyDewPointTemperature  212887 non-null  float64
 24  HourlyDryBulbTemperature  212887 non-null  float64
 25  HourlyRelativeHumidity   212887 non-null  float64
 26  HourlyStationPressure    212887 non-null  float64
 27  HourlyVisibility         212887 non-null  float64
 28  HourlyWetBulbTemperature  212887 non-null  float64
 29  HourlySeaLevelPressure   212887 non-null  float64
dtypes: float64(18), int64(3), object(9)
memory usage: 48.7+ MB
```

Out[ ]:

	PYMDH_code	FL_DATE	ORIGIN_AIRPORT_ID	ORIGIN	ORIGIN_CITY_NAME	ORIGIN_STATE_NM	DEST_AIRPORT_
0	LAX.2019.07.13.20	13-07-2019	12892	LAX	Los Angeles, CA	California	147
1	SFO.2019.12.29.22	2019-12-29	14771	SFO	San Francisco, CA	California	107
2	LAX.2019.12.03.22	2019-12-03	12892	LAX	Los Angeles, CA	California	146
3	SJC.2019.08.09.14	2019-08-09	14831	SJC	San Jose, CA	California	108
4	LGB.2019.07.02.15	02-07-2019	12954	LGB	Long Beach, CA	California	148
...	...	...	...	...	...	...	
212882	LGB.2019.09.09.17	2019-09-09	12954	LGB	Long Beach, CA	California	147
212883	BUR.2019.08.25.16	2019-08-25	10800	BUR	Burbank, CA	California	148
212884	LAX.2019.12.13.10	2019-12-13	12892	LAX	Los Angeles, CA	California	137
212885	LGB.2019.11.08.17	2019-11-08	12954	LGB	Long Beach, CA	California	128
212886	SAN.2019.11.05.11	2019-11-05	14679	SAN	San Diego, CA	California	112

212887 rows × 30 columns

In [ ]:

```
final_data['DEP_DEL15']=final_data['DEP_DEL15'].astype(bool)
final_data['ARR_DEL15']=final_data['ARR_DEL15'].astype(bool)
final_data.to_excel("final_data.xlsx")

#splitting the dataset for training and testing
from sklearn.model_selection import train_test_split

X=final_data[['DEP_DELAY','DEP_DEL15','DISTANCE','HourlyAltimeterSetting','HourlyDewPointTemperature','
Y=final_data[['ARR_DEL15','ARR_DELAY']]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.60, random_state=42)
X_train.reset_index(drop=True,inplace=True)
X_test.reset_index(drop=True,inplace=True)
Y_train.reset_index(drop=True,inplace=True)
Y_test.reset_index(drop=True,inplace=True)
CY_train=Y_train[['ARR_DEL15']]
CY_test=Y_test[['ARR_DEL15']]
RY_train=Y_train[['ARR_DELAY']]
RY_test=Y_test[['ARR_DELAY']]
```

In [ ]:

```
#Training the model for classification task

import xgboost as xgb
boost =  xgb.XGBClassifier(
 learning_rate = .1,
 n_estimators=459,
 max_depth=5,
 min_child_weight=1,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27).fit(X_train, CY_train)
Y_pred=boost.predict(X_test)

Y_pred=pd.DataFrame(Y_pred)
Y_pred
```

Out[ ]:

	0
0	False
1	False
2	False
3	False
4	False
...	...
127728	True
127729	False
127730	False
127731	False
127732	False

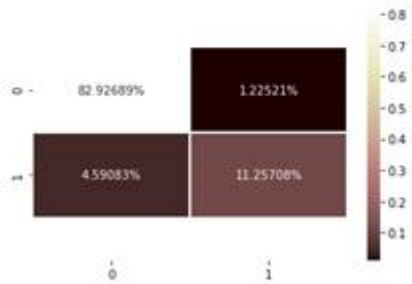
127733 rows × 1 columns

In [ ]:

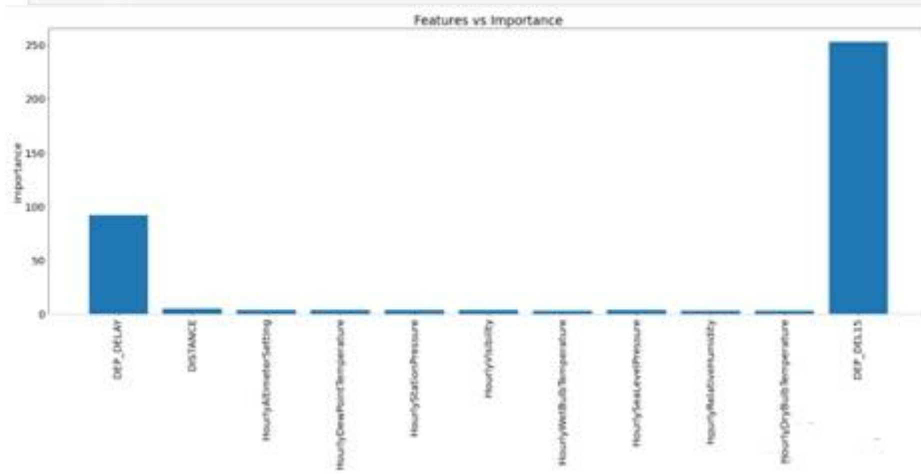
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score

#performance metrics for classification model

c_m=confusion_matrix(CY_test,Y_pred)
a_s=accuracy_score(CY_test,Y_pred)
p=precision_score(CY_test,Y_pred)
r=recall_score(CY_test,Y_pred)
f=f1_score(CY_test,Y_pred)
c_r=classification_report(CY_test,Y_pred)
auc=roc_auc_score(CY_test,Y_pred)

print("Performance metrics for classfication model\n-------------------------------------- ")
print("confusion matrix:",c_m,sep='\n')
print("accuracy score:",a_s)
print("precision:",p)
print("recall:",r)
print("f1_score:",f)
print("auc score:",auc)
print("classification report:",c_r,sep='\n')
```

```
Performance metrics for classfication model
--------------------------------------
confusion matrix:
[[105925   1565]
 [  5864  14379]]
accuracy score: 0.9418396185793805
precision: 0.9018439538384345
recall: 0.71031961665761
f1_score: 0.7947052809019814
auc score: 0.8478800613756001
classification report:
              precision    recall  f1-score   support

       False       0.95      0.99      0.97    107490
        True       0.90      0.71      0.79     20243

    accuracy                           0.94    127733
   macro avg       0.92      0.85      0.88    127733
weighted avg       0.94      0.94      0.94    127733
```

```
In [ ]: import matplotlib.pylab as plt
        import seaborn as sns
        ax=sns.heatmap(c_m/np.sum(c_m),annot=True ,linewidth=.5,
                    fmt='.5%', cmap='pink')
        bottom, top = ax.get_ylim()
        ax.set_ylim(bottom + 0.5, top - 0.5)

Out[ ]: (2.5, -0.5)
```




```
In [ ]: d=boost.get_booster().get_score(importance_type= 'gain')
        l=d.items()
        x=[]
        y=[]
        for i in l:
            x.append(i[0])
            y.append(i[1])

        from matplotlib import pyplot as plt
        x_pos = np.arange(len(x))
        plt.rcParams['font.size']=20
        plt.figure(figsize=(30,10))
        plt.bar(x_pos, y , align='center', alpha=1)
        plt.xticks(x_pos, x,rotation=90)
        plt.ylabel('Importance')
        plt.title('Features vs Importance')

        plt.show()
```



```
In [ ]: RX_train=pd.concat([X_train,CY_train],axis=1)
        RX_test=pd.concat([X_test,Y_pred],axis=1)
        RSX_test=pd.concat([X_test,CY_test],axis=1)

        RX_test.rename(columns={0:'ARR_DEL15'},inplace=True)

        #Training the model for regression task
        from sklearn.linear_model import LinearRegression
        from sklearn.linear_model import Ridge
        from sklearn.linear_model import ElasticNet

        #model = Lasso()
        #model = Ridge()
        #model=ElasticNet()
        model=LinearRegression()
        model.fit(RX_train,RY_train)

        RY_pred=model.predict(RX_test)
        RSY_pred=model.predict(RSX_test)
```

```
In [ ]: from sklearn.metrics import mean_absolute_error
        from sklearn.metrics import mean_squared_error
        from sklearn.metrics import mean_squared_log_error
        from sklearn.metrics import r2_score
        import math as m

        #performance metrics for regression model
        smae=mean_absolute_error(RY_test, RSY_pred)
        smse=mean_squared_error(RY_test, RSY_pred)
        sr2s=r2_score(RY_test, RSY_pred)

        #performance metrics for regression model
        mae=mean_absolute_error(RY_test, RY_pred)
        mse=mean_squared_error(RY_test, RY_pred)
        r2s=r2_score(RY_test, RY_pred)

        print("Performance metrics for regression model\n------------------------------------------")
        print("mean absolute error:",smae)
        print("root mean squared error:",m.sqrt(smse))
        print("r2-score:",sr2s)
        print()

        print("Performance metrics for whole model\n------------------------------------")

        print("mean absolute error:",mae)
        print("root mean squared error:",m.sqrt(mse))
        print("r2-score:",r2s)

        Performance metrics for regression model
        ------------------------------------------
        mean absolute error: 8.058008166944312
        root mean squared error: 10.770722923658475
        r2-score: 0.9303379872773441

        Performance metrics for whole model
        ------------------------------------
        mean absolute error: 9.123398426216875
        root mean squared error: 12.709693337978122
        r2-score: 0.9029989457371684
```

8. TESTING AND RESULTS

8.1 Result of Classification model:

accuracy score: 94.1%

precision: 90.1 %

recall: 71.0%

f1 score: 79.4%

auc score: 84.7%

confusion matrix:

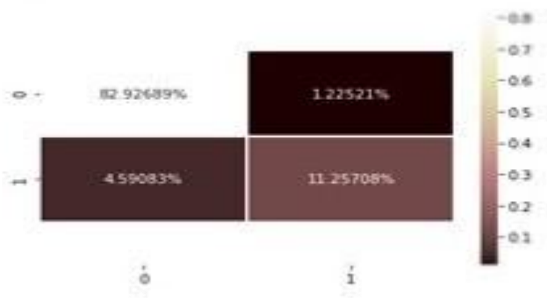


Fig 8.1: Confusion matrix

8.2 Result of Regression model:

mean absolute error: 8.05 mins

root mean square error: 10.77 mins

r2-score: 0.93

8.3 Result of Overall model:

mean absolute error: 9.12 mins

root mean square error: 12.70 mins

r2-score: 0.90

9. CONCLUSION

9.1 Conclusion

This project aims to predict the flights delay along with the estimation of delay time in minutes using machine learning algorithms namely Decision Tree Algorithm (XGBoost) and Linear regression. Data set of both flight data and weather data will be taken to compare with the given inputs and validate them by applying classification and Regression concepts of Machine Learning. We have also done feature extraction, handling missing values using appropriate methods, sampling in order to handle imbalanced data and also tuning the hyper parameters with which we were able to achieve better accuracy

9.2 Future Scope

In future, we would like to enhance the application by predicting the flight delays considering more factors like heavy air traffic, security etc... and deploy this model in real time.

10. BIBILIOGRAPHY

10.1 References

[1] https://www.transtats.bts.gov

[2] https://www.ncdc.noaa.gov

[3] https://pandas.pydata.org/

[4] https://scikit-learn.org/stable/

[5] https://xgboost.readthedocs.io/en/latest/python/index.html

[6] https://matplotlib.org/

[7] https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/