A

Major Project

On

Predicting Flight Delays using Machine Learning

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

Tushar Patel (177R1A0558)

Chintala Chereesh (17601A0529)

Nitish Singh (177R1A0542)

Under the Guidance of

DR. G. SOMASEKHAR

(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGC Act.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2017-21

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled “**Predicting flight delays using machine learning**” being submitted by **Tushar Patel** (**177R1A0558**), **Chintala Chereesh (17601A0529) & Nitish Singh (177R1A0542)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2020-21.

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

DR. G. SOMASEKHAR
Assocaite Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HoD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on

ACKNOWLEGDEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We take this opportunity to express my profound gratitude and deep regard to my guide

Dr. G. Somasekhar, Associate Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) Coordinators: **Mr. J. Narasimha Rao, Mr. B. P. Deepak Kumar, Mr. K. Murali, Dr. Suwarna Gothane** and **Mr. B. Ramji** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju**, Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. We would like to express our sincere gratitude to **Sri. Ch. Gopal Reddy**, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

Tushar Patel (177R1A0558)

Chintala Chereesh (17601A0529)

Nitish Singh (177R1A0542)

ABSTRACT

Flight delay is a major problem in the aviation sector. During the last two decades, the growth of the aviation sector has caused air traffic congestion, which has caused flight delays. Flight delays result not only in the loss of fortune also negatively impact the environment. Flight delays also cause significant losses for airlines operating commercial flights. Therefore, they do everything possible in the prevention or avoidance of delays and cancellations of flights by taking some measures. In this paper, using machine learning models such as Logistic Regression, Decision Tree Regression, Bayesian Ridge, Random Forest Regression and Gradient Boosting Regression we predict whether the arrival of a particular flight will be delayed or not. Many attempts have been by researchers in the past for predicting flight delays using Machine Learning, Deep Learning and Big Data approaches. Kalliguddi(author) constructed regression models like Decision Tree Regressor, Random Forest regressor on flight data for predicting both departure and arrival delays. The main issues are to find the error rate in terms of predictions and reducing the error factor in the model

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

The future scope of this paper can include the application of more advanced, modern and innovative pre-processing techniques, automated hybrid learning and sampling algorithms, and deep learning models adjusted to achieve better performance. To evolve a predictive model, additional variables can be introduced. In this paper we used data from the US only, therefore in future, the model can be trained with data from other countries as well.

1.2 PROJECT PURPOSE

Accurate flight delay prediction is fundamental to establish the more efficient airline business. Recent studies have been focused on applying machine learning methods to predict the flight delay. Most of the previous prediction methods are conducted in a single route or airport. Scope of factors which may potentially influence the flight delay, and compares several machine learning-based models in designed generalized flight delay prediction tasks. The first thing to look at is what pieces contribute to flight delays. We all know a severe storm halts travel plans, and our tools are pretty good at predicting them. What we are concerned with are the little ripples in airline traffic that create smaller surprise delays.

1.3 PROJECT FEATURES

The prediction, analysis and cause of flight delays have been a major problem for air traffic control, decision making by airlines and ground delay response programs. Studies are conducted on the delay propagation of the sequence. Also, studying the predictive model of arrival delay and departure delay with meteorological features is encouraged.

The growing aviation industry has resulted in air-traffic causing flight delays. Flight delays have huge economic impact on airlines and also have harmful environmental effects. Therefore, it is important to detect the major factors influencing flight delay. The factors influencing the flight delay range from natural factors like weather to factors like day, month, etc. On observing wide variety of flight data, a list of factors responsible for the delay was generated. There are other factors influencing the delay as well, but their scope is limited. This list primarily categorizes delay into two types, namely departure delay and arrival delay. Some of these features influence the flight delay drastically while others have a minor impact. In order to develop an efficient flight delay prediction system, these features along with their degree of impact on the delay must be taken into consideration.

For all airlines, flight delays represent the source of financial and technical difficulties. This article aims both to identify and analyse the factors causing delays and to suggest some possibilities on how to eliminate these delays. We have chosen an airline operating in the European region and used its data from the period 2008–2014. The database includes information on departures from most major European airports. The so called IATA1 Airlines delay codes, see Tab. I, have been used for the primary classification of delays. The differentiation of flights goes as follows: regular (J), charter (C), and empty (P). The causes of delays obtained directly from the airline were confronted with further factors derived from the data (the time of day, the months selected, the length of the delay, the type of flight and type of aircraft). The article by Campanelli et al. (2014) provides an analysis of the chain effect of flight delays. The article is dealing with an air transport system behaving in nonlinear manner which is difficult to anticipate. Similarly, the paper by Rebollo, Balakrishnan (2014) introduces models for predicting air traffic delays. The work of Ionescu et al. (2016) suggests that flight plans often fail to respond to the possibility of delays associated with unforeseen events, late reporting following a technical malfunction, or congestion of airports or air space. The proposed models improve the chance to predict the potential delays. A similar analysis of factors causing flight delays at the Czech international airports was performed in the article by Zámková, Prokop (2015b).

The paper by Zámková, Prokop (2015a) focused on financial impacts of delayed flights on airlines. Zámková, Prokop (2016) later focused on factors influencing flight delays, but their analysis was limited to the airports located near Czech tourists' favourite destinations only. Flight delay is studied vigorously in various research in recent years. The growing demand for air travel has led to an increase in flight delays. According to the Federal Aviation Administration (FAA), the aviation industry loses more than $3 billion in a year due to flight delays and, as per BTS, in 2016 there were 860,646 arrival delays. The reasons for the delay of commercial scheduled flights are air traffic congestion, passengers increasing per year, maintenance and safety problems, adverse weather conditions, the late arrival of plane to be used for next flight. In the United States, the FAA believes that a flight is delayed when the scheduled and actual arrival times differs by more than 15 minutes. Since it becomes a serious problem in the United States, analysis and prediction of flight delays are being studied to reduce large costs.

In recent years, flight delay problem blocks the development of the civil aviation industry all over the world. And delay propagation always is a main factor that impacts the flight’s delay.

All kinds of delays often happen in nearly-saturated or overloaded airports. Factors such as distance covered by an aircraft, the day of the flight and the scheduled departure time were considered when developing a multiple regression model to predict flight delays (Burgauer & Peters, 2000). In that study, those factors were found key in predicting flight delays. Time duration and number of airports were used in developing probability models for flight delays (Mueller & Chatterji, 2002). The probability models developed using these factors were the normal distribution and the poisson distribution. An econometric model that was used to study flight delays used various variables such asairline and airport of destination, frequency, aircraft size, occupancy rate and fare(Hsiao & Hansen, 2006). This model was used to make a study on flight cancellation. A correlation analysis has been done between flight delays and capacities of airports (Isonet al., 2015). The results from this analysis indicated that a correlation exists between flight delays and airport capacities.

However, these results called for further investigation since it was not shown with certainty that one variable caused another one to happen. There was a possibility of another variable that could have caused the correlation. Factors such as the airport capacity, departure delays and arrival delays have been used to model the cost of a delayed flight using advanced analytical techniques in operation research such as simulations (Zou & Hansen, 2012). In a study by Kalliguddi & Leboulluec (2017), predictive modelling of aircraft delay was done using variables such as, national air system delay, departure delay, carrier delay, taxi in and taxi out, weather delay, late aircraft delay, distance and security delay. Lawson & Castillo, (2012) used a dataset of flights and included several years, which resulted in an impressive number of data points. They limited their features to weather data only (33 feat ures only in the end) obtaining 40% recall only. A study by Sridhar et al. (2009)focused on weather related delays but used more sophisticated features such as key airspace metrics and also took into account the delay of previous flights to predict a new delay. Predicting flight on-time performance was done using predictive features such as: the previous aircraft arriving late, weather, and departure time and achieved an average F1 score of 58.7%. (Arjounet al., 2013). Bandyopadhyay & Guerrero (2012) I predicted airline delays usingflight departure times, and weather conditions. Their prediction algorithms achieved F1 score of 56.6%. This clearly shows that for different studies, different factors will be used for predictive modelling. This will mainly be influenced by the data collected from the airlines and airports.

Accurate flight delay prediction is fundamental to establish the more efficient airline business. Recent studies have been focused on applying machine learning methods to predict the flight delay. Most of the previous prediction methods are conducted in a single route or airport. Scope of factors which may potentially influence the flight delay, and compares several machine learning-based models in designed generalized flight delay prediction tasks

2. LITERATURE SURVEY

2.LITERATURE SURVEY

2.1 RELATED WORK

Much research has been done on studying flight delays. The prediction, analysis and cause of flight delays have been a major problem for air traffic control, decision-making by airlines and ground delay response programs. Studies are conducted on the delay propagation of the sequence. Also, studying the predictive model of arrival delay and departure delay with meteorological features is encouraged. In the past, researchers have tried to predict flight delays with Machine Learning. Chakrabarty et al. used supervised automatic learning algorithms (random forest, Gradient Boosting Classifier, Support Vector Machine and the k-nearest neighbour algorithm) to predict delays in the arrival of operated flights including the five busiest US airports. The maximum precision achieved was 79.7% with gradient booster as a classifier with a limited data set. Choi et al. applied machine learning algorithms like decision tree, random forest, AdaBoost and kNearest Neighbours to predict delays on individual flights. Flight schedule data and weather forecasts have been incorporated into the model. Sampling techniques were used to balance the data and it was observed that the accuracy of the classifier trained without sampling was more that of the trained classifier with sampling techniques. Cao et al. used a Bayesian Network model to analyse the turnaround time of a flight and delay prediction.

Juan José Rebollo and Hamsa Balakrishnan used a hundred pairs of origin and destination to summarise the result of various regression and classification models. The find outs reveal that among all the methods used, random forest has the highest performance. However, predictability may additionally range because of factors such as the number of origin destination pairs and the forecast horizon. Sruti Oza, Somya Sharma used multiple linear regression to predict weather induced flight delays in flight-data, as well as climatic factors and probabilities due to weather delays. The forecasts were based on some key attributes, such as carrier, departure time, arrival time, origin and destination. Anish M. Kalliguddi and Aera K. Leboulluec predicted both departure and arrival delays using regression models such as Decision Tree

Regressor, Multiple Linear Regression and Random Forest Regressor in flight-data. It has been observed that the longer forecast horizon is useful for increasing the accuracy with a minimum forecast error for random forests. Etani J Big Data A supervised model of on-schedule arrival fight is used using weather data and flight data. The relationship between flight data and pressure patterns of Peach Aviation is found. On-Schedule arrival flight is predicted with 77% accuracy using Random Forest as a Classifier.

Flight operations translate into complex problems to solve, high-lighted by a myriad of possible scenarios with many internal (e.g.,traffic management) and external variables (e.g., weather). The studied problems include planning offlight routes (Amin-Naseri, Yazdekhasti,& Salmasnia, 2018), detecting flight trajectory anomalies (Di Ciccio,Van der Aa, Cabanillas, Mendling, & Prescher, 2016), air traffic management (Wandelt & Sun, 2014), recognizing aircraft events such as landing (Zhang & Chandrasekar, 2017), among others. Given the chained nature of air transportation business processes, most of those problems are interconnected, making it challenging to narrow the re-search focus in a specific problem. The challenges inherent of such complexity have led researchers to devote efforts to adopting state-of-the-art techniques such as modern optimisation and data mining to address them.The departure delay prediction of aflight is a widely studied problem in air transportation literature (Rebollo & Balakrishnan, 2014).The operational impact offlight delays is enormous, resulting in economic losses estimated by Zou and Hansen (Zou & Hansen, 2012)in$7.1–13.5 billion in 2007 for US airlines when comparing the ideal operational performance to the real one. Ball et al. (Ball et al., 2010)corroborate that range by estimating a value of around $8.3 billion toairline losses. Flight delays affect passengers, causing distress, and consequently undermining the airline brand image. Although airlines attempt to compensate for the delays, travellers are unable to perceive the different services offered by the competition, which does not favour brand loyalty, with travellers often choosing to shift to another airline(Torlak, Sevkli, Sanal, & Zaim, 2011).

Delays have a more emphasised impact on high-income managers who frequently need to air travel,with airlines being especially concerned to satisfy this type of travellers(Pai, 2010). Besides passengers, delays can also cause labour disruption by affecting personnel work period, especially in the air transportation business, where there is strict legislation regarding flying times and off-times (Clausen, Larsen, Larsen, & Rezanova, 2010). Also, as it was previously stressed, the propagation effect of delays to the daily airline and airport operation can cause otherflights' delay which is difficult to overcome since resources are optimised to reduce slacks (Kohl et al.,2007; Wu & Law, 2019;Kafle & Zou, 2016). Another critical issue re-lated to delay propagation is the air transportation network (Wandelt,Sun, & Zhang, 2019). The complexity of such networks implies that awide range of external factors (i.e., not related to the airline) can affect directly or indirectly flights, leading to delays (Hansen & Zou, 2013). AsSun et al. (Sun, Wandelt, & Cao, 2017) pointed out, country networks are different between countries, with some of them being more influenced by the passenger traffic of their neighbourhood countries.Table 1summarises four articles focusing on predicting flight departure delays. The first noteworthy common characteristic is the fact that all four studies used the US publicly available data. This data was supplied by the U.S. Department of Transportation (Tu et al., 2008), the Federal Aviation Administration - FAA (Balakrishna, Ganesan, &Sherry, 2010; Rebollo & Balakrishnan, 2014), or the Bureau of Trans-portation Statistics - BTS (Choi et al., 2016). Table 1stresses out (1) the open data policy of the US authorities (Huijboom & Broek, 2011), and(2) the need to study other countries' airspace and air transportation operations. Some studies aim to predict the delay time value, thus making of it a regression problem, although most of them are focused inpredicting if the flight is delayed or not, therefore building a classifier(Witten, Frank, Hall, & Pal, 2016). The latter poses the challenge of defining the threshold above which the flight is considered delayed.Rebollo and Balakrishnan (Rebollo & Balakrishnan, 2014) considered a flight delayed if departures are 60 min after schedule, while Choi et al.(Choi et al., 2016) considered 15 min. However, the former authors stated that the US Department of Transportation“only counts a flight as delayed if it incurs a delay of more than 15 min”(Rebollo &Balakrishnan, 2014). Moreover, literature acknowledges the 15 min delay published by the US Department of Transportation as a standard threshold definition (Lan, Clarke, & Barnhart, 2006; Zou& Hansen,2012). Furthermore, in Europe, the Eurocontrol (2018)states that an“aircraft should takeoff within 15 minutes of the time stated in itsflight plan”.

Managers at the European-based airline used for the em-pirical research also provided support for the 15 min definition which is currently used in the aviation business.The results achieved by the studies mentioned in Table 1are not directly comparable since the cases are different (i.e., different time-frame, airports, and features considered). Also, the goals do not precisely match, with Balakrishna et al. (Balakrishna et al., 2010) focusing specifically on taxi-out (a likely reason for better accuracy than there maining), while the others aimed to predict departure delay in general, although using different metrics for evaluating the results. The two most recent studies (Choi et al., 2016;Rebollo & Balakrishnan, 2014)achieved a similar accuracy of 81%, although considering different features. In fact, apart from Tu et al. (Tu et al., 2008), the remaining are too narrow in the features used, focusing on understanding a specific phenomenon (e.g., weather influence; or the connection - origin destiny airport - influence). Also, most studies are using the features directly obtained from a single or two public datasets, without attempting toN. Fernandes, et al.Research in Transportation Business & Management unveil other interesting features that may delays. Thus,the focus is on improving the model's accuracy by testing different models (Choi et al., 2016). Finally, while predicting departure delay is a challenging task, it would be interesting to get insights from the most accurate models on the features' contribution to the delay. Such knowledge may help air transportation managers to understand this complex problem. However, from the studies analysed, only Rebolloand Balakrishnan (Rebollo & Balakrishnan, 2014) provided and dis-cussed their findings regarding understanding features' contribution.

The first thing to look at is what pieces contribute to flight delays. We all know a severe storm halts travel plans, and our tools are pretty good at predicting them. What we are concerned with are the little ripples in airline traffic that create smaller surprise delays. Factors recorded in our data set are departure time, taxi out, taxi in, arrival time, cancellations, diversions, distance, weather delays, and security delays just to name a few.

Once we isolated which pieces of data to use we could start identifying and visualizing correlations. Logically, we expect departure time and arrival time to have a strong correlation along with distance and air time as well. What was most interesting is the shape of a departure delay versus a late-arriving aircraft. This shows that not all late departures result in a late arrival.

We now have an idea of how the features in the data work together. My hypothesis is that late flights arriving will cause a reverse ripple effect measured over time in late departures at the destination airport. Logically this makes sense because if a Boeing 737 is booked to leave Chicago at 8:00 AM Central and arrive in Miami at 11:32 AM Eastern, and is delayed by security for 20 minutes, the subsequent flight that aircraft is has been booked for out of Miami will be affected by this delay. I suspect the arc in the scatter plot above is created due to various countermeasures the airlines and ATC employ to reduce the reverse ripple effect. If an early aircraft takes the flight that the late one could not make due to a delay, our delays become reduced and hard to track. This is where measuring a local delay ratio comes into play.

The delay ratio is calculated by summing all the flights that have been delayed at the origin, and dividing by the total number of flights made at the origin. The trick is narrowing your scope by location and time. Doing so produces meaningful measurement that does not generalize too much.

Our work benefits from considering as many factors as possible that may potentially influence the flight delay. For instance, airports information, weather of airports, traffic flow of airports, traffic flow of routes. The contributions of this paper can be summarized as follows:

We explore a broader scope of factors which may potentially influence the flight delay and quantize those selected factors. Thus we obtain an integrated aviation dataset. Our experimental results indicate that the multiple factors can be effectively used to predict whether a flight will
delay.

Several machine learning based-network architectures are proposed and are matched with the established aviation dataset. Traditional flight prediction problem is a binary classification task. To comprehensively evaluate the performance of the architectures, several prediction tasks covering classification and regression are designed. Conventional schemes mostly focused on a single route or a single airport. However, our work covers all routes and airports

2.2 Python

Python is a programming language, which means it'a a language both people and computers can understand. Python was developed by a Dutch software engineer named Guido van Rossum, who created the language to solve some problems he saw in computer languages of the time. Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation.

One significant advantage of learning Python is that it’s a general-purpose language that can be applied in a large variety of projects. Below are just some of the most common fields where Python has found its use:

- Data science
- Scientific and mathematical computing
- Web development
- Computer graphics
- Basic game development
- Mapping and geography (GIS software)

Python’s ecosystem is growing over the years and it’s more and more capable of the statistical analysis. It’s the best compromise between scale and sophistication (in terms od data processing). Python emphasizes productivity and readability. Python is used by programmers that want to delve into data analysis or apply statistical techniques (and by devs that turn to data science) There are plenty of Python scientific packages for data visualization, machine learning, natural language processing, complex data analysis and more. All of these factors make Python a great tool for scientific computing and a solid alternative for commercial packages such as MatLab. The most popular libraries and tools for data science are:

Pandas: a library for data manipulation and analysis. The library provides data structures and operations for manipulating numerical tables and time series.

NumPy: the fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

SciPy: a library used by scientists, analysts, and engineers doing scientific computing and technical computing.

Being a free, cross-platform, general-purpose and high-level programming language, Python has been widely adopted by the scientific community. Scientists value Python for its precise and efficient syntax, relatively flat learning curve and the fact that it integrates well with other languages (e.g. C/C++). As a result of this popularity there are plenty of Python scientific packages for data visualization, machine learning, natural language processing, complex data analysis and more. All of these factors make Python a great tool for scientific computing and a solid alternative for commercial packages such as MatLab.

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 allows you to generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, and more.

NumPy : NumPy is the fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

NetworkX : NetworkX is a library for studying graphs which helps you create, manipulate, and study the structure, dynamics, and functions of complex networks.

TomoPy : TomoPy is an open-sourced Python toolbox to perform tomographic data processing and image reconstruction tasks. TomoPy provides a collaborative framework for the analysis of synchrotron tomographic data with the goal to unify the effort of different facilities and beamlines performing similar tasks.

Theano : Theano is a numerical computation Python library. Theano allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

SymPy : SymPy is a library for symbolic computation and includes features ranging from basic symbolic arithmetic to calculus, algebra, discrete mathematics and quantum physics. It provides computer algebra capabilities either as a standalone application, as a library to other applications, or live on the web.

SciPy : SciPy is a library used by scientists, analysts, and engineers doing scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

Scikit-learn : Scikit-learn is a machine learning library. 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.

Scikit-image : Scikit-image is a image processing library. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more.

ScientificPython : ScientificPython is a collection of modules for scientific computing. It contains support for geometry, mathematical functions, statistics, physical units, IO, visualization, and parallelization.

SageMath : SageMath is mathematical software with features covering many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus. SageMath uses the Python, supporting procedural, functional and object-oriented constructs.

Veusz : Veusz is a scientific plotting and graphing package designed to produce publication-quality plots in popular vector formats, including PDF, PostScript and SVG.

Graph-tool : Graph-tool is a module for the manipulation and statistical analysis of graphs.

SunPy : SunPy is a data-analysis environment specializing in providing the software necessary to analyze solar and heliospheric data in Python.

Bokeh : Bokeh is a Python interactive visualization library that targets modern web browsers for presentation. Bokeh can help anyone who would like to quickly and easily create interactive plots, dashboards, and data applications. Its goal is to provide elegant, concise construction of novel graphics in the style of D3.js, but also deliver this capability with high-performance interactivity over very large or streaming datasets.

TensorFlow : TensorFlow is an open source software library for machine learning across a range of tasks, developed by Google to meet their needs for systems capable of building and training neural networks to detect and decipher patterns and correlations, analogous to the learning and reasoning which humans use. It is currently used for both research and production at Google products, often replacing the role of its closed-source predecessor, DistBelief.

Nilearn : Nilearn is a Python module for fast and easy statistical learning on NeuroImaging data. Nilearn makes it easy to use many advanced machine learning, pattern recognition and multivariate statistical techniques on neuroimaging data for applications such as MVPA (Mutli-Voxel Pattern Analysis), decoding, predictive modelling, functional connectivity, brain parcellations, connectomes.

Dmelt : DataMelt, or DMelt, is a software for numeric computation, statistics, analysis of large data volumes ("big data") and scientific visualization. The program can be used in many areas, such as natural sciences, engineering, modeling and analysis of financial markets. DMelt can be used with several scripting languages including Python/Jython, BeanShell, Groovy, Ruby, as well as with Java.

Python-weka-wrapper : Weka is a suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions. The python-weka-wrapper package makes it easy to run Weka algorithms and filters from within Python.

Dask : Dask is a flexible parallel computing library for analytic computing composed of two components: 1) dynamic task scheduling optimized for computation, optimized for interactive computational workloads, and 2) Big Data collections like parallel arrays, dataframes, and lists that extend common interfaces like NumPy, Pandas, or Python iterators to larger-than-memory or distributed environments.

Python Saves Time : Even the classic “Hello, world” program illustrates this point:

```
print("Hello, world")
```

For comparison, this is what the same program looks like in Java:

```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, world");
   }
}
```

3. SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

SYSTEM ANALYSIS

The prediction, analysis and cause of flight delays have been a major problem for air traffic control, decision making by airlines and ground delay response programs. Studies are conducted on the delay propagation of the sequence. Also, studying the predictive model of arrival delay and departure delay with meteorological features is encouraged.

3.1 PROBLEM DEFINITION

The growing aviation industry has resulted in air-traffic causing flight delays. Flight delays have huge economic impact on airlines and also have harmful environmental effects. Therefore, it is important to detect the major factors influencing flight delay. The factors influencing the flight delay range from natural factors like weather to factors like day, month, etc. On observing wide variety of flight data, a list of factors responsible for the delay was generated. There are other factors influencing the delay as well, but their scope is limited. This list primarily categorizes delay into two types, namely departure delay and arrival delay. Some of these features influence the flight delay drastically while others have a minor impact. In order to develop an efficient flight delay prediction system, these features along with their degree of impact on the delay must be taken into consideration.

3.2 EXISTING SYSTEM

Many attempts have been made by researchers in the past for predicting flight delays using Machine Learning, Deep Learning and Big Data approaches. Kalliguddi(author) constructed regression models like Decision Tree Regressor, Random Forest regression on flight data for predicting both departure and arrival delays. The main issue is to find the error rate in terms of predictions and reducing the error factor in the model.

3.2.1LIMITATIONS OF EXISTING SYSTEM

•Just simple data mining for the data set and observations.

•No scaling on the data of the prediction.

•No proper resolution of how the flights delay has been happening.

•Impacts of the delay on next flight and time schedules

3.3 PROPOSED SYSTEM

Accurate flight delay prediction is fundamental to establish the more efficient airline business. Recent studies have been focused on applying machine learning methods to predict the flight delay. The primary goal of this project is to predict airline delays caused by various factors and Error rate on models. Flight delays lead to negative impacts, mainly economical for commuters, airline industries and airport authorities. To carry out the predictive analysis, which encompasses a range of statistical techniques from supervised machine learning and, data mining, that studies current and historical data to make predictions or just analyse about the future delays, with help of Regression Analysis using regularization technique in Python.

3.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

•Proper analysis of the data in this model leads to the identification of delays and types of delays.

•In this model analysis helps us to overcome other flight delays and management issues.

•This model helps in predicting the delays of the flight under different circumstances like weather and technical problems.

•Many attempts have been by researchers in the past for predicting flight delays

3.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis are

•Economic Feasibility
•Technical Feasibility
•Social Feasibility

3.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.
The following are some of the important financial questions asked during preliminary investigation:

•The costs conduct a full system investigation.
•The cost of the hardware and software.
•The benefits in the form of reduced costs or fewer costly errors.
Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

3.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.4.3 BEHAVIORAL FEASIBILITY

This includes the following questions:
•Is there sufficient support for the users?
•Will the proposed system cause harm?
The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the

project is behaviorally feasible.

3.5 HARDWARE & SOFTWARE REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

• Processor	:	Intel Dual Core@ CPU 2.90GHz.
• Hard disk	:	500GB and Above.
• RAM	:	4GB and Above.
• Monitor	:	5 inches or above.

3.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

Operating system	:	Windows 8, 10
Languages	:	Python
Backend	:	Machine Learning
IDE	:	Jupyter

4. ARCHITECTURE

4. ARCHITECTURE

4.1 PROJECT ARCITECTURE

This project architecture shows the procedure followed for flight delay using machine learning, starting from input to final prediction.

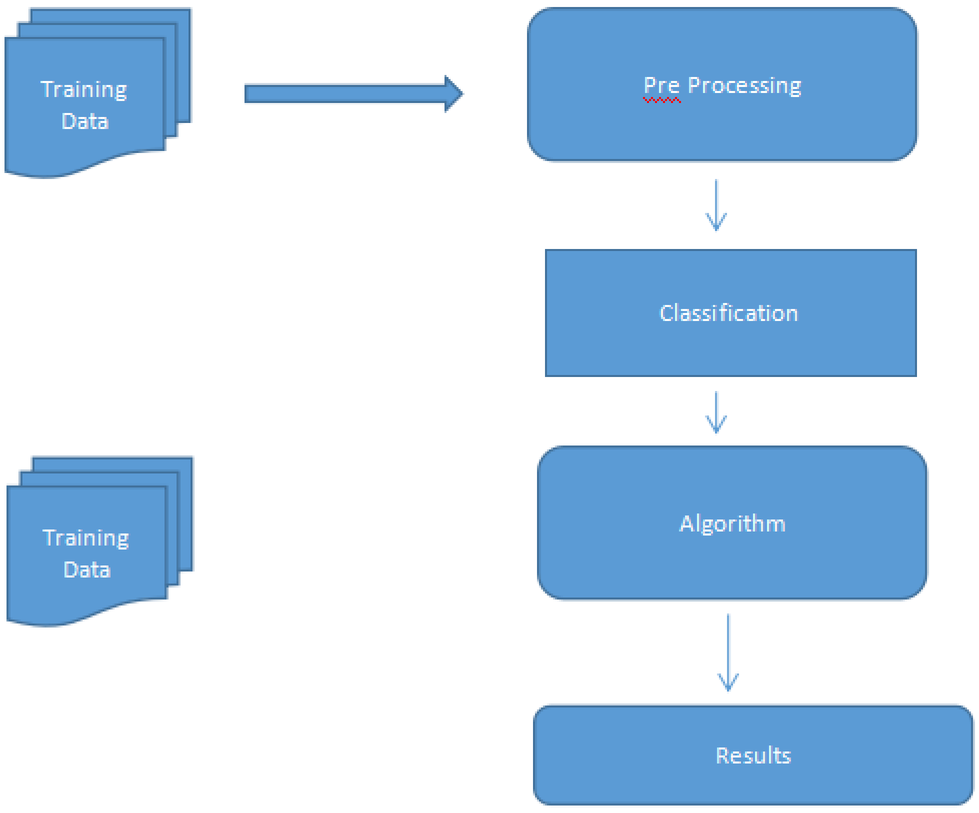


Figure 4.1: Project Architecture

4.2 DESCRIPTION

Input Data: Input data is generally in csv format where the data is fetched and mapped in the data framed from the source columns.

Reading Data: pandas sklearn library is used to read the data into the data frame.

Separating Features: In this following step we are going to separate the features which we take to train the model by giving the target value i.e. 1/0 for the particular of features.

Normalization: Normalization is a very important step while we are dealing with the large values in the features as the higher bit integers will cost high computational power and time. To achieve the efficiency in computation we are going to normalize the data values.

Training and test data: Training data is passed to the MLP classifier to train the model. Test data is used to test the trained model whether it is making correct predictions or not.

4.3 USE CASE DIAGRAM

The use case graph is for demonstrating the direct of the structure. This chart contains the course of action of use cases, performing pros and their relationship. This chart might be utilized to address the static perspective of the structure.

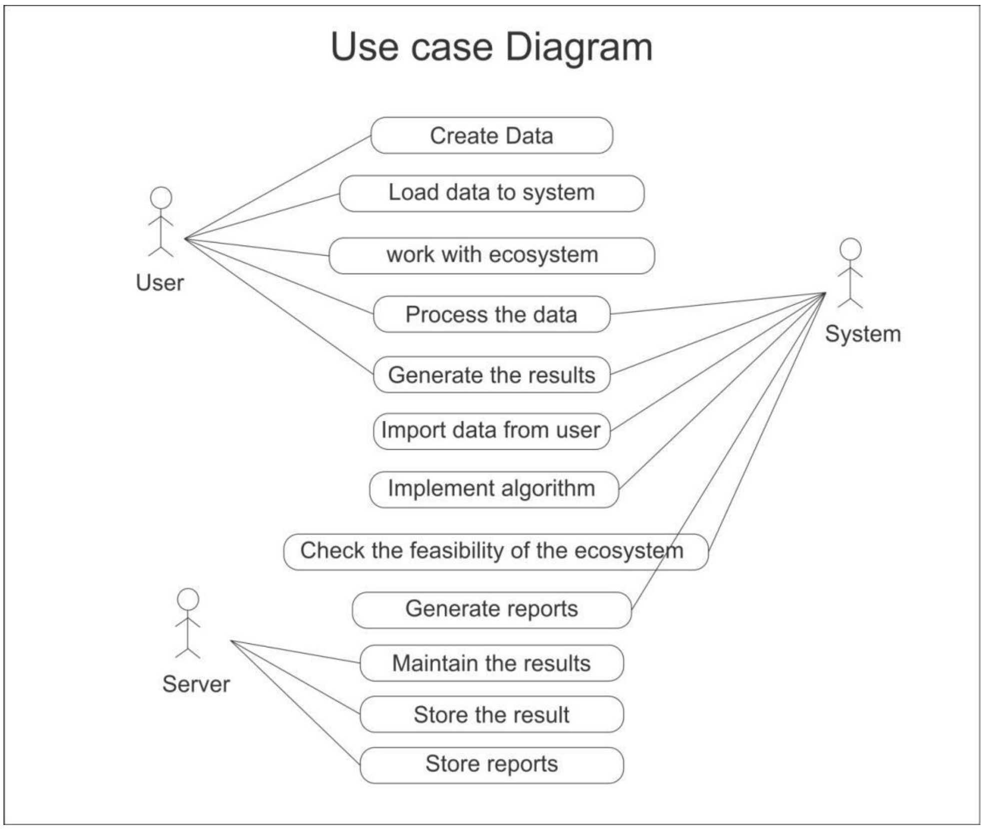


Figure 4.2: Use Case Diagram

4.4 CLASS DIAGRAM

The class graph is the most normally pulled in layout UML. It addresses the static course of action perspective of the structure. It solidifies the strategy of classes, interfaces, joint attempts and their affiliations.

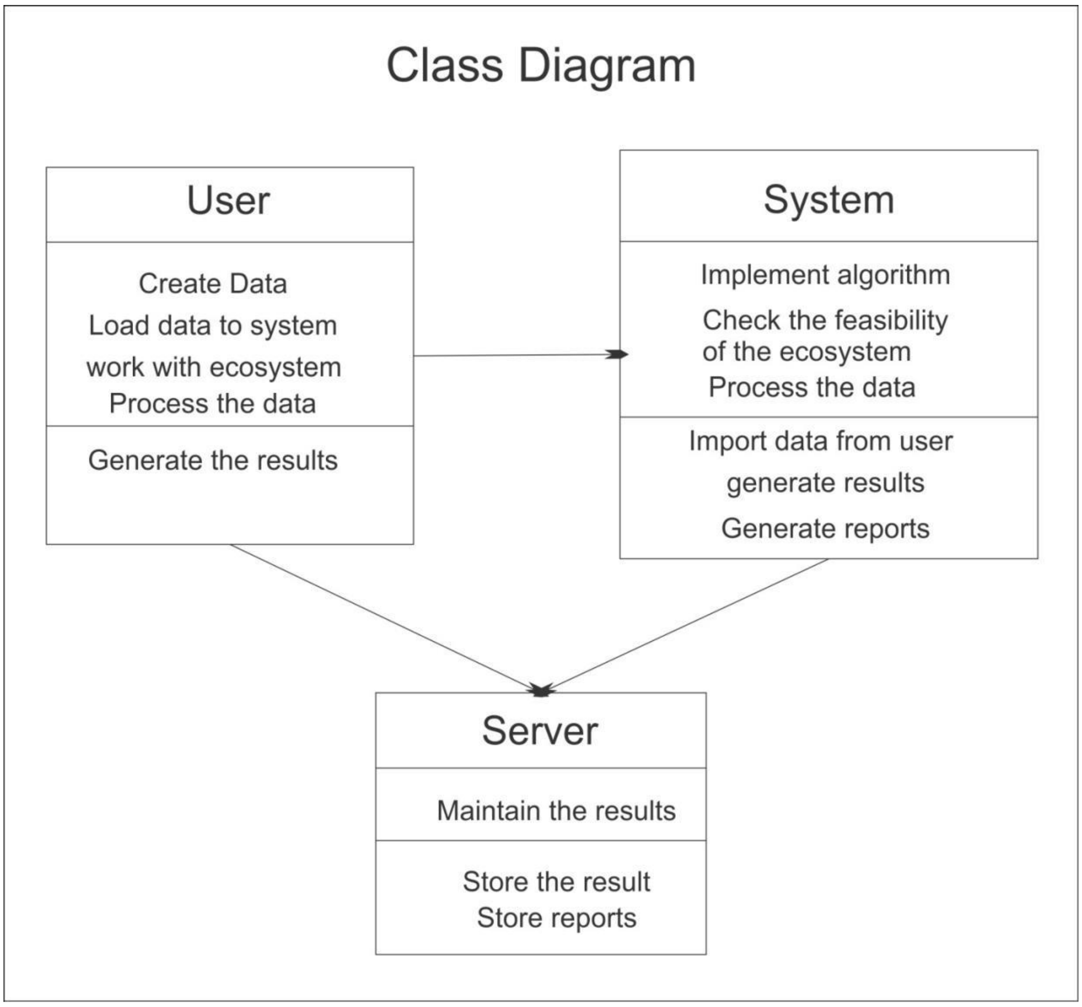


Figure 4.3: Class Diagram

4.5 SEQUENCE DIAGRAM

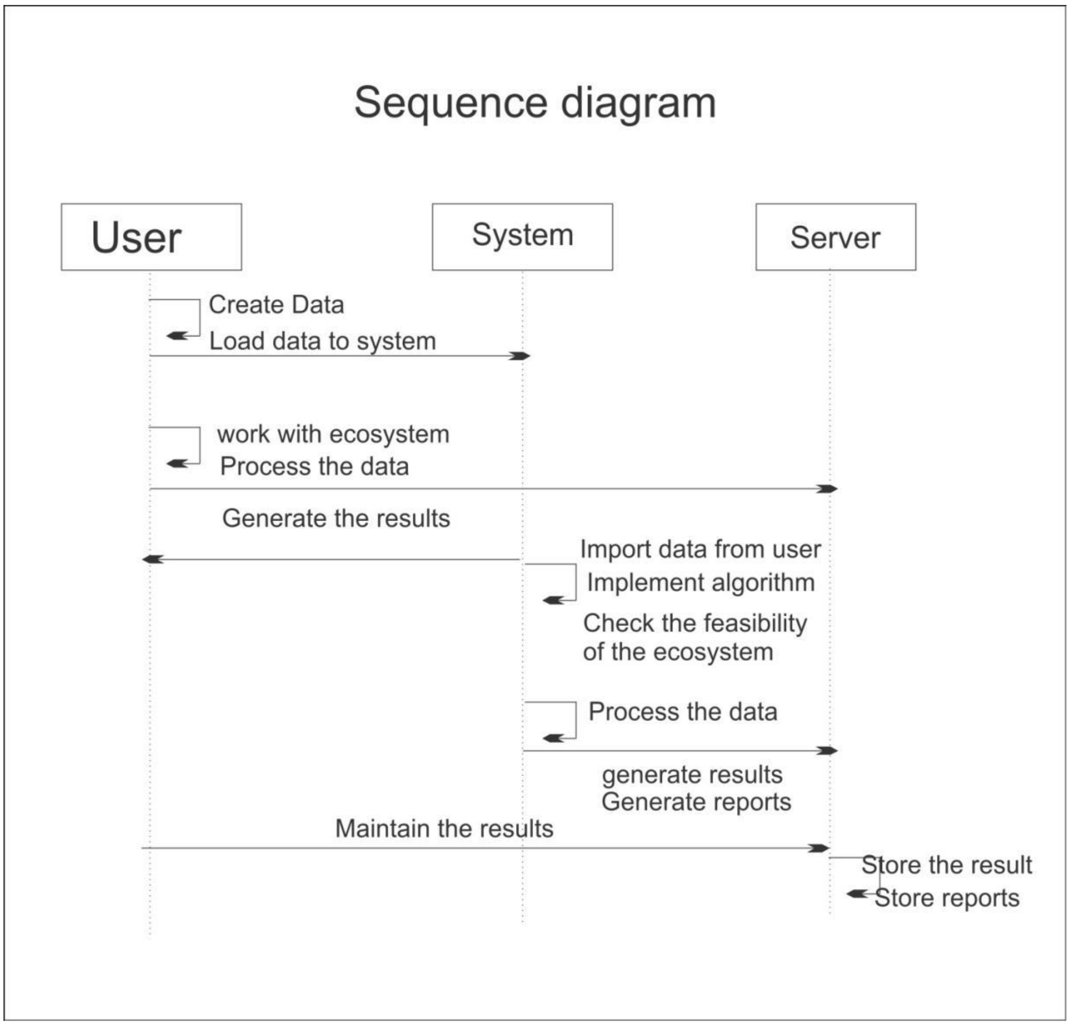


Figure 4.4: Sequence Diagram

4.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

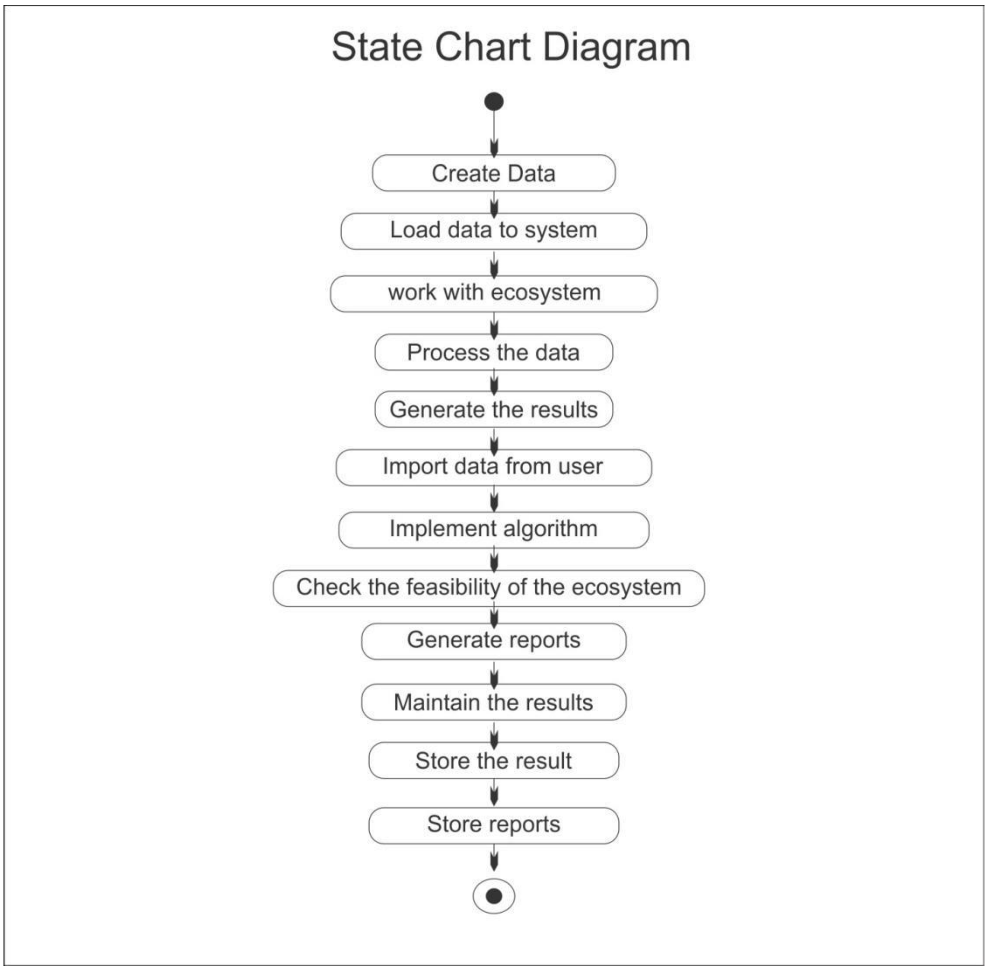


Figure 4.5: Activity Diagram

5. **IMPLEMENTATION**

5. IMPLEMENTATION

5.1 SAMPLE CODE

```
# import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time

df = pd.read_csv("C:/Users/sravan/Desktop/2020/New
projects/Satish/flight/DelayedFlights/DelayedFlights.csv")
df = df.drop("Unnamed: 0",1)
df = df[df["Month"].isin([10,11,12])]
df.head()

df['TaxiOut'].fillna(0, inplace=True)
cancelled = df[df['Cancelled']==1]
cancelled.tail()

font = {'size'   : 16}
plt.rc('font', **font)

days_cancelled = cancelled['Cancelled'].groupby(df['DayOfWeek']).count()
days_total = df['Cancelled'].groupby(df['DayOfWeek']).count()
days_frac = np.divide(days_cancelled, days_total)
x=days_frac.index.values
week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

fig, ax = plt.subplots(figsize = (12,6))
ax.bar(x,days_frac*100, align='center')
ax.set_ylabel('Percentage of Flights Cancelled')
ax.set_xticks(x)
ax.set_xticklabels(week, rotation = 45)
```

```
plt.show()

df['CRSDepTime'].head(10)
fig, ax = plt.subplots(figsize = (12,6))
ax.hist([df['CRSDepTime'], cancelled['CRSDepTime']], normed=1, bins=20,
label=['All', 'Cancelled'])
ax.set_xlim(0,2400)
ax.set_xlabel('Scheduled Departure Time')
ax.set_title('Normalized histogram of Scheduled Departure Times')
plt.legend()
plt.show()

df['DayofMonth'].head(10)
fig, ax = plt.subplots(figsize = (12,6))
ax.hist([df['DayofMonth'], cancelled['DayofMonth']], normed=1, bins=31,
label=['All', 'Cancelled'])
ax.set_xlim(0,31)
ax.set_xlabel('Day of Month')
ax.set_title('Normalized histogram of Day of Month')
plt.legend()
plt.show()

fig, ax = plt.subplots(figsize = (12,6))
ax.hist([df['Month'], cancelled['Month']], normed=1, bins=3, label=['All', 'Cancelled'])
ax.set_xlim(10,12)
ax.set_xlabel('Month')
ax.set_title('Normalized histogram of Months')
plt.legend()
plt.show()

fig, ax = plt.subplots(figsize = (12,6))
ax.hist([df['Distance'], cancelled['Distance']], normed=1, bins=20, label=['All',
'Cancelled'])
```

```
ax.set_xlim(0,3000)
ax.set_xlabel('Flight Distance in miles')
ax.set_title('Normalized histogram of Flight Distances')
plt.legend()
plt.show()

fig, ax = plt.subplots(figsize = (12,6))
all_delay_log = np.log10(df['DepDelay'])
adl_mean = np.log10(np.mean(df['DepDelay']))
cancelled_delay_log = np.log10(cancelled['DepDelay'])
cdl_mean = np.log10(np.mean(cancelled['DepDelay']))
ax.hist([all_delay_log, cancelled_delay_log], normed=1, bins=20, label=['All',
'Cancelled'])
#ax.hist([df['DepDelay'], cancelled['DepDelay']], normed=1,
bins=np.logspace(0.8,2.5, 30), label=['All', 'Cancelled'])
ax.plot([adl_mean, adl_mean],[-1,2],color='#1f77b4')
ax.plot([cdl_mean, cdl_mean],[-1,2],color='#ff7f0e')
ax.set_xlim(0.7,3)
ax.set_ylim(-0.1,1.1)
#ax.set_xscale("log", nonposx='clip')
#plt.gca().set_xscale("log")
ax.set_xlabel('log10(Departure Delay in minutes)')
ax.set_title('Normalized histogram of Departure Delay Times')
plt.legend()
plt.show()

df['total_delay'] = (df['CarrierDelay'] + df['WeatherDelay']
        + df['NASDelay'] + df['SecurityDelay'] + df['LateAircraftDelay'])
df_delayed = df[~np.isnan(df['total_delay'])]
df['total_delay'].fillna(0, inplace=True)
df_delayed.head()
carrier_group =
df_delayed['CarrierDelay'].groupby(df_delayed['UniqueCarrier']).mean()
```

```
weather_group =
df_delayed['WeatherDelay'].groupby(df_delayed['UniqueCarrier']).mean()
nas_group = df_delayed['NASDelay'].groupby(df_delayed['UniqueCarrier']).mean()
security_group =
df_delayed['SecurityDelay'].groupby(df_delayed['UniqueCarrier']).mean()
late_group =
df_delayed['LateAircraftDelay'].groupby(df_delayed['UniqueCarrier']).mean()
w_bottom = carrier_group.values
n_bottom = w_bottom + weather_group.values
s_bottom = n_bottom + nas_group.values
l_bottom = s_bottom + security_group.values
x = carrier_group.index.values
fig, ax = plt.subplots(figsize = (12,6))
ax.set_xticks(np.arange(len(x)))
ax.set_xticklabels(x, rotation = 45)
ax.bar(np.arange(len(x)),carrier_group.values, align='center', label='Carrier Delay')
ax.bar(np.arange(len(x)),weather_group.values, align='center', bottom=w_bottom,
label='Weather Delay')
ax.bar(np.arange(len(x)),nas_group.values, align='center', bottom=n_bottom,
label='NAS Delay')
ax.bar(np.arange(len(x)),security_group.values, align='center', bottom=s_bottom,
label='Security Delay')
ax.bar(np.arange(len(x)),late_group.values, align='center', bottom=l_bottom,
label='Late Aircraft Delay')
ax.set_xlabel('Aircraft Carrier Code')
ax.set_ylabel('Departure Delay in minutes')
plt.legend()
plt.show()

cancelled_group =
cancelled.groupby(['UniqueCarrier','CancellationCode']).size().reindex(fill_value=0.0
).unstack()
cg = cancelled_group.fillna(0)
```

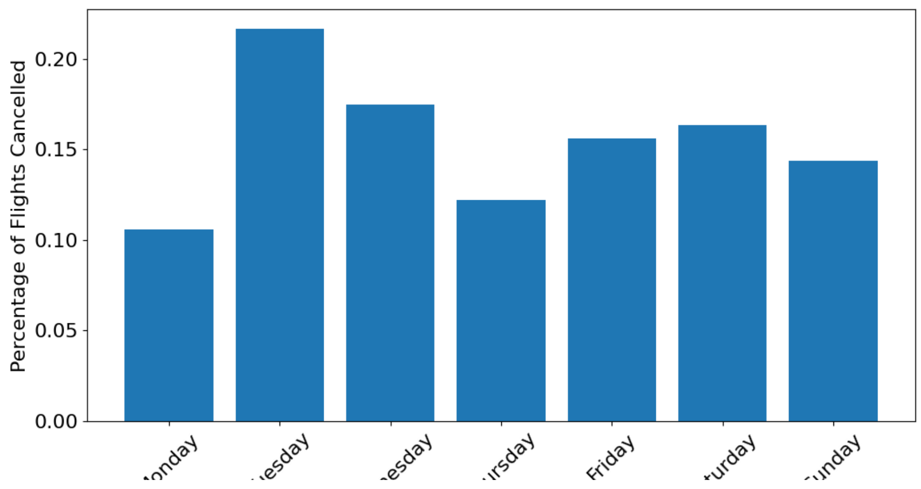
```
b_bottom = cg.loc[:,'A'].values
c_bottom = b_bottom + cg.loc[:,'B'].values
d_bottom = c_bottom + cg.loc[:,'B'].values
x = cg.loc[:,'A'].index.values
fig, ax = plt.subplots(figsize = (12,6))
ax.set_xticks(np.arange(len(x)))
ax.set_xticklabels(x, rotation = 45)
ax.bar(np.arange(len(x)),cg.loc[:,'A'].values, align='center', label='Carrier')
ax.bar(np.arange(len(x)),cg.loc[:,'B'].values, align='center', bottom=b_bottom,
label='Weather')
ax.bar(np.arange(len(x)),cg.loc[:,'C'].values, align='center', bottom=c_bottom,
label='NAS')
#ax.bar(np.arange(len(x)),cancelled_group.loc[:,'D'].values, align='center',
bottom=d_bottom, label='Security')
ax.set_xlabel('Aircraft Carrier Code')
ax.set_ylabel('Number of Cancellations')
plt.legend()
plt.show()
total_flights_per_carrier = df['UniqueCarrier'].groupby(df['UniqueCarrier']).count()
fig, ax1 = plt.subplots(figsize = (12,6))
x = total_flights_per_carrier.index.values
ax1.set_xticks(np.arange(len(x)))
ax1.set_xticklabels(x, rotation = 45)
ax1.bar(np.arange(len(x)),total_flights_per_carrier.values, align='center')
ax1.set_xlabel('Aircraft Carrier Code')
ax1.set_ylabel('Total Number of Flights')
plt.show()

df['Carrier mean delay'] =
df['total_delay'].groupby(df['UniqueCarrier']).transform('mean')
df['Carrier mean distance'] =
df['Distance'].groupby(df['UniqueCarrier']).transform('mean')
df['Carrier cancellations'] =
df['Cancelled'].groupby(df['UniqueCarrier']).transform('mean')
df['Origin cancellations'] = df['Cancelled'].groupby(df['Origin']).transform('mean')
df['Dest cancellations'] = df['Cancelled'].groupby(df['Dest']).transform('mean')
df['Origin TaxiOut'] = df['TaxiOut'].groupby(df['Origin']).transform('mean')
df['Origin Delay'] = df['total_delay'].groupby(df['Origin']).transform('mean')
df['Origin'] = df['Origin'].astype('category').cat.codes
df['Dest'] = df['Dest'].astype('category').cat.codes
df['CancellationCode'] = df['CancellationCode'].astype('category').cat.codes
df.fillna(0, inplace=True)

X = df[['Month', 'DayofMonth', 'DayOfWeek', 'CRSDepTime', 'Origin', 'Dest',
'Distance', 'Carrier mean distance',
       'Origin Delay', 'Origin TaxiOut']]
y = df['Cancelled']

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

nnclf = MLPClassifier(hidden_layer_sizes = [5,5], solver='adam', alpha=0.0003,
activation='relu',
                  max_iter = 100, random_state = 47).fit(X_train_scaled, y_train)
y_predicted = nnclf.predict(X_test_scaled)
confusion = confusion_matrix(y_test, y_predicted)
```

```
print('Accuracy: {:.3f}'.format(accuracy_score(y_test, y_predicted)))
confusion = confusion_matrix(y_test, y_predicted)
print(confusion)

from sklearn.svm import SVC
np.set_printoptions(formatter={'float': lambda x: "{0:0.3f}".format(x)})
svm = SVC(kernel='rbf', C=1000, gamma=6, random_state=47).fit(X_train_scaled,
y_train)
y_pred = svm.predict(X_test_scaled)

print('Recall: {:.3f}'.format(recall_score(y_test, y_pred)))
print('Precision: {:.3f}'.format(precision_score(y_test, y_pred)))
print('Accuracy: {:.3f}'.format(accuracy_score(y_test, y_pred)))
print('F1: {:.3f}'.format(f1_score(y_test, y_pred)))
confusion = confusion_matrix(y_test, y_pred)
print(confusion)

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
clf = RandomForestClassifier(n_estimators=50, random_state=47).fit(X_train,
y_train)

# sum(y_test)
# clf.score(X_test, y_test)
y_predicted = clf.predict(X_test)
confusion = confusion_matrix(y_test, y_predicted)
#confusion
#sum(y_predicted)

print('Recall: {:.3f}'.format(recall_score(y_test, y_predicted)))
print('Precision: {:.3f}'.format(precision_score(y_test, y_predicted)))
print('Accuracy: {:.3f}'.format(accuracy_score(y_test, y_predicted)))
print('F1: {:.3f}'.format(f1_score(y_test, y_predicted)))
confusion = confusion_matrix(y_test, y_predicted)
print(confusion)
print('Feature importances: {}'.format(clf.feature_importances_))

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train,y_train)
gnb.score(X_train, y_train)
y_pred = gnb.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

6. SCREENSHOTS

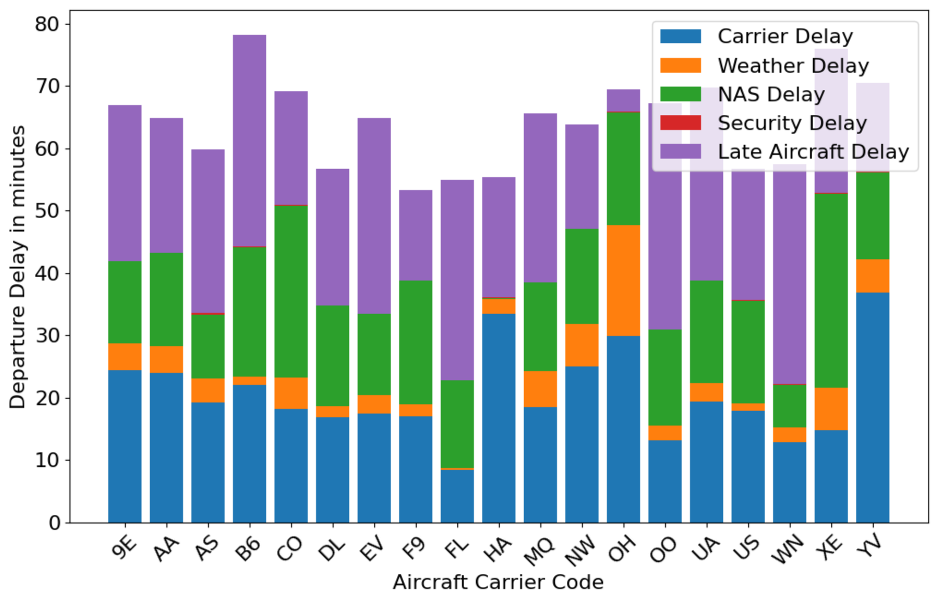
6. SCREEN SHOTS

6.1 Day wise flights



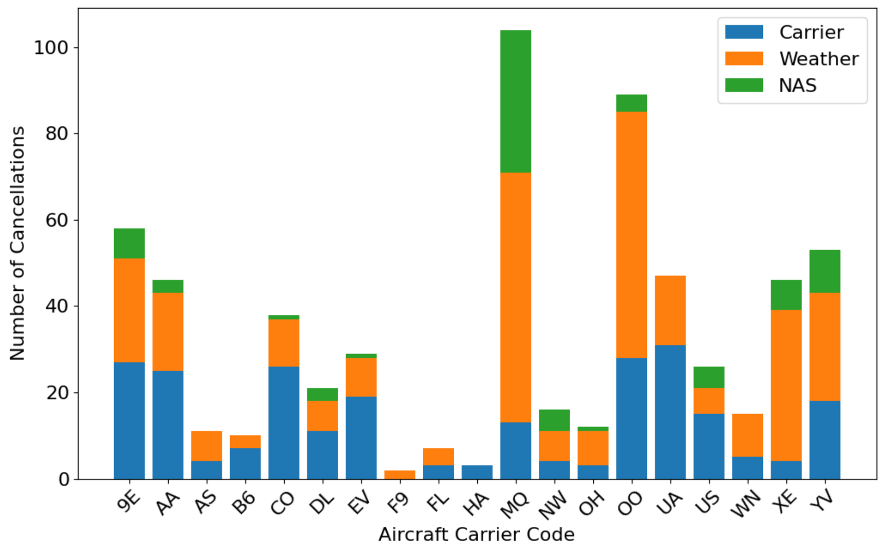
Screenshot 6.1: Day wise percentage of Flights

6.2 Delay with Features



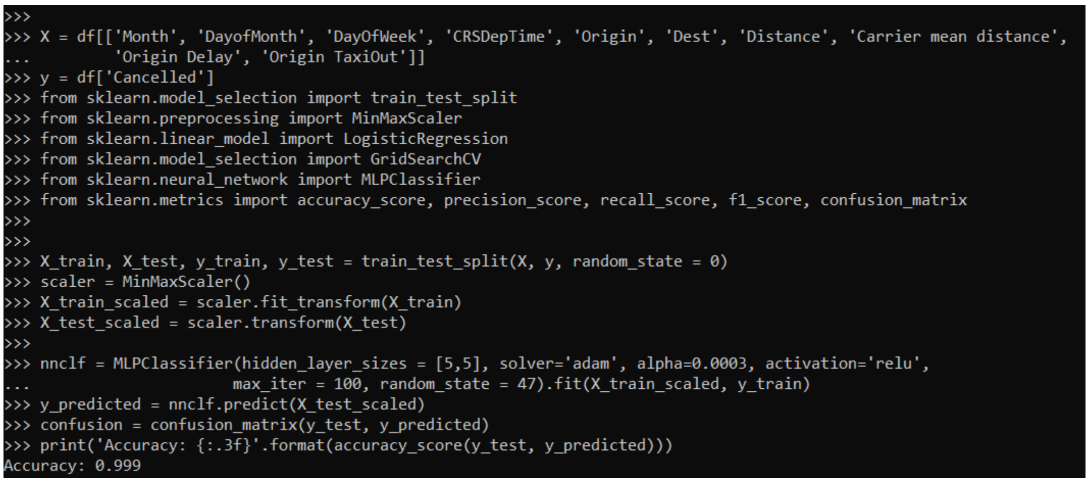
Screenshot 6.2: Delays caused for the flights in departure

6.3 Flights Canceled with Features



Screenshot 6.3: Flights canceled due to the features

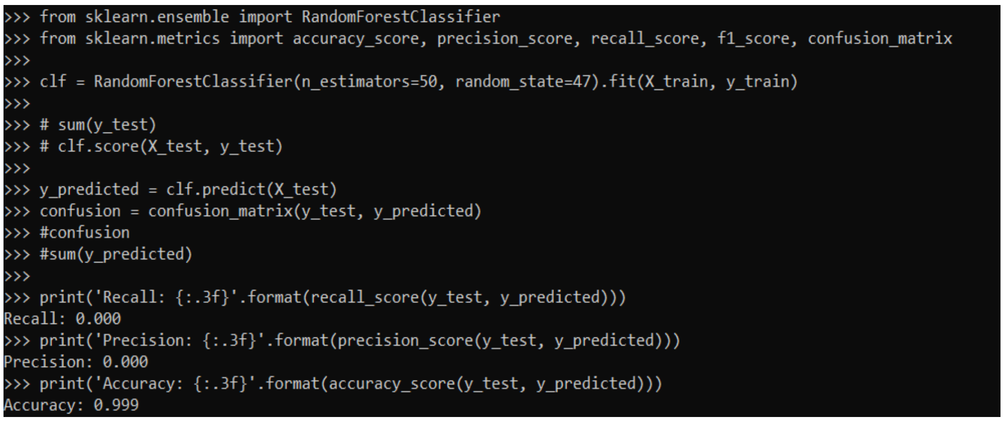
6.4 MLP Classifier



```
>>>
>>> X = df[['Month', 'DayofMonth', 'DayOfWeek', 'CRSDepTime', 'Origin', 'Dest', 'Distance', 'Carrier mean distance',
...           'Origin Delay', 'Origin TaxiOut']]
>>> y = df['Cancelled']
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.preprocessing import MinMaxScaler
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.model_selection import GridSearchCV
>>> from sklearn.neural_network import MLPClassifier
>>> from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
>>>
>>>
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
>>> scaler = MinMaxScaler()
>>> X_train_scaled = scaler.fit_transform(X_train)
>>> X_test_scaled = scaler.transform(X_test)
>>>
>>> nnclf = MLPClassifier(hidden_layer_sizes = [5,5], solver='adam', alpha=0.0003, activation='relu',
...                       max_iter = 100, random_state = 47).fit(X_train_scaled, y_train)
>>> y_predicted = nnclf.predict(X_test_scaled)
>>> confusion = confusion_matrix(y_test, y_predicted)
>>> print('Accuracy: {:.3f}'.format(accuracy_score(y_test, y_predicted)))
Accuracy: 0.999
```

Screenshot 6.4: MLP Classifier

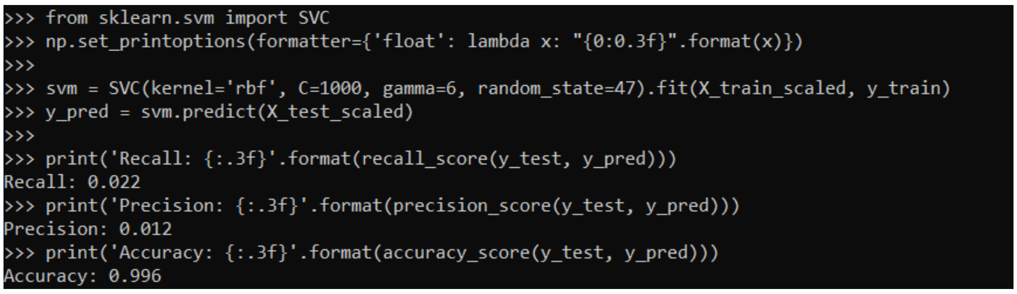
6.5 RF Classifier



```
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
>>>
>>> clf = RandomForestClassifier(n_estimators=50, random_state=47).fit(X_train, y_train)
>>>
>>> # sum(y_test)
>>> # clf.score(X_test, y_test)
>>>
>>> y_predicted = clf.predict(X_test)
>>> confusion = confusion_matrix(y_test, y_predicted)
>>> #confusion
>>> #sum(y_predicted)
>>>
>>> print('Recall: {:.3f}'.format(recall_score(y_test, y_predicted)))
Recall: 0.000
>>> print('Precision: {:.3f}'.format(precision_score(y_test, y_predicted)))
Precision: 0.000
>>> print('Accuracy: {:.3f}'.format(accuracy_score(y_test, y_predicted)))
Accuracy: 0.999
```

Screenshot 6.5:Random Forest Classifier

6.6 SVM Classifier



```
>>> from sklearn.svm import SVC
>>> np.set_printoptions(formatter={'float': lambda x: "{0:0.3f}".format(x)})
>>>
>>> svm = SVC(kernel='rbf', C=1000, gamma=6, random_state=47).fit(X_train_scaled, y_train)
>>> y_pred = svm.predict(X_test_scaled)
>>>
>>> print('Recall: {:.3f}'.format(recall_score(y_test, y_pred)))
Recall: 0.022
>>> print('Precision: {:.3f}'.format(precision_score(y_test, y_pred)))
Precision: 0.012
>>> print('Accuracy: {:.3f}'.format(accuracy_score(y_test, y_pred)))
Accuracy: 0.996
```

Screenshot 6.6: SVM Classifier

7. TESTING

7. TESTING

7.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Testing is a procedure, which uncovers blunders in the program. Programming testing is a basic component of programming quality affirmation and speaks to a definitive audit of determination, outline and coding. The expanding perceivability of programming as a framework component and chaperon costs related with a product disappointment are propelling variables for we arranged, through testing. Testing is the way toward executing a program with the plan of finding a mistake. The plan of tests for programming and other built items can be as trying as the underlying outline of the item itself It is the significant quality measure utilized amid programming improvement. Amid testing, the program is executed with an arrangement of experiments and the yield of the program for the experiments is assessed to decide whether the program is executing as it is relied upon to perform.

A technique for programming testing coordinates the outline of programming experiments into an all around arranged arrangement of steps that outcome in fruitful improvement of the product. The procedure gives a guide that portrays the means to be taken, when, and how much exertion, time, and assets will be required. The procedure joins test arranging, experiment configuration, test execution, and test outcome gathering and assessment. The procedure gives direction to the specialist and an arrangement of points of reference for the chief. Due to time weights, advance must be quantifiable and issues must surface as ahead of schedule as would be prudent

Keeping in mind the end goal to ensure that the framework does not have blunders, the distinctive levels of testing techniques that are connected at varying periods of programming improvement are here.

Framework testing includes in-house testing of the whole framework before conveyance to the client. Its point is to fulfill the client the framework meets all necessities of the customer's determinations. This testing assesses working of framework from client perspective, with the assistance of particular report. It doesn't require any inward learning of framework like plan or structure of code.

It contains utilitarian and non-useful zones of utilization/item. Framework Testing is known as a super arrangement of a wide range of testing as all the significant sorts of testing are shrouded in it. In spite of the fact that attention on sorts of testing may differ on the premise of item, association procedures, course of events and necessities. Framework Testing is the start of genuine testing where you test an item all in all and not a module/highlight.

Acknowledgment testing, a testing method performed to decide if the product framework has met the prerequisite particulars. The principle motivation behind this test is to assess the framework's consistence with the business necessities and check in the event that it is has met the required criteria for conveyance to end clients. It is a pre-conveyance testing in which whole framework is tried at customer's site on genuine information to discover blunders. The acknowledgment test bodies of evidence are executed against the test information or utilizing an acknowledgment test content and afterward the outcomes are contrasted and the normal ones.

The acknowledgment test exercises are completed in stages. Right off the bat, the essential tests are executed, and if the test outcomes are palatable then the execution of more intricate situations are done

Bottom up Approach : Testing can be performed beginning from littlest and most reduced level modules and continuing each one in turn. In this approach testing is directed from sub module to primary module, if the fundamental module is not built up a transitory program called DRIVERS is utilized to recreate the principle module.

At the point when base level modules are tried consideration swings to those on the following level that utilization the lower level ones they are tried exclusively and afterward connected with the already inspected bring down level modules.

Top down Approach : In this approach testing is directed from fundamental module to sub module. in the event that the sub module is not built up an impermanent program called STUB is utilized for mimic the sub module. This sort of testing begins from upper level modules. Since the nitty gritty exercises more often than not performed in the lower level schedules are not given stubs are composed. A stub is a module shell called by upper level module and that when achieved legitimately will restore a message to the calling module demonstrating that appropriate association happened.

7.2 TYPES OF TESTING

7.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. Unit Testing is done on singular modules as they are finished and turned out to be executable. It is restricted just to the planner's prerequisites. It centers testing around the capacity or programming module. It Concentrates on the interior preparing rationale and information structures. It is rearranged when a module is composed with high union.

It is otherwise called Functional testing. A product testing strategy whereby the inward workings of the thing being tried are not known by the analyzer. For instance, in a discovery test on a product outline the analyzer just knows the information sources and what the normal results ought to be and not how the program touches base at those yields. The analyzer does not ever inspect the programming code and does not require any further learning of the program other than its determinations. In this system some experiments are produced as information conditions that completely execute every single practical prerequisite for the program. This testing has been utilizations to discover mistakes in the accompanying classifications.

7.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

It is otherwise called Glass box, Structural, Clear box and Open box testing . A product testing procedure whereby express learning of the inner workings of the thing being tried are utilized to choose the test information. Not at all like discovery testing, white box testing utilizes particular learning of programming code to inspect yields. The test is precise just if the analyzer comprehends what the program should do. He or she would then be able to check whether the program veers from its expected objective. White box testing does not represent blunders caused by oversight, and all obvious code should likewise be discernable. For an entire programming examination, both white box and discovery tests are required.

In this the experiments are produced on the rationale of every module by drawing stream diagrams of that module and sensible choices are tried on every one of the cases. It has been utilizations to produce the experiments in the accompanying cases

7.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

7.3 TEST CASES

7.3.1 UPLOADING IMAGES

Test case ID	Test case name	Purpose	Test Case	Output
1	User uploads image	Use it for identification	The user uploads the a dog image	Uploaded successfull y
2	User uploads 2nd image	Use it for identification	The user uploads the a non-dogimage	Uploaded successfull y

7.3.2 CLASSIFICATION

Test case ID	Test case name	Purpose	Input	Output
1	Classification test 1	To check if the classifier performsits task	A dog image is given	Breed is predicted.
2	Classification test 2	To check if the classifier performsits task	A human image is given	It predicted as human.
3	Classification test 3	To check if the classifier performsits task	A frog imageis given	Predicted as not a dog.

8. CONCLUSION & FUTURE SCOPE

8.CONCLUSION & FUTURE SCOPE

8.1 PROJECT CONCLUSION

The primary goal of this project is to predict airline delays caused by various factors and Error rate on models. Flight delays lead to negative impacts, mainly economical for commuters, airline industries and airport authorities. To carry out the predictive analysis, which encompasses a range of statistical techniques from supervised machine learning and, data mining, that studies current and historical data to make predictions or just analyze about the future delays, with help of Regression Analysis using regularization technique in Python.

8.2 FUTURE SCOPE

In future we can use other convolutional neural networks by downloading the modules directly into the project files. The software can be developed further to include lot of modules because the proposed system is developed on the view of future. We can connect to other data bases by including them .

9. BIBILOGRAPHY

9. BIBILOGRAPHY

9.1 REFERENCES

[1] N. G. Rupp, "Further Investigation into the Causes of Flight Delays," in Department of Economics, East Carolina University, 2007.

[2] "Bureau of Transportation Statistics (BTS) Databases and Statistics," [Online]. Available: http://www.transtats.bts.gov.

[3] "Airports Council International, World Airport Traffic Report," 2015,2016.

[4] E. Cinar, F. Aybek, A. Caycar, C. Cetek, "Capacity and delay analysis for airport manoeuvring areas using simulation," Aircraft Engineering and Aerospace Technology, vol. 86, no. No. 1,pp. 43-55, 2013.

[5] Navoneel, et al., Chakrabarty, "Flight Arrival Delay Prediction Using Gradient Boosting Classifier," in Emerging Technologies in Data Mining and Information Security, Singapore, 2019.

[6] Y. J. Kim, S. Briceno, D. Mavris, Sun Choi, "Prediction of weatherinduced airline delays based on machine learning algorithms," in 35th Digital Avionics Systems Conference (DASC), 2016.

[7] W.-d. Cao. a. X.-y. Lin, "Flight turnaround time analysis and delay prediction based on Bayesian Network," Computer Engineering and Design, vol. 5, pp. 1770-1772, 2011.

[8] J. J. Robollo, Hamsa, Balakrishnan, "Characterization and Prediction of Air Traffic Delays".

[9] S. Sharma, H. Sangoi, R. Raut, V. C. Kotak, S. Oza, "Flight Delay Prediction System Using Weighted Multiple Linear Regression," International Journal of Engineering and Computer Science, vol. 4, no. 4, pp. 11668 - 11677, April 2015.

[10] A. M. Kalliguddi, Area K., Leboulluec, "Predictive Modelling of Aircraft Flight Delay," Universal Journal of Management, pp. 485 491, 2017.

[11] Noriko, Etani, "Development of a predictive model for on-time arrival fight of airliner by discovering correlation between fight and weather data," 2019.

[12] C. J. Willmott, Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square(RMSE) in assessing average model performance," Climate Research, vol. 30, no. 1, pp. 79 - 82, 2005.

9.2 WEBSITES

[1] https://web.stanford.edu/class/cs231a/prev_projects_2016/output%20(1).pdf

[2] https://towardsdatascience.com/metrics-toevaluate-your-machine-learning-algorithm f10ba6e38234-.