

EFFICIENT ALGORITHM FOR ESTIMATION OF FLIGHT DELAY PREDICTION ON BIG DATA MECHANISM

Submitted in partial fulfillment of the requirements for the
award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

SANGEETA BURMAN (37110678)

SALONI JAIN. R (36110669)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL-2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SANGEETA BURMAN (37110678)** and **SALONI JAIN. R (36110669)** who carried out the project entitled "**EFFICIENT ALGORITHM FOR ESTIMATION OF FLIGHT DELAY PREDICTION BASED ON BIG DATA MECHANISM**" under our supervision from **NOVEMBER 2020** to **APRIL 2021**.

INTERNAL GUIDE

Dr. G.MEERA GANDHI, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

SANGEETA BURMAN (37110678) and SALONI JAIN. R (36110669) hereby declare that the Project Report entitled **“EFFICIENT ALGORITHM FOR ESTIMATION OF FLIGHT DELAY PREDICTION BASED ON BIG DATA MECHANISM”** done by me under the guidance of **Dr.Meeragandhi.G , M.E, Ph.D.,** , is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering in Computer Science.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean**, School of Computing and also **Dr. S. VIGNESHWARI, M.E., Ph.D., Dr. L. LAKSHMANAN, ME., Ph.D., Heads of the Department, Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide **Dr.Meeragandhi.G , M.E, Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Flight delay is a nondeterministic problem. Modelling and estimating flight delay is very important in the flight delay research. It is also the precondition to calculate delay propagation. A flight-delay and delay propagation model is established based on Bayesian network (BN). After trained the network by real data with Expectation Maximization (EM) arithmetic, the influences from the Arrival-delay and the Flight cancellation on the departure-delay are analyzed under different all kinds of states. The conclusion of study indicates that the propagation from arrival-delays to the departure delays is very obvious, especially when the degree of arrival-delay was MEDIUM or HEAVY, the influence of propagation could not be eliminated. The flight cancellation is one efficient method to deal with the influence. Flight cancellation is involved to build our model. After trained, the model proved that Flight cancellation could be adopted when delay propagation happened seriously which can weaken the propagation in a certain extent and the weakening degree is related to the number of cancelled flights. Then the proposed method mixed by the both structure learning and pure parameter learning to build a network model for a reduced Flight Chain, the model structure is established based on the learned topology. The delay estimation by the model proves much better than the existing model trained by pure parameter learning.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	viii
1	INTRODUCTION	1
	1.1 OBJECTIVES	2
	1.2 EXISTING SYSTEM	3
	1.3 FLIGHT DELAY BY SCENARIO	3
	1.4 PROPOSED SYSTEM	4
	1.5 ADVANTAGES OF PROPOSED SYSTEM	5
2	LITERATURE SURVEY	6
3	METHODOLOGY	9
	3.1 AIM AND SCOPE	9
	3.2 HARDWARE REQUIREMENTS	9
	3.3 SOFTWARE REQUIREMENTS	10
	3.4 SOFTWARE FEATURES	10
	3.4.1 Hadoop Distributed File System	10
	3.4.2 Ubuntu	13
	3.4.3 Apache Spark	14
	3.4.4 Scala	15
	3.5 METHODS	16
	3.5.1 Flight Data	16
	3.5.2 Predictive Model	17
	3.5.3 Statistical Analysis	17
	3.5.4 Data Correlation	18
	3.5.5 Exploring Delay Distribution	18
	3.5.6 Flight Delay Prediction	18
	3.6 ALGORITHMS	19
	3.6.1 NAÏVE BAYES CLASSIFICATION	19
	3.6.2 K-MEANS ALGORITHM	20
	3.7 METHODS	21

	3.8 STEP BY STEP DESCRIPTION OF OPERATION	22
	3.9 UNIFIED MODELLING LANGUAGE	23
	3.9.1 Use Case Diagram	23
	3.9.2 Sequence Diagram	25
4	RESULTS AND DISCUSSION	26
	4.1 ROOT OF FLIGHT DELAY	27
	4.2 DELAY PROPAGATION	28
5	CONCLUSION AND FUTURE ENHANCEMENT	29
	5.1 CONCLUSION	29
	5.2 FUTURE ENHANCEMENT	30
	REFERENCES	31
	APPENDICES	34
	A.SAMPLE CODE	34
	B.SCREENSHOTS	40
	C.PUBLICATION WITH PLAGARISM	43
	REPORT	

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO.
1.1	Flight Delay Scenario	4
3.1	Big Data Preview	11
3.2	HDFS Architecture	12
3.3	Ubuntu Home Page	13
3.4	Spark	15
3.5	Table of Flight Data	17
3.6	Flight Delay Prediction	19
3.7	Clustering Example	21
3.8	Use Case Diagram	24
3.9	Sequence Diagram	25
4.1	Delayed Flights	26
4.2	Arrival Time vs Average Delay	27
5.1	Collected Data Sample	40
5.2	Predicated Delay or No Delay	40
5.3	Predicated Delay or No Delay by Origin, Destination	41
5.4	Departure Delay by DOFW	41
5.5	Final Output	42

LIST OF ABBREVIATIONS

HDFS	- Hadoop Distributed File System
UML	- Unified Modelling Language
NAS	- National Aviation System
M Lib	- Master of Library
NoSQL	- Non Structured Query Language
RDD	- Resilient Distributed Dataset

CHAPTER 1

INTRODUCTION

Delay is one of the most remembered performance indicators of any transportation system. Notably, commercial aviation players understand delay as the period by which a flight is late or postponed. Thus, a delay may be represented by the difference between scheduled and real times of departure or arrival of a plane. Country regulator authorities have a multitude of indicators related to tolerance thresholds for flight delays. Indeed, flight delay is an essential subject in the context of air transportation systems. In 2019, 36% of flights delayed by more than five minutes in Europe, 31.1% of flights delayed by more than 15 minutes in the United States, and 16.3% of flights were cancelled or suffered delays greater than 30 minutes in Brazil. This indicates how relevant this indicator is and how it affects no matter the scale of airline meshes.

Flight delays have negative impacts, mainly economic, for passengers, airlines, and airports. Given the uncertainty of their occurrence, passengers usually plan to travel many hours earlier for their appointments, increasing their trip costs, to ensure their arrival on time. On the other hand, airlines suffer penalties, fines and additional operation costs, such as crew and aircrafts retentions in airports. Furthermore, from the sustainability point of view, delays may also cause environmental damage by increasing fuel consumption and gas emissions.

With the rapid development of the civil aviation industry, airspace has become increasingly crowded. This crowdedness causes increasingly frequent delays in most major airports worldwide. This situation seriously affects airports, airlines, and passengers. From 2007 to 2019, the annual flights in China consistently increased from 3.65million to 10.83million, with an average increasing rate of approximately 12.2% in the past five years. Meanwhile, the rate of flights arriving on time decreased from 83.19% in 2007 to 71.67% in 2019. The annual cost of flight delays in China was estimated to be more than \$7.4 billion. Such high economic costs of delay necessitate delay causal factor analysis and delay-reduction strategies.

OBJECTIVES

- The rapid growth in urban population has placed an enormous strain on urban transportation infrastructures.
- This is particularly the case in developing countries which experience the fastest urbanization, but suffer from far less developed urban transportation infrastructures. Conventionally, there are two primary models of urban transport systems, namely, public transit services such as buses.
- Subway and private passenger services such as taxis, shared shuttles, ride-hailing services.

However, delays show a certain aggregation pattern in the temporal dimension, high delays are normally clustered, and low delays tend to be surrounded by low delays. In other words, the delay value of samples with shorter distance between them is normally similar compared to the delay value of delays with longer distance between them. The correlation between two delay values depends on their spatial attribute such as spatial location and spatial distance. Without doubt, there is high degree of spatial dependence among delays in a space organized by hour and by day. Given that most of the for the mentioned methods were based on certain assumptions which either ignore or simplify the correlation of samples in the dataset, Diana initially introduced the approach of spatial analysis for delay prediction, which is able to take the spatial dependence in every direction into account.

There are two reasons why a flight may be delayed or cancelled. One is the factor of company circumstances. Another is the factor of the inevitability. As the company circumstances, the airline is responsible for machine parts trouble and the system failure. As the inevitability, there is the natural disaster including a heavy snow and the typhoon. So, on-time flight is focused on and on-time flight prediction is developed because the flight performance is not affected by the company circumstances and the inevitability. The key research in this paper is to discover the correlation between flight data and weather data. Flight data set of Flight Stats and weather data of Japan Meteorological Agency from March 1, 2012 to December 3, 2018 are gathered. Flight Stats also offers on-time arrival performance of flight which does not reach 15 min delay.

By using those data resources, the feature of flight performance and pressure patterns of weather is analyzed, and the sea-level pressures of 3 weather observation spots, which are Wakkanai as the most northern spot, Minami-Torishima as the most

eastern spot, and Yonagunijima as the most western spot, are selected as pinpoint of weather data in order to extract the features of weather related to pressure pattern as this method of selecting pinpoint data is useful in big data analytics to discover the correlation among data. The relation between those pinpoint data and the pressure pattern can be proved. As a result, a predictive model of on-time arrival flight is developed by using the flight data and the weather data. The model is evaluated by flight data and weather data from January 1, 2019 to January 31, 2019. Furthermore, a tool is developed to evaluate the feasibility of the model.

EXISTING SYSTEM

- Existing public transits operate along fixed routes with fixed time tables, where the transit capacity offered do not always match the time-varying trip demands. Consequently, many urban residents rely heavily on private cars and other transport modes (e.g., motor cycles, bikes) to get around a city, creating urban road congestion
- The choice of urban transit services from a passenger depends on the QoE and cost of the trip.
- Where the QoE hinges on many potential factors, including in-vehicle time, level of inconvenience, and the trip cost depends on the service operation cost.
- Private transit services in general offer high QoE, with low in vehicle time and high level of convenience, but at a high cost of trip fare.

THE FLIGHT DELAY SCENARIO

Commercial aviation is a complex distributed transportation system. It deals with valuable resources, demand fluctuations, and a sophisticated origin-destination matrix that need orchestration to provide smooth and safety operations. Furthermore, individual passenger follows her itineraries while airlines plan various schedules for aircrafts, pilots and flight attendants. This Figure illustrates a typical operation of a commercial flight. Stages can take place at terminal boundaries, airports, runways,

and airspace, being susceptible to different kinds of delays. Some examples include mechanical problems, weather conditions, ground delays, air traffic control, runway queues and capacity constraints.

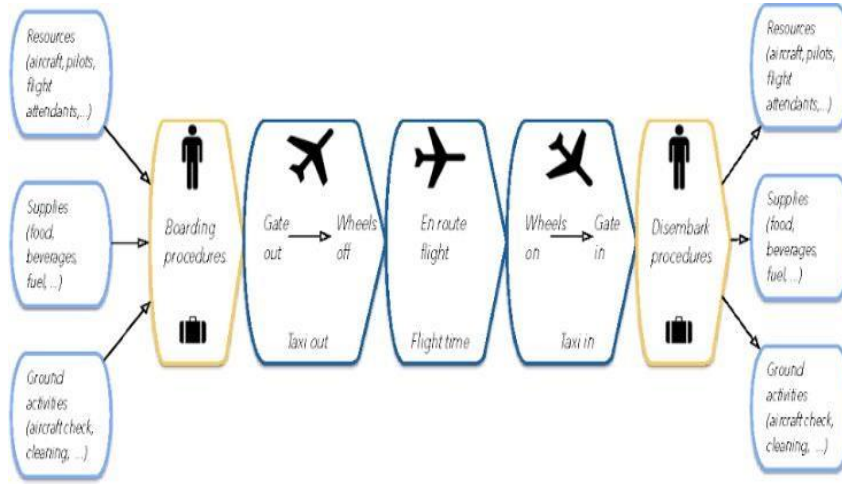


Fig 1.1 Flight Delay Scenario

PROPOSED SYSTEM

- Source-destination location pairs with a large number of trip demands, e.g., commute trips between a residential area and a commercial/working area, point-to-point mode is preferred.
- On the other hand, for those source destination pairs with less trip demands, hub-and-spoke mode is more promising to aggregate trip demands and reduce the operation cost by leveraging economics of scale. To balance such trade-offs, we propose to investigate the hybrid hub-and-spoke planning problem.
- Distribution changes dramatically over time and follows a stable diurnal pattern. To better cope with the trip demand dynamics, we divide each day into fixed time intervals, and develop City Lines solutions for different intervals.
- For the rest part of this paper, we focus on solving the hybrid hub-and-spoke planning problem for a given time interval.

The past few decades have seen rapid urbanization at the world scale. It is reported that the world urban population has reached 54% in 2014, and it is projected that by 2050, two-thirds of the world population will be urban. Conventionally, there are two primary models of urban transport systems, namely, public transit services such as buses, subway, and private passenger services such as taxis, shared shuttles, ride-hailing services (e.g., Airline Systems or Uber). Both systems have limitations following fundamental trade-offs in transit service efficiency and costs in fulfilling passengers' demands or "quality-of-experience" (QoE), especially during peak demand hours.

However, it can be a challenging task to plan and implement the hub- and-spoke mode in urban transportation for the following reasons:

- (i) Urban transits operate on an extremely large spatio-temporal scale, thus it is non-trivial to develop a scalable hub-and-spoke network to dynamically serve vast volumes of trip demands over time. In operations research, the hub location problem (HLP) has been studied for industry planning, e.g., airline route planning. These solutions, however, are limited to a maximum scale of 200 regions/spokes.
- (ii) In a real urban area with diverse distributions of trip demands, it is desirable but yet challenging to integrate both point-to-point and hub-and-spoke modes in an adaptive and dynamic fashion.

ADVANTAGES OF PROPOSED SYSTEM

- The need for a complicated multi-speed transmission and clutch is eliminated in series hybrids since only the electric motor directly drives the transmission, which may have one gear.
- Gasoline engines in series hybrids tend to be smaller and more efficient since they do not directly power the vehicle and are not subject to the highly variable power demands of stop-and-go driving.
- The smaller, more efficient engine and greater use of electric power helps reduce harmful gas emissions in series hybrids.

CHAPTER 2

LITERATURE SURVEY

A lot of research on flight delay has been done. Flight delay prediction, its analysis, and its causes have been an important topic for air traffic control, airline decision-making and ground delay response programs. In the United States, transit at hub airports is distinctive. Delay model studies in delay propagation of the sequence have been conducted. In order to use the delay model for prediction, it has been developed to study predictive models. Moreover, the study of predictive model for arrival delay and departure delay with weather data has been promoted.

1. In , Kim et al. was focusing on flight sequence, and the proposed Recurrent Neural Networks (RNN) predicted departure and arrival delay with using weather data of National Oceanic and Atmospheric Administration (NOAA). This prediction showed the accuracy of delayed flight prediction is 91.81% of McCarran International Airport (LAS) to 71.34% of Sky Harbor International Airport (PHX) because of the difference

Page 3 of 17Etani in data volume.

2. In, Choi et al. were focusing on a relation between weather, which data was gathered from NOAA, and flight delay. As a result, the proposed Random Forest, which is an ensemble learning method, predicted 80.36% of arrival delay.

3. In, Belcastro et al. predicted the arrival flight delay due to bad weather by Random Forest in Map Reduce. This study, which used weather data of NOAA, showed the result of prediction as follows: (1) with a delay threshold of 15 min, an accuracy of 74.2% and 71.8% recall on delayed flights are achieved, and (2) with a threshold of 60 min, the accuracy is 85.8% and the delay recall is 86.9%.

4. In, Thiagarajan et al. proposed a predictive model by Gradient Boosting, which is a machine learning technique for regression and classification with weather data of the World Weather Online API consisting of weather attributes of origin and destination airports. They predicted 94.35% of arrival delay and 86.48% of departure delay.

Outside the United States, in, Prasad et al. predicted departure and arrival delays by

Decision Tree, which uses a tree-like model of decisions and their possible consequences, in domestic flights of Brazil with weather data of the same data resource as that of Thiagarajan. In arrival delay by Decision Tree classifier, the accuracy is 78%. In departure delay by Regression classifier, the accuracy is 77%.

5. Dieterich Lawson and William Castillo, in their project "Predicting Flight Delays" in 2012 (Lawson & Castillo, 2012), used the same dataset of flights but included several years, which resulted in an impressive number of data points (135 million of flights). However they limited their features to weather data only (33 features only in the end) obtaining 40% recall only.

6. Raj Bandyopadhyay and Rafael Guerrero, in their project "Predicting airline delays" in 2012 (Bandyopadhyay&Guerrero,2012),focused only on one airport and one airline company for 2 years, and only took into account weather data. Even though they were able to achieve a good score, their model is restricted to a very small subset of data and may not generalize well to other airlines and airports.

7. "Modeling Flight Delays and Cancellations in the US" by researchers from the NASA and the FAA (Wang et al., 2009). They 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. They also showed that Neural Networks tend to perform better than "regular" classifiers on this problem due to their ability to grasp the complex structure of the relations.

8. "Characterization and Prediction of Air Traffic Delays" by researchers at MIT used both classification and regression to attack this problem (Rebollo & Balakrishnan, 2014). Their classification was done using a Random Forest classifier, with an absolute test error of 19% on quite small sets (training + test = 4000 samples).

9. "Analysis of aircraft arrival and departure delay characteristics" by researchers from the NASA analyses what are the main causes of delays (E.R.Mueller & G.B.Chatterji, 2002). They provide good insights on which feature to consider to better predict flight delays: for instance origin airport and hour of departure are very relevant features.

10. The work was developed by S. Lin, C. Lai, C. Wu, and C. Lo[4] “A Trustworthy QoS based Collaborative Filtering Approach for Web Service Discovery”2012 This work proposes a trustworthy two-phase web service discovery mechanism based on QoS and collaborative filtering, which discovers and recommends the needed web services effectively for users in the distributed environment, and also solves the problem of services with incorrect QoS information. In the experiment, the theoretical analysis and simulation experiment results show that the proposed method can accurately recommend the needed services to users, and improve the recommendation quality.

Disadvantage: Communication is not satisfied.

11. The work was developed by G.Trinh, C.Rungie, M.Wright, C.Driesener, and J.Dawes [3] “Predicting Future Purchases with The Poisson Log normal Model” 2008.This paper proposes a model using a log-normal distribution in place of the gamma distribution, hence creating a Poisson log-normal (PLN) distribution. The PLN distribution has stronger theoretical grounding than the NBD as NBD has a natural interpretation relying on the central limit theorem. Empirical analysis of brands in multiple categories shows that the PLN distribution gives better predictions than the NBD.

Disadvantage: Process not a user friendly

Most researches focused on departure or arrival delay by using machine learning method. In this paper, the data model of predicting on-time arrival flight is designed by discovering the data correlation between flight data and weather data by big data analytics. The contribution is to find the pinpoint data of the sea-level pressure in the relation to the flight features. A predictive model that can explain the rationale of data is developed and implemented by using machine learning.

CHAPTER 3

METHODOLOGY

AIM & SCOPE

The primary goal of this project is to predict airline delays caused by various factors. Flight delays lead to negative impacts, mainly economical for commuters, airline industries and airport authorities.

Different sources and affect airports, airlines, enroute airspace or an ensemble of them can induce delays. For analysis purposes, one may assume a simplified system where only one of these actors or any combination of them is considered. On the other hand, only airlines are considered when comparing the performance of two airlines under the same conditions. There are many possibilities to ensemble scopes. This becomes important when studying the dynamics of air transportation systems, mainly when targeting root delay.

HARDWARE REQUIREMENTS

Software engineers use the hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system as the starting point for the system design. It shows what the system does and it should be implemented.

PROCESSOR	:	PENTIUM DUAL CORE
RAM	:	1 GB RAM
MONITOR	:	15" COLOR
HARD DISK	:	120 GB
KEYBOARD	:	STANDARD 102 KEYS
MOUSE	:	3 BUTTON

SOFTWARE REQUIREMENTS

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

OPERATING SYSTEM	:	UBUNTU
PROGRAMMING LANGUAGE	:	SCALA
TOOLS	:	INTELLIJ IDEA
DATABASE	:	HDFS

SOFTWARE FEATURES

In this part various software are discussed which are used to develop the system. The software that are used to develop the system are Apache Spark and Scala

HADOOP DISTRIBUTED FILE SYSTEM-HDFS

Hadoop is one of the most popularly used Big Data Analytics tool in the big data industry. Hadoop is an open-source framework for processing big data and it stores the data on cost-effective commodity hardware. It uses clustered architecture for parallel processing of the data. Hadoop is written in Java.

Apache Hadoop offers a scalable, flexible and reliable distributed computing big data framework for a cluster of systems with storage capacity and local computing power by leveraging commodity hardware. Hadoop follows a Master Slave architecture for the transformation and analysis of large datasets using Hadoop Map Reduce paradigm. The 3 important hadoop components that play a vital role in the Hadoop architecture are :

- Hadoop Distributed File System (HDFS) – Patterned after the UNIX file system
- Hadoop Map Reduce
- Yet Another Resource Negotiator (YARN)

BIG DATA

Recent years have witnessed a dramatic increase in our ability to collect data from various sensors, devices, in different formats, from independent or connected applications. This data has outpaced our capability to process, analyze, store and understand these datasets. Consider the Internet data. The web pages indexed by Google were around one million in 1998, but quickly reached 1 billion in 2000 and have already exceeded 1 trillion in 2008. This rapid expansion is accelerated by the dramatic increase in acceptance of social networking applications, such as Facebook, Twitter etc., that allow users to create contents.

The data used in this research is available publicly. Since it is the hourly data of the last ten years, the data is huge in quantity. Hence, it has to be managed cleverly and all of it must be taken into consideration. The study of the weather is carried out keeping the flights and their journeys in mind. This enables the researchers to understand the effect of weather on a particular journey. This is a very unique study. It will help in predicting the delays or preventing them in certain cases.



Fig 3.1 Big Data Preview

APPLICATIONS OF BIG DATA

- Improving and Optimizing Cities and Countries
- Financial TRADING
- Improving Sports Performance
- Improving Science and Research

- Optimizing Machine and Device Performance
- Improving Security and Law Enforcement

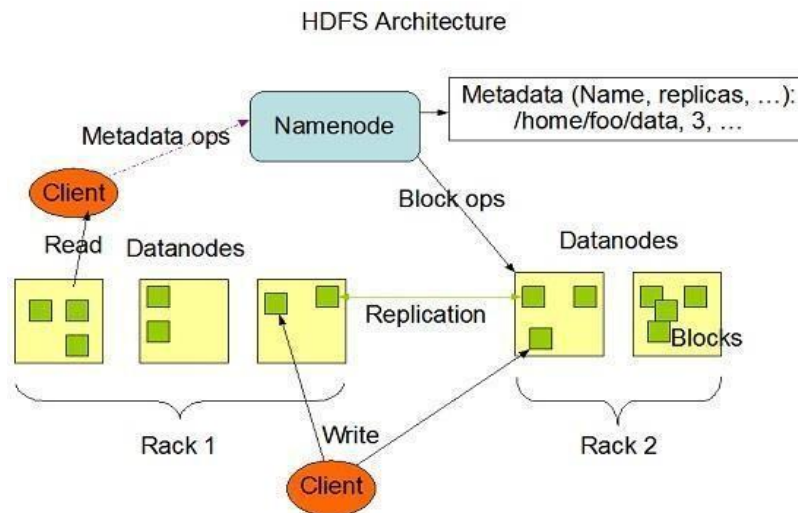


Fig 3.2: HDFS Architecture.

A file on HDFS is split into multiple blocks and each is replicated within the Hadoop cluster. A block on HDFS is a blob of data within the underlying file system with a default size of 64MB. The size of a block can be extended up to 256 MB based on the requirements.

Hadoop follows a master slave architecture design for data storage and distributed data processing using HDFS and Map Reduce respectively. The master node for data storage is Hadoop HDFS is the Name Node and the master node for parallel processing of data using Hadoop Map Reduce is the Job Tracker. The slave nodes in the Hadoop architecture are the other machines in the Hadoop cluster which store data and perform complex computations. Every slave node has a Task Tracker daemon and a Data Node that synchronizes the processes with the Job Tracker and Name Node respectively. In Hadoop architectural implementation the master or slave.

UBUNTU

Ubuntu(pronounced oo-BOON-too) is an open source Debian-based Linux distribution. Sponsored by Canonical Ltd., Ubuntu is considered a good distribution for beginners. The operating system was intended primarily for personal computers (PCs) but it can also be used on servers.

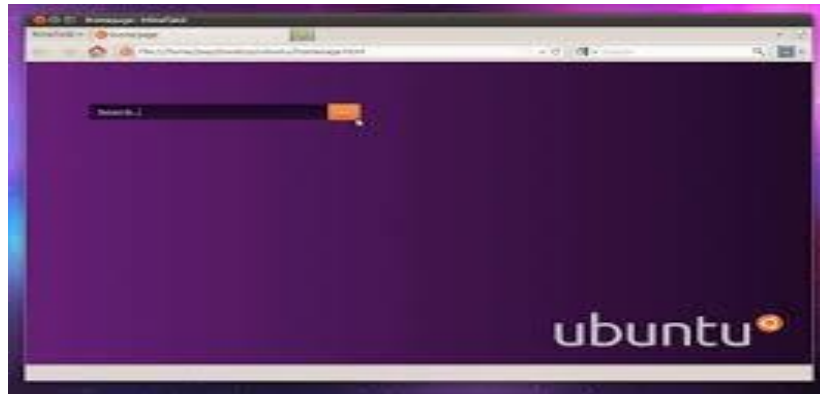


Fig 3.3 Ubuntu Home Page

FEATURES OF UBUNTU

A default installation of Ubuntu contains a wide range of software that includes LibreOffice, Firefox, Thunderbird, Transmission, and several lightweight games such as Sudoku and chess. Many additional software packages are accessible from the built in Ubuntu Software (previously Ubuntu Software Center) as well as any other APT-based package management tools. Many additional software packages that are no longer installed by default, such as Evolution, GIMP, Pidgin, and Synaptic, are still accessible in the repositories still installable by the main tool or by any other APT-based package management tool. Cross-distribution snap packages and flat-picks are also available, that both allow installing software, such as some of Microsoft's software, in most of the major Linux operating systems (such as any currently supported Ubuntu version and in Fedora). The default file manager is GNOME Files, formerly called Nautilus.

UBUNTU FRAMEWORK

A snap can be strictly confined, which means that it operates in a secure box with only pre-defined points of access to there of the system. For third-party applications, this means that user will have a very high level of confidence that the app can only see

appropriate data that user have provided to it. Snaps can also be 'classic' which means that they behave more like debs, and can see everything on your system. User should make sure user have a high level of confidence in the publisher of any classic snap user install, since a compromise or bad faith behaviour in that code is not confined to the app itself. It is also common to consume Ubuntu as an image on a public cloud, or as a container. Ubuntu is published by Canonical on all major public clouds, and the latest image for each LTS version will always include security updates rolled up to at most two weeks ago. User may benefit from installing newer updates than that, but the base image user boot on the cloud should always be the current one from Canonical to ensure that it is broadly up to date and the number of updates needed for full security is minimal.

Canonical also publishes a set of images and containers that user can download for use with VMware or other local hypervisors and private cloud technologies. These include standard Ubuntu images on the Docker Hub, and standard images for use with LXD and MAAS. These images are also kept up to date, with publication of rolled up security updated images on a regular cadence, and user should automate your use of the latest images to ensure consistent security coverage for your users.

APACHE SPARK

Apache Spark can process data from a variety of data repositories, including the Hadoop Distributed File System (HDFS), NoSQL databases and relational data stores, such as Apache Hive. Spark supports in-memory processing to boost the performance of big data analytics applications, but it can also perform conventional disk-based processing when data sets are too large to fit into the available system memory. The Spark Core engine uses the resilient distributed data set, or RDD, as its basic data type.

The RDD is designed in such a way so as to hide much of the computational complexity from users. It aggregates data and partitions it across a server cluster, where it can then be computed and either moved to a different data store or run through an analytic model.

The user doesn't have to define where specific files are sent or what computational resources are used to store or retrieve files. In addition, Spark can handle more than the batch processing applications that Map Reduce is limited to running.



Fig 3.4 Spark

SPARK LIBRARIES

The Spark Core engine functions partly as an application programming interface(API) layer and under pins a set of related tools form an aging and analyzing data. Aside from the Spark Core processing engine, the Apache Spark API environment comes packaged with some libraries of code for use in data analytics applications. These libraries include:

- Spark SQL - One of the most commonly used libraries, Spark SQL enables users to query data stored in disparate applications using the common SQL language.
- Spark Streaming - This library enables users to build applications that analyze and present data in real time.
- M Llib - A library of machine learning code that enables users to apply advanced statistical operations to data in their Spark cluster and to build applications around these analyses.
- Graph X- A built-in library of algorithms for graph-parallel computation.

SCALA

Scala is a general-purpose programming language, and for the vast majority of tasks it fundamentally doesn't matter too much which language user pick. It's really a matter of style. Scala is a great language if user want an IDE that can warn user about errors almost immediately, even before user compile your code, and definitely before user unit. That makes Scala particularly good at tasks like writing large analytics or machine learning engines. It's very hard to write effective unit tests for these types of applications, so user generally want to rely on other methods to prevent errors, such as types and property-based tests. Scala is great for this—at my job, we have an application

that we had to spend 3 months with 6 people building before getting any user feedback, and despite that we only had one real bug in the logic when it was released in beta (and that bug was fixed with in a day.)

Scala is not the only attempt to create a “better Java”. Alternatives such as Kotlin and Ceylon have also gone down that path, but they made the fundamental decision to remain very close in syntax to the Java language itself, so as to minimize the learning curve. This may seem like a great idea, but it is ultimately somewhat self-defeating in that it forces user to stay within a number of those very same Java paradigms that were the reason for wanting to create a “better Java” in the first place.

In contrast, Scala was created specifically with the goal of being a better language, shedding those aspects of Java which it considered restrictive, overly tedious, or frustrating for the developer. As a result, there are indeed code distinctions and paradigm shifts that can make early learning of Scala programming a bit more difficult, but the result is a much cleaner and well organized language that is ultimately easier to use and increases productivity.

METHODS

The flight delay prediction problem may be modelled in many ways, depending on the objectives of the research. Methods were divided into six groups, The numbers next to each category represent the number of related papers.

3.5.1. FLIGHT DATA

The type of datasets from the air transportation system are mainly related to airlines, airports or ensemble. Since airlines and airports commonly do not share their databases with the entire community, they are often used by collaborators of those institutions. Ensemble datasets may include both carriers, airports, and additional information provided by governmental agencies, regulatory authorities, and service providers.

predicted_arr_delay	carrier	origin	dest	dep_delay	taxi_out	distance
-8.311	AA	DFW	LAX	-3	18	1,235
-7.32	AA	DFW	LAX	-2	18	1,235
-6.329	AA	DFW	LAX	-1	18	1,235
-5.339	AA	DFW	LAX	0	18	1,235
-4.348	AA	DFW	LAX	1	18	1,235
-3.357	AA	DFW	LAX	2	18	1,235
-2.367	AA	DFW	LAX	3	18	1,235
-1.376	AA	DFW	LAX	4	18	1,235
-0.385	AA	DFW	LAX	5	18	1,235
0.605	AA	DFW	LAX	6	18	1,235
1.596	AA	DFW	LAX	7	18	1,235
2.587	AA	DFW	LAX	8	18	1,235
3.577	AA	DFW	LAX	9	18	1,235
4.568	AA	DFW	LAX	10	18	1,235

Fig 3.5 Table of Flight Data

PREDICTIVE MODEL

Flight data and weather data are used to propose and create two models of on-time arrival with weather data and without weather data because Japan Meteorological Agency does not offer the future weather data. And four models of delayed arrival with weather data and without weather data, cancel with weather data and without weather data are prepared in order to evaluate the models. The predictive models are implemented by SVM Classifier, Gradient Boosting Classifier, Random Forest Classifier, AdaBoost in machine learning library of scikit-learn, and those models' performance are evaluated. In generating each predictive model, balanced parameter was set to the class weight parameter in order to cope with unbalanced data, and hyper parameters were adjusted by grid search.

STATISTICAL ANALYSIS

Statistical analysis usually encompasses the use of regression models, correlation analysis, econometric models, parametric tests, non-parametric tests, and multivariate analysis (MVA). When it comes to regression models, both delay multiplier and recursive models can help airlines to understand delay propagation effects through the network and to estimate the costs of delays. Some works focus on statistical inference. Pathomsiri et al. used a non-parametric function to evaluate the efficiency of airports of the United States regarding delays. Reynolds et al.(103) computed the

correlation between levels of delays and capacities of the European airports. They also suggested different approaches to deal with the congestion problem, describing their advantages and disadvantages.

DATA CORRELATION

In big data mining, first, big data with characteristics called 3 V is collected. 3 V is Volume which is data volume, Variety which is diversity of data to handle, and Velocity which is speed to generate data. In addition to 3 V, Veracity which is the data accuracy and Value which is data used for decision making are collected as 5 V. Second, those collected data is analyzed to find some pattern such as pattern recognition and predictive analysis. Third, any correlation between collected data or classified data should be discovered.

EXPLORING DELAY DISTRIBUTION

The first step to analyze delay pattern is to explore the distribution. By defining a space with x coordinate of day of week and y coordinate of hour, the delay is added to each hour unit as an attribute. The delay distribution can be plotted with different colors as the delay minutes. The time when an intense delay occurred is recognized in the distribution map. 3D trend analysis can be used to visualize the departure delay distribution and trend in the temporal dimension.

FLIGHT DELAY PREDICTION

The main problems related to flight delay prediction are identified and organized in a taxonomy. It includes scopes, models, and ways of handling flight delay prediction problem. It considers flight domain features, such as problem and scope, and Data Science perspectives, such as data and methods. Figure 2 depicts the entire taxonomy while next subsections describe each component of the taxonomy and Related work. Regarding the available literature on flight delay prediction, we have conducted a systematic mapping study.

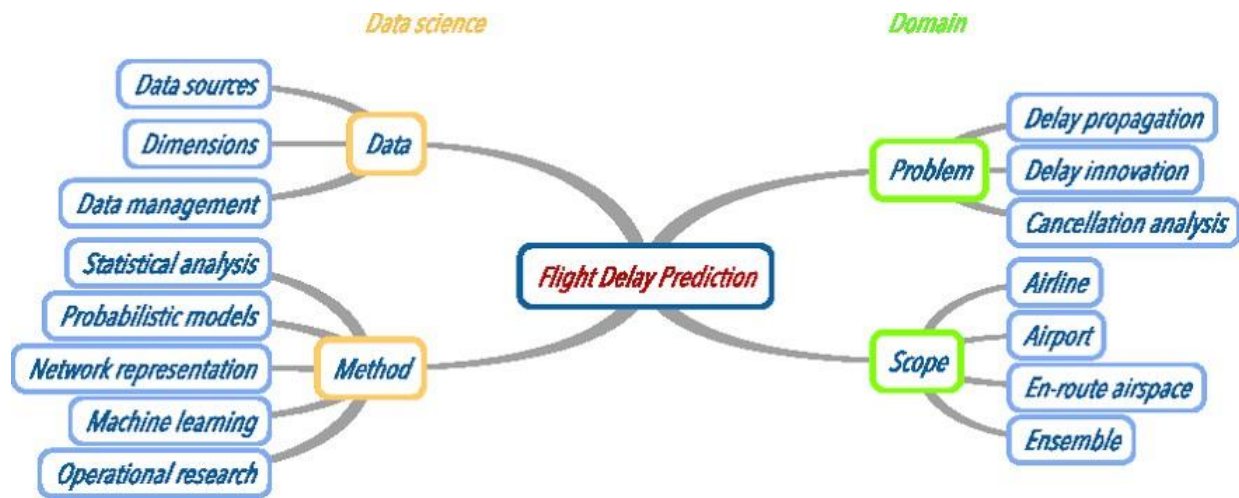


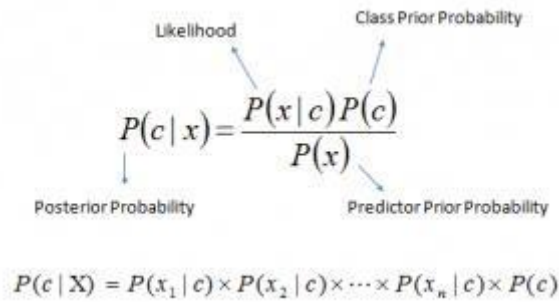
Fig 3.6 Flight Delay Prediction

ALGORITHMS

NAIVE BAYES CLASSIFICATION

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

- In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters.
- Naive Bayes learners and classifiers can be extremely fast compared to methods that are more sophisticated. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.



$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$

The following code helps in predicting the probability of ratio wrong and correct so that the flights are filtered those are delayed and not delayed according to the count of flight delays.

```
val correct = lp.filter($"label" === $"prediction").count()
val wrong = lp.filter(not($"label" === $"prediction")).count()
val ratioWrong = wrong.toDouble / counttotal.toDouble
val ratioCorrect = correct.toDouble / counttotal.toDouble
val truep = lp.filter($"prediction" === 0.0).filter($"label" === $"prediction").count() / count
total.toDouble
val truen = lp.filter($"prediction" === 1.0).filter($"label" === $"prediction").count() / count
total.toDouble
val falsep = lp.filter($"prediction" === 0.0).filter(not($"label" === $"prediction")).count() /
count total.toDouble
val falsen = lp.filter($"prediction" === 1.0).filter(not($"label" === $"prediction")).count()
/ count total.toDouble
```

K-MEANS ALGORITHM

K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. Partition of objects into k non-empty subsets. Identifying the cluster centroids (mean point) of the current partition. Assigning each point to a specific cluster. Compute the distances from each point and allot points to the cluster where the distance from the centroid is minimum.

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K . The k -means algorithm is a centroid-based clustering algorithm. K-means

partitions n observations into k clusters which minimize the within-cluster sum of squares. K means minimizes the following objective function of the clusters.

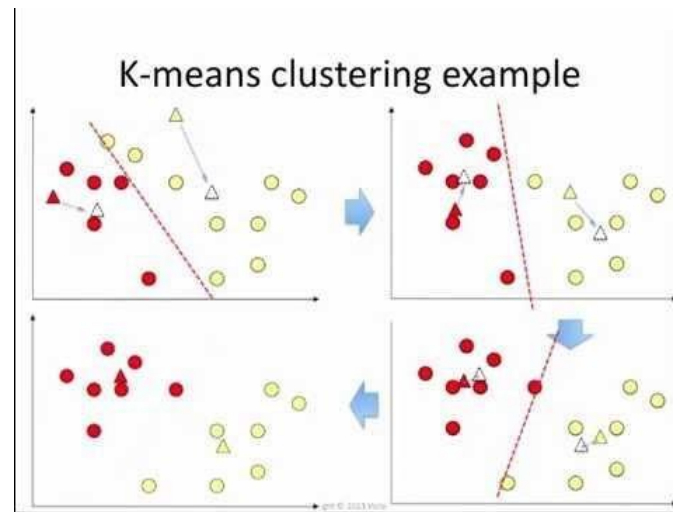


Fig 3.7 Clustering Example

MODULES

1. CLUSTERING MODEL

Descriptive analytics, which embraces elements from unsupervised machine learning (e.g. clustering, outlier detection), pattern recognition and statistics to discover regular patterns in data. In essence, models falling within this category attempt at summarizing, condensing and/or explaining the collected data, which is done by removing all pointless information and by inferring regularities (i.e. patterns) in the form of groups or fitted statistical distributions.

2. FEATURE SELECTION

Predictive analytics, where learning algorithms are applied over supervised data examples to capture the relationship between a set of observed features or predictors and a target variable. Once trained, such models are capable of predicting the value of an unknown target variable for new input data, not necessarily equal to any of the historical examples upon which the model was built. Applications of these learning models are manifold in the transportation realm, including self-driving (e.g. in real-time perception and prediction of traffic scenes

3. ANALYTICS USING MODEL

Prescriptive analytics, which suggest the best action or decision to be made among a range of possibilities by virtue of optimization techniques, expert systems and other elements from Computational intelligence, Mathematical Programming and Operations Research. To this end, prescriptive analytics take advantage of the knowledge gained by their prior descriptive and predictive counterparts

STEP BY STEP DESCRIPTION OF OPERATION

1. DATA COLLECTION

The different sensors like wind sensor, rain sensor and thermo-hygro sensor records different parameters like rainfall, wind, humidity and temperature. The recorded data is stored in the form of datasheet. This data set is send for pre-processing and then to the statistical software.

2. PRE-PROCESSING

The pre processing step is used to remove the unwanted data or noise recorded by the sensors during transmission or it may refer to the selection of a particular area for consideration for prediction purpose.

3. DATA TRANSFER

The recorded data is transferred to the Statistical Software in order to give an input data.

4. DATA PROCESSING

The Data Mining Technique is to be applied to the transferred data in order to validate data. This technique will be implemented by using Statistical Data Miner Software and by quantitative analysis. Quantitative Analysis is the process of presenting and interpreting numerical data. It can allow for greater objectivity and accuracy of results. Generally, quantitative methods are designed to provide summaries of data that support generalizations about the phenomenon under study. In order to accomplish this,

quantitative research usually involves few variables and many cases, and employs prescribed procedures to ensure the value validity and reliability of the data.

5. CLASSIFICATION

After predicting data, what will be the weather in upcoming future after some periods, the Classification will take place. In Classification , it will display what will be the future weather, whether it will be sunny day or rainy or cloudy day what will be the change in speed of wind, humidity etc. the Classification Technique will help for taking some prevention from the climatic hazards that are identified during classification of flight delays.

UML(UNIFIED MODELING LANGUAGE)

Design Engineering deals with the various UML [Unified Modeling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

USE CASE DIAGRAM

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).Hence use case diagrams are used for description of flight delay prediction.

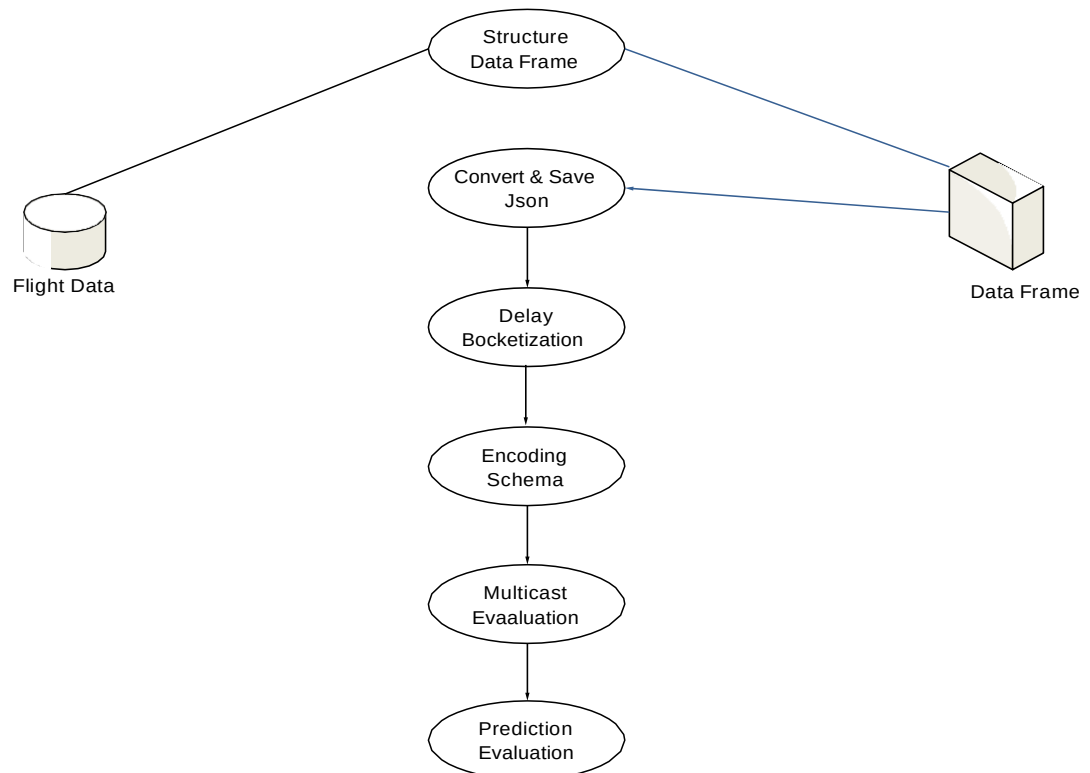


Fig 3.8 Use Case Diagram

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors. An actor is represents a user or another system that will interact with the system modeled. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

SEQUENCE DIAGRAM

A sequence diagram shows an interaction arranged in time sequence. It shows object partitioning in interaction by the message they exchange that are arranged in time sequence. Vertical dimensions represents time and horizontal dimension represent object.

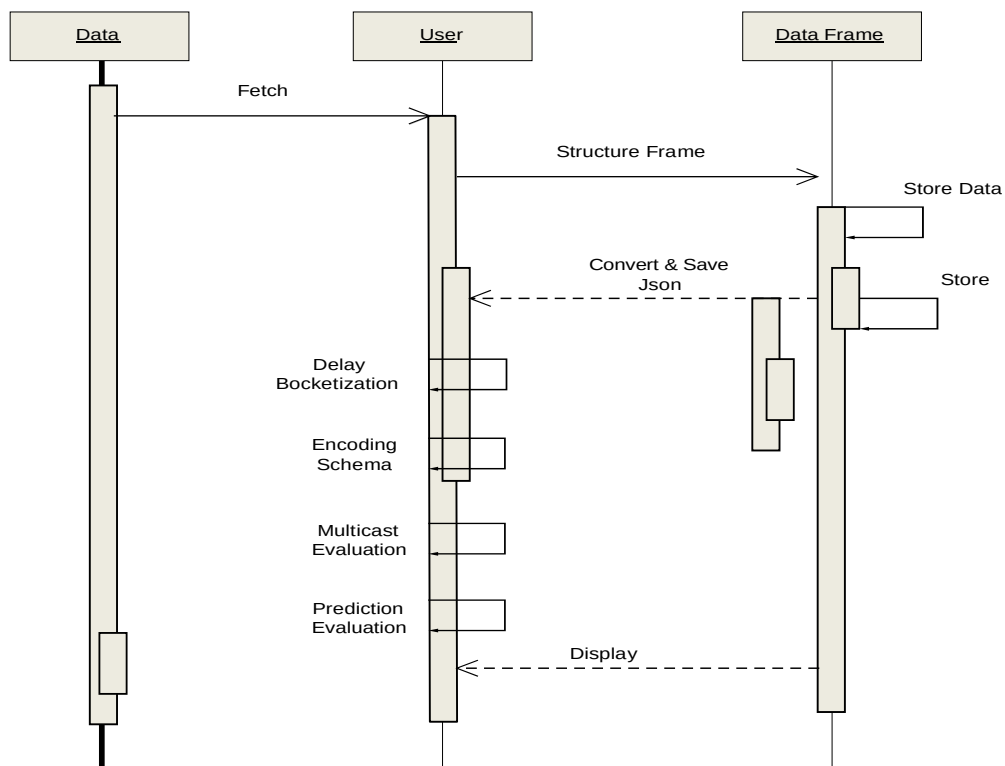


Fig 3.9 Sequence Diagram

CHAPTER 4

RESULTS AND DISCUSSIONS

The below figure depicts the percentage of delayed flights associated to a single delay cause or a combination of them. For example 13.2% of delayed flights are only due to air carrier delays, 11.9% due to combination of late-arriving aircraft and NAS, or 8.9% due to combination of air carrier delay, late-arriving aircraft and NAS.

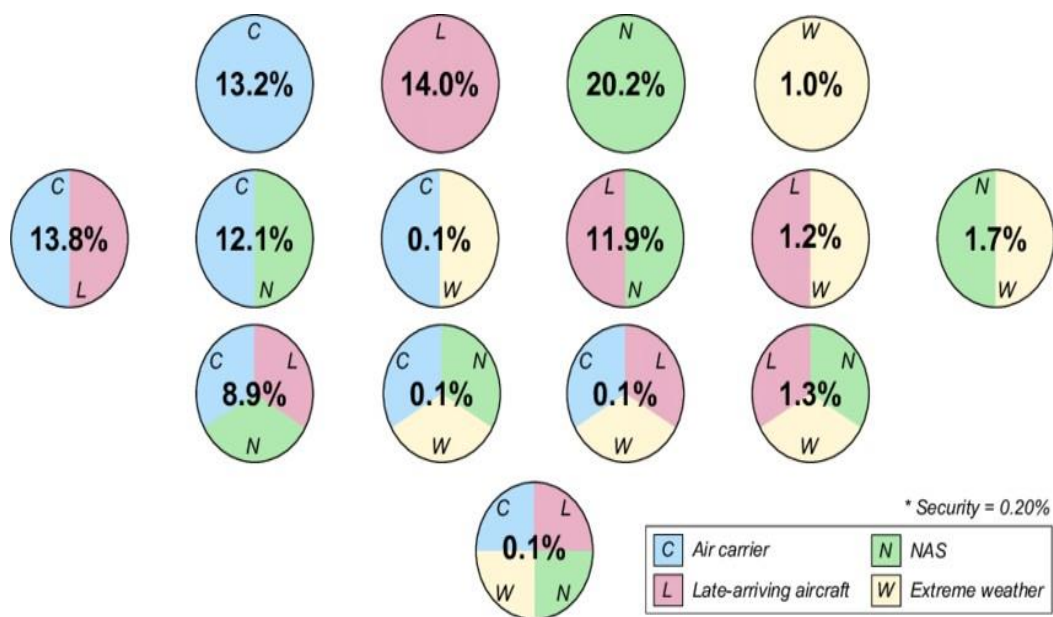


Fig 4.1 Delayed Flights

Due to a single delay cause or a combination of them.

- **Air carrier:** The cause of delay was due to circumstances within the airline's control(e.g. maintenance or crew problems, aircraft cleaning, baggage loading, fueling).
- **Late-arriving aircraft:** A previous flight with the same aircraft arrived late, so causing the present flight to depart late.
- **National Aviation System (NAS):** Delays due to the National Aviation System that refer to a large set of conditions, such as non-extreme weather conditions, airport operations, heavy traffic volume, and air traffic control.

- **Extreme weather:** Significant meteorological conditions (actual or forecast) that, in the judgment of the carrier, delays or prevents the operation of a flight such as tornado, blizzard or hurricane.
- **Security:** Delays caused by evacuation of a terminal, re-boarding of aircraft because of security breach, inoperative screening equipment and/or long lines in excess of 29 minutes at screening areas

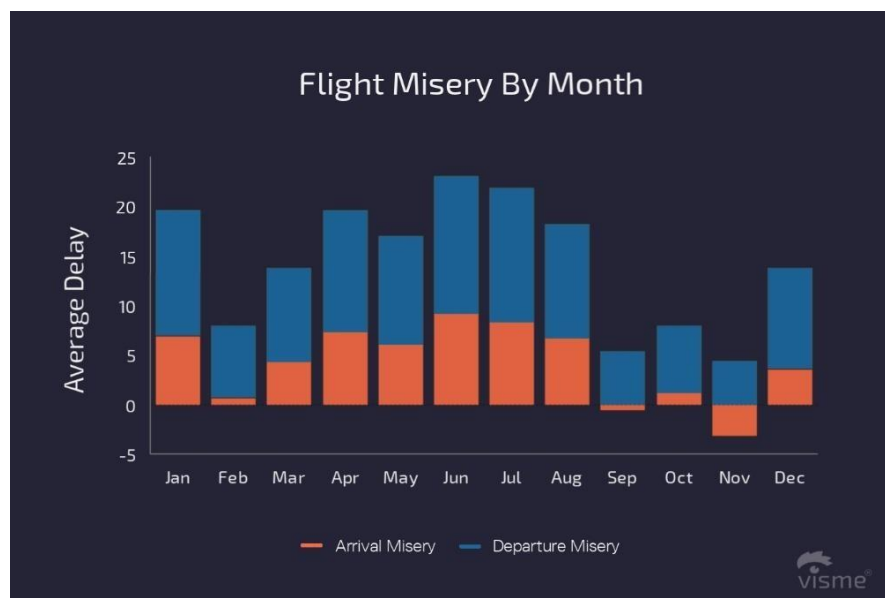


Fig 4.2 Arrival Time vs Average Delay

Since flight delays cause economic consequences to passengers and airlines, recognizing them through prediction may improve marketing decisions. Due to that, several forecast models have been built over the last twenty years. These models have ought to understand how delays propagate through the network of flights or airports, to predict root delay in the system or to comprehend the cancellation process. Beyond these three points of view for treating the flight delay prediction problem, models could also differ by their scope of application, data issues and methods.

ROOT OF FLIGHT DELAY

Considering that new delay (root delay) may happen eventually, these root delays impair the performance of transportation network. Researchers create prediction models

to tackle root delay, predicting when and where a delay will occur and what are its reasons and sources. This includes models that efficiently seek to estimate the number of minutes, probability or level of delay for a specific flight, airline or airport. A relevant number of works focused on predicting and estimating delay duration. Some approaches considered probabilistic models and innovation distribution, whereas others find conditions for the occurrence of a root delay, such as passenger demand, fares, flight frequency, aircraft size, and taxi-out time. Particular circumstances, such as weather conditions, acts of God, aircraft problems, may lead airlines to cancel flights. Besides, airlines may directly cancel a flight, when factors like seat occupancy or cost savings are taking into consideration.

DELAY PROPAGATION

In delay propagation, the primary objective is to understand how delay propagates through airlines and airports based on the assumption that an initial delay has already occurred in the transportation system. A particular scenario happens when delays are spread to other flights of the same airline as chain reactions. Under this situations, it is important to measure how stable and reliable carriers can be to recover from delay propagation. Also, a delay may continue to propagate due to the scheduling of critical resources or retentions in other airports. When scheduled time for take-off or landing is not fulfilled, flights need new slots that may be unavailable. In this scenario, it is important to understand the effects that a root delay in flight may produce to both departure and arrival airports. Such phenomenon may increase the number of flights at some period, generating capacity problems and queues.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

CONCLUSION

The main goal of this work is to predict several days in advance the arrival delay of a scheduled flight due to weather conditions. Accurate prediction of flight delays is an important problem to be addressed given the economic impact of flight delay to both airlines and travelers. In our model, the predicted arrival delay takes into consideration both flight information (origin airport, destination airport, scheduled departure and arrival time) and weather conditions at origin airport and destination airport according to the flight timetable.

Two open datasets of airline flights and weather observations have been analyzed to discover initial insights, evaluate the quality of data, and identify potentially interesting subsets. Then, data cleaning and transformation (joining and balancing operations) have been performed to make data ready for modeling. Finally, a scalable parallel version of the Random Forest data classification algorithm has been developed, iteratively calibrating its settings to optimize results in terms of accuracy and recall. The data preparation and mining tasks have been implemented as MapReduce programs that have been executed on a Cloud infrastructure to achieve scalability. The results show a high accuracy in prediction of delays above a given threshold.

For instance, with a delay threshold of 60 minutes we achieve an accuracy 85.8% and a delay recall of 86.9%. We have obtained such good performance results considering different weather observations at origin and destination airports and selecting flights that are really delayed by weather conditions. Moreover, if weather conditions are not considered the model achieves an accuracy of 69.1%. Therefore, the proposed methodology identifies a very useful pattern of flight delay that may help airlines in reducing delays.

FUTURE ENHANCEMENT

Generate new features that capture the nuances of the co-dependence between a flight's delay and the next flights delay (as of now, we only account for first order time dependencies, but the interactions in reality are more complex). Use additional features (twitter data, finer resolution terror data, etc.) to account for factors that contribute to delays but are invisible by our algorithms as of now due to the absence of proxy variables. Improve our neural network. Due to lack of time, we could not explore as much as we wanted to neural network approach, but results already were promising using only out-of-the-box algorithms. Use more of the already available data to improve our estimates on the delayed class.

REFERENCES

- [1] Shawn Allan, JA Beesley, Jim Evans, and Steve Gaddy. 2001. Analysis of delay causality at Newark International Airport. In 4th USA/Europe Air Traffic Management R&D Seminar.
- [2] Michael Ball, Cynthia Barnhart, Martin Dresner, Mark Hansen, Kevin Neels, Amedeo Odoni, Everett Peterson, Lance Sherry, Antonio A Trani, and Bo Zou. 2010. Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the United States. (2010).
- [3] Michael Callaham, James DeArmon, Arlene Cooper, Jason Goodfriend, Debra Moch-Mooney, and George Solomos. 2001. Assessing NAS performance: Normalizing for the effects of weather. In 4th USA/Europe Air Traffic Management R&D Symposium.
- [4] Pablo Fleurquin, Jose J Ramasco, and Victor M Eguiluz. 2013a. Data-driven modeling of systemic delay propagation under severe meteorological conditions. ArXiv e-prints (2013).
- [5] Pablo Fleurquin, Jose J Ramasco, and Victor M Eguiluz. 2013b. Systemic delay propagation in the US airport network. Scientific reports 3 (2013).
- [6] Alexander Klein, Richard Jehlen, and Diana Liang. 2007. Weather Index With Queuing Component For National Airspace System Performance Assessment. In FAA/Eurocontrol ATM Sem. Barcelona, Spain.
- [7] Alexander Klein, Sadegh Kavoussi, and Robert S Lee. 2009. Weather forecast accuracy: study of impact on airport capacity and estimation of avoidable costs. In Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009).
- [8] Eric R Mueller and Gano B Chatterji. 2002. Analysis of aircraft arrival and departure delay characteristics. In AIAA aircraft technology, integration and operations (ATIO) conference.

- [9] Banavar Sridhar, Yao Wang, Alexander Klein, and Richard Jehlen. 2009. Modeling Flight Delays and Cancellations at the National, Regional and Airport Levels in the United States. In 8th USA/Europe ATM R&D Seminar, Napa, California (USA).
- [10] Domenico Talia and Paolo Trunfio. 2012. Service-oriented distributed knowledge discovery. Chapman and Hall/CRC. ISBN 978-1-4398-7531-5.
- [11] Yao Wang and Deepak Kulkarni. 2011. Modeling Weather Impact on Ground Delay Programs. Technical Report. SAE Technical Paper.
- [12] Tom White. 2009. Hadoop: The Definitive Guide (1st ed.). O'Reilly Media, Inc.
- [13] Ning Xu, George Donohue, Kathryn Blackmond Laskey, and Chun-Hung Chen. 2005. Estimation of delay propagation in the national aviation system using Bayesian networks. In 6th USA/Europe Air Traffic Management Research and Development Seminar.
- [14] Ning Xu, Lance Sherry, and Kathryn Blackmond Laskey. 2008. Multifactor model for predicting delays at us airports. *Transp. Research Record: Journal of the Transportation Research Board* 2052, 1 (2008), 62–71.
- [15] Antanas Verikas, Adas Gelzinis, and Marija Bacauskiene. 2011. Mining data with random forests: A survey and results of new tests. *Pattern Recognition* 44, 2 (2011), 330–349.
- [16] Gabor Takacs. 2014. Predicting flight arrival times with a multistage model. In *Big Data (Big Data)*, 2014 IEEE International Conference on. 78–84.
- [17] Juan Jose Rebollo and Hamsa Balakrishnan. 2014. Characterization and prediction of air traffic delays. *Transportation Research Part C: Emerging Technologies* 44 (2014), 231–241.

- [18] Nikolas Pyrgiotis, Kerry M Malone, and Amedeo Odoni. 2013. Modelling delay propagation within an airport network. *Transportation Research Part C: Emerging Technologies* 27 (2013), 60–75.
- [19] Shervin AhmadBeygi, Amy Cohn, Yihan Guan, and Peter Belobaba. 2008. Analysis of the potential for delay propagation in passenger airline networks. *Journal of air transport management* 14, 5 (2008), 221–236.
- [20] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, and others. 2006. Handling imbalanced datasets: A review. *GESTS Int. Trans. on Computer Science and Engineering* 30, 1 (2006), 25–36.
- [21] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
- [22] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113.
- [23] Spyros Blanas, Jignesh M Patel, Vuk Ercegovic, Jun Rao, Eugene J Shekita, and Yuanyuan Tian. 2010. A comparison of join algorithms for log processing in mapreduce. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 975–986.
- [24] FlightCaster 2009. How FlightCaster Squeezes Predictions from Flight Data. <http://www.datawrangling.com/how-flightcaster-squeezes-predictions-from-flight-data>. (2009).
- [25] Federal Meteorological Handbook 2005. Federal Meteorological Handbook No. 1 (FMH-1), "Surface Weather Observations and Reports". <http://www.ofcm.gov/fmh-1/pdf/FMH1.pdf>. (2005).

APPENDICES

A. SAMPLE CODE

```
import org.apache.spark._
import org.apache.spark.ml._
import org.apache.spark.ml.feature._
import org.apache.spark.ml.classification._
import org.apache.spark.ml.evaluation._
import org.apache.spark.ml.tuning._
import org.apache.spark.sql._
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._

object Flight {
  case class Flight(_id: String, doFW: Integer, carrier: String, origin: String,
    dest: String, crsdephour: Integer, crsdeptime: Double, depdelay: Double,
    crsarrrtime: Double, arrdelay: Double, crselapsedtime: Double, dist: Double)
    extends Serializable

  val schema = StructType(Array(StructField("_id", StringType, true),
    StructField("doFW", IntegerType, true),
    StructField("carrier", StringType, true),
    StructField("origin", StringType, true),
    StructField("dest", StringType, true),
    StructField("crsdephour", IntegerType, true),
    StructField("crsdeptime", DoubleType, true),
    StructField("depdelay", DoubleType, true),
    StructField("crsarrrtime", DoubleType, true),
    StructField("arrdelay", DoubleType, true),
    StructField("crselapsedtime", DoubleType, true),
    StructField("dist", DoubleType, true)))
```

```

class Flight(_id: String, doFW: Integer, carrier: String, origin: String,
  dest: String, crsdephour: Integer, crsdeptime: Double, depdelay: Double,
  crsarrrtime: Double, arrdelay: Double, crselapsedtime: Double, dist: Double) extends
Serializable
val schema = StructType(Array(
  StructField("_id", StringType, true),
  StructField("doFW", IntegerType, true),
  StructField("carrier", StringType, true),
  StructField("origin", StringType, true),
  StructField("dest", StringType, true),
  StructField("crsdephour", IntegerType, true),
  StructField("crsdeptime", DoubleType, true),
  StructField("depdelay", DoubleType, true),
  StructField("crsarrrtime", DoubleType, true),
  StructField("arrdelay", DoubleType, true),
  StructField("crselapsedtime", DoubleType, true),
  StructField("dist", DoubleType, true)
))
import spark.implicits._
val
df=spark.read.format("json").schema(schema).load("file:///home/jpasolutions/Desktop/
code/flights20170102.json")
df.first()
  df.count()
  df.createOrReplaceTempView("flights")
  spark.catalog.cacheTable("flights")
  df.show
val
test=spark.read.format("json").schema(schema).load("file:///home/jpasolutions/Deskt
op/code/flights20170304.json")
test.count()
  test.first()
  test.show

```

```

println("what is the average departure delay by day of the week")
spark.sql("SELECT doFW, avg(depdelay) as avgdelay FROM flights GROUP BY doFW
ORDER BY avgdelay desc").show
println("what is the count of departure delay by origin airport where delay minutes >40")
spark.sql("select origin, count(depdelay) from flights where depdelay > 40 group by
origin ORDER BY count(depdelay) desc").show
val delaybucketizer=newBucketizer().setInputCol("depdelay")
.setOutputCol("delayed").setSplits(Array(0.0, 40.0, Double.PositiveInfinity))
val df4 = delaybucketizer.transform(df)
df4.groupBy("delayed").count.show
df4.createOrReplaceTempView("flight")
println("what is the count of departure delay and not delayed by origin")
spark.sql("select origin, delayed, count(delayed) from flight group by origin, delayed
order by origin").show
val delaybucketizer = new Bucketizer().setInputCol("depdelay")
.setOutputCol("delayed").setSplits(Array(0.0, 40.0, Double.PositiveInfinity))
val df4 = delaybucketizer.transform(df)
df4.groupBy("delayed").count.show
df4.createOrReplaceTempView("flight")
println("what is the count of departure delay and not delayed by origin")
spark.sql("select origin, delayed, count(delayed) from flight group by origin, delayed
order by origin").show
println("what is the count of departure delay by dest")
spark.sql("select dest, delayed, count(delayed) from flight where delayed=1 group by
dest, delayed order by dest").show
println("what is the count of departure delay by origin, dest")
spark.sql("select origin,dest, delayed, count(delayed) from flight where delayed=1
group by origin,dest, delayed order by origin,dest").show
println("what is the count of departure delay by doFW")
spark.sql("select doFW, delayed, count(delayed) from flight where delayed=1 group
by doFW, delayed order by doFW").show

```

```

println("what is the count of departure delay by hour where delay minutes >40")
spark.sql("select crsdephour, delayed, count(delayed) from flight where delayed=1
group by crsdephour, delayed order by crsdephour").show
println("what is the count of departure delay carrier where delay minutes >40")
spark.sql("select carrier, delayed, count(delayed) from flight where delayed=1 group
by carrier, delayed order by carrier").show
val fractions = Map(0.0 -> .29, 1.0 -> 1.0)
val strain = df4.stat.sampleBy("delayed", fractions, 36L)
strain.groupBy("delayed").count.show
val categoricalColumns = Array("carrier", "origin", "dest", "dofW")
val stringIndexers = categoricalColumns.map { colName =>
  new StringIndexer()
    .setInputCol(colName)
    .setOutputCol(colName + "Indexed")
    .fit(df)
}
val encoders = categoricalColumns.map { colName =>
  new OneHotEncoder()
    .setInputCol(colName + "Indexed")
    .setOutputCol(colName + "Enc")
}
val labeler = new Bucketizer().setInputCol("depdelay")
.setOutputCol("label").setSplits(Array(0.0, 40.0, Double.PositiveInfinity))
val featureCols = Array("carrierEnc", "destEnc", "originEnc",
  "dofWEnc", "crsdephour", "crselapsedtime", "crsarrrtime", "crsdeptime", "dist")
val assembler = new VectorAssembler()
.setInputCols(featureCols).setOutputCol("features")
val dTree = new DecisionTreeClassifier()
.setLabelCol("label").setFeaturesCol("features").setMaxBins(7000)
val steps = stringIndexers ++ encoders ++ Array(labeler, assembler, dTree)
val pipeline = new Pipeline().setStages(steps)

```

```

    val paramGrid = new ParamGridBuilder().addGrid(dTree.maxDepth, Array(4, 5,
6)).build()
val evaluator = new MulticlassClassificationEvaluator()
.setLabelCol("label").setPredictionCol("prediction").setMetricName("accuracy")
    // Set up 3-fold cross validation with paramGrid
    val crossval = new CrossValidator()
.setEstimator(pipeline).setEvaluator(evaluator).setEstimatorParamMaps(paramGrid).s
etNumFolds(3)
    val ntrain = strain.drop("delayed").drop("arrdelay")
    println(ntrain.count)
    ntrain.show
    val cvModel = crossval.fit(ntrain)
//note if you want to do analyze the decision tree it is easier not to use OneHotEncoding
    // val featureImportances = bestModel.stages.last
.asInstanceOf[RandomForestRegressionModel].featureImportances.toArray
    // val treeModel = cvModel.bestModel.
asInstanceOf[org.apache.spark.ml.PipelineModel].stages.last.asInstanceOf[DecisionTr
eeClassificationModel]
    // val featureImportances = treeModel.featureImportances.toArray
    // val fis = s"features importances:\n ${featureCols.zip(featureImportances).map(t =>
s"\t${t._1} = ${t._2}")}.mkString("\n")}\n"
    // println(fis)
    val predictions = cvModel.transform(test)
    val accuracy = evaluator.evaluate(predictions)
    val lp = predictions.select("label", "prediction")
    val counttotal = predictions.count()
    val label0count = lp.filter($"label" === 0.0).count()
    val pred0count = lp.filter($"prediction" === 0.0).count()
    val label1count = lp.filter($"label" === 1.0).count()
    val pred1count = lp.filter($"prediction" === 1.0).count()
    val correct = lp.filter($"label" === $"prediction").count()
    val wrong = lp.filter(not($"label" === $"prediction")).count()
    val ratioWrong = wrong.toDouble / counttotal.toDouble

```

```

    val ratioCorrect = correct.toDouble / counttotal.toDouble
    val truep = lp.filter($"prediction" === 0.0).filter($"label" === $"prediction").count() /
counttotal.toDouble
    val truen = lp.filter($"prediction" === 1.0).filter($"label" === $"prediction").count() /
counttotal.toDouble
    val falsep = lp.filter($"prediction" === 0.0).filter(not($"label" ===
$"prediction")).count() / counttotal.toDouble
    val falsen = lp.filter($"prediction" === 1.0).filter(not($"label" ===
$"prediction")).count() / counttotal.toDouble
    println("ratio correct", ratioCorrect)
    predictions.createOrReplaceTempView("flight")
    println("what is the count of predicted delay/notdelay by scheduled departure hour")
    spark.sql("select crsdephour, prediction, count(prediction) from flight group by
crsdephour, prediction order by crsdephour").show
    println("what is the count of predicted delay/notdelay by origin")
    spark.sql("select origin, prediction, count(prediction) from flight group by origin,
prediction order by origin").show
    println("what is the count of predicted delay/notdelay by destination")
    spark.sql("select dest, prediction, count(prediction) from flight group by dest,
prediction order by dest").show
    println("what is the count of predicted delay/notdelay by origin,dest")
    spark.sql("select origin,dest, prediction, count(prediction) from flight group by
origin,dest, prediction order by origin,dest").show
    println("what is the count of predicted delay/notdelay by day of the week")
    spark.sql("select dofW, prediction, count(prediction) from flight group by dofW,
prediction order by dofW").show
    println("what is the count of predicted delay/notdelay by carrier")
    spark.sql("select carrier, prediction, count(prediction) from flight group by carrier,
prediction order by carrier").show

    // cvModel.write.overwrite().save("/user/user01/data/cfModel")

}}

```

B. SCREEN SHOTS

```
scala>
scala> // Set up 3-fold cross validation with paramGrid
scala> val crossval = new CrossValidator().setEstimator(pipeline).setEvaluator(evaluator).setEstimatorParamMaps(paramGrid).setNumFolds(
3)
20/03/06 05:24:34 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
20/03/06 05:24:34 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
crossval: org.apache.spark.ml.tuning.CrossValidator = cv_0b7c4e172069
scala>
scala> val ntrain = strain.drop("delayed").drop("arrdelay")
ntrain: org.apache.spark.sql.DataFrame = [_id: string, dofw: int ... 9 more fields]
scala> println(ntrain.count)
15113
scala> ntrain.show
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
 _id|dofw|carrier|origin|dest|crsdephour|crsdeptime|depdelay|crsarrrtime|crselapsedtime| dist|
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
AA_2017-01-01_ATL...| 7| AA| ATL| LGA| 17| 1700.0| 0.0| 1912.0| 132.0| 762.0|
AA_2017-01-01_LGA...| 7| AA| LGA| MIA| 7| 700.0| 0.0| 1017.0| 127.0| 1096.0|
AA_2017-01-01_DEN...| 7| AA| DEN| ORD| 12| 1235.0| 0.0| 1609.0| 145.0| 888.0|
AA_2017-01-01_MIA...| 7| AA| MIA| IAH| 20| 2045.0| 80.0| 2230.0| 173.0| 964.0|
AA_2017-01-01_MIA...| 7| AA| MIA| LGA| 20| 2025.0| 0.0| 2325.0| 188.0| 1096.0|
AA_2017-01-01_ORD...| 7| AA| ORD| BOS| 15| 1500.0| 0.0| 1818.0| 130.0| 807.0|
AA_2017-01-01_MIA...| 7| AA| MIA| LGA| 18| 1750.0| 85.0| 2049.0| 173.0| 1096.0|
AA_2017-01-01_MIA...| 7| AA| MIA| DEN| 16| 1621.0| 43.0| 1846.0| 265.0| 1769.0|
DL_2017-01-01_ATL...| 7| DL| ATL| ORD| 15| 1500.0| 4.0| 2152.0| 125.0| 594.0|
DL_2017-01-01_BOS...| 7| DL| BOS| ATL| 15| 1450.0| 6.0| 1750.0| 180.0| 946.0|
DL_2017-01-01_SFO...| 7| DL| SFO| ATL| 6| 610.0| 0.0| 1345.0| 275.0| 2139.0|
DL_2017-01-01_ATL...| 7| DL| ATL| ORD| 15| 1500.0| 41.0| 1605.0| 125.0| 606.0|
DL_2017-01-01_ORD...| 7| DL| ORD| ATL| 12| 1240.0| 27.0| 1543.0| 123.0| 606.0|
DL_2017-01-01_ATL...| 7| DL| ATL| EWR| 19| 1930.0| 25.0| 2145.0| 135.0| 746.0|
DL_2017-01-01_ORD...| 7| DL| ORD| ATL| 10| 1000.0| 2.0| 1300.0| 120.0| 606.0|
DL_2017-01-01_IAH...| 7| DL| IAH| ATL| 6| 600.0| 0.0| 859.0| 119.0| 609.0|
DL_2017-01-01_BOS...| 7| DL| BOS| ATL| 13| 1330.0| 49.0| 1633.0| 183.0| 946.0|
DL_2017-01-01_ATL...| 7| DL| ATL| DEN| 9| 940.0| 0.0| 1102.0| 202.0| 1199.0|
DL_2017-01-01_DEN...| 7| DL| DEN| ATL| 6| 600.0| 10.0| 1102.0| 182.0| 1199.0|
DL_2017-01-01_DEN...| 7| DL| DEN| ATL| 13| 1317.0| 70.0| 1808.0| 171.0| 1199.0|
only showing top 20 rows
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
scala> val cvModel = crossval.fit(ntrain)
cvModel: org.apache.spark.ml.tuning.CrossValidatorModel = cv_0b7c4e172069
scala>
scala> // note if you want to do analyze the decision tree it is easier not to use OneHotEncoding
scala>
scala> // val featureImportances = bestModel.stages.last.asInstanceOf[RandomForestRegressionModel].featureImportances.toArray
```

Fig 4.1 Collected Data Sample

It was showing the entire data of all flights with delayed time, arrival time, distance to be reached, destination time, departure time. First collection of data is most important in this project.

```
scala> println("what is the count of predicted delay/notdelay by scheduled departure hour")
what is the count of predicted delay/notdelay by scheduled departure hour
scala> spark.sql("select crsdephour, prediction, count(prediction) from flight group by crsdephour").show
-----+-----+-----+-----+
|crsdephour|prediction|count(prediction)|
-----+-----+-----+-----+
|0|0.0|188|
|0|1.0|1|
|1|1.0|1|
|1|0.0|29|
|5|0.0|593|
|6|0.0|2043|
|7|0.0|3582|
|8|1.0|4|
|8|0.0|3010|
|9|1.0|64|
|9|0.0|2443|
|10|1.0|161|
|10|0.0|2936|
|11|1.0|277|
|11|0.0|2546|
|12|1.0|197|
|12|0.0|2496|
|13|1.0|95|
|13|0.0|2820|
|14|0.0|2200|
only showing top 20 rows
-----+-----+-----+-----+
scala> println("what is the count of predicted delay/notdelay by origin")
what is the count of predicted delay/notdelay by origin
scala> spark.sql("select origin, prediction, count(prediction) from flight group by origin, prediction order by origin").show
-----+-----+-----+-----+
|origin|prediction|count(prediction)|
-----+-----+-----+-----+
|ATL|1.0|775|
|ATL|0.0|6037|
|BOS|0.0|3910|
|BOS|1.0|335|
|DEN|0.0|4352|
|DEN|1.0|484|
|EWR|0.0|4467|
|IAH|0.0|3479|
|IAH|1.0|400|
|LGA|0.0|5387|
|MIA|1.0|229|
|MIA|0.0|4468|
|ORD|0.0|5878|
|ORD|1.0|923|
|SFO|1.0|354|
|SFO|0.0|3970|
-----+-----+-----+-----+
```

Fig 4.2 Predicted Delay or No Delay

In the above figure we predicted whether there is a delay or no delay by origin and predicted departure hours.

```
scala> spark.sql("select dest, prediction, count(prediction) from flight group by dest, prediction order by dest").show
-----+-----+
|dest|prediction|count(prediction)|
-----+-----+
|ATL|0.0|6836|
|BOS|0.0|4202|
|BOS|1.0|68|
|DEN|1.0|108|
|DEN|0.0|4685|
|EMR|0.0|3322|
|EMR|1.0|1147|
|IAH|0.0|3896|
|IAH|1.0|2|
|LGA|0.0|5364|
|MIA|1.0|7|
|MIA|0.0|4689|
|ORD|0.0|6783|
|ORD|1.0|4|
|SFO|1.0|2164|
|SFO|0.0|2171|
-----+-----+

scala> println("what is the count of predicted delay/notdelay by origin,dest")
what is the count of predicted delay/notdelay by origin,dest
scala> spark.sql("select origin,dest, prediction, count(prediction) from flight group by origin,dest, prediction order by origin,dest")
.show
-----+-----+
|origin|dest|prediction|count(prediction)|
-----+-----+
|ATL|BOS|0.0|820|
|ATL|DEN|0.0|860|
|ATL|EMR|0.0|391|
|ATL|EMR|1.0|508|
|ATL|IAH|0.0|413|
|ATL|LGA|0.0|1373|
|ATL|MIA|0.0|1041|
|ATL|ORD|0.0|905|
|ATL|SFO|1.0|287|
|ATL|SFO|0.0|234|
|BOS|ATL|0.0|818|
|BOS|DEN|0.0|350|
|BOS|EMR|1.0|195|
|BOS|EMR|0.0|394|
|BOS|IAH|0.0|260|
|BOS|LGA|0.0|822|
|BOS|MIA|0.0|412|
|BOS|ORD|0.0|741|
|BOS|SFO|1.0|230|
|BOS|SFO|0.0|113|
-----+-----+
only showing top 20 rows

scala> println("what is the count of predicted delay/notdelay by day of the week")
what is the count of predicted delay/notdelay by day of the week
```

Fig 4.3 Predicted Delay or No Delay by Origin, Destination.

Here in the above figure it is the content of table that displays count of predicted delay or no delay by origin, destination.

```
scala> println("what is the count of departure delay by origin, dest")
what is the count of departure delay by origin, dest
scala> spark.sql("select origin,dest, delayed, count(delayed) from flight where delayed=1 group by origin,dest, delayed order by origin,dest").show
-----+-----+
|origin|dest|delayed|count(delayed)|
-----+-----+
|ATL|BOS|1.0|48|
|ATL|DEN|1.0|44|
|ATL|EMR|1.0|144|
|ATL|IAH|1.0|26|
|ATL|LGA|1.0|158|
|ATL|MIA|1.0|66|
|ATL|ORD|1.0|76|
|ATL|SFO|1.0|89|
|BOS|ATL|1.0|59|
|BOS|DEN|1.0|13|
|BOS|EMR|1.0|45|
|BOS|IAH|1.0|11|
|BOS|LGA|1.0|64|
|BOS|MIA|1.0|24|
|BOS|ORD|1.0|46|
|BOS|SFO|1.0|53|
|DEN|ATL|1.0|54|
|DEN|BOS|1.0|16|
|DEN|EMR|1.0|74|
|DEN|IAH|1.0|40|
-----+-----+
only showing top 20 rows

scala> println("what is the count of departure delay by dofw")
what is the count of departure delay by dofw
scala> spark.sql("select dofw, delayed, count(delayed) from flight where delayed=1 group by dofw, delayed order by dofw").show
-----+-----+
|dofw|delayed|count(delayed)|
-----+-----+
|1|1.0|956|
|2|1.0|720|
|3|1.0|499|
|4|1.0|645|
|5|1.0|594|
|6|1.0|430|
|7|1.0|714|
-----+-----+

scala>
scala> println("what is the count of departure delay by hour where delay minutes >40")
what is the count of departure delay by hour where delay minutes >40
scala> spark.sql("select crsdephour, delayed, count(delayed) from flight where delayed=1 group by crsdephour, delayed order by crsdephour").show
-----+-----+
|crsdephour|delayed|count(delayed)|
-----+-----+
|1|1.0|956|
|2|1.0|720|
|3|1.0|499|
|4|1.0|645|
|5|1.0|594|
|6|1.0|430|
|7|1.0|714|
-----+-----+
```

Fig 4.4 Departure Delay by DOFW.

Here in the above screen shot first we have taken the data of all flights and then count the number of delayed flights by this we are displaying the most rated delayed flight in the order.

```

|crsdephour|delayed|count(delayed)|
|-----|-----|-----|
|0|1.0|9|
|1|1.0|11|
|5|1.0|15|
|6|1.0|72|
|7|1.0|115|
|8|1.0|193|
|9|1.0|179|
|10|1.0|288|
|11|1.0|285|
|12|1.0|237|
|13|1.0|345|
|14|1.0|358|
|15|1.0|336|
|16|1.0|355|
|17|1.0|484|
|18|1.0|486|
|19|1.0|388|
|20|1.0|234|
|21|1.0|164|
|22|1.0|66|
|-----|-----|-----|
only showing top 20 rows

scala>
scala> println("what is the count of departure delay carrier where delay minutes >40")
what is the count of departure delay carrier where delay minutes >40
scala> spark.sql("select carrier, delayed, count(delayed) from flight where delayed=1 group by carrier, delayed order by carrier").show

|carrier|delayed|count(delayed)|
|-----|-----|-----|
|AA|1.0|783|
|DL|1.0|1065|
|UA|1.0|2458|
|WN|1.0|252|
|-----|-----|-----|

scala>
scala> val fractions = Map(0.0 -> .29, 1.0 -> 1.0)
fractions: scala.collection.immutable.Map[Double,Double] = Map(0.0 -> 0.29, 1.0 -> 1.0)
scala> val strain = df4.stat.sampleBy("delayed", fractions, 36L)
strain: org.apache.spark.sql.DataFrame = [_id: string, dofw: int ... 11 more fields]
scala> strain.groupBy("delayed").count.show

|delayed|count|
|-----|-----|
|0.0|10555|
|1.0|4558|
|-----|-----|

```

Fig 4.5 Final Output

The data is which we have collected should get displayed in an order like the most delayed flight will have the highest count so it will be the highest rated flight with the maximum delay count in the negative order.

C. PUBLICATION WITH PLAGARISM REPORT

