eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

Computer Engineering Degree

Computer Science

Flight Delay Forecast due to Weather Using Data Mining

Author

Adrian Alexander Arteche Simmons

Director

Iñaki Inza Cano

Firstly I would like to thank my director, Iñaki Inza for supporting and believing in me throughout this project even at tough times, when things seemed as if they were not going to be finished on time.

I would also like to thank Pablo Rozas Larraondo, who despite living and working in Australia, has helped to shed light on METARs and web scrapping matters and without whose help everything would have been much more obscure and complicated.

Thank you, too, to my family, especially to my mother who kept on pushing me towards the completion of my project, by insisting and insisting that I was running out of time. She was also there to help me with any doubts about my English and to prepare my project's oral presentation.

I would like to mention one particular teacher, who gave me advice whenever I needed it and who has helped me to accomplish tasks not only about the project but also about myself. José Miguel Blanco.

Finally, I would like to thank you for reading this memory project, and perhaps together we could develop an application which can forecast flight delays and save a great deal of time and money to everyone.

Abstract

Flight delays are present every day in every part of the world. There can be flight delays due to weather, to excessive traffic, to runway construction work and to other factors, but most delays are due to weather and people assume it's part of flying. But what if we could accurately predict, at least with ~70% accuracy, if a flight was going to be delayed due to weather within 10 days of the flight date? If possible, it would save passengers a great deal of time and money because passengers would be able to book flight connections with enough time to spare. By searching thoroughly through the Internet I was able to collect a list of variables that will help me in creating a flight delay due to weather forecaster. First WEKA and then R will be used to build the model by trying different classifiers and selecting the one with the best results. WEKA is going to be used for its ease of use in accessing different classifiers and testing different settings and R will be used as a more complex and updated approach. The flight variables used for the model have to be available at the time of ticket purchase and at the same time the weather variables will have to be obtained from a weather website. An early thought is that Naïve Bayes will provide the best results. As it turned out, however, the best accuracy in R was achieved with the linear discriminant analysis classifier, achieving over 74% accuracy and Naïve Bayes which was able to predict with 69% accuracy. At the same time in WEKA the best accuracy was achieved with a cost sensitive classifier and the CfsSubsetEval.

Table of Contents

Table Listing

Figure Listing

Chapter 1

Introduction

The main aim of this project is to create a data-mining model, which is able to forecast flight delays due to weather observations. The whole idea lies on the fact that when a user enters a flight destination, date, time, airline, origin and some weather observations, the system will respond without a time lapse with an answer that represents whether the flight entered may or not be delayed. The model is trained and tested against 2013 flight and weather records, which means that all the used data are facts and nothing was invented. In the near future this model will be used as part of an online app. with its own interface, being used in real time to assist travelers from all over the world and save them time and money when booking their connecting flights.

At the same time, there is also a personal aim of this project, which is to learn a whole procedure to a real life data-mining problem and to get to know how to understand the problem, collect the data, clean the data, build a model and extract the conclusions of this type of problem by using the latest software available for this purpose.

Motivation

Have you ever been stranded at an airport because you had a close connection which you missed due to bad weather at your point of origin?

For frequent fliers such as myself, it’s a “pain in the neck” to miss connections when you go on holiday, and even more for business people who travel every single day, or at least every single week, and can’t miss work meetings or closing deals with partners. That’s why, if beforehand you know that your flight will most likely leave late due to weather, you will be able to book a connection with enough time for you to catch it, and not be at an unwanted airport for an unwanted amount of time. The whole idea relies on the fact that weather pages can give you a 10-day forecast, so you must book the ticket within that space of time or just leave it to luck and weather records.

At the end of the project, you may wonder why he chose Chicago's airport? Chicago Airport, commonly known as O'Hare International Airport is the busiest airport in the world when talking about aircraft movements. It has 881933 aircraft movements a year. I wanted to go big and face a real life problem with its real life data problems such as where to get the data from, to gauge how long it would take to process the data, to build models and to make conclusions. In other words to do everything I needed to do. This project with this airport is the perfect start to learn the data-mining field and to familiarize myself with it, to see if this field fulfills me and can become a strong point of interest in my future job. This is the main motivation for doing this project.

Chapter 2

Basic concepts

This chapter summarizes the variables that appear in the weather and flight datasets. These variables' definitions must be known to understand what the project's report is talking about at each time. Most of these variables were part of the initial flight and weather dataset. However, there are some that had to be created. The variables that have been created and the variables used for the merge of both datasets state so in their definition and the reason for their creation.

2.1 Flight data

Every flight has a series of fields, which give information about that specific flight. Next, those variables[*] will be described for a better understanding when reading this report.

FL_DATE_MDD_MMDD[**]

This variable has been created specifically for this project, to be able to have an order as an int, for it to be processed by R and WEKA. Without this variable, there isn't a possibility to set an order in time for each flight. With this new variable it does have an order, (month, day, day) or (month, month, day, day). So the numbers go from 101 (Jan 01) to 1231 (Dec 31). This was necessary because R doesn´t work correctly with dates as far as I've read, and in WEKA, there is a date data type, but Naïve Bayes´ classifier doesn't tolerate it.

FL_DATE[**]

It's the flight date with the following format (*ddmmyyyy*). This variable is used for the merge.

[*] (Bureau of Transportation Statistics 2013)
[**] These variables have been used for the model building in R and WEKA

UNIQUE CARRIER[**]

Unique carrier code. These are two character codes assigned by the IATA[2]. When multiple carriers have used the same code, a numeric suffix is used for earlier users.

ORIGIN[*]

Origin Airport Code. These are three character codes assigned by the IATA[3]. This project only uses one origin airport, ORD, which is Chicago O'Hare International Airport.

ORIGIN CITY NAME[*]

Origin airport, city name. In this project it doesn't change, as the unique origin city is Chicago, ILL

DEST[**]

Destination Airport. These are three character codes assigned by the IATA[3]. Only US airports are used.

DEST CITY NAME[**]

Destination Airport, City Name. As mentioned before, only US cities are used.

ROUNDED_TIME[**]

This field has been created specifically for this project to be able to merge the weather dataset with the flight dataset on a common variable. Every weather observation is made every hour starting at 00:51. So the flights within an hour have been taken and their flight time has been rounded to this field. For example, if a flight left in the following hour range: 6:51am – 7:50am their rounded time would be 6:51am. It's more likely for those flights to be suffering from the weather observations that have been spotted, than from the weather observations that haven't.

[*] 2 International Air Transportation Association (Aircarrier: United States Census Bureau n.d.)
3 Airport names by the IATA (IATA Codes: One World Nations Online n.d.)
* These variables have been used for the model building in WEKA
** These variables have been used for the model building in R and WEKA

CRS_DEP_TIME[**]

CRS Departure Time. The plane must have left the gate by that time. Format (*hhmm*).

DEP_TIME[*]

Actual Departure Time. When the plane actually leaves the gate. Format (*hhmm*).

DEP_DELAY[*]

Difference in minutes between scheduled and actual departure time. Early departures show negative numbers.

DEP_DELAY_GROUP[*]

Departure delay intervals, every 15min for (< -15 to >180).

DEP_DEL15[**]

Departure delay indicator, 15 minutes or more (1=Yes)

DEP_TIME_BLK[*]

CRS Departure time block. Hourly intervals.

CRS_ARR_TIME[**]

CRS Arrival time. The plane must have got to the gate by that time. Format (*hhmm*).

ARR_TIME[*]

Actual Arrival Time. When the plane actually gets to the gate. Format (*hhmm*).

ARR_TIME_BLK[*]

CRS Arrival time block. Hourly Intervals.

[*] * These variables have been used for the model building in WEKA
** These variables have been used for the model building in R and WEKA

ARR_DEL15*

Arrival delay indicator, 15 minutes or more (1=Yes).

2.2 Weather data

Every piece of weather data has a number of fields that gives further information about that weather observation. A short description will be given next, for a better understanding. These descriptions haven't been invented. They have been obtained from different web pages, which will be referenced at the end of the reference section and at every footnote there's also a reference to those references.

Conditions**

This is the type of weather observed at the reporting time. These conditions[3] may include *types of precipitation, the condition of the air…*

DewPointC**

Dew point is the temperature to which air must be cooled for saturation to occur, providing there is no change in water content. The dew point is used to predict the formation of dew, frost and fog. If dew point and temperature are close together in the late afternoon when the air begins to turn colder, fog is likely during the night. High dew point indicates high vapor content; low dew point indicates low vapor content. Therefore a high dew point indicates a better chance of rain and severe thunderstorms. Measured in C.

Events*

Event is just the weather phenomena, without a quantifier in front of the term. It has nine different values: *Fog, Fog-Rain, Fog-Rain-Thunderstorm, Fog-Snow, Rain, Rain-Snow, Rain-Thunderstorm, Thunderstorm and Snow*.

GustSpeedKm.h*

It is the maximum wind speed recorded over a specific time period. When wind speeds are measured and the peak wind speed during the measuring period is

3 (Documentation: Wunderground 2015)
* These variables have been used for the model building in WEKA
** These variables have been used for the model building in R and WEKA

roughly 10 knots more than the average wind speed, a wind gust is reported. Measured in km/h.

Humidity[**]

Humidity is the amount of water vapor in the air. Humidity indicates the likelihood of precipitation, dew or fog.

Precipitationmm[*]

This is the amount of liquid equivalent precipitation measured over a particular range of time. Snow is melted down and measured. Measured in mm.

Sea Level PressurehPa[**]

The sea level pressure is a correction of the station pressure to sea level. This correction takes into account the standard variation of pressure with height and the influence of temperature variations with height on the pressure. The temperature used in the sea level correction is a twelve-hour mean, eliminating diurnal effects. Once calculated, horizontal variations of sea level pressure may be compared for location of high and low pressure areas and fronts. Measured in hPa.

TemperatureC[**]

The temperature is a measure of the internal energy that a substance contains. This is the most measured quantity in the atmosphere. Measured in C.

VisibilityKm[**]

The visibility is the maximum distance an object may be seen considering air conditions. Precipitation, fog, haze, pollutants and suspended dust all contribute to lowering visibility. Measured in Km.

Wind Direction[**]

This is the direction from which the air is moving. This direction is in terms of category. There are these categories (*Calm, East, ENE, ESE, NE, NNE, NNW, North, NW, SE, South, SSE, SSW, SW, Variable, West, WNW, WSW*).

* These variables have been used for the model building in WEKA
** These variables have been used for the model building in R and WEKA

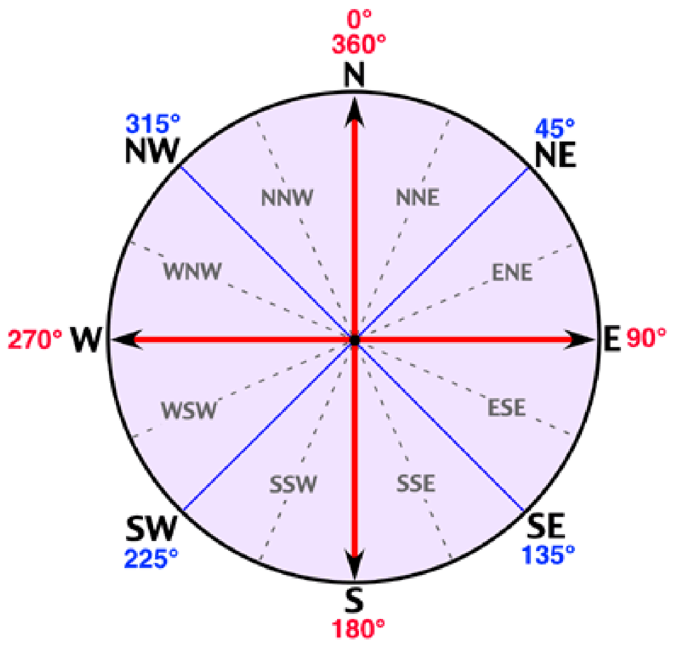


Fig. 2.1 Wind compass describing principal directions
Source: http://www.physicalgeography.net/fundamentals/7n.html

WindDirDegrees**

This is the direction from which the air is moving. The directions are in terms of degrees from true north (0 degrees), and the angle increases in a clockwise direction. In other words, if the direction is 45 degrees, the wind is from the northeast. Measured in degrees.

Wind SpeedKm/h**

The wind speed is a measure of the average speed of movement of the wind at a specific point. When measured, the value represents an average taken over a couple of minutes. Measured in km/h.

TimeCST

This is the Central Standard Time. It is used only for the merge of both datasets. It has the following format (*hhmm*).

DateUTC

This is the Universal Coordinated Time. It is used only for the merge of both datasets. It has the following format (*yyyy-mm-dd*).

** These variables have been used for the model building in R and WEKA

Part 1: Project Development

In the next few chapters the whole data mining process will be described, from the data collection part, through the data preparation and finally the data mining. At the end of this part the software and hardware used will also be described. Data cleaning and formatting can account for the 70%-80% of the whole project time according to several Data scientists.

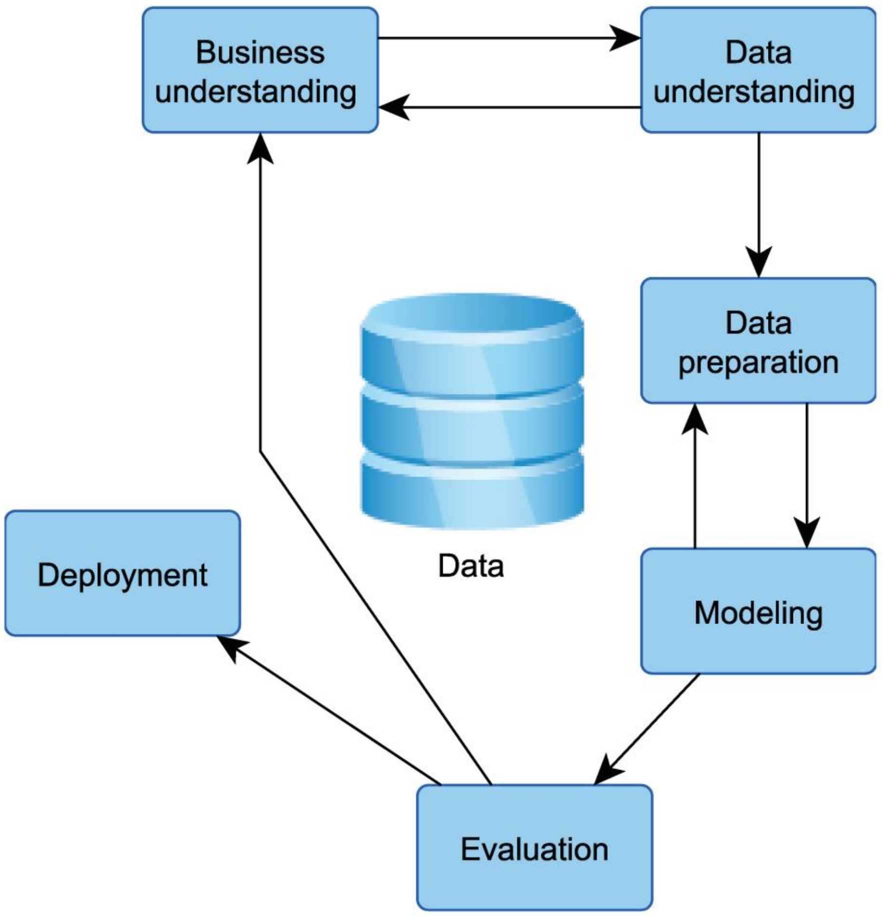


Fig. P1.1 Phases to a Data Mining project

Source: http://www.ontodm.com/doku.php?id=ontodm-kdd

"We all worry about algorithms, they are fascinating, but most of us know that data mining in practice is mostly data prep work" – Usama Fayyad (Chief Data Officer & Executive VP Yahoo! Inc.)

Chapter 3

Data collection

Once the project is understood, the next step is to collect all the required data for the future model. Data collection at first seemed to be an easy part of the project, just finding the data and downloading it. However nothing is as easy as it may seem. The focus of this project is not one of an academic project. Therefore, it doesn´t use academic datasets and academic amounts of data. This means that academic projects, used in everyday classrooms, tend to work with small amounts of data to be able to extract conclusions within an hour or two and all the data can be downloaded from a given platform. Instead, this project faces a much bigger and real problem where the data isn´t located at a given source with the needed format and that deep searching is needed to be able to find this data.

The data collection is divided into two main sections, weather data and flight data.

3.1 Flight Data

For each flight, at least the origin, destination, date, time, airline and whether the flight was delayed or not needs to be collected. At first, pages such as "Flightstats" or "Flightaware", which are pages for flight lovers, were researched; at Flightaware you have to pay for the data in order to be able to download it. So it isn't a real option. Finally, hours and hours of research paid off. The Federal Aviation Administration's website hosts a link to the United States Department of Transportation. At that website, there is a section where the fields required for this project can be selected and then the monthly data sets can be downloaded. However, to come across this website, at least 50 other website links were opened and 30 Google terms were searched for.

For this project, initially the whole year datasets from 2009 to 2013 were downloaded, but later only the 2013 dataset has been used, due to the already large amount of data: over 800 thousand cases with more than 30 variables, with every

Excel file taking up 300Mb. To put this into perspective, 800 thousand cases equals 20952 MS Word pages full of data.

To make things clear, it would have been better to deal with five years' worth of data, around 4 million cases. The model would be more accurate, however a more powerful computer would have been needed to be able to deal with such a large amount of data.

3.2 Weather Data

Regarding each weather observation, as many variables as possible are needed, some of which in a future prototype will be able to be collected by a user interface or through a live feed. These variables have to be easily accessed. At the start of the project, the METAR concept was totally unknown, so research had to be done regarding weather data, its formats and its different uses had to be known to be able to choose the best option, or at least the best feasible option.

3.2.1 METAR

According to Wikipedia, METAR is a format for reporting weather information. A METAR weather report is predominantly used by pilots in fulfillment of a part of a pre-flight weather briefing, and by meteorologists, who use aggregated METAR information to assist in weather forecasting.

METARs was the first choice to be the selected weather data for the project, because also according to Wikipedia it is the most common format in the world for the transmission of observational weather data. Looking for the right METARs wasn't an easy task either, as before many different websites were searched for METARs, over 10 different weather websites were looked at before coming across the final one. Once the source was found, a tool was required to be able to extract the information from the webpage. Research was conducted, and a tool for this purpose was found, which is called "web scraper". This tool allows the user of it to run the program and download whatever piece of information from the target webpage and save it into a text file. A web scraper can be programmed in any language but this one was programmed in Python to extract METAR knowledge from the website. This web scraper will be described later on. This website had one major problem, it would only let you download 1 month of weather data at a time, so this process took up to 12 days. Once the data was downloaded, a unique format was attempted to be given to the data, in this case CSV. However, due to the fact that each METAR is too variable and doesn't have a common format between each other due to the remarks (1), which add some information to the current METAR, this format couldn't be given to it because the key to the question is, where to put the commas.

An initial approach was to parse each METAR with regular expressions. For this purpose, how METARs are structured and which fields they have was contrasted between different sites and learned. However, it was far too complicated, because as has been stated before, each METAR is too variable and the regular expression has to take that into account. Perhaps some fields can be parsed correctly, but when you get to the remarks' part it all gets mixed up. This is the reason why this weather format was given up, after so much effort had been put into it.

(1): So each remark can be a whole new METAR. For example:
<ins>METAR with remark:</ins>
KORD 312351Z 16014G20KT 6SM BR OVC008 03/02 A2984 **RMK AO2 SLP112 T00330017**
<ins>METAR without remark:</ins>
KORD 312351Z 16014G20KT 6SM BR OVC008 03/02 A2984

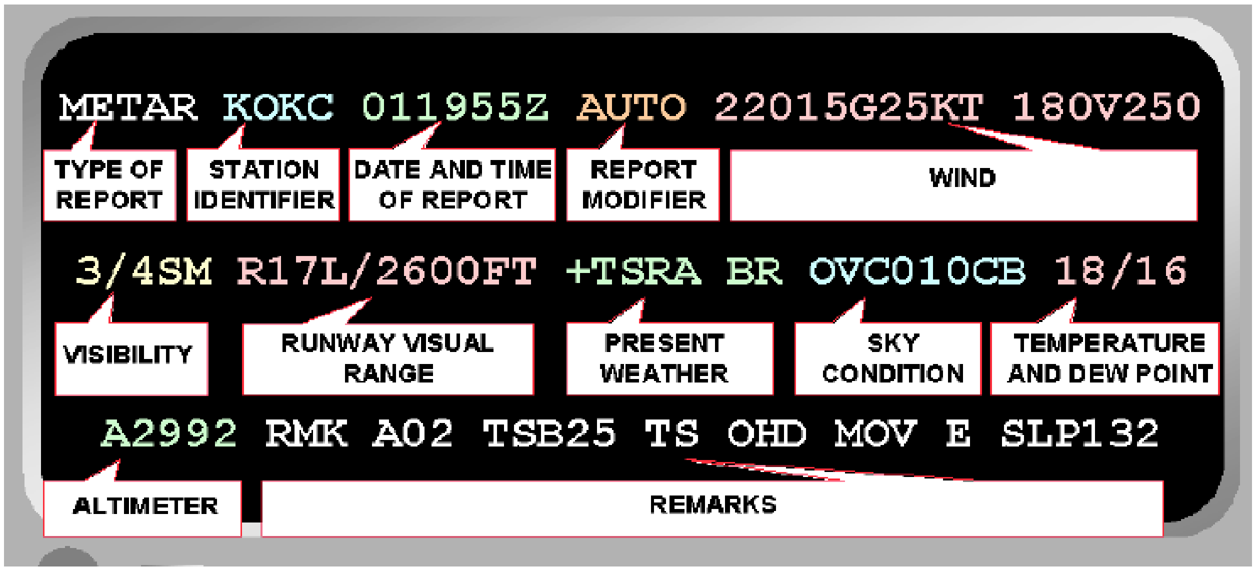

METAR KOKC 011955Z AUTO 22015G25KT 180V250
TYPE OF REPORT
STATION IDENTIFIER
DATE AND TIME OF REPORT
REPORT MODIFIER
WIND
3/4SM R17L/2600FT +TSRA BR OVC010CB 18/16
VISIBILITY
RUNWAY VISUAL RANGE
PRESENT WEATHER
SKY CONDITION
TEMPERATURE AND DEW POINT
A2992 RMK AO2 TSB25 TS OHD MOV E SLP132
ALTIMETER
REMARKS

Fig. 3.1 METAR example explanation

Source: http://www.avionicswest.com/Articles/MetarTaf.html

3.2.2 NON-METAR weather observations

After encountering the METAR problems, the steps to the METAR finding had to be backtracked and an alternative source of weather data had to be found, or weather data itself had to be given up. Leaving aside weather data was contemplated, however it would also mean giving up reliability, so another source of weather data had to be found. After many hours and many different searching terms, that webpage was found.

Wunderground was the solution to the problem; it offers an hourly weather history and observations with the weather variables mentioned in the previous chapter. However, this website, in contrast to the flight data source, doesn’t have the monthly data to download, so at first it seemed impossible to be able to get all the weather data. The model is built with 2013 flight data and that has to be taken into account, because the weather data needed had to be the same date range as the flight one. The website that was found would let you see the weather for that specific date, however if you looked at the URL it could be seen that the day, month and year were part of the PHP GET parameters and, by changing those variables, any date could be accessed. However entering 365 days was too tedious, and then converting it into the CSV format, which was needed, was even more tedious.

As a result the web scraper used for the METAR downloading was tweaked and this allowed the data from this webpage to be downloaded into a text file for future processing. In addition to the information given above about the web scraper, for it to work correctly a web site link pattern had to be found. For this website the pattern was found, as you are able to see next.

For example: […]/airport/KORD/"+"2013"+"/"+"**%m**"+"/"+"**%d**"+"/[…]

```
weather_scraper()
while startDate <= endDate loop
    request the data to the website
    write the data onto a text file
end loop;
```

Chapter 4

Data preparation

Once all the data was collected, the final dataset for future processing was built. This whole process is done in different steps and, as mentioned in the previous chapter, this project has two different datasets, so this process has to be done twice and then both datasets can be merged to obtain the final dataset. The only reason why, both datasets aren’t merged at the beginning is because it is easier to work with lighter files referring to size than to work with a bigger data file, because the computer tends to freeze up.

4.1 Data cleaning and transforming

Data cleaning is a very important step in data mining; here is where each dataset is cleaned. This means the model is prepared to fit the needed format, by eliminating unwanted values such as outliers, extreme values or values, which aren’t wanted to be processed, transformed or adapted to the desired format. If this step isn’t done correctly, the final results may turn out to be incorrect or may vary a lot from the correct ones. That’s why as mentioned in the first line of this paragraph, it is a very important step, and it doesn’t take the same amount of time and effort to do it well than to do it wrongly. A thorough cleaning must be performed.

4.1.1 Flight Data

The flight dataset initially started out being twelve separate CSV file, which were merged into one so it could be opened in one Excel window. The sum-up of the twelve files had over 800 thousand entries with twenty-two variables, some of which are explained in the first chapter but some others aren’t because they are totally irrelevant for this project. In conclusion, the resulting dataset is a very big one, which just by having it open in MS Excel consumes a lot of system resources.

Having over 800 thousand entries meant that programming scripts was the only way to cope with all the different tasks that had to be done. Because even if it only took 10 seconds to clean each entry, it would take over 65 days to process all the data, and this wasn't an option. The second option was to use Excel's filters, but again human process is too slow. Therefore as said, the scripts had to be programmed, so although time had to be invested into that, the quickness of the scripts would make up for the time invested here. In addition, the computer used throughout the project was just enough for this amount of data, so that meant things had to be done little by little or else the system collapsed and Excel would not respond.

Firstly, just the flights whose origin was Chicago O'Hare International Airport (ORD) had to be kept, so a specific script in Visual Basic was built for such a purpose. To process over 800 thousand entries consumes too much RAM memory, and that's why it couldn't be done in just a loop from 1 to 800000. The script had to be executed different times with different loop ranges. Before executing the scripts, the variables that weren't definitely useful for sure were eliminated, and this made the dataset 'lighter' size-wise.

```
removeOtherAirports()
do while not isEmpty(Cell)
    Check if the cell value isn't Chicago if
    not increase counter
End loop
End
```

Secondly, by having the weather data also in the window, two variables had to be adjusted or created to be able to do an inner join and merge both datasets into a single one. After looking at both datasets, those variables were found. The first variable consisted of transforming the date variable of the format (*ddmmyyyy*) to the format (*yyyy-mm-dd*). At first, by common sense one would think that Excel provides the necessary tools to do such a thing by going into date format, but it actually doesn't. Therefore a script was created for this purpose. As before, there were still too many entries, over 400 thousand, so it had to be executed many different times.

```
changeDate()
For i=1..End loop
    Change current date format to required format.
End loop
End
```

The second variable used for the merge had to be created. Therefore, a new variable was created under the name "Rounded Time". In chapter 2 this variable is explained for the weather dataset, so now the rounded time variable for this dataset has a similar meaning. All the flights between 4:51am and 5:50am have a rounded time of 4:51 and so on. One might think that this way accuracy may be lost, but the weather data is collected in the same way, so one might also wonder why not set 5:50am flights to rounded time 5:51. The answer is because the 5:51am weather data, at 5:50, hasn't been collected yet, and the threshold is every hour, so that's why the new rounded time is at every hour.

```
roundedTime()
For i=1..End loop
    Check to see if the time was in certain range and
    change it to the new format.
End loop
End
```

Thirdly, the last cleaning task performed regarding the flight dataset was done once the class variable was defined for the model prediction, which in this case is DEP_DEL15. The rows where the class variable was empty had to be removed.

```
NotNADEP_DEL15()
For i=2..End loop
    If value equals NULL
    Delete row
End loop
```

```
End
```

Finally, to be able to have a date order when the data mining model was being built, a new variable had to be created, which was given the name of FL_DATE_MDD_MMDD, and this time human processing had to be done, because building a script seemed more time-consuming than just processing it human-wise. So by using the function ‘Search and Replace” and executing it 365 times, this field was built. For example 1012013, that is in the format (ddmmyyyy or dmmyyyy), which was found and replaced by 101, which is in the format (mmdd or mdd).

4.1.2 Weather Data

The weather dataset started out being a long txt file; therefore it had to be converted into a CSV file format. Once that had been done, by just changing the extension of the file from txt to CSV, it was realized that when the CSV file was opened in excel, blank entries were found in-between each weather entry. A script for removing those blank rows was created.

```
deleteBlankRows()
For i=1..End loop
    If row is empty then delete it.
End loop
End
```

```
keepOnlyHoursNeeded()
For i=1..End loop
    If hour not needed then delete entire row
End loop
End
```

4.2 Data Merging

This part of the project was a real quest. Both datasets, weather and flights, had to be merged to obtain just one. At first, one would think this is an easy task; just do an SQL inner join on date and time, and that's the end to this story. However, it wasn't so easy.

4.2.1 First Option

The first thought was to use an SQL query, such as an inner join on date and time, as mentioned above. However, one of the main drawbacks was that a database had to be built or that the CSV file had to be transformed into an SQL file. Both drawbacks are complicated in terms of time.

4.2.2 Second Option

The second thought was to use Google Fusion Tables, which is a piece of software created by Google for data management. It is a really good piece of software and must be recommended. This software has an implicit merge-option, which was used to do the merging of both datasets. However, for some reason it didn't work correctly, so it had to be discarded.

4.2.3 Final Option

Finally, a point was reached where there was no clue as to what to do next to be able to merge both datasets, which had to be done. At that point, advice from J.M. Blanco came to mind: "You don't need complicated software to do complicated things, you can use everyday software to do those things". An idea arose: If Excel is one of the best data processing software ever made, it has to have the ability to do what was needed. Researching on the Internet with the search term "Excel and SQL" provided some web pages where the solution to the problem was described.

Excel has an option where you can work with two or more spreadsheets as if they were databases and you can do SQL queries, among other things. Therefore, this option was used, and it worked perfectly. Both datasets could be merged on the ROUNDED_TIME and the FL_DATE from the flight dataset and the TimeCST and DateUTC from the weather dataset into one dataset, which was the one used to build the model later on.

Further information regarding this topic: (Excel: WikiHow n.d.)

Chapter 5

Data Mining and Evaluation of the results

Data mining is one of the last parts to the project. This part and the next one, which is the interpretation and evaluation, contribute to 30% of the total data mining project time. To be able to construct the models you had to go through the other parts first. This part is considered by most people as the best and most entertaining part of a data-mining project, together with the interpretation and evaluation part.

5.1 Building the model with WEKA

As a more academic approach, my model was built using WEKA, just because this software is the one used in the Data Mining subject at university and I could get by with it. However, it wasn’t as easy as it first seemed. When the dataset was imported into WEKA, a series of problems arose, which will be described later on. To solve these problems some transformations had to be done to the data before being able to do anything else.

5.1.1 Model fixing

Every data mining software doesn’t work the same way and doesn’t have the same data requirements. So before doing anything, some settings had to be adjusted before being be able to deal with the model without any more problems.

5.1.1.1 Changing variable types for WEKA

Some variable types had to be changed, because WEKA automatically decides the variable type when you import a dataset. At first the following changes were made.

Table 5.1 Change of variable type

Variable	From	To
FL_DATE	Nominal	Date
DEP_DELAY_GROUP	Numeric	Nominal
ARR_DEL15	Numeric	Nominal
GustSpeedKm/h	Nominal	Numeric
Preicipitationmm	Nominal	Numeric
Missing values	-9999	?

Nominal: Variable with values, which have no numerical value, such as gender.
Numerical: Variable with values, which have a numerical value.

5.1.1.2 Removing variables with filters

Useless variables were removed using a WEKA filter. This removed the Origin and Origin city, and that's because it never changes, so the origin city is Chicago and the origin is ORD and both of these can be assumed. In addition to those variables, some variables that couldn't be known beforehand were removed such as the: DEP_TIME, DEP_DELAY, DEP_DELAY_GROUP, DEP_TIME_BLK, ARR_TIME, ARR_TIME_BLK, ARR_DEL_15, ROUNDED_TIME.

5.1.2 Model Building

So after the WEKA preprocessing I was at the point where the model could be built and conclusions could be extracted. To make it clear now, the confusion matrixes in WEKA follow the next layout.

Table 5.2 Confusion matrix layout WEKA

		Prediction	
		0=No	**1=Yes**
Actual	**0=No**		
	1=Yes		

5.1.2.1 First approach

As a first approach, a Meta classifier called 'Cost Sensitive Classifier' is used. The base classifier in this first approach is another meta classifier, called 'Attribute Selected classifier', which then uses the Naïve Bayes classifier and the CfsSubsetEval as an evaluator. The Cost Sensitive Classifier makes its base classifier cost sensitive, which means it uses a cost matrix to penalize some types of errors more than others. At the same time, the CfsSubsetEval evaluates the value of a subset of attributes by considering the predictive ability of each individual variable along with the degree of redundancy between them. Therefore, subsets of variables that are highly correlated with the class while having low intercorrelation are preferred.

As a frequent flier I've decided to use the following cost matrix and after I will give my reasoning for this.

Class 0: Not Delayed
Class 1: Delayed

Cost Matrix	
0	1
6.5	0

So by looking at the cost matrix it can be seen that field (2,1) has a 6.5 value instead of the normal value of 1. This is because I'm telling WEKA that I want to penalize class 1 as class 0. This is because, from a flier's point of view, it is better to plan things as being delayed and then if they aren't, you can look at duty free shops or have a coffee, rather than plan things as not being delayed and then if they are, be stuck in the middle of nowhere. So that's why this cost matrix has been chosen.

We will just look at the confusion matrix for the results

Confusion Matrix	
30759	36299
6636	17987

Average Cost: (6636*6.5 + 36299*1)/91681 = 0.86

5.1.2.2 Second approach

As a second approach the same meta classifier, Cost Sensitive Classifier, has been used but this time the base classifier isn't another meta classifier, it is just the Naïve Bayes classifier, which takes all attributes from every instance as normal and the label with the lowest cost is selected by getting all label probabilities and determining the expected cost of each possible labeling. However, to be able to use this classifier the date format had to be changed, as was explained in the first chapter. The Naïve Bayes classifier is based on the Bayes theorem.

$$c^* = arg\,\max_{c} p(C = c) \prod_{i=1}^{n} p(X_i = x_i | C = c)$$

Fig. 5.1 Naïve Bayes Classifier
Source: Clasificadores Bayesianos 2013/14 Minería de Datos EHU/UPV

So for this classifier I've used the same cost matrix.

Class 0: Not Delayed
Class 1: Delayed

Cost Matrix	
0	1
6.5	0

But now the results are slightly different.

Confusion Matrix	
31891	12832
8006	8392

Average Cost: (8006*6.5 + 12832*1)/61121 = 1.06

5.1.2.3 Third Approach

For the third and last approach, the 'Attribute Selected Classifier', which reduces the dimensionality of training and testing data by doing a selection of attributes, has been used. CfsSubsetEval has been used as the evaluator and Naïve Bayes as the classifier but with no cost matrix this time. The results for this approach are the following:

Confusion Matrix	
44103	620
15649	749

Average Cost: (15649*1 + 620*1)/61121 = 0.26

5.1.3 Comparing Results

1st Approach

Confusion Matrix	
30759	36299
6636	17987

Cost: 0.86

2nd Approach

Confusion Matrix	
31891	12832
8006	8392

Cost: 1.06

3rd Approach

Confusion Matrix	
44103	620
15649	749

Cost: 0.26

So the best option is to use the first approach, which is a cost-sensitive classifier with the CfsSubsetEval, as it makes fewer unwanted errors when forecasting a delayed flight. It is true that it also makes more errors when forecasting a non-delayed flight, but this is better because, as was explained previously, as a flier it is better to get a later connection than to miss a flight.

Table 5.3 Best option confusion matrix WEKA

	Confusion Matrix (Cases = 91681)	
	Not Delayed	**Delayed**
Not Delayed	30759	36299
Delayed	6636	17987

5.1.4 Problems Found

Many problems didn’t have to be dealt with, however there were at least three that I can account for. The first one, as said in a previous chapter, Naïve Bayes doesn’t support the date data type, which is why the date variables had to be transformed into another data type.

Secondly, due to the considerable amount of data, problems with the Java heap appeared. Therefore, the heap had to be increased to avoid the “Out of memory” error.

Lastly, due to the large amount of data, some types of classifiers were unable to be used, because it would just keep running, and perhaps 24h would go by, and it would still be running. It makes sense, for example, that the J48 classifier or any classifiers with trees are some of those classifiers that couldn’t be used.

5.2 Building the model with R

As a more professional approach, R was chosen to be the program, not only because it is one of the most used data mining software but because, apart from that, I wanted to learn and use a software that would allow me to look for a job in this field in the near future, and knowing R would give me a bit more credit. It took me time to learn something about R, which is completely different from WEKA in the way that WEKA has a user-friendly interface and R is mostly commands. Therefore, I decided to read a book called “Data Mining with R and Rattle” and this gave me some alternatives to start with. After reading the book I was determined to use Rattle, because like WEKA it has a user-friendly interface. However, when I imported the dataset into Rattle, I would get a lot of problems due to its size and that Rattle is good but only has a series of classifiers, mostly trees, and, as I said before with WEKA, a tree classifier wasn’t an option for me. So discarding Rattle, I was facing another problem, which was dealing with R and its command line. Researching on the Internet I came up with a package named “Caret”, which is one of the latest data mining trends, so I decided to stick with this package and learn about it.

5.2.1 Model fixing

Once all the problems had been dealt with, which will later be explained, to be able to use Caret my dataset had to be changed slightly to be able to construct the model.

When importing the dataset into R, R would select the variable types automatically, and sometimes R would guess the right type but others not. The class variable had to be transformed from int to factor. Factor variables are variables with different types, for example in this case 1 or 0. The class variable must always be a factor. The following command was ran:

FlightData$DEP_DEL15 <- factor(FlightData$DEP_DEL15)

In addition to the class variable, some numeric variables were transformed into factor variables. At this point one would wonder, why not transform it into numeric with a cast and that's it? This is the general thought, but when transforming these variables into numerical ones was tried, the variable values weren't the ones that there should have been. So researching on the Internet, specifically Stack overflow, a solution to the problem can be found and this solution had the following command to do so.

Generic: *as.numeric(levels(f))[f]* //Replace f with the factor to change
Particular:
FlightData$Precipitationmm<-
as.numeric(levels(FlightData$Precipitationmm))[FlightData$Precipitationmm]

Finally, the last fix was to change the class variable levels, from (1,0) to (Yes, No), this way avoiding any conflicts with the R variables (X0, X1), which seems to be a very common mistake when data modeling. The command used is the following.

levels(FlightData$DEP_DEL15)[match("1",levels(FlightData$DEP_DEL15))]<- "Yes" and the same command but changing Yes for No.

5.2.2 Model building

Before doing anything, the training and testing data partitions have to be created and a seed for random numbers has to be set. To do that, the following commands have to be run. This command allows the future creation of the testing partition and of the training partition.

set.seed(13)

TestTrain ← createDataPartition(y=FlightDataProcess$DEP_DEL15, p=.75, list=FALSE)

Variable in call	Explanation
y	Vector of outcomes. This case it will be the class variable.
p	Percentage used towards training. In this case 75% which are 212300 cases.
list	Whether the result should be in a list or not. This case = FALSE.

Next, the training and testing partitions will be created.

trainingW ← FlightDataProcess[TestTrain,]
testingW ← FlightDataProcess[-TestTrain,]

'training' takes all the variables with 212300 cases and at the same time 'testing' takes all the variables but with 283065 – 212300 = 70765 cases.

Before creating the classifier, a trainControl has to be created. The trainControl controls the different nuances of the train function.

controlW ← trainControl(method = "repeatedcv", repeats=3, classProbs=TRUE)

Variable in call	Explanation
method	The resampling method. This case refers to a repeated cross validation.

repeats	It is the number of complete sets of folds to compute. In this case it will be 3.
classProbs	This is telling R to compute the class probabilities for the classification models along with the predicted values in each resample.

5.2.2.1 Linear Discriminant Analysis Model

The first classifier to be used will be the '*Linear Discriminant Analysis*', which finds a linear combination of features that characterizes or separates two or more classes of objects or events. This classifier is used only for classification models and has no tuning parameters. First the model will be trained using the '*train*' function, then it will be tested against the testing data using the '*predict*' function and lastly the confusion matrix will be calculated by using the '*confusionMatrix*' function.

ldaModel3x10cvW ← train (DEP_DEL15 ~., data=trainingW, method="lda", trControl = controlW, preProc=c("center", "scale"))

Variable in call	Explanation
DEP_DEL15~,	It is the variable to predict. The ~, denotes the other variables which are all the predictors.
data	It is the data frame from which variables specified in the formula are preferentially to be taken.
method	A string specifying which classification or regression model to use. In this case the lda: linear discriminant analysis is being used.
trControl	It is a list of values that define how this function acts.
preProc	It is a string vector that defines a pre-processing of the predictor data.

ldaPredictW ← predict(ldaModel3x10cvW, newdata=testingW, type="raw")

Variable in call	Explanation
ldaModel3x10cvW	It accounts for the object.
newdata	Set of data to predict on.

type=raw	The class predictions are being used, that's why it's raw.

confusionMatrix(data=ldapredictW, testingW$DEP_DEL15)

Variable in call	Explanation
data	It is a factor of predicted classes
testingW$DEP_DEL15	It is a factor of classes to be used as the true results.

5.2.2.2 Naïve Bayes Model

The second classifier to be used will be the '*Naïve Bayes*' classifier. This classifier is also used only for classification models. First the model will be trained using the '*Naïve Bayes*' function. Due to problems while using the Caret package with this classifier, an alternative package was used for this classifier, the '*e1071*' package. After training the model, it will be tested against the testing data using the '*predict'* function and finally the confusion matrix will be obtained by the use of the '*confusionMatrix'* function.

nbModelW ← NaiveBayes(DEP_DEL15~., data=trainingW)

Variable in call	Explanation
DEP_DEL15~.,	It is the variable to predict. The ~, denotes the other variables which are all the predictors.
data	It is the dataframe of predictors.

predictionsW ← predict(nbModelW, testingW)

Variable in call	Explanation
nbModelW	It is the object
testingW	Set of data to predict on.

confusionMatrix(predictionsW$class, testingW$DEP_DEL15)

Variable in call	Explanation
predictionsW$class	It is a factor of predicted classes

testingW$DEP_DEL15	It is a factor of classes to be used as the true results.

5.2.3 Comparing results

First of all, the confusion matrix layout in R is slightly different; therefore the layout is shown next.

Table 5.4 Confusion matrix layout R

		Actual	
Prediction		**0=No**	**1=Yes**
	0=No		
	1=Yes		

With the Linear Discriminant Analysis the following results are obtained.

Confusion Matrix		
	0	**1**
0	48765	15875
1	2355	3770

Accuracy: 74%

As can be seen, although the accuracy seems to be good, the incorrect classification of the red square (1,2) is too big and that can be a point to highlight as a main drawback of this model. With this square, the interpretation is “when a flight is delayed, the model tells the client it’s going to leave on time”, from my point of view as an expert, this is an unwanted result.

At the same time Naïve Bayes has the following results.

Confusion Matrix		
	0	**1**
0	40213	10889
1	10907	8756

Accuracy: 69%

Using this classifier, the accuracy is a bit worse, however the confusion matrix itself appears to be better. There are fewer well-classified cases as not being delayed, but there are more well-classified cases as being delayed. At the same time, the red highlighted square (1,2) with the same interpretation as before has fewer cases, which is better, but in contrast to that it also happens that there are more cases where the blue highlighted square (2,1) is. This last square has the following interpretation: "when a flight is not delayed, the model says it will be", from my point of view as an expert, this is a better result than the red square, because it is telling the client to book with more time between flights, whereas the red square tells the client that its original flight isn't going to be delayed, so he does not need that much time between flights.

In conclusion, between both models, the best option is the one that uses Naïve Bayes even though it loses some accuracy. However, this loss of accuracy is better when comparing both whole confusion matrixes because there are less cases badly classified where the red cell is.

Table 5.5 Best option confusion matrix R

	Confusion Matrix (Cases = 70765)	
	Not Delayed	**Delayed**
Not Delayed	40213	10889
Delayed	10907	8756

5.2.4 Problems

The biggest problem is a strange and very serious problem[4]. I had W7 on my desktop computer with 8GB RAM and an i5 Intel Core, which was the perfect computer for doing the models. The case is that I installed the 64bit version of R and when I tried to install the Caret package I would get an error, telling me it wasn't possible to install Caret. So I tried to use the 32-bit version of R and install Caret, this time it would let me. The main drawback of using the 32-bit version is that you are just able to use 4GB of RAM, which wasn't enough for such a big dataset.

[4] More information about the problem (Stack Overflow n.d.)

The solution I had to apply was that, instead of using my powerful desktop computer, I had to use my laptop to be able to at least access those 8 GB of RAM, because if not I would obtain the error: “Can’t allocate memory”. For more information about this problem, you can visit the link below.

Chapter 6

Software and Hardware used

In this chapter the tools and software used throughout the project will be described. This includes not only the preprocessing part and the model building, but also the planning part of the project and the report writing. These tools are just some of the tools that can be used to do these tasks. Other people may use other tools for the same thing.

6.1 Software

***Sublime Text 2*[5]**

Sublime Text 2 is a cross-platform text and source code editor with a Python application-programming interface. It supports many programming languages and markup languages, and its functionality can be extended by users with plugins.

It was developed by Jon Skinner and released the 18th January 2008. It can be used on any operating system, from Linux to Mac OS X passing through Windows.

Why use Sublime Text 2 and not another thing? Sublime Text 2 is a full text and code editor and, apart from that, you are able to run the programs directly from the program. It also depends on the language you chose, but it provides you with a hint of the tag you are looking for.

Sublime Text 2 is used to program the Python script that was used to collect the weather data.

[5] (Sublime Text 2: Wikipedia n.d.)

Python[6]

To collect data a web scraper programmed in Python was used. According to Wikipedia Python's syntax allows programmers to express concepts in fewer lines of codes.

Guido van Rossum at CWI in the Netherlands started Python's implementation in December 1989. Python 2.0 was released on October 16th 2000 and Python 3.0 was released December 3rd 2008.

Python is intended to be a highly-readable language. It is designed to have an uncluttered visual layout, frequently using English keywords where other languages use punctuation. Furthermore, Python has a smaller number of syntactic exceptions and special cases than C or Pascal. Python uses whitespace indentation.

Why use Python for web scraping and not another thing? Python offers a module called 'urllib2', which has suitable functions to open websites and extract information easily.

Python is used to program the web scraper that is in charge of collecting the weather data for the model.

MS Excel

Microsoft Excel is a spreadsheet application developed by Microsoft for Windows and Mac OS X. It features calculation, graphing tools, pivot tables and a macro-programming language. The first version was released in 1987.

Why choose MS Excel versus another similar type of software? MS Excel is a very complete spreadsheet application tool, which supports almost any kind of file extension, and it has a lot of features. Its user-friendly interface helps you most of the time. However, if this doesn't seem enough, I will say that, apart from the typical things a normal user would do in Excel (Charts, Calculation…), it enables

[6] (Python: Wikipedia n.d.) and (Urllib2: Python n.d.)

you to use the VBA language to create functions to use on the spreadsheets you've created. Excel can also be used as if it were an SQL database as was explained in a previous chapter. Having said this, for me it is the perfect program.

MS Excel is used a lot throughout the project, to visualize the data and perform cleaning tasks on it. It is also used to do the merging and finally to do the time charts for the project management chapter.

VBA[7]

Visual Basic for Applications is a dialect of Visual Basic. Programming with VBA allows spreadsheet manipulation that is annoying or impossible with standard spreadsheet techniques. You are able to write code directly into Excel, as it has its own window for doing this. From my point of view, this tool is one of the best things that Excel has, as I have definitely saved time. Instead of processing all the data row by row, I could just run a program which would do all the work for me. The only things you have to have clear is that the program works correctly, so I advise you to try out the program you've built with a small data table and if it works, then use It on the whole data table. If you are processing large amounts of data and run the program without trying it first, it may take several hours before completion and it may be wrong, so you would have wasted your time.

This programming language is used to program the scripts that are used to clean the weather and flight data. It is used within the MS Excel program.

Google Fusion Tables[8]

Google Fusion Tables is a web service provided by Google for data management. It can be used for gathering, visualizing and sharing data tables. Data is stored in multiple tables that Internet users can view and download. The web service provides means for visualizing data with pie charts, bar charts…

[7] (VBA: Excel Easy n.d.) and (VBA Programming: Wikipedia n.d.)
[8] (Help: Fusion Tables Google n.d.) and (Google Fusion Tables: Wikipedia n.d.)

It was launched in June 2009 by Google but announced by Alon Halevy and Rebecca Shapley.

Why use Google Fusion Tables anyway? It is true that most of its features are provided by MS Excel or Free Office, however, there is one that isn’t: Google Fusion Drive has the option to create geographical maps. This is the most interesting option from my point of view. However, it also has a drawback, which is it can’t deal with large amounts of data.

Google Fusion Tables were going to be used to merge both datasets and to create some geographical maps for a better presentation of the data, however, it can´t deal with large amounts of data so it was discarded.

WEKA[9]

WEKA, which stands for Waikato Environment for Knowledge Analysis, is a popular suite of machine learning written in Java and developed at the University of Waikato in New Zealand.

It is a workbench that contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user-friendly interfaces for easy access to its functionality.

WEKA supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection. WEKA also provides access to SQL databases using the Java Database Connectivity and can process the result returned by a database query.

Why use WEKA? From my point of view WEKA is a great piece of software, although it is more oriented towards academic purposes. It has a good interface and I think it is the perfect tool to learn data-mining skills. However, later on I would use a more developed piece of software such as R. WEKA, which can let you do a

[9] (WEKA n.d.) and (WEKA: Wikipedia n.d.)

lot of things, and see the results clearly, although as just mentioned, it's a more academic piece of software.

WEKA is used to build some models using the merged flight and weather dataset. It was used as a more academic approach, but it offers good models to make the appropriate conclusions on.

R[10]

R is a programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Another feature is that R is an interpreted language; if a user types 2 + 2, the computer replies 4. R also supports procedural programming with functions.

Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand created R and it first appeared in 1993. R is named partly after the first names of the first two R authors.

Why use R instead of another piece of software? R has the possibility to download and install any kind of library, so this functionality gives R the ability to cope with any kind of problem. It can implement graphical techniques, classical statistical tests, time-series analysis, classification, clustering and others. The main drawback is that it is mainly a command line, however, once you get used to it, you won't care much about this.

R is used for a more advanced approach. It is used to create models using the merged flight and weather dataset. Good conclusions can be extracted from these models.

[10] (R: Wikipedia n.d.)

RStudio[11]

RStudio is a free and open source integrated development environment for R. RStudio is available in two editions: RStudio desktop, where the program runs locally, and RStudio Server, which allows accessing RStudio by using a web browser while it is running on a remote server. RStudio was created in December 2010.

Why use RStudio instead of using nothing? RStudio provides the user with a neat environment. Its windows, where the user is able to see his datasets, the command line, and the graphics… and can even download, install and use packages directly. If you don't like to use the command line for everything, RStudio provides an alternative to this with its interface.

RStudio is used in this project as a workbench for the R programming language.

Caret[12]

The caret package, which stands for Classification And Regression Training, is a set of functions that attempt to streamline the process of creating predictive models. The package contains tools for data splitting, pre-processing, feature selection, model tuning using resampling, variable importance estimation, as well as other functionalities.

Why use Caret? Caret, as mentioned in the previous paragraph, has a lot of functionalities and that's one of the main reasons, but not the only one. From a user's point of view, it is better to have everything in a same package, and data mining with caret seems simpler than with other packages. The commands make sense and all follow a certain order. Caret is able to use almost every classifier and there is plenty of help regarding this package, so if you get stuck or you don't know how to do something, don't worry just ask.

[11] (RStudio: Wikipedia n.d.)
[12] (The Caret Package n.d.)

Caret is used to build and test the R models. Different classifiers are used within the Caret package. Some of these classifiers are: Naïve Bayes and Linear Discriminant Analysis.

Rattle[13]

Rattle, stands for R Analytic Tool To Learn Easily, provides its users with an interface. The aim is to provide a simple and intuitive interface that allows a user to quickly load data from a CSV file, transform and explore the data, build and evaluate models, and export those models as PMML (Predictive Modeling and Markup Language) or as scores.

Why use Rattle? From a user’s point of view, Rattle offers the perfect and simple interface to do all the things mentioned in the previous paragraph. Apart from that, all the R commands are logged and commented on through a log tab, and the user can download them as a script file. However as previously stated, it has a main drawback, which is that some types of classifiers aren´t available.

Rattle isn’t used in this project, however it was intended to be used at first. But as some classifiers aren´t allowed, it was discarded when that problem was found.

MS Word[14]

Microsoft Word is a word processor developed by Microsoft. It was first released in 1983 under the name Multi-Tool Word. Among its features, Word includes a built-in spell check, a thesaurus, a dictionary and utilities for manipulating and editing text.

Why use MS Word? MS Word is a worthy piece of software. As mentioned in the previous paragraph, it enables you to do a lot of things to text documents. Apart from those already mentioned, you can create fancy titles or place watermarks. From my point of view, it’s a great tool, it has a full user-friendly interface and

[13] (Rattle: R n.d.)
[14] (Microsoft Word: Wikipedia n.d.)

apart from the interface it has more advanced tools. It also has the possibility to program macros in VBA to do more automatic things. The interface is divided into main headings, such as Home, Layout…, and when you want to give format to the text, you can easily do it by going into Home and selecting the appropriate style for each section. Having done this, your index will be made automatically. So these are some of the reasons for which I've chosen MS Word.

MS Word is used in this project to write the project report. It´s interface makes the writing part quicker and simpler.

Dropbox[15]

Dropbox is a file-hosting service operated by Dropbox Inc., headquartered in San Francisco, California and founded in 2007 by MIT students Drew Houston and Arash Ferdowsi, which offers cloud storage, file synchronization, personal cloud and client software. Dropbox allows users to create a special folder on their computers, which Dropbox then synchronizes so that it appears to be the same folder regardless of the computer.

The Dropbox client enables users to drop any file into a designated folder. The file is then automatically uploaded to Dropbox's cloud-based service and made available to any other of the user's computers and devices.

Why use Dropbox? I've been able to use other cloud storage apps, such as Mega or Google Drive, however for me Dropbox is the best one. It's true that the limited space is one of Dropbox's main drawbacks, however the good implementation of version control makes up for it. Version control is a very important feature, and other software what they do is to duplicate the file under a different name. What Dropbox does is to replace the file with the new changes. So if you have Dropbox's folder locally on your computer and you are working from Dropbox, every time you save the file it will sync automatically. Apart from this feature, if you have Dropbox installed on your phone, you can make your phone upload every new photo you take to Dropbox, so if your phone breaks down you won't lose your

[15] (Dropbox: Wikipedia n.d.)

pictures. As another advantage, apart from its features, I like its clean and light-colored interface. It’s easy to understand what’s happening at every moment. With its neat sharing interface, it’s also easy to share folders or files.

Dropbox is used as the main storage for the project. The whole project is stored in Dropbox in a folder hierarchy. Every time any part of the project is saved, it is also updated in Dropbox. This avoids the loss of any document.

6.2 Hardware

The hardware during this project has been one of the most important things. Without a powerful computer I wouldn’t have been able to do so many things. However, there were times when I couldn’t do some algorithms due to my lack of powerful hardware.

A MacBook Pro Retina 13” with a 2.6GHz i5 processor, 8GB RAM memory and 128GB of SSD has been used. As I’ve described, it is a quite powerful computer and as I’ve mentioned at times I had problems with it. Other times, I lacked RAM memory but SSD memory would be used instead.

Is this hardware enough? For most of the project it was enough, however if you are planning on having more data I would recommend a more powerful computer. With at least 16GB RAM and a cluster with parallel computing, you would save on time and you would be able to do more classifiers, such as trees.

Part 2: Project Improvements

In the next few chapters some improvements to this project will be commented on. These improvements have been thought through carefully, but haven't been developed due to a lack of time.

Chapter 7

Improvements

This project has a long list of improvements, which doesn't mean that the project's results are wrong or anything like that. It only means that I wasn't too ambitious at the start of the project. I already had an overly big problem to cope with. I had chosen one of the biggest and busiest airports in the world, with over 300 thousand outgoing flights a year to US destinations.

7.1 Data improvements

In this section, some possible data improvements that can be made in the future will be described. Weather data and flight data will be referred to separately.

7.1.1 Flight Data

There are different levels of flight data improvements. To start with, all the ingoing and outgoing flights to and from US destinations, where O'Hare international airport is part of the equation, would be used. So this is any flight that leaves any US airport and lands in O'Hare and any flight that leaves O'Hare and lands anywhere in the US. These would be the first-level improvements.

The second-level improvements would be to choose all ingoing and outgoing flights, where O'Hare is part of the equation, but to and from all over the world. So this is any flight that leaves or lands in O'Hare from or to any part of the world, even if it's just to fuel up.

The third-level improvements would be to not do only one airport, which in this case is O'Hare, but to be able to use any airport in the world. However, these improvements are the most difficult ones to develop, because the flight data for non-US airports is difficult or even impossible to find.

7.1.2 Weather Data

Regarding weather data improvements, one will be mainly described. The METARs would be used instead of the actual weather data. This would be a really big and important improvement, because METAR is the most common format in the world, used by pilots, control towers and so on. This format, makes the weather data be more accessible, however, as stated before, regular expressions must be used to parse the different fields. So if you don’t get along with RegEx, you need a lot of time to find the key.

7.2 Model improvements

The model has at least one important improvement. The class variable values would be changed. Now, it has just two values, 1 or “Yes” that means the flight is going to be delayed and 0 or “No” that means the flight isn’t going to be delayed. The delayed threshold is set at 15 minutes. So from a flier’s point of view, I would prefer to know a more exact delay degree. So at least I would have a class variable with five values.

Table 7.1 Class variable improvements

Class Variable	
Value	**Delay in minutes**
0	<= 14
1	15 to 29
2	30 to 44
3	45 to 59
4	+1h

Another improvement would be to try other more powerful classifiers, which may work worse, but at least you know that’s the result. Having a more powerful computer lets you try with trees and with Naïve Bayes trees and then compare the results to the other classifiers and choose the best one.

Chapter 8

Front-end improvements

This is one of the points I would have loved to work on, but because of lack of time I couldn't. This is a must in the future. "A front-end interface where the final user or client is able to interact with the model". So this app would be hosted on a server and every time a client was to book a flight, they would go to this app and enter their flight data and the app would return a quite accurate prediction, but it is a prediction that can help to book a better connection.

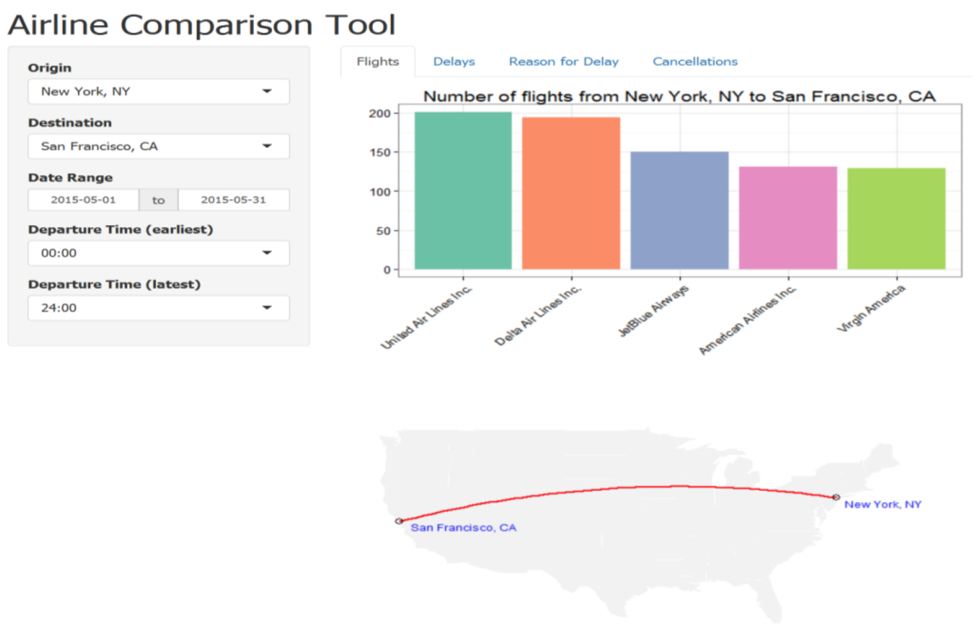
I've only said that the user would enter the flight data, but who or what provides the application the weather data?

8.1 Shiny[16]

The app would be made with Shiny, which is a web application framework that lets you build interactive web applications with R. Automatic reactive binding between inputs and outputs and extensive pre-built widgets make it possible to build beautiful, responsive and powerful applications.

For a better understanding of how Shiny works, reactive programming is a programming paradigm oriented around data flows and the propagation of change. In an imperative programming setting, 'a:= b + c' would mean that 'a' is being assigned the result of 'b + c' when the expression is evaluated. After being evaluated the values of 'b' and 'c' can change with no effect on 'a'. However, in reactive programming the value of 'a' would be automatically updated based on the new values.

[16] (Shiny GitHub n.d.) and (Shiny n.d.)


Airline Comparison Tool

Origin: New York, NY

Destination: San Francisco, CA

Date Range: 2015-05-01 to 2015-05-31

Departure Time (earliest): 00:00

Departure Time (latest): 24:00

Fig. 8.1 Example Shiny Interface

Source: http://nycdatascience.com/airline-performance-comparison-with-rshiny/

8.2 RWeather[17]

In the case the original plan remains untouched, which is to use the weather data in the format I have it now, and not use the METAR format, I would rely on RWeather as a live feed of weather. So this means that every time the user introduces the flight data into Shiny, automatically RWeather provides weather data for predicting the new case. But in the case where the METAR improvement is developed, I would have to look for a live feed of METAR format weather data.

[17] (RWeather: R n.d.)

Part 3: Other aspects of the project

Chapter 9

Conclusions

Throughout this chapter I will present the project´s conclusions, my personal conclusions and I will then refer to the code used in this project.

9.1 Project´s conclusions

All through the project presented here a data-mining model has been created, which is capable of predicting if a flight is going to leave late due to certain weather conditions, with over 69 percent success. I would have never imagined I was going to achieve most of my initial goals. It all started as a simple idea that I presented to my project director, which he liked so I decided to go ahead with it and here I am today.

The first three phases, which are mostly data collecting and preprocessing, have stood for 70% of the project time. This is what prevented me from achieving my last goal, which is to create a front-end system for the user. However, I firmly believe, that after presenting this project and when I have a bit of free time, I would like to create this front-end interface and perhaps carry out some of the data improvements mentioned previously.

Finally, it’s worth highlighting that this idea has been developed from scratch, from the data collecting to the model building, and this has fulfilled my personal expectations.

9.2 Personal conclusions

This project has been a great and difficult personal experience. I’ve been able to deal with some really difficult problems that almost led me to quitting.

Thanks to this project, I’ve learned to overcome obstacles and to keep calm in distended moments.

During the entire project I´ve worked with programming languages such as Python, Visual Basic and R that were not taught at my university. However, this degree has given me the ability to cope with any programming language. All of these programming languages are similar in one way; they follow similar programming patterns and once you’ve learned the distinctive things about it, you are able to create a program.

Finally, I must say that the recent experience with R and Data Science provided by this project has helped me pass the job interviews at a technologic consulting firm in Bilbao.

9.3 Learned lessons

There is a learned lesson that I must point: Ambition.

Ambition is a tricky concept, and everything in it’s right measure is good, even this. Being and overly ambitious person isn’t good as most may think and now I will tell you why. I’ve never been too ambitious, until this project, when I decided to go for high stakes and tackle one of the biggest and busiest airports in the world. However, as a result of my ambition, I had to give up the front-end interface due to a lack of time. If I had gone with a smaller airport, perhaps 100 thousand flights a year, I would have been able to achieve all my goals.

Therefore, the conclusion is that ambition is a good thing, but be realistic about your limits and don’t go beyond them. Most of the time, being too ambitious will play against you.

9.4 Code

Weather Scraper

```
import urllib2
import re
import datetime

weather = open("weather2013.csv", "w")

startDate = datetime.date(2013, 1, 1);
endDate = datetime.date(2013, 12, 31);
delta = datetime.timedelta(days=1)

delta = datetime.timedelta(days=1)
while startDate <= endDate:
   print startDate.strftime("%y-%m-%d ")
   request =
startDate.strftime("http://www.wunderground.com/history/airport/KORD/"+"201
3"+"/"+"%m"+"/"+"%d"+"/DailyHistory.html?req_city=&req_state=&req_statename
=&reqdb.zip=&reqdb.magic=&reqdb.wmo=&format=1")
   response = urllib2.urlopen(request)
   for line in response.read().split('<br />'):

         print >> weather, line
   startDate += delta
```

Explanation

The weather scraper enters a webpage with the selected parameters and keeps the information of that webpage in a variable. Then it looks for the selected tag, which in this case is the '<br />' tag, in the information that was previously kept in that variable, and writes the line up to this tag in a CSV document.

Remove Other Airports

```
Sub removeOtherAirports ()
Dim i As Long

  i = 2
  Do While Not IsEmpty(Cells(i, 3))
    If ((Cells(i,3).Value <> "ORD")) Then
      Rows(i).Delete
    Else
       i = i + 1
    End If
  Loop
End Sub
```

Explanation

This program removes other airports which don't correspond to the origin airport ORD. It does this by searching for the condition in the If and using the Rows(i).Delete.

Change Date

```
Sub changeDate()
Dim i As Long

  For i = 1 To 32500
    If (Cells(i, 1).Value = 1122013) Then
      Cells(i, 1).Value = "2013-12-01"
    ElseIf (Cells(i, 1).Value = 2122013) Then
      Cells(i, 1).Value = "2013-12-02"
    ElseIf (Cells(i, 1).Value = 3122013) Then
      Cells(i, 1).Value = "2013-12-03"
    ElseIf (Cells(i, 1).Value = 4122013) Then
      Cells(i, 1).Value = "2013-12-04"
    ElseIf (Cells(i, 1).Value = 5122013) Then
      Cells(i, 1).Value = "2013-12-05"
       [.........................................]
   ElseIf (Cells(i, 1).Value = 26122013) Then
      Cells(i, 1).Value = "2013-12-26"
    ElseIf (Cells(i, 1).Value = 27122013) Then
      Cells(i, 1).Value = "2013-12-27"
    ElseIf (Cells(i, 1).Value = 28122013) Then
      Cells(i, 1).Value = "2013-12-28"
    ElseIf (Cells(i, 1).Value = 29122013) Then
      Cells(i, 1).Value = "2013-12-29"
    ElseIf (Cells(i, 1).Value = 30122013) Then
      Cells(i, 1).Value = "2013-12-30"
    ElseIf (Cells(i, 1).Value = 31122013) Then
      Cells(i, 1).Value = "2013-12-31"
    End If
  Next i
End Sub
```

Explanation

First of all, the loop range goes from 1 to 32500 this is to avoid Excel from freezing due to the large amount of data. The program is used to change the format to the date parameter. It has to be changed from the continuous form ‘1122013’ to the hyphen separated form ‘2013-12-01’.

Rounded Time

```
Sub roundedTime()
Dim i As Long

   For i = 1 To 32500
      If (Cells(i, 9).Value >= 45100 And Cells(i, 9).Value < 55100) Then
         Cells(i, 8).Value = "451"
      ElseIf (Cells(i, 9).Value >= 55100 And Cells(i, 9).Value < 65100) Then
         Cells(i, 8).Value = "551"
      ElseIf (Cells(i, 9).Value >= 65100 And Cells(i, 9).Value < 75100) Then
         Cells(i, 8).Value = "651"
      ElseIf (Cells(i, 9).Value >= 75100 And Cells(i, 9).Value < 85100) Then
         Cells(i, 8).Value = "751"
      ElseIf (Cells(i, 9).Value >= 85100 And Cells(i, 9).Value < 95100) Then
         Cells(i, 8).Value = "851"
      ElseIf (Cells(i, 9).Value >= 95100 And Cells(i, 9).Value < 105100) Then
         Cells(i, 8).Value = "951"
      ElseIf (Cells(i, 9).Value >= 105100 And Cells(i, 9).Value < 115100) Then
         Cells(i, 8).Value = "1051"
                [......................................]
      ElseIf (Cells(i, 9).Value >= 175100 And Cells(i, 9).Value < 185100) Then
         Cells(i, 8).Value = "1751"
      ElseIf (Cells(i, 9).Value >= 185100 And Cells(i, 9).Value < 195100) Then
         Cells(i, 8).Value = "1851"
      ElseIf (Cells(i, 9).Value >= 195100 And Cells(i, 9).Value < 205100) Then
         Cells(i, 8).Value = "1951"
      ElseIf (Cells(i, 9).Value >= 205100 And Cells(i, 9).Value < 215100) Then
         Cells(i, 8).Value = "2051"
      ElseIf (Cells(i, 9).Value >= 215100 And Cells(i, 9).Value < 225100) Then
         Cells(i, 8).Value = "2151"
      ElseIf (Cells(i, 9).Value >= 225100 And Cells(i, 9).Value < 235100) Then
         Cells(i, 8).Value = "2251"
      ElseIf (Cells(i, 9).Value >= 235100 And Cells(i, 9).Value < 245100) Then
         Cells(i, 8).Value = "2351"
         Else
                Rows(i).EntireRow.Delete
      End If
   Next i
End Sub
```

Explanation

As previously mentioned, most of these programs use small loop ranges. In this case, the program is used in order to have a rounded time that represents all the values during one hour. For example, it selects every time from 23:51 inclusive to 24:51 and gives them a 23:51 value.

Delete Blank Rows

```
Sub deleteBlankRows()
Dim i As Long

   For i = 1 To 32500
      If (IsEmpty(Cells(i, 1)) Then
         Rows(i).EntireRow.Delete
      End If
   Next i
End Sub
```

Explanation

When the weather scraper was used, it wrote between each line of data a blank row as well. This program is used to delete those blank rows.

Keep Only Weather Hours Needed

```
Sub keepOnlyWeatherHoursNeeded()
Dim i As Long

   For i = 1 To 32500
      If (Cells(i, 1).Value < 451 And  Cells(i, 1).Value > 2351)
         Rows(i).EntireRow.Delete
      End If
   Next i
End Sub
```

Explanation

The weather data, apart from the hours needed, had more hours. This program is used to eliminate those hours that weren't needed. It does this by searching for the condition and eliminating that entire row.

Not NA DEP_DEL15

```
Sub notNADEP_DEL15 ()
Dim i As Long

   For i = 2 To 10500
      If Cells(i, 13).Value = vbNullString

         Rows(i).EntireRow.Delete
      End If
   Next i
End Sub
```

Explanation

The whole point of this program is to eliminate the rows where the class variable, which in this case is DEP_DEL15, is not a value.

Chapter 10

Project Management

This project has had different working stages. During data collection I also managed to read and learn about R. Then there was a series timeline, when I cleaned and data mined the data, first with one program and then with the other one, and finally I wrote the report.

10.1 Task management

At the start of the project, I defined the scope and broke down the project into different tasks. However, I didn't think that defining an entire project plan was efficient because there was too big of a time-gap and the project was too open to changes in schedule, so I decided to do small time gap plans to be able to use them as a project guideline. Although that guideline was there, in most of the cases I wasn't able to follow it, because risks that initially seen as possible occurred, so I had to postpone tasks or change them in the timeline. In the 10.1 figure I will show a work breakdown structure, which represents the main tasks of the project.

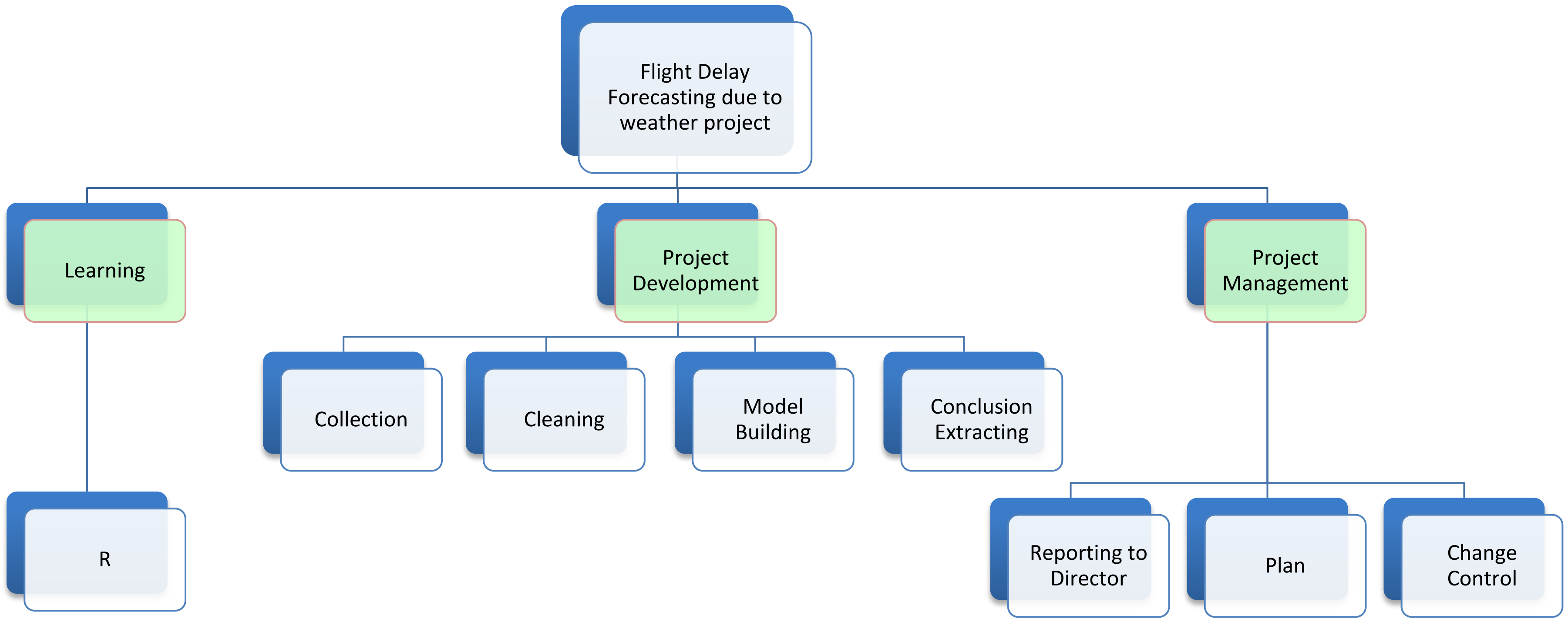


Fig. 10.1 Work breakdown structure tree. The main categories are showed in light green. The tasks are showed in light blue.

10.2 Scope management

In the next table I will talk about my defined objectives and describe whether I've accomplished them or not and with what grade of quality. I will just describe these two parameters, because the procedure and the task itself has been described previously.

Table 10.1 Objectives

Objectives	Accomplished?	Quality
R Knowledge	✓	Acquired basic R knowledge. Caret.
Dataset Collecting	✓	Weather and flights with good data quality.
Obtaining Cleaned Dataset	✓	Reliable dataset.
Data mining model	✓	Different models to be able to compare.
Making Conclusions	✓	Good and reliable conclusions.
Shiny	✗	Not done.

I've accomplished 5 out of 6, though the reason why I didn't accomplish Shiny was due to a lack of time close to the date-limit. I had to choose between presenting in November or leaving Shiny out and presenting in September and, after meditating both options I've gone for the one of leaving Shiny out.

10.3 Time management

The project started in September 2014 but, because I had other subjects with earlier marking dates, there would be times where I had to focus on those and leave the project development for later on. The project ends in September 2015. Next I will show the most relevant milestones that have taken place throughout the project. As the different milestones took more than one day, I will state the period of time when those milestones took place.

End September 2014: Had collected flight data
End October 2014: Had cleaned flight data
End February 2015: Had collected weather data
Mid March 2015: Had cleaned weather data
End March – First April: Had merged both datasets
End April: WEKA model
Mid May – End July: Caret
End April – End July: Conclusions over model

10.4 Risk Management

Throughout the project some planned risks occurred. Some tasks had to be rescheduled or postponed due to a lack of time. In the following table, the most important risks that occurred will be presented and the solution will be explained.

Table 10.2 Risk management

Risk that appeared	Solution taken
Subject Assignments	Postpone tasks until more free time
Subject Exams	Postpone tasks until after exams
Christmas	Dedicate less time but at least some
Software problems	Try different computers
Caret problems	Ask on Stackoverflow
July 2015 Holidays	Take some days off but try to catch up after

10.5 Communication Management

Throughout the project different methods of communication have been used. Most of the communication has been via email. We found email to be a good way to communicate different ideas, questions, status, and thoughts about the project. There has been at least 1 email every two weeks. When the follow up was too complicated via email, we would have a meeting and comment on the status of the project, what was done, how was it done and what there was left to do until the next meeting.

10.6 Cost Management

In the next subchapter the project cost's represented in hours and minutes will be presented and then explained.

Table 10.3 Project cost (hours)

Task	Cost (H)
Problem Understanding	5
Data Collection	23
Data Cleaning	84
Data Merging + Transformation	35
Data Mining (WEKA)	17
Data Mining (R)	52
Books (R and Python)	40
Plan	25
Memory	60
Project Presentation (estimated)	23

In the next page there are three charts. The first one corresponds to the project hours as a percentage, the second one corresponds to the preprocessing vs the WEKA model and the third one corresponds to the preprocessing vs the R model. As I said at the introduction of part 1, data preprocessing accounts for 70%-80% of a data mining project time, these percentages can be seen in the last two charts.

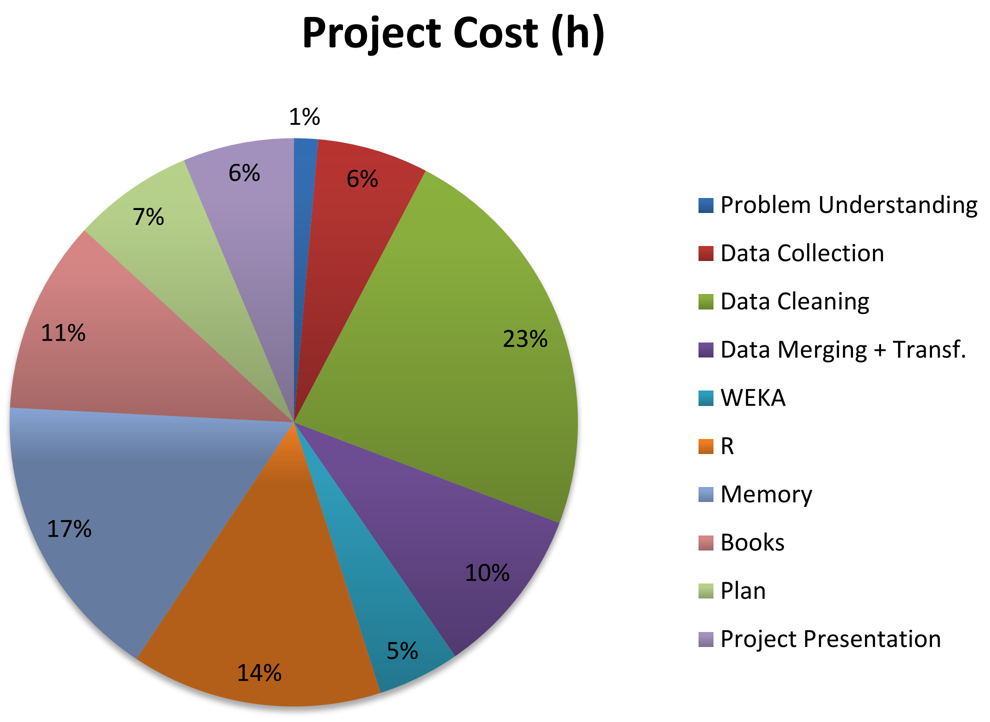


Fig. 10.2 Project cost pie chart

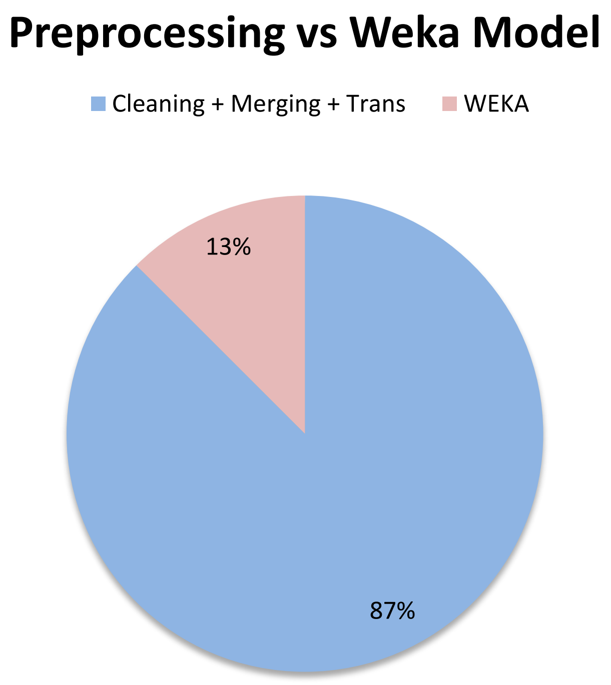


Fig. 10.3 Preprocessing vs Weka Model

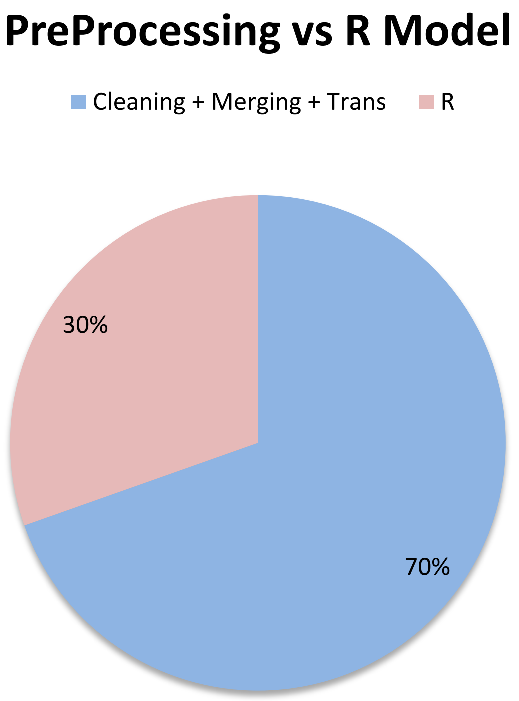


Fig. 10.4 Preprocessing vs R Model

Bibliography and References

Aircarrier: United States Census Bureau. https://www.census.gov/foreign-trade/reference/codes/aircarrier/acname.txt (accessed 2015).

Airport History: Wunderground. 2013. http://www.wunderground.com/history/airport/KORD/2013/2/1/DailyHistory.html?req_city=KORD&req_state=IL&req_statename=Illinois&reqdb.zip=60666&reqdb.magic=5&reqdb.wmo=99999&MR=1.

Bureau of Transportation Statistics. 2013. http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time (accessed 2014).

CfsSubsetEval: Pentaho. http://wiki.pentaho.com/display/DATAMINING/CfsSubsetEval (accessed 2015).

Cost Sensitive Classifier: Sourceforge. 2015. http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/CostSensitiveClassifier.html.

Cost Sensitive Classifier: WEKA Wikispace. 2015. https://weka.wikispaces.com/CostSensitiveClassifier.

Delaney, Samuel. "Planespotting. Using machine learning algorithms to predict flight delays at time of ticket purchase." Paper, CSE 773C, University of Nevada Reno.

Documentation: Wunderground. 2015. http://www.wunderground.com/weather/api/d/docs?d=resources/phrase-glossary&MR=1#current_condition_phrases.

Dropbox: Wikipedia. https://en.wikipedia.org/wiki/Dropbox_(service).

Estimate Model Accuracy in R: Machine Learning Mastery. http://machinelearningmastery.com/how-to-estimate-model-accuracy-in-r-using-the-caret-package/ (accessed 2015).

ETH Zurich. "Flight Delay Prediction." Master's Thesis, ETH Zurich, 2011-2012.

Excel: WikiHow. http://es.wikihow.com/insertar-una-consulta-SQL-en-Microsoft-Excel (accessed 2015).

Fayyad, Usama. Yahoo Research. 2007. www2.cs.uh.edu/~ceick/DM/fayyad.ppt (accessed 2015).

Google Fusion Tables: Wikipedia. https://en.wikipedia.org/wiki/Google_Fusion_Tables.

Help: Fusion Tables Google. https://support.google.com/fusiontables/answer/2571232.

IATA Codes: One World Nations Online. http://www.nationsonline.org/oneworld/IATA_Codes/airport_code_list.htm.

Kock, Matthew de. "Weather Forecasting using Dynamic Bayesian Networks." Honours Project, Computer Science, University of Cape Town, 2008.

McKinney, Wes. *Python for Data Analysis.* Edited by Melanie Yarbrough, Julie Steele and Meghan Blanchette. O'Reilly, 2013.

Metar FAQ: Wunderground. http://www.wunderground.com/metarFAQ.asp (accessed 2015).

METAR: Atmospheric Sciences A&M Texas University. http://www.met.tamu.edu/class/metar/quick-metar.html (accessed 2015).

METAR: Wikipedia. https://en.wikipedia.org/wiki/METAR (accessed 2015).

Microsoft Word: Wikipedia. https://en.wikipedia.org/wiki/Microsoft_Word#Features_and_flaws.

MIT. "A Network Based Model for Predicting Air Traffic Delays." Computer Science, MIT, 2012.

Naive Bayes: Wikipedia. https://en.wikipedia.org/wiki/Naive_Bayes_classifier (accessed 2015).

Python: Wikipedia. https://en.wikipedia.org/wiki/Python_(programming_language).

R: Wikipedia. https://en.wikipedia.org/wiki/R_(programming_language) (accessed 2015).

Rattle: R. https://cran.r-project.org/web/packages/rattle/index.html.

RStudio: Wikipedia. https://en.wikipedia.org/wiki/RStudio.

"RWeather: R." *RWeather.* https://cran.r-project.org/web/packages/RWeather/RWeather.pdf.

Shiny. http://shiny.rstudio.com/.

Shiny GitHub. http://rstudio.github.io/shiny/tutorial/.

Springer. *Data Mining with Rattle and R.* Edited by Robert Gentleman, Kurt Hornik and Giovani G. Parmigiani. Springer.

Stack Overflow. http://stackoverflow.com/questions/24853799/how-do-i-diagnose-unable-to-create-socket (accessed 2015).

Stanford. "Predicting Flight Delays." Computer Science, Stanford, 2012.

Stefanski, Tim. "Predicting Flight Delays Through Data Mining." Computer Science 105, Boston University.

Sublime Text 2: Wikipedia. https://en.wikipedia.org/wiki/Sublime_Text (accessed 2015).

The Caret Package. http://topepo.github.io/caret/index.html.

Training: The Caret Package. http://topepo.github.io/caret/training.html (accessed 2015).

Urllib2: Python. https://docs.python.org/2/library/urllib2.html (accessed 2015).

VBA Programming: Wikipedia. https://en.wikipedia.org/wiki/Microsoft_Excel#VBA_programming.

VBA: Excel Easy. http://www.excel-easy.com/vba.html.

WEKA. http://www.cs.waikato.ac.nz/ml/weka/.

WEKA: Wikipedia. https://en.wikipedia.org/wiki/Weka_(machine_learning).