UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado Ingeniería Informática

Trabajo Fin de Grado

Empirical study of dimensionality reduction methodologies for classification problems

Autor: Daniel Gerald Orbegoso Barrantes

Trabajo realizado en: DBIS Goethe Universität

País: Frankfurt am Main

Tutor: José Luis Castillo Sequera

Cotutor: Kim Hee

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

FECHA:

A mi padre y a

mi madre, por creer en aquel niño cuyos

profesores decían que jamás llegaría a ser nada.

Agradecimientos

A mi madre, por ser la constancia, apoyo en mis momentos de debilidad y la luz en mis momentos de ceguera.
A mi padre, por ser la roca cuando yo era frágil, por enseñarme que un no puede ser un si, por regalarme su vida.
A mi hermana, por enseñarme que el mundo es grande y hermoso, a no tener miedo a lo nuevo y a luchar por lo imposible.
A mi tío Cornelio, por ser un padre cuando le necesitaba, por darme su tiempo sin condiciones.
A Martin García, Santiago Cataldo, Fernanda Paraizo y Santi Nuñez, gracias por haberme apoyado y estado a mi lado en mi camino.
A mi profesor tutor José Luis Castillo Sequera y mi profesora co-tutora Kim Hee, por confiar en un joven que solo tenía inquietudes y quería aprender, por regalarme su tiempo.

Gracias.

Table of contents

Table of Tables

Table of Figures

Resumen

Cuando hablamos de “Dimensionality Reduction” en Informática o “Big Data” nos referimos al proceso de reducción de variables previamente examinadas de un conjunto de datos para poder así obtener un conjunto de variables menor que nos permitirá construir un modelo de datos igual o con mejor precisión y menor cantidad de datos.

Con este propósito se aplican técnicas de “Feature Selection” y “Feature Extraction”, con la primera de ellas extraemos un conjunto de características importantes de un dataset mediante el uso de distintos algoritmos de “machine learning”, mientras que con la segunda obtendremos un nuevo conjunto de características obtenidas a partir de las características originales.

En este trabajo de fin de grado hacemos un estudio empírico sobre las distintas metodologías para clasificación de problemas utilizando un dataset médico llamado NCS-1 de pacientes clínicos con distintas patologías médicas, estudiamos los distintos algoritmos que se pueden aplicar a cada caso determinado con dicho dataset, y finalmente con los datos obtenidos realizamos un benchmark que nos permite entender mejor los distintos modelos estudiados.

Palabras clave

Data mining, Dimensionality Reduction, Python, NCS1, Feature Engineering, Mental health, DSM-III-R disorders.

Abstract

When we speak about Dimensionality reduction in informatics or big data, we refer to the process of reducing the number of random variables under consideration, and so, obtaining a set of principle variables which allow us to build a data model with the same or similar accuracy and a lower amount of data.

For this purpose, we apply feature selection and feature extraction techniques. With feature selection we select a subset of the original feature set using techniques of machine learning, and with feature extraction we are going to build a new set of features from the original feature set.

In this Project, we are going to make an empirical study about the different methodologies for classification problems using a medical dataset called NCS-1 of clinical patients with different medical pathologies, we study the different algorithms that can be applied for each case with this dataset, and finally with obtained data developing a Benchmark to understand the different applied models.

Keywords

Data mining, Dimensionality Reduction, Python, NCS1, Feature Engineering, Mental health, DSM-III-R disorders.

Extended Abstract

Below we summarize in more detail the structure that we will see in this end-of-grade paper, with detailed descriptions of each part. I hope that this end-of-grade paper will help the next generations as support in their eagerness to understand how to process and work with the data, to make them more efficient.

- **Description:**

  The result of this research is a detailed memory in which we can find separated blocks explaining the development and the necessary information to understand each step realized across:

  - **Block 1: Introduction and Objectives:** In the Introduction, we are going to speak about the challenges we faced in this project, the reason that we selected this topic, the hypothesis we want to validate and the methodologies used, giving a small overview of the problem.
  - **Block 2: Background:** In the Background, we give an introduction to different fields, which need to be understood if we want to understand the results of this project. The first part gives an introduction to what is Big Data, Data Scientist and Machine Learning. The second part speaks about the different dimensionality reduction techniques and the machine learning algorithms used in this project. The third part focus on some data mining techniques we thought have to be explained and a small introduction to the metrics used to understand the results. Finally, we give some information about important libraries and applications needed to develop this project.

- **Block 3: Methodology and Experiment design:** In this block, we clarify first the methodology followed to develop this project, the CRISP-DM methodology for data mining projects. In second place, we explain step by step the procedure applied to our project, since the beginning, when we started to collect the data until the moment we obtained our results. This part is going to cover both, Experiment and Experiment design at the same time
- **Block 4: Results and analysis (Discussion):** This part is focused in the comprehension of the results obtained with this experiment. After this, we will analyze the information obtained, discussing the reason for the results obtained and giving our own appreciation of them.
- **Block 5: Conclusion and Future research:** We make a summary of the project in the conclusion, answering the initial questions about this field. As well, we give our opinion about possible further directions.

Block 1

Introduction

1.1 Overview

Nowadays the quantity of data is emerging exponentially with a high density of noise, and data are becoming more complex than ever before. One of the main reasons is that complexity reduction is not a priority when data are stored in a big data lake. The issues in a highly complex data set are as follows: it hinders from achieving data mining objectives and it demands high computational costs because it requires more time to capture a meaningful insight from such data sets. In order to solve the above-mentioned problem, one solution is to use dimensionality reduction techniques. A dimensionality reduction technique is able to extract meaningful knowledge from a large-scale and complex data set that sits in a high dimensional space.

These studies have been used in the past in many different fields or cases, for example in text domains (*George Forman; 2003.)* or bioinformatics (*Yvan Saeys, Iñaki Inza ,Pedro Larrañaga, 2007)* [1], [2]

The extraction of huge amounts of data in medical fields is becoming a heated issue in the realm of data mining. Usually, medical datasets involved in diagnostic models are high-dimensional, which means increasing the complexity of the classification, reducing the effect of the models. Data mining and machine learning algorithms depend on classification, and the growth of the size exceeds the ability of the classifiers to make correct predictions. Thus, before building models, it is necessary to reduce the dimensions of the dataset, because usually most of the information can be described by a few dimensions.

Dimension reduction methods are primarily divided in Feature Selection and Feature Extraction Algorithms.

With Feature extraction, we transform the original feature space into a new one of lower dimension. However, feature extraction algorithms generate new parameters after dimensionality reduction, eliminating interpretability. In this experiment, the algorithms PCA and LDA will be performed to reduce the number of dimensions as we can see in Figure 1[3].

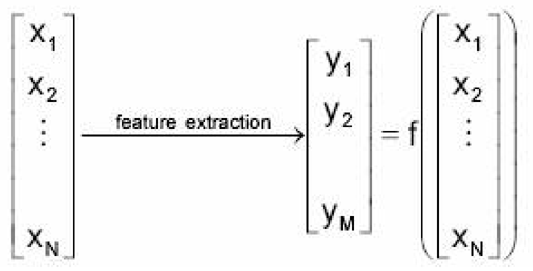


Figure 1. Feature Extraction approach reducing dimensionality

Figure 2[3] illustrates the Feature selection technic, which selects an optimal feature subset from the original features. Although Feature selection is robust against irrelevant features, occasionally fail to retain important information present in the original Feature space. In this experiment, the feature selection algorithms performed are MRMR, Low Variance, Random Forest, Univariate Feature Selection and Extratreeclassifier.

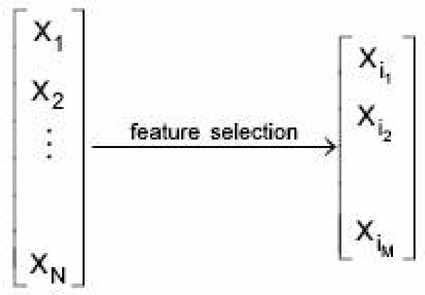


Figure 2. Feature Selection approach reducing dimensionality

1.2 Project objectives

The main purpose of this final degree is both a personal learning of the subject the one I find quite interesting and complement my knowledge of Big Data Technologies and Analytics, as well as a first approximation to this Data Scientist field.

The background knowledge in fields like analysis, design, and implementation of several states of the art dimensionality reduction techniques on a binary dataset or the different scientific methodologies to extract knowledge from the data can be considered the main objective. It could be considered the capacity to use and implement new tools as Jupyter Notebook or the library Sci-kit in the laboratory of the DBIS in the Goethe Universität the second objective.

The different files used for testing the different algorithms in the laboratory can be reused and in the future giving the possibility to test new algorithms in the different files, modify them or even be used in the classes by the professors who want to experiment with them. This document is the resultant memory.

1.3 Methodological approach

In this paper, we implement the data mining process CRISP-DM on the NCS-1 dataset, (The National Comorbidity Survey). It is a robust and well-proven methodology which describes common approaches used by data scientist experts. We are going to use it because of the flexibility and usefulness applying Feature Engineering, seeking to understand which one of the different algorithms perform better and give us a better prediction on a binary classification problem.

Block 2

Background

Before we begin to go deeper, we are going to explain some terms that should be understood and clarify doubts that could appear in this field. We will begin by explaining the difference between fields that could be confused as the data scientist or big data. After this, we will deal with the different classification algorithms introducing the contrasts between the different types of algorithms and giving a small introduction to them.

Since we are going to analyze the functioning of the algorithms, we must understand the metrics used to analyze them and what measures have been taken to evaluate our dataset. Finally, we will explain which applications or libraries have been used to carry out this project.

2.1 Big Data, Data Science, Machine Learning and Data Mining

A strong confusion exists among the concepts **data science, big data, data mining and machine learning** nowadays. To be honest, at the beginning for me was the same situation and I think would be correct to start this TFG with a small introduction of this terminologies to clarify some doubts about what are we doing, how are the technologies connected and where can we place this project.

Nowadays many companies use **big data** to extract and collect data from many sources, but big data needs of the data science specialist to make the data useful and extract value. This is the next step in the process and something that is not within everyone's grasp [4].

The **data science** is the process of understanding, analyzing and creating actionable insights from data. With data science, the data scientist tries to answer a particular question or solve particular problems. There are many industries that could use it to obtain a privileged insight:

- Biotech
- Energy
- Finance
- Gaming and hospitality
- Government
- Healthcare
- Insurance
- Internet
- Manufacturing
- Pharmaceutical
- Retail
- Telecom
- Travel and transportation
- Utilities

Data science can be used to measure regional market share [5], understand consumer sentiment [6], predict elections [7], and even make better wine.[8]

As we can see in figure 3[9], when we speak about **Data science**, we are speaking about a lot of different fields the one involves **Big data**, Data Analytics, Methods and Algorithms, Data Mining and Machine Learning. For this purpose, we use specific tools and methodologies as will be described in the next pages.

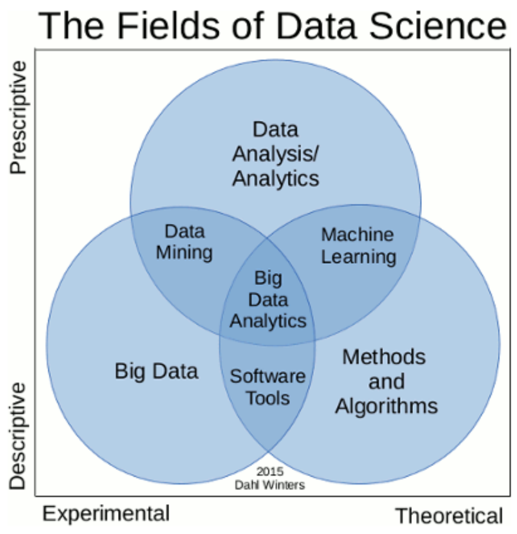


Figure 3. The fields of data science

This TFG is going to be focused on **feature engineering**[10], it is the process of using domain knowledge of the data to create features that make machine learning algorithms work. **Machine learning**[11] is a sub-field of data science, focused on designing algorithms with the idea of learning and make predictions of the data. We don't have to confuse it with **data mining**, which is the analysis of the data trying to find unknown properties of the data. We don't have to overlap those terms because all they work together and we won't need all of them in our project, but it would be a nice point to start explaining the correlation, the differences, and interaction between them. We can see in Figure 4 the machine learning process [12].

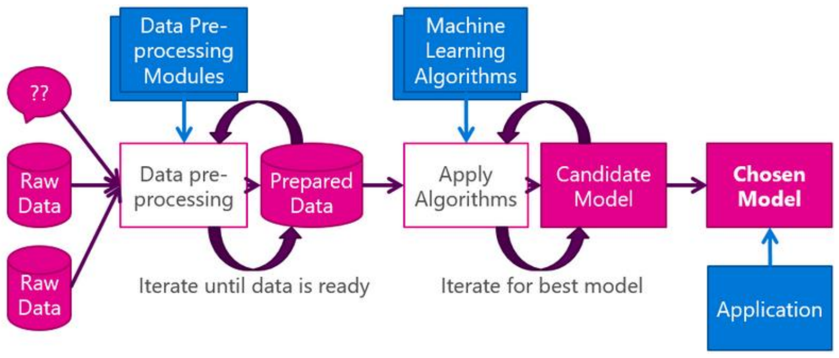


Figure 4. Machine learning process

2.1.1 Big Data

We can understand Big data as mostly digital unstructured data in huge quantities which are so complex to be processed by traditional data processing application software. We could say that it is a part of the data science, where our data is so big that we require overcoming logistical challenges to deal with it.

The main purpose of big data is to capture, extract and process the data, analyzing information from large datasets. For this purpose, there are special techniques and tools such as Hadoop ecosystem, Hortonworks ecosystem, and Apache tools. A primary component of big data is the so-called Three Vs (3Vs) model, the one represents the characteristics and challenges of big data. IBM purposed later one challenge more, "Veracity":

- **Volume**: nowadays the quantity of data is increasing exponentially each year, but it is not that this data was not there before, or that it is something totally new. The data was there, but before we didn't have the tools to collect them properly. In 2010 Thomson Reuters estimated that the world had 800 exabytes of data and EMC moreover thought that it was close to 900 exabytes and it would grow around 50 percent every year[13].We can appreciate this information in the figure 5[14].

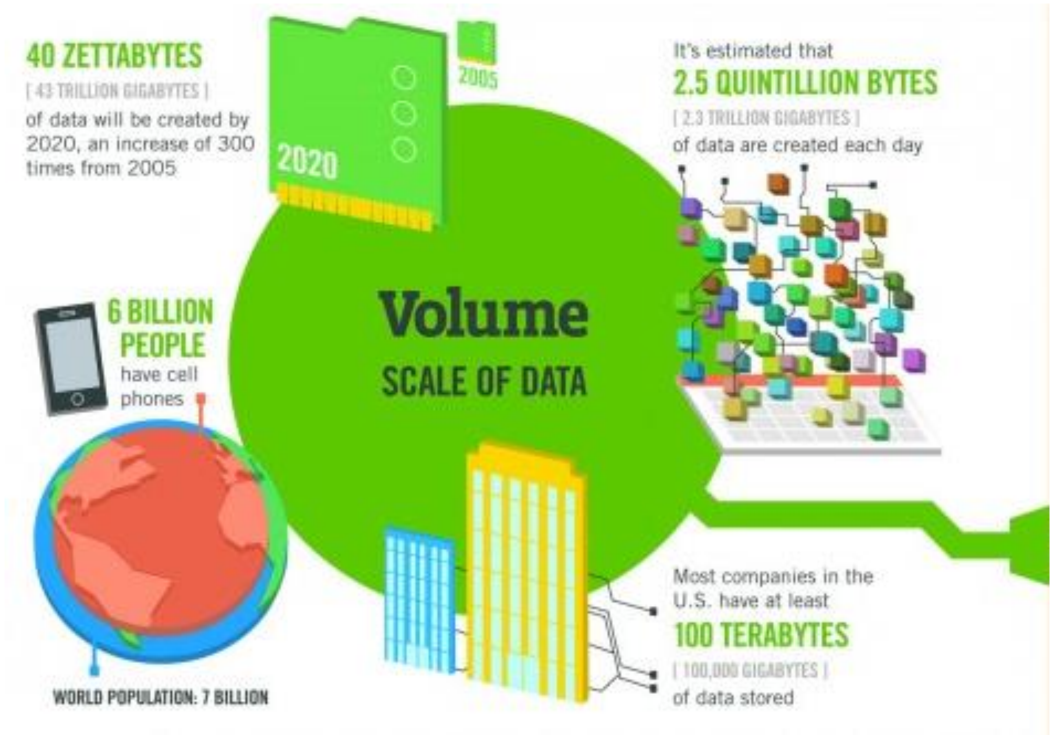

40 ZETTABYTES
[43 TRILLION GIGABYTES]
of data will be created by 2020, an increase of 300 times from 2005
2020
2005
It's estimated that
2.5 QUINTILLION BYTES
[2.3 TRILLION GIGABYTES]
of data are created each day
Volume
SCALE OF DATA
6 BILLION PEOPLE
have cell phones
WORLD POPULATION: 7 BILLION
Most companies in the U.S. have at least
100 TERABYTES
[100,000 GIGABYTES]
of data stored

Figure 5. Volume dimension.

- **Variety**: more and more information is being digitized. The traditional data types are structured like in a bank statement where everything is organized, but nowadays this kind of data is augmented with unstructured data, where information from Twitter, audio files, web pages, weblogs, are classified. All kind of data that can be captured and has not a meta-model fits in the classification as unstructured data.

With unstructured data, we have not any rules. A picture, voice recording or tweet can be different but express ideas and thoughts based on human understanding. The big data takes this unstructured data and make sense of it. We can appreciate this information in figure 6[14].

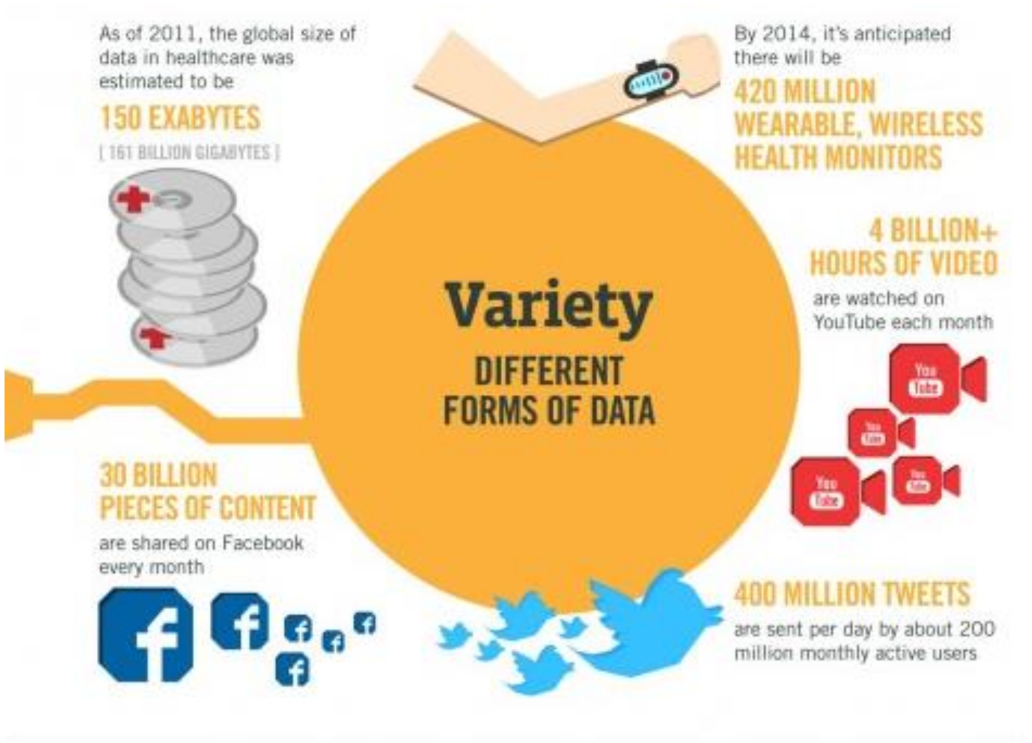

As of 2011, the global size of data in healthcare was estimated to be
150 EXABYTES
[161 BILLION GIGABYTES]
By 2014, it's anticipated there will be
420 MILLION WEARABLE, WIRELESS HEALTH MONITORS
Variety
DIFFERENT FORMS OF DATA
4 BILLION+ HOURS OF VIDEO
are watched on YouTube each month
30 BILLION PIECES OF CONTENT
are shared on Facebook every month
400 MILLION TWEETS
are sent per day by about 200 million monthly active users

Figure 6. Variety dimension.

- **Veracity**: can we trust the data we are collecting? Do we have inherent discrepancies in all the data collected? Veracity refers to the trustworthiness of the data, the noise, and abnormality. When you scope out your big data strategy, you need to clean and keep dirty data from accumulating in the system. We can appreciate this information the figure 7[14]

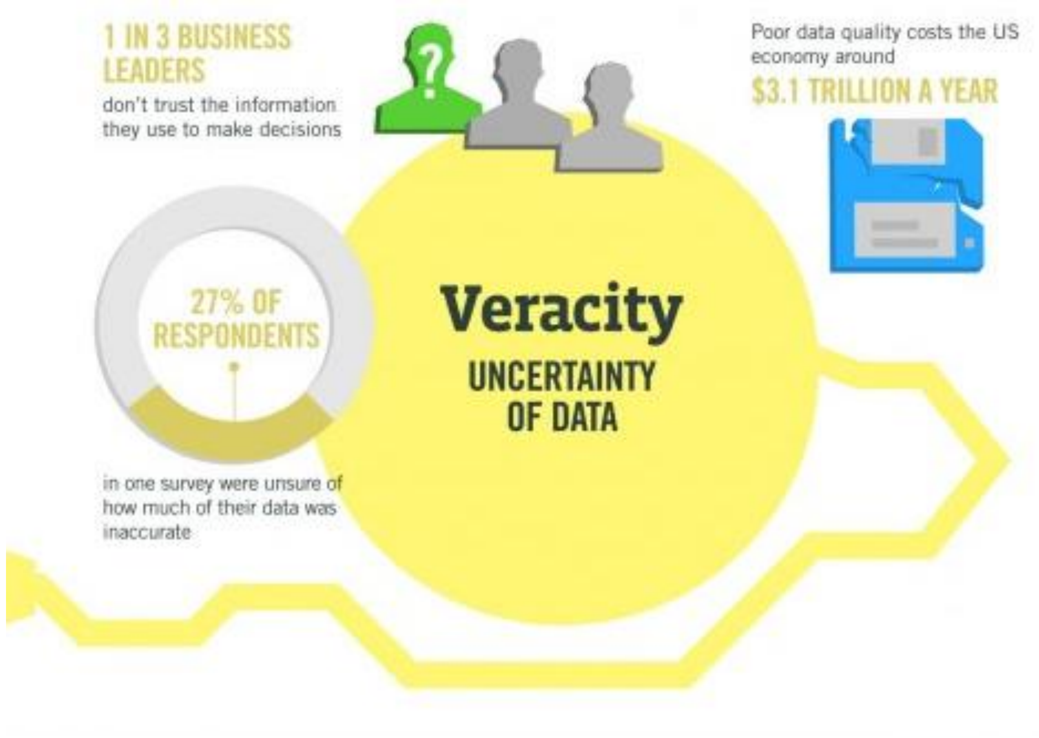

1 IN 3 BUSINESS LEADERS
don't trust the information they use to make decisions
Poor data quality costs the US economy around
$3.1 TRILLION A YEAR
27% OF RESPONDENTS
in one survey were unsure of how much of their data was inaccurate
Veracity
UNCERTAINTY OF DATA

Figure 7. Veracity dimension.

- **Velocity**: with this term, we refer to the frequency in the incoming data needed to be processed. Data comes from everywhere and each source is different, the flow of data is massive and continuous, therefore if we are able to handle the velocity of the incoming data, we would obtain a close-up view of the situation giving us a great advantage over our competition. We can appreciate this information the figure 8[14].

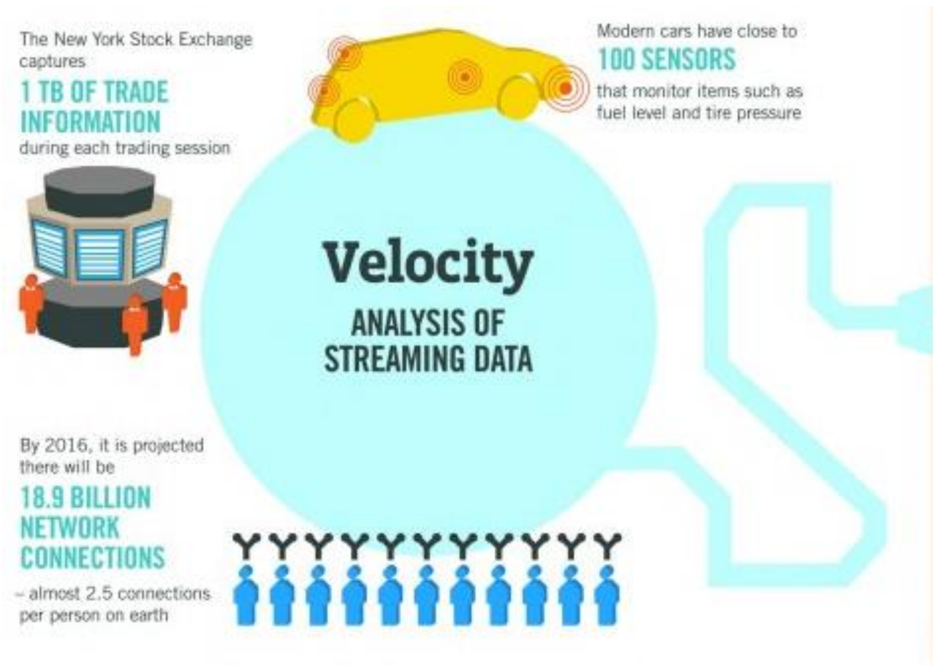

The New York Stock Exchange captures
1 TB OF TRADE INFORMATION
during each trading session
Modern cars have close to
100 SENSORS
that monitor items such as fuel level and tire pressure
Velocity
ANALYSIS OF STREAMING DATA
By 2016, it is projected there will be
18.9 BILLION NETWORK CONNECTIONS
– almost 2.5 connections per person on earth

Figure 8. Velocity dimension

When we speak about Big data we speak about **Hadoop** and the **Hadoop ecosystem.** Hadoop is an open source software framework created for distributed storage and processing of dataset of big data using the MapReduce programming model. The Hadoop core is composed of the distributed file system HDFS, where we are going to store the data and Map Reduce, which will process large-scale data. Hadoop and its entire ecosystem are being developed by Apache. We can appreciate the Hadoop ecosystem in Figure 9[15].

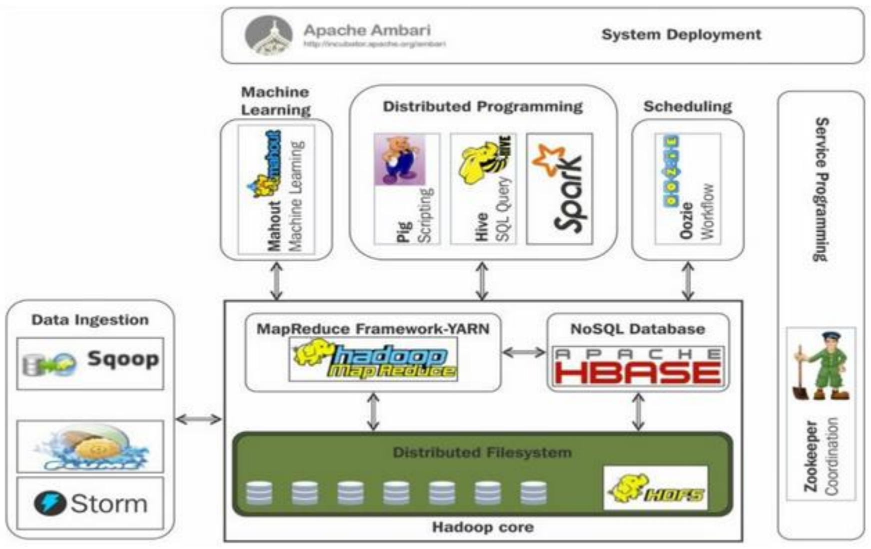


Figure 9. Hadoop ecosystem

I would like to mention some technologies from the Hadoop ecosystem like Apache Pig, Apache HBase, Apache Hive and especially **Apache spark**, which allows us to process the data faster than MapReduce and is one of the states of the art technologies and a referent in the Materia. It was developed in the University of Berkley, as a solution and improve for Map Reduce. It allows us to apply machine learning algorithms, stream data, process graphs and give us SQL support. It allows you to process data faster and easier than MapReduce. We can see in figure 10 the ecosystem of Apache Spark[16]

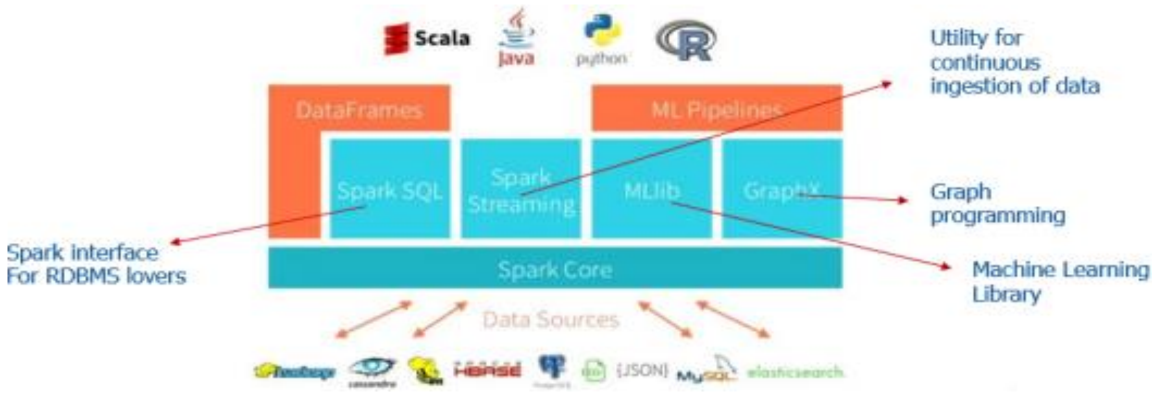


Figure 10. Apache Spark ecosystem.

2.1.2 Data Science

Data science is quite complex as we could see in the diagram on the beginning. It involves many specific domains and skills, everything related to the preparation, cleansing, and analysis of the data. It deals with structured and unstructured data to extract information and knowledge.

We can speak about Data Science like a cluster which involves many fields like mathematics and statistics, computer science and programming, statistical modeling, database technologies, signal processing, data modeling, artificial intelligence and learning, natural language processing, visualization, predictive analytics, and so on.

Machine learning [17] is an essential method included in the process of extracting the knowledge using different modeling algorithms. Data Science includes too:

- problem formulation
- exploratory data analysis
- data model compiling
- data visualization
- data extraction

We don't have to forget as we mentioned before that **Big Data** is a special application of data science. The data is everywhere and is found in huge quantities. With the Data science, we can:

- visualize how this data is discovered
- extract the data
- process and analyze the data
- interpret the data

- model the data with different algorithms
- visualize the data to obtain patterns
- report on the data and present it regardless of the size of the data processed.

I was able to find an interesting sheet in figure 11 with the steps performed by IBM, it can be followed to understand the data science process[18].

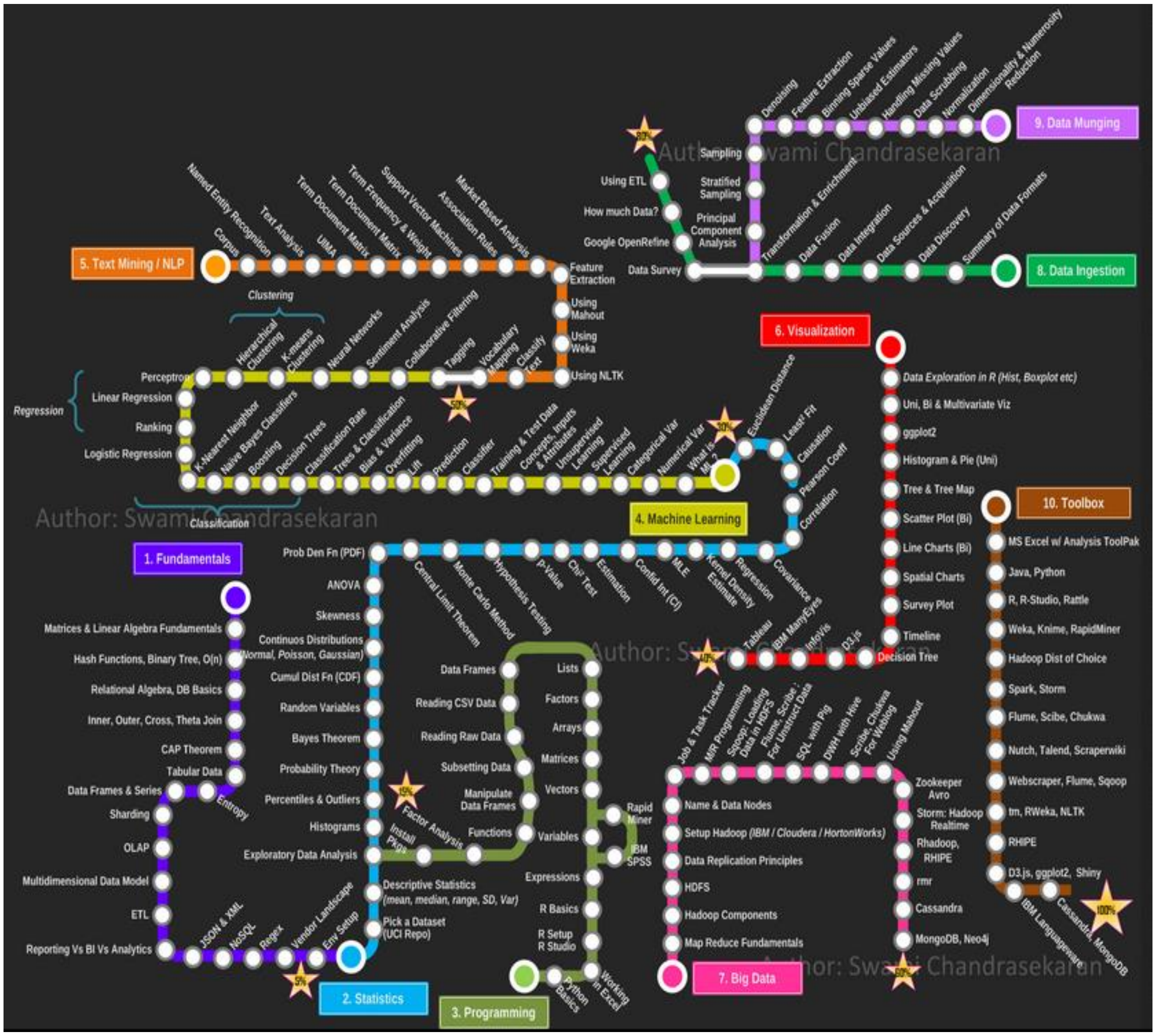


Figure 11. Data science domains.

The number of fields to be applied is huge too, including:

- <u>Social media</u> – the average American spends 2 hours each day in social networking. Data scientists are learning how to describe surprisingly accurate stories about people from that accumulation of information.
- **<u>Medicine and Healthcare</u>** – as in our case, we are trying to predict with the patterns discovered in the health history of a patient the likelihood of having a disease or not. In the Wired magazine, we can find how data science is supplementing doctors, with an exponential use since 2013 of HER(electronic health records)[19]
- <u>Security</u> – Thanks to recently cyberattacks, data science is booming now and the security analytics market is set to reach 8 million dollars by 2023 [20].
- <u>Social sciences</u> – Nowadays social sciences benefit from the growing accessibility and availability of data sources, without forgetting the developments in computational tools for data collection and analysis[21]
- <u>Engineering</u> – Exploit the availability of data to its fullest extent to improve the decisions we take or increase productivity and deepen our understanding of scientific questions is one biggest today's challenge. Several institutions are offering a master stadium in this direction.
- <u>Defense</u> – For the military and counter-terrorist forces, the challenge for data science is to blend automated detection. Getting the data from drones or sensors, we can convert a stream of data into contextual information which the soldiers can use [22]
- <u>Business</u> – Nowadays the technologist able to map out, mash up and mine data for a strategic benefit is pivotal to an organization's direction. [23]

- Economics and Finance – There are several incubators and companies supporting new startups in Economics and Finance, and it is not an accident. The profile of the economist with programming skills is very much in demand nowadays by fintechs companies. Companies like Appzen or Cyence have born in San Francisco with the purpose to give the user the facility of detecting a fraud or bringing together cybersecurity and economics. [24] [25]
- Marketing – For years the marketing roles have dominated this field, leaving out of the game the data works. But many companies started to realize the benefits of using data science for analytics or visualization[26].
- Geolocation – Monsanto started to track the different trucks of the companies, giving the plant managers information about the food. Where is the truck at this moment, which temperature do we have in this point, how much time is going to take to the destination, etc... all this information allow us to prioritize which trucks made it into the manufacturing plan first, improving its supply chain.[27]
- and much more.

2.1.3 Machine Learning

If I would have to use a phrase to explain what is Machine Learning, I would say that *"Machine learning teaches computers to do what comes naturally to humans: learn from experience"*[28]. Machine learning is the science of getting computers to act without being explicitly programmed. It is the ability of computer programs to analyze big data, searching patterns in the data useful for the researchers and that usually are not visible for human's point of view, obtaining information automatically and learning from it. In figure 12 we can see the machine learning process [17].

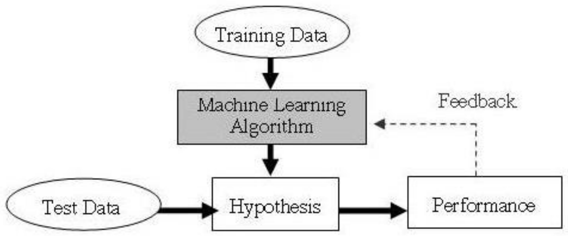


Figure 12. Machine Learning process

It implies that we need algorithmic models that describe a certain process, making predictions for a subject and adjusting it itself in the case the accuracy is not correct at all. For example, Amazon's machine learning algorithm recommend you a product based on your customer's browsing and purchasing behavior[29]. We can find an interesting sheet in figure 13, used by Microsoft Azure, explaining the process used to apply a different algorithm depending on our goals and the data we are analyzing [30].

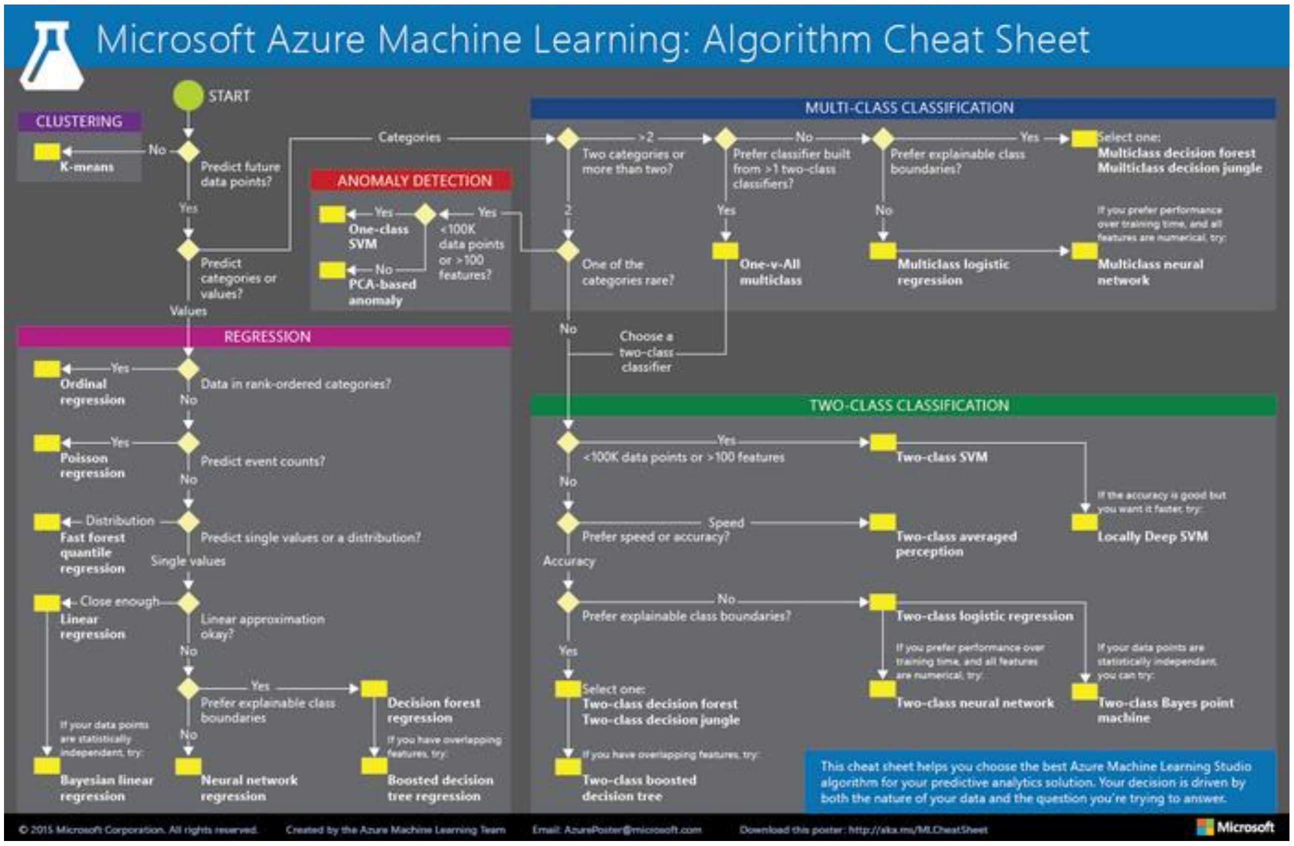

Microsoft Azure Machine Learning: Algorithm Cheat Sheet
START
CLUSTERING
K-means
No
Predict future data points?
Yes
Predict categories or values?
Categories
Values
ANOMALY DETECTION
One-class SVM
Yes
Yes
<100K data points or >100 features?
No
PCA-based anomaly
MULTI-CLASS CLASSIFICATION
>2
Two categories or more than two?
2
One of the categories rare?
No
Prefer classifier built from >1 two-class classifiers?
Yes
One-v-All multiclass
No
Choose a two-class classifier
Yes
Prefer explainable class boundaries?
Select one:
Multiclass decision forest
Muilticlass decision jungle
No
Multiclass logistic regression
If you prefer performance over training time, and all features are numerical, try:
Multiclass neural network
REGRESSION
Yes
Ordinal regression
Data in rank-ordered categories?
No
Yes
Poisson regression
Predict event counts?
No
Distribution
Fast forest quantile regression
Predict single values or a distribution?
Single values
Close enough
Linear regression
Linear approximation okay?
No
If your data points are statistically independent, try:
Bayesian linear regression
Yes
Prefer explainable class boundaries
Decision forest regression
If you have overlapping features, try:
Boosted decision tree regression
No
Neural network regression
TWO-CLASS CLASSIFICATION
Yes
<100K data points or >100 features
Two-class SVM
If the accuracy is good but you want it faster, try:
Locally Deep SVM
No
Speed
Prefer speed or accuracy?
Two-class averaged perception
Accuracy
No
Prefer explainable class boundaries?
Two-class logistic regression
If you prefer performance over training time, and all features are numerical, try:
Two-class neural network
If your data points are statistically independent, you can try:
Two-class Bayes point machine
Yes
Select one:
Two-class decision forest
Two-class decision jungle
If you have overlapping features, try:
Two-class boosted decision tree
This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.
© 2015 Microsoft Corporation. All rights reserved.
Created by the Azure Machine Learning Team
Email: AzurePoster@microsoft.com
Download this poster: http://aka.ms/MLCheatSheet
Microsoft

Figure 13. Machine Learning: Algorithm Cheat Sheet.

Other companies as Google or Facebook are using extensively machine learning algorithms to show the ads to relevant users, predict which one can be next trend, and even as we started to realize in the last years, predict the behavior of the users to detect terrorists [31].

Other applications we can find are:

- Banking and Financial Services: Machine learning can predict customers who are not going to pay credit card bills or paying loans. This is quite important because a good machine learning algorithm would help the bank to identify a customer who can be granted a loan or a special credit card.
- Healthcare: As in our case, a **well trained machine learning algorithm could diagnose a deadly disease based on the symptoms of patients**, comparing it with past data of patients with similar symptoms.
- Retail: we can identify products which sell more frequently, helping the retailers to decide what kind of product has been sold next season or which one has to be retired. Machine learning can help us to identify patterns about which products are likely to be sold together.

2.1.3.1 Machine Learning Algorithms

Machine Learning algorithms are able to read a text, twitter or hear a song and predict if this person is lying, if the tweet is a bot or if this song is going to be the next new hit. The algorithms adaptively improve their performance as the number of samples available for learning increases.

We will train the algorithm always trying to satisfy the requirements of the company, looking for higher performance or accuracy in the predictions. With this purpose, we are going to work with different models, which can be divided into different categories[28]:

- Supervised learning: this technique trains a model on known input and output data, predicting future outputs. Those models are called predictive models and are used for example when a marketing company is trying to find out which customers are likely to churn. The famous weather predictor is an example of this technique. It uses the information accumulated during the last years to train and compare with the new incomes, providing better insights into weather forecasts. We can divide them in two:
  - Classification: Models are going to separate the categorical data in different classes. predicting categorical responses. For example, whether an email is genuine or spam, whether a tumor is cancerous or benign, or in our case if a patient has a disease or not. This TFG analyzes a binary classification problem, where the target is going to be classified in two classes. The Figure 14 shows how we separate a dataset in different classes (represented by blue, yellow and red color) [32]. The classification algorithm makes the discrimination we can see with the green lines.

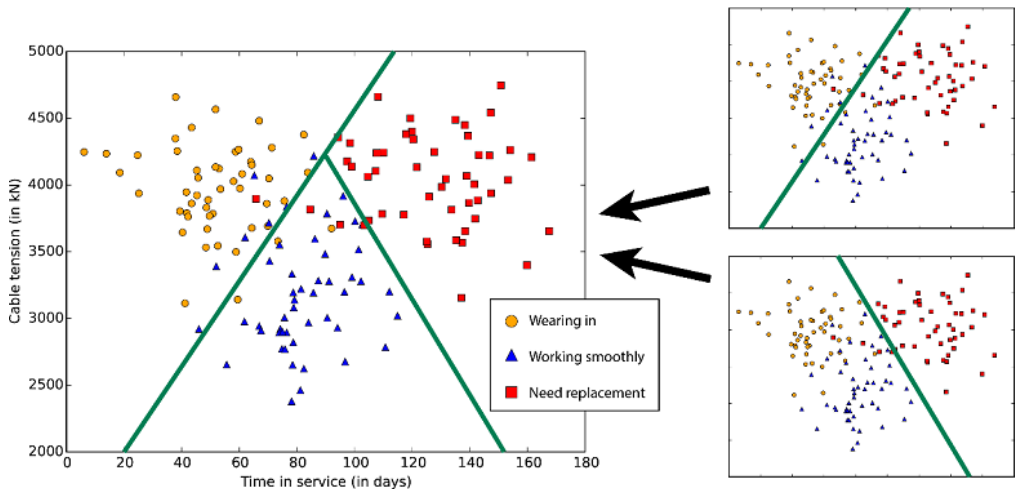


Figure 14. Classifying elements in 3 classes.

It is performed by classification algorithms like:

- Linear Support Vector Machines

- Naïve Bayes
- Random Forest Classifier
- Decision trees classifier and Extra tree classifier
- Neural Network
- Logistic regression
- Gradient Boosting Classifier
- K-nearest Neighbors.

Some typical applications include medical imaging, image, speech recognition, and credit scoring.

- Regression: Predicts continuous responses, for example, the fluctuation in the weather or the values in the stock market. In this project, we are going to focus on the classification techniques. In figure 15 [33] we can appreciate a scheme showing different algorithms for each group.

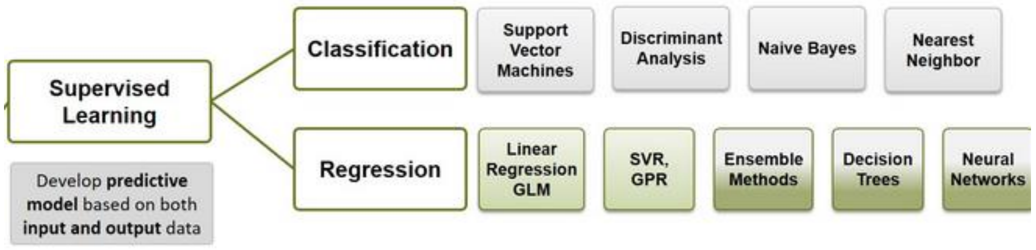


Figure 15. Supervised Learning algorithms classification.

- Unsupervised Learning: It is focused in Raw data, unstructured data where we are not able to identify the different classes. Its main task is to convert this unstructured data in structured data through hidden patterns or intrinsic structures in data. Some popular algorithms can be:
  - Linear Regression

- Logistic Regression
- Polynomial Regression
- Stepwise Regression
- Ridge Regression
- Lasso Regression
- ElasticNet Regression

Clustering is the most common unsupervised learning technique. This technique is able to find patterns in the Raw data and group the different elements, creating labels to structure the data. In figure 16[34] we can appreciate the different classes discovered in our data with blue, red and green color.

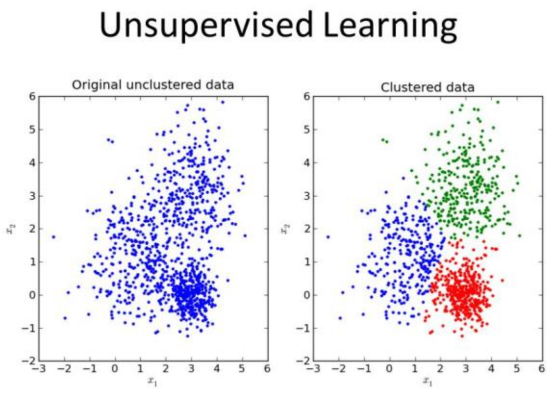


Figure 16. Original vs Clustered dataset using unsupervised learning algorithm.

Nowadays there is a huge amount of raw data in every field, for example when a retailer needs to find out the combinations of products that the costumers tend to buy more frequent. In the pharmaceutical industry, unsupervised learning may be used to predict which diseases are likely to occur along with another disease. Furthermore, in Amazon, we can find suggestions about products in combination with other ones.

An example of an algorithm used here is: K- means Clustering Algorithm, like the figure 17[35].

It is an interesting field that could be used to continue this Research, using clustering to find which diseases are likely to occur with the one selected.

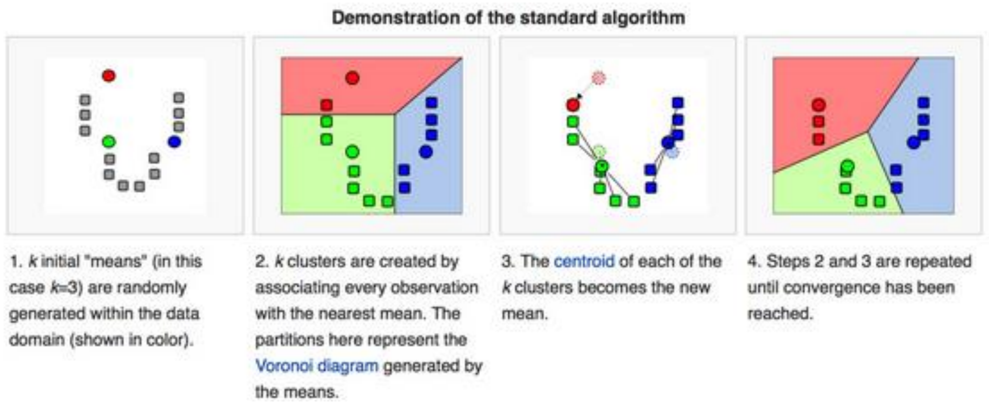

Demonstration of the standard algorithm
1. k initial "means" (in this case k=3) are randomly generated within the data domain (shown in color).
2. k clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.
3. The centroid of each of the k clusters becomes the new mean.
4. Steps 2 and 3 are repeated until convergence has been reached.

Figure 17. K means clustering Algorithm, explaining the steps to perform the algorithm

There are some steps we can follow to perform a machine learning task [36]. We are going to mention them here but we will go deeper in the section block 3. The steps used are the next:

- <u>Collecting data</u>: The better variety, density, and volume of data, better the learning prospects from the algorithms.
- <u>Preparing the data</u>: The quality of the data is particularly important for any analytical process. Therefore, one needs to spend time taking steps for fixing issues such as missing data and treatment of outliers.
- <u>Training a model</u>: In this step, we take the appropriate algorithm and the data we are interested in, we split the preprocessed data in two parts, and we will train our algorithm with the training part. The test part is used only as a reference for our predictions.

- Evaluating the model: The accuracy of the algorithm is going to be evaluated with the test part of the data. We will determine the precision based on the outcomes. It is important that this part of the data is not seen before by the algorithm until the moment we predict to prevent overfitting.
- Improving the performance: This step checks how other parameters work in our algorithm, or even choosing new algorithms to compare the performance and augment the efficiency. That's the reason we spent a lot of time in data collection and preparation.

2.1.4 Data Mining

It refers to the science or process of collecting and analyzing data, with the purpose of discovering patterns or properties with the use of machine learning algorithms, statistics, and mathematics. Those patterns can be groups of data records, unusual records in our dataset, or as in our case, dependencies between the different records. It converts messy data in useful information, unlocking for the company various insights.

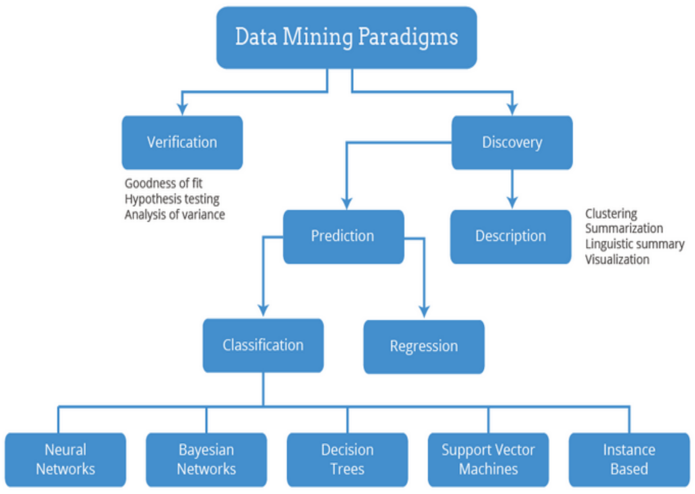


Figure 18. Data mining paradigms

Data mining is closely related to data analysis and it involves fields like database and data management, data pre-processing, modeling, complexity considerations, post-processing of discovered structures, visualization, etc. With data mining, we are extracting information from large datasets and not extracting data as the name suggests. We can see the Data mining paradigms in Figure 18 [37].

Figure 19. Data mining

Prepare the data or collect the data, such as other steps, are not part of the data mining process, but they belong to the data mining CRISP-DM process. In our research, we are using the methodology Cross Industry Standard Process for Data Mining (commonly known by its acronym CRISP-DM) to obtain, prepare, analyze and report the data.

2.1.4.1 Feature Engineering, Importance and Features

As we said, we are going to focus our attention on **Feature Engineering**. A very good definition would be that feature engineering is the process to create Features that makes machine algorithms work. It is quite important and will determine the success of our project, without matter your skills in statistical or computer techniques. A quotation I like is the next one:

"At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used. If you have many independent features that each correlate well with the class, learning is easy. On the other hand, if the class is a very complex function of the features, you may not be able to learn it. Often, the raw data is not in a form that is amenable to learning, but you can construct features from it. This is typically where most of the effort in a machine learning project goes." [38]

It is the core of a data mining project, and comes to life in the **Data Preparation phase**, covering the phases of selecting the data, cleaning the data, constructing the data and attribute selection. This phase can be performed several times taking in some cases almost 70% of the time in a data mining project.

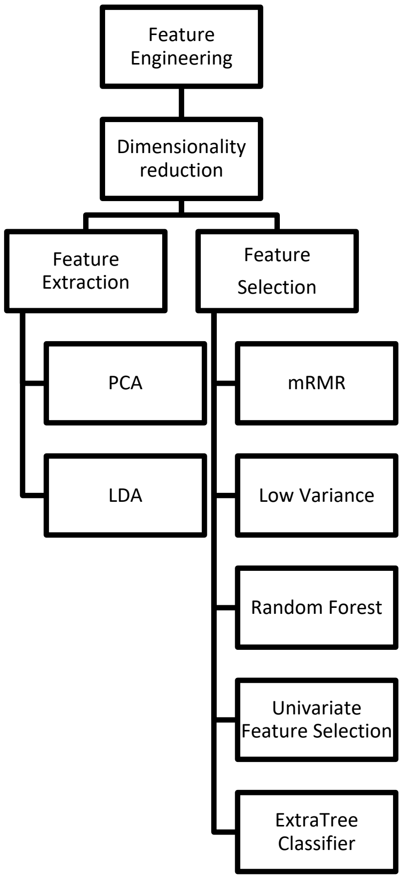


Figure 20. Feature Engineering diagram

The features will directly influence the obtained results, so the better we choose and prepare the features, the results will be more accurate and effective. Those features will be prepared with different dimensionality reduction algorithms. We can appreciate some of these algorithms in Figure 20.

The results obtained will depend on many factors, among which we have the metrics used to evaluate our results, the features used, and how we have prepared them. Therefore, we are going to need quality features, which correctly describe the inherent structures of our dataset.

Having quality features will give us flexibility, allowing us to even choose the wrong models and still get decent results. By having more flexibility, we will be allowed to use easier models, which allow us to run faster, being more understandable and maintainable. We will not even need to tune our models with the best parameters, but we will still get good results.

A **feature** is an attribute that we use for modeling. A table is to be composed of columns and rows, where the rows are usually the observations of our study, and the columns are the features. A feature is a part of the attributes of an observation that will have a meaning for our objectives. They will be an essential part of solving our problem

An image is an observation in computer vision, but a feature could be a line in the image. In twitter, a tweet can be an observation, but maybe a phrase or word can be a feature. When we use Siri von Apple, an utterance can be an observation, but maybe a phoneme is a feature.

- The process of Feature Engineering:

  We must know where feature engineering fits into the machine learning process. It is a process that will be repeated until the moment we decide that the result obtained solves the problem we raised in the beginning. The process could be as follows:

1. Brainstorm features
2. Devise features
3. Select features
4. Evaluate models

We need to have a well-defined problem to know when do we have to stop this process and try other models, configurations, etc. We can find some examples of feature Engineering in this reference. [39]

2.2 Dimensionality Reduction and Modelling Algorithms selected for this research

In the Present work, we are interested in evaluating the combination of the different dimensionality reduction algorithms with different classification algorithms approaches, such as Logistic Regression or Random Forest, which are known to be well adapted to the classification problems.

First, we are going to show which ones are the different Dimensionality Reduction Algorithms we have chosen and the Modelling algorithms used for our predictions, giving a small introduction about Dimensionality Reduction and Machine learning algorithms, focusing on the dimensionality reduction algorithms.

2.2.1 Dimensionality Reduction

Suppose our system is going to be only effective working with a dataset with a small number of components. Usually, in Data Science, we are going to have big datasets with thousands and even millions of features. The problem of dimensionality reduction appears when the data have big dimensions, and it is intractable with our infrastructure, as we explain in Figure 19[40]. Therefore, several times would be recommended to reduce our dataset into a manageable size, keeping as much original information as possible.

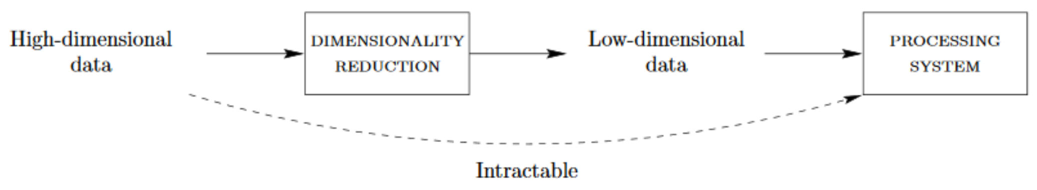


Figure 19. Dimensionality reduction Problem.

With dimensionality reduction, we are looking for a model the one is going to have only the most important features. It has several benefits:

- We are going to make our model easier to be interpreted
- We are going to reduce the variance of our model, and it means preventing overfitting.
- We are going to reduce the computational time and reduce costs of training a model.

When we speak about Dimensionality Reduction techniques, usually we speak about Feature Selection and Feature Extraction.

2.2.1.1 Feature Selection Algorithms

We are going to perform four different feature selection algorithms. They are going to select a subset of relevant features from the original dataset and create a new one with the selected features. We are going to delete redundant features trying to keep as much information as possible from our original dataset[41]. Those algorithms are going to be quite effective when we are trying to understand the data and it is easier to implement them in compare with feature extraction.

I would like to mention that we are going to have three different types of feature selection algorithms:

- Filter Methods: Filter methods are going to use a proxy measure to score a feature. This measure is going to be computed faster but is not going to be so effective as the wrapper method. Usually, filter methods are used as a preprocessing for wrapper methods because they don’t have a high computational complexity.

- Embedded Methods: This method is going to combine the advantages of filter and wrapper methods. The most famous algorithm is going to be LASSO method for constructing a linear model, penalizing the regression coefficients with an L1 penalty and decreasing some of them to 0
- Wrapper Methods: they are going to score feature subsets, and each new subset is going to be used for training the model. The test is going to be performed on a hold-out set, the one is going to obtain an error, and depending on this error this subset obtains a score. It is going to need high computational complexity.

2.2.1.1.1 Random Forest

Random Forests are often used for feature selection [42]in a data science project and the reason is the tree-based strategies performed by random forests. Random Forests are a collection of decision trees classifiers which are fitted on various subsamples of the dataset and use averaging to improve the prediction accuracy. A Decision Tree is basically a set of decisions on whether or not classify something. We will go deeper later in this algorithm but now we will focus in Random Forest.

And how it works? Random Forest ranks how well the nodes improves the purity[43]. It means, that the Nodes with the greatest decrease in impurity happen at the beginning and the nodes with the least decrease in impurity at the end (Gini impurity). The outcome of the Random Forest can be visualized by the “Gini Importance” and be used as an indicator of relevance between the features in the dataset. With this feature importance, we can create a ranking of the features and if we prude the tree below a particular node, we will keep a determinate number of features creating a new subset of the most important features.

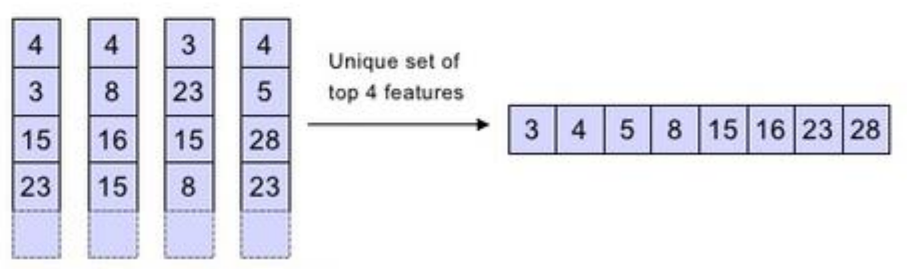


Figure 20. Random Forest Selection of top features

- Implementation in Python

  This algorithm is already implemented in scikit-learn and we can import it directly into our project. Below we have the class RandomForestClassifier and the parameters by default[44].

  class **`sklearn.ensemble.RandomForestClassifier`(***n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*)**

  We have a certain number of attributes predefined like:
  - *feature_importances_*
  - *estimators_*
  - *classes_*
  - …

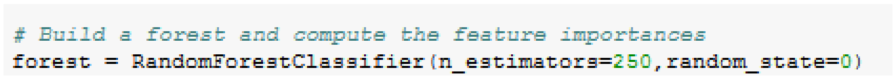
  and methods like:
  - *apply(X)*, to apply trees in the forest to X

- *fit(X, y[, sample_weight]),* to build a forest of trees from the training set (x,y)
- *predict(X)*, to predict class for X.
- …

For more information on the parameters, attributes, and functions, you can find a complete guide on the scikit-learn page.

- Process

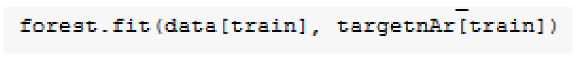
To apply our feature selection algorithm we have to follow some steps:
First, we are going to create a random Forest Classifier tuning the algorithm with the parameters we consider necessary as we can see in figure 21.



```
# Build a forest and compute the feature importances
forest = RandomForestClassifier(n_estimators=250,random_state=0)
```

Figure 21. Creation of a RandomForestClassifier

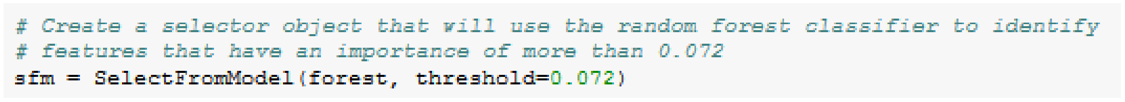
We are going to train the classifier on the training data (we have to use only the training dataset if we want to avoid overfitting). This process is shown in Figure 22.



```
forest.fit(data[train], targetnAr[train])
```

Figure 22. Training of RandomForestClassifier

We will create a selector object to identify features with an importance of more than a certain threshold. We calculated that the value which is going to give us the predetermined is 0.072.



```
# Create a selector object that will use the random forest classifier to identify
# features that have an importance of more than 0.072
sfm = SelectFromModel(forest, threshold=0.072)
```

Figure 23. Apply a threshold to RandomForestClassifier

In figure 24 we can see how we will train the selector again only with the training dataset.

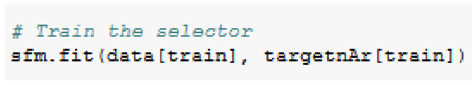

```
# Train the selector
sfm.fit(data[train], targetnAr[train])
```

Figure 24. Training of the selector

Finally, in figure 25 we can appreciate how we create a new dataset containing only the most important features with the function transform on the training part and the test part of the dataset.

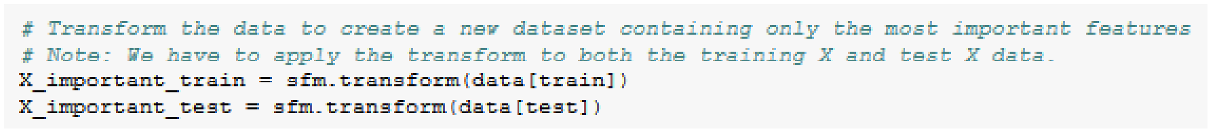

```
# Transform the data to create a new dataset containing only the most important features
# Note: We have to apply the transform to both the training X and test X data.
X_important_train = sfm.transform(data[train])
X_important_test = sfm.transform(data[test])
```

Figure 25. Creation of new dataset with RandomForestClassifier

2.2.1.1.2 Low Variance

Low variance is a basic feature selection technique. The main idea is the next, considering a feature like a constant, only the features with a specific variance are going to be used. For example, if we want only the features with a variance higher than 0.6, we will use a threshold of 0.6 to delete the features with a lower variance from our new subspace. Therefore, a heuristic approach is first removing all features whose variance is below the threshold

By default, it is going to remove all zero-valance features, i.e. features that have the same value in all samples like all 5 for example or the same word.

- Implementation in Python

  This algorithm can be implemented in Python with the use of the function **VarianceThreshold**[45]**.** This function removes all low-variance features, looking only the X features and not the desired outputs. Below we have the class VarianceThreshold and the parameters by default.

class `sklearn.feature_selection.VarianceThreshold`(*threshold=0.0*)

We have as predefined attributes *variances_*

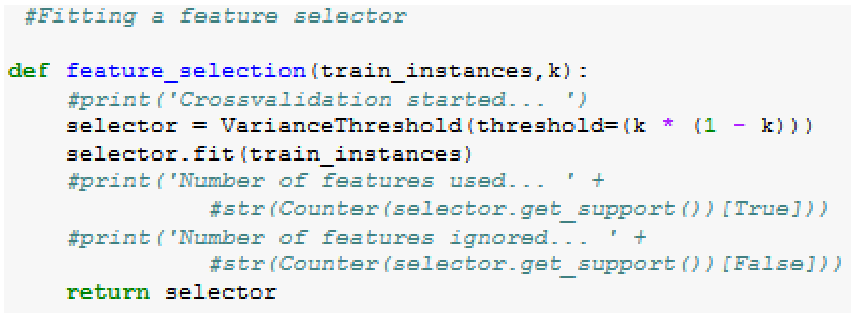
and methods like:

- ○ *fit(X[, y]), to learn empirical variances from X*
- ○ *fit_transform (X[, y]), to fit the data, and then transform it.*
- ○ *get_parameters([deep]), to get parameters for this estimator.*
- ○ ...

For more information on the parameters, attributes, and functions, you can find a complete guide on the scikit-learn page.

- Process

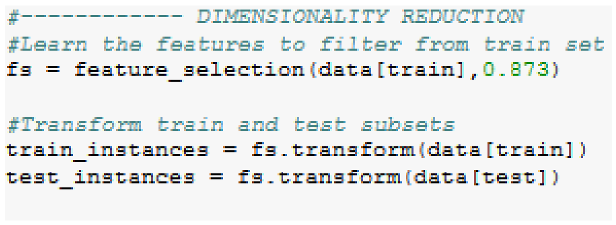
There are several ways of implementing this algorithm, but we suggest the next one:



```
#Fitting a feature selector

def feature_selection(train_instances,k):
    #print('Crossvalidation started... ')
    selector = VarianceThreshold(threshold=(k * (1 - k)))
    selector.fit(train_instances)
    #print('Number of features used... ' +
              #str(Counter(selector.get_support())[True]))
    #print('Number of features ignored... ' +
              #str(Counter(selector.get_support())[False]))
    return selector
```

Figure 26. Definition of feature selection function for Low Variance

As we can see in Figure 26, we are going to define a function, the one is going to receive the train_instances, the X part of our dataset and the K value, which is the threshold selected to be evaluated. After training our algorithm we are going to send it back, prepared to transform the data in a new subspace. As we can see in Figure 27, on one side we are going to transform the training dataset, and separated we are going to transform the testing part.



```
#------------ DIMENSIONALITY REDUCTION
#Learn the features to filter from train set
fs = feature_selection(data[train],0.873)

#Transform train and test subsets
train_instances = fs.transform(data[train])
test_instances = fs.transform(data[test])
```

Figure 27. Reduction of dimensions Low Variance

2.2.1.1.3 mRMR

The mRMR (minimum Redundancy Maximum Relevance) feature selection algorithm seeks and select features with a high correlation with the output class, which is called maximum relevance selection, and at the same time, these values may contain a low correlation between them, looking for values to be as far as possible. This is the scheme that follows our mRMR algorithm. Some subsets contain material that is relevant but redundant at the same time. mRMR attempts to solve this problem by eliminating those values.

This algorithm is used in methods to accurately identify genes and phenotypes but is also commonly used in cancer detection or speech recognition

- Implementation in Python

This algorithm can be implemented in Python through the use of the skfeature library, the one we can find in scikit-feature. We can install it following the steps in the reference attached here[46].

We are going to receive as input parameters:

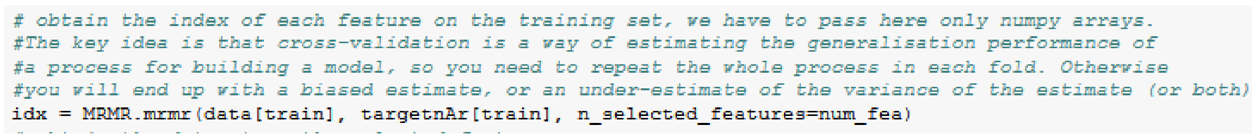
- *X: {numpy array}, shape (n_samples, n_features) input data, guaranteed to be dis crete*
- *y: {numpy array}, shape (n_samples,) input class labels*
- *kwargs: {dictionary} n_selected_features: {int} number of features to select*
- *We are going to give as output parameters:*

- *X: {numpy array}, shape (n_samples, n_features) input data, guaranteed to be dis crete*
- *y: {numpy array}, shape (n_samples,) input class labels*
- *kwargs: {dictionary} n_selected_features: {int}*

  number of features to select

- Process

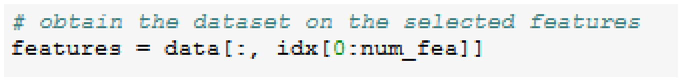
The process to apply mRMR is quite simple and it was followed the examples given by the developers[47].



```
# obtain the index of each feature on the training set, we have to pass here only numpy arrays.
#The key idea is that cross-validation is a way of estimating the generalisation performance of
#a process for building a model, so you need to repeat the whole process in each fold. Otherwise
#you will end up with a biased estimate, or an under-estimate of the variance of the estimate (or both)
idx = MRMR.mrmr(data[train], targetnAr[train], n_selected_features=num_fea)
```

Figure 28. Collecting indexes with mRMR

As we can appreciate in Figure 28, we have to pass only numpy arrays, therefore at the beginning, the data was transformed. As input, we are going to give the training part of our numpy arrays and the number of features we are interesting to consider.



```
# obtain the dataset on the selected features
features = data[:, idx[0:num_fea]]
```

Figure 29. New dataset with selected features

In figure 29 we can appreciate how the number of features is reduced, using the index obtained before to select only those features from our dataset.

2.2.1.1.4 Univariate Feature Selection

This algorithm is going to examine each feature individually to understand which ones have the strongest relationship with the output variable. This method is simple to run and understand, giving us a good idea how to understand the data, but probably we won't obtain the best optimization in our dataset.

There are several options to implement Univariate feature selection, but the one to use is the provided by sci-kit learn, the SelectKBest classs[48]. This class is going to remove all but the k highest scoring features. This algorithm can be used with different statistical tests to select a specific number of features, like for regression f_regression, mutual_info_regression and for classification chi2 or f_classif.

- Implementation in Python

  This algorithm can be implemented in Python with the use of the function **SelectKBest,** from sci-kit learn. It selects features according to the k highest, ignoring the lower ones. Below we have the class SelectKBest and the parameters by default.

  Class `sklearn.feature_selection.SelectKBest` (*score_func=<function f_classif>, k=10*)

  We have a certain number of attributes predefined like:

  - *scores_*
  - *pvalues_*
  - ...

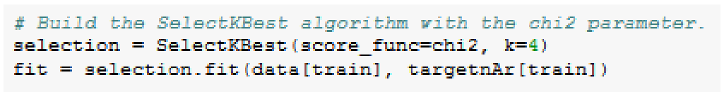
  and methods like:

  - *fit(X, y), it runs score function on (x,y) and gets the appropriate features.*
  - *fit_transform (X[, y]), to fit the data, and then transform it.*
  - *get_params([deep]), to get parameters for this estimator.*
  - ...

  For more information of the parameters, attributes, and functions, you can find a complete guide on the scikit-learn page.

- Process

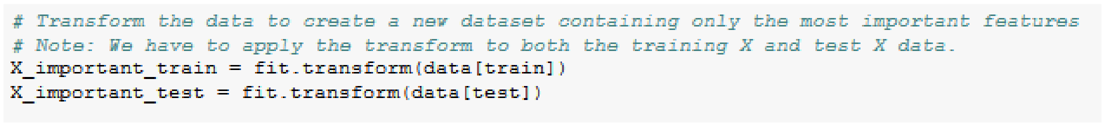
  The process to apply Univariate feature selection is quite simple and we are going to use the example given in the website of the library.



```
# Build the SelectKBest algorithm with the chi2 parameter.
selection = SelectKBest(score_func=chi2, k=4)
fit = selection.fit(data[train], targetnAr[train])
```

Figure 30. Creating and modeling SelectKBest algorithms

As we can appreciate in Figure 30, we are using the core_func chi2, it is a statistical test for non-negative features. We are giving the number of features we are interested in and then we train the algorithm with the training part of our dataset, giving the features and the target to the model.



```
# Transform the data to create a new dataset containing only the most important features
# Note: We have to apply the transform to both the training X and test X data.
X_important_train = fit.transform(data[train])
X_important_test = fit.transform(data[test])
```

Figure 31. Transforming data

In figure 31 we can see how we transform the data in the new dataset, transforming both parts, the train part and the test part.

2.2.1.2 Feature Extraction Algorithms

We are going to focus our attention on the Feature Extraction algorithms, which are a little bit more complex in comparison with feature selection, but at the same time are the proper ones if we are trying to discriminate values in a dataset with the intention of classifying.

Feature Extraction involves reduceing the number of amount of data used to describe a large dataset, which is one of the major problems performing analysis of complex data. Usually, as much bigger is the dimension of my dataset, the computation complexity and amount of memory increase. With Feature Extraction, we build combinations of the variables, solving the dimensionality problem, and still describing the data with sufficient accuracy.

2.2.1.2.1 PCA

Principal component analysis is a method of extracting important variables from a data set, identifying the different patterns in data. PCA aims to find the correlation between the variables in the data, searching the principal components of the dataset and transforming the data into a lower-dimensional subspace.

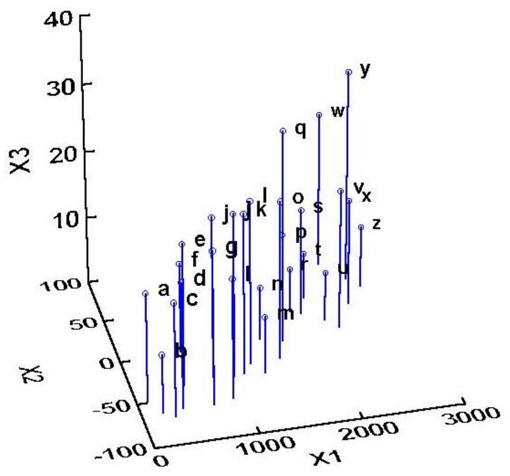


Figure 32. Some values in a 3-dimension graph. Blue lines represent distance of X3 axis.

Generally speaking, we are going to search the bigger variance between our points. We try to project our data on a simple line, visualizing it like the shadow of the points. As we can see in the figure 32, when we don’t cover properly the area, we are not going to maximize the variance on the projected dimension, therefore we have to look for the maximum variance and at the same time minimize the mean squared distance between the data and the projections, resulting in a line which is closest to the data.

We would obtain like this what is called the **first principal component**, reducing everything in one dimension capturing the largest variance of the data. No other principal component can have a higher variability than the first principal component.

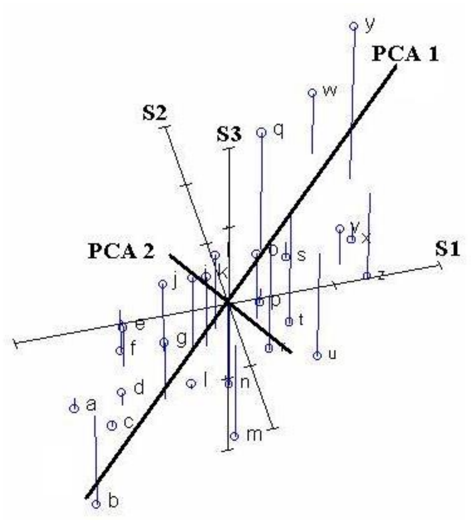


Figure 33. The same data is standardized and centered. We can appreciate the Principal components PCA 1 and PCA 2. The axes S1, S2 and S3 represent the new axes.

The second principal component is also a linear combination of the predictors or points in our graph, taking the remaining bigger variance in our dataset, having a correlation between the first and the second principal component of 0. If the correlation is 0, then our PC have to be orthogonal.

We can appreciate in the figures 32, 33 and 34 the process in a graphical way. It can help us to understand for example how do we convert a 3d dimensional problem in a 2-dimensional problem, easier to understand as we can see in figure 38. The final axes are going to be represented by the PCA axes [49].

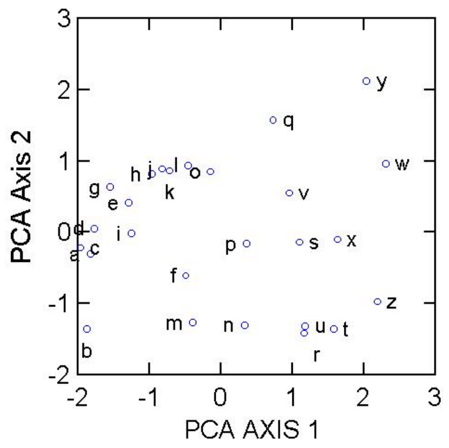


Figure 34. Data in the new subspace represented by the PCA axes.

All succeeding PCs follows a similar concept, capturing the remaining variance in the dataset. But, how many PC do we need? Most of the variance in your dataset would be explained by a small number of components, usually in 10% of the data can we find 95% of the variance.

If we speak about mathematics, we can understand the PCs as the eigenvector of the covariance matrix. As the covariance is symmetric, the eigenvectors are orthogonal, as well as our PCs.

- Implementation in Python

  This algorithm is already implemented in scikit-learn and we can use import it directly into our project. Below we have the class PCA implementation and the parameters by default [50].

  class **`sklearn.decomposition.PCA`** (*n_components=None, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', random_state=None*)

  We have a certain number of attributes predefined like:

  - *components_*
  - *explained_variance_*
  - *mean_*
  - …

  and methods like:

  - *Fit_transform(X[, y*]), fit the model with X and apply the dimensionality reduction on X
  - *fit(X [, y]]), fit the model with X.*
  - *transform(X), apply dimensionality reduction with* X.

- ○ …

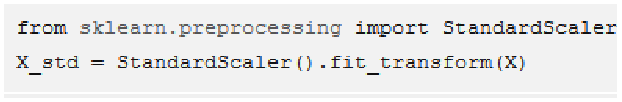
For more information on the parameters, attributes, and functions, you can find a complete guide on the scikit-learn page.

- Process

The first thing we are going to do is preprocess the data. With this purpose, we have to realize that if we want a proper performance of PCA, we are going to need standardized data. So, we need to apply normalization on the dataset, and we are going to explain what does it mean.

Usually, we are going to have features with different scales in a dataset. For example, centimeters, meters, kilometers, kilograms, and grams, etc. If we perform PCA on an un-normalized dataset, we will always depend on the feature with higher variance, and this is undesirable[51].

In our case wouldn't be necessary to perform normalization because we have already the dataset normalized and all the features have the same measure, but in the case, would be necessary the code shown in Figure 35:



```
from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(X)
```

Figure 35. Scaling data for PCA

We are going to compute it step by step, but at the end, we will give also the implementation from sci-kit learn in python.

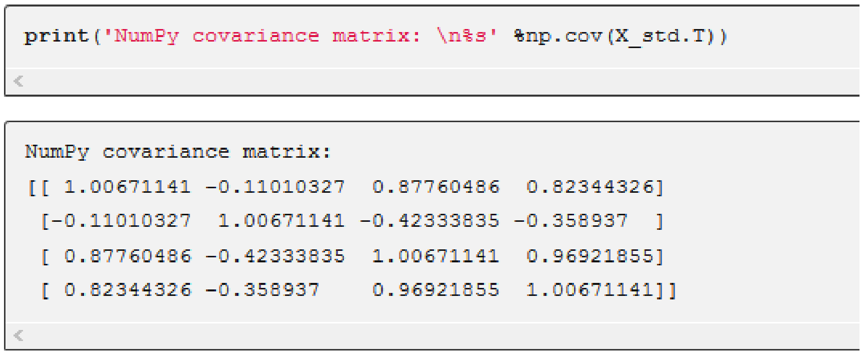
We are going to obtain first the eigenvectors and eigenvalues, the ones are going to represent the PCs in our dataset, and those values will be obtained from the covariance matrix, the core of a PCA. Therefore, the eigenvectors represent the directions of the new feature space, and the eigenvalues the magnitude (the variance along the eigenvector).

1. **Covariance Matrix:** We are going to perform eigen decomposition on our covariance matrix, where each element represents the covariance between two features. This formula in figure 36 represents the covariance between two features:

$$\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} \left(x_{ij} - \bar{x}_j\right)\left(x_{ik} - \bar{x}_k\right).$$

Figure 36. Covariance matrix algorithm

In figure 37, we can see how we estimate the covariance matrix, using "np.cov" and giving to this function the data we want to process.

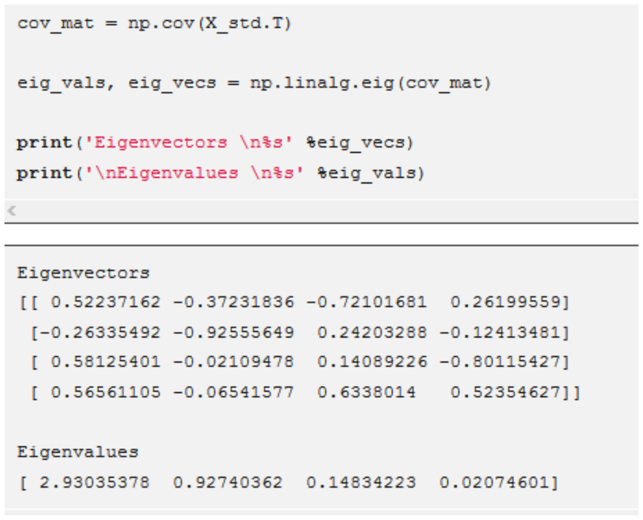


```
print('NumPy covariance matrix: \n%s' %np.cov(X_std.T))
```

```
NumPy covariance matrix:
[[ 1.00671141 -0.11010327  0.87760486  0.82344326]
 [-0.11010327  1.00671141 -0.42333835 -0.358937  ]
 [ 0.87760486 -0.42333835  1.00671141  0.96921855]
 [ 0.82344326 -0.358937    0.96921855  1.00671141]]
```

Figure 37. Printing Covariance matrix

2. **Eigenvectors and Eigenvalues:** We are going to perform an eigendecomposition on the covariance matrix:

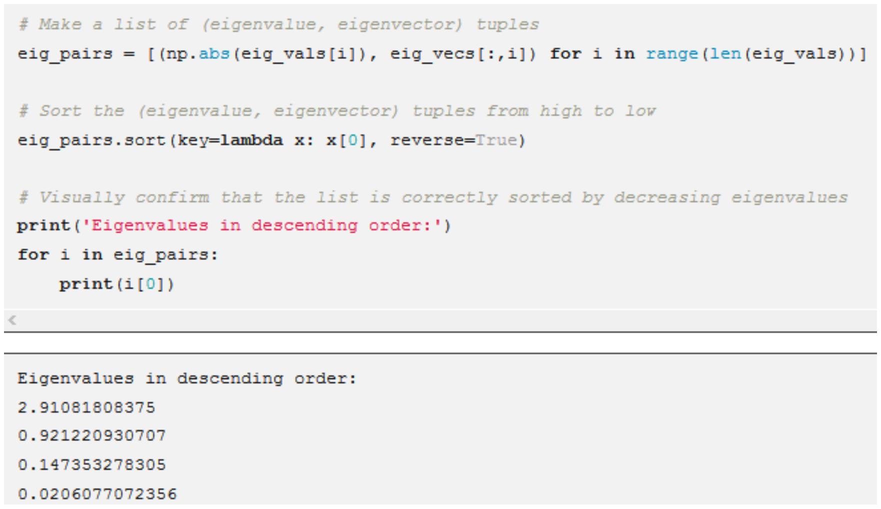


```
cov_mat = np.cov(X_std.T)

eig_vals, eig_vecs = np.linalg.eig(cov_mat)

print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

```
Eigenvectors
[[ 0.52237162 -0.37231836 -0.72101681  0.26199559]
 [-0.26335492 -0.92555649  0.24203288 -0.12413481]
 [ 0.58125401 -0.02109478  0.14089226 -0.80115427]
 [ 0.56561105 -0.06541577  0.6338014   0.52354627]]

Eigenvalues
[ 2.93035378  0.92740362  0.14834223  0.02074601]
```

Figure 38. Printing Eigenvectors and Eigenvalues

Obtaining the eigenvectors and eigenvalues as we can see in figure 38. In order to decide which eigenvector(s) can be dropped, we are going to rank the eigenvalues from highest to lowest and choose the highest ones. This process is shown in figure 39.

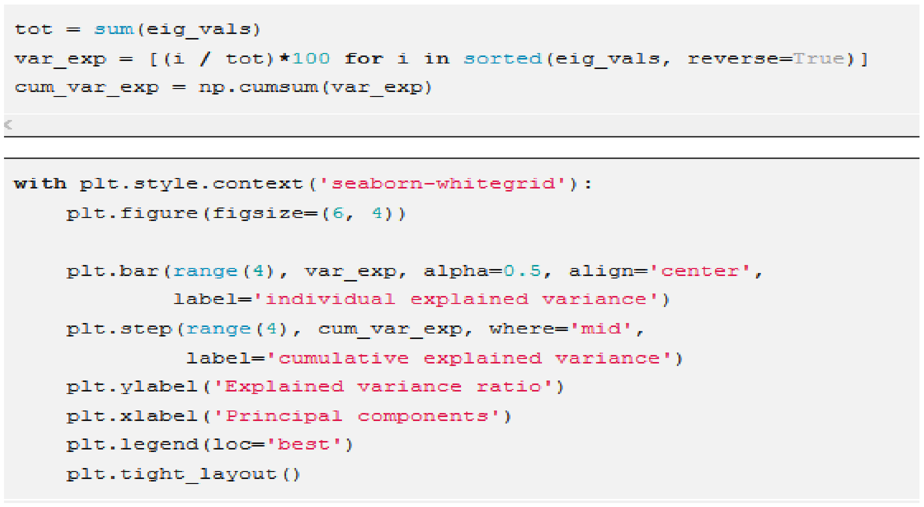


```
# Make a list of (eigenvalue, eigenvector) tuples
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the (eigenvalue, eigenvector) tuples from high to low
eig_pairs.sort(key=lambda x: x[0], reverse=True)

# Visually confirm that the list is correctly sorted by decreasing eigenvalues
print('Eigenvalues in descending order:')
for i in eig_pairs:
    print(i[0])
```

```
Eigenvalues in descending order:
2.91081808375
0.921220930707
0.147353278305
0.0206077072356
```

Figure 39. Printing eigenvalues in order

3. The question is now, how do we know if the correct number of eigenvalues we decided to use is the correct one? how many dimensions do we want without losing so much information? I would recommend obtaining the explained variance, the one is going to indicate us how much information can be attributed to each PC. We can see how I obtained it in figure 40.



```
tot = sum(eig_vals)
var_exp = [(i / tot)*100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
```

```
with plt.style.context('seaborn-whitegrid'):
    plt.figure(figsize=(6, 4))

    plt.bar(range(4), var_exp, alpha=0.5, align='center',
            label='individual explained variance')
    plt.step(range(4), cum_var_exp, where='mid',
             label='cumulative explained variance')
    plt.ylabel('Explained variance ratio')
    plt.xlabel('Principal components')
    plt.legend(loc='best')
    plt.tight_layout()
```

Figure 40. Calculating the explained variance

We are going to use accumulative sum [52] of the explained variance with the purpose of representing it in a graph making easier the comprehension. and the graph where we are showing the variance, then it is our decision how many information we want to keep in our new data-set. In figure 41 we can appreciate the principal components and the cumulative explained variance, obtaining with the first two principal component around a 95 percent information.

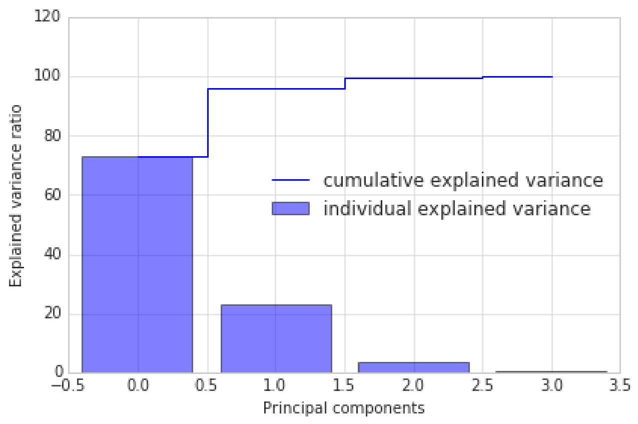
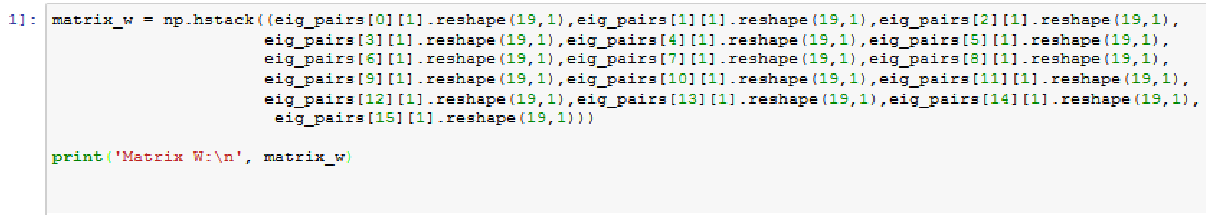


Figure 41. Graph showing the explained Variance

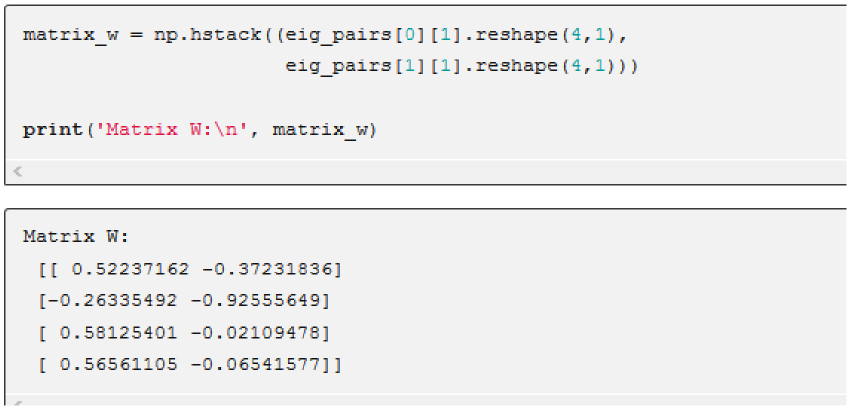
4. Now we are going to reduce the N-dimensional feature space to a new m-dimensional feature space. For this purpose, we will choose the top eigenvectors with the highest eigenvalues to construct our d x k-dimensional eigenvector matrix. For example, here we are going to see a code in Figure 42 where we implemented to understand how to reduce the dimension from 19 until 15.



```
1]: matrix_w = np.hstack((eig_pairs[0][1].reshape(19,1),eig_pairs[1][1].reshape(19,1),eig_pairs[2][1].reshape(19,1),
                         eig_pairs[3][1].reshape(19,1),eig_pairs[4][1].reshape(19,1),eig_pairs[5][1].reshape(19,1),
                         eig_pairs[6][1].reshape(19,1),eig_pairs[7][1].reshape(19,1),eig_pairs[8][1].reshape(19,1),
                         eig_pairs[9][1].reshape(19,1),eig_pairs[10][1].reshape(19,1),eig_pairs[11][1].reshape(19,1),
                         eig_pairs[12][1].reshape(19,1),eig_pairs[13][1].reshape(19,1),eig_pairs[14][1].reshape(19,1),
                          eig_pairs[15][1].reshape(19,1)))

    print('Matrix W:\n', matrix_w)
```

Figure 42. Creation of the matrix used to reduce the dimensions.

The matrix obtained is quite big, and we wouldn't be able to show it here, therefore in figure 43 is anoother example of projection with less dimension, where we can appreciate the new matrix.

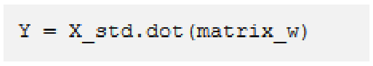


```
matrix_w = np.hstack((eig_pairs[0][1].reshape(4,1),
                      eig_pairs[1][1].reshape(4,1)))

print('Matrix W:\n', matrix_w)
```

```
Matrix W:
 [[ 0.52237162 -0.37231836]
 [-0.26335492 -0.92555649]
 [ 0.58125401 -0.02109478]
 [ 0.56561105 -0.06541577]]
```

Figure 43. Printing matrix with four dimensions

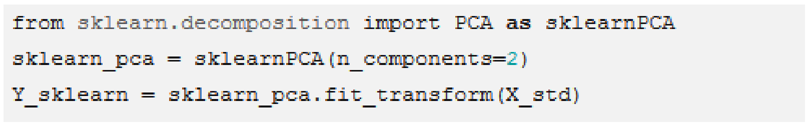
5. Finally, we are going to project the matrix onto the new Feature Space obtaining a new dataset with reduced dimensions, as we can see in figure 44.



```
Y = X_std.dot(matrix_w)
```

Figure 44. Creating the new reduced dataset

I think is interesting to explain the procedure of PCA, understand how this algorithm works and give an easy example of the implementation, everything for educational purposes. But we have an implementation from sci-kit learn also, the one we are going to see in the figure 45:



```
from sklearn.decomposition import PCA as sklearnPCA
sklearn_pca = sklearnPCA(n_components=2)
Y_sklearn = sklearn_pca.fit_transform(X_std)
```

Figure 45. PCA using the algorithm from the scikit library.

As we can appreciate, the number of components of the new subspace is given as a parameter, and we are going to transform it directly applying on the standardized dataset. I'm going to add here two references I found quite interesting and show the process of PCA in a simple and readable way[53], [54], [55]

2.2.1.2.2 LDA

When we speak about **Linear Discriminant Analysis**, we speak usually about dimensionality Reduction applied to use pattern-classification and machine learning algorithms. The idea is to transform our dataset onto a lower-dimensional space, applying class separability in order to avoid overfitting and at the same time reduce the computational costs.

The general LDA approach is quite similar in comparison to PCA, but in this case, we are looking for the axes that maximize the separation between the classes[56]. It is going to prevent us to avoid overfitting by minimizing the error in parameter estimation, and not only reducing computational costs.

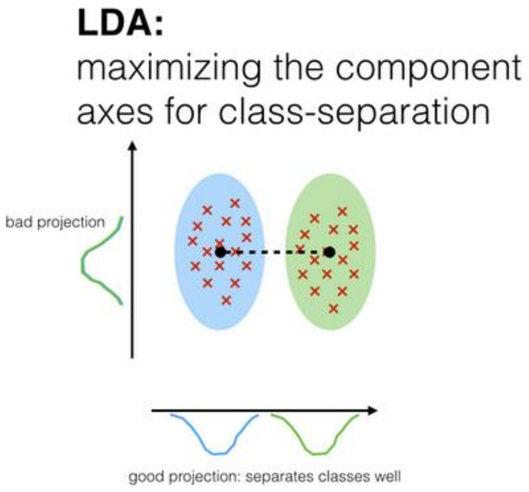

LDA:
maximizing the component axes for class-separation
bad projection
good projection: separates classes well

Figure 46. LDA algorithm separating classes

A small difference between PCA and LDA is that with PCA we could say that is unsupervised, because we don't care about the class label, and in LDA we compute the axes that maximize the separation between multiple classes. We can consider that LDA tends to outperform PCA, but it is not always like this. In practice, it is interesting to test both in combination, PCA followed by LDA for dimensionality reduction.

We are going to make a small summarize of the LDA approach:

1 Compute the d-dimensional mean vectors for the different classes.

2 Compute the scatter matrices (in-between-class and within-class scatter matrix)

3 Compute eigenvectors and eigenvalues.

4 Sort eigenvectors by decreasing eigenvalues and choose k eigenvector with the largest eigenvalues.
5 Use the eigenvector matrix to transform the dataset onto a new subspace

LDA algorithm works like this: it is going to search a linear combination of variables that best separates two classes as we can see in figure 46.

- Implementation in Python

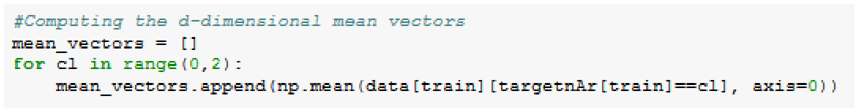
  This algorithm is already implemented in scikit-learn, but the version we have is going to give us only one dimension. This version fits a Gaussian density to each class, assuming that all classes share the same covariance matrix. We are not going to be able to use this algorithm to compute the dimensionality reduction because it is going to give us **a mostly vertical discriminant vector**, separating the 2 classes. Therefore if we want to implement it, we have to do on our own as we did, obtaining the eigenvalues and eigenvector and projecting it on a subspace using the Fisher linear discriminant[57]. In this case, we are going to follow the steps explained by Sebastian Raschka on his website, explaining LDA in 5 steps [56].

  We make an assumption, it is that LDA must always be used with normalized data, it means that all our attributes have to be normalized. In the case we make a reduction of dimensions could give us a reasonable result, but in the case, we use this algorithm to make a classification, it won't work at all.

- Procedure

  As we explained we are going to follow some steps to perform the LDA algorithm.

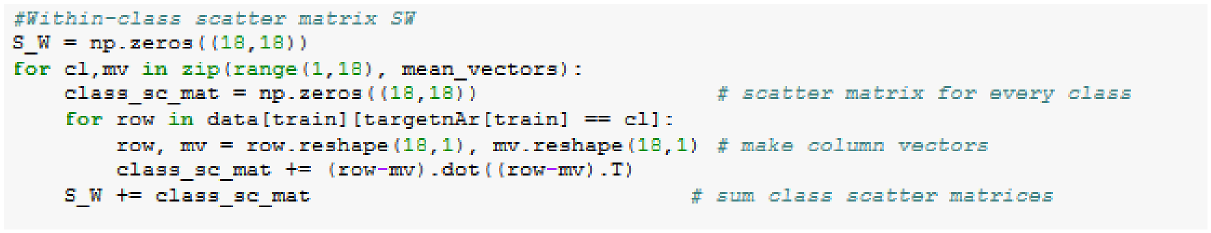
1. **Computing the d-dimensional mean vectors**: in the first step, we will compute the mean vector of the different classes. As we can see in the code from figure 47, we are calculating the values 0 and 1 the mean vectors.



```
#Computing the d-dimensional mean vectors
mean_vectors = []
for cl in range(0,2):
    mean_vectors.append(np.mean(data[train][targetnAr[train]==cl], axis=0))
```

Figure 47. Calculating mean vectors

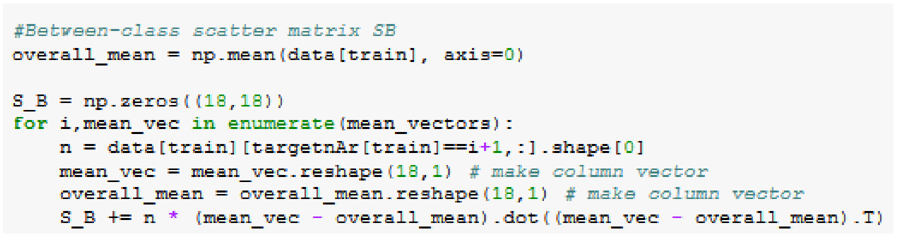
2. The second step will be to calculate the **Scatter Matrices**, with the correspondent within-class scatter matrix SW and the between-class matrix SB



```
#Within-class scatter matrix SW
S_W = np.zeros((18,18))
for cl,mv in zip(range(1,18), mean_vectors):
    class_sc_mat = np.zeros((18,18))                  # scatter matrix for every class
    for row in data[train][targetnAr[train] == cl]:
        row, mv = row.reshape(18,1), mv.reshape(18,1) # make column vectors
        class_sc_mat += (row-mv).dot((row-mv).T)
    S_W += class_sc_mat                              # sum class scatter matrices
```

Figure 48. Calculating within-class

In figure 48 we can observe we calculate the within-class scatter matrix SW following some steps in order, first calculating the scatter matrix for each class and after that making the column vectors and finally summing the class scatter matrices obtained.

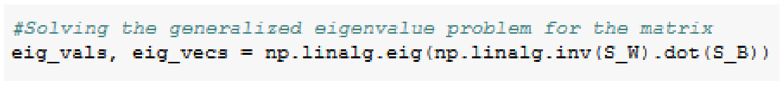


```
#Between-class scatter matrix SB
overall_mean = np.mean(data[train], axis=0)

S_B = np.zeros((18,18))
for i,mean_vec in enumerate(mean_vectors):
    n = data[train][targetnAr[train]==i+1,:].shape[0]
    mean_vec = mean_vec.reshape(18,1) # make column vector
    overall_mean = overall_mean.reshape(18,1) # make column vector
    S_B += n * (mean_vec - overall_mean).dot((mean_vec - overall_mean).T)
```

Figure 49. Calculating Between-class

In figure 49, we are calculating the between-class scatter matrix. The process is similar to the within-class scatter matrix, but this time we are going to calculate the size of the respective classes “n”, the sample mean “mean_vec” and the overall mean “overall_mean”.

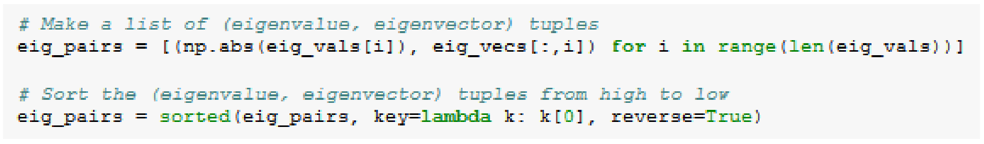
3. The third step we are going to perform is to obtain or solve the eigenvalue problem for the matrix $\frac{SB}{Sw}$, obtaining like this the linear discriminants as we can see in the figure. 50



```
#Solving the generalized eigenvalue problem for the matrix
eig_vals, eig_vecs = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))
```

Figure 50. Obtaining eigen values and eigen vectors

4. In step four, we are going to sort the eigenvector by decreasing the eigenvalues. We are not only interested projecting the data in a new subspace, but also reduce the dimension of the dataset. We are going to drop the eigenvectors with the lower eigenvalues, because they have less information about the distribution of the data. This process is shown in Figure 51.



```
# Make a list of (eigenvalue, eigenvector) tuples
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the (eigenvalue, eigenvector) tuples from high to low
eig_pairs = sorted(eig_pairs, key=lambda k: k[0], reverse=True)
```

Figure 51. Obtaining and sorting eigenpairs.

After this step, we are going to perform some similar steps like in PCA, calculating the W matrix (k x d-dimensional eigenvector matrix), reducing the initial dimension space in the new one. In figure 52 we can see this process.

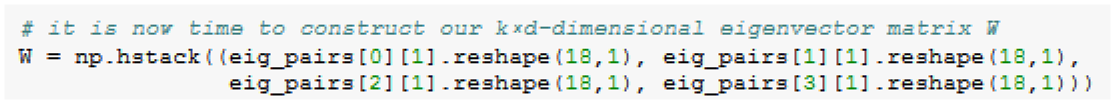

```
# it is now time to construct our k×d-dimensional eigenvector matrix W
W = np.hstack((eig_pairs[0][1].reshape(18,1), eig_pairs[1][1].reshape(18,1),
               eig_pairs[2][1].reshape(18,1), eig_pairs[3][1].reshape(18,1)))
```

Figure 52. Obtaining W Eigenvector matrix.

5. In the final step, we are going to transform the samples onto the new subspace, using the W matrix obtained, and multiplying it by the features train part and the features test part. This final step can appreciate in figure 53.

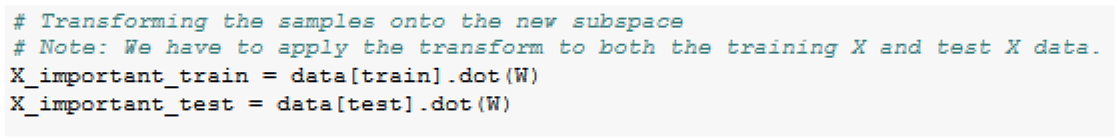

```
# Transforming the samples onto the new subspace
# Note: We have to apply the transform to both the training X and test X data.
X_important_train = data[train].dot(W)
X_important_test = data[test].dot(W)
```

Figure 53. Transforming dataset with LDA

2.2.2 Machine Learning Algorithms

The different machine learning algorithms we are going to use are the next ones:

- Decision Tree Classifier: the reason we want to use this algorithm is that they are easy to read and understand what our algorithm does to take a decision. The algorithm is going to add a node to the root of the tree, and all nodes are going to receive a list of rows as input, but the node is going to be the only one which receives the entire training set. Each node, will make a question and depend on the answer will classify the input in two subsets classifying the information. Each time we will make a new question, learning how and when to do them, purifying the information as much as possible.

  But do decide how much importance has a question, we are going to use a metric called Gini impurity, which measures uncertainty. To reduce this uncertainty, we are going to use a concept called information gain. We will make questions until the moment there are not more questions to make. We can find more information about this algorithm in sci-kit[58].

- Extra Trees Classifier: This algorithm seeks to randomize tree building in the context of numerical input features. Is quite similar to Random Forest, but this one drops the idea of using bootstrap copies of the learning (advantage in terms of bias) and selecting a cut-point at random (excellent variance reduction effect). It leads often to increased accuracy because of smoothing. We can find more information about this algorithm in sci-kit[59].

- Neural Network: This algorithm was created inspired by the biological neural networks, tested successfully solving hard problems. The core of our neural network is going to be the "perceptron", which is going to be the representation of a neuron, and with them, we are going to build a neural net. Our neural network is going to have on one side the input layer, at least 2 inputs and one output layer, and everything in the middle is going to be called the hidden layer. If we create many layers in the hidden layers is going to be called deep network. The information is going to be sent from the input layer through the synapsis to the neurons, multiplying the entrance by a weight. The neurons are going to process the information, adding the output information from all the synapses, and applying an activation function, modeling complex nonlinear pattern. We can find more information about this algorithm in sci-kit[60].

- Random Forest: It is one of the most popular machine learning algorithms, being able to perform classification and regression tasks. Other advantages of this algorithm are the capacity of handling missing values or large dataset with higher dimensionality, but it is going to have some disadvantages also, like obtaining good results for classification, but not so good for regression or we are not going to be able to control properly what our algorithm is doing. In general, we will build a forest with some trees, and the most trees in our forest, the more robust the prediction, obtaining higher accuracy. To build our algorithm we are going to use different approaches, like information gain or Gini index approach. It works as a large collection of decorrelated decision trees, using them to make a classification. We will give our information to the classifier, and each tree of our forest based on attributes gives a classification, saving the tree votes for that class. The algorithm decides the class having the most votes over all the other trees in the forest. We can find more information about this algorithm in sci-kit [44].

- Gradient Boosting Classifier: the idea of this algorithm is the next. It is going to work on the principle of ensemble, creating collections of predictors. It is going to combine a set of weak learners, delivering improved prediction accuracy. Boosting is a technique where at the beginning the learners are going to fit simple models, and are going to be focus on the difficult parts of the data. Therefore, at any instant t, the outcomes are weighed based on the outcomes previous instant t-1. If our predictions were correct, we will give a lower weight, and if the predictions were wrong we will give higher weight. Therefore, after many iterations, focusing on the difficult part the result will be better and all models give a weight depending on the results, and obtaining a consolidated result. We can find more information about this algorithm in sci-kit[61].

- Gaussian NB: this algorithm works on conditional probability, which is the probability that something will happen, based on something which has already occurred. Therefore, using the knowledge of something that already happened, we can predict the probability of an event that didn't occur yet. So, the classifier is going to predict membership probabilities for each class, like the probability of a record belongs to a class or another one. The class with the highest probability is going to be considered as the most likely class. As we are interested in tuning our algorithm, we used the CalibratedClassifierCV in python to be able to tune this algorithm. We can find more information about this algorithm in sci-kit[62]–[64].

2.3 Techniques needed

In this section, we will explain the techniques that have been used to balance our dataset and evaluate the results of our algorithms.

2.3.1 Handling imbalanced data

Oversampling and undersampling are techniques in data analysis used to adjust the class distribution of a dataset, but before explaining what are these techniques, we are going to explain what is imbalanced data. It is a topic which has been studied for about two decades in machine learning, being the subject of many papers and workshops. A vast number of techniques have been tried, and just a few answers were able to answer how to deal with this kind of situation. At the end, it depends on the data and what we are looking for[65].

In the beginning, we realized that our dataset is **imbalanced**, it means that we don't have the same number of examples of each class in our dataset. The goal of a classification algorithm is to try to learn a separator (classifier) that can distinguish the different classes. There are many ways of doing this, based on various mathematical, statistical, or geometric assumptions, but in the real life, the datasets usually are not balanced.

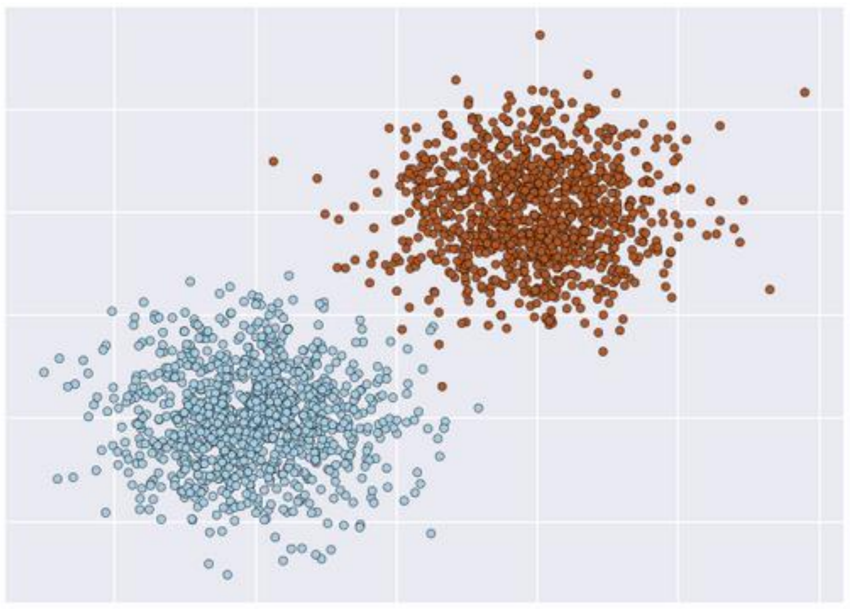


Figure 54. Balanced dataset

In figure 54 [65] we can see a dataset with two balanced classes, and in figure 55 the dataset we use for this experiment. The main problem is that these classes are imbalanced, the green points are greatly outnumbered by the blue ones.

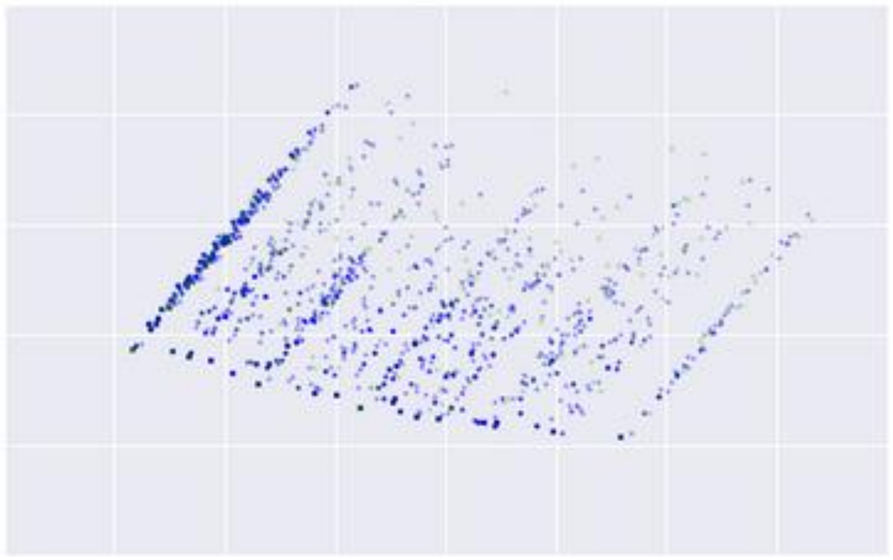


Figure 55. Imbalanced dataset

Research on imbalanced classes often considers imbalanced to mean a minority class of 10% to 20%. In reality, datasets can get far more imbalanced than this as in our case, where the minor class is around 3% of the total. Here are some examples:

- 2% of the credit card accounts are defrauded per year, most fraud detection datasets are imbalanced.
- HIV prevalence in the USA is ~0.4%, the datasets are extremely imbalanced.
- Factory production defects rates are around 0,1%.

Conventional algorithms are often biased towards the majority class because their loss functions attempt to optimize quantities like error rate, without taking the data distribution into consideration. In the worst case, minority examples are treated as outliers of the majority class and ignored. Sometimes this situation can be perfectly acceptable if we are looking to maximize our accuracy, minimizing the error rate, but for us is more important to classify the rare class because it is the one that represents the number of patients with a disease.

2.3.1.1 Approaches

Usually, in data scientist, we ask ourselves how to work with our imbalanced data, and the answer is that it depends on the data. We have many useful approaches:

- Sometimes it works properly without doing nothing, and we can work directly using the natural distribution.
- We can balance the dataset applying oversampling, undersampling or synthesizing a new minority class.
- We can adjust before or after the class weight, adjust the decision threshold or modify the algorithms to be more sensitive to rare classes.
- Build new algorithms to perform in imbalanced data

We can only make progress if we are measuring the right things. Therefore, there are some steps we would have to consider before start training the classification algorithms.

- When we work with imbalanced data, we don't have to use accuracy to evaluate our classifier. Accuracy applies a naïve of 0.50 threshold to decide between classes, which is wrong when the classes are imbalanced. Usually, our algorithm will be more sensitive to the major class, therefore accuracy is not a good pattern to measure the performance of our classifier. I would suggest using a ROC curve.
- Get the probabilities instead of the labels. With the probabilities, we will be able to build the Roc curve, instead of just predicting points the ones won't be so helpful.
- Usually, we get probabilities estimation with a threshold of 0.5 to separate the classes, which is not the best idea. Take a look at the Roc curve, and decide which one would be the best threshold for your case.

- We have to test always in the natural part of the data, it doesn't matter what we do in the training set, at the end the test part has to be the original one.

2.3.1.2 Oversampling and Smote Algorithm

Oversampling [65] replicates the minority class instances to create a bigger population of this type as we can see in figure 56[50]. In the other hand, undersampling throws away data of the bigger class. On the internet, we are able to find papers supporting oversampling because we create more data, instead of undersampling, but the truth is that it is not that easy as it looks like.

Oversampling for one side would replicate the data randomly, and at the same time replicate the errors that we have in our dataset, it means, create noise. For example, if a classifier makes a false negative error on the original minority data set, and that data set is replicated five times, the classifier will make six errors on the new set.

By using oversampling, our dataset would show a smaller variance, because the bigger quantity of synthetic data and by undersampling the data would show a bigger variance than they really do.

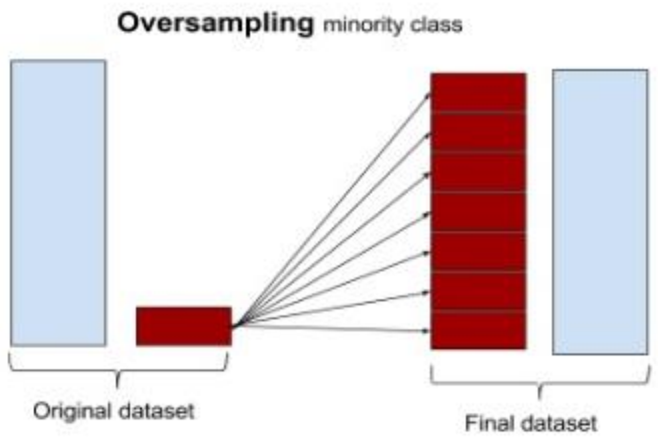


Figure 56. Oversampling

There is a big number of methods available to oversample the dataset used in our experiment for a classification problem. One of the most common techniques and the one used in our experiment is **SMOTE**, Synthetic Minority Over-sampling Technique [66]. It works like this:

- First, we are going to ignore the majority class examples. Then we are going to choose a sample in this feature space and consider the k nearest neighbors as we can see in Figure 57[65].

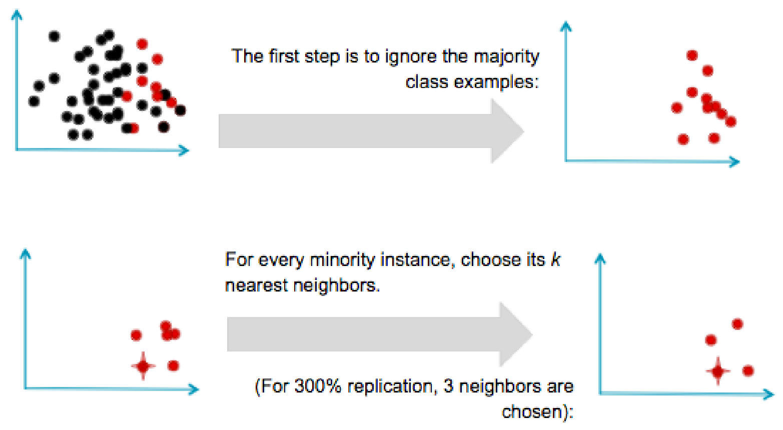

The first step is to ignore the majority class examples:
For every minority instance, choose its *k* nearest neighbors.
(For 300% replication, 3 neighbors are chosen):

Figure 57. Ignoring majority class and choosing k nearest neighbors.

- Finally, we can see in Figure 58[65] how the algorithm creates a synthetic point taking the vector between one of those k neighbors, and the sample we took and multiply the vector by a random number between 0 and 1. It will finally add this to the current data point to create the new synthetic data point.

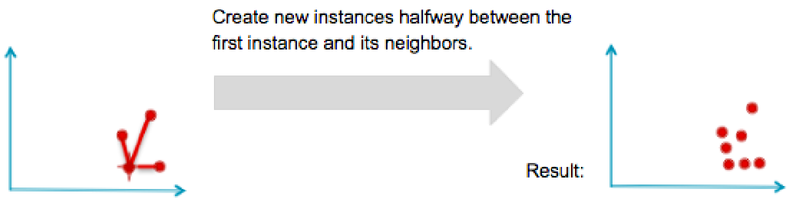

Create new instances halfway between the first instance and its neighbors.
Result:

Figure 58. The process to create synthetic data.

2.3.1.3 Undersampling

In figure 59[65] we can appreciate how Undersampling works. It consists in down-size the majority class by removing randomly samples of this type until the moment the dataset is balanced. It is one of the most common and simplest strategies to handle imbalanced data. One problem of undersampling would be that it can discard potentially useful data [67].

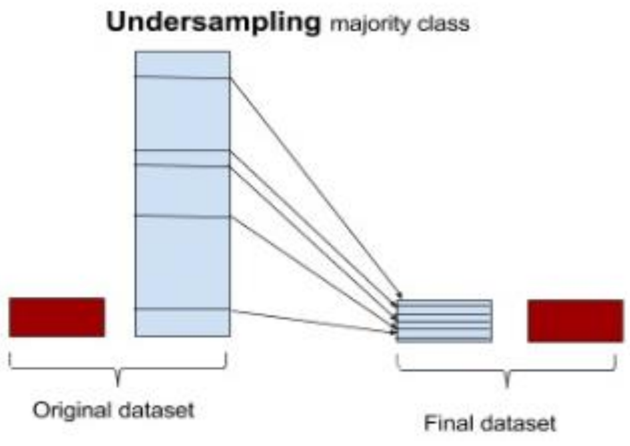


Figure 59. Undersampling

During the last years, different techniques have been proposed, usually more advanced methods, but they didn't bring any improvement with respect to simply selecting samples at random. By undersampling, we solve the class imbalance issue and increase the sensitivity of our models. But in general, the more imbalanced the dataset is, the more samples will be discarded by undersampling, therefore throwing away potentially useful information.

2.3.1.4 Libraries and approach used

To carry out this project, we have imported the imbalanced-learn library[68], since it gives us many solutions to this problem and the algorithms work correctly. We will need certain requirements that are specified in the README file.

Until this moment there is not a clear correlation between improvement and balanced data, but it is true that many algorithms work better when the data is balanced, not discriminating the minority class in the dataset [69], [70].

For this project, we used both implementations in python, obtaining a better performance. Here we are going to find a link where we can find the implementation of both techniques [71].

2.3.2 Hold-out validation vs cross-validation

With the "single hold-out set estimator" strategy (it is called "train/test split method" too), we are going to split our data into two parts, train and test, as we can see in Figure 60[72]. With the training part, we are going to train our algorithms, it doesn't matter if we are applying dimensionality reduction algorithms or modeling algorithms, they will be always trained in this part. Common proportions are 70%/30% training/test.

It is the easiest and fastest way to split the data, but we are going to find some inconveniences which have to be understood:

- Single train-test split is unreliable measure of model quality (unless you have very large dataset)

- If you have a large set of data (>50 000 samples) then train-test split might be enough

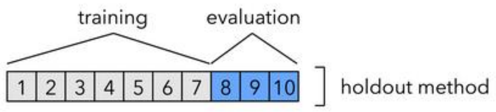


Figure 60. Hold-out method showing both parts.

In the other hand, we can find K-fold cross-validation. It is similar to the Hold-out validation, but here we are going to split first our dataset into K subsets (folds) as we can appreciate in Figure 61, training our algorithms on K-1 subsets and testing with the remains subset. We are going to iterate it K times, changing in each iteration the testing fold. At the end, we will obtain the result for each fold and we can calculate the average of the results obtained.

The advantages using K-fold cross-validation is a lower variance, reduced by averaging over the k different partitions, obtaining a less sensitive performance to the partitioning of the data.[73]

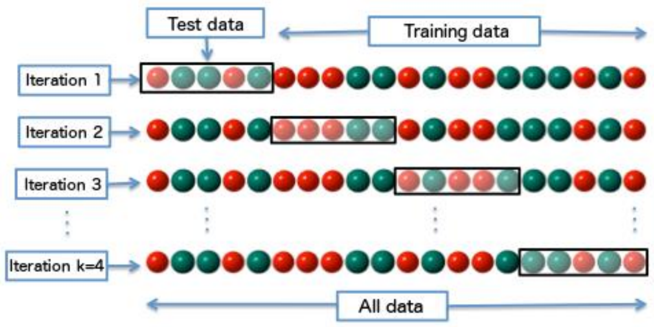


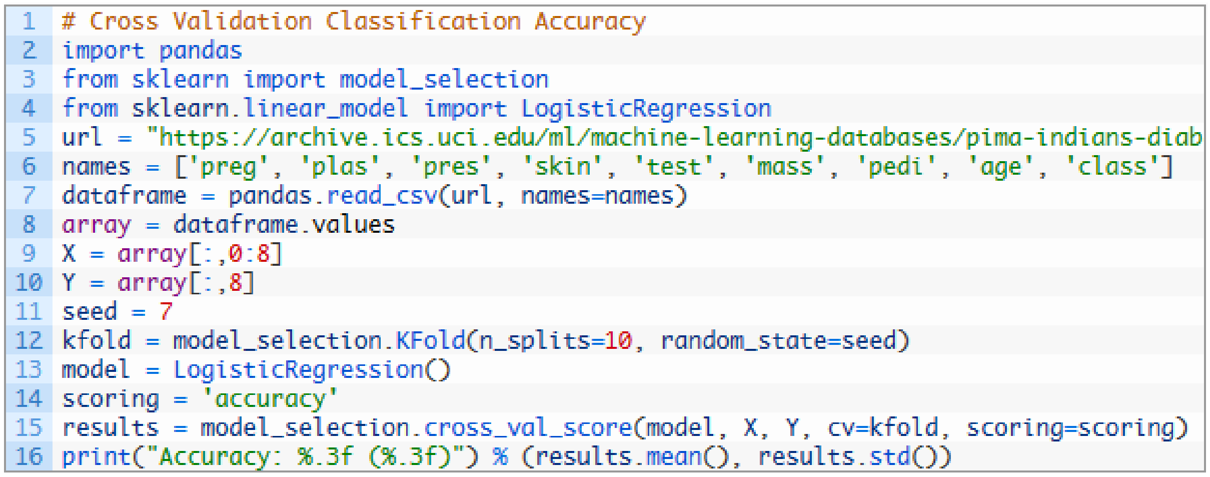
Figure 61. Diagram of k-fold cross-validation with k =4.

2.3.3 Metrics

We are going to explain the Metrics used in our project.

2.3.3.1 Classification Accuracy

It is the most common metric for classification problems. We will calculate the sum of correctly calculated predictions as a ratio of all predictions made. It is discouraged to use this method to evaluate our classification algorithm, except in cases where our dataset is balanced and the predictions have the same importance as the errors, which is unusual. In figure 62, we can appreciate the process in Python[74].



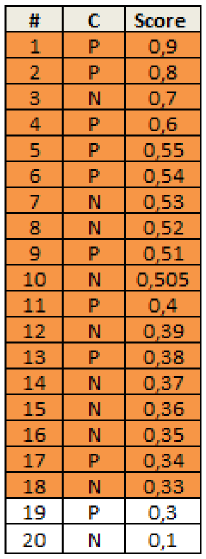
```python
# Cross Validation Classification Accuracy
import pandas
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diab
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = LogisticRegression()
scoring = 'accuracy'
results = model_selection.cross_val_score(model, X, Y, cv=kfold, scoring=scoring)
print("Accuracy: %.3f (%.3f)") % (results.mean(), results.std())
```

Figure 62. Classification Accuracy

2.3.3.2 Area Under the ROC curve

We use the ROC curve[75] to see how well our classifier can separate positive and negative values, identifying at the same time the best threshold for separating them. By default, this value is going to be 0.5, but we can change it whenever we want and in many cases, would be recommended to change it (usually in medical projects, the threshold is smaller, but it depends on our goals).

To be able to use the ROC curve, our classifier should be able to rank examples such that the ones with higher rank are more likely to be positive.



#	C	Score
1	P	0,9
2	P	0,8
3	N	0,7
4	P	0,6
5	P	0,55
6	P	0,54
7	N	0,53
8	N	0,52
9	P	0,51
10	N	0,505
11	P	0,4
12	N	0,39
13	P	0,38
14	N	0,37
15	N	0,36
16	N	0,35
17	P	0,34
18	N	0,33
19	P	0,3
20	N	0,1

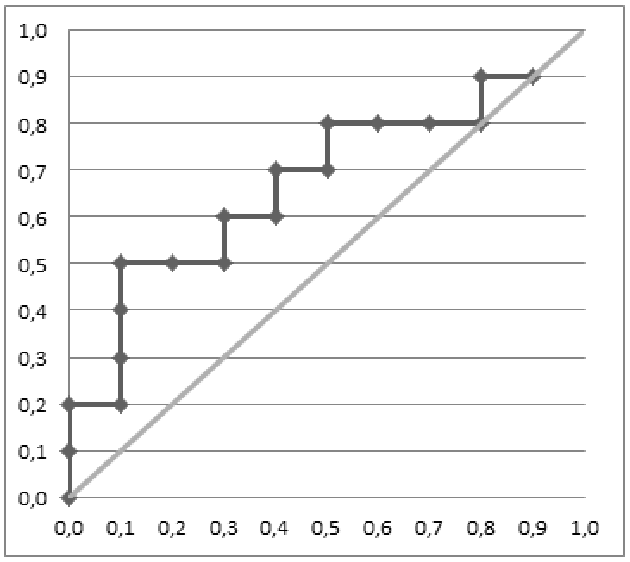


Figure 63. Building a ROC curve. Each point of the ROC curve can be considered as a miniclassifier

As we can appreciate in Figure 63[76], to build a ROC curve first we have to order the test examples by them score. We will start in our graphic in the point 0.0 and for each example x we will move one position up in the case our classifier predicted it is positive, and in the case, it predicts a negative value, we will move right. We can realize that as the examples with a higher score, are more likely to be classified as positive.

The ROC curve can be broken down into sensitivity and specificity. A binary classification problem is really a trade-off between **sensitivity** and **specificity**.

- Sensitivity is the true positive rate also called the recall. It is the number of positive instances predicted correctly.
- Specificity is also called the true negative rate. It is the number of negative instances predicted correctly.

When we have our ROC curve we can obtain the AUC, which is a metric for binary classification. The AUC represents a model's ability to discriminate between positive and negative classes. An area of 1 represents a perfect test; an area of .5 represents a worthless test[76].

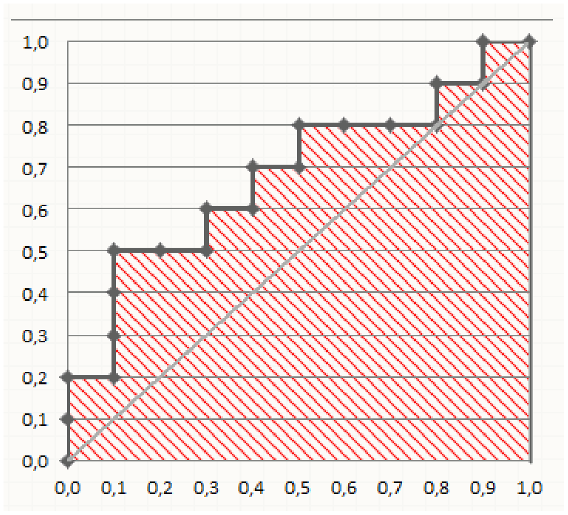


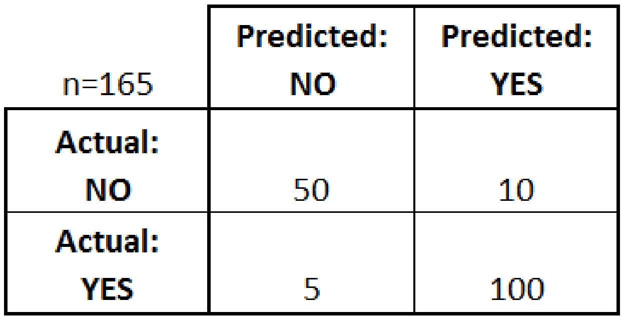
Figure 64. The area under the curve.

The whole point of using AUC is that it considers all possible thresholds.

2.3.3.3 Confusion Matrix

The confusion matrix shows us the precision of a model with two or more classes. The table presents predictions on the x-axis and accuracy outcomes on the y-axis. Each cell represents the number of predictions made by our algorithm.

For example, our algorithm can predict that a value has been predicted as a number 1, and may not be the case and this is a 0. Because it was considered as a false positive since despite being a negative value, we have predicted as positive. In figure 65[77] we can appreciate an example of the confusion Matrix, where we can see from a total of 165 objects, 50 negative values were predicted as false and 100 positive values predicted as true. 15 values were predicted false.



n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Figure 65. Confusion Matrix

2.4 Libraries, applications, and sources

We are going to give a small introduction about the programming language used, applications and libraries needed in this project.

2.4.1 Introduction to Python

As we mentioned in the beginning of the TFG, we are going to give a small introduction about what is Python and some tutorials would be interesting to check. A deep introduction to python could take many pages and it is no the purpose of this TFG. Because of the field, we are touching in this research, we are going to give some information and hints for new data scientists and beginner students of machine learning.

We would recommend taking a Python course before starting this TFG, here we are not going to explain or go deeply in basic commands of python, but explaining algorithms and libraries necessary for the execution of the algorithms.

Figure 66. Python

Python is a powerful, dynamic and interpreted language. It is quite easy to start programming in python because of the readability and the advantage of giving the programmers the possibility of expressing concepts in fewer lines of code that might be used in Java or C++. In Python, we don’t have type declarations of variables, parameters, function or methods in the source code, making the code quite short and flexible.

I would recommend taking some Edx courses about data science and python for data scientist, to understand the first steps and take some experience in this field[78].

Some characteristics we can find in Python are:

- We don’t need any compilation, we can run our code directly from source code.
- Multiparadigm programming language: it is able to support object-oriented and structured programming, such as:
    - Many languages features support functional programming and aspect-oriented programming
    - Other paradigms are supported via extensions like logic programming.
- Free and Open source, you can freely distribute copies of the software, read it, develop it, reuse or change it. It is based in the paradigm FLOSS, which concept is a community which shares knowledge.
- High-level language, we don’t need to take care of low level details such as memory management.
- Portable, you can use Python on several platforms, such as:
    - Linux
    - Windows.
    - Macintosh
    - Solaris

    - Amiga
    - …
- Extensible, if we want a piece of code to run fast or we don't want some piece of the algorithm to be open, we can just code this part of the program in C or C++ and use it from our Python program.

2.4.2 Anaconda

To install python, we would recommend installing Anaconda. We can find a screenshot of the platform in Figure 67 [79], it is the most popular Python data science platform and it will help us to install python and other interesting packages like NumPy, Scipy, etc.

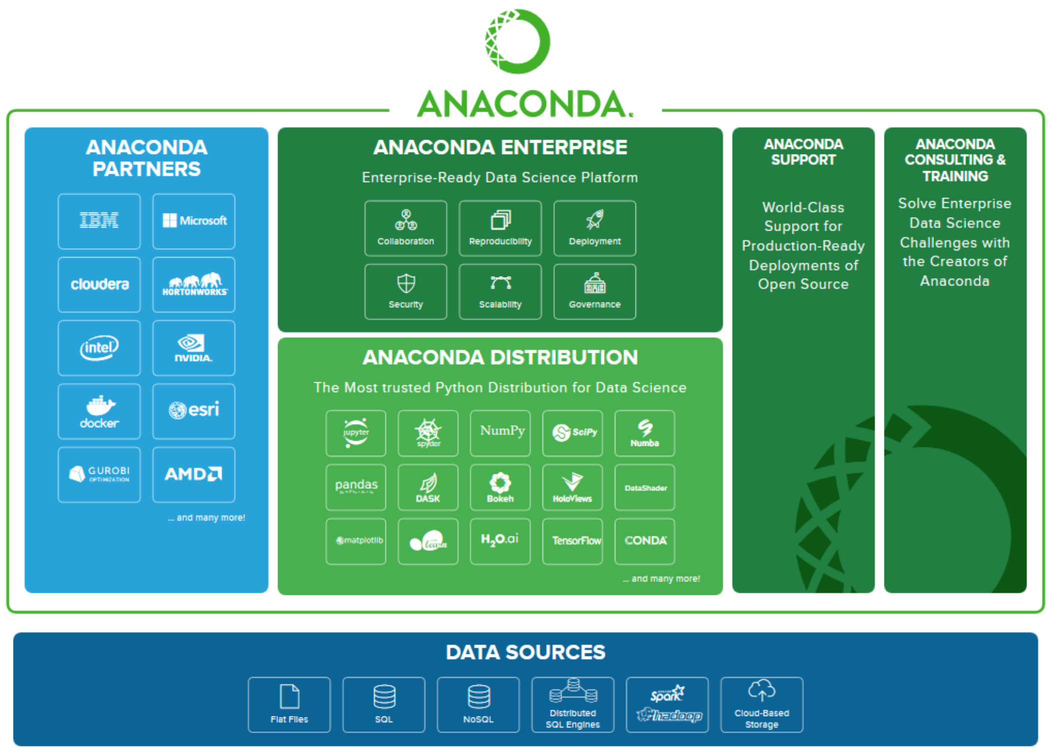


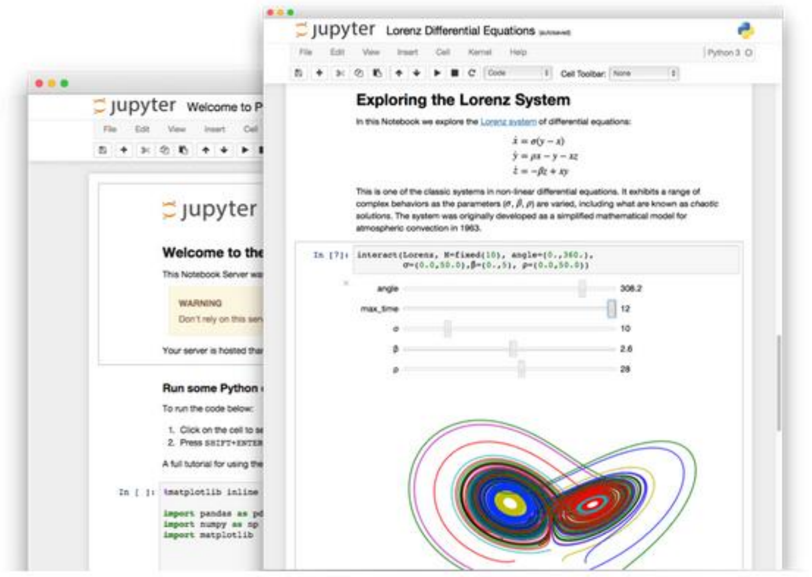
Figure 67. Anaconda

2.4.3 Jupyter

Jupyter Notebook is an interactive open source web application created for python. We recommend installing it because it is quite intuitive to use, allowing us to create and share documents containing live code, equations, visualizations and explanatory text. We can see the interface in figure 68[80].

It allows us to use other languages as Scala, R or Julia, all Data Science popular programming languages. We are going to have big data integration, leveraging big data tools like Apache Spark from Python.

We just need to program in our notebook and run the code. We are able to share our notebooks with other users using email, GitHub, and the Jupyter Notebook Viewer.



jupyter Lorenz Differential Equations
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None
jupyter Welcome to P

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as chaotic solutions. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

```
angle     306.2
max_time  12
σ         10
β         2.6
ρ         28
```

Figure 68. Jupyter interface

2.4.4 Cygwin

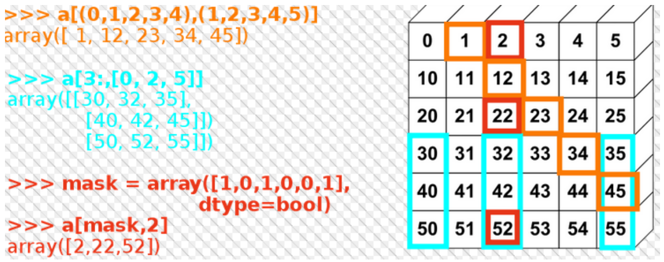
It is a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows. It is a Unix-like environment and command-line interface for Microsoft Windows. It is not necessary to be installed to work with Jupyter.

2.4.5 Libraries

The Stack is quite big, there is more than dozens of libraries and we would like here to recommend the core packages we are going to need for the different algorithms in python:

- NumPy: a fundamental package for scientific computing with Python, which is a collection of software specifically designed for scientific computing in Python (do not confuse with SciPy library). It provides an abundance of useful features for operations on n-arrays and matrices in Python. The library provides vectorization of mathematical operations on the NumPy array type, which ameliorates performance and accordingly speeds up the execution.

  One of NumPy's major strength is that most operations are implemented as C/C++ and FORTRAN code for efficiency.

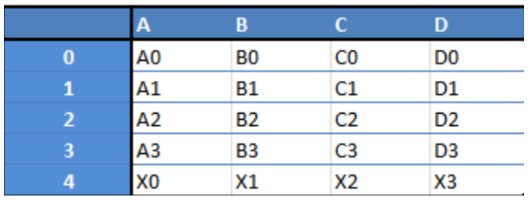


```
>>> a[(0,1,2,3,4),(1,2,3,4,5)]
array([ 1, 12, 23, 34, 45])

>>> a[3:,[0, 2, 5]]
array([[30, 32, 35],
       [40, 42, 45]])
       [50, 52, 55]])

>>> mask = array([1,0,1,0,0,1],
                 dtype=bool)
>>> a[mask,2]
array([2,22,52])
```

Figure 69. Indexing applications[81]

- Scipy: It is a Python-based ecosystem of open-source software for mathematics, science, and engineering. It is considered to be one of the core packages for scientific computing routines. As a useful expansion of the NumPy core functionality, it contains a broad range of functions for linear algebra, interpolation, integration, clustering, etc. [82]

- Pandas: It is a package designed to work with labeled and relational data, quite simple and intuitive. It was designed for quick and easy data manipulation, aggregation and visualization. It comes with a powerful Dataframe object, which is a multi-dimensional array object for efficient numerical operations similar to NumPy's *ndarray* with additional functionalities.[83]



	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	X0	X1	X2	X3

Figure 70. Dataframe example.

If we use Pandas on this Dataframe, we are able to delete and add columns, convert data structures to Dataframe objects, handle missing data, apply grouping, etc.

- Matplotlib: It is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. It is a top piece of software which is making Python a strong competitor to such scientific tools as MatLab or Mathematica. The code is low level, meaning that we have to write the proper code to obtain the most advanced levels of visualizations, but the effort worth it [84]. In figure 73 we can find an example of this library[85].

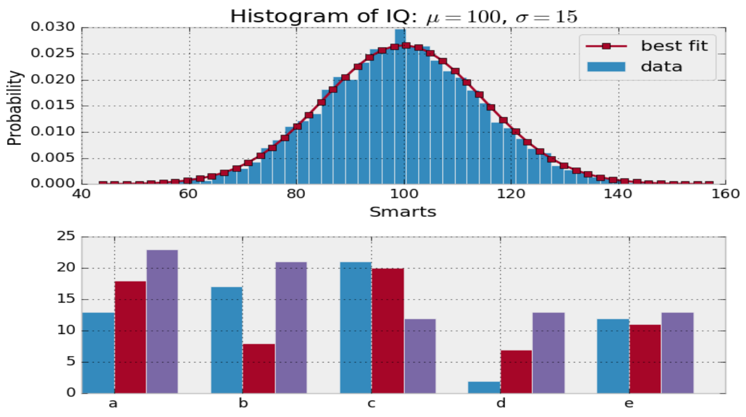
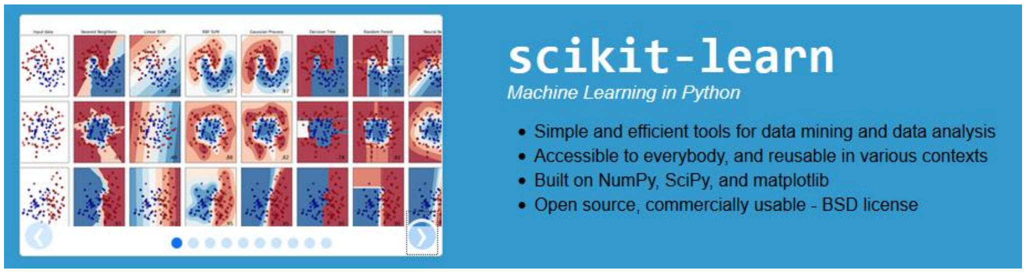


Figure 73. A histogram in Matplotlib.

- Sympy: It is a library for symbolic mathematical computations. It has a broad range of features, ranging from calculus, algebra, quantum physics, etc. [86]
- Scikit-learn: Free software machine library for the Python programming language, the most popular general machine library for Python. The package is built on the top of SciPy and makes heavy use of its math operations. It includes a broad range of different classifiers, cross-validation and other model selection methods, dimensionality reduction techniques, modules for regression and clustering analysis, and a useful data-preprocessing module. [87]

The library combines quality code and well-documented functions, with several examples to make easy its implementation.

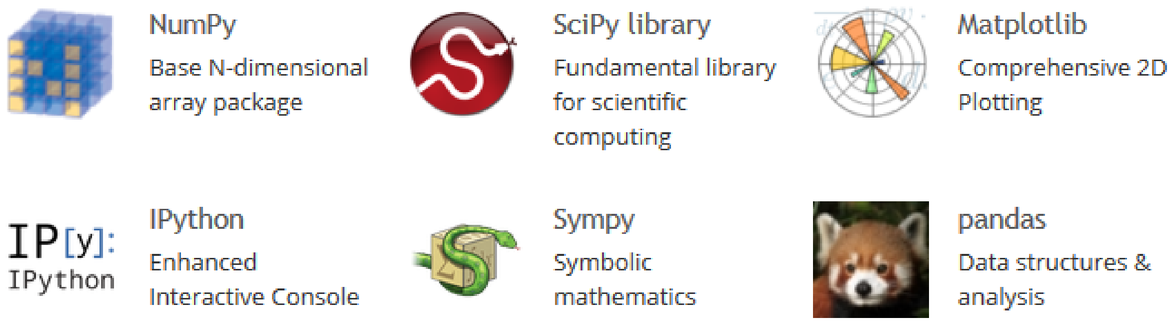


scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Figure 71. Scikit-learn [87]



NumPy
Base N-dimensional array package

SciPy library
Fundamental library for scientific computing

Matplotlib
Comprehensive 2D Plotting

IPython
Enhanced Interactive Console

Sympy
Symbolic mathematics

pandas
Data structures & analysis

Figure 72. Core libraries[82].

Block 3

Methodology and Experiment design

This block is going to focus on the Design of the Experiment. With this purpose, we would have to understand the Methodology used and the Dataset we are going to analyze.

3.1 Cross Industry Standard Process for Data Mining

CRISP-DM is a data mining process model used commonly by data mining experts to tackle problems. This methodology provides a structured approach for carrying out a data mining project. It is a robust, flexible and useful methodology, using analysis to solve problems. The model is a sequence of idealized phases but in truth, many of the phases can be solved in any order and very often it will be necessary to repeat many of them or backtrack to previous tasks in order to contrast improvements in the analysis process. Phases such as modeling, preparation of the data or understanding of the data will be usually repeated more than once as we have experience in this project. This methodology can be applied to any of the projects used regardless of the area to which it belongs, business, medicine, automotive, etc. It is possible to find several guides explaining this methodology online [88]. It breaks the process of data mining into six major phases [89].

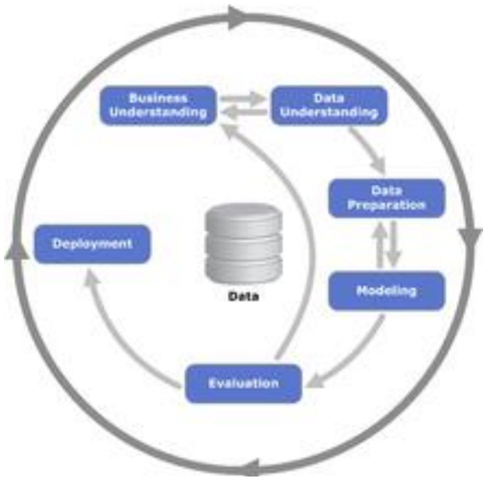


Figure 73. Process diagram showing the relationship between the different phases of CRISP-DM

In Figure 73, we can find a diagram showing the order of the phases in the CRISP-DM methodology and the dependencies between the different phases. The outer circle indicates us the cyclic nature of data mining itself. The phases of CRISP-DM are described below

3.1.1 Business Understanding

In this initial phase, we are going to understand and define which ones are the business problems that need to be solved and the objectives, everything from a business perspective. With all this information, a data mining problem will be defined and a plan designed to achieve the objective.

This first phase is important because if you do not understand the objectives of the research, you could invest a lot of time and effort in producing the correct answers for the incorrect questions

Some of the tasks in this phase are:

- Determine Business Objectives: It seeks to describe the main objective and plan required from a business perspective, as well as addressing other types of questions or objectives. We are going to set the criteria used to determine if the project has succeeded or not, answering the questions specified in the beginning. Tasks of this phase:
    - Background,
    - Business Objectives,
    - Business success criteria
- Asses Situations: We will make an inventory of the necessary resources, as well as people, data, computer resources, and software. Here we list all the requirements of the project, including the complete schedule, the comprehensibility and quality of the

required results. The limitations are listed too, and it can be the availability of resources, technological limitations such as the size of the dataset to model when being small, etc. Here we list also the risks that may delay the project or cause it to fail. Finally, in the case of having costs, a small analysis could be done to understand if it would be worth with the benefits that would be obtained. Tasks of this phase:

    - Inventory of Resources
    - Requirements
    - Assumptions and Constraints
    - Risks and Contingencies
    - Terminology
    - Costs and Benefits.

- Determine Data Mining Goal: We will determine the criteria to be able to say that the project has been successfully completed, and the results obtained have been as expected. Tasks of this phase:
    - Data Mining Goals
    - Data Mining Success Criteria.
- Produce Project Plans: The planned plan for achieving the objectives of the project is described. You must specify the tools considered in the project and the initial techniques to be implemented. The plan made initially for each stage of the project can be shown in a diagram. Tasks of this phase:
    - Project plan
    - Initial Assessment of Tools and Techniques.

3.1.2 Data Understanding

In this phase, we are going to start collecting data, which means we will get the data from internal or external sources and with specific characteristics including data volume, variety, formats and so on, as well as whether the data is in data files, in HDFS, it is life or stream.

We will identify data quality problems, discover insights into the data, detect if we have inconsistent in the data or duplicate values, the degree to which data is missing, etc.

Some of the tasks in this phase are:

- Collect Initial Data: Here we are going to perform the "Initial Data Collection Report" It will indicate where the data has been acquired, the source, the problems obtained to acquire them and the solution found to acquire them. This will help both the future execution of the project and the creation of similar projects.
- Describe Data: Here we are going to perform the "Data Description Report" The properties of the acquired data will be examined. Describing its format, the quantity, identities of the fields and evaluating if the data acquired satisfies our requirements.
- Explore Data: Here we are going to perform the "Data Exploration Report". In this step, we will address data mining questions, directly addressing project objectives, refining the description of data or quality reports and even transforming and preparing data for further analysis
- Verify Data Quality: Here we are going to perform the "Data Quality Report". We are going to examine the quality of the data, if the data is complete or if we are missing values in the data.

3.1.3 Data Preparation

It seeks to prepare the data to be used by machine-learning algorithms. This process involves a number of tasks including cleaning, filtering, transformation, feature selection or dimensionality reduction. The objective is to improve the set of features eliminating variables which are not necessary and could produce noise or contribute to a poor precision of the algorithms. Feature Engineering is going to be performed here. Some of the tasks in this phase are:

- Select Data: Here we are going to perform the "Rationale for inclusion/exclusion criteria". We decide which data is going to be used for the analysis. These decisions are made based on the goals of our project, the type of data, etc. The selection of the attributes can be both rows and columns. If it is possible, it can be explained why this data is excluded and the reasons.
- Clean Data: Here we are going to perform the "Data Cleaning Report". We will raise the quality of the data to the technical level of the analysis that we have selected. To do this we will select subsets of the data, insert new values or apply techniques estimating missing values through modeling.
- Construct Data: By this, we mean construction operations such as the production of derived attributes or transformation of values to existing attributes. The derived attributes will be generated through one or more existing values. On the other hand, new attributes are going to be generated that may not be in our dataset, but after studying it we prove that they would be necessary to achieve our objectives. Tasks of this phase:
    - Derived attributes
    - Generated Records

- Integrate Data: Here we are going to perform the "Merged Data". When integrating data, it is sought to mix or merge information from different datasets or different databases. In this process, the aggregations are made, calculating new values summarizing information from different registers or tables.
- Format Data: Here we are going to perform the "Reformatted Data". We are going to seek the correct format to work with the machine learning algorithms.

3.1.4 Modeling

The development of a machine-learning model is based on the calibration and evaluation of different techniques to obtain the best performance.

These models are used to predict, analyze, or search for patterns, associations, or groups in our dataset. Different techniques can be used to solve a problem, but it depends on the type of data we are working, so it is recommended to study the dataset first to understand the data and the algorithms that can work properly with it. In case it is necessary we could return to the data preparation phase, to improve our dataset. We will encounter two types of machine-learning algorithms: supervised and unsupervised.

The tasks in this phase are the next:

- Select Modeling Techniques: In this section, we will select the techniques of modeling, and it is possible that many of them have changed with the techniques we were supposed to use in the beginning. We must explain the chosen modeling technique and the assumptions we have made to carry them out, for example, all elements must be numbered or that the elements must be normalized. Tasks of this phase:
    - Modeling Technique
    - Modeling Assumptions

- Generate Test Design: Here we are going to generate the Test Design. Before creating a model, we will have to prove the quality and the validity of the model.

  In a situation like ours, where data mining is supervised to classify different models, certain patterns are used as measures of quality to indicate that a model works correctly or not. So, the dataset is divided into two parts, "test" which is the set to estimate the quality of the final algorithm and "Train", where we train and build our algorithm. We must describe the techniques we used and especially how we have performed this testing process.
- Build Model: We are going to run the modeling tools, creating one or more models. We will adjust the parameters for each algorithm, performing small tests checking which parameters work best in which case. It would be helpful to make a description of the resulting models, informing about the interpretation of the models and the problems that we had in order to implement them. Tasks of this phase:
  - Parameter Settings
  - Models
  - Models Description
- Asses Model: We must interpret the models with the knowledge that we have of that field and the criteria we have about data mining. After this, we have to discuss with experts in the field such as mathematicians or business analysts the results obtained.

  We must classify the models and evaluate them according to the evaluation criteria. We must consider the goals and understand that it is usual to apply a technique more than once, obtaining results generated by different techniques. The results obtained will be evaluated, enumerating the qualities of the models generated and with relation to each other. Check that the algorithms have been tuned correctly and that the chosen parameters are indeed correct for that model. Tasks of this phase:

  - Model Assessment
  - Revised Parameter Settings

3.1.5 Evaluation

After developing a model, we will evaluate if the performance is the expected and we are predicting the classes correctly. For them, it is recommended to review the steps followed from the beginning. For example, in the case of predictions, we try to evaluate the number of elements that were predicted correctly. After this phase, we must express if the results are satisfactory and determine which models we will use. The tasks in this phase are the next:

- Evaluate Results: We will evaluate the results, checking the accuracy obtained from the models. We are going to summarize the results of the evaluation, stating whether or not our model finally met the specified objectives in principle or not. In case we agree with the results obtained, we will give the models as good and we will approve them to use them in a real situation. In case the budget or time allows, other test models with real data can also be evaluated. Tasks of this phase:
  - Assessment of Data Mining Results
  - Business Success Criteria
  - Approved Models.
- Review Process: We are going to do a review of the models to verify that we have not forgotten some important task. Questions will be asked to check if the process has been performed correctly. If it is possible we have to remark which activities should be repeated or probably have been missed.

- Determine Next Steps: In this part, we can decide how to proceed with the results obtained. In case we have possible potential actions to implement, we should put pros and cons to implement them. Tasks of this phase:
    - List of Possible Action
    - Decisions.

3.1.6 Deployment

In the final step, if we are happy with the model we created, now we are going to deploy the models to run in different environments It does not mean that this one is the end of the project, even if we were just looking to improve the performance of the model, the knowledge we gained have to be organized and presented in a way the customers can understand and use. The different environments include applications, Database management systems, Apache Hadoop, Apache Spark or just a simple spreadsheet. In many cases will be the customer the one which will carry out the deployment steps, and not the analyst. Tasks in this phase:

- Plan Deployment: At this point, it is where we will describe the procedure followed to create the model so that we can use it later.
- Plan monitoring and maintenance: Here we are going to deploy the monitoring and maintenance plan. It would be helpful to detail a data mining plan, with the process we are following.
- Produce Final Report: Finally, we are going to redact a report, explaining the experiences obtained and making a report. It could be a presentation of the data mining results to Tasks of this phase:
    - Final Report
    - Final Presentation.

- Review Project: In the Experience Documentation, we are going to explain what went right and what went wrong, summarizing the experience obtained in this project. At the same time would be helpful to explain any pitfalls we encountered, misleading approaches, etc. for future research.

With this introduction, I wanted to clarify some doubts with the terminologies in this Area and give an idea about the direction of this Research for those Students interested in this Field.

3.2 Experiment design

Seven different classification algorithms have been selected, basing the estimations on the AUC results for each pair. “Classification Model-Reduction algorithm”.

Additionally, results obtained for the sensitivity, the specificity, the accuracy of each classification are shown and a graphic where we can see the results of the different folds and the ROC obtained for each one.

The next diagram in Figure 74 [90]explains briefly the procedure followed, according to the CRISP-DM methodology:

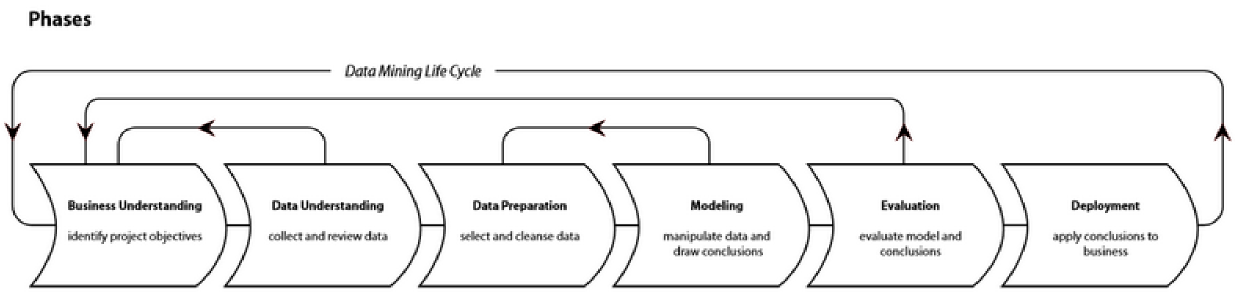


Figure 74. CRISP-DM phases followed to understand the experiment.

3.2.1 Business Understanding and Data Understanding

In the phase **Business Understanding**, we determined as requirements would be needed a student with knowledge of the subject and a teacher as cotutor of the DBIS laboratory of the Goethe Universität to carry out this project. We will also need a computer to work with a minimum of 8 Gb Ram and a core processor i5-i7 working on a Windows 10 or Linux system.

As a programming language, Python 2.7.13 has been chosen and the necessary software is:

- Anaconda 4.3.1 for 64-bit
- scikit-learn library
- Cygwin64 Terminal
- Jupiter Notebook

CRISP-DM was chosen as the methodology to be followed in this project, because it is a robust, flexible and useful methodology, using analysis to solve problems.

Speaking about the data, a dataset would be needed (in this case a medical dataset) with the intention of applying dimensionality reduction techniques on a patient classification problem for a given disease. Since a medical dataset deals with sensitive information, it will be difficult to acquire it, spending a lot of time searching one with the quality and information necessary to analyze the data. Speaking of quality, we refer to the information of the data and the facility to understand what is expressed in the dataset. This at first delayed us quite a lot, since it was not easy for us to find the right one with the diseases required by the teacher.

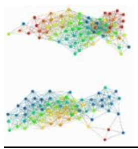
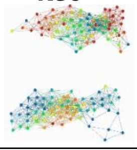
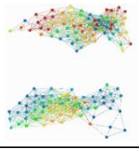
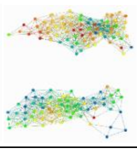
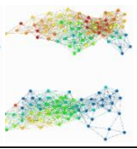
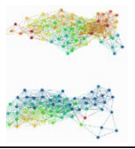
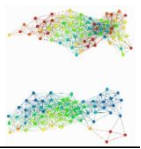
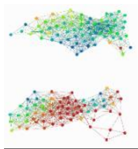
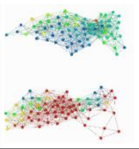
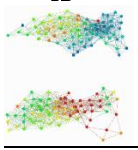
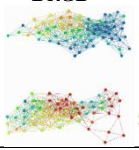
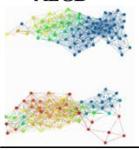
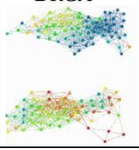
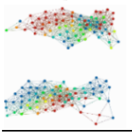
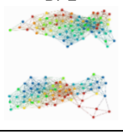
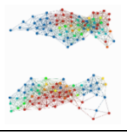
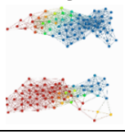
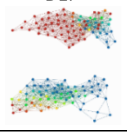
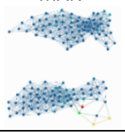
Another problem we encountered was the amount of data to analyze since it was not large enough, we had to use data mining techniques to perform acceptable algorithm training. As we work with a conventional computer, the computer performance won't be so high and the processing time may be very slow for certain algorithms. With the right tools and hardware, we could have worked with larger datasets and more complex algorithms.

At the beginning of this project, a diagram was given where we can find the different stages proposed for this project. It is expected that after finishing this project, a better understanding of the different dimensional reduction algorithms will be obtained.

Consequently, in the second phase **Data Understanding** was searched a certain dataset for our experiment, choosing the medical dataset NCS-1 Diagnosis/Demographic Data, from the University of Michigan, related to the National Comorbidity Survey: Baseline (NCS-1) [41], 1990-1992 (ICPSR 6693) [41]. It consists of 8,098 patients who have been diagnosed a mood disorder at least once in their lifetime. The patients who have been diagnosed multiple mood disorders are the key to identify a psychiatric morbidity and comorbidity. The dataset consists of nineteen disorders including major depression (DEP), mania (MAN), dysthymia (DYS), panic attack (PT), panic disorder (PD), agoraphobia without panic (AGO), social phobia (SOC), simple phobia (SIM), generalized anxiety disorder (GAD), alcohol abuse (ALCA), alcohol dependence (ALCD), drug abuse (DRGA), drug dependence (DRGD), and antisocial personality disorder (ASP).

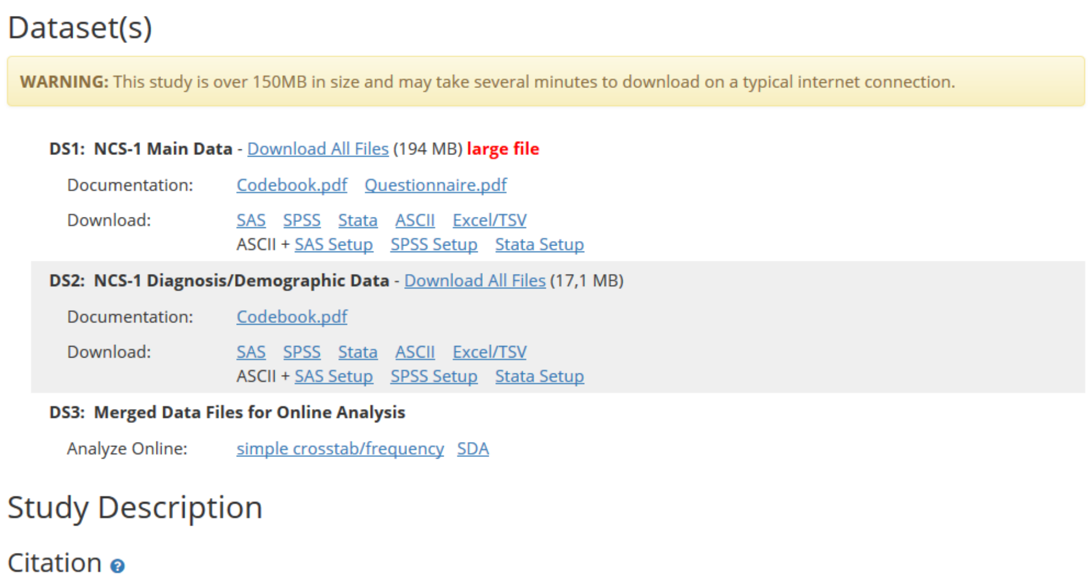
Topology techniques were used through Ayasdi [91], with the intention of understanding the patterns for each of the diseases. The following table 1 shows nineteen projections, one for each disease. The color nodes represent the concentration of this disease, having for a red node a high concentration of this disease and blue for a low. We can verify the covariance intuitively, for example in the case of having similar patterns between two diseases, we can check similar colors for both diseases compared. In the first row can be verified a number of associations between diseases. In the last row, we have placed diseases without clear associations

Table 1. A Topological Summary of NCS-1 Dataset. Nineteen Mood Disorders Are Projected in Different Color Patterns

PT	*AGO*	*GAD*	*PTSD*	*SOC*	*SIM*	*DYS*
						
AAB	*ASP*					
						
CD	*DRGD*	*ALCD*	*DRGA*			
						
PD	*BP1*	*NAP*	*ALCA*	*DEP*	*MAN*	
						

The dataset used treats mainly psychiatric illnesses, but in this study, we can also find other datasets with information on life habits, family, marital status, education, region of residence, race or ethnicity, etc.

This Survey contains other 2 datasets, but in this case, we were interested in the Diagnosis, the one contains information about the diseases of the different patients. The data were explored and described, verifying the quality of the data and understanding the connection between the features of the dataset.


Dataset(s)

WARNING: This study is over 150MB in size and may take several minutes to download on a typical internet connection.

DS1: NCS-1 Main Data - Download All Files (194 MB) **large file**

Documentation: Codebook.pdf Questionnaire.pdf

Download: SAS SPSS Stata ASCII Excel/TSV
ASCII + SAS Setup SPSS Setup Stata Setup

DS2: NCS-1 Diagnosis/Demographic Data - Download All Files (17,1 MB)

Documentation: Codebook.pdf

Download: SAS SPSS Stata ASCII Excel/TSV
ASCII + SAS Setup SPSS Setup Stata Setup

DS3: Merged Data Files for Online Analysis

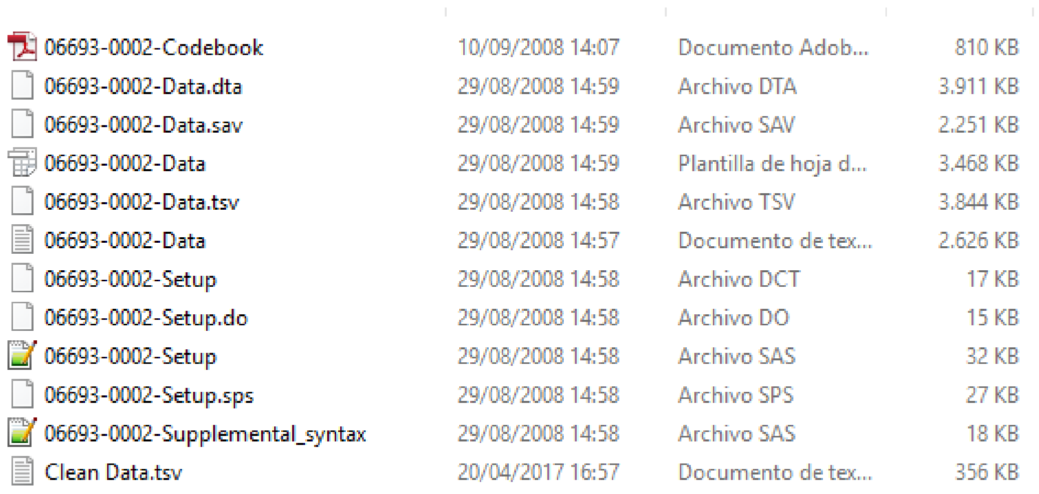
Analyze Online: simple crosstab/frequency SDA

Study Description

Citation

Figure 75. Datasets included in the National Comorbidity Survey (NCS-1)

During the first few months, the experiment was anchored in these two points, since other datasets were initially considered but did not meet the objectives described in phase 1. Other reasons were the complexity to understand the content of the dataset or the lack of information, incomplete codebook and low quality of the dataset. We were asking in different institutions to provide us some Medical Data, but most of the time we didn’t receive any reply or they had not this kind of data. Therefore, it is recommended to have an agreement with a hospital or medical center, which provides us the data, or if possible in any other case, obtain such data by oneself.



06693-0002-Codebook	10/09/2008 14:07	Documento Adob...	810 KB
06693-0002-Data.dta	29/08/2008 14:59	Archivo DTA	3.911 KB
06693-0002-Data.sav	29/08/2008 14:59	Archivo SAV	2.251 KB
06693-0002-Data	29/08/2008 14:59	Plantilla de hoja d...	3.468 KB
06693-0002-Data.tsv	29/08/2008 14:58	Archivo TSV	3.844 KB
06693-0002-Data	29/08/2008 14:57	Documento de tex...	2.626 KB
06693-0002-Setup	29/08/2008 14:58	Archivo DCT	17 KB
06693-0002-Setup.do	29/08/2008 14:58	Archivo DO	15 KB
06693-0002-Setup	29/08/2008 14:58	Archivo SAS	32 KB
06693-0002-Setup.sps	29/08/2008 14:58	Archivo SPS	27 KB
06693-0002-Supplemental_syntax	29/08/2008 14:58	Archivo SAS	18 KB
Clean Data.tsv	20/04/2017 16:57	Documento de tex...	356 KB

Figure 76. File with the dataset. Codebook of the dataset and different formats of the dataset

3.2.2 Data Preparation and Modelling

This part of the experiment will show the technique used after understanding which one would be the most correct and suitable for this problem, so it could vary depending on the focus that is being sought to achieve the goals and does not follow always a constant pattern.

The core of this project focuses on the feature engineering, which will be carried out in the **data preparation** phase. By means of feature engineering, a new dataset was created from the original, having a certain number of features after a detailed study of which one of the features would give us greater information, and eliminating those whose weight was smaller or contained redundant information. As we can see in figure 77, with the help of Ayasdi[91] it was possible to verify the relation between the features, and which ones had a greater weight, and which ones a minor one. After this, the lifetime disease features were selected to create the new dataset, and PTSDLT as the target, because of having more clear relationships with other diseases. We selected lifetime diseases because include whether a patient has a disease during his life and not just for a period, eliminating redundancy between features.

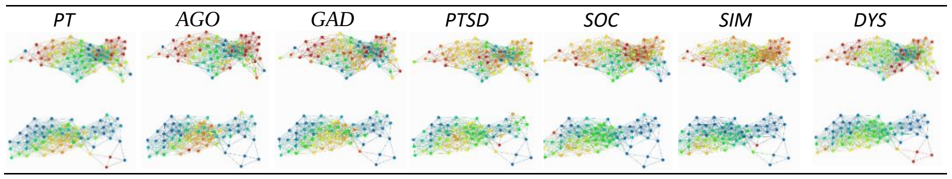
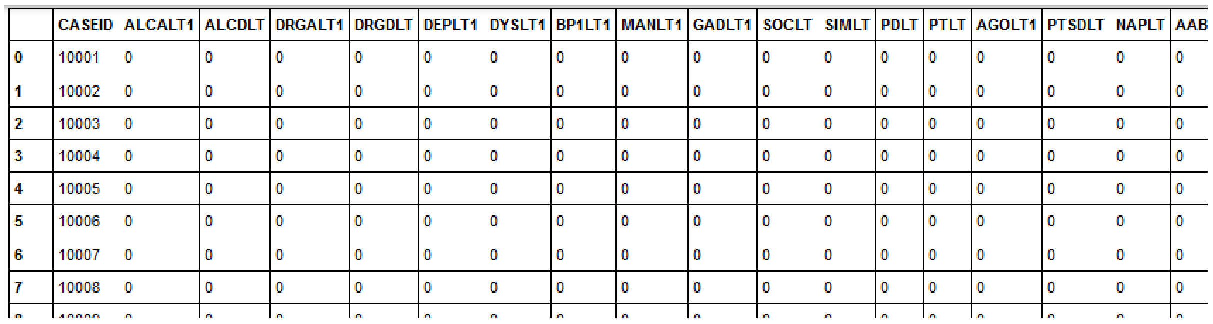


Figure 77 Topological Summary of NCS-1 Dataset. Mood disorders related to PTSD

Finally, was selected a total of 18 different features corresponding to lifetime diseases and one feature as the target of the research, shown in figure 78. The dataset is going to be called "CleanData.tsv", created from the original NCS-1, and is going to be the basis of our project, where we will apply the different dimensionality reduction algorithms and model our classifiers. As we can see here, cleaning the data and constructing the datasets are important steps to do even before applying algorithms of feature selection or feature extraction, reducing dimensions with noise or data without any value.



	CASEID	ALCALT1	ALCDLT	DRGALT1	DRGDLT	DEPLT1	DYSLT1	BP1LT1	MANLT1	GADLT1	SOCLT	SIMLT	PDLT	PTLT	AGOLT1	PTSDLT	NAPLT	AAB
0	10001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	10002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	10003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	10004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	10005	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	10006	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	10007	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	10008	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 78. Dataset with 19 diseases

The new dataset will be studied and analyzed in depth again in the phase **Data preparation** and the results will be used in the phase **Modelling**. To do this, a great deal of time was invested in applying different combinations of techniques, both processing and modeling, moving between these phases again and again until the best technique combination was found, thus obtaining the best performance for our algorithms (even when this project was not intended).

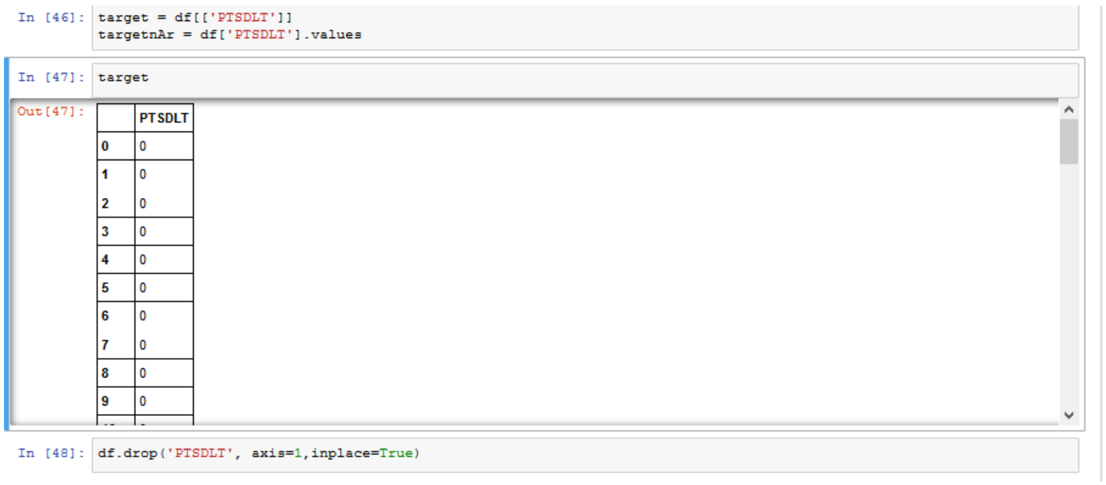
It was considered that it would be important to show a correct processing of the dataset for future research, therefore this process is shown below.

3.2.2.1 Preliminary steps

First of all, we assume that all the different elements are normalized and are numbers. For each different **Dimensionality Reduction** algorithm, a separated python file was created as we can see here below. These files were executed in Jupyter and for each one of them, a kernel was given. This is shown in Figure 79.

TFG LDA subsampling oversampling.ipynb Running

Figure 79. Python file in Jupyter

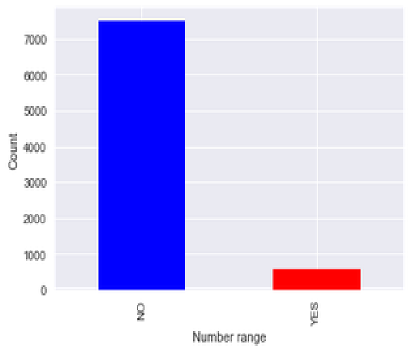


```
In [46]: target = df[['PTSDLT']]
         targetnAr = df['PTSDLT'].values

In [47]: target

Out[47]:
   PTSDLT
0  0
1  0
2  0
3  0
4  0
5  0
6  0
7  0
8  0
9  0

In [48]: df.drop('PTSDLT', axis=1,inplace=True)
```

Figure 80. Creating the target Dataset

In figure 80, we perform the next step, separating the target and the features in two different datasets, “Features” and “Target”, it was verified that the dataset was imbalanced, since a large number of patients did not show this disease, and on the other hand a small number of patients had it, as we can appreciate in figure 81.

```
Here are 8098 patients, represented with blue color the ones without disease, and red with he disease.
```



```
[(0, 7507), (1, 591)]
```

Figure 81. Imbalanced dataset. Under the graphic can be appreciated the number of patients with the disease indicated with number 1, and with 0 the patients without a disease.

Therefore, for a correct operation of the algorithms, the dataset had to be treated and for that purpose resampling techniques were applied. It will be done through **oversampling** or **undersampling** techniques, to obtain a better performance of our classification algorithms in the moment of fitting the model. Note however, all these steps (model selection, feature selection, resampling etc.) must be performed independently in each fold of the cross-validation procedure, or the resulting performance estimate would be optimistically biased. This point is raised first, since it is the way in which the experiment should be understood.

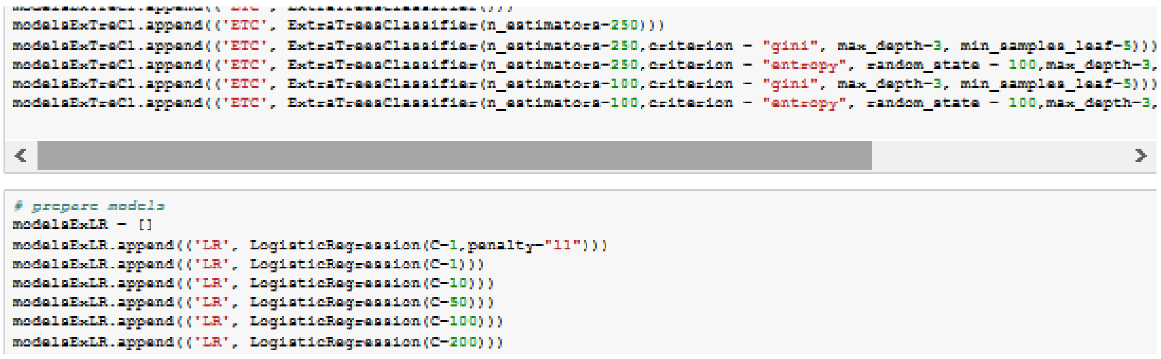
3.2.2.2 Selection of the technique to follow

After understanding what is needed to obtain a clean dataset, properly preprocessed, was the moment to apply the dimensionality reduction algorithm techniques, but first, it had to be understood which strategy was going to be followed. As the dataset doesn't contain a huge quantity of patients, the "single hold-out set estimator" strategy was not going to give a great result, and the variation in the performance estimation for different samples of data or partitions could be important. In the other hand, K-fold cross-validation estimator has a lower variance than a single hold-out set estimator, reducing the variance by averaging over k different partitions, obtaining a less sensitive performance to the partitioning of the data.

As it will be evaluated which one is the best performance obtained for a dimensional reduction algorithm in a classification problem, and not which one is the best tuned for a classification algorithm, K-fold cross-validation was the technique selected to evaluate our algorithms.

3.2.2.3 Tuning of the Algorithms

The algorithms were tuned to obtain the best performance in this research as we can see in Figure 82, even when it is not the objective of this study. This code is added to each file although it has not been used for the final demonstration. Experimentally, better results have been obtained by configuring the main parameters of each algorithm.

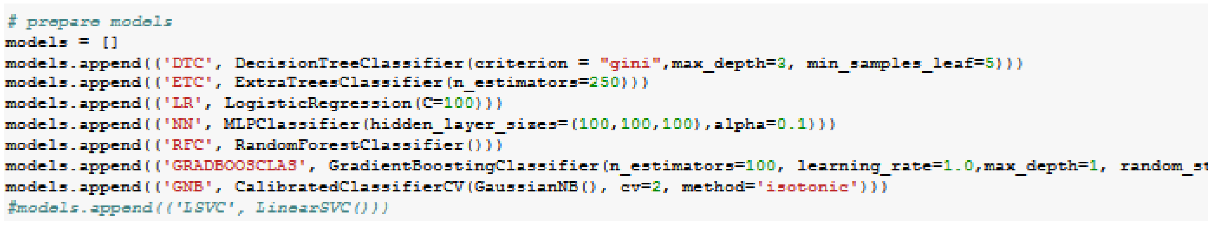


```
modelsExTreCl.append(('ETC', ExtraTreesClassifier(n_estimators=250)))
modelsExTreCl.append(('ETC', ExtraTreesClassifier(n_estimators=250,criterion = "gini", max_depth=3, min_samples_leaf=5)))
modelsExTreCl.append(('ETC', ExtraTreesClassifier(n_estimators=250,criterion = "entropy", random_state = 100,max_depth=3,
modelsExTreCl.append(('ETC', ExtraTreesClassifier(n_estimators=100,criterion = "gini", max_depth=3, min_samples_leaf=5)))
modelsExTreCl.append(('ETC', ExtraTreesClassifier(n_estimators=100,criterion = "entropy", random_state = 100,max_depth=3,
```

```
# prepare models
modelsExLR = []
modelsExLR.append(('LR', LogisticRegression(C=1,penalty="l1")))
modelsExLR.append(('LR', LogisticRegression(C=1)))
modelsExLR.append(('LR', LogisticRegression(C=10)))
modelsExLR.append(('LR', LogisticRegression(C=50)))
modelsExLR.append(('LR', LogisticRegression(C=100)))
modelsExLR.append(('LR', LogisticRegression(C=200)))
```

Figure 82. Tuning of the models.

The classification algorithms were created in the variable “models”, the one is going to be used later in our cross-validation algorithm, as we can see in Figure 83.



```
# prepare models
models = []
models.append(('DTC', DecisionTreeClassifier(criterion = "gini",max_depth=3, min_samples_leaf=5)))
models.append(('ETC', ExtraTreesClassifier(n_estimators=250)))
models.append(('LR', LogisticRegression(C=100)))
models.append(('NN', MLPClassifier(hidden_layer_sizes=(100,100,100),alpha=0.1)))
models.append(('RFC', RandomForestClassifier()))
models.append(('GRADBOOSCLAS', GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,max_depth=1, random_st
models.append(('GNB', CalibratedClassifierCV(GaussianNB(), cv=2, method='isotonic')))
#models.append(('LSVC', LinearSVC()))
```

Figure 83. Creation of model variable with the classification algorithms selected.

In Table 2 we can find a table with the value of the parameters applied for each algorithm.

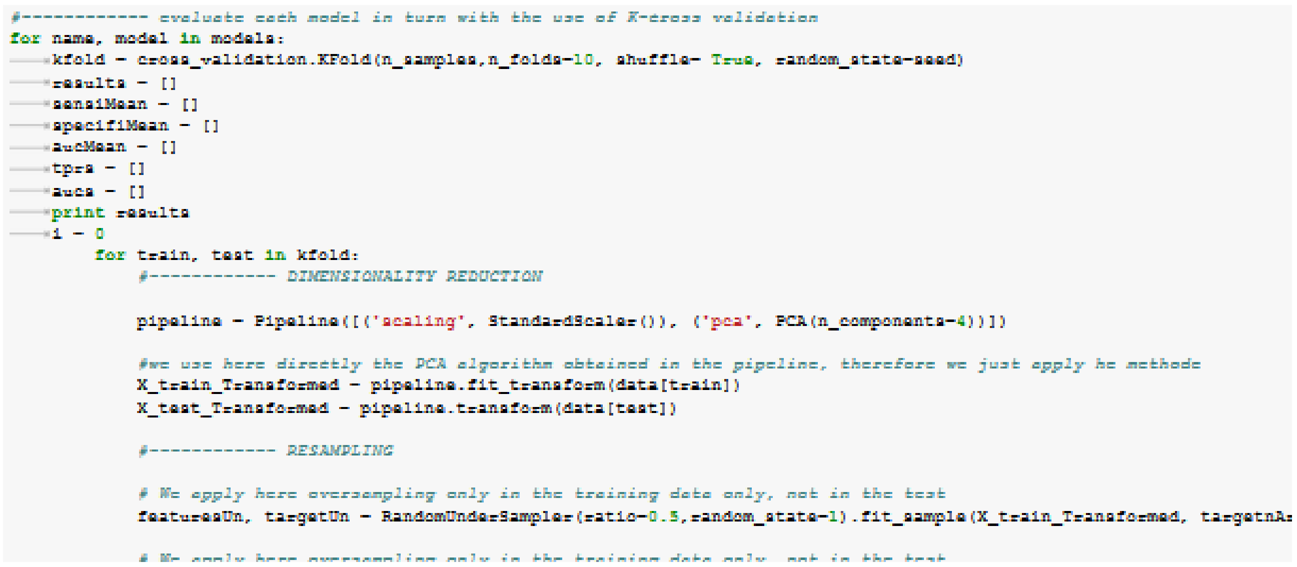
Table 2. Parameters of the models created

Models / Parameters	DTC	ETC	LR	NN	RFC	GRADBOOSCLAS	GNB
Criterion	gini	-	-	-	-	-	-
Max_depth	3	-	-	-	-	1	-
min_sample_leaf	5	-	-	-	-	-	-
cv	-	-	-	-	-	-	2
n_estimators	-	250	-	-	-	100	-
C	-	-	100	-	-	-	-
Hidden_layer_sizes	-	-	-	(100,100,100)	-	-	-
Alpha	-	-	-	0.1	-	-	-
learning_rate	-	-	-	-	-	1.0	-
method	-	-	-	-	-	-	isotonic
random_state	-	-	-	-	-	0	-

Those parameters need to be understood if we want to improve the performance of the default algorithms. For example, in cases like MLPClassifier we can for example set up the number of neurons in each middle layer of the neural network, obtaining a better performance instead of by default configuration.

3.2.2.4 KFold-Cross Validation

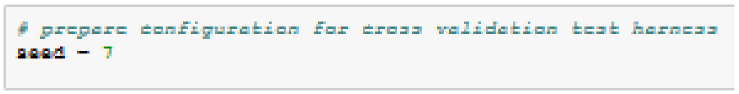
The complete dataset will be entered into a "kfold-cross validation" loop for 10 folds and repeated n times according to the number of classification algorithms created in the variables “models”, applying resampling (undersampling and oversampling explained before), reduction of dimensions, modelling techniques and plotting for each one of the different folds created. This means that for a given classification model, 10 folds are created and the preprocessing techniques are applied each time in a different fold, being evaluated as if they would be a new dataset and so on N times by each sorting algorithm selected. In figure 84 we show the steps followed at the beginning, first applying dimensionality reduction and then resampling



```
#------------ evaluate each model in turn with the use of K-cross validation
for name, model in models:
    kfold = cross_validation.KFold(n_samples,n_folds=10, shuffle= True, random_state=seed)
    results = []
    sensiMean = []
    specifiMean = []
    aucMean = []
    tprs = []
    aucs = []
    print results
    i = 0
        for train, test in kfold:
            #------------ DIMENSIONALITY REDUCTION

            pipeline = Pipeline([('scaling', StandardScaler()), ('pca', PCA(n_components=4))])

            #we use here directly the PCA algorithm obtained in the pipeline, therefore we just apply he methode
            X_train_Transformed = pipeline.fit_transform(data[train])
            X_test_Transformed = pipeline.transform(data[test])

            #------------ RESAMPLING

            # We apply here oversampling only in the training data only, not in the test
            featuresUn, targetUn = RandomUnderSampler(ratio=0.5,random_state=1).fit_sample(X_train_Transformed, targetnA:

            # We apply here oversampling only in the training data only, not in the test
```

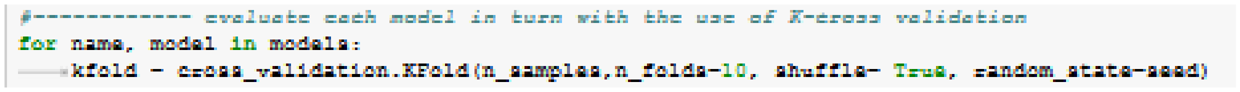
Figure 84. Code of the cross-validation procedure

For a fair comparison of the different algorithms, the procedure is configured with a random seed to ensure that the same splits for the training data are performed and each algorithm is evaluated in the same way[92]



```
# prepare configuration for cross validation test harness
seed = 7
```

Figure 85. Creation of the random seed



```
#------------ evaluate each model in turn with the use of K-cross validation
for name, model in models:
    kfold = cross_validation.KFold(n_samples,n_folds=10, shuffle= True, random_state=seed)
```

Figure 86. Creation of the cross validation with the seed

As was explained above, in the k-cross validation loop both preprocessing and modeling phases will be performed, and in this case, the order will be important and will be governed by predetermined patterns so that there is no inconsistency in the data or overfitting when creating models[93].

3.2.2.4.1 Dimensionality Reduction

Firstly, as it can be appreciated, dimensional reduction techniques have been performed to eliminate irrelevant features and to compact the dataset. It has been reduced from 18 to 4 features, eliminating 78% of the data and compacting the information in only 22% of the relevant data. We decided to reduce the dimensions to four dimensions previously, because when we were analyzing how much variance we would retain reducing the dimensions, with 4 features we were able to retain 60% of the data as we can see in figure 87, and the rest of the features were not able to retain much information, the variance was quite small.

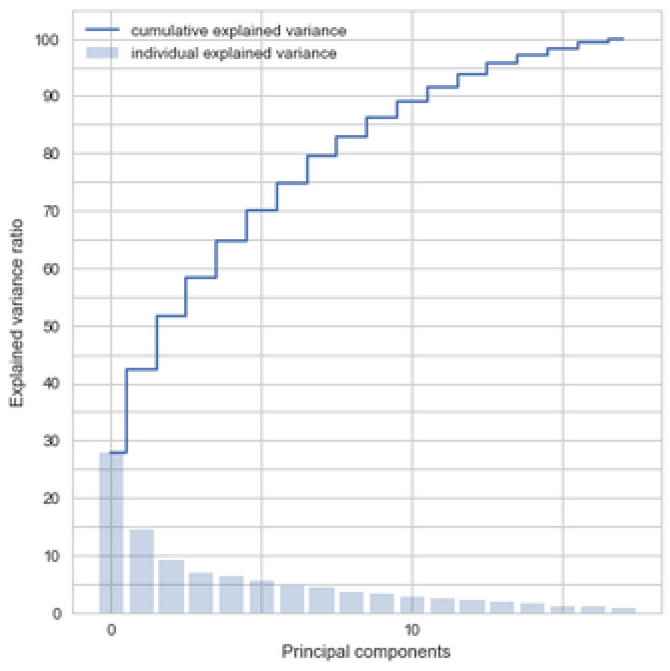
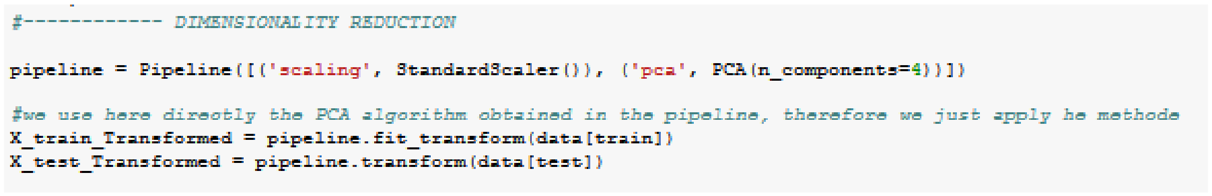


Figure 87. A graphic with the PC's and the explained variance ratio

This is a considerable reduction and prepares the way for the next step as we can see in figure 87.



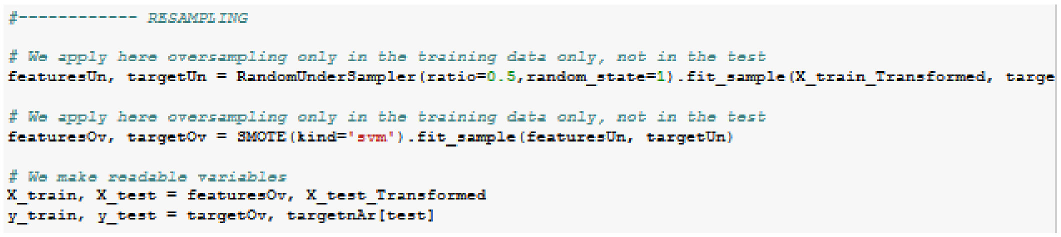
```
#------------ DIMENSIONALITY REDUCTION

pipeline = Pipeline([('scaling', StandardScaler()), ('pca', PCA(n_components=4))])

#we use here directly the PCA algorithm obtained in the pipeline, therefore we just apply he methode
X_train_Transformed = pipeline.fit_transform(data[train])
X_test_Transformed = pipeline.transform(data[test])
```

Figure 88. Dimensionality Reduction

3.2.2.4.2 Resampling and normalization

In our case, both techniques were used. First undersampling, trying to reduce the noise of the data, and then oversampling creating synthetic data, finally obtaining a balanced dataset with the same number of patients for each class.

In figure 89 is shown the different algorithms applied to process with the resampling of the dataset. In figure 90 we can appreciate in blue color the patients without a disease and in green color the patients with the disease. The techniques selected are SMOTE for oversampling, and for undersampling “RandomUndersampling”.



```
#------------ RESAMPLING

# We apply here oversampling only in the training data only, not in the test
featuresUn, targetUn = RandomUnderSampler(ratio=0.5,random_state=1).fit_sample(X_train_Transformed, targe

# We apply here oversampling only in the training data only, not in the test
featuresOv, targetOv = SMOTE(kind='svm').fit_sample(featuresUn, targetUn)

# We make readable variables
X_train, X_test = featuresOv, X_test_Transformed
y_train, y_test = targetOv, targetnAr[test]
```

Figure 89. Resampling: Undersampling and Oversampling

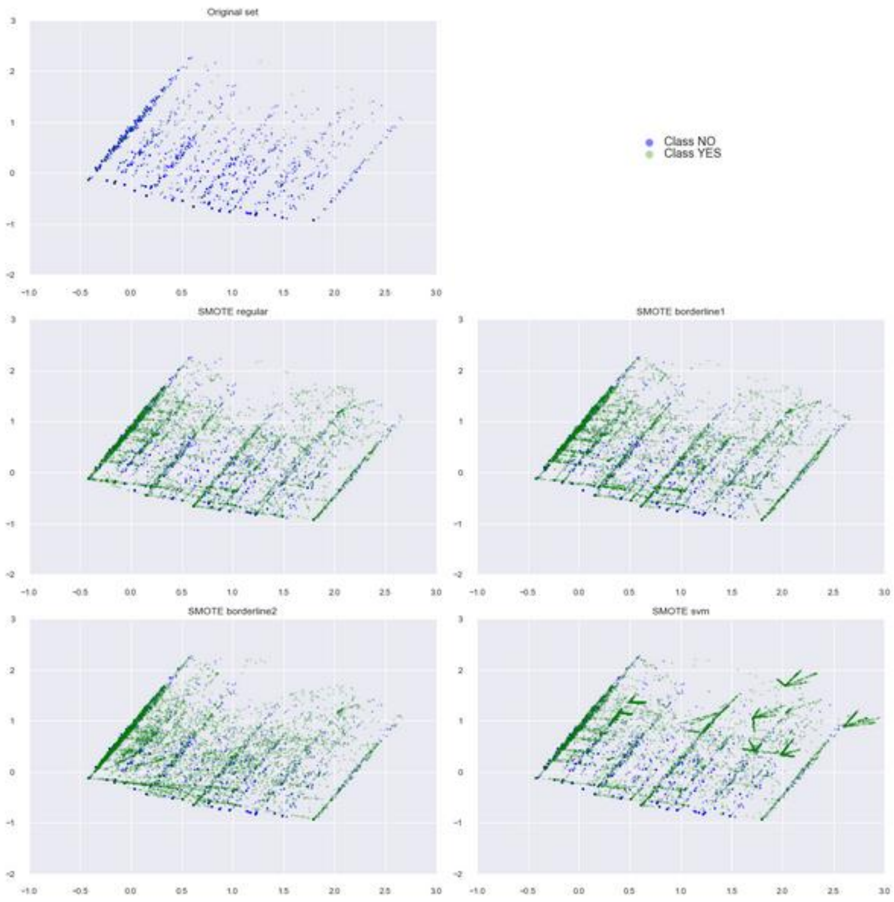
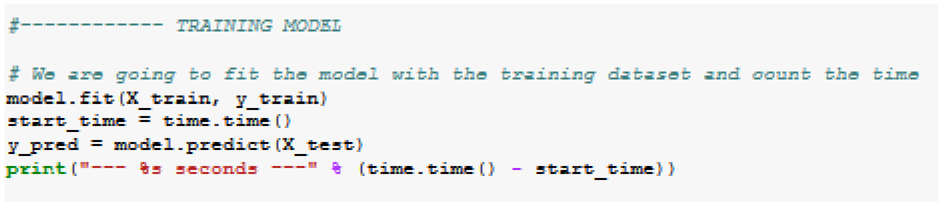


Figure 90. Oversampling with SMOTE with different patterns shown in four graphics. We can visualize the original dataset and the result after applying SMOTE

It was understood that we didn't normalize the data because the values are in the same range. At the beginning between 1 and 0, and after applying dimensionality reduction and resampling techniques, the values do not vary so much, and normalization was not needed.

3.2.2.4.3 Modelling

Finally, in figure 91 we can see how the classifier algorithms were modeled using the data obtained. The **Modelling** phase was repeated several times until the moment the results reached a degree of success.



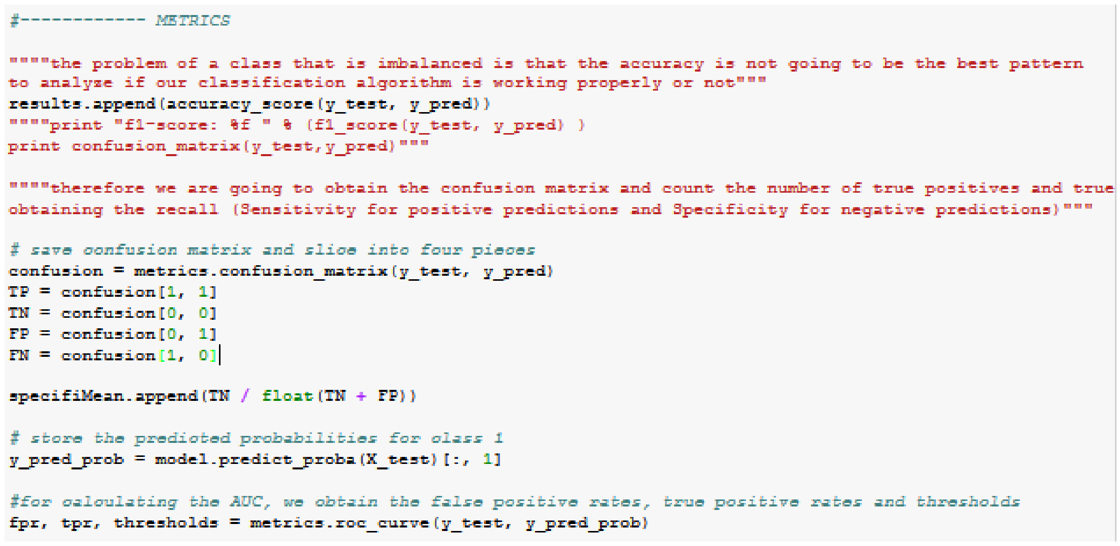
```
#------------ TRAINING MODEL

# We are going to fit the model with the training dataset and count the time
model.fit(X_train, y_train)
start_time = time.time()
y_pred = model.predict(X_test)
print("--- %s seconds ---" % (time.time() - start_time))
```

Figure 91. Modelling Data

We are not going to go deeper in the explanation because all the models were explained before and they are not the objectives of this project.

3.2.3 Evaluation Metrics

The evaluation metrics is going to be performed inside of the **K-cross validation**, but it belongs to another phase, therefore it was explained here. Since it is sought to solve a problem of binary classification, the appropriate technique to measure is the **AUC** or the F1-score. AUC was chosen, because the F1 score is applicable for any particular point in the ROC curve, representing only a particular threshold value (it is a smart way to represent recall and precision). In the other hand, the ROC curve allows us to obtain the AUC score, which includes integration over the whole range of precision/recall tradeoffs, it means, it is able to evaluate different levels of thresholds and many F scores values, instead of as just one point as would be the F1 score value. In figure 92 we can appreciate how do we obtain the different metrics in our experiment.



```
#------------ METRICS

"""the problem of a class that is imbalanced is that the accuracy is not going to be the best pattern
to analyze if our classification algorithm is working properly or not"""
results.append(accuracy_score(y_test, y_pred))
"""print "f1-score: %f " % (f1_score(y_test, y_pred) )
print confusion_matrix(y_test,y_pred)"""

"""therefore we are going to obtain the confusion matrix and count the number of true positives and true
obtaining the recall (Sensitivity for positive predictions and Specificity for negative predictions)"""

# save confusion matrix and slice into four pieces
confusion = metrics.confusion_matrix(y_test, y_pred)
TP = confusion[1, 1]
TN = confusion[0, 0]
FP = confusion[0, 1]
FN = confusion[1, 0]

specifiMean.append(TN / float(TN + FP))

# store the predicted probabilities for class 1
y_pred_prob = model.predict_proba(X_test)[:, 1]

#for calculating the AUC, we obtain the false positive rates, true positive rates and thresholds
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred_prob)
```

Figure 92. Obtaining the metrics.

Other parameters as Sensitivity, Specificity or accuracy are collected in an informative way because they are not the correct way of measuring binary classifier problem, but it gives us an idea how well is going to predict the discriminative class in our experiment. We could go deeper in this field, but it is not the aim of the experiment.

3.2.4 Deployment

In our case are the results obtained will be explained in the next section. The files used in this experiment will be attached with this Project. This phase is going to be developed in the next block.

Block 4

Results and analysis-discussion

This block will focus on explaining the results obtained in this experiment for each of the different dimensionality reduction algorithms and will subsequently perform an analysis-discussion of the same.

4.1 Results

For each case, it is going to find a summary of the results for each classification algorithms with the mean values obtained for Sensitivity, Specificity, AUC, and Accuracy. A chart, with a legend explaining the meaning of each element in our graphic, will clarify the elements of the graphic. In figure 93 we can find an example.

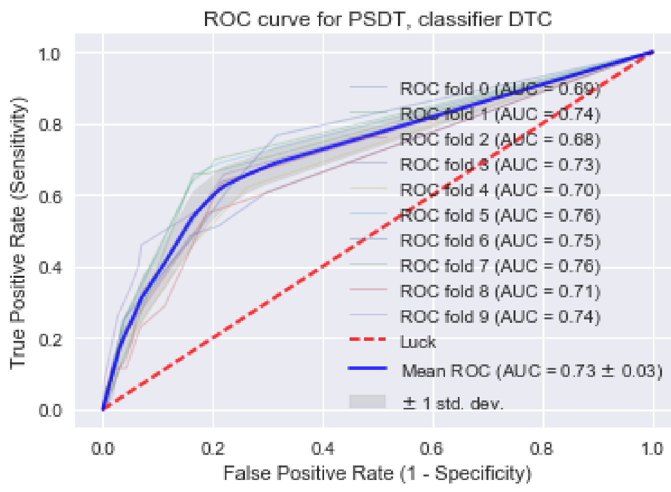


Figure 93. Result for each case

Each graphic represents the next points:

- On the top of the Graph we can find the evaluation metric used, the disease analyzed and the classifier selected.
- The false positive and true positive rates are expressed in the x and y axes, respectively.

- The best classifier can be found with the sensitive and specificity mean.
- In the graph, it will be found a legend which explains the meaning of the different elements that appear in it:
    - Several lines in soft- color represent the different results obtained in each one of the folds, indicating the value of the AUC obtained in that fold.
    - The value of luck is represented in red and with broken lines, which indicates the border for a classifier that is as good as random.
    - In a deep blue color, we will find the mean of the AUC value for all folds, bordered by a gray color which represents the standard deviation.

As well we are going to show a Boxplot with the representation of the AUC means. In figure 94 we can find an example.

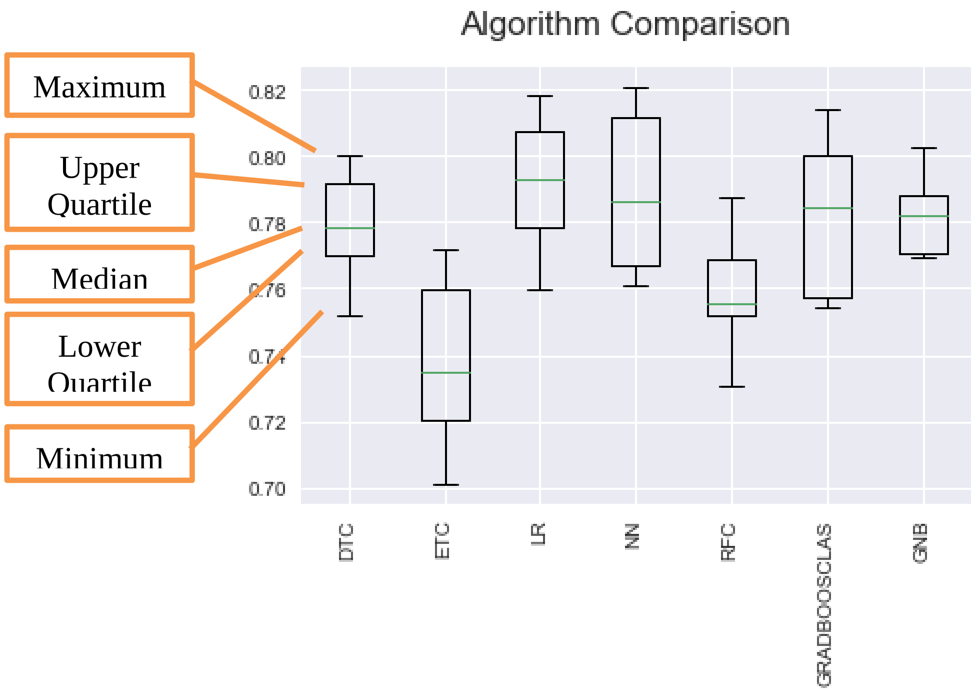


Figure 94. Bloxpot with the comparisons of the classification algorithms.

Each graphic represents the next points:

- The AUC range and the classification models are expressed in the y and x-axes, respectively.
- Each box is going to be composed by the next components:
  - Maximum: Greatest Auc value, excluding outliers.
  - Upper Quartile: 25% of the data greater than this value.
  - Median: 50% of the data greater than this value; middle of the Dataset. The green line is the mean Auc.
  - Lower Quartile: 25% of the data less than this value.
  - Minimum: Least value.

Finally, it is going to be shown a table with the results obtained. We will repeat the same steps for all the different algorithms.

We would like to point out that for all cases it has been taken as a threshold of 0.5 which is the default value.

4.1.1 Base Case: No Dimensionality Reduction applied

As we need to compare our results to measure the performance, the first situation we are going to find is the base case without performing any Dimensionality Reduction Algorithm.

- **Decision Tree Classifier**

```
o   Sensitivity mean: 0.653312
o   Specificity mean: 0.758691
o   AUC mean: 0.731695
o   Accuracy DTC: 0.750933 (0.033620)
```

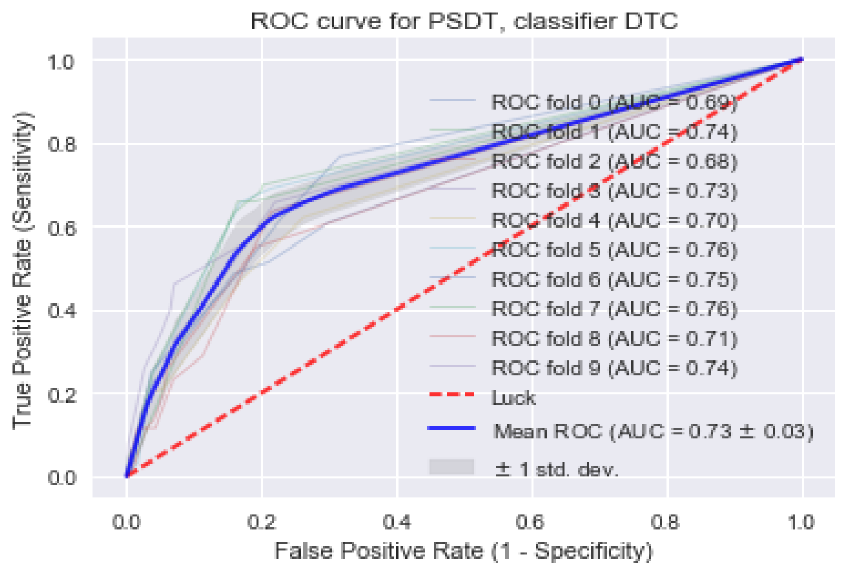


Figure 95. Mean ROC for Decision Tree classifier - No Dimensionality Reduction.

In Figure 95 we can appreciate that the Auc mean for Decision Tree classifier - No Dimensionality Reduction makes a “fair” discrimination with a value of 0.73. The discrimination is higher at the beginning for positive values, but after reaching the point with the best classifier, the predictions by negative values start to grow.

- **Extra Tree Classifier**

```
o  Sensitivity mean: 0.563670
o  Specificity mean: 0.792673
o  AUC mean: 0.731210
o  Accuracy ETC: 0.776240 (0.017553)
```

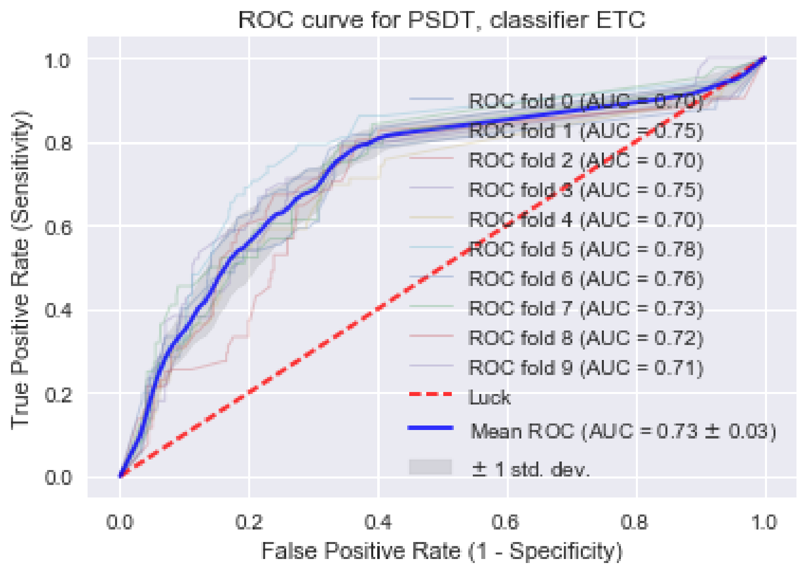


Figure 96. Mean ROC for Decision Tree classifier - No Dimensionality Reduction.

In figure 96 we can appreciate that the Auc mean for Decision Tree classifier – No Dimensionality Reduction makes a "fair" discrimination with a value of 0.73. The discrimination is higher at the beginning for positive values, before reaching the best classifier we can find some inconsistencies but after reaching the point with the best classifier, the negative predictions start to grow.

- **Logistic Regression Classifier**

```
o  Sensitivity mean: 0.678457
o  Specificity mean: 0.763701
o  AUC mean: 0.782738
o  Accuracy LR: 0.756974 (0.014506)
```

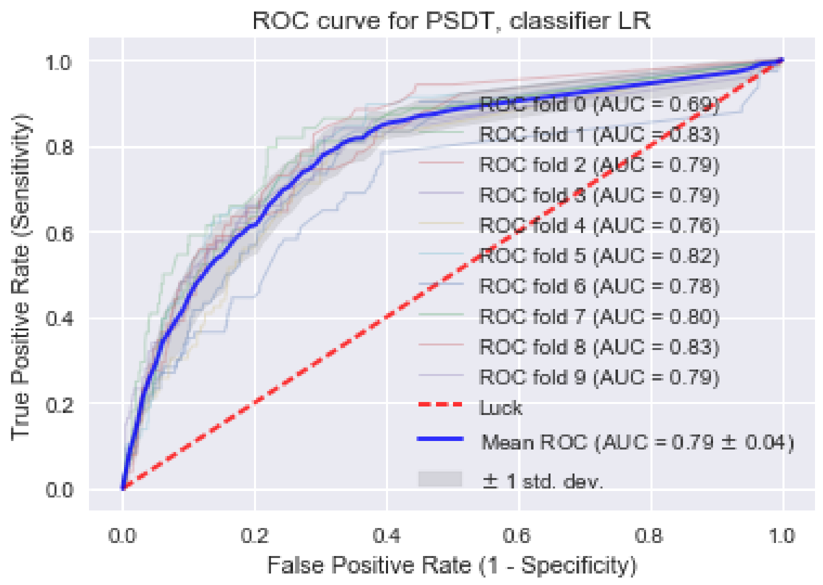


Figure 97. Mean ROC for Logistic Regression classifier- No Dimensionality Reduction.

In figure 97 we can appreciate that the Auc mean for Logistic Regression classifier- No Dimensionality Reduction makes a "fair-good" discrimination with a value of 0.78. The discrimination is higher at the beginning for positive values, before reaching the best classifier we can find some inconsistencies but after reaching the point with the best classifier, the negative predictions start to grow.

- **Neural Network Classifier**

```
o  Sensitivity mean: 0.633677
o  Specificity mean: 0.764011
o  AUC mean: 0.749987
o  Accuracy NN: 0.754384 (0.031358)
```

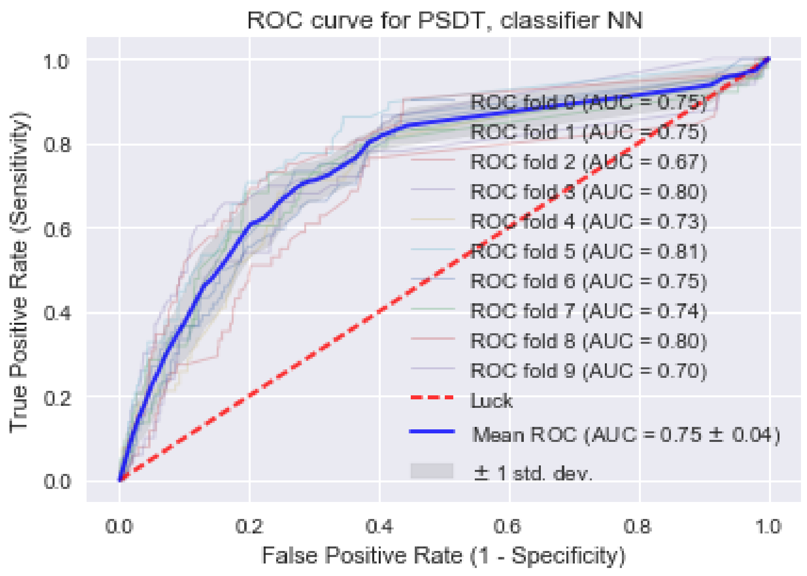


Figure 98. Mean ROC for Neural Network classifier - No Dimensionality Reduction.

In figure 98 we can appreciate that the Auc mean for Neural Network classifier - No Dimensionality Reduction makes a “fair” discrimination with a value of 0.75. The discrimination is higher at the beginning for positive values, before and after reaching the best classifier we can find some inconsistencies, it looks like the classifier can discriminate properly. After reaching the point with the best classifier, the negative predictions start to grow.

- **Random Forest Classifier**

```
o  Sensitivity mean: 0.596784
o  Specificity mean: 0.773315
o  AUC mean: 0.734712
o  Accuracy RFC: 0.760682 (0.018889)
```

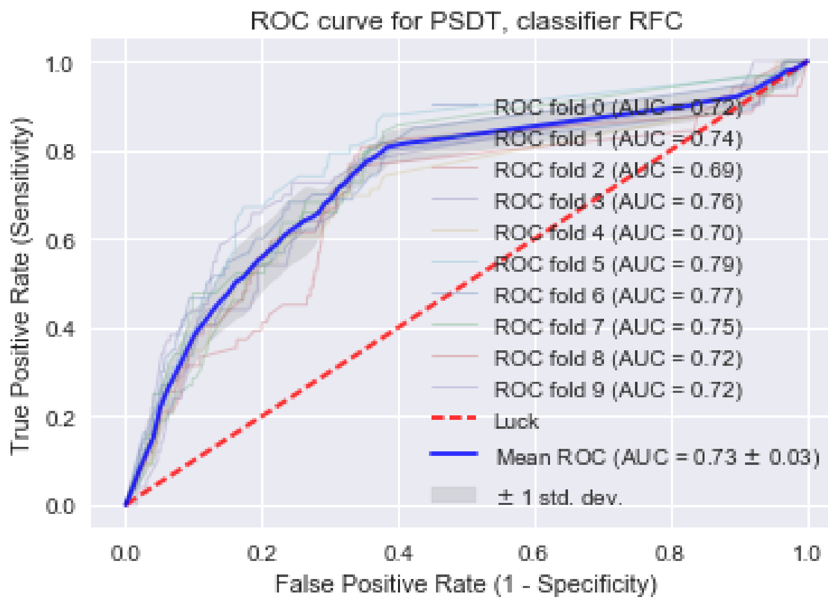


Figure 99. Mean ROC for Random Forest classifier - No Dimensionality Reduction.

In figure 99 we can appreciate that the Auc mean for Random Forest classifier - No Dimensionality Reduction makes a “fair” discrimination with a value of 0.73. The discrimination is higher at the beginning for positive values, before and after reaching the best classifier we can find some inconsistencies, it looks like the classifier can discriminate properly. After reaching the point with the best classifier, the negative predictions start to grow.

- **Gradient Boosting Classifier**

```
o  Sensitivity mean: 0.609569
o  Specificity mean: 0.800046
o  AUC mean: 0.777659
o  Accuracy GRADBOOSCLAS: 0.785748 (0.015748)
```

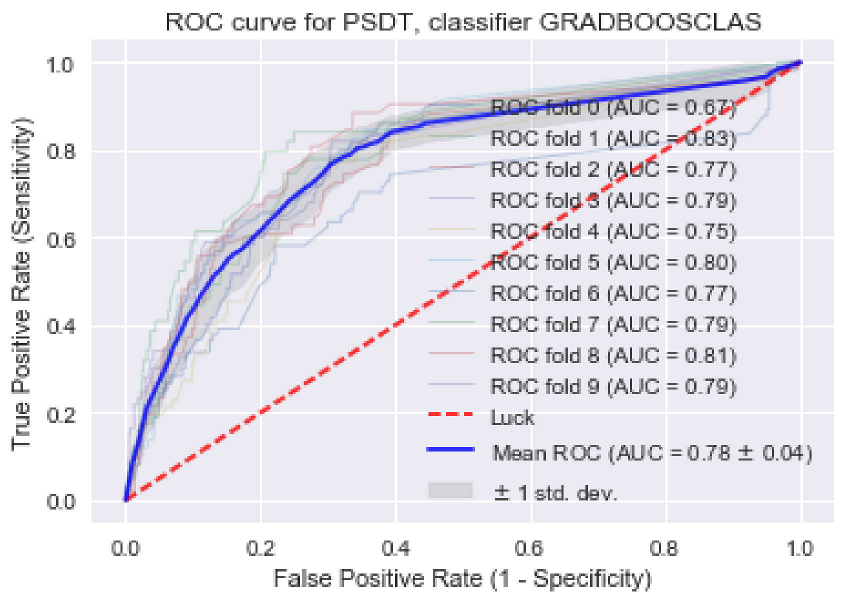


Figure 100. Mean ROC for Gradient Boosting classifier - No Dimensionality Reduction.

In figure 100 we can appreciate that the Auc mean for Gradient Boosting classifier - No Dimensionality Reduction makes a "fair" discrimination with a value of 0.77. The discrimination is higher at the beginning for positive values. After reaching the point with the best classifier, the negative prediction starts to grow.

- **Gaussian NB Classifier**

```
o  Sensitivity mean: 0.828677
o  Specificity mean: 0.634184
o  AUC mean: 0.791134
o  Accuracy GNB: 0.647819 (0.058135)
```

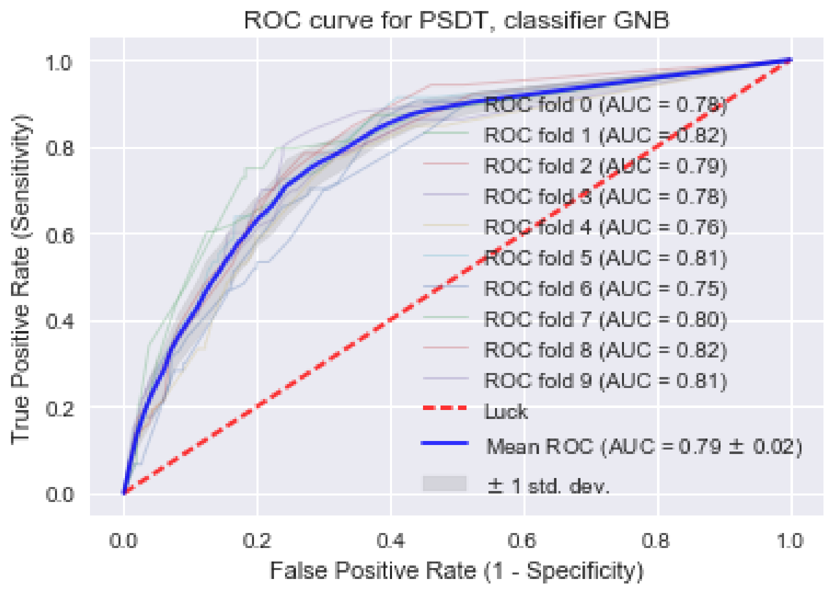


Figure 101. Mean ROC for Gaussian Naïve Bayes classifier - No Dimensionality Reduction.

In figure 101 we can appreciate that the Auc mean for Gaussian Naïve Bayes classifier - No Dimensionality Reduction makes a "good" discrimination with a value of 0.79. The discrimination is higher at the beginning for positive values, and continue growing until reach the point for the best classifier. After reaching the point with the best classifier, the negative prediction starts to grow.

- **Plotbox Algorithm Comparison.**

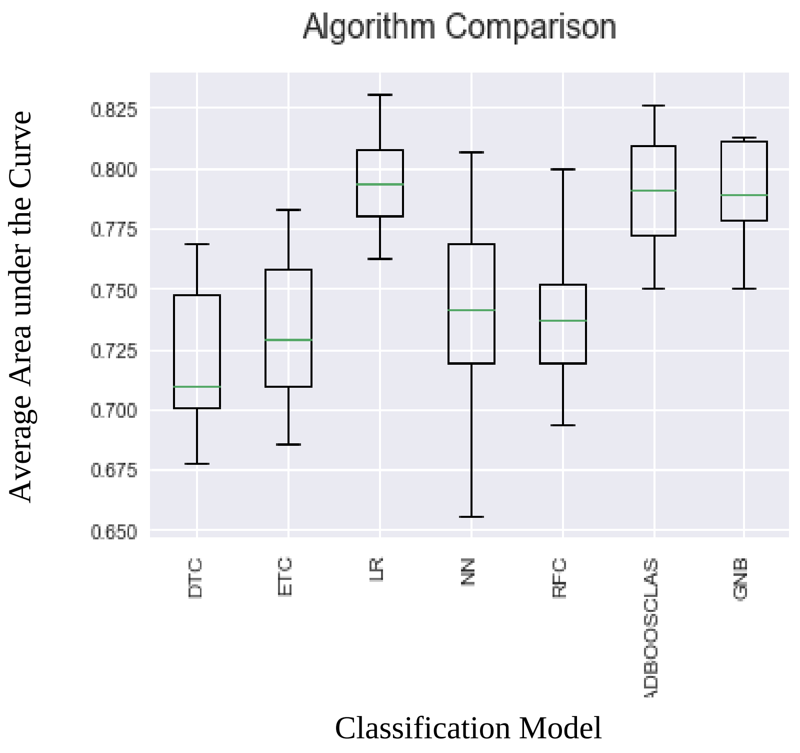


Figure 102. Comparison of classification algorithms without Dimensionality Reduction.

In figure 102 we can appreciate the Comparison between the different AUC values obtained from the classification algorithms. DTC, ETC, NN, and RFC are bigger compared with LR, GRADBOOSCLAS AND GNB. This suggests the AUC of the first group are distributed in a bigger range, increasing their variance and at the same time showing a poorer result than those belonging to the second group. In the second group, the median is similar in the three cases but GNB values are distributed in a smaller range, with the box and the upper whisker very close, representing 75% of the data. The second group is going to have the best models for this case, with GNB as the best model.

- **Table with results of this experiment for no DR**

In table 3, we are able to see the results for each classification model on the whole dataset with 18 features. We are going to focus our attention in the Area under the Curve and the sensitivity. As we can realize here the algorithm the one performs better is the GNB for no Dimensionality reduction and in second place we can find the LR algorithm. This table is going to be our base and we are going to use it to compare how good works the new subspaces obtained with the DR algorithms with only 4 features.

Table 3. Comparison table with the results for each Classification model without Dimensionality reduction

Model	AUC	SENSITIVITY	SPECIFICITY	ACCURACY
DTC	0.731695	0.653312	0.758691	0.750933
ETC	0.731210	0.563670	0.792673	0.776240
LR	0.782738	0.678457	0.763701	0.756974
NN	0.749987	0.633677	0.764011	0.754384
RFC	0.734712	0.596784	0.773315	0.760682
GBC	0.777659	0.609569	0.800046	0.785748
GNB	0.791134	0.634184	0.828677	0.647819

4.1.2 PCA

The first case we are going to analyze is the feature extraction algorithm PCA.

- **Decision Tree Classifier**

```
o Sensitivity mean: 0.800921
o Specificity mean: 0.656946
o AUC mean: 0.773194
o Accuracy DTC: 0.666698 (0.036368)
```

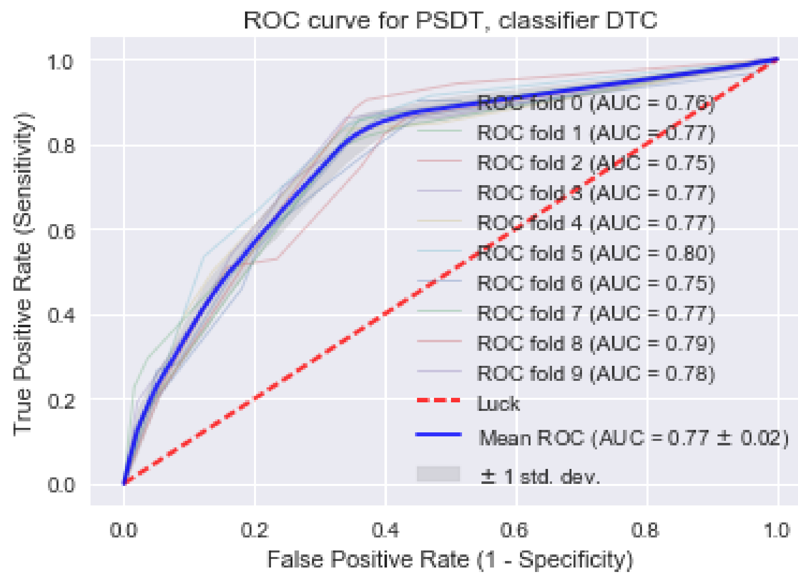


Figure 103. Mean ROC for Decision Tree classifier – PCA.

In figure 103 we can appreciate that the Auc mean for the pair Decision Tree classifier - PCA makes a “fair” discrimination with a value of 0.77. The discrimination is higher at the beginning for positive values, and continue growing after reaching the point for the best classifier. After reaching the point with the best classifier, the negative prediction starts to grow.

- **Extra Tree Classifier**

```
o Sensitivity mean: 0.644840
o Specificity mean: 0.759546
o AUC mean: 0.738591
o Accuracy ETC: 0.751177 (0.017062)
```

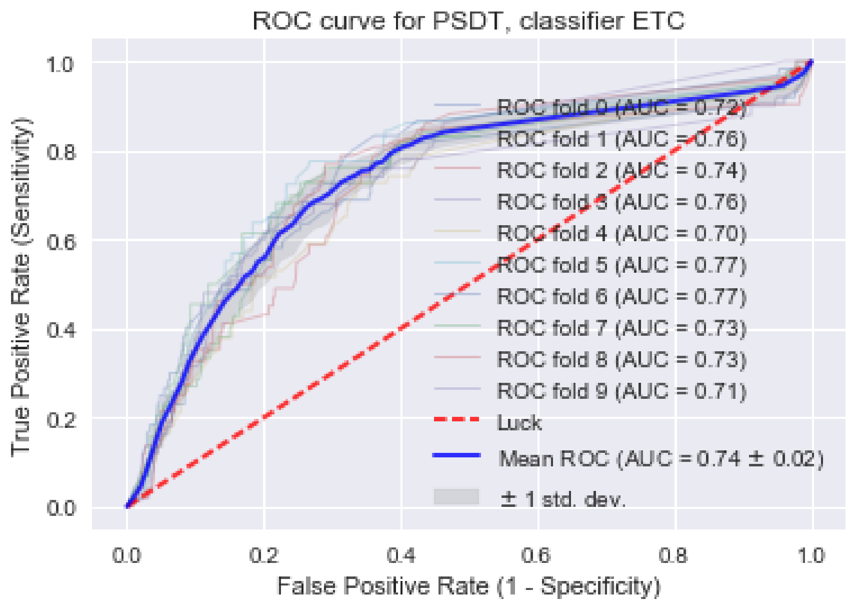


Figure 104. Mean ROC for Extra Tree classifier – PCA.

In figure 104 we can appreciate that the Auc mean for the pair Extra Tree classifier - PCA reduction makes a “fair” discrimination with a value of 0.73. The discrimination is higher at the beginning for positive values, reaching 0.6 sensitivity quite fast. After this point, it continues growing to reach the best classifier point but decreasing the growth. After reaching the point with the best classifier, the negative prediction starts to grow, even beyond the limit of Luck at the end.

- **Logistic Regression Classifier**

```
o Sensitivity mean: 0.612776
o Specificity mean: 0.808554
o AUC mean: 0.791019
o Accuracy LR: 0.794021 (0.014047)
```

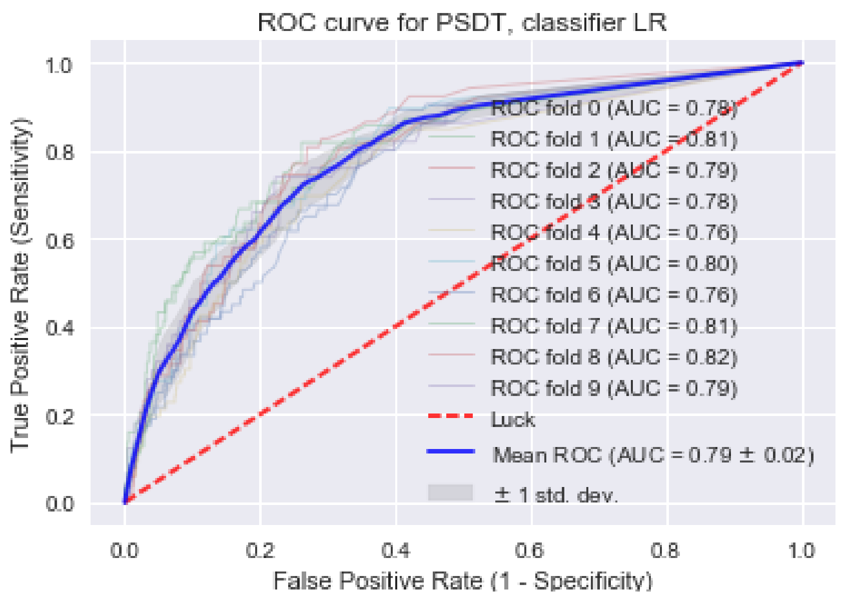


Figure 105. Mean ROC for Extra Tree classifier – PCA.

In figure 105 we can appreciate that the Auc mean for the pair for Extra Tree classifier - PCA reduction makes a "good" discrimination with a value of 0.79. The discrimination is higher at the beginning for positive values, reaching 0.8 sensitivity quite fast. After reaching the point with the best classifier, the negative prediction starts to grow and decrease the positive predictions.

- **Neural Network Classifier**

```
o  Sensitivity mean: 0.760683
o  Specificity mean: 0.709322
o  AUC mean: 0.788770
o  Accuracy NN: 0.713017 (0.021141)
```

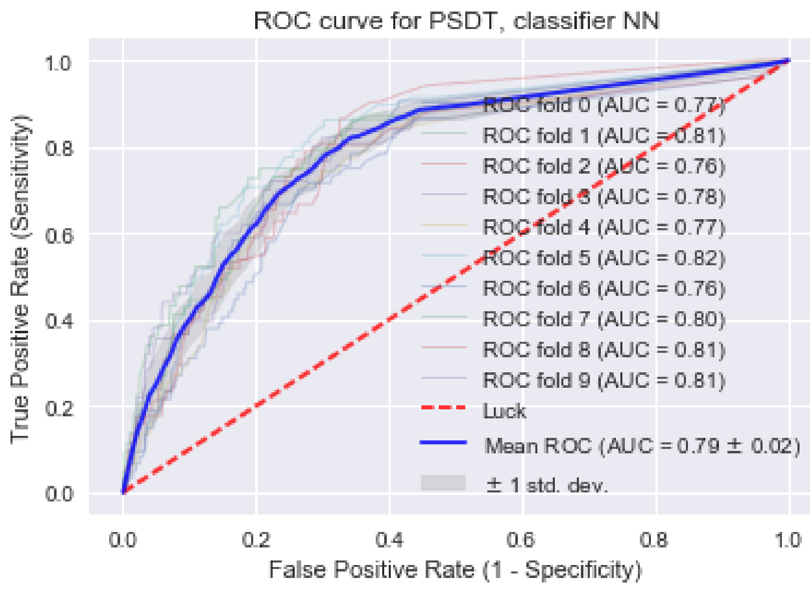


Figure 106. Mean ROC for Neural Network classifier – PCA.

In figure 106 we can appreciate that the Auc mean for the pair Neural Network classifier - PCA reduction makes a "good" discrimination with a value of 0.79. The discrimination is higher at the beginning for positive values, reaching 0.8 sensitivity quite fast. After reaching the point with the best classifier, the negative prediction starts to grow and decrease the positive predictions. Results are similar with LR, but the change between both phases is clearer.

- **Random Forest Classifier**

```
o  Sensitivity mean: 0.630653
o  Specificity mean: 0.769692
o  AUC mean: 0.761680
o  Accuracy RFC: 0.759451 (0.016666)
```

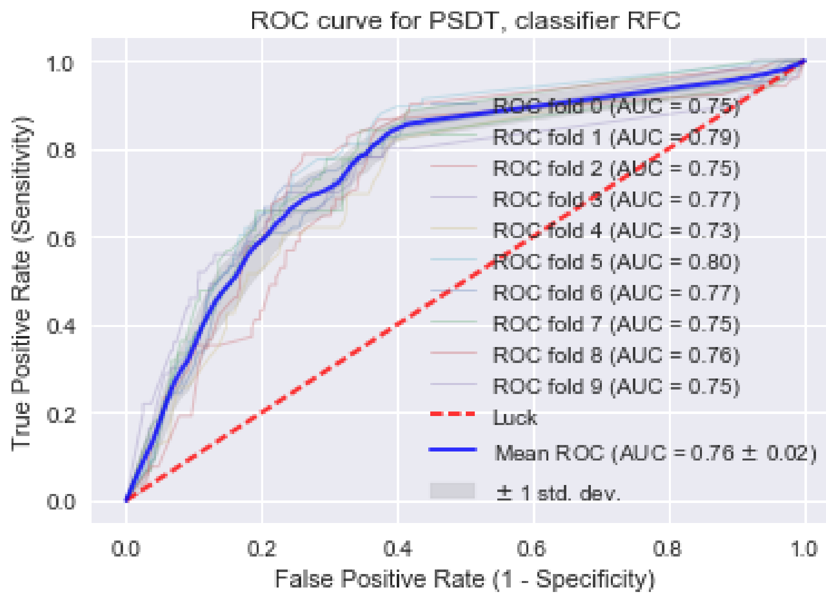


Figure 107. Mean ROC for Random Forest classifier – PCA.

In figure 107 we can appreciate that the Auc mean for the pair Random Forest classifier - PCA reduction makes a "fear" discrimination with a value of 0.76. The discrimination is higher at the beginning for positive values, but in the middle of the curve, we can find inconsistencies. After reaching the point with the best classifier, the curve decreases and grows, eventually ending with an increase in negative predictions.

- **Gradient Boosting Classifier**

```
o Sensitivity mean: 0.727397
o Specificity mean: 0.726887
o AUC mean: 0.781235
o Accuracy GRADBOOSCLAS: 0.726722 (0.030344)
```

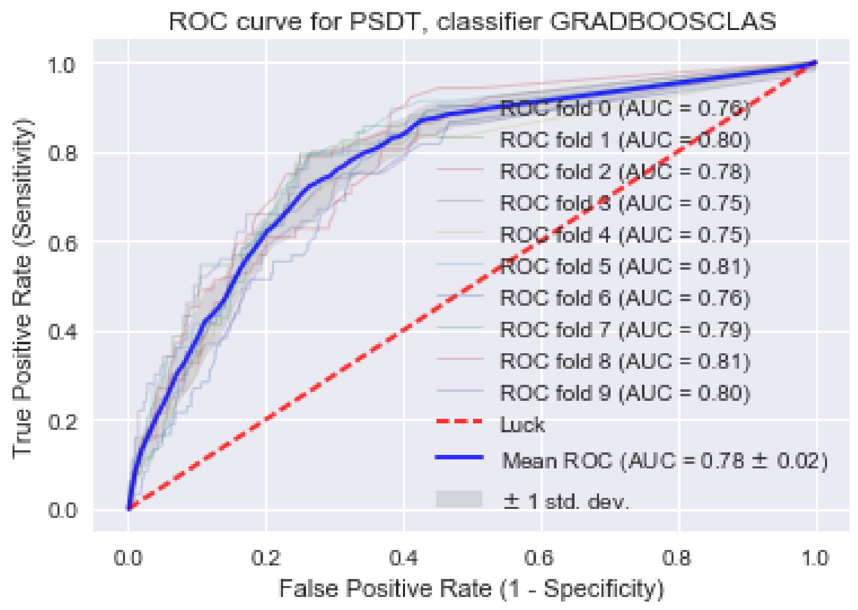


Figure 108. Mean ROC for Gradient Boosting classifier – PCA.

In figure 108 we can appreciate that the Auc mean for the pair Gradient Boosting classifier - PCA reduction makes a "fair-good" discrimination with a value of 0.78. The discrimination is higher at the beginning for positive values, with the sensitivity and specificity mean quite similar. After reaching the point with the best classifier, the negative prediction starts to grow and decrease the positive predictions.

- **Gaussian NB Classifier**

```
o  Sensitivity mean: 0.804367
o  Specificity mean: 0.622194
o  AUC mean: 0.775913
o  Accuracy GNB: 0.635213 (0.021522)
```

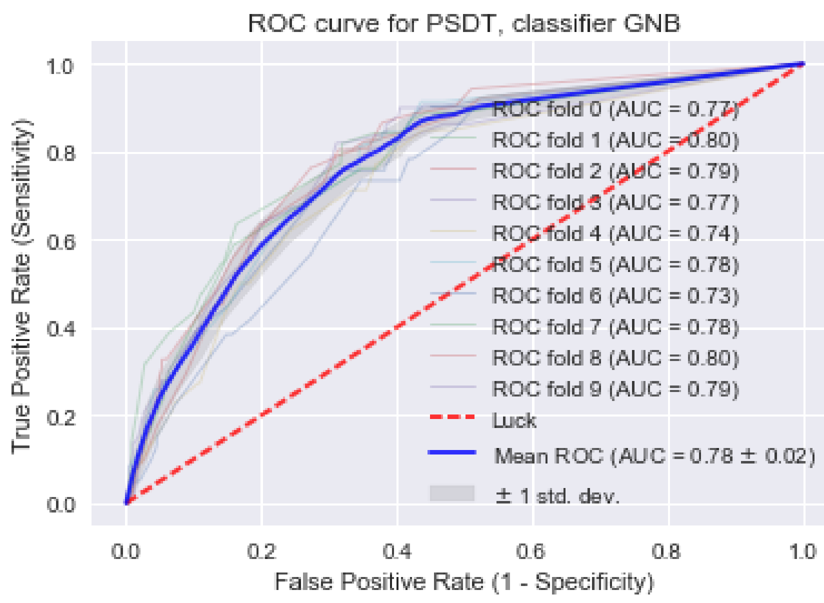


Figure 109. Mean ROC for Gaussian Naïve Bayes classifier – PCA.

In figure 109 we can appreciate that the Auc mean for the pair Gaussian Naïve Bayes classifier - PCA makes a "fair" discrimination with a value of 0.77. The discrimination is higher at the beginning for positive values, but after reaching the point with the best classifier, the negative predictions start to grow.

- **Plotbox Algorithm Comparison.**

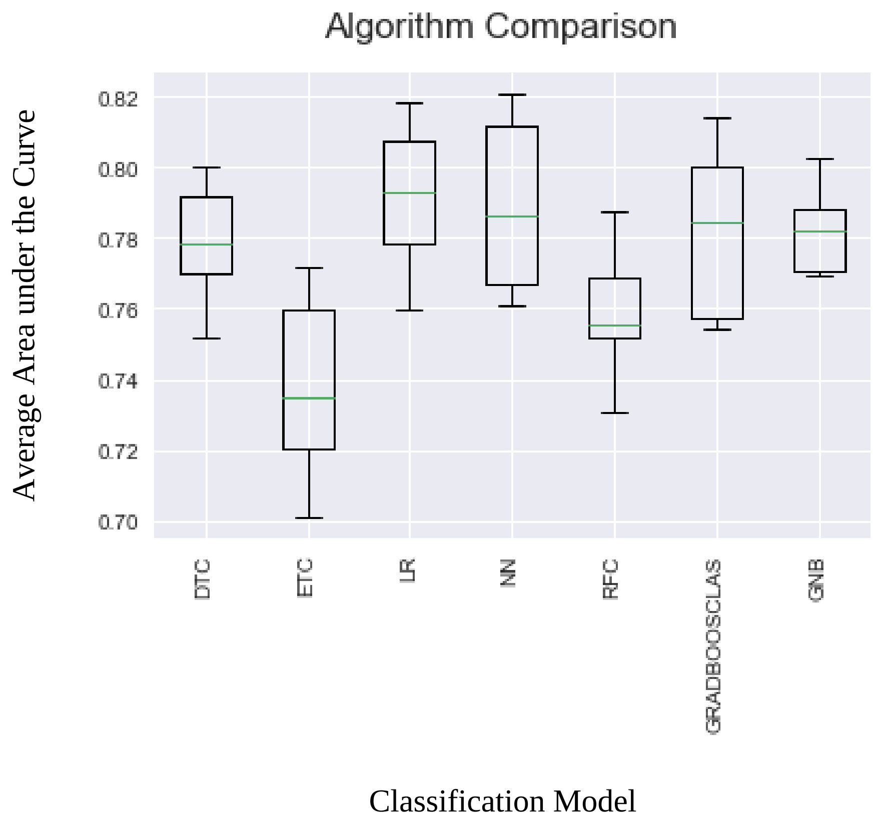


Figure 110. Comparison of classification algorithms with PCA.

In figure 110 we can appreciate the comparison between the different AUC values obtained for the classification algorithms. ETC results are quite low in comparison with the rest of the algorithms. DTC, RFC, GRADBOOSCLAS, and GNB have their boxes in arrange under 0.80, lower in comparison with LR and NN. This suggests the AUC of the first group are distributed in a lower range, showing a poorer result than those belonging to the second group. In the third group, the middle box exceeds 0.8 and the median is bigger in comparison with the second group. LR values are distributed in a smaller range in compare with NN, with the box and the upper whisker closer, representing 75% of the data. The third group is going to have the best models for this case, with LR as the best model.

- **Table with results of this experiment for PCA**

In table 4, we are able to see the results for each classification model in the new subspace created with PCA. This subspace is going to be created with 4 features and we are going to focus our attention in the Area under the Curve and the sensitivity. As we can realize here the algorithm the one performs better is the LR for no Dimensionality reduction and in second place we can find the NN algorithm.

Table 4. Comparison table with the results for each Classification model with PCA

Model	**AUC**	**SENSITIVITY**	**SPECIFICITY**	**ACCURACY**
DTC	0.773194	0.800921	0.656946	0.666698
ETC	0.738591	0.644840	0.759546	0.751177
LR	0.791019	0.612776	0.808554	0.794021
NN	0.788770	0.760683	0.709322	0.713017
RFC	0.761680	0.630653	0.769692	0.759451
GBC	0.781235	0.727397	0.726887	0.726722
GNB	0.775913	0.804367	0.622194	0.635213

4.1.3 LDA

The third case is going to analyze the performance of the feature extraction algorithm LDA.

- **Decision Tree Classifier**

```
o  Sensitivity mean: 0.784193
o  Specificity mean: 0.680089
o  AUC mean: 0.782340
o  Accuracy DTC: 0.687329 (0.020675)
```

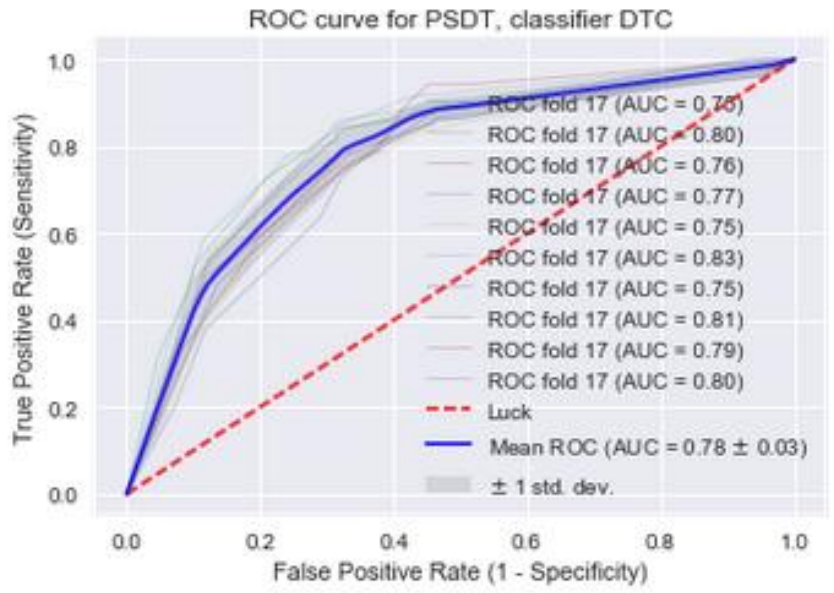


Figure 111. Mean ROC for Decision Tree classifier – LDA.

In figure 111 we can appreciate that the Auc mean for the pair Decision Tree classifier - LDA makes a "fair-good" discrimination with a value of 0.78. It can be appreciated three different phases. In the first one, the discrimination is higher at the beginning for positive values, but then growth starts to decline. After reaching the point with the best classifier, the predictions by negative values start to grow.

- **Extra Tree Classifier**

```
o  Sensitivity mean: 0.609136
o  Specificity mean: 0.792978
o  AUC mean: 0.741359
o  Accuracy ETC: 0.779448 (0.018464)
```

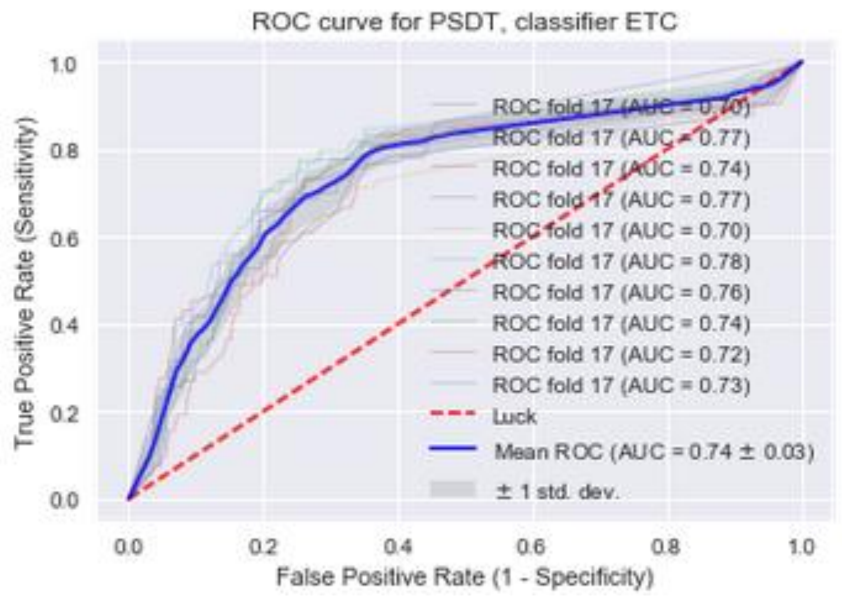


Figure 112. Mean ROC for Extra Tree classifier - LDA.

In figure 112 we can appreciate that the Auc mean for the pair Extra Tree classifier - LDA makes a “fair” discrimination with a value of 0.74. The discrimination is higher at the beginning for positive values, reaching 0.6 sensitivity quite fast. After this point, it continues growing, reaching the best classifier point but decreasing the growth. After reaching the point with the best classifier, the negative prediction starts to grow, even beyond the limit of Luck at the end.

- **Logistic Regression Classifier**

```
o  Sensitivity mean: 0.668888
o  Specificity mean: 0.766367
o  AUC mean: 0.791076
o  Accuracy LR: 0.759079 (0.016083)
```

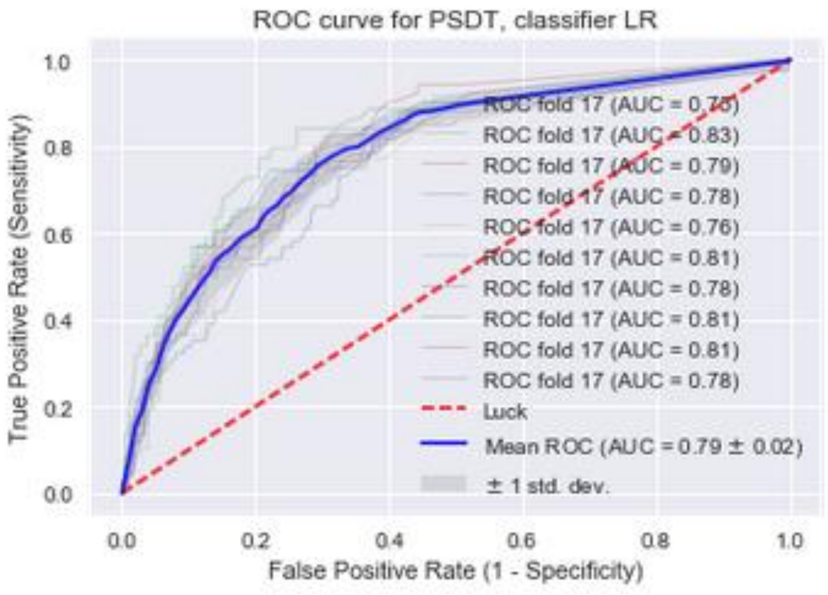


Figure 113. Mean ROC for pair Logistic Regression classifier – LDA.

In figure 113 we can appreciate that the Auc mean for the pair Logistic Regression classifier - LDA makes a "good" discrimination with a value of 0.79. We can differentiate three types of phases. Until we reach 0.4 of sensitivity the discrimination is higher for positive values, but then a change in growth happens, becoming more stable and from 0.85 the growth becomes almost horizontal increasing the negative predictions.

- **Random Forest Classifier**

```
o  Sensitivity mean: 0.592789
o  Specificity mean: 0.784940
o  AUC mean: 0.744956
o  Accuracy RFC: 0.770809 (0.015454)
```

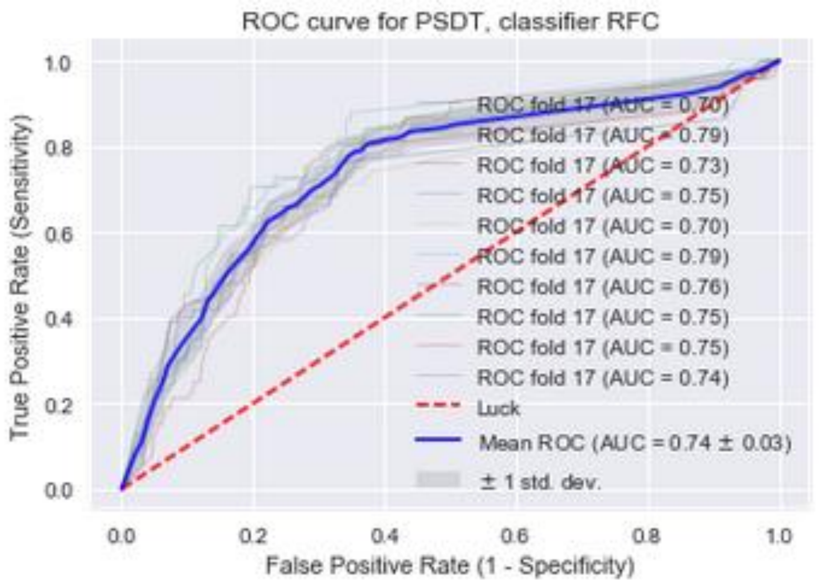


Figure 114. Mean ROC for Random Forest Classifier – LDA.

In figure 114 we can appreciate that the Auc mean for the pair Random Forest Classifier - LDA makes a “fair” discrimination with a value of 0.74. The curve grows steadily predicting positive values until it reaches 0.8 of sensitivity, then there is an increase in predictions of negative values.

- **Gradient Boosting Classifier**

```
o  Sensitivity mean: 0.716820
o  Specificity mean: 0.718947
o  AUC mean: 0.771401
o  GRADBOOSCLAS: 0.718319 (0.024142)
```

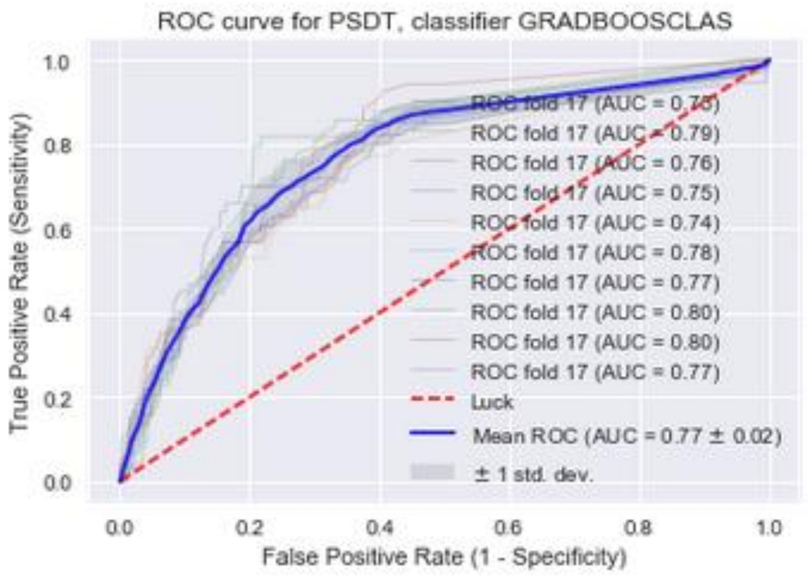


Figure 115. Mean ROC for GRADBOOS Classifier – LDA.

In figure 115 we can appreciate that the Auc mean for the pair GRADBOOS Classifier - LDA makes a "fair" discrimination with a value of 0.77. The curve grows steadily predicting positive values until it reaches 0.6 sensitivity. Then it starts to stabilize the growth and after reaching 0.85 of sensitivity, we can find an increase of negative predictions.

- **Gaussian NB Classifier**

```
o  Sensitivity mean: 0.816249
o  Specificity mean: 0.604423
o  AUC mean: 0.781445
o  Accuracy GNB: 0.618779 (0.063153)
```

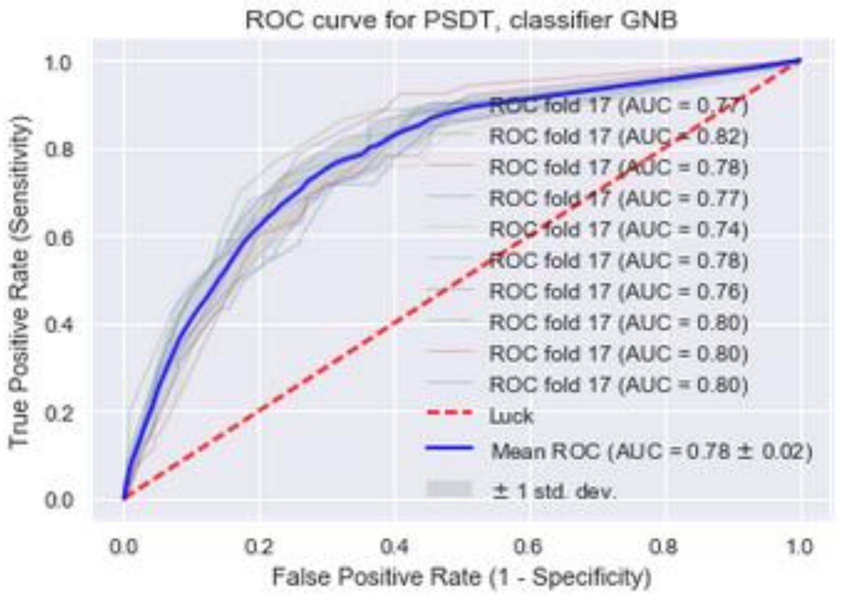
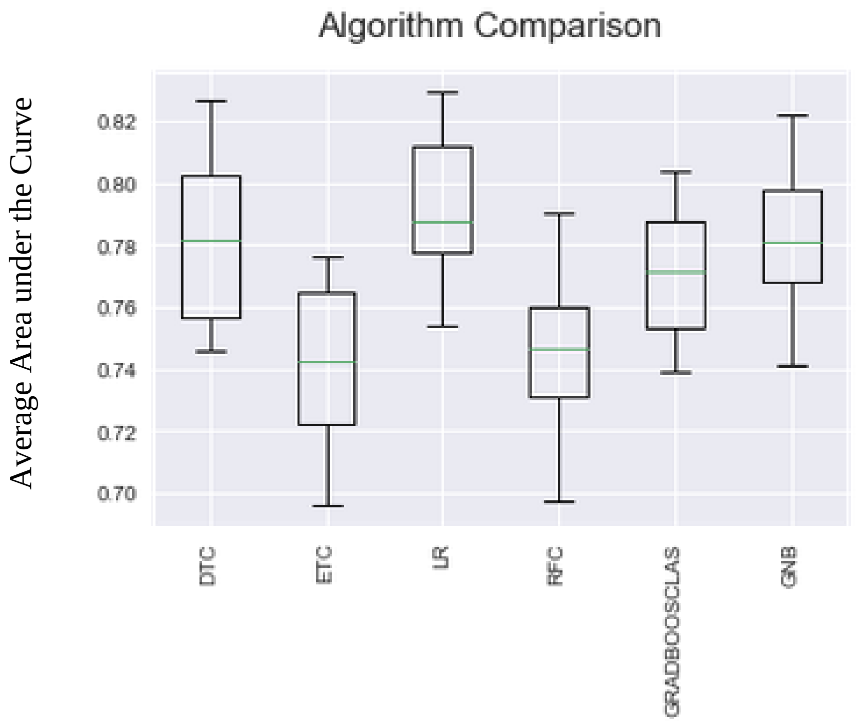


Figure 116. Mean ROC for Gaussian Naïve Bayes Classifier – LDA.

In figure 116 we can appreciate that the Auc mean for the pair Gaussian Naïve Bayes Classifier - LDA makes a "fair" discrimination with a value of 0.78. The curve grows steadily predicting positive and negative values in the first half, and increasing the growth of negative prediction in the second half, becoming almost a horizontal growth.

- **Plotbox Algorithm Comparison.**



Classification Model

Figure 117. Comparison of classification algorithms with LDA

In figure 117 we can appreciate the Comparison between the different AUC values obtained for the classification algorithms. ETC and RFC results are quite low in comparison with the rest of the algorithms. GRADBOOSCLAS, GNB, and DTC have their boxes in a range under or close to 0.80, lower in comparison with LR. This suggests the AUC of the second group are distributed in a lower range, showing a poorer result in compare to LR. In the second group, the medians of DTC and GNB are closer to the median of LR, exceeding 0.78. LR values are distributed in a smaller range, and with a higher Auc value, leading the classification as the best model.

- **Table with results of this experiment for LDA**

  In the table, we are able to see the results for each classification model in the new subspace created with LDA. This subspace is going to be created with 4 features and we are going to focus our attention in the Area under the Curve and the sensitivity.

  As we can realize here the algorithm the one performs better is the LR for no Dimensionality reduction and in second place we can find the DTC algorithm.

Table 5. Comparison table with the results for each Classification model with LDA

Model	AUC	SENSITIVITY	SPECIFICITY	ACCURACY
DTC	0.782340	0.784193	0.680089	0.687329
ETC	0.741359	0.609136	0.792978	0.779448
LR	0.791076	0.668888	0.766367	0.759079
RFC	0.744956	0.592789	0.784940	0.770809
GBC	0.771401	0.716820	0.718947	0.718319
GNB	0.781445	0.816249	0.604423	0.618779

4.1.4 Low variance

The fourth case to analyze is the feature selection algorithm Low Variance.

- **Decision Tree Classifier**

```
o  Sensitivity mean: 0.662864
o  Specificity mean: 0.728164
o  AUC mean: 0.727408
o  Accuracy DTC: 0.723130 (0.040228)
```

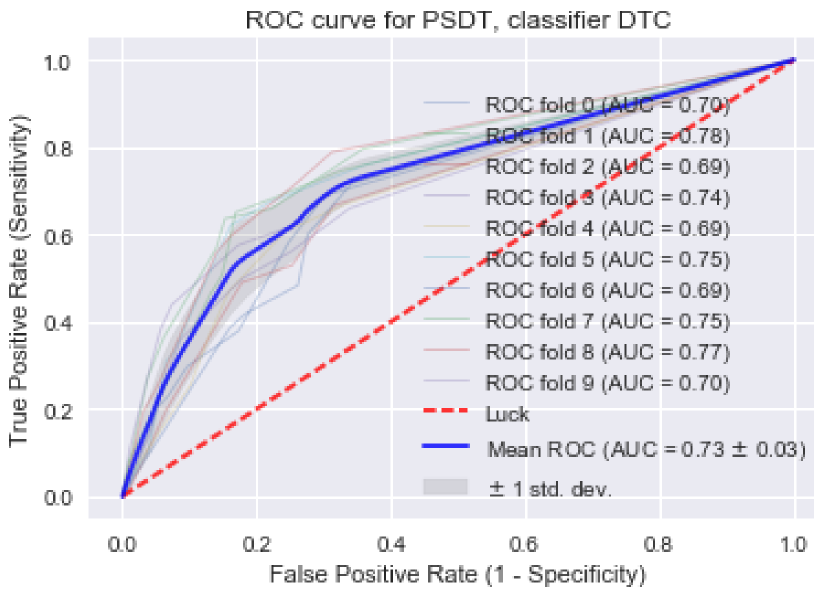


Figure 118. Mean ROC for Decision Tree Classifier – Low Variance.

In figure 118 we can appreciate that the Auc mean for the pair Decision Tree Classifier – Low Variance makes a "fair" discrimination with a value of 0.72. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.55 value of sensitivity. In the second phase, we can see that when approaching to the best classification point makes suddenly a change and grows negatively to re-grow again positively. Finally, from 0.7 of sensitivity onwards, the predicted negative elements prevail increasing the horizontal growth.

- **Extra Tree Classifier**

```
o  Sensitivity mean: 0.693760
o  Specificity mean: 0.699250
o  AUC mean: 0.736028
o  Accuracy ETC: 0.698804 (0.027076)
```

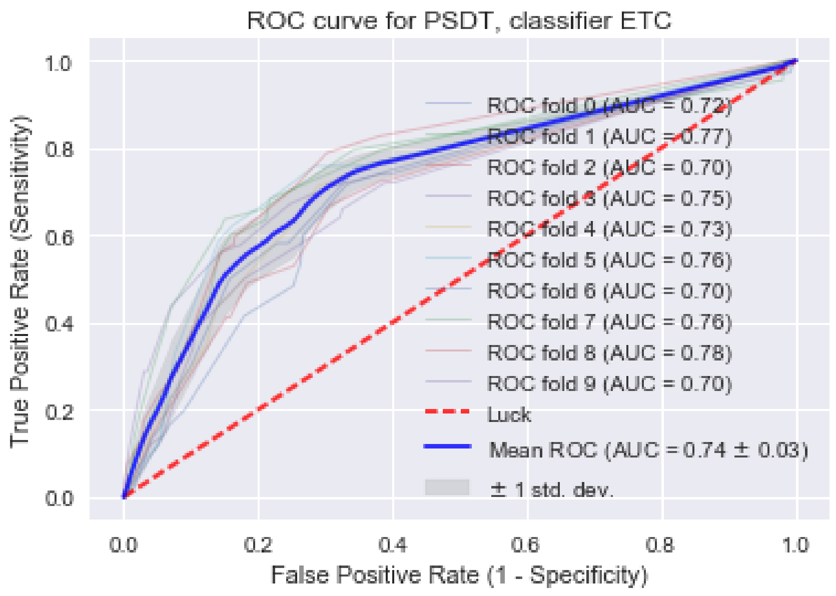


Figure 119. Mean ROC for Extra Tree Classifier – Low Variance.

In figure 119 we can appreciate that the Auc mean for the pair Extra Tree Classifier – Low Variance makes a "fair" discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.55 value of sensitivity. In the second phase, we can see that when approaching to the best classification point, the curve makes suddenly a change and grows negatively to re-grow again positively. Finally, from 0.75 of sensitivity onwards, the predicted negative elements prevail increasing the horizontal growth.

- **Logistic Regression Classifier**

```
o  Sensitivity mean: 0.640049
o  Specificity mean: 0.742074
o  AUC mean: 0.736918
o  Accuracy LR: 0.734747 (0.029219)
```

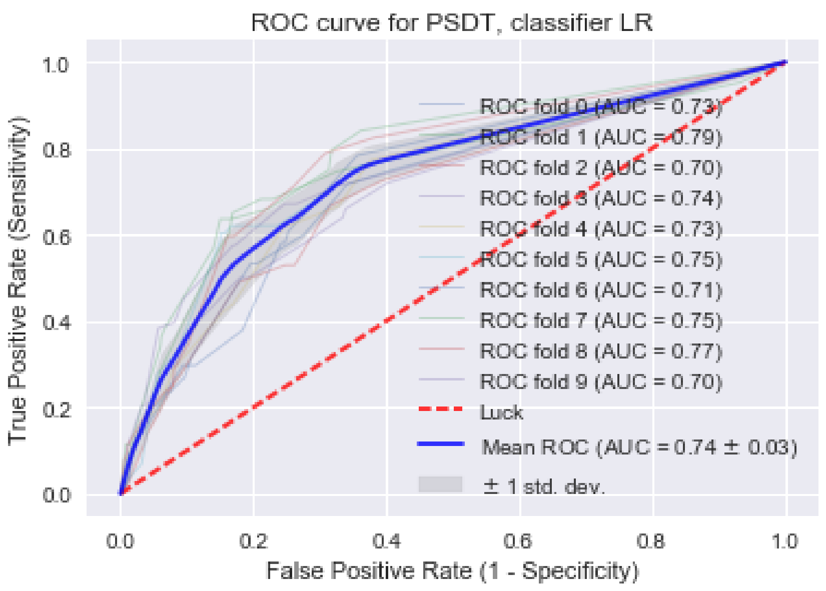


Figure 120. Mean ROC for Logistic Regression Classifier – Low Variance.

In figure 120 we can appreciate that the Auc mean for the pair Logistic Regression Classifier – Low Variance makes a "fair" discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.55 value of sensitivity. In the second phase, we can see that when approaching to the best classification point, the curve grows stable, reaching a growth parallel to the Luck line. Finally, from 0.78 of sensitivity onwards, the predicted negative elements prevail increasing the horizontal growth.

- **Neural Network Classifier**

```
o  Sensitivity mean: 0.695888
o  Specificity mean: 0.701019
o  AUC mean: 0.740440
o  Accuracy NN: 0.700416 (0.043444)
```

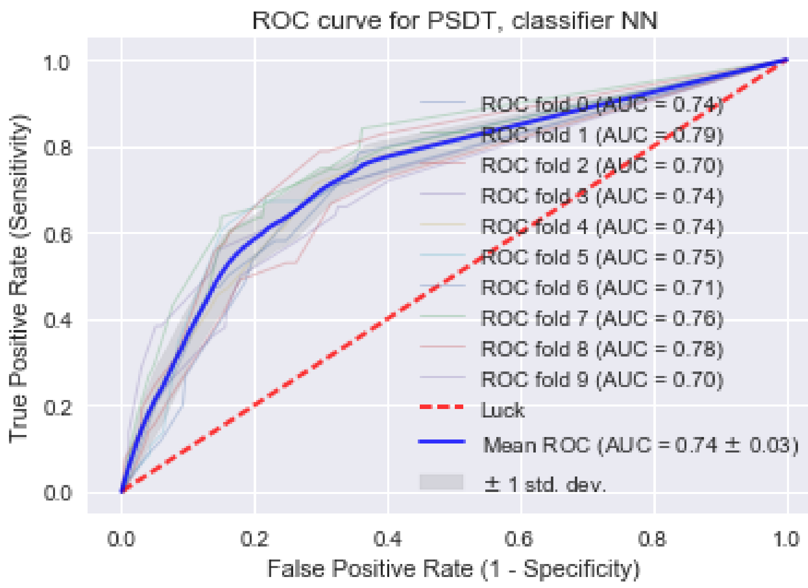


Figure 121. Mean ROC for Neural Network Classifier – Low Variance.

In figure 121 we can appreciate that the Auc mean for the pair Neural Network Classifier – Low Variance makes a "fair" discrimination with a value of 0.74. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.55 value of sensitivity. In the second phase, we can see that when approaching to the best classification point, the curve grows stable, reaching a growth parallel to the Luck line. Finally, from 0.78 of sensitivity onwards, the predicted negative elements prevail increasing the horizontal growth.

- **Random Forest Classifier**

```
o  Sensitivity mean: 0.677924
o  Specificity mean: 0.708993
o  AUC mean: 0.733339
o  Accuracy RFC: 0.706466 (0.027090)
```

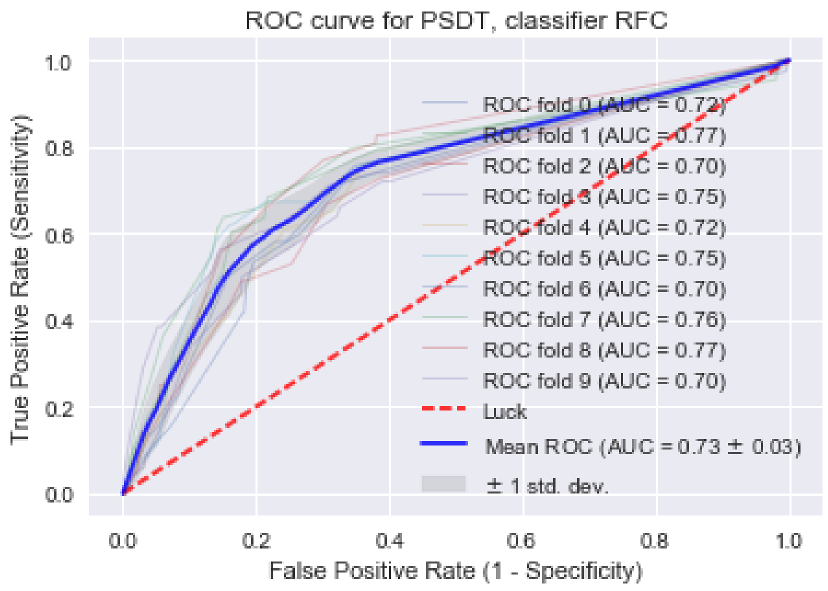


Figure 122. Mean ROC for Random Forest Classifier – Low Variance.

In figure 122 we can appreciate that the Auc mean for the pair Random Forest Classifier – Low Variance makes a "fair" discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.58 value of sensitivity. In the second phase, we can see that when approaching to the best classification point, the curve grows stable, reaching a growth parallel to the Luck line. Finally, from 0.75 of sensitivity onwards, the predicted negative elements prevail increasing the horizontal growth.

- **Gradient Boosting Classifier**

```
o  Sensitivity mean: 0.641639
o  Specificity mean: 0.740915
o  AUC mean: 0.733166
o  Accuracy GRADBOOSCLAS: 0.733259 (0.026124)
```

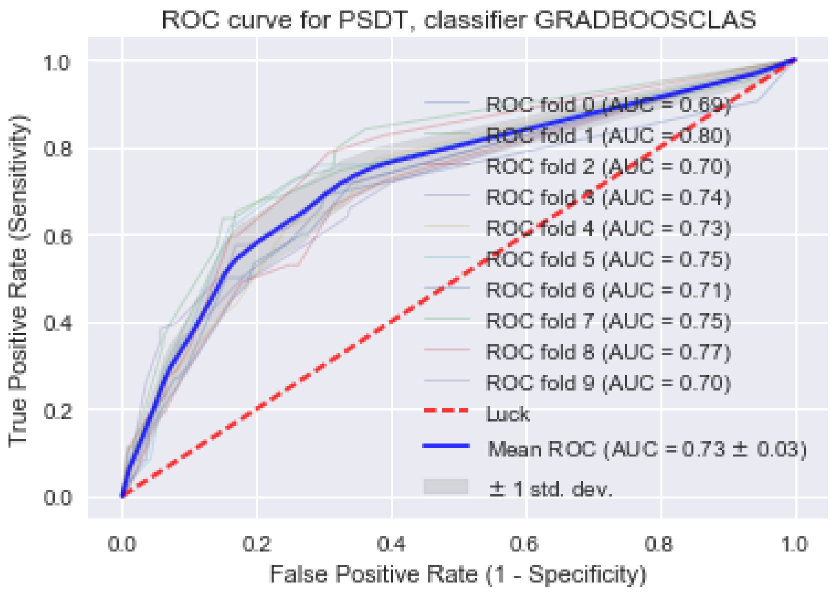


Figure 123. Mean ROC for Gradient Boosting Classifier – Low Variance.

In figure 123 we can appreciate that the Auc mean for the pair Gradient Boosting Classifier – Low Variance makes a "fair" discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.58 value of sensitivity. In the second phase, we can see that when approaching to the best classification point, the curve grows stable, reaching a growth parallel to the Luck line. Finally, from 0.75 of sensitivity onwards, the predicted negative elements prevail increasing the horizontal growth.

- **Gaussian NB Classifier**

```
o  Sensitivity mean: 0.771766
o  Specificity mean: 0.577533
o  AUC mean: 0.731844
o  Accuracy GNB: 0.591270 (0.175175)
```

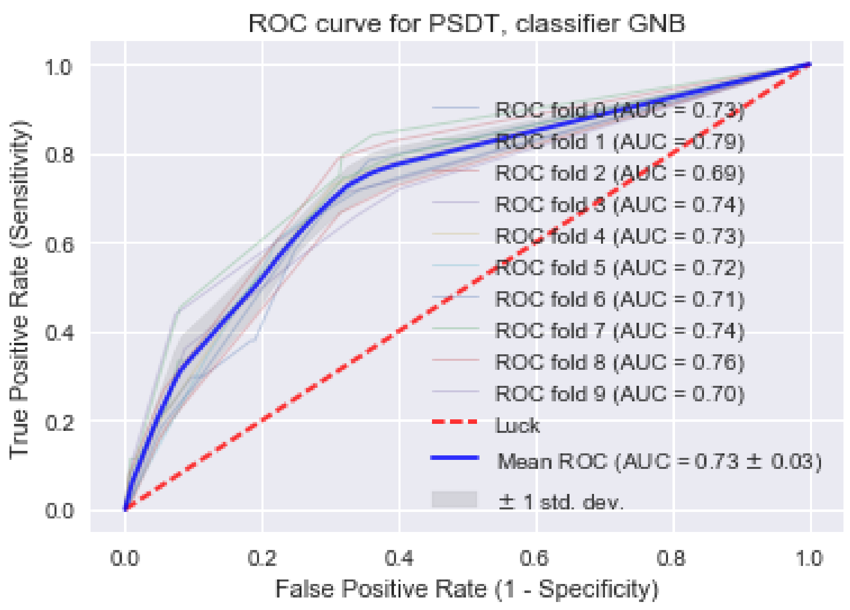


Figure 124. Mean ROC for Gaussian Naïve Bayes Classifier – Low Variance.

In figure 124 we can appreciate that the Auc mean for the pair Gaussian Naïve Bayes Classifier – Low Variance makes a "fair" discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.3 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a bigger number of positive values rather than negative values. Finally, when approaching to the best classification point, the curve makes a change and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Plotbox Algorithm Comparison.**

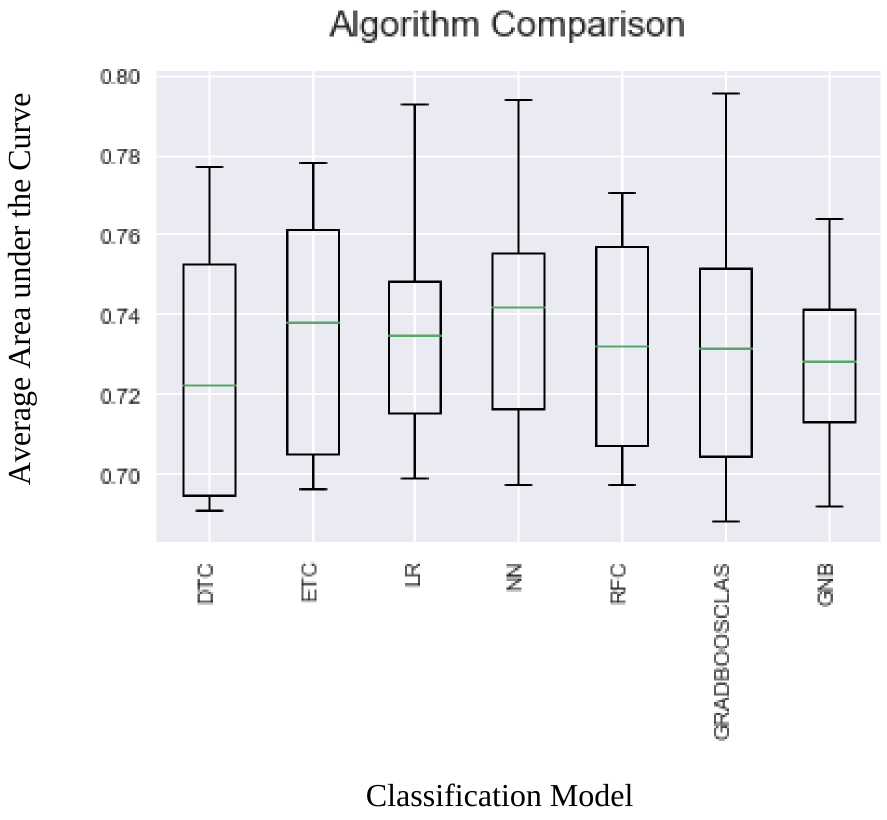


Figure 125. Comparison of classification algorithms with Low Variance

In figure 125 we can appreciate that the Comparison between the different AUC values obtained for the classification algorithms. We can appreciate here the results for a poor classification, where the variance of the Auc values obtained is spread over a wide range. IT is quite difficult to appreciate which one of the algorithms is going to give us the better performance. LR and NN have the smaller boxes and the wider whiskers, concentrating a 50% of the results between 0.71 and 0.75. This suggests the AUC of this group is going to be bigger, and giving us NN as better classifier model because it values is distributed in a bigger box but with the same whiskers as LR.

- **Table with results of this experiment for Low Variance**

  In the table, we are able to see the results for each classification model in the new subspace created with Low Variance. This subspace is going to be created with 4 features and we are going to focus our attention in the Area under the Curve and the sensitivity. As we can realize here the algorithm the one performs better is the LR for no Dimensionality reduction and in second place we can find the NN algorithm.

Table 6. Comparison table with the results for each Classification model with Low Variance

Model	**AUC**	**SENSITIVITY**	**SPECIFICITY**	**ACCURACY**
DTC	0.727408	0.662864	0.728164	0.723130
ETC	0.736028	0.693760	0.699250	0.698804
LR	0.736918	0.640049	0.742074	0.734747
NN	0.740440	0.695888	0.701019	0.700416
RFC	0.733339	0.677924	0.708993	0.706466
GBC	0.733166	0.641639	0.740915	0.733259
GNB	0.731844	0.771766	0.577533	0.591270

4.1.5 mRMR

The fifth case to analyze is the feature selection algorithm minimum-redundancy maximum-relevancy

- **Decision Tree Classifier**

```
o  Sensitivity mean: 0.663776
o  Specificity mean: 0.762198
o  AUC mean: 0.733713
o  Accuracy DTC: 0.755002 (0.018223)
```

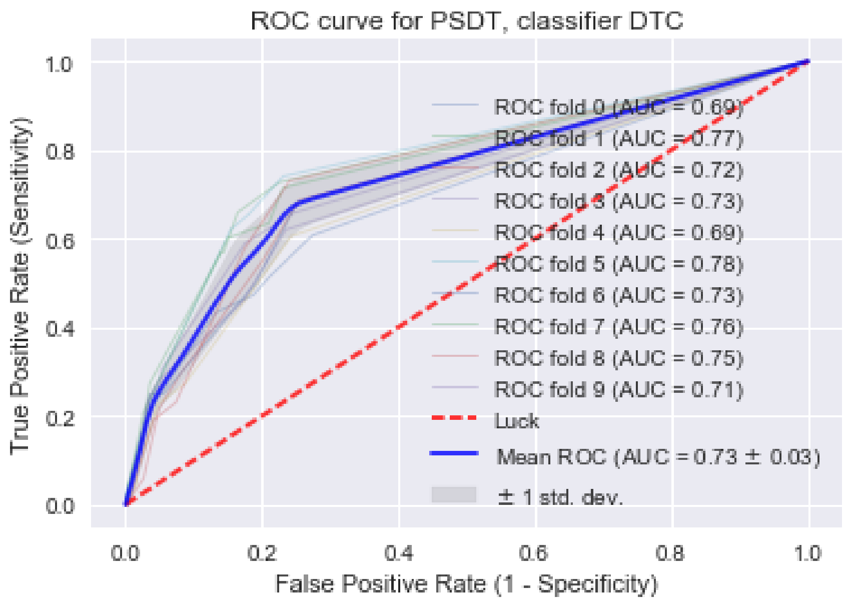


Figure 126. Mean ROC for Decision Tree Classifier – mRMR.

In figure 126 we can appreciate that the Auc mean for the pair Decision Tree Classifier – mRMR makes a "fair" discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.25 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a bigger number of positive values rather than negative values. Finally, when approaching to the best classification point, the curve makes a change and onwards the predicted negative elements prevail increasing the horizontal growth

- **Extra Tree Classifier**
  - Sensitivity mean: 0.666896
  - Specificity mean: 0.761495
  - AUC mean: 0.734415
  - Accuracy ETC: 0.754261 (0.017432)

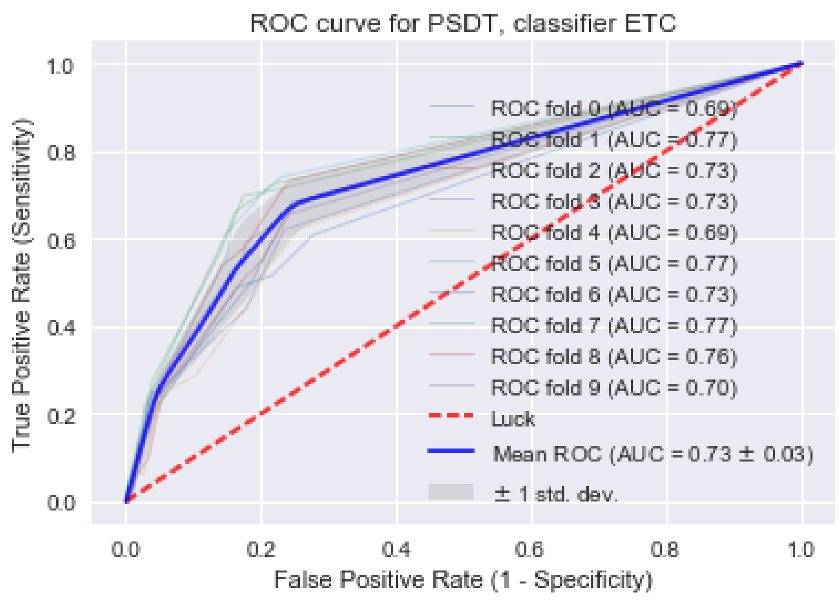


Figure 127. Mean ROC for pair Extra Tree Classifier – mRMR.

In figure 127 we can appreciate that the Auc mean for the pair Extra Tree Classifier – mRMR makes a "fair" discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.25 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a bigger number of positive values rather than negative values. Finally, when approaching to the best classification point, the curve makes a change and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Logistic Regression Classifier**

```
o  Sensitivity mean: 0.626143
o  Specificity mean: 0.782715
o  AUC mean: 0.734658
o  Accuracy LR: 0.771304 (0.030170)
```

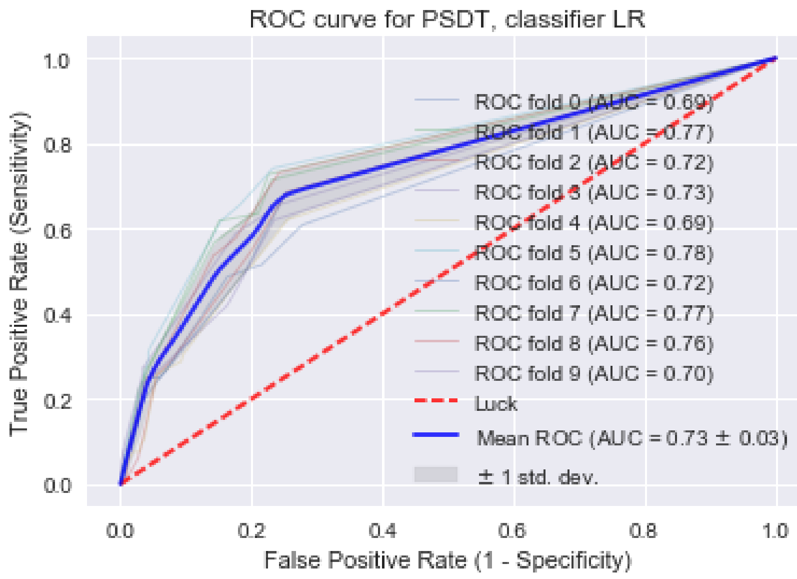


Figure 128. Mean ROC for Logistic Regression Classifier – mRMR.

In figure 128 we can appreciate that the Auc mean for the pair Logistic Regression Classifier – mRMR makes a "fair" discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.27 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a bigger number of positive values rather than negative values. Finally, when approaching to the best classification point, the curve makes a change and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Neural Network Classifier**

```
o  Sensitivity mean: 0.668411
o  Specificity mean: 0.760559
o  AUC mean: 0.733612
o  Accuracy NN: 0.753520 (0.017693)
```

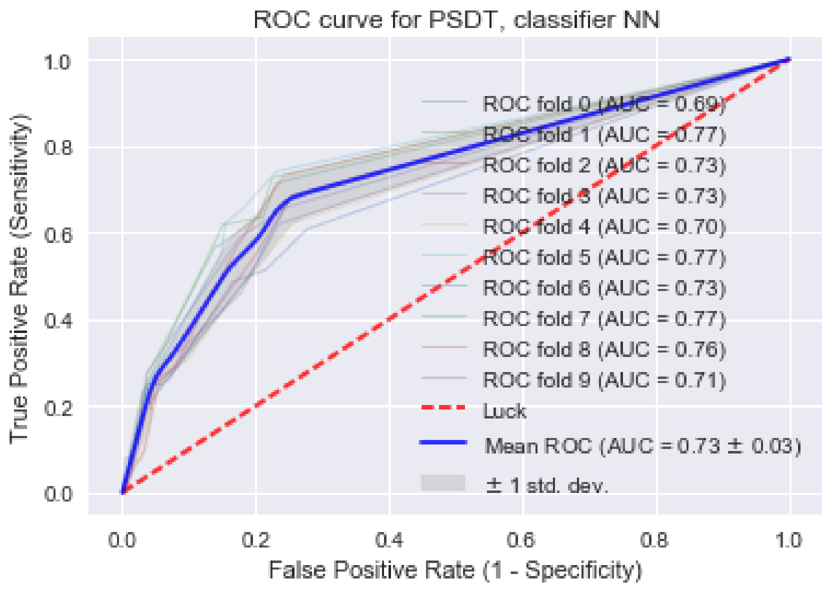


Figure 129. Mean ROC for Neural Network Classifier – mRMR.

In figure 129 we can appreciate that the Auc mean for the pair Neural Network Classifier – mRMR makes a “fair” discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.23 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a bigger number of positive values rather than negative values. In the last part of this phase it makes a strange, meaning that in this point the classifier is not able to discriminate properly. Finally, when approaching to the best classification point, the curve makes a change and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Random Forest Classifier**

```
o  Sensitivity mean: 0.679017
o  Specificity mean: 0.755848
o  AUC mean: 0.733939
o  Accuracy RFC: 0.750063 (0.016814)
```

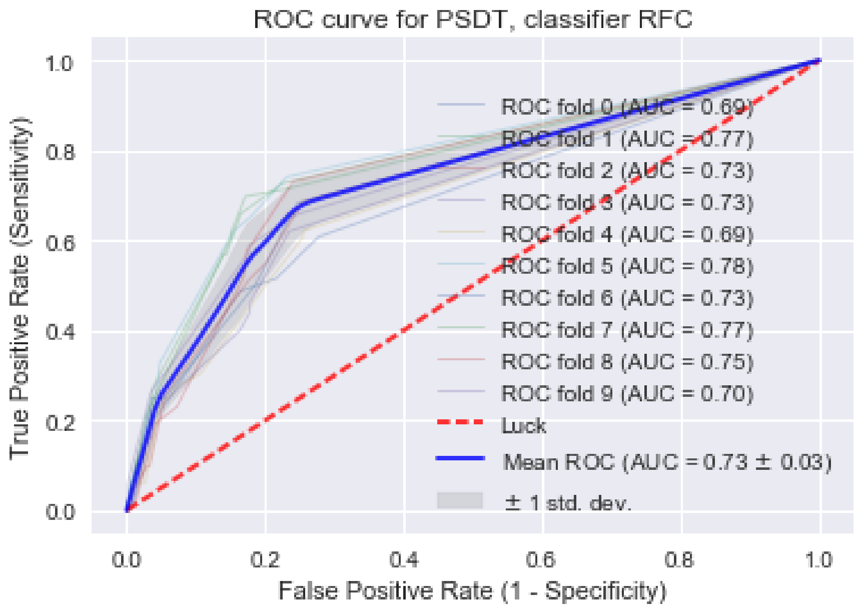


Figure 130. Mean ROC for Random Forest Classifier – mRMR.

In figure 130 we can appreciate that the Auc mean for the pair Random Forest Classifier – mRMR makes a "fair" discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.27 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a bigger number of positive values rather than negative values. Finally, when approaching to the best classification point, the curve makes a change and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Gradient Boosting Classifier**

```
o  Sensitivity mean: 0.613932
o  Specificity mean: 0.786908
o  AUC mean: 0.735768
o  Accuracy GRADBOOSCLAS: 0.774514 (0.028499)
```

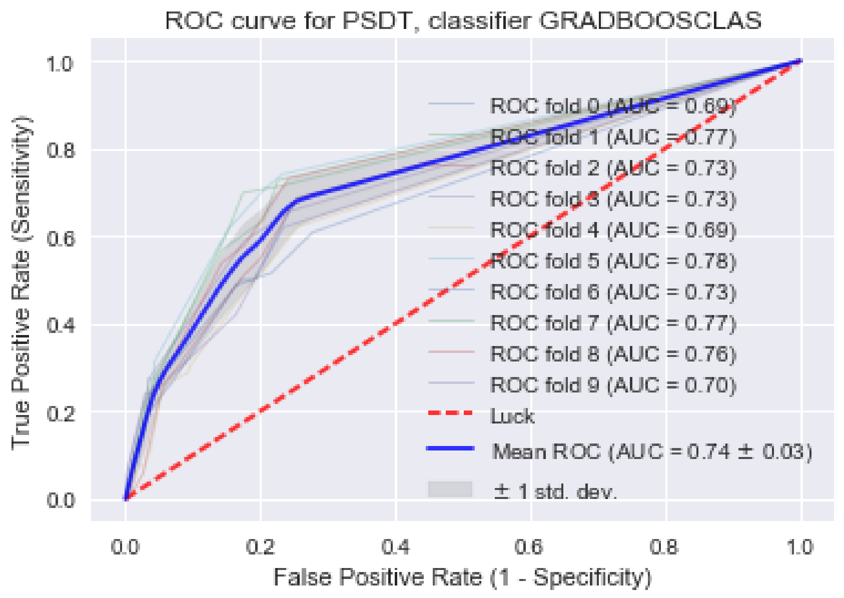


Figure 131. Mean ROC for Gradient Boosting Classifier – mRMR.

In figure 131 we can appreciate that the Auc mean for the pair Gradient Boosting Classifier – mRMR makes a “fair” discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.27 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a bigger number of positive values rather than negative values. In the last part of this phase it makes a strange, meaning that in this point the classifier is not able to discriminate properly. Finally, after reaching the best classification point, the curve makes a change in the value 0.67 and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Gaussian NB Classifier**

```
o  Sensitivity mean: 0.679017
o  Specificity mean: 0.755050
o  AUC mean: 0.732916
o  Accuracy GNB: 0.749323 (0.016757)
```

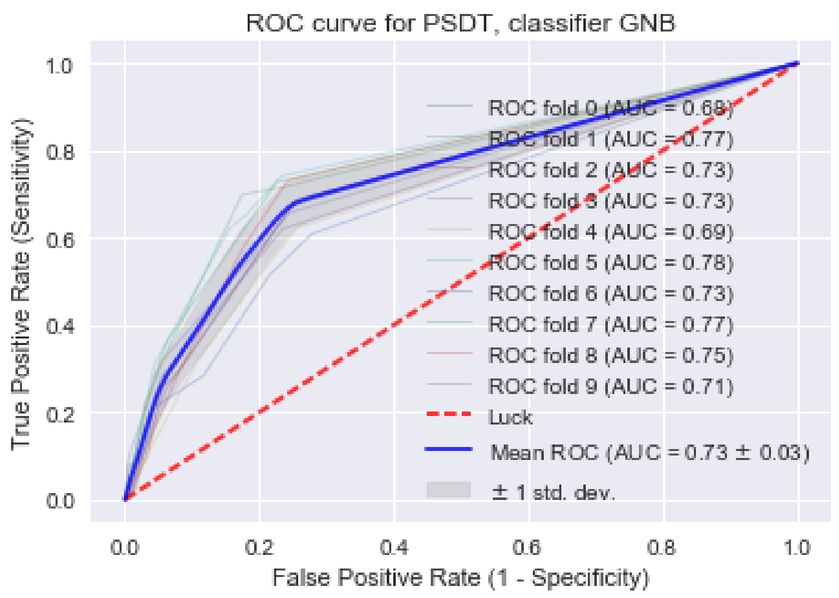


Figure 132. Mean ROC for Gaussian Naïve Bayes Classifier – mRMR.

In figure 132 we can appreciate that the Auc mean for the pair Gaussian Naïve Bayes Classifier – mRMR makes a “fair” discrimination with a value of 0.73. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.29 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a bigger number of positive values rather than negative values. Finally, when approaching to the best classification point, the curve makes a change in the value 0.69 and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Plotbox Algorithm Comparison.**

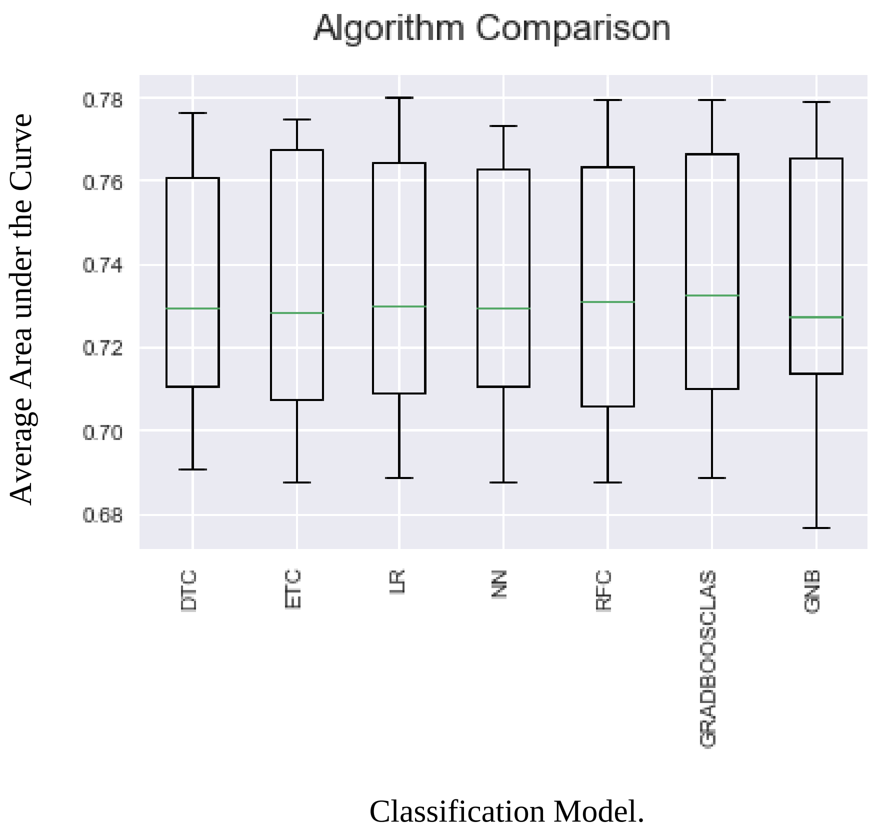


Figure 133. Comparison of classification algorithms with mRMR

In figure 133 we can appreciate the comparison between the different AUC values obtained for the classification algorithms. We can appreciate here the results for a poor classification, where the variance of the Auc values obtained is spread over a wide range. It is quite difficult to appreciate which one of the algorithms is going to give us the better performance. We have to take a look the results obtained in the ROC curve to select the better algorithm. In this case, LR and NN were selected as the better algorithms, with a small advantage over the other algorithms, but at the same time all with a low prediction accuracy.

- **Table with results of this experiment for mRMR**

In table 7, we are able to see the results for each classification model in the new subspace created with mRMR. This subspace is going to be created with 4 features and we are going to focus our attention in the Area under the Curve and the sensitivity.

As we can realize here the algorithm the one performs better is the GBC for no Dimensionality reduction and in second place we can find the LR algorithm.

Table 7. Comparison table with the results for each Classification model with MRMR

Model	AUC	SENSITIVITY	SPECIFICITY	ACCURACY
DTC	0.733713	0.663776	0.762198	0.755002
ETC	0.734415	0.666896	0.761495	0.754261
LR	0.734658	0.626143	0.782715	0.771304
NN	0.733612	0.668411	0.760559	0.753520
RFC	0.733939	0.679017	0.755848	0.750063
GBC	0.735768	0.613932	0.786908	0.774514
GNB	0.732916	0.679017	0.755050	0.749323

4.1.6 Random Forest

The sixth case to analyze is the feature selection algorithm, Random Forest

- **Decision Tree Classifier**

```
o  Sensitivity mean: 0.656386
o  Specificity mean: 0.766256
o  AUC mean: 0.737802
o  Accuracy DTC: 0.757972 (0.030962)
```

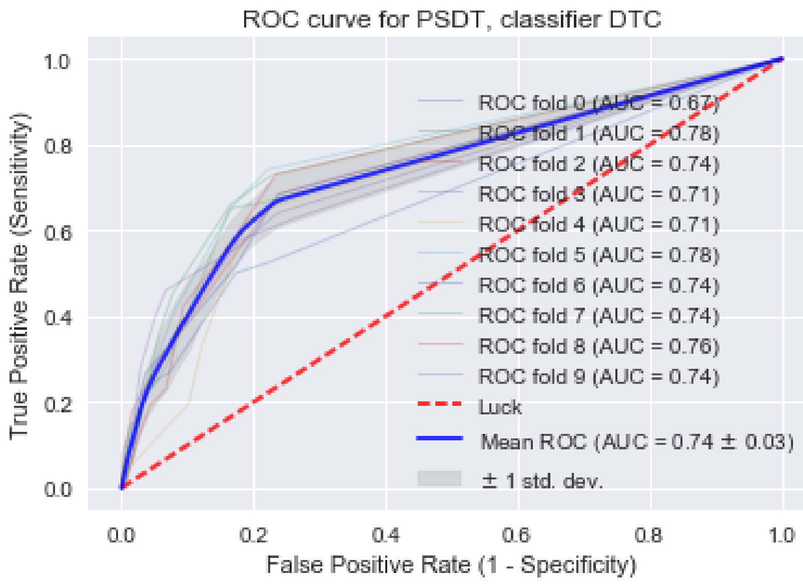


Figure 134. Mean ROC for Decision Tree Classifier – Random Forest.

In the figure 134, we can appreciate that the Auc mean for the pair Decision Tree Classifier – Random Forest makes a "fair" discrimination with a value of 0.73. We can find two phases. In the first one, the curve grows more or less steadily vertical until it reaches the best classification point, where it makes suddenly a change and grows negatively. From here on, the negative predicted elements prevail, increasing the horizontal growth.

- **Extra Tree Classifier**

```
o  Sensitivity mean: 0.665858
o  Specificity mean: 0.757913
o  AUC mean: 0.751584
o  Accuracy ETC: 0.750933 (0.028047)
```

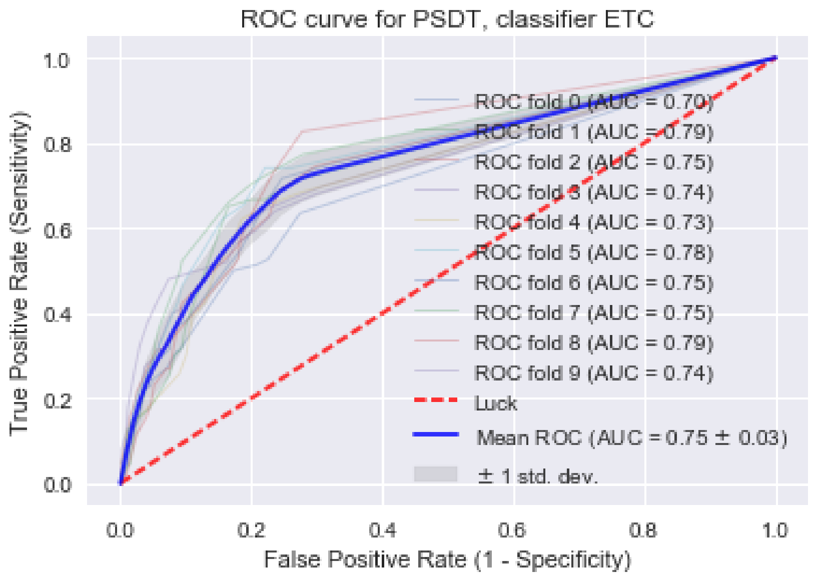


Figure 135. Mean ROC for Extra Tree Classifier – Random Forest.

In the figure 135, we can appreciate that the Auc mean for the pair Extra Tree Classifier – Random Forest makes a "fair" discrimination with a value of 0.75. We can find two phases. In the first one, the curve grows more or less steadily vertical until it reaches the best classification point, where it makes suddenly a change and grows negatively. From here on, the negative predicted elements prevail, increasing the horizontal growth.

- **Logistic Regression Classifier**

```
o  Sensitivity mean: 0.656858
o  Specificity mean: 0.776266
o  AUC mean: 0.753428
o  Accuracy LR: 0.767358 (0.024289)
```

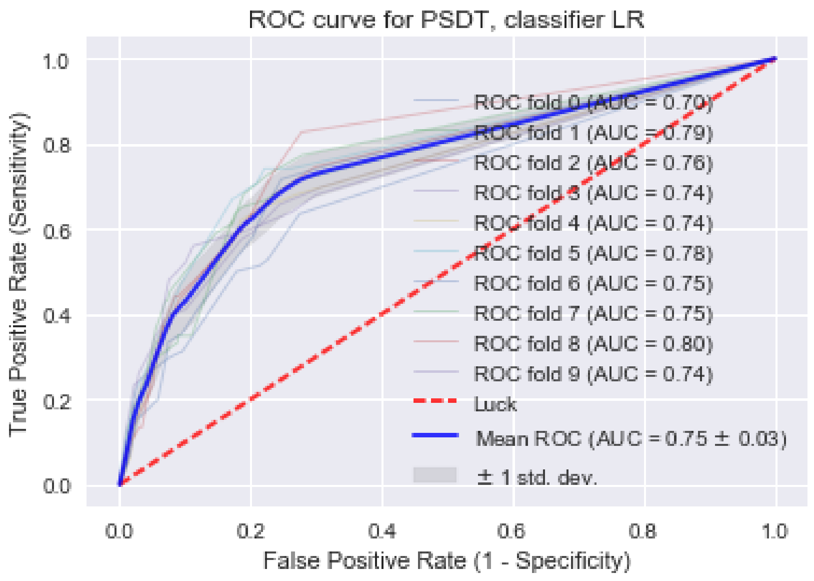


Figure 136. Mean ROC for Logistic Regression Classifier – Random Forest.

In the figure 136, we can appreciate that the Auc mean for the pair Logistic Regression Classifier – Random Forest makes a "fair" discrimination with a value of 0.75. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.4 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a smaller number of positive values. Finally, when approaching to the best classification point, the curve makes a change in the value 0.65 and onwards the predicted negative elements prevail increasing the horizontal growth

- **Neural Network Classifier**

```
o  Sensitivity mean: 0.679980
o  Specificity mean: 0.749392
o  AUC mean: 0.752433
o  Accuracy NN: 0.744019 (0.031626)
```

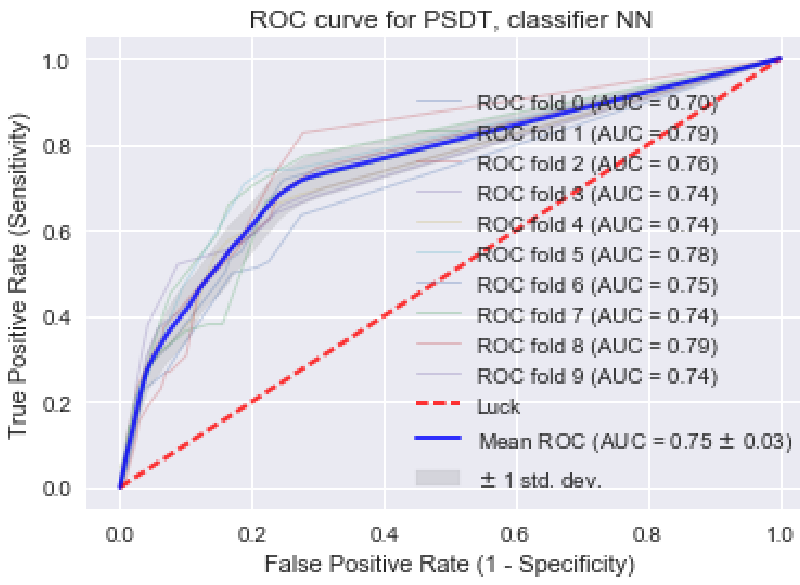


Figure 137. Mean ROC for Neural Network Classifier – Random Forest.

In the figure 137, we can appreciate that the Auc mean for the pair Neural Network Classifier – Random Forest makes a “fair” discrimination with a value of 0.75. We can find three phases. In the first one, the curve grows vertically stable until it reaches the 0.3 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a smaller number of positive values. Finally, when approaching to the best classification point, the curve makes a change in the value 0.67 and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Random Forest Classifier**

```
o  Sensitivity mean: 0.676756
o  Specificity mean: 0.752582
o  AUC mean: 0.750989
o  Accuracy RFC: 0.746984 (0.032204)
```

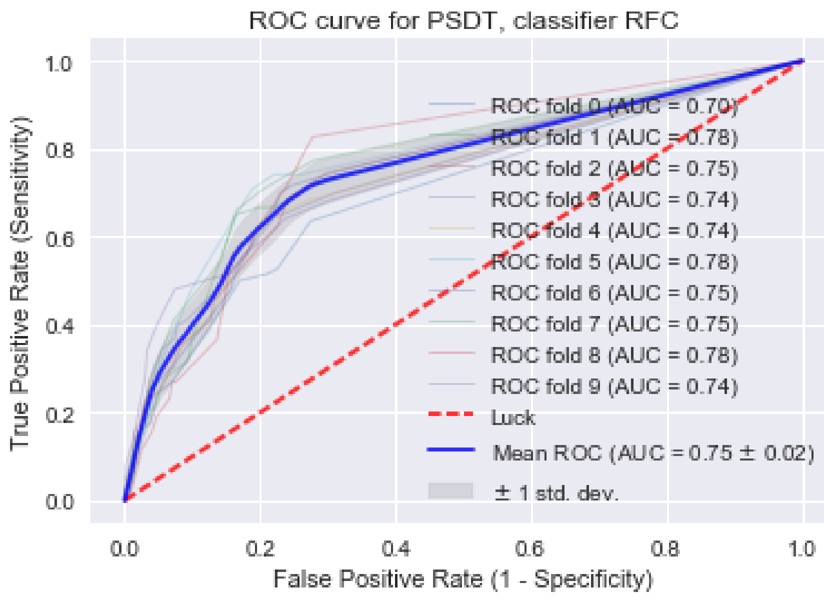


Figure 138. Mean ROC for pair Random Forest Classifier – Random Forest.

In the figure 138, we can appreciate that the Auc mean for the pair Random Forst Classifier – Random Forest makes a "fair" discrimination with a value of 0.75. We can find three phases. In the first one, the curve grows vertically steadily until it reaches the 0.3 value of sensitivity. In the second phase, we can see that it continues growing vertically with some inconsistencies, predicting a smaller number of positive values. Finally, when approaching to the best classification point, the curve makes a change in the value 0.65 and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Gradient Boosting Classifier**

```
o  Sensitivity mean: 0.646035
o  Specificity mean: 0.782966
o  AUC mean: 0.752183
o  Accuracy GRADBOOSCLAS: 0.772666 (0.021746)
```

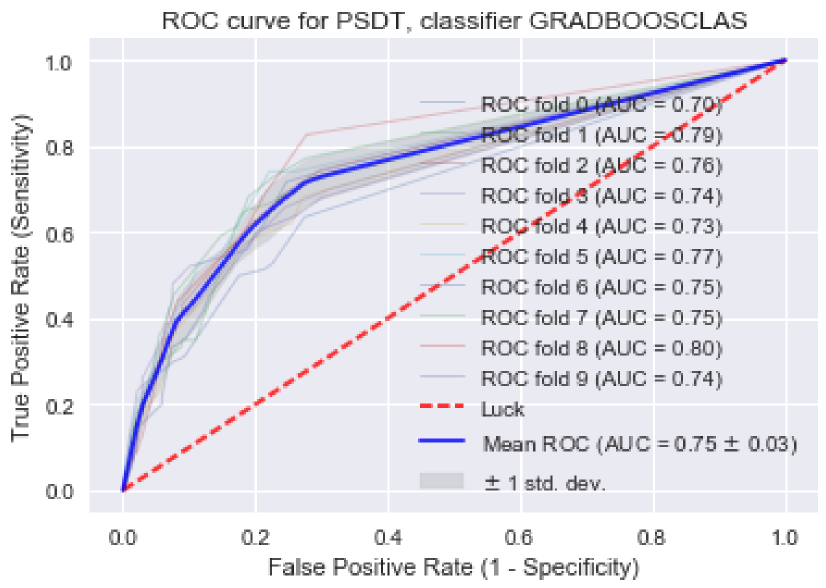


Figure 139. Mean ROC for Gradient Boosting Classifier – Random Forest.

In the figure 139, we can appreciate that the Auc mean for the pair Gradient Boosting Classifier – Random Forest makes a “fair” discrimination with a value of 0.75. We can find three phases. In the first one, the curve grows vertically steadily until it reaches the 0.4 value of sensitivity. In the second phase, we can see that it continues growing vertically, predicting a smaller number of positive values. Finally, when approaching to the best classification point, the curve makes a change in the value 0.65 and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Gaussian NB Classifier**

```
o  Sensitivity mean: 0.693048
o  Specificity mean: 0.740508
o  AUC mean: 0.752328
o  Accuracy GNB: 0.736354 (0.028027)
```

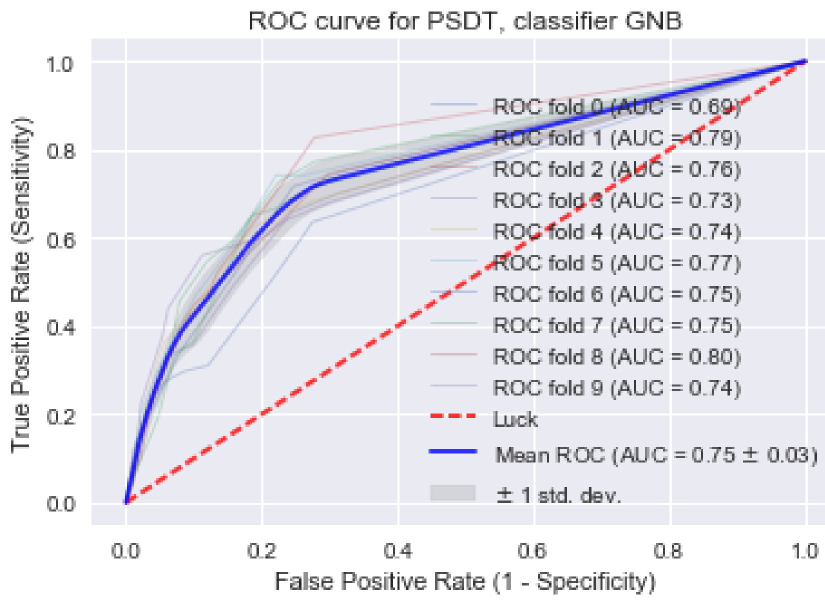


Figure 140. Mean ROC for Random Forest Classifier – Random Forest.

In the figure 140, we can appreciate that the Auc mean for the pair Random Forest Classifier – Random Forest makes a “fair” discrimination with a value of 0.75. We can find three phases. In the first one, the curve grows vertically steadily until it reaches the 0.3 value of sensitivity. In the second phase, we can see that it continues growing predicting a smaller number of positive values. Finally, when approaching to the best classification point, the curve makes a change in the value 0.69 and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Plotbox Algorithm Comparison.**

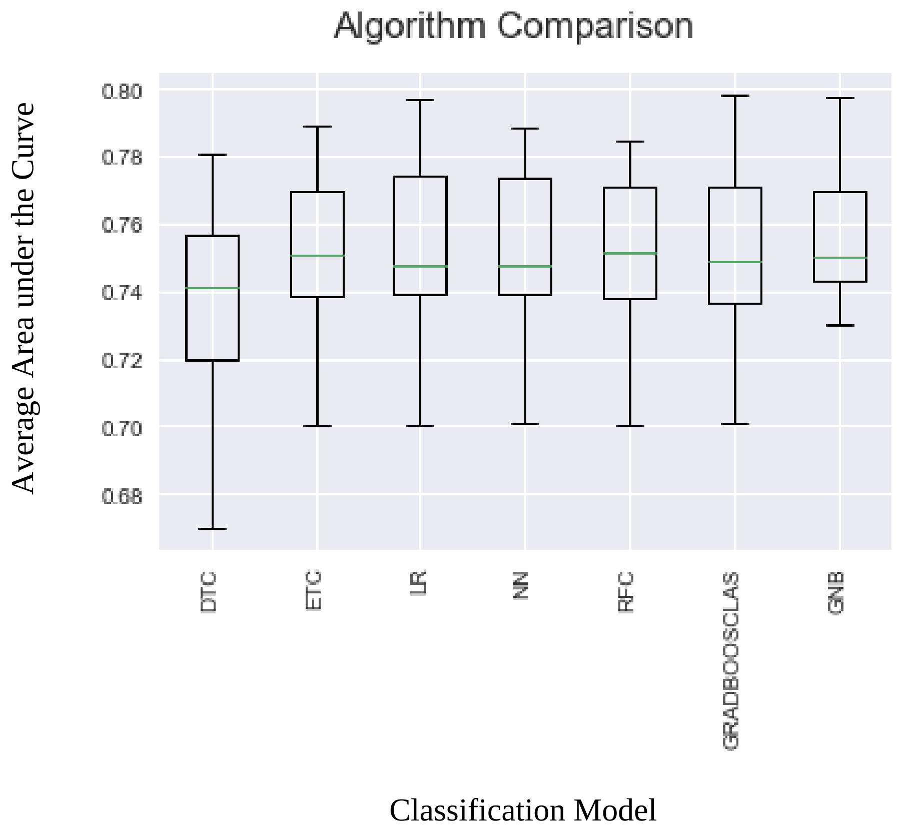


Figure 141. Comparison of classification algorithms with Random Forest.

In the figure 141, we can appreciate the comparison between the different AUC values obtained for the classification algorithms. We can appreciate here the results for a poor classification, where the variance of the Auc values obtained is spread over a wide range. It is quite difficult to appreciate which one of the algorithms is going to give us the better performance. DTC and GNB are the outliers in this comparison. LR and NN have the bigger boxes and the lower whiskers are similar in comparison to ETC, RFC, and GRADBOOSCLAS. In the other hand, the upper whiskers are wider and the upper box limit is higher. This suggests the AUC of this group is going to be bigger, and LR is going to lead the classification as the better algorithm.

- **Table with results of this experiment for Random Forest**

  In the table, we are able to see the results for each classification model in the new subspace created with Random Forest. This subspace is going to be created with 4 features and we are going to focus our attention in the Area under the Curve and the sensitivity. As we can realize here the algorithm the one performs better is the LR for no Dimensionality reduction and in second place we can find the NN algorithm.

Table 8. Comparison table with the results for each Classification model with Random Forest

Model	AUC	SENSITIVITY	SPECIFICITY	ACCURACY
DTC	0.737802	0.656386	0.766256	0.757972
ETC	0.751584	0.665858	0.757913	0.750933
LR	0.753428	0.656858	0.776266	0.767358
NN	0.752433	0.679980	0.749392	0.744019
RFC	0.750989	0.676756	0.752582	0.746984
GBC	0.752183	0.646035	0.782966	0.772666
GNB	0.752328	0.693048	0.740508	0.736354

4.1.7 Univariate Feature Selection

The last case to analyze is the feature selection algorithm Univariate Feature Selection.

- **Decision Tree Classifier**

```
o  Sensitivity mean: 0.653276
o  Specificity mean: 0.791957
o  AUC mean: 0.734563
o  Accuracy DTC: 0.781677 (0.016187)
```

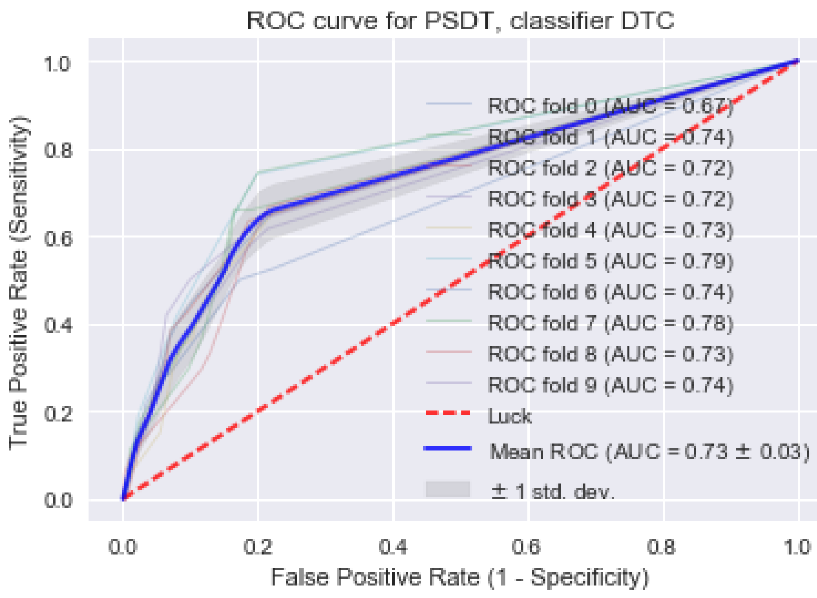


Figure 142. Mean ROC for Decision Tree Classifier – Univariate Feature Selection.

In figure 142 we can appreciate that the Auc mean for the pair Decision Tree Classifier – Univariate Feature Selection makes a "fair" discrimination with a value of 0.73. We can find two phases. In the first one, the curve grows more or less steadily vertical until it reaches the best classification point, where it makes suddenly a change and grows negatively. From here on, the negative predicted elements prevail, increasing the horizontal growth.

- **Extra Tree Classifier**

```
o Sensitivity mean: 0.676701
o Specificity mean: 0.771364
o AUC mean: 0.754153
o Accuracy ETC: 0.764265 (0.030083)
```

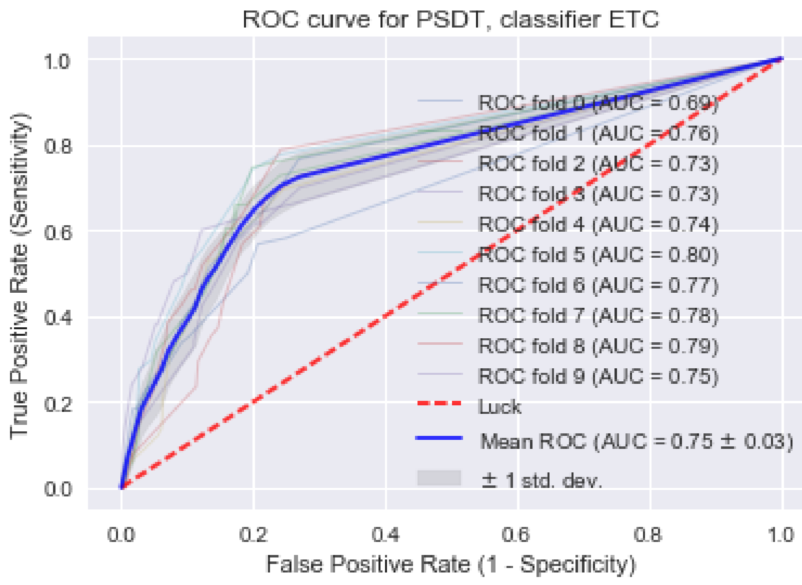


Figure 143. Mean ROC for Extra Tree Classifier – Univariate Feature Selection.

In figure 143 we can appreciate that the Auc mean for the pair Extra Tree Classifier – Univariate Feature Selection makes a "fair" discrimination with a value of 0.75. We can find two phases. In the first one, the curve grows more or less steadily vertical until it reaches the best classification point, where it makes suddenly a change and grows negatively. From here on, the negative predicted elements prevail, increasing the horizontal growth.

- **Logistic Regression Classifier**

```
o Sensitivity mean: 0.616027
o Specificity mean: 0.796613
o AUC mean: 0.756793
o Accuracy LR: 0.783401 (0.035699)
```

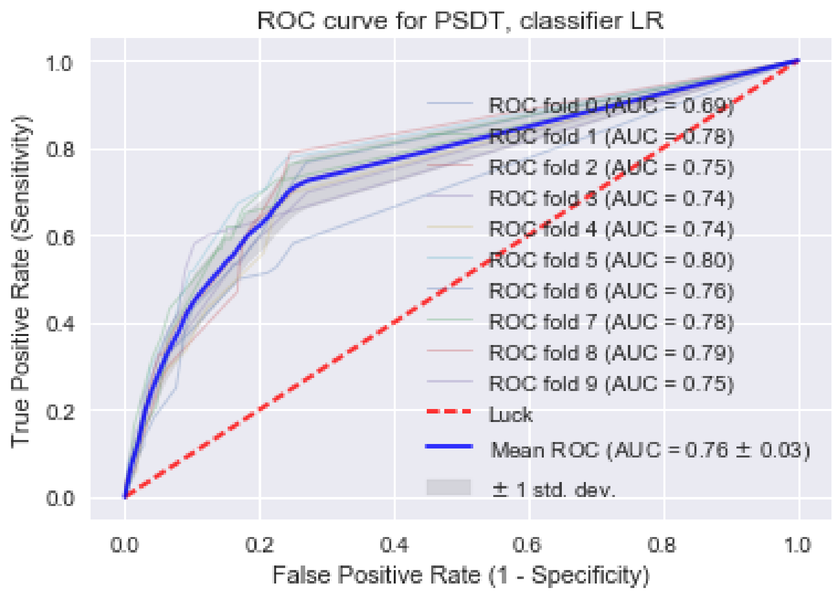


Figure 144. Mean ROC for Logistic Regression Classifier – Univariate Feature Selection.

In figure 144 we can appreciate that the Auc mean for the pair Logistic Regression Classifier – Univariate Feature Selection makes a "fair" discrimination with a value of 0.75. We can find two phases. In the first one, the curve grows more or less steadily vertical until it reaches the best classification point, where it makes a strange. It means that in this point the classifier is not able to discriminate properly. From here on, the negative predicted elements prevail, increasing the horizontal growth.

- **Neural Network Classifier**

```
o  Sensitivity mean: 0.696031
o  Specificity mean: 0.757638
o  AUC mean: 0.753044
o  Accuracy NN: 0.753277 (0.026646)
```

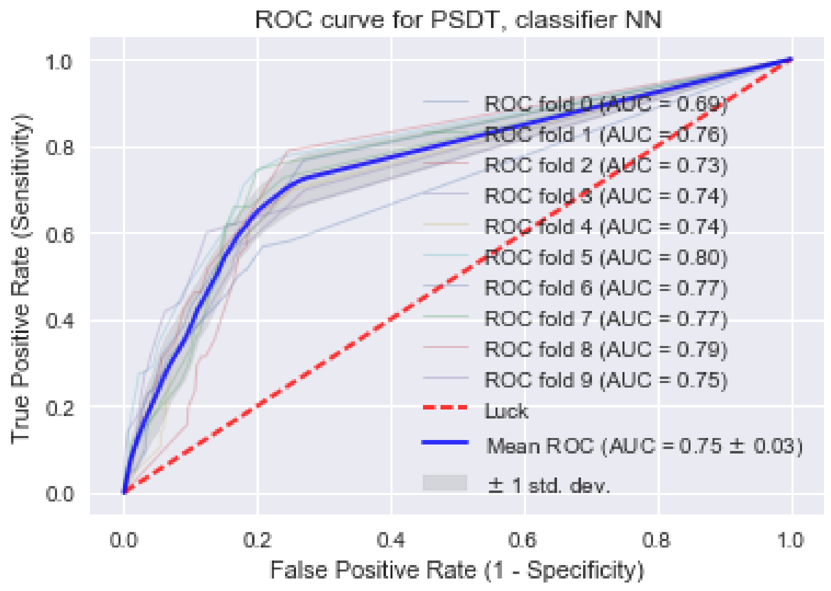


Figure 145. Mean ROC for Logistic Regression Classifier – Univariate Feature Selection.

In figure 145 we can appreciate that the Auc mean for the pair Logistic Regression Classifier – Univariate Feature Selection makes a "fair" discrimination with a value of 0.75. We can find two phases. In the first one, the curve grows more or less steadily vertical until it reaches the best classification point, where it makes a strange. It means that at this point is not able to discriminate properly. From here on, the negative predicted elements prevail, increasing the horizontal growth.

- **Random Forest Classifier**

```
o  Sensitivity mean: 0.665034
o  Specificity mean: 0.776431
o  AUC mean: 0.752944
o  Accuracy RFC: 0.768092 (0.028285)
```

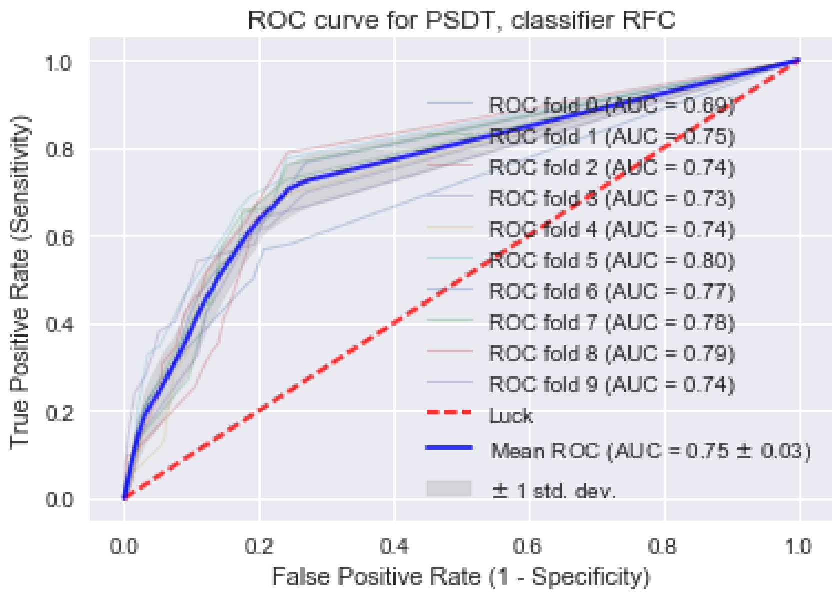


Figure 146. Mean ROC for Random Forest Classifier – Univariate Feature Selection.

In figure 146 we can appreciate that the Auc mean for the pair Random Forest Classifier – Univariate Feature Selection makes a “fair” discrimination with a value of 0.75. We can find three phases. In the first one, the curve grows vertically steadily until it reaches the 0.2 value of sensitivity. In the second phase, we can see that it continues growing predicting a smaller number of positive values. Finally, when approaching to the best classification point, the curve makes a change in the value 0.66 and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Gradient Boosting Classifier**

```
o  Sensitivity mean: 0.636512
o  Specificity mean: 0.787294
o  AUC mean: 0.756333
o  Accuracy GRADBOOSCLAS: 0.776611 (0.033979)
```

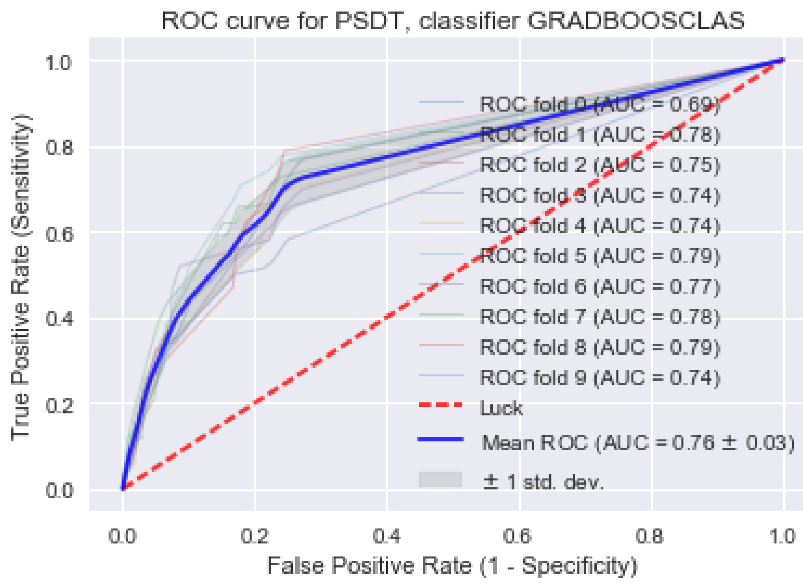


Figure 147. Mean ROC for Gradient Boosting Classifier – Univariate Feature Selection.

In figure 147 we can appreciate that the Auc mean for the pair Gradient Boosting Classifier – Univariate Feature Selection makes a “fair” discrimination with a value of 0.75. We can find four phases. In the first one, the curve grows vertically steadily until it reaches the 0.4 value of sensitivity. In the second phase, we can see that it continues growing predicting a smaller number of positive values. In the next phase, after approaching to the best classification point, the curve makes a strange, where we can consider that the classifier is not able to discriminate properly the values. Finally, when the curve reaches the value 0.70 and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Gaussian NB Classifier**

```
o  Sensitivity mean: 0.695493
o  Specificity mean: 0.758019
o  AUC mean: 0.756899
o  Accuracy GNB: 0.753401 (0.026865)
```

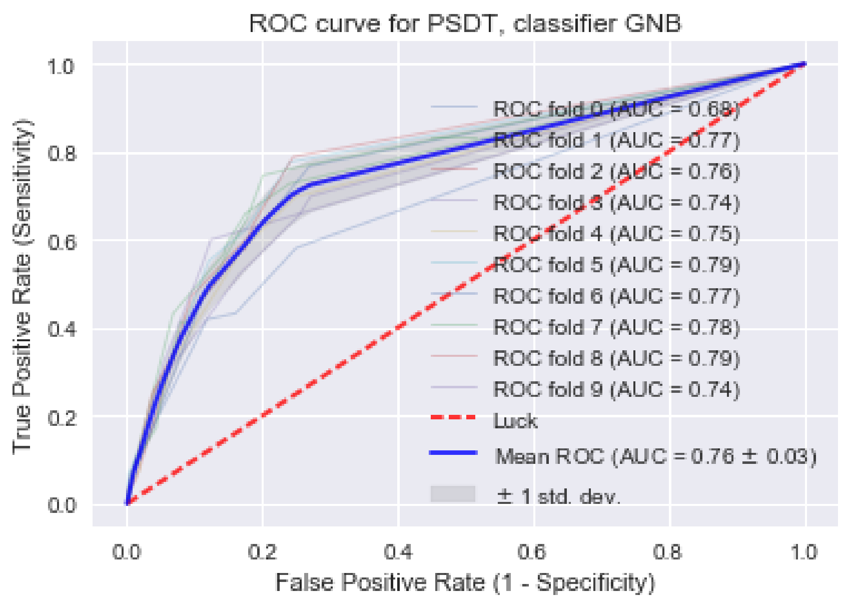


Figure 148. Mean ROC for Gaussian Naïve Bayes Classifier – Univariate Feature Selection.

In figure 148 we can appreciate that the Auc mean for the pair Gaussian Naïve Bayes Classifier – Univariate Feature Selection makes a “fair” discrimination with a value of 0.75. We can find three phases. In the first one, the curve grows vertically steadily until it reaches the 0.2 value of sensitivity. In the second phase, we can see that it continues growing predicting a smaller number of positive values. Finally, when approaching to the best classification point, the curve makes a change in the value 0.69 and onwards the predicted negative elements prevail increasing the horizontal growth.

- **Plotbox Algorithm Comparison.**

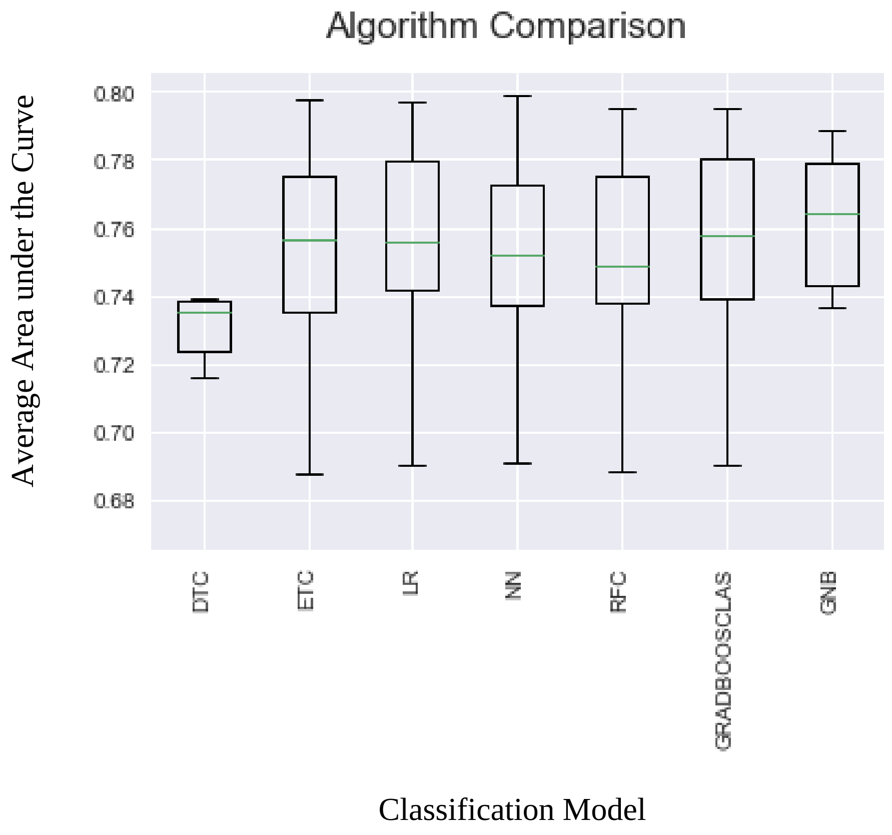


Figure 149. Comparison of classification algorithms with Univariate Feature Selection.

In figure 149 we can appreciate the comparison between the different AUC values obtained for the classification algorithms. We can appreciate here the results for a poor classification, where the variance of the Auc values obtained are spread over a wide range, except for DTC, where the values are quite similar. It is quite difficult to appreciate which one of the algorithms is going to give us the better performance. DTC and GNB are the outliers in this comparison. LR and GNB have the bigger boxes and the lower and upper whiskers of LR are similar in comparison to ETC, NN, RFC, and GRADBOOSCLAS. In the other hand, GNB has the values concentrated between 0.79

and 0.74. This suggests the AUC of the second group is going to be bigger, and LR is going to lead the classification as the better algorithm.

- **Table with results of this experiment for Univariate Feature Selection**

  In the table, we are able to see the results for each classification model in the new subspace created with Univariate Feature Selection. This subspace is going to be created with 4 features and we are going to focus our attention in the Area under the Curve and the sensitivity. As we can realize here the algorithm the one performs better is the GNB for no Dimensionality reduction and in second place we can find the LR algorithm.

Table 9. Comparison table with the results for each Classification model with Univariate Feature Selection

Model	**AUC**	**SENSITIVITY**	**SPECIFICITY**	**ACCURACY**
DTC	0.734563	0.653276	0.791957	0.781677
ETC	0.754153	0.676701	0.771364	0.764265
LR	0.756793	0.616027	0.796613	0.783401
NN	0.753044	0.696031	0.757638	0.753277
RFC	0.752944	0.665034	0.776431	0.768092
GBC	0.756333	0.636512	0.787294	0.776611
GNB	0.756899	0.695493	0.758019	0.753401

4.2 Analysis and Discussion

Table 10 shows us a comparison between the dimensionality reduction algorithms used in this project and the classification model which performed better.

Table 10. Results for each dimensionality reduction algorithm. Light yellow represents the best result, dark yellow represents the second best result.

Dimensionality Reduction	Class Model	AUC	SENSITIVITY
LDA	LR	0.791076	0.668888
PCA	LR	0.791019	0.612776
No Dimensionality Reduction	LR	0.782738	0.678457
Univariate Feature Selection	LR	0.756793	0.616027
Random Forest	LR	0.753428	0.656858
Low Variance	LR	0.736918	0.640049
MRMR	LR	0.734658	0.626143
PCA	NN	0.788770	0.760683
Random Forest	NN	0.752433	0.679980
Low Variance	NN	0.740440	0.695888
No Dimensionality Reduction	GNB	0.791134	0.634184
Univariate Feature Selection	GNB	0.756899	0.695493
MRMR	GBC	0.735768	0.613932
LDA	DTC	0.782340	0.784193

The yellow rows represent the best **Auc** performance for this classifier algorithm and the gold color represents the second-best performance for the same classifier algorithm, obtaining a mean AUC between 0.73-0.8 with a sensitivity mean of 0.61-0.78, which are acceptable values for a Research in the medical-psychological field [94], [95].

The third value included in this table is the **sensitivity**, a value which expresses the true positive rates (probability of obtaining a positive prediction when the disease is present). If we take a look at our ROC curve, we would be able to find the **specificity** for this pair, corresponding to a particular decision threshold. The Sensitivity is shown only in an informative way because it is not the correct way of measuring a classifier algorithm, but it gives us an idea how well is going to predict the discriminative class in this experiment.

Clearly, the Feature extraction algorithms perform better on our binary dataset, with PCA and LDA in the top of the AUC ranking as we present below in table 11:

Table 11. Best results for a Dimensionality Reduction algorithm

Dimensionality Reduction	Class Model	AUC
No Dimensionality Reduction	GNB	0.791134
LDA	LR	0.791076
PCA	LR	0.791019
PCA	NN	0.788770

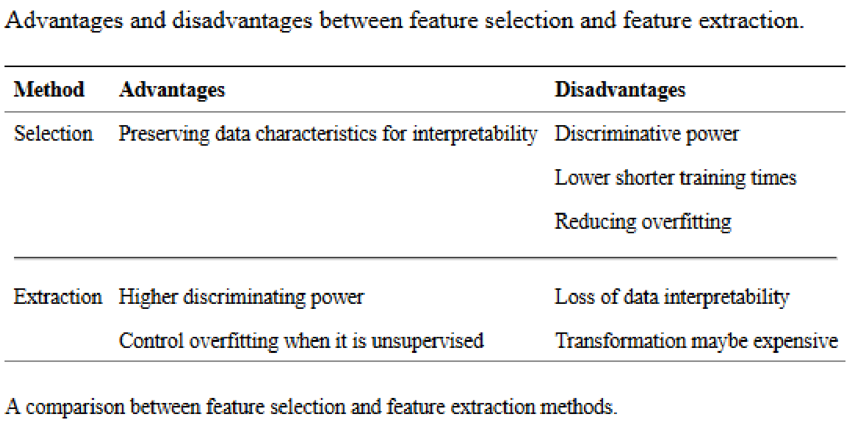
Why does feature extraction perform better on a binary classification problem? To answer this question, it is necessary to understand the difference between a **Generative and Discriminative algorithm**.

The **Generative models** try to lean the **model** that generates the data estimating the assumptions and distributions of the model. That means it tries to create a model (some characterizations of the entire population) of the positive values and another one with the negative values, looking for a boundary in space where one model becomes more likely than another model. It will be able to predict unseen data using this knowledge, assuming that the model learned captures the real model. Generative approaches use probabilistic methods to take decisions. They are able to use labeled and unlabeled data.

Discriminative efforts work different, it puts all the effort into modeling the **boundary** between the two classes. It is going to look closely at the points near to the boundary to make its decisions, therefore the discriminative model is going to ignore most of our data, just focusing on the important points. They can't really use unlabeled data and they are not going to be able to perform unsupervised tasks as generative models.

Discriminative approaches are usually more powerful when we have labeled data, providing a better discriminatory ability than the best-selected subsets by feature selection, but we are not going to be able to retain original data and we are going to lose interpretability[96]. It could be used both at the same time too, first applying feature extraction to create new features with higher discriminative power, and then with feature selection, discarding the features with low discriminative power, maybe a question for a future research[97].

Table 12. Feature Extraction VS Feature Selection



Advantages and disadvantages between feature selection and feature extraction.

Method	Advantages	Disadvantages
Selection	Preserving data characteristics for interpretability	Discriminative power
		Lower shorter training times
		Reducing overfitting
Extraction	Higher discriminating power	Loss of data interpretability
	Control overfitting when it is unsupervised	Transformation maybe expensive

A comparison between feature selection and feature extraction methods.

After this small explanation about Discriminative algorithms and Generative algorithms, we are going to come back to our binary classification problem.

Feature extraction algorithms will usually follow the discriminatory approach. As we aim to discriminate if a patient has a disease or not, our Feature Extraction algorithms are going to have always the advantage of obtaining a better and stable performance in terms of area under the ROC. On the other hand, the feature selection algorithms follow the generative approach and in this situation, it is going to be overtaken by feature extraction approaches.

Other experiments in this field (Masoumeh Zareapoor, 2015) are able to corroborate this experiment. In this case was compared the performance between the Feature extraction and selection algorithms on a binary classification problem, trying to classify emails correctly between spam or not. In this case, the results obtained were similar, giving a better AUC performance for Feature Extraction algorithm in different dimensions. See full citation [98].

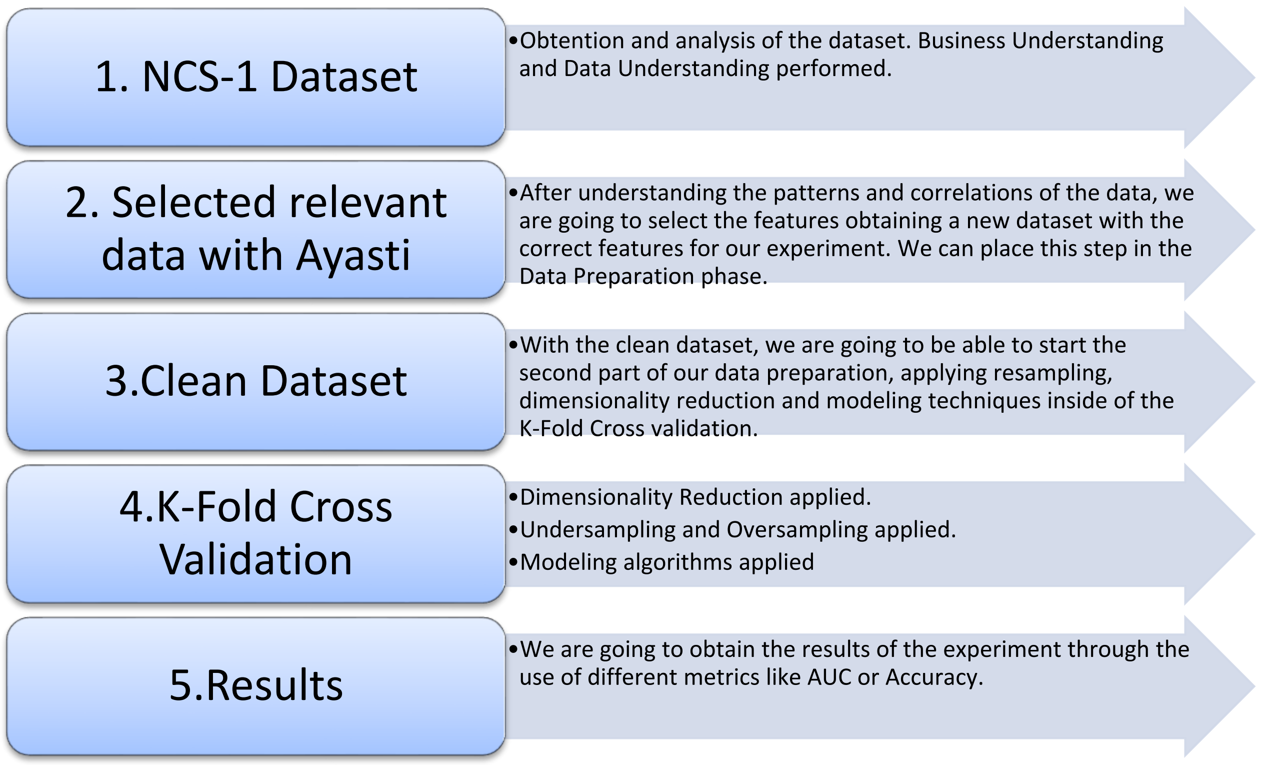

1. NCS-1 Dataset
•Obtention and analysis of the dataset. Business Understanding and Data Understanding performed.
2. Selected relevant data with Ayasti
•After understanding the patterns and correlations of the data, we are going to select the features obtaining a new dataset with the correct features for our experiment. We can place this step in the Data Preparation phase.
3.Clean Dataset
•With the clean dataset, we are going to be able to start the second part of our data preparation, applying resampling, dimensionality reduction and modeling techniques inside of the K-Fold Cross validation.
4.K-Fold Cross Validation
•Dimensionality Reduction applied.
•Undersampling and Oversampling applied.
•Modeling algorithms applied
5.Results
•We are going to obtain the results of the experiment through the use of different metrics like AUC or Accuracy.

Figure 150. Diagram with the process applied

Block 5

Conclusion

This experiment has explained the necessary process to carry out a reduction of dimensions, using different state of the art dimensionality reduction algorithms on a binary dataset seeking to answer a series of questions.

The first question we did is about the performance implementing a dimensionality reduction algorithm, instead of using Raw Data. We realized that using the appropriate dimensionality reduction algorithms, the obtained performance is going to be the same or even better because we are deleting the noise of the data and overfitting. We could appreciate this case applying **LDA with LR**, obtaining an **AUC of 0.791**. The same modeling algorithm was applied without using any dimensionality reduction algorithm and we obtained an **AUC of 0.782.** Observing the table 11, we could appreciate how the AUC is the same with 4 features, instead of the original 18 with an error of ~0.001 for the best results obtained in our experiment.

It is true that is going to have a cost, algorithms must be trained to understand the correlation between the different features of the data, and then reduce the dimensions. But it is going to be obtained a cleaner dataset with better features and a smaller number of dimension, giving a better interpretability, reducing the memory or computational resources needed and time fitting prediction models.

The second question refers to what kind of algorithm works better in our binary classification problem and why. We should understand first which kind of data we are going to reduce and the aim we are looking for. In this case, our Feature extraction algorithms perform better because we are working on a discriminative problem. But sometimes our feature extraction algorithm won't fit our wishes, for example, if we want to reduce the data to obtain the most important features, to understand for example why the economy of Europe is decreasing, we would like to keep the original data, instead of creating new data, losing interpretability.

In this case PCA reduces the feature space and lower the complexity of classification, keeping the main features of the dataset in a smaller one with fewer dimensions, however, some useful information is lost during reduction. Experiments and results show that the performance is the same one, reducing the dimensions by 78%.

Experiments and evaluation metrics show that the performance with other dimensionality reduction algorithms is similar in comparison to the original dataset, even when a 78% of the dimensions is reduced from the dataset, which leads to lower the cost and complexity of classification process, improving the performance.

We hope that with this project, students and teachers can begin to understand the process to analyze different datasets, in our case a medical dataset. The project remains open, since the technique we have followed has given us good results but not excellent, so they can be improved and modified in any case, allowing to extend this research. The main idea has been covered, explaining how to apply dimensionality reduction algorithms, but in the background of all this we can go beyond, not only by modifying the dataset applying classification algorithms to predict, but obtaining connections between the different values of our dataset. Why does a person have a disease or not? Why a value grows and at the same time, another value grows too or diminishes? Could we predict a disease in advance by knowing your medical history? All these processes can be applied to different frameworks but in the medicine field, it could revolutionize the industry we know and not only being a tool for patients or giving support to doctors, but reducing mortality in whole populations in many cases.

As Vinod Khosla says, “*Eighty percent of what doctors do, tech can do at a fraction of the cost — especially your rural doctor in India*” [99]. It doesn’t mean top doctors are going to be replaced anytime soon, but we think that in the next years the Data Science field is going to continue growing, improving the algorithms and software used, and changing drastically the health-care business.

List of references

[1] Forman,G. (2003). ‘*An Extensive Empirical Study of Feature Selection Metrics for Text Classification*’, *J. Mach. Learn. Res.*, vol. 3 (no. Mar), pp. 1289–1305.

[2] Y. Saeys, (2007) ‘*A review of feature selection techniques in bioinformatics*’, *Bioinformatics,* vol. 23 (Issue 19), pp. 2507-2517.

[3] Wang, S., Zhang, Y., Liu, G., Phillips, P. and Yuan, T. (2015). Detection of Alzheimer’s Disease by Three-Dimensional Displacement Field Estimation in Structural Magnetic Resonance Imaging. *Journal of Alzheimer's Disease,* 50(1), pp.233-248.

[4] *Big Data needs Data Science - Hortonworks*. (2013). *Hortonworks*, from https://es.hortonworks.com/blog/big-data-needs-data-science/. Accessed 21 August 2017

[5] Smith, D., & Smith, +. (2013). *Explore smartphone market share with Nanocubes*. *Revolutions*, from http://blog.revolutionanalytics.com/2013/08/explore-smartphone-market-share-with-nanocubes.html. Accessed 21 August 2017

[6] Smith, D., & Smith, +. (2012). *Analytic applications are built by data scientists*. *Revolutions*, from http://blog.revolutionanalytics.com/2012/02/analytic-applications-are-built-by-data-scientists.html. Accessed 21 August 2017

[7] Smith, D., & Smith, +. (2012). *How Nate Silver won the election with Data Science*. *Revolutions*, from http://blog.revolutionanalytics.com/2012/11/in-the-2012-election-data-science-was-the-winner.html. Accessed 21 August 2017

[8] Smith, D., & Smith, +. (2011). *Because it's Friday: Data Mining Wine*. *Revolutions*, from http://blog.revolutionanalytics.com/2011/11/because-its-friday-data-mining-wine.html. Accessed 21 September 2017

[9] S.Batia, J.(2016)‘What is difference between Big Data and Machine Learning? - Quora’. [Online]., Available: https://www.quora.com/What-is-difference-between-Big-Data-and-Machine-Learning. Accesed 16-07-2017

[10] Guerra, L. (2016). *Feature engineering? Start here!*. *Datasciencecentral.com*., from http://www.datasciencecentral.com/profiles/blogs/feature-engineering-start-here. Accessed 17 September 2017

[11] Amatriain, X. (2015)‘What’s the relationship between machine learning and data mining? - Quora’. [Online]., Available: https://www.quora.com/Whats-the-relationship-between-machine-learning-and-data-mining. Accessed 22 August 2017

[12] Kearn, M. (2016). *Machine Learning is for Muggles too!*. *Martin Kearn*., from https://blogs.msdn.microsoft.com/martinkearn/2016/03/01/machine-learning-is-for-muggles-too/. Accessed 16 September 2017

[13] Williamson, J. (undated). *The 4 V’s of Big Data - dummies*. *dummies*., from http://www.dummies.com/careers/find-a-job/the-4-vs-of-big-data/. Accessed 11 August 2017

[14] *The Four V's of Big Data*. (undated). *IBM Big Data & Analytics Hub*., from http://www.ibmbigdatahub.com/infographic/four-vs-big-data. Accessed 21 September 2017

[15] Jhajj, R. (2016). *The Hadoop Ecosystem Explained*. *Examples Java Code Geeks*., from https://examples.javacodegeeks.com/enterprise-java/apache-hadoop/hadoop-ecosystem-explained/. Accessed 21 September 2017

[16] *Vardan*. (undated) *Apache Spark vs Hadoop MapReduce*. *Edureka Blog*., from https://www.edureka.co/blog/apache-spark-vs-hadoop-mapreduce. Accessed 11 August 2017.

[17] Savinkin, I. (2016). *Big Data, Data Analytics, Data Mining, Data Science, Machine Learning*. *Web Scraping*., from http://scraping.pro/data-analytics-data-analysis-data-mining-data-science-machine-learning-big-data/. Accessed 05 August 2017

[18] Chandrasekaran, S. (2013). *Becoming a Data Scientist – Curriculum via Metromap ← Pragmatic Perspectives*. *Nirvacana.com*., from http://nirvacana.com/thoughts/becoming-a-data-scientist/. Accessed 17 September 2017

[19] A. L. Burtch, B. W. L. Burtch, and B. Works. (undated). *Tell Your Kids to Be Data Scientists, Not Doctors*. *WIRED*., from https://www.wired.com/insights/2014/06/tell-kids-data-scientists-doctors/. Accessed 23 August 2017

[20] Shankar Siva Kumar, R., & Rioux, C. (2016). *4 trends in security data science for 2017*. *O'Reilly Media*., from https://www.oreilly.com/ideas/4-trends-in-security-data-science-for-2017. Accessed 23 August 2017

[21] Stelmaszak, M., & Hukal, P. (2017). *When data science meets social sciences: the benefits of the data revolution are clear but careful reflection is needed*. *Impact of Social Sciences*., from http://blogs.lse.ac.uk/impactofsocialsciences/2017/03/01/when-data-science-meets-social-sciences-the-benefits-of-the-data-revolution-are-clear-but-careful-reflection-is-needed/. Accessed 7 July 2017

[22] *Data Science in Defense and Counterterrorism | Careers in Defense for Masters-Prepared Data Scientists*. (undated). *Datasciencegraduateprograms.com*., from http://www.datasciencegraduateprograms.com/defense-and-counterterrorism/. Accessed 01 July 2017

[23] Plesser, A. (2017). *(VIDEO) For Modern Businesses, Data Is King, IBM's Bob Lord*. *HuffPost*., from http://www.huffingtonpost.com/entry/video-for-modern-businesses-data-is-king-ibms_us_599ca508e4b056057bddcf25. Accessed 14 September 2017

[24] Lehe, D. (2017). *Birmingham needs more graduate education programs to plug the brain drain*. *AL.com*., from http://www.al.com/opinion/index.ssf/2017/08/plugging_the_brain_drain_start.html. Accessed 15 August 2017

[25] Wittenberg, D. (2017). *Concur Technologies, Inc. (NASDAQ:CNQR), Netsuite Inc (NYSE:N) - 5 InsurTech Startups Changing The Game In The Bay Area*. *Benzinga*., from https://www.benzinga.com/fintech/17/08/9956914/5-insurtech-startups-changing-the-game-in-the-bay-area. Accessed 28 June 2017

[26] Pedraza, B. (2017). *Cite a Website - Cite This For Me*. *Searchenginejournal.com*., from https://www.searchenginejournal.com/data-science-basics/209858/. Accessed 2 July 2017

[27] McGowan, B. (2017). *Data science key to Monsanto improving its supply chain*. *CIO*., from https://www.cio.com/article/3214664/data-science/data-science-key-to-monsanto-improving-its-supply-chain.html. Accessed 10 August 2017

[28] *Machine Learning in MATLAB - MATLAB & Simulink - MathWorks United Kingdom*. (undated). *De.mathworks.com*., from https://de.mathworks.com/help/stats/machine-learning-in-matlab.html?requestedDomain=de.mathworks.com. Accessed 13 July 2017

[29] Bhatia, R. (2017). *Forbes Welcome*. *Forbes.com*., from https://www.forbes.com/sites/forbestechcouncil/2017/08/07/what-is-machine-learning/. Accessed 23 August 2017

[30] Marsman, J. *How to win a hackathon using Azure Machine Learning*. (2015). *Jennifer Marsman*., from https://blogs.msdn.microsoft.com/jennifer/2015/09/10/how-to-win-a-hackathon-using-azure-machine-learning/. Accessed 17 September 2017

[31] Burgess, M. (2017). *How Facebook is using AI to tackle terrorism online*. *WIRED UK*., from http:// http://www.wired.co.uk/article/facebook-ai-terrorism. Accessed 24 August 2017

[32] *How to choose machine learning algorithms*. (2017). *Docs.microsoft.com*., from https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice. Accessed 18 September 2017

[33] Pilotte, P. (2016). *Cite a Website - Cite This For Me*. *Embedded-computing.com*., from http://www.embedded-computing.com/embedded-computing-design/analytics-driven-embedded-systems-part-2-developing-analytics-and-prescriptive-controls. Accessed 18 September 2017

[34] Chamaki, F. (2016). *Franki Chamaki» Blog Archive » Data-driven market segmentation – more effective marketing to segments using AI*. *Frankichamaki.com*., from http://www.frankichamaki.com/data-driven-market-segmentation-more-effective-marketing-to-segments-using-ai/. Accessed 18 July 2017

[35] Landman, N., & Pang, H. (undated). *k-Means Clustering | Brilliant Math & Science Wiki*. *Brilliant.org*., from https://brilliant.org/wiki/k-means-clustering/. Accessed 17 August 2017

[36] Jain, K., & newbie, M. (2015). *Machine Learning Basics For A Newbie | Machine Learning Applications*. *Analytics Vidhya*., from https://www.analyticsvidhya.com/blog/2015/06/machine-learning-basics/. Accessed 11 June 2017

[37] *Data mining | Better Evaluation*. (undated). *Betterevaluation.org*., from http://www.betterevaluation.org/en/evaluation-options/data_mining. Accessed 4 June 2017

[38] Domingos, P. (2012). A few useful things to know about machine learning. *Communications Of The ACM*, *55*(10), 78. http://dx.doi.org/10.1145/2347736.2347755.

[39] Brownlee, J. (2014). *Discover Feature Engineering, How to Engineer Features and How to Get Good at It - Machine Learning Mastery*. *Machine Learning Mastery*., from https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/. Accessed 10 June 2017

[40] Carreria-Perpinan, M. (2017). *Dimensionalit y reduction*., from http://faculty.ucmerced.edu/mcarreira-perpinan/papers/phd-ch04.pdf. Accessed 9 July 2017

[41] *National Comorbidity Survey: Baseline (NCS-1), 1990-1992*. (2017). *Icpsr.umich.edu*. Accessed 29 May 2017, from http://www.icpsr.umich.edu/icpsrweb/ICPSR/studies/6693.

[42] L. Breiman, (2001), ‘Random Forests’, *Mach. Learn.*, vol. 45, no. 1, pp. 5–32.

[43] B. H. Menze *et al.*, (2009) ‘A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data’, *BMC Bioinformatics*, vol. 10, p. 213.

[44] *3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.19.0 documentation*. (undated). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed 22 July 2017

[45] *sklearn.feature_selection.VarianceThreshold — scikit-learn 0.19.0 documentation*. (undated). *Scikit-learn.org*., from http://scikit-

learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html#sklearn.feature_selection.VarianceThreshold Accessed 18 July 2017.

[46] *jundongl/scikit-feature*. (undated). *GitHub*., from https://github.com/jundongl/scikit-feature. Accessed 3 July 2017

[47] *jundongl/scikit-feature*. (undated). *GitHub*., from https://github.com/jundongl/scikit-feature/blob/master/skfeature/example/test_MRMR.py. Accessed 11 August 2017

[48] *sklearn.feature_selection.SelectKBest — scikit-learn 0.19.0 documentation*. (undated). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest. Accessed 13 July 2017

[49] M. Palmer, (undated),*Principal Components Analysis, Ordination.okstate.edu*., from http://ordination.okstate.edu/PCA.htm. Accessed 10 July 2017

[50] *sklearn.decomposition.PCA — scikit-learn 0.19.0 documentation*. (undated). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA. Accessed 29 August 2017

[51] *Practical Guide to Principal Component Analysis (PCA) in R & Python*. (2016). *analyticsvidhya*., from https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/. Accessed 18 June 2017

[52] *numpy.cumsum — NumPy v1.13 Manual*. (undated). *Docs.scipy.org*., from https://docs.scipy.org/doc/numpy/reference/generated/numpy.cumsum.html. Accessed 12 July 2017

[53] Raschka, S. (2014). *Implementing a Principal Component Analysis (PCA)*. *Sebastian Raschka's Website*., from http://sebastianraschka.com/Articles/2014_pca_step_by_step.html. Accessed 3 July 2017

[54] Willems, K. (2017). *Python Machine Learning: Scikit-Learn Tutorial*. *DataCamp Community*., from http://www.datacamp.com/community/tutorials/machine-learning-python. Accessed 17 June 2017

[55] Hamilton, L. (2014). *Introduction to Principal Component Analysis (PCA) - Laura Diane Hamilton*. *Lauradhamilton.com*., from http://www.lauradhamilton.com/introduction-to-principal-component-analysis-pca. Accessed 10 July 2017

[56] Raschka, S. (2014). *Linear Discriminant Analysis*. *Sebastian Raschka's Website*., from http://sebastianraschka.com/Articles/2014_python_lda.html. Accessed 3 August 2017.

[57] Welling, M. *Fisher Linear Discriminant Analysis*., from https://www.ics.uci.edu/~welling/teaching/273ASpring09/Fisher-LDA.pdf. Accessed 19 July 2017.

[58] *1.10. Decision Trees — scikit-learn 0.19.0 documentation*. (undated). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/tree.html. Accessed 11 June 2017.

[59] *3.2.4.3.3. sklearn.ensemble.ExtraTreesClassifier — scikit-learn 0.19.0 documentation*. (undated). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html#sklearn.ensemble.ExtraTreesClassifier. Accessed 4 July 2017.

[60] *sklearn.neural_network.MLPClassifier — scikit-learn 0.19.0 documentation*. (undated). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. Accessed 13 August 2017.

[61] *3.2.4.3.5. sklearn.ensemble.GradientBoostingClassifier — scikit-learn 0.19.0 documentation*. (Untitled). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html. Accessed 16 July 2017.

[62] *Probability calibration of classifiers — scikit-learn 0.19.0 documentation*. (Untitled). *Scikit-learn.org*., from http://scikit-learn.org/stable/auto_examples/calibration/plot_calibration.html. Accessed 7 August 2017.

[63] *sklearn.naive_bayes.GaussianNB — scikit-learn 0.19.0 documentation*. (Untitled). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html. Accessed 7 August 2017.

[64] *sklearn.calibration.CalibratedClassifierCV — scikit-learn 0.19.0 documentation*. (Untitled). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html. Accessed 7 August 2017.

[65] Fawcett, T. (2016). *Learning from Imbalanced Classes - Silicon Valley Data Science*. *Silicon Valley Data Science*., from https://svds.com/learning-imbalanced-classes/. Accessed 10 July 2017.

[66] Chawla, N., & Bowyer, K. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*. *Cs.cmu.edu*., from https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/chawla02a-html/chawla2002.html. Accessed 20 August 2017.

[67] Pozzolo, A., & Caelen, O. (2015). *When is undersampling effective in unbalanced classification tasks?*. *Slideshare.net*., from https://www.slideshare.net/dalpozz/when-is-undersampling-effective-in-unbalanced-classification-tasks. Accessed 19 August 2017.

[68] *scikit-learn-contrib/imbalanced-learn*. (Untitled). *GitHub*., from https://github.com/scikit-learn-contrib/imbalanced-learn. Accessed 17 July 2017.

[69] Estabrooks, A., Jo, T., & Japkowicz, N. (2004). *A Multiple Resampling Method for Learning from Imbalanced Data Sets*., from http://onlinelibrary.wiley.com/doi/10.1111/j.0824-7935.2004.t01-1-00228.x/abstract. Accessed 20 August 2017.

[70] Batista, G., Prati, R., & Monard, M. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6*(1), 20. http://dx.doi.org/10.1145/1007730.1007735.

[71] *General examples — imbalanced-learn 0.3.0 documentation*. (Untitled). *Contrib.scikit-learn.org*., from http://contrib.scikit-learn.org/imbalanced-learn/stable/auto_examples/index.html#example-using-over-sampling-class-methods. Accessed 20 August 2017.

[72] Raschka, S. (2017). *holdout-vs-2fold*. Retrieved from https://sebastianraschka.com/images/blog/2016/model-evaluation-selection-part3/holdout-vs-2fold.png.

[73] *Why do researchers use 10-fold cross validation instead of testing on a validation set?*. (2017). *Stats.stackexchange.com*., from https://stats.stackexchange.com/questions/49692/why-do-researchers-use-10-fold-cross-validation-instead-of-testing-on-a-validati. Accessed 10 July 2017.

[74] Brownlee, J. (2016). *Metrics To Evaluate Machine Learning Algorithms in Python - Machine Learning Mastery*. *Machine Learning Mastery*., from

https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/. Accessed 25 May 2017.

[75] *sklearn.metrics.roc_curve — scikit-learn 0.19.0 documentation*. (Untitled). *Scikit-learn.org*., from http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html. Accessed 18 August 2017.

[76] *ROC Analysis - ML Wiki*. (2017). *Mlwiki.org*., from http://mlwiki.org/index.php/ROC_Analysis. Accessed 19 August 2017.

[77] Markham, K. (2014). *Simple guide to confusion matrix terminology*. *Data School*., from http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/. Accessed 15 July 2017.

[78] *Course | 6.00.1x | edX*. (2017). *Courses.edx.org*., from https://courses.edx.org/courses/course-v1:MITx+6.00.1x+2T2017/course/. Accessed 23 September 2017.

[79] *Downloads*. (2017). *Anaconda*., from https://www.anaconda.com/download/. Accessed 30 August 2017.

[80] *Project Jupyter*. *Jupyter.org*., from http://www.jupyter.org. Accessed 30 August 2017.

[81] *1.3.1. The NumPy array object — Scipy lecture notes*. *Scipy-lectures.org*., from http://www.scipy-lectures.org/intro/numpy/array_object.html. Accessed 30 August 2017.

[82] *SciPy.org — SciPy.org*. *Scipy.org*., from https://www.scipy.org/. Accessed 31 August 2017.

[83] *Python Data Analysis Library — pandas: Python Data Analysis Library*. *Pandas.pydata.org*., from http://pandas.pydata.org/. Accessed 30 August 2017.

[84] *Matplotlib: Python plotting — Matplotlib 2.0.2 documentation*. *Matplotlib.org*., from http://matplotlib.org/. Accessed 30 August 2017.

[85] *Matplotlib Style Gallery*. *Tonysyu.github.io*., from https://tonysyu.github.io/raw_content/matplotlib-style-gallery/gallery.html. Accessed 30 August 2017.

[86] *SymPy Gamma*. *Sympygamma.com*., from http://www.sympygamma.com/. Accessed 30 August 2017.

[87] *Cite a Website - Cite This For Me*. *Scikit-learn.org*., from http://scikit-learn.org/stable/. Accessed 30 August 2017.

[88] , from http://Free guide to using CRISP DM to evaluate data mining tools . Accessed 30 August 2017.

[89] Marbn, s., Mariscal, G., & Segovi, J. (2009). *A Data Mining &amp; Knowledge Discovery Process Model*., from https://www.intechopen.com/books/data_mining_and_knowledge_discovery_in_real_life_applications/a_data_mining__amp__knowledge_discovery_process_model. Accessed 14 July 2017

[90] Villena Román, J. (2016). *s|ngular - CRISP-DM: La metodología para poner orden en los proyectos de Data Science*. *Data.sngular.team*., from https://data.sngular.team/es/art/25/crisp-dm-la-metodologia-para-poner-orden-en-los-proyectos-de-data-science. Accessed 9 September 2017.

[91] *Artificial Intelligence, Machine Intelligence and Machine Learning | Ayasdi*. (2017). *Ayasdi*., from https://www.ayasdi.com/. Accessed 10 May 2017

[92] Brownlee, J. (2016). *How To Compare Machine Learning Algorithms in Python with scikit-learn - Machine Learning Mastery*. *Machine Learning Mastery*., from

https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/. Accessed 1 June 2017

[93] Naseriparsa, M., & Mansour Riahi Kashani, M. (2013). Combination of PCA with SMOTE Resampling to Boost the Prediction Rate in Lung Cancer Dataset. *International Journal Of Computer Applications*, *77*(3), 33-38. http://dx.doi.org/10.5120/13376-0987.

[94] Roli, L., Pecoraro, V., & Trenti, T. (2017). Can NGAL be employed as prognostic and diagnostic biomarker in human cancers? A systematic review of current evidence. *The International Journal Of Biological Markers*, *32*(1), 0-0. http://dx.doi.org/10.5301/jbm.5000245.

[95] Ranjan Prusty, M. (2015)., from https://www.researchgate.net/post/What_is_the_value_of_the_area_under_the_roc_curve_AUC_to_conclude_that_a_classifier_is_excellent. Accessed 10 August 2017.

[96] Zheng, H., & Zhang, Y. (2008). Feature selection for high-dimensional data in astronomy. *Advances In Space Research*, *41*(12), 1960-1964. http://dx.doi.org/10.1016/j.asr.2007.08.033.

[97] Hira, Z., & Gillies, D. (2015). A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data. *Advances In Bioinformatics*, *2015*, 1-13. http://dx.doi.org/10.1155/2015/198363.

[98] Zareapoor, M., & K. R, S. (2015). Feature Extraction or Feature Selection for Text Classification: A Case Study on Phishing Email Detection. *International Journal Of Information Engineering And Electronic Business*, *7*(2), 60-65. http://dx.doi.org/10.5815/ijieeb.2015.02.08.

[99] Lardinois, F. (2013). *Vinod Khosla: In The Next 10 Years, Data Science Will Do More For Medicine Than All Biological Sciences Combined*. *TechCrunch*., from https://techcrunch.com/2013/09/11/vinod-khosla-in-the-next-10-years-data-science-will-do-more-for-medicine-than-all-biological-sciences-combined/. Accessed 23 September 2017