Omar Alejandro Garduza Riveroll

Benemérita Universidad Autónoma de Puebla

Ph. D. José Arturo Olvera López

03/06/20

Abstract

This is a Data Mining project, that contains a basic implemented ensemble learning method for classification. This project is related to the phases of Preprocessing, Knowledge extraction and Visualization from the process of KDD (Knowledge Discovery in Databases)

BASIC ENSEMBLE LEARNING METHOD

Data Mining

ÍNDICE

INTRODUCTION

***Data mining*** (Saxena, n.d.) in general terms means mining or digging deep into data, which is in different forms to gain patterns, and to gain knowledge on that pattern. In the process of data mining, large data sets are first sorted, then patterns are identified, and relationships are established to perform data analysis and solve problems.

***Classification*** (Saxena, n.d.)is a data analysis task, for instance: the process of finding a model that describes and distinguishes data classes and concepts. Classification is the problem of identifying to which of a set of categories (subpopulations), a new observation belongs to, based on a training set of data containing observations and whose categories membership are known.

***Ensemble learning*** (Polikar, 2008)is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a specific computational intelligence problem. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one. Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting.

***Cross-validation*** (Brownlee, 2018)is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

The current document is a report of a basic ensemble learning method development process, which is divided into the following sections:

* Data Analysis
* [PCA](https://builtin.com/data-science/step-step-explanation-principal-component-analysis) (Principal Component Analysis)
* Classification models and Cross Validation
* Building the ensemble learning method
* Plotting the results

Note: For your knowledge, the Python programming language was used to develop and program all the following notes and codes, using the *Jupyter Notebook* development tool for Python.

1. Data Analysis

This section is focused mainly on the following points:

* Reading from file
* Plotting the data into histograms
* Data Summary
* Outliers detection

The data set file structure must be as follows:

1. number\_of\_samples
2. number\_of\_features
3. number\_of\_different\_class\_targets
4. *samples* (**the samples must be described with all its attributes separated by a single coma and each new sample input in a new line**)

Here is a short example of a valid file:

5

5

3

140,125,0,0,0,1

12,34,4.5,0,0,2

34,137,0,0,0,2

220,149,0,0,0,3

232,149,0,0,0,1

* 1. First steps

First we need to read the file and retreive the data, for the following notes, they will be used 2 different data files, one with 300 input samples, 150 features and 4 class targets, which we will call **File A** for practical purpouses, and the other one with 409 input samples, 385 features and 6 class targets, which we’ll call **File B**.

* 1. Data Summary

We will create a data summary for each feature in our data set. The functions included in the data summary section are:

* Minimum of a data set
* First quartile – 25th Percentile
* Median
* Mean
* Third quartile – 75th Percentile
* Maximum of a data set
  1. Turning information into histograms and outlier detection

For the outliers detection it was used the Interquartile rangers metric (IQR). With a K value of 1.5. So, the range for acceptance will be:

Note: Some of the features had values that barely variated from the range between 0 and 1, or practically had no variation at all.

The original file use for this project contained 19 features and so that, here are some examples of the graphics created. First we will show a few graphics from the File A:

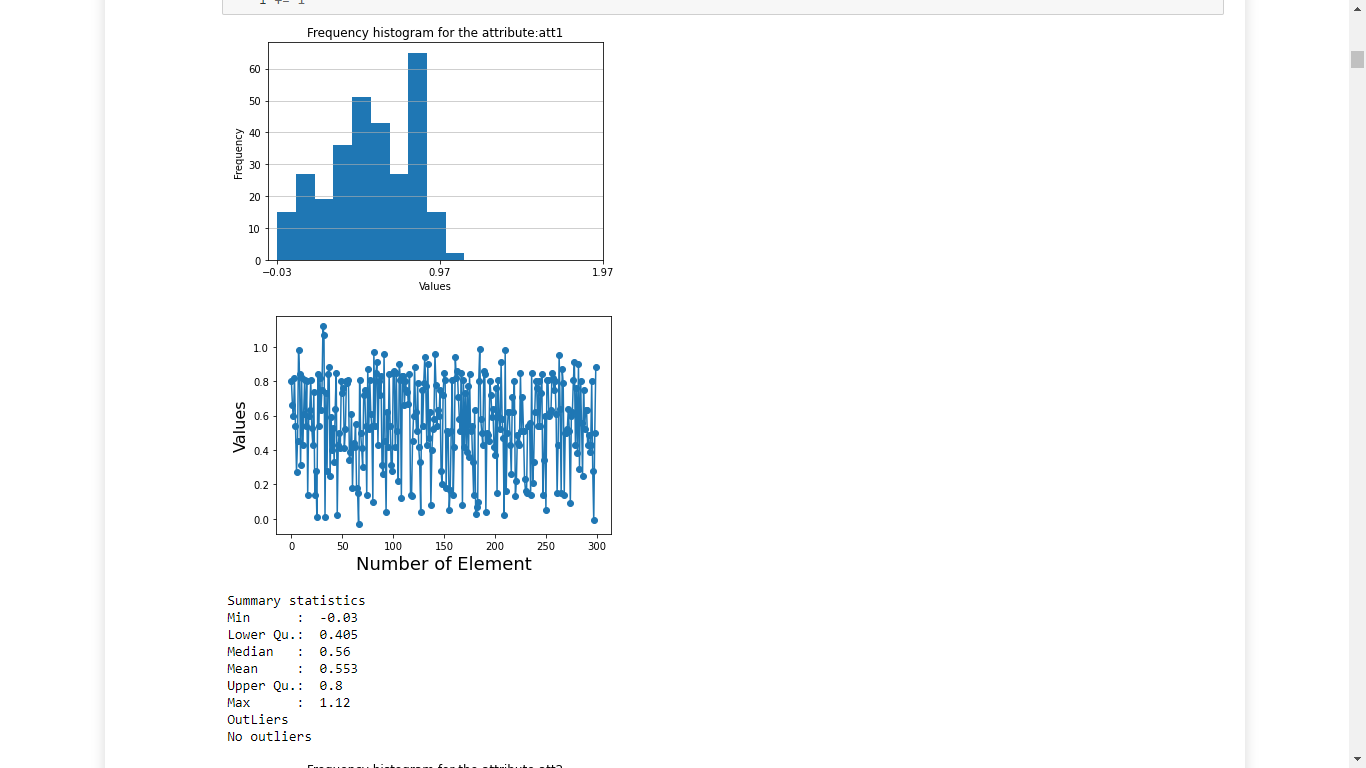


Figure 1 Frequency histogram for the att1 data in the File A

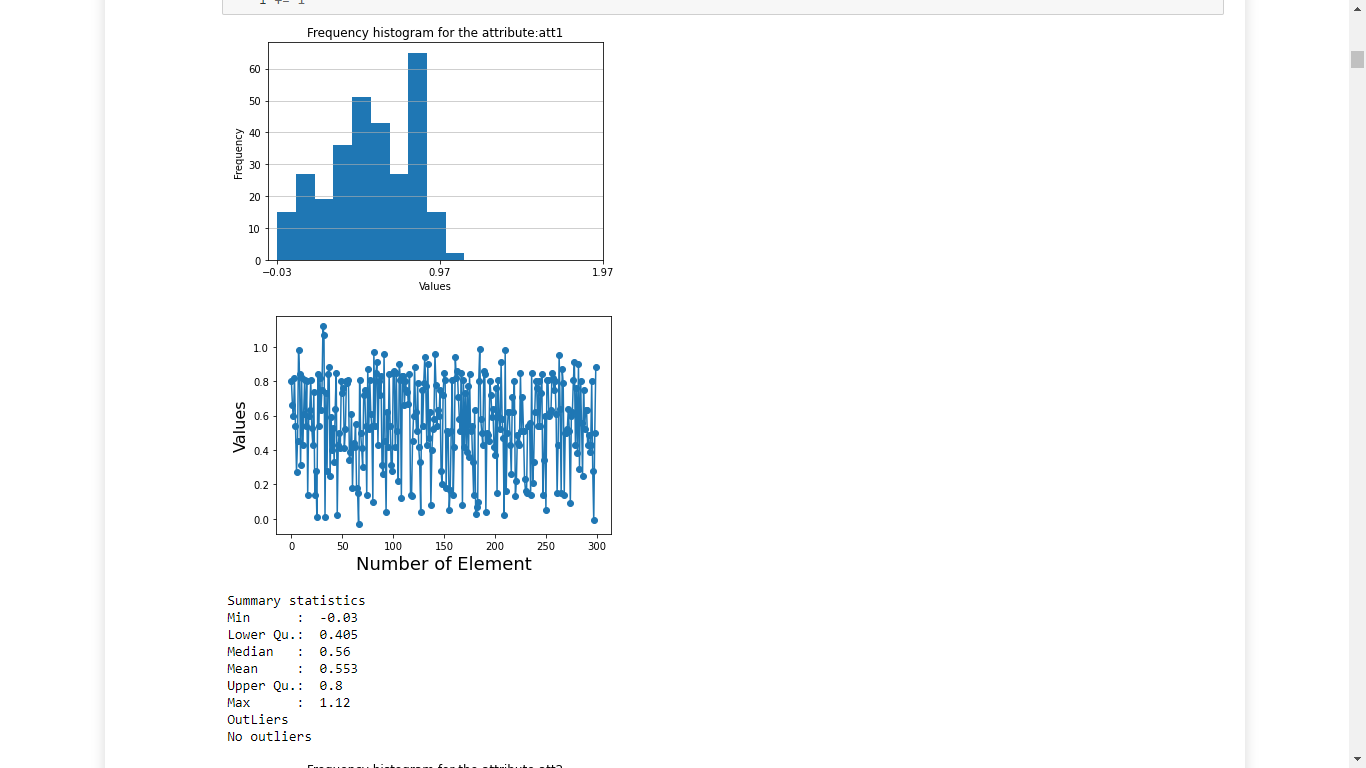


Figure 2 Scatter plot for the att1 data in the File A

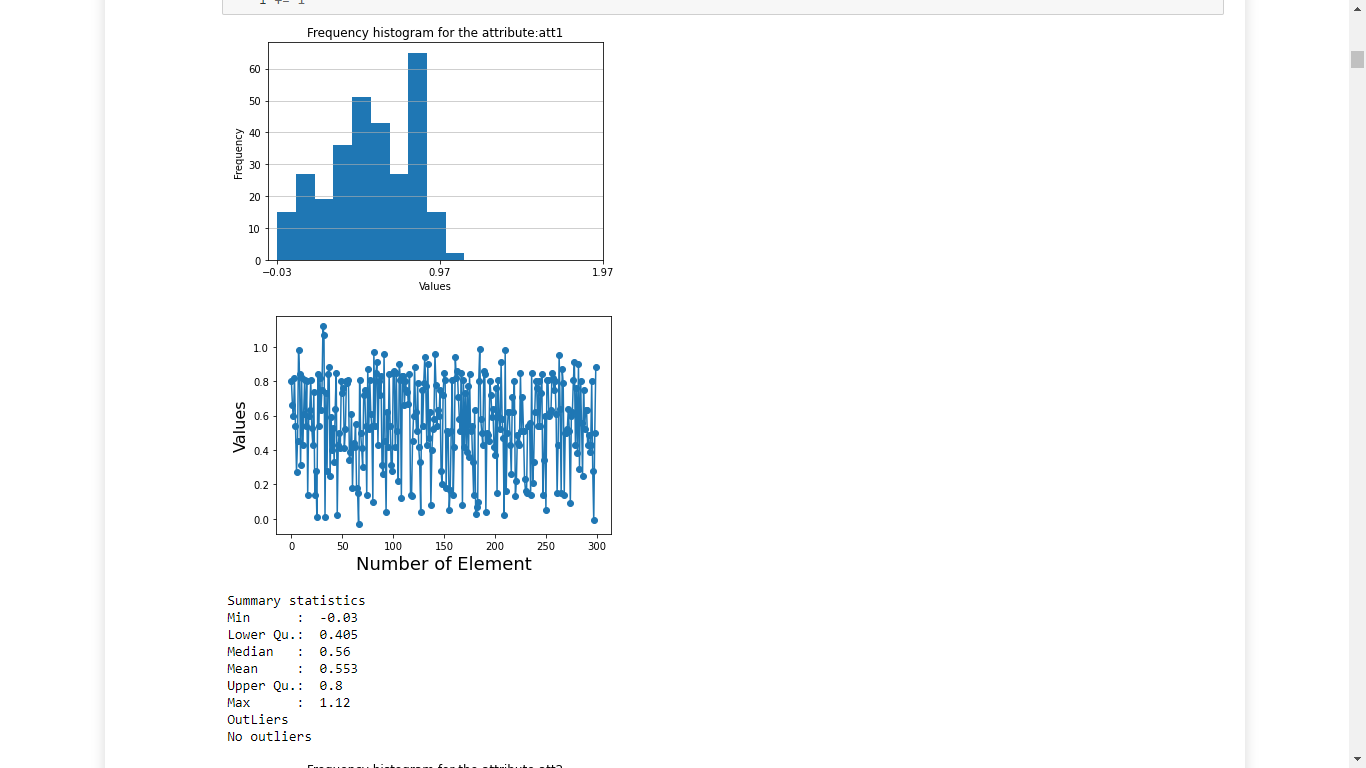


Figure 3 Summary statistics for the att1 data in the File A

For these kinds of features, which their distribution was quite “normal”, in a matter of speaking, there are no outliers, because the data is well distributed between all the range of values. There is very little deviation in the features of the File A. Most of them barely a few outliers or even none, like the example before.

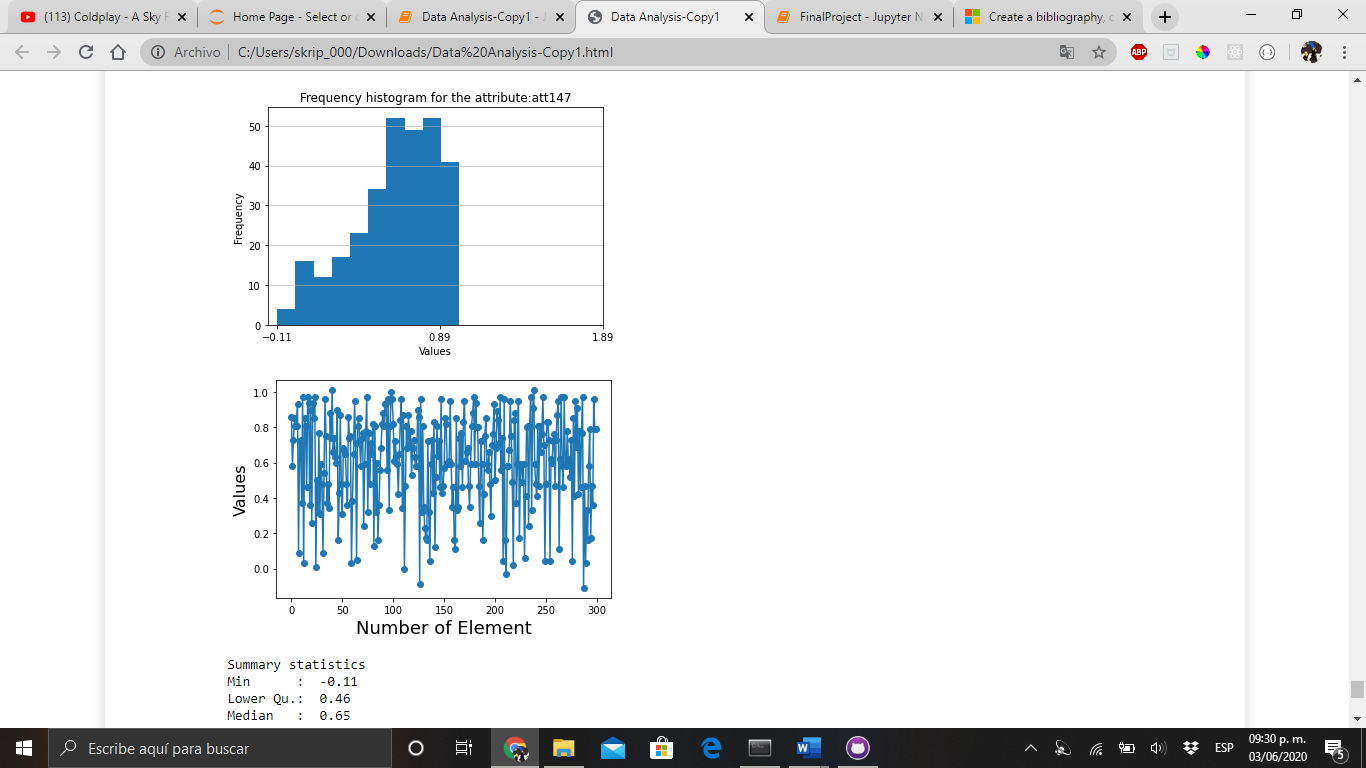


Figure 4 Frequency histogram for the att147 data in the File A

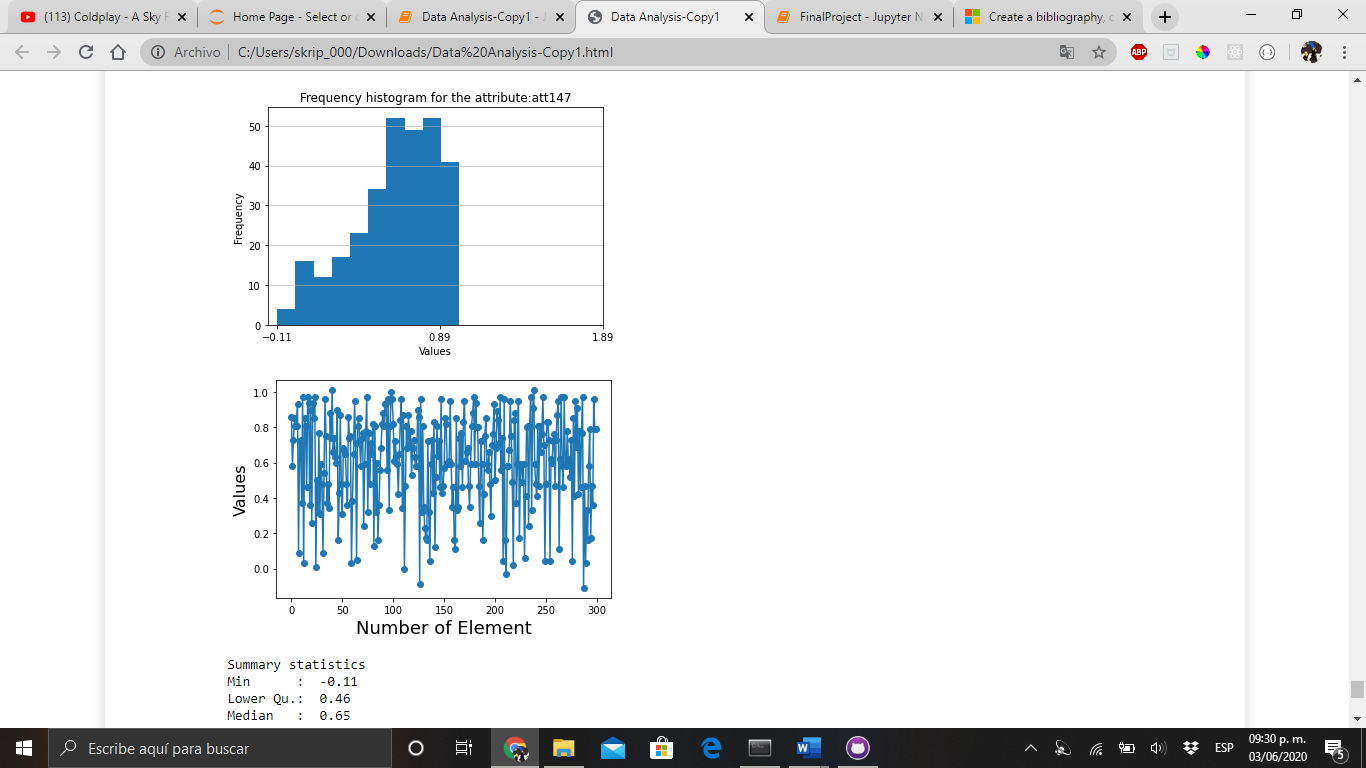


Figure 5 Scatter plot for the att147 data in the File A

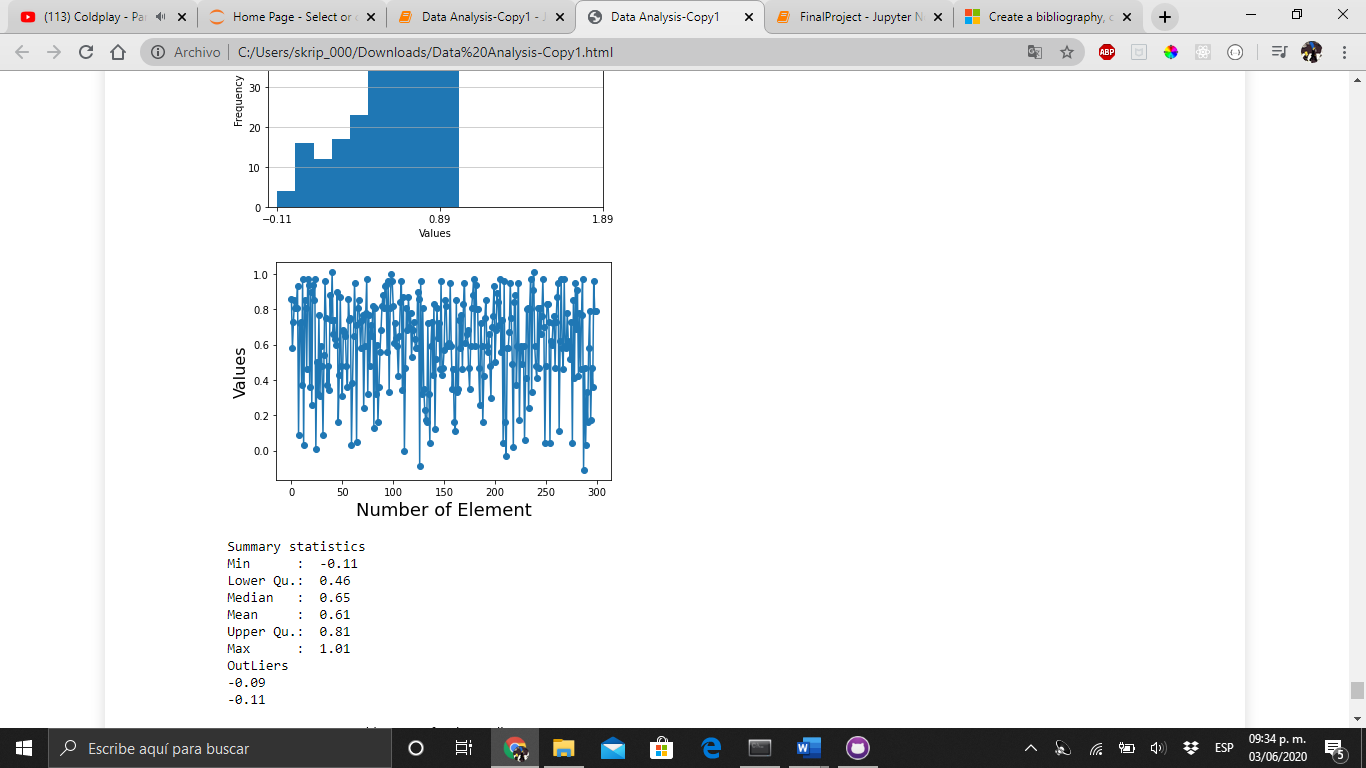


Figure 6 Summary statistics for the att147 data in the File A

Now let’s take a look to File B features:

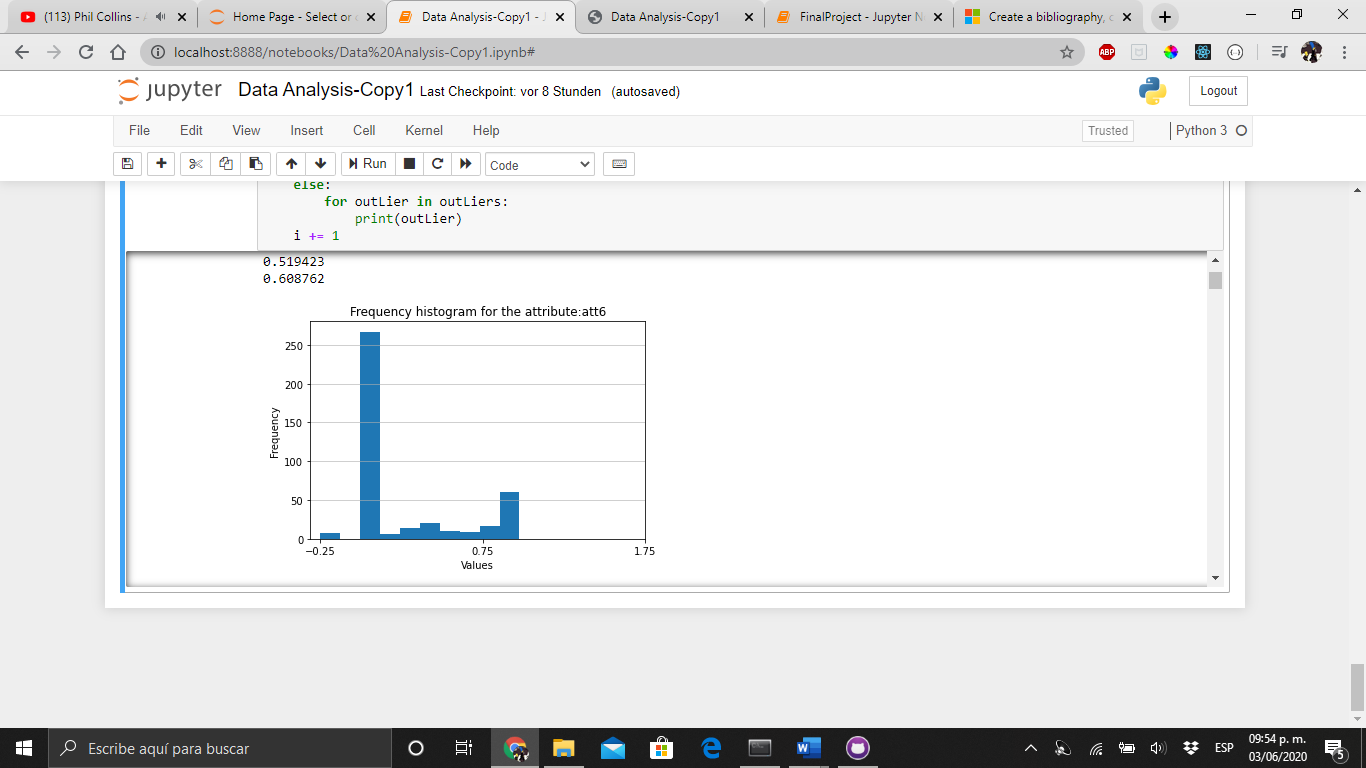


Figure 7 Frequency histogram for the att6 data in the File B

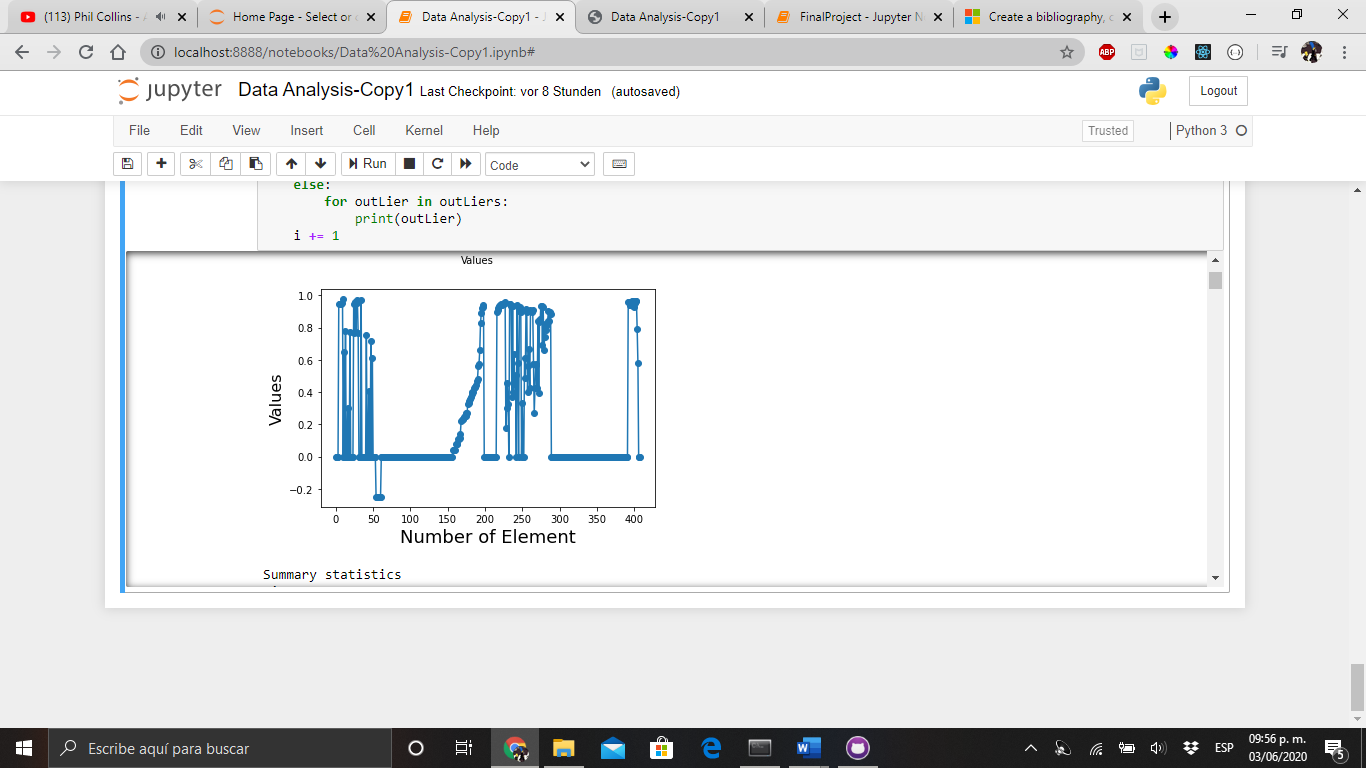


Figure 8 Scatter plot for the att6 data in the File B

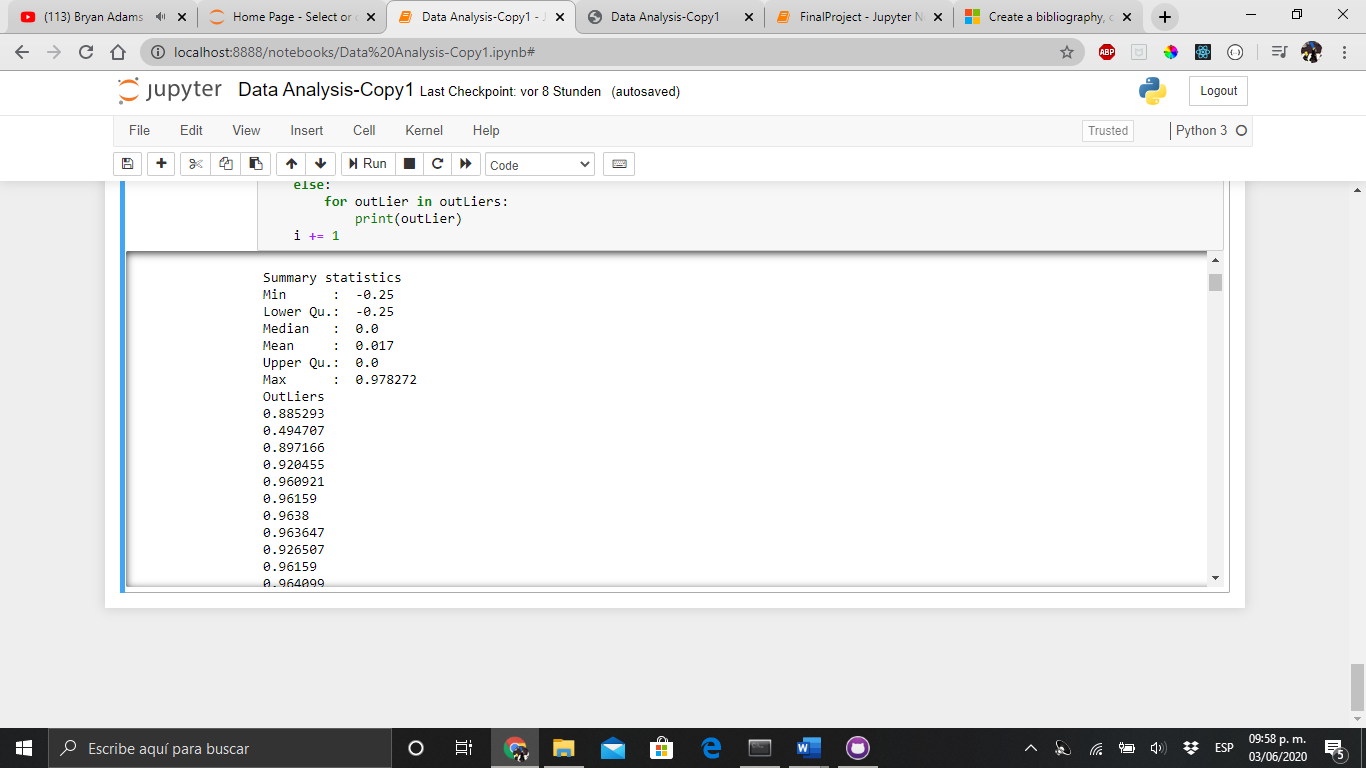


Figure 9 Summary statistics for the att6 data in the File B

Here we can see that most of the features had outliers in their data, and the deviation was bigger in this file than in the other.

1. Principal Component Analysis (PCA)
   1. Create the principal components

We ask the user to insert how many components he wants, in order to perform the PCA. For this example, we will try the File A and B with 30 principal components:

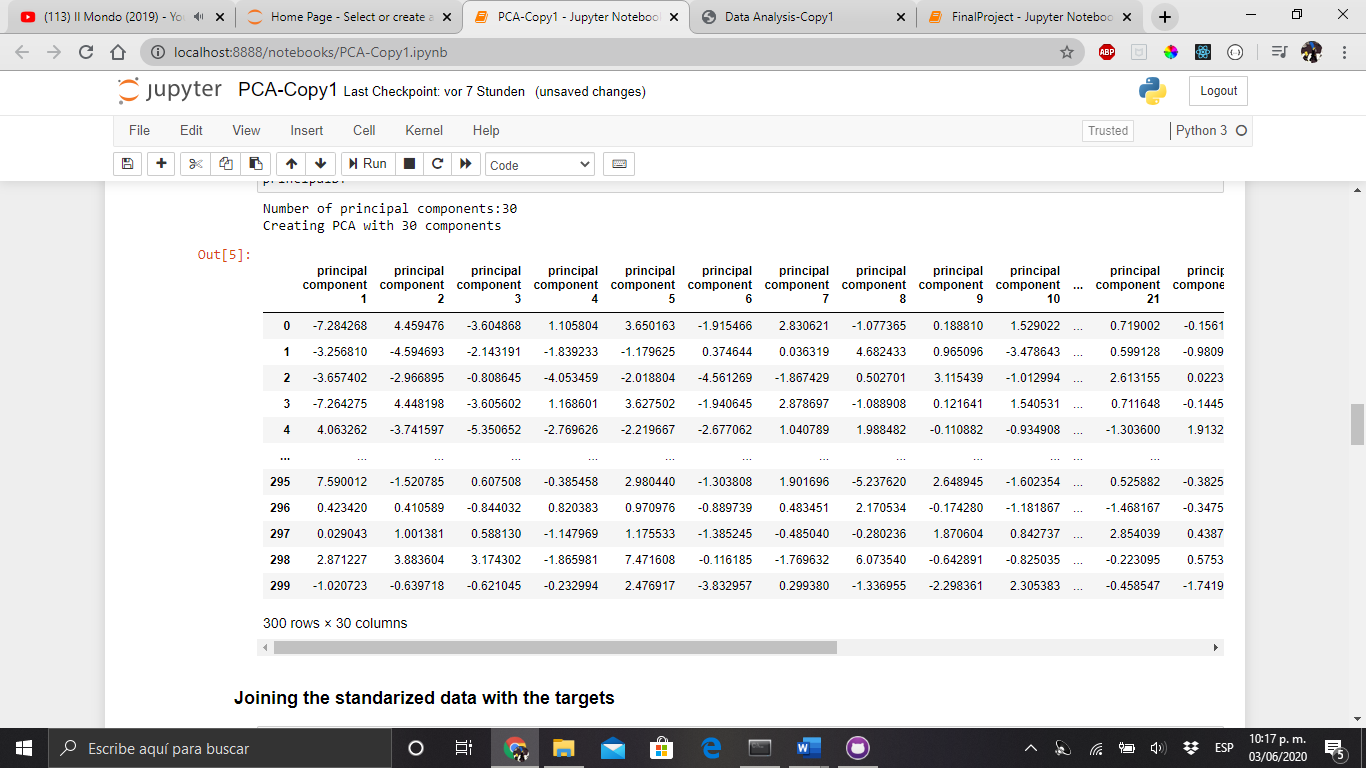


Figure 10 PCA with 30 principal components for the File A

* 1. Plotting the data into a 2-dimensional space

With our PCA done, now we can plot our data into a 2D scatter plot, this give us a visual representation of the data distribution

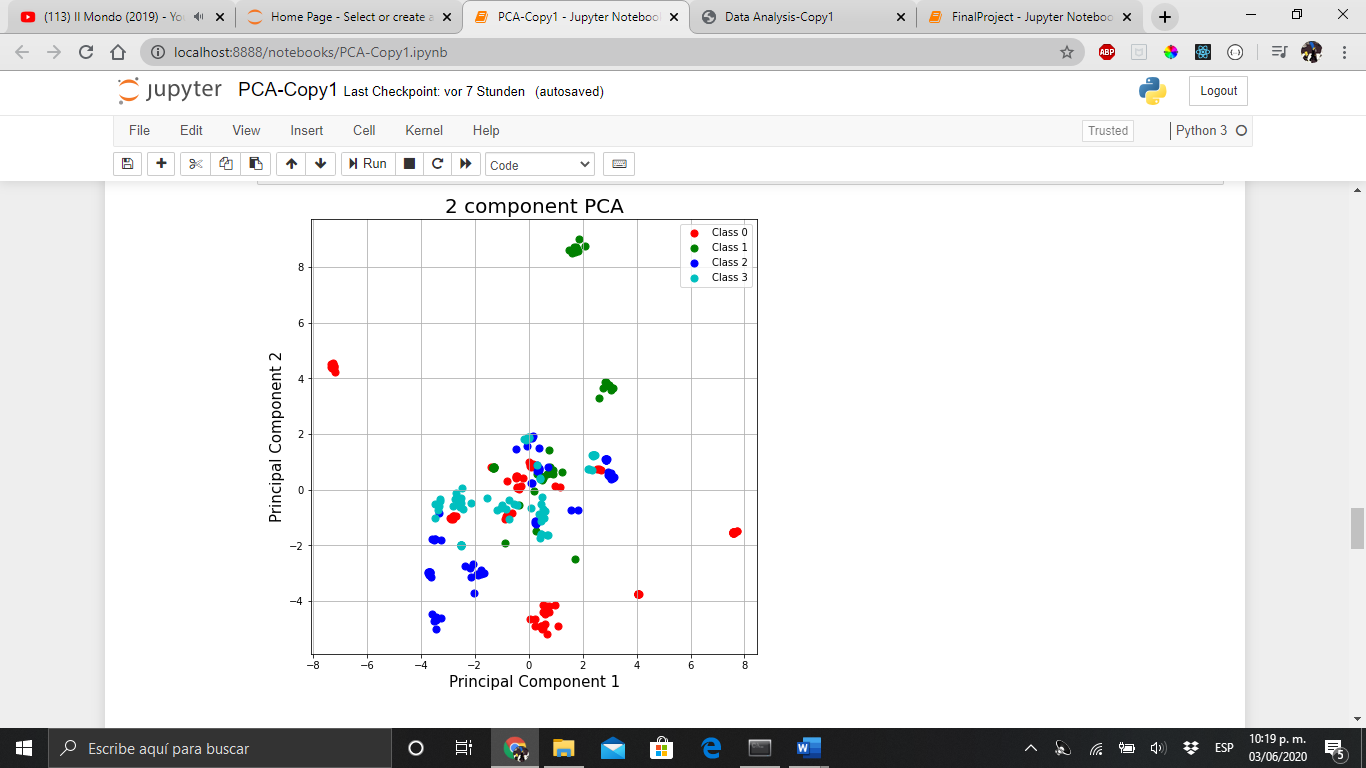


Figure 11 2-Dimensional space PCA for the File A

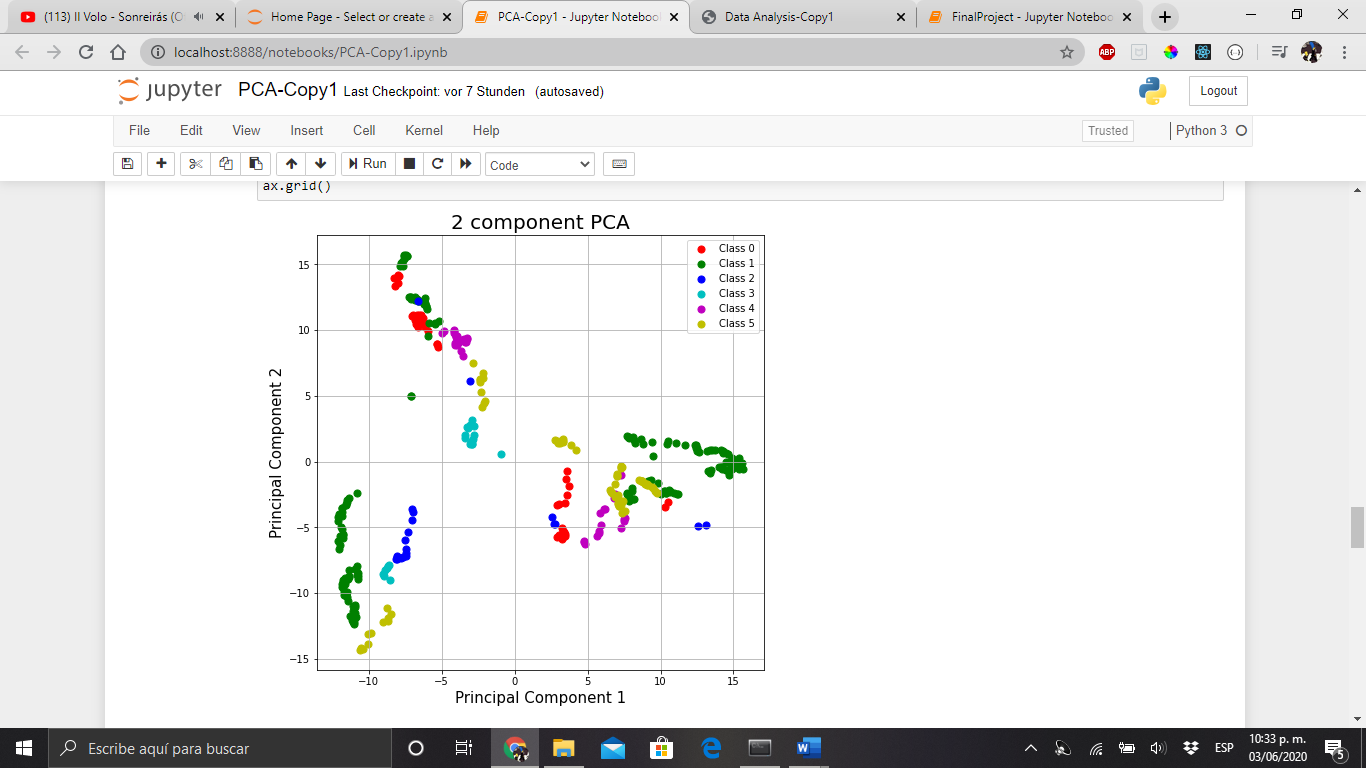


Figure 12 2-Dimensional Space PCA for the File B

* 1. Variance explanation

As we know, the PCA tries to build the components from the ones that represent the most variance in the data. It means, that the first principal component has the most variance in the data, the second one has a variance less than the first one but greater than the third and so on…

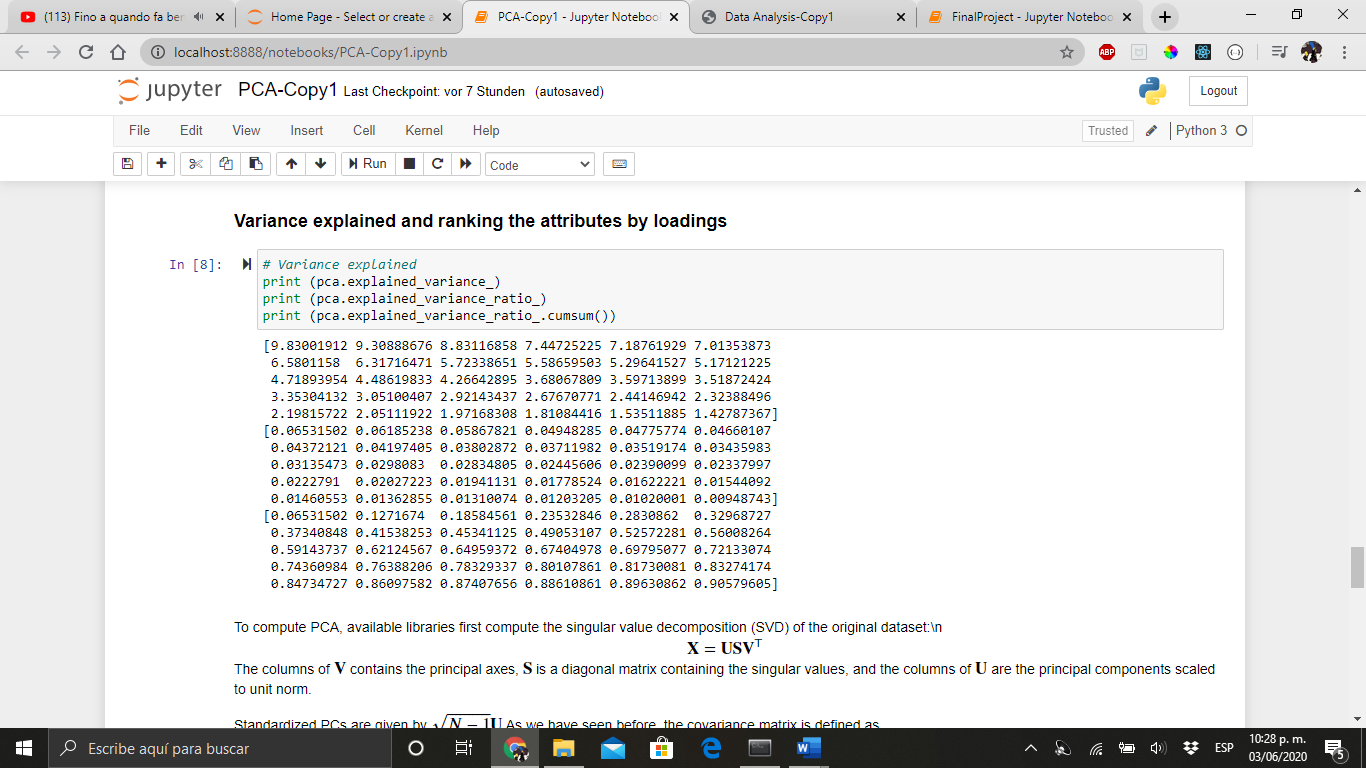


Figure 13 Variance explanation for the File A

We can notice that we were able to reduce the problems dimensionality from 150 to just 30 features. And with this number, is enough to represent a very good percentage of the date, reaching a 90% of data information.

In the File B, the variance had the following behavior with the same number of components:

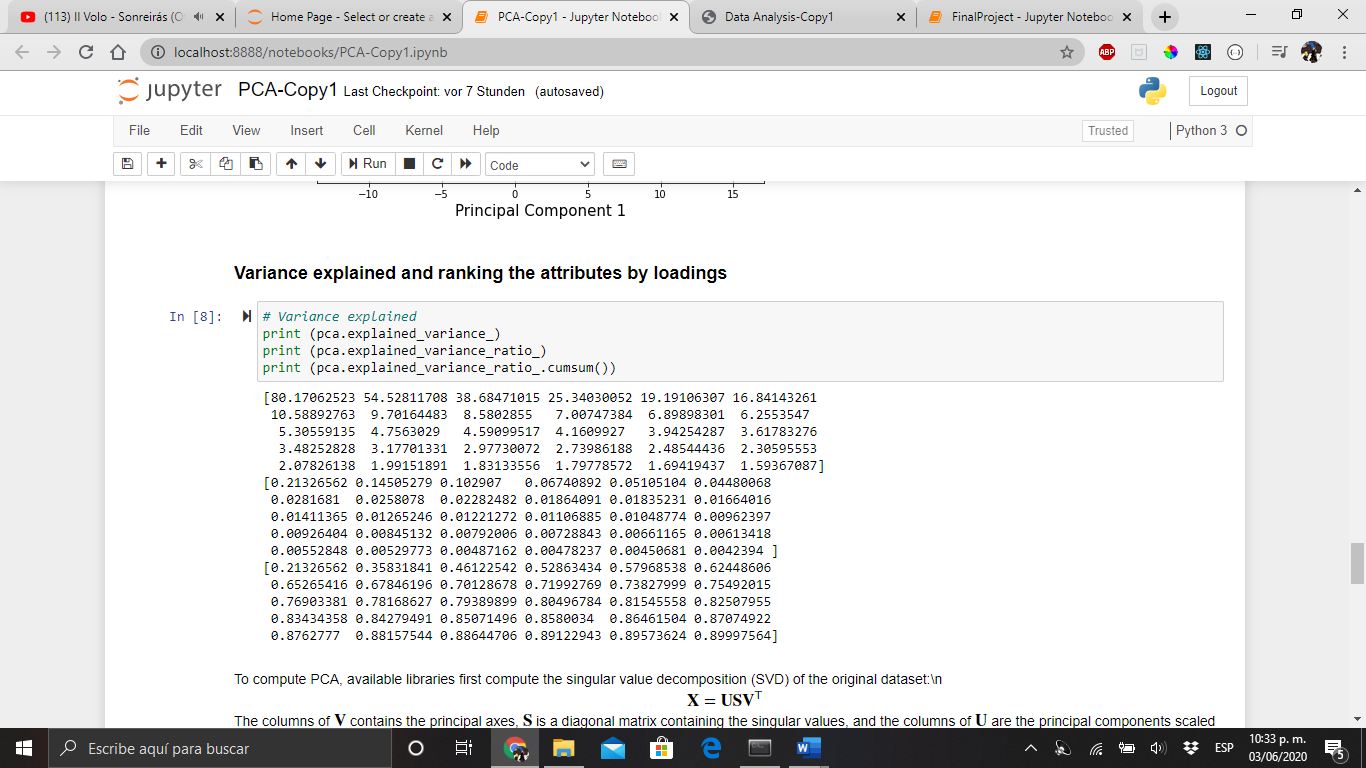


Figure 14 Variance explained for the File B

In both files, 30 components are more than enough to represent a very good percentage of the data information.

1. Classification models and cross validation
   1. Defining the classification models for the ensemble

For the ensemble learning method, we will use 3 supervised classification models:

* K Nearest Neighbors
* Naïve-Bayes
* Neural Network
  1. Testing classification models with different cross validations

We will design the ensemble so the user can give us the following values as an input:

* Number of Folds for the Cross Validation
* Number of K-Nearest Neighbors for the KNN (K-Nearest Neighbors) classifier
* Number of hidden layers in the Neural Network and the number of *perceptrons* in every layer
* Activation function for the perceptron:
  + *Identity*: No-op activation, useful to implement linear bottleneck, returns f(x) = x
  + *Logistic:* The logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
  + *Tanh*: The hyperbolic tan function, returns f(x) = tanh(x).
  + *Relu*: The rectified linear unit function, returns f(x) = max (0, x)
* Number of iterations for the Neural Network (epochs)
* Momentum of the neural network

We will test the efficiency of the classification models with a normal cross validation, after applying PCA to the data distribution and then train the models with that not-correlated features. Both files were tested with the next parameters:

* 5 Folds for the Validation Process
* 5 Nearest Neighbors
* 3 Hidden Layers for the Neural Network, each one with a number of 10, 15 and 10 Perceptrons respectively
* The tanh as activation function
* A number of 3000 epochs
* Momentum of 0.8 for the neural network

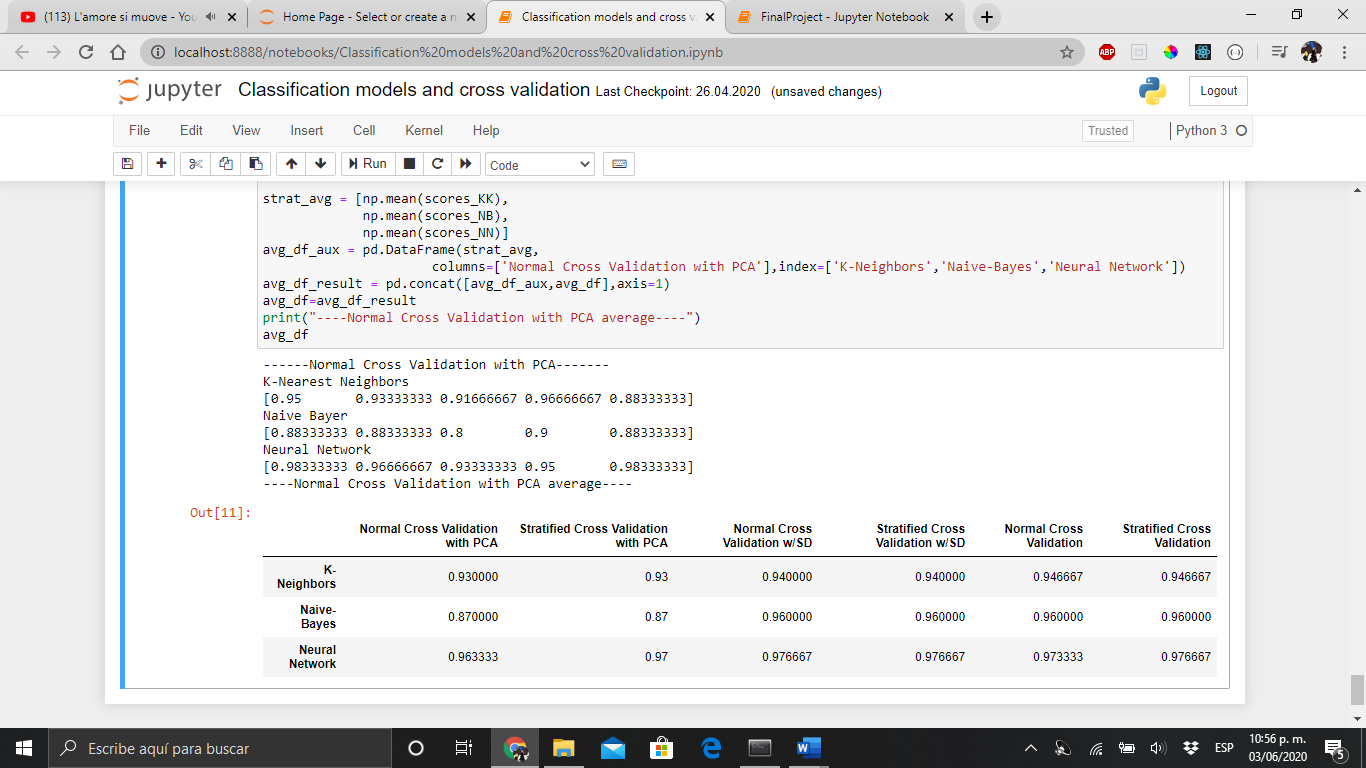


Figure 15 Scores for different cross validations for the file A

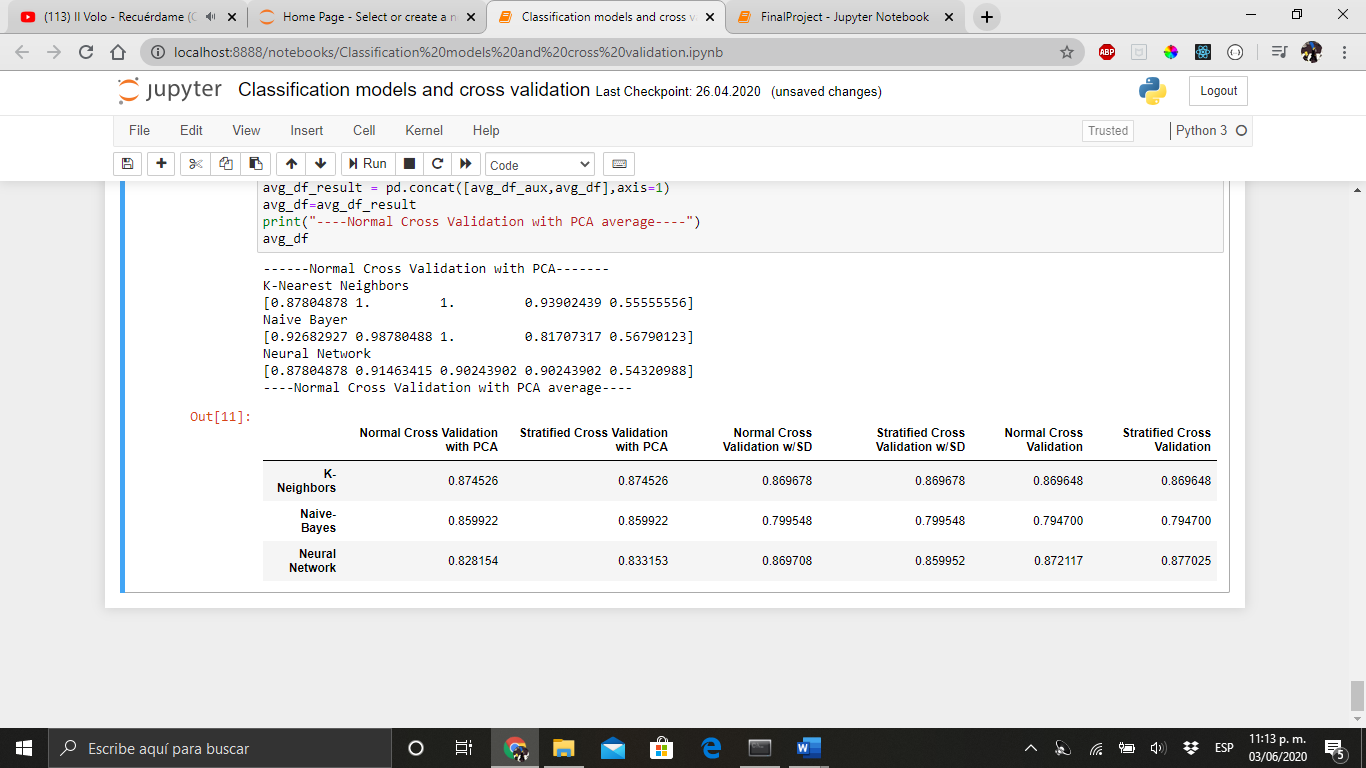


Figure 16 Scores for different cross validations for the file B

1. Building the ensemble learning method
   1. Defining the data distribution for the ensemble

For every classification model, the data will be previously preprocessed and transform with a PCA, in order to improve the performance for the models and avoid wrong answers due to the nature of the model, like the case of the Naïve Bayes and the neural network.

* 1. Ilustrating the neural network

We will try to put a visual representation of the neural network used to test the model:

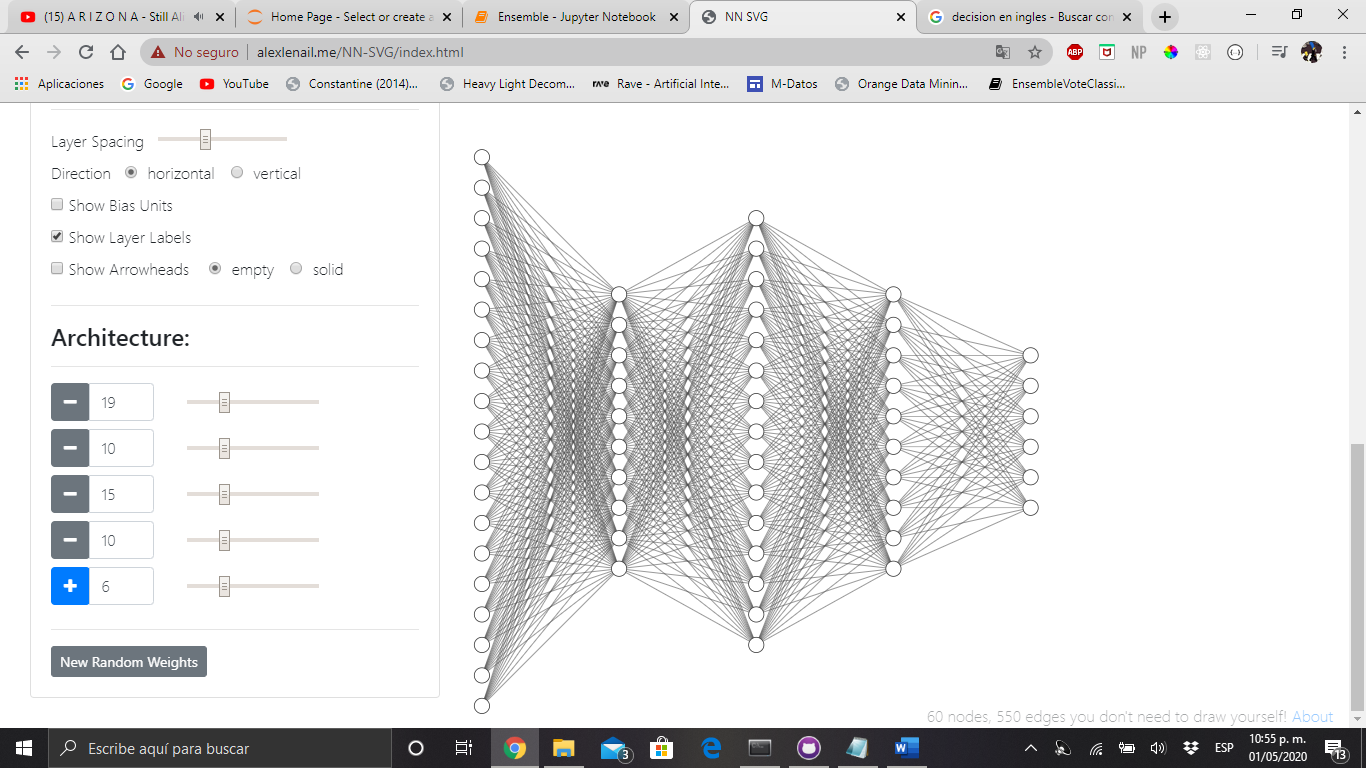


Figure 17 Neural Network

* 1. Building the Ensemble

For the ensemble learning method was used the Hard-Voting algorithm, which consists in selecting the majority of the predicted labels from the models and taking that as the new answer for the ensemble. This is the ensembles accuracy:

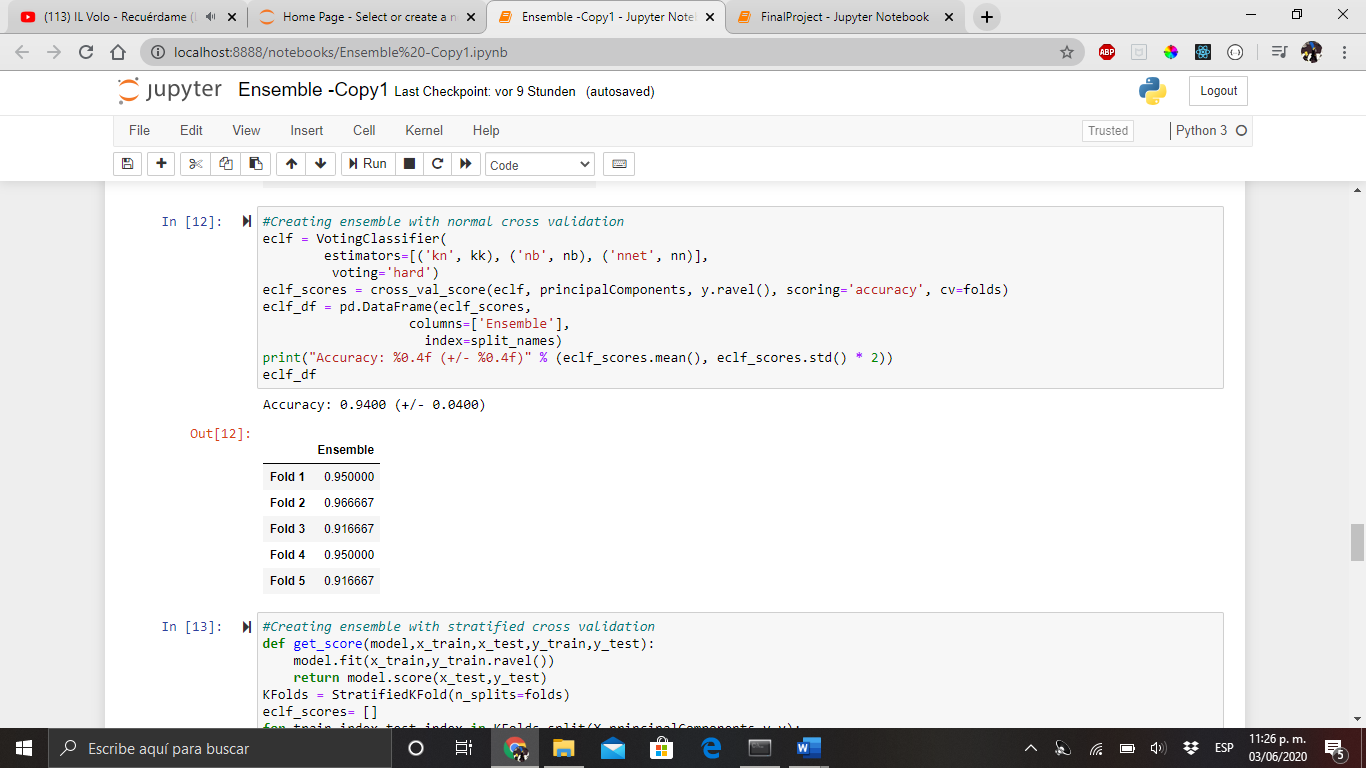


Figure 18 Ensembles Accuracy with the file A

Now we can see that in every fold, the accuracy of the ensemble was not only better than the models’ individual results but also, we got a better accuracy percentage.

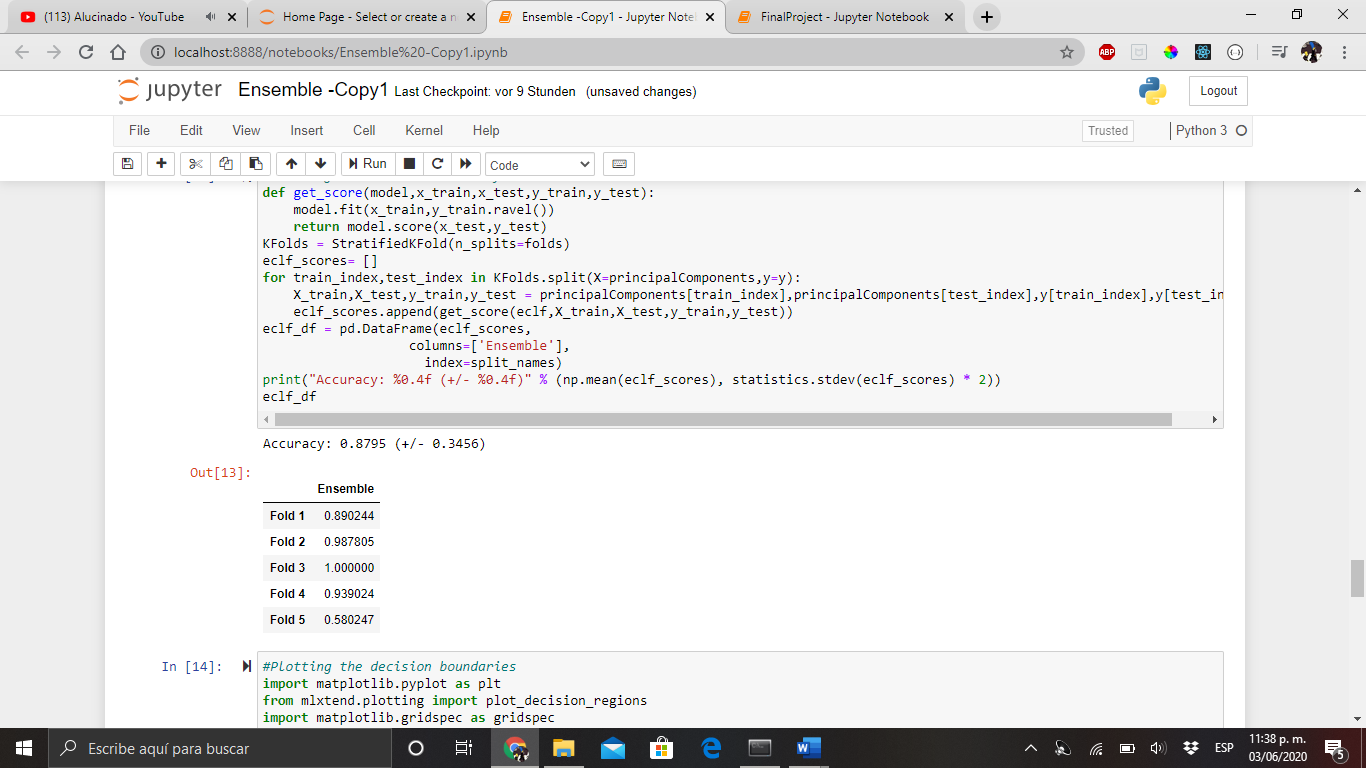


Figure 19 Ensemble's accuracy for the file B

* 1. Plotting decision boundaries

We can plot the decision boundaries of each model in order to have a visual representation of the classification rules that they have created inside the box. This is possible if we plot our data into a two-dimensional space with a little help from the PCA:

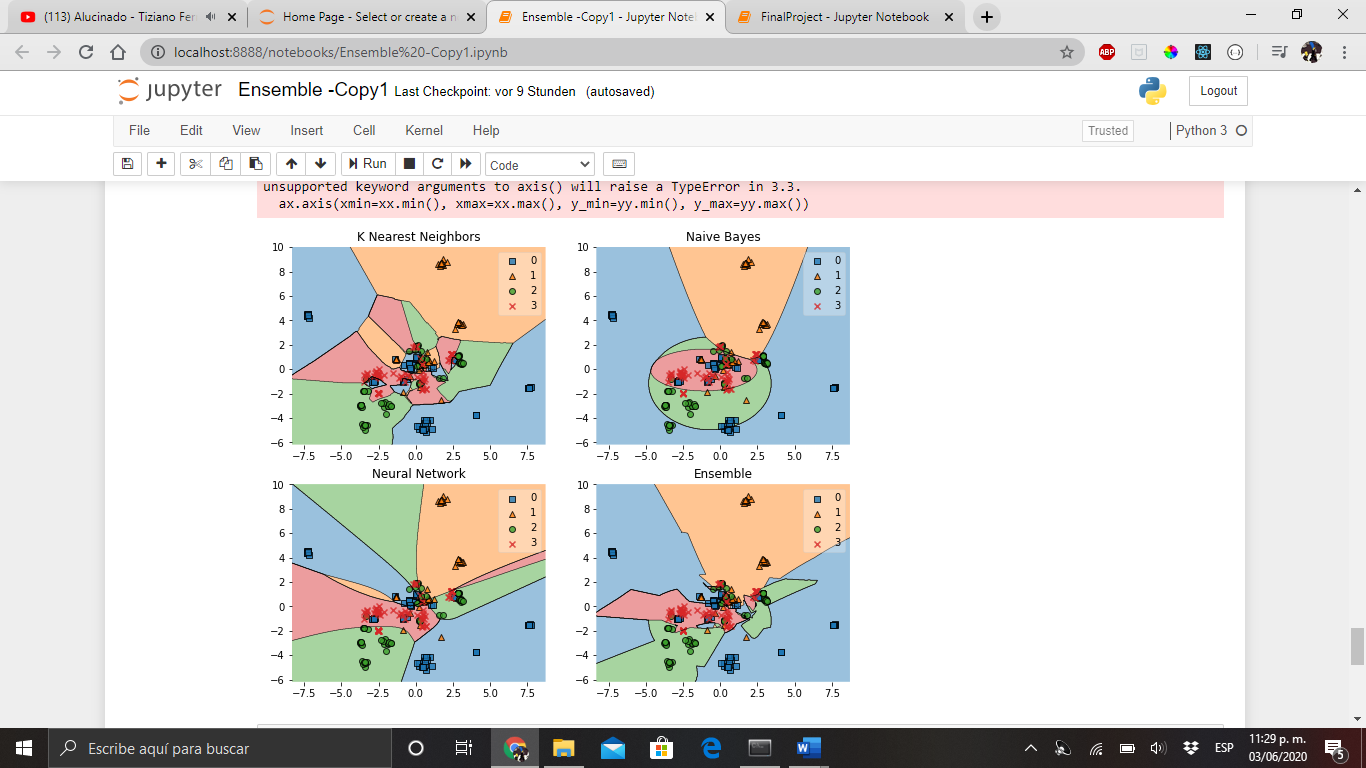


Figure 20 Ensemble's Decision boundaries for the file A

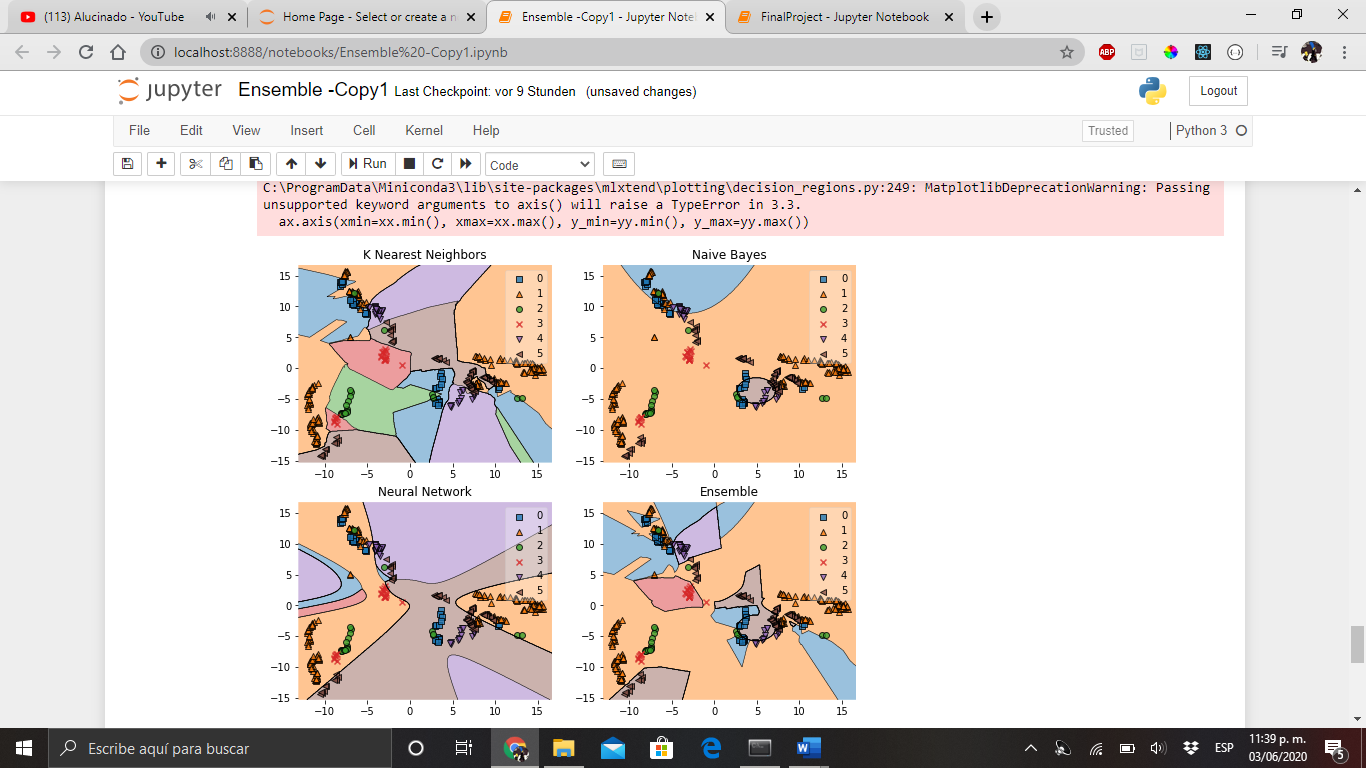


Figure 21 Ensemble's Decision boundaries for the file B

* 1. Logarithmic Loss

And finally, we will measure the logarithmic loss (related to cross-entropy) of the ensemble, which measures the performance of a classification model where the prediction input is a probability value between 0 and 1. The goal of our machine learning models is to minimize this value. A perfect model would have a log loss of 0. Log loss increases as the predicted probability diverges from the actual label. So, predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high log loss. Accuracy is the count of predictions where your predicted value equals the actual value. Accuracy is not always a good indicator because of its yes or no nature. LogLoss considers the uncertainty of your prediction based on how much it varies from the actual label. This gives us a more nuanced view into the performance of our model.

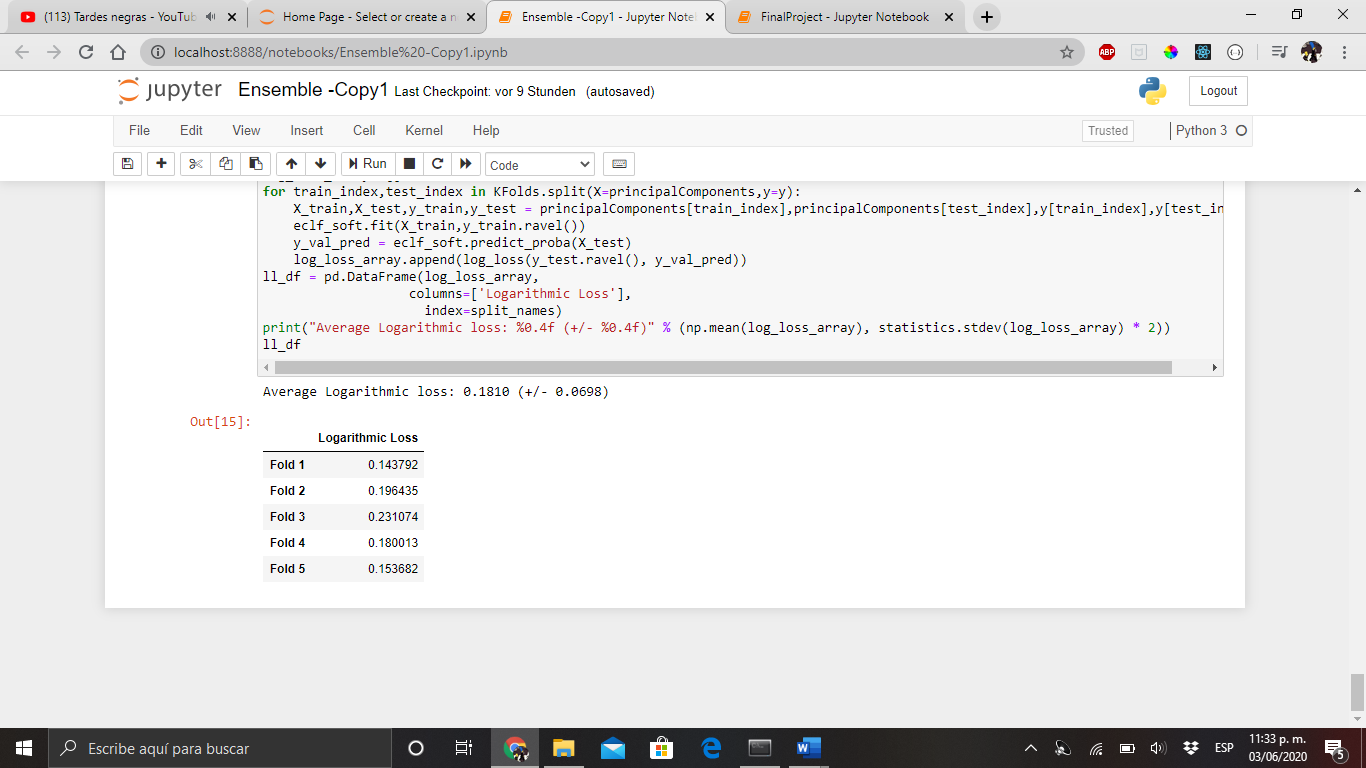


Figure 22 Logarithmic Loss for the file A

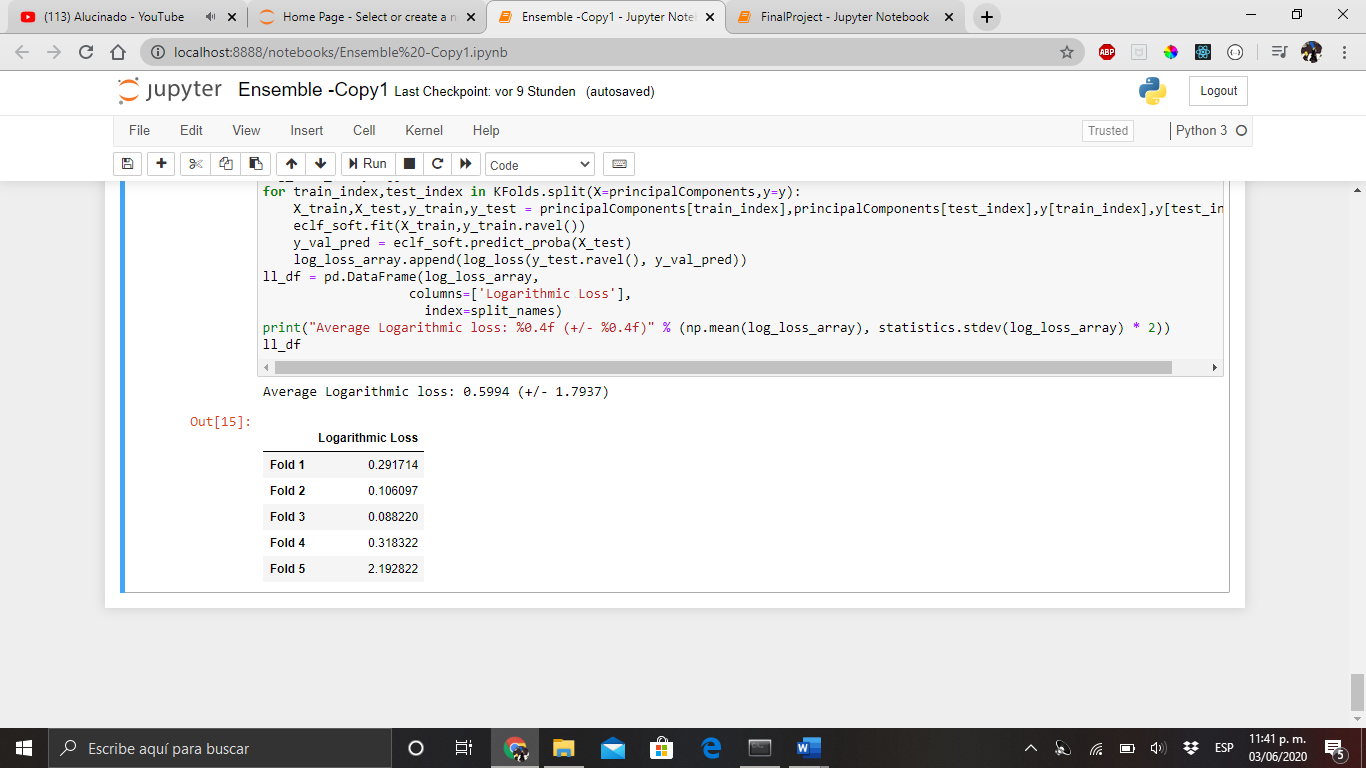


Figure 23 Logarithmic Loss for the file B

1. Plotting the results
   1. Classification report

In order to make a complete classification report from our ensemble learning method, we will use some of the following metrics to describe the accuracy from a more general perspective:

* **Precision**: Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives. Said another way, “for all instances classified positive, what percent was correct?”
* **Recall:** Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. Said another way, “for all instances that were actually positive, what percent was classified correctly?”
* **F1 Score:** The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.
* **Support:** Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn’t change between models but instead diagnoses the evaluation process.

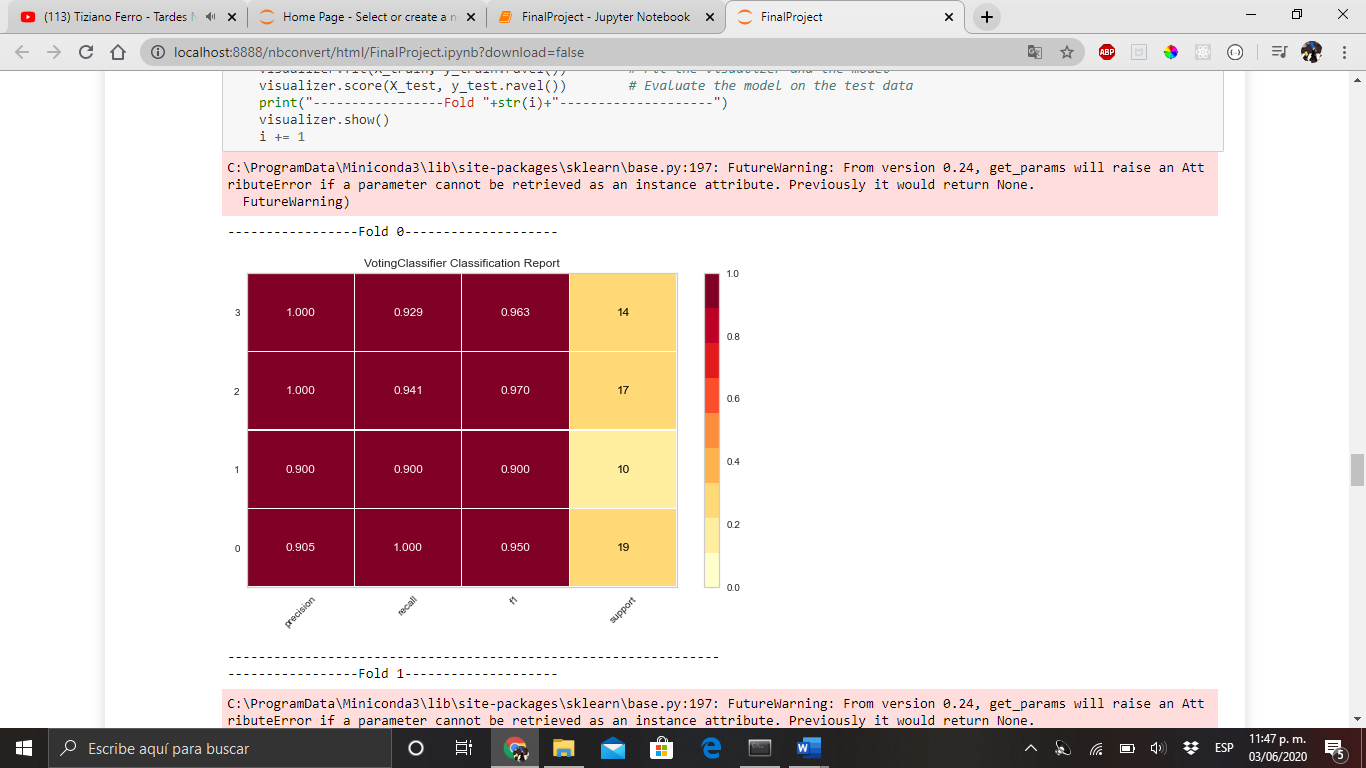


Figure 24 Fold 0 Classification report for the file A

//foto b

* 1. Confusion matrix

As well as with the classification reports, a bar color gives us an idea of how concentrated the predictions from our ensemble for every target class are:

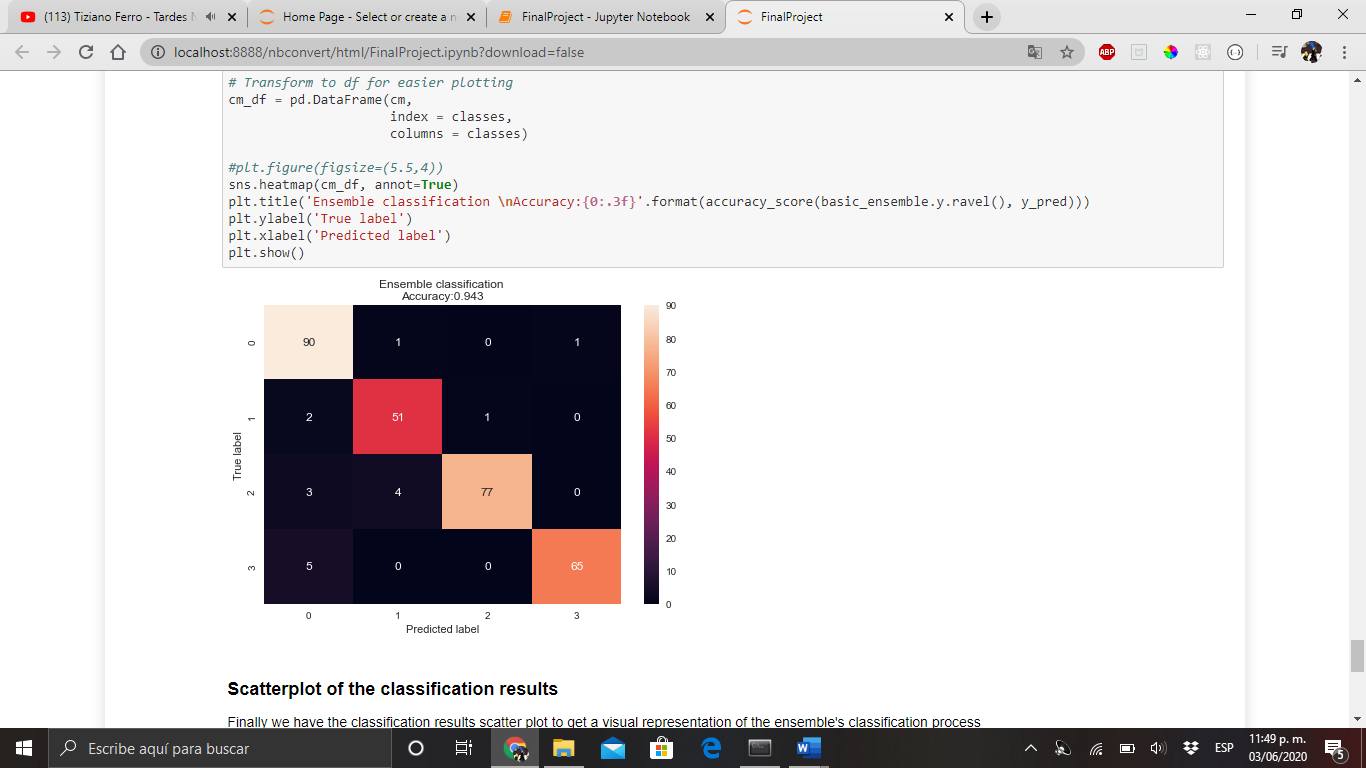


Figure 25 Confusion matrix for the fie A

REFERENCIAS