**F21DL:**

**Data Mining & Machine Learning**

*Coursework 1*

*Students:*

Mohammad ALKHALDI

Anis BENAMER

Quentin DUCASSE

Sylvain TOUANEN

*Lecturers:*

Diana BENTAL

Ekaterina KOMENDANTSKAYA



Table of Contents

[Introduction: 4](#_Toc23860078)

[File Management, Data Pre-Processing, Transformation and Selection 5](#_Toc23860079)

[Naïve Bayes Nets 6](#_Toc23860080)

[Complex Bayes Nets 7](#_Toc23860081)

[Clustering 8](#_Toc23860082)

[Research Question 9](#_Toc23860083)

# 

# Introduction

In the scope of the F21DL-Data Science and Machine Learning course, the first coursework uses a bank of images of street signs and wants to predict the type of a given image. In order to respond to this objective, several tools are used.

The first part of the coursework, and therefore the first one of this report is centered around data *pre-processing*, how we managed files, how we processed, transformed and selected the data available. The second part is focused on *Naïve Bayes* and both its accuracy and issues. Along with this part comes a reflection on the information we learned about the dataset. The third part revolves around *Complex Bayesian Architectures*, how to build them and how they perform compared to the previous model. Along with this part come reflections on the new properties found about the data as well as the help *Bayesian Nets* provide over *Naïve Bayes*. The next part is centered on *K-Means and Clustering* and how it can be applied to our problem. This part comes with a discussion on the results obtained along with the clustering part. Finally, a research question is asked and a solution to it is provided in the final part.

This coursework has been the occasion to train the machine learning skills we obtained throughout the course, either during lectures or labs. As the subject was completely different from the ones in the labs, because it is related to computer vision, we were taking the coursework as a new challenge to prove our understanding of the course. Our initial objective was to use Python in order to complete the coursework as it is looking like the most used language in industry. However, parts of our reflection are based on Weka as well due to its accessibility.

The project is hosted on GitHub under [*https://github.com/QDucasse/dm\_cw1*](https://github.com/QDucasse/dm_cw1)along with the installation instructions and milestones of the project. In order for it to work, the actual datasets need to be downloaded following this link (LINK NEEDED) and then dropped inside the root project. The project was run under Weka 3.8.3 and Python 3.7.4.

# Data Pre-Processing

The first part of the coursework revolves around *Data Pre-Processing*, and we will here explain the objectives we set for ourselves. The four main points we took in consideration are *File Management, Pre-Processing, Transformation* and *Selection.*

***File Management***

In order to load the dataset, it is needed to link the “base” dataset (12660 instances of images composed of 2304 pixels each) and the labels those images are given. Images are stored under “./data/x\_train\_gr\_smpl.csv” while the labels are in “./data/y\_train\_smpl.csv”. Moreover, Boolean masks are provided under “./data/x\_train\_smpl\_<NB>.csv” where NB is the wanted label. Based on those files were generated the files in the arff folder allowing the dataset to be loaded in Weka. The conversion is made possible thanks to the arff\_converter.py file and can be done again by executing the file with the following command while in the root project:

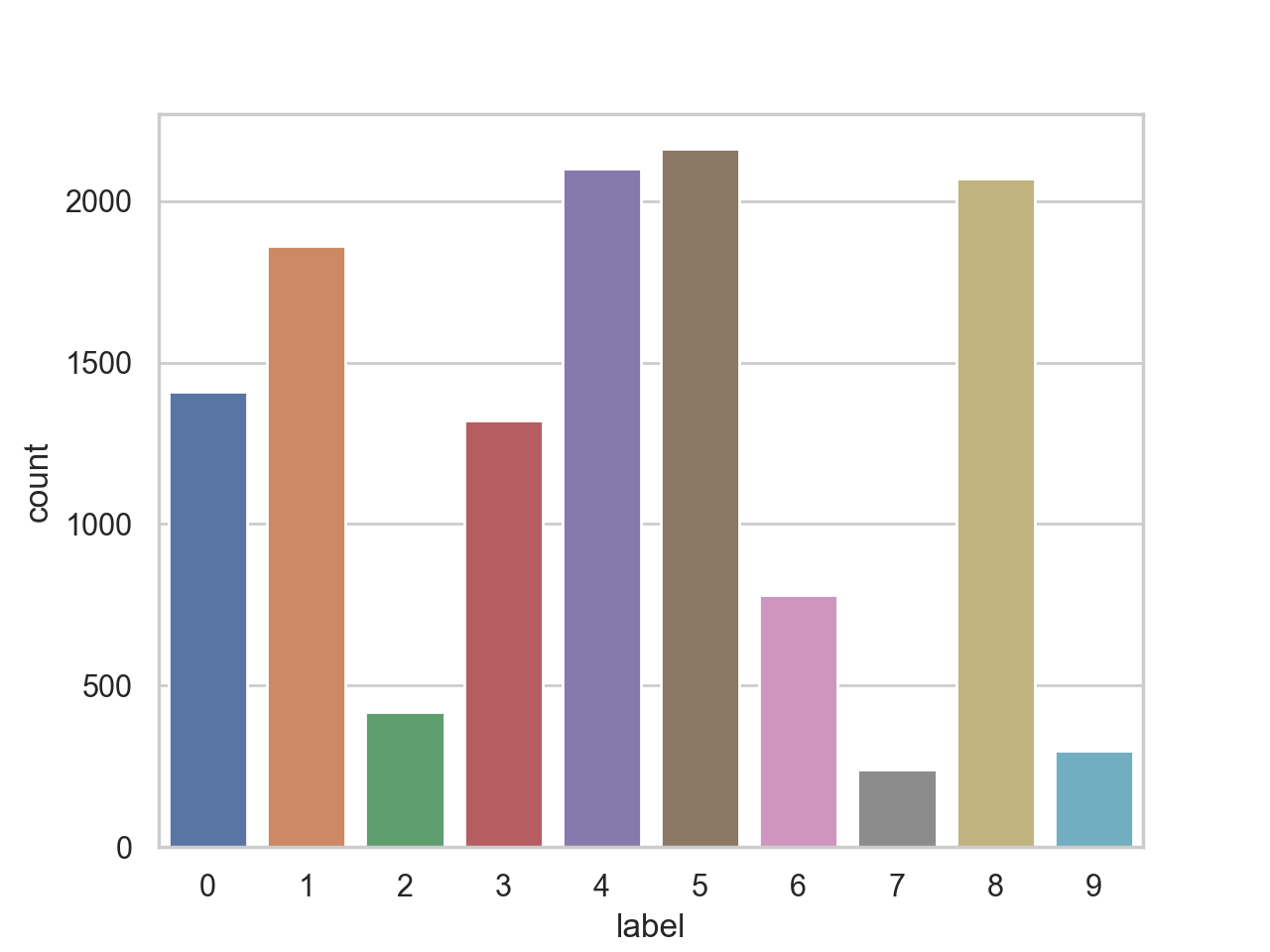
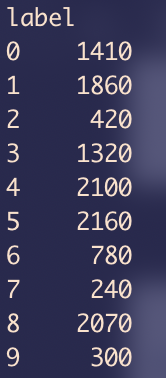
*bash command*

$ python dm\_cw1/arff\_converter.py

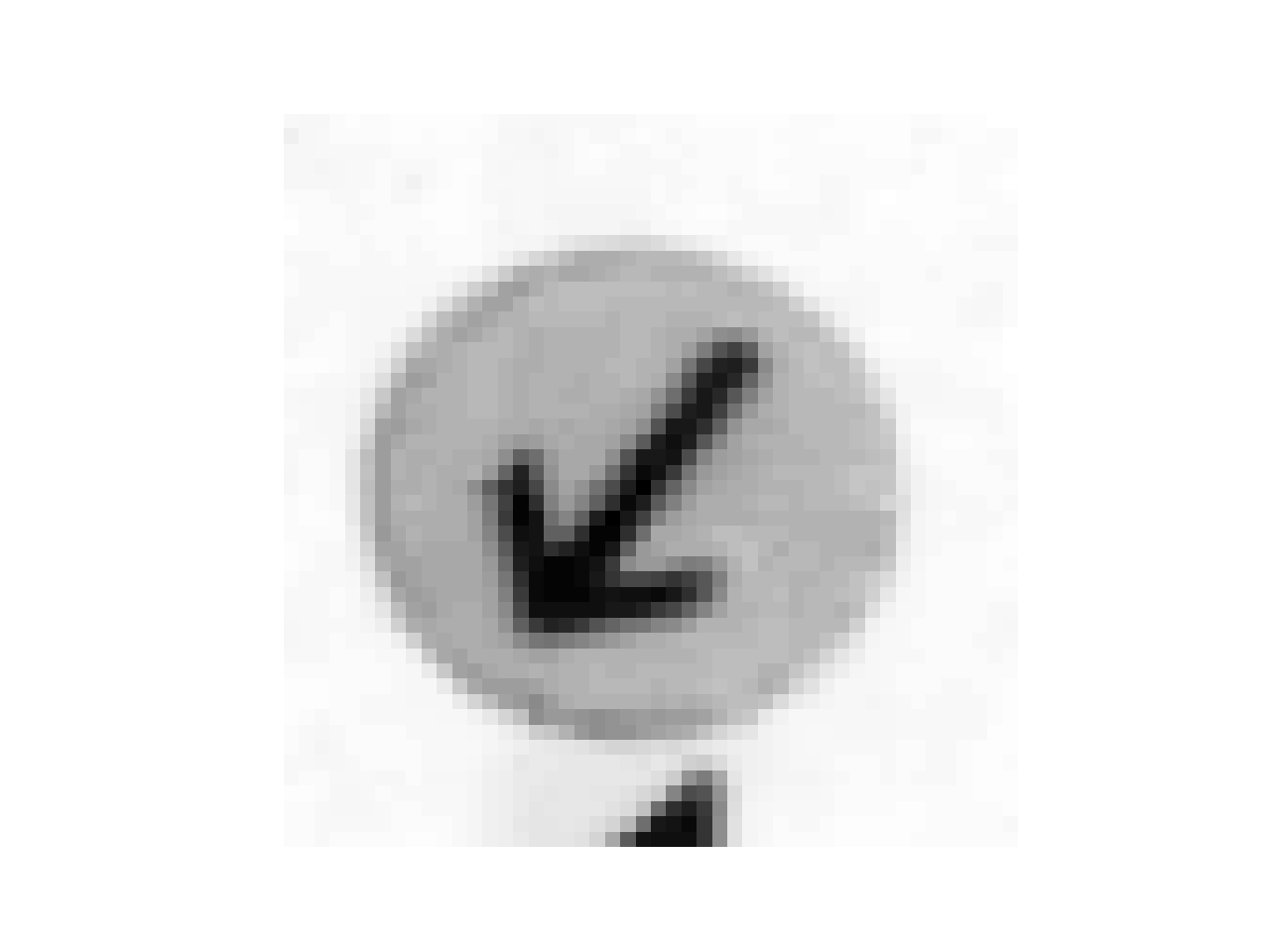
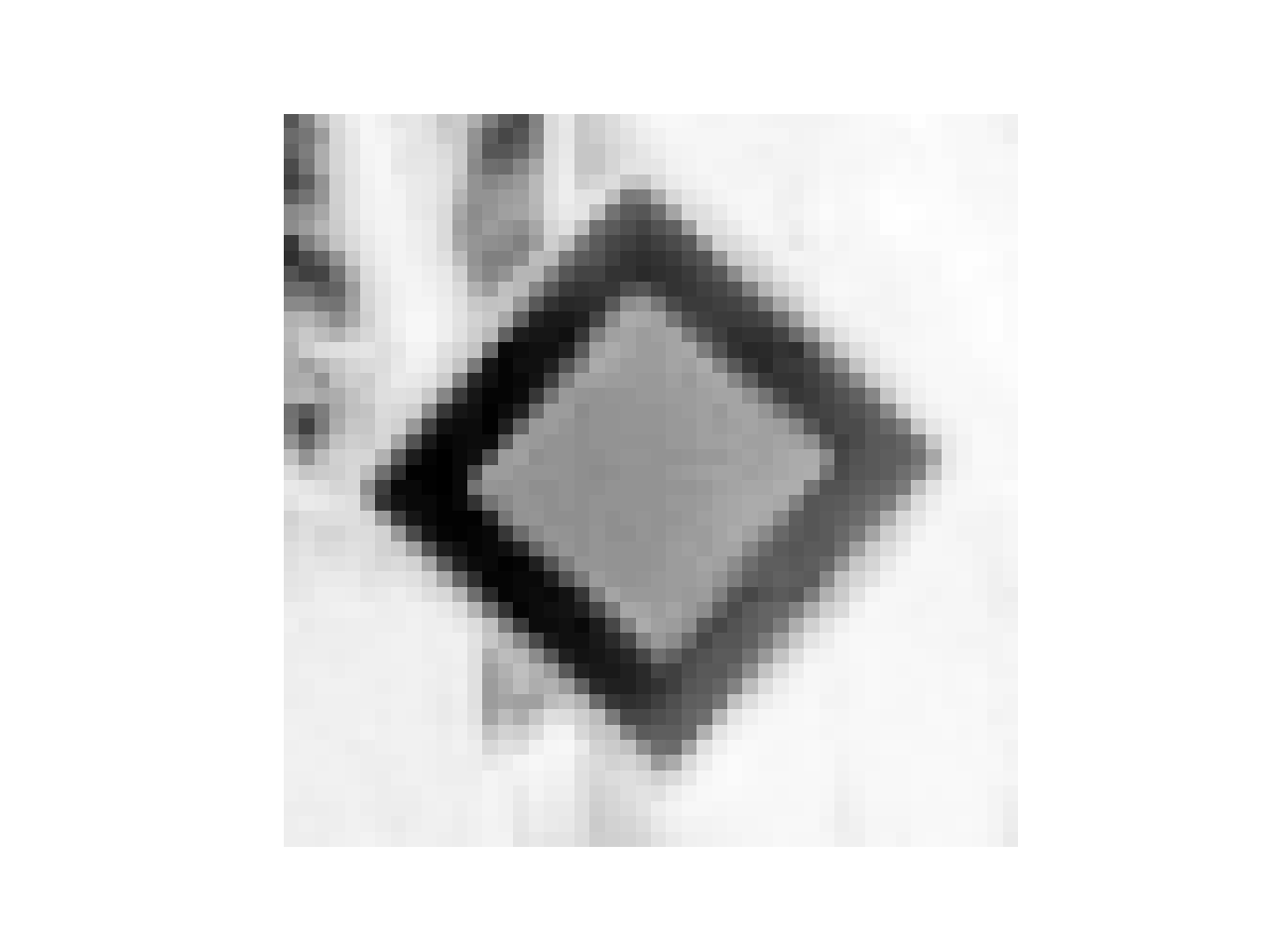
The converter in arff\_converter.py performs similar actions to the loader.py. Both of those scripts fully load the dataset, either the *Weka’s* way through *arff* files or the *Python* way using *pandas*. The labels are linked to the corresponding images and it is possible to load a Boolean mask of the labels rather than the labels themselves.

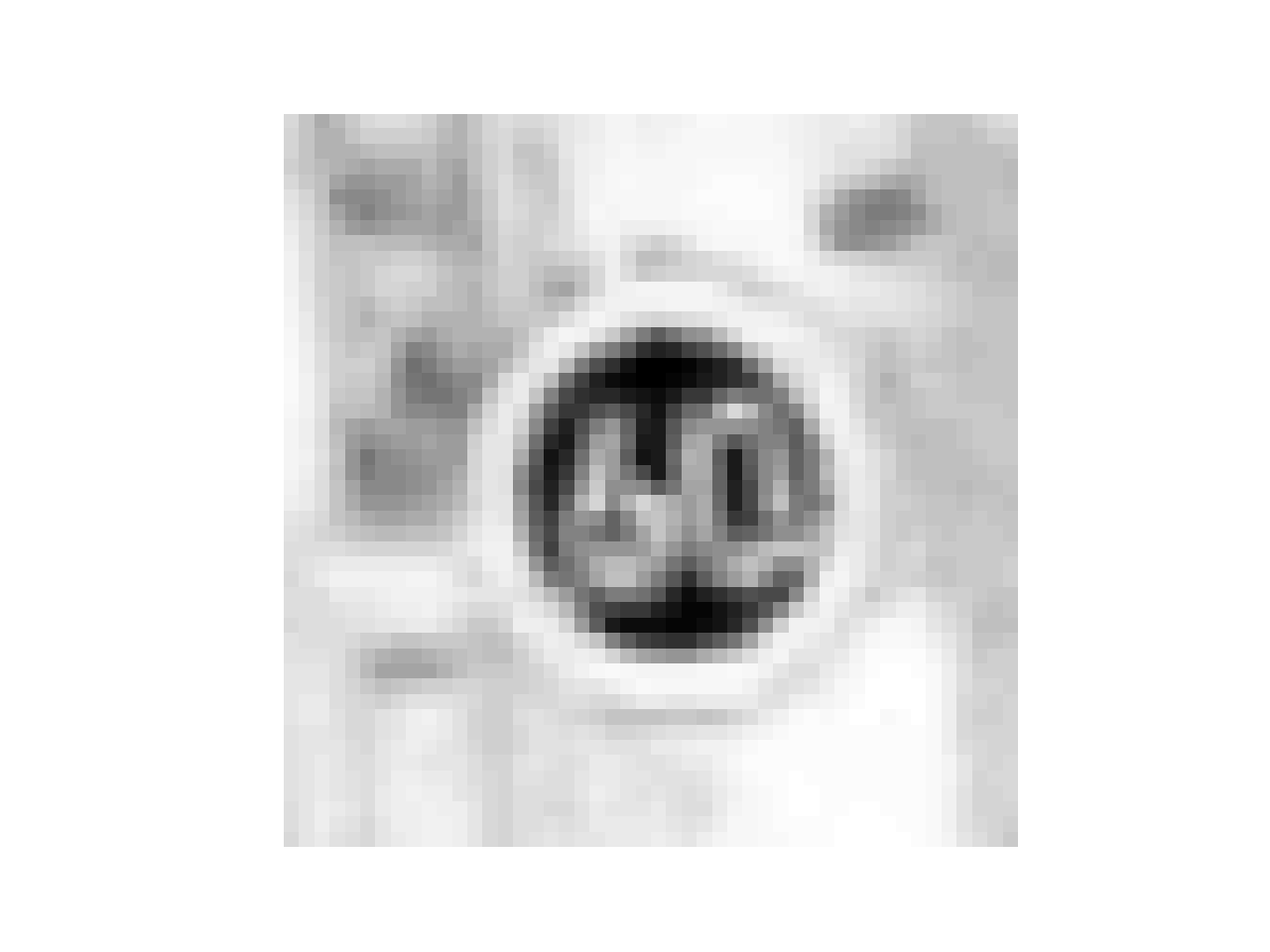
***Visualisation***

Python could handle the whole dataset, so we tried to load it in order to visualize it and have a better understanding of what it is made of. Using *seaborn,* we can plot the number of instances by label as well as their size using groupby(‘label’).size():



The first thing we can notice is that the numbers of instances of the different type of signs is extremely unbalanced, while label 5 is represented by 2160 instances, label 7 only holds 240 instances. This imbalance may cause the filter to perform badly and will need to be fixed. We can then inspect the images by using the display\_nth\_sign() function (indexes 1300, 5600, 10400 and 12550 here).

******

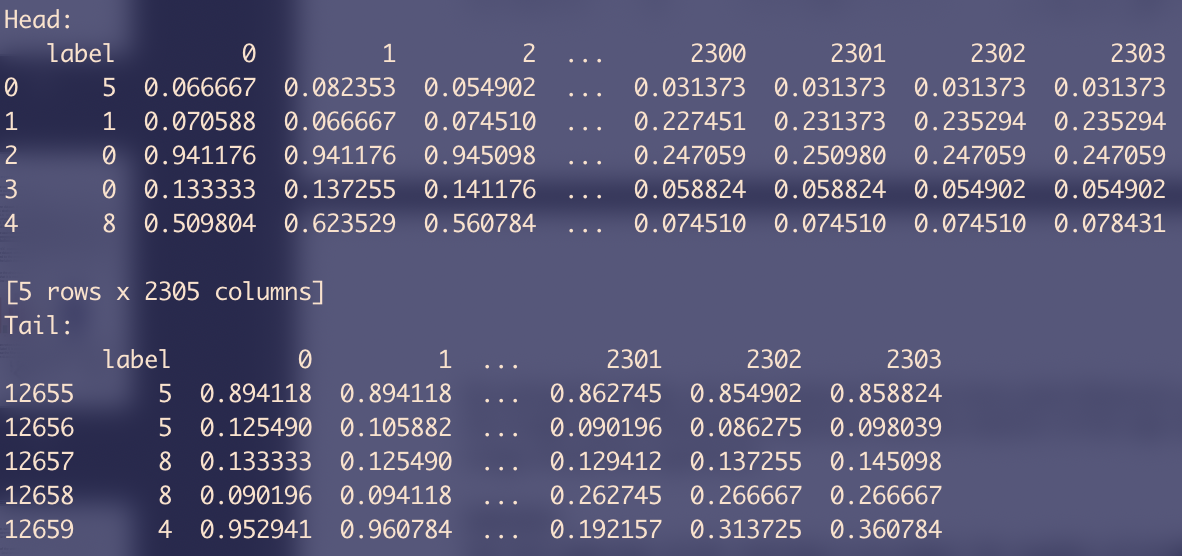
******

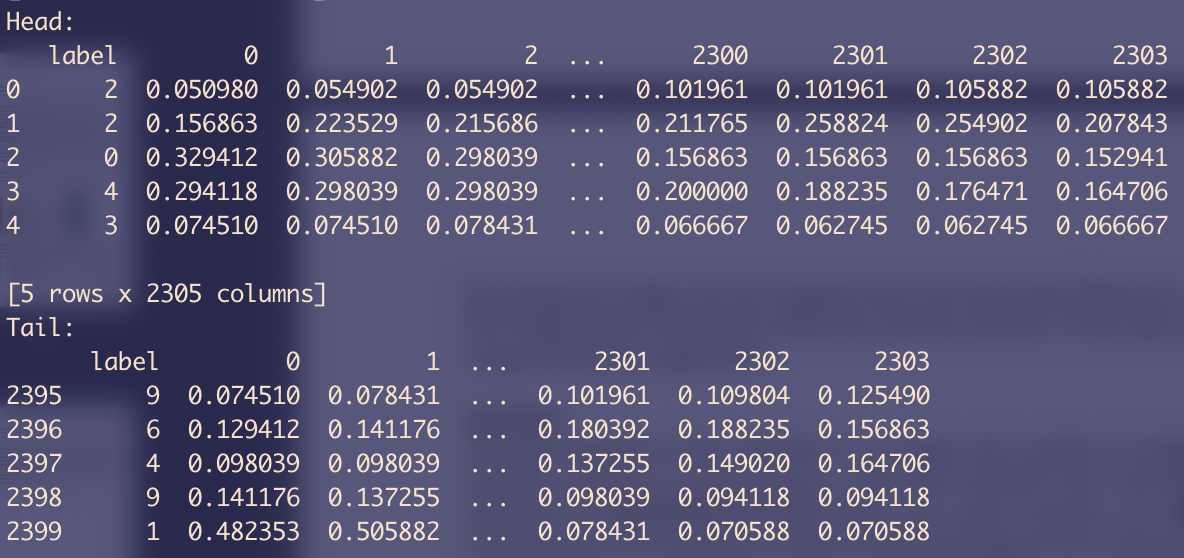
By seeing the image, we can suppose that not every pixel (feature) will be equivalent. The center of the image seems extremely important and the exterior of the sign and therefore the border of the image should less significant.

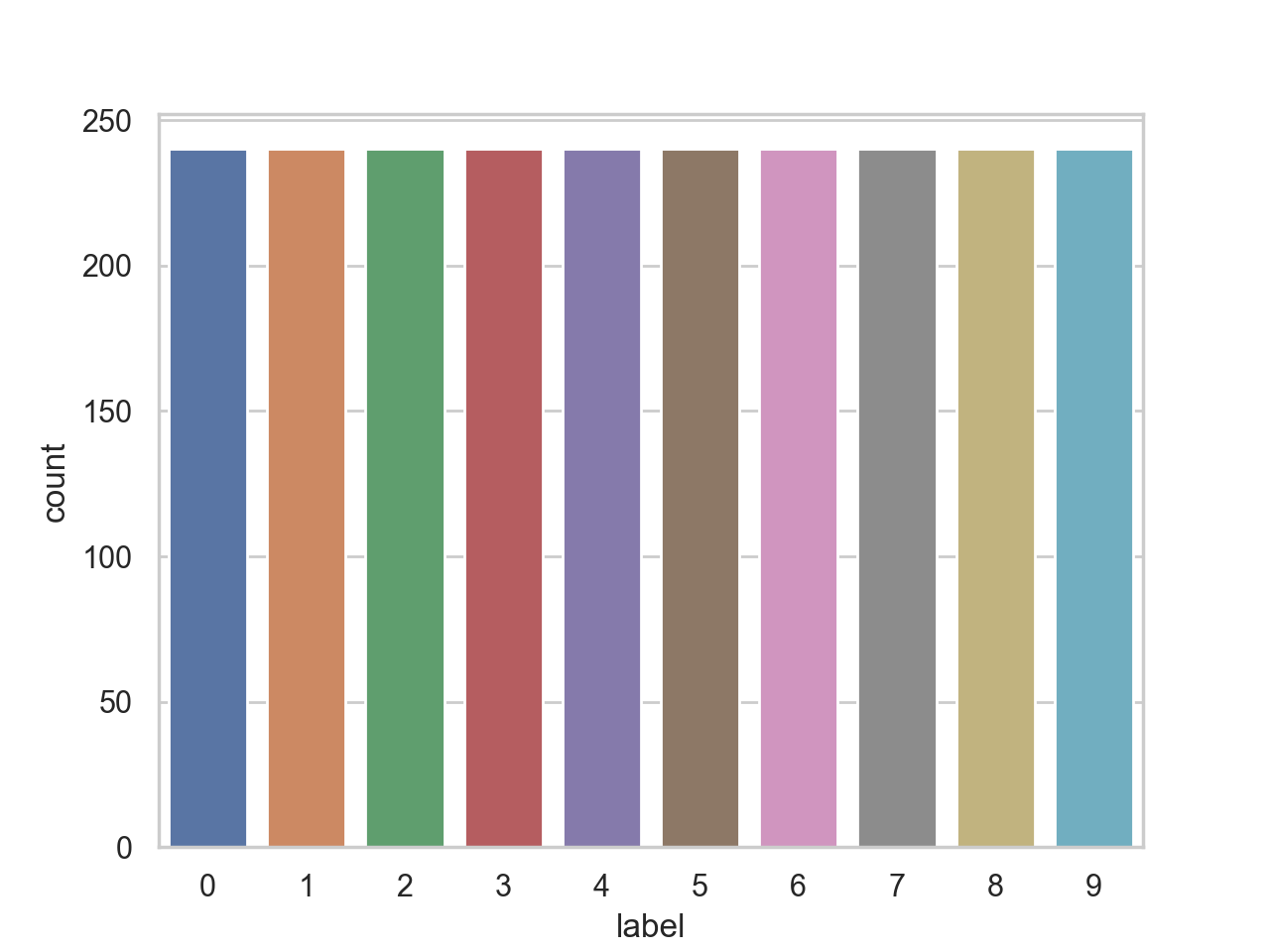
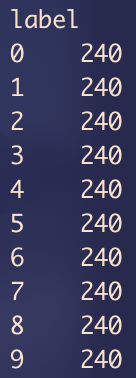
***Selection***

The first objective of the selection is to reduce the number of instances. To do so, we created the function select\_instances() that will compute the minimal number of instances a class attribute is represented by and select this same number out of all the other class attributes representants. The selection is either made by taking the first instances representative of the different class attributes or by picking them randomly. This option can be triggered by specifying the function parameter rd=True or False.

Moreover, the instances are put in random order each time a new computation is done. This is made possible by using the function sample(frac=1) applied to the correct *pandas* data frame. The final datasets are the following (obtained by using print\_head\_tail()).







***Transformation***

The main transformation that we did on the images was to normalize everything. To do so, we used to different techniques. The first one, as the images are composed of greyscale values (between 0 and 255) was to divide everything by 255 to get a value between 0 and 1. The second method uses the function scale() from the *preprocessing* *sklearn* subpackage and will produce a Gaussian distribution with zero mean and variance. You can see in the logs above the result of using the divise\_by\_255() filter on the dataset.

***Arff Conversion***

The conversion of the dataset to .arff files in order to load the dataset in Weka is made through the arff\_converter.py file. As Weka cannot handle large files, the selection is done at generation here following the same guidelines the *pandas* data frame would handle. Executing the file will generate the base dataset, two sampled datasets (random selection or not) and the datasets with the Boolean masks.

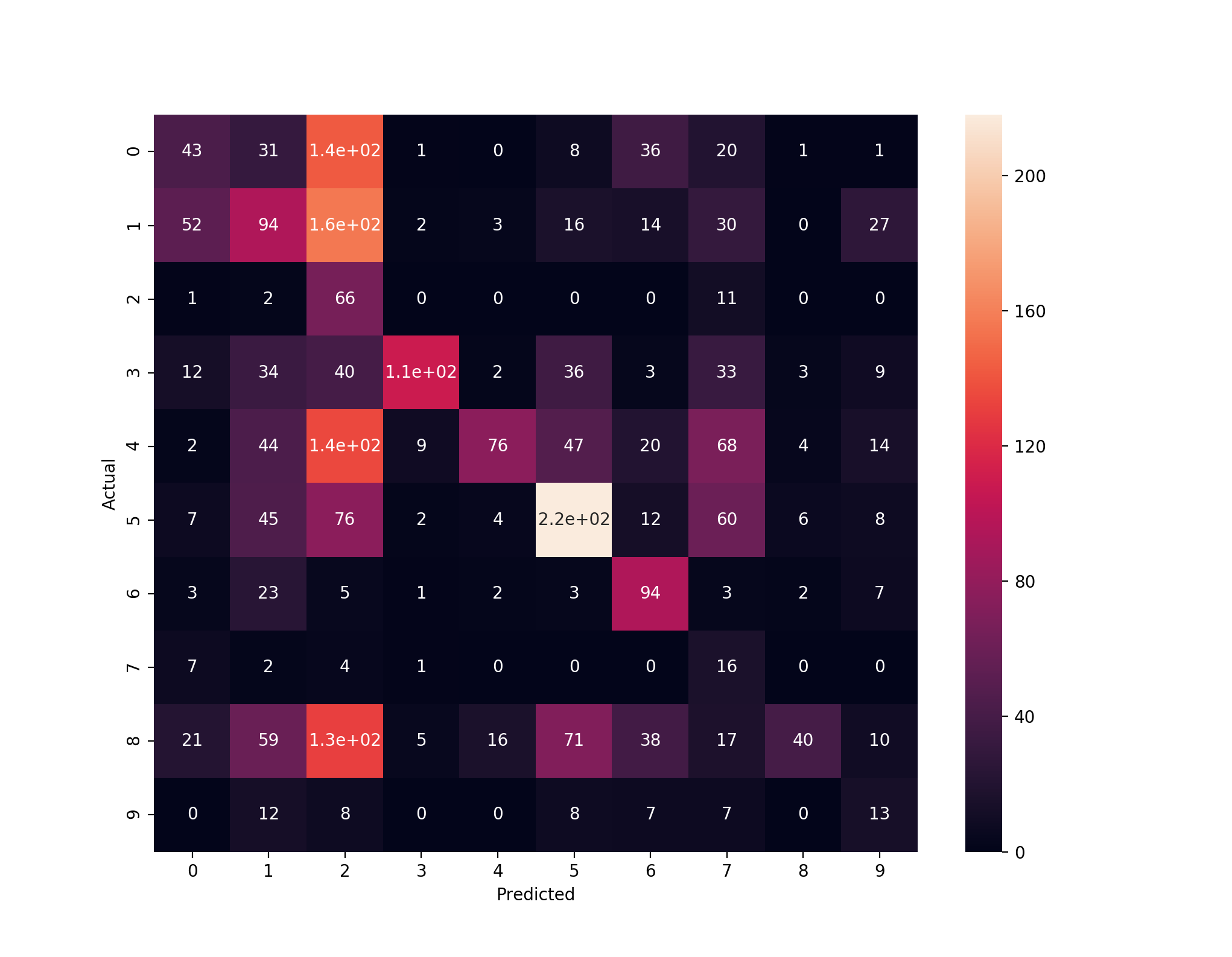
# Naïve Bayes Nets

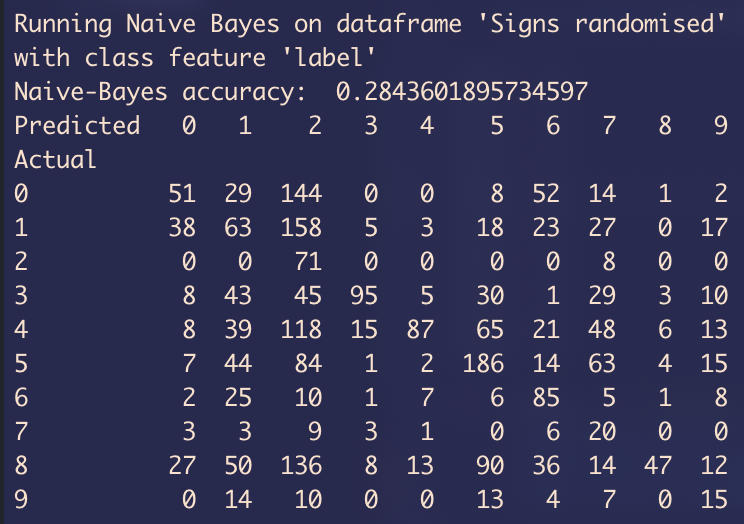
***First use***

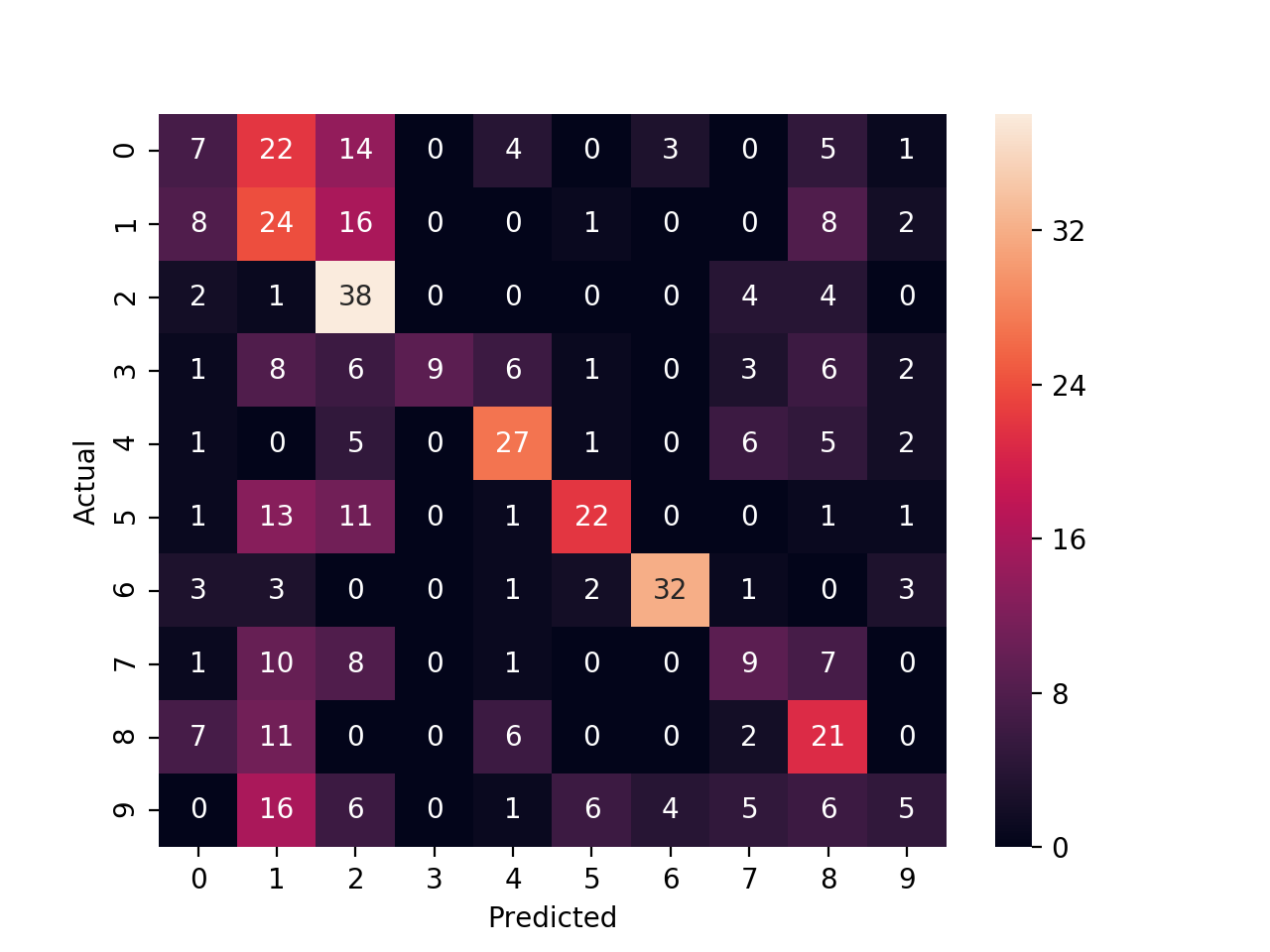
Once the dataset is reduced, randomised and normalised, we can apply the first model and algorithm in order to try to identify the different signs. We will run the algorithm on both Weka and Python in order to compare the results and see if the whole dataset that Python can compute holds better results. For all the runs, a confusion matrix will be provided. We will run the *Naïve Bayes classifier* on the following datasets and under the languages:

* Full dataset (randomised version) in Python (1)
* Small dataset (randomised version) in Python (2)
* Small dataset (randomised version) in Weka (3)

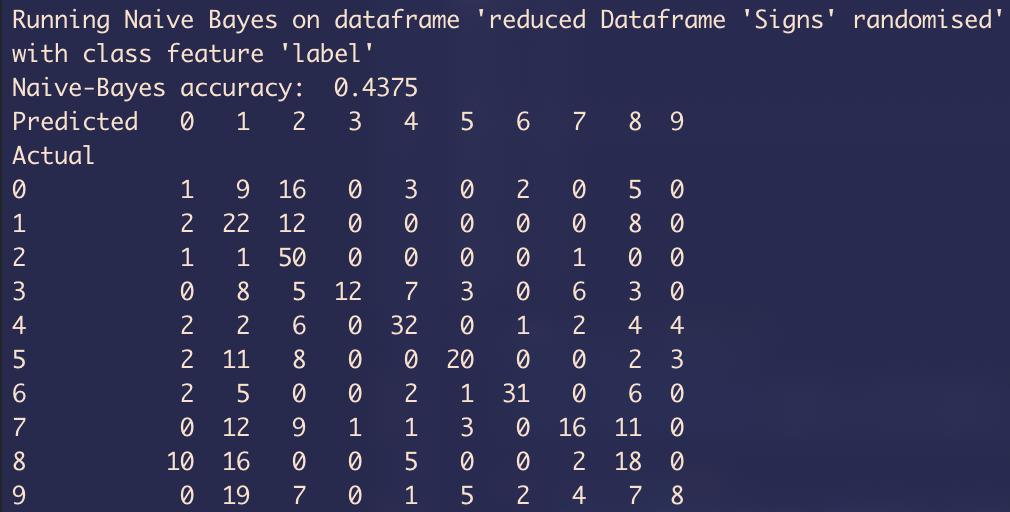
We obtain the following:



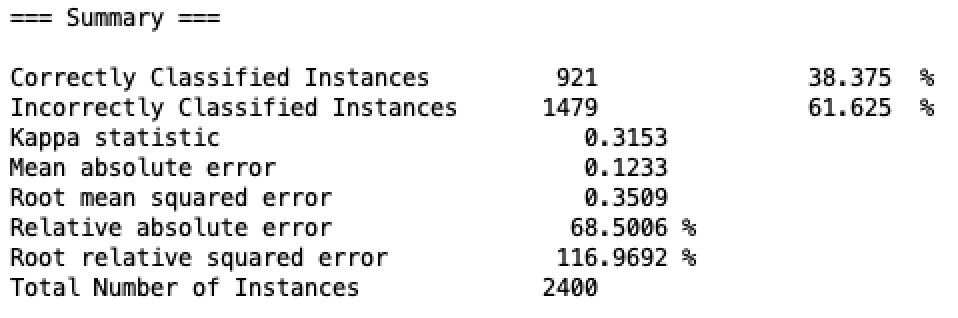
(1)

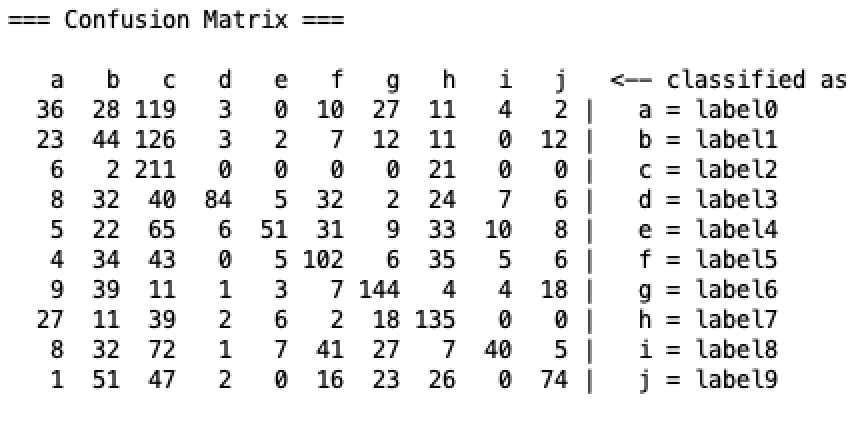


(2)



From Python point of view, the first thing to notice is that the classifier performs better on the reduced dataset (28.4% against 43.8%) even though it struggles to output correct answers as its accuracy stays under 50%. Another thing when looking at the confusion matrix is that the classifier tends to predict signs labelled 1 a lot and often miss the point, especially when looking at signs labelled 0 or 9 for example. The same issues are noticeable on the Weka logs shown below.

(3)



***Determining Best Attributes***

# Complex Bayes Nets

# Clustering

# Research Question

Bayesian optimisation

<https://machinelearningmastery.com/what-is-bayesian-optimization/>