BASIC ENSEMBLE LEARNING METHOD

Data Mining

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)

Omar Alejandro Garduza Riveroll

Benemérita Universidad Autónoma de Puebla

PhD. José Arturo Olvera López

03/06/20

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 (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. All the complete code is found here: https://github.com/OmarGard/DataMining **(Garduza, 2020). It's just necessary to run the 06 – FinalProject file in order to run the final version, the other files were made during all the process and are more specific parts of the project**

I. 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

a. First steps

First we need to read the file and retreive the data, for the following notes, there 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**.

b. 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

c. Plotting 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:

$$K * IQR = 1.5 * IQR$$

First we will show a few graphics from the File A:

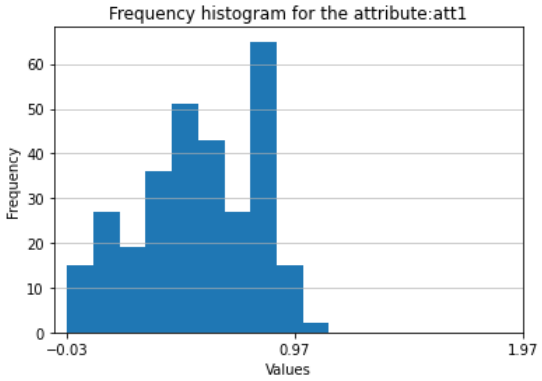

Figure 1 Frequency histogram for the att1 data in the File A

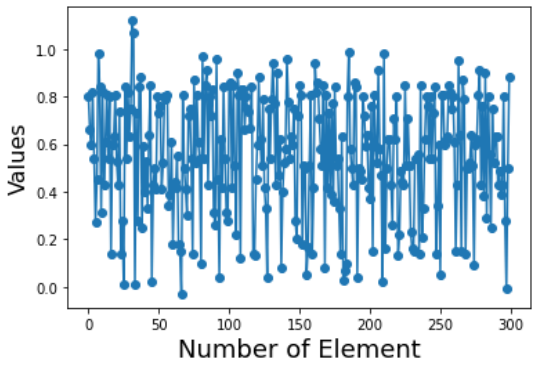

Figure 2 Scatter plot for the att1 data in the File A

```
Summary statistics
Min       :  -0.03
Lower Qu.:  0.405
Median    :  0.56
Mean      :  0.553
Upper Qu.:  0.8
Max       :  1.12
OutLiers
No outliers
```

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 had barely a few outliers or even none, like the example before.

Here is an example of a feature with a few outliers:

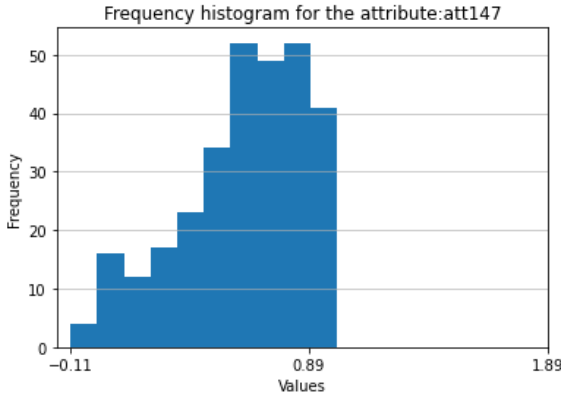

Figure 4 Frequency histogram for the att147 data in the File A

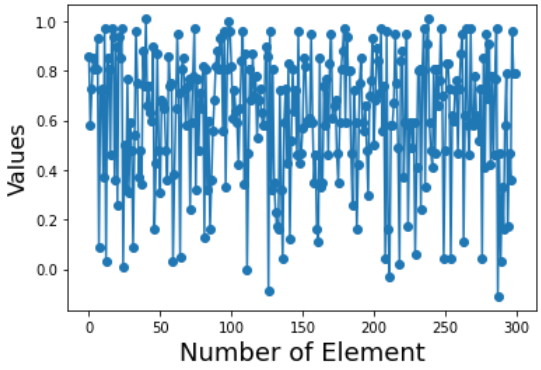

Figure 5 Scatter plot for the att147 data in the File A

```
Summary statistics
Min      :  -0.11
Lower Qu.:  0.46
Median   :  0.65
Mean     :  0.61
Upper Qu.:  0.81
Max      :  1.01
OutLiers
-0.09
-0.11
```

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

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

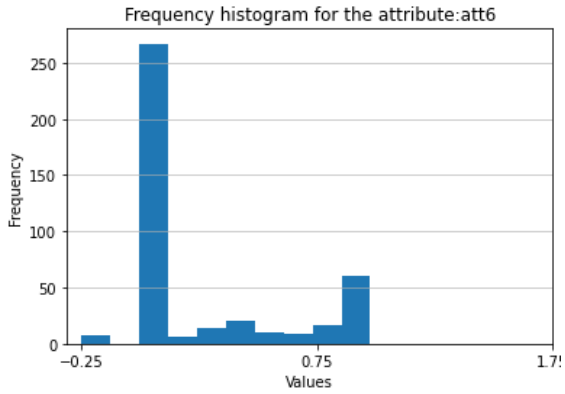

Figure 7 Frequency histogram for the att6 data in the File B

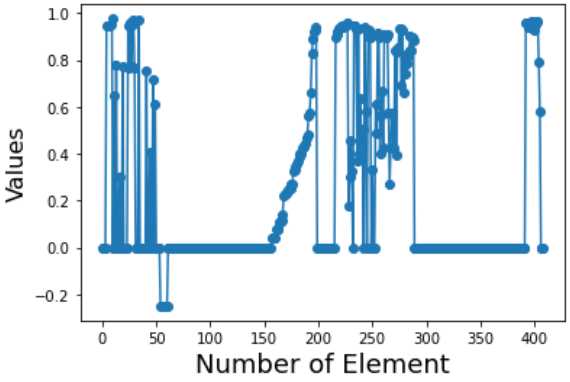

Figure 8 Scatter plot for the att6 data in the File B

```
Summary statistics
Min      :  -0.25
Lower Qu.:  -0.25
Median   :  0.0
Mean     :  0.017
Upper Qu.:  0.0
Max      :  0.978272
OutLiers
0.885293
0.494707
0.897166
0.920455
0.960921
0.96159
0.9638
0.963647
0.926507
0.96159
```

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.

II. Principal Component Analysis (PCA)

a. 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:

:

	principal component 1	principal component 2	principal component 3	principal component 4	principal component 5	principal component 6	principal component 7	principal component 8	principal component 9	principal component 10	...	principal component 21	princip compone
0	-7.284268	4.459476	-3.604868	1.105804	3.650163	-1.915466	2.830621	-1.077365	0.188810	1.529022	...	0.719002	-0.1561
1	-3.256810	-4.594693	-2.143191	-1.839233	-1.179625	0.374644	0.036319	4.682433	0.965096	-3.478643	...	0.599128	-0.9809
2	-3.657402	-2.966895	-0.808645	-4.053459	-2.018804	-4.561269	-1.867429	0.502701	3.115439	-1.012994	...	2.613155	0.0223
3	-7.264275	4.448198	-3.605602	1.168601	3.627502	-1.940645	2.878697	-1.088908	0.121641	1.540531	...	0.711648	-0.1445
4	4.063262	-3.741597	-5.350652	-2.769626	-2.219667	-2.677062	1.040789	1.988482	-0.110882	-0.934908	...	-1.303600	1.9132
...	...	...	...	...	...	...	...	...	...	...	...	...	
295	7.590012	-1.520785	0.607508	-0.385458	2.980440	-1.303808	1.901696	-5.237620	2.648945	-1.602354	...	0.525882	-0.3825
296	0.423420	0.410589	-0.844032	0.820383	0.970976	-0.889739	0.483451	2.170534	-0.174280	-1.181867	...	-1.468167	-0.3475
297	0.029043	1.001381	0.588130	-1.147969	1.175533	-1.385245	-0.485040	-0.280236	1.870604	0.842737	...	2.854039	0.4387
298	2.871227	3.883604	3.174302	-1.865981	7.471608	-0.116185	-1.769632	6.073540	-0.642891	-0.825035	...	-0.223095	0.5753
299	-1.020723	-0.639718	-0.621045	-0.232994	2.476917	-3.832957	0.299380	-1.336955	-2.298361	2.305383	...	-0.458547	-1.7419

300 rows × 30 columns

Figure 10 PCA with 30 principal components for the File A

b. Plotting the data into a 2-dimensional space

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

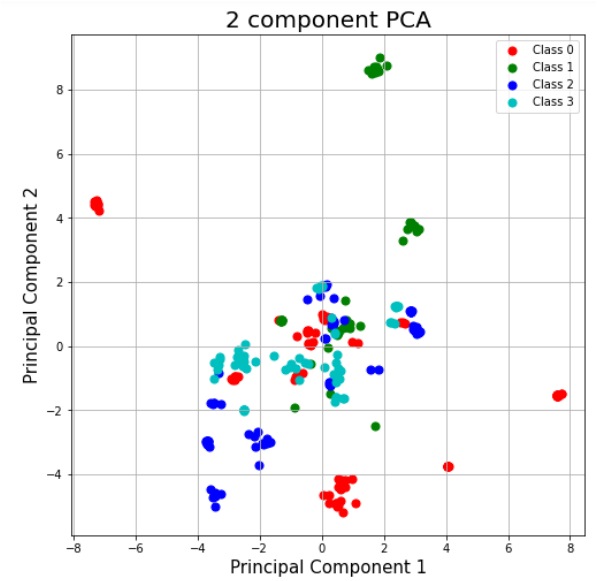

Figure 11 2-Dimensional space PCA for the File A

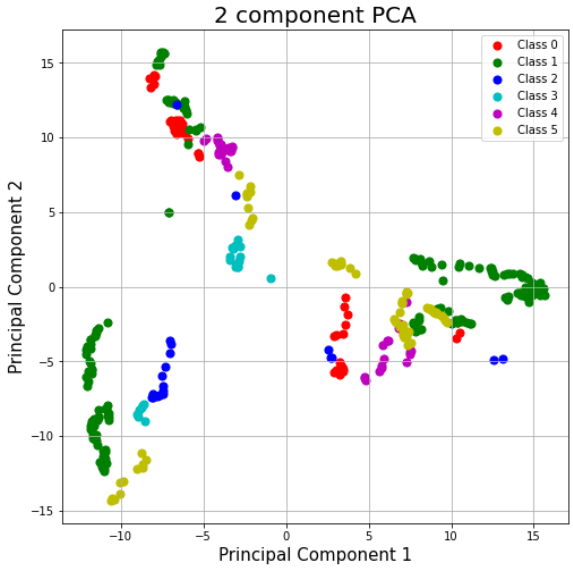

Figure 12 2-Dimensional Space PCA for the File B

c. 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…

```
# Variance explained
print (pca.explained_variance_)
print (pca.explained_variance_ratio_)
print (pca.explained_variance_ratio_.cumsum())
```

```
[9.83001912 9.30888676 8.83116858 7.44725225 7.18761929 7.01353873
 6.5801158  6.31716471 5.72338651 5.58659503 5.29641527 5.17121225
 4.71893954 4.48619833 4.26642895 3.68067809 3.59713899 3.51872424
 3.35304132 3.05100407 2.92143437 2.67670771 2.44146942 2.32388496
 2.19815722 2.05111922 1.97168308 1.81084416 1.53511885 1.42787367]
[0.06531502 0.06185238 0.05867821 0.04948285 0.04775774 0.04660107
 0.04372121 0.04197405 0.03802872 0.03711982 0.03519174 0.03435983
 0.03135473 0.0298083  0.02834805 0.02445606 0.02390099 0.02337997
 0.0222791  0.02027223 0.01941131 0.01778524 0.01622221 0.01544092
 0.01460553 0.01362855 0.01310074 0.01203205 0.01020001 0.00948743]
[0.06531502 0.1271674  0.18584561 0.23532846 0.2830862  0.32968727
 0.37340848 0.41538253 0.45341125 0.49053107 0.52572281 0.56008264
 0.59143737 0.62124567 0.64959372 0.67404978 0.69795077 0.72133074
 0.74360984 0.76388206 0.78329337 0.80107861 0.81730081 0.83274174
 0.84734727 0.86097582 0.87407656 0.88610861 0.89630862 0.90579605]
```

Figure 13 Variance explanation for the File A

We can notice that we were able to reduce the problem 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:

```
# Variance explained
print (pca.explained_variance_)
print (pca.explained_variance_ratio_)
print (pca.explained_variance_ratio_.cumsum())
```

```
[80.17062523 54.52811708 38.68471015 25.34030052 19.19106307 16.84143261
 10.58892763  9.70164483  8.5802855   7.00747384  6.89898301  6.2553547
  5.30559135  4.7563029   4.59099517  4.1609927   3.94254287  3.61783276
  3.48252828  3.17701331  2.97730072  2.73986188  2.48544436  2.30595553
  2.07826138  1.99151891  1.83133556  1.79778572  1.69419437  1.59367087]
[0.21326562 0.14505279 0.102907   0.06740892 0.05105104 0.04480068
 0.0281681  0.0258078  0.02282482 0.01864091 0.01835231 0.01664016
 0.01411365 0.01265246 0.01221272 0.01106885 0.01048774 0.00962397
 0.00926404 0.00845132 0.00792006 0.00728843 0.00661165 0.00613418
 0.00552848 0.00529773 0.00487162 0.00478237 0.00450681 0.0042394 ]
[0.21326562 0.35831841 0.46122542 0.52863434 0.57968538 0.62448606
 0.65265416 0.67846196 0.70128678 0.71992769 0.73827999 0.75492015
 0.76903381 0.78168627 0.79389899 0.80496784 0.81545558 0.82507955
 0.83434358 0.84279491 0.85071496 0.8580034  0.86461504 0.87074922
 0.8762777  0.88157544 0.88644706 0.89122943 0.89573624 0.89997564]
```

Figure 14 Variance explained for the File B

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

III. Classification models and cross validation

a. 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

b. 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 classification models' efficiency 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

The cross-validation tests made to measure the classification models accuracy were the following:

- Train the models and test them with a Stratified Cross-Validation
- Train the models and test them with a Normal Cross-Validation
- Train the models and test them with a Stratified Cross-Validation with Standardized Data
- Train the models and test them with a Normal Cross-Validation with Standardized Data
- Train the models and test them with a Stratified Cross-Validation with PCA transformed data
- Train the models and test them with a Normal Cross-Validation with PCA transformed data

	Normal Cross Validation with PCA	Stratified Cross Validation with PCA	Normal Cross Validation w/SD	Stratified Cross Validation w/SD	Normal Cross Validation	Stratified Cross Validation
K-Neighbors	0.930000	0.93	0.940000	0.940000	0.946667	0.946667
Naive-Bayes	0.870000	0.87	0.960000	0.960000	0.960000	0.960000
Neural Network	0.963333	0.97	0.976667	0.976667	0.973333	0.976667

Figure 15 Scores for different cross validations for the file A

	Normal Cross Validation with PCA	Stratified Cross Validation with PCA	Normal Cross Validation w/SD	Stratified Cross Validation w/SD	Normal Cross Validation	Stratified Cross Validation
K-Neighbors	0.874526	0.874526	0.869678	0.869678	0.869648	0.869648
Naive-Bayes	0.859922	0.859922	0.799548	0.799548	0.794700	0.794700
Neural Network	0.828154	0.833153	0.869708	0.859952	0.872117	0.877025

Figure 16 Scores for different cross validations for the file B

IV. Building the ensemble learning method

a. 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 and a normal cross-validation will be used to test the ensemble's accuracy.

b. Ilustrating the neural network

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

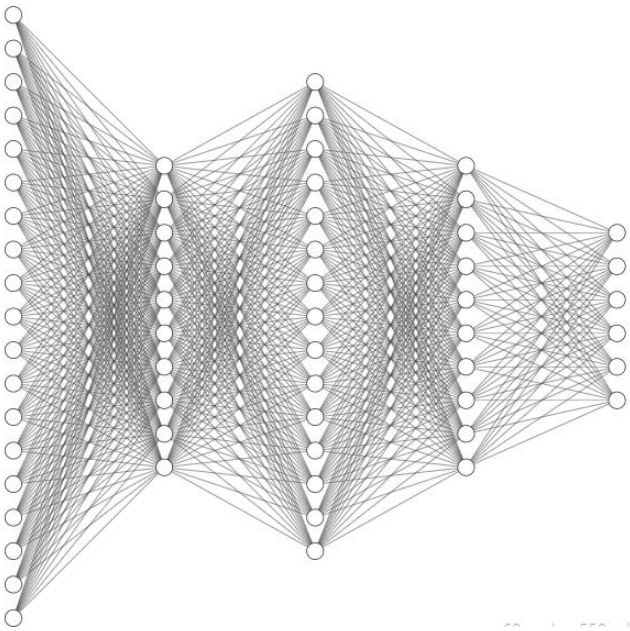

Figure 17 Neural Network

c. 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:

```
Accuracy: 0.9400 (+/- 0.0400)
```

	Ensemble
Fold 1	0.950000
Fold 2	0.966667
Fold 3	0.916667
Fold 4	0.950000
Fold 5	0.916667

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.

```
Accuracy: 0.8795 (+/- 0.3456)
```

	Ensemble
Fold 1	0.890244
Fold 2	0.987805
Fold 3	1.000000
Fold 4	0.939024
Fold 5	0.580247

Figure 19 Ensemble's accuracy for the file B

d. 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:

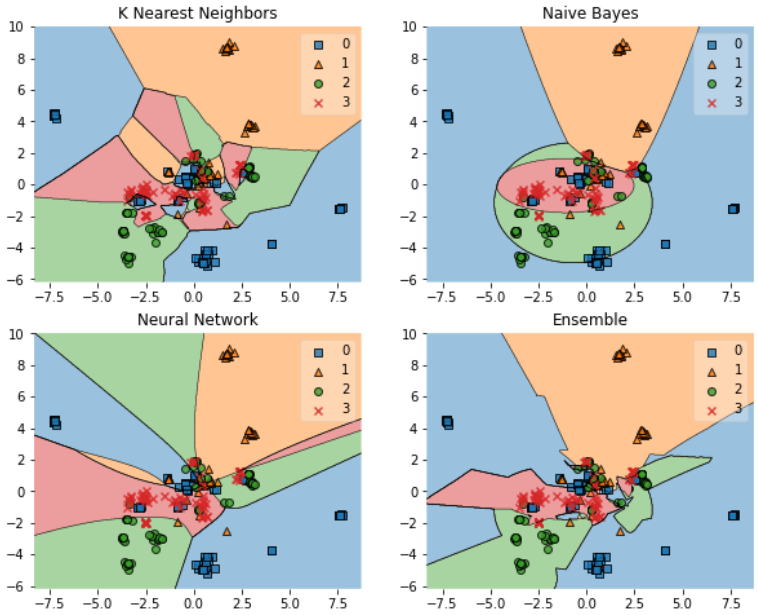

Figure 20 Ensemble's Decision boundaries for the file A

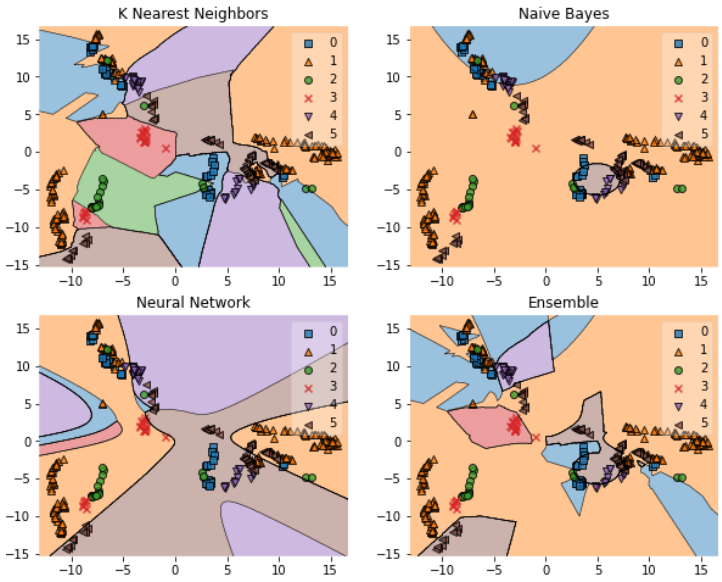

Figure 21 Ensemble's Decision boundaries for the file B

e. Logarithmic Loss

Now, 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. Log Loss 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.

```
Average Logarithmic loss: 0.1810 (+/- 0.0698)
```

	Logarithmic Loss
Fold 1	0.143792
Fold 2	0.196435
Fold 3	0.231074
Fold 4	0.180013
Fold 5	0.153682

Figure 22 Logarithmic Loss for the file A

```
Average Logarithmic loss: 0.5994 (+/- 1.7937)
```

	Logarithmic Loss
Fold 1	0.291714
Fold 2	0.106097
Fold 3	0.088220
Fold 4	0.318322
Fold 5	2.192822

Figure 23 Logarithmic Loss for the file B

V. Plotting the results

a. 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 F_1 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, F_1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F_1 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.

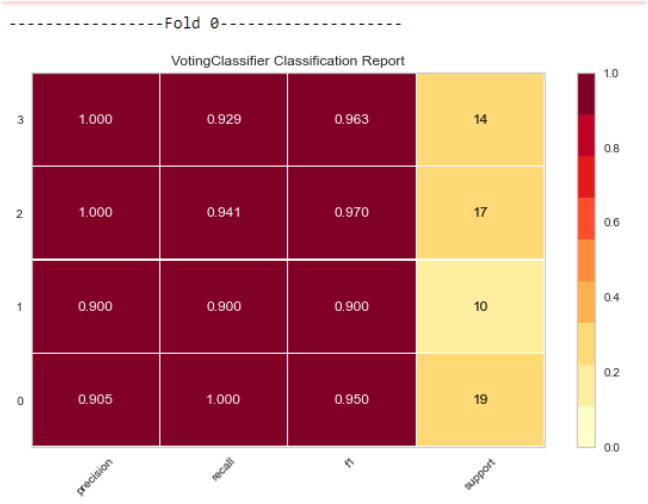

Figure 24 Fold 0 Classification report for the file A

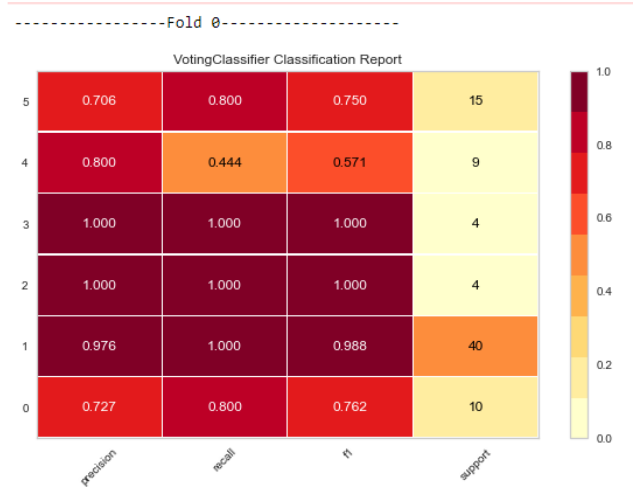

Figure 25 Fold 0 Classification report for the file B

b. 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:

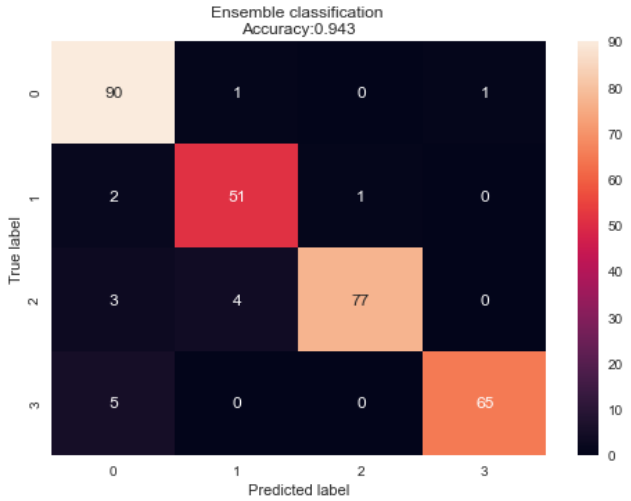

Figure 26 Confusion matrix for the fie A

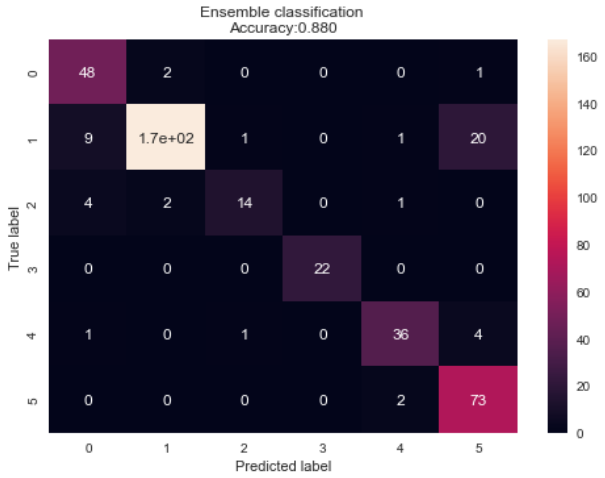

Figure 27 Confusion matrix for the fie B

c. Classification results scatter plot

Finally, In order to give an even better idea of how our ensemble well is doing, we can plott our results into a scatter plot. This will show us visually, how our results are being set into every single one of the targets in the classes. We just need to think our classification process as a probabilistic one with a soft-voting instead of hard.

In soft voting, every individual classifier provides a probability value that a specific data point belongs to a particular target class. The predictions are weighted by the classifier's importance and summed up. Then the target label with the greatest sum of weighted probabilities wins the vote. So, with this, we can plot our probabilistic results into a scatter plot in order to visualize what is happening inside the ensemble:

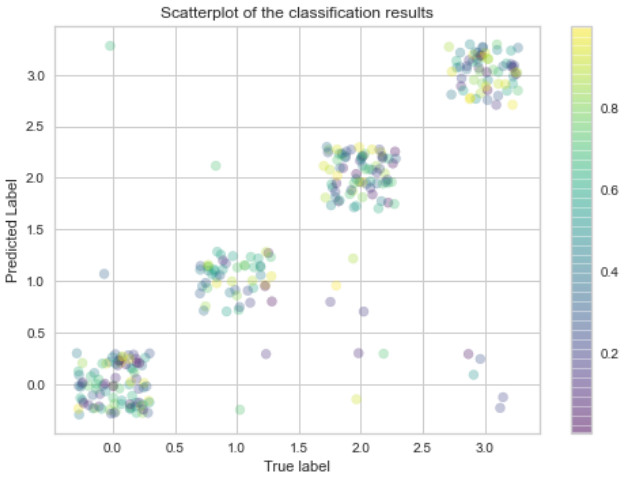

Figure 28 Classification results scatter plot for the file A

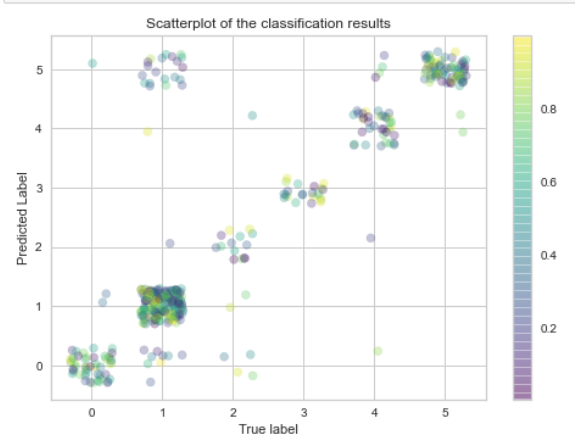

Figure 29 Classification results scatter plot for the file B

Conclusion

The way the majority systems are build, is based on the human behavior in day life. This document is not but the reflection of hours of hard work of thousands of engineers trying to create models that can produce answers to very common problems in the current world, such as: Whether a person is a good candidate for a credit bank or even the medical diagnosis for a rare illness. The models used here were assemble together in order to create or produce a much accurate answer to the reality of the results. And this just not proved to be a better solution, but also showed that the cooperative work between entities is sometimes the solution to a bigger problem, sometimes it is not necessary to build a better solution in order to improve the performance of some process or some algorithms, but to take a look back to whatever already works and try to find a way to make them cooperate. Sometimes it is not useful to reinvent the wheel, but to find a better use for it.

References

Brownlee, J. (2018, May 23). *A Gentle Introduction to k-fold Cross-Validation*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/k-fold-cross-validation/

Garduza, O. (2020, 04 20). *DataMining*. Retrieved from Github: https://github.com/OmarGard/DataMining

Polikar, D. R. (2008, December 22). *Ensemble learning*. Retrieved from Scholarpedia: http://www.scholarpedia.org/article/Ensemble_learning

Saxena, S. (n.d.). *Basic Concept of Classification (Data Mining)*. Retrieved June 03, 2020, from GeeksforGeeks: https://www.geeksforgeeks.org/basic-concept-classification-data-mining/