Development of a Data Mining Model to Report classification of any numeric Dataset

Francisco Silva (LJ1808409)
Master in Computer Science

Beihang University and FEUP
line 4: Beijing, China

Contact:
francisjssilva@gmail.com

I. INTRODUCTION (*HEADING 1*)

This work was developed in order to create a small application to evaluate classification algorithms and its differences when used in multiple and different Datasets filled with numeric data. In order to achieve this, multiple datasets have been used to test the application.

total sulfur dioxide	density	pH	sulphates	alcohol	quality
34	0.9978	3.51	0.56	9.4	5
67	0.9968	3.2	0.68	9.8	5
54	997	3.26	0.65	9.8	5
60	998	3.16	0.58	9.8	6
34	0.9978	3.51	0.56	9.4	5
40	0.9978	3.51	0.56	9.4	5

II. RELATED WORK

A. Scenario

The scenario of this application is based in the use of two different algorithms (Naive bayes and Decision Tree classification algorithm) to develop an application capable of:

- Evaluate the accuracy of the algorithm when a dataset is provided.
- Evaluate precision
- Evaluate recall
- Evaluate f1-score
- Be capable of understand the differences between each other.

B. Methodology

The methodology of this work was to collect some datasets from "kaggle.com" and use them to test the application.

The main dataset used for this paper was "winequality.csv" which is a dataset that evaluates the quality of a wine based in its different compositions (*fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol and the output quality*).

This dataset is composed of 11 classes, being one of them, the output class that we want to predict.

As said before, two distinct algorithms were used it this work, Naive Bayes and Decision Tree Algorithms. Naive bayes was implemented from scratch and Decision Tree was implemented with the help os "scikit-learn" library.

The whole project was made in python 3.0.

III. ARCHITECTURE AND TOPOLOGY

The architecture used for this work represents 5 phases. First handle the data of the dataset, then summarize it, prepare the algorithm, summarise the dataset and its attributes by class, make predictions and finally output the classification report.

As said before, the whole application was made in python 3.0 and scikit-learn library was used to help in the creation of the decision tree.

A. Architecture of Naive bayes

Naive bayes was implemented from scratch and its architecture is composed by four phases, as follow:

- Handle data: in this phase, the data is read from the file path provided with the use of loadCsv function from python library's. Then, it we converted into float, split in train set and test set with a split ration provided by the user. This

means if the user inputs a 0.6 split ratio, the dataset will be split in random 60% training set and 30% random test set.

• Summarize data: After handle data, the data is summarized with the use of gaussian function. Here, five sub steps were done. The data is separated by class, mean is calculated using gaussian distribution in order to understand its tendency, standard deviation is done, and then, the dataset and its attribute by classes are summarized.

• Make prediction: Prediction is made in base of gaussian probability.

• Output Classification report: Finally, with the help of *scikit-learn library,* classification report is outputted by calling *confusion_matrix()* and *classification_report()*

B. Architecture of Decision Tree Algorithm

Decision Tree was implemented with the help of *scikit-learn* library and its architecture is composed by four phases:

• Handle data: in this phase, the same is done as in Naive Bayes Algorithm. LoadCsv was called, then data was treated using the functions provided in *scikit-learn library*, namely *np.delete()* to split output class from other classes, after *train_test_split()* was called in order to split dataset in training set and test set.

• Creation of Decison Tree: In this step, decision tree is created using again scikit-learn library calling *DecisionTreeClassifier()* and *classifier.fit()*.

• Make Prediction: Then, predictions are made using *classifier.predict(X_test)*.

• Output Classification report: Finally, and again, with the help of *scikit-learn library,* classification report is outputted by calling *confusion_matrix()* and *classification_report()*.

Both of the algorithms seems to be well implemented, the results are as expected regarding both of them.

IV. USER INTERFACE

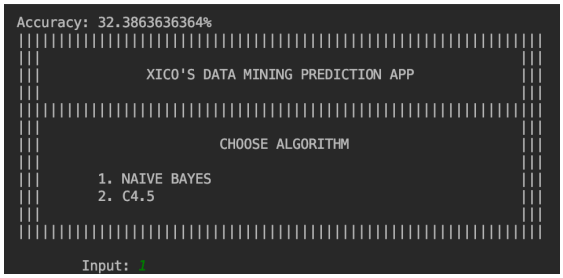
The user interface is very simple, outputted in command line, in text.

I want o emphasise that all the input/output operations verify if the user is inputing wrong characters or wrong column names and even wrong file names.

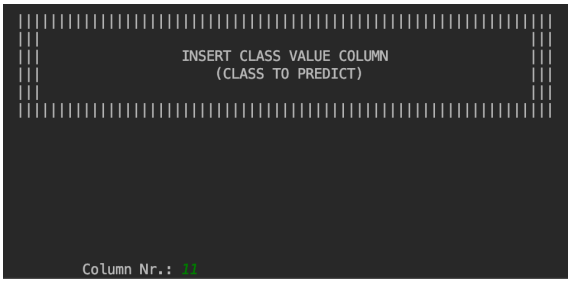
The interface is composed of four steps, namely:

• Choose an algorithm: Both algorithm names are outputted and user can choose between 1 and 2. The user just need to write the number 1 or 2 in order to choose the option he wants. This pattern is the same in the next three steps. It possible to understand better how user should use the application looking for the following images:

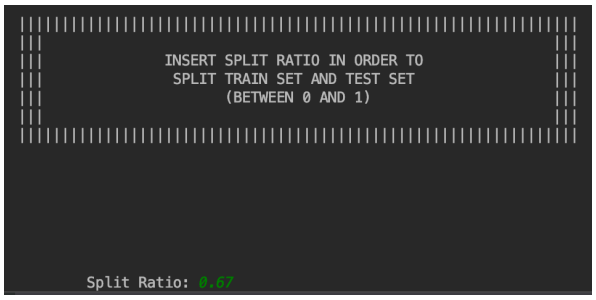
• Input path name: Here, user should input the file path of the dataset.



• Choose class output number: User must input the output class number (column number). This means that if the user wants to predict the quality of the wine, he should introduce the correspondent column number.



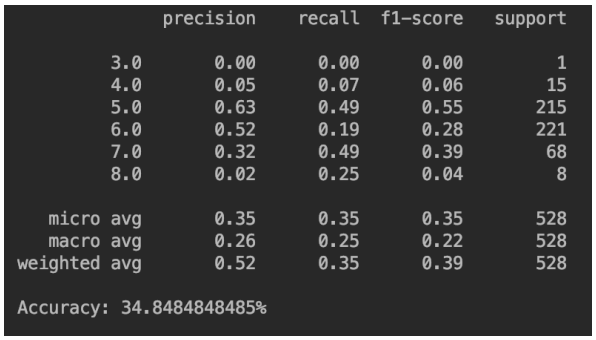
• Input Split Ratio: Finally user should input split ration in order to split the training set and test set.



V. OBSERVATIONS

Regarding to observations of the main dataset chosen to test this application *(winequality.csv)* it was possible to conclude that Decision tree is more accurate than Naive Bayes algorithm. In order to conclude this I start by running the program with both algorithms side by side.

The output message for Naive Bayes was the following:

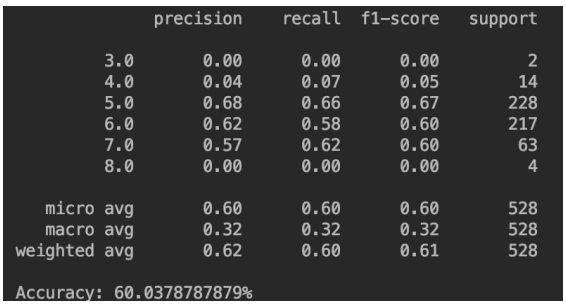


	precision	recall	f1-score	support
3.0	0.00	0.00	0.00	1
4.0	0.05	0.07	0.06	15
5.0	0.63	0.49	0.55	215
6.0	0.52	0.19	0.28	221
7.0	0.32	0.49	0.39	68
8.0	0.02	0.25	0.04	8
micro avg	0.35	0.35	0.35	528
macro avg	0.26	0.25	0.22	528
weighted avg	0.52	0.35	0.39	528

Accuracy: 34.8484848485%

It is possible to understand that the accuracy of this algorithm for this dataset was not the best, also precision is bad for almost of all the output possibilities. This means that either the algorithm is not so good or that the evaluation of the wine in the data set is not accurate.

As for Decision tree we can observe some critical differences in the output results:



	precision	recall	f1-score	support
3.0	0.00	0.00	0.00	2
4.0	0.04	0.07	0.05	14
5.0	0.68	0.66	0.67	228
6.0	0.62	0.58	0.60	217
7.0	0.57	0.62	0.60	63
8.0	0.00	0.00	0.00	4
micro avg	0.60	0.60	0.60	528
macro avg	0.32	0.32	0.32	528
weighted avg	0.62	0.60	0.61	528

Accuracy: 60.0378787879%

Regarding the image above is easy to understand that there's an increase of almost 200% of the accuracy even tough precision, recall and f1-score are pretty similar to Naive Bayes. In this one is notable that the algorithm seems to be a accurate but still 60% of accuracy is not so good as expected for prediction which probably means that the *winequality.csv* dataset is not very coherent.

Regarding both of the output results I can conclude that Decision tree is a more optimisable algorithm than Naive Bayes. Even tough results are very different from both of them, which is natural, since naive bays is not so efficient as Decision tree algorithms.

VI. CONCLUSION

With this work was possible to create a pretty simple tool to test the feasibility of a dataset and its accuracy.

Also was possible to conclude that Decision tree algorithms are more accurate than naive bayes.

VII. REFERENCES

- https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/
- https://www.kaggle.com/datasets
- https://www.upwork.com/hiring/data/15-python-libraries-data-science/
- https://machinelearningmastery.com/
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- https://en.wikipedia.org/wiki/Decision_tree_learning