Classifying the White Wine Quality Using Machine Learning

Capstone Project Report
Machine Learning Engineer Nanodegree

ELEFTHERIOS KOULIERAKIS
March 28, 2018

Contents

List of Figures

List of Tables

Definition

Project Overview

Wine, one of the most enjoyed and popular beverages, is produced and consumed for almost 8000 years [1]. It has become an inseparable part of the human life and the wine and spirits industry is one of the most challenging sectors nowadays. Companies that are active in this sector, usually hire experts in order to assert and estimate the quality of their products. This process can be quite time consuming and expensive, and for that reason, there is a need to change the way the quality of the wine is evaluated using machine learning.

For this project, we will use the data provided in the Machine Learning Repository [2] and we will analyse the white wine dataset. This dataset was introduced by Paulo Cortez et al [3] in 2009. Each entry in this dataset is given a label from zero to ten and this label corresponds to the quality of wine according to experts. The scope of this project is to use machine learning in order to develop a model that predicts the quality of the wine given its physiochemical features. This is a challenging task because the data is unbalanced since there are munch more normal wines than excellent or poor ones and it also has several features that have different formats. Apart from that there is no information about the grape type, the wine brand and the wine selling price which is quite important for such a task.

Problem Statement

The goal of this project is to develop a machine learning model in order to predict the quality class for each entry in our white-wine dataset. The information about the quality of the wine is given in a scale from 0 to 10. For this project, we will transform this problem to a binary classification task. In particular, we distinguish two different classes for every wine: Class 0 means that the quality score is lower than 7 and class 1 means that the quality score is equal to or greater than 7. In other words, class 0 corresponds to bad or normal quality wines and class 1 corresponds to good quality wines. The machine learning model to be developed, should provide the class prediction.

This model should be able to correctly classify most of the entries in our data. For that reason we employ several algorithms that have been extensively used for classification purposes. These algorithms are Decision Trees, Random Forests and Adaptive Boosting. The algorithms that had the best performance were selected, fine-tuned, and compared to a benchmark model. Other techniques such as developing a stacking meta-classifier and a soft voting classifier are also used in order to achieve the best performance. The final algorithm is expected to be useful for classifying the quality of wine, given its physiochemical features.

Metrics

This is a classification task with unbalanced data. For that reason the most appropriate metric to use is the F_1 score because it considers both the precision and the recall [5]. In order to define the F_1 score, we should at first define precision and recall. Precision is given by the following formula:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall is given by the following formula:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Finally, the F_1 score is given by the following formula:

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

In order to define these metrics using the project context, the true positives are the correctly classified good quality wines (class 1), the false negatives are wines that belong to class 1 but were classified in class 0, and the false negatives are wines that belong to class 0 and were classified in class 1. In other words, if the precision score is high we are sure then that most of the wines that belong for instance to class 1 were correctly classified in that class. If the recall score is high, then we are sure that most of the wines that were for instance classified in class 1 indeed belonged to that class.

The F_1 score can be interpreted as a weighted average of the precision and recall, where an F_1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F_1 score are equal. Additional to the F_1 score, the *accuracy* metric is also used to see how many of the wine entries were correctly classified. As a metric, *accuracy* is quite easy to understand: 82% accuracy means that 82% of the tested entries were correctly classified.

Analysis

Data Exploration

The White Wine Quality Dataset [2] was introduced in 2009 and contains information about 4898 wine entries. There is information about several physiochemical features: fixed acidity, volotile acidity, citric acidity, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quiality, which will be used to create our target column. The descriptive statistics about all the aforementioned features are shown on Table 1.

Table 1: Descriptive Statistics of the Dataset.

	mean	std	min	25%	50%	75%	max
fixed acidity	6.854788	0.843868	3.80000	6.300000	6.80000	7.3000	14.200
volatile acidity	0.278241	0.100795	0.08000	0.210000	0.26000	0.3200	1.100
citric acid	0.334192	0.121020	0.00000	0.270000	0.32000	0.3900	1.660
residual sugar	6.391415	5.072058	0.60000	1.700000	5.20000	9.9000	65.800
chlorides	0.045772	0.021848	0.00900	0.036000	0.04300	0.0500	0.346
free sulfur dioxide	35.308085	17.007137	2.00000	23.000000	34.00000	46.0000	289.000
total sulfur dioxide	138.360657	42.498065	9.00000	108.000000	134.00000	167.0000	440.000
density	1.548363	6.838541	0.98711	0.991723	0.99374	0.9961	103.898
pH	3.188267	0.151001	2.72000	3.090000	3.18000	3.2800	3.820
sulphates	0.489847	0.114126	0.22000	0.410000	0.47000	0.5500	1.080
alcohol	10.514181	1.230618	8.00000	9.500000	10.40000	11.4000	14.200
quality	5.877909	0.885639	3.00000	5.000000	6.00000	6.0000	9.000

There are few interesting points to mention: According to the table above, the average value of the pH is 3.188 and the maximum is 3.82. That means that the pH of most wines is very close to the mean. Apart from that, the quality score of around 75% of the total entries is less or equal to 6. Finally, we observe that all the features are numerical. In addition we create the target column: if the quality core of the wine is equal to or less than 6 then the class of the wine is 0 otherwise the class of the wine is 1. After applying this rule, we observe that there are 3838 entries that belong to class 0 and 1060 entries that belong to class 1. In other words, 78.4% of the entries belong to class 0 and 21.6% of the entries belong to class 1.

Exploratory Visualization

We would like to explore whether there is a clear correlation between the features, and for that reason, we created the feature pairplot shown on Figure 1. In this pairplot, a scatter plot is made for both features, and on the diagonal, there is a univariate distribution for each feature. By observing this figure, we can't really see a very strong correlation between the different features.

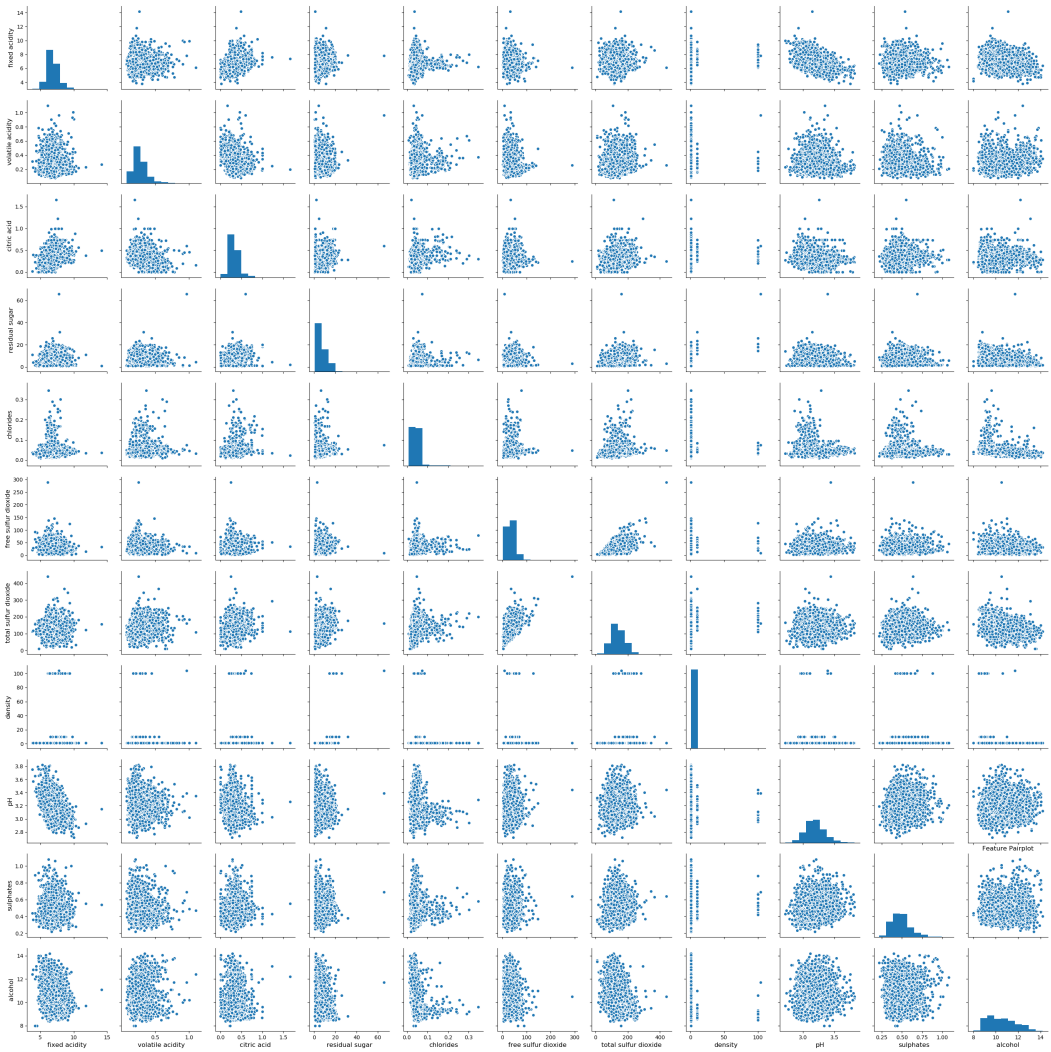


Figure 1: Feature Pairplot.

Our experience suggests that in general, good quality wines have high alcohol concentrations compared to the average or bad quality wines. For that reason, we would like to see if there is a difference between the two classes when it comes to alcohol. For that reason, we create the following density class plot shown on Figure 2, which encodes the density of observations on the x axis for alcohol with height along the other axis for both classes. Obviously, the area for both density distributions is 1.

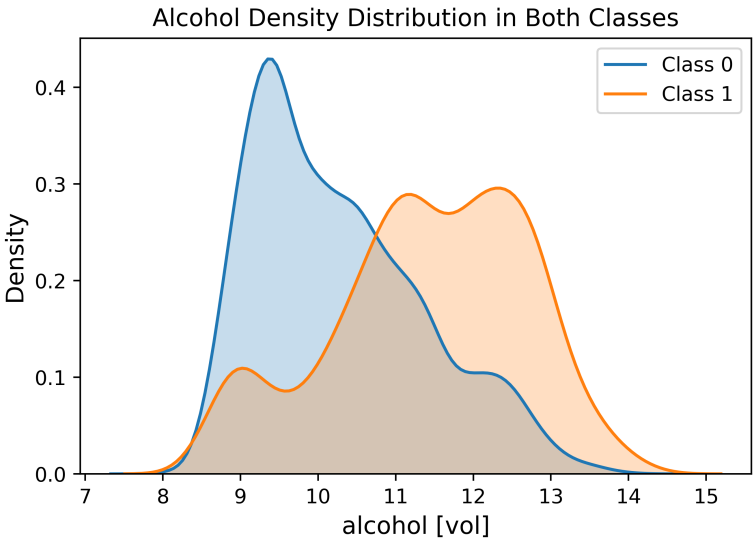


Figure 2: Alcohol Density Distribution for Both Classes.

We observe that indeed our expectation was correct. Wines that belong to class 1 (good quality) are expected to have higher alcohol concentration compared to wines that belong to class 0. In the following part of this report, we will show how important the alcohol concentration is in order to classify the wines. Apart from the alcohol concentration, we created the same density class plots for all features, but there was not such a significant difference between the two classes.

Algorithms and Techniques

For this project, we initially used three different algorithms: Decision Trees, Random Forest, and Adaptive Boosting. Apart from these we also developed a stacking meta classifier and a soft voting classifier. Each of these algorithms is briefly discussed below:

Decision Trees

Decision Trees are one of the most popular algorithms and they consist a non-parametric supervised learning method used for both classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features [6].
Important Training Parameters:
max_depth: The maximum depth of the tree. *min_samples_split:* The minimum number of samples required to split an internal node. *min_samples_leaf:* The minimum number of samples required to be at a leaf node.

Random Forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting [7].

Important Training Parameters:
Random Forest has similar training parameters with the decision tree. Apart from that, there is one more important parameter called *n_estimators* which is the number of trees in the forest.

Adaptive Boosting

An Adaptive Boosting classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases [8].
Important Training Parameters:
base_estimator: The base estimator from which the boosted ensemble is built. *n_estimators:* The maximum number of estimators at which boosting is terminated.

Stacking Meta-Classifier

Stacking is an ensemble learning technique to combine multiple classification models via a meta-classifier. The individual classification models are trained based on the complete training set; then, the meta-classifier is fitted based on the outputs -meta-features- of the individual classification models in the ensemble. The meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble [9].

Voting Classifier

This is a soft voting or majority rule classifier for unfitted estimators. For both the stacking Meta-Classifier and the soft voting classifier, we used the Benchmark model (Naive Bayes), the Decision Tree and the Random Forest Classifier as estimators. We didn't use the Adaptive boosting classifier because it performed worse than the rest.

Benchmark

One of the most fundamental classification algorithms in machine learning is Naive Bayes, and this is the algorithm that we will use as a benchmark. Naive Bayes is based on the Bayes' theorem with the 'naive' assumption of indipendence between every pair of features. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

Methodology

Data Preprocessing

The dataset obtained from the machine learning repository was quite clean. Our first step was to create an additional target feature based on the quality score column: If the quality score was equal to or greater than 7, then the wine was classified in the class 1, otherwise the wine was classified in class 0. It is important to mention that the `quality` score column was not further used for our analysis. The rest of the columns were used as training features for the classification task. Before applying any algorithm, the data were randomly divided into a training and a testing set. We used the `train_test_split` function from the `sklearn.model_selection` module. We used 67% of the data to train the models and 33% to test them.

Implementation

For this project we used the Python programming language and several supporting libraries. All the implementation is provided and described at the `White_Wine_Quality_Analysis.ipyn`. In general, the following steps were taken in the implementation part:

- Import the data
  This is done using the `pandas` library.
- Descriptive statistics and exploratory visualizations
  We used `matplotlib.pyplot` for most of the visualizations of this module. Apart from that, we also used `seaborn.kdeplot` to plot a few density distribution plots to compare some features for both classes. We also used the `.apply()` method to create the right labels for both classes.
- Randomly split the data into a train and a test set
  We used the `train_test_split` function from `sklearn` to split the data.
- Train and test the Initial Models
  We created the `model_performance` function that returns the training time of the model, its accuracy and the F_1 score.
  - train and test the benchmark model
    From `sklearn.naive_bayes` we imported GaussianNB and used the default settings.
  - train and test the rest models
    From `sklearn.tree` we imported `DecisionTreeClassifier` and used the default settings and from `sklearn.ensemble` we imported `RandomForestClassifier` and `AdaBoostClassifier` and also used the default settings.
- Fine-tune the models that performed better than the benchmark model
  Since the performance of the Random Forest and the Decision Tree was the best, we used grid search on the parameters described in the Algorithms section (`n_estimators` for the Random Forest and `min_samples_split`, `min _samples_leaf` and `min_samples_leaf` for both) to further improve them. These parameters are shown on Table 2. Superscript rf means that this was only used in the parameter grid of the random forest and superscript dt means that this was only used in the parameter grid of the decision tree.
- Develop the meta-classifiers
  - Develop the stacking classifier
    From `mlxtend.classifier` we imported `StackingClassifier` and used the default settings.
  - Develop the voting classifier From `sklearn.ensemble` we imported `VotingClassifier` and used the default settings.

We used the `sklearn` and the `mlxtend` libraries in python to train and test all the developed models. In order to do so, first we instantiate the model and then we call the `.fit()` and `.predict()` methods to train the model and make predictions respectively. Apart from that we also used the `.feature_importances_` attribute of the Decision Trees object to see which are the most dominant features in the classification task.

There was one complication during the fine tuning of the random forest. This task was very time-consuming because we also tried to optimise the `n_estimators` feature. For that reason

we had to limit the parameter grid in the rest of the parameters and this is why some parameters have the dt superscript since they were only used to optimise the decision tree.

Table 2: Grid Search Parameters

n_estimators^{rf}	8	10	12	14	15	16	17
max_depth	None	15	20	23	25	27	30
min_samples_split	2	3	4	5^{dt}			
min_samples_leaf	1	2	3	4^{dt}	5^{dt}		

In order to compute the *accuracy* and the F_1 score we imported the `f1_score` and the `accuracy_score` from the `sklearn.metrics` module. The accuracy was only computed just to give an estimation on how good the model performed in general, but all the models were compared based on the F_1 score. Apart from that, we wanted to see how fast each the initial classifiers is trained, and for that reason we imported the `time` library.

Refinement

The benchmark model and the rest of the initial classifiers (Decision Trees, Random Forest, and Adaptive Boosting) were trained and tested on the same train and test part of the data. We found out that the worst performance both in training time and F_1 score was achieved by the Adaptive Boosting Classifier and for that reason we did not try to further improve it. The best performance in terms of the F_1 score was achieved by the Random Forest classifier. The second best was achieved by the Decision Trees, and the third best F_1 score was achieved by the Benchmark model (Naive Bayes Classifier).

In our first step of refinement, we worked on fine-tuning the Decision Tree and the Random Forest classifiers. For that reason, we used both grid search and cross validation and we observed that only the performance of the Random Forest was improved. Our second step was to create a stacking meta-classifier and as estimators, we used the Naive Bayes, the fine-tuned Decision Tree and the fine-tuned Random Forest Classifiers. We observed that the stacking classifier was not better that the fine-tuned Random Forest. Our last step was to create a simple Voting Classifier, and as estimators for this model we used again the Naive Bayes, the fine-tuned Decision Tree and the fine-tuned Random Forest. We observed that the performance of the Voting Classifier was slightly improved compared to the fine-tuned Random Forest. In other words, the voting classifier achieved the best F_1 score. All the results and the scores of each model are analysed in the next section.

Results

Model Evaluation and Validation

Our first step was to compare the benchmark model (Naive Bayes) with the rest classifiers (Decision Tree, Random Forest, and Adaptive Boosting). Before computing the *accuracy* and the F_1 score for each of the aforementioned models, we were also interested in the time each model needed to be trained. The results for the training time are shown in Figure 3.

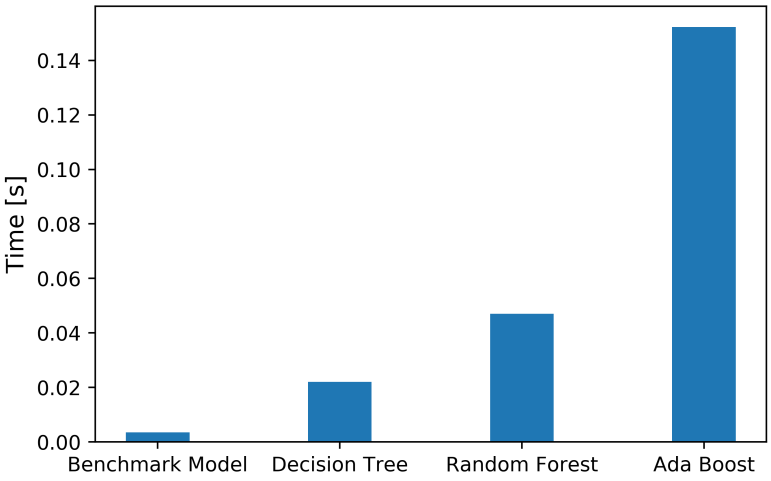


Figure 3: Training Time for the Initial Models.

We observe that the training of the Adaptive Boosting classifier is a time consuming process. On the other hand, the Naive Bayes is trained significantly faster than every other algorithm. Apart from that, the Decision Trees were trained faster than the Random Forest. Our next step was to compute the accuracy for every model. The results are shown in Figure 4.

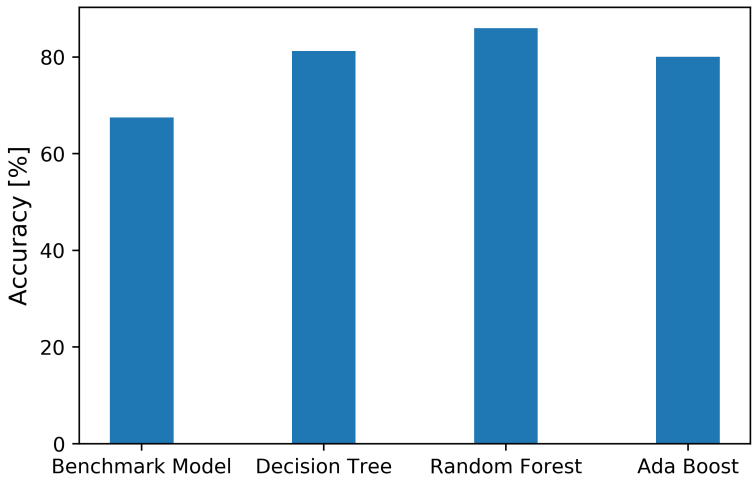


Figure 4: Accuracy Score for the Initial Models.

We observe that the best accuracy score was achieved by the Random Forest Classifier (86%) and the worst accuracy was obtained by the Benchmark Model (67.5%). Our last and most

important step in the validation of the initial models was the computation of their F_1 score. The results are shown in Figure 5.

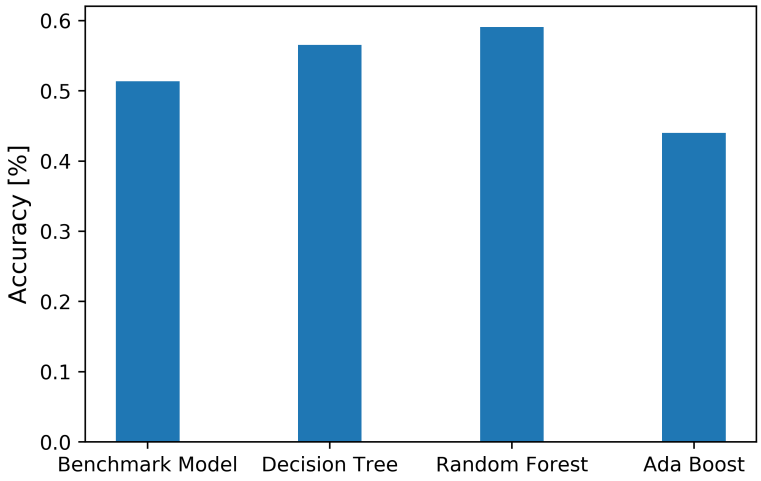


Figure 5: F_1 Score for the Initial Models.

We observe that the best score was achieved by the Random Forest Classifier (0.591). The Decision Tree had comparable performance (0.565) to the Random Forest. The Benchmark model had worse performance (0.514) than the Decision Tree but not as bad as the performance of the Adaptive Boosting (0.44). Apart from that we were interested to know which are the most informative features. The results are shown in Figure 6. As expected, we observe that indeed the alcohol concentration is the most important feature for this classification task. This comes to complete agreement with what we discussed about Figure 2. We also observe that the rest features all less informative than the alcohol concentration.

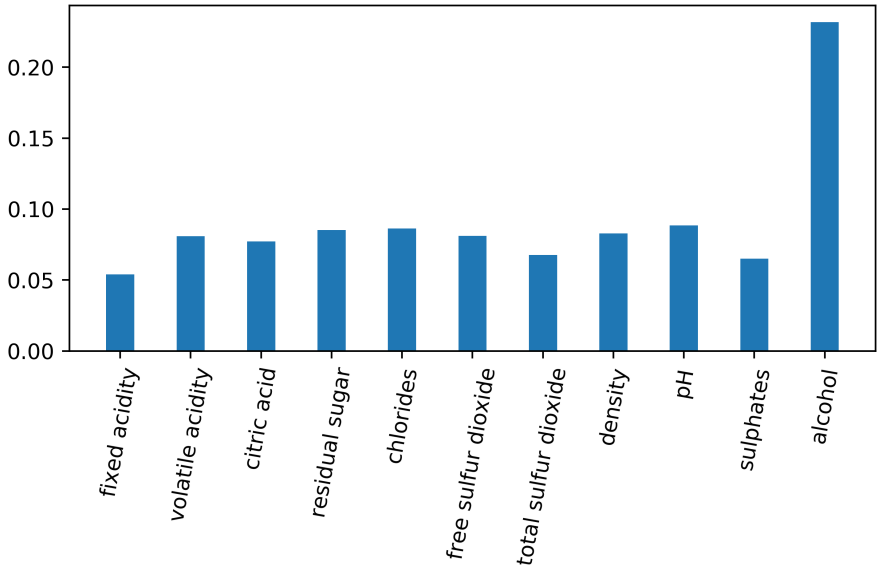


Figure 6: Decision Tree Feature Importance.

Since the performance of the Random Forest and the Decision Tree was the best, we used grid search on the parameters described in the Algorithms section (`n_estimators` for the Random Forest and `min_samples_split`, `min_samples_leaf` and `min_samples_leaf` for both) to further improve them. The performance of the Decision Tree slightly improved (0.571) but the fine tuned Random Forest had a better F_1 score (0.621). Our next step was to build a stacking Meta-Classifier by using the fine-tuned Random Forest along with the fine-tuned Decision Tree and the Naive Bayes classifier. On top of these estimators, a logistic regression was added to develop the complete stacking classifier. The F_1 score of this classifier was 0.567 which is lower than the F_1 score of the fine tuned Random Forest. Our last step was to make a Voting Classifier. Again, we used the same estimators: Naive Bayes, fine-tuned Decision Tree and fine-tuned Random Forest. The F_1 score achieved by the voting classifier was the best achieved (0.656). The results are summarized in Figure 7.

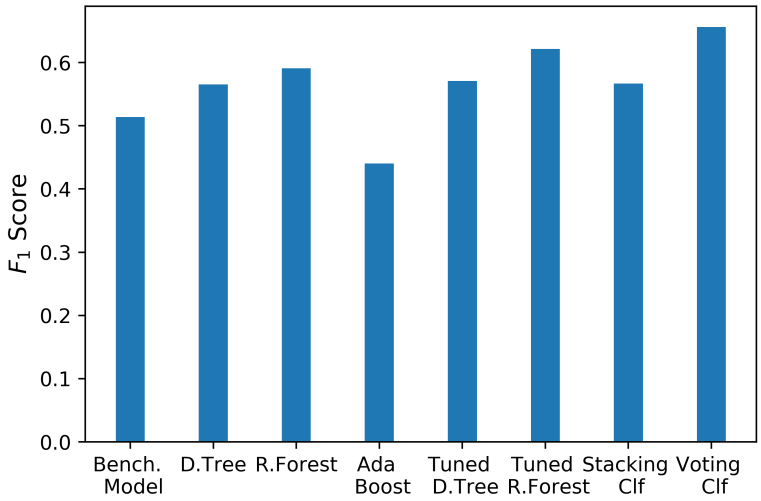


Figure 7: F_1 Score for All Models.

Since the Voting Classifier Achieved better F_1 score than the fine tuned Random Forest, we select this algorithm as the best model developed for this project. Another reason to select this model is that even though the difference between them is not very significant, there might be some extra features learned better by the Gaussian Naive Bayes and the fine-tuned Decision Tree and this is why the voting classifier results in higher F_1 score and performs better.

Justification

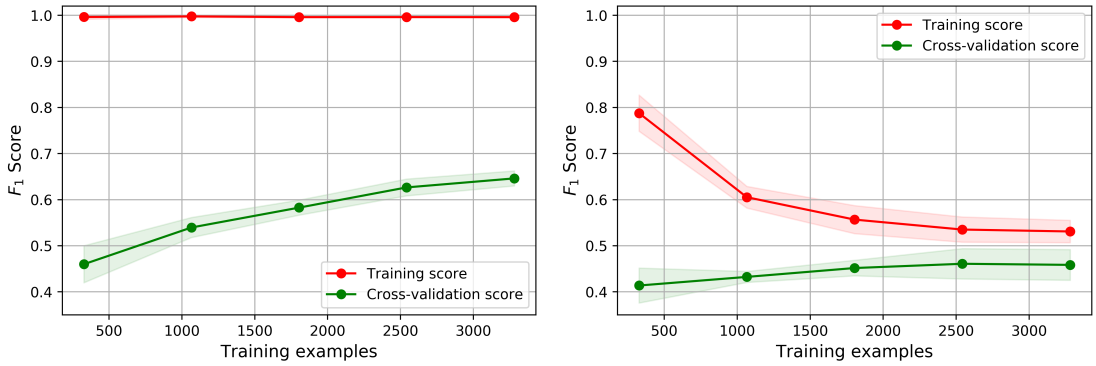
As mentioned before, the best model in terms of the F_1 score is the Voting Classifier (0.656). The F_1 score of the Benchmark Model is 0.514. The Voting classifier is better than the Benchmark model also in terms of the Accuracy score (0.851 against 0.675). We consider that the Voting Classifier is decent. The final results are quite good but not excellent. Further improvement could of course be realized by employing different algorithms such as Support Vector Machines and Neural Networks.

We also plotted the learning curves of the voting classifier (which are shown in Figure 8 and are further discussed in the Free-Form Visualization subsection). We observed that there was high variance in the model, but the positive point was the fact that adding more data to the model results in better performance and higher values of the F_1 score. That means that the model will better generalize if it is provided with more data. To summarize, given the fact that the voting classifier achieved the highest F_1 score and is able to generalize (if provided with more data), we select this as our final and best development for this project. Given also the accuracy of the model which was 84.3%, we further support that this model is capable to solve the problem.

Conclusion

Free-Form Visualization

A way to estimate if a model would perform better by adding more data is to plot its learning curves. Figure 8a shows the learning curves of the Voting Classifier in terms of F_1 score. The training F_1 score is much higher than the validation score and close to 1. This is astrong indication of having high variance and the model has learned train dataset patterns that are not available in the test set. In order to further improve the model we could add more data and to further support this argument we may observe that increasing the training examples results in higher F_1 score.



(a) Voting Classifier

(b) Adaptive Boosting

Figure 8: Learning Curves of the training and cross-validation F_1 score

In order to further support this argument, we show in Figure 8b the learning curves of the Adaptive Boosting Classifier which was the model that had the worst performance in terms of F_1 score. In this case, the difference between the coss-validation score and the training score is big as it is in the case of the voting classifier and the model seems to be capable to generalize, but we may argue that adding more data to this model will not improve its performance and this is also shown in the graph: Increasing the number of the training examples does not result in higher values of F_1 score. Apart from that, the cross validation F_1 score of the voting classifier using less than 500 training examples is higher than the cross validation F_1 score of the Adaptive Boosting classifier using more than 3000 training examples. This is one additional reason (apart from the highest F_1 score) why the Voting Classifier is more promising even though there is an indication of high variance.

Reflection

In this project, we developed an end to end solution to classify the quality of the wine. The delivered model is able to support wine tasting evaluations for the white type of wine. Several steps were taken before reaching to the final solution. Our initial implementation was the training and testing of the Naive Bayes Benchmark along with the rest classifiers (Adaptive Boosting, Decision Trees, Random Forest). The Decision Tree and the Random Forest classifier seemed to perform better than the rest, and for that reason we used grid search and cross validation to

fine tune them. The Random forest was improved and was used as an estimator for the stacking meta-classifier along with the Naive Bayes and the fine-tuned Decision Tree. The stacking meta-classifier though did not achieve better results and our final implementation was to create a voting classifier using the fine tuned random forest with the naive bayes and the fine-tuned decision tree. The voting classifier performed better than the fine tuned Random Forest.

One challenging and interesting aspect of this project was the fact that not all features were as informative as the alcohol concentration was. The feature importances of the Decision Tree plotted in Figure 6 indicated how dominant this feature is when it comes to the quality of wine. Another challenge was that the dataset was imbalanced since only 21.6% of the wines belonged to class 1 (good quality). Apart from this, the final challenge was to develop a better model than the fine-tuned Random Forest classifier. Our first attempt with the stacking meta-classifier was not successful in doing so, and the voting classifier slightly outperformed the fine-tuned random forest.

Improvement

Our final implementation was the development of a model that can quite accurately classify the quality of the white wine. Further improvements can be realized to create a model that can perform this classification task more efficiently. The grid search combined with the cross validation was a time-consming process, and for that using randomized search cross validation [11] should be considered next time. Gradient boosting [12] has become very popular nowadays and has proven to be an inseparable algorithm from kaggle competitions and it would be interesting to employ this algorithm to see if our final solution can be improved. Apart from Gradient Boosting, Neural Networks have also become one of the most traditional approaches for almost every classification task. Many methods can be used to further improve a neural network such as adding dropout layers, fine-tuning the Neural Network hyper-parameters and optimizing the cost function. All the aforementioned improvements can be compared to our final solution that is presented in this report and probably combined with the voting or stacking classifier in order to achieve even better performance.

References

[1] Gina Hames *Alcohol in History*.
Themes in World History, Routlege Publications 2009, Page 17.

[2] The link to the repository:
https://archive.ics.uci.edu/ml/datasets/wine+quality

[3] The link to Paublo Cortez's personal page:
http://www3.dsi.uminho.pt/pcortez/Home.html

[4] Paulo Cortez, Antonio Cerdeira, Fernando Almeida, Telmo Matos,Jose Reis *Modeling wine preferences by data mining from physicochemical properties*. Decision Support Systems Volume 47, Issue 4, November 2009, Pages 547-553.

[5] F_1 score metric at sklearn:
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[6] Decision Trees at sklearn:
http://scikit-learn.org/stable/modules/tree.html

[7] Random Forest Classifier at sklearn:
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[8] Adaptive Boosting Classifier at sklearn:
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[9] Stacking Classifier at mlxtend:
https://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/

[10] Naive Bayes Classifier at sklearn library:
http://scikit-learn.org/stable/modules/naive_bayes.html

[11] Randomized Search Cross Validation at sklearn library:
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.h

[12] Gradient Boosting:
https://en.wikipedia.org/wiki/Gradient_boosting