Machine Learning
Teacher: Stephan Schiffel
Reykjavík University

Fall 2021
T-504-ITML, Vélrænt Gagnanám

Project 1: Supervised Learning

Anton Björn Mayböck Helgason, antonm19@ru.is
Björgvin Ægir Elisson, bjorgvine19@ru.is

Group 1

October 5, 2021

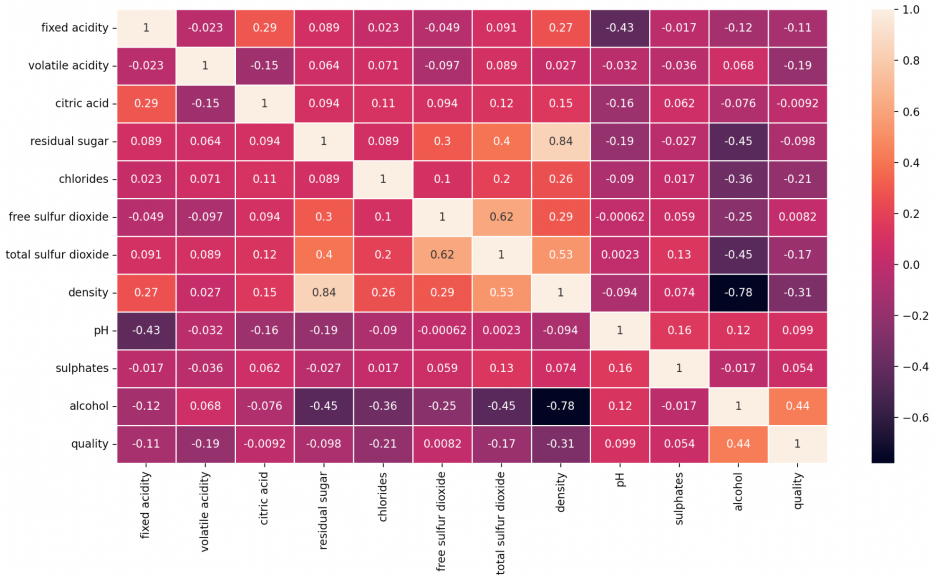
I. Introduction

The data is a .csv file downloaded from Kaggle. It has a total of 4898 variants of a type of white wine, each with 11 input variables, all of which describe physicochemical properties of the respective instance, like citric acid, pH, alcohol, etc and an output variable, or the class, which is "quality". The wine type in question is the Portuguese "Vinho Verde" wine which is a type of wine originating from the historic Minho province in northern Portugal. Our data only includes variants of white "Vinho Verde" wine although "Vinho Verde" wine can also be red, rosé, and more. The purpose of this project is to determine a machine learning classifier that classifies the data most accurately. That is determined using hyperparameter tuning and Grid-Search Cross Validation, hereafter called GS-CV.

II. Process

The process, as was shortly described in the introduction, of determining the best classifier included hyperparameter tuning using GS-CV. Although exhaustive, GS-CV is a powerful tool for reporting on the accuracy of different models with different hyperparameters to find the best set of hyperparameters that result in the best accuracy. Before the process of GS-CV was implemented, some preprocessing was done on the data.

Here to the right is a heatmap which shows the correlation between all variables in the dataset. From the heatmap we can see that the highest correlation is between the variables "density" and "residual sugar". We decided to drop one of those variables entirely but the decision of which one came down to the correlation with the output variable. Note that the correlation between residual sugar and quality is incredibly weak while the



correlation between density and quality is much stronger than for residual sugar. For that reason, the residual sugar column got dropped. As we were going to use K-nearest neighbors Classifier we used the StandardScaler to standardize the data to remove mean and scaling the data to unit variance.

Additionally, before the hyperparameter tuning ensued, all models were trained on the data with no hyperparameters set. These accuracy scores were stored in a dictionary along with the best accuracy score after the hyperparameter tuning of each model. Furthermore, each reported set of the best hyperparameters for each model was stored in a separate dictionary. This is to give a good overview of the results.

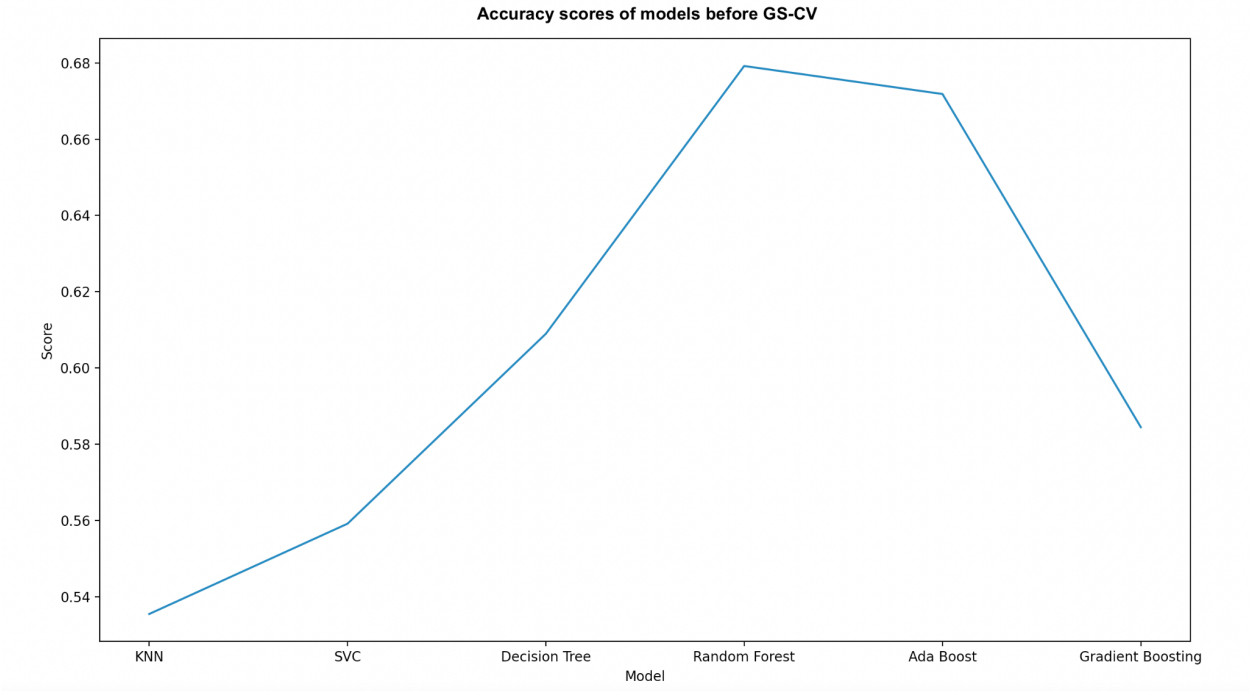
See Appendix 1 for tables on each set of hyperparameters that was used in the GS-CV for each different model. These specific hyperparameters were chosen after looking at each classifier on their respective documentation and looking at examples of which parameters were most often used. The different values for the parameters were chosen in the same way. From the lectures and examples in the documentation, we could find out the purpose of each hyperparameter which further helped our decision process.

The classifiers, (K-nearest neighbor, Support Vector Classifier, Decision Tree Classifier, Random Forest Classifier, AdaBoost Classifier and Gradient Boost Classifier), that were chosen were classifiers that had been gone over in lectures, been gone over extensively in assignments or were recommended to us by a teacher and that we felt most confident in being able to explain and use to their fullest extent

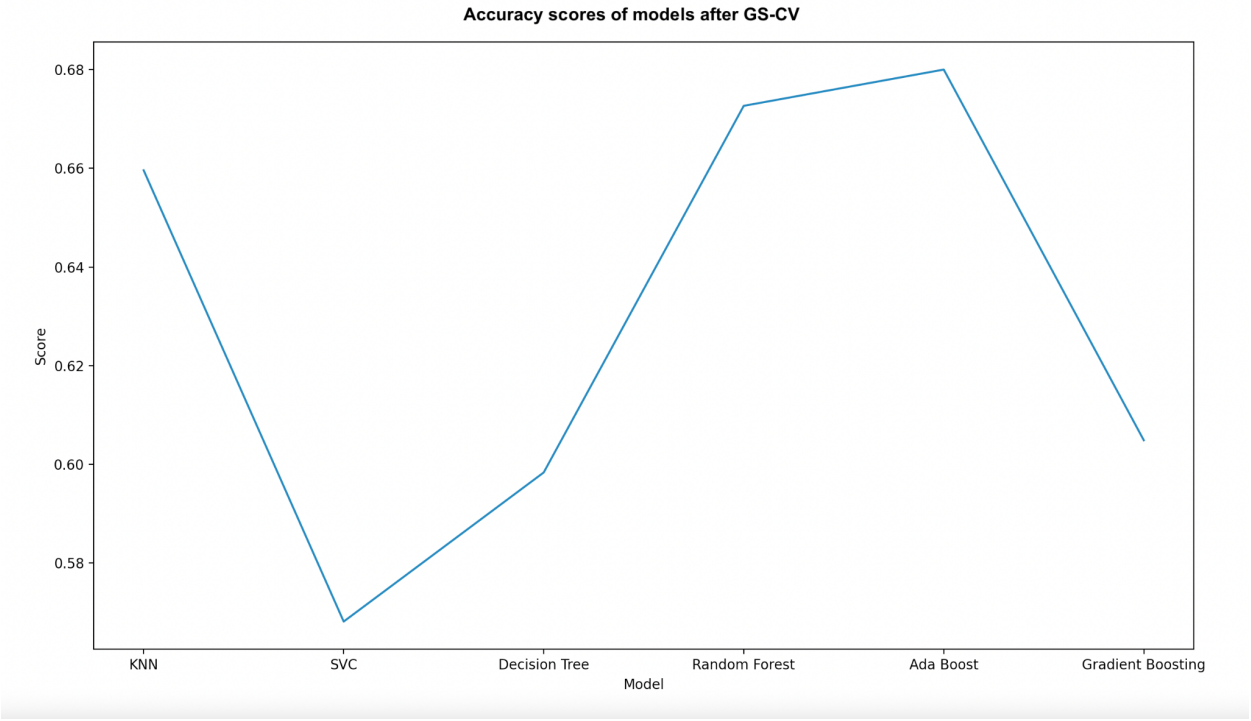
III. Results

First, let's look at how each model performed before conducting the GS-CV to find the best set of hyperparameters. We can see that the random forest classifier has the best accuracy score before the GS-CV, with a score of just under 68%. The weakest classifier is the K-nearest neighbors with an accuracy score of just under 54%. Below we can see this information in a table and on a line plot.

Classifier	Accuracy before GS-CV	Accuracy after GS-CV
K-nearest neighbors	53.55%	65.96%
Support Vector Classifier	55.92%	56.82%
Decision Tree	60.90%	59.84%
Random Forest Classifier	67.92%	67.65%
AdaBoost classifier	67.18%	68.00%
Gradient Boosting classifier	58.45%	60.49%



Now let's look at the accuracy scores of each model after the GS-CV. The two plots look fairly similar except now the best performing classifier is the AdaBoost with a score of 68%, and the K-nearest neighbors took a big jump in accuracy scoring, with a new accuracy score of 66%.



Below is a table containing each classifier and its best set of hyperparameters that the GS-CV found. The models tested were K-nearest neighbors, support vector classifier, decision tree, random forest classifier, AdaBoost classifier, and gradient boosting.

Classifier	**Hyperparameters**
K-nearest neighbors	algorithm: auto n_neighbors: 20 weights: distance
Support Vector Classifier	C: 10 gamma: scale kernel: rbf
Decision Tree	criterion: gini max_depth: None splitter: best
Random Forest Classifier	criterion: entropy max_depth: None n_estimators: 130

AdaBoost classifier	algorithm: SAMME learning_rate: 10 n_estimators: 40
Gradient Boosting classifier	loss: deviance max_features: auto n_estimators: 250

IV. Conclusions

As the results show, without any hyperparameter tuning Random Forest has the best accuracy score but after the hyperparameters have been tuned, AdaBoost gets better and would be the preferred choice of a classifier, also factoring computation time. AdaBoost being the best classifier can be explained because the base estimator for it was the Random Forest with its best set of hyperparameters, so it builds a new classifier on top of a good one. Note that the dataset is very small and the Random Forest classifier tries to use bagging to help which could be the reason the accuracy lowered after the hyperparameters were set. The big jump in accuracy for K-nearest neighbors is a perfect example of how the correct hyperparameters can change the accuracy of a model to the better, but also how dangerous they can be as seen with the Gradient Boost Classifier, where the hyperparameters lowered the accuracy by 0.2%. Unsurprisingly, the Decision Tree classifier performed worse than Random Forest as Random Forest was built upon the Decision Tree, much like how the AdaBoost was built upon the Random Forest. When also factoring computation time into the mix, which for some instances reached tens of minutes, the performance of the Support Vector Classifier is much worse than was before. This also seemed to be the case with the Gradient Boost Classifier.

V. Future Work

For more extensive work on this particular project some other methods may be taken in classifying the data. For example, the dataset could be looked at with feature selection to determine which physicochemical attributes of the wine affect the quality more. If more GS-CV research were to be done, the accuracy for each value of each parameter could be plotted and compared to better find the best set of hyperparameters. The GS-CV could even be performed multiple times with the range of the numerical hyperparameters being shortened in each iteration but that would require an immense amount of computation time. Also as Gradient Boost

Classifier is really dependent on the hyperparameters that are given and the accuracy varies wildly because of that it would be better to remove it from future work. Furthermore, the jump in accuracy the K-nearest neighbors classifier had compared to the other models was really interesting and could be worth looking further into. Maybe, with future research, such as finer tuning of the hyperparameters, that model could end up being the best, knocking AdaBoost off its throne.

VI. References

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.

Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Appendix 1: Tables of chosen hyperparameters for different models

K-Nearest Neighbors classifier

Parameter	**Values**
n_neighbors	1, 2, 5, 10, 20
algorithm	Auto, ball_tree, kd_tree, brute
weights	Uniform, distance

Support Vector Classifier

Parameters	**Values**
C	0.1, 1, 10, 100
gamma	scale, auto
kernel	linear, rbf

Decision tree classifier

Parameters	**Values**
criterion	gini, entropy
splitter	best, random
max_depth	4, 5, 7, 9, 10, 11, None

Random forest classifier

Parameters	**Values**
n_estimators	70, 80, 100, 130, 150
criterion	gini, entropy
max_depth	4, 5, 7, 9, 10, 11, None

AdaBoost Classifier

Parameters	**Values**
n_estimators	40, 50, 60, 65, 70, 80, 100
learning_rate	0.01, 0.1, 0.05, 0.5, 1, 10
algorithm	SAMME, SAMME.R

Gradient Boost classifier

Parameters	**Values**
loss	deviance, exponential
learning_rate	0.01, 0.2, 0.3, 0.1, 0.05, 0.5, 1
n_estimators	100, 200, 220, 230, 240, 250
max_features	auto, sqrt, log2