**Quality of Wine Predictions Using Classification**

# Wine Quality Classification Dataset

Morgan Hardin, [mhardin5@bellarmine.edu](mailto:mhardin5@bellarmine.edu)

**ABSTRACT**

This project uses a wine dataset and tries to determine if the predicted outcome is either good quality wine or bad quality. This can be determined through using classification techniques to train the model and determine which technique is the most accurate. The highest accuracy score will determine which model is the most accurate in predicting the quality of the wine. The mean squared error determines the error percentage of the model and which one has the lowest error rate. Overall, this project analyzes the dataset and tries to predict the quality of future wine data.

1. **INTRODUCTION**

A wine dataset was downloaded from Kaggle and analyzed, determining if the quality of wine is good or bad based on certain characteristics that impact its taste, smell, and texture. Based on the info section of the columns from Kaggle, if the quality of wine is bad, then it will be zero and if the quality of wine is good, then it will be one. This is used for when the good and bad categorical data was transformed into numerical data. The model was trained and tested with four classifiers: logistic regression, random forest, decision tree, and support vector classifier. A voting classifier is used to find the most accurate of the four classifiers for training the model. The goal is to find which has the most accurate score for this model in its predictions of the quality of wine.

1. **BACKGROUND**
   1. *Data Set Description*

The wine dataset was downloaded from Kaggle and was chosen since it was designed specifically to use and learn classification techniques. The data was collected by Naresha Bhat and was made with classification modeling in mind. Bhat got the data from the UCI Machine Learning Repository already cleaned up and ready for classification. The dataset was chosen because it was created for classification purposes.

* 1. *Machine Learning Model*

The machine learning techniques that were used were logistic regression, decision tree, random forest, and support vector classification (SVC). A voting classifier is used to find the averaged, highest accuracy score for the model.

* Logistic regression classifies data as being above or below 0.5. It separates zeros and ones into the two categories using the logistic function:

sig(x) = 1 / 1+e-X

This gives an output between 0 and 1, classifying the data.

* Decision trees work as a binary tree structure and separate from the root node into 2 leaves and continues branching with two child nodes to a parent node. Entropy was the type of criterion chosen since it gave the best results, meaning the information gain is impacted based on the formula:

IENTROPY = jΣi = 1 (pi \* log2 \* pi)

This is important because the purpose is to maximize the information gain and, in this model, this equation and formula maximizes the information gain.

* Random forest works with decision trees since it takes a random subset and tries to find the best features in that specific path. Although it can also be used with bagging, it is used with decision trees in this project to add a randomness to the decision tree to help control the model and help it perform better.
* Support vector machine (SVC) tries to find a hyperplane that clearly distinguishes two different classification groups. The margin should be maximized to achieve the best accuracy in separating the two groups.
* Voting classifier takes various implemented classifiers and determines the highest accuracy score. It can either taken on `hard` voting or `soft` voting. In this case, `soft` voting gave a higher accuracy score. This means that the probabilities of each classifier are taken and given a weight, then are averaged together to get the final accuracy score of all the classifiers combined.

These are four of the main classification techniques that can be used to improve certain aspects of the dataset. They are put into the voting classifier to be averaged together to give an averaged accuracy score and determine which classifier gives the highest accuracy score. Overall, the main objective for this classification problem is to maximize the functions and find the average scores in order to reach the highest possible accuracy score. The goal for these techniques is to determine which has the highest score for this particular wine dataset.

1. **EXPLORATORY ANALYSIS**

The wine dataset has a total of 12 columns, each with 1599 non-null entries. Of the twelve columns, all were numerical data (float64) except the quality column which held categorical data. This was easily encoded to numerical data with the ordinal encoder since there was order to the categorical data, with only bad and good entries. There was no missing data and the histograms show that there was mostly a normal distribution across all the twelve columns. The `total sulfur dioxide’ column was the only one that had a higher range than the rest, with the max being around 300 units while the others hardly broke 100. For the quality, which is the main outcome that will be predicted, the values were not skewed so it does not have to be stratified when `train\_test\_split` is used. All the x variables (all columns excluding `quality`) will need to be scaled since SVC is used. It can be scaled early after splitting and training the dataset since the scaling will not impact the other classification techniques but is needed for the support vector classification. Ultimately, this dataset seemed to have mostly been cleaned up except for a few small things like encoding the `quality` column and scaling the data. With this being done, the dataset can be moved on to the classification and prediction section.

**Table 1: Data Types**

|  |  |
| --- | --- |
| *Variable Name* | *Data Type* |
| wine | Tabular Data (Holds Strings and Integers) |
| `fixed acidity`, `volatile acidity`, `citric acid`, `residual sugar`, `chlorides`, `free sulfur dioxide`, `total sulfur dioxide`, `density`, `pH`, `sulphates`, and `alcohol` | Floating Point Integers (float64) |
| `quality` | Started: Categorical (object / string)  Encoded to: Floating Point Integers (float64) |
| `x`, `y`, `x\_train`, `x\_test`, `y\_train`, and `y\_test` | Tabular Data (Holds Strings and Integers) |
| `encoder` | Object of: OrdinalEncoder() |
| `sc\_x` | Object of: StandardScaler() |
| `log\_clf` | Object of: LogisticRegression() |
| `rnd\_clf` | Object of: RandomForestClassifier() |
| `dt\_clf` | Object of: DecisionTreeClassifier() |
| `svm\_clf` | Object of: SVC() |
| `voting\_clf` | Object of: VotingClassifier() |
| `y\_pred` | Array of Integers |
| `RMSE` | Floating Point Integer (float64) |
| `y\_pred\_voting` | Array of Integers |

1. **METHODS**
   1. *Data Preparation*

This dataset did not need any columns to be dropped since each one was very important in impacting the quality of the wine (good or bad quality). With this being said, the x values (all columns excluding the `quality` column) needed to be scaled since it is a classification problem and SVC is one of the techniques used. This is because it makes all of the x data scaled so it is more accurate and allows the model to perform better. It also does not impact the other three techniques outcomes. This means that even if the data was not scaled, the outcome will still be the same for logistic regression, decision trees, and random forest. It impacts SVC for the better and has better results with the x values being scaled, which is why it is implemented and used in this model.

* 1. *Experimental Design*

Three experiments were done, each having different parameters for the train, test, and split section. For experiment 1, the sets were split 80% into the training set and 20% into the testing set. This experiment gave decent results, but it definitely could have been better. It resulted in around a 77% accuracy and a mean squared error of about 48%. For experiment 2, the sets were split 85% into the training set and 15% into the testing set. This experiment gave the best results with around an 80% accuracy and a mean squared error of about 45%. Finally, experiment 3 was split 70% into the training set and 30% into the testing set. This gave the worst results of around a 76% accuracy and a mean squared error of about 48%. I also found that the parameters for logistic regression (solver = `lbfgs`, multi\_class = `multinomial`, max\_iter = 1000), random forest (n\_estimators = 1000, max\_depth = 10), decision tree (criterion = ‘entropy’, max\_depth = 10), random\_state = 0), and support vector classifier (gamma = `auto`, probability = True) were the highest across the board for each of the three experiments. The other possibilities were roughly the same, but were off about 1-2%, so these parameters gave the highest accuracy score / lowest mean squared error percentage. Overall, these parameters for the classifiers did not impact the scores as much as changing the training and testing set sizes, which is why that these parameter changes were not included in the experiments.

Table X: Experiment Parameters

|  |  |
| --- | --- |
| **Experiment Number** | **Parameters** |
| 1 | All four features were split into 80 / 20 for the training and testing sets   * Gave Decent Results |
| 2 | All four features were split into 85 / 15 for the training and testing sets   * Gave Best Results |
| 3 | All four features were split into 70 / 30 for the training and testing sets   * Gave Worst Results |

* 1. *Tools Used*

The following tools were used for this analysis: Python running through the Google Colab environment on an Apple Macintosh computer, all for analysis and implementation. In addition to base Python, the following libraries were also used: pandas, numpy, matplotlib.pyplot, seaborn, and sklearn. These tools were chosen and used since they are excellent for exploratory analysis of datasets and allow the users to manipulate them to predict and classify data.

1. **RESULTS**
   1. *Mean Squared Error Calculation / Confusion Matrix / Classification Report / ROC and AUC*

The mean squared error calculation for each classifier was put in the for loop alongside the accuracy scores. For logistic regression, the mean squared error was 0.51 which is not great. For random forest, the mean squared error was the best with 0.44. For decision tree, the mean squared error was 0.49. Finally, for SVC, the mean squared error was 0.49. By taking the average of the weighted classifiers, the mean squared error for the soft voting classifier was 0.44. This means that there is an error percentage of about 44% which is not great, but still could be worse.

The confusion matrix was created and printed based on the actual y values and the predicted y values. There were 87 true positives, 25 false positives, 23 false negatives, and 109 true negatives. This means that the model was correct, or true, for 196 values. It also means that the model was incorrect, or false, for 48 of the values. This shows that the model was sometimes incorrect and could be trained better to be more accurate to get more true values.

The classification report shows that the model did better predicting good quality wines than it did bad quality wines. This is seen through the precision scores, recall scores, and f1-scores. The precision is measured by all the positives in the model while the recall is the model’s ability to define a true positive. The f1-score finds the mean of these two scores. Overall, this classification report shows that this model did a better job of labeling good values than it did with bad values. With this being said, the total accuracy score for these was 80%, meaning the scores were pretty good, but still could be better.

The ROC curve was plotted showing the false positive rate compared to the true positive rate. This graph shows that the model performed well enough since these values fell between 0.5 and 1 (on the closer side of 1). This is good since it means that the model was trained correctly. To make the model even better, the value for true positive should be 1 and false positive should be 0. The AUC score was also calculated, being at about 80%, matching the other results from before and showing that the model is accurate, but still could perform better with further training.

* 1. *Discussion of Results*

The model that performed the best was the random forest classifier. It did the best since it took the decision tree classifier and added a randomness to it the allows it to perform better and be more accurate. The worst model was logistic regression. This is probably because the classifier is simpler and works better for easily separated data. Not all data is perfectly separated into two categories to use logistic regression, so random forest works better since it takes only some of the data (randomly) and branches out in a controlled manner. This is a much better algorithm for efficiency since it is quicker and does not need to process all the training data. The voting classifier also states that the random forest classifier is the best model since its accuracy score lines up with the random forest score, meaning it had the highest weight of all four classifiers.

* 1. *Problems Encountered*

It was very difficult to find a dataset that was interesting and had a classification problem. This dataset turned out to be pretty good but was not big enough to give a higher accuracy score. The scores were also difficult since changing the parameters in the classifiers did not seem to do much to the score. Also, the voting classifier worked well but originally took a very long time to execute during the beginning of the project. Finding out which parameters worked the best and gave the highest accuracy score was a pretty big problem that was able to be figured out.

* 1. *Limitations of Implementation*

This dataset is not big enough to get a higher accuracy score so it has a limited score with the classification techniques. It also has limitations with the parameters with the classifiers since they did not seem to change the score much. This is probably once again due to the data set being fairly small. Although it is not tiny, it is still not big enough for a higher accuracy score. Classification is what the dataset was intended for, so other classification techniques that are more complex or geared toward somewhat smaller datasets might work better than these models.

* 1. *Improvements/Future Work*

Making the dataset bigger by gathering more data and trying more classification techniques would be a good place to start in improving the model. More expierments would also be helpful or having a different dataset altogether might work as well.

1. **CONCLUSION**

This project was a classification project that intended to predict the quality of wine being good or bad based on the wine dataset downloaded from Kaggle. Four classification techniques were used: Logistic Regression, Decision Tree, Random Forest, and SVC. These classifiers were all put into a voting classifier to be weighted to determine the highest scoring model. The random forest classifier turned out to be the best fit for the model with an 80% accuracy as seen through the various scores reports in the project. Ultimately, the model turned out to be decent, but needs more work. With only an 80% accuracy, it has a lot of room to perform better, but still is decent enough for the problem at hand.

**REFERENCES**

Wine Dataset:

<https://www.kaggle.com/datasets/nareshbhat/wine-quality-binary-classification>

Machine Learning Textbook:

<https://platform.virdocs.com/r/s/0/doc/591190/sp/197640749/mi/611682288>

SKLearn Logistic Regression:

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

SKLearn Decision Tree Classifier:

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

SKLearn Random Forest Classifier:

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

SKLearn Support Vector Machine (SVC):

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

SKLearn Voting Classifier:

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html>