Sheetal Sattiraju
13 December 2022

Wine Dataset Analysis

Introduction

The wine dataset, aka 'load_wine' from scikit-learn, is a popular dataset for classification machine learning. This particular dataset contains three different classes, class 0, 1 and 2, with 178 samples of wine and 13 distinct features. These features are based on various chemical properties of different types of wine. The feature type of this dataset is positive and real. The properties are as follows: Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines and Proline. The goal of this final is to make use of the classifier and model techniques I have learned from this semester and apply it to this numpy array dataset. In this paper, I will be attempting to get the highest accuracy possible, using 5 different models: SVM, RF, DT, MLP and GNB.

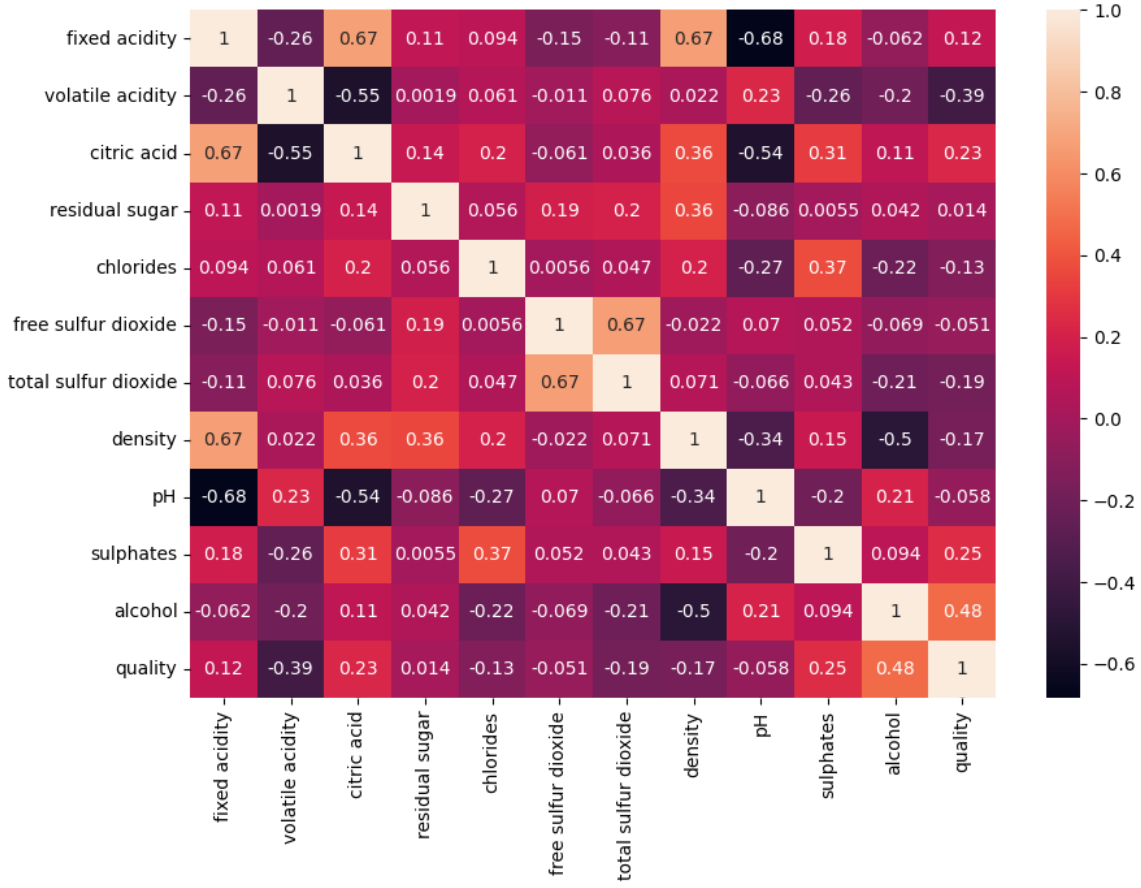


Figure 1

Here, we can note the relationships between all of the 13 attributes of the dataset and how they compare to each other. We can see how certain features have very distinct relationships, such as fixed acidity and density or ph and citric acid.

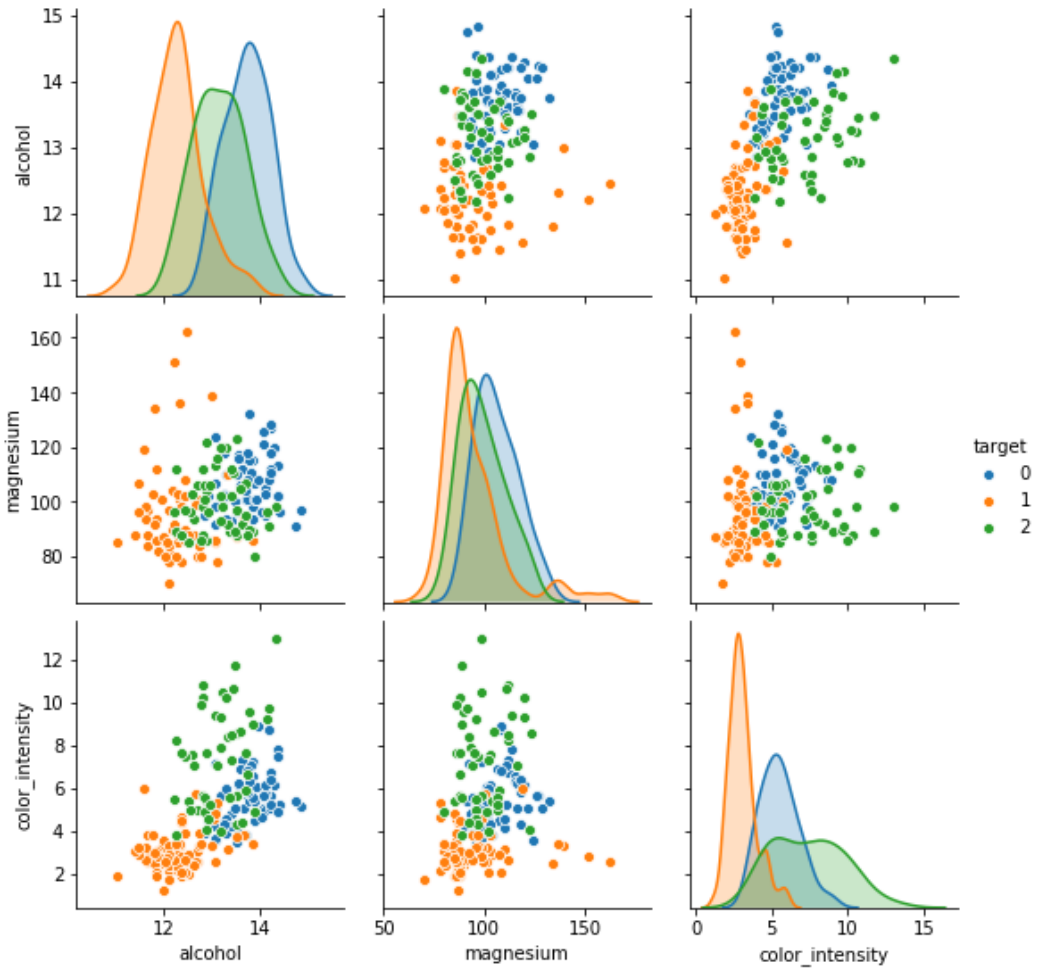


Figure 2

In this figure, we can see that some features in this data are more easily distinguishable, like color intensity vs alcohol in this figure. However, we can also note that class 1 and 2 are not linearly separable in magnesium vs color intensity especially and magnesium vs alcohol. This figure showcases a couple instances (magnesium vs color intensity and magnesium vs alcohol) where the model may struggle to correctly identify the appropriate class.

Experiments

I. SVM

SVM, or Support Vector Machine, is a type of supervised learning model that can be used for classification tasks. This classifier makes use of a generalized maximal margin classifier, but with a soft margin instead. This classifier is effective for the load_wine dataset, due to the high amount of features to compare. Additionally, we cannot separate the three classes: class 0, 1 and 2 linearly, requiring a versatile classifier.

<u>The accuracy rate lies around: 94.44% to 100%.</u>

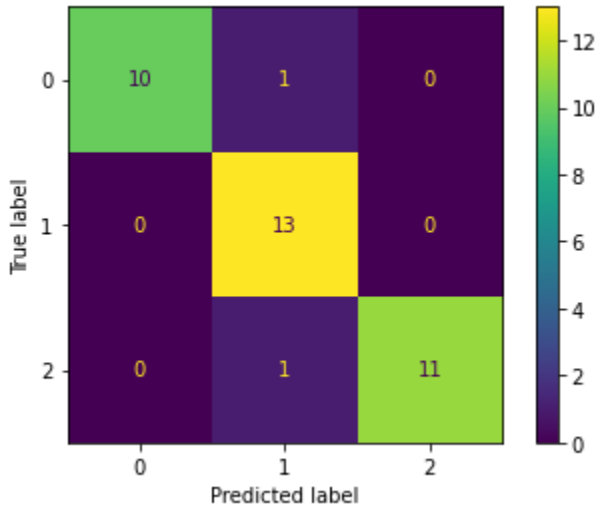
For this project, I preprocessed the dataset using StandardScaler. Afterwards, I trained the model using the SVC classifier. See SVM on Wine Dataset for the code. I also ran cross validation on this classifier, which will be shown at the end of this report, in the analysis section.

These were the results of the SVC after several iterations:

SVM Run 1:

Accuracy: 0.9444444444444444

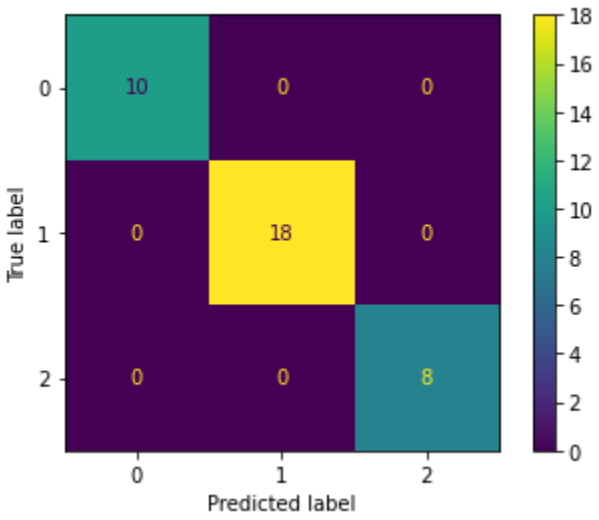
	precision	recall	f1-score	support
0	1.00	0.91	0.95	11
1	0.87	1.00	0.93	13
2	1.00	0.92	0.96	12
accuracy			0.94	36
macro avg	0.96	0.94	0.95	36
weighted avg	0.95	0.94	0.95	36



SVM Run 2:

Accuracy: 1.0

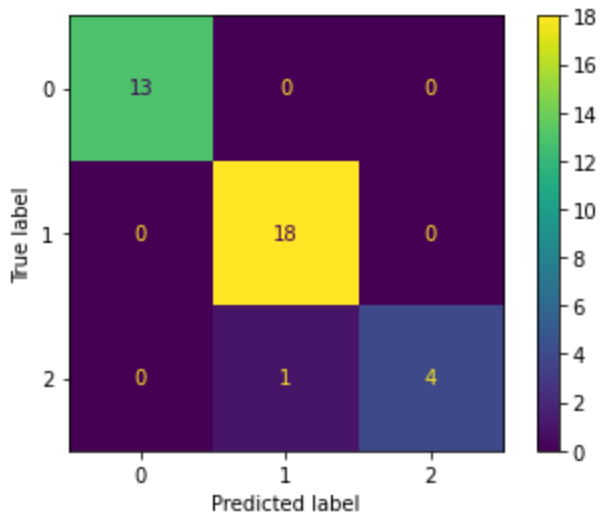
	precision	recall	f1-score	support
0	1.00	1.00	1.00	16
1	1.00	1.00	1.00	14
2	1.00	1.00	1.00	6
accuracy			1.00	36
macro avg	1.00	1.00	1.00	36
weighted avg	1.00	1.00	1.00	36



SVM Run 3:

Accuracy: 0.9722222222222222

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.95	1.00	0.97	18
2	1.00	0.80	0.89	5
accuracy			0.97	36
macro avg	0.98	0.93	0.95	36
weighted avg	0.97	0.97	0.97	36



Brief Analysis:

We can note that SVM is highly accurate in separating the classes appropriately. However, we can see that it struggled with a couple of samples in recognizing class 1, as expected. We saw earlier how a few samples in this class overlap with the other classes for specific features, possibly magnesium or total acidity levels.

SVM GridSearch:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16
1	1.00	0.93	0.96	14
2	0.86	1.00	0.92	6
accuracy			0.97	36
macro avg	0.95	0.98	0.96	36
weighted avg	0.98	0.97	0.97	36

```
Out[11]: 0.9722222222222222
```

Out of curiosity, I ran a Gridsearch on the SVM model, which brought up the accuracy rate to a max of 97.22% for all runs. It either brought up the accuracy slightly, or kept it the same.

II. RF

Random Forest Classifier is a supervised algorithm used for classification tasks. This particular classifier makes use of the average decision from a large number of decision trees. The predictive nature of this algorithm allows for accurate classification, while the randomness prevents overfitting of the data, which is key to the load_wine dataset. As we saw from Figure 2, we need to be able to accurately predict Class 1 vs 2 to get a good accuracy rate.

The accuracy rate lies around: 97.77% to 100%

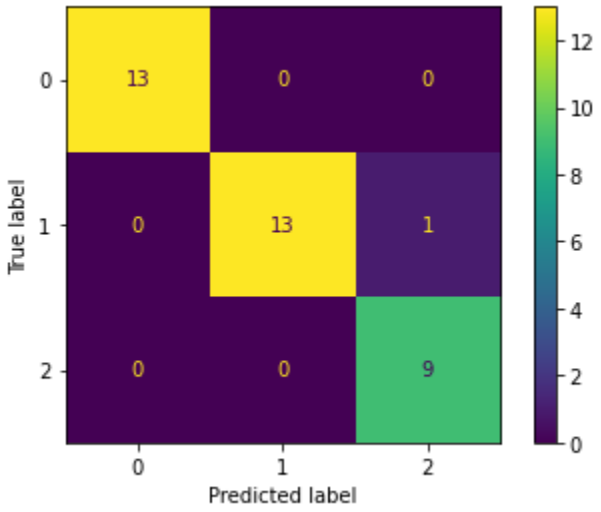
For this project, I preprocessed the dataset using StandardScaler. Afterwards, I trained the model using the RF classifier. See RF on Wine Dataset for the code. I also ran cross validation on this classifier, which will be shown at the end of this report, in the analysis section.

These were the results of the RFC after several iterations:

RF Run 1:

Accuracy: 0.9722222222222222

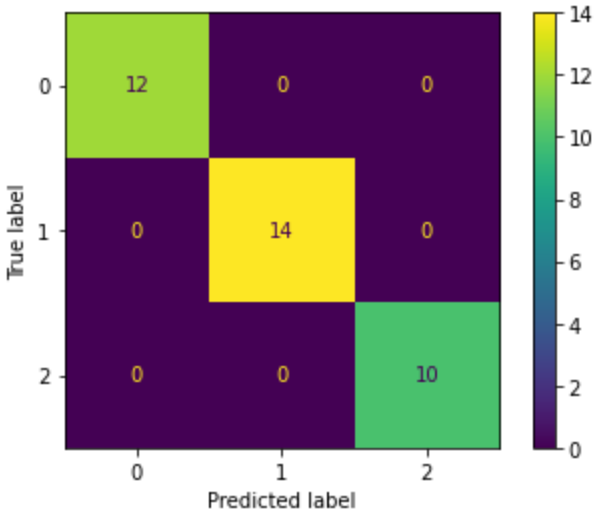
	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	0.93	0.96	14
2	0.90	1.00	0.95	9
accuracy			0.97	36
macro avg	0.97	0.98	0.97	36
weighted avg	0.98	0.97	0.97	36



RF Run 2:

Accuracy: 1.0

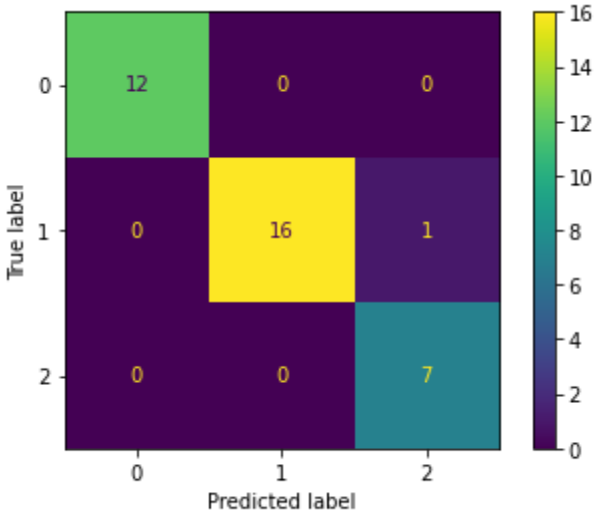
	precision	recall	f1-score	support
0	1.00	1.00	1.00	12
1	1.00	1.00	1.00	14
2	1.00	1.00	1.00	10
accuracy			1.00	36
macro avg	1.00	1.00	1.00	36
weighted avg	1.00	1.00	1.00	36



RF Run 3:

Accuracy: 0.9722222222222222

	precision	recall	f1-score	support
0	1.00	1.00	1.00	12
1	1.00	0.94	0.97	17
2	0.88	1.00	0.93	7
accuracy			0.97	36
macro avg	0.96	0.98	0.97	36
weighted avg	0.98	0.97	0.97	36



Brief Analysis:

We can see that RandomForestClassifier, once again, had a pretty high accuracy rate, slightly better than SVM. This model seemed to struggle with a sample or two from class 2, while SVM

had an issue with classifying class 1 accurately. RandomForest, from these experiments, is the better model in classifying this particular dataset.

RF GridSearch

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	1.00	1.00	14
2	1.00	1.00	1.00	9
accuracy			1.00	36
macro avg	1.00	1.00	1.00	36
weighted avg	1.00	1.00	1.00	36

```
Out[3]: 1.0
```

```
Out[15]: 0.9722222222222222
```

Out of curiosity, I ran a Gridsearch on the RF model, which brought up the accuracy rate to a max of 100% or 97.22% for all runs. Once again, it either brought up the accuracy slightly, or kept it the same.

III. Decision Trees

Decision Tree Classifier is a similar type of classification model like RandomForest, but only combines some decisions as opposed to several or an average of them. This model is not as highly accurate as RandomForest on some datasets but is very fast. I doubt this model will have a great accuracy rate, but I chose to test it to see how viable it can be on a dataset of this size, due to its speedy nature.

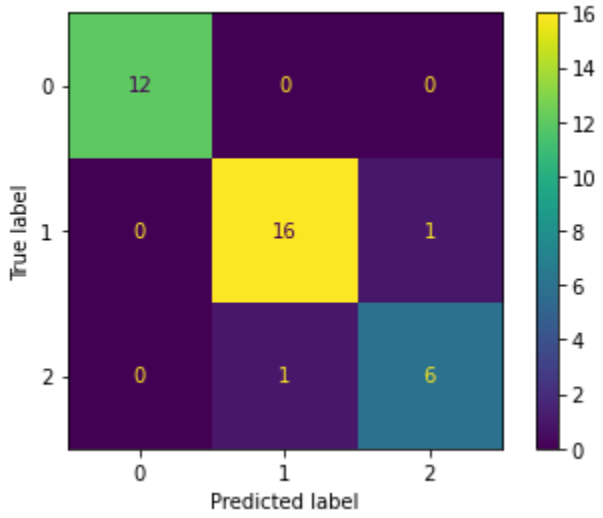
The accuracy rate lies around: 83.33% to 94.44%

For this project, I preprocessed the dataset using StandardScaler. Afterwards, I trained the model using the decision tree classifier. See Decision Trees on Wine Dataset for the code. I also ran cross validation on this classifier, which will be shown at the end of this report, in the analysis section.

These were the results of the DTC after several iterations:

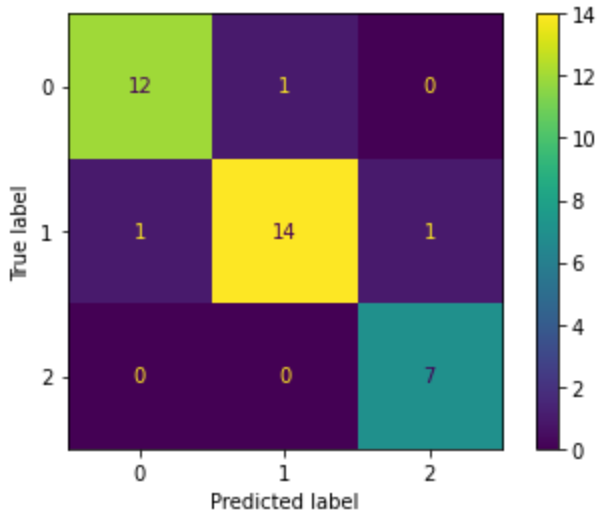
DT Run 1:

```
Accuracy: 0.9444444444444444
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.94      0.94      0.94        17
           2       0.86      0.86      0.86         7

    accuracy                           0.94        36
   macro avg       0.93      0.93      0.93        36
weighted avg       0.94      0.94      0.94        36
```



DT Run 2:

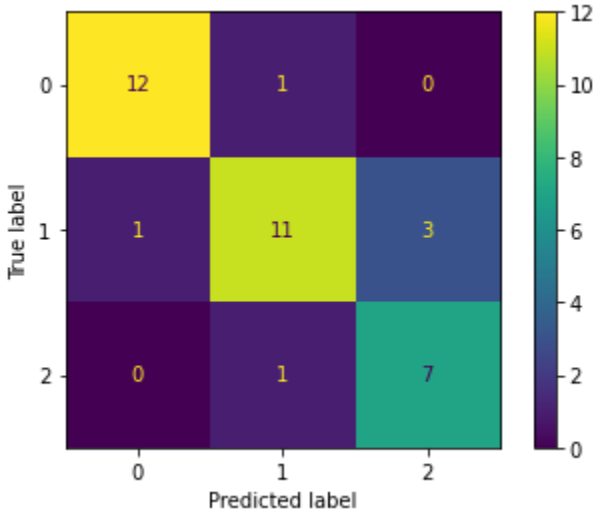
```
Accuracy: 0.9166666666666666
              precision    recall  f1-score   support

           0       0.92      0.92      0.92        13
           1       0.93      0.88      0.90        16
           2       0.88      1.00      0.93         7

    accuracy                           0.92        36
   macro avg       0.91      0.93      0.92        36
weighted avg       0.92      0.92      0.92        36
```



DT Run 3:

Accuracy: 0.8333333333333334

	precision	recall	f1-score	support
0	0.92	0.92	0.92	13
1	0.85	0.73	0.79	15
2	0.70	0.88	0.78	8
accuracy			0.83	36
macro avg	0.82	0.84	0.83	36
weighted avg	0.84	0.83	0.83	36



Brief Analysis:

As predicted, the model was not that highly accurate for this dataset, but was a very fast model. This model had issues for all three classes and struggled with the overlapping data points. RandomForest still stands as the best classifier model for this dataset.

IV. MLP

MLP, aka Multilayer perceptron, is a neural network deep learning algorithm, in which the mapping between inputs and outputs is non-linear. MLP makes use of hidden layers and takes a set of inputs to provide an output using an activation function. This deep neural network model has a decent chance of being a highly accurate model for this type of dataset.

The accuracy rate lies around: 94.44% to 100%

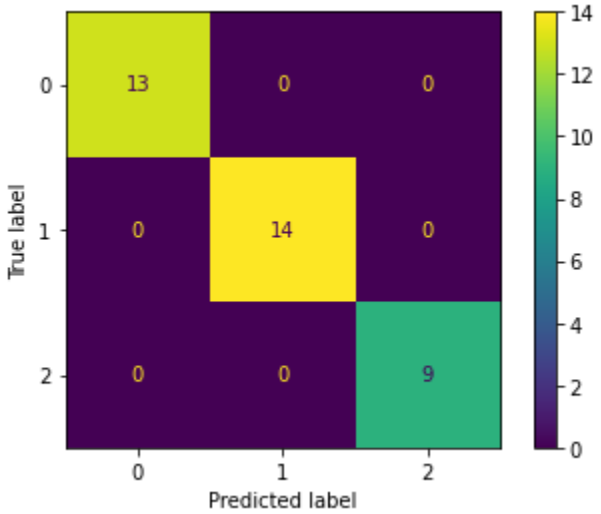
For this project, I preprocessed the dataset using StandardScaler. Afterwards, I trained the model using MLP. See MLP on Wine Dataset for the code. I also ran cross validation on this classifier, which will be shown at the end of this report, in the analysis section.

These were the results of the MLP model after several iterations:

MLP Run 1:

Accuracy: 1.0

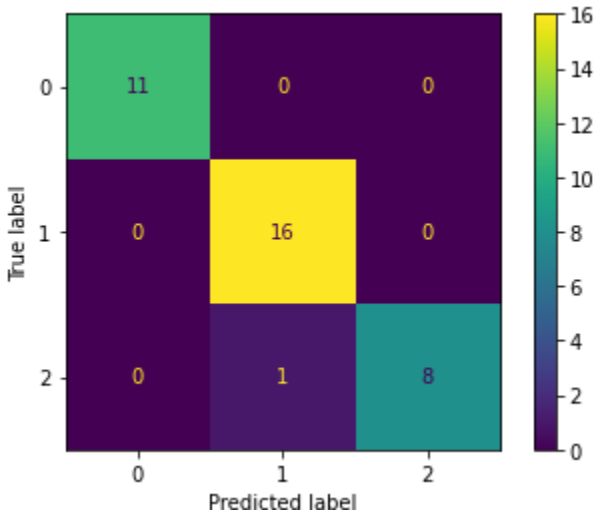
	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	1.00	1.00	14
2	1.00	1.00	1.00	9
accuracy			1.00	36
macro avg	1.00	1.00	1.00	36
weighted avg	1.00	1.00	1.00	36



MLP Run 2:

Accuracy: 0.9722222222222222

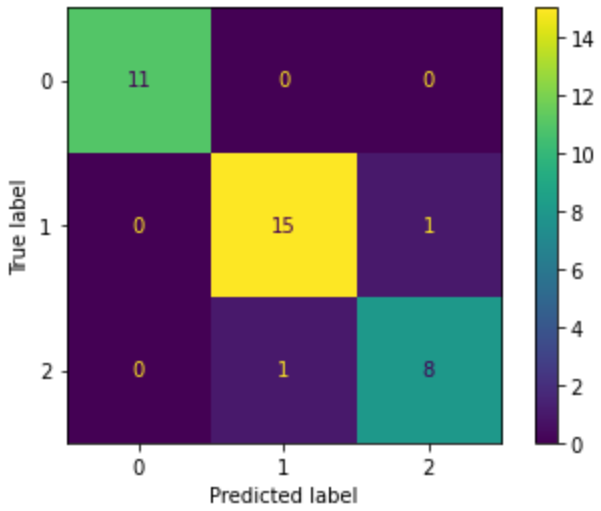
	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	0.94	1.00	0.97	16
2	1.00	0.89	0.94	9
accuracy			0.97	36
macro avg	0.98	0.96	0.97	36
weighted avg	0.97	0.97	0.97	36



MLP Run 3:

Accuracy: 0.9444444444444444

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	0.94	0.94	0.94	16
2	0.89	0.89	0.89	9
accuracy			0.94	36
macro avg	0.94	0.94	0.94	36
weighted avg	0.94	0.94	0.94	36



Brief Analysis:

MLP was highly accurate and fast, making it an equally good classification model to SVM and RandomForest. It had similar issues with a sample or two in Class 1 or 2, as expected for Run 2 and 3. Overall, this model's accuracy rates were really high and viable for these types of datasets.

V. GNB

GNB, or Gaussian Naive Bayes, is a supervised classifier, which makes use of a probabilistic algorithm based off of Bayes' Theorem. This algorithm makes use of Z-score, mean and standard deviation to fit the model, based on each datapoint. This algorithm works well for continuous and big datasets, which is what we need when trying to separate the three classes in the wine dataset. (I learned about this algorithm in my AI class and decided to test it)

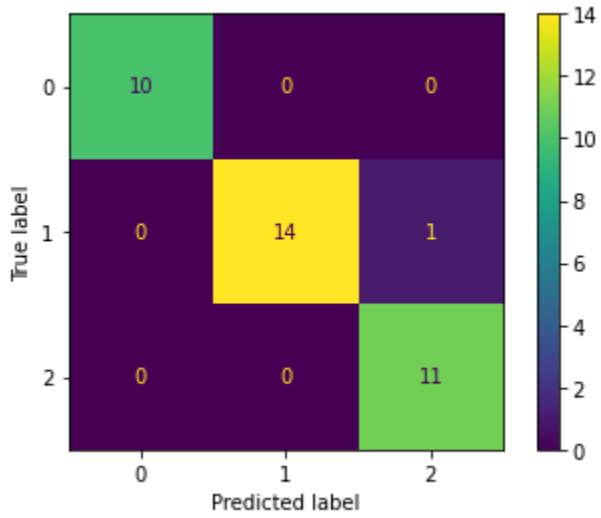
The accuracy rate lies around: 94.44% to 100%

For this project, I preprocessed the dataset using StandardScaler. Afterwards, I trained the model using GNB. See GNB on Wine Dataset for the code. I also ran cross validation on this classifier, which will be shown at the end of this report, in the analysis section.

These were the results of the GNBmodel after several iterations:

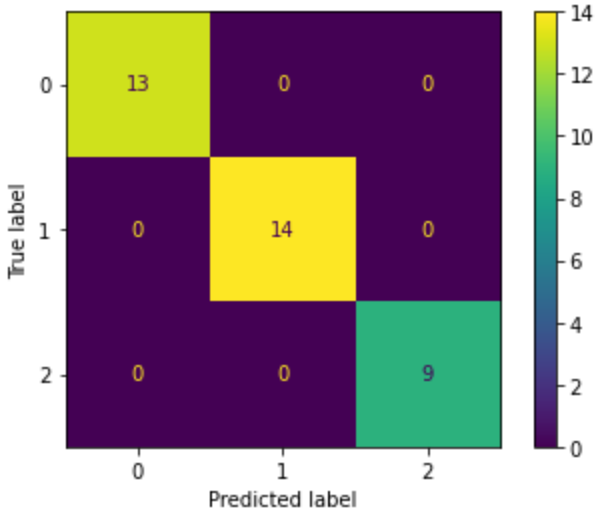
GNB Run 1:

```
Accuracy: 0.9722222222222222
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.93      0.97        15
           2       0.92      1.00      0.96        11

    accuracy                           0.97        36
   macro avg       0.97      0.98      0.97        36
weighted avg       0.97      0.97      0.97        36
```



GNB Run 2:

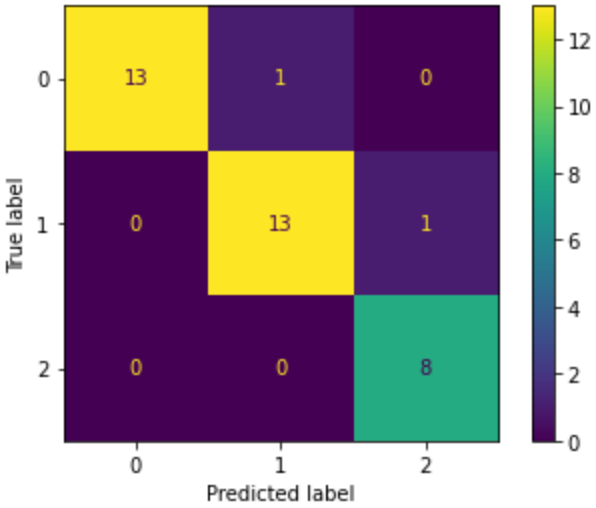
```
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         9

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```



GNB Run 3:

Accuracy: 0.9444444444444444

	precision	recall	f1-score	support
0	1.00	0.93	0.96	14
1	0.93	0.93	0.93	14
2	0.89	1.00	0.94	8
accuracy			0.94	36
macro avg	0.94	0.95	0.94	36
weighted avg	0.95	0.94	0.94	36



Brief Analysis:

As expected, the model did pretty well and scored similar high accuracy rates to SVM, RF and MLP, making it also a viable classifier for this dataset. Class 2 seemed to be the main issue for this model, but overall, performed well.

VI. Cross Validation Analysis

Run 1:

Models Used:	Accuracy Score (%)	Cross Validation Score (%)
SVM	**94.44%**	**98.571%**
RF	97.22%	96.467%
Decision Trees	94.44%	91.571%
MLP	100%	97.857%
GNB	**97.22%**	**97.238%**

Run 2:

Models Used:	Accuracy Score (%)	Cross Validation Score (%)
SVM	100%	97.19%
RF	100%	97.952%
Decision Trees	**91.66%**	**93.619%**

MLP	**97.22%**	**98.571%**
GNB	100%	96.524%

Run 3:

Models Used:	Accuracy Score (%)	Cross Validation Score (%)
SVM	**97.22%**	**99.286%**
RF	**97.22%**	**97.905%**
Decision Trees	**83.33%**	**95.81%**
MLP	**94.44%**	**98.571%**
GNB	**94.44%**	**100%**

After performing cross validation on all five models for 3 separate trials, we can see that Cross Validation helped in the bolded cases. The key trend we can pull from this is that if the model's accuracy rate happened to be slightly lower than usual, the cross validation tended to bring the accuracy up. When the model's accuracy score was close to the average accuracy rate of the model, the cross validation score tended to be slightly lower or higher than the model's accuracy. We can note that the cross validation did not always work, as seen in Table 1 and Table 2.

Analysis of Experiments

From these five experiments, we can see that four methods (SVM, RF, MLP and GNB) can be used to deliver a highly accurate model. Almost all the models have an accuracy rate above 94.44%, with the exception of the Decision Tree Classifier. RandomForest had the most consistent accuracy between 97.77% to 100%, but all of the accurate models were able to reach 100% in some test runs. SVM, MLP and GNB had almost identical results at 94.44-100%, while Decision Tree's accuracy lied around 83.33% - 94.44%. None of the models had an overfitting issue and seemed sustainable. I used GridSearch for SVM and RandomForest, and had results in which turning the hyperparameters either helped the model reach a slightly higher accuracy or no change at all. After cross validation, the accuracies were pretty high for all models, with the exception of a few cases. These few cases did not bring down the accuracy significantly.

We can see with all of the classifiers we have tested today, outside of the 100% accuracy cases,they all either mislabel class 1 or 2 of the load_wine dataset. None of these classifiers had trouble labeling the class 0, as predicted at the beginning of this paper. *Figure 1* showcased these varying relations between the 13 different features and the specifications of the alcohol. We can see from this figure why this dataset is difficult to separate linearly and requires the use of SVM, RF, MLP and GNB type classifiers. This also explains the lower accuracy rate of the DecisionTree model, which is more appropriate for linear models. *Figure 2* pinpoints a couple of these non-linear relationships; an example relation of this is magnesium vs color intensity: we

can see that this graph has overlapping data points, for Class 1 and 2, forcing us to use non-linear classification models for high accuracy and efficiency.

Conclusion and Future Works

In short, I experimented with SVM, RF, MLP and GNB due to their abilities to target non-linearly separable classes. I also experimented with DecisionTrees, due to its similarity to the RandomForest Classifier, even though it is meant for linear classes to see how much more inaccurate the classifier's accuracy would be. Four of these models are highly accurate and successful. In the future, I would likely try SVM with a different kernel (such as linear, polynomial) to see if that produces less or more accurate results. I would also try other optimizer techniques besides GridSearch and see if that would improve the results for this dataset. Perhaps, using a different preprocessing model than StandardScaler would be an interesting idea. Other possible models for testing would be CNN, NLP, other ensemble learning methods, and etc. As long as the model can separate non-linear classes, it should result in a high accuracy.

References:

- Dataset Description: https://scikit-learn.org/stable/datasets/toy_dataset.html#wine-dataset (For the introduction)
- Cross Validation: https://www.kaggle.com/code/dhruvalpatel30/wine-quality-prediction-with-cross-validation (For this snippet)

```
rfc_scores = cross_val_score(rfc, x_train, y_train, cv=10)
print(rfc_scores)
print('Mean Score =', rfc_scores.mean().round(5)*100,'%')
```


Figures:

1: https://www.kaggle.com/code/dhruvalpatel30/wine-quality-prediction-with-cross-validation
2: seaborn command