Machine Learning Engineer Nanodegree

Red Wine Quality

Ng Hai Ming

May 5th, 2019

I. Definition

Project Overview

Out of various human senses, taste itself is the least understood [1], complicated by the fact it can be a highly subjective evaluation exercise. There are even instances where changes in stated wine prices influenced the volunteers' perception of its tastes as well as the humans' brain regions associated with the experience of pleasure [2]. The ability to improve the objective identification of wine quality via supervised learning algorithms can facilitate the eventual stratification of various wine quality classes, and the concomitant provision of objective & highly visible benchmarks to set fair prices of wines against [1].

Data for the project, as attached with this project submission, can be downloaded from *https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009*. This dataset contains 1599 variants of the Portugese "Vinho Verde" wine, which includes 11 input physicochemical properties and 1 output sensory variable. Each input and output variables would be further elaborated and explored in the Data Exploration Section below.

Problem Statement

This project aims to use publicly available data from the UCI machine learning repository to classify the quality of red variants of Portuguese "Vinho Verde" wine based on input physicochemical properties. This is essentially an exercise for which objective numerical inputs are "regressed" against subjective binary quality outputs. Here, an arbitrary (sensory output variable) quality value of 7 would be utilised, above which the wine is classified as good quality and below which, poor quality.

To tackle this task at hand, the dataset would first be split into training and testing (validation) sets using sklearn's cross_validation train_test_split. Three different algorithms would be tried on the training dataset to identify the one best suited for classifying the problem. Once identified, tuning of the selected algorithm's hyperparameters and its relevant training would then take place. During training, the performance of the model would be evaluated using two key metrics, namely $F_{\beta=0.5}$ and accuracy score. When completed, the trained algorithm would then be used to predict the wine quality on the testing dataset. The objective of this exercise is to train this selected algorithm, i.e. AdaBoostClassifier (with DecisionTreeClassifier as the base estimator), to generalise well on the testing dataset and classify it with sufficiently high

accuracy and/or high $F_{\beta=0.5}$-scores. The primary target is to generate accuracy scores comparable with the benchmark set in the reference journal article [1].

Evaluation Metrics

Given this is a binary classification problem, this problem can be represented with a standard confusion matrix shown in **Figure 1a** below. When proposing evaluation metrics, the appropriate ones must be selected in order to get the desired model performance. After all, the model weights are fine-tuned off the model's performance on the selected metric for each training episode.

Figure 1a: Confusion Matrix

	Predicted: Not Good	Predicted: Good
Actual: Not Good	True Negatives (TN)	False Positives (FP)
Actual: Good	False Negatives (FN)	True Positives (TP)

There are several measures by which the model performances for supervised learning problems (since actual labels are provided) can be gauged, of which the three most common ones are:

a) Accuracy [3]:

The ratio of correctly predicted observations to total observations [4]

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

b) Precision [5]:

The ratio of correctly predicted observations to all positive predictions [4]. In other words, for all wines labelled as Good Quality, how many are actually Good Quality?

$$Precision = \frac{TP}{TP + FP}$$

c) Recall [5]:

The ratio of correctly predicted observations to all observations labelled as Good Quality [4]. In other words, for all wines that are Good Quality, how many wines are correctly labelled as Good Quality?

$$Recall = \frac{TP}{TP + FN}$$

Based on the above definitions, the following two evaluation metrics would be proposed and used here:

1) Accuracy:

Accuracy is a straightforward, simple and intuitive measure for model performance, since it's a direct measure of correctly predicted observations to total observations. However, it is a great measure only for symmetric datasets, where values of False Positives and False Negatives are almost the same [4]. The distribution of the dataset quality figures are as provided below in **Figure 1b**:

Figure 1b: Original Distribution

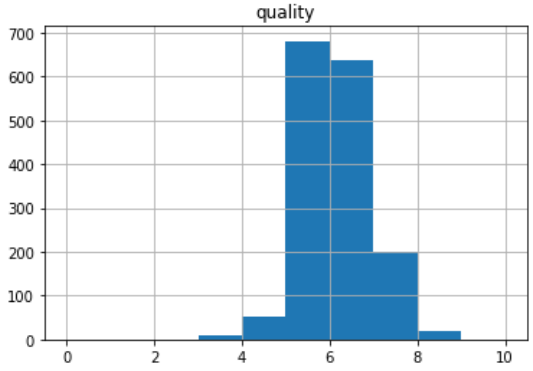
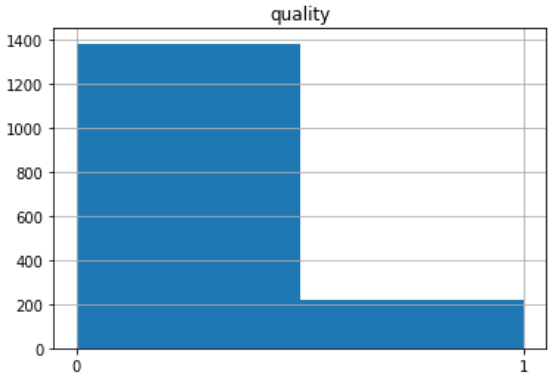


Figure 1c: Transformed Distribution



Given the dataset transformation to a binary classification problem (Quality >= 7 as Good Quality), the resulting dataset distribution shown in **Figure 1c** would become much less symmetric when compared to the original **Figure 1b**, and accuracy metrics would become less reliable for such a problem statement. This led to the eventual choice of $F_{\beta=0.5}$ as the primary evaluation metric for model training purposes. However, the use of accuracy metrics would still be retained for purposes of comparison against the highest benchmark of **89.0%** set in the reference journal article [1].

2) F-beta score - $F_{\beta=0.5}$ [6]:

F-beta score is the harmonic average of the precision and recall as per the equations below [6]. While not as intuitive to understand, it takes into account both FPs and FNs, and is especially useful for dataset with uneven class distribution like **Figure 1c** [4].

$$F_\beta = (1+\beta^2)\frac{Precision\ .\ Recall}{(\beta^2\ .\ Precision) + Recall}$$

$$F_{0.5} = (1+0.5^2)\frac{Precision\ .\ Recall}{(0.5^2\ .\ Precision) + Recall}$$

The choice of $F_{\beta=0.5}$, where recall are weighed lower than precision, would help attenuate the influence of false negatives [6]. This is particularly pertinent for the distribution shown in **Figure 1c**, where the possibility of False Negatives would likely outweigh the possibility of False Positives, given the skewed distribution (counts) towards the FN side (Not Good Quality).

II. Analysis

Data Exploration

The dataset consists of 1599 rows and 12 columns of data, of which 11 columns are input physicochemical properties and the remaining one being the output sensory variable. The first 5 rows of the dataset has been provided below:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

The 11 input physicochemical properties are [1, 7]:

1) Fixed Acidity: The content of Tartaric Acid, an acid which do not evaporate quickly
2) Volatile Acidity: The content of Acetic Acid, which at too high levels can lead to unpleasant and vinegar taste in wine
3) Citric Acid: The content of Citric Acid content, which can add “freshness” and “flavours” to wine
4) Residual Sugar: The amount of sugar remaining post-fermentation, where levels exceeding 45 g/L would be considered sweet
5) Chlorides: The amount of salt present in the wine
6) Free Sulfur Dioxide: Equilibrium concentration of free sulfur dioxide, which helps prevent microbial growth and wine oxidation
7) Total Sulfur Dioxide: Amount of free and bound forms of sulfur dioxide, which remains undetectable in wine in low concentrations. Above 50ppm, it becomes evident in the smell and taste of wine
8) Density: The density of the wine, which should be similar to that of water. The extent of deviations from water density would indicate the corresponding extent of alcohol and/or sugar content
9) pH: The level of the wine acidity and/or basicity, where low pHs (acidic) would give rise to a sour taste and high pHs (alkalinity) bitter taste
10) Sulphates: The content of wine additive, which contributes to SO_2 levels that contains antioxidant and antimicrobial properties discussed in *point 6*
11) Alcohol: The percentage of alcohol

The single output sensory variable consists of [1, 7]:

1) Quality: The sensory analysis performed by a minimum of 3 sensory assessors (human tasters) using blind tastes, based on their expert experience and knowledge, on a scale ranging from 0 to 10

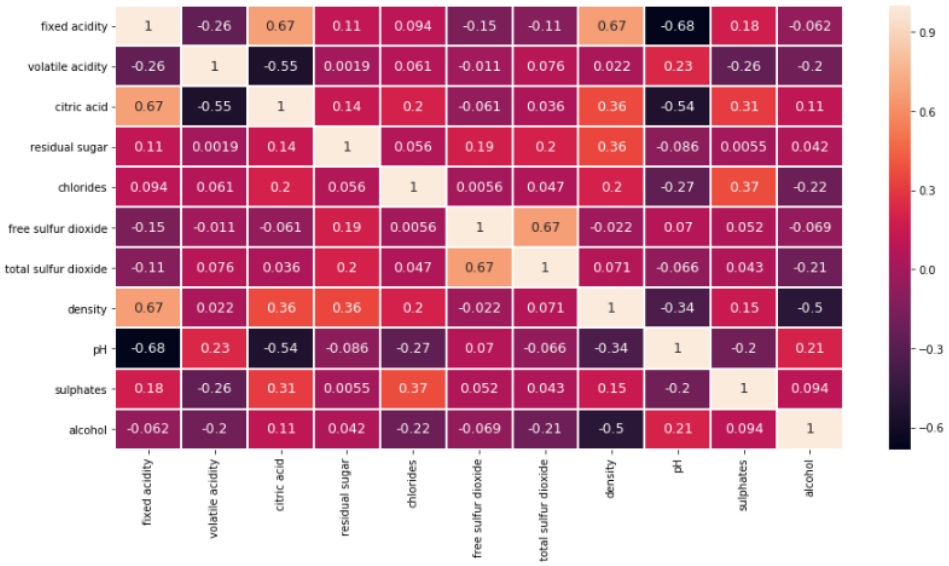
Exploratory Visualisation

Table 2a provides the descriptive statistics for the dataset. The wide numerical ranges for various input physicochemical properties observed in **Table 2a**, such as chlorides ranging from 0.012 to 0.611 and free sulfur dioxide ranging from 1 to 72, would necessitate the application of **Min-Max Scaling**, shown in the later **Data Preprocessing** Section, on the dataset prior to its training. This would ensure each of the 11 features are treated equally by the supervised learner, and avoid favouring features with higher absolute values, for optimal model performance.

Table 2a: Summary Statistics

	mean	std	min	25%	50%	75%	max
fixed acidity	8.319637	1.741096	4.6	7.1	7.9	9.2	15.9
volatile acidity	0.527821	0.17906	0.12	0.39	0.52	0.64	1.58
citric acid	0.270976	0.194801	0	0.09	0.26	0.42	1
residual sugar	2.538806	1.409928	0.9	1.9	2.2	2.6	15.5
chlorides	0.087467	0.047065	0.012	0.07	0.079	0.09	0.611
free sulfur dioxide	15.874922	10.460157	1	7	14	21	72
total sulfur dioxide	46.467792	32.895324	6	22	38	62	289
density	0.996747	0.001887	0.99007	0.9956	0.99675	0.997835	1.00369
pH	3.311113	0.154386	2.74	3.21	3.31	3.4	4.01
sulphates	0.658149	0.169507	0.33	0.55	0.62	0.73	2
alcohol	10.422983	1.065668	8.4	9.5	10.2	11.1	14.9
quality	5.636023	0.807569	3	5	6	6	8

Figure 2b: Correlation Matrix



Given there is the possibility of highly correlated features, Figure 2b and 2c are generated to identify their potential presences. Highly correlated features can give rise to the issue of multicollinearity, which can introduce "regression noises" and adversely affect the supervised learners' performance. While not removed in the eventual model training, it would be used as a basis of performance comparison against a reduced model in the Improvements section.

Figure 2c: Scatter Matrix

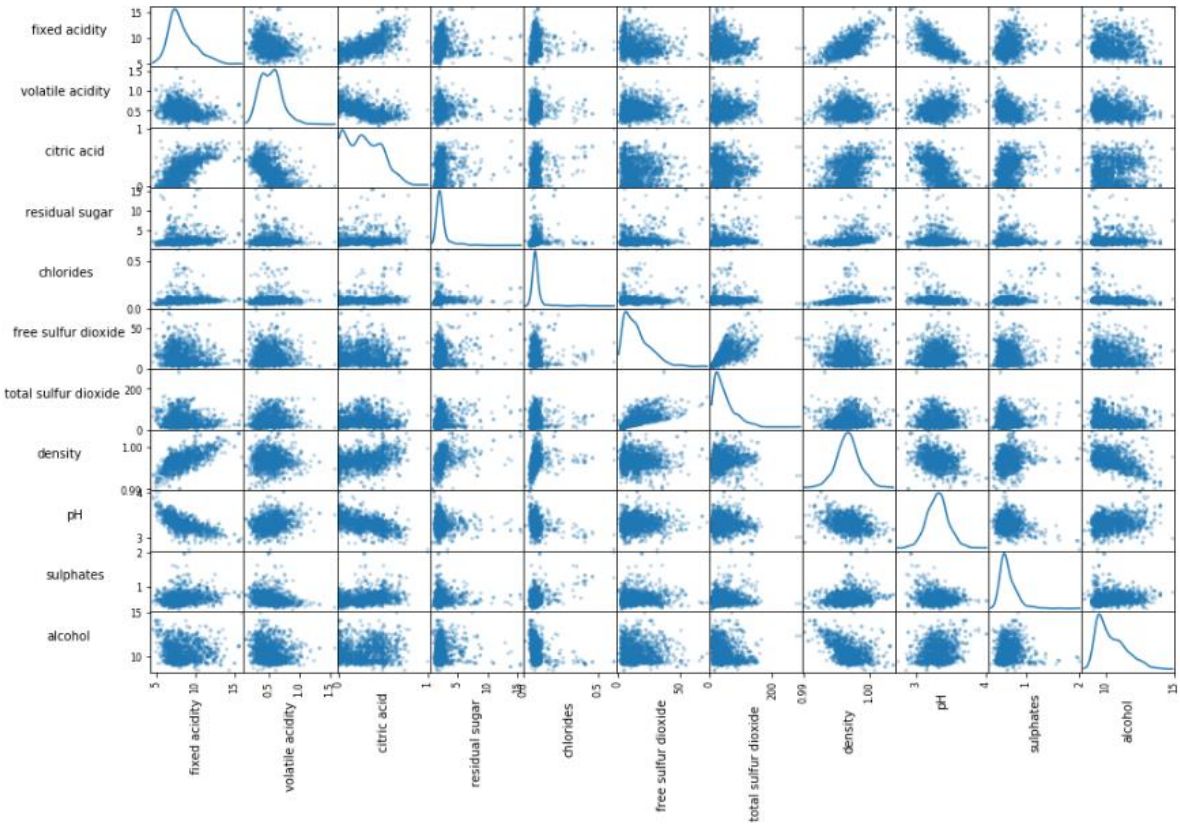


Table 2d: High Correlation Feature Pairs (Correlation ≥ 0.5)

Feature 1	Feature 2	Correlation Sign	Magnitude
Citric Acid	Fixed Acidity	Positive	0.67
Density	Fixed Acidity	Positive	0.67
pH	Fixed Acidity	Negative	-0.68
Citric Acid	Volatile Acidity	Negative	-0.55
pH	Citric Acid	Negative	-0.54
Alcohol	Density	Negative	-0.5

From Figure 2b and 2c, a few correlations are highly evident as tabulated in Table 2d above. Some feature-pairs correlations are highly explicable, for example pH decreasing with increasing concentration of Citric Acid or Fixed Acidity, and Density decreasing with increasing alcohol content (given alcohols' low densities). Interestingly, Citric Acid's high positive and negative correlation with Fixed Acidity and Volatile Acidity might be indicative of its complementary and substitutive use with Tartaric Acid and Acetic Acid in red wines respectively.

Algorithms & Techniques

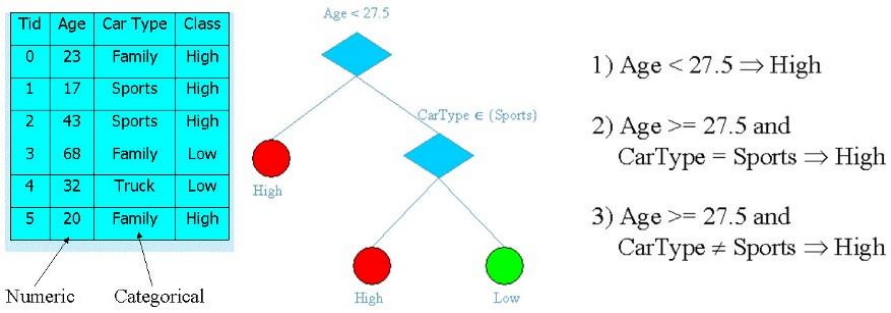
In this project, 3 supervised learning algorithms would be utilised, and the one performing the best on the base dataset would be the eventual algorithm of choice for the classification of the testing (validation) dataset. The 3 algorithms are:

1) DecisionTreeClassifier

a) *Description*

A Decision Tree works on the principle of creating a tree for the entire dataset, where each node, link and leaf represents a feature, decision and outcome respectively [8]. An example of how this can be represented is as per **Figure 2e** below:

Figure 2e: Example of Decision Tree [8]



The sklearn library utilises an optimised mode of CART algorithm to generate the Decision Tree [9], where Gini Indices (see Equation below) are used as cost functions to evaluate splits in the dataset [8].

$$= 1 - \sum_{t=0}^{t=1} P_t^2$$

b) *Justification* [10]

Decision Tree works best when there's correlated variables, since it specialises at elucidating interactions between these variables. As observed in **Figure 2b**, there's substantial correlations between some of the input physicochemical properties to justify this classifier's inclusion.

Also, given the subjective determination of quality figures in the first place, it is likely the human assessors' "underlying model" for each input physicochemical properties would follow discontinuous piecewise models, where each feature value can be potentially segmented into finite and distinct nodes for the determination of the wine's quality. For example, in the case of the residual sugar feature, the numerical ranges in the human assessors' minds can potentially be segmented into

distinct buckets like 0.9-1.3 g/L, 1.3-1.6 g/L etc. These are situations where the DecisionTreeClassifier excels at.

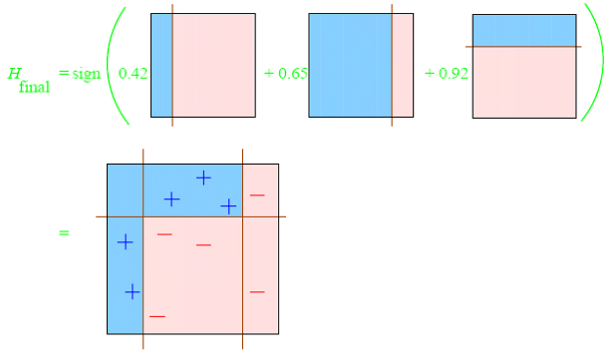
2) AdaBoostClassifier

a) Description [11]

AdaBoost works on the principle of combining various weak learners in a weighted sum to represent the final output of the boosted classifier (strong learner) for the prediction of a red wine's quality. Each weak learners are Decision Tree Classifier by default, and they are created using various features the algorithm have identified in/about the dataset, whether it's the Citric Acid, pH or Density etc. These weak learners would then be used in combination to model a set of rules and decisions for the accurate prediction of a red wine's quality.

During the training process, the algorithm looks at instances where it classified the red wine incorrectly and prioritises their correct classification in subsequent rounds of training, through an increasing penalty function attached to these misclassification errors. The algorithm would then attempt to find the best learner (or decision tree) to incorporate into the ensemble, repeating the process until a pre-specified number of rounds is reached, or if the classification error function cannot be further reduced.

Figure 2f: Example of AdaBoostClassifier [12]



Once completed, as explained above, all the individual learners assembled hitherto are then combined in a weighted sum to generate the final ensemble output, and this sum would determine if the wine is of good quality. For a binary classification problem, AdaBoost works on a principle of majority voting, where past a threshold, a classification of Yes ensues, and below a threshold, a classification of No results.

b) Justification [13]

AdaBoost is generally computationally expensive to use, since it involves the combination of multiple hypothesises to form a better hypothesis, thus adding learning time and memory constraints to the problem. However, since this dataset contains only 1599 rows of data, it would not make AdaBoost that computationally prohibitive to use.

Besides, AdaBoost is good at generalisation, i.e. less susceptible to overfitting issues than other algorithms, and allows the various subjective nuances, i.e. individual feature or combined interactions of features, of what makes a wine good quality to be tackled by specific weak leaners suited to its particular needs.

3) Gaussian Naïve-Bayes

a) *Description* [14]

Gaussian Naïve-Bayes assumes the dataset follows a Gaussian distribution and the algorithm works by calculating the mean and standard deviation for each input variable for each class value. Predictions are made by calculating probabilities of new input value for that class using the Gaussian Probability Density Function.

b) *Justification* [14]

This project is primarily tackling a binary classification problem, which Gaussian Naïve-Bayes is very well suited for. Besides, as observed in **Figure 2c**, with the exception of Citric Acid, the distribution of the input physicochemical properties roughly follows a Gaussian distribution, albeit with some positive skewness for a few input features. Therefore, the underlying critical assumption of Gassuain distribution for Gaussian Naïve-Bayes classifier can still approximately hold, key to this algorithm classification performance.

Benchmark Model

The trained model performance will be benchmarked against the original paper's overall accuracies, which utilises Neural Network and Support Vector Machine (SVM) architectures to predict the wine's quality values. There is some differences however, since the original paper retains the scale of 0-10 for the classification task (a multiclass classification), whereas in this proposal a binary classification task with an arbitrary cut-off quality value of 7 has been proposed instead. Notwithstanding, the goal would be to match the highest accuracy figure of **89.0%** achieved by the SVM architecture used in the paper [1]. As a basis of comparison, the original paper's results have been tabulated in **Table 2g** below:

Table 2g: Reference Benchmark Accuracies

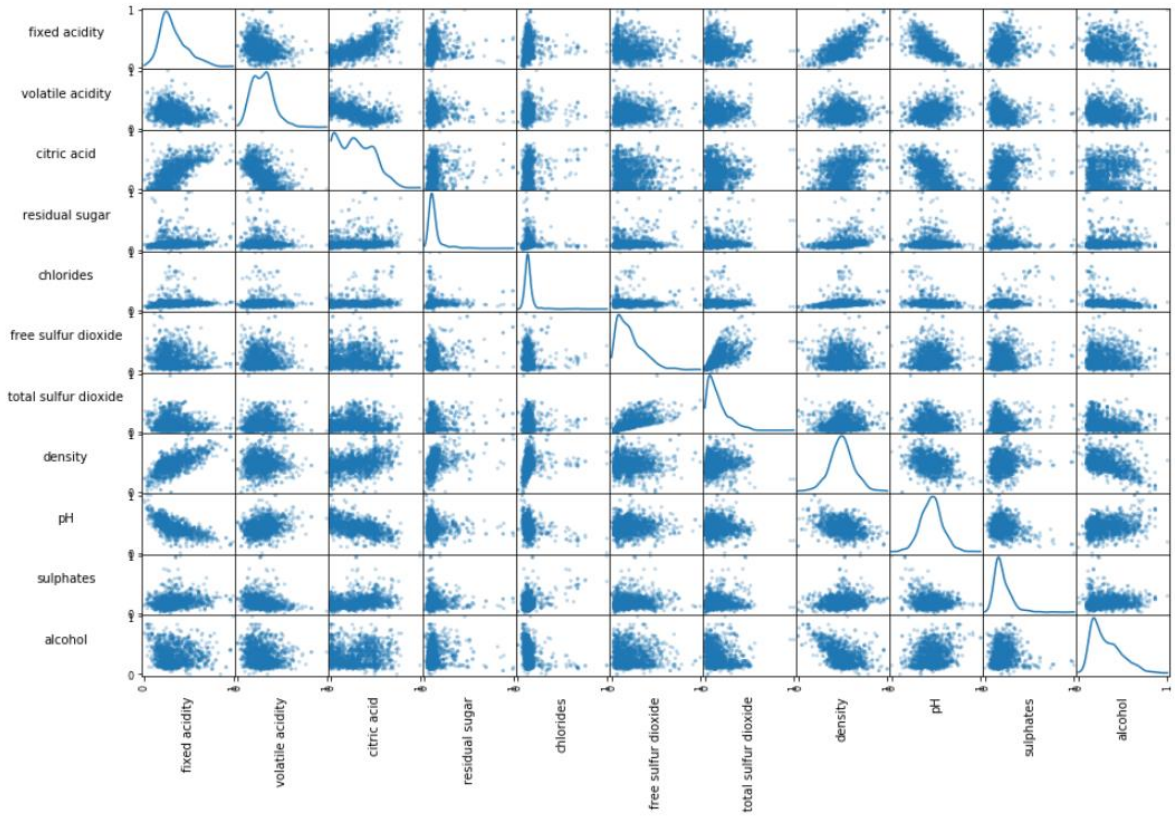
	Benchmark Accuracy
Multiple Regression	88.6 ± 0.1
Neural Network	88.8 ± 0.2
Support Vector Machines	89.0 ± 0.2

III. Methodology

Data Preprocessing

Before one can proceed with the training of the supervised learners, the dataset has to be pre-processed. Given there are no non-numeric columns, no one-hot encoding pre-processing would be required here. As discussed earlier in the **Exploratory Visualisation** section, **Min-Max Scaling** would still have to be performed here to ensure each of the 11 numeric features are treated equally by the supervised learner. The transformed scatter matrix generated is shown in **Figure 3a** below.

Figure 3a: Transformed Scatter Matrix



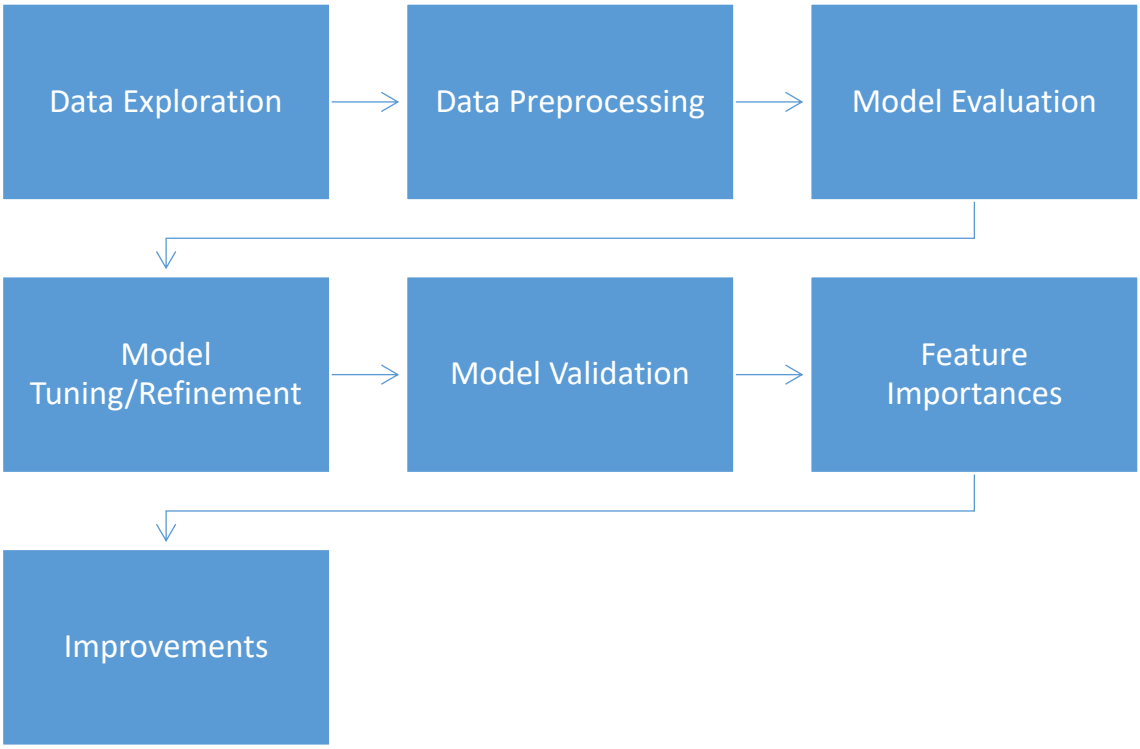
Besides Min-Max Scaling, the transformation of the quality column would have to be applied to convert this dataset into a binary classification problem. The resulting quality distribution would be as shown in **Figure 1c** earlier.

The final steps of the Data Preprocessing involve splitting the features and target columns into training and testing (validation) datasets in the desired 9:1 ratio. The training dataset is primarily used for the training of the supervised learners, while the testing (validation) dataset to validate the model's accuracy, i.e. generalises to out-of-sample dataset.

Implementation

The implementation of supervised learners on the dataset would follow the process pathway detailed below in **Figure 3b**:

Figure 3b: Implementation Process Pathway



a) Data Exploration

This section primarily deals with exploratory analysis, as described in the **Exploratory Visualisation** section earlier, where summary statistics, scatter matrices & correlation matrices were generated to identify transformations required as well as possible correlations between features. The resulting distribution generated in the scatter matrices also helped facilitate the identification of supervised learners suitable for the competent evaluation of this dataset .

b) Data Preprocessing

Here, as described in the prior **Data Preprocessing** section, min-max scaling of numeric columns, transformation (mapping) of the dataset's quality column to a binary classification problem and the splitting of the features & target columns into training & testing (validation) datasets respectively were sequentially performed to prepare this dataset for use with the 3 supervised learners selected.

c) Model Evaluation

For this segment, 3 different supervised learners would be trained against 1%, 10% and 100% of the training dataset and their performances evaluated against the testing (validation) dataset. The resulting comparison figures have been generated in **Figure 3c and 3d** respectively. **Appendix A** consists of two additional **Figures 6a and 6b** containing side-by-side comparisons of the 3 algorithms' confusion matrices and classification reports generated alongside Figure 3c and 3d.

Figure 3c: Comparison of Summary Statistics

AdaBoostClassifier

	1%	10%	100%
acc_test	0.756250	0.843750	0.906250
acc_train	0.738707	0.890202	1.000000
f_test	0.357143	0.388889	0.666667
f_train	0.347658	0.585034	1.000000
pred_time	0.001008	0.018982	0.017972
train_time	0.002000	0.065969	0.203160

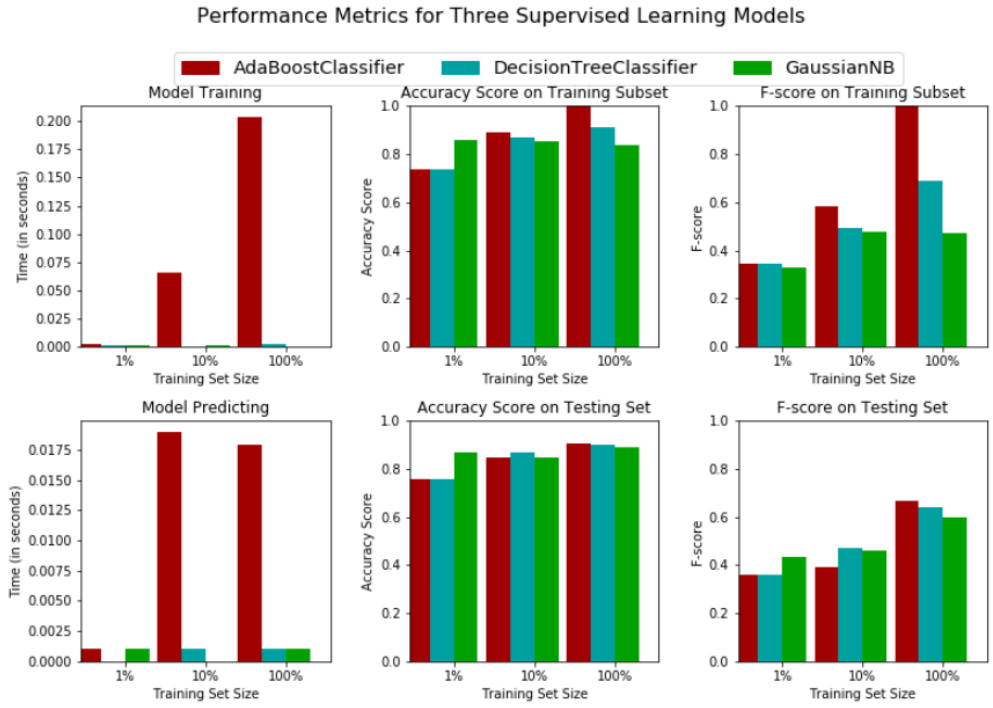
DecisionTreeClassifier

	1%	10%	100%
acc_test	0.756250	0.868750	0.900000
acc_train	0.738707	0.868659	0.913829
f_test	0.357143	0.472973	0.641026
f_train	0.347658	0.495768	0.691937
pred_time	0.000000	0.000999	0.001009
train_time	0.001000	0.000000	0.002998

GaussianNB

	1%	10%	100%
acc_test	0.868750	0.843750	0.887500
acc_train	0.859625	0.851286	0.839472
f_test	0.431034	0.461538	0.600000
f_train	0.329087	0.476815	0.472242
pred_time	0.000999	0.000000	0.000999
train_time	0.000999	0.000999	0.000000

Figure 3d: Comparison of Performance Metrics



From the results above, of the 3 algorithms utilised, AdaBoost performed the best on both 100% training and testing datasets for the two evaluation metrics of accuracy and $F_{\beta=0.5}$ scores. Consequently, it would be deemed as the algorithm of choice for the subsequent Model Tuning and Validation segments.

Interestingly, one can also evidently observe how accuracy scores are not reliable measure of the model's performance. Despite being trained on only 1% of the dataset, all three classifiers managed to achieve surprisingly high accuracy scores exceeding 70% for both training and testing dataset in **Figure 3d**. This indubitably justify the choice of $F_{\beta=0.5}$ score as the key evaluation metric instead, for which model's weights would be tuned against.

d) Model Tuning/Refinement

The hyperparameters of the algorithm of choice, AdaBoost, would be tuned here using GridSearchCV to meet, if not exceed, the benchmark targets of 89% accuracy score,

besides achieving high $F_{\beta=0.5}$ score, the key evaluation metric here. The intermediate and final values obtained have been documented in the following **Refinement** section, with a two-step tuning approach used to obtain the optimal hyperparameter set in a computationally efficient way.

e) Model Validation

The tuned model would be validated against the testing (validation) dataset for its final accuracy and $F_{\beta=0.5}$ scores, with the final confusion matrices and classification reports generated as well for further analysis. The final hyperparameter values set would also be correspondingly tabulated for reference.

f) Feature Importance

Not all features are equally critical to the contribution to a red wine's quality, where the plotting of a feature importance graph can facilitate the identification of important features. Besides, with highly correlated features identified in earlier sections, such a plot can help confirm these surmises, as well as facilitate the implementation of potential improvements. Improvements could include the retraining of model with reduced features to achieve higher model performances, possibly due to removal of "regression noises" etc.

g) Improvements

Two possible improvements were explored, and their corresponding performances were then compared with the tuned AdaBoostClassifier model found:

1) Re-training of the optimal model found against a dataset with reduced features
2) Principal Component Analysis (PCA)

Refinement

This section details the intermediate and final values utilised in the GridSearchCV search space for the choice classifier's hyperparameter tuning. Prior to tuning the algorithm of choice AdaBoost itself, its default base estimator, i.e. DecisionTreeClassifier, was first tuned in isolation, and the resulting hyperparameter values obtained were then utilised as starting estimates for the AdaBoost's base_estimator ranges. Such a two-step approach greatly reduced the magnitude of hyperparameter search space required to find the optimal value set (mainly base_estimator) that generalises well to the validation dataset, thus reducing the relevant computational costs.

a) Decision Tree Hyperparameter Tuning for AdaBoost

The three hyperparameters tuned for DecisionTreeClassifier are maximum depth, minimum samples leaf as well as minimum samples split. Given Decision Tree's tractability for graphical visualisation (as compared to AdaBoost), a Graphviz plot of the

tree has been generated in **Appendix B - Figure 7** to provide some insights pertaining to the underlying classifier criterion at each tree node, i.e. individual features, for the eventual determination of the red wine quality.

Table 3e: Decision Tree Tuned Hyperparameters Set

Parameters	Description	Values Tested	Best Value
max_depth	The maximum tree depth	[2, 3, 4, 5, …, 30, 31, 32]	6
min_samples_leaf	Minimum number of samples required to split an internal node	[1, 2, 3, 4]	1
min_samples_split	Minimum number of samples at a leaf node	[2, 3, 4]	4

b) AdaBoostClassifier Hyperparameter Tuning

Using the above tuned DecisionTreeClassifier hyperparameter values as starting estimates, the GridSearchCV's base_estimator ranges would be generated, for which the AdaBoost would be tuned against. Despite three hyperparameters for Decision Tree being calibrated above, only maximum depth was utilised for the GridSearchCV here, given the introduction of the other two leading to poorer $F_{\beta=0.5}$ scores when cross-validated on the testing dataset. This could potentially be attributed to model overfitting, hence the lower scores observed when either min_samples_leaf or min_samples_split hyperparameter were included.

Table 3f: AdaBoostClassifier Hyper-parameters Tuning Results

Parameters	Description	Values Tested			Best Value
base_estimator	The base estimator, i.e. weak learners, from which the boosted ensemble is built	DecisionTreeClassifier	max_depth	3	DecisionTreeClassifier(max_depth=10, random_state=42)
				4	
				5	
				6	
				7	
				8	
				9	
				10	
			random_state	42	
learning_rate	The value which determines how much contribution each classifier is shrunk by	[0.8, 0.85, 0.9, 0.95, 1]			0.85

IV. Results

Model Evaluation/Validation

The tuned model's final hyperparameter values are tabulated in **Table 4a** below:

Table 4a: Optimal AdaBoostClassifier Hyperparameters

Algorithm:	SAMME.R	
Base Estimator:	DecisionTreeClassifier	
	Max Depth:	10
	Min Samples Leaf:	1
	Min Samples Split:	2
Learning Rate:	0.85	
N Estimators:	50	

The corresponding unoptimised and optimised model accuracy and $F_{\beta=0.5}$ scores on the validation dataset have been provided below, alongside the confusion matrix and classification report provided in **Figure 4b**:

```
Unoptimized model
------
Accuracy score on testing data: 0.8313
F-score on testing data: 0.2273

Optimized Model
------
Final accuracy score on the testing data: 0.9187
Final F-score on the testing data: 0.7432
```

The optimised model's accuracy score of **91.87%** generated exceeded the Benchmark Model target scores tabulated in **Figure 2g**. Also, a sufficiently high F-score of **74.32%** was achieved as well. The only possible drawback of this optimised model is the low recall of 0.50 observed in the classification report, where quite a number of good wines were falsely identified as bad wines. Possible improvements to tackle this low recall ratio, such as Features Reduction and PCA, will be implemented in the following section and compared against the performances found here. Other improvements, though out of this report scope, could include implementation of alternative models like XGBoost.

Figure 4b: Confusion Matrix & Classification Report


```
Classification Report
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.93      0.99      0.95       138
           1       0.85      0.50      0.63        22

avg / total       0.91      0.92      0.91       160
```

To check the robustness of the optimised model found, it was ran with the same set of hyperparameters, though with different random states. The results are as tabulated below in **Table 4c** below:

Table 4c: Robustness Test of Optimised Model

Random State	Accuracy Score	F-score
26	90.00%	64.10%
30	90.00%	64.29%
34	90.00%	64.10%
38	90.00%	63.95%
42	91.88%	74.32%
46	90.00%	64.10%
50	89.38%	60.81%
54	90.63%	67.07%
58	91.25%	70.51%
62	88.75%	57.14%

Across various random states, while accuracy score remain within a tight range of **88.75%** to **91.88%**, F-score on the other hand varied widely from **57.14%** to **74.32%**. This might be indicative of the model's sensitivity to small changes in data, especially since accuracy score have demonstrated its unreliability as a performance measure and F-score being the more reliable one. Therefore, this optimised model is unlikely to be generalisable to unseen data, where slight differences in red wine inputs can effect large changes in model performance. The improvements suggested above can potentially tackle this issue, and robustness check would be performed to evaluate these improvements' impact on the model robustness, as will be later observed.

Comparison with Benchmark Model

The benchmark model achieved a top accuracy score of **89.0%** via SVM architecture as tabulated in **Figure 2g**. A side-by-side comparison with this value would be made against **Table 4c**'s accuracy scores in **Table 4d** below:

Table 4d: Accuracy Score Comparison with Benchmark Model

Random State	Accuracy Score	Difference
26	90.00%	1.12%
30	90.00%	1.12%
34	90.00%	1.12%
38	90.00%	1.12%
42	91.88%	3.23%
46	90.00%	1.12%
50	89.38%	0.42%
54	90.63%	1.83%
58	91.25%	2.53%
62	88.75%	-0.28%

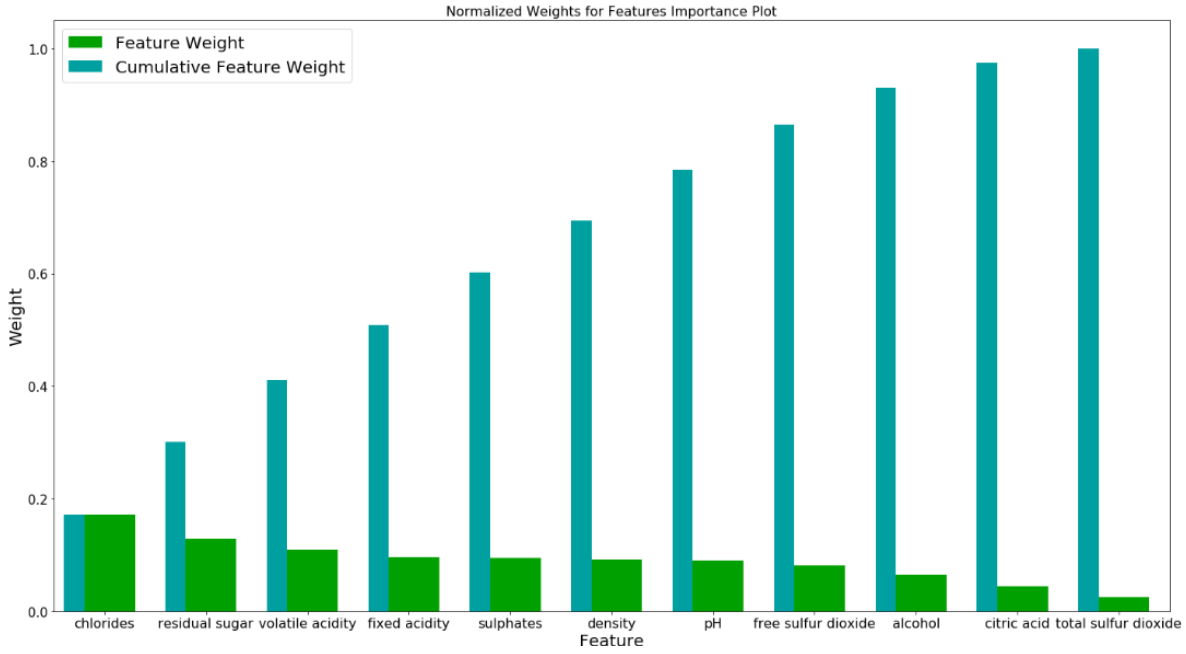
Though a slightly different problem statement, the optimised model has demonstrated accuracy scores comparable with the benchmark model values, indicating the optimised model has to a certain extent solved the problem. Improvements, implemented in subsequent sections, will push these accuracy scores higher across the various random state values.

V. Conclusions

Free-Form Visualisation

The feature importance plot is generated in Figure 5a below:

Figure 5a: Feature Importance Plot



Based on the normalised weights above, it can be observed the first 6 features alone, i.e. chlorides, residual sugar, volatile acidity, fixed acidity, sulphates and density accounted for more than 75% of the cumulative feature weights, indicating there's potential scope for features reduction. This could potentially remove regression noises and help improve model performance, as observed in the later Improvements section. As identified in **Table 2d**, Citric Acid is highly correlated with multiple features, i.e. Fixed Acidity, Volatile Acidity and pH, resulting in it being very much captured in the latter features, hence explaining its feature insignificance observed in **Figure 5a**. This, to a lesser extent, could also explain why density and pH are less significant features for the determination of red wine quality, given their high correlations with Fixed Acidity and other features too.

Improvements

Two improvements were implemented below for comparison purposes with the optimised model found above:

a) Features Reduction

Borrowing on the feature importance plot found above, the number of chosen input features would be reduced to the first 6 in **Figure 6a** and the optimised model found in **Section IV** retrained against this subset of chosen features. The resulting model performance has been provided below, alongside side-by-side comparisons of confusion matrices and classification reports in **Figure 6b and 6c** respectively:

```
Final Model trained on full data
------------------------------------
Accuracy on testing data: 0.9187
F-score on testing data: 0.7432

Final Model trained on reduced data
------------------------------------
Accuracy on testing data: 0.9313
F-score on testing data: 0.8108
```

The model performances did indeed improve, with accuracy and $F_{\beta=0.5}$ scores increasing from **91.87%** to **93.13%** and **74.31%** to **81.08%** respectively. This could potentially be attributed to, as explained previously, the removal of potential regression noises. Recall did improve from **0.50** to **0.55** upon the application of features reduction. The difference between the optimised and reduced predictions model was 1 False Positive and Negative, very fine margins but for a dataset of 1599 rows, led to significant improvements in the resulting accuracy and $F_{\beta=0.5}$ scores.

Figure 6b: Side-by-side Comparison of Confusion Matrices

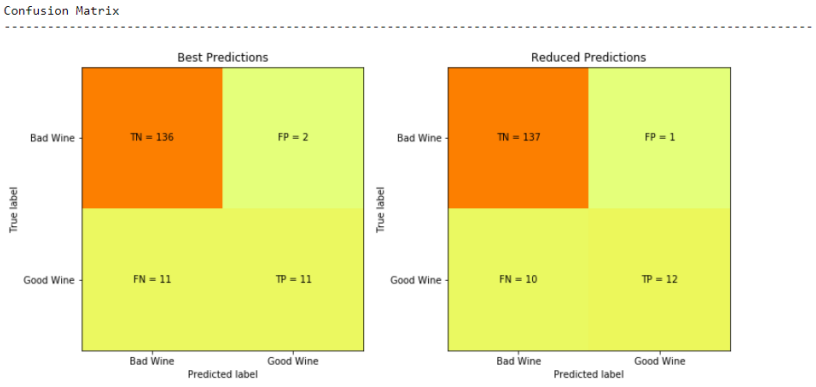


Figure 6c: Side-by-side Comparison of Confusion Matrices

```
Classification Report for Best Predictions
--------------------------------------------------------------------

             precision    recall  f1-score   support

          0       0.93      0.99      0.95       138
          1       0.85      0.50      0.63        22

avg / total       0.91      0.92      0.91       160


Classification Report for Reduced Predictions
--------------------------------------------------------------------

             precision    recall  f1-score   support

          0       0.93      0.99      0.96       138
          1       0.92      0.55      0.69        22

avg / total       0.93      0.93      0.92       160
```

Robustness checks will also be performed here to evaluate impact this improvement has on the variability of F-score observed in the prior section. The results are tabulated in **Table 6d** below:

Table 6d: Robustness Test of Reduced Model

Random State	Accuracy Score	Difference
26	91.88%	75.76%
30	92.50%	74.47%
34	91.25%	69.77%

38	90.00%	63.95%
42	93.13%	81.08%
46	91.88%	73.17%
50	91.88%	73.17%
54	91.25%	70.51%
58	92.50%	76.92%
62	91.88%	74.32%

Across various random states, though accuracy score still remained within a tight range of **90.00%** to **93.13%**, F-score varied widely from **63.95%** to **81.08%**. This is likely due to the intrinsic distribution of the dataset, where the low absolute number of good wines would amplify the effect incorrect and/or correct labelling of it on the F-score values computed.

b) Principal Component Analysis (PCA)

Another algorithm, PCA, was tried here for purposes of handling highly correlated features within the dataset. For tractability, PCA is an algorithm that attempts to combine correlated features into "new dimensions", while simultaneously being able to reduce the input sizes and hence computational complexity. When utilised here, a variance of 90% was specified, which can be thought of as the percentage of information retained after PCA transformation. The resulting accuracy and $F_{\beta=0.5}$ scores are generated below:

```
Final Model trained on PCA data
------------------------------------
Accuracy on testing data: 0.9062
F-score on testing data: 0.6818
```

The accuracy and $F_{\beta=0.5}$ scores was poorer than that of both the reduced data and optimised models, possibly indicative of information losses when transforming the dataset.

Reflections

The steps for this project can be summarised with the following steps:

1. The search for a problem and its relevant dataset
2. Analysis and preprocessing of the dataset
3. Identification of benchmark targets
4. Evaluation of 3 different supervised learners to identify one most suited for the problem
5. Tuning of selected learners
6. Validation of tuned learners against validation dataset
7. Identification of important features
8. Implementation and comparison of improvements implemented

The most interesting part of the project was being able to generate the various visuals for the predictions made by the various classifiers, especially the Graphviz plot for the DecisionTree and Feature Importance plot. It facilitates one's understanding of and provide insights into the prediction results made by the classifiers. Indeed, these would be very valuable takeaways when implementing various supervised learning algorithms in the future.

Bibliography

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reise, "Modeling wine preferences by data mining from physicochemical properties," *Elsevier,* p. 7, 2009.

[2] K. Svitil, "CalTech," 14 January 2008. [Online]. Available: https://www.caltech.edu/about/news/wine-study-shows-price-influences-perception-1374.

[3] A. Long, "Understanding Data Science Classification Metrics in Scikit-Learn in Python," 6 August 2018. [Online]. Available: https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019. [Accessed 21 April 2019].

[4] "Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures," 9 September 2016. [Online]. Available: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/. [Accessed 1 May 2019].

[5] "Precision and recall," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall. [Accessed 1 May 2019].

[6] "F1_Score," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/F1_score. [Accessed 21 April 2019].

[7] "Red Wine Quality," Kaggle, [Online]. Available: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009. [Accessed 21 April 2019].

[8] M. Sanjeevi, "Chapter 4: Decision Trees Algorithms," Medium, 7 Oct 2017. [Online]. Available: https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1. [Accessed 5 May 2019].

[9] Russell, "Creating and Visualizing Decision Trees with Python," Medium, 6 Aug 2017. [Online]. Available: https://medium.com/@rnbrown/creating-and-visualizing-decision-trees-with-python-f8e8fa394176. [Accessed 5 May 2019].

[10] "What are the disadvantages of using a decision tree for classification?," Quora, [Online]. Available: https://www.quora.com/What-are-the-disadvantages-of-using-a-decision-tree-for-classification. [Accessed 5 May 2019].

[11] H. Jung, "Adaboost for Dummies: Breaking Down the Math (and its Equations) into Simple Terms," Towards Data Science, 10 April 2018. [Online]. Available: https://towardsdatascience.com/adaboost-for-dummies-breaking-down-the-math-and-its-equations-into-simple-terms-87f439757dcf. [Accessed 5 May 2019].

[12] "Boosting," UPenn, [Online]. Available: https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=lectures.boosting. [Accessed 5 May 2019].

[13] A. Ravanshad, "Ensemble Methods," Medium, 28 April 2018. [Online]. Available: https://medium.com/@aravanshad/ensemble-methods-95533944783f. [Accessed 4 May 2019].

[14] J. Brownlee, "Naive Bayes for Machine Learning," Machine Learning Mastery, 11 April 2016. [Online]. Available: https://machinelearningmastery.com/naive-bayes-for-machine-learning/. [Accessed 5 May 2019].

VI. Appendix A

Two additional figures, provided in **Figure 6a** and **6b** below, delineates the side-by-side comparison of the 3 supervised learners' confusion matrices and classification reports.

Figure 6a: Comparison of Confusion Matrices

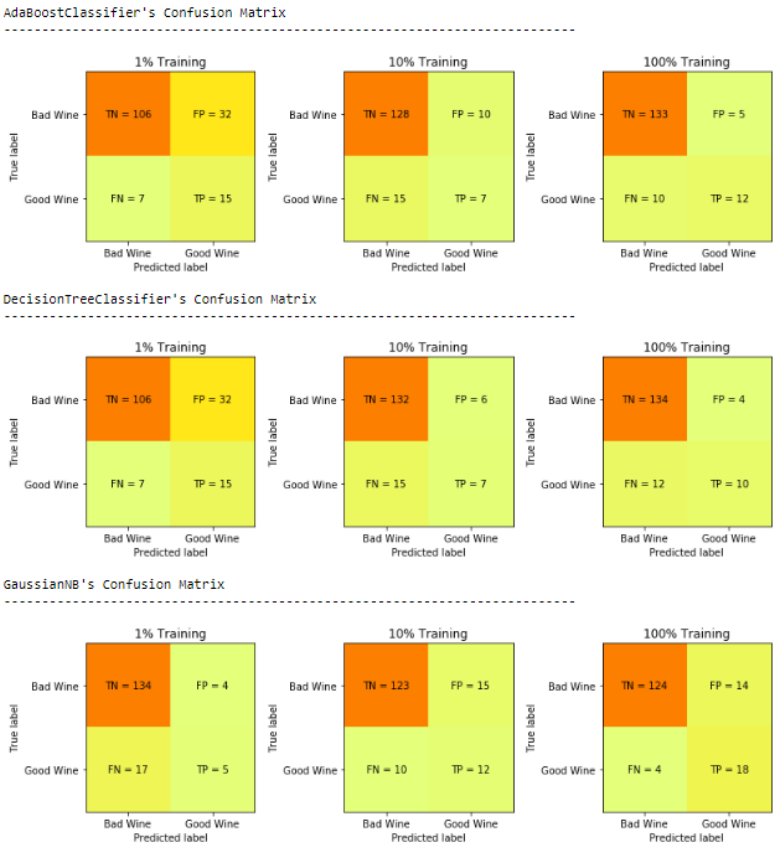


Figure 6b: Comparison of Classification Reports

AdaBoostClassifier's Classification Report Matrix

--------- 1% Training: ---------

	precision	recall	f1-score	support
0	0.94	0.77	0.84	138
1	0.32	0.68	0.43	22
avg / total	0.85	0.76	0.79	160

--------- 10% Training: ---------

	precision	recall	f1-score	support
0	0.90	0.93	0.91	138
1	0.41	0.32	0.36	22
avg / total	0.83	0.84	0.84	160

--------- 100% Training: ---------

	precision	recall	f1-score	support
0	0.93	0.96	0.95	138
1	0.71	0.55	0.62	22
avg / total	0.90	0.91	0.90	160

DecisionTreeClassifier's Classification Report Matrix

--------- 1% Training: ---------

	precision	recall	f1-score	support
0	0.94	0.77	0.84	138
1	0.32	0.68	0.43	22
avg / total	0.85	0.76	0.79	160

--------- 10% Training: ---------

	precision	recall	f1-score	support
0	0.90	0.96	0.93	138
1	0.54	0.32	0.40	22
avg / total	0.85	0.87	0.85	160

--------- 100% Training: ---------

	precision	recall	f1-score	support
0	0.92	0.97	0.94	138
1	0.71	0.45	0.56	22
avg / total	0.89	0.90	0.89	160

GaussianNB's Classification Report Matrix

--------- 1% Training: ---------

	precision	recall	f1-score	support
0	0.89	0.97	0.93	138
1	0.56	0.23	0.32	22
avg / total	0.84	0.87	0.84	160

--------- 10% Training: ---------

	precision	recall	f1-score	support
0	0.92	0.89	0.91	138
1	0.44	0.55	0.49	22
avg / total	0.86	0.84	0.85	160

--------- 100% Training: ---------

	precision	recall	f1-score	support
0	0.97	0.90	0.93	138
1	0.56	0.82	0.67	22
avg / total	0.91	0.89	0.90	160

VII. Appendix B

Figure 7: Graphviz Visualisation of Intermediate Decision Tree