Insights

Machine Learning

EDA

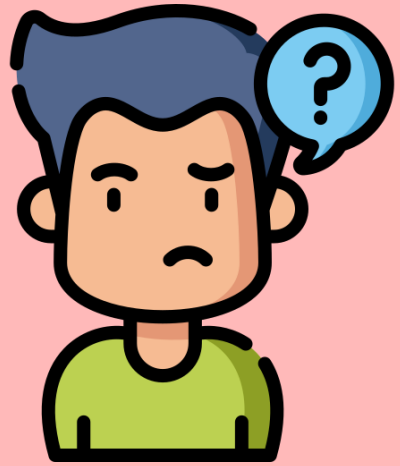
Problem Formulation

CZ1115 Mini Project: Red Wine Quality Classification

Chua Yong Xuan, Lim Xin Yi, Timothy Teh

Introduction & Practical Motivation

- Too many choices and variety of wine
- How do we distinguish between the wines and make an informed decision as a consumer?
- What determines good and bad wine?



Problem Definition:

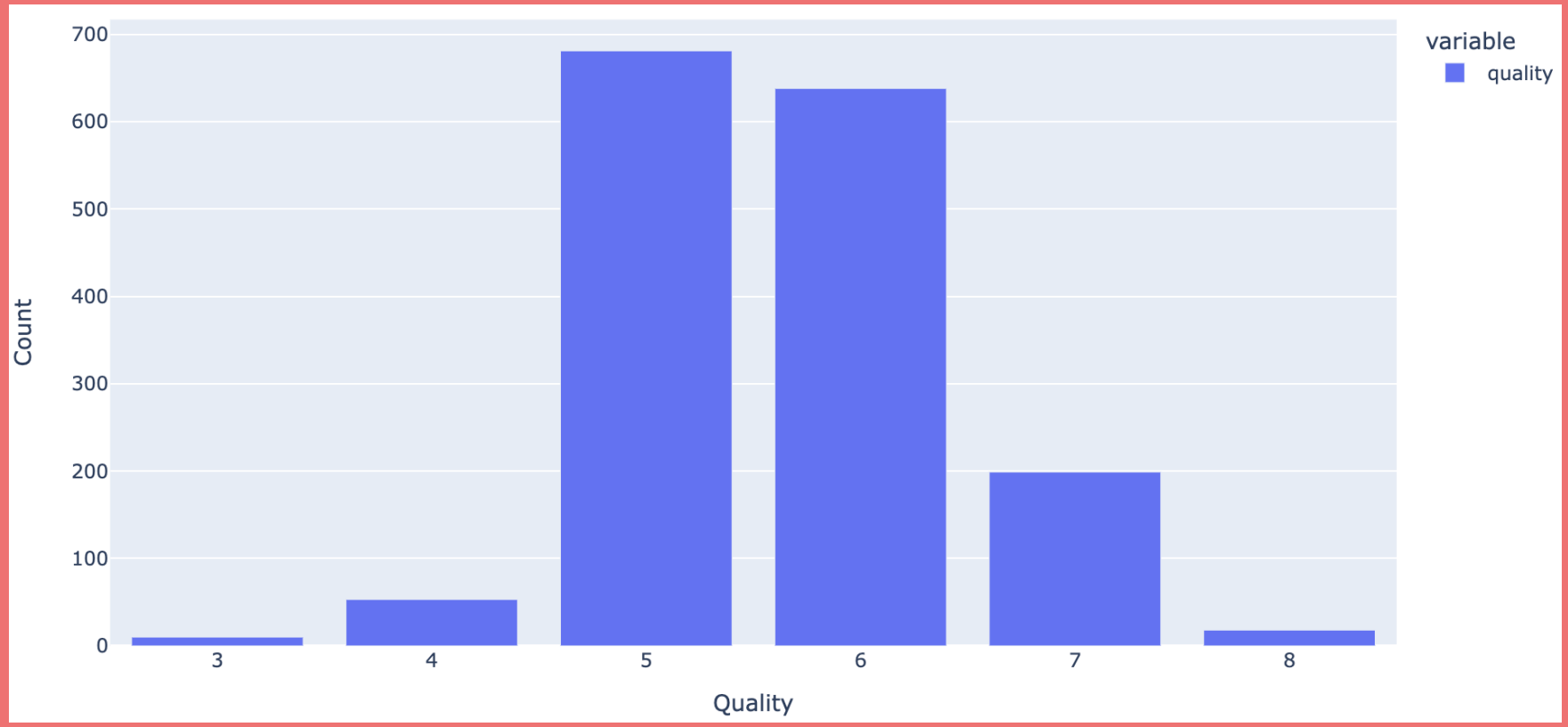
Using the physicochemical properties of wine, predict the wine quality.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

Red Wine Dataset:

12 Attributes (eg. Fixed acidity, alcohol level)

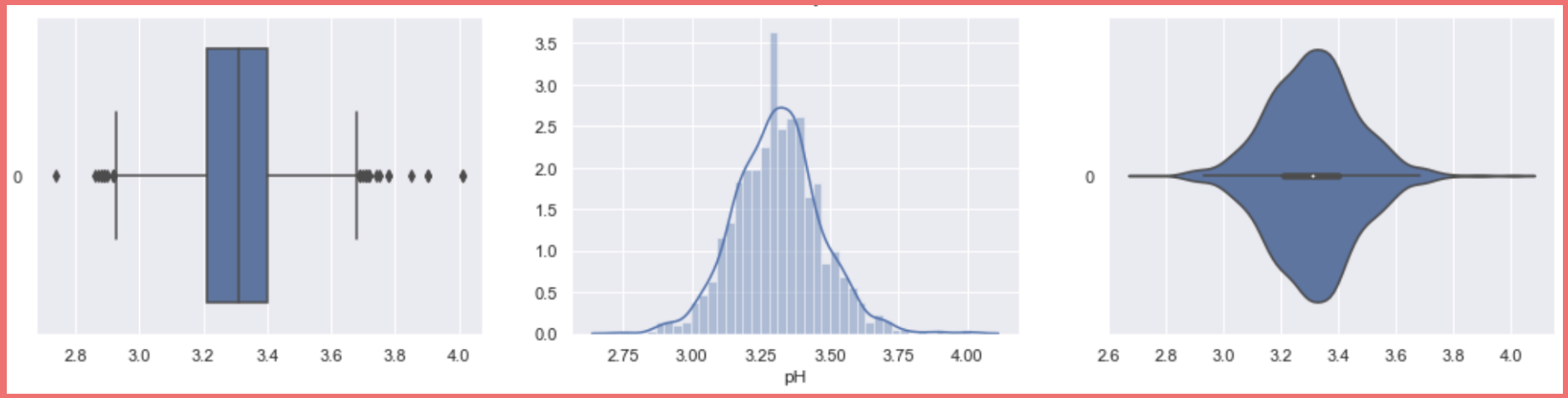
What is the quality of wines in the dataset?



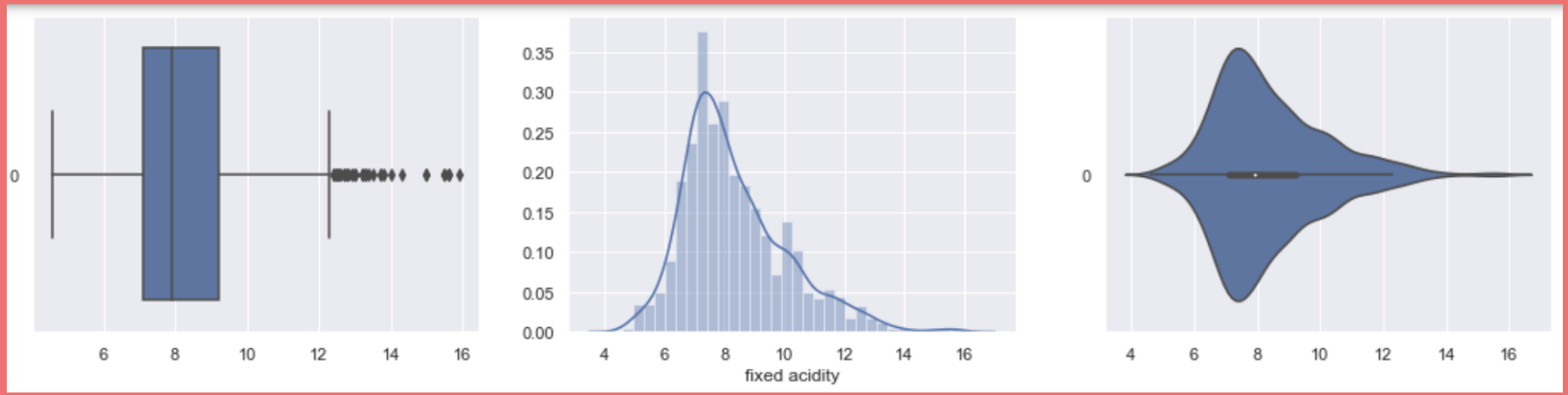
<u>Resembles a normal distribution</u>:

- mean (5.63) ≙ median (6.0)
- bell curve that is almost symmetrical about mean

How are the other attributes distributed?

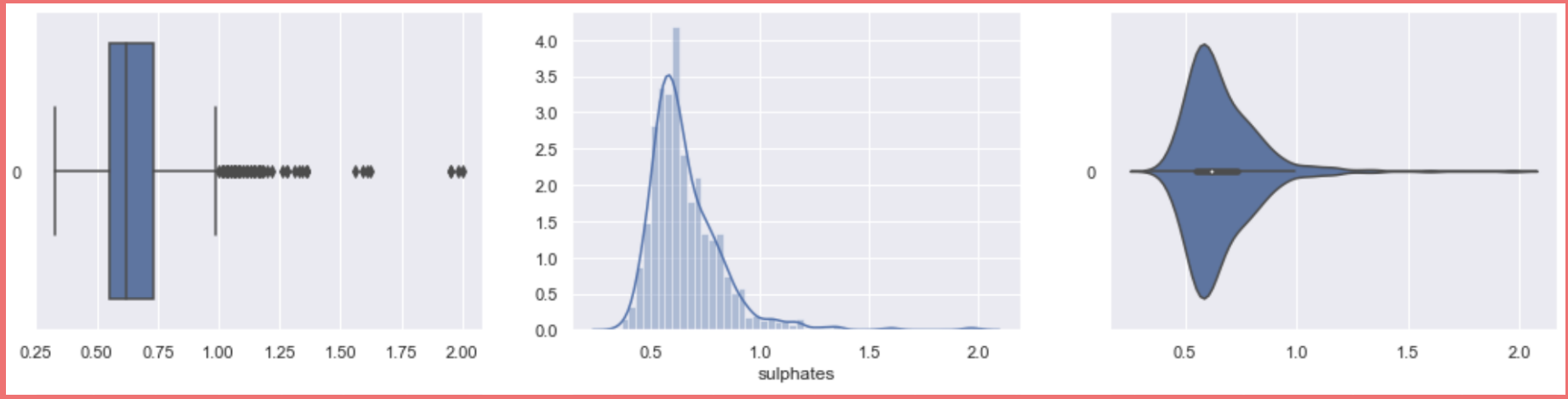


Some are almost an ideal normal distribution

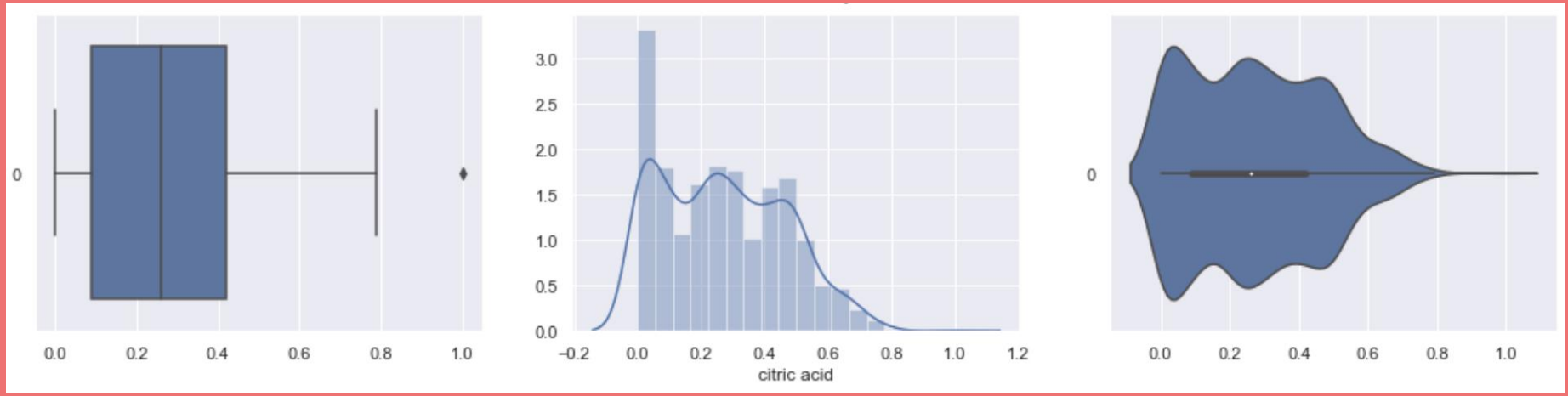


Some are quite close to a normal distribution

How are the other attributes distributed?

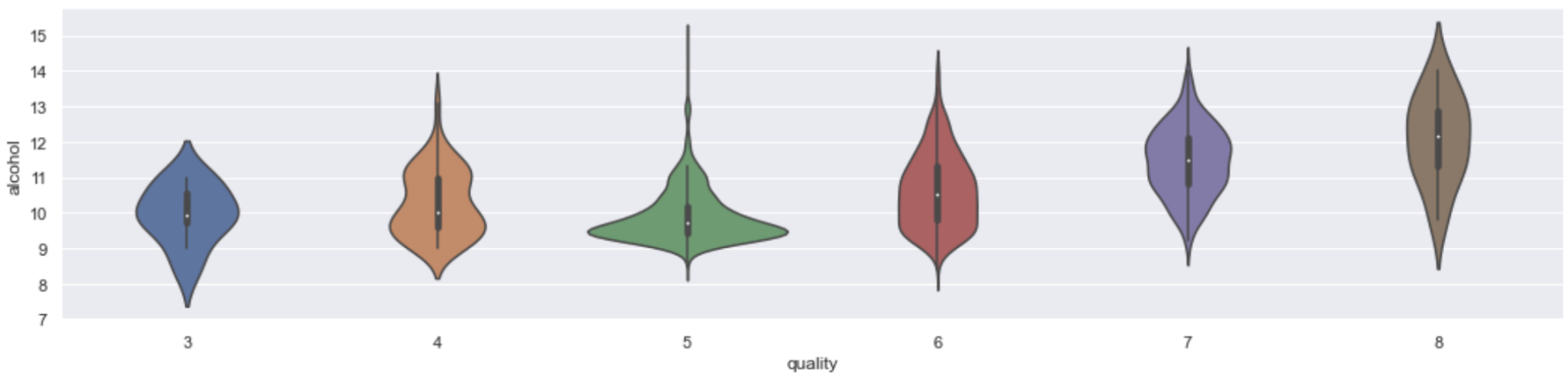


Some are skewed to the left or to the right

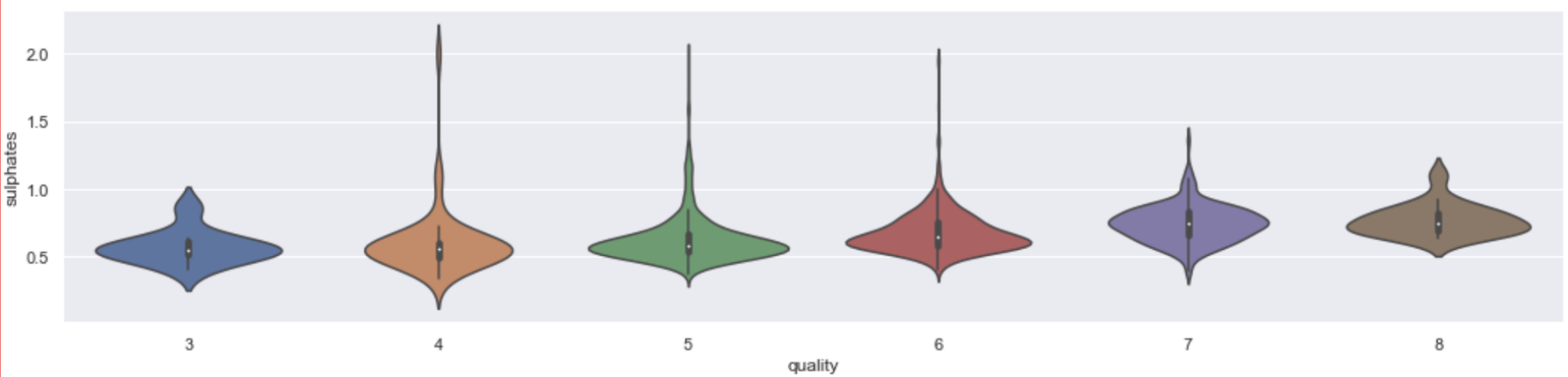


Some are no where near a normal distribution

How is each attribute related to the quality?



Alcohol has the strongest relationship with quality

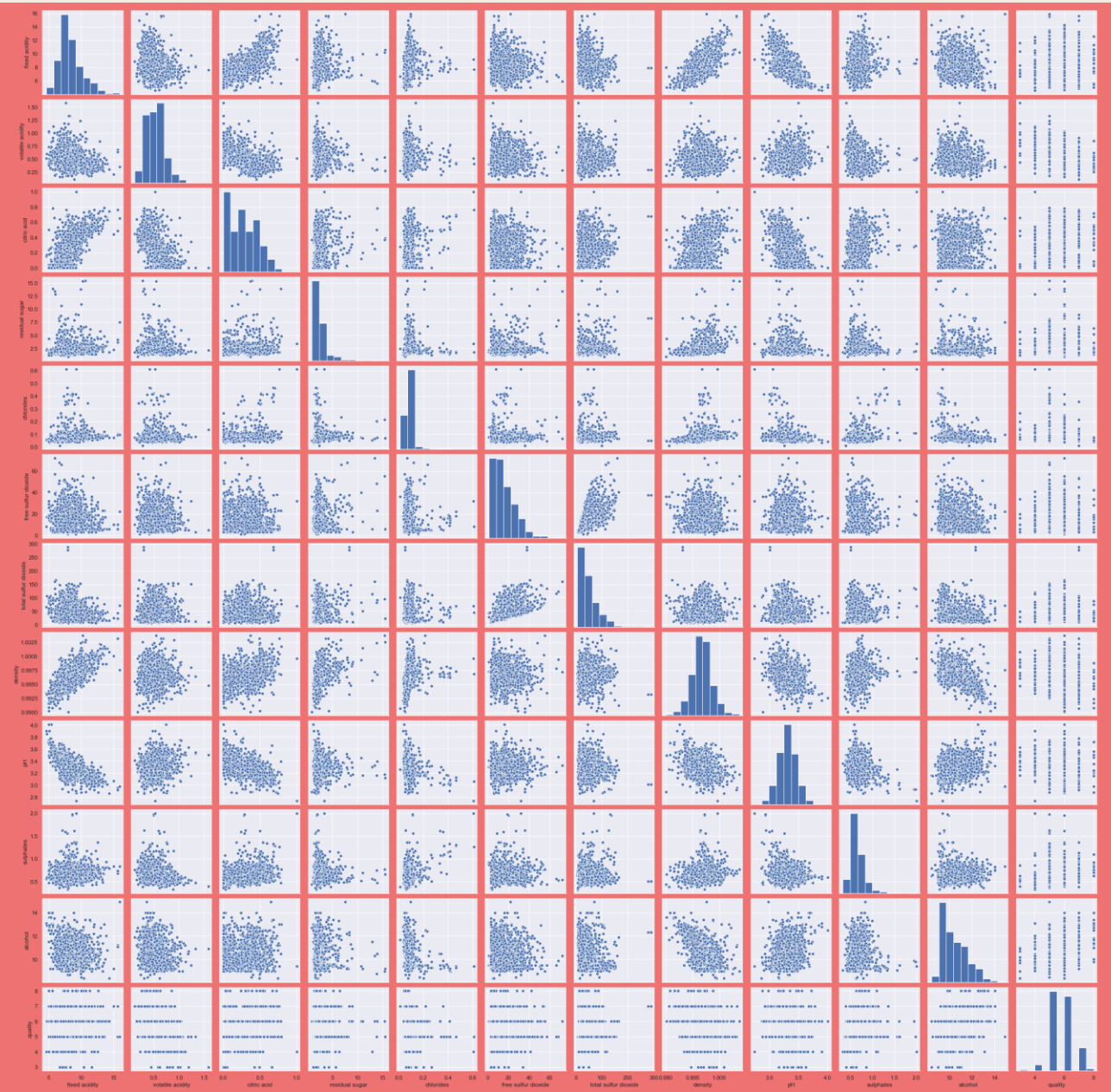


While most of the other attributes do not show a strong relationship with quality

Insights

Machine Learning

A more extensive representation: Pair Plots



EDA

Problem Formulation

Data Cleaning

Checking for NULL values

```
1 wine_data.isnull().sum()
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

Removing duplicates

```
1 wine_data.loc[wine_data.duplicated()]
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
4	7.4	0.700	0.00	1.90	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
11	7.5	0.500	0.36	6.10	0.071	17.0	102.0	0.99780	3.35	0.80	10.5	5
27	7.9	0.430	0.21	1.60	0.106	10.0	37.0	0.99660	3.17	0.91	9.5	5
40	7.3	0.450	0.36	5.90	0.074	12.0	87.0	0.99780	3.33	0.83	10.5	5
65	7.2	0.725	0.05	4.65	0.086	4.0	11.0	0.99620	3.41	0.39	10.9	5
...	...	...	...	...	...	...	...	...	...	...	...	...

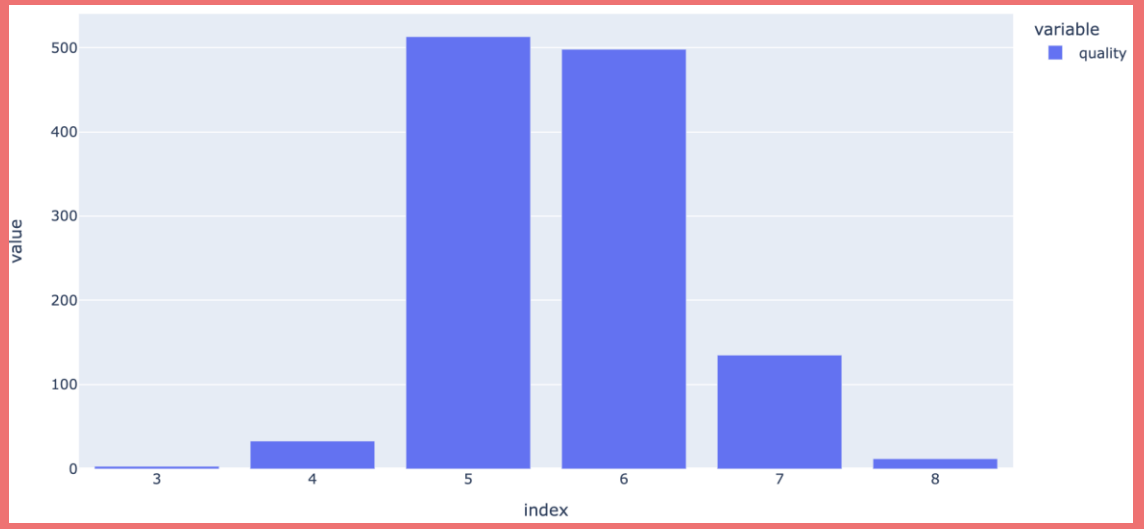
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000
mean	8.310596	0.529478	0.272333	2.523400	0.088124	15.893304	46.825975	0.996709	3.309787	0.658705	10.432315	5.623252
std	1.736990	0.183031	0.195537	1.352314	0.049377	10.447270	33.408946	0.001869	0.155036	0.170667	1.082065	0.823578
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996700	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.430000	2.600000	0.091000	21.000000	63.000000	0.997820	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

Reformatting ‘quality’ into a binary variable

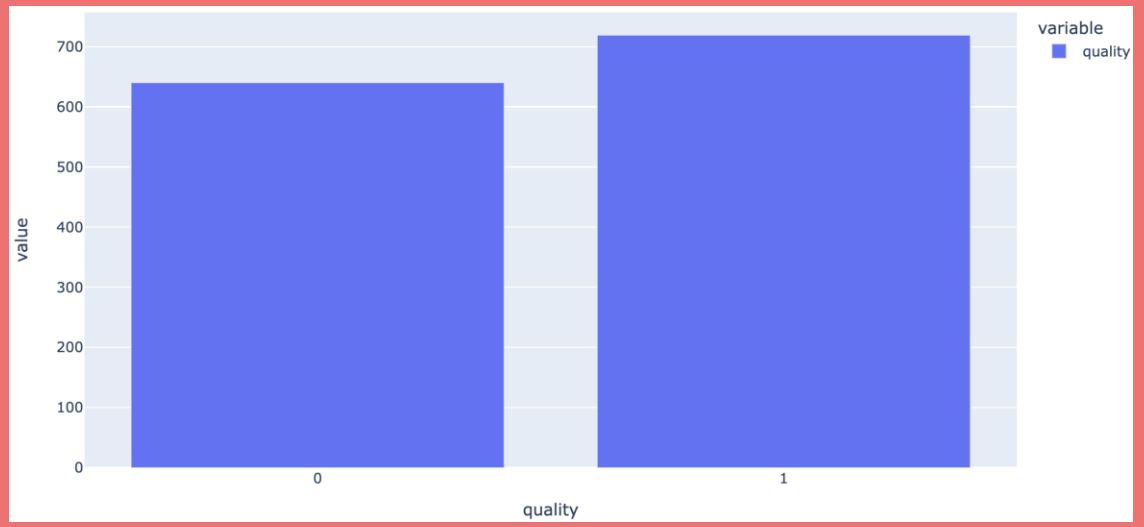
Quality scores below 3 or above 8 are not represented in this dataset, which makes it hard to predict underrepresented bins.

Hence, we fit the quality score into two bins, to consider only 'bad' and 'good' wines (0 and 1). This converts our problem into a binary classification problem.

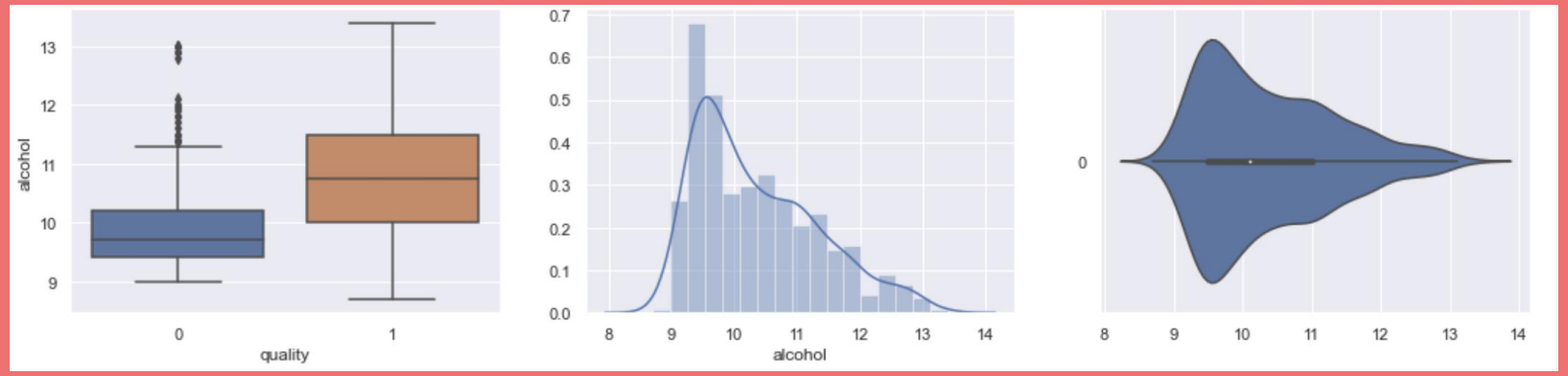
Before

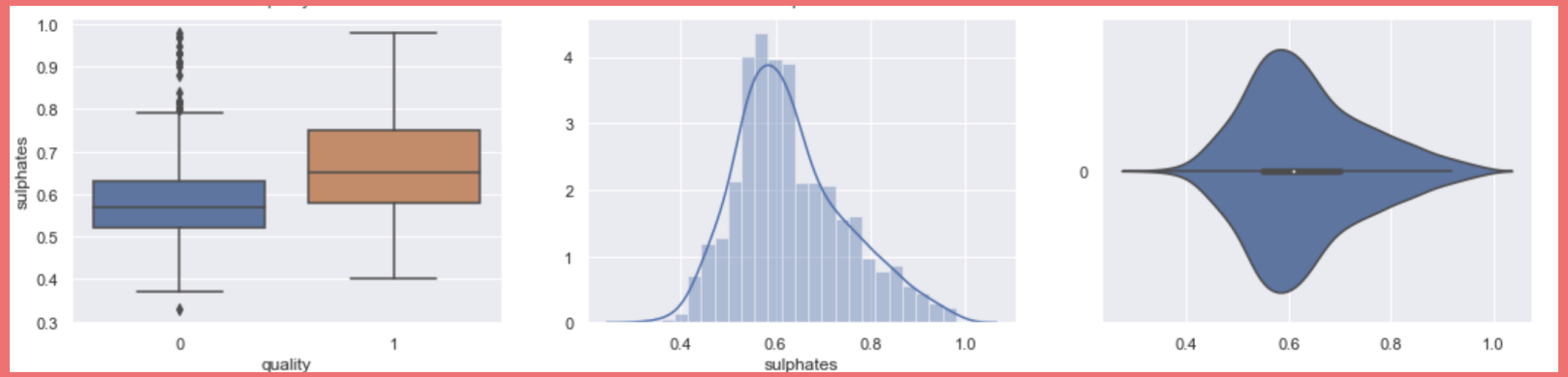


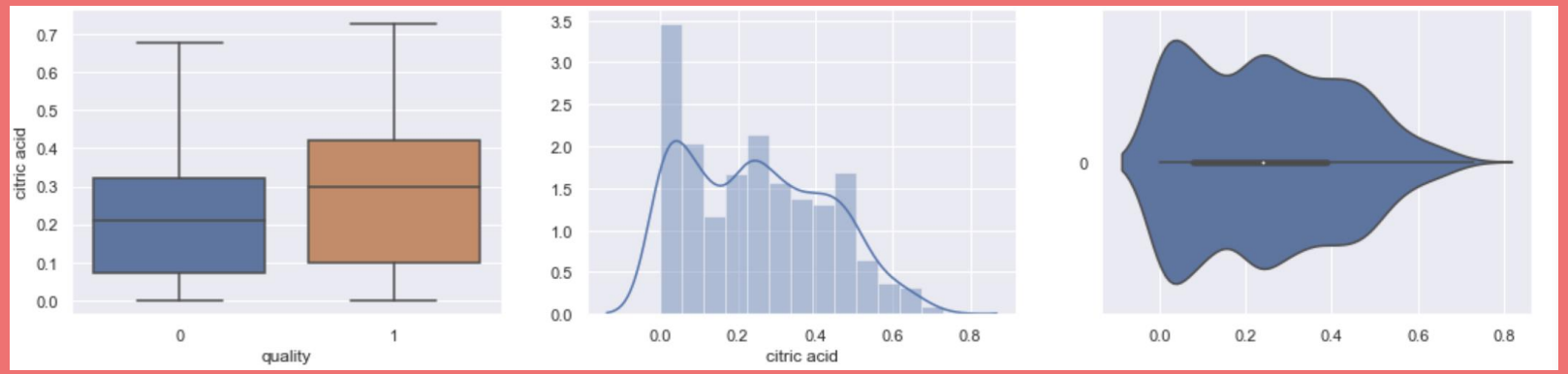
After



How are the other attributes distributed?





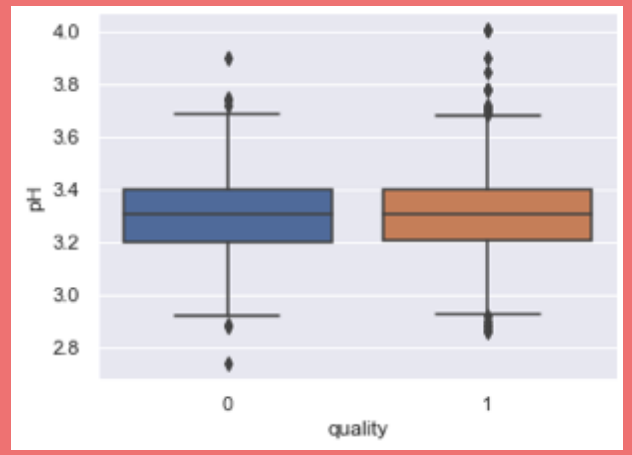


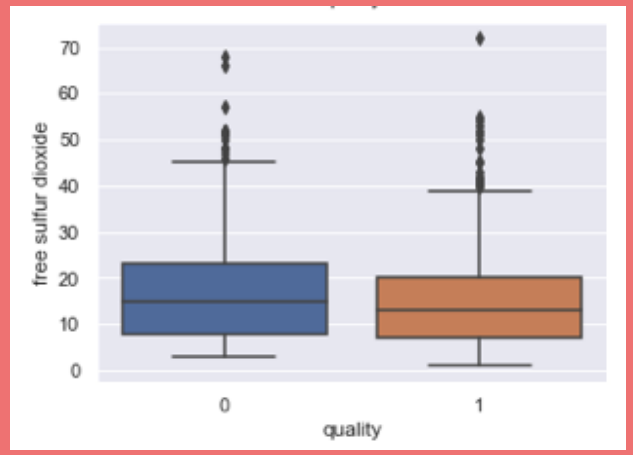
Other attempts

1. Feature Selection:

Removal of insignificant attributes

- Attributes Removed: pH and free sulphur dioxide
- When plotted against quality, there seems to be no correlation





- However, the ML accuracy **dropped** for: Decision Tree, Random Forest and CatBoost

Other attempts

2. Outlier Removal

Remove samples with predictor variable values beyond 1.5 * IQR of the 1st or 3rd quartile

```
# Cleaning dataset: Removing outliers
wine_predictors = wine_data.drop('quality', axis = 1)
wine_quality = wine_data['quality']


Q1_joint = wine_predictors.quantile(0.25)
Q3_joint = wine_predictors.quantile(0.75)
IQR_joint = Q3_joint - Q1_joint
idx_joint_cleaned = ~((wine_predictors < (Q1_joint - 1.5 * IQR_joint)) |
            (wine_predictors > (Q3_joint + 1.5 * IQR_joint))).any(axis=1)

wine_predictors_cleaned = wine_predictors.loc[idx_joint_cleaned]

wine_data_cleaned = wine_predictors_cleaned.join(wine_quality)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1019.000000	1019.000000	1019.000000	1019.000000	1019.000000	1019.000000	1019.000000	1019.000000	1019.000000	1019.000000	1019.000000	1019.000000
mean	8.169872	0.522507	0.250805	2.198822	0.078452	14.959764	42.388616	0.996547	3.322385	0.631237	10.385967	0.538763
std	1.475073	0.167364	0.182313	0.452214	0.014973	8.838951	26.625802	0.001616	0.132589	0.114861	0.992372	0.498740
min	5.100000	0.120000	0.000000	1.200000	0.039000	1.000000	6.000000	0.992350	2.940000	0.330000	8.700000	0.000000
25%	7.100000	0.390000	0.080000	1.900000	0.069000	8.000000	22.000000	0.995500	3.230000	0.550000	9.500000	0.000000
50%	7.800000	0.520000	0.240000	2.100000	0.078000	13.000000	36.000000	0.996560	3.320000	0.610000	10.100000	1.000000
75%	9.000000	0.630000	0.400000	2.500000	0.087000	20.000000	56.000000	0.997600	3.400000	0.700000	11.000000	1.000000
max	12.300000	1.010000	0.730000	3.650000	0.122000	42.000000	124.000000	1.001000	3.680000	0.980000	13.400000	1.000000

Other attempts

2. Outlier Removal

However, the ML accuracy **dropped** for: Decision Tree, Random Forest, AdaBoost, CatBoost and GradientBoost

Why?

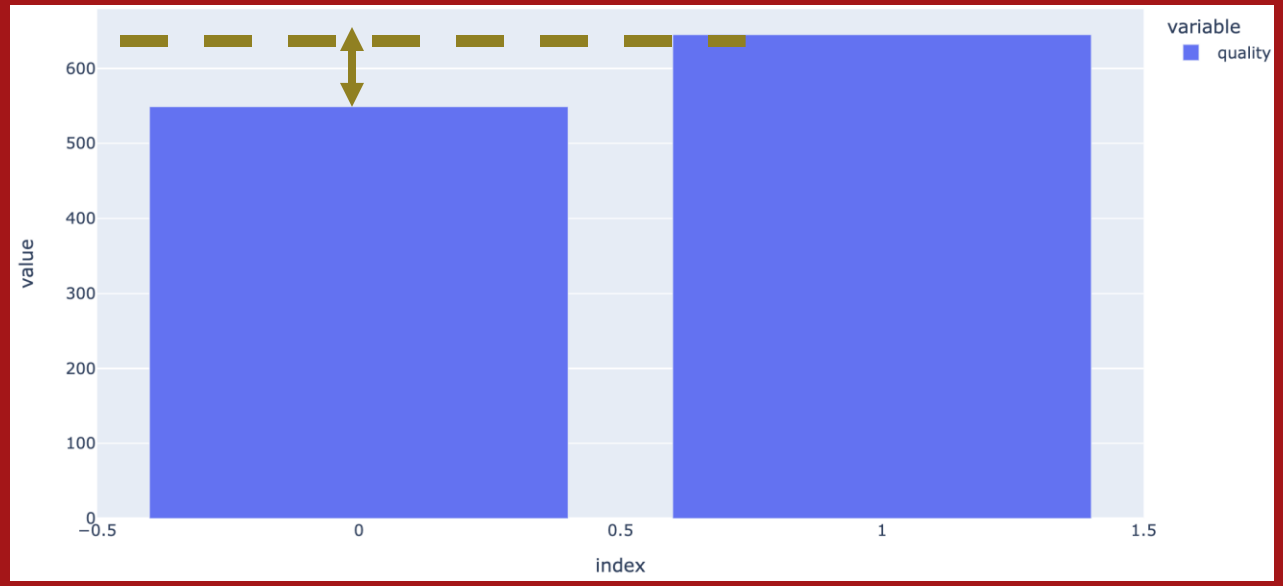
Outlying predictor values might have been justifiable, and fits into model trained.

Preparation for Machine Learning

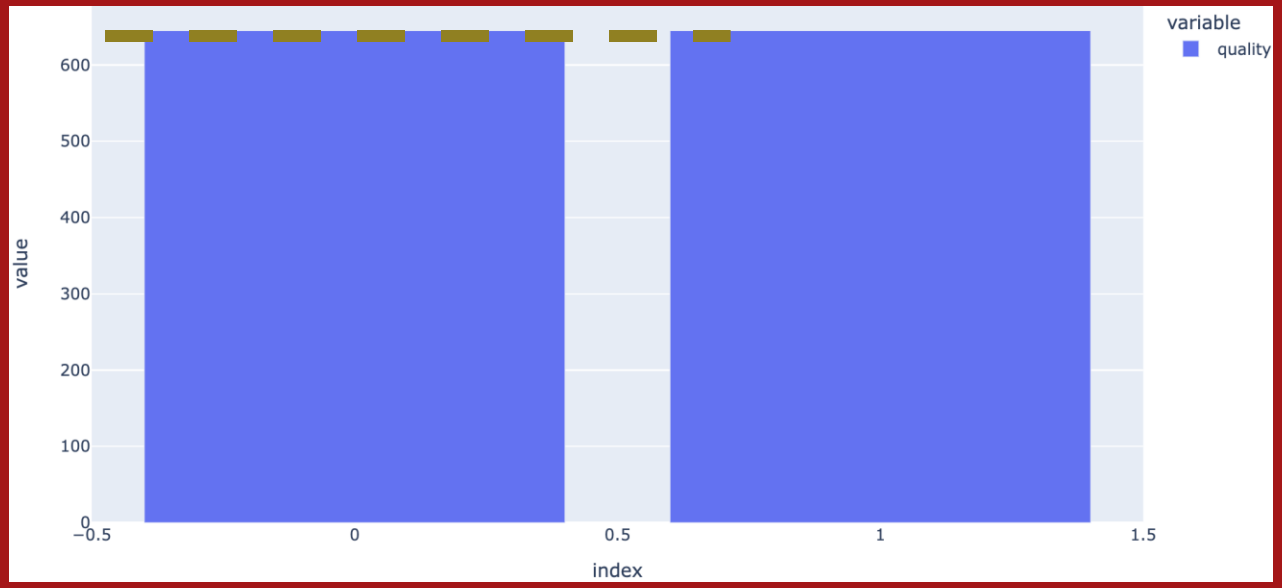
1. SMOTE
2. Train Test Split
3. Hyperparameter Tuning

SMOTE: Synthetic Minority Oversampling Technique

Purpose: To balance the counts of good and bad wine in the dataset



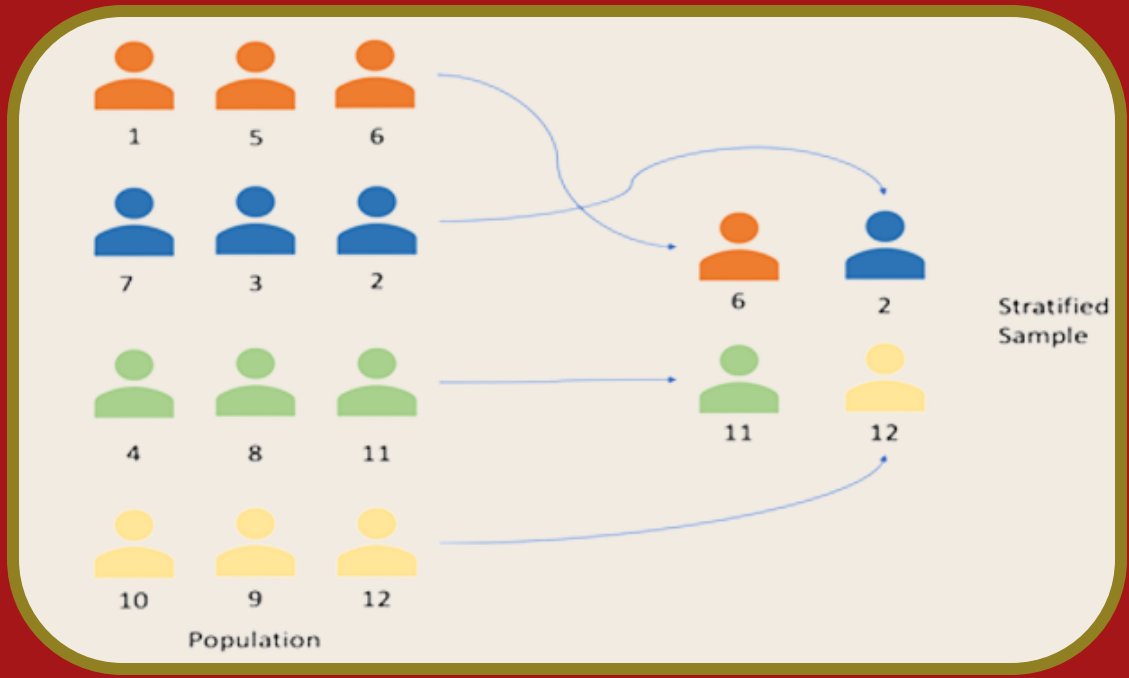
SMOTE does this by generating new instances from existing minority cases supplied



Train Test Split

train_test_split() function to split the dataset into train and test sets, with *test size* = 0.2

stratify = *y_smote*
Ensures consistent ratio of *good* : *bad* values throughout train and set sets



Hyperparameter Tuning: GridSearchCV

To determine the hyperparameters of the ML model that provide the highest accuracy

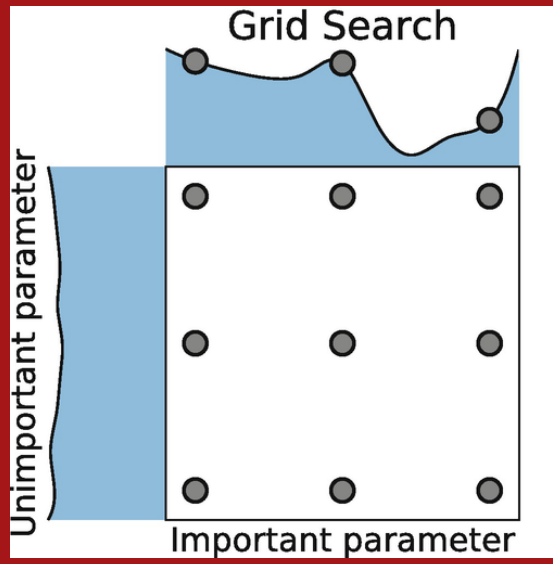
Includes 5-fold cross-validation to obtain the average unbiased results

```
param_grid = {'penalty':['l1', 'l2', 'elasticnet'],
              'C': np.logspace(-3, 3, 7),
              'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
             'random_state': [42]}

grid_search_logreg = GridSearchCV(LogisticRegression(), param_grid,
                  cv=KFold(5, shuffle=True, random_state=42), n_jobs=1)
grid_search_logreg.fit(X_train,y_train)
```

Grid Search looks through every combination of specified hyperparameter values

- Computationally expensive
- More optimal



Machine Learning Models

Mathematical Models

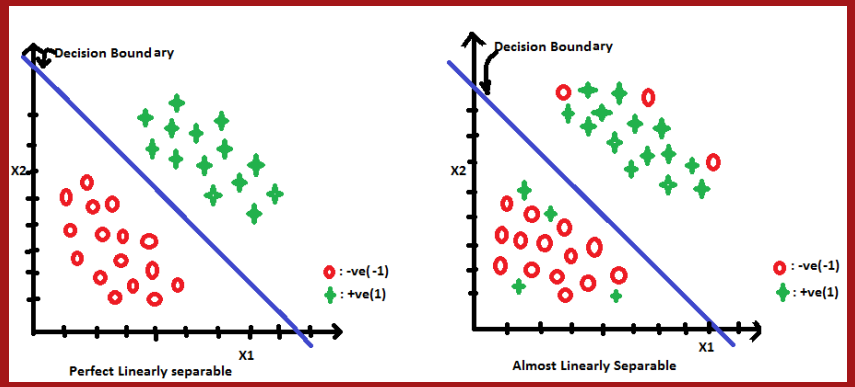
1. Logistic Regression
2. Gaussian Naïve Bayes

Tree-based ML Models

1. Decision Tree
2. Random Forest
3. AdaBoost
4. CatBoost
5. GradientBoost

1 Logistic Regression

Explanation

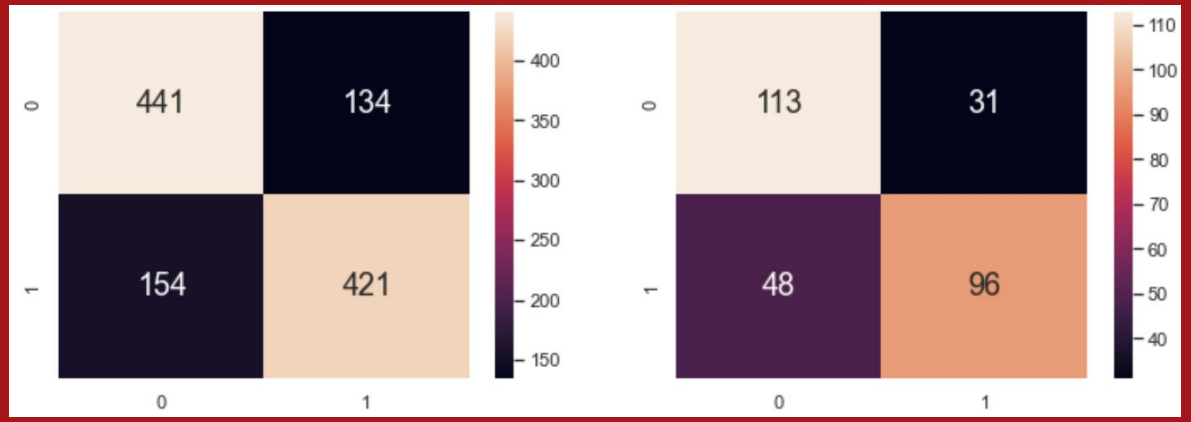


Models the probability of a discrete outcome given an input variable

For classification problems, with linearly separable variables

Results

```
Goodness of Fit of Model          Train Dataset
Classification Accuracy           : 0.7495652173913043
True Positive Rate                : 0.7321739130434782
False Positive Rate               : 0.23304347826086957
Goodness of Fit of Model          Test Dataset
Classification Accuracy           : 0.725694444444444
True Positive Rate                : 0.6666666666666666
False Positive Rate               : 0.2152777777777778
```



Train Set

Test Set

2 Gaussian Naïve Bayes

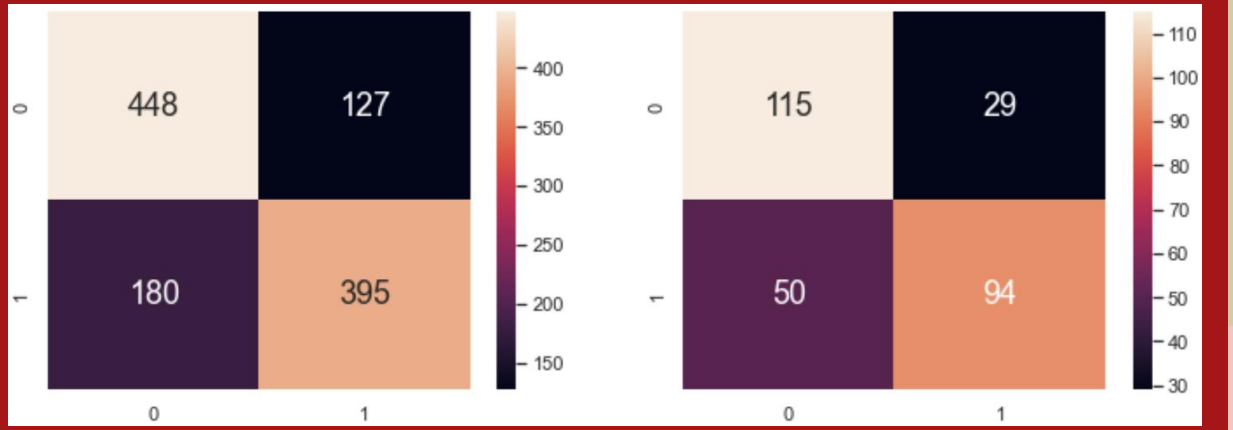
Explanation

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

GNB applies Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

Results

```
Goodness of Fit of Model        Train Dataset
Classification Accuracy         : 0.7330434782608696
True Positive Rate              : 0.6869565217391305
False Positive Rate             : 0.22086956521739132
Goodness of Fit of Model        Test Dataset
Classification Accuracy         : 0.725694444444444
True Positive Rate              : 0.6527777777777778
False Positive Rate             : 0.2013888888888889
```



Train Set

Test Set

3 Decision Tree

Explanation

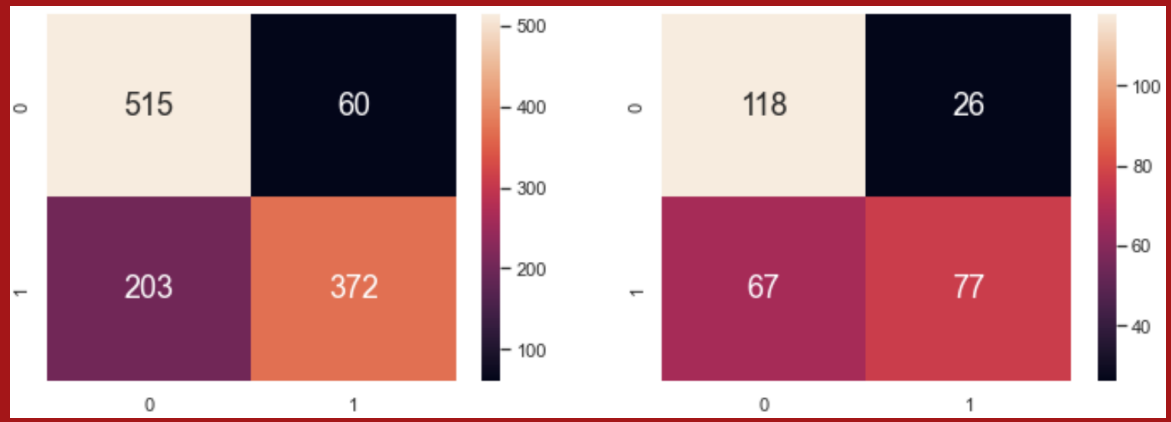
Use the dataset features to create yes/no questions.

Continually splits the dataset until all data points belonging to each class are isolated.

Can handle both categorical and numerical data (numerical data as predictors in this dataset)

Results

Goodness of Fit of Model	Train Dataset
Classification Accuracy	: 0.7713043478260869
True Positive Rate	: 0.6469565217391304
False Positive Rate	: 0.10434782608695652
Goodness of Fit of Model	Test Dataset
Classification Accuracy	: 0.6770833333333334
True Positive Rate	: 0.5347222222222222
False Positive Rate	: 0.18055555555555555

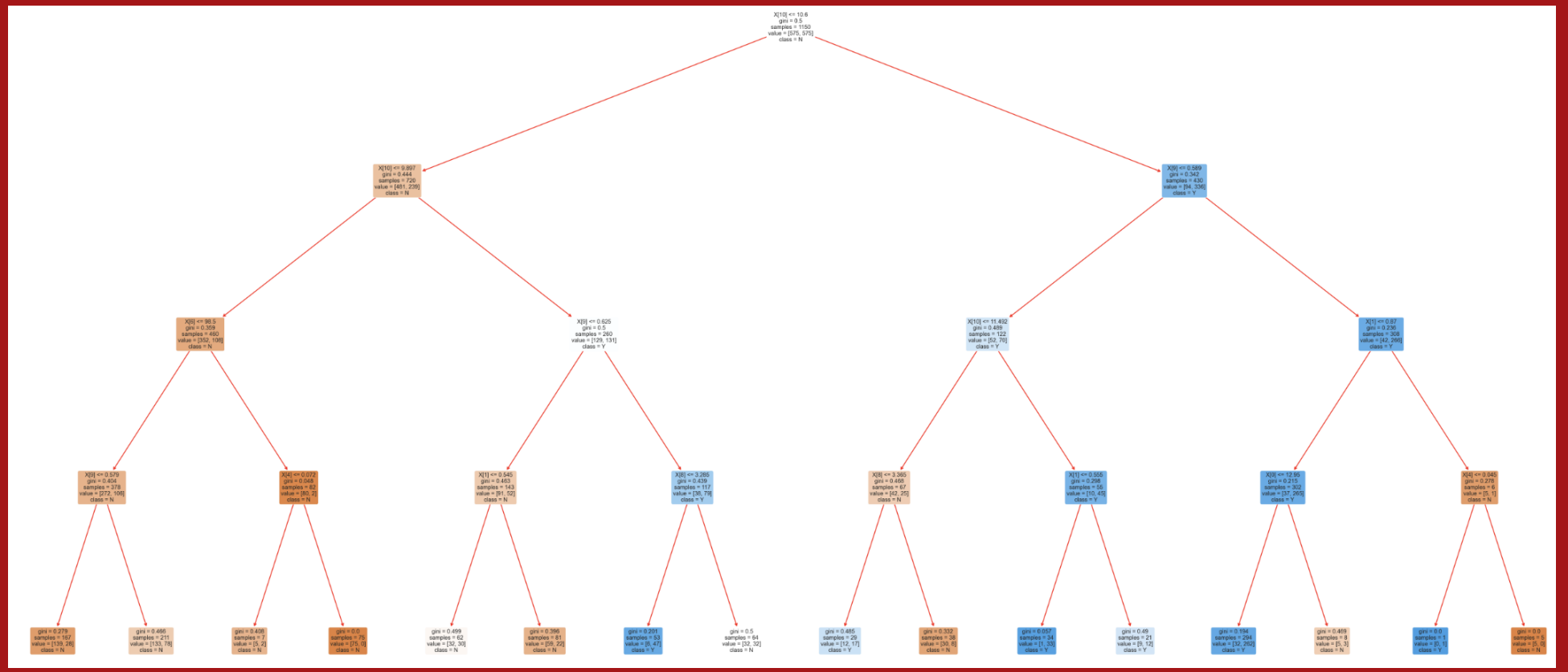


Train Set

Test Set

3 Decision Tree

Max depth 4 because it is the best hyperparameter from GridSearchCV



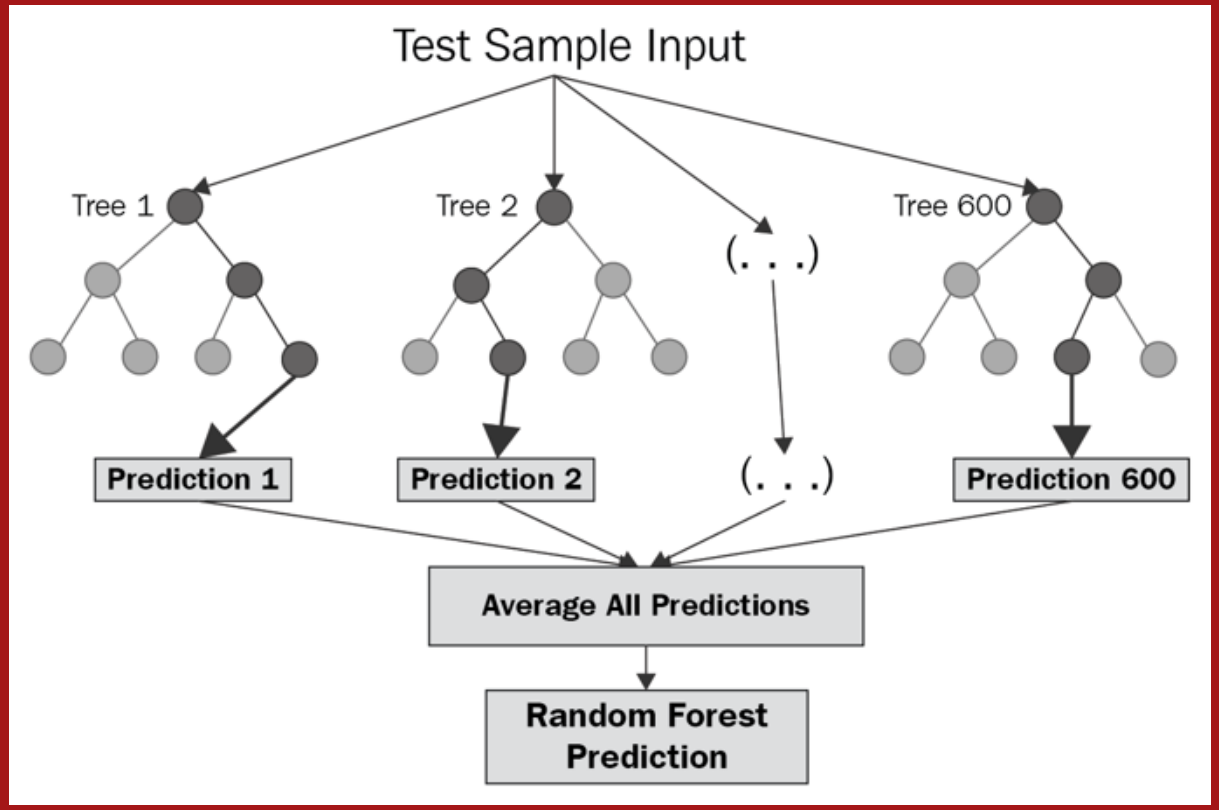
4 Random Forest

Explanation

Adds randomness when growing the trees

Multiple decision trees merged together

➡ More accurate & stable predictions

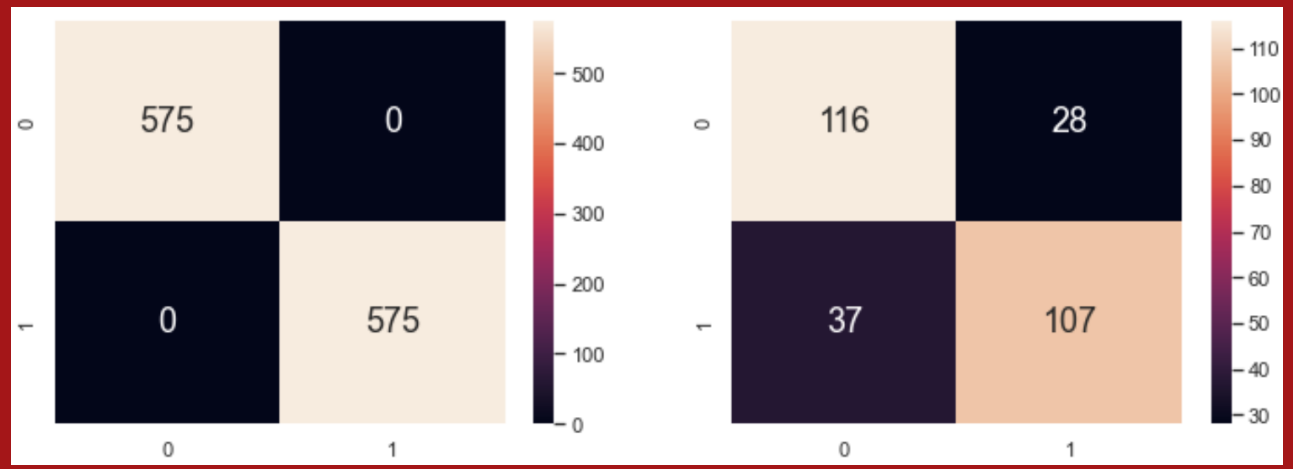


4 Random Forest

Results

Overfitted

```
Goodness of Fit of Model          Train Dataset
Classification Accuracy           : 1.0
True Positive Rate                : 1.0
False Positive Rate               : 0.0
Goodness of Fit of Model          Test Dataset
Classification Accuracy           : 0.7743055555555556
True Positive Rate                : 0.7430555555555556
False Positive Rate               : 0.19444444444444445
```



Train Set

Test Set

Boosting Classifiers

AdaBoost

CatBoost

Gradient Boost

Weighted ensemble method

A base learner takes assigns equal weight or attention to each observation.

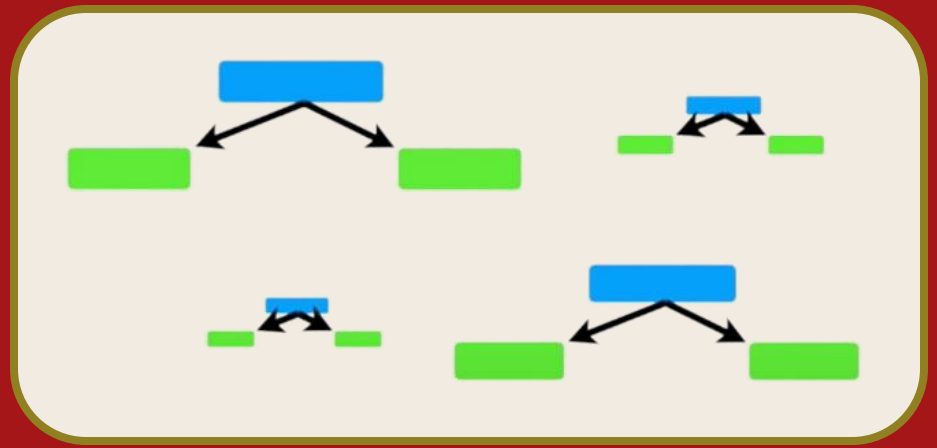
If there is any prediction error caused by first base learning algorithm → pay higher attention to it

These steps are iterated to get the highest accuracy possible

5 AdaBoost Classifier

Explanation

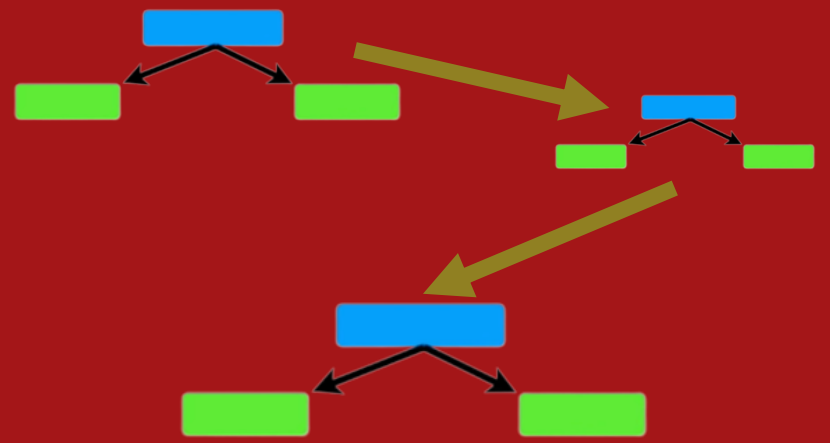
$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$



Combines multiple poorly performing ‘stumps’ to get a high accuracy strong classifier.

Each stump learns from previous stump's error

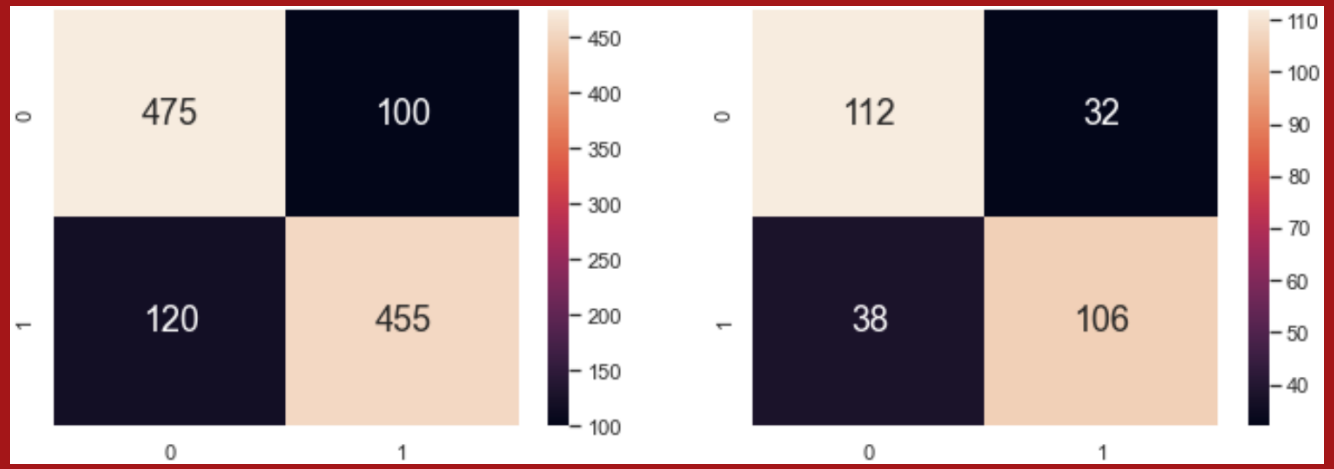
Stumps with higher accuracy get more say.



5 AdaBoost Classifier

Results

```
Goodness of Fit of Model          Train Dataset
Classification Accuracy           : 0.808695652173913
True Positive Rate                : 0.7913043478260869
False Positive Rate               : 0.17391304347826086
Goodness of Fit of Model          Test Dataset
Classification Accuracy           : 0.7569444444444444
True Positive Rate                : 0.7361111111111112
False Positive Rate               : 0.2222222222222222
```



Train Set

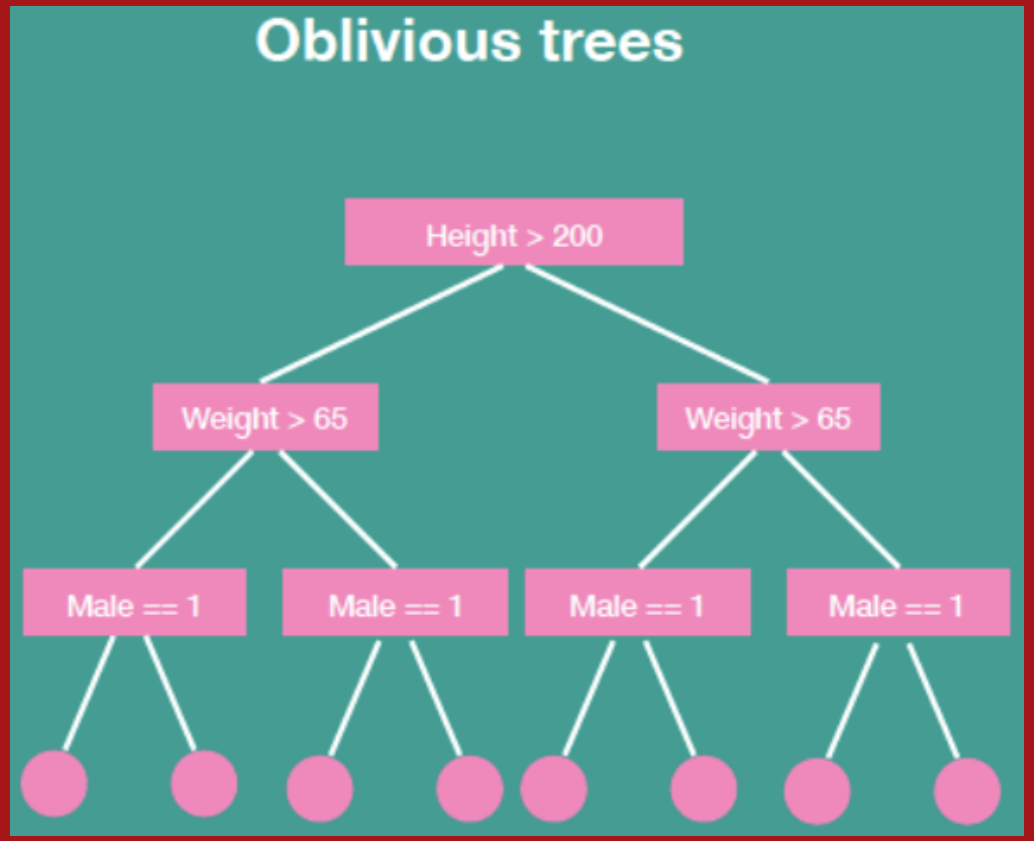
Test Set

6 CatBoost Classifier

Explanation

CatBoost grows oblivious trees
→ trees are grown by imposing the rule that all nodes at the same level, test the same predictor with the same condition.

Such trees are balanced.

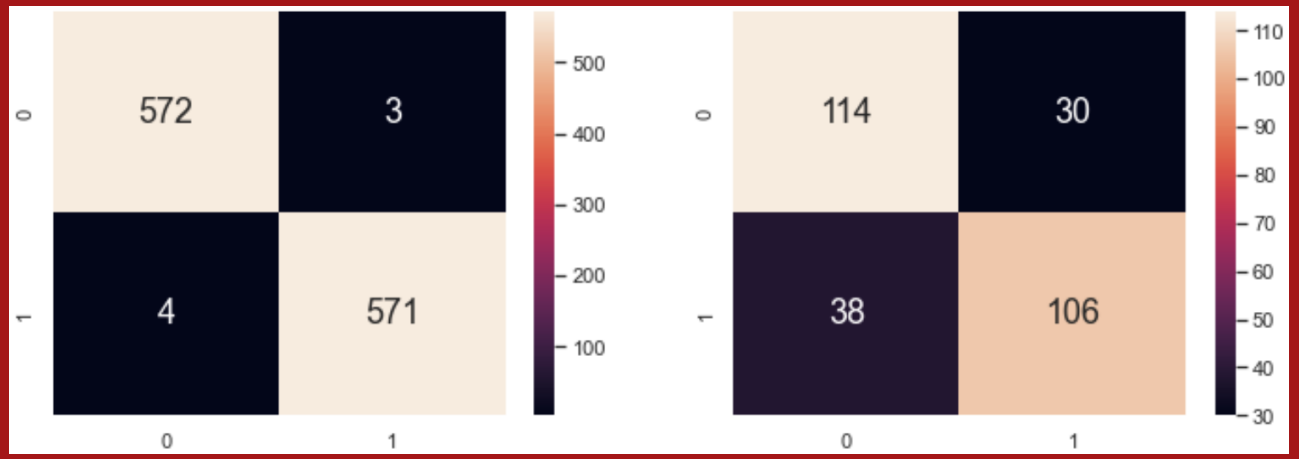


6 CatBoost Classifier

Overfitted

Results

```
Goodness of Fit of Model          Train Dataset
Classification Accuracy           : 0.9939130434782608
True Positive Rate                : 0.9930434782608696
False Positive Rate               : 0.005217391304347826
Goodness of Fit of Model          Test Dataset
Classification Accuracy           : 0.7638888888888888
True Positive Rate                : 0.7361111111111112
False Positive Rate               : 0.20833333333333334
```



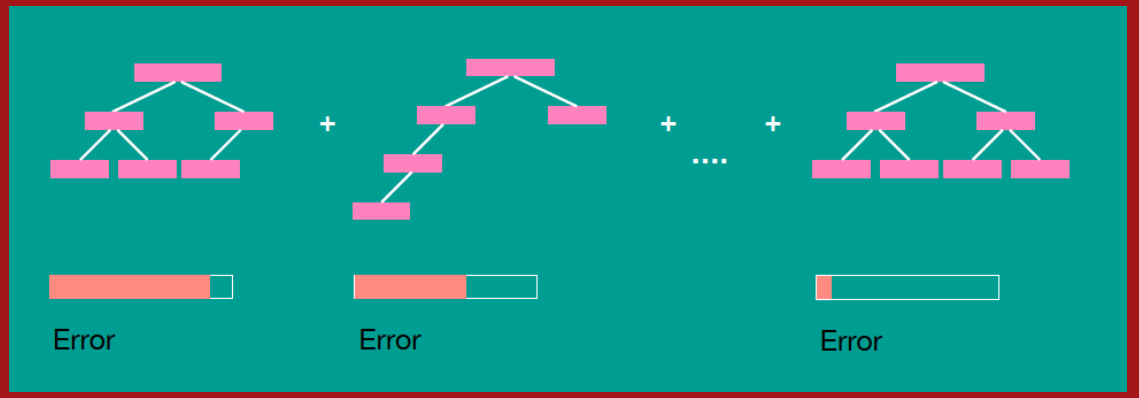
Train Set

Test Set

7 GradientBoost Classifier

Explanation

Uses gradient descent to minimize the loss function, or the difference between the actual class value of the training example and the predicted class value.



$$s = -\nabla F = \begin{pmatrix} -L'_z(y_1, a_{n-1}(x_1)), \\ \dots \\ -L'_z(y_\ell, a_{n-1}(x_\ell)) \end{pmatrix}$$

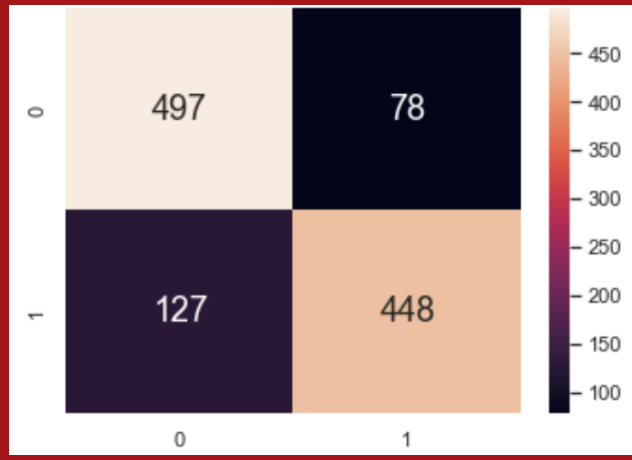
$$b_n(x) = \operatorname{argmin}_b \frac{1}{\ell} \sum_{i=1}^{\ell} (b(x_i) - s_i)^2$$

$$a_n(x) = \sum_{m=1}^{n} b_m(x)$$

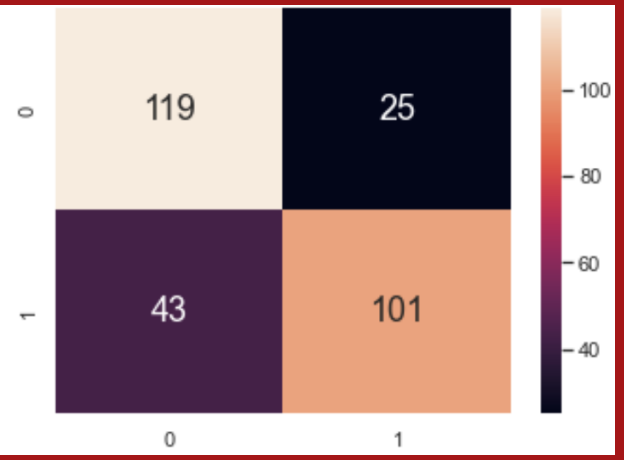
7 GradientBoost Classifier

Results

```
Goodness of Fit of Model          Train Dataset
Classification Accuracy           : 0.8217391304347826
True Positive Rate                : 0.7791304347826087
False Positive Rate               : 0.1356521739130435
Goodness of Fit of Model          Test Dataset
Classification Accuracy           : 0.7638888888888888
True Positive Rate                : 0.7013888888888888
False Positive Rate               : 0.1736111111111111
```



Train Set



Test Set

1 Decision Tree performed the worst

- Greedy and deterministic
- A single decision tree is sensitive to data variations → if you add or delete one more row the result can be different
- Prone to overfitting, especially when a tree is particularly deep.

2 Random Forest Classifier performed the best

It is an ensemble of single decision trees, adding additional randomness to the model, while growing the trees.

Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

- Much more robust than a single decision tree.
- Aggregate many decision trees to limit overfitting

3 Random Forest and CatBoost models were overfitted however…

Random Forest is an ensemble of decision trees and a single decision tree is very sensitive to data variations, hence it can easily overfit to noise in the data (as a result of SMOTE), especially since each tree is weighted equally.

Yet, these models still performed the best after cross-validation. This suggests that the data in the test set resembles that in the train dataset (likely due to stratifying) which still results in high accuracy for these models.

4 Boosting models performed worse than Random Forest

AdaBoost uses decision stumps, which has only 1 node and 2 leaves, instead of the conventional decision trees.

More weights are assigned to the incorrectly classified samples, such that the model learns from the mistakes of the previous stumps.

It emphasises on even the smallest error and might overfit to the noise. And it is likely that our dataset has substantial noisy data as a result of SMOTE.

Similarly, decision trees in gradient boosting suffer from the same disadvantage.

5 Not all data cleaning methods may be appropriate

Feature selection and *Outlier removal* was actually counter-productive for the ML models

Removing outliers without understanding the context of these outliers is not helpful. In this case, removing outliers resulted in valid data points being deleted.

Feature selection does not help in our case because our predictors are inherently weakly related to the response variable. Removing more attributes hence worsens the performance.

Conclusion

YES! We can predict the quality of wine given its physicochemical properties to a high accuracy

We have identified the ***Random Forest Classifier*** to be the most effective ML model to use in predicting wine quality

77.43% accuracy

We have identified the ***Decision Tree Classifier*** to be the least effective ML model to use in predicting wine quality

67.71% accuracy

References

https://archive.ics.uci.edu/ml/datasets/wine+quality

https://www.kaggle.com/code/firuzjuraev/red-wine-quality-forecasting/notebook#--Modeling!-

https://www.analyticsvidhya.com/blog/2021/04/distinguish-between-tree-based-machine-learning-algorithms/

https://towardsdatascience.com/the-right-way-of-using-smote-with-cross-validation-92a8d09d00c7

https://www.sciencedirect.com/science/article/abs/pii/S0167923609001377?via%3Dihub

https://blog.paperspace.com/adaboost-optimizer/#:~:text=AdaBoost%20is%20an%20ensemble%20learning,turn%20them%20into%20strong%20ones

https://towardsdatascience.com/introduction-to-gradient-boosting-on-decision-trees-with-catboost-d511a9ccbd14

Work Allocation

Timothy Teh:

Data Extraction, Data Cleaning, Data Visualisation

Chua Yong Xuan:

Implementation of Machine Learning Models: Logistic Regression, Gaussian Naive Bayes, Decision Tree

Lim Xin Yi:

Implementation of Machine Learning Models: Random Forest, AdaBoost, CatBoost, Gradient Boosting Classifier

THANK YOU!