Wine Quality Prediction

Group 11A

Alex Zelaya
I-Ching Lu
Kaidi Huang
Chaithanya Sai Krishna Bokka

Wine Quality Predictions - Outline

Abstract

The most frequently asked question about wine is how to evaluate the quality of wine. The reason why people are particularly concerned about this question is that the quality of wine is a characteristic of wine, which indicates the degree of its superiority or inferiority. For wine lovers, the judgment of wine quality directly determines their purchasing behavior and also provides a reference for the reasonable price of wine.

Our report will provide an in-depth analysis of red wine quality by applying machine learning techniques with the aim of identifying the main physicochemical characteristics that affect the quality of red wine. In this process, we collected data on red wine samples containing a variety of chemical profiles (e.g., acidity, sugar content, and alcohol percentage), as well as red wine quality scores associated with these chemical profiles. We used a variety of machine learning models, including Decision Tree and Random Forest, to classify and predict the red wine samples. During model training, we performed necessary preprocessing of the data, such as feature selection, data normalization, and cross-validation, to ensure the accuracy and generalization ability of the models.

The analysis results show that different models have different degrees of accuracy in predicting the quality of red wines (e.g. Corelation Heatmap). Among them, factors such as PH value and alcohol content have the most significant influence on the quality of red wines. By comparing the performance of the models before and after preprocessing, we found that preprocessing steps such as Standardization significantly improved the prediction accuracy of the models.
The results of this study provide an important basis for red wine producers to understand which chemical characteristics have the most significant impact on red wine quality, and also provide a scientific basis for consumers to choose red wines. Although the analyses in this study were based on a specific dataset and region, the methodology and conclusions have implications for broader research on red wine quality. Future research could extend such analysis to more regions and datasets to further validate and deepen our findings.

1. Introduction

1.1 Background

To meet the growing demand for diverse wine preferences, wineries must consider specific attributes to maintain a "fine" quality label. From developing wines that consist of a sour, yet fine taste to a sweet, but bitter flavor, wineries must treat their wines with delicacy to meet the expectations of the many wine enthusiasts. With the many different groups of wine enthusiasts around the world, wineries must consider what attributes they should focus on to gain a better rating from the public. If certain wines do not meet the expectations of the public based on certain attributes such as the acidity or the density of the sugars, then there is a likely chance that the wines would not be rated of high quality. Not only do wineries need to consider what attributes to put more concentration on than others, they also have to look into some combinations of attributes that would make it so that the wine made would have a high rating and acquire the label of “fine” quality. In the eyes, or in this case the taste, of the consumer, wineries must take in consideration of what makes a certain wine more “fine” compared to other variety of wines. As the consumption of wine becomes more frequent in professional and grand events, the wine industry must acclimate their production to meet the needs for different consumers. In order to acclimate the rising demands, wineries must take in consideration on certain attributes that contribute to the taste and flavor for wines to have a “fine” quality label. However, wine cannot be labeled as good quality unless the different features meet a criteria that matches the rating to be considered “good” quality.

1.2 Business Idea

Our research aims to develop an innovative system for assessing the quality of different wines by utilizing machine learning techniques. The core meaning of this business idea is to meet the growing consumer demand for high-quality wines while providing winemakers with a more accurate quality control tool. By analyzing the chemical properties of the variety of wines to predict their quality, this approach will not only help consumers to make more informed choices, but also provide the wine industry with a competitive advantage in the marketplace and scientific decision support. In addition,

the application of advanced data analytics to wine manufacturing will promote technological innovation and development in the industry, providing an example of modernization and transformation of a traditional industry.

1.3 Business Implications

The winemaking industry faces diverse challenges from brewing to distribution. By accurately predicting wine quality, winemakers can not only improve the marketability of their products, but also add value to their products while maintaining production costs. For example, if a winemaker can accurately predict which winemaking processes and component ratios will produce high-quality wines, they can avoid costly trial-and-error methods and go straight to the production strategies that are most likely to be successful.

Additionally, high-quality wines established through data analytics can also provide consumers with a more appealing product, which not only increases consumer willingness to buy, but also increases customer loyalty and brand reputation. On the marketing side, accurate quality prediction models enable marketing teams to more effectively promote products to targeted consumer segments and optimize advertising to maximize return on marketing investment. The ability to accurately predict wine quality also provides retailers with a significant business advantage. Retailers can use this information to optimize inventory management, reduce the risk of overstocking, and take a more targeted approach to their promotional campaigns and sales strategies. For example, they can highlight wines that are predicted to be of high quality during specific seasons or events, attracting more consumers and increasing sales.

In short, accurately predicting wine quality is not only critical for producers, but has far-reaching implications for every link in the entire supply chain. Through data-driven decision support systems, all players in the industry can gain valuable insights to improve efficiency, strengthen market positioning, and ultimately grow revenue. In the long run, wine quality prediction modeling is not only a technological advancement but also a leap forward in marketing and brand strategy.

2. DATA

2.1 Data Summary

We get the original wine dataset and utilize the `HEAD()` function and `SHAPE()` function to view the dataset. From our wine dataset, we have about 1,599 entries of different wines that consist of 12 different attributes. Each attribute consist of different ranges.

	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

2.2 Data Description

Within our wine dataset, it consist of 1,599 entries of different types of wine with each entry having a total of 12 different attributes, where the “quality” attribute is the classification column. Using the 11 different attributes, we are expecting the find that attributes that contribute the most toward developing the wine entry to have a quality of at least 7 or higher. The description of each attribute in the dataset is shown in the table below:

Attribute	Description	
fixed acidity	Fixed acidity is due to the presence of non-volatile acids in wine. For example, tartaric, citric or malic acid. This acid combines to balance the flavors of the wine, giving a fresh taste	

Attribute	Description	
volatile acidity	Volatile acidity, which is part of the acid in wine that can be smelled through the nose, is one of the most common defects. Excessive levels of acetic acid in wine can lead to unpleasant vinegar flavors	
citric acid	It can be used to acidify wines (increase acidity), collect wines, and clean filters to prevent fungal and mold infections. Small amounts of citric acid add "freshness" and flavor to wines	
residual sugar	Amount of sugar remaining after fermentation stops	
chlorides	Content of some minerals (e.g. salt) in wine	
free sulfur dioxide	The free form of sulfur dioxide exists in equilibrium between the molecules SO_2 (as a dissolved gas) and bisulfite ions; it prevents the growth of microorganisms and the oxidation of wine	
total sulfur dioxide	Amount of SO_2 in both free and bound forms; at low concentrations, SO_2 is barely detectable in wine, but at free SO_2 concentrations above 50 ppm, SO_2 becomes evident in the nose and flavor of the wine	
density	The density of wine can be less or more than that of water, and its value is mainly determined by the alcohol concentration and the sugar content.	
pH	Describe the acidity or alkalinity of the wine from 0 (very acidic) to 14 (very alkaline)	
sulphates	A wine additive that promotes sulfur dioxide gas (SO_2) levels, acts as an antimicrobial agent and antioxidant	
alcohol	Percentage of alcohol content of wines	
quality	A number of wine experts have rated these wines based on their personal sensory perception, with scores between 0 (very poor) and 10 (very good), and are only used as a reference for the final prediction of results	

Gain overall information from dataset: We get the original wine dataset and utilize the `info()` function to view the dataset and the columns. Based on our function, we can see that there are no missing values found in each column, making the general process of our data take less time.

```
wine_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
wine_df.shape
```

```
(1599, 12)
```

Find the NULL value: The number of nulls for all features columns is 0, indicating that there are no existing Null values within our dataset.

```
#null values check

wine_df.isna().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
```

```
# unique features in every column

wine_df.nunique()
```

```
fixed acidity           96
volatile acidity       143
citric acid             80
residual sugar          91
chlorides              153
free sulfur dioxide     60
total sulfur dioxide   144
density                436
pH                      89
sulphates               96
alcohol                 65
quality                  6
dtype: int64
```

Looking into the research, it is known that the three common characteristics of wine are acidity, alcohol content, and sweetness.

The Acidity can be considered in two ways. On the one hand, acids can be categorized into fixed acidity and volatile acidity, the total content of which reflects the concentration of the wine. Citric acid and volatile acidity are good indicators of the health of a wine, helping us to judge the quality of the wine and predict how difficult it will be to stored. On the other hand, pH (acidity and alkalinity) is also an indicator of acidity, reflecting the richness of a wine's flavor on the palate.

In addition, sulfur dioxide, as an acidic oxide, has an effect on acidity as well as sterilizing and anti-oxidizing the wine, and there are two related fields in this dataset, free sulfur dioxide and total sulfur dioxide.

The alcohol attribute is the amount of alcohol in a wine, where most wines have an alcohol content of between 10-15%.

sweetness is determined by the residual sugar in the wine. In some instances, the sweetness of the wine can alter whether a wine can be considered a fine quality for some people. If the Sweetness happens to be too high, consumers

may feel uninterested in wines that have a sugar level of candy, but some consumers would not enjoy the most bitter of wines either.

The density of a wine is mainly determined by the alcohol and sugar content, which are also important indicators of alcohol and sweetness.

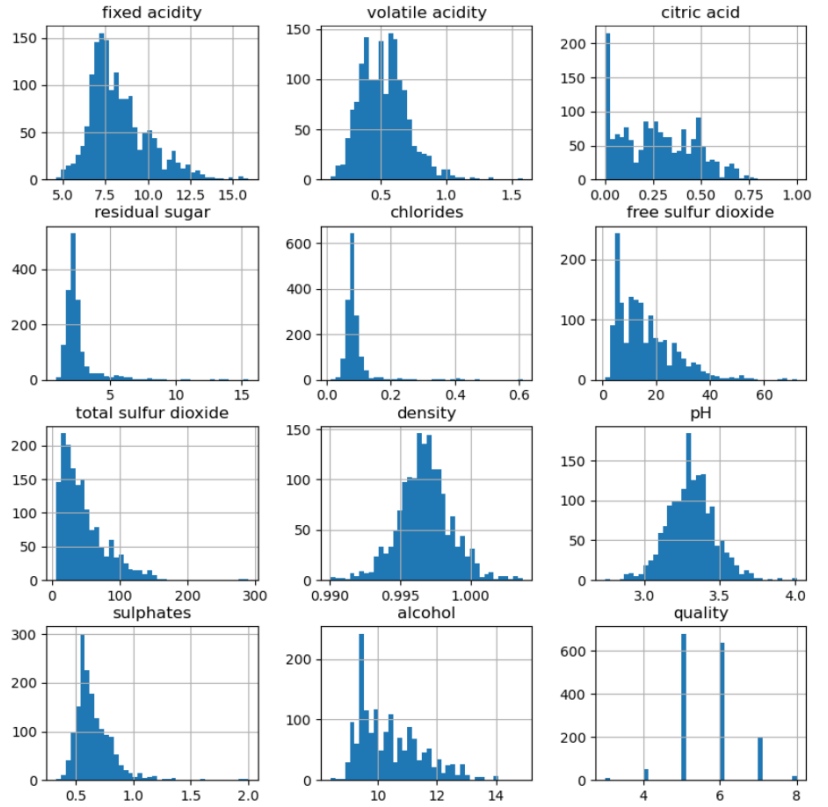
Chlorides and sulphates are mineral salts found in wine, usually at levels of 0.1-0.4g/L and 0.25-0.85g/L, respectively.

2.3 Data Visualization

Histogram

A histogram is a type of statistical reporting chart that consists of a series of vertical bars or lines of varying heights representing the distribution of data, with the horizontal axis representing the type of data and the vertical axis representing the distribution. Within our dataset, we decided to look into the distribution of the dataset and separate them into different graphs. We can see than some attributes have a slightly even distribution, whereas other attributes have a heavily skewed data points.

```
## Visulaization of data column wise heat map
wine_df.hist(bins=40, figsize=(10, 10))
plt.show()
```



This set of histograms reveals the distribution of various physicochemical properties in the wine data set. Most of the properties show some degree of right skewness, indicating that there are relatively few extremes of high values, while most of the wine property values are concentrated in the lower range. In particular, the contents of volatile acidity, residual sugars, chlorides, and sulphates show a significant right-skewed distribution on the histograms, suggesting that for these

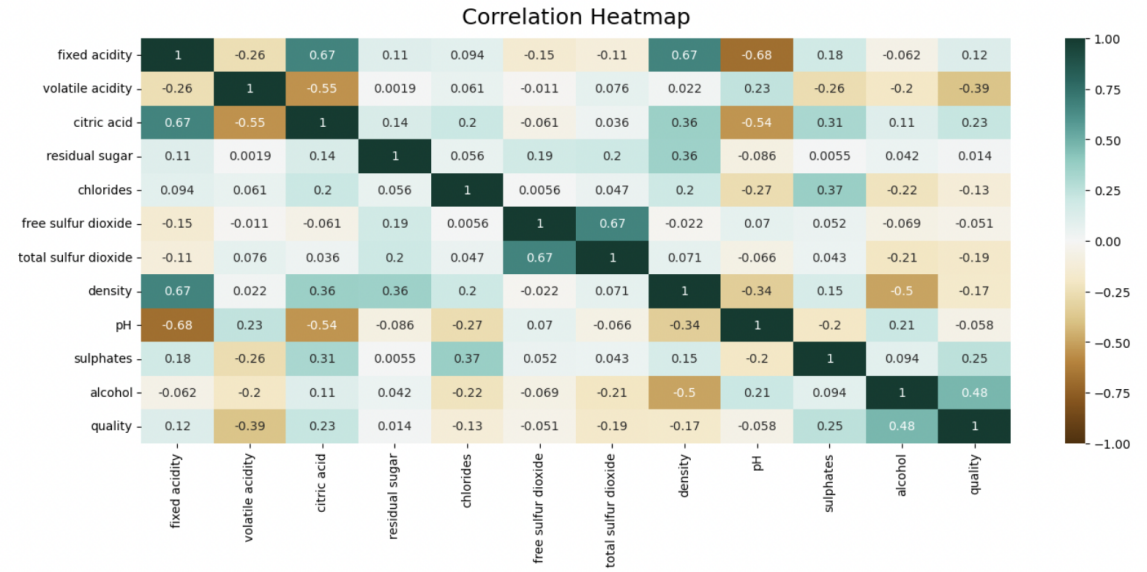
metrics, the majority of red wines tend to have values concentrated in the lower ranges, while high values are more sparsely sampled. The distribution of total sulfur dioxide is particularly right-skewed, implying that most red wines are low. The alcohol content is also right-skewed, but the quality scores of the wines are slightly left-skewed, suggesting that most wines tend to be scored at medium or quality levels, with some high-quality levels, and very fun low qualities. This distribution information is critical to understanding which chemical indicators have the greatest impact on red wine quality and how winemaking processes can be improved.

In the case for our “quality” attribute, we can see a small issue with the current dataset prior to pre-processing. Within the histogram of the “quality”.

Heat Map

A heat map is a statistical chart that displays data by coloring blocks of color. Using a heat map, we can visualize the similarity between two attributes in a dataset based on the colors in the map, as measured by the Pearson correlation coefficient. We plot the heat map as follows:

```
## identifying how strong the columns are corellated

plt.figure(figsize=(16, 6))
heatmap = sns.heatmap(wine_df.corr(), vmin=-1, vmax=1, annot=True, cmap='BrBG')
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':18}, pad=12);
# save heatmap as .png file
# dpi - sets the resolution of the saved image in dots/inches
# bbox_inches - when set to 'tight' - does not allow the labels to be cropped
plt.savefig('heatmap.png', dpi=300, bbox_inches='tight')
```



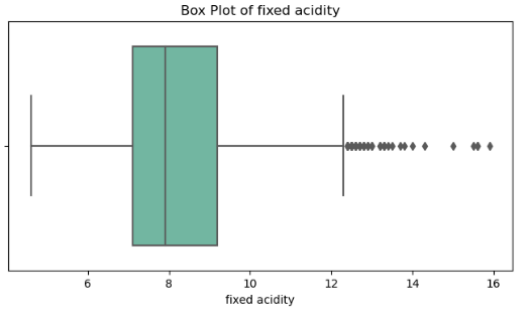
This heat map shows the correlation between the different chemical features in the wine data set. Positive correlation values (close to +1) imply a direct correlation between properties, while negative correlation values (close to -1) indicate an inverse correlation between properties. For example, the high positive correlation between fixed acidity and citric acid (0.67) content suggests that as fixed acidity increases, citric acid content also tends to increase. Conversely, the high negative correlation between fixed acidity and pH (-0.68) implies that fixed acidity increases as pH decreases (acidity increases).

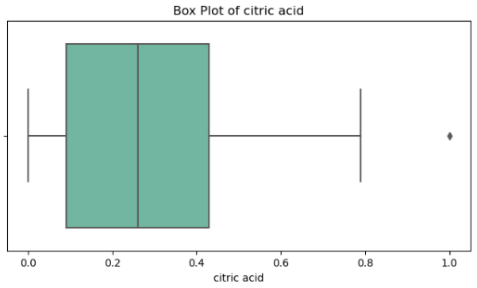
In the case of the correlation between quality and the highest correlation attribute, alcohol, whenever there is a high amount of alcohol content in the wine, the more likely it is to receive a high-quality rating of 7 or higher. The next highest correlations for quality would be the sulphates and the citric acid in the wine. Although the top three attributes that have a positive correlation with quality may play a factor to make a certain wine receive a high quality label, wineries would still need to consider other attributes that can cause negative impacts to their ratings as well. Overall, the heat map provides wineries with an insight into which chemical characteristics have a significant impact on the quality of the final product, which is essential for improving the production process and increasing the quality of wine.

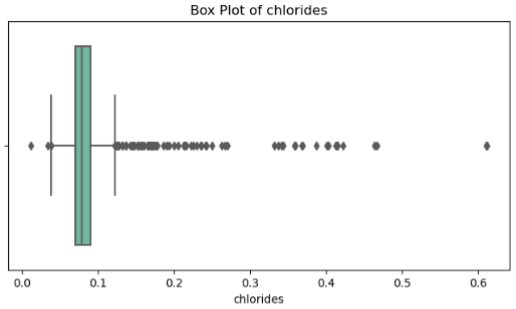
Box Plot

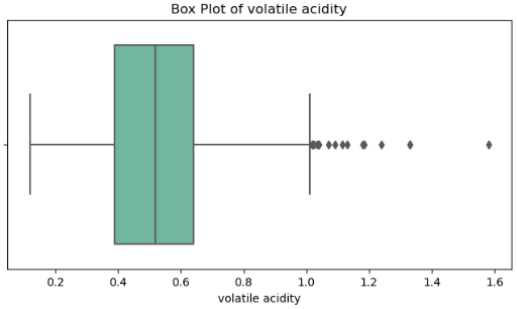
The box plot is used to reflect the center position and scattering range of one or more groups of continuous quantitative data distribution. It is not only able to analyze the level differences between different categories of data at various levels, but also reveals the degree of dispersion, outliers, distributional differences, etc. between the data. Based on the wine dataset, we use the box plots with all the attributes:

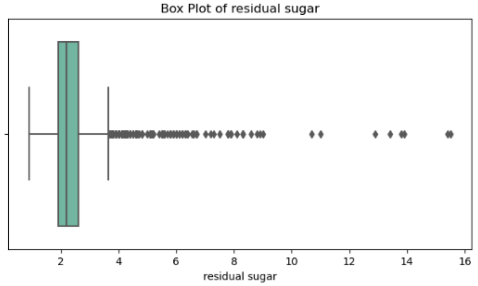
```
##updated plots after data cleaning

columns_to_plot = wine_df.columns
for col in columns_to_plot:
    plt.figure(figsize=(8, 4))
    plt.title(f'Box Plot of {col}')
    sns.boxplot(data=wine_df, x=col, palette="Set2")
    plt.show()
```

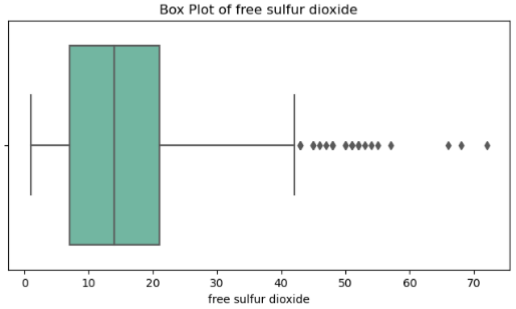

Box Plot of fixed acidity
6
8
10
12
14
16
fixed acidity

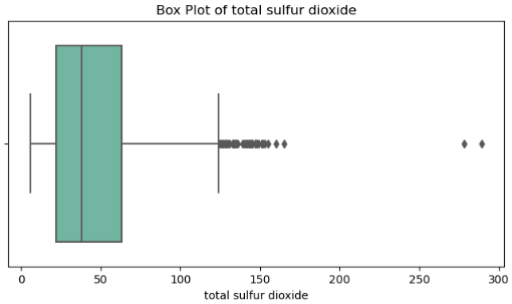

Box Plot of citric acid
0.0
0.2
0.4
0.6
0.8
1.0
citric acid

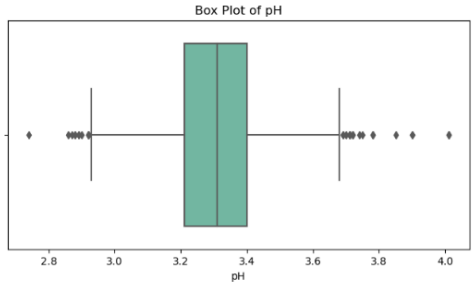

Box Plot of chlorides
0.0
0.1
0.2
0.3
0.4
0.5
0.6
chlorides

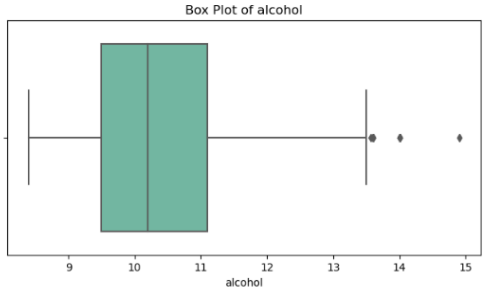

Box Plot of volatile acidity
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
volatile acidity

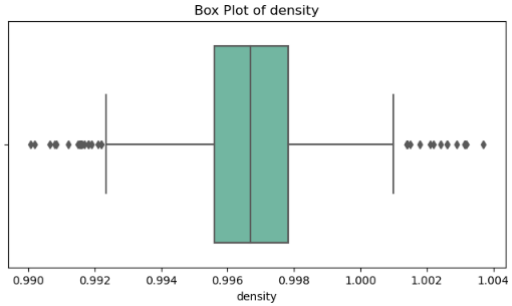

Box Plot of residual sugar
2
4
6
8
10
12
14
16
residual sugar

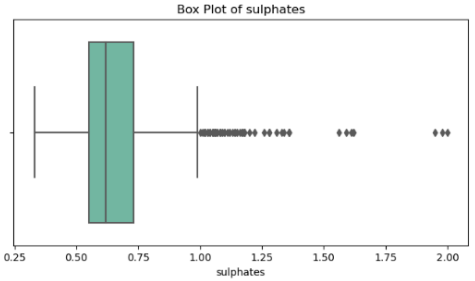

Box Plot of free sulfur dioxide
0
10
20
30
40
50
60
70
free sulfur dioxide

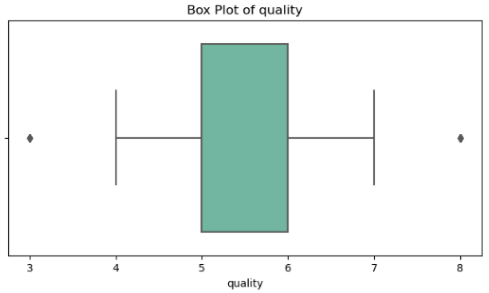












This set of box plots provides an important perspective on the distribution and variability of individual chemical attributes in the wine dataset. The medians, interquartile ranges, and outliers shown in each plot reveal the degree of variability and concentration trends for each chemical component. For example, the box plots for volatile acidity, citric acid, and alcohol content appear to be more compact, suggesting that these attributes are relatively consistent across the majority of wine samples, whereas the box plots for residual sugar and total sulfur dioxide show wider interquartile ranges, implying that these attributes are more variable across wine samples. The large number of outliers, especially evident in the box plots for chloride and sulfate, may indicate that there was some specific winemaking process or condition that led to the outliers for these chemical constituents. For red wine producers, it is critical to understand the distributional characteristics of these variables, as they are directly related to the control of the winemaking process and the quality and flavor of the final product.

3. Analysis

Model baseline accuracy without pre processing steps.

The baseline accuracy of a model serves as a benchmark or reference point for evaluating the performance of more complex models. We would initially run the dataset using without pre-processing to find the accuracy of certain models using an uncured dataset. Regardless if our dataset is considered clean, the baseline model would include possible errors such as outliers, duplicates, and other potential issues that could cause for the accuracy to skew in either direction. The end goal of any machine learning model is to outperform this baseline.

In the baseline model we developed, we choose to include the ‘fixed acidity', 'volatile acidity', 'sulphates', 'density','citric acid’ five variables to build the model. Those are manually picked within the 11 attributes, and we got the accuracy of 0.8531 for Support Vector Machines, 0.9093 for Random forest model, 0.85 for logistic regression and 0.83 for KNN model respectively.

```
wine_df['good_quality'] = [1 if x >= 7 else 0 for x in wine_df['quality']]

## dividing train and test samples data

X = wine_df[['fixed acidity', 'volatile acidity', 'sulphates', 'density','citric acid']]
y = wine_df.good_quality
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state= 42)


classifiers = {
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC()
}
```

```
## list to store the model accuracy

baseline_models_accuracy = []
```

```
## model execution before pre processing

for clf_name, clf in classifiers.items():
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    baseline_models_accuracy.append((clf_name, accuracy))
```

```
# sorting of accuracy between two baseline models

baseline_models_accuracy.sort(key=lambda x: x[1])
```

```
# model accuracy before pre processing

esults_df_baseline_models = pd.DataFrame(baseline_models_accuracy, columns=["Method Name", "Accuracy"])
esults_df_baseline_models
```

	Method Name	Accuracy
0	SVM	0.853125
1	Random Forest	0.909375

```
wine_df['good_quality'] = [1 if x >= 7 else 0 for x in wine_df['quality']]

## dividing train and test samples data

X = wine_df[['fixed acidity', 'volatile acidity', 'sulphates', 'density','citric acid']]
y = wine_df.good_quality
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state= 42)


classifiers = { 'Logistic Regression': LogisticRegression(),
                'KNN': KNeighborsClassifier()}
```

```
## list to store the model accuracy

baseline_models_accuracy = []
```

```
## model execution before pre processing

for clf_name, clf in classifiers.items():
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    baseline_models_accuracy.append((clf_name, accuracy))
```

```
# sorting of accuracy between two baseline models

baseline_models_accuracy.sort(key=lambda x: x[1])
```

```
## model accuracy before pre processing

results_df_baseline_models = pd.DataFrame(baseline_models_accuracy, columns=["Method Name", "Accuracy"])
results_df_baseline_models
```

	Method Name	Accuracy
0	KNN	0.831250
1	Logistic Regression	0.853125

3.1 The Pre-processing

The outliers: As demonstrated from earlier, some attributes have a significant amount of outliers within the box plot. Given the size of or wine dataset, we decided to keep the data with considerable amount of outliers since out dataset is neither excessively large nor excessively small. Since the count of our dataset isn't as high as a large dataset, removing the data that are found to have outliers could cause the data to skew in a certain direction. We would also want to keep the outliers while conducting our machine learning models because in some attributes, we can consider the outliers as

extreme cases. In addition to possibly skewing our data after removing the outliers, our machine-learning models could also shift in terms of their accuracy, causing some models to become less accurate than what we initially desired.

Finding and removing the duplicate values: Now that we conducted our baseline models, we then proceed to commit to pre-processing our data to see whether we can acquire a more accurate machine learning method compared to the baseline models. We start with checking our wine dataset if there are any rows of data that have the same data entries within all 11 attributes. After running the dataset for duplicates, we found 240 rows of data that had exact entries to the initial data entry, so we will drop those duplicates. Once we dropped the 240 rows of data, we were then left with 1,359 rows of data for our new wine dataset. It's important to note that the total count of attributes mentioned in our dataset is 13, as we would need to keep the ‘quality’ column to display the rating of the data entry. We would also move onto creating a new column called ‘good_quality’ to represent if the data entry has reached a rating of 7 or higher. Therefore, the primary dataset contains 1,359 instances with 11 attributes, while the additional two columns are utilized for dataset classification, where the ‘quality’ attributes represent our rating system for a wine entry and the ‘good_quality’ column represents the classification if a wine entry has a rating of 7 or higher.

```
## drop duplicates

wine_df[wine_df.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
...	...	...	...	...	...	...	...	...	...	...	...	...
1563	7.2	0.695	0.13	2.00	0.076	12.0	20.0	0.99546	3.29	0.54	10.1	5
1564	7.2	0.695	0.13	2.00	0.076	12.0	20.0	0.99546	3.29	0.54	10.1	5
1567	7.2	0.695	0.13	2.00	0.076	12.0	20.0	0.99546	3.29	0.54	10.1	5
1581	6.2	0.560	0.09	1.70	0.053	24.0	32.0	0.99402	3.54	0.60	11.3	5
1596	6.3	0.510	0.13	2.30	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6

240 rows × 12 columns

```
wine_df.drop_duplicates(inplace=True)
wine_df.reset_index(drop=True, inplace=True)
wine_df.shape
```

```
(1359, 13)
```

Standardizing the data

Standardizing the data helps to ensure that all features are on a similar scale, preventing certain features from dominating the learning algorithm due to differences in magnitude. Although the data in each column does not have any major differences from the initial ranges, we would want to standardize the data to ensure that our machine learning model runs efficiently.

```
scaler = StandardScaler()
X_train_standardized = scaler.fit_transform(X_train)
X_test_standardized = scaler.transform(X_test)
```

```
X_train_standardized
```

```
array([[-0.01635802, -0.08787126,  4.18848373, ...,  0.30846774,
         1.44826134,  0.08564777],
       [-0.02185189, -0.63309979, -0.19559237, ..., -0.84752686,
         0.57075393, -0.89443913],
       [ 0.14931504,  0.44953976, -0.61310619, ..., -0.11174228,
        -1.44146465,  0.41234341],
       ...,
       [ 1.22710838, -0.66164379, -0.09944751, ...,  1.55686845,
         1.86153565, -0.13129216],
       [-0.40292849, -0.92384394,  0.55484811, ...,  0.46220312,
         0.66015426, -0.74594111],
       [ 0.70155857, -0.80441927, -0.93753794, ..., -0.88041916,
         0.39745189, -1.04293715]])
```

```
X_test_standardized
```

```
array([[ 0.4054996 , -0.86666778,  1.22725902, ..., -0.75238627,
         0.70042114, -0.15321049],
       [-1.03618357, -1.12485224, -1.8696734 , ..., -2.94080244,
        -0.21028453,  1.7082294 ],
       [ 0.9542488 , -1.24075783,  2.02840949, ...,  0.39538004,
         0.76318693, -0.53804389],
       ...,
       [ 0.78975255, -1.03593313, -0.60051663, ..., -0.77125713,
         0.48781413, -0.97988943],
       [-0.56628401,  0.46604241, -0.07692497, ..., -0.74842117,
        -0.40388465, -0.54835922],
       [-0.12680672, -0.3267206 ,  0.03575731, ...,  0.75942484,
         0.34491142,  0.53114182]])
```

```
results_standardized = []
```

3.2 Using SMOTE to balance the data

After removing the duplicates and standardizing the data, we then move on to creating synthetic data for the minority classification to match the majority classification using SMOTE. Since the dataset shows a classification of “bad wines”, or 0, with an entry of 1175, whereas the classification of “good wines”, or 1, has only 184 entries, there is a major difference

in the two classifications that we would need to make an accurate prediction for our “good” wines. With this imbalance, we would need to use SMOTE to bring an even distribution of both good and bad-labeled wines.

```
In [74]: ## different wine quality

         wine_df['good_quality'].value_counts()
```

```
Out[74]: good_quality
         0    1175
         1     184
         Name: count, dtype: int64
```

3.3 Feature selection

With the information provided by the heat map, we can see the correlation between quality and ‘fixed acidity', 'volatile acidity', 'sulphates', 'alcohol', 'density','citric acid','total sulfur dioxide are the highest seven attributes. We put these into SMOTE to remove the bias.

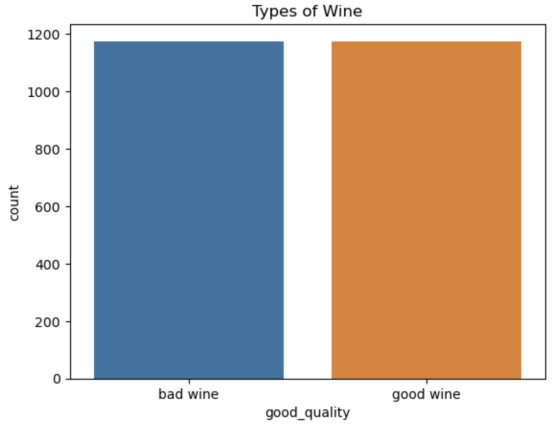
```
## balacing the skewness in wine quality

from imblearn.over_sampling import SMOTE, ADASYN
#parameter for requires seed
random_value = 2000

X = wine_df[['fixed acidity', 'volatile acidity', 'sulphates', 'alcohol', 'density','citric acid','total sulfur dioxide']]
y = wine_df.good_quality
oversample = SMOTE()
X_ros, y_ros = oversample.fit_resample(X, y)
sns.countplot(x=y_ros)
plt.xticks([0,1], ['bad wine','good wine'])
plt.title("Types of Wine")
plt.show()
```

X is the feature matrix containing selected red wine characteristics such as fixed acidity, volatile acidity, sulfate, alcohol, density, citric acid, and total sulfur dioxide; **y** is the target variable indicating the quality of the red wine, where it is assumed that wine_df.good_quality is a binary variable labeling the red wine as good or bad.

We used the SMOTE algorithm, which is an oversampling technique to solve the sample imbalance problem in classification problems. It balances the category distribution by generating synthetic samples from a few categories.

`oversample = SMOTE()` : creates an instance of SMOTE.

`X_ros, y_ros = oversample.fit_resample(X, y)` : Applying the SMOTE algorithm to the features and target variables generates the oversampled datasets X_ros and y_ros.



The graph shows that after SMOTE oversampling, the previously unbalanced wine quality labels are balanced into the same number of "good" and "bad" wines, which helps to train more unbiased and accurate machine learning models. Because models trained on unbalanced datasets may favor the majority class at the expense of the minority class, SMOTE solves this problem by creating a synthetic sample of the minority class.

`sns.countplot(x=y_ros)` plots the distribution of the oversampled target variable using Seaborn's countplot function, showing the newly balanced "bad wine" and "good wine" numbers "The distribution of the target variable after oversampling was plotted using Seaborn's countplot function.

Data Preparation

Classify data

```
## Intializing model classifiers after pre processing the data

classifiers = {
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC()
}
```

In [68]:
```
results_standardized = []
for clf_name, clf in classifiers.items():
    clf.fit(X_train_standardized, y_train)
    y_pred = clf.predict(X_test_standardized)
    accuracy = accuracy_score(y_test, y_pred)
    results_standardized.append((clf_name, accuracy))
```

In [69]:
```
results_standardized.sort(key=lambda x: x[1])
```

In [70]:
```
## results standardizing the data

##results after standardizing the data

results_df_standardized = pd.DataFrame(results_standardized, columns=["Method Name", "Accuracy"])
results_df_standardized
```

Out[70]:

	Method Name	Accuracy
0	Logistic Regression	0.814894
1	KNN	0.870213
2	SVM	0.880851
3	Random Forest	0.917021

It defines a dictionary in Python called classifiers, which contains several different machine learning models and their corresponding initialization objects. The dictionary facilitates the selection of the best model for the dataset classification task by referring to and comparing different classifiers during training, training each model, and evaluating its performance.

3.4 K-Nearest Neighbors (KNN): 0.83—>0.88

Definition:

K-Nearest Neighbors is a non-parametric and lazy supervised learning algorithm used for classification and regression. In KNN, an object is classified by a majority vote of its k nearest neighbors, where k is a positive integer (usually small).

Advantages:

1. Simple and Intuitive: KNN is easy to understand and implement, making it suitable for quick prototyping and simple scenarios.
2. No Training Phase: KNN is a lazy learner, meaning it does not have a separate training phase. The model is built during prediction time.
3. Versatile: KNN can be applied to both classification and regression tasks.
4. Non-Parametric: KNN makes no assumptions about the underlying data distribution.

Disadvantages:

1. Computationally Expensive: The prediction time complexity is higher, especially as the size of the dataset increases.
2. Sensitivity to Noise: KNN is sensitive to noisy data and outliers.
3. Curse of Dimensionality: KNN's performance can degrade in high-dimensional spaces due to the curse of dimensionality.

4. Need to Choose K: The choice of the parameter k is crucial and may impact the model's performance.

3.5 Logistic Regression: 0.85—>0.82

Definition:

Logistic Regression is a parametric statistical model used for binary classification. Despite its name, it is used for classification, not regression. Logistic Regression models the probability that a given instance belongs to a particular category.

Advantages:

1. Efficient and Fast: Logistic Regression is computationally efficient and can handle large datasets with relatively low computational cost.
2. Outputs Probabilities: Provides probabilities for class membership, allowing for nuanced interpretation.
3. Regularization: Can be regularized to prevent overfitting.
4. Interpretable: The coefficients in logistic regression can be interpreted in terms of the log-odds of the features.

Disadvantages:

1. Assumes Linearity: Logistic Regression assumes a linear relationship between the independent variables and the log-odds of the dependent variable.
2. Limited Expressiveness: Logistic Regression may not perform well when the relationship between features and target is highly non-linear.
3. Sensitive to Outliers: Logistic Regression can be sensitive to outliers.
4. Not Suitable for Complex Relationships: It may not capture complex relationships in the data as effectively as more complex models.

Several reasons why a logistic regression model may not fit the data well. Here are some common reasons:

1. **Non-Linear Relationships:**
    - Logistic regression assumes a linear relationship between the features and the log-odds of the target variable. If the true relationship is non-linear, logistic regression may not fit the data well.
2. **Incorrect Model Assumptions:**
    - Logistic regression assumes that the errors are binomially distributed and that there is no multicollinearity (high correlation) among the independent variables. If these assumptions are violated, the model may not be a good fit.
3. **Outliers:**
    - Outliers in the data can disproportionately influence the logistic regression model, leading to a poor fit. Consider identifying and addressing outliers during data preprocessing.
4. **Model Complexity:**
    - Logistic regression is a linear model and may not be suitable for highly complex relationships in the data. In such cases, more flexible models like decision trees or ensemble methods might be more appropriate.
5. **Optimization Issues:**
    - If the optimization algorithm used to train the logistic regression model doesn't converge or converges to a suboptimal solution, it can affect the fit. Adjusting hyperparameters or trying a different optimization algorithm may be necessary.
6. **Data Distribution Changes:**

- If the distribution of the data changes over time or between training and testing sets, the logistic regression model may not generalize well. Monitoring and accounting for distribution shifts are important.

3.6 SVC Model

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
y_pred = svc.predict(X_test_standardized)
print(metrics.accuracy_score(y_test, y_pred))
```

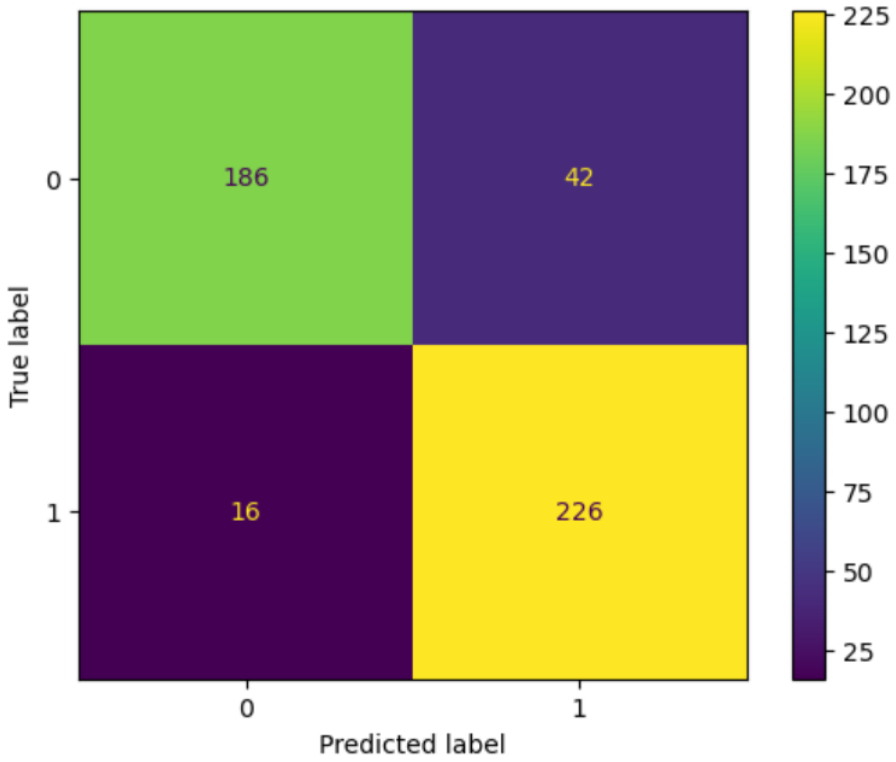
```
0.8765957446808511
```

The accuracy of this model was about 0.8766, or the model correctly predicted about 87.66% of the test data. The code demonstrates that the model's prediction accuracy is quite high, which means that the SVC model was able to differentiate between good and bad wine with a high degree of accuracy on this particular test set.

This process demonstrates that in real-world machine learning applications, finding the optimal parameters for a model is an iterative process, and that the performance of the final model may not be significantly improved by parameter fine-tuning, especially if the model is already relatively well-optimized.

Confusion Matrix

```
Model accuracy after pre processing and gridsearch 0.8765957446808511
```



The matrix shows the performance of the model in distinguishing between high quality (labeled 1) and low quality (labeled 0) red wines：

- True-negative example (TN, green area): the model correctly identifies 186 low-quality red wine samples, which means that the actual quality of these samples is low and predicted by the model to be low.

- False Positive Example (FP, blue area): The model incorrectly predicts 42 low-quality wine samples as high-quality, which may result in consumer expectations of these wines not matching the actual experience.
- True Example (TP, yellow area): The model correctly identifies 226 high-quality red wine samples, which suggests that the actual quality of these samples is high, as predicted by the model.
- False Negative Example (FN, purple area): The model incorrectly classifies 16 red wines that are actually high quality as low quality, which could mean that some good wines are not being recognized as they should be.

The accuracy is 87.66%, which indicates that our model can reliably distinguish high-quality and low-quality red wines in most cases, but there is still a certain percentage of misclassifications. These misclassifications may have an impact on wine sales and brand reputation, so further optimization of the model to reduce these misclassifications is key to improving the performance of the red wine quality classification system.

```
from sklearn.metrics import classification_report
cr = classification_report(y_test, y_pred_best)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.92      0.82      0.87       228
           1       0.84      0.93      0.89       242

    accuracy                           0.88       470
   macro avg       0.88      0.87      0.88       470
weighted avg       0.88      0.88      0.88       470
```

Precision: Precision is the percentage of samples predicted to be in the positive class that are actually in the positive class. In this example, for the negative class (0), the precision is 0.92, which means that 92% of the samples predicted by the model to be in the negative class are actually in the negative class. For the positive class (1), the precision is 0.84, which means that 84% of the samples predicted by the model to be positive calss are actually positive class.

Recall: Recall is the percentage of samples that are positive classes that are correctly predicted by the model to be positive classes. Recall is 0.82 for negative classes and 0.93 for positive classes, which means that the model captures 82% of the actual negative samples and 93% of the actual positive samples.

F1-Score: The F1-score is the reconciled average of precision and recall, and is a composite of precision and recall. An F1-score of 0.87 for the negative classes and 0.89 for the positive classes usually indicates that the model performs reasonably well in predicting both types of samples.

Support: Support refers to the actual number of samples in the test data for each category. The negative classes has 228 samples and the positive classes has 242 samples.

The recall reflects the model's ability to recognize "good wines" (positive class) and "bad wines" (negative class):
A positive recall of 0.93 means that almost all high-quality wines are correctly identified by the model. In other words, if there are 242 bottles of red wine that are actually good wine, then the model is able to correctly identify 231 bottles; a negative class recall of 0.82 indicates that the model is able to identify most, but not all, of the low-quality red wines. If there were 228 bottles of red wine that were actually bad wine, then the model was able to correctly identify about 186 bottles.
Thus, a high recall indicates that there are very few false negative (FN) cases, i.e., very few samples that are actually in the positive category are incorrectly predicted to be in the negative category. For wine quality assessment, a high recall means that the model does a good job of capturing wines that are actually of high quality, reducing the risk of missing good wines.

The way to Improve the accuracy of SVM

Hyperparameter tuning is a crucial step in optimizing the performance of a machine learning model. To perform hyperparameter tuning for SVM, you can use techniques like Grid Search. C (Regularization Parameter) controls the trade-off between having a smooth decision boundary and classifying the training points correctly. A smaller C leads to a larger margin but may misclassify some points, while a larger C aims to classify all points correctly. The choice of kernel determines the type of decision boundary. Common choices include 'linear', 'poly' (polynomial), and 'rbf' (radial basis function). The kernel parameter interacts with other hyperparameters, and its selection can significantly impact the model's performance.

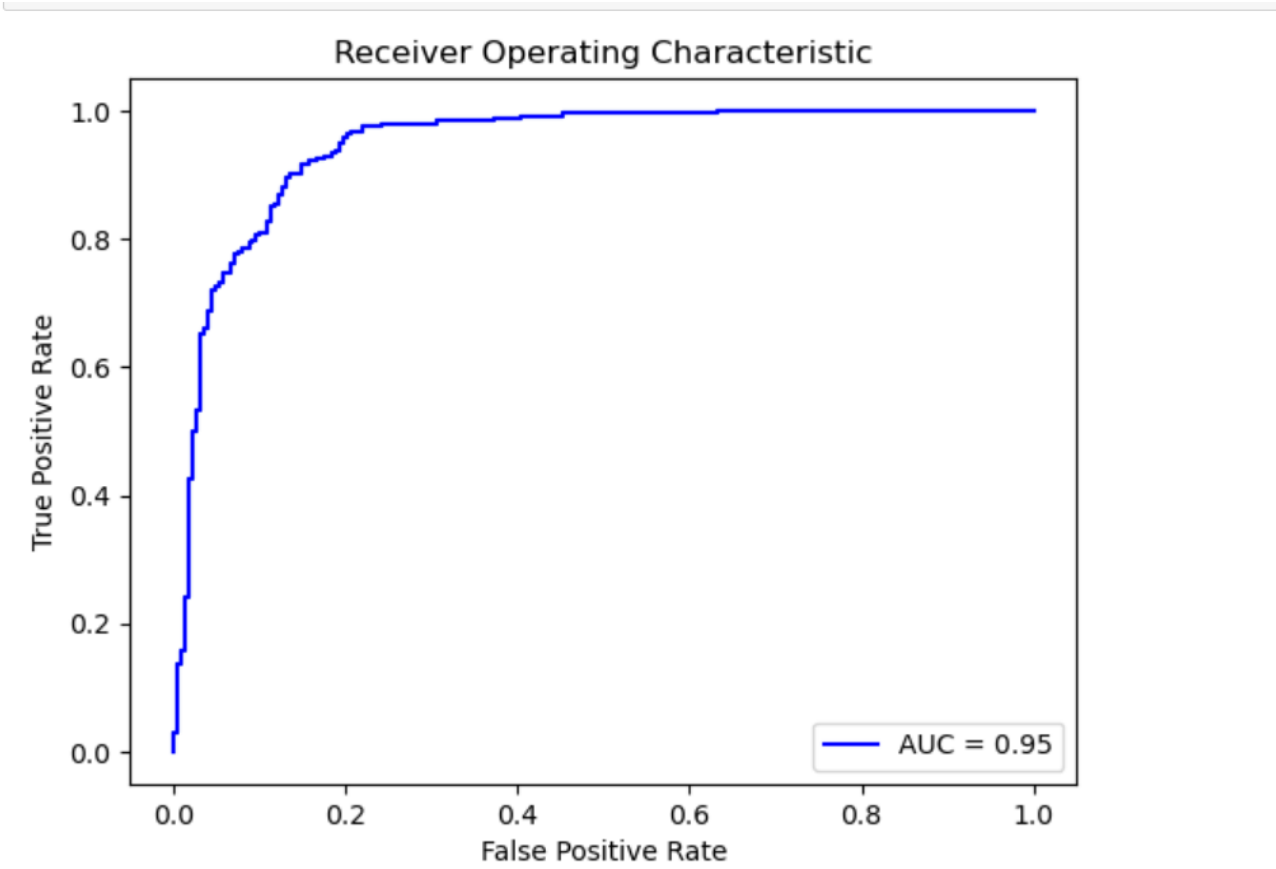
```
from sklearn.model_selection import GridSearchCV
param_dist = {'C': [0.8,0.9,1.0,1.2,1.3,1.4],
              'kernel':['linear', 'rbf','poly']}
svc_cv = GridSearchCV(svc, param_dist, cv=10)
svc_cv.fit(X_train_standardized,y_train)
print(svc_cv.best_params_)
```

```
{'C': 1.4, 'kernel': 'rbf'}
```

```
## runing the model with implementing latest parameters

svc_new = SVC(C = 1.4, kernel = "rbf", random_state = 42, probability=True)
svc_new.fit(X_train_standardized, y_train)
y_pred_best = svc_new.predict(X_test_standardized)
```

```
## confusion matrix for updated tree parameters

import matplotlib.pyplot as plt
import numpy
from sklearn import metrics

print('Model accuracy after pre processing and gridsearch', metrics.accuracy_score(y_test, y_pred_best))

confusion_matrix = metrics.confusion_matrix(y_test, y_pred_best)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [0, 1])

cm_display.plot()
plt.show()
```

```
Model accuracy after pre processing and gridsearch 0.8765957446808511
```



The ROC (Receiver Operating Characteristic) curve is a tool for evaluating the performance of a binary classification model, which shows the True Positive Rate (TPR) and the False Positive Rate (FPR) of the model at different thresholds. Visualization of the ROC curve shows the blue curve, which represents the performance of the SVM model, and the red dashed line, which represents the performance of random guessing. The closer the blue curve is to the upper left corner, the better the performance of the model. The AUC value in the figure is 0.95 , which is a relatively high value, indicating that the model does a fairly good job of distinguishing between positive and negative classes. It also indicates that our

model is able to distinguish between high and low quality red wines with relatively high reliability and has high predictive power.

3.7 Random Forest Model

After standardize the datasets, the random forest have the accuracy of 0.91.

We can do the cross validation to see whether it do have a more accurate estimate of a model's performance than a single train-test split by using multiple subsets of the data for training and testing.

```
results_df_standardized = pd.DataFrame(results_standardized, columns=["Method Name", "Accuracy"])
results_df_standardized
```

	Method Name	Accuracy
0	SVM	0.876596
1	Random Forest	0.910638

```
# Random Forest classifier initialization
rfc = RandomForestClassifier(n_estimators=150, random_state=random_value)
```

```
# Cross Validation

from sklearn.metrics import recall_score
from sklearn.model_selection import cross_val_score

rf_score = cross_val_score(estimator = rfc,
                           X = X_train_standardized, y= y_train,
                           scoring = 'recall',cv = 10,
                           verbose = 3, n_jobs=-1)
# Fit data training
rfc.fit(X_train_standardized, y_train)
# Predict data test
y_pred = rfc.predict(X_test_standardized)
print('Avarage Recall score', np.mean(rf_score))
print('Test Recall score', recall_score(y_test, y_pred))
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of  10 | elapsed:    6.3s remaining:   14.8s
[Parallel(n_jobs=-1)]: Done   7 out of  10 | elapsed:    6.3s remaining:    2.7s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    6.3s finished
```

```
Avarage Recall score 0.9463967055593686
Test Recall score 0.9338842975206612
```

Use the training data (which has been standardized) to fit a random forest model. The fitted model is then used to make predictions on the test data. The average recall is shown to be about 0.9464, indicating that the model correctly identifies about 94.64% of the positive class samples on average during cross-validation; The test recall is shown to be about 0.9339, indicating that the model correctly identifies about 93.39% of the positive class samples on the test set.

These metrics indicate that the Random Forest model performs well in recalling positive class samples and can identify most of the high-quality red wines. For the assessment of red wine quality, a high recall is beneficial because it reduces the possibility of misclassifying high-quality red wines as low-quality, ensuring that good wines are not missed.

```
from sklearn.model_selection import RandomizedSearchCV

rf_grid = {
                'n_estimators': [50, 100, 200],
                'max_depth': [3, 5, 10],
                'min_samples_split': [2, 5, 10]
                }

# Use RandomizedSearchCV
rf_cv = RandomizedSearchCV(estimator=rfc, param_distributions=rf_grid,
                           scoring='recall', cv=10)
```

```
# Fit to model

rf_cv.fit(X_train, y_train)

# Best Score and parameters using grid search
print(f'Best score: {rf_cv.best_score_}')
print(f'Best params: {rf_cv.best_params_}')
```

```
Best score: 0.9474605353466027
Best params: {'n_estimators': 100, 'min_samples_split': 2, 'max_depth': 10}
```

```
# Cross Validation

# Random Forest Regression initialization
rf_tuned = RandomForestClassifier(**rf_cv.best_params_, random_state=random_value)

rf_tuned_score = cross_val_score(estimator = rf_tuned,
                                 X = X_train_standardized, y= y_train,
                                 scoring = 'recall', cv = 10,
                                 verbose = 0)
# Fit data training
rf_tuned.fit(X_train_standardized, y_train)
# Predict data test
y_pred_tuned = rf_tuned.predict(X_test_standardized)
print('Avarage Recall score', np.mean(rf_score))
print('Test Recall score', recall_score(y_test, y_pred))
print('Avarage Recall score Tuning', np.mean(rf_tuned_score))
print('Test Recall score Tuning', recall_score(y_test, y_pred_tuned))
```

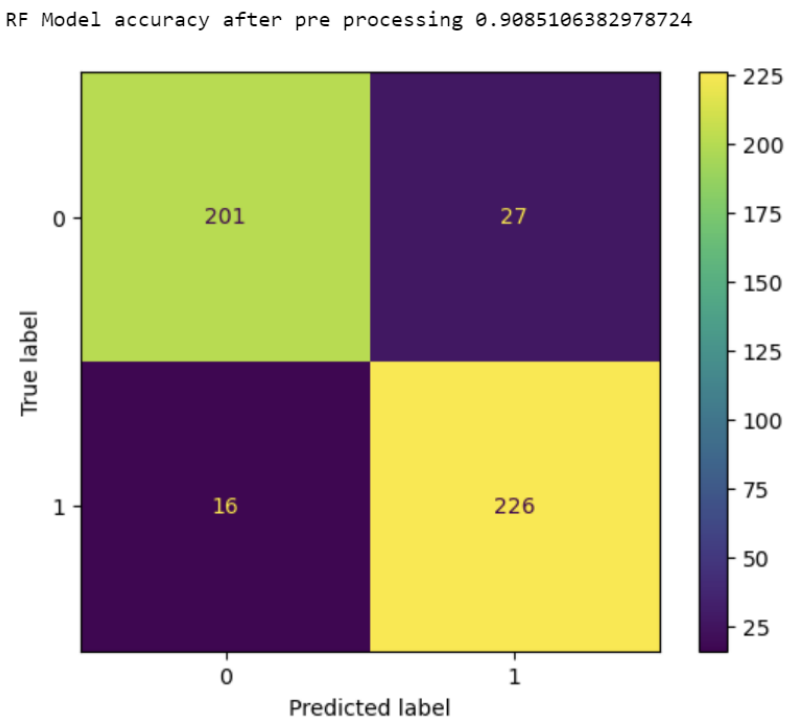
```
Avarage Recall score 0.9463967055593686
Test Recall score 0.9338842975206612
Avarage Recall score Tuning 0.945309997712194
Test Recall score Tuning 0.9462809917355371
```

Improve the recall score

RandomizedSearchCV is hyperparameter tuning technique helps you find the best set of hyperparameters for your machine learning model.

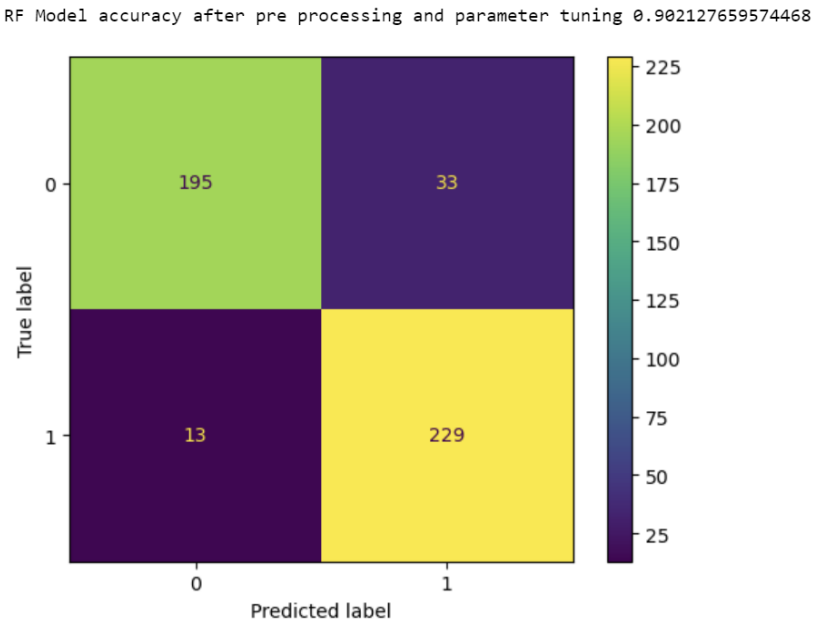
1. n_estimators is number of trees in the forest. Increasing the number of trees generally improves the performance of the Random Forest, but it also increases computation time. It is a critical parameter, and a higher number of trees can lead to better generalization.
2. Maximum depth of each decision tree. Controls the maximum depth of each tree. Deeper trees can capture more complex patterns in the data but may lead to overfitting. It is important to tune this parameter to find the right balance between model complexity and generalization.
3. Minimum number of samples required to split an internal node. Specifies the minimum number of samples a node must have before it can be split. Larger values prevent the tree from splitting nodes with a small number of samples, which can help prevent overfitting.

Random Forest 1st run



In this confusion matrix:
True Negatives (TN): The quantity of low quality red wine correctly predicted by the model is 201.
False Positives (FP): The number of low-quality wines that the model incorrectly predicts as high-quality is 27.
False Negatives (FN): The model incorrectly predicts 16 high quality wines as low quality.

Random Forest 2nd run (still need the improve cm)



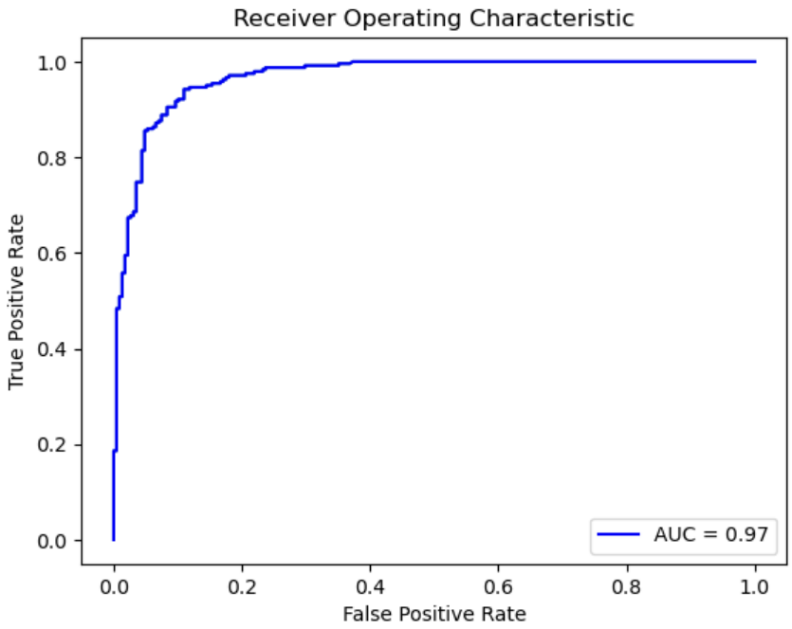
In this confusion matrix:
True Negatives (TN): The quantity of low quality red wine correctly predicted by the model is 195.
False Positives (FP): The number of low-quality wines that the model incorrectly predicts as high-quality is 33.
False Negatives (FN): The model incorrectly predicts 13 high quality wines as low quality.
True Positives (TP): The number of high quality wines correctly predicted by the model is 229.

True Positives (TP): The number of high quality wines correctly predicted by the model is 226.

This confusion matrix also shows that the accuracy of the model is about 90.85%. This means that the model correctly predicted the quality of about 90.85% of the red wines in all the samples tested. Accuracy is an important metric because it directly reflects the overall performance of the model on the test data. In the process of red wine quality assessment, a high accuracy rate means that the model is very effective in distinguishing between high and low quality red wines, and is able to provide consumers and producers with a reliable quality classification.

The average recall for cross-validation an initial 0.9464 to 0.9453.
The recall of the test data was improved from the initial 0.9339 to 0.9463.

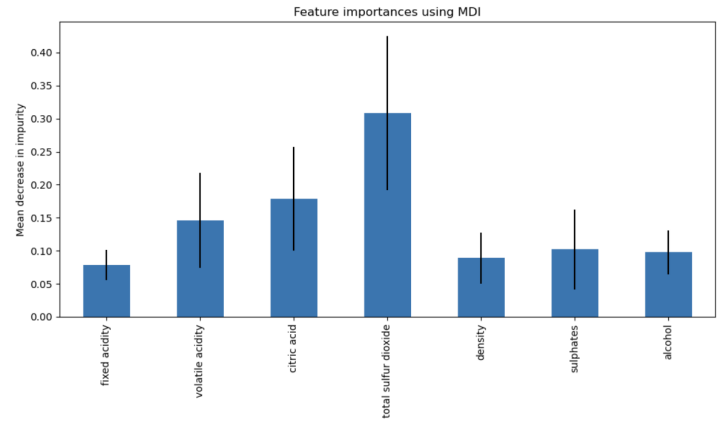
The accuracy score is 0.9021, which means that the model correctly predicted about 90.21% of the test samples.



The ROC curve is above the diagonal, indicating that the classifier outperforms the random guess, while the AUC value of 0.97 shows that the model has a good ability to distinguish between the two classes.

During the red wine quality assessment, the ROC curve proves the effectiveness of the random forest model in distinguishing between high and low quality red wines. The high AUC value implies that the model has a high degree of accuracy, which in practical applications ensures that high-quality red wines are correctly identified.

```
forest_importances = pd.Series(importances, index = feature_names_from_dataset)
# Set the size of the figure
fig, ax = plt.subplots(figsize=(10, 6))

# Plot the bar chart
forest_importances.plot.bar(yerr=std, ax=ax)

# Set titles and labels
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")

# Adjust layout for better appearance
fig.tight_layout()

# Display the plot
plt.show()
```



Using a selected amount of features for our random forest, we can see that out of the chosen features from our wine dataset, the top three attributes with the most importance for predicting if a wine would be labeled as “good” or “bad” wine are:

- Total Sulfur Dioxide
- Citric Acid
- Volatile Acidity

4. What we learned from this Project

4.1 Key Takeaways

From this analysis, we learned that data mining is more than just the process of applying algorithms and models; it involves a deep understanding of the data and how to improve model performance through data preprocessing and feature engineering. Key-takeways points include:

- The importance of data preprocessing: Unprocessed data often leads to poor model performance. Although in some instances our model accuracy's can be high or even 100% accurate, having unprocessed data can cause the data to have potential issues in the long run such as overfitting the data or having the model learn from reoccurring instances, or duplicates. Model accuracy and recall can be significantly improved by implementing normalization, dealing with missing values, and selecting appropriate features.
- Model selection and optimization: it is crucial to select the right model for a given problem. We learned that although the accuracy of our unprocessed data for K-mean nearest neighbors and logistic regression models was around 83%-85%,the same models may not have a better accuracy once we begin to process the data. Parameters of Random Forest and SVM models are tuned, and optimal model configurations can be found through cross-validation and grid search.
- Comprehensive Understanding of Evaluation Metrics: A variety of metrics such as accuracy, recall, precision, and F1 score provide a comprehensive performance evaluation.ROC curves and AUC values provide an intuitive view of a model's discriminative ability.

4.2 help business managers understand the implications

The results of the analysis show that the optimized model can differentiate between high and low-quality wine with sufficient accuracy. We can help business managers in the following ways:

Quality assurance: The model ensures that only high-quality wines are labeled and marketed as high quality, reducing brand reputation risk.

Concentration of resources: With the random forest model, wineries can improve their wine production to reach the "good" quality benchmark by making changes to the attributes that play a significant role in the rating.
Inventory management: The predictive model can help determine which wines should be brought to market and which may need further quality improvement through quality judgment. Once wineries are able to improve certain attributes to meet the rating criteria, then they would be able to place their wines into the markets.

Pricing strategy: Predictions of wine quality can be used as the basis for pricing and marketing strategies, helping brands determine price ranges for different quality wines.

5. Conclusion

In this study, the quality of wine was assessed and categorized by applying various machine learning techniques. In particular, the random forest and support vector machine (SVM) models demonstrated significant accuracy on the preprocessed data, reaching about 90.94% and 85.31% accuracy, respectively. In addition, through cross-validation and parameter tuning, we further improved the performance of the random forest model, which finally achieved a test recall of about 94.63% and an average recall of 94.53%, implying that the model can reliably identify most of the high-quality wines.

The analysis of the confusion matrix and ROC curves provided a deeper understanding of the model performance. Specifically, the ROC curve analysis demonstrated the model's ability to highly discriminate between the two categories (high-quality and low-quality wines), such as the random forest model with an AUC value of 0.97. These metrics indicate that the selected model's prediction of the quality of red wines is not only accurate, but also highly reliable.

In addition, our study reveals the significant effect of data preprocessing on model performance. Unprocessed data were less accurate than carefully preprocessed data on the benchmark model, which emphasizes the importance of the data preparation stage in modeling efforts. The analysis of the proportions of the class variables also showed that the distribution of high and low-quality wine in the dataset had a direct impact on model training, suggesting a data balancing issue that may need to be considered in future work. With the data balancing issues occurring, we would then need to commit to using SMOTE in order to avoid an imbalance in the training data for our machine learning models.

Overall, the results of this study provide a powerful tool for wine producers and distributors to assess wine quality scientifically and may serve as a positive guide for future red wine production and assessment. Future work could explore additional feature engineering methods, consider more complex models, or further validate the generalization ability of the model on larger and more diverse datasets.