VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ADVANCED SCIENCE - (SAS)

Winter Semester - 2021-2022

WINE QUALITY PREDICTION USIGN MACHINE LEARNING

PROJECT REVIEW

Submitted fulfilment for the J-component of CSC5007

EXPLORATORY DATA ANALYSIS

CAL COURSE
In

MSC. DATASCIENCE

BY

RESHMA P - 21MDT0104

RAHUL ROSARIO S – 21MDT0139
POOJA K - 21MDT0140

VARUN PAANDIAN M– 21MDT0141

Under the guidance of

Prof.Dr. PALLAVI MISHRA

ABSTRACT

The quality of a wine is important for the consumers as well as the wine industry. The traditional (expert) way of measuring wine quality is time-consuming. Nowadays, machine learning models are important tools to replace human tasks. In this case, there are several features to predict the wine quality but the entire features will not be relevant for better prediction. So, our thesis work is focusing on what wine features are important to get the promising result. For the purpose of classification model and evaluation of the relevant features, we used three algorithms namely support vector machine (SVM), naïve Bayes (NB), and artificial neural network (ANN). In this study, we used two wine quality datasets red wine and white wine. To evaluate the feature importance we used the Pearson coefficient correlation and performance measurement matrices such as accuracy, recall, precision, and f1 score for comparison of the machine learning algorithm. A grid search algorithm was applied to improve the model accuracy. Finally, we achieved the artificial neural network (ANN) algorithm has better prediction results than the Support Vector Machine (SVM) algorithm and the Naïve Bayes (NB) algorithm for both red wine and white wine datasets.

1.INTRODUCTION

The quality of the wine is a very important part for the consumers as well as the manufacturing industries. Industries are increasing their sales using product quality certification. Nowadays, all over the world wine is a regularly used beverage and the industries are using the certification of product quality to increases their value in the market. Previously, testing of product quality will be done at the end of the production, this is time taking process and it requires a lot of resources such as the need for various human experts for the assessment of product quality which makes this process very expensive. Every human has their own opinion about the test, so identifying the quality of the wine based on humans experts it is a challenging task. There are several features to predict the wine quality but the entire features will not be relevant for better prediction. The research aims to what wine features are important to get the promising result by implementing the machine learning classification algorithms such as Support Vector Machine (SVM), Naïve Bayes (NB), and Artificial Neural Network (ANN), using the wine quality dataset. The wine quality dataset is publically available on the UCI machine learning repository (Cortez et al., 2009). The dataset has files wine variants of the Portuguese "Vinho Verde" wine. It contains a large collection of datasets that have been used for the machine learning community. The wine dataset contains 1599 instances. The files contain 11 input features and 1 output feature. Input features are based on the physicochemical tests and output variable based on sensory data is scaled in 11 quality classes from 0 to 10 (0-very bad to 10-very good).

2.BACKGROUND

A wide range of machine learning algorithms is available for the learning process. This section describes the classification algorithms used in wine quality prediction and related work.

3.CLASSIFICATION ALGORITHM

3.1 SUPPORT VECTOR MACHINE

The support vector machine (SVM) is the most popular and most widely used machine learning algorithm. It is a supervised learning model that can perform classification and regression tasks. However, it is primarily used for classification problems in machine learning (Gandhi, 2018).

The SVM algorithm aims to create the best line or decision boundary that can separate n-dimensional space into classes. So we can put the new data points easily in the correct groups. This best decision boundary is called a hyperplane.

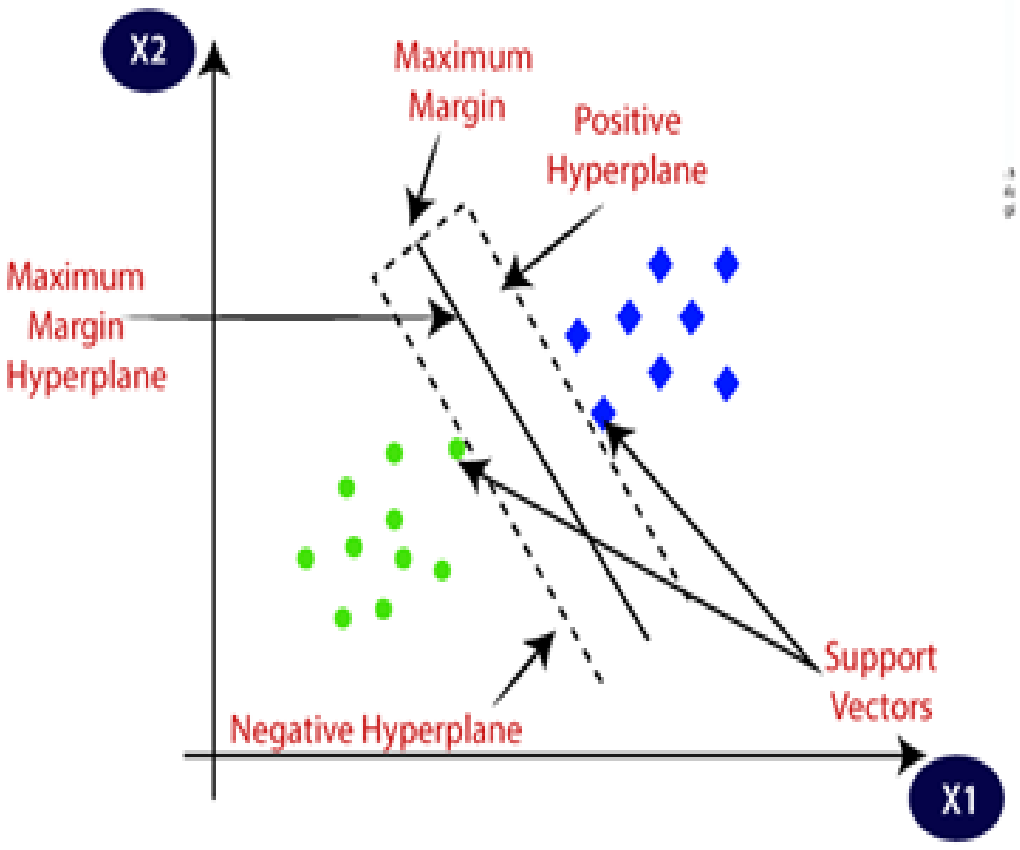


FIG. SUPPORT VECTOR MACHINE

The support vector machine selects the extreme data points that helping to create the hyperplane. In FIG , two different groups are classified by using the decision boundary or hyperplane:

The SVM model is used for both non-linear and linear data. It uses a nonlinear mapping to convert the main preparing information into a higher measurement. The model searches for the linear optimum splitting hyperplane in this new measurement. A hyperplane can split the data into two classes with an appropriate nonlinear mapping to suitably high measurements and for the finding, this hyperplane SVM uses the support vectors and edges (J. Han et al., 2012). The SVM model is a representation of the models as a point in space, the different classes are isolated by the gap to mapped with the aim that instances are wide as would be careful. The model can perform out a nonlinear form of classification (Kumar et al., 2020).

3.2 NAIVE BAYESIAN

The naive Bayesian is the simple supervised machine learning classification algorithm based on the Bayes theorem. The algorithm assumes that the feature conditions are independent of the given class (Rish, 2001). The naive Bayes algorithm helps to build fast machine learning models that can make a fast prediction. The algorithm finds whether a particular portion has a spot by a particular class it utilizes the probability of likelihood (Kumar et al., 2020).

3.3 ARTIFICIAL NEURAL NETWORK

The artificial neural network is a collection of neurons that can process information. It has been successfully applied to the classification task in several industries, including the commercial, industrial, and scientific filed (Zhang, 2000). The algorithm model is a connection between the neurons that are interconnected with the input layer, a hidden layer, and an output layer (Hewahi, 2017).

The neural network is constant because while an element of the neural network is failing, it can continue its parallel nature without any difficulties (Mhatre et al., 2015).

The implementation of the artificial neural network consists of three layers: input, hidden, and, output as shown in FIG . The function at the input layer is mapped the input attribute which passes input to the hidden layer. The hidden layer is a middle layer where all

input with the weights is received to each node in the hidden layer. The output layer is mapped to the predicted elements (says, 2020).

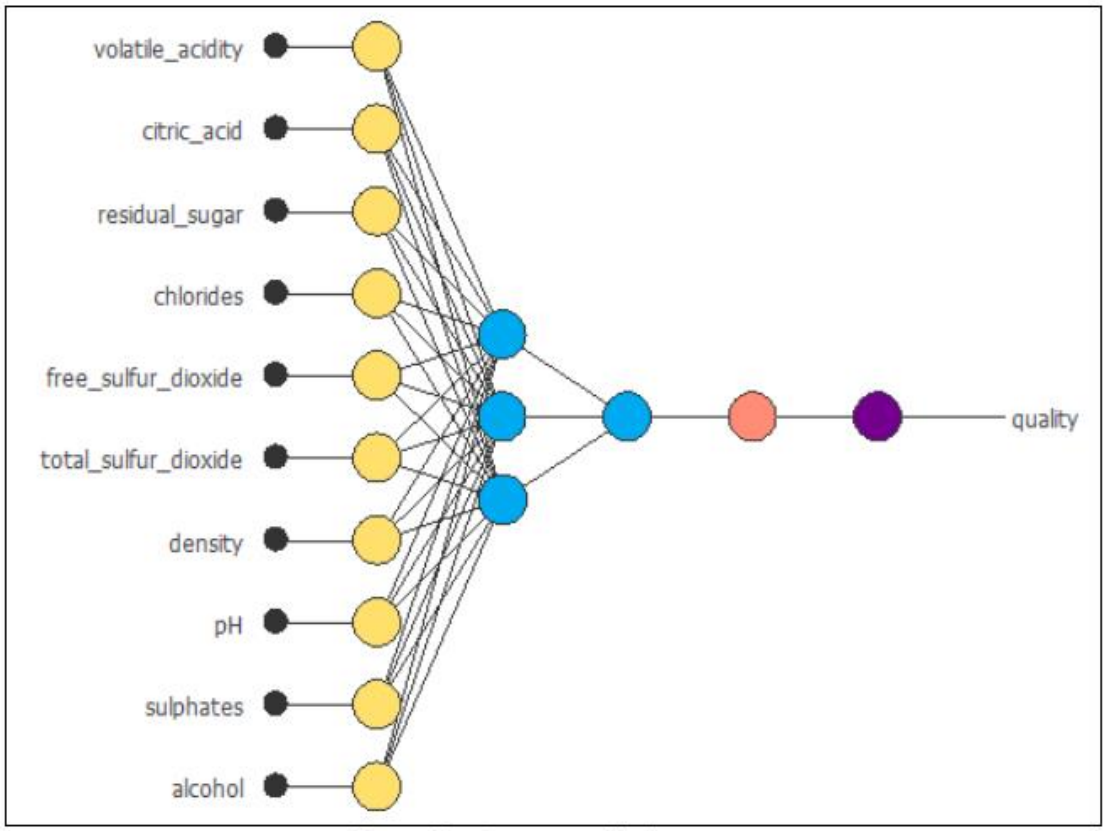


Image Courtesy: neuraldesigner.com

FIG : NEURAL NETWORK OF WINE PREDICTION

The implementation of the artificial neural network consists of three layers: input, hidden, and, output as shown in Figure 2. The function at the input layer is mapped the input attribute which passes input to the hidden layer. The hidden layer is a middle layer where all input with the weights is received to each node in the hidden layer. The output layer is mapped to the predicted elements (says, 2020).

3.4 RELATED WORK

Kumar et al. (2020) have used prediction of red wine quality using its various attributes and for the prediction, they used random forest, support vector machine, and naive Bayes techniques (Kumar et al., 2020). They have calculated the performance measurement such as precision, recall, f1-score, accuracy, specificity, and misclassification error. Among these three techniques, they achieved the best result from the support vector machine as

compare to the random forest and naive Bayes techniques. They achieved the accuracy of the support vector machine technique is 67.25%.

Gupta, (2018) has used important features from red wine and white wine quality using various machine learning algorithms such as linear regression, neural network, and support vector machine techniques. They used two ways to determine the wine quality. Firstly the dependency of the target variable on the independent variable and secondly predicting the value of the target variable and conclusion that all features are not necessary for the prediction instead of selecting only necessary features to predict the wine quality (Gupta, 2018).

Dahal et al., (2021) has predicted the wine quality based on the various parameters by applying various machine learning models such as rigid regression, support vector machine, gradient boosting regressor, and multi-layer artificial neural network. They compare the performance of the models to predict wine quality and from their analysis, they found gradient boosting regressor is the best model to other model performances with the MSE, R, and MAPE of 0.3741, 0.6057, and 0.0873 respectively(Dahal et al., 2021).

Er, and Atasoy, (2016) has proposed the method to classify the quality of the red wine and white wine using three machine learning algorithm such as k-nearest-neighborhood, random forest, and support vector machine. They used principal component analysis for the feature 7 selection and they have achieved the best result using the random forest algorithm (Er, 2016).

Lee et al., (2015) has proposed a method decision tree-based to predict the wine quality and compare their approach using three machine learning algorithm such as support vector machine, multi-layer perceptron, and BayesNet. They found their proposed method is better compared to other stated methods (Lee et al., 2015).

P. Appalasamy et al., (2012) have predicted the wine quality based on the physiochemical data. They used both red wine and white wine datasets and applied the decision tree and naive Bayes algorithms. They compare the results of these two algorithms and conclude that the classification approach can help to improve the wine quality during production (P. Appalasamy et al., 2012)

4.PROBLEM

4.1 PROBLEM DEFINITION

The significance of each feature for the wine quality prediction is not yet quantified. And in terms of performance, the current accuracy is about 67.25%. Thus, in this thesis, we considered two aspects of the problems mentioned above. The first one is the study of the importance of the features for the prediction of wine quality. The secondly, performance of the prediction model can be improved using a neural network with other ordinary classifiers used by the articles cited above.

4.2 RESEARCH AIM

The following research question and hypothesis are formulated.

1. What wine features are important to get a promising result?

   The researchers have used a neural network for the regression task but for the classification task neural network was never used. Hypothetically, the current prediction model that has been obtained by researchers will be improved by using the neural network. To address the research question the following objectives are formulated :

- ✓ To balance the dataset.
- ✓ To analyze the impact of the features.
- ✓ To optimize the classification models through hyperparameter tuning.
- ✓ To model and evaluate the approaches.

5. METHOD AND APPROACH

5.1 DATA DESCRIPTION

The red wine and white wine datasets have been used in this paper which is obtained from the UCI machine learning repository it contains a large collection of datasets that have been used

FIXED ACIDITY	**Fixed acids, numeric from 3.8 to 15.9**

for the machine learning community. The dataset contains two excel files, related to red wine and white wine variants of the Portuguese "Vinho Verde" wine (Cortez et al., 2009). The red wine dataset contains 1599 instances and the white wine dataset contains 4898 instances. Both datasets have 11 input variables (based on physicochemical tests): fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulfates, alcohol, and 1 output variable (based on sensory data): quality. Sensory data is scaled in 11 quality classes from 0 to 10 (0-very bad to 10-very good). Below Table description of the attributes.

VOLATILE ACIDITY	Volatile acids, numeric from 0.1 to 1.6
CITRIC ACID	Citric acids, numeric from 0.0 to 1.7
RESIDUAL SUGAR	residual sugar, numeric from 0.6 to 65.8
CHLORIDES	Chloride, numeric from 0.01 to 0.61
FREE SULFUR DIOXIDE	Free sulfur dioxide, numeric: from 1 to 289
TOTAL SULFUR DIOXIDE	Total sulfur dioxide, numeric: from 6 to 440
DENSITY	Density, numeric: from 0.987 to 1.039
PH	pH, numeric: from 2.7 to 4.0
SULFATES	Sulfates, numeric: from 0.2 to 2.0
ALCOHOL	Alcohol, numeric: from 8.0 to 14.9
QUALITY	Quality, numeric: from 0 to 10, the output target

6.ATTRIBUTE DISTRIBUTION

7.FEATURE SELECTION

Feature selection is the method of selection of the best subset of features that will be used for classification (Fauzi et al., 2017). Most of the feature selection method is divided into a filter and wrapper, the filter uses the public features work individually from the learning algorithm and the wrapper evaluates the features and chooses attributes based on the estimation of the accuracy by using a search algorithm and specific learning model (onan and Korukoğlu, 2017). In this study, for a better understanding of the features and to examines the correlation between the features. The Pearson correlation coefficient is calculated for each feature in Table 1, this shows the pairwise person correlation coefficient P, which is calculated by using the below formula (Dastmard, 2013).

$$Px, = \frac{\text{cov (X, Y)}}{\sigma X, \Sigma y}$$

Where the σ is the standard deviation of the features X and Y and cov is the covariance. The range of the correlation coefficient from -1 to 1. Point 1 value implies linear equation is describes the correlation between X and Y strong positive, which is all data points are lying on a line for Y increases as X increases. Where point -1 value indicates that strong negative correlations between data points. All data points lie on a line in which Y decreases as X increases. And point 0 indicates that there is an absence of correlation between the points (Dastmard, 2013).

8. HYPERPARAMETR TUNING

The grid search is a basic method for hyperparameter tuning. Perform an inclusive search on the hyperparameter set specified by the user. Grid search is suitable for several hyperparameters with limited search space. The grid search algorithm is straightforward with enough resources, the most accurate prediction can be drawn and users can always find the best combination (Joseph, 2018). Running grid search in parallel is easy because each test is run separately without affected by the time series. The results of one experiment are independent of the results of other experiments. Computing resources can be allotted in a very flexible way. In addition, grid search can accept a limited sampling range, because too many settings are not suitable. In practice, grid search is almost preferable only when the user has enough knowledge with these hyperparameters to allow the definition of a narrow search space, and it is not necessary to adjust more than three hyperparameters simultaneously. Although other search algorithms may have more useful features, grid search is still the most widely used method due to its mathematical simplicity (Yu and Zhu, 2020).

9. EVALUATION

The performance measurement is calculated and evaluate the techniques to detect the effectiveness and efficiency of the model. There are four ways to check the predictions are correct or incorrect:

- True Positive: Number of samples that are predicted to be positive which are truly positive.
- False Positive: Number of samples that are predicted to be positive which are truly negative.
- False Negative: Number of samples that are predicted to be negative which are truly positive.
- True Negative: Number of samples that are predicted to be negative which are truly negative.

Below listed techniques, we use for the evaluation of the model.

9.1 ACCURACY

Accuracy is defined as the ratio of correctly predicted observation to the total observation. The accuracy can be calculated easily by dividing the number of correct predictions by the total number of predictions.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{False Negative} + \text{True Negative}}$$

9.2 PRECISION

Precision is defined as the ratio of correctly predicted positive observations to the total predicted positive observations.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

9.3 RECALL

Recall is defined as the ratio of correctly predicted positive observations to all observations in the actual class. The recall is also known as the True Positive rate calculated as,

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

9.4 F1 SCORE

F1 score is the weighted average of precision and recall. The f1 score is used to measure the test accuracy of the model. F1 score is calculated by multiplying the recall and precision is divided by the recall and precision, and the result is calculated by multiplying two.

$$\text{F1 score} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall + Precision}}$$

Accuracy is the most widely used evaluation metric for most traditional applications. But the accuracy rate is not suitable for evaluating imbalanced data sets, because many experts have observed that for extremely skewed class distributions, the recall rate for minority classes is typically 0, which means that no classification rules are generated for the minority class. Using the terminology in information retrieval, the precision and recall of the minority categories are much lower than the majority class. Accuracy gives more weight to the majority class than to the minority class, this makes it challenging for the classifier to implement well in the minority class. 13 For this purpose, additional metrics are coming into widespread usage (Guo et al., 2008).

The F1 score is the popular evaluation matric for the imbalanced class problem (Estabrooks and Japkowicz, 2001). F1 score combines two matrices: precision and recall. Precision state how accurate the model was predicting a certain class and recall state that the opposite of the regrate misplaced instances which are misclassified. Since the multiple classes have multiple F1 scores. By using the unweighted mean of the F1 scores for our final scoring. We want our models to get optimized to classify instances that belong to the minority side, such as wine quality of 3, 8, or 9 equally well with the rest of the qualities that are represented in a larger number.

10. EXPERIMENTAL DESIGN USING PYTHON

```
#Importing required packages.
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
%matplotlib inline
```

```
#loading dataset
wine = pd.read_csv("D:/winequality prediction.csv")
wine
```

Out[43]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

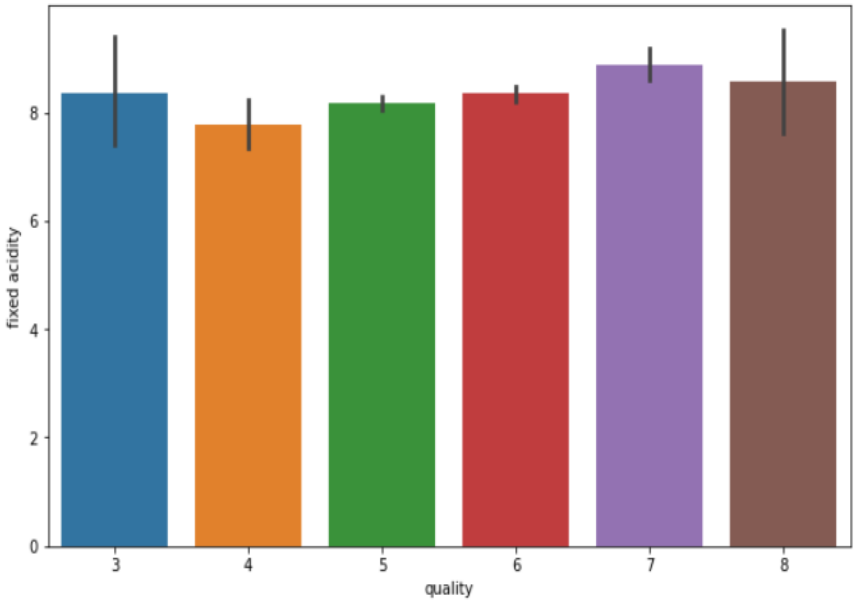
```
#Information about the data columns
wine.info()
```

```
<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
```

Let's Do Some Plotting To Know How The Data Columns Are Distributed In The Dataset

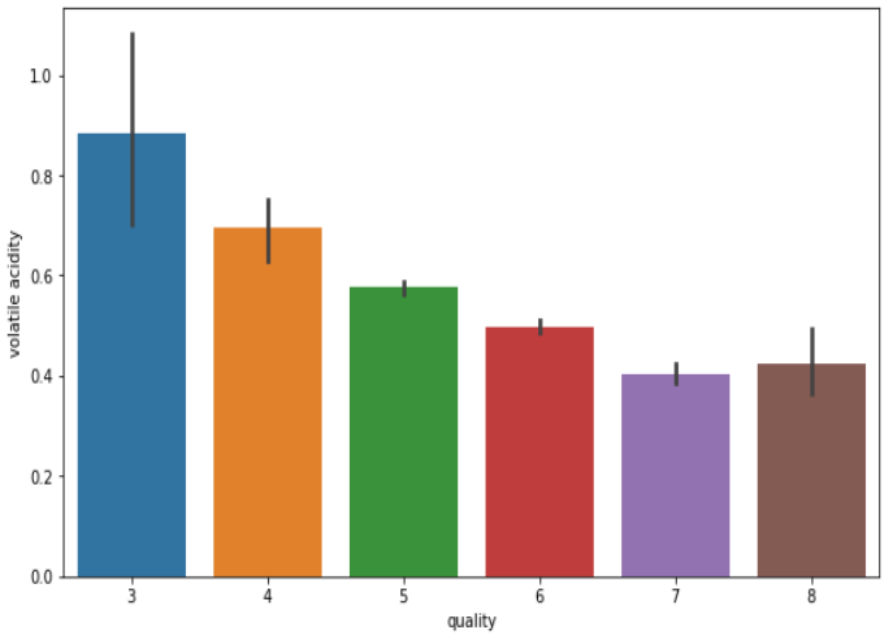
In [45]:
```
#Here we see that fixed acidity does not give any specification to classify the quality.
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'fixed acidity', data = wine)
```

Out[45]: <AxesSubplot:xlabel='quality', ylabel='fixed acidity'>



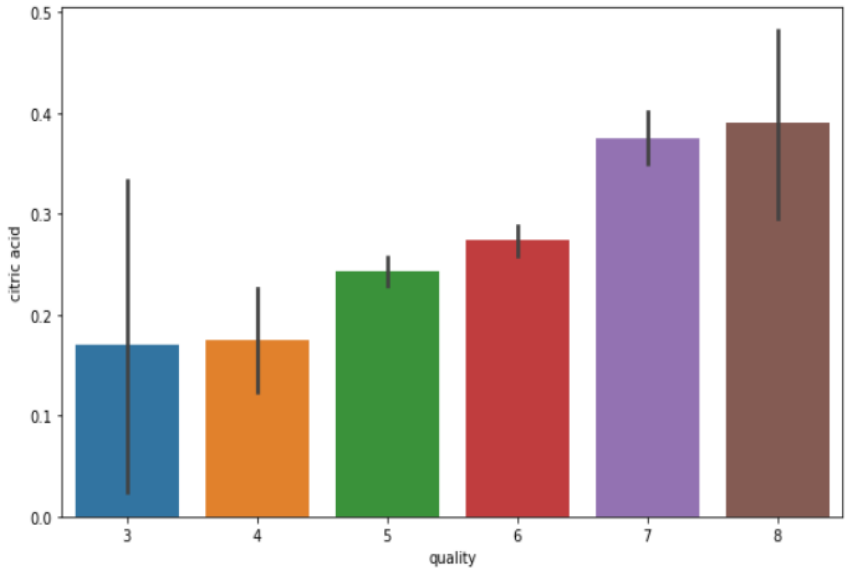
In [46]:
```
#Here we see that its quite a downing trend in the volatile acidity as we go higher the quality
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'volatile acidity', data = wine)
```

Out[46]: <AxesSubplot:xlabel='quality', ylabel='volatile acidity'>



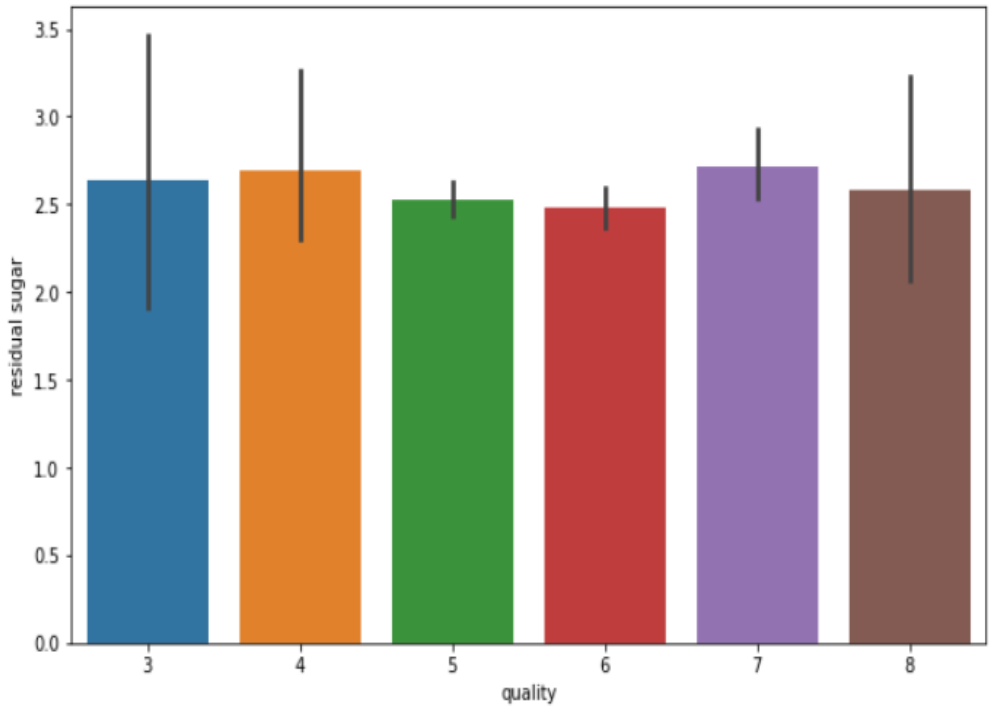
In [47]:
```
#Composition of citric acid go higher as we go higher in the quality of the wine
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'citric acid', data = wine)
```

Out[47]: <AxesSubplot:xlabel='quality', ylabel='citric acid'>



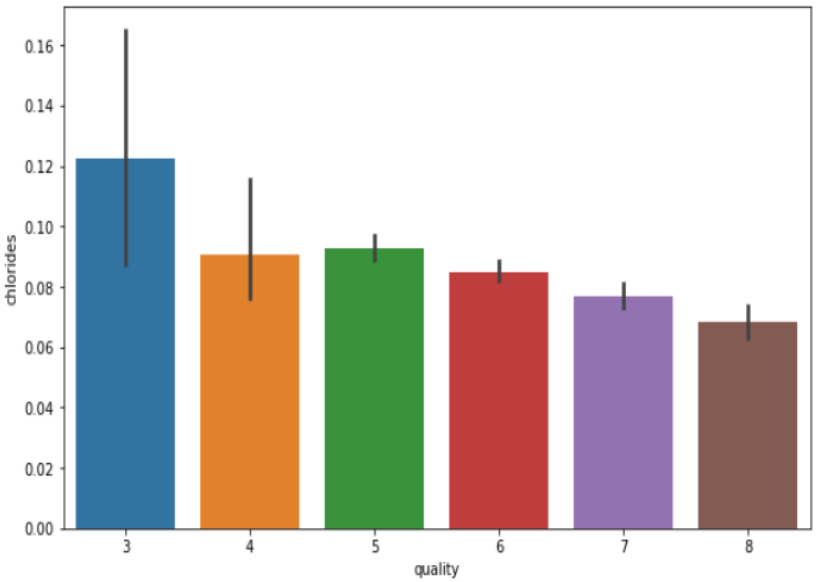
In [48]:
```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'residual sugar', data = wine)
```

Out[48]: <AxesSubplot:xlabel='quality', ylabel='residual sugar'>



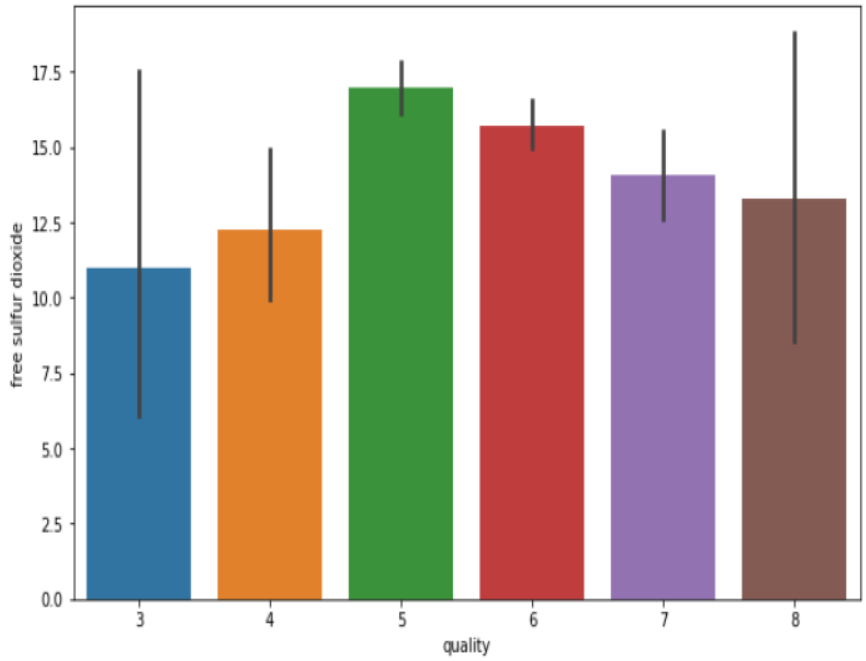
In [49]:
```
#Composition of chloride also go down as we go higher in the quality of the wine
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'chlorides', data = wine)
```

Out[49]: <AxesSubplot:xlabel='quality', ylabel='chlorides'>



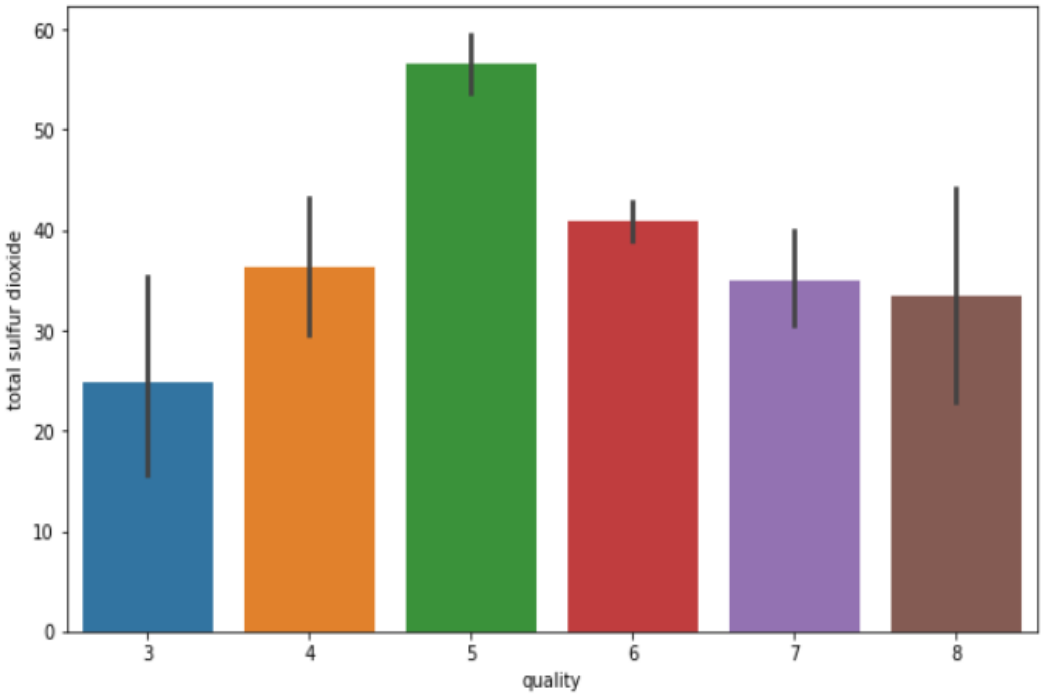
In [50]:
```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = wine)
```

Out[50]: <AxesSubplot:xlabel='quality', ylabel='free sulfur dioxide'>



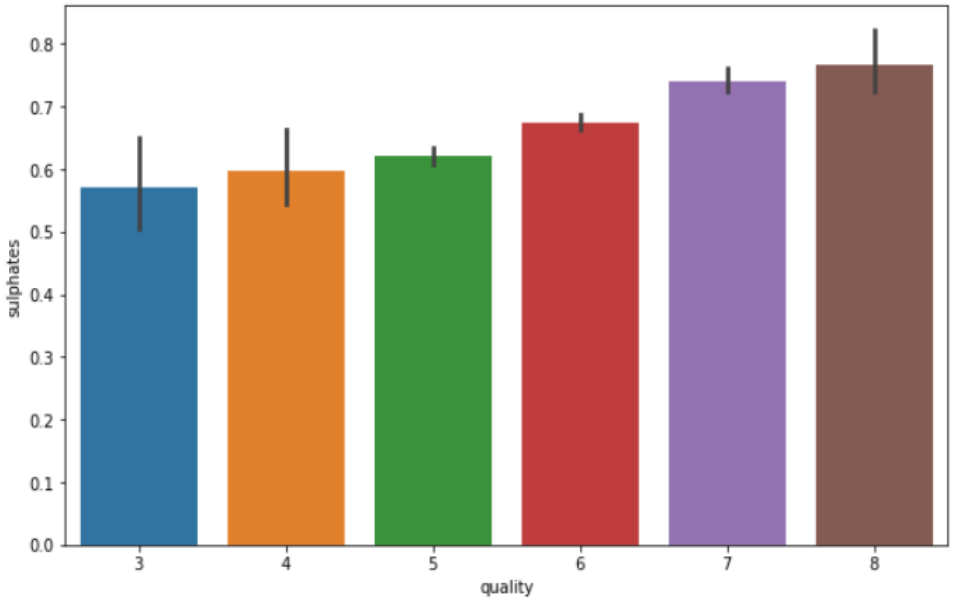
n [51]:
```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'total sulfur dioxide', data = wine)
```

ut[51]: <AxesSubplot:xlabel='quality', ylabel='total sulfur dioxide'>



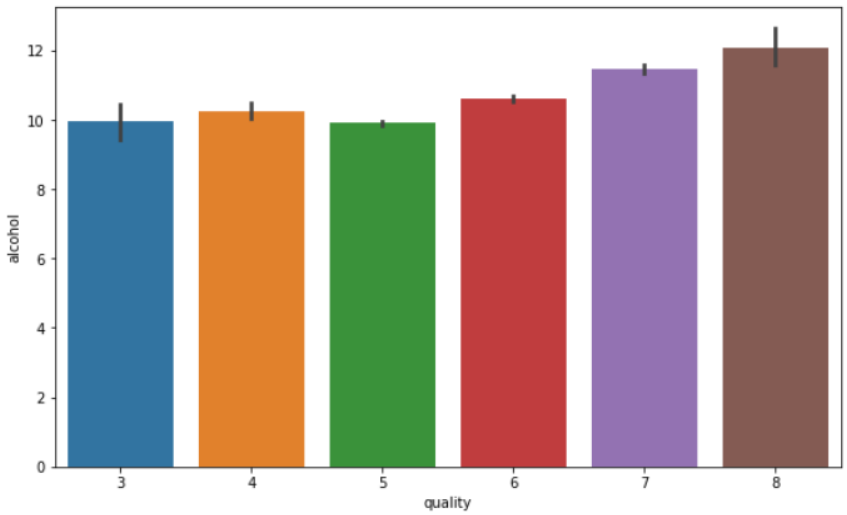
In [52]:
```
#Sulphates level goes higher with the quality of wine
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'sulphates', data = wine)
```

Out[52]: <AxesSubplot:xlabel='quality', ylabel='sulphates'>



In [53]:
```
#Alcohol level also goes higher as te quality of wine increases
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'alcohol', data = wine)
```

Out[53]: <AxesSubplot:xlabel='quality', ylabel='alcohol'>



Preprocessing Data For Performing Machine Learning Algorithms

In [54]:
```
#Making binary classificaion for the response variable.
#Dividing wine as good and bad by giving the limit for the quality
bins = (2, 6.5, 8)
group_names = ['bad', 'good']
wine['quality'] = pd.cut(wine['quality'], bins = bins, labels = group_names)
```

In [55]:
```
#Now lets assign a labels to our quality variable
label_quality = LabelEncoder()
```

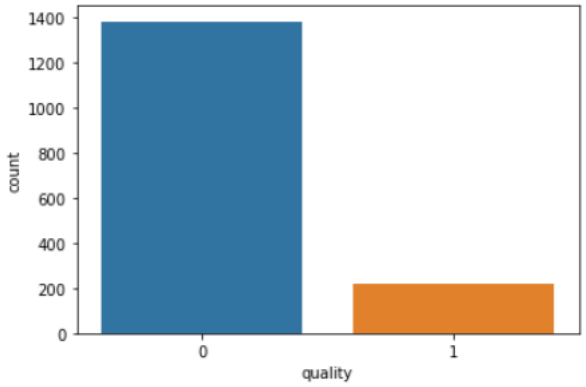
In [56]:
```
#Bad becomes 0 and good becomes 1
wine['quality'] = label_quality.fit_transform(wine['quality'])
```

In [57]:
```
wine['quality'].value_counts()
```

Out[57]:
```
0    1382
1     217
Name: quality, dtype: int64
```

In [58]:
```
sns.countplot(wine['quality'])
```

58]: <AxesSubplot:xlabel='quality', ylabel='count'>



59]:
```
#Now seperate the dataset as response variable and feature variabes
X = wine.drop('quality', axis = 1)
y = wine['quality']
```

In [60]:
```
#Train and Test splitting of data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

In [61]:
```
#Applying Standard scaling to get optimized result
sc = StandardScaler()
```

In [62]:
```
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Our Training And Testing Data Is Ready Now To Perform Machine Learning Algorithm

Random Forest Classifier

```
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
pred_rfc = rfc.predict(X_test)
```

```
#Let's see how our model performed
print(classification_report(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.90      0.97      0.93       273
           1       0.68      0.36      0.47        47

    accuracy                           0.88       320
   macro avg       0.79      0.67      0.70       320
weighted avg       0.87      0.88      0.87       320
```

Random forest gives the accuracy of 87%

```
#Confusion matrix for the random forest classification
print(confusion_matrix(y_test, pred_rfc))
```

```
[[265   8]
 [ 30  17]]
```

Stochastic Gradient Decent Classifier

```
sgd = SGDClassifier(penalty=None)
sgd.fit(X_train, y_train)
pred_sgd = sgd.predict(X_test)
```

```
print(classification_report(y_test, pred_sgd))
```

```
              precision    recall  f1-score   support

           0       0.90      0.85      0.88       273
           1       0.34      0.45      0.39        47

    accuracy                           0.79       320
   macro avg       0.62      0.65      0.63       320
weighted avg       0.82      0.79      0.80       320
```

84% accuracy using stochastic gradient descent classifier

```
print(confusion_matrix(y_test, pred_sgd))
```

```
[[233  40]
 [ 26  21]]
```

Support Vector Classifier

```
[69]: svc = SVC()
      svc.fit(X_train, y_train)
      pred_svc = svc.predict(X_test)
```

```
[70]: print(classification_report(y_test, pred_svc))
```

```
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       273
           1       0.71      0.26      0.37        47

    accuracy                           0.88       320
   macro avg       0.80      0.62      0.65       320
weighted avg       0.86      0.88      0.85       320
```

Support vector classifier gets 86% accuracy

Let's try to increase our accuracy of models

Grid Search CV

```
[4]: #Finding best parameters for our SVC model
     param = {
         'C': [0.1,0.8,0.9,1,1.1,1.2,1.3,1.4],
         'kernel':['linear', 'rbf'],
         'gamma' :[0.1,0.8,0.9,1,1.1,1.2,1.3,1.4]
     }
     grid_svc = GridSearchCV(svc, param_grid=param, scoring='accuracy', cv=10)
```

```
[5]: from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
     %matplotlib inline
     grid_svc.fit(X_train, y_train)
```

```
[5]: GridSearchCV(cv=10, estimator=SVC(),
                  param_grid={'C': [0.1, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4],
                              'gamma': [0.1, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4],
                              'kernel': ['linear', 'rbf']},
                  scoring='accuracy')
```

```
[6]: from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
     %matplotlib inline
     #Best parameters for our svc model
     grid_svc.best_params_
```

```
[6]: {'C': 1.2, 'gamma': 0.9, 'kernel': 'rbf'}
```

```
#Let's run our SVC again with the best parameters.
svc2 = SVC(C = 1.2, gamma =  0.9, kernel= 'rbf')
svc2.fit(X_train, y_train)
pred_svc2 = svc2.predict(X_test)
print(classification_report(y_test, pred_svc2))
```

```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       273
           1       0.89      0.34      0.49        47

    accuracy                           0.90       320
   macro avg       0.89      0.67      0.72       320
weighted avg       0.90      0.90      0.88       320
```

SVC improves from 86% to 90% using Grid Search CV

Cross Validation Score for random forest and SGD

```
In [82]: #Now lets try to do some evaluation for random forest model using cross validation.
         rfc_eval = cross_val_score(estimator = rfc, X = X_train, y = y_train, cv = 10)
         rfc_eval.mean()
```

```
Out[82]: 0.9124569389763779
```

Random forest accuracy increases from 87% to 91 % using cross validation score.

10.1 NOW FOR THE SAME DATASET WE ARE GOING TO SEE DIFFERENT TYPES OF GRAPHS VISUALIZATION

***Pandas** is a useful library in data analysis, **Numpy** library used for working with arrays, **Seaborn** and **Matplotlib** are used in data visualization.*

```
In [12]: # import libraries

         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sb
```

READING DATA:

```
# loading the data
Dataframe = pd.read_csv("D:/winequality prediction.csv")
Dataframe
```

*Pandas **read_csv** function used for reading the csv file.*

DATA CHECKING:

```
]: # show rows and columns
   Dataframe.head()
```

]:

	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

```
# getting info.
Dataframe.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
```

```
Dataframe.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	c
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.0
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.6
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.8
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.0
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.0
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.0
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.0
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.0

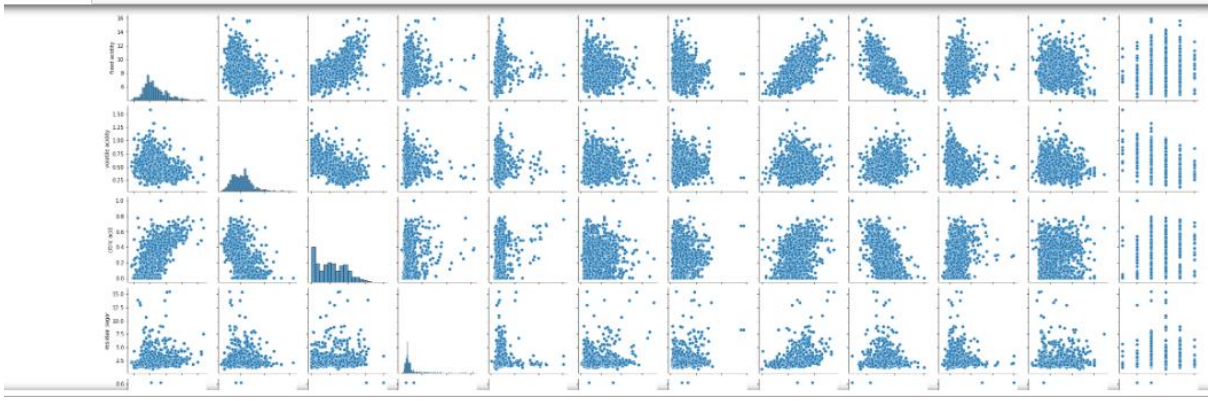
CHECKING NULL VALUES:

```
# null value check
Dataframe.isnull().sum()
```

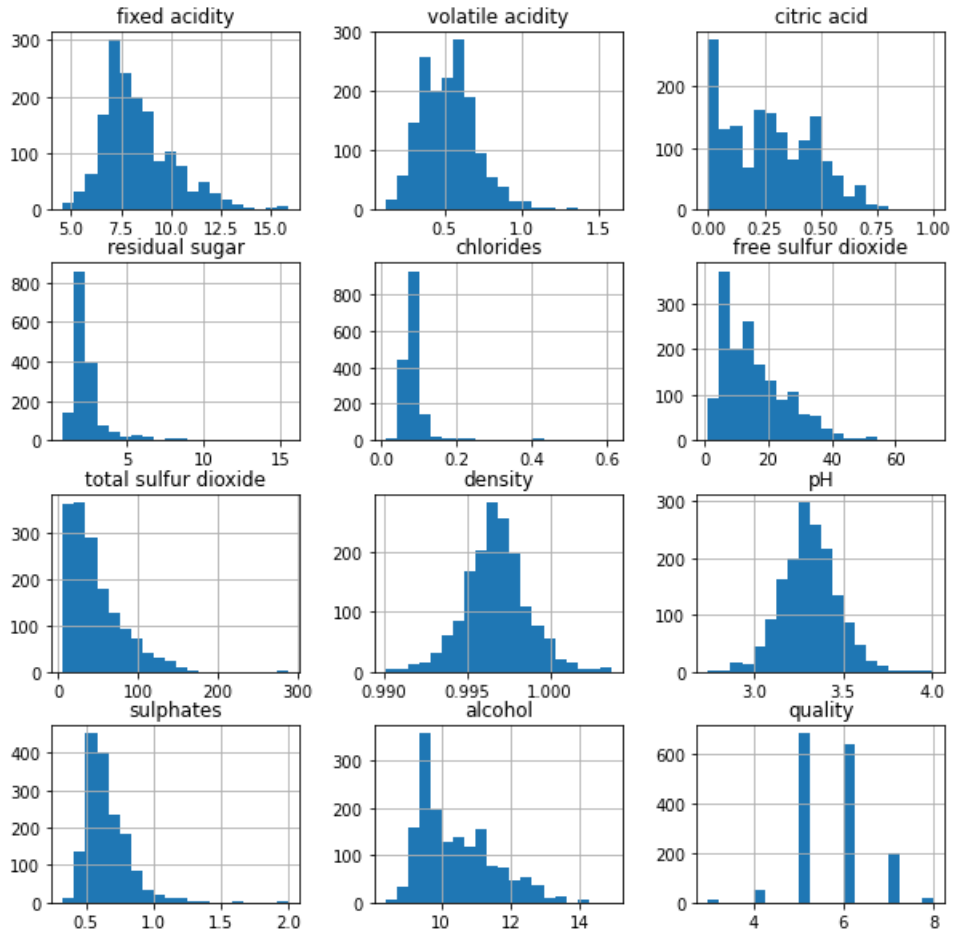
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality   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

DATA VISUALIZATION:

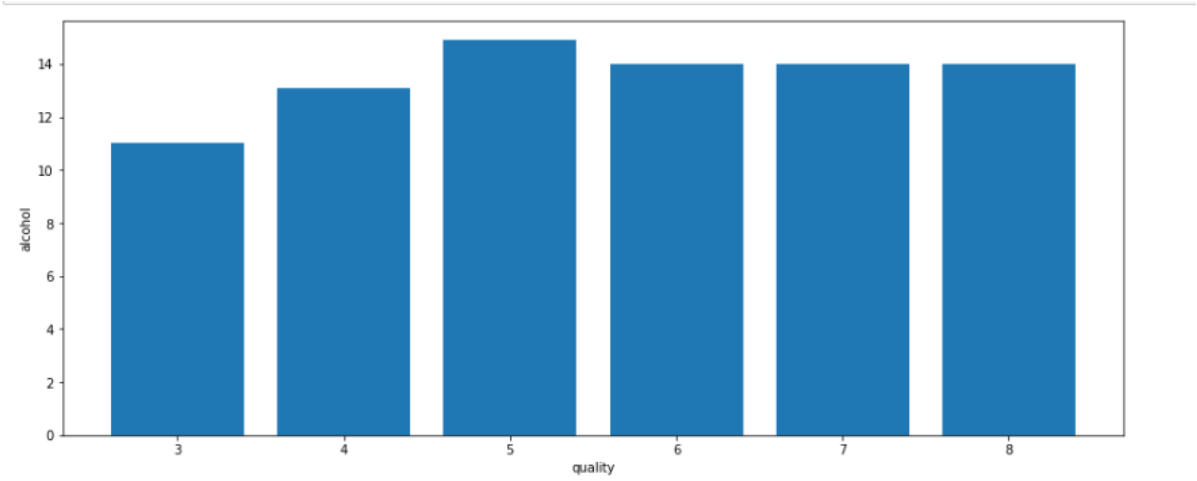
```
In [21]: # plot pairplot
         sb.pairplot(Dataframe)
         #show graph
         plt.show()
```



```
#plot histogram
Dataframe.hist(bins=20,figsize=(10,10))
#plot showing
plt.show()
```




```
plt.figure(figsize=[15,6])
plt.bar(Dataframe['quality'],Dataframe['alcohol'])
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.show()
```

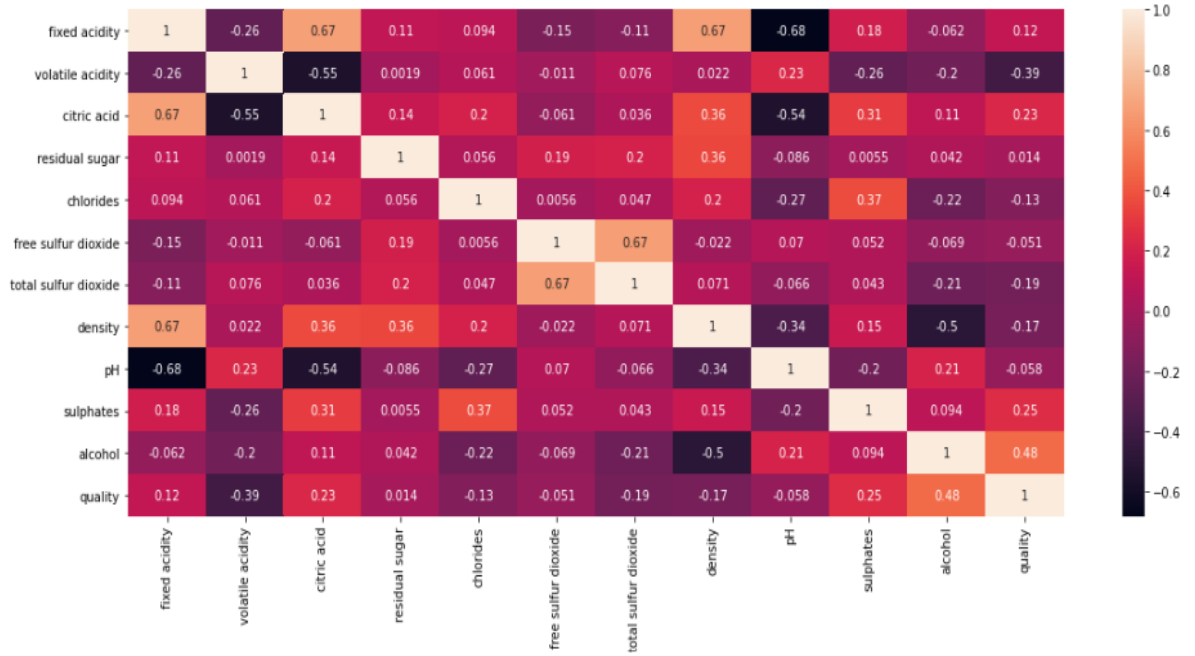


We check how the quality of wine increases with increase the percent of alcohol in the wine.

CHECKING THE CORRELATION:

Here we use statistical method which is used to evaluate the strength of bonding of the relationship between two quantitative variables.

```
# correlation by visualization
plt.figure(figsize=[18,7])
# plot correlation
sb.heatmap(Dataframe.corr(),annot=True)
plt.show()
```



From this correlation visualization, we will find which features are correlated with other features. so we will use a python program to find those features.

By Implementing Svc , Random Forest Classifier, Gradient Decent Classifier In Python We Found The Accuracy Or Quality Level For Wine. These Are The Ways To Find The Accuracy Or Quality Of Wine.

11. CONCLUSIONS AND FUTURE WORK

11.1 CONCLUSION

This report uses the wine dataset r, of Portuguese "Vinho Verde" wine to predict the quality of the wine based on the physicochemical properties. First, we used oversampling to balance the dataset in the data preprocessing stage to optimize the performance of the model. Then we look for features that can provide better prediction results. For this, we used Pearson coefficient correlation matrices and ranked the features according to the high correlation among the features. After applying the sampling datasets which is balancing dataset the performance of the model is improved. In general, removing irrelevant features of the datasets improved the performance of the classification model. To conclude that the minority classes of a dataset will not get a good representation on a classifier and representation for each class can be solved by oversampling and under sampling to balance the representation classes over datasets.

The accuracy of the support vector classifier (SVC) algorithm is 86% from the wine ,the random forest classifier 87% (RFC) from the wine, and the accuracy using stochastic gradient descent classifier 84% from wine, after using Grid Search CV, SVC improves from 86% to 90%.

Among these three machine learning algorithms, we achieved the best accuracy result from the SUPPORT VECTOR CLASIFFIER wine datasets.

Therefore, in the classification algorithms by selecting the appropriate features and balancing the data can improve the performance of the model.

11.2 FUTURE WORK

In the future, to improve the accuracy of the classifier, it is clear that the algorithm or the data must be adjusted. We recommend feature engineering, using potential relationships between wine quality, or applying the boosting algorithm on the more accurate method.

In addition, by applying the other performance measurement and other machine learning algorithms for the better comparison on results. This study will help the manufacturing industries to predict the quality of the different types of wines based on certain features, and also it will be helpful for them to make a good product.

REFERENCES

1. Chawla, N.V., 2005. Data Mining for Imbalanced Datasets: An Overview, in: Maimon, O., Rokach, L. (Eds.), Data Mining and Knowledge Discovery Handbook. Springer US, Boston, MA, pp. 853–867. https://doi.org/10.1007/0-387-25465-X_40
2. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J., 2009. Modeling wine preferences by data mining from physicochemical properties. Decis. Support Syst. 47, 547–553. https://doi.org/10.1016/j.dss.2009.05.016
3. Dahal, K., Dahal, J., Banjade, H., Gaire, S., 2021. Prediction of Wine Quality Using Machine Learning Algorithms. Open J. Stat. 11, 278–289. https://doi.org/10.4236/ojs.2021.112015
4. Dastmard, B., 2013. A statistical analysis of the connection between test results and field claims for ECUs in vehicles.
5. Drummond, C., Holte, R.C., 2003. C4.5, Class Imbalance, and Cost Sensitivity: Why Under-sampling beats Over-sampling. pp. 1–8.
6. Er, Y., Atasoy, A., 2016. The Classification of White Wine and Red Wine According to Their Physicochemical Qualities. Int. J. Intell. Syst. Appl. Eng. 4, 23–26. https://doi.org/10.18201/ijisae.265954
7. Er, Y., Atasoy, Ayten, 2016. The Classification of White Wine and Red Wine According to Their Physicochemical Qualities. Int. J. Intell. Syst. Appl. Eng. 23–26. https://doi.org/10.18201/ijisae.265954.
8. Guo, X., Yin, Y., Dong, C., Yang, G., Zhou, G., 2008. On the Class Imbalance Problem, in: 2008 Fourth International Conference on Natural Computation. Presented at the 2008 Fourth International Conference on Natural Computation, pp. 192–201. https://doi.org/10.1109/ICNC.2008.871
9. Gandhi, R., 2018. Support Vector Machine — Introduction to Machine Learning Algorithms [WWW Document]. Medium. URL https://towardsdatascience.com/support-vector-machineintroduction-to-machine-learning-algorithms-934a444fca47 (accessed 6.6.21).
10. Wolf, L., Shashua, A., 2005. Feature Selection for Unsupervised and Supervised Inference: The Emergence of Sparsity in a WeightBased Approach 33.
11. Yu, T., Zhu, H., 2020. Hyper-Parameter Optimization: A Review of Algorithms and Applications. ArXiv200305689 Cs Stat.
12. Zhang, P., 2000. Neural Networks for Classification: A Survey. Syst. Man Cybern. Part C Appl. Rev. IEEE Trans. On 30, 451–462. https://doi.org/10.1109/5326.89707.

13. Lee, S., Park, J., Kang, K., 2015. Assessing wine quality using a decision tree, in: 2015 IEEE International Symposium on Systems Engineering (ISSE). Presented at the 2015 IEEE International Symposium on Systems Engineering (ISSE), IEEE, Rome, Italy, pp. 176–178. https://doi.org/10.1109/SysEng.2015.7302752.
14. P. Appalasamy, N.D. Rizal, F. Johari, A.F. Mansor, A. Mustapha, 2012. Classification-based Data Mining Approach for Quality Control in Wine Production [WWW Document]. https://doi.org/10.3923/jas.2012.598.601.
15. Panday, D., Cordeiro de Amorim, R., Lane, P., 2018. Process. Lett. 129, 44–52. https://doi.org/10.1016/j.ipl.2017.09.005
16. Onan, A., Korukoğlu, S., 2017. A feature selection model based on genetic rank aggregation for text sentiment classification. J. Inf. Sci. 43, 25–38. https://doi.org/10.1177/0165551515613226.
17. Joseph, R., 2018. Grid Search for model tuning [WWW Document]. Medium. URL https://towardsdatascience.com/grid-search-formodel-tuning-3319b259367e (accessed 6.6.21)
18. Hewahi, N.M., Abu Hamra E, 2017. A Hybrid Approach Based on Genetic Algorithm and Particle Swarm Optimization to Improve Neural Network Classification. J. Inf. Technol. Res. JITR 10, 48–68. https://doi.org/10.4018/JITR.2017070104
19. Fauzi, M., Arifin, A.Z., Gosaria, S., Prabowo, I.S., 2017. Indonesian News Classification Using Naïve Bayes and Two-Phase Feature Selection Model. Indones. J. Electr. Eng. Comput. Sci. 8, 610–615. https://doi.org/10.11591/ijeecs.v8.i3.pp610-615
20. Er, Y., Atasoy, Ayten, 2016. The Classification of White Wine and Red Wine According to Their Physicochemical Qualities. Int. J. Intell. Syst. Appl. Eng. 23–26. https://doi.org/10.18201/ijisae.265954.