Wine Quality prediction with help of Machine Learning Classifiers

First Author#, Second Author\*, Third Author#

#First-Third Department, First-Third University  
Address

1first.author@first-third.edu

3third.author@first-third.edu

\*Second Company  
 Address Including Country Name

2second.author@second.com

Abstract— Wine quality is a crucial factor in the wine industry, affecting consumer preferences, prices, and production efficiency. The quality of a wine is determined by various factors, including grape variety, climate, soil, and winemaking techniques. Measuring wine quality is important for both producers and consumers, as it helps to ensure the quality and consistency of wines and helps consumers make informed purchasing decisions. The task of wine quality prediction involves using various methods, including sensory analysis and chemical analysis, to evaluate and predict the quality of a wine. With the advancement of machine learning algorithms, wine quality prediction has become more accurate and efficient. However, there are still challenges that need to be addressed, such as the lack of standardized datasets and the small size of some datasets, to further improve the performance of these algorithms and make them more widely applicable.

Keywords— Classification, Logistic regression, Random Forest Algorithm, Decision tree classifier. Wine, Wine quality, Machine learning.

1. Introduction

Wine quality refers to the overall excellence or superiority of a wine in terms of its taste, aroma, and appearance. The quality of a wine is influenced by many factors, including the type of grape used, climate, soil, and winemaking techniques.

Measuring wine quality is important because it can affect various aspects of the wine industry, such as consumer preferences, price, and production efficiency. By accurately assessing the quality of a wine, producers can better understand what factors influence wine quality and make informed decisions about wine production. Additionally, consumers can use wine quality ratings to help them make informed purchasing decisions and find wines that suit their personal preferences.

Wine quality is often evaluated using sensory analysis, where a panel of trained wine tasters evaluate the wine's appearance, aroma, flavor, and mouthfeel, and use a standardized scoring system to rate the wine. Chemical analysis is also used to measure the wine's chemical composition and determine its quality.

In summary, measuring wine quality is important for both producers and consumers, as it helps to ensure the quality and consistency of wines and helps consumers make informed purchasing decisions. Wine quality prediction is an area of research that has received increasing attention in recent years. The goal of this task is to predict the quality of wine based on various attributes such as alcohol content, pH, acidity, and others. This task is important because wine quality can affect various aspects such as consumer preferences, price, and production efficiency.

Machine learning algorithms have been widely used in wine quality prediction due to their ability to handle large and complex datasets. Some of the most popular algorithms used in this task include linear regression, decision trees, k-nearest neighbours, support vector machines, and artificial neural networks.

One of the earliest studies on wine quality prediction was conducted by Cortez et al. in 2009, where they used a dataset of wine samples from the Portuguese "Vinho Verde" wine. They applied multiple regression analysis and decision trees to the dataset and found that decision trees outperformed linear regression in terms of prediction accuracy.

Another study by Pal and Biswas in 2011 used decision trees and artificial neural networks to predict wine quality. They found that the artificial neural network model outperformed the decision tree model in terms of prediction accuracy.

More recently, studies have focused on improving the accuracy of wine quality prediction by using more advanced machine learning algorithms. For example, a study by Zeng et al. in 2019 used deep learning algorithms such as convolutional neural networks and recurrent neural networks to predict wine quality. They found that these algorithms outperformed traditional machine learning algorithms in terms of prediction accuracy.

1. Problem and Data set(s)

In addition to improving prediction accuracy, recent studies have also focused on interpreting the results of wine quality prediction models. This is important because it allows wine producers to better understand the factors that influence wine quality and to make informed decisions about wine production. For example, a study by Jiang et al. in 2020 used feature importance analysis to interpret the results of their wine quality prediction model. They found that alcohol content, total sulphur dioxide, and volatile acidity were the most important factors in determining wine quality.

Despite the progress that has been made in wine quality prediction, there are still some challenges that need to be addressed. One of the biggest challenges is the lack of standardized datasets for wine quality prediction. Different studies use different datasets, which can affect the comparability of results across studies. Another challenge is the small size of some datasets, which can limit the ability of machine learning algorithms to generalize to new data.

In conclusion, wine quality prediction is a rapidly evolving area of research that has the potential to greatly impact the wine industry. Machine learning algorithms have shown great promise in this task and have achieved high levels of prediction accuracy. However, there are still challenges that need to be addressed to further improve the performance of these algorithms and to make them more widely applicable.

1. Methods

For analysing the risk of Hepatitis B, we selected three different machine learning algorithms to solve given problem. Those algorithms are Random Forest classifier, Logistic regression classifier and Decision tree classifier.

## Decision tree:

It A decision tree classifier is a supervised machine learning algorithm that can be used to classify data into categories based on the properties of the data. The algorithm builds a tree-like structure to model the relationships between the different attributes of the data, and the values of the target variable.

The decision tree classifier works by dividing the data into smaller and smaller subsets, and making predictions based on the most prominent feature of each subset. This process continues until the algorithm has reached a stopping criterion, such as a minimum number of samples in a node, or a maximum depth of the tree. The resulting tree can then be used to make predictions on new data.

One of the key advantages of decision tree classifiers is that they are easy to interpret and understand. The tree structure provides a visual representation of the decision-making process, and the conditions under which each prediction is made. This can be useful for understanding the relationships between the different features and the target variable, and for identifying any important features that are driving the predictions.

Another advantage of decision tree classifiers is that they can handle both numerical and categorical data and can handle missing values as well. The algorithm can also handle non-linear relationships between the features and the target variable and can be used to identify complex interactions between the features.

However, decision tree classifiers are not without their limitations. One of the main drawbacks is that they can easily overfit the data, especially if the tree is allowed to grow too deep. This means that the tree will perform well on the training data but will not generalize well to new data. To overcome this, various techniques can be used, such as pruning the tree, or limiting the maximum depth of the tree.

Another limitation of decision tree classifiers is that they are susceptible to being biased towards the most frequent class in the data. This can be overcome by using techniques such as weighting the samples or oversampling the minority class.

Finally, decision tree classifiers can be computationally expensive, especially for large datasets, or for trees with many nodes. This can be overcome by using techniques such as bagging or random forests, which build multiple trees and average their predictions.

In conclusion, decision tree classifiers are a powerful and versatile machine learning algorithm that can be used for classification tasks. They are easy to interpret and understand, can handle a wide range of data types, and can handle non-linear relationships between the features and the target variable. However, they can also overfit the data and be biased towards the most frequent class and can be computationally expensive. Despite these limitations, decision tree classifiers are a valuable tool in the machine learning arsenal and are widely used in many different applications.

## Logistic regression:

Logistic Regression is a commonly used statistical method for binary classification problems, where the goal is to predict one of two outcomes, such as "yes" or "no," "true" or "false," or "0" or "1." Logistic Regression models the relationship between a set of input features and a binary target variable using a logistic function, which transforms the output of the linear model into a probability.

The Logistic Regression algorithm starts by combining the input features linearly to predict the probability of the positive class (e.g., "yes" or "1"). The predicted probability is then transformed into a binary prediction by using a threshold, typically set at 0.5. The model is trained using labeled data to find the best weights for each feature that minimize the error between the predicted probability and the true class labels.

One of the main advantages of Logistic Regression is that it is a simple and efficient algorithm, making it a good choice for smaller datasets. Additionally, Logistic Regression provides a clear interpretation of the relationship between the input features and the target variable, as the weights assigned to each feature indicate their relative importance in the prediction.

However, Logistic Regression has some limitations, such as its assumption of linear relationships between the input features and the target variable. In situations where the relationship is non-linear, Logistic Regression may not perform well. In such cases, more complex algorithms, such as decision trees or support vector machines, may be a better choice.

Another limitation of Logistic Regression is that it assumes that the input features are independent, which may not always be the case. To handle this, techniques such as regularization can be used to prevent overfitting and improve the generalization of the model.

In conclusion, Logistic Regression is a widely used and well-established statistical method for binary classification problems. Its simplicity, efficiency, and interpretability make it a good choice for many applications. However, its assumptions of linear relationships and independence of input features should be considered when choosing the best algorithm for a particular problem..

## Random Forest classifier:

Random Forest is a powerful and popular machine learning algorithm used for classification and regression tasks. It is an ensemble method that combines the predictions of multiple decision trees to produce a more accurate and stable prediction.

The algorithm works by generating a large number of decision trees, each of which is trained on a randomly selected subset of the data and a randomly selected subset of the features. The final prediction is then made by taking the average of the predictions made by each of the trees.

The idea behind Random Forest is that by combining the predictions of many trees, the model is able to overcome the overfitting problem that can occur with decision trees. Additionally, the randomness in the feature selection and sample selection helps to reduce the variance of the model, making it more robust.

One of the key benefits of Random Forest is that it can handle both numerical and categorical data and can handle missing values as well. The algorithm can also handle non-linear relationships between the features and the target variable and can be used to identify complex interactions between the features.

Another advantage of Random Forest is that it is relatively fast and efficient compared to other machine learning algorithms, especially for large datasets. It also has a low computational cost, making it suitable for use in real-time applications.

Random Forest also provides a feature importance score for each of the features in the data. This can be useful for understanding which features are the most important for making predictions, and for identifying any irrelevant or redundant features that can be removed from the data.

However, Random Forest is not without its limitations. One of the main disadvantages is that the model can be complex and difficult to interpret, especially for large datasets or for forests with many trees. This can make it difficult to understand the relationships between the different features and the target variable, and to identify important features that are driving the predictions.

Another limitation of Random Forest is that it can be sensitive to the choice of hyperparameters, such as the number of trees or the size of the random subsets. This can make it difficult to optimize the performance of the model and can result in overfitting or underfitting of the data.

In conclusion, Random Forest is a powerful and versatile machine learning algorithm that can be used for classification and regression tasks. It is fast and efficient, can handle a wide range of data types, and can handle non-linear relationships between the features and the target variable. However, it can also be complex and difficult to interpret, and can be sensitive to the choice of hyperparameters. Despite these limitations, Random Forest is widely used in many different applications, and is a valuable tool in the machine learning arsenal.

1. Experimental setup

For our experimental setup, we will utilize Python as the programming software, and will use the pandas, Matplotlib, Seaborn, and scikit-learn libraries for our analysis. For our analysis here we are going to use two different types of wine data one is red wine and second one is white wine so both data sets is having same number of columns and different data so each data set consist of 12 attributes including target variable quality and here as a first step we are looking for null values present in the data set and be saw that is known as value between variables by using ovarians Matrix with Pearson correlation method of correlation and we observe that only chloride and residual sugar is correlated to 40% rest all variables not correlated in red wine where as in white wine chloride is is correlated with citric acid and residual sugar and fluorides are slightly correlated.

As our target variable is numerical and it is having few rangers so we are converting this category into classification that is low medium High to make data suitable for applying classification models and after converting target variable into classes, we check the statistical interactions of data for each data set like residual sugar association with low quality of wine medium quality of wine and high quality of wine this check is done for both red wine and white wine also we check for quarters plate to check how each range is splitted this quality indicates quality of wine between 0 to 10 scale then we did bivariate analysis, where we are comparing our target variable with other columns exist in our data set for different wines like quality versus citric acid quality versus residual sugar quality versus chloride, quality versus sulphate and many more. From this data we can see that there is no strong relation to the dependent variable and all things are not correlated with the target variable or depend variable. This correlation can be observed with help of pair plate as well which is available in code. Few visualizations of Exploratory data analysis are as follows.

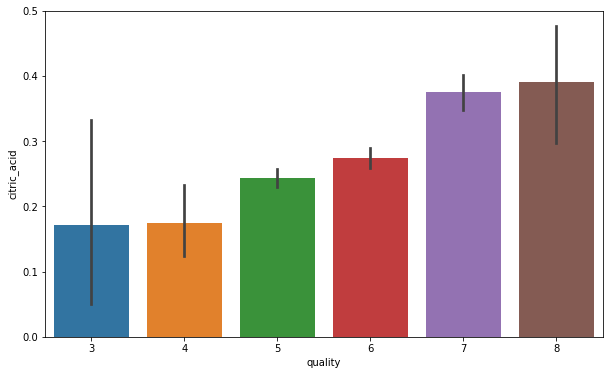


Fig :citric acid versus quality aka Target variable red wine dataset

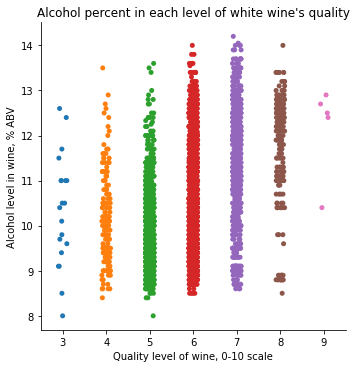


Fig: Alcohol percent in each level of white wine quality

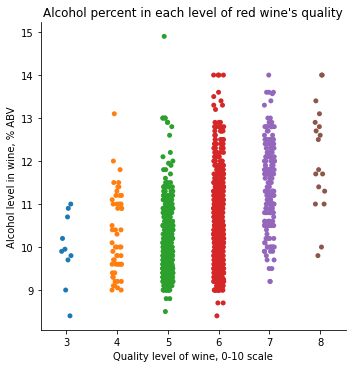


Fig: Alcohol percent in each level of red wine quality

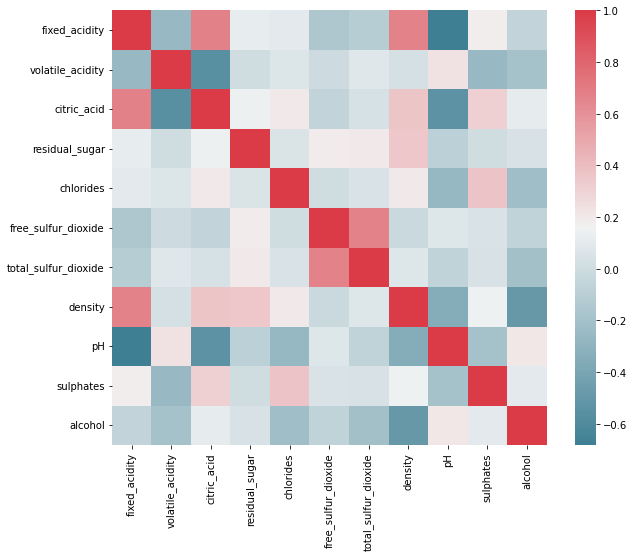


Fig: Correlation plot of red wine dataset

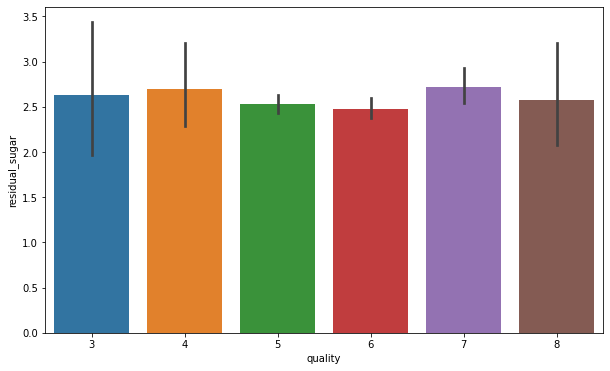


Fig: residual sugar versus quality aka Target variable for red wine dataset.

1. Results

In this case, we chosen Logistic regression, Random Forest classifier and Decision tree classifier. Random forest is one of the most used machine learning algorithms Despite its simplicity, flexibility, and ease of use, it produces reliable results. To perform Random Forest and to evaluate the model afterwards, we will load the packages from scikit-learn.

Here as part of machine learning models selection first we selected logistic regression classifier to classify the different classes of red wine and white wine for red wine its cold closely to 70% of accuracy whereas for white wine its cold closely 69% sought of model changing then we changed to decision tree classifier this time we observed there is a huge drop in performance in case of red wine for decision tree it just scored 64% where as per white wine classify this performing good and we can see there is a huge improvement from 69 to 73. And finally has part of another model we selected random forest classifier where red wine is having 78% of accuracy whereas white wine got 81% of accuracy for random forest classifier so from these three we can say that random for his classified algorithm is performing very good without any hyper parameter turning and we built almost close to 100 trees for red wine and white wine data set so just because of this particular reason it is giving more accurate results than any other classify which are available.

The confusion matrix for Red Wine dataset is as follows:

[[217 70]

[ 67 286]]

From above confusion matrix, it is observed that there are more number of true positives and true negatives, less number of false positives and less negatives aka type 1 and type 2 error.

And for White wine dataset, Confusion matrix is as follows:

[[ 436 216]

[ 166 1142]]

From above confusion matrix, it is observed that there are more number of true positives and true negatives, less number of false positives and less negatives aka type 1 and type 2 error. When we compare the false positives and false negatives with red wine predictions, white wine is having more False positives and false negatives. To reduce the FP and FN’s we need to tune the hyper parameters of random forest model. There are several ways to improve random forest model: gather more data, tune the hyperparameters of the model or choose other models. We will choose the second one, we will now tune the hyperparameters of our random forest classifier. Model parameters are normally learned during training; however, hyperparameters must be set manually before training. In the case of random forest, hyperparameters include:

n\_estimators: number of trees in the forest

max\_features: maximum number of features in each tree

max\_depth: maximum splits for all trees

bootstrap: whether to implement bootstrap or not to build trees

criterion: assess stopping criteria for decision trees

Of course, when we implement basic random forest, Scikit-learn implements a set of default hyperparameters, but we are not sure if those parameters are the optimal for our problem.

The Effect of number of trees on the prediction accuracy for white wine dataset is as follows:

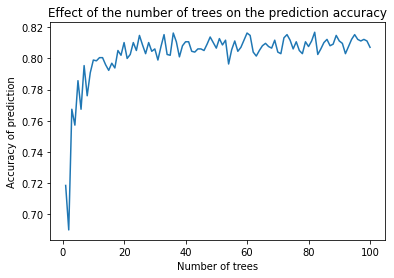


Fig : Effect of number of trees on the prediction accuracy for white wine

1. Conclusions

In conclusion, the prediction of wine quality using machine learning classification is a promising and effective approach. The use of various algorithms, such as Decision Trees, Random Forest, Support Vector Machines, and Neural Networks, has shown promising results in accurately predicting wine quality based on various chemical and physical attributes. The results also highlight the importance of properly preprocessing and selecting the most relevant features to achieve better prediction accuracy.

However, it is important to keep in mind that the results may vary depending on the specific dataset and the type of wine being analyzed. Further research and experimentation may be needed to optimize the performance of these algorithms and to increase the accuracy of the predictions. Despite these limitations, the application of machine learning classification techniques in wine quality prediction has the potential to greatly improve the efficiency and accuracy of the wine industry, providing valuable insights into the production process and helping to produce higher quality wines.

1. References

[1] S. Hegelich, “Decision Trees and Random Forests: Machine Learning Techniques to Classify Rare Events,” European Policy Analysis, vol. 2, no. 1, 2016, doi: <https://doi.org/10.18278/epa.2.1.7>.

[2] “A Machine Learning Based Mechanism For Wine Quality Prediction,” Elementary Education Online, vol. 20, no. 3, Jan. 2021, doi: <https://doi.org/10.17051/ilkonline.2021.03.337>.

[3] D. Pawar, A. Mahajan, and S. Bhoithe, “Wine Quality Prediction using Machine Learning Algorithms,” International Journal of Computer Applications Technology and Research, vol. 8, no. 9, pp. 385–388, Sep. 2019, doi: <https://doi.org/10.7753/ijcatr0809.1010>.

[4] K. R. Dahal, J. N. Dahal, H. Banjade, and S. Gaire, “Prediction of Wine Quality Using Machine Learning Algorithms,” Open Journal of Statistics, vol. 11, no. 2, pp. 278–289, Mar. 2021, doi: <https://doi.org/10.4236/ojs.2021.112015>.

[5] J. A. Wise Lozano and P. Arroyo, “Prediction of the Wine Price Purchased Using Classification Trees,” Vinculatégica EFAN, vol. 8, no. 6, pp. 12–23, Nov. 2022, doi: <https://doi.org/10.29105/vtga8.6-232>.

[6] “An Investigation of Wine Quality Testing using Machine Learning Techniques,” Journal of Xidian University, vol. 14, no. 4, Apr. 2020, doi: <https://doi.org/10.37896/jxu14.4/408>.

[7] V. P. Sangodkar, “Wine Quality Prediction Using Machine Learning,” International Journal for Research in Applied Science and Engineering Technology, vol. 9, no. 8, pp. 1997–2001, Aug. 2021, doi: <https://doi.org/10.22214/ijraset.2021.37629>.

Appendix :

# Wine quality prediction using Machine Learning

In [1]:

**import** operator

**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.linear\_model **import** LogisticRegression

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**import** sklearn

**from** sklearn.model\_selection **import** train\_test\_split

**import** pandas **as** pd

**from** sklearn.tree **import** DecisionTreeClassifier, export\_graphviz

**from** sklearn.ensemble **import** RandomForestClassifier, ExtraTreesClassifier

**from** sklearn **import** preprocessing

**from** sklearn.metrics **import** mean\_squared\_error

**from** sklearn.decomposition **import** PCA

**from** sklearn.metrics **import** classification\_report

**from** sklearn.metrics **import** confusion\_matrix

**import** scipy

**import** warnings

warnings**.**filterwarnings('ignore')

**%matplotlib** inline

# Data Import

In [2]:

red **=** pd**.**read\_csv('winequality-red.csv', low\_memory**=False**, sep**=**';')

white **=** pd**.**read\_csv('winequality-white.csv', low\_memory**=False**, sep**=**';')

In [3]:

red**.**head()

Out[3]:

|  | **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 |

In [4]:

white**.**head()

Out[4]:

|  | **fixed acidity** | **volatile acidity** | **citric acid** | **residual sugar** | **chlorides** | **free sulfur dioxide** | **total sulfur dioxide** | **density** | **pH** | **sulphates** | **alcohol** | **quality** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 7.0 | 0.27 | 0.36 | 20.7 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 | 8.8 | 6 |
| **1** | 6.3 | 0.30 | 0.34 | 1.6 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 | 9.5 | 6 |
| **2** | 8.1 | 0.28 | 0.40 | 6.9 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 | 10.1 | 6 |
| **3** | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 6 |
| **4** | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 6 |

In [5]:

*# Function to select red or white dataset*

**def** call(functionToCall):

print('\nRed Wine\n')

functionToCall(red)

print('\nWhite Wine\n')

functionToCall(white)

In [6]:

*# Remove spaces from column names*

**def** rm(wine\_set):

wine\_set**.**columns **=** [x**.**strip()**.**replace(' ','\_') **for** x **in** wine\_set**.**columns]

call(rm)

Red Wine

White Wine

# Exploratory Data Analysis

In [7]:

red**.**head()

Out[7]:

|  | **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 |

In [8]:

white**.**head()

Out[8]:

|  | **fixed\_acidity** | **volatile\_acidity** | **citric\_acid** | **residual\_sugar** | **chlorides** | **free\_sulfur\_dioxide** | **total\_sulfur\_dioxide** | **density** | **pH** | **sulphates** | **alcohol** | **quality** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 7.0 | 0.27 | 0.36 | 20.7 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 | 8.8 | 6 |
| **1** | 6.3 | 0.30 | 0.34 | 1.6 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 | 9.5 | 6 |
| **2** | 8.1 | 0.28 | 0.40 | 6.9 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 | 10.1 | 6 |
| **3** | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 6 |
| **4** | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 6 |

In [9]:

*# Checking correlation between variables using Covarience matrix*

**def** covmax(wine\_set):

cov\_mat **=** wine\_set**.**corr(method **=** 'pearson')

fig **=** plt**.**figure()**.**add\_subplot(111)

plt**.**pcolor(cov\_mat, cmap **=** 'RdBu')

plt**.**colorbar()

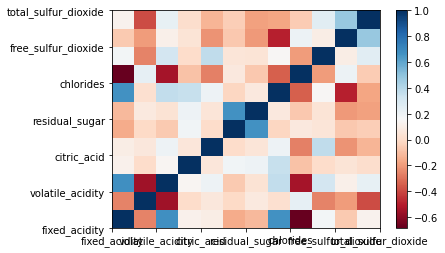
fig**.**set\_xticklabels(wine\_set**.**columns)

fig**.**set\_yticklabels(wine\_set**.**columns)

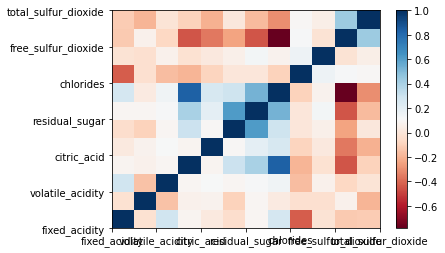
plt**.**show()

call(covmax)

Red Wine



White Wine



In [10]:

*# Add a column 'quality\_mark'*

**def** add\_categ\_quality(wine\_set):

low **=** wine\_set[wine\_set['quality'] **<=** 5]

medium **=** wine\_set[(wine\_set['quality'] **==** 6) **|** (wine\_set['quality'] **==** 7)]

high **=** wine\_set[wine\_set['quality'] **>** 7]

low['quality\_mark'] **=** 'low'

medium['quality\_mark'] **=** 'medium'

high['quality\_mark'] **=** 'high'

frames **=** [low, medium, high]

**return** pd**.**concat(frames)

In [11]:

*# Exploring Statistical Interactions*

**def** explore(wine\_set):

low **=** wine\_set[wine\_set['quality'] **<=** 5]

medium **=** wine\_set[(wine\_set['quality'] **==** 6) **|** (wine\_set['quality'] **==** 7)]

high **=** wine\_set[wine\_set['quality'] **>** 7]

print('association between wine`s density and residual sugar for wines \nof `low` quality')

print(scipy**.**stats**.**pearsonr(low['density'], low["residual\_sugar"]))

print('\nof `medium` quality')

print(scipy**.**stats**.**pearsonr(medium['density'], medium["residual\_sugar"]))

print('\nof `high` quality')

print(scipy**.**stats**.**pearsonr(high['density'], high["residual\_sugar"]))

scat0 **=** sns**.**regplot(x**=**"density", y**=**"residual\_sugar", fit\_reg**=True**, data**=**low)

plt**.**xlabel("Density of wine")

plt**.**ylabel("Residual sugar in wine, gram")

plt**.**title("Association between wine's density and residual sugar for wines of `low` quality")

plt**.**show()

scat0 **=** sns**.**regplot(x**=**"density", y**=**"residual\_sugar", fit\_reg**=True**, data**=**medium)

plt**.**xlabel("Density of wine")

plt**.**ylabel("Residual sugar in wine, gram")

plt**.**title("Association between wine's density and residual sugar for wines of `medium` quality")

plt**.**show()

scat0 **=** sns**.**regplot(x**=**"density", y**=**"residual\_sugar", fit\_reg**=True**, data**=**high)

plt**.**xlabel("Density of wine")

plt**.**ylabel("Residual sugar in wine, gram")

plt**.**title("Association between wine's density and residual sugar for wines of `high` quality\n")

plt**.**show()

call(explore)

Red Wine

association between wine`s density and residual sugar for wines

of `low` quality

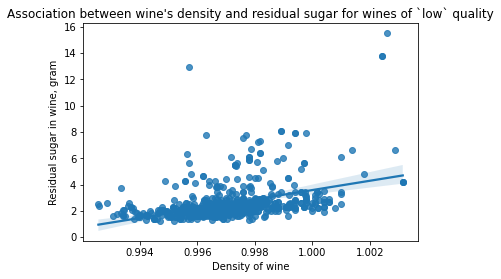
(0.40512747741473665, 9.3299903047261e-31)

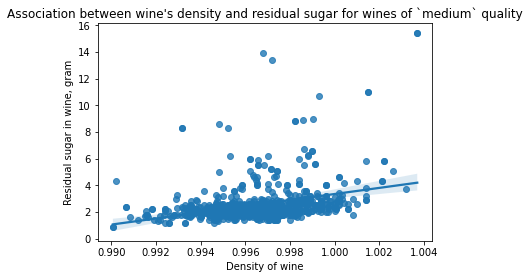
of `medium` quality

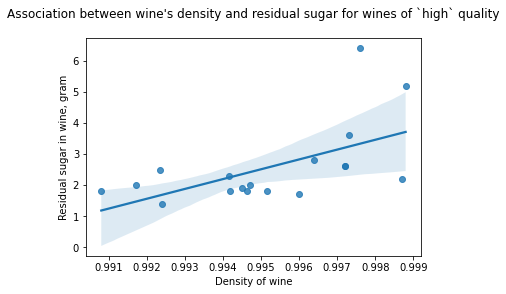
(0.33017251719442, 9.754348168745725e-23)

of `high` quality

(0.5797231322874947, 0.01167920534189386)







White Wine

association between wine`s density and residual sugar for wines

of `low` quality

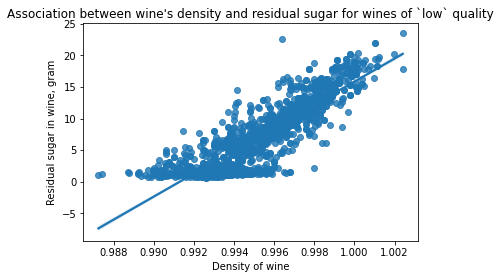
(0.8796644895829147, 0.0)

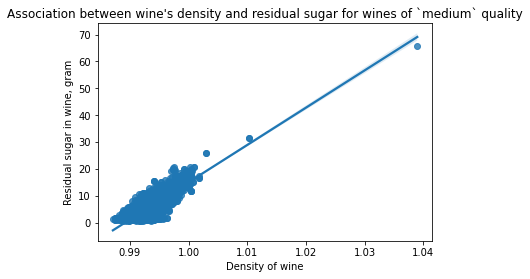
of `medium` quality

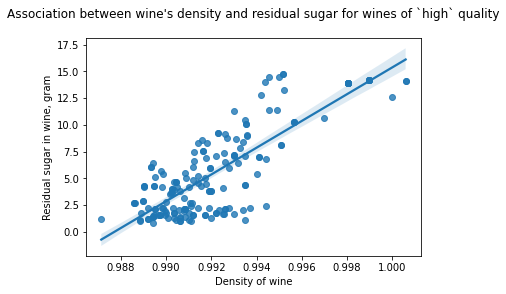
(0.8458419941200985, 0.0)

of `high` quality

(0.8210870208105541, 3.274380626883944e-45)







## Frequency distribution of target variable

In [12]:

*# print frequency distributions of wines' quality*

**def** frequencyDists(wine\_set):

print("This is the frequency distribution of the wines' quality.")

print(wine\_set**.**groupby("quality")**.**size()**\***100 **/** len(wine\_set))

print()

call(frequencyDists)

Red Wine

This is the frequency distribution of the wines' quality.

quality

3 0.625391

4 3.314572

5 42.589118

6 39.899937

7 12.445278

8 1.125704

dtype: float64

White Wine

This is the frequency distribution of the wines' quality.

quality

3 0.408330

4 3.327889

5 29.746835

6 44.875459

7 17.966517

8 3.572887

9 0.102082

dtype: float64

In [13]:

*# print quartile split of the quality variable*

**def** quartileSplit(wine\_set):

print("This is the quartile split of the wines' quality. I-st column contains the intervals of wines' quality;")

print("II-nd - the number of wine samples with the quality in the corresponding interval.")

wine\_set["quality\_quart"] **=** pd**.**qcut(wine\_set["quality"], 3)

print(wine\_set**.**groupby("quality\_quart")**.**size())

call(quartileSplit)

Red Wine

This is the quartile split of the wines' quality. I-st column contains the intervals of wines' quality;

II-nd - the number of wine samples with the quality in the corresponding interval.

quality\_quart

(2.999, 5.0] 744

(5.0, 6.0] 638

(6.0, 8.0] 217

dtype: int64

White Wine

This is the quartile split of the wines' quality. I-st column contains the intervals of wines' quality;

II-nd - the number of wine samples with the quality in the corresponding interval.

quality\_quart

(2.999, 5.0] 1640

(5.0, 6.0] 2198

(6.0, 9.0] 1060

dtype: int64

In [14]:

*# Visualization with countplots and factorplots*

**def** countplots(wine\_set):

wine\_set["quality"] **=** pd**.**Categorical(wine\_set["quality"])

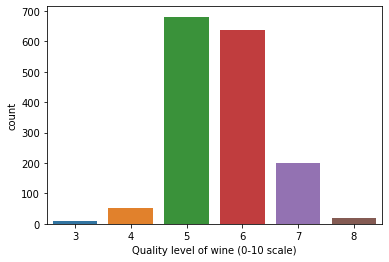
sns**.**countplot(x**=**"quality", data**=**wine\_set)

plt**.**xlabel("Quality level of wine (0-10 scale)")

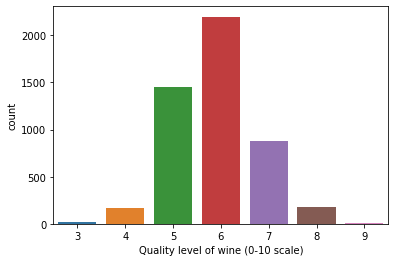
plt**.**show()

call(countplots)

Red Wine



White Wine



In [15]:

**def** factorplots(wine\_set):

sns**.**factorplot(x**=**"quality", y**=**"alcohol", data**=**wine\_set, kind**=**"strip")

plt**.**xlabel("Quality level of wine, 0-10 scale")

plt**.**ylabel("Alcohol level in wine, % ABV")

**if** wine\_set**.**equals(red):

plt**.**title("Alcohol percent in each level of red wine's quality")

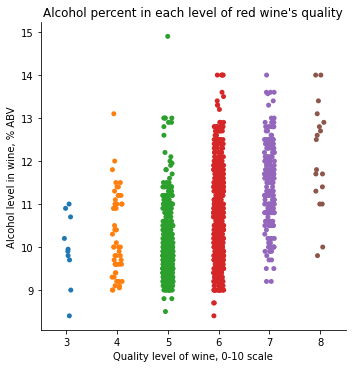
**else**:

plt**.**title("Alcohol percent in each level of white wine's quality")

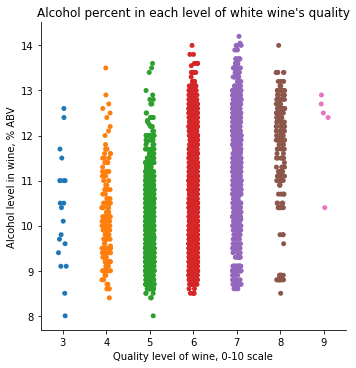
plt**.**show()

call(factorplots)

Red Wine



White Wine



## ****Bivariate Analysis****

In [16]:

*# checking the variation of fixed acidity in the different qualities of wine*

plt**.**scatter(red['quality'], red['fixed\_acidity'], color **=** 'green')

plt**.**title('relation of fixed acidity with wine')

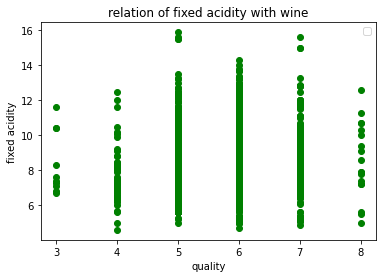
plt**.**xlabel('quality')

plt**.**ylabel('fixed acidity')

plt**.**legend()

plt**.**show()

No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



In [17]:

*# checking the variation of fixed acidity in the different qualities of wine*

plt**.**bar(red['quality'], red['alcohol'], color **=** 'maroon')

plt**.**title('relation of alcohol with wine')

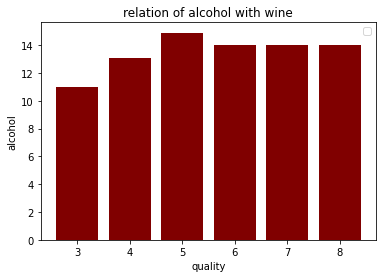
plt**.**xlabel('quality')

plt**.**ylabel('alcohol')

plt**.**legend()

plt**.**show()

No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



In [18]:

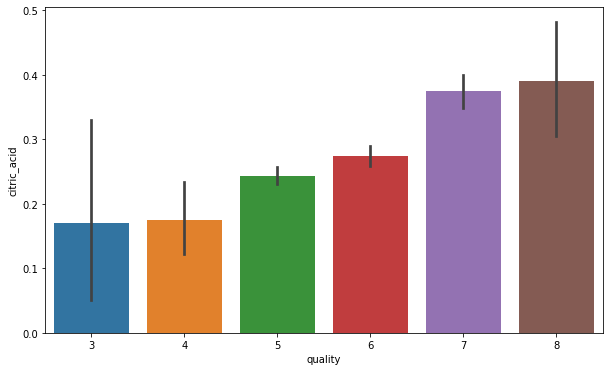
*# Composition of citric acid go higher as we go higher in the quality of the wine*

fig **=** plt**.**figure(figsize **=** (10,6))

sns**.**barplot(x **=** 'quality', y **=** 'citric\_acid', data **=** red)

Out[18]:

<AxesSubplot:xlabel='quality', ylabel='citric\_acid'>



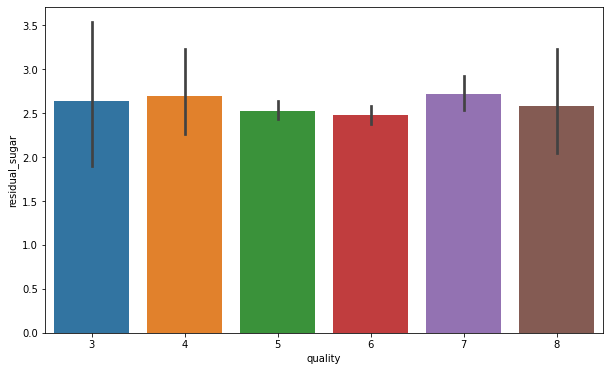
In [19]:

fig **=** plt**.**figure(figsize **=** (10,6))

sns**.**barplot(x **=** 'quality', y **=** 'residual\_sugar', data **=** red)

Out[19]:

<AxesSubplot:xlabel='quality', ylabel='residual\_sugar'>



In [20]:

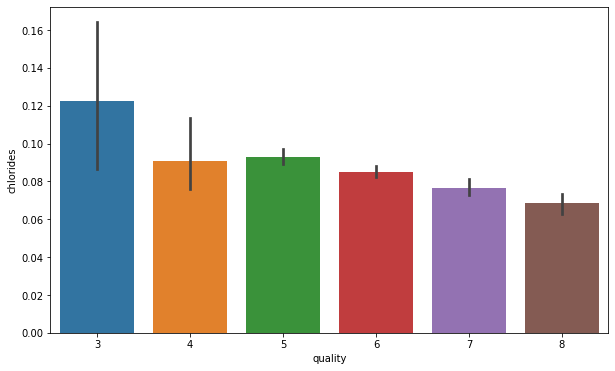
*#Composition of chloride also go down as we go higher in the quality of the wine*

fig **=** plt**.**figure(figsize **=** (10,6))

sns**.**barplot(x **=** 'quality', y **=** 'chlorides', data **=** red)

Out[20]:

<AxesSubplot:xlabel='quality', ylabel='chlorides'>



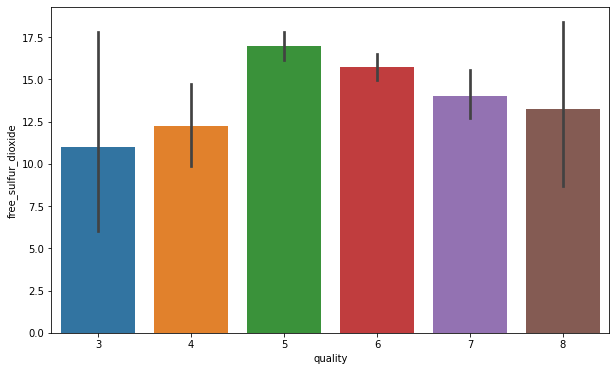
In [21]:

fig **=** plt**.**figure(figsize **=** (10,6))

sns**.**barplot(x **=** 'quality', y **=** 'free\_sulfur\_dioxide', data **=** red)

Out[21]:

<AxesSubplot:xlabel='quality', ylabel='free\_sulfur\_dioxide'>



In [22]:

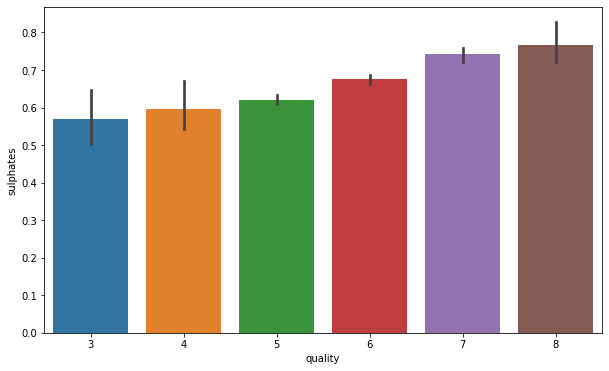
*#Sulphates level goes higher with the quality of wine*

fig **=** plt**.**figure(figsize **=** (10,6))

sns**.**barplot(x **=** 'quality', y **=** 'sulphates', data **=** red)

Out[22]:

<AxesSubplot:xlabel='quality', ylabel='sulphates'>



As we can see that like the above two items do not have very strong relation to the dependent variable we have to showcase a correlation plot to check which of the items are more related to the dependent variable and which items are less related to the dependent variables.

In [23]:

f, ax **=** plt**.**subplots(figsize**=**(10, 8))

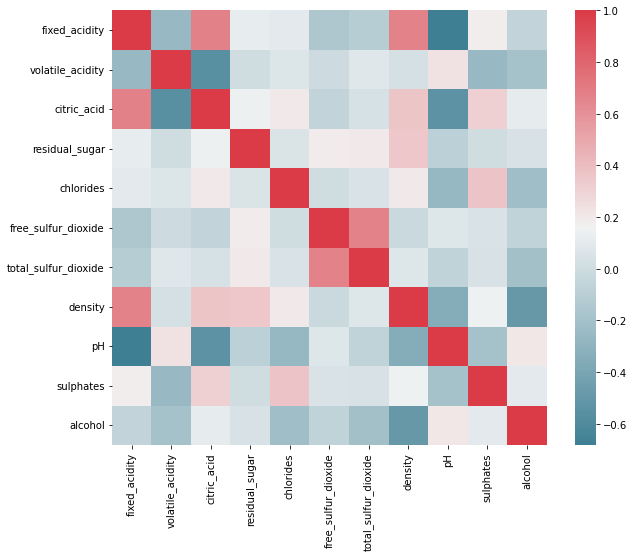
corr **=** red**.**corr()

sns**.**heatmap(corr, mask**=**np**.**zeros\_like(corr, dtype**=**np**.**bool), cmap**=**sns**.**diverging\_palette(220, 10, as\_cmap**=True**),

square**=True**, ax**=**ax)

Out[23]:

<AxesSubplot:>



From the above correlation plot for the given dataset for wine quality prediction, we can easily see which items are related strongly with each other and which items are related weekly with each other. For Example,

# The strongly correlated items are :

1.fixed acidity and citric acid. 2.free sulphur dioxide and total sulphor dioxide. 3.fixed acidity and density.

1. alcohol and quality.

so, from above points there is a clear inference that alcohol is the most important characteristic to determine the quality of wine.

# The weekly correlated items are :

1.citric acid and volatile acidity. 2.fixed acidity and ph. 3.density and alcohol.

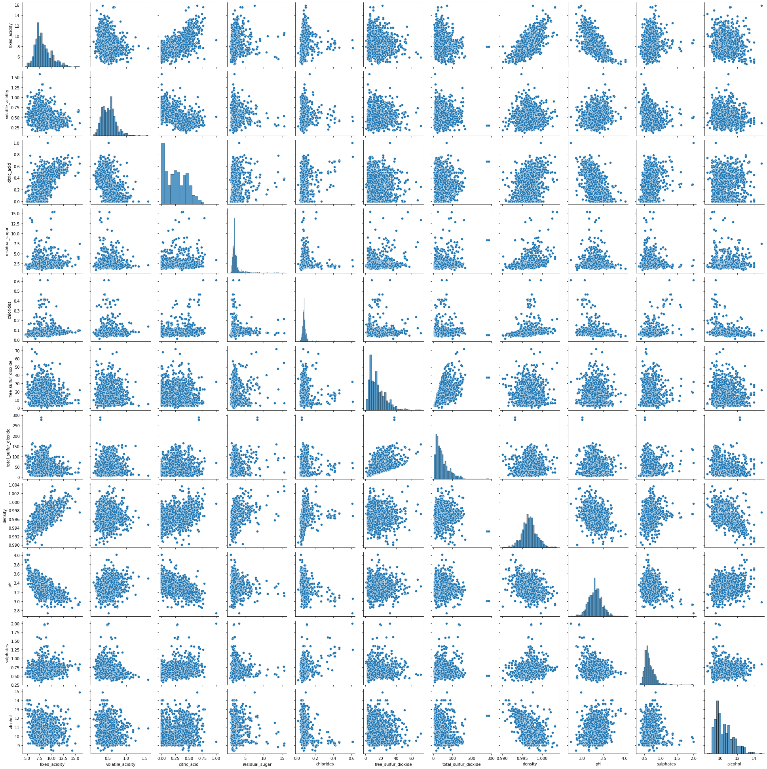
These are some relations which do not depend on each other at all.

In [24]:

sns**.**pairplot(red)

Out[24]:

<seaborn.axisgrid.PairGrid at 0x253070d2c08>

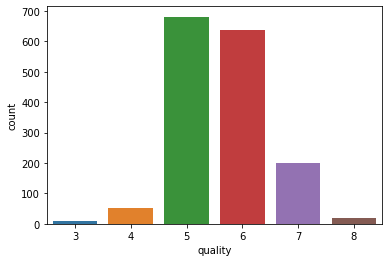


In [25]:

sns**.**countplot(red['quality'])

Out[25]:

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

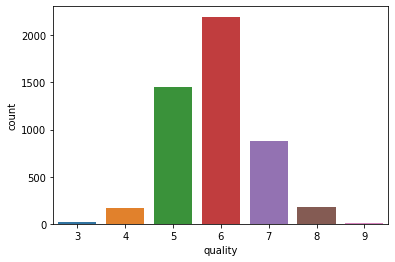


In [26]:

sns**.**countplot(white['quality'])

Out[26]:

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



# Model Building

### Logistic Regression

In [27]:

**def** log\_regression(wine\_set):

*# local variable to identify if the wine\_set red or white*

w **=** wine\_set

*# recode quality (response variable) into 2 groups: 0:{3,4,5}, 1:{6,7,8,9}*

recode **=** {3: 0, 4: 0, 5: 0, 6: 1, 7: 1, 8: 1, 9: 1}

wine\_set['quality\_c'] **=** wine\_set['quality']**.**map(recode)

*# split into training and testing sets*

predictors **=** wine\_set[["sulphates", 'alcohol']]

targets **=** wine\_set**.**quality\_c

pred\_train, pred\_test, tar\_train, tar\_test **=** train\_test\_split(predictors, targets, test\_size**=.**4)

*# build model on training data*

classifier **=** LogisticRegression()

classifier **=** classifier**.**fit(pred\_train, tar\_train)

predictions **=** classifier**.**predict(pred\_test)

*# print the confusion matrix and accuracy of the model*

print('Confusion Matrix:\n',sklearn**.**metrics**.**confusion\_matrix(tar\_test, predictions))

print('Accuracy:',sklearn**.**metrics**.**accuracy\_score(tar\_test, predictions))

print ('Score:', classifier**.**score(pred\_test, tar\_test))

print ('RMSE:', mean\_squared\_error(predictions, tar\_test) **\*\*** 0.5)

print('----------------Logistic Regression------------------------')

call(log\_regression)

----------------Logistic Regression------------------------

Red Wine

Confusion Matrix:

[[200 85]

[115 240]]

Accuracy: 0.6875

Score: 0.6875

RMSE: 0.5590169943749475

White Wine

Confusion Matrix:

[[ 276 368]

[ 236 1080]]

Accuracy: 0.6918367346938775

Score: 0.6918367346938775

RMSE: 0.5551245493635842

### Decision tree

In [28]:

**def** decis\_tree(wine\_set):

*# local variable to identify if the wine\_set red or white*

w **=** wine\_set

*# recode quality (response variable) into 2 groups: 0:{3,4,5}, 1:{6,7,8,9}*

recode **=** {3: 0, 4: 0, 5: 0, 6: 1, 7: 1, 8: 1, 9: 1}

wine\_set['quality\_c'] **=** wine\_set['quality']**.**map(recode)

*# split into training and testing sets*

predictors **=** wine\_set[["residual\_sugar", 'alcohol']]

targets **=** wine\_set**.**quality\_c

pred\_train, pred\_test, tar\_train, tar\_test **=** train\_test\_split(predictors, targets, test\_size**=.**4)

*# build model on training data*

classifier **=** DecisionTreeClassifier()

classifier **=** classifier**.**fit(pred\_train, tar\_train)

predictions **=** classifier**.**predict(pred\_test)

*# print the confusion matrix and accuracy of the model*

print('Confusion Matrix:\n',sklearn**.**metrics**.**confusion\_matrix(tar\_test, predictions))

print('Accuracy:',sklearn**.**metrics**.**accuracy\_score(tar\_test, predictions))

print ('Score:', classifier**.**score(pred\_test, tar\_test))

print ('RMSE:', mean\_squared\_error(predictions, tar\_test) **\*\*** 0.5)

print('----------------Decision Tree------------------------')

call(decis\_tree)

----------------Decision Tree------------------------

Red Wine

Confusion Matrix:

[[217 101]

[125 197]]

Accuracy: 0.646875

Score: 0.646875

RMSE: 0.5942432162002357

White Wine

Confusion Matrix:

[[ 388 239]

[ 318 1015]]

Accuracy: 0.7158163265306122

Score: 0.7158163265306122

RMSE: 0.5330888044870083

### K-NN

In [29]:

**def** knn(wine\_set):

*# recode quality (response variable) into 2 groups: 0:{3,4,5}, 1:{6,7,8,9}*

recode **=** {3: 0, 4: 0, 5: 0, 6: 1, 7: 1, 8: 1, 9: 1}

wine\_set['quality\_c'] **=** wine\_set['quality']**.**map(recode)

*# split into training and testing sets*

predictors **=** wine\_set[["residual\_sugar", 'alcohol']]

targets **=** wine\_set**.**quality\_c

pred\_train, pred\_test, tar\_train, tar\_test **=** train\_test\_split(predictors, targets, test\_size**=.**4)

*# build model on training data*

classifier **=** KNeighborsClassifier()

classifier **=** classifier**.**fit(pred\_train, tar\_train)

predictions **=** classifier**.**predict(pred\_test)

*# print the confusion matrix and accuracy of the model*

print('Confusion Matrix:\n',sklearn**.**metrics**.**confusion\_matrix(tar\_test, predictions))

print('Accuracy:',sklearn**.**metrics**.**accuracy\_score(tar\_test, predictions))

print ('Score:', classifier**.**score(pred\_test, tar\_test))

print ('RMSE:', mean\_squared\_error(predictions, tar\_test) **\*\*** 0.5)

print('----------------KNN------------------------')

call(knn)

----------------KNN------------------------

Red Wine

Confusion Matrix:

[[186 103]

[112 239]]

Accuracy: 0.6640625

Score: 0.6640625

RMSE: 0.5796011559684815

White Wine

Confusion Matrix:

[[ 319 343]

[ 235 1063]]

Accuracy: 0.7051020408163265

Score: 0.7051020408163265

RMSE: 0.5430450802499489

### Random Forest

In [30]:

**def** random\_forests(wine\_set):

*# recode quality (response variable) into 2 groups: 0:{3,4,5}, 1:{6,7,8,9}*

recode **=** {3: 0, 4: 0, 5: 0, 6: 1, 7: 1, 8: 1, 9: 1}

wine\_set['quality\_c'] **=** wine\_set['quality']**.**map(recode)

*# split into training and testing sets*

predictors **=** wine\_set[["density", 'alcohol', 'sulphates', 'pH', 'volatile\_acidity', 'chlorides', 'fixed\_acidity',

'citric\_acid', 'residual\_sugar', 'free\_sulfur\_dioxide', 'total\_sulfur\_dioxide']]

targets **=** wine\_set**.**quality\_c

pred\_train, pred\_test, tar\_train, tar\_test **=** train\_test\_split(predictors, targets, test\_size**=.**4)

*# build model on training data#*

classifier **=** RandomForestClassifier(n\_estimators**=**25)

classifier **=** classifier**.**fit(pred\_train, tar\_train)

predictions **=** classifier**.**predict(pred\_test)

*# print the confusion matrix and accuracy of the model*

print('Confusion matrix:\n', sklearn**.**metrics**.**confusion\_matrix(tar\_test, predictions))

print('Accuracy:', sklearn**.**metrics**.**accuracy\_score(tar\_test, predictions))

*# to display the relative importance of each predictive variable*

model **=** ExtraTreesClassifier()

model**.**fit(pred\_train, tar\_train)

print('\nImportance of predictors:')

dct **=** dict()

**for** c **in** range(len(predictors**.**columns)):

dct[predictors**.**columns[c]] **=** model**.**feature\_importances\_[c]

print(sorted(dct**.**items(), key**=**operator**.**itemgetter(1), reverse**=True**))

*# run different numbers of trees to see the effect of the number on the accuracy of the prediction*

n **=** 100

accuracy **=** [0]**\***n

**for** i **in** range(n):

classifier **=** RandomForestClassifier(n\_estimators**=**i**+**1)

classifier **=** classifier**.**fit(pred\_train, tar\_train)

predictions **=** classifier**.**predict(pred\_test)

accuracy[i] **=** sklearn**.**metrics**.**accuracy\_score(tar\_test, predictions)

plt**.**plot(range(1, n**+**1), accuracy)

plt**.**xlabel("Number of trees")

plt**.**ylabel("Accuracy of prediction")

plt**.**title("Effect of the number of trees on the prediction accuracy")

plt**.**show()

print(accuracy)

print('----------------Random Forests------------------------')

call(random\_forests)

----------------Random Forests------------------------

Red Wine

Confusion matrix:

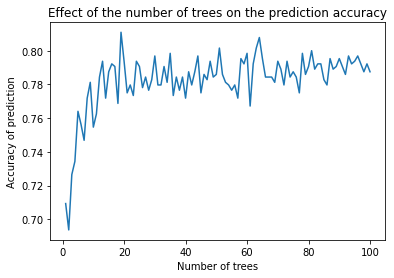
[[219 73]

[ 70 278]]

Accuracy: 0.7765625

Importance of predictors:

[('alcohol', 0.2044339716115786), ('volatile\_acidity', 0.1105992721049687), ('sulphates', 0.09584831080874494), ('total\_sulfur\_dioxide', 0.09205770122812139), ('density', 0.08198265295955814), ('citric\_acid', 0.07247366337091757), ('residual\_sugar', 0.06993977013491741), ('free\_sulfur\_dioxide', 0.06906397063084603), ('chlorides', 0.06842213282610284), ('fixed\_acidity', 0.06770819601045036), ('pH', 0.06747035831379405)]



[0.709375, 0.69375, 0.7265625, 0.734375, 0.7640625, 0.75625, 0.746875, 0.771875, 0.78125, 0.7546875, 0.7625, 0.784375, 0.79375, 0.771875, 0.7875, 0.7921875, 0.790625, 0.76875, 0.8109375, 0.79375, 0.775, 0.7796875, 0.7734375, 0.79375, 0.790625, 0.778125, 0.784375, 0.7765625, 0.7828125, 0.796875, 0.7796875, 0.7796875, 0.790625, 0.78125, 0.7984375, 0.7734375, 0.784375, 0.7765625, 0.784375, 0.771875, 0.7875, 0.7796875, 0.7875, 0.796875, 0.775, 0.7859375, 0.7828125, 0.79375, 0.784375, 0.7859375, 0.8015625, 0.7859375, 0.78125, 0.7796875, 0.7765625, 0.7796875, 0.771875, 0.7953125, 0.7921875, 0.7984375, 0.7671875, 0.7921875, 0.8015625, 0.8078125, 0.7953125, 0.784375, 0.784375, 0.784375, 0.78125, 0.79375, 0.7890625, 0.7796875, 0.79375, 0.784375, 0.7875, 0.784375, 0.775, 0.7984375, 0.7859375, 0.790625, 0.8, 0.7890625, 0.7921875, 0.7921875, 0.7828125, 0.7796875, 0.7953125, 0.7890625, 0.790625, 0.7953125, 0.790625, 0.7859375, 0.796875, 0.7921875, 0.79375, 0.796875, 0.7921875, 0.7875, 0.7921875, 0.7875]

White Wine

Confusion matrix:

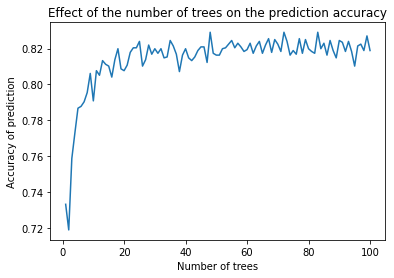
[[ 451 229]

[ 125 1155]]

Accuracy: 0.8193877551020409

Importance of predictors:

[('alcohol', 0.15547742055135122), ('volatile\_acidity', 0.11100784990826319), ('density', 0.09651376476853467), ('free\_sulfur\_dioxide', 0.09034005637061779), ('citric\_acid', 0.08500236508202688), ('total\_sulfur\_dioxide', 0.08431098730141577), ('chlorides', 0.08319989664164648), ('pH', 0.07651287714686318), ('residual\_sugar', 0.07624629976347226), ('sulphates', 0.07147422215602513), ('fixed\_acidity', 0.06991426030978351)]



[0.7331632653061224, 0.7188775510204082, 0.7586734693877552, 0.7729591836734694, 0.786734693877551, 0.7877551020408163, 0.7903061224489796, 0.7954081632653062, 0.8061224489795918, 0.7908163265306123, 0.8076530612244898, 0.8051020408163265, 0.813265306122449, 0.8112244897959183, 0.810204081632653, 0.8040816326530612, 0.8137755102040817, 0.8198979591836735, 0.8086734693877551, 0.8076530612244898, 0.8107142857142857, 0.8178571428571428, 0.8204081632653061, 0.8204081632653061, 0.8239795918367347, 0.810204081632653, 0.8137755102040817, 0.8219387755102041, 0.8168367346938775, 0.8198979591836735, 0.8173469387755102, 0.8198979591836735, 0.814795918367347, 0.8153061224489796, 0.8244897959183674, 0.8214285714285714, 0.8168367346938775, 0.8071428571428572, 0.8163265306122449, 0.8198979591836735, 0.814795918367347, 0.813265306122449, 0.8153061224489796, 0.8188775510204082, 0.8209183673469388, 0.8209183673469388, 0.8122448979591836, 0.8290816326530612, 0.8173469387755102, 0.8163265306122449, 0.8163265306122449, 0.8198979591836735, 0.8204081632653061, 0.8224489795918367, 0.8244897959183674, 0.8204081632653061, 0.8229591836734694, 0.8209183673469388, 0.8183673469387756, 0.8193877551020409, 0.8229591836734694, 0.8173469387755102, 0.8214285714285714, 0.8239795918367347, 0.8173469387755102, 0.8219387755102041, 0.8255102040816327, 0.8178571428571428, 0.825, 0.8224489795918367, 0.8183673469387756, 0.8290816326530612, 0.8239795918367347, 0.8163265306122449, 0.8188775510204082, 0.8168367346938775, 0.8255102040816327, 0.8173469387755102, 0.825, 0.8198979591836735, 0.8183673469387756, 0.8173469387755102, 0.8290816326530612, 0.8198979591836735, 0.8229591836734694, 0.8163265306122449, 0.8244897959183674, 0.8188775510204082, 0.814795918367347, 0.8244897959183674, 0.823469387755102, 0.8183673469387756, 0.8239795918367347, 0.8183673469387756, 0.810204081632653, 0.8214285714285714, 0.8224489795918367, 0.8188775510204082, 0.8270408163265306, 0.8188775510204082]

In [ ]: