**Final Project**

by Gökberk Pekerkan, 56992

Course: QMBU/INTL 450

Spring 2020

Koç University

INDEX

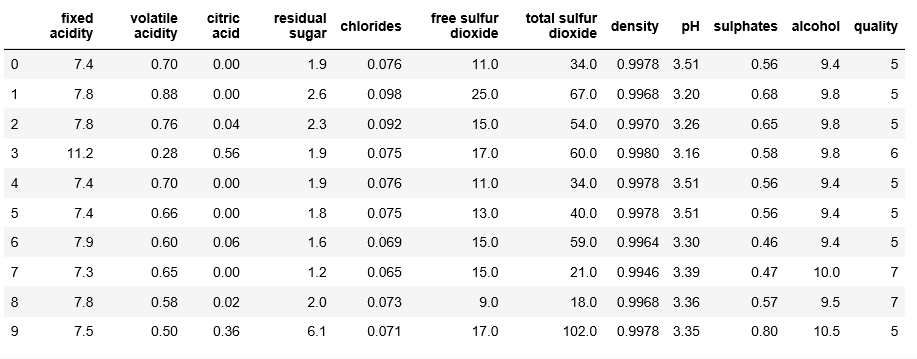
1. Introduction
2. Hypothesis
3. Futher Notes
4. Reading the Data
5. Algorithms
6. Conclusion
7. **Introduction**: Deciding which wine is good has many things to consider about. Actually it can be pretty hard to decide which wine is actually a good if you are not a conasseur or don’t know anything which makes a good quality wine. This kept me thinking, why not use machine learning algorithms which could use all the metrics and predict the quality for you. Hence, this project provides just that.

Hereby, the relationship of these components will be measured and a relationship to the quality of wine calculated. Machine Learning tools thaught in this course will be implemented and the accuracy scores resulting from these tools will be presented to show which proves to be the best.

1. **Hypothesis:** The different data entries will be able to predict the wine quality.
2. **Further Notes:** This dataset was downloaded from kaggle.com and can be found there.
3. **Reading the data**

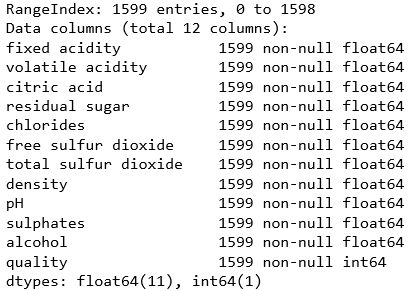
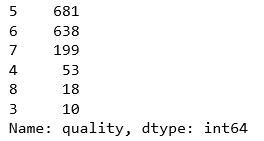
*wine = pd.read\_csv("winequality.csv")*

After reading the data, I looked how the data was distributed on the following commands:

*wine.head(10)*  


We have 11 different features

wine.info() wine['quality'].value\_counts()



Here is a simple view of all the ‘quality’ values for every data entry. Quality grades range from 3 to 8.

The data has a lot of entries as can be seen in the figure.

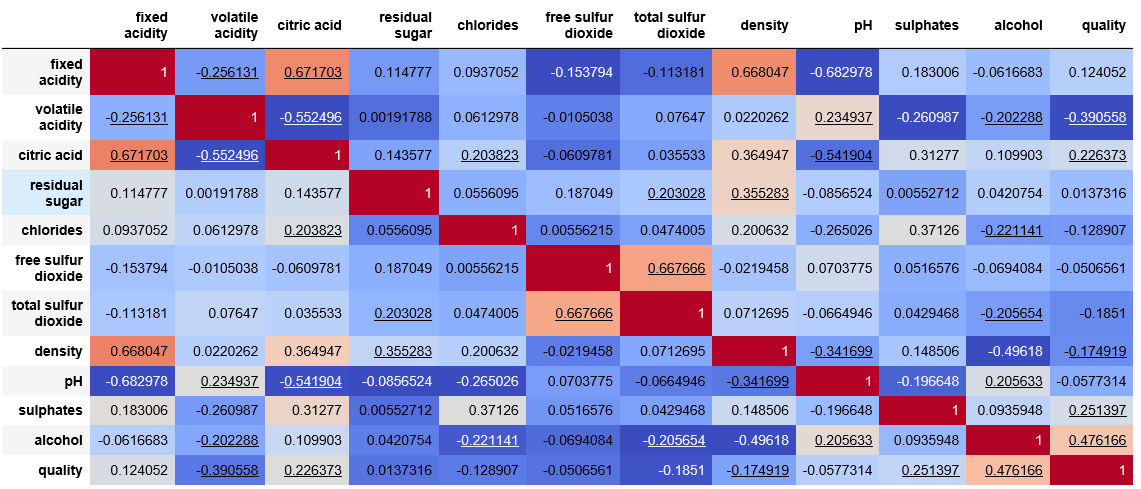
The correlation matrix gives a good view on how the metrics are correlated to eachother.

*rs = np.random.RandomState(0) #shows a good corr. matrix*

*df = pd.DataFrame(rs.rand(10, 10))*

*corr = wine.corr()*

*corr.style.background\_gradient(cmap='coolwarm')*



Alcohol and Quality show a positive correlation.

Volatile Acidity and Quality show a negativ correlation.

Now it is time to classify ‘quality’ values into binaries:

*reviews = []*

*for i in wine['quality']:*

*if i <= 6:*

*reviews.append('0')*

*elif i >= 7 :*

*reviews.append('1')*

*wine['quality'] = reviews*

Reviews which are equal or smaller than 6 are considered bad, whereas reviews equal or larger than 7 are considered as good quality.

*wine['quality'].value\_counts()*

This shows us how ‘good’ and ‘bad’

classified quality wines are distributed.

Outcome 1 denotes good quality whereas outcome 0 stands for bad.

Separating the dataset: splitting X and y variables

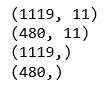
*X = wine.drop('quality', axis=1)*

*y = wine['quality']*

Wine dataset is split up into training and testing data. Here, 70% is used for training and 30% for testing.

*Xtrain, Xtest, ytrain, ytest = train\_test\_split(X, y, test\_size=0.3,random\_state=1)*

To be sure it would be good to check if everything is in order.

*Xtrain.shape*

*Xtest.shape*

Seems all good.

*ytrain.shape*

*ytest.shape*

1. **Algorithms**

In the upcoming parts, I will use the following algorithms:

1. Gaussian Naive Bayes
2. Decision Trees
3. Support Vector Machines (SVM)
4. Random Forests
5. Kneighbors

Gaussian Naive Bayes

GaussianNB was implemented to predict the probability of different classes based on various attributes of the wine data set.

*dtc = DecisionTreeClassifier()*

*dtc = dtc.fit(Xtrain,ytrain)*

*ypred = dtc.predict(Xtest)*

*"Accuracy:",metrics.accuracy\_score(ytest, ypred)\*100*



This score actually provides a good accuracy score.

Decision Trees

Since we have a lot of data entries it would be wise to try it out with the Decision Tree model. Also it works well with classified problems which is the case in this project.

*dtc = DecisionTreeClassifier()*

*dtc = dtc.fit(Xtrain,ytrain)*

*ypred = dtc.predict(Xtest)*

*"Accuracy:",metrics.accuracy\_score(ytest, ypred)\*100*



We got a higher score than the previous model. This model doesn’t seem to disappoint.

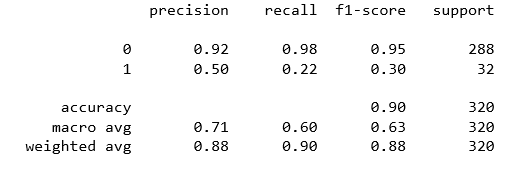
Support Vector Machines – SVM

*svc = SVC()*

*svc.fit(Xtrain,ytrain)*

*pred\_svc =svc.predict(Xtest)*

*classification\_report(ytest,pred\_svc)*



Random Forest

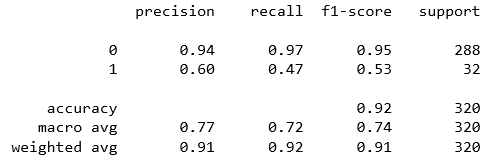
Here I decided to use 250 trees in the forest to give a better performance.

*rfc = RandomForestClassifier(n\_estimators=250)*

*rfc.fit(Xtrain, y\_train)*

*pred\_rfc = rfc.predict(Xtest)*

*classification\_report(ytest, pred\_rfc)*



This seems to be the highest accuracy score by far.

Kneighbors Classifier

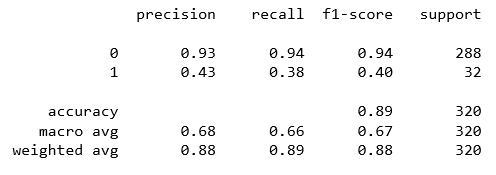
My last algorithm will be KNN which is also good for large data sets

*knn = KNeighborsClassifier()*

*knn.fit(Xtrain,ytrain)*

*pred\_knn=knn.predict(Xtest)*

*classification\_report(y\_test, pred\_knn)*



1. **Conclusion**

Eventually, we can predict the wine quality with a 92% score if the Random Forest Classifier is selected. Also my hypothesis that from the metrics, the quality of wine can be measured seems to be legit since we have a high accuracy score as a result. Even though SVM an KNN were pretty close to eachother, GaussianNB showed the smallest accuracy score.