Machine Learning Nanodegree

Finding whether the quality of wine is good or bad using Supervised Learning

Patnala Sai Kiran

July 2nd, 2018

I. Definition

Project Overview

Wine is an alcoholic beverage produced through the fermentation of grapes. Other fruits and plants, such as berries, apples, cherries, palm, and rice can also be fermented. It is one of the most consumed drink by the people. It plays an important role in human's life. Yeast consumes the sugar in the grapes and converts it to ethanol and carbon dioxide. Different varieties of grapes and strains of yeasts produce different styles of wine. These variations result from the complex interactions between the biochemical development of the grape, the reactions involved in fermentation, the terroir, and the production process. Many countries having legal names intended to define styles and qualities of wine. These typically restrict the geographical origin and permitted varieties of grapes, as well as other aspects of wine production.

Wine has been produced for thousands of years. The earliest known traces of wine are from China, Georgia, Iran and Sicily. The Wine we are going to study here is RedWine. The red-wine production process involves extraction of colour and flavour components from the grape skin. Red wine is made from dark-coloured grape varieties. The red colour comes from some pigments (called anthocyanins) present in the skin of the grape. In addition we can have some chemicals as ingredients. So, in my dataset iam using some features like volatile acidity, pH, chlorides, free sulphates, free sulfur dioxide,total sulfur dioxide, density and alcohol etc to predict the quality of the wine to better

The interest has been increased for good quality wine in people in recent years which demands growth in this industry. Therefore, companies are investing in new technologies to improve wine production and selling. In this direction, our wine quality certification plays a very important role for the people to choose the best quality wine. According to the technical point of view, a good quality wine consists of factors meeting specific criteria and set according to considerations of scientific, chemical and technical factors, which can always be verified analytically. The quality of wine is mainly affected by the ingredients used for making it. So, we use these ingredients to predict the quality of the wine.

Reference1: http://www.diwinetaste.com/dwt/en2013073.php

Reference2:
https://www.winesandvines.com/news/article/190064/National-Impact-of-the-Wine-Industry2199-Billion

Reference3: https://en.wikipedia.org/wiki/Wine

Problem Statement

Determination of the quality of wine is very important, since the people who are willing to consume the wine for potential benefits may anticipate it to be a good one and the current project deals with achieving the factors mentioned in the above section.

Since the task is to figure out whether the quality of the wine is good or bad based on the chemical composition in the manufacturing the wine.And hence we have to tackle with a binary classification problem that has two possible outcomes ('0' for bad quality and '1' for good quality).

Features and Description

In the present data set I segregated the given data set into 11 potential input features which are physicochemical (inputs)

- Fixed acidity -It is the level of fixed acidity
- Volatile acidity -it is the level of volatile acidity
- Citric acid -amount of citric acid
- Residual sugar -amount of residual sugar
- Chlorides -amount of chlorides
- Free sulfur dioxide -amount of free sulphur dioxide
- Total sulfur dioxide -amount of total sulphur dioxide
- Density -density of the wine
- pH -pHhe wine
- sulphates- amount of sulphates in the wine
- Alcohol - alcohol content in the wine

and the output variable is(based on sensory data)

- Quality (score between 0 and 10)

Reference : https://archive.ics.uci.edu/ml/datasets/Wine+Quality

By considering all the above features, i will predict the quality of the wine to be good or bad. So, for this i will convert my target variable into the form that is suitable to perform classification i.e; as (0 and 1). Then I will apply diverse machine learning classification models to predict the quality of the wine and compare the performances of the diverse models and eventually determine the final model for the input data. I think all the provided features have priority in estimating the quality of the wine.

Metrics

In this project I will use the evaluation metric of **fbeta_score**. It measures the effectiveness of retrieval with respect to a user who attaches beta times as much importance to recall as precision.

$$F\beta = (1+\beta 2).precision.recall/(\beta 2.precision)+recall$$

The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

When **beta=0.5** more emphasis is placed on precision. This is called f 0.5 score. For my project, I will calculate fbeta_score following sklearn documentation as follows

sklearn.metrics.fbeta_score(*y_true, y_pred, beta, labels=None, pos_label=1, average='bi nary', sample_weight=None*)

In my project I will be calculating the fbeta_score with beta=0.5.

Reference: https://docs.orange.biolab.si/3/data-mining-library/reference/evaluation.cd.html

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.fbeta_score.html

Therefore, I have selected F – score as my metric because there is no imbalance in the data and I want to select the model based on the balance between Precision and Recall. I will also check training and testing time.

II . Analysis

Data Exploration

In the data Exploration phase, I found the total number of records and attributes in the considered dataset. Then printed the instances and attributes present in the dataset.

```
print("The number of instances is :",len(data))
print("The number of attributes is :",len(data.columns))
print(data.shape)
```

Output:

The number of instances is : 1599
The number of attributes is : 12
(1599, 12)

Then i displayed the dataset to have a clear picture of it.

data.describe() - This gave me the following output

```
data.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8

From above result, we can clearly say that all **features are of numerical** type and continuous values except Quality which is discrete in nature.And also there are **no missing values** in the data which can be clearly seen from the above output.

Then i explored the features that are present in my dataset by running the code-- data.head()

```
data.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

So, we can conclude the features as follows

Predictors:

1 - fixed acidity

2 - volatile acidity

3 - citric acid

4 - residual sugar

5 - chlorides

6 - free sulfur dioxide

7 - total sulfur dioxide

8 - density

9 - pH

10 - sulphates

11 - alcohol

Output variable:(Target variable)

12 - quality (score between 0 and 10)

Exploratory Visualization

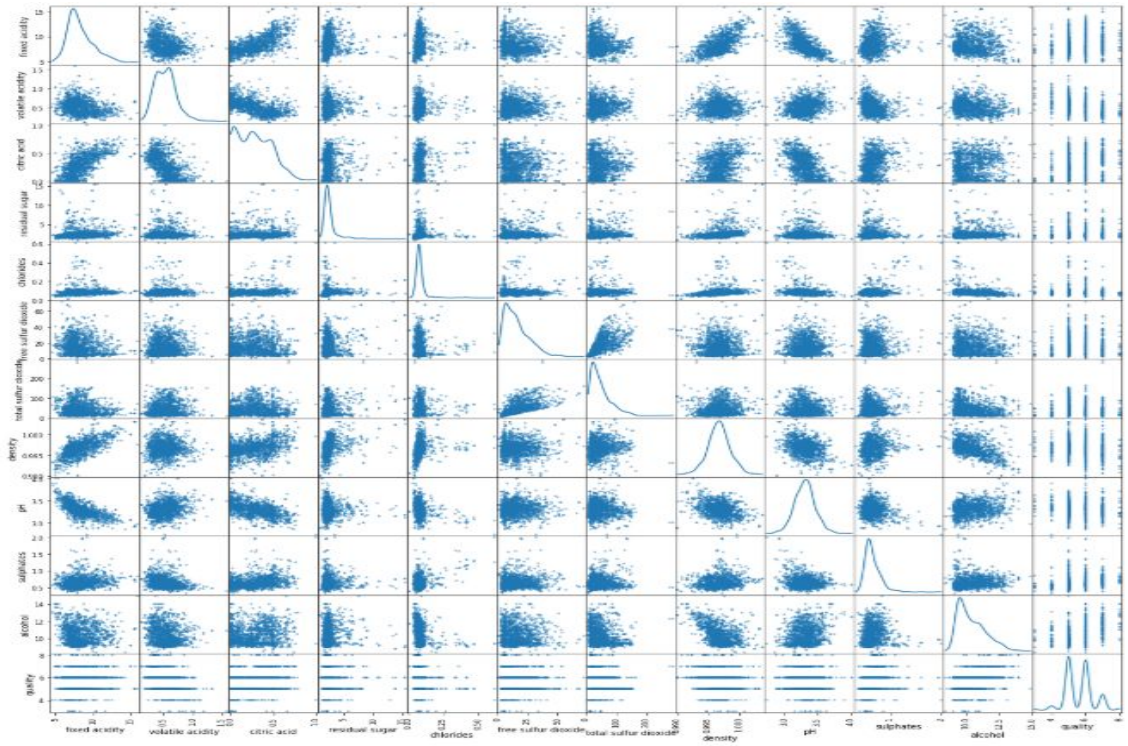
In this phase i plotted the scatter matrix of all the features of the dataset, because to see the distribution of each feature in the dataset.

```
pd.scatter_matrix(data , alpha = 0.5 , figsize = ( 20 , 20 ) , diagonal = 'kde' );
plt.show()
```

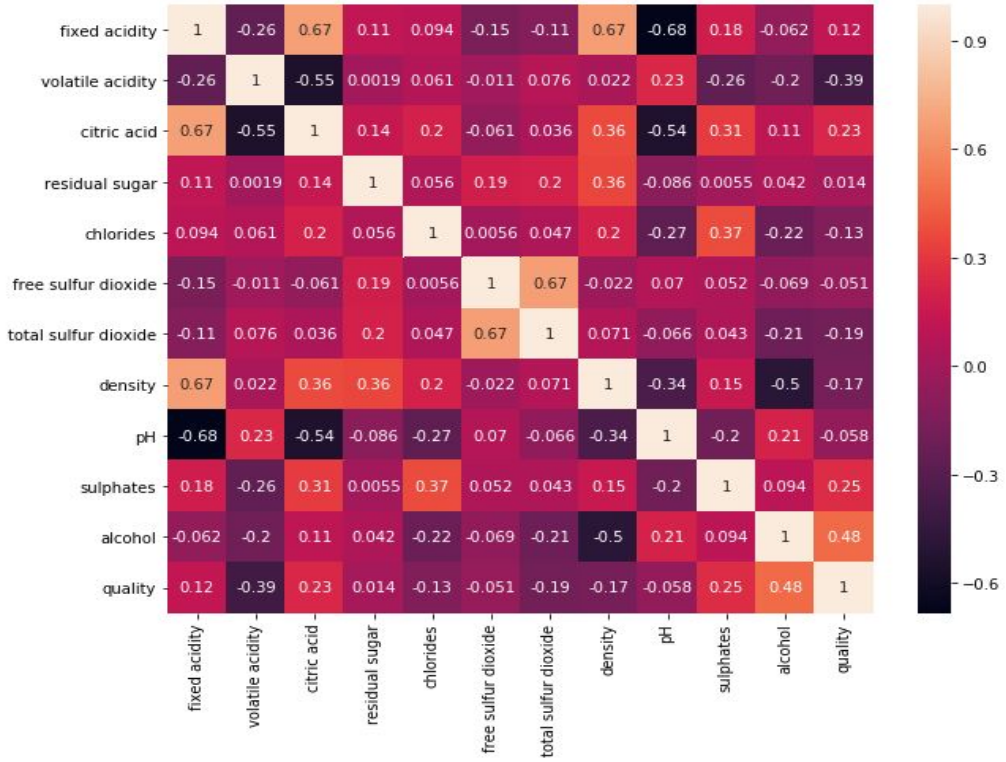
This gives the following scatter matrix:

By seeing the below scatter matrix image we can clearly say that most of the distributions are skewed.And very few features are normally distributed.

Then i plotted the Heatmap of all the features.I made this plot in order to the correlation of each and every feature with the output variable(‘quality’).



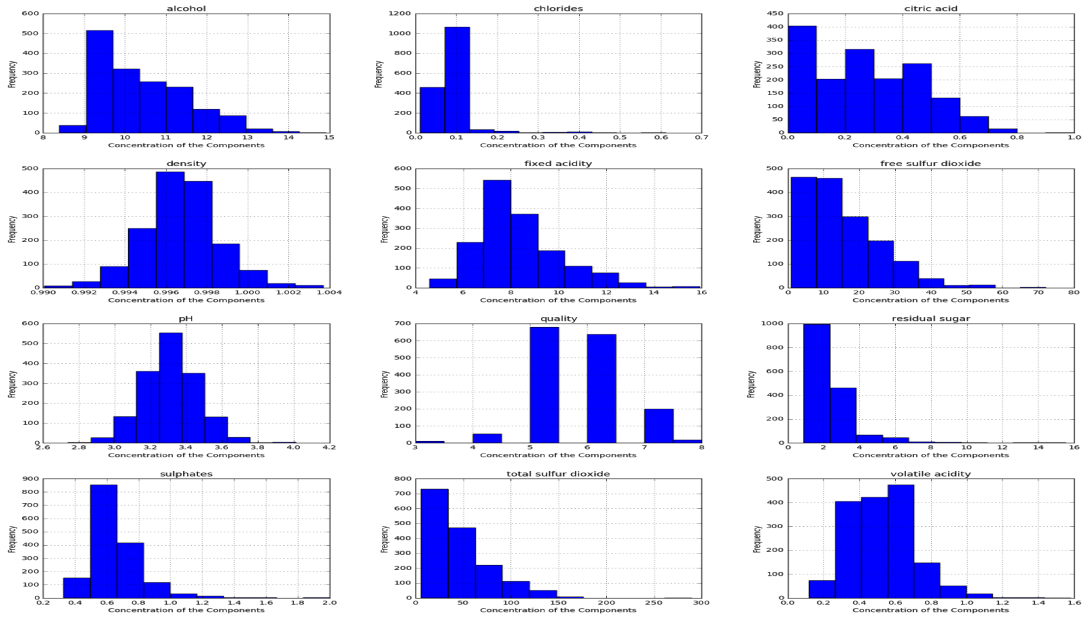
The Heatmap is as follows:



We can look few features with good correlation with the **target variable** as follows:

- **alcohol :0.48**
- **citric acid : 0.23**
- **sulphates : 0.25**

I plotted the heatmap because, i can find the features which are highly correlated with the output variable and found very few features. And we can also see that most of the features are skewly distributed in the following image.



Algorithm and Techniques

In Machine Learning we face the major challenge with choosing the best algorithm that suits our data, which predicts better. We have many algorithms like Logistic Regression, Decision Tree Classifier,KNN Classifier, Adaboost Classifier and Random Forest etc.Among these i am going to choose few algorithms that i believe that suits my dataset.

1. SVM -- I selected SVM,because it is good for binary classification but when we have multiple features then Kernel SVM can be used. Perhaps the biggest limitation of the support vector approach lies in choice of the kernel. A second limitation is speed, both in training and testing.

   Reference: http://www.svms.org/disadvantages.html

2. AdaBoost -- AdaBoost (Adaptive Boosting) is a powerful classifier that works well on both basic and more complex recognition problems. AdaBoost works by creating an accurate classifier by combining many relatively weak classifiers. It is a powerful classification algorithm that is applied in a wide variety of fields, such as biology, computer vision, and speech processing. Unlike other powerful classifiers, such as SVM, AdaBoost can achieve similar classification results with much less use parameters . But, AdaBoost can be sensitive to noisy data and outliers. In some problems, however, it can be less prone to the overfitting than most learning algorithms.Any way we are removing the few outliers from our dataset.

   Reference : http://www.nickgillian.com/wiki/pmwiki.php/GRT/AdaBoost

3. Decision Tree Classifier -- The reason behind choosing this is, it is very powerful classification algorithm that converts our classification task into tree like structure and it is easy to interpret and understand.This is relatively easy and fast to predict the class of Target variable.The number of hypertune parameters are almost null. But, this algorithm also leads to overfitting which can be solved by Ensemble learning algorithm.

4. Random Forest: --This algorithm creates the forest with a number of trees. This model gives better accuracy than other models because it consists of more number of trees. The **higher the number** of trees in the forest, it gives **the high accuracy** results.How ever using random forest, there is no need for feature normalization, individual decision trees can be trained in parallel,random forests are widely used, they reduce overfitting.

   Reference : http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/

Technique

Grid Search Technique

In my project I will perform hyper parameter tuning using grid search cv to further improve my optimal model's accuracy.Hyper parameter tuning is very important in improving performance of the model. To find the best Hyper parameters, we give a set of values to the parameters which are to be tuned and give this space to Grid Search algorithm . It will allocate each value in the space at a time and returns the best parameters. I will use the grid search cv documentation as follows:

sklearn.model_selection.GridSearchCV(estimator, param_grid, scoring=None, fit_params=None,n_jobs=1, iid=True, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score='raise',return_train_score='warn')

GridSearch:
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV

BenchMark Model

Logistic Regression:

As a benchmark model I have used Logistic Regression classification model because it doesn't require linear relationships to exist in between predictors and target variables. The problem with this algorithm is it is more prone to overfitting.

Logistic regression uses an equation as the representation, very much like linear regression.

Below is an example logistic regression equation:

$$y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x).

Reference : https://machinelearningmastery.com/logistic-regression-for-machine-learning/

F Beta score is calculated so that any improvement will add to the performance of model. F Beta will be used for measuring performance. The fbeta score for our Logistic Regression model is calculated.

Bench Mark Model's Fbeta_score: 0.761316872428

III . Methodology

Data Preprocessing

In this phase, I am going to seperate the target variable from the original dataset using the following code

```
y= pd.Series(data['quality'])
X=data.drop('quality', axis=1)
display(X.head())
```

It provides the following output:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

After this I will check for any missing values in my data and by observing the results as shown in the image given below we can clearly say that there are no missing values in my dataset.

```
data.isnull().sum()
```

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

Here we have no NULL values in our data set .So, we need not remove any of the values.

Now the data is checked for the outliers and if found any, they are removed from the data in order to improve accuracy of the estimators. But sometimes the presence of an outlier has a certain meaning, which explanation can be lost if the outlier is deleted.

The following code will give 120 outliers that are present in our dataset.By removing these outliers we will get 1479 instances remaining in our dataset.

```
l=0
cnt=0
tot_outliers = np.array([])
l2=np.array([])
for val in X.keys():
    Q1= np.percentile(X[val], q=25)
    Q3= np.percentile(X[val],q=75)
    IQR = Q3 - Q1
    lower_bound = Q1 - (IQR * 1.5)
    upper_bound = Q3 + (IQR * 1.5)
    print("outliers for:{}".format(val))
    outliers = X[((X[val] < lower_bound) | (X[val] > upper_bound))]
    tot_outliers = np.append(outliers.index.get_values(),tot_outliers)
com_outliers = []
#print(Counter(tot_outliers))
x=dict(Counter(tot_outliers))
for i in x:
    if x[i]>1:
        com_outliers.append(int(i))
print(com_outliers)
```

Now we will convert the target variable suitable for the classification. So that , we can classify our data points into the two specified classes. We will convert the target variables to 0 and 1. If the value of the target variable is greater than 5 then it is changed to 1 else it is changed to 0. We will convert as follow

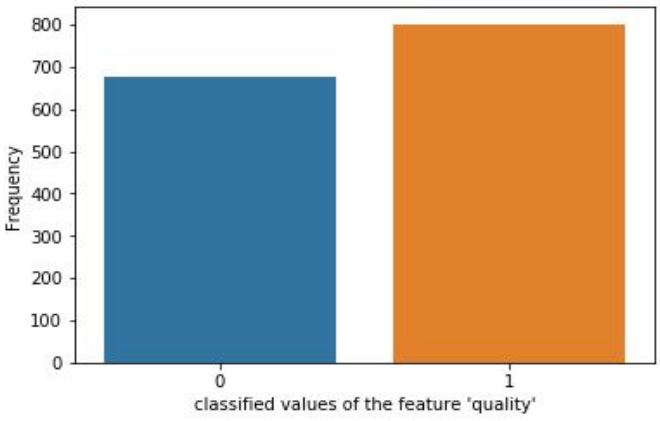
y = pd.Series([1 if x > 5 else 0 for x in y])

Then we will plot the distribution of the 0 and 1 in the target variable and it gives the count of 1 as 801 and count of 0 as 678.

Feature-Scaling(Min Max Scaling)

Feature scaling is a method of standardizing the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

In our project we are performing the Min Max scaling which standardizes the values in the range of 0,1



Reference : https://en.wikipedia.org/wiki/Feature_scaling

Here in our dataset, after the feature scaling the values are scaled to values between [0,1] and [-1,1]. We will calculate the scaled value by using the following formula.

$$x' = x - \min(x)'/\max(x)-\min(x)$$

Where x' is standardized value and

x is the original value

The code for min max scaling is as follows:

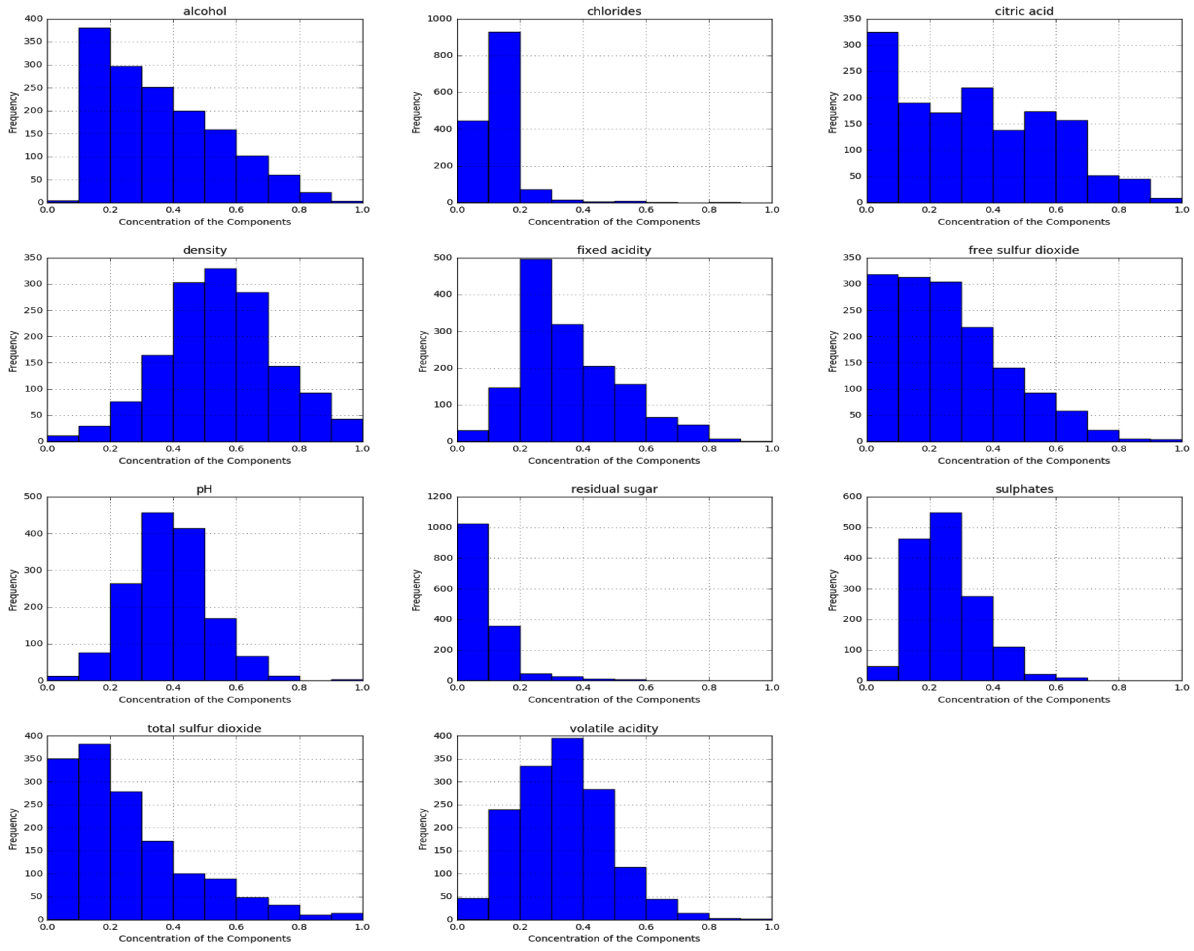
```
: # Initialize a scaler, then apply it to the features
  scaler = MinMaxScaler() # default=(0, 1)

  good_data[:]= scaler.fit_transform(good_data[:])

  # Show an example of a record with scaling applied
  display(good_data.head(5))
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	0.269231	0.479339	0.000000	0.055118	0.108883	0.188679	0.187919	0.663158	0.557522	0.186992	0.178571
1	0.307692	0.628099	0.000000	0.110236	0.171920	0.452830	0.409396	0.557895	0.283186	0.284553	0.250000
2	0.307692	0.528926	0.051282	0.086614	0.154728	0.264151	0.322148	0.578947	0.336283	0.260163	0.250000
3	0.634615	0.132231	0.717949	0.055118	0.106017	0.301887	0.362416	0.684211	0.247788	0.203252	0.250000
4	0.269231	0.479339	0.000000	0.055118	0.108883	0.188679	0.187919	0.663158	0.557522	0.186992	0.178571

After scaling the distributions are as below:



Train-Test Split

Now we will split the data into training and testing sets with 25% data as testing data and 75% of data as training data.

```
: X_train , X_test , y_train , y_test = train_test_split(good_data, y , test_size = 0.25 , random_state = 145)

print(X_train.shape , y_train.shape)
print(X_test.shape , y_test.shape)
```

```
(1109, 11) (1109,)
(370, 11) (370,)
```

Implementation

I tried different algorithms as follows found their fbeta_scores to find the best algorithm

Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier
dec = DecisionTreeClassifier(random_state=1)
dec.fit(X_train,y_train)
dec_pred = dec.predict(X_test)

accuracy = fbeta_score(y_test, dec_pred,beta=0.5)
print(accuracy)
```

Fbeta_score : 0.7647

SVM :

```
from sklearn.svm import SVC
svm_model_linear = SVC().fit(X_train, y_train)
svm_predictions = svm_model_linear.predict(X_test)

# model accuracy for X_test
accuracy = fbeta_score(y_test, svm_predictions,beta=0.5)
print(accuracy)
```

Fbeta_score : 0.7489

Adaboost Classifier:

```
from sklearn.ensemble import AdaBoostClassifier
dec = AdaBoostClassifier (random_state=1)
dec.fit(X_train,y_train)
dec_pred = dec.predict(X_test)

accuracy = fbeta_score(y_test, dec_pred,beta=0.5)
print(accuracy)
```

Fbeta_score : 0.7716

Random Forest:

```
from sklearn.ensemble import RandomForestClassifier
dec = RandomForestClassifier (random_state=1)
dec.fit(X_train,y_train)
dec_pred = dec.predict(X_test)

accuracy = fbeta_score(y_test, dec_pred,beta=0.5)
print(accuracy)
```

Fbeta_score : 0.7839

Now we have obtained the fbeta_scores of all the above stated algorithms and now we have to find the best model among them .We can directly eliminate the SVM model as it has the least fbeta score when compared to other models we tested so far.

The Decision Tree classifier and AdaBoost classifier has less difference between their fbeta scores but if we observe the training time AdaBoost algorithm takes more time to train the model. And Decision Tree classifier has less hyper parameters (almost null). Sometimes it leads to overfitting.

I choose **Random Forest as my final model** because its fbeta_score is larger than all the other algorithms and this algorithm creates the forest with a number of trees. This model gives better accuracy than other models because it consists of more number of trees.The higher the number of trees the more accuracy we get.Random Forest model also takes much time for hyper parameter tuning but ultimately it gives higher accuracy.

Refinement

I am using Grid Search technique to refine my model performance. The initial and final model solutions are significantly different. Previously , without tuning I got an Fbeta_score of 0.7839 and after tuning, I got 0.815.

The grid Search is implemented as follows:

```
from sklearn.metrics import make_scorer

param_grid = {'n_estimators':[10,50,100,150],'max_depth': [1,2,3,4,5,8]}
#cvs=ShuffleSplit(X_train.shape[0],n_iter=10,test_size=0.2,random_state=1)
clf=RandomForestClassifier()

scoring=make_scorer(fbeta_score,beta=0.5)
grid_search = GridSearchCV(clf, param_grid,cv=5,scoring=scoring)
kfit=grid_search.fit(X_train, y_train)
model = kfit.best_estimator_
final = model.predict(X_test)
```

```
final_score=fbeta_score(y_test,final,beta=0.5)
print(final_score)
```

And this gave me a **Fbeta _score :0.815**

IV .Results

Model Evaluation and Validation :

The final model (Random Forest Classifier with hyper parameter tuning using Grid Search) achieved an Fbeta_ score of around 81% which is a good score.

The final model parameters are :

-- **n_estimators** : 100

The default value for n_estimators is 50 but our best_model is considering 100 i.e., it is taking more

no. of estimators to build strong learner.

Max_depth : 8

Therefore the un-optimized and optimized models are:

Un - optimized model : RandomForestClassifier before tuning.

Optimized model : RandomForestClassifier after tuning.

Justification

Consider the following table

	Benchmark Model	Unoptimized Model	Optimized Model
Fbeta_score	76.13	78.39	81.5

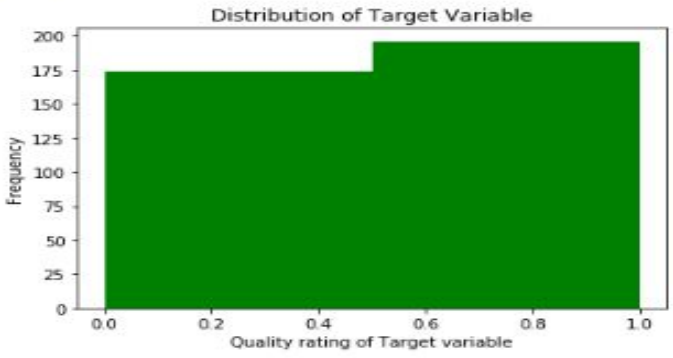
From the we can infer that the final model is better than the BenchMark model and the Unoptimized Model. Our BenchMark model is not accurate as it is giving the accuracy of only 76% and which is less compared to the 81.5 obtained by our optimized model which is using Random Forest Classifier.

V. Conclusion

Free-Form Visualization

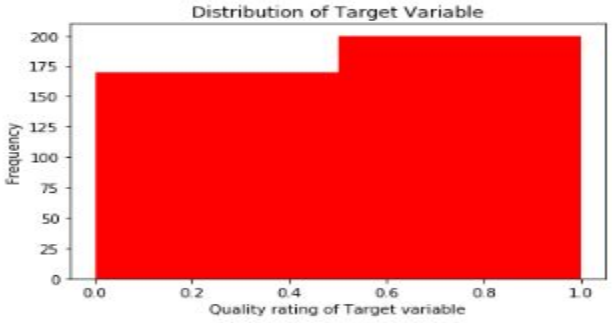
```
plt.hist(y_test , bins = 2 , color = 'g')

plt.xlabel('Quality rating of Target variable')
plt.ylabel('Frequency')
plt.title('Distribution of Target Variable')
plt.show()
```



```
plt.hist(final , bins = 2 , color = 'r')

plt.xlabel('Quality rating of Target variable')
plt.ylabel('Frequency')
plt.title('Distribution of Target Variable')
plt.show()
```



When we compare these two graphs, If you see y-axis at 175 the values in the first graph are included but not in second graph. This shows that the in the testing data some of the values are wrongly predicted as 1.But our final; model is predicting this correctly. This is because of some imbalance in the input data. This can be avoided if we have correctly balanced data.

Reflection

- In the First step, I obtained a dataset from the repository and then started the analysis of the data.
- In the analysis phase, I read the csv input file into dataFrame and after that i knew the number of rows and columns in my dataset.
- After that i observed the features that are present in the dataset and also described the features of it.Then I explained what are the predicting variables and what is the target variable .
- After that I started with the visualization phase ,In this i observed the distribution of the features using the scatter matrix and after that plotted the heat map to find the correlation of the target variable with the predicting variables.And found few features that are positively correlated with the output variable. Then tried to find what are the features that are skewly distributed.
- And then started with the data preprocessing phase. In phase i separated the target variable from the dataset, checked for null values and after that checked for outliers and found about 573 outliers . Among them I removed the most common outliers for more than one feature. This way i removed 120 instances from the data set.
- After that i changed my target variable into the form suitable for classification. Then i scaled all my features using minmax scaler. As a result all the feature values are scaled to values in the range [0,1].
- In the next step I split my dataset into training and testing sets .with 25% for testing and 75% for training.

- Then I checked my Benchmark model for accuracy, which is less.
- Then I tried different algorithms like Adaboost, Decision Tree, SVM and RandomForest and obtained their respective fbeta scores. Then I selected one model(Random Forest classifier) with best score.
- To perform the hyper parameter tuning on the final model, i used the Grid search technique which improved my fbeta score.

Improvement

- I think perfect results can be derived by completely removing the outliers but it is not ideal all the time.Because some outliers may have strong correlation the output variable.
- Perfectly tuning our hyper parameters would help to improve our accuracy much more.
- As we removed outliers algorithms like KNN can also be applied to get better result.

References:

1. http://www.diwinetaste.com/dwt/en2013073.php
2. https://www.winesandvines.com/news/article/190064/National-Impact-of-the-Wine-Industry2199-Billion
3. https://en.wikipedia.org/wiki/Wine
4. https://docs.orange.biolab.si/3/data-mining-library/reference/evaluation.cd.html
5. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.fbeta_score.html
6. http://www.svms.org/disadvantages.html
7. http://www.nickgillian.com/wiki/pmwiki.php/GRT/AdaBoost.
8. http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/
9. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV
10. https://machinelearningmastery.com/logistic-regression-for-machine-learning/
11. https://en.wikipedia.org/wiki/Feature_scaling