# Machine Learning Nanodegree

## Capstone project report

## Predicting Wine Quality

Chikka Udaya Sai

June 9th , 2018

## I . Definition

### Project Overview

Wine has become an important part of human life and often considered as a luxury product. The role of wine is significant in human's life because it is an important source of nutrition and the art of wine making has its own importance. Wine sector contributes a lot of economy to most of the countries like UK , Portugal etc., There are a lot of evidences (Evidence is given in the references) which suggests that moderate consumption of wine helps people to live longer , protecting them against certain cancers and improve mental health.

From the past , wine was used to treat various health conditions . Monks lived longer than other people because of their regular and moderate consumption of wine. This fact is also proven scientifically . For all this to happen , the quality of wine has to be great and wine makers give utmost importance to quality of wine. Producing a good quality of wine is not an easy task . There are so many factors influence the quality of wine but most of the factors related to the chemical composition involved in the manufacturing of wine. There is a research going on to produce the best quality of wine. Some of them I found interesting are:

https://www.researchgate.net/publication/259922445_Proficiency_test_on_FTIR_wine_analysis

https://www.emeraldinsight.com/doi/abs/10.1108/17511061111186514

http://www.guadoalmelo.it/en/winemaking-high-quality-artisanal-production/

https://health.gov/dietaryguidelines/2015/guidelines/appendix-9/

The reason behind my interest in this project is to know how Machine Learning algorithms perform on real – world applications. Our model is well suited for the field in testing the quality of wine.

### Problem Statement

The aim of this project is to predict the quality of wine based on the chemical composition involved in the manufacturing of wine . If the quality of wine is good , then my model will give output as **1** else **0.**

The problem is a binary classification problem which have two possible outputs ( '1' for Good quality of wine and '0' for bad quality of wine ).I am going to use 11 features (chemical composition involved in the manufacturing of wine ) in the data as predictor variables and Quality ( Quality rating ) as the target variable. For this purpose , I am modifying the data of Quality attribute to 1 and 0 depending on the quality rating of wine greater than 5 or not. In the project , I am going to use various Machine Learning Algorithms to predict the quality of wine and compare their performance and finally declare my final model.

**A brief outline of my solution :**

My solution to the problem will look like giving output 1 when the quality of the wine is good and giving output 0 when the quality of the wine is bad.

As the problem is a binary classification problem, I am going to use three classification algorithms namely Logistic Regressor, Decision Tree Classifier and Adaptive Boosting Classifier. I want to

select the best algorithm based on F - score ( My evaluation metric to measure performance ) and then using Grid Search technique, I want to improve the performance of my model to a little bit.

### Metrics

The evaluation metric that I am using to measure the performance of our model F – score . Accuracy and F – score are good metrics to measure the performance of classification model . F- score takes account of both precision and recall and accuracy tells how many of our predictions are right. I didn't choose accuracy score as my metric because my data is unbalanced.

Therefore , I am choosing F – score as my metric because I want to select the model based on the balance between Precision and Recall. In addition to that , I will also check training and testing time .

## II . Analysis

## Data Exploration

The dataset that I am working is downloaded from **UCI Machine Learning Repository**. It contains chemical composition of both red wine and white wine but I am working only with white wine data .

**Dataset URL :**

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

**Citation:**

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties.

In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Available at: https://www.sciencedirect.com/science/article/pii/S0167923609001377?via%3Dihub

The data is open – sourced and can be used for research purpose with proper citation included.

There are 12 features and 4898 instances of data . The feature that I am interesting is Quality ( Target Feature ) . All data in the dataset are of numerical type and continuos except Quality which is discrete ( It has values in between 1 to 10 ) . There are no missing values in the data. I am going to use all this data for my project.

The features are :

**Insights about the Data**

```
In [3]: # No. of Attributes and Instances  in the data

print(' Shape of the data :',wine_data.shape)

# Names of Features

print('\nThe features present in the data are :\n\n',wine_data.columns)

# Displaying First five Observations

display(wine_data.head())
```

```
 Shape of the data : (4898, 12)

The features present in the data are :

 Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

| | 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 |

1. Fixed Acidity :  Measure of total concentration of tartaric acid that are present in the wine which are produced in the fermentation process.
2. Volatile Acidity :  It is the acidity produced in the wine which is caused by the bacteria which gives a characteristic flavour & aroma to wine.
3. Citric Acid : Amount of Citric acid added to the wine to increase the acidity of wine and to give fresh flavour to it.
4. Residual Sugar : Amount of sugar that are left after the completion of fermentation step. The sweetness of wine depends on this amount.
5. Chlorides : Amount of chloride present in the wine. This is added to increase the saltiness.
6. Free Sulfur dioxide : Amount of free sulfur dioxide present in the wine . It is added to wine to preserve it from bacteria.
7. Total Sulfur dioxide : Amount of total sulfur dioxide ( free sulfur dioxide + other sulphate content ) present in the wine.
8. Density : It is the density of wine.
9. pH : It is a measure of acidity present in the wine . If acidity is more , pH is low.
10. Sulphates : The amount of sulphate content present in the wine. This is added to the wine to stabilize it.
11. Alcohol : It is the volume of alcohol present in the wine.
12. Quality : It is the measure of quality of wine from 1 to 10.

**Describing the Data**

```
In [4]: # Description of the data to get some insights

wine_data.describe()
```

Out[4]:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | qu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 4898.000000 | 4898.000000 | 4898.000000 | 4898.000000 | 4898.000000 | 4898.000000 | 4898.000000 | 4898.000000 | 4898.000000 | 4898.000000 | 4898.000000 | 4898.00 |
| mean | 6.854788 | 0.278241 | 0.334192 | 6.391415 | 0.045772 | 35.308085 | 138.360657 | 0.994027 | 3.188267 | 0.489847 | 10.514267 | 5.87 |
| std | 0.843868 | 0.100795 | 0.121020 | 5.072058 | 0.021848 | 17.007137 | 42.498065 | 0.002991 | 0.151001 | 0.114126 | 1.230621 | 0.88 |
| min | 3.800000 | 0.080000 | 0.000000 | 0.600000 | 0.009000 | 2.000000 | 9.000000 | 0.987110 | 2.720000 | 0.220000 | 8.000000 | 3.00 |
| 25% | 6.300000 | 0.210000 | 0.270000 | 1.700000 | 0.036000 | 23.000000 | 108.000000 | 0.991723 | 3.090000 | 0.410000 | 9.500000 | 5.00 |
| 50% | 6.800000 | 0.260000 | 0.320000 | 5.200000 | 0.043000 | 34.000000 | 134.000000 | 0.993740 | 3.180000 | 0.470000 | 10.400000 | 6.00 |
| 75% | 7.300000 | 0.320000 | 0.390000 | 9.900000 | 0.050000 | 46.000000 | 167.000000 | 0.996100 | 3.280000 | 0.550000 | 11.400000 | 6.00 |
| max | 14.200000 | 1.100000 | 1.660000 | 65.800000 | 0.346000 | 289.000000 | 440.000000 | 1.038980 | 3.820000 | 1.080000 | 14.200000 | 9.00 |

From the above code execution , we can clearly say that all **features are of numerical** type and continuous values except Quality which is discrete in nature.

There are **no missing values** in the data which can be clearly identified from the below execution.

```
In [7]: # Finding Null values in the data

        wine_data.isnull().any()
```

```
Out[7]: fixed acidity           False
        volatile acidity        False
        citric acid             False
        residual sugar          False
        chlorides               False
        free sulfur dioxide     False
        total sulfur dioxide    False
        density                 False
        pH                      False
        sulphates               False
        alcohol                 False
        quality                 False
        dtype: bool
```

**There are no missing values present in the data and therefore we can proceed further.**

Next task is to find outliers . Below execution finds the outliers present in the data.

**Outlier Treatment**

```
In [8]: # Identifying outliers in each attribute

        for feature in wine_data.keys():

            Q1 = np.percentile(wine_data[feature], q=25)

            Q3 = np.percentile(wine_data[feature], q=75)

            IQR = Q3 - Q1
            step = 1.5 * IQR

            print("Outliers for the feature '{}':".format(feature))
            display( wine_data[~((wine_data[feature] >= Q1 - step) & (wine_data[feature] <= Q3 + step))] )
```

```
Outliers for the feature 'fixed acidity':
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 98 | 9.8 | 0.360 | 0.46 | 10.50 | 0.038 | 4.0 | 83.0 | 0.99560 | 2.89 | 0.30 | 10.1 | 4 |
| 169 | 9.8 | 0.420 | 0.48 | 9.85 | 0.034 | 5.0 | 110.0 | 0.99580 | 2.87 | 0.29 | 10.0 | 5 |
| 207 | 10.2 | 0.440 | 0.88 | 6.20 | 0.049 | 20.0 | 124.0 | 0.99680 | 2.99 | 0.51 | 9.9 | 4 |
| 294 | 9.1 | 0.590 | 0.38 | 1.60 | 0.066 | 34.0 | 182.0 | 0.99680 | 3.23 | 0.38 | 8.5 | 3 |
| 358 | 10.0 | 0.200 | 0.39 | 1.40 | 0.050 | 19.0 | 152.0 | 0.99400 | 3.00 | 0.42 | 10.4 | 6 |
| 551 | 9.2 | 0.250 | 0.34 | 1.20 | 0.026 | 31.0 | 93.0 | 0.99160 | 2.93 | 0.37 | 11.3 | 7 |
| 555 | 9.2 | 0.250 | 0.34 | 1.20 | 0.026 | 31.0 | 93.0 | 0.99160 | 2.93 | 0.37 | 11.3 | 7 |
| 656 | 9.0 | 0.270 | 0.35 | 4.90 | 0.028 | 27.0 | 95.0 | 0.99320 | 3.04 | 0.40 | 11.3 | 6 |
| 774 | 9.1 | 0.270 | 0.45 | 10.60 | 0.035 | 28.0 | 124.0 | 0.99700 | 3.20 | 0.46 | 10.4 | 9 |
| 847 | 9.2 | 0.340 | 0.27 | 1.20 | 0.026 | 17.0 | 73.0 | 0.99210 | 3.08 | 0.39 | 10.8 | 5 |
| 873 | 10.3 | 0.170 | 0.47 | 1.40 | 0.037 | 5.0 | 33.0 | 0.99390 | 2.89 | 0.28 | 9.6 | 3 |

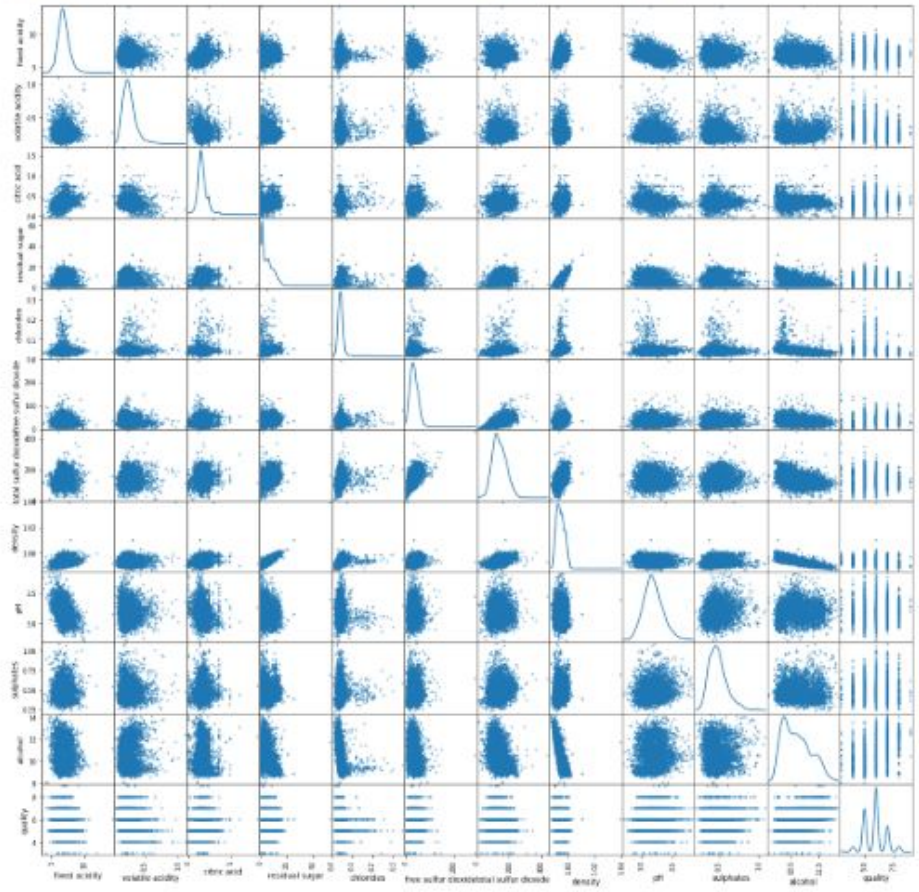
**There are more outliers present in the data. But , I am not going to remove them because sometimes wine manufacturers add exta composition of chemicals than required to improve the taste and quality. So , instead of considering them as outliers I am going to consider them as business strategies of different wine manufacturers.**

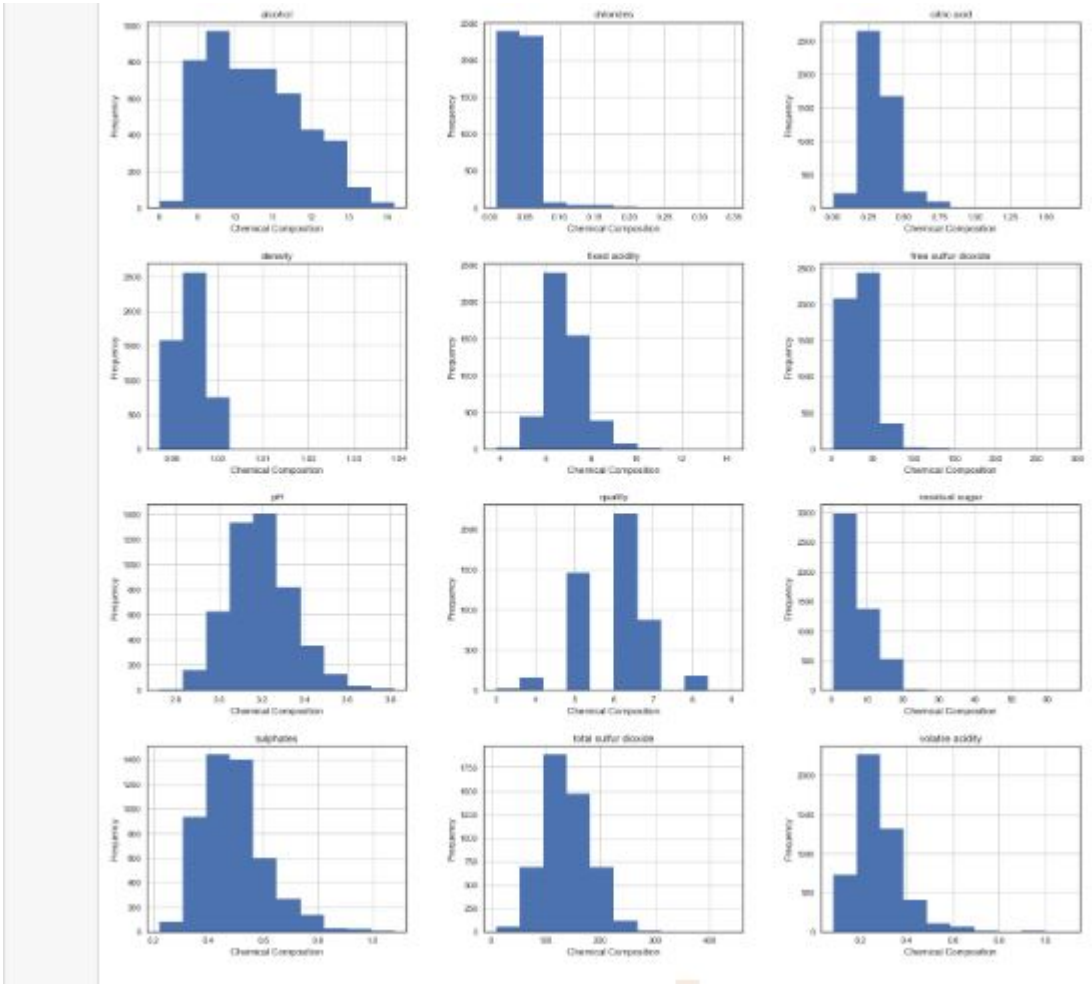
There are more outliers present in the data. But , I am not going to remove them because sometimes wine manufacturers add extra composition of chemicals than required to improve the taste and quality. So , instead of considering them as outliers I am going to consider them as business strategies of different wine manufacturers.

**Exploratory Visualizations**

The scatter plot below shows the relation between the attributes.

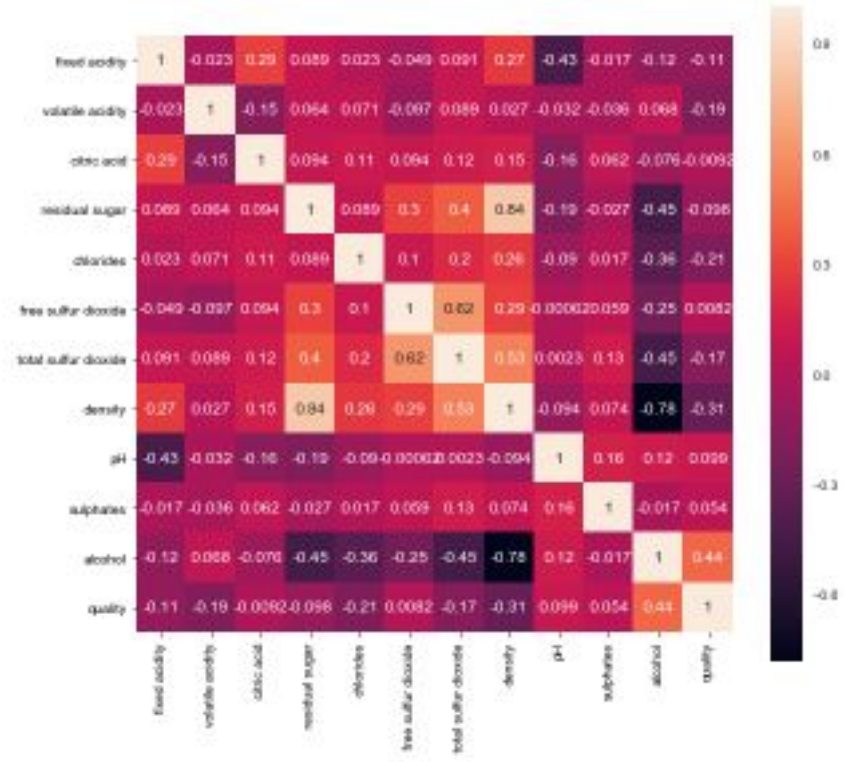
In [5]:
```
# plotting a scatter matrix to know the relationship ( correlation ) between the features
pd.plotting.scatter_matrix( wine_data , alpha = 0.5 , figsize = ( 20 , 20 ) , diagonal = 'kde' );
```



From the above visualizations , it is clear that most of the features are skewly - distributed . The problem with this type of data is very large values and very small values will affect the performance of our model. So , it is better to standardize them so that we will get good performance of model.

I wrote a piece of code to visualize a heatmap because it is easy to predict the correlation between the features.




From the above heatmap , what we can infer is there is that there is both positive and negative correlation exists in between the features .

Most of the features are not highly correlated . But for some , there exists a reasonable amount of correlation.

**pH - Fixed Acidity : -0.43**

It is reasonable because pH depends on acidity . It is negative correlation because as acidity increases pH decreases.

**Residual sugar - Density : 0.84**

Residual sugar and Density are positively correlated . If there is more residual sugar , then definitely density of wine also more.

**Free sulfur di-oxide - Total sulfur di-oxide : 0.62**

Total sulfur di-oxide constitutes Free sulfur d-oxide . If the content of one feature increases ,then other feature will also increase . Therefore , there exists positive correlation.

**Residual sugar - Total sulfur di-oxide : 0.4**

This correlation exists because if content of total sulfur di-oxide increases , then fermentation will stop earlier leaving most of the residual sugar which results in the higher density.

**Density - Total sulfur di-oxide : 0.53**

This correlation exists because if content of total sulfur di-oxide increases , then fermentation will stop earlier leaving most of the residual sugar which results in the higher density.

**Alcohol - Quality : 0.44**

The correlation is not strong because the increase in the alcohol content in wine increases the quality of wine only to a certain extent.

**Alcohol - Density : -0.78**

If alcohol content is more in wine , then it definitely leads to less density because liquid becomes less thick.

**Alcohol - Residual sugar : -0.45**

Since alcohol and density are correlated with one another , there exists a correlation in between residual sugar and density are correlated.

**Alcohol - Total sulfur - dioxide :**

Since alcohol and density are correlated with one another , there exists a correlation in between total sulfur di-oxide and density are correlated.

## Algorithms and Techniques

**Algorithms:**

The problem discussed here is a binary classification task( good quality - 1 or bad quality - 0) . So , I am going to choose three Supervised Classification Algorithms .Then I compare the performance of each model and select the best one. The three algorithms that I chosen are:

1. Logistic Regression :

I choose Logistic Regression for this problem because it is very simple classification algorithm which belongs to the family of Generalized Linear Model. It is more robust and to apply this algorithm , it doesn't require linear relationships exists in between predictors and target variables. The problem with this algorithm is it is more prone to over-fitting.

2. Decision Tree Classifier :

The Second algorithm that I choose is Decision Tree Classifier. The reason behind 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 algorithm is relatively easy and fast to predict the class of Target variable. The no. of Hyper tune parameters are almost null. The disadvantage of this algorithm is also over-fitting which can be solved by Ensemble learning algorithms.

3. Adaptive Boosting Classifier :

The final algorithm that I choose is Adaptive Boosting Classifier. It belongs to the family of Ensemble learners. It is one of the most powerful and efficient algorithm which boosts the performance (predictive power ) of a model by combining a set of weak learners into a single strong learner. This algorithm is less prone to Over-Fitting because it will allocate weights to set of weak learners which

then contribute to strong learner. If weak learner performs well , then it will contribute more weight to the weak learner while adding it to strong learner.

**Techniques :**

The technique that I am going to use to improve the performance of model is Grid Search.

1. Grid Search :

I am using Grid Search technique is to tune Hyper parameters because tuning Hyper parameters is very important part in improving performance of the model. Finding best Hyper parameters is a difficult task. So , 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.

**Working of Grid Search :**

Grid Search is one technique to find hyper parameters for better optimization where we give a set of possible values to parameters to be tunes (known as space) . The Grid search then assigns one value in space at a time and run the model for validation and this process continues and finally it returns the best model with hyper parameters which has high validation score.

## Benchmark :

The Benchmark model that I am taking is to predict the class of target variable as **Good Quality 1** irrespective of the chemical composition of the wine . But this will give very bad results and we'll use SL algorithms to achieve reasonable accuracy. The reason why I choose this as a Benchmark model because most of the wines in the data are good quality and no wine manufacture wants to prepare bad quality wine which is a loss to his company because nobody wants to buy the bad quality wine. Sometimes , due to several reasons Bad quality wine is formed.

**The performance of the model can be seen from the figure below:**

**Performance of Benchmark model**

Our Benchmark model will provide the wine quality always 1 irrespective of chemical composition. So , performance ( F- score ) will be pretty low. Let's check it.

```
In [12]: y_pred = np.ones(980)
         print('F-score of Benchmark model is',fbeta_score( y_test , y_pred , beta = 0.5))
F-score of Benchmark model is 0.7102210549354344
```

**As per our guess , the F - score is low ( around 71 % ).**

So , using Machine Learning classification algorithms we have to obtain better performance than Benchmark model.

# III . Methodology

## Data Pre-processing :

At this Pre-processing step , the things I did were:

1. I divided the data into Predictor variables ( X ) and Target variable ( y ).
2. I modified the data of the Quality attribute ( Target variable ) to suit it for classification problem. I am modifying the data of Quality attribute to 1 and 0 depending on the quality rating of wine greater than 5 or not.

**Converting Quality Variable to suit it for Classification task**

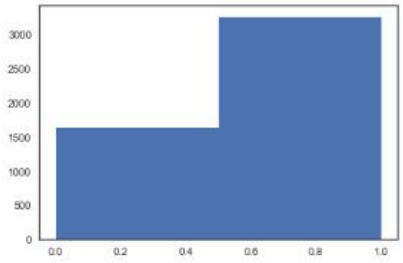
```
In [9]: # Dividing the data into Predictors and Target variables

        X = wine_data.drop( 'quality' , axis = 1 )


        # If quality of wine is greater than 5 ,then quality is good i.e., target variable is 1 else 0 for bad quaity
        y = pd.Series([ 1 if x > 5 else 0 for x in wine_data['quality'] ])

        plt.hist( y , bins = 2 )

Out[9]: (array([1640., 3258.]), array([0. , 0.5, 1. ]), <a list of 2 Patch objects>)
```




3. I splitted the total data into training and testing data.

**Splitting the total data into training and Testing data**

```
In [10]: # Dividing the data into training and testing set


         X_train , X_test , y_train , y_test = train_test_split( X , y , test_size = 0.2 , random_state = 29 )

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

(3918, 11) (3918,)
(980, 11) (980,)
```

4. I standardize the data because most of the ML algorithms works on the assumption that the data is normalized.

**Standardizing the Data**

```
In [11]: # Standardizing the data


         scaler = StandardScaler().fit(X_train)

         X_train_scaled = scaler.transform(X_train)
         X_test_scaled = scaler.transform(X_test)

         print(X_train_scaled.mean(axis=0))
         print(X_train_scaled.std(axis=0))
         print(X_test_scaled.mean(axis=0))
         print(X_test_scaled.std(axis=0))

[ 4.74239217e-16  2.39386527e-16 -4.71518916e-17  9.06767146e-19
 -1.77726361e-16  9.52105504e-17 -1.19693263e-16 -2.36616353e-14
  2.66226834e-15 -2.98326391e-16  7.45362594e-16]
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
[ 0.03969588  0.01213103 -0.01524392 -0.05946248 -0.0761042  -0.0549612
 -0.04591478 -0.04677059 -0.08594093  0.04567527  0.01062148]
[1.03701062 1.02449068 1.06264727 0.96376595 0.82703223 1.13395603
 0.98950384 0.94137479 1.00230283 1.04273032 0.98884971]
```

## Implementation :

To solve the problem , I choose three popular classification algorithms ( Logistic Regressor , Decision Tree Classifier and Random Forest Classifier ) and I wrote a function to check the performance of each model which can been seen from the figure below.

```
In [13]: def performance(clf, X_train_scaled, y_train, X_test_scaled, y_test):

             print('Classifier Performance :',clf.__class__.__name__)

             print()


             t1 = time()

             clf.fit(X_train , y_train)     # Calculating the training time

             t2 = time()

             print('Training time :',t2 - t1)

             t1 = time()

             y_preds = clf.predict(X_test)

             t2 = time()

             print('Testing time : ',t2 - t1)    # Calculating the Testing time

             print('F-score is',fbeta_score( y_test , y_preds , beta = 0.5))

             print('\n\n')
```

Then I checked the performance of each model by fitting data with each classifier and measure their performance using above function. The results can be seen from the figure below.

```
In [27]: clf1 = LogisticRegression(random_state = 1)
         clf2 = DecisionTreeClassifier(random_state = 1)
         clf3 = AdaBoostClassifier(random_state = 1)


         for clf in [clf1 , clf2 ,clf3]:
             performance(clf , X_train_scaled, y_train, X_test_scaled, y_test)

Classifier Performance : LogisticRegression

Training time : 0.026017427444458008
Testing time :  0.0
F-score is 0.7977130411108087


Classifier Performance : DecisionTreeClassifier

Training time : 0.03302192687988281
Testing time :  0.013011693954467773
F-score is 0.8424678985280301


Classifier Performance : AdaBoostClassifier

Training time : 0.2375178337097168
Testing time :  0.010000944137573242
F-score is 0.8311837202636858
```

Now , the crucial step is to find the best classifier. I eliminated Logistic Regressor because the F-score is low compared to Decision Tree Classifier and AdaBoostClassifier.

We got somewhat similar F - score for both AdaBoost classifier & Decision Tree classifier but the training time was very high in case of Ada Boost Classifier.

I am going to choose Ada Boost Classifier as my final model because it has more hyper parameters to tune than Decision Trees which will increase F - score to a quite bit .

**If I choose Decision tree as my final model , then there will be less no. of hyper parameters to tune (almost null ) and Decision Trees has over-fitting problem if more complex data is added to the dataset. That's why I am choosing AdaBoost classifier as my final model.**

**Complications that I faced during the Coding process :**

I haven't faced any difficulties during the coding process much because of Udacity well-formatted teaching. The only problem that I faced is choosing the right evaluation metric as my data is unbalanced. I choose F -score and the coding process for that is so simple.

## Refinement :

I used Grid Search technique to refine my model performance. The initial and final model solutions are not significantly different. Previously , without tuning I got an F-score of 83.11% and after tuning I got an accuracy of 83.50 %.

The results are here :

**Hyper Parameter Tuning using Grid Search**

In [15]:
```
clf = AdaBoostClassifier()

h_params = { 'n_estimators' : [1 , 10 , 50 , 75 , 100] , 'learning_rate' : [0.1 , 0.2, 0.3 , 0.35 , 0.5] }

scorer = make_scorer( fbeta_score , beta=0.5 )

grid = GridSearchCV(clf , param_grid=h_params , scoring=scorer )

best_model  = grid.fit( X_train , y_train ).best_estimator_
```

In [16]:
```
print('Best Model Performance')
print('Model:',best_model)

performance(best_model , X_train_scaled, y_train, X_test_scaled, y_test)

y_preds = best_model.predict(X_test)
```

```
Best Model Performance
Model: AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=0.3, n_estimators=100, random_state=None)
Classifier Performance : AdaBoostClassifier

Training time : 0.5118050575256348
Testing time :  0.020000219345092773
F-score is 0.8350014297969687
```

# IV . Results

## Model Evaluation and Validation :

The final model ( AdaBoost Classifier with hyper parameter tuning using Grid Search ) achieved an F - score of around 84% which is pretty good. The final model parameters are :

learning_rate : 0.3

If Learning rate is less , then our model will learn slowly which gives better performance than default value (1.0)

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.

Un - optimized model : AdaBoostClassifier before tuning

Optimized model : AdaBoostClassifier after tuning

## Justification :

| Metric | Benchmark model | Un-optimized model | Optimized model |
|---|---|---|---|
| F-score | 0.7102 | 0.8311 | 0.8350 |

From the above table , we can clearly say that our final model works pretty well than our Benchmark model . As our final model got F-score of around 84% the model is quite significant to solve the problem because our benchmark model always predicting the quality of wine to be good whereas our classifier trying to figure out patterns in the data and then predicts the class of Target variable.

# V . Conclusion

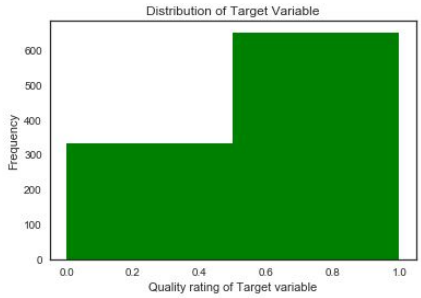
## Free – form Visualization :

Below I provide some visualizations to emphasize the performance of my model against actual values in the target variable . The predictions that I am comparing are obtained from Optimised model ( Ada Boost Classifier with parameter tuning.)

```
In [31]: plt.hist(y_test , bins = 2 , color = 'g')

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

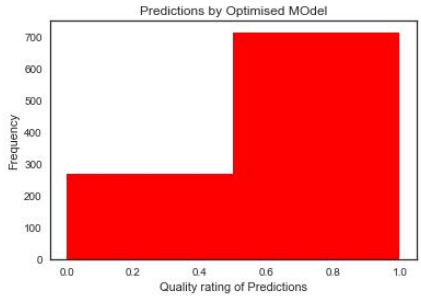
```
Out[31]: Text(0.5,1,'Distribution of Target Variable')
```




```
In [32]: plt.hist(y_preds , bins = 2 , color = 'r')

         plt.xlabel('Quality rating of Predictions')
         plt.ylabel('Frequency')
         plt.title('Predictions by Optimised MOdel')
```

```
Out[32]: Text(0.5,1,'Predictions by Optimised MOdel')
```




From the above visualizations , it is clear that our model is wrongly predicting bad quality of wine as good quality. This is expected because our data contains more examples of good quality wine than bad quality wine.

So, if the data is more or balanced , then we may even get better performance.

## Reflection :

- First , I downloaded the data from UCI machine learning repository and read the data using Pandas DataFrame.
- Then I defined my problem statement as to predict the quality of wine as good or bad depending upon the chemical composition of wine.
- Then I found the no. of instances and attributes present in the data at the Data exploration step.
- Then I performed Bi-variate analysis to see whether the features are correlated or not. Some of the features are correlated and I briefly described the reason behind Correlation.
- At the Data Pre-processing step , I tried to find missing values but there were no missing values. Then I tried to find out Outliers in the data and I just gave my opinion about not removing outliers in the data and after that I standardized the data.

- Then I defined a Benchmark model which will predict the quality of wine as always good but it gave me bad performance as expected.
- So , I choose three algorithms Logistic Regressor , Decision Tree and Ada Boost Classifier out of which I choose Ada Boost Classifier with proper reasoning.
- I tried to improve the performance of Ada Boost Classifier with Grid Search technique ( Hyper parameter optimization) and F-score increased a little bit.
- Overall , I had a great experience with the project . I struggled a little bit at the Hyper parameter optimization since the F-score was not that much significant than un-optimized model ( even Decision Tree has better performance but it will overfit if more complex data is added).

## Improvement :

I think results can be improved further if :

- we have a lot more balanced data about the chemical composition of wine
- Using CNN we can achieve better performance because CNN's has become the most effective tool in ML which is well suited for Classification problems.
- XGBoost and LIghtGBM models also achieve good performance to this problem (suggested by reviewer 1 ) to boost the performance and to improve the predictive power.