

Table of Contents[](https://machinelearningprojects.net/wine-quality-prediction/)

* [Step 1 – Importing libraries required for Wine Quality Prediction.](https://machinelearningprojects.net/wine-quality-prediction/#Step_1_-_Importing_libraries_required_for_Wine_Quality_Prediction)
* [Step 2 – Read input data.](https://machinelearningprojects.net/wine-quality-prediction/#Step_2_-_Read_input_data)
* [Step 3 – Describe the data.](https://machinelearningprojects.net/wine-quality-prediction/#Step_3_-_Describe_the_data)
* [Step 4 – Take info from the data.](https://machinelearningprojects.net/wine-quality-prediction/#Step_4_-_Take_info_from_the_data)
* [Step 5 – Plot out the data.](https://machinelearningprojects.net/wine-quality-prediction/#Step_5_-_Plot_out_the_data)
* [Step 6 – Count the no. of instances of each class.](https://machinelearningprojects.net/wine-quality-prediction/#Step_6_-_Count_the_no_of_instances_of_each_class)
* [Step 7 – Make just 2 categories good and bad.](https://machinelearningprojects.net/wine-quality-prediction/#Step_7_-_Make_just_2_categories_good_and_bad)
* [Step 8 – Alloting 0 to bad and 1 to good.](https://machinelearningprojects.net/wine-quality-prediction/#Step_8_-_Alloting_0_to_bad_and_1_to_good)
* [Step 9 – Again check counts.](https://machinelearningprojects.net/wine-quality-prediction/#Step_9_-_Again_check_counts)
* [Step 10 – Balancing the two classes.](https://machinelearningprojects.net/wine-quality-prediction/#Step_10_-_Balancing_the_two_classes)
* [Step 11 – Again check the counts of classes in the new dataframe.](https://machinelearningprojects.net/wine-quality-prediction/#Step_11_-_Again_check_the_counts_of_classes_in_the_new_dataframe)
* [Step 12 – Checking the correlation between columns.](https://machinelearningprojects.net/wine-quality-prediction/#Step_12_-_Checking_the_correlation_between_columns)
* [Step 13 – Splitting the data into train and test.](https://machinelearningprojects.net/wine-quality-prediction/#Step_13_-_Splitting_the_data_into_train_and_test)
* [Step 14 – Finally training our Wine Quality Prediction model.](https://machinelearningprojects.net/wine-quality-prediction/#Step_14_-_Finally_training_our_Wine_Quality_Prediction_model)

Step 1 – Importing libraries required for Wine Quality Prediction.

import numpy as np

import pandas as pd

import seaborn as sns

from sklearn.svm import SVC

import matplotlib.pyplot as plt

from sklearn.linear\_model import SGDClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

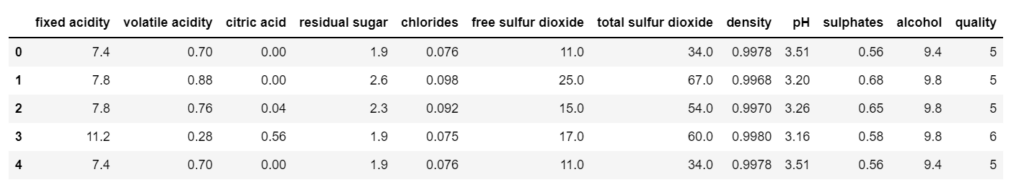
from sklearn.metrics import classification\_report,confusion\_matrix,accuracy\_score

%matplotlib inline

Step 2 – Read input data.

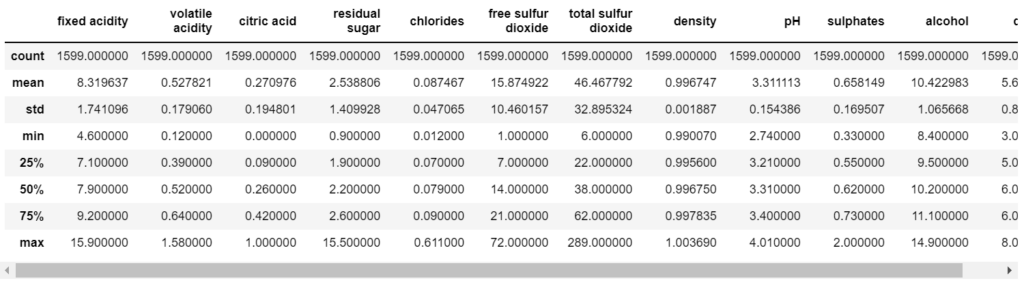
wine = pd.read\_csv**(**'winequality-red.csv'**)**

wine.head**()**

Input data

Step 3 – Describe the data.

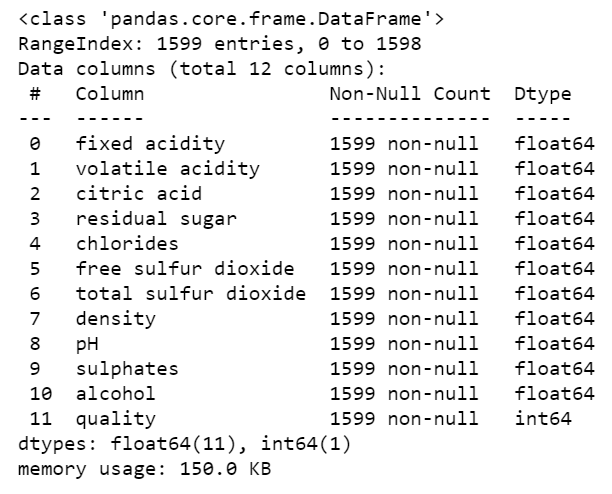
wine.describe**()**

Description of our data

Step 4 – Take info from the data.

wine.info**()**

* From the data below, we can infer that there is no NULL value in our data.

Info of our data

Step 5 – Plot out the data.

fig = plt.figure**(**figsize=**(**15,10**))**

plt.subplot**(**3,4,1**)**

sns.barplot**(**x='quality',y='fixed acidity',data=wine**)**

plt.subplot**(**3,4,2**)**

sns.barplot**(**x='quality',y='volatile acidity',data=wine**)**

plt.subplot**(**3,4,3**)**

sns.barplot**(**x='quality',y='citric acid',data=wine**)**

plt.subplot**(**3,4,4**)**

sns.barplot**(**x='quality',y='residual sugar',data=wine**)**

plt.subplot**(**3,4,5**)**

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

plt.subplot**(**3,4,6**)**

sns.barplot**(**x='quality',y='free sulfur dioxide',data=wine**)**

plt.subplot**(**3,4,7**)**

sns.barplot**(**x='quality',y='total sulfur dioxide',data=wine**)**

plt.subplot**(**3,4,8**)**

sns.barplot**(**x='quality',y='density',data=wine**)**

plt.subplot**(**3,4,9**)**

sns.barplot**(**x='quality',y='pH',data=wine**)**

plt.subplot**(**3,4,10**)**

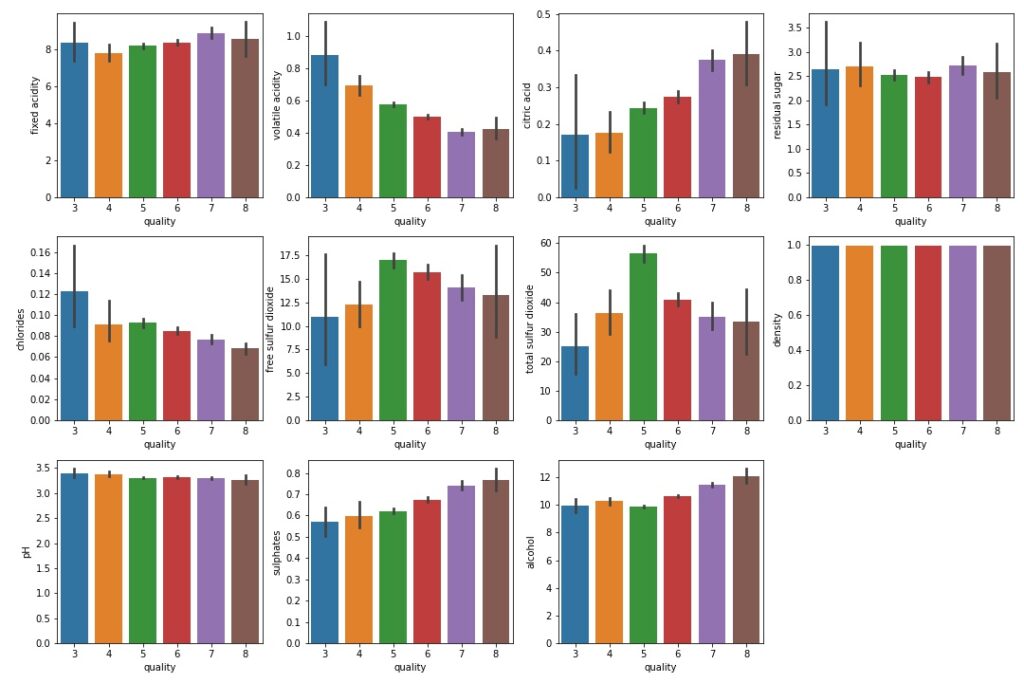
sns.barplot**(**x='quality',y='sulphates',data=wine**)**

plt.subplot**(**3,4,11**)**

sns.barplot**(**x='quality',y='alcohol',data=wine**)**

plt.tight\_layout**()**

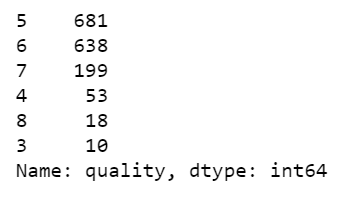
* From the plots below we can infer:
* Quality is high when volatile acidity is less.
* Quality is high when citric acid is high.
* Quality is high when chlorides are less.
* Quality is high when sulphates are more.
* Quality is high when alcohol is more.

plots

Step 6 – Count the no. of instances of each class.

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

* We can see that we have 6 classes of quality that are 3,4,5,6,7,8 but we don’t want it like this.
* So what we will do is we will mark every rating from 3 to 6 as BAD and ratings of 7 and 8 as GOOD.

value counts

Step 7 – Make just 2 categories good and bad.

ranges = **(**2,6.5,8**)**

groups = **[**'bad','good'**]**

wine**[**'quality'**]** = pd.cut**(**wine**[**'quality'**]**,bins=ranges,labels=groups**)**

* Here we are cutting bins use [pd.cut()](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.cut.html" \t "_blank) in 2 categories 2-6.5 as BAD and 6.5-8 as GOOD.

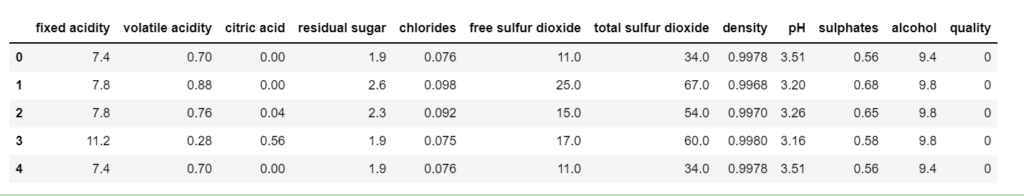
Step 8 – Alloting 0 to bad and 1 to good.

le = LabelEncoder**()**

wine**[**'quality'**]** = le.fit\_transform**(**wine**[**'quality'**])**

wine.head**()**

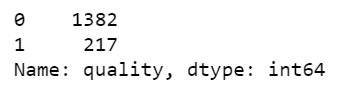
* Replace BAD with 0.
* Replace GOOD with 1.
* For reference see the quality column in the image below.

labels encoded to 0/1

Step 9 – Again check counts.

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

* Now we have just 2 classes, 0 and 1 or BAD and GOOD.
* But as we can see that the data is highly unbalanced, so we will balance it in the next step.



Unequal value counts

Step 10 – Balancing the two classes.

good\_quality = wine**[**wine**[**'quality'**]**==1**]**

bad\_quality = wine**[**wine**[**'quality'**]**==0**]**

bad\_quality = bad\_quality.sample**(**frac=1**)**

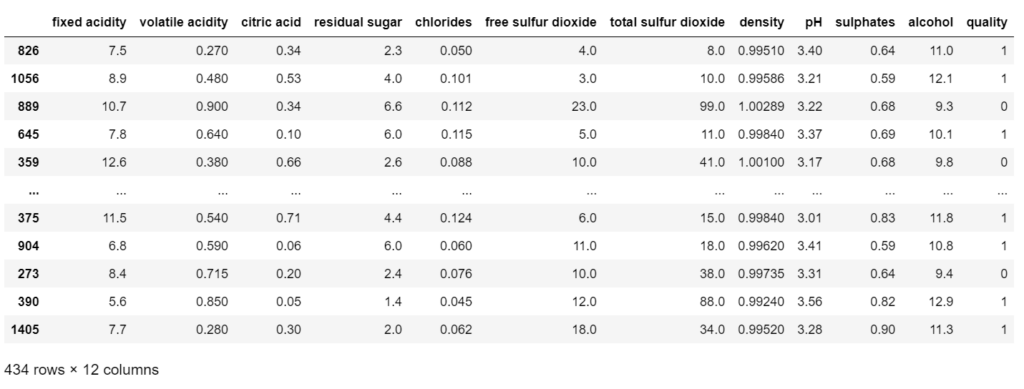
bad\_quality = bad\_quality**[**:217**]**

new\_df = pd.concat**([**good\_quality,bad\_quality**])**

new\_df = new\_df.sample**(**frac=1**)**

new\_df

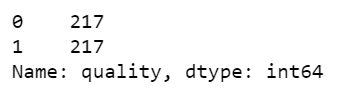
* In this step, we are simply balancing our dataset.
* We are making a new data frame good\_quality in which we will have data of just good\_quality wine or we can say where the quality is 1.
* Similarly, we are making for bad\_quality.
* Then we are simply shuffling bad quality data using [df.sample(frac=1)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sample.html" \t "_blank). It means shuffle the data and take a 100% fraction of the data.
* Then we are taking out 217 samples of bad\_quality because we have just 217 samples of good\_quality.
* Then we are joining both 217 samples of each class and our final data frame will have 217\*2=434 rows.
* Finally, again shuffling the data.

balanced dataset

Step 11 – Again check the counts of classes in the new dataframe.

new\_df**[**'quality'**]**.value\_counts**()**

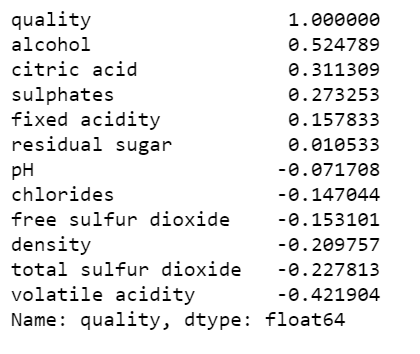
* Now we can see that both the classes have 217 instances and hence our data is shuffled.

Equal value counts

Step 12 – Checking the correlation between columns.

new\_df.corr**()[**'quality'**]**.sort\_values**(**ascending=False**)**

* From the image below we can infer that quality is highly dependent on the alcohol quantity in the wine.



Correlations

Step 13 – Splitting the data into train and test.

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

X = new\_df.drop**(**'quality',axis=1**)**

y = new\_df**[**'quality'**]**

X\_train, X\_test, y\_train, y\_test = train\_test\_split**(**X, y, test\_size=0.3, random\_state=101**)**

Step 14 – Finally training our Wine Quality Prediction model.

param = **{**'n\_estimators':**[**100,200,300,400,500,600,700,800,900,1000**]}**

grid\_rf = GridSearchCV**(**RandomForestClassifier**()**,param,scoring='accuracy',cv=10,**)**

grid\_rf.fit**(**X\_train, y\_train**)**

print**(**'Best parameters --> ', grid\_rf.best\_params\_**)**

# Wine Quality Prediction

pred = grid\_rf.predict**(**X\_test**)**

print**(**confusion\_matrix**(**y\_test,pred**))**

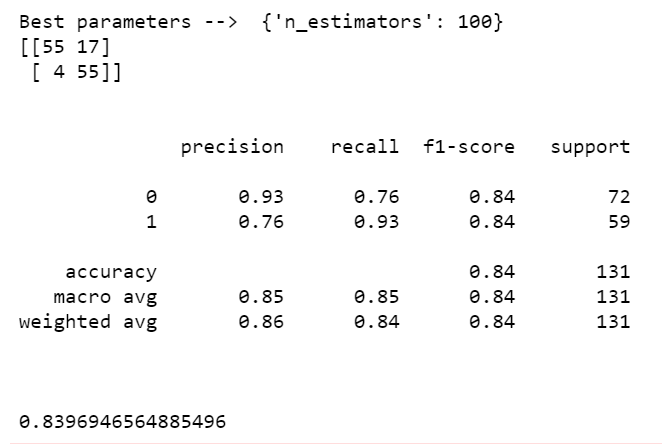
print**(**'\n'**)**

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

print**(**'\n'**)**

print**(**accuracy\_score**(**y\_test,pred**))**

* I also used some other algorithms like SVM and SGD Classifier but Random Forest stood out as always.
* Here I have used GridSearchCV with Random Forest to find the best value of the ‘n\_estimators’ parameter.
* Finally, we ended up with an accuracy of 83.9% which is very good for this much small dataset.

final metrics

