Name: Kelvin J Anil
Reg No: 21BCE0002
Email: kelvin.janil2021@vitstudent.ac.in
Assignment 4

Colab Link:
https://colab.research.google.com/drive/1HfkORhIzm5FbYZ52ircMKZkclFXTcAS-?usp=sharing

**Project Title:**

Grapes to Greatness: Machine Learning in Wine Quality Prediction

**Description:**

Predicting wine quality using machine learning is a common and valuable application in the field of data science and analytics. Wine quality prediction involves building a model that can assess and predict the quality of a wine based on various input features, such as chemical composition, sensory characteristics, and environmental factors.

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For more details, consult the reference [Cortez et al., 2009]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones).

**Dataset:** link

**Task**:

- Load the Dataset
- Data preprocessing including visualization
- Machine Learning Model building
- Evaluate the model
- Test with random observation

```
#Load the Dataset
import pandas as pd
import numpy as np
wine_ds=pd.read_csv("/content/winequality-red.csv")
wine_ds
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7.4 | 0.700 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.99780 | 3.51 | 0.56 | 9.4 | 5 |
| 1 | 7.8 | 0.880 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.99680 | 3.20 | 0.68 | 9.8 | 5 |
| 2 | 7.8 | 0.760 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.99700 | 3.26 | 0.65 | 9.8 | 5 |
| 3 | 11.2 | 0.280 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.99800 | 3.16 | 0.58 | 9.8 | 6 |
| 4 | 7.4 | 0.700 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.99780 | 3.51 | 0.56 | 9.4 | 5 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1594 | 6.2 | 0.600 | 0.08 | 2.0 | 0.090 | 32.0 | 44.0 | 0.99490 | 3.45 | 0.58 | 10.5 | 5 |
| 1595 | 5.9 | 0.550 | 0.10 | 2.2 | 0.062 | 39.0 | 51.0 | 0.99512 | 3.52 | 0.76 | 11.2 | 6 |
| 1596 | 6.3 | 0.510 | 0.13 | 2.3 | 0.076 | 29.0 | 40.0 | 0.99574 | 3.42 | 0.75 | 11.0 | 6 |
| 1597 | 5.9 | 0.645 | 0.12 | 2.0 | 0.075 | 32.0 | 44.0 | 0.99547 | 3.57 | 0.71 | 10.2 | 5 |
| 1598 | 6.0 | 0.310 | 0.47 | 3.6 | 0.067 | 18.0 | 42.0 | 0.99549 | 3.39 | 0.66 | 11.0 | 6 |

1599 rows × 12 columns

```
#Checking for missing values
wine_ds.isnull().sum()
```

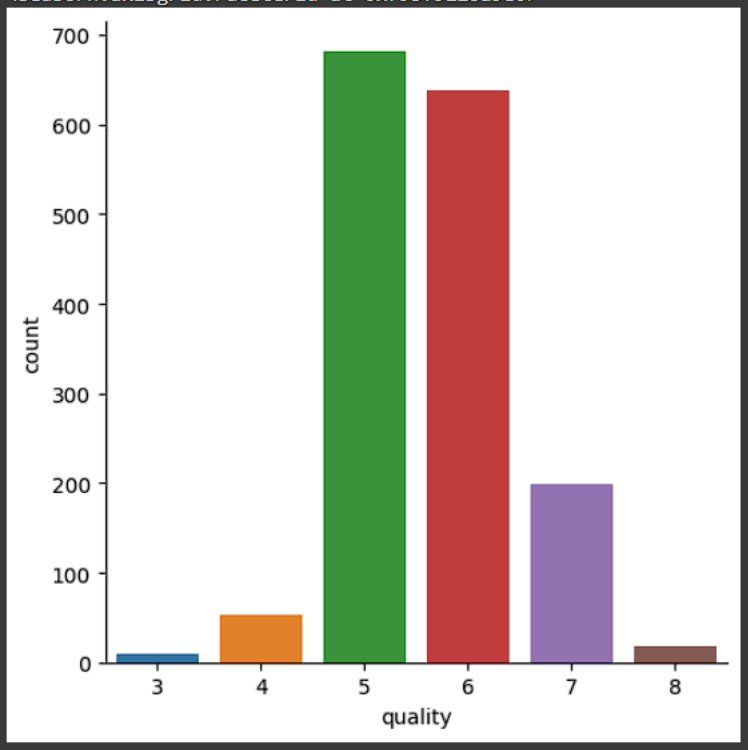
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality   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

```
# Since there are no missing values, next step Data Analysis
# Statistical Description
wine_ds.describe()
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 |
| mean | 8.319637 | 0.527821 | 0.270976 | 2.538806 | 0.087467 | 15.874922 | 46.467792 | 0.996747 | 3.311113 | 0.658149 | 10.422983 | 5.636023 |
| std | 1.741096 | 0.179060 | 0.194801 | 1.409928 | 0.047065 | 10.460157 | 32.895324 | 0.001887 | 0.154386 | 0.169507 | 1.065668 | 0.807569 |
| min | 4.600000 | 0.120000 | 0.000000 | 0.900000 | 0.012000 | 1.000000 | 6.000000 | 0.990070 | 2.740000 | 0.330000 | 8.400000 | 3.000000 |
| 25% | 7.100000 | 0.390000 | 0.090000 | 1.900000 | 0.070000 | 7.000000 | 22.000000 | 0.995600 | 3.210000 | 0.550000 | 9.500000 | 5.000000 |
| 50% | 7.900000 | 0.520000 | 0.260000 | 2.200000 | 0.079000 | 14.000000 | 38.000000 | 0.996750 | 3.310000 | 0.620000 | 10.200000 | 6.000000 |
| 75% | 9.200000 | 0.640000 | 0.420000 | 2.600000 | 0.090000 | 21.000000 | 62.000000 | 0.997835 | 3.400000 | 0.730000 | 11.100000 | 6.000000 |
| max | 15.900000 | 1.580000 | 1.000000 | 15.500000 | 0.611000 | 72.000000 | 289.000000 | 1.003690 | 4.010000 | 2.000000 | 14.900000 | 8.000000 |

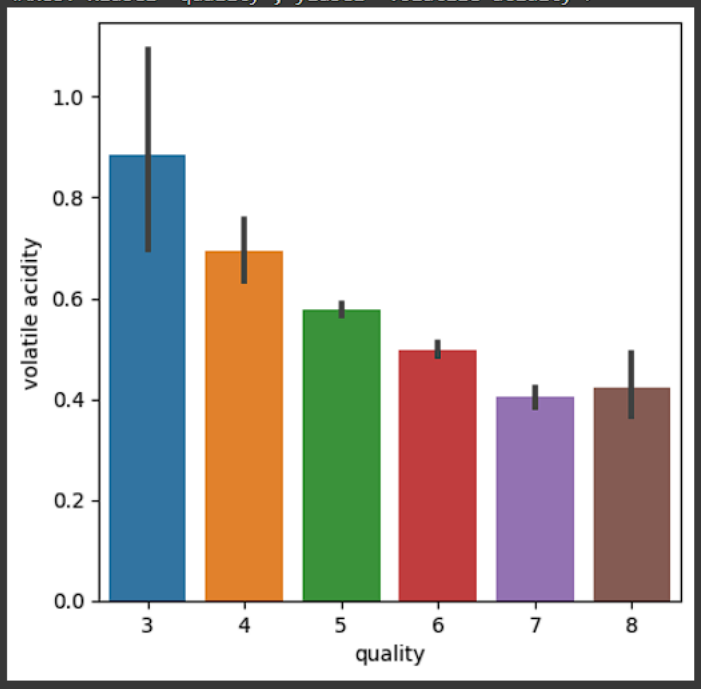
```python
# quality values
import seaborn as sns
sns.catplot(x = 'quality', data = wine_ds , kind = 'count')
```

```
<seaborn.axisgrid.FacetGrid at 0x795f9228a9e0>
```



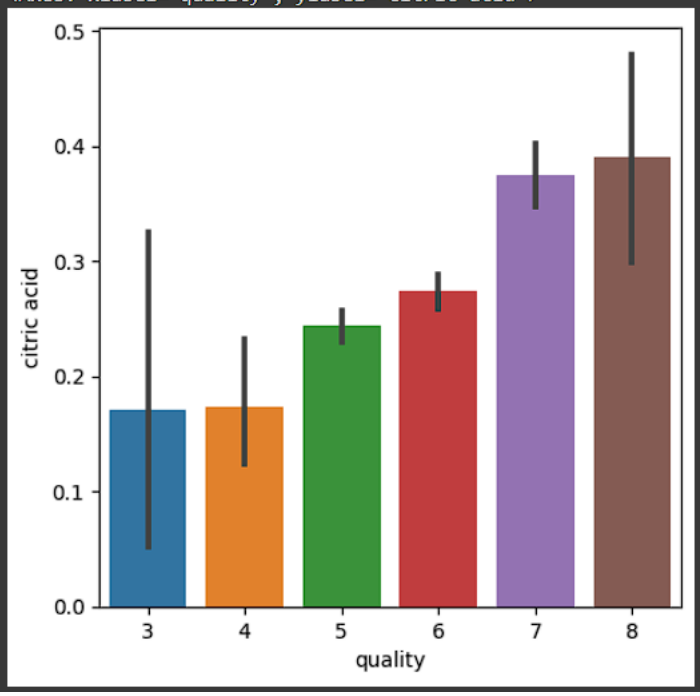
```python
# find which feature is related to quality
# lets take the first one
# 1. quality vs volatile acidity
import matplotlib.pyplot as plt
plot = plt.figure(figsize = (5,5))
sns.barplot(x = 'quality', data = wine_ds , y='volatile acidity' )
```

```
<Axes: xlabel='quality', ylabel='volatile acidity'>
```



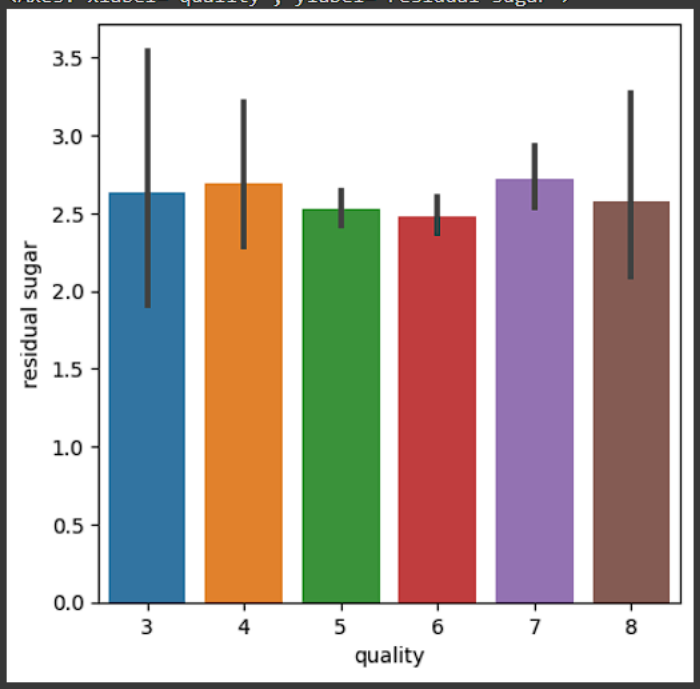
```python
# 2.  quality vs citric acidity
plot = plt.figure(figsize = (5,5))
sns.barplot(x = 'quality', data = wine_ds , y = 'citric acid')
```

```
<Axes: xlabel='quality', ylabel='citric acid'>
```



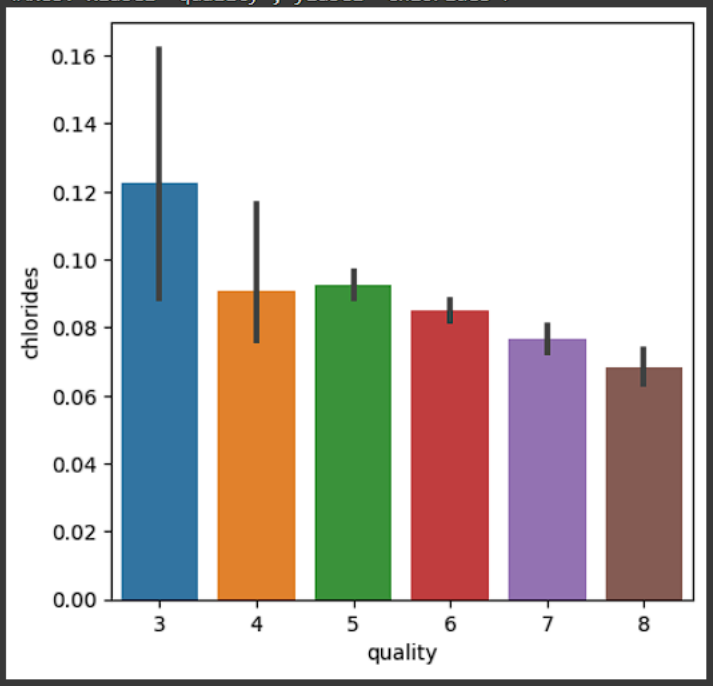
```
# 3.  quality vs residual sugar
plot = plt.figure(figsize = (5,5))
sns.barplot(x = 'quality', data = wine_ds, y = 'residual sugar')
```

```
<Axes: xlabel='quality', ylabel='residual sugar'>
```



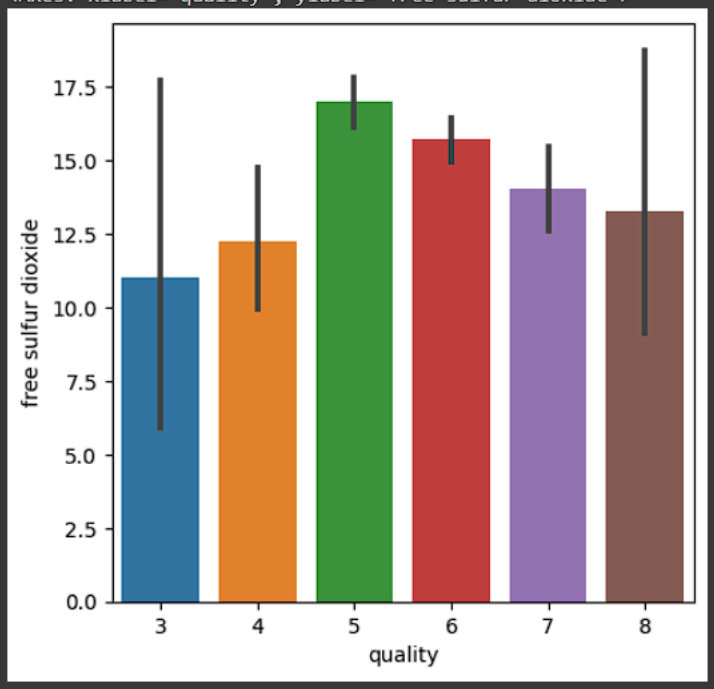
```
# 4. quality vs chlorides
plot = plt.figure(figsize = (5,5))
sns.barplot(x = 'quality', data = wine_ds, y ='chlorides')
```

```
<Axes: xlabel='quality', ylabel='chlorides'>
```



```
# 5. quality vs free sulphur dioxide
plot = plt.figure(figsize = (5,5))
sns.barplot(x = 'quality' , data = wine_ds , y='free sulfur dioxide')
```

```
<Axes: xlabel='quality', ylabel='free sulfur dioxide'>
```

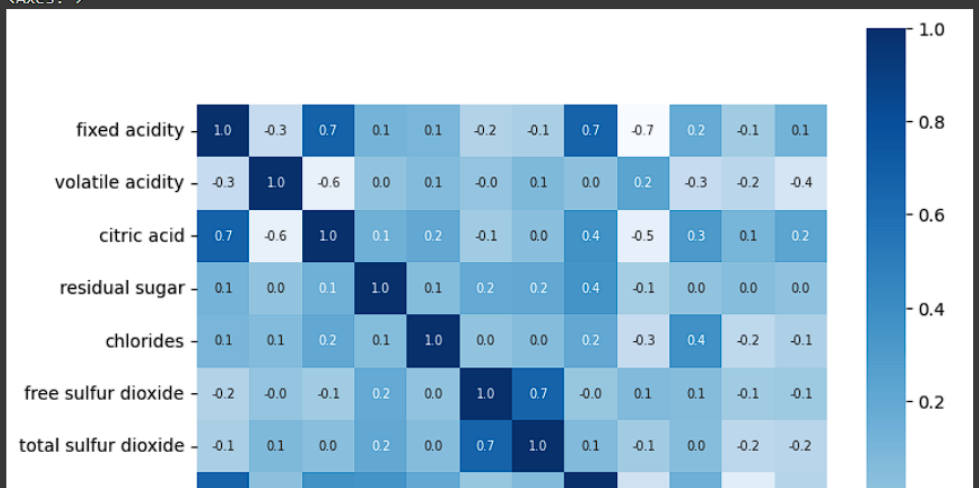


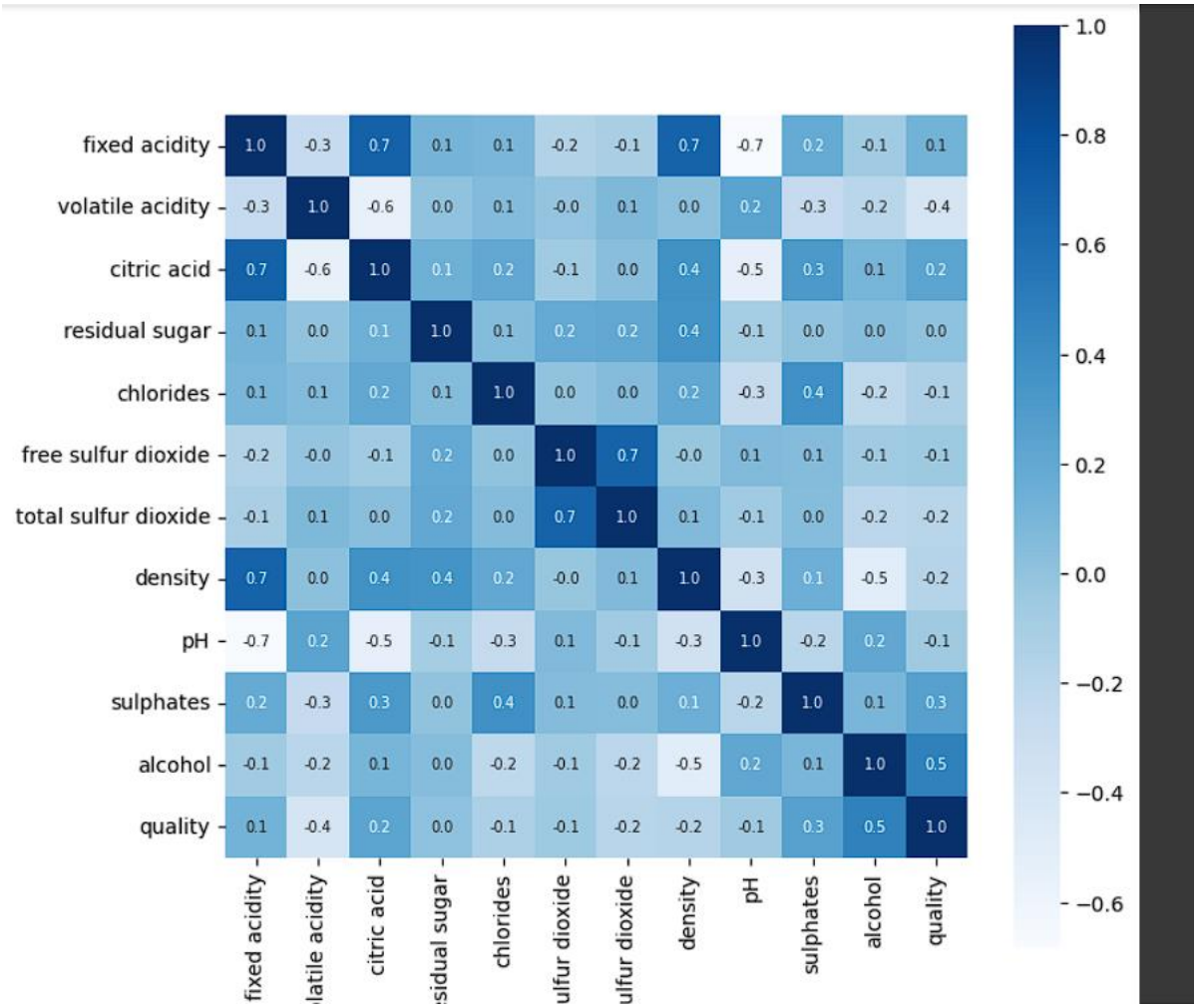

```
# we will find the correlation between quality with each feature
cor = wine_ds.corr()
cor
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| fixed acidity | 1.000000 | -0.256131 | 0.671703 | 0.114777 | 0.093705 | -0.153794 | -0.113181 | 0.668047 | -0.682978 | 0.183006 | -0.061668 | 0.124052 |
| volatile acidity | -0.256131 | 1.000000 | -0.552496 | 0.001918 | 0.061298 | -0.010504 | 0.076470 | 0.022026 | 0.234937 | -0.260987 | -0.202288 | -0.390558 |
| citric acid | 0.671703 | -0.552496 | 1.000000 | 0.143577 | 0.203823 | -0.060978 | 0.035533 | 0.364947 | -0.541904 | 0.312770 | 0.109903 | 0.226373 |
| residual sugar | 0.114777 | 0.001918 | 0.143577 | 1.000000 | 0.055610 | 0.187049 | 0.203028 | 0.355283 | -0.085652 | 0.005527 | 0.042075 | 0.013732 |
| chlorides | 0.093705 | 0.061298 | 0.203823 | 0.055610 | 1.000000 | 0.005562 | 0.047400 | 0.200632 | -0.265026 | 0.371260 | -0.221141 | -0.128907 |
| free sulfur dioxide | -0.153794 | -0.010504 | -0.060978 | 0.187049 | 0.005562 | 1.000000 | 0.667666 | -0.021946 | 0.070377 | 0.051658 | -0.069408 | -0.050656 |
| total sulfur dioxide | -0.113181 | 0.076470 | 0.035533 | 0.203028 | 0.047400 | 0.667666 | 1.000000 | 0.071269 | -0.066495 | 0.042947 | -0.205654 | -0.185100 |
| density | 0.668047 | 0.022026 | 0.364947 | 0.355283 | 0.200632 | -0.021946 | 0.071269 | 1.000000 | -0.341699 | 0.148506 | -0.496180 | -0.174919 |
| pH | -0.682978 | 0.234937 | -0.541904 | -0.085652 | -0.265026 | 0.070377 | -0.066495 | -0.341699 | 1.000000 | -0.196648 | 0.205633 | -0.057731 |
| sulphates | 0.183006 | -0.260987 | 0.312770 | 0.005527 | 0.371260 | 0.051658 | 0.042947 | 0.148506 | -0.196648 | 1.000000 | 0.093595 | 0.251397 |
| alcohol | -0.061668 | -0.202288 | 0.109903 | 0.042075 | -0.221141 | -0.069408 | -0.205654 | -0.496180 | 0.205633 | 0.093595 | 1.000000 | 0.476166 |
| quality | 0.124052 | -0.390558 | 0.226373 | 0.013732 | -0.128907 | -0.050656 | -0.185100 | -0.174919 | -0.057731 | 0.251397 | 0.476166 | 1.000000 |

```
# heatmap of all the features
plt.figure(figsize = (8,8))
sns.heatmap(cor, cbar = True, square = True, fmt = '.1f', annot = True, annot_kws = {'size': 7} , cmap = 'Blues')
```

```
<Axes: >
```



```
# Data Preprocessing
# Separate the quality feature
X = wine_ds.drop('quality',axis = 1)
X
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7.4 | 0.700 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.99780 | 3.51 | 0.56 | 9.4 |
| 1 | 7.8 | 0.880 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.99680 | 3.20 | 0.68 | 9.8 |
| 2 | 7.8 | 0.760 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.99700 | 3.26 | 0.65 | 9.8 |
| 3 | 11.2 | 0.280 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.99800 | 3.16 | 0.58 | 9.8 |
| 4 | 7.4 | 0.700 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.99780 | 3.51 | 0.56 | 9.4 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1594 | 6.2 | 0.600 | 0.08 | 2.0 | 0.090 | 32.0 | 44.0 | 0.99490 | 3.45 | 0.58 | 10.5 |
| 1595 | 5.9 | 0.550 | 0.10 | 2.2 | 0.062 | 39.0 | 51.0 | 0.99512 | 3.52 | 0.76 | 11.2 |
| 1596 | 6.3 | 0.510 | 0.13 | 2.3 | 0.076 | 29.0 | 40.0 | 0.99574 | 3.42 | 0.75 | 11.0 |
| 1597 | 5.9 | 0.645 | 0.12 | 2.0 | 0.075 | 32.0 | 44.0 | 0.99547 | 3.57 | 0.71 | 10.2 |
| 1598 | 6.0 | 0.310 | 0.47 | 3.6 | 0.067 | 18.0 | 42.0 | 0.99549 | 3.39 | 0.66 | 11.0 |

1599 rows × 11 columns

```
[47] # Label Binarization
     # quality >= 7 = Good and quality < 7 = Bad
     # Good = 1 and Bad = 0
     Y= wine_ds['quality'].apply(lambda y_value : 1 if y_value >= 7 else 0)
     Y
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1594    0
1595    0
1596    0
1597    0
1598    0
Name: quality, Length: 1599, dtype: int64
```

```
[48] # Train and Test Split
     # 0.2 means 20 percent of original data to be test data
     from sklearn.model_selection import train_test_split
     X_train, X_test, Y_train, Y_test = train_test_split (X,Y, test_size = 0.2 , random_state = 3)

[49] print(Y.shape,Y_train.shape,Y_test.shape)

     (1599,) (1279,) (320,)

[50] # Model Training
     # Random Forest Classifier
     from sklearn.ensemble import RandomForestClassifier
     model = RandomForestClassifier()
```

```
     model.fit(X_train, Y_train)
     # X_train contains all the data values like volatile acidity, citric acidity..
     # Y_train contains the quality values i.e., 0 or 1

     ▸ RandomForestClassifier

                                                                    + Code

[54] # Model Evalution
     # Accuracy Score Value
     from sklearn.metrics import accuracy_score

[58] # accuracy on test data
     X_test_prediction = model.predict(X_test)
     # compare original label values and values predicted by model
     test_data_accuracy = accuracy_score(X_test_prediction,Y_test)

[59] print("Accuracy: ",test_data_accuracy)

     Accuracy:  0.934375
```

```
# Testing with random observation
# 10th observation
input = (7.8,0.58,0.02,2,0.073,9,18,0.9968,3.36,0.57,9.5)
# changing into numpy array
input_arr = np.asarray(input)
# reshape the data as we are predicting for only 1 value
input_arr_reshaped = input_arr.reshape(1,-1)
# prediction
prediction = model.predict(input_arr_reshaped)
if(prediction ==1):
  print("Good Quality")
else:
  print("Bad Quality")
```

```
Good Quality
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

```
# Testing with random observation
# 23rd observation
input = (7.6,0.39,0.31,2.3,0.082,23,71,0.9982,3.52,0.65,9.7)
# changing into numpy array
input_arr = np.asarray(input)
# reshape the data as we are predicting for only 1 value
input_arr_reshaped = input_arr.reshape(1,-1)
# prediction
prediction = model.predict(input_arr_reshaped)
if(prediction ==1):
  print("Good Quality")
else:
  print("Bad Quality")
```

```
Bad Quality
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```