MAJOR PROJECT

A

PROJECT ON

WINE QUALITY ANALYSIS

Under Guidance of

Abin Varghese

Submitted By:-

Pritiranjan Biswal

INTRODUCTION: -

The aim of this project is to predict the quality of wine on a scale of 0–10 given a set of features as inputs. Input variables are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates, and alcohol. And the output variable is quality .We are dealing only with red wine. We have quality being one of these values: [3, 4, 5, 6, 7, 8]. The higher the value the better the quality. In this project we will treat each class of the wine separately and their aim is to be able and find decision boundaries that work well for new unseen data.

Dataset Description:

The dataset contains a total of 12 variables, which were recorded for 1,599 observations. This data will allow us to create different regression models to determine how different independent variables help predict our dependent variable, quality.

Input variables are

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

12) Quality

Preparing Wine Data

Correctly prepared data is the cornerstone of an effective machine learning model and accurate predictions.

1. **Standardizing feature variables**. The process of transforming the data to get a mean of 0 and a standard deviation of 1 in the data distribution. This helps even out the range of the wine data.
2. **Splitting data**. The process of splitting wine data into training and testing sets. This is essential to performing cross-validation of the ML models to identify the most effective approach to quality prediction.
3. **Building an ML model**. When the wine quality data is all set, one can start building, training, and testing a machine learning model by using different classification approaches.

Feature Importance

Having all the necessary data on hand is not enough. It's also critical to understand exactly how each of the features relates to wine quality and what role it plays in the ML modeling process.

Finding the Best Method for Wine Quality Prediction

Based on the current research, the most effective ML methods for wine quality analysis are Logistic Regression, Support Vector Machine (SVM), K-nearest Neighbors, and Random Forest. Although there are different opinions among ML researchers, we tried to collect all the results and provide the simple average for the accuracy of the ML models in predicting wine quality:

- Logistic regression-87.815
- Support Vector Machine-89.375
- K-nearest Neighbors-89
- Random Forest-90.93

From above study we finally concluded that Random Forest Classifier model is best model to predict the wine quality with 90.93% of accuracy score.

#importing all necessary libraries and Dataset 1.Pandas is a useful library in data handling. 2.Numpy library used for working with arrays. 3.Seaborn/Matplotlib are used for data visualisation purpose. 4.Sklearn – This module contains multiple libraries having pre-implemented functions to perform tasks from data preprocessing to model development and evaluation.

In [104]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
```

In [105]:

```
#loading the dataset to a pandas DataFarme
wine_data=pd.read_csv("E:\\Red_wine.csv")
wine_data
```

Out[105]:

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

1599 rows × 12 columns

First 5 rows of the dataset

In [106]:

```
wine_data.head()
```

Out[106]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9

Last 5 rows of the datset

In [107]:

```
wine_data.tail()
```

Out[107]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	al
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.52	0.58	
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.52	0.75	
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.52	0.71	
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.52	0.66	

Doing Some Exploratory Data Analysis

In [108]:

```
#count the numbers of data present in each column of the dataset
wine_data.count()
```

Out[108]:

```
fixed acidity           1599
volatile acidity        1599
citric acid             1599
residual sugar          1599
chlorides               1599
free sulfur dioxide     1599
total sulfur dioxide    1598
density                 1599
pH                      1598
sulphates               1599
alcohol                 1599
quality                 1598
dtype: int64
```

Number of rows and columns in the dataset

In [109]:

```
wine_data.shape
```

Out[109]:

```
(1599, 12)
```

In [110]:

```
wine_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1598 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1598 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1598 non-null   float64
dtypes: float64(12)
memory usage: 150.0 KB
```

In [111]:

```
wine_data.isnull().sum()
```

Out[111]:

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    1
density                 0
pH                      1
sulphates               0
alcohol                 0
quality                 1
dtype: int64
```

In [112]:

```
wine_data.isnull().sum().sum()
```

Out[112]:

```
3
```

As there is 3 missing value present in the whole datset we have to drop that rows which contains missing values because in this dataset only few missing values are present.

In [113]:

```
#fetching those rows which having missing values
wine_data[wine_data['total sulfur dioxide'].isnull() | wine_data['pH'].isnull()| wine_data[
```

Out[113]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alc
9	7.5	0.50	0.36	6.1	0.071	17.0	NaN	0.9978	3.35	0.80	
123	8.0	0.71	0.00	2.6	0.080	11.0	34.0	0.9976	3.44	0.53	
184	6.7	0.62	0.21	1.9	0.079	8.0	62.0	0.9970	NaN	0.58	

In [114]:

```
wine_data.dropna(inplace=True)
```

In [115]:

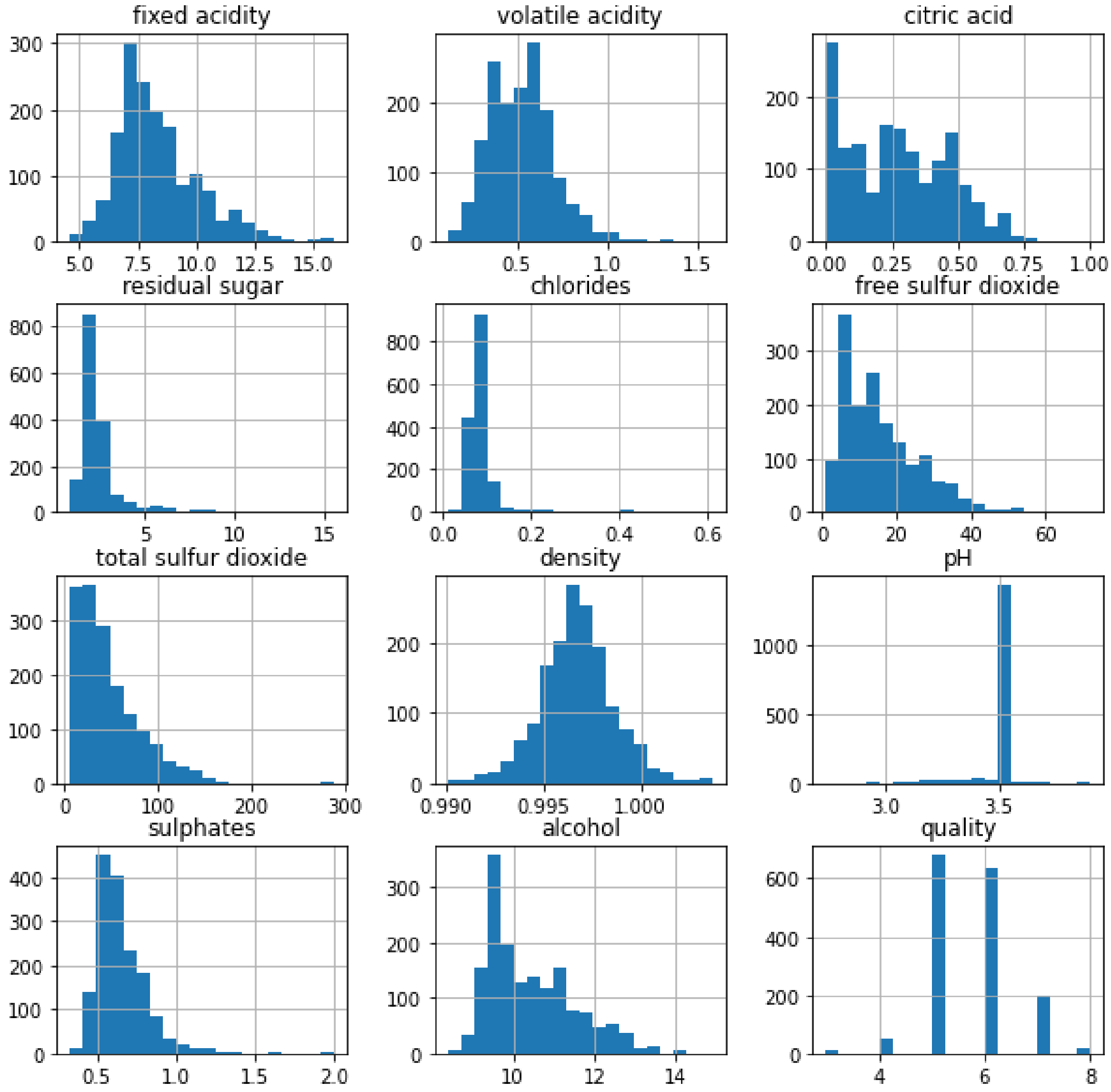
```
wine_data.isnull().sum().sum()
```

Out[115]:

```
0
```

In [116]:

```
wine_data.hist(bins=20 , figsize=(10,10))
plt.show()
```



Data Analysis and Visualization

In [117]:

```
wine_data.describe()
```

Out[117]:

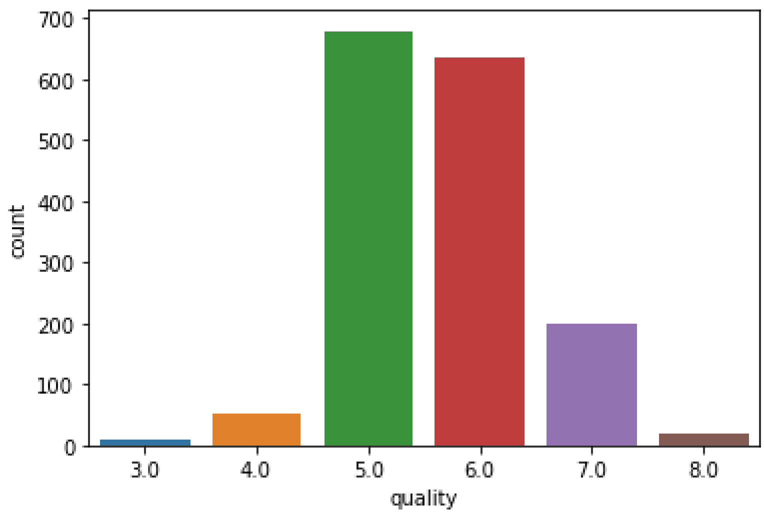
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total su diox
count	1596.000000	1596.000000	1596.000000	1596.000000	1596.000000	1596.000000	1596.000
mean	8.321366	0.527666	0.271128	2.536936	0.087487	15.882206	46.431
std	1.742121	0.179154	0.194847	1.408341	0.047107	10.467380	32.893
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000

In [118]:

```
sns.countplot(x='quality',data=wine_data)
```

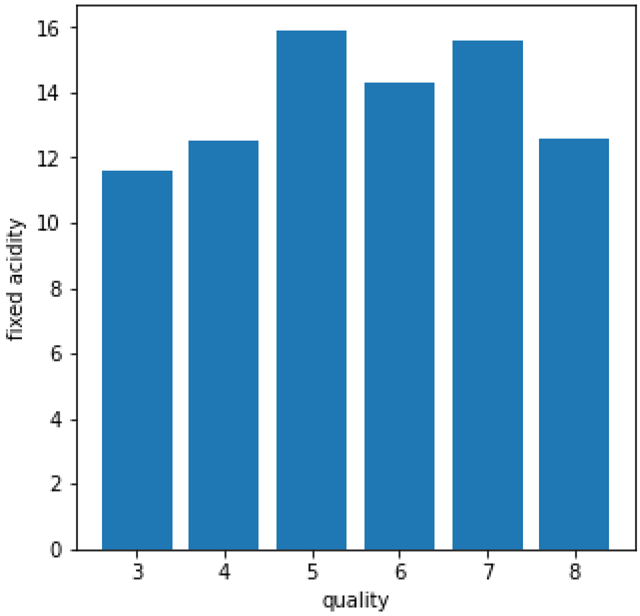
Out[118]:

<AxesSubplot:xlabel='quality', ylabel='count'>



In [119]:

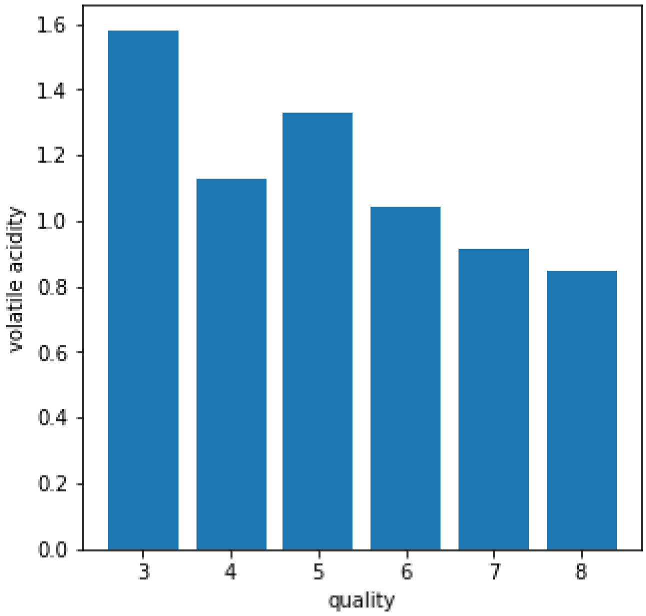
```
#fixed acidity vs quality
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['fixed acidity'])
plt.xlabel('quality')
plt.ylabel('fixed acidity')
plt.show()
```



Fixed acidity is directly proportional to the quality of the wine

In [120]:

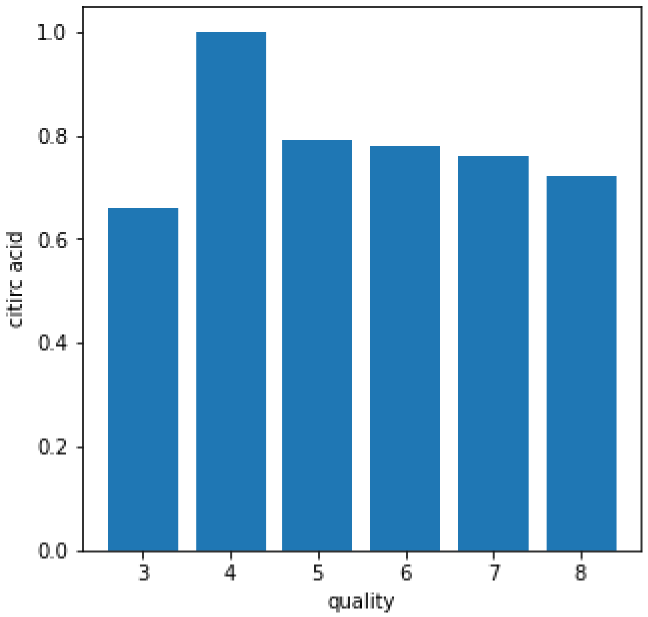
```
#volatile acidity vs quality
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['volatile acidity'])
plt.xlabel('quality')
plt.ylabel('volatile acidity')
plt.show()
```



here volatile acidity is inversely propertional to quality, means as the volatile acidity increases the quality of wine decreases.

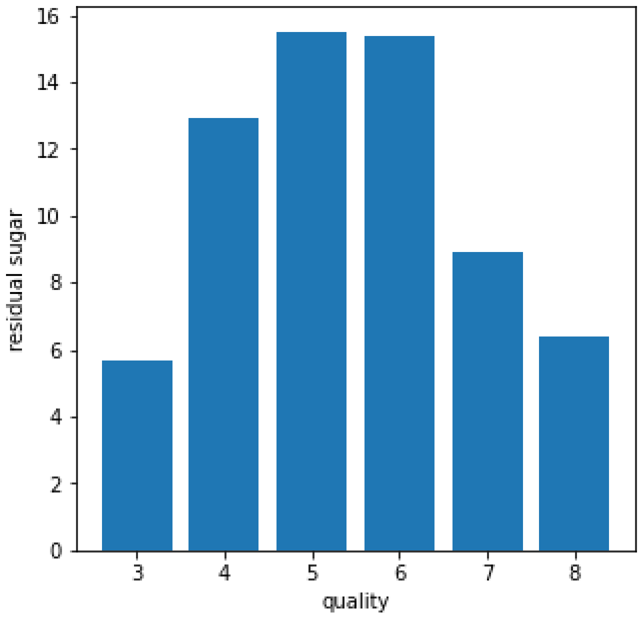
In [121]:

```
#citric acid vs quality
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['citric acid'])
plt.xlabel('quality')
plt.ylabel('citirc acid')
plt.show()
```



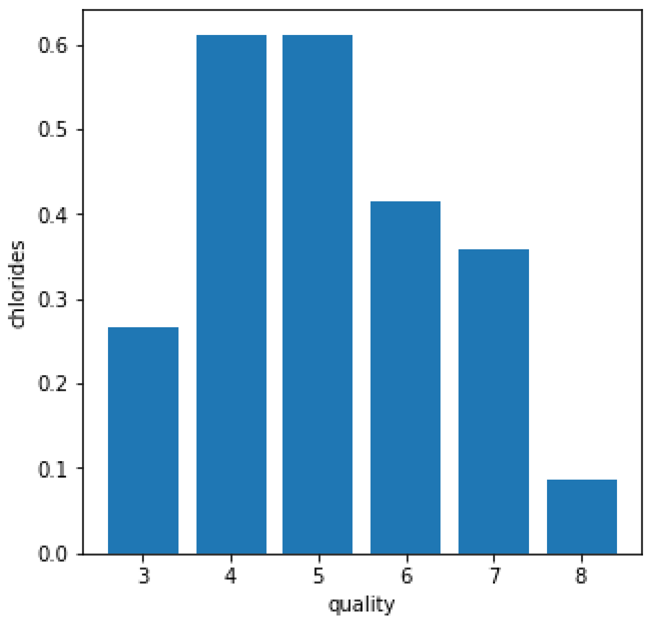
In [122]:

```
#residual sugar vs quality
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['residual sugar'])
plt.xlabel('quality')
plt.ylabel('residual sugar')
plt.show()
```



In [123]:

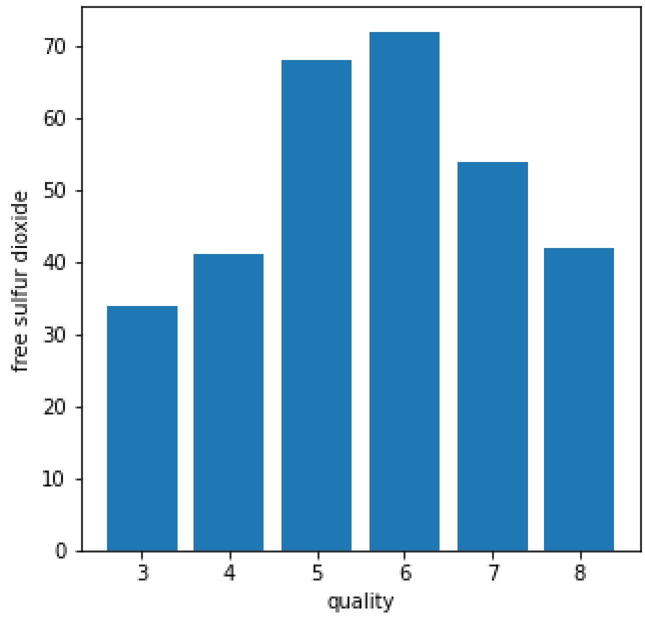
```
#chlorides vs quality
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['chlorides'])
plt.xlabel('quality')
plt.ylabel('chlorides')
plt.show()
```



in above plot we found that Less Chlorides present in Better quality of wine

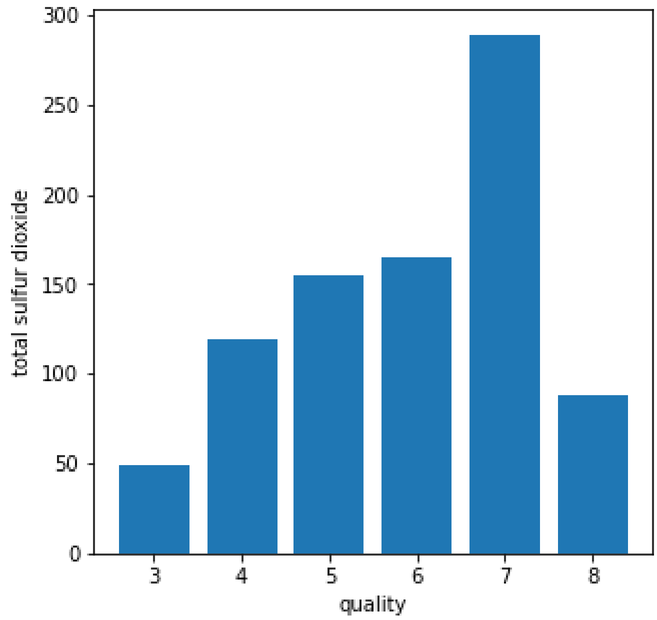
In [124]:

```
#free sulfer diaoxide vs quality
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['free sulfur dioxide'])
plt.xlabel('quality')
plt.ylabel('free sulfur dioxide')
plt.show()
```



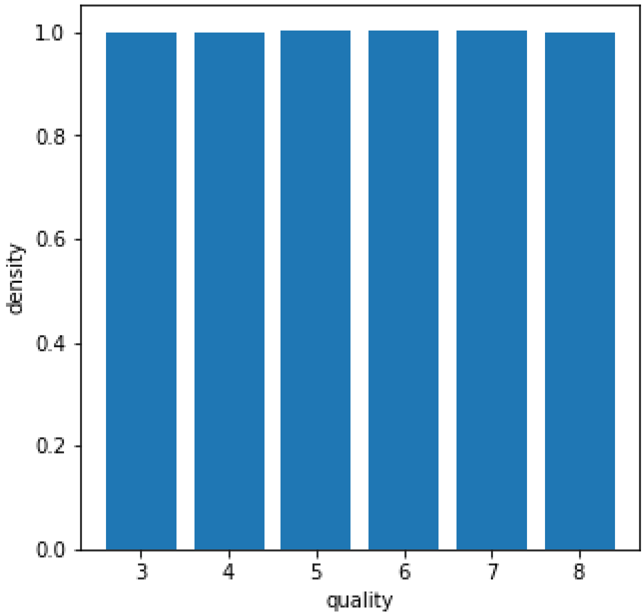
In [125]:

```
#total sulfur dioxide vs quality
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['total sulfur dioxide'])
plt.xlabel('quality')
plt.ylabel('total sulfur dioxide')
plt.show()
```



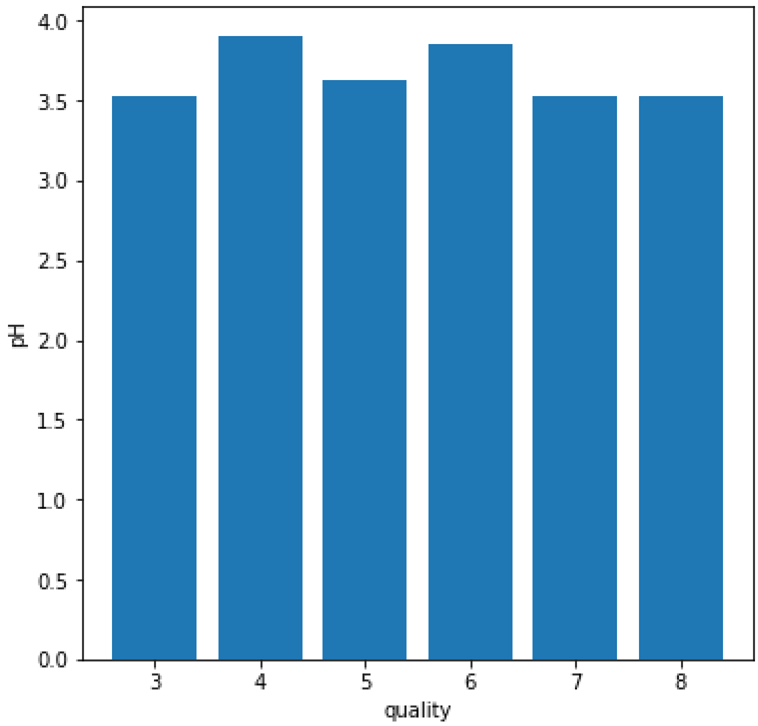
In [126]:

```
#density vs quality
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['density'])
plt.xlabel('quality')
plt.ylabel('density')
plt.show()
```



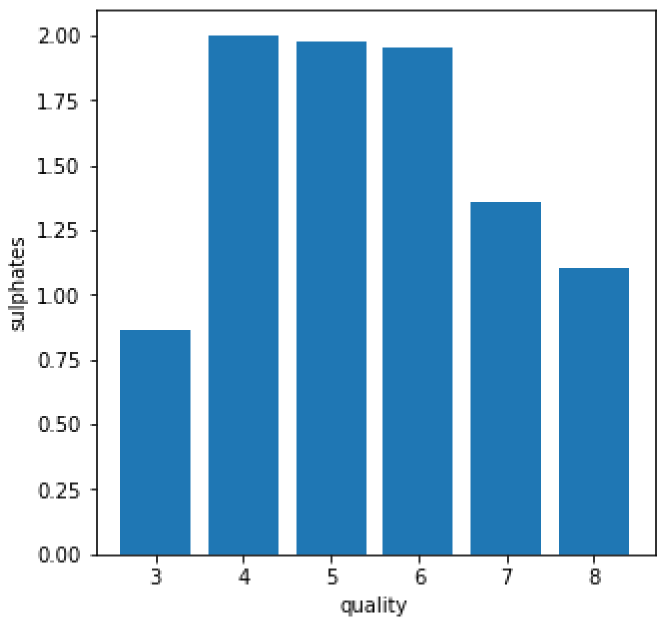
In [127]:

```
#pH vs quality
plot=plt.figure(figsize=(6,6))
plt.bar(wine_data['quality'],wine_data['pH'])
plt.xlabel('quality')
plt.ylabel('pH')
plt.show()
```



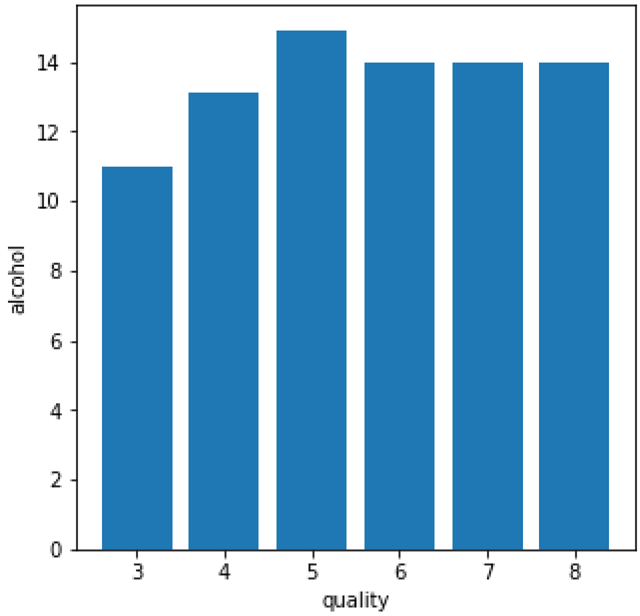
In [128]:

```
#sulphate vs quality
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['sulphates'])
plt.xlabel('quality')
plt.ylabel('sulphates')
plt.show()
```



In [129]:

```
plot=plt.figure(figsize=(5,5))
plt.bar(wine_data['quality'],wine_data['alcohol'])
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.show()
```



From above we found that as alcohol containing in wine is increases we got better quality of wine,means higher the quantity of alcohol having higher quality of wine.

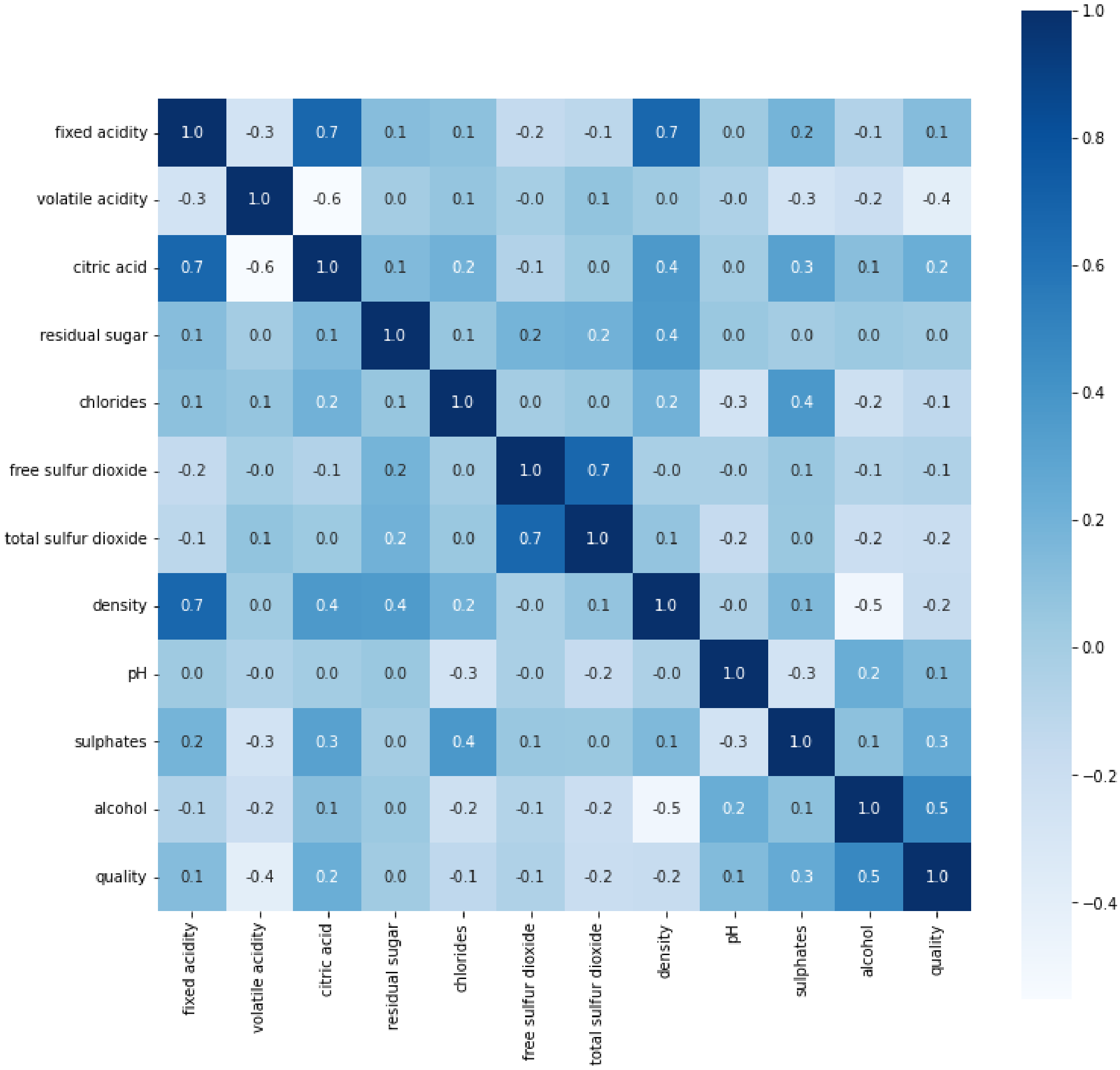
Correlation

In [130]:

```
correlation=wine_data.corr()
```

In [131]:

```
#constructing a heatmap to undertsnad the correlation between the columns
plt.figure(figsize=(12, 12))
sns.heatmap(correlation, annot=True,square=True ,fmt='0.1f',cbar=True,cmap='Blues')
plt.show()
```



Label Binarization of quality

In [132]:

```
wine_data['quality'].unique()
```

Out[132]:

```
array([5., 6., 7., 4., 8., 3.])
```

In [133]:

```
wine_data['quality'].value_counts()
```

Out[133]:

```
5.0    679
6.0    637
7.0    199
4.0     53
8.0     18
3.0     10
Name: quality, dtype: int64
```

In [134]:

```
wine_data['quality']=[1 if x>=7 else 0 for x in wine_data['quality']]
```

In [135]:

```
wine_data['quality'].unique()
```

Out[135]:

```
array([0, 1], dtype=int64)
```

In [136]:

```
wine_data['quality'].value_counts()
```

Out[136]:

```
0    1379
1     217
Name: quality, dtype: int64
```

from this we conclude that there are few observation good quality of wines and many observation are available for wrost quality of wine .

In [138]:

```
#Separate the data and Label
x=wine_data.drop('quality',axis=1)
y=wine_data['quality']
```

In [139]:

```
x
```

Out[139]:

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

1596 rows × 11 columns

In [140]:

```
y
```

Out[140]:

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

Train and Test split

In [141]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [142]:

```
x_train.shape
```

Out[142]:

```
(1276, 11)
```

In [143]:

```
x_test.shape
```

Out[143]:

```
(320, 11)
```

In [144]:

```
y_train.shape
```

Out[144]:

```
(1276,)
```

In [145]:

```
y_test.shape
```

Out[145]:

```
(320,)
```

Feature Scaling: it is require for to put our features into same scale.we use here standard scaler

In [146]:

```
from sklearn.preprocessing import StandardScaler
```

In [147]:

```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
```

In [148]:

```
x_train
```

Out[148]:

```
array([[-1.27525952, -1.17673516, -0.10937401, ...,  0.26790084,
         1.78453028,  0.16600727],
       [ 0.18116246, -1.11993923,  0.66111495, ...,  0.26790084,
        -1.03418363, -0.95377392],
       [ 0.99675877, -1.34712296,  0.66111495, ...,  0.26790084,
         1.43938164, -0.67382862],
       ...,
       [-0.63443385,  0.52714281, -1.08532668, ...,  0.26790084,
        -0.74655976, -0.67382862],
       [-0.86746137,  0.64073468, -1.23942447, ...,  0.26790084,
        -0.45893589, -0.02062293],
       [ 1.28804316, -0.5519799 ,  0.4042853 , ...,  0.26790084,
         0.17383662,  0.07269217]])
```

In [149]:

```
x_test
```

Out[149]:

```
array([[ 6.27433587e-01, -6.27099805e-01,  5.09888328e-01, ...,
         2.56940727e-01, -6.08266259e-01,  6.46477640e-02],
       [-1.06436885e+00,  2.64425855e-01, -1.38558351e+00, ...,
         2.56940727e-01, -2.10895991e-01,  1.02461456e+00],
       [ 8.16908638e-02, -5.02302601e-02,  5.61117296e-01, ...,
         2.56940727e-01, -8.73179772e-01, -9.91315712e-01],
       ...,
       [ 2.75583021e+00,  2.21242581e-03,  2.66150501e+00, ...,
         2.56940727e-01,  7.82529679e-01,  2.46456476e+00],
       [ 5.72859315e-01, -7.31985177e-01,  7.14804202e-01, ...,
         2.56940727e-01,  1.04744319e+00,  4.48634483e-01],
       [-1.17351740e+00,  7.88852715e-01, -1.33435454e+00, ...,
         2.56940727e-01,  5.40175208e-02,  8.32621201e-01]])
```

Building Machine Learning Model on Our Dataset

Logistic Regression: As it is a classification problem,here we have to find out wine quality is good or bad

In [150]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [151]:

```
model = LogisticRegression()
model.fit(x_train,y_train)
```

Out[151]:

```
LogisticRegression()
```

In [152]:

```
y_pred=model.predict(x_test)
```

In [153]:

```
accuracy_score(y_test,y_pred)
```

Out[153]:

```
0.878125
```

SVC:Support Vector Classifier

In [154]:

```
from sklearn import svm
```

In [155]:

```
svm = svm.SVC()
```

In [156]:

```
svm.fit(x_train,y_train)
```

Out[156]:

```
SVC()
```

In [157]:

```
y_pred2=svm.predict(x_test)
```

In [158]:

```
accuracy_score(y_test,y_pred2)
```

Out[158]:

```
0.89375
```

KNeighbors Classifier

In [159]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [160]:

```
knn=KNeighborsClassifier()
```

In [161]:

```
knn.fit(x_train,y_train)
```

Out[161]:

```
KNeighborsClassifier()
```

In [162]:

```
y_pred3=knn.predict(x_test)
```

In [163]:

```
accuracy_score(y_test,y_pred3)
```

Out[163]:

```
0.89375
```

Model Training: randomForestClassifier

In [175]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [176]:

```
model=RandomForestClassifier()
model.fit(x_train,y_train)
```

Out[176]:

```
RandomForestClassifier()
```

In [177]:

```
y_pred4=model.predict(x_test)
```

In [178]:

```
accuracy_score(y_test,y_pred4)
```

Out[178]:

```
0.909375
```

In [ ]:

```

```

In [179]:

```
import pandas as pd
```

In [180]:

```
final_data=pd.DataFrame({'Models':['LR','SVC','KNN','RF'],
                        'ACC':[accuracy_score(y_test,y_pred)*100,
                               accuracy_score(y_test,y_pred2)*100,
                               accuracy_score(y_test,y_pred3)*100,
                               accuracy_score(y_test,y_pred4)*100]})
```

In [181]:

```
final_data
```

Out[181]:

	Models	ACC
0	LR	87.8125
1	SVC	89.3750
2	KNN	89.3750
3	RF	90.9375

In [182]:

```
import seaborn as sns
```

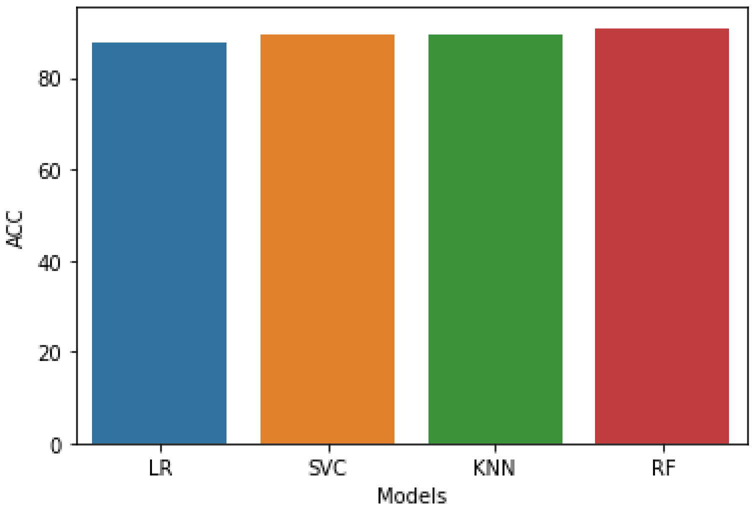
In [183]:

```
sns.barplot(final_data['Models'],final_data['ACC'])
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other argu
ments without an explicit keyword will result in an error or misinterpretati
on.
  warnings.warn(
```

Out[183]:

```
<AxesSubplot:xlabel='Models', ylabel='ACC'>
```



From above study we finally concluded that RandomForest Classifier model is best model to predict the wine quality with 90.93% of accuracy_score.