Wine Quality prediction Analysis:

Tools used for Analzing Data : Python, Machine Leaning, Excel.

About Dataset :

- The primary goal of this project is to build a predictive model that can accurately estimate wine quality based on its chemical composition. The dataset used in this project is the Wine Quality dataset. It contains important chemical features such as: *Fixed Acidity, Volatile Acidity, Citric Acid, Residual Sugar, Chlorides, Free Sulfur Dioxide, Total Sulfur Dioxide, Density pH, Sulphates, Alcohol, Quality (target variable)

By analyzing the data, we aim to:

- Understand the factors that influence wine quality.
- Develop robust machine learning models.
- Gain experience with a typical data science workflow.

Name of the Dataset:

In this project, we:

1. Project follows a well-defined workflow for building and deploying the model:
2. Data Loading and Exploration : Load data using Pandas, explore data types, missing values, and summary statistics.
3. Model Selection & Training : Select ML models (Logistic Regression, ElasticNet, etc.) and train using the data.
4. Model Evaluation : Evaluate the models using MSE, RMSE, and R² Score for performance analysis.
5. MLOps : Implement practices such as Experiment Tracking with MLFlow and Version Control with Dagshub.

Importing Libraries

In [45]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from warnings import filterwarnings
filterwarnings(action='ignore')
```

Loading Dataset

In [46]:
```
wine = pd.read_csv('Wine Quality Dataset.csv')
print("Successfully Imported Data!")
wine.head()
```

```
Successfully Imported Data!
```

Out[46]:

	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

In [47]:
```
print(wine.shape)
```

```
(4898, 12)
```

Description

In [48]:
```
wine.describe(include='all')
```

Out[48]:

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

Finding Null Values

In [49]:
```
print(wine.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
```

Calulate the correlation between columns in a data set

In [50]:
```
wine.corr()
```

Out[50]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.022697	0.289181	0.089021	0.023086	-0.049396	0.091070	0.265331	-0.425858	-0.017143	-0.120881	-0.113663
volatile acidity	-0.022697	1.000000	-0.149472	0.064286	0.070512	-0.097012	0.089261	0.027114	-0.031915	-0.035728	0.067718	-0.194723
citric acid	0.289181	-0.149472	1.000000	0.094212	0.114364	0.094077	0.121131	0.149503	-0.163748	0.062331	-0.075729	-0.009209
residual sugar	0.089021	0.064286	0.094212	1.000000	0.088685	0.299098	0.401439	0.838966	-0.194133	-0.026664	-0.450631	-0.097577
chlorides	0.023086	0.070512	0.114364	0.088685	1.000000	0.101392	0.198910	0.257211	-0.090439	0.016763	-0.360189	-0.209934
free sulfur dioxide	-0.049396	-0.097012	0.094077	0.299098	0.101392	1.000000	0.615501	0.294210	-0.000618	0.059217	-0.250104	0.008158
total sulfur dioxide	0.091070	0.089261	0.121131	0.401439	0.198910	0.615501	1.000000	0.529881	0.002321	0.134562	-0.448892	-0.174737
density	0.265331	0.027114	0.149503	0.838966	0.257211	0.294210	0.529881	1.000000	-0.093591	0.074493	-0.780138	-0.307123
pH	-0.425858	-0.031915	-0.163748	-0.194133	-0.090439	-0.000618	0.002321	-0.093591	1.000000	0.155951	0.121432	0.099427
sulphates	-0.017143	-0.035728	0.062331	-0.026664	0.016763	0.059217	0.134562	0.074493	0.155951	1.000000	-0.017433	0.053678
alcohol	-0.120881	0.067718	-0.075729	-0.450631	-0.360189	-0.250104	-0.448892	-0.780138	0.121432	-0.017433	1.000000	0.435575
quality	-0.113663	-0.194723	-0.009209	-0.097577	-0.209934	0.008158	-0.174737	-0.307123	0.099427	0.053678	0.435575	1.000000

Get the Average of groupby

In [51]:
```
wine.groupby('quality').mean()
```

Out[51]:

quality	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
3	7.600000	0.333250	0.336000	6.392500	0.054300	53.325000	170.600000	0.994884	3.187500	0.474500	10.345000
4	7.129448	0.381227	0.304233	4.628221	0.050098	23.358896	125.279141	0.994277	3.182883	0.476135	10.152454
5	6.933974	0.302011	0.337653	7.334969	0.051546	36.432052	150.904598	0.995263	3.168833	0.482203	9.808840
6	6.837671	0.260564	0.338025	6.441606	0.045217	35.650591	137.047316	0.993961	3.188599	0.491106	10.575372
7	6.734716	0.262767	0.325625	5.186477	0.038191	34.125568	125.114773	0.992452	3.213898	0.503102	11.367936
8	6.657143	0.277400	0.326514	5.671429	0.038314	36.720000	126.165714	0.992236	3.218686	0.486229	11.636000
9	7.420000	0.298000	0.386000	4.120000	0.027400	33.400000	116.000000	0.991460	3.308000	0.466000	12.180000

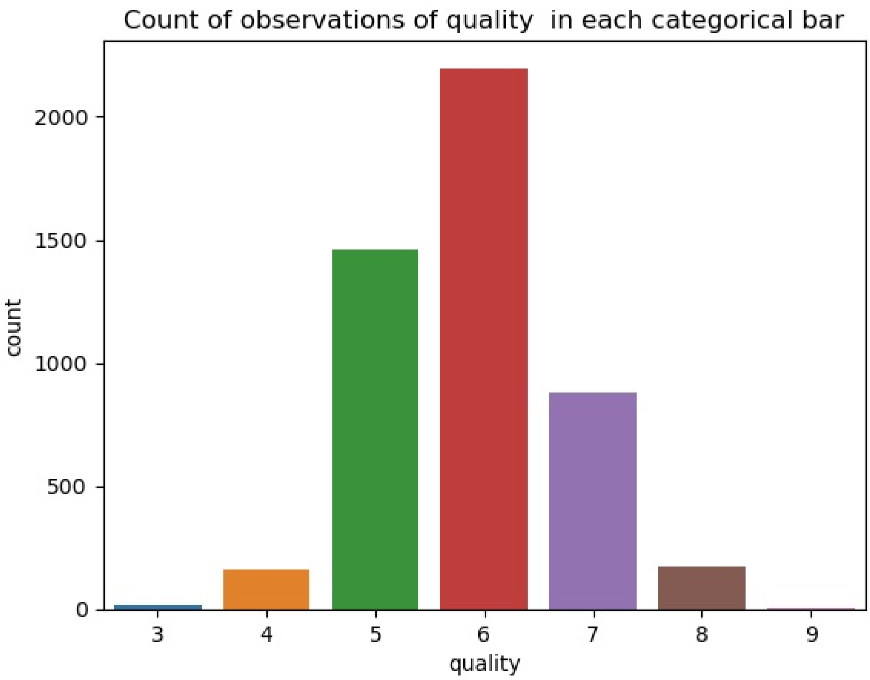
Data Analysis

Countplot:

- It will show the counts of observations in each categorical bar

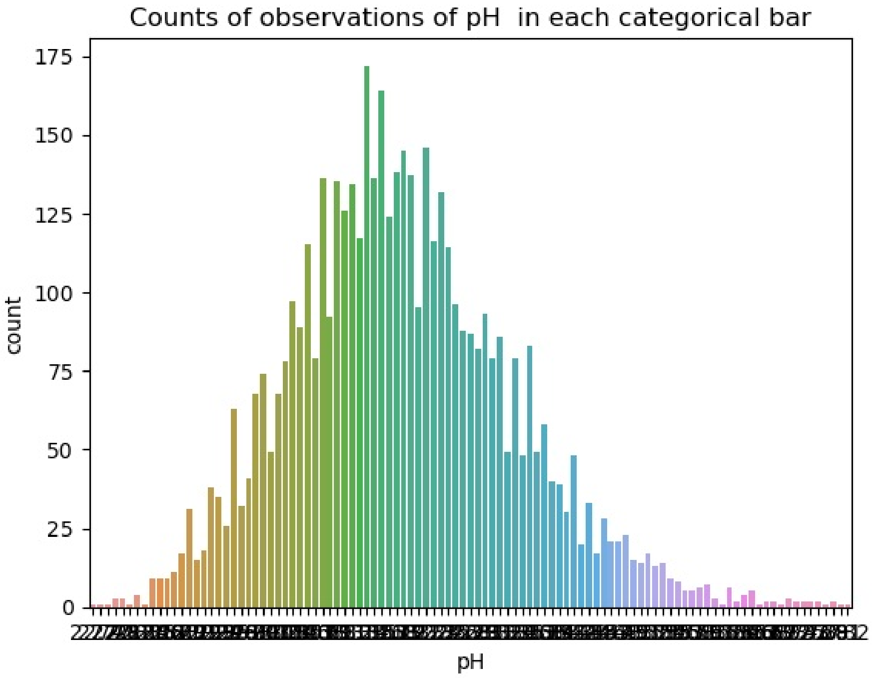
In [52]:

```
sns.countplot(wine['quality'])
plt.title("Count of observations of quality  in each categorical bar")
plt.show()
```

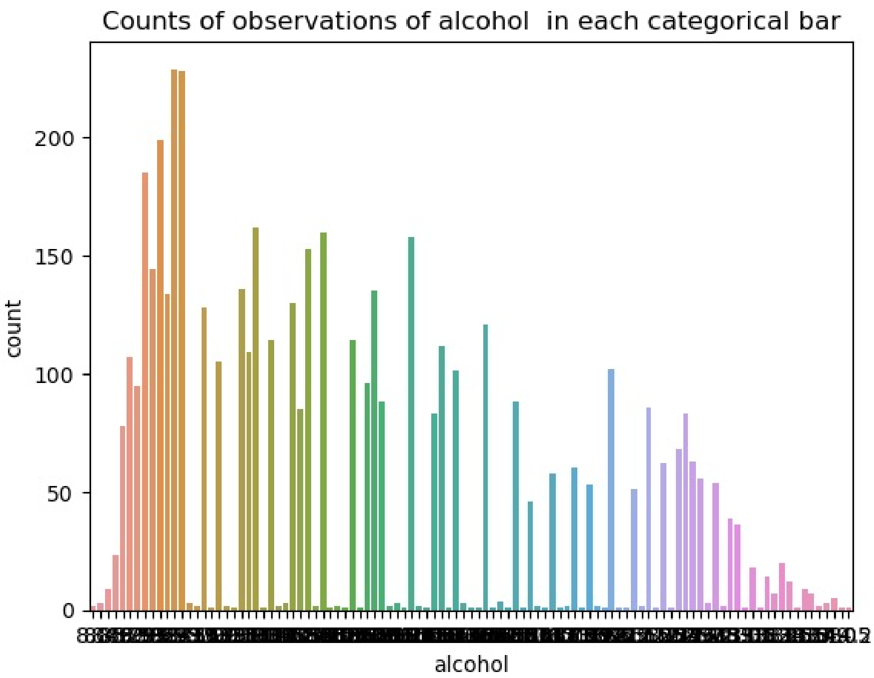


In [53]:

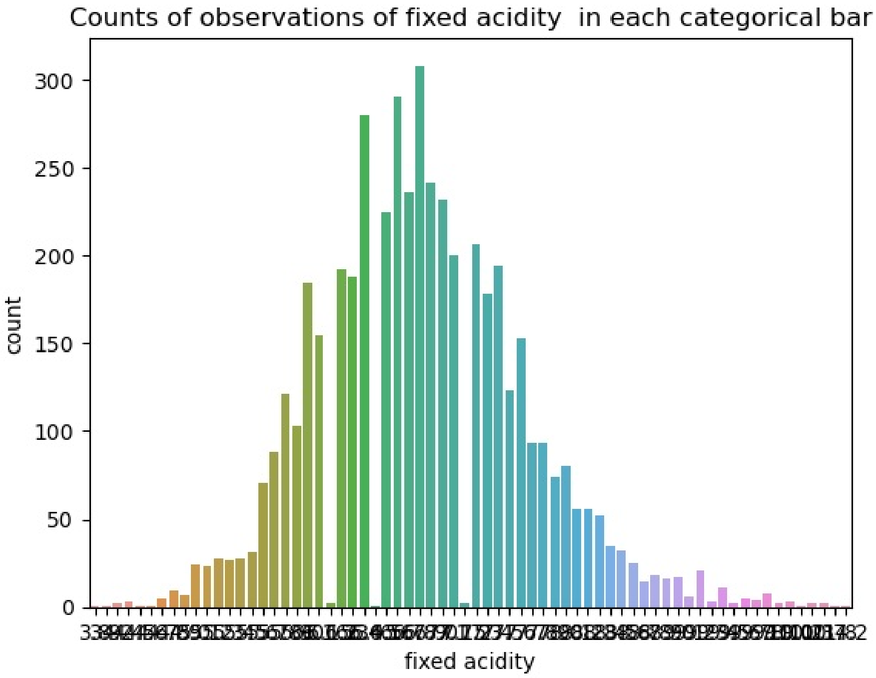
```
sns.countplot(wine['pH'])
plt.title("Counts of observations of pH  in each categorical bar")
plt.show()
```



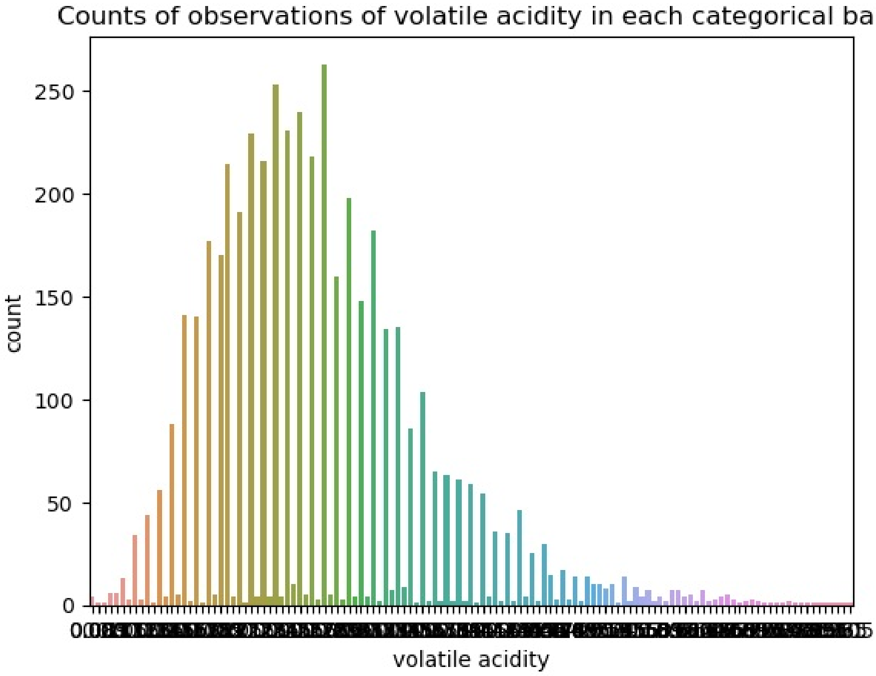
In [54]:
```
sns.countplot(wine['alcohol'])
plt.title("Counts of observations of alcohol  in each categorical bar")
plt.show()
```



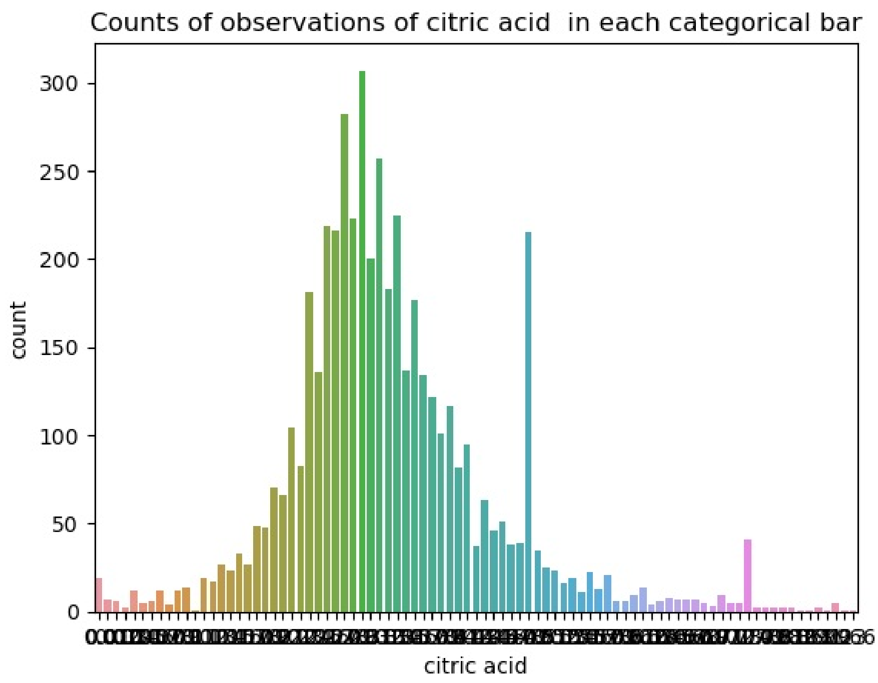
In [55]:
```
sns.countplot(wine['fixed acidity'])
plt.title("Counts of observations of fixed acidity  in each categorical bar")
plt.show()
```



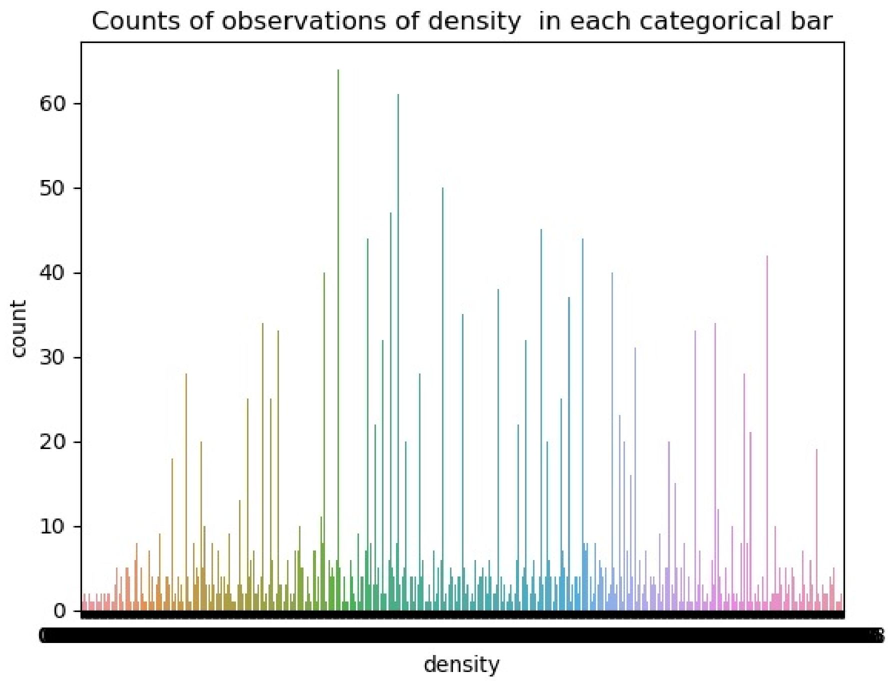
In [56]:
```
sns.countplot(wine['volatile acidity'])
plt.title("Counts of observations of volatile acidity in each categorical bar")
plt.show()
```



In [57]:
```
sns.countplot(wine['citric acid'])
plt.title("Counts of observations of citric acid  in each categorical bar")
plt.show()
```



In [58]:
```
sns.countplot(wine['density'])
plt.title("Counts of observations of density  in each categorical bar")
plt.show()
```

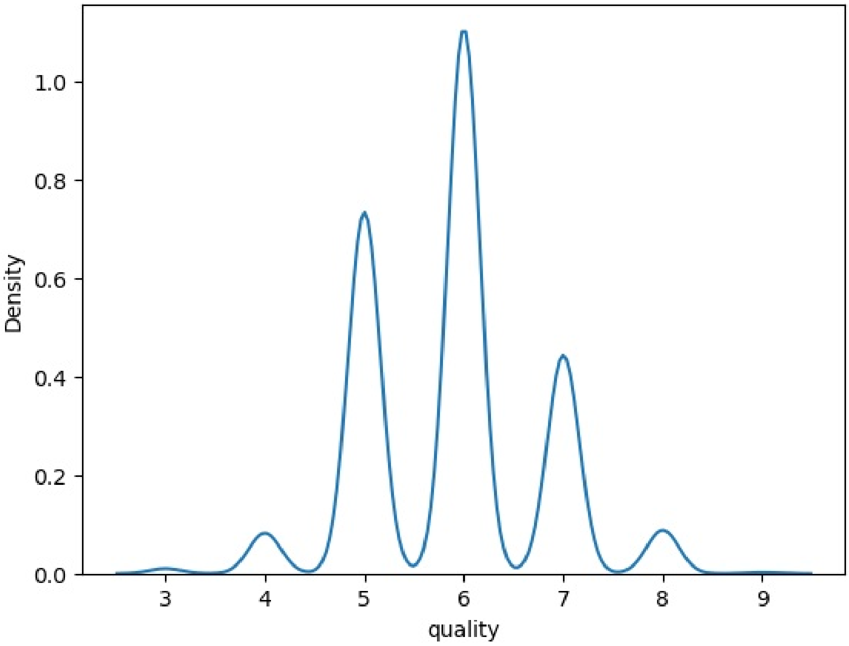


KDE Plot:

- kernel density estimate plot is a non parametric way to estimate the probability density function of a continuous variable,providing insights into data distribution,shape and central tendency.

In [59]:
```
sns.kdeplot(wine.query('quality>2').quality)
```

Out[59]:
```
<AxesSubplot:xlabel='quality', ylabel='Density'>
```

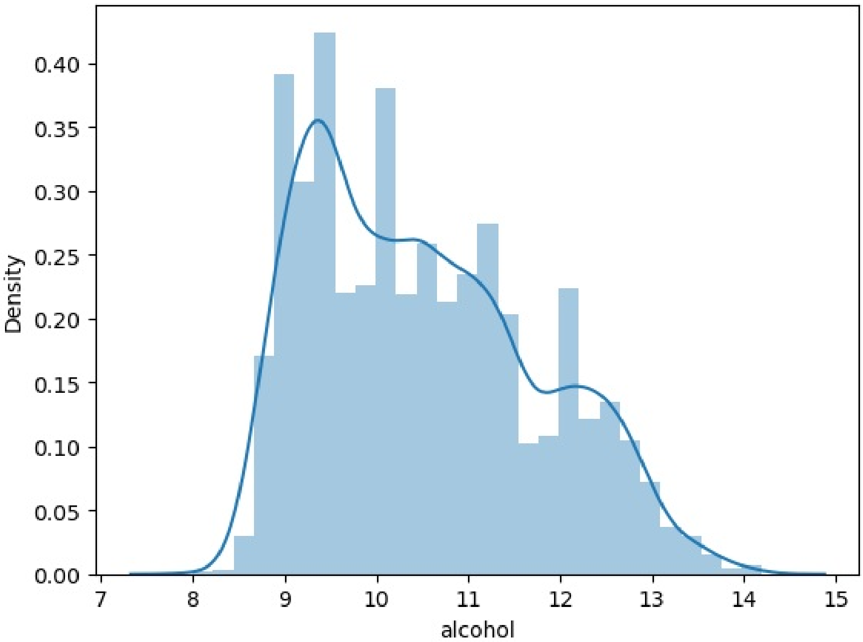


Distplot:

- A Distplot or distribution plot,depicts the variation in the data distribution.It is a visualization of the distribution of data using a histogram and a line.

In [60]:
```
sns.distplot(wine['alcohol'])
```

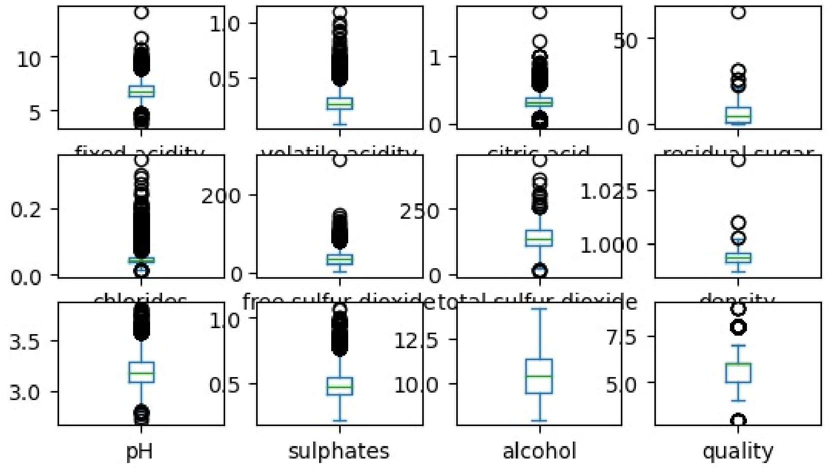
Out[60]:
```
<AxesSubplot:xlabel='alcohol', ylabel='Density'>
```



AxesSubplot:

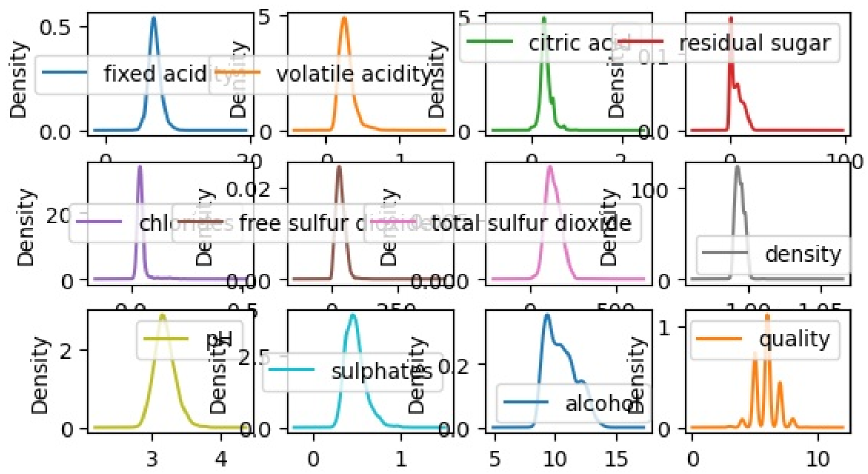
In [61]:
```
wine.plot(kind='box',subplots=True,layout=(4,4),sharex=False)
```

```
Out[61]: fixed acidity          AxesSubplot(0.125,0.712609;0.168478x0.167391)
volatile acidity       AxesSubplot(0.327174,0.712609;0.168478x0.167391)
citric acid            AxesSubplot(0.529348,0.712609;0.168478x0.167391)
residual sugar         AxesSubplot(0.731522,0.712609;0.168478x0.167391)
chlorides                 AxesSubplot(0.125,0.511739;0.168478x0.167391)
free sulfur dioxide    AxesSubplot(0.327174,0.511739;0.168478x0.167391)
total sulfur dioxide   AxesSubplot(0.529348,0.511739;0.168478x0.167391)
density                AxesSubplot(0.731522,0.511739;0.168478x0.167391)
pH                         AxesSubplot(0.125,0.31087;0.168478x0.167391)
sulphates               AxesSubplot(0.327174,0.31087;0.168478x0.167391)
alcohol                 AxesSubplot(0.529348,0.31087;0.168478x0.167391)
quality                 AxesSubplot(0.731522,0.31087;0.168478x0.167391)
dtype: object
```




```
In [62]: wine.plot(kind='density',subplots=True,layout=(4,4),sharex=False)
```

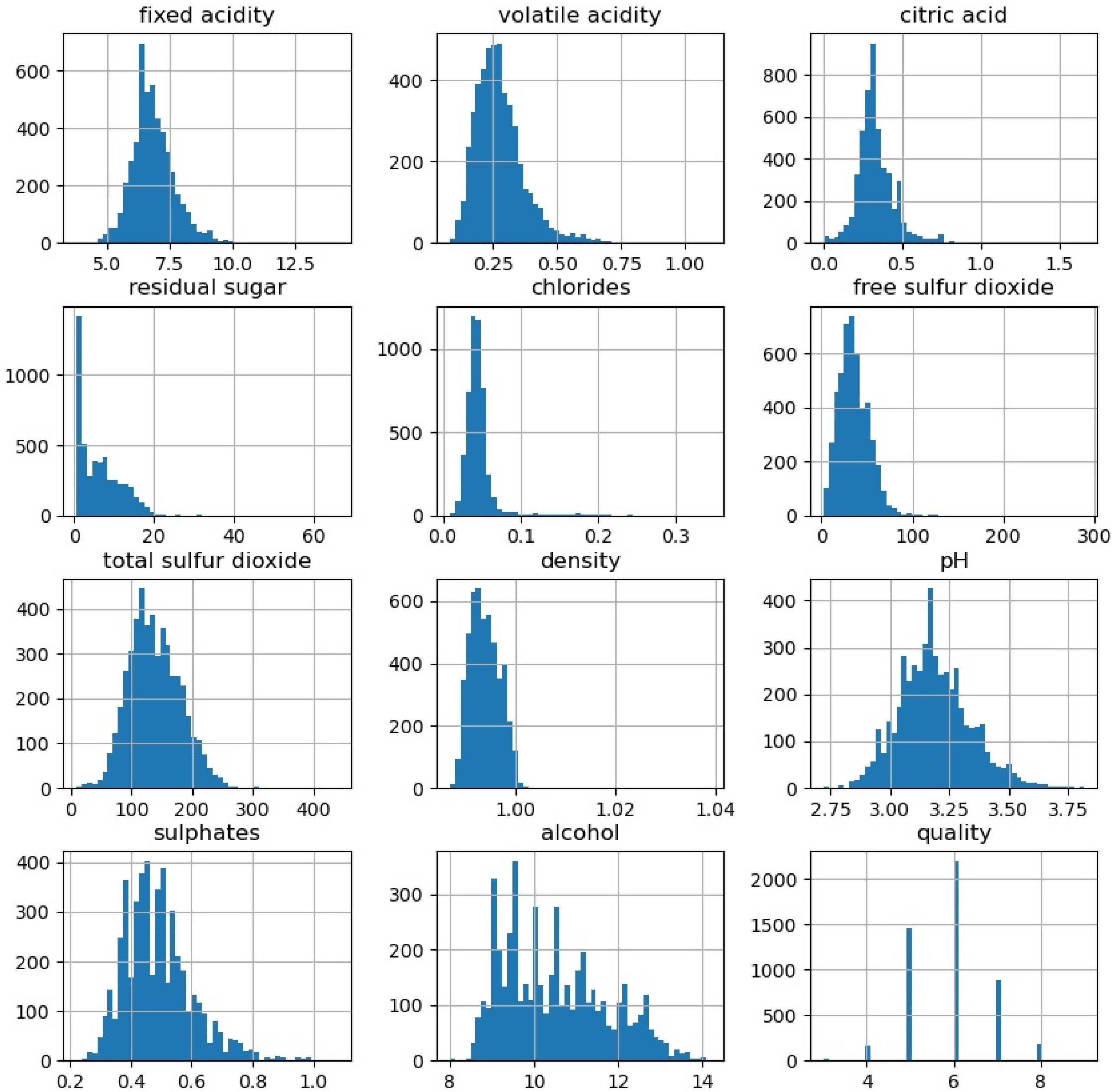
```
Out[62]: array([[<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
         <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
        [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
         <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
        [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
         <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
        [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
         <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>]],
       dtype=object)
```



Histogram:

- It is a graph showing the number of observations within each given interval.
- A histogram is used to represent data provided in the form of some groups.It is an accurate method for the graphical representation of numerical data distribution.

```
In [63]: wine.hist(figsize=(10,10),bins=50)
plt.show()
```

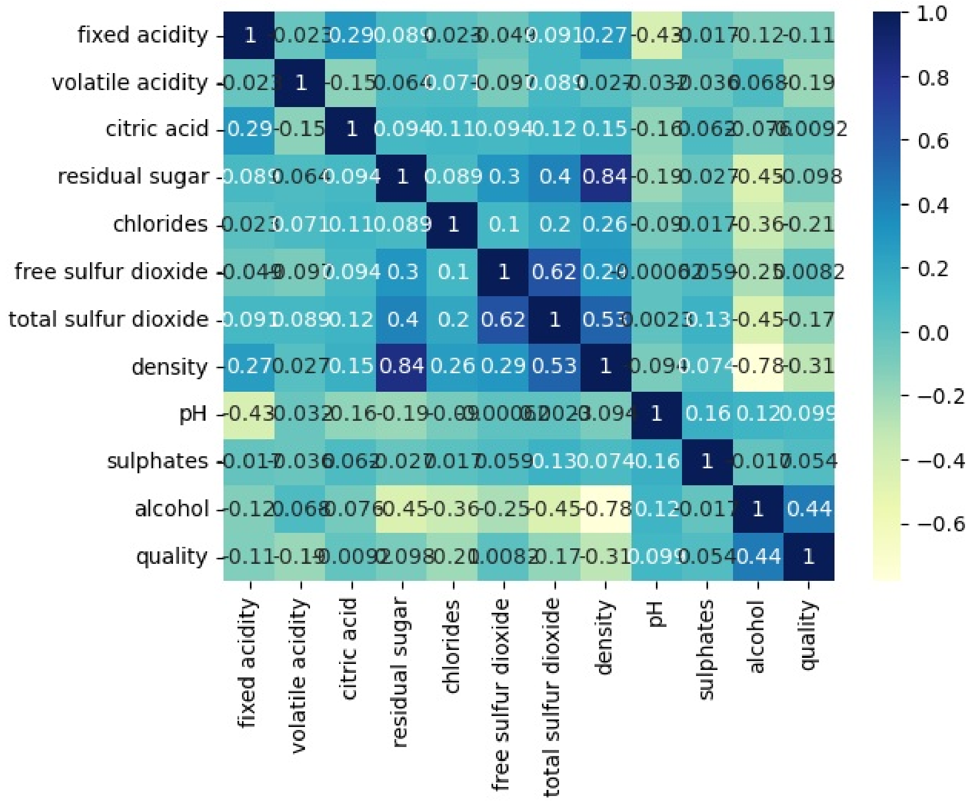


Heatmap for expressing correlation:

A correlation heatmap is a graphical tool that displays the correlation between multiple variables as a color coded matrix.

In [64]:
```
corr=wine.corr()
sns.heatmap(corr,cmap='YlGnBu',annot=True)
```

Out[64]: <AxesSubplot:>

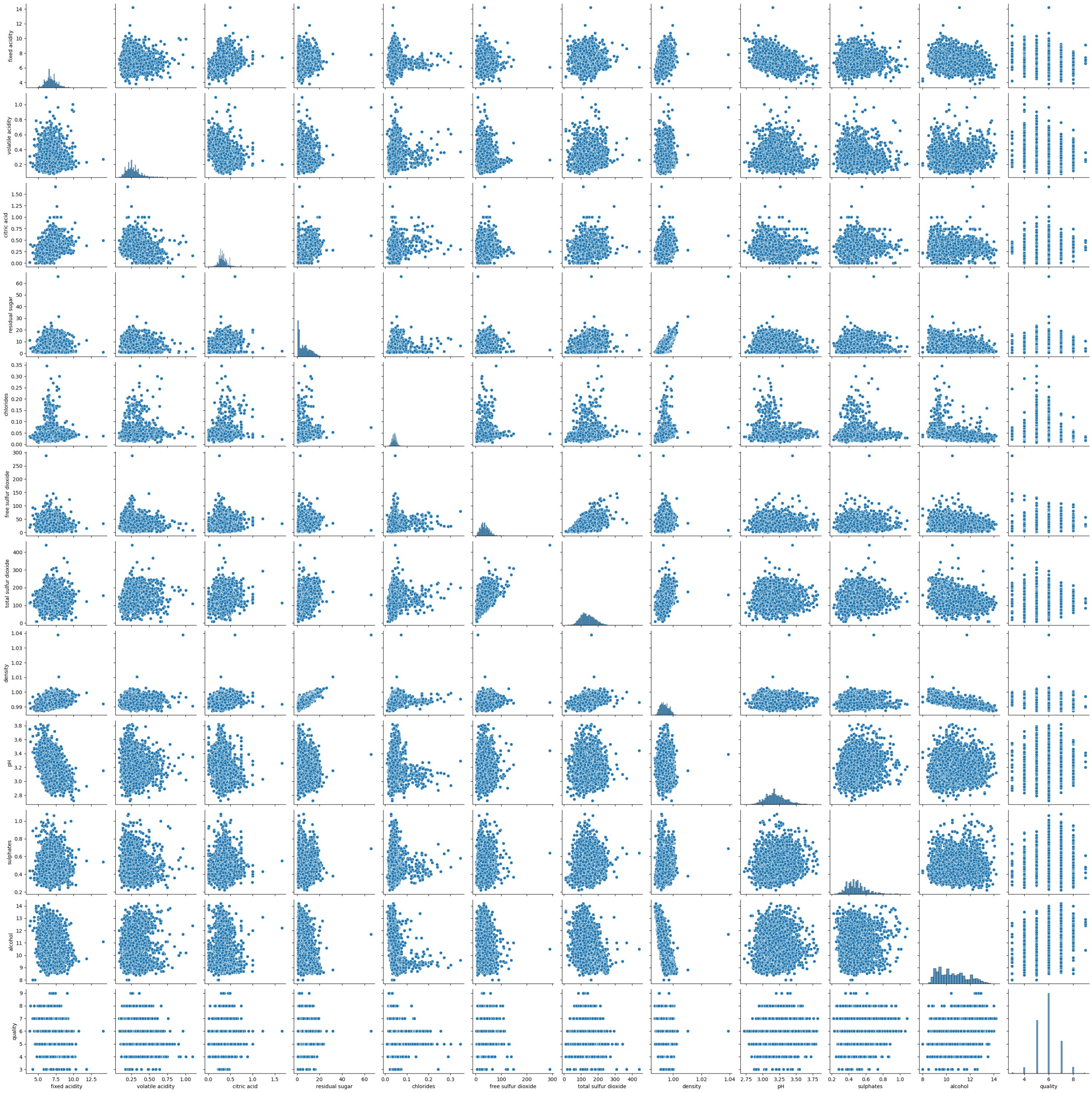


Pair Plot:

To plot multiple pairwise bivariate distributions in a dataset.

In [65]:
```
sns.pairplot(wine)
```

Out[65]:
```
<seaborn.axisgrid.PairGrid at 0x2557a943340>
```



In [66]:
```
wine.head(1)
```

Out[66]:

	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	1.001	3.0	0.45	8.8	6

Feature Selection

In [67]:
```
# Create Classification version of target variable
wine['goodquality']=[1 if x>=7 else 0 for x in wine['quality']]    #separate feature variables and target varia
```

```
x=wine.drop(['quality','goodquality'],axis=1)
y=wine['goodquality']
```

In [68]:
```
#See proportion of good vs bad wines
wine['goodquality'].value_counts()
```

Out[68]:
```
0    3838
1    1060
Name: goodquality, dtype: int64
```

In [69]:
```
x
```

Out[69]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	8.8
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	9.5
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	10.1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9
...	...	...	...	...	...	...	...	...	...	...	...
4893	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2
4894	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6
4895	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4
4896	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8
4897	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8

4898 rows × 11 columns

In [70]:
```
print(y)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
4893    0
4894    0
4895    0
4896    1
4897    0
Name: goodquality, Length: 4898, dtype: int64
```

Feature Importance

In [71]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
from sklearn.ensemble import ExtraTreesClassifier
Classifiern=ExtraTreesClassifier()
Classifiern.fit(x,y)
score = Classifiern.feature_importances_
print(score)
```

```
[0.0718218  0.08550306 0.07340215 0.08315334 0.08337822 0.07957521
 0.07911649 0.10078579 0.08352738 0.07912185 0.18061472]
```

Splitting Dataset

In [72]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=7)
```

Logistic Regression

In [73]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)

from sklearn.metrics import accuracy_score,confusion_matrix
print("Accuracy Score:",accuracy_score(y_test,y_pred))
```

```
Accuracy Score: 0.7979591836734694
```

In [74]:
```
confusion_mat=confusion_matrix(y_test,y_pred)
print(confusion_mat)
```

```
[[1102   59]
 [ 238   71]]
```

Using KNN:

In [75]:
```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=3)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,y_pred))
```

```
Accuracy Score: 0.7789115646258503
```

Using SVC

In [76]:
```
from sklearn.svm import SVC
model=SVC()
model.fit(x_train,y_train)
pred_y=model.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,pred_y))
```

```
Accuracy Score: 0.789795918367347
```

Using Decision Tree:

In [77]:
```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(criterion='entropy',random_state=7)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy score:",accuracy_score(y_test,y_pred))
```

```
Accuracy score: 0.8414965986394558
```

Using GaussianNB:

In [78]:
```
from sklearn.naive_bayes import GaussianNB
model13=GaussianNB()
model13.fit(x_train,y_train)
y_pred3=model13.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,y_pred3))
```

```
Accuracy Score: 0.7204081632653061
```

Using Random Forest:

In [81]:
```
from sklearn.ensemble import RandomForestClassifier
model2=RandomForestClassifier(random_state=1)
model2.fit(x_train,y_train)
y_pred2=model2.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,y_pred2))
```

```
Accuracy Score: 0.8768707482993198
```

In [80]:
```
results = pd.DataFrame({
    'Model': ['Logistic Regression','KNN', 'SVC','Decision Tree' ,'GaussianNB','Random Forest'],
    'Score': [0.797,0.778,0.789,0.841,0.720,0.876,]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df
```

Out[80]:

	Model
Score	
0.876	Random Forest
0.841	Decision Tree
0.797	Logistic Regression
0.789	SVC
0.778	KNN
0.720	GaussianNB

Hence I will use Random Forest algorithms for training my model.

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js