AIM: Implementation of Naïve Bayes Classifier

THEORY:

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

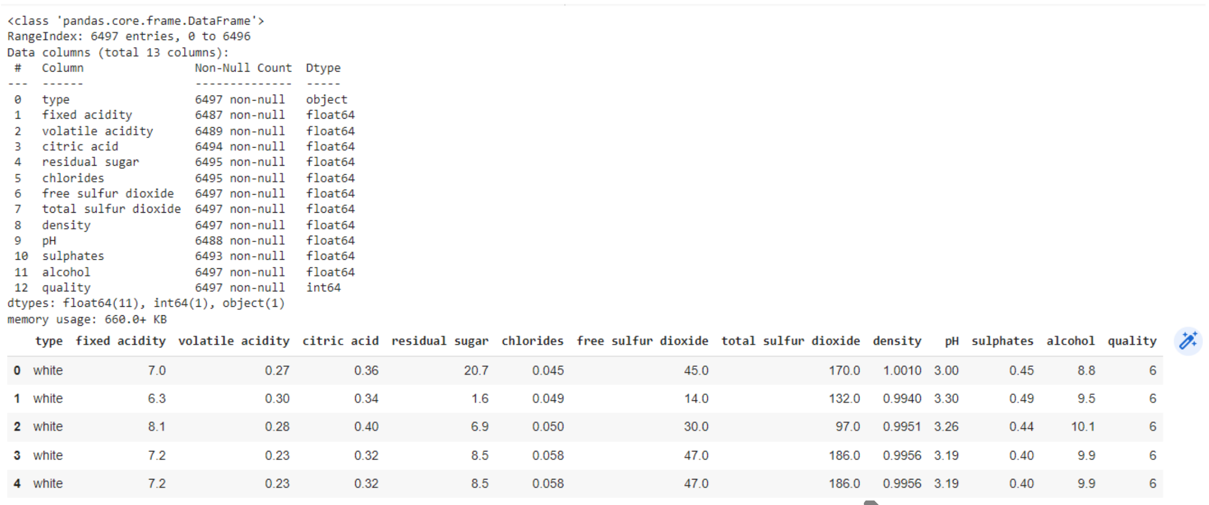
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

1) **IMPORTING LIBRARIES:**

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

2) **DATA PREPROCESSING:**

```
Dataframe = pd.read_csv('winequalityN.csv')
# getting info.
Dataframe.info()
Dataframe.describe()
# null value check
Dataframe.isnull().sum()
Dataframe = Dataframe.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
Dataframe.head()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   type                  6497 non-null   object 
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

3) DATA SPLITTING INTO TRAINING DATASET & TESTING DATASET & CREATING MODEL:

```
x = Dataframe.drop(columns = ['quality','type'])
y = Dataframe['quality']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30,random_state=1)
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(x_train,y_train)
```

```
GaussianNB()
```

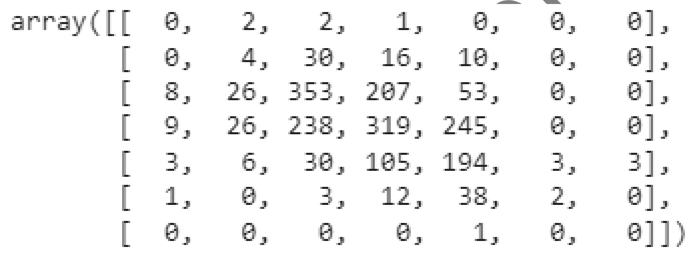
4) PREDICTING QUALITY OF THE WINE:

```
model.score(x_test,y_test)
y_pred = model.predict(x_test)
np.set_printoptions(threshold=np.inf)
y_pred
```

```
array([6, 6, 6, 7, 7, 6, 7, 7, 7, 6, 6, 7, 6, 5, 6, 7, 6, 5, 5, 5, 7, 5,
       6, 5, 6, 7, 5, 7, 5, 5, 7, 6, 5, 6, 6, 7, 7, 5, 6, 8, 5, 7, 7, 7,
       6, 5, 6, 7, 6, 6, 5, 4, 6, 6, 7, 7, 5, 5, 5, 6, 5, 5, 7, 7, 7, 6,
       7, 7, 6, 7, 6, 7, 5, 7, 6, 5, 6, 6, 4, 3, 7, 5, 3, 5, 6, 5, 7, 6,
       6, 5, 6, 5, 6, 7, 7, 5, 7, 6, 7, 7, 5, 6, 6, 3, 6, 5, 5, 7, 6, 6,
       5, 5, 6, 6, 5, 5, 7, 6, 7, 5, 6, 4, 6, 7, 6, 6, 6, 5, 5, 7, 5, 5,
       5, 5, 6, 4, 5, 7, 6, 7, 6, 5, 7, 6, 5, 5, 5, 5, 7, 7, 6, 7, 7, 7,
       5, 6, 6, 7, 6, 7, 5, 7, 6, 5, 7, 7, 7, 7, 7, 5, 6, 7, 7, 5, 5, 6,
       7, 5, 4, 6, 6, 5, 7, 7, 5, 6, 6, 7, 5, 5, 6, 7, 5, 5, 7, 6, 5, 5,
       5, 5, 6, 5, 7, 6, 6, 5, 6, 5, 6, 6, 5, 7, 7, 7, 6, 5, 5, 5, 5, 6,
       7, 5, 7, 7, 5, 6, 5, 7, 5, 5, 5, 6, 6, 7, 6, 6, 7, 6, 7, 6, 7, 7,
       5, 6, 5, 5, 7, 7, 8, 7, 6, 6, 6, 5, 6, 6, 7, 6, 5, 5, 5, 6, 5, 5,
       5, 6, 5, 7, 5, 5, 6, 5, 7, 7, 7, 6, 5, 5, 5, 5, 6, 7, 6, 4, 5, 7,
       5, 6, 6, 7, 6, 7, 7, 4, 7, 7, 7, 3, 6, 6, 5, 7, 7, 7, 7, 4, 6, 7,
       7, 5, 6, 5, 6, 7, 6, 6, 7, 7, 6, 5, 5, 5, 7, 7, 7, 5, 5, 6, 5, 7,
       5, 5, 5, 6, 5, 5, 5, 7, 5, 6, 4, 5, 5, 6, 7, 6, 5, 7, 5, 7, 5, 3,
       6, 7, 6, 7, 6, 6, 6, 6, 5, 5, 6, 5, 6, 6, 5, 5, 5, 7, 5, 4, 6, 7,
       5, 5, 6, 5, 5, 5, 6, 4, 6, 3, 6, 7, 5, 6, 6, 5, 5, 5, 5, 6, 5, 6,
       5, 6, 6, 5, 7, 6, 6, 4, 6, 6, 7, 5, 6, 5, 6, 4, 5, 5, 6, 5, 5, 6,
       5, 5, 6, 6, 5, 6, 7, 6, 5, 5, 7, 5, 5, 6, 6, 7, 5, 5, 7, 7, 7, 7,
       5, 6, 7, 6, 6, 6, 7, 6, 5, 6, 7, 5, 5, 7, 6, 5, 6, 5, 7, 6, 7, 7,
       7, 4, 7, 7, 5, 6, 7, 7, 6, 5, 6, 4, 5, 5, 5, 6, 6, 5, 7, 6, 6, 6,
       6, 6, 5, 5, 7, 6, 5, 6, 6, 7, 6, 5, 5, 7, 5, 5, 5, 6, 5, 5, 6, 6,
       5, 6, 5, 6, 5, 7, 7, 6, 4, 6, 6, 6, 7, 6, 5, 6, 5, 7, 5, 6, 5, 6,
       6, 7, 7, 6, 7, 6, 6, 6, 5, 7, 5, 5, 6, 6, 6, 5, 5, 6, 7, 7, 5, 7,
       6, 4, 5, 6, 6, 7, 6, 5, 5, 7, 7, 7, 7, 5, 5, 6, 6, 4, 6, 5, 4, 5,
       6, 7, 4, 7, 7, 6, 6, 7, 5, 5, 5, 7, 7, 7, 6, 5, 4, 6, 5, 5, 5, 6,
       6, 5, 7, 6, 5, 5, 7, 6, 7, 7, 6, 6, 6, 6, 5, 6, 6, 7, 7, 6, 3, 7,
       5, 5, 5, 7, 6, 5, 3, 5, 7, 5, 7, 6, 6, 5, 7, 7, 5, 7, 6, 5, 6, 5,
       6, 5, 5, 6, 7, 4, 7, 5, 6, 6, 7, 5, 7, 6, 6, 6, 5, 5, 6, 6, 5, 5,
       5, 5, 7, 7, 3, 4, 7, 6, 5, 7, 5, 5, 5, 7, 6, 7, 7, 5, 5, 6, 5, 5,
       6, 6, 7, 6, 7, 6, 6, 7, 7, 7, 7, 5, 5, 5, 5, 6, 7, 7, 6, 6, 6, 5,
       7, 6, 6, 7, 5, 7, 5, 6, 7, 7, 6, 7, 6, 4, 5, 3, 4, 5, 7, 6, 6, 5,
       5, 7, 5, 7, 7, 5, 6, 6, 6, 5, 7, 6, 5, 5, 7, 6, 5, 6, 7, 7, 6, 5,
       5, 6, 6, 5, 5, 5, 5, 7, 5, 6, 6, 6, 6, 3, 6, 7, 6, 5, 6, 6, 7, 5,
       5, 7, 7, 5, 6, 5, 5, 3, 4, 7, 5, 7, 7, 7, 6, 7, 7, 6, 4, 6, 6, 7,
       5, 6, 6, 5, 7, 5, 5, 7, 6, 6, 5, 5, 7, 5, 7, 5, 7, 7, 6, 3, 6, 6,
       7, 5, 6, 7, 5, 7, 5, 7, 6, 5, 4, 5, 5, 7, 5, 6, 6, 3, 4, 7, 6, 6,
       5, 4, 6, 7, 5, 5, 6, 7, 7, 7, 6, 7, 7, 5, 5, 4, 7, 6, 6, 6, 7, 7,
       7, 6, 5, 6, 6, 5, 6, 5, 6, 5, 6, 6, 5, 7, 6, 6, 5, 7, 5, 7, 6, 7,
       5, 7, 5, 7, 5, 5, 6, 7, 7, 6, 5, 6, 5, 6, 5, 4, 6, 7, 7, 5, 6, 5,
       6, 5, 6, 6, 6, 6, 6, 7, 6, 6, 6, 4, 6, 6, 5, 5, 6, 4, 6, 6, 7, 5,
       6, 6, 7, 5, 7, 6, 7, 7, 6, 7, 5, 7, 6, 5, 7, 7, 7, 7, 5, 6, 5, 7,
       5, 6, 6, 7, 5, 7, 7, 7, 6, 5, 5, 5, 7, 5, 7, 6, 6, 7, 7, 6, 6, 5,
       6, 5, 5, 6, 7, 7, 7, 9, 6, 5, 6, 6, 6, 5, 7, 5, 6, 6, 6, 5, 5, 6,
       5, 7, 7, 4, 7, 6, 7, 6, 7, 5, 6, 6, 5, 6, 5, 7, 5, 7, 5, 5, 6, 7,
       5, 6, 7, 6, 5, 6, 5, 6, 6, 5, 7, 5, 5, 6, 6, 6, 5, 7, 5, 6, 5, 6,
       5, 7, 6, 6, 7, 6, 4, 4, 4, 6, 7, 6, 5, 6, 7, 6, 5, 7, 6, 7, 5, 5,
       6, 5, 7, 5, 7, 7, 7, 7, 6, 6, 7, 7, 6, 5, 7, 6, 7, 7, 6, 6, 5, 5,
       5, 6, 6, 7, 7, 5, 6, 7, 6, 6, 7, 7, 7, 6, 6, 5, 5, 7, 7, 5, 5, 7,
       7, 5, 6, 5, 6, 5, 7, 5, 6, 6, 6, 6, 6, 5, 5, 4, 4, 6, 6, 6, 6, 7,
       6, 6, 7, 7, 5, 7, 6, 6, 4, 4, 5, 5, 6, 6, 5, 8, 5, 4, 5, 6, 5, 8,
       5, 4, 6, 3, 7, 6, 6, 6, 4, 5, 7, 5, 7, 7, 7, 6, 6, 6, 5, 6, 6, 6,
       6, 6, 7, 7, 5, 7, 5, 5, 6, 4, 5, 7, 7, 6, 5, 6, 6, 6, 6, 7, 6, 6,
       4, 7, 7, 5, 5, 7, 7, 5, 5, 5, 8, 6, 5, 7, 6, 5, 5, 6, 7, 6, 7, 6,
       5, 4, 7, 5, 7, 5, 5, 6, 6, 5, 6, 7, 5, 5, 5, 5, 7, 6, 5, 5, 6, 6,
       5, 5, 6, 5, 5, 5, 7, 7, 6, 5, 5, 6, 5, 5, 7, 5, 7, 6, 6, 6, 5, 6,
       5, 7, 5, 5, 6, 5, 6, 7, 7, 7, 6, 7, 6, 6, 5, 7, 7, 6, 6, 5, 7, 5,
       6, 5, 7, 7, 6, 5, 5, 6, 7, 6, 6, 6, 5, 5, 5, 5, 5, 7, 7, 7, 6, 5,
       5, 6, 5, 5, 6, 5, 7, 7, 6, 5, 6, 7, 7, 6, 6, 6, 5, 5, 5, 6, 5, 6,
       6, 6, 7, 6, 5, 6, 6, 3, 7, 6, 5, 6, 5, 6, 6, 6, 6, 7, 6, 7, 5, 7,
       6, 5, 6, 5, 5, 6, 6, 5, 6, 6, 7, 7, 5, 5, 5, 5, 6, 5, 5, 7, 6, 5,
       5, 5, 6, 5, 5, 6, 5, 4, 7, 5, 6, 5, 4, 6, 5, 7, 6, 5, 6, 5, 5, 6,
       5, 7, 5, 6, 7, 5, 7, 7, 7, 4, 7, 5, 5, 5, 6, 7, 6, 7, 5, 5, 5, 7,
       7, 6, 6, 6, 6, 6, 6, 5, 5, 6, 5, 7, 5, 6, 6, 6, 7, 7, 7, 5, 5, 7,
       7, 7, 6, 6, 7, 6, 7, 5, 5, 6, 7, 7, 5, 6, 6, 6, 5, 6, 7, 5, 6, 6,
       7, 5, 7, 7, 5, 5, 6, 5, 7, 6, 6, 5, 6, 6, 5, 5, 7, 6, 5, 7, 7, 5,
       7, 7, 7, 5, 7, 7, 7, 6, 6, 5, 5, 7, 7, 7, 5, 5, 6, 6, 5, 7, 5, 7,
       7, 4, 5, 6, 6, 6, 5, 5, 6, 7, 6, 5, 6, 6, 6, 5, 3, 6, 7, 6, 7, 9,
       6, 7, 5, 5, 6, 5, 6, 7, 3, 7, 4, 7, 5, 5, 5, 5, 5, 6, 7, 5, 7, 6,
       4, 7, 6, 6, 6, 7, 7, 5, 5, 5, 5, 5, 7, 5, 6, 6, 7, 5, 5, 5, 5, 5,
       7, 5, 6, 7, 6, 5, 6, 5, 5, 6, 7, 7, 5, 7, 5, 6, 6, 6, 5, 7, 6, 5,
       7, 7, 5, 5, 5, 7, 5, 5, 6, 6, 5, 7, 5, 7, 7, 6, 6, 7, 6, 5, 5, 5,
       7, 5, 5, 6, 7, 7, 7, 7, 6, 5, 7, 5, 5, 5, 7, 5, 7, 7, 7, 6, 7, 7,
       7, 7, 5, 5, 5, 6, 6, 7, 3, 6, 7, 5, 6, 6, 7, 7, 4, 5, 7, 6, 6, 5,
       5, 7, 6, 5, 5, 5, 5, 6, 6, 5, 6, 6, 5, 5, 5, 6, 5, 7, 7, 7, 5, 6,
       7, 6, 4, 5, 6, 5, 6, 7, 7, 6, 5, 6, 6, 6, 5, 6, 5, 7, 6, 5, 5, 5,
       7, 5, 6, 6, 6, 5, 6, 5, 5, 5, 5, 5, 5, 6, 5, 5, 7, 5, 5, 6, 6, 6,
       7, 7, 5, 5, 6, 6, 7, 7, 7, 6, 6, 6, 7, 6, 6, 7, 6, 7, 7, 6, 6, 7, 5,
       5, 5, 5, 5, 7, 5, 5, 6, 5, 4, 7, 6, 5, 6, 6, 5, 5, 5, 5, 5, 5, 7,
       6, 7, 7, 6, 7, 5, 6, 7, 7, 3, 7, 5, 6, 6, 7, 6, 6, 6, 6, 5, 4, 7,
       7, 5, 7, 6, 6, 6, 6, 7, 5, 7, 5, 7, 6, 5, 5, 6, 6, 5, 5, 7, 6, 7,
       5, 5, 5, 6, 6, 7, 5, 5, 6, 4, 6, 6, 7, 7, 5, 6, 4, 7, 7, 7, 6, 6,
       7, 7, 7, 5, 6, 7, 7, 6, 7, 5, 5, 5, 6, 6, 7, 7, 6, 5, 7, 6, 6, 7,
       7, 3, 7, 5, 7, 6, 7, 6, 6, 5, 6, 6, 7, 4, 6, 6, 9, 5, 6, 5, 5, 5,
       6, 5, 7, 5, 4, 4, 6, 5, 5, 6, 7, 6, 7, 4, 5, 5, 5, 5, 5, 6, 6, 5,
       7, 6, 7, 5, 6, 5, 5, 7, 6, 6, 6, 7, 6, 7, 5, 6, 6, 7, 6, 5, 6, 7,
       6, 5, 7, 5, 7, 5, 6, 7, 5, 7, 7, 6, 7, 7, 5, 6, 6, 5, 5, 7, 7, 7,
       7, 6, 5, 5, 7, 6, 5, 5, 7, 5, 5, 5, 6, 6])
```

5) CONFUSION MATRIX:

metrics.confusion_matrix(y_test,y_pred)



```
array([[  0,   2,   2,   1,   0,   0,   0],
       [  0,   4,  30,  16,  10,   0,   0],
       [  8,  26, 353, 207,  53,   0,   0],
       [  9,  26, 238, 319, 245,   0,   0],
       [  3,   6,  30, 105, 194,   3,   3],
       [  1,   0,   3,  12,  38,   2,   0],
       [  0,   0,   0,   0,   1,   0,   0]])
```

CONCLUSION:

From this practical, I have learned the implementation of naïve bayes classifier in python.