**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:

Naïve Bayes Classifier Algorithm

1. **IMPORTING LIBRARIES:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import metrics

import seaborn as sns

1. **DATA PREPROCESSING:**

Dataframe = pd.read\_csv('winequalityN.csv')

# getting info.

Dataframe.info()

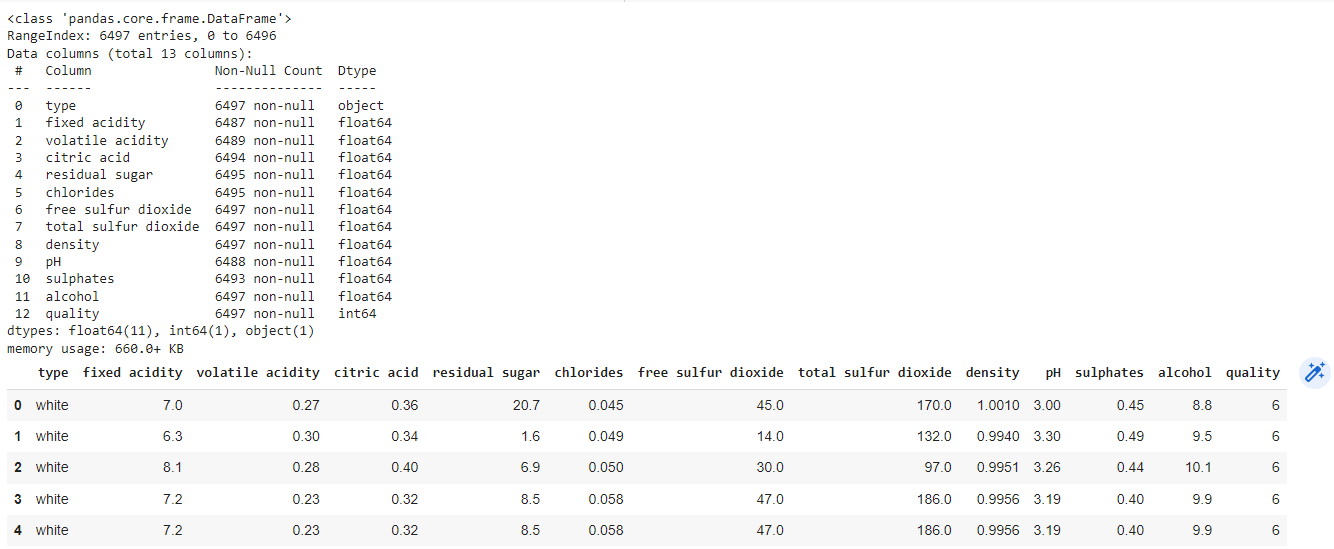
Dataframe.describe()

# null value check

Dataframe.isnull().sum()

Dataframe = Dataframe.replace((np.inf, -np.inf, np.nan), 0).reset\_index(drop=True)

Dataframe.head()



1. **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)



1. **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, 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,

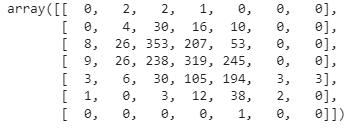
7, 6, 7, 5, 6, 5, 5, 7, 6, 6, 6, 7, 6, 7, 5, 6, 6, 7, 6, 5, 6, 7,

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])

1. **CONFUSION MATRIX:**

metrics.confusion\_matrix(y\_test,y\_pred)



**CONCLUSION:**

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