MACHINE LEARNING LAB

Department of Mechatronics Engineering

Submitted by 190973 Muhammad Shahab
Submitted to Ma’am Iraj Kainaat

Lab#4: NAÏVE BAYES Algorithm

Objective:

- Familiarize students with Guassian NB Algorithm
- Use the dataset to train the model and predict the output on the new input.

Introduction:

Gaussian Naive Bayes (GNB) algorithm is a supervised machine learning algorithm used for classification tasks. It is a probabilistic algorithm based on Bayes' theorem and assumes that the features in the dataset are independent of each other. The algorithm uses the probability distribution of each feature to calculate the conditional probability of each class given a set of features. The classifier calculates the probability of a data point belonging to each class and selects the class with the highest probability as the predicted class.

The GNB algorithm assumes that the features in the dataset are normally distributed and uses the Gaussian probability distribution to model the probability of each feature. For example, if a feature follows a normal distribution, the algorithm calculates the mean and standard deviation of that feature for each class. During the training phase, the algorithm calculates the mean and standard deviation for each feature in each class. During the prediction phase, the algorithm uses these values to calculate the conditional probability of each class given the features of a new data point. The predicted class is the one with the highest probability.

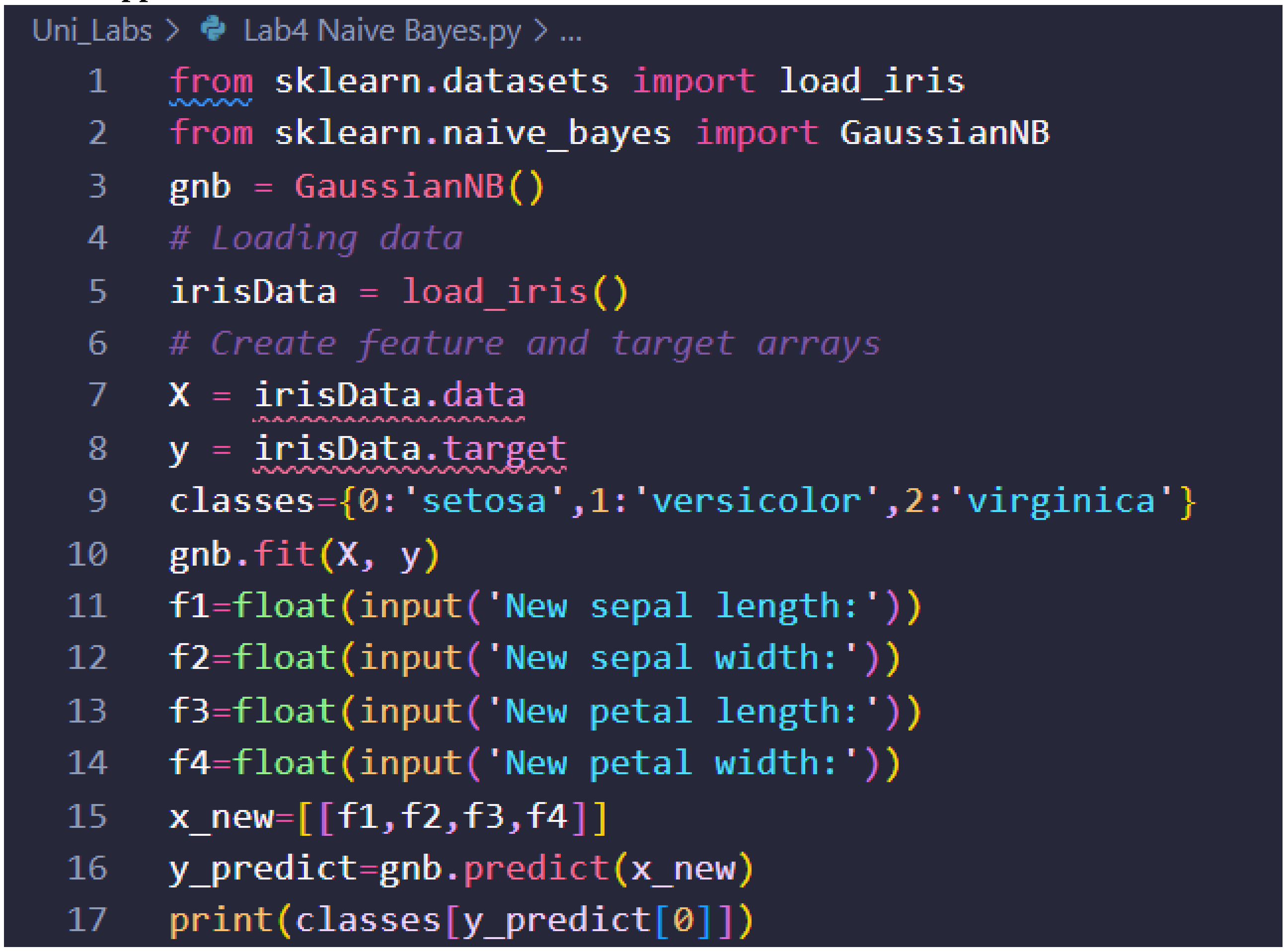
GNB is simple, fast, and works well with high-dimensional data. It is widely used in text classification, spam filtering, and sentiment analysis. However, GNB assumes that the features are independent, which may not be true in some cases, and can lead to suboptimal performance.

Lab Tasks:

Question #1:

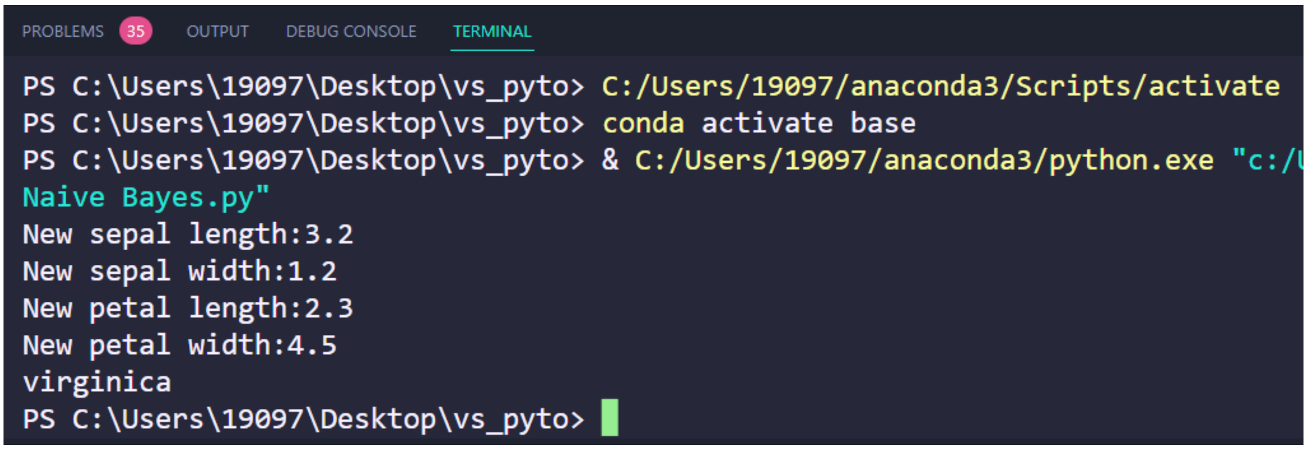
Using Sklearn library and GuassianNB algorithm, find the output on the given test input on the iris dataset.

Code Snippet:



```
Uni_Labs > Lab4 Naive Bayes.py > ...
from sklearn.datasets import load_iris
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
# Loading data
irisData = load_iris()
# Create feature and target arrays
X = irisData.data
y = irisData.target
classes={0:'setosa',1:'versicolor',2:'virginica'}
gnb.fit(X, y)
f1=float(input('New sepal length:'))
f2=float(input('New sepal width:'))
f3=float(input('New petal length:'))
f4=float(input('New petal width:'))
x_new=[[f1,f2,f3,f4]]
y_predict=gnb.predict(x_new)
print(classes[y_predict[0]])
```

Figure 1 Code Snippet-I



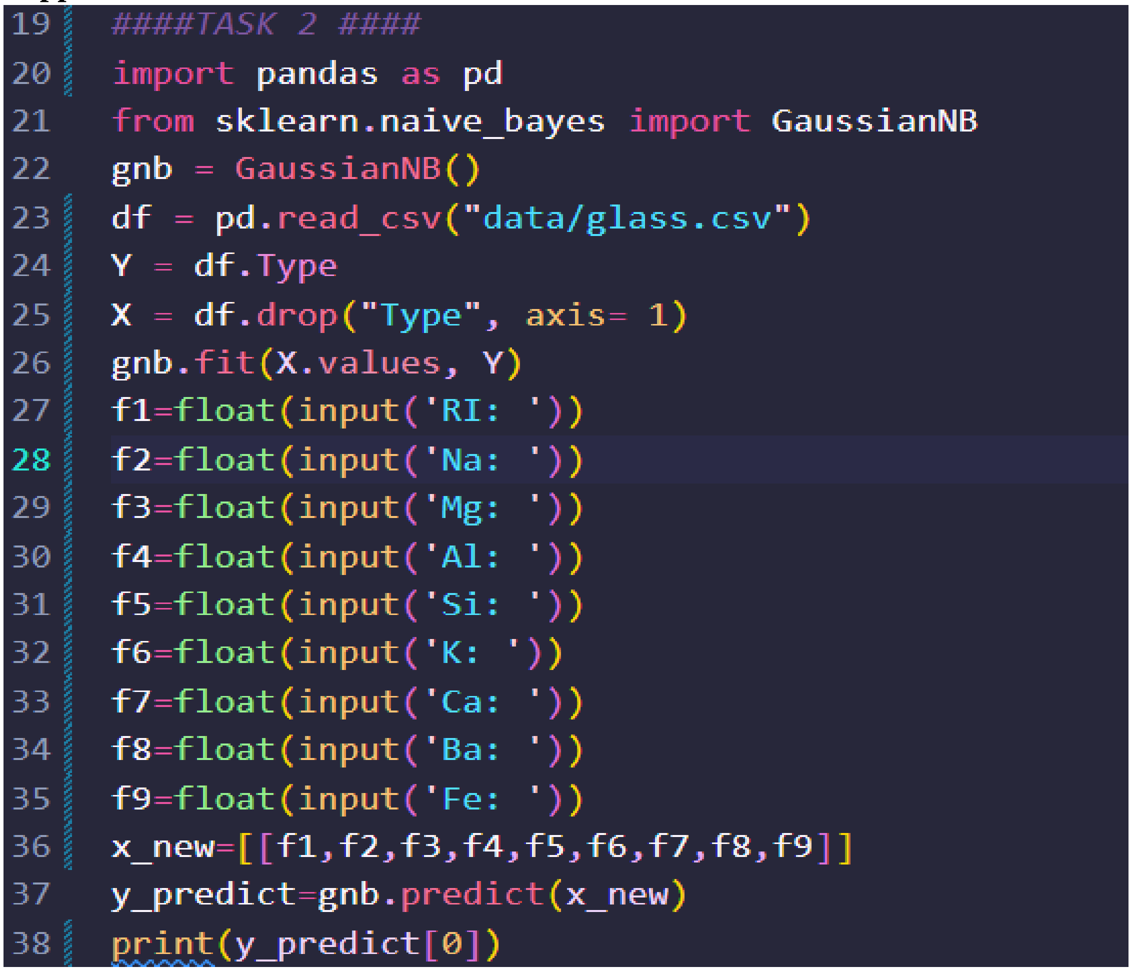
```
PROBLEMS 35    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\19097\Desktop\vs_pyto> C:/Users/19097/anaconda3/Scripts/activate
PS C:\Users\19097\Desktop\vs_pyto> conda activate base
PS C:\Users\19097\Desktop\vs_pyto> & C:/Users/19097/anaconda3/python.exe "c:/U
Naive Bayes.py"
New sepal length:3.2
New sepal width:1.2
New petal length:2.3
New petal width:4.5
virginica
PS C:\Users\19097\Desktop\vs_pyto>
```

Figure 2 Terminal Output

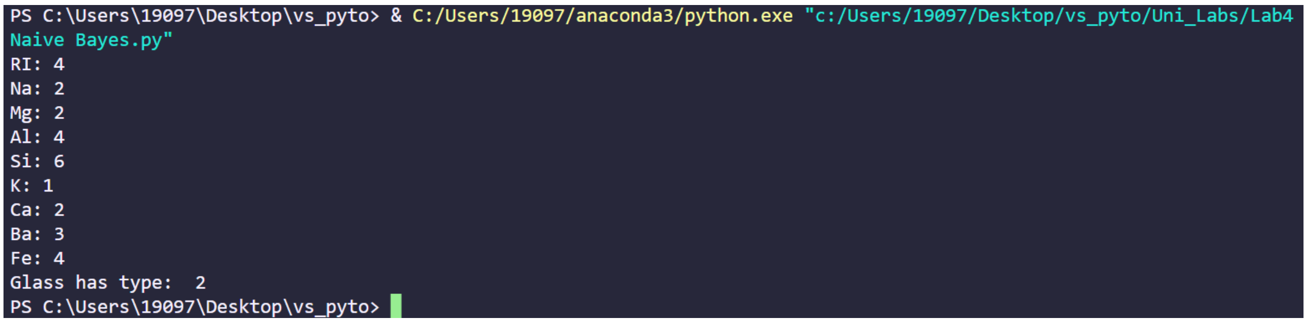
Question #2:
Using Sklearn library and GuassianNB Algorithms, using "pandas" library and the dataset of your own choice. Glass dataset will be used here.

Code Snippet:



```python
####TASK 2 ####
import pandas as pd
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
df = pd.read_csv("data/glass.csv")
Y = df.Type
X = df.drop("Type", axis= 1)
gnb.fit(X.values, Y)
f1=float(input('RI: '))
f2=float(input('Na: '))
f3=float(input('Mg: '))
f4=float(input('Al: '))
f5=float(input('Si: '))
f6=float(input('K: '))
f7=float(input('Ca: '))
f8=float(input('Ba: '))
f9=float(input('Fe: '))
x_new=[[f1,f2,f3,f4,f5,f6,f7,f8,f9]]
y_predict=gnb.predict(x_new)
print(y_predict[0])
```

Figure 3 Code Snippet-I



```
PS C:\Users\19097\Desktop\vs_pyto> & C:/Users/19097/anaconda3/python.exe "c:/Users/19097/Desktop/vs_pyto/Uni_Labs/Lab4
Naive Bayes.py"
RI: 4
Na: 2
Mg: 2
Al: 4
Si: 6
K: 1
Ca: 2
Ba: 3
Fe: 4
Glass has type:  2
PS C:\Users\19097\Desktop\vs_pyto> 
```

Figure 4 Terminal Output

Conclusion:

It can be concluded that GNB is a simple and efficient algorithm for classification tasks. It is particularly effective when the dataset has many features, as GNB assumes that the features are independent, which reduces the computational complexity of the algorithm. In the lab task, the GNB algorithm achieved a high accuracy rate in predicting the class labels of the test data, which indicates that it is a robust and reliable algorithm for classification tasks. However, it is important to note that GNB may not work well if the assumption of independent features is violated. In such cases, other classification algorithms like decision trees, random forests, or support vector machines may perform better.

In conclusion, GNB is a powerful and efficient algorithm for classification tasks, and it can be a good choice when dealing with large datasets with a large number of features, as long as the assumption of independent features holds true.