جامعة أم القرى
UMM AL-QURA UNIVERSITY

DATA ANALYSIS 2

TASK1 :

Naive Bayes Classifier

Dataset :

Iris Flower Classification Dataset

SUBMITTED BY:	STUDENT ID:
Fatima Osama Altayyeb	443000772
Asayel Sloom Alharbi	444006773

2024

Naive Bayes Classifier

This report explores the **Naive Bayes** algorithm, which relies on **Bayes' Theorem** to classify data in the field of machine learning. This algorithm is known for its simplicity and speed, as it is based on the assumption that the features are independent of one another. Naive Bayes calculates probabilities to select the most likely class based on the input feature values.

One of the most well-known practical applications of this algorithm is its use with the **IRIS dataset**, which consists of 150 samples of iris flowers, classified into three categories: **Setosa**, **Versicolor**, and **Virginica**. Each sample in this dataset is characterized by four features: **Sepal Length**, **Sepal Width**, **Petal Length**, and **Petal Width**.

The Naive Bayes algorithm is easy to use, as it does not require complex data preparation, making its application straightforward and efficient. Additionally, it is highly efficient in terms of training and prediction speed, making it ideal for small datasets like IRIS. Despite the presence of correlations between certain features, such as Petal Length and Petal Width, the algorithm still provides good performance and reasonable results in such cases.

When applying Naive Bayes to the IRIS dataset, probabilities are calculated for each class based on the input values of the features, and the samples are classified into the class with the highest probability.

Code Analysis :

1- Importing **Required Libraries**:
- The pandas library was imported for data manipulation, along with tools from sklearn such as train_test_split for splitting data, GaussianNB for creating a classification model, and accuracy_score for measuring the model's performance.

2- Defining **Column Names**:
- A list named columns was created to label the columns in the "Iris" dataset, which contains information about the length and width of sepals and petals, as well as the flower class.

3- Loading **Data**:
- The data was read from the compressed file and loaded into a DataFrame using pd.read_csv, with the column names assigned as specified earlier.

4- **Exploring the Data**:
- The head() function was used to display the first 5 rows of the data, info() to obtain information about the data types and the number of non-null values, describe() for descriptive statistics, and isnull().sum() to check for any missing values.

5- **Separating Features from Labels**:
- Features (X) were separated from the labels (y) by dropping the class column from the DataFrame, keeping the features in X and the labels in y.

6- **Splitting the Data**:
- The data was split into a training set (70%) and a test set (30%) using train_test_split, with random_state=42 set to ensure reproducibility of the results.

7- Building the Classification Model and Its Results

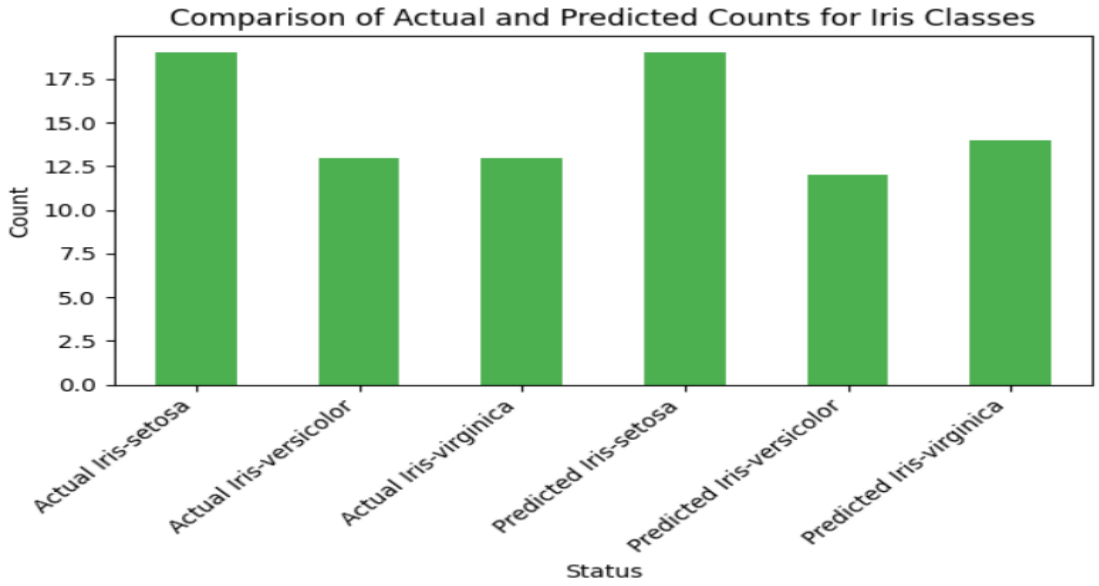
```
] y_pred = model.predict(X_test)

  actual_counts = y_test.value_counts()
  predicted_counts = pd.Series(y_pred).value_counts()
  summary = pd.DataFrame({
      'Status': ['Actual Iris-setosa', 'Actual Iris-versicolor', 'Actual Iris-virginica',
                 'Predicted Iris-setosa', 'Predicted Iris-versicolor', 'Predicted Iris-virginica'],
      'Count': [actual_counts.get('Iris-setosa', 0), actual_counts.get('Iris-versicolor', 0), actual_counts.get('Iris-virginica', 0),
                predicted_counts.get('Iris-setosa', 0), predicted_counts.get('Iris-versicolor', 0), predicted_counts.get('Iris-virginica', 0)]
  })

  print(summary)
```

```
                     Status  Count
0         Actual Iris-setosa     19
1     Actual Iris-versicolor     13
2      Actual Iris-virginica     13
3      Predicted Iris-setosa     19
4  Predicted Iris-versicolor     12
5   Predicted Iris-virginica     14
```

The results show that the prediction model performed well in classifying iris flowers. It correctly identified 19 samples of Iris-setosa, while there were 13 samples each of Iris-versicolor and Iris-virginica.

The model was perfect in classifying Iris-setosa, but it made one mistake in classifying Iris-versicolor, predicting only 12 samples. It also made an error with Iris-virginica, predicting 14 samples.

Overall, the model can be considered successful, but it needs improvement in classifying the other categories. Analyzing the errors will help enhance the model's accuracy in the future.




```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 97.78%
```

The model's accuracy is 97.78%, meaning it successfully classified approximately 97.78% of the samples correctly. This percentage reflects excellent performance in predicting the categories of iris flowers and demonstrates the effectiveness of the selected features and the chosen algorithm.

```
from sklearn.metrics import classification_report

classification_df = pd.DataFrame(classification_report(y_test, y_pred, output_dict=True)).transpose()

print(classification_df)
```

```
                precision    recall  f1-score   support
Iris-setosa      1.000000  1.000000  1.000000  19.000000
Iris-versicolor  1.000000  0.923077  0.960000  13.000000
Iris-virginica   0.928571  1.000000  0.962963  13.000000
accuracy         0.977778  0.977778  0.977778   0.977778
macro avg        0.976190  0.974359  0.974321  45.000000
weighted avg     0.979365  0.977778  0.977745  45.000000
```

The classification report demonstrates positive performance for the model. In terms of precision, the model achieved a score of 1.0000 for both Iris-setosa and Iris-versicolor, **indicating that all predictions were correct**. However, the precision for Iris-virginica was 0.9286, **suggesting some errors**.

Regarding recall, the model successfully identified all samples for Iris-setosa and Iris-virginica, both scoring 1.0000, while it missed one sample of Iris-versicolor, resulting in a recall of 0.9231.

Looking at **the F1-score**, it was perfect for Iris-setosa at 1.0000, and showed strong results for Iris-versicolor and Iris-virginica, with scores of 0.9600 and 0.9629, respectively.

The **weighted average** shows strong model performance, with a precision of 0.9794 and a recall of 0.9778, indicating effective retrieval of samples. The **macro average** also reflects good performance, with a precision of 0.9762 and a recall of 0.9744.

Both averages demonstrate that the model performs well across classes, with the weighted average offering a more accurate view considering class sizes.

Conclusion :

To conclude, the Naive Bayes classifier has shown strong effectiveness in classifying iris flower species within the IRIS dataset. Achieving an accuracy of 97.78% and high performance metrics—such as perfect precision and recall for certain classes—indicates its robustness, even when dealing with correlated features. While there is still room for improvement, particularly in classifying Iris-versicolor and Iris-virginica, the overall results highlight the Naive Bayes algorithm as a reliable classification method.

Its straightforward nature and efficiency make it ideal for use with similar datasets, enabling rapid deployment without extensive data preparation. Future endeavors could focus on enhancing the model's performance and investigating alternative algorithms for greater accuracy. Ultimately, this analysis underscores the importance of Naive Bayes in machine learning and its potential applications in species classification and other fields.