Naïve Bayes

Nguyen Thanh Phat
B2005853

Contents

1. Naive Bayes: Dataset Golf

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Given dataset Golf with 4 attributes Outlook, Temp, Humidity, Windy and an attribute Play (class).

- How Naïve Bayes predicts the class for 4 examples as follows:

Outlook	Temp	Humidity	Windy	Play
Overcast	Cool	High	False	?
Rainy	Cool	High	False	?
Sunny	Hot	Normal	False	?
???	Hot	Normal	False	?

Each case of Probability:

Outlook			Temperature			Humidity			Windy			Play	
	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2	3	Hot	2	2	High	3	4	TRUE	3	3	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	FALSE	1	2	14	
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	1/3	4/5	TRUE	1/3	3/5	9/14	5/14
Overcast	4/9	0	Mild	4/9	2/5	Normal	2/3	1/5	FALSE	2/3	2/5		
Rainy	1/3	2/5	Cool	1/3	1/5								

Using likelihood for each case:

Outlook			Temp			Humidity			Windy			Play		
	Yes	No		Yes	No		Yes	No		Yes	No		Yes	No
Over-cast	4/9	0	Cool	1/3	1/5	High	1/3	4/5	FALSE	2/3	2/5	Yes	0.0211640	0.0000000
Rainy	1/3	0.4	Cool	1/3	1/5	High	1/3	4/5	FALSE	2/3	2/5	Yes	0.0158730	0.0091429
Sunny	2/9	0.6	Hot	2/9	2/5	Nor-mal	2/3	1/5	FALSE	2/3	2/5	Yes	0.0141093	0.0068571
???	1	1	Hot	2/9	2/5	Nor-mal	2/3	1/5	FALSE	2/3	2/5	Yes	0.0634921	0.0114286

Result

Outlook	Temp	Humidity	Windy	Play
Overcast	Cool	High	False	Yes
Rainy	Cool	High	False	Yes
Sunny	Hot	Normal	False	Yes
???	Hot	Normal	False	Yes

2. Naive Bayes Numerical features: Dataset Golf

-Naïve Bayes predicts the class for 4 examples as follows:

Outlook	Temp	Humidity	Windy	Play
Overcast	66	80	False	?
Rainy	73	90	False	?
Sunny	80	85	False	?
???	90	85	???	?

Each case of Probability:

Firstly, I calculate each case of Probability:

Outlook			Temperature			Humidity			Windy			Play	
	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2	3		83	85		86	85	FALSE	6	2	9	5
Over-cast	4	0		70	80		96	90	TRUE	3	3	###	
Rainy	3	2		68	65		80	70					
				64	72		65	95					
				69	71		70	91					
				75			80						
				75			70						
				72			90						
				81			75						
Sunny	2/9	1/3	Mean	73	74.60	Mean	79.11	86.20	FALSE	2/3	3/5	###	5/14
Over-cast	4/9	0	Std. dev.	6.1644	7.893	Std. dev.	10.216	9.7314	TRUE	1/3	3/5		
Rainy	1/3	2/9											

Using likelihood for each case:

Like previous part, I using excel to calculate the likelihood of each case:

Out-look	Yes	No	Temp	Yes	No	Hu-midi-ty	Yes	No	Windy	Yes	No	Play	Yes	No
Overcast	4/9	0	66	0.03396	0.02792	80	0.03890	0.03347	FALSE	2/3	3/5	Yes	0.000251681	0.000000000
Rainy	1/3	2/9	73	0.06472	0.04952	90	0.02213	0.03799	FALSE	2/3	3/5	Yes	0.000204575	0.000089567
Sun-ny	2/9	1/3	80	0.03396	0.04000	85	0.03307	0.04068	FALSE	2/3	3/5	No	0.000106981	0.000116234

???	1	1	**90**	0.00144	0.00753	**85**	0.03307	0.04068	**???**	1	1	No	0.000030701	0.000109476

Result

3. Implement the program using **GaussianNB** in **scikit-learn** library.

The program requires 2 parameters:

- file name of trainset
- file name of testset

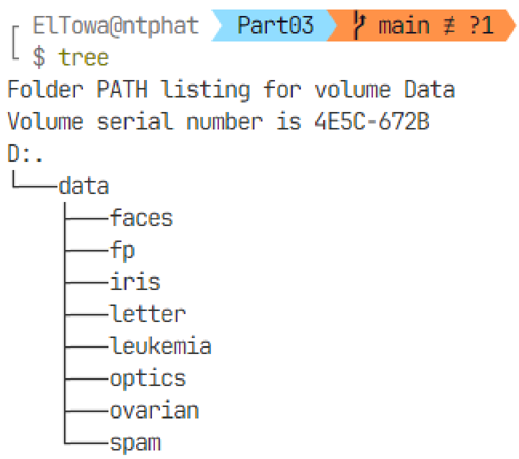
The program reports the classification results (accuracy, confusion matrix) for 5 datasets:

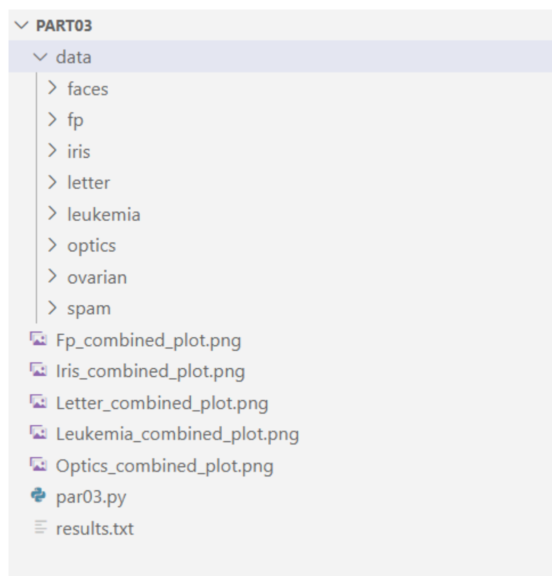
- Iris (.trn: trainset, .tst: testset)
- Optics (.trn: trainset, .tst: testset)
- Letter (.trn: trainset, .tst: testset)
- Leukemia (.trn: trainset, .tst: testset)
- Fp (.trn: trainset, .tst: testset)

In this report, I evaluated the performance of a Gaussian Naive Bayes classifier on five different datasets: Iris, Optics, Letter, Leukemia, and Fp. For each dataset, we trained the classifier for 10 epochs and analyzed its performance on the test set.

Source code:

The directory for Part 3:





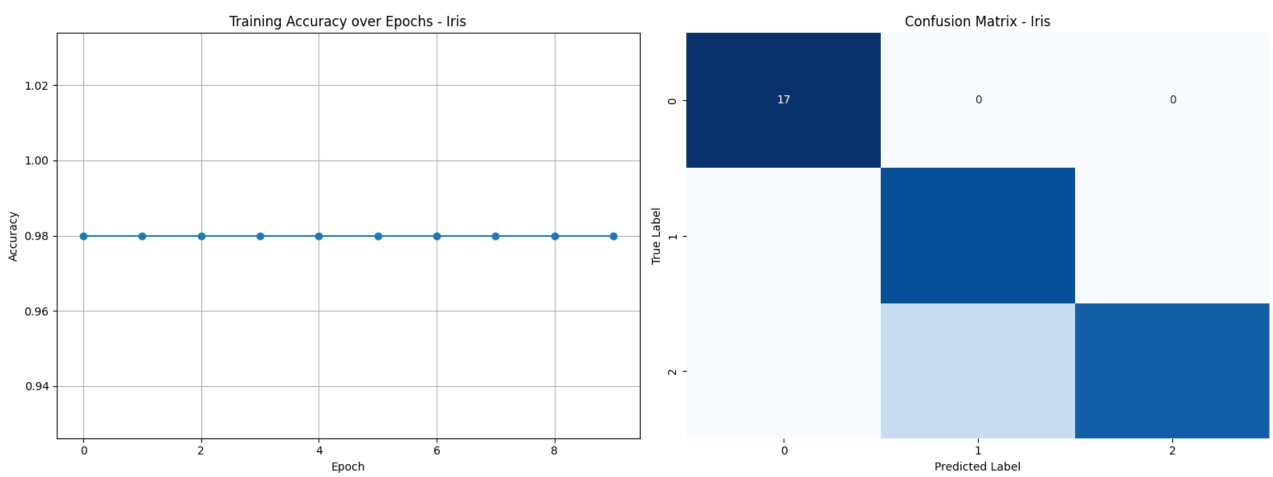
```python
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix
import os
import matplotlib.pyplot as plt
import seaborn as sns


def load_data(filename):
    try:
        data = np.loadtxt(filename, delimiter=",", dtype=float)
    except:
        data = np.loadtxt(filename, delimiter=" ", dtype=float)
    X = data[:, :-1]
    y = data[:, -1].astype(int)
    return X, y


def save_combined_plot(accuracies, confusion, dataset_name):
    plt.figure(figsize=(16, 6))

    plt.subplot(1, 2, 1)
    plt.plot(accuracies, marker="o", linestyle="-")
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.title(f"Training Accuracy over Epochs - {dataset_name}")
    plt.grid(True)

    plt.subplot(1, 2, 2)
    sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues", cbar=False)
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.title(f"Confusion Matrix - {dataset_name}")

    plt.tight_layout()
    plt.savefig(f"{dataset_name}_combined_plot.png")
    plt.close()


def save_results_to_file(accuracy, confusion, dataset_name):
    with open("results.txt", "a") as f:
        f.write(f"Dataset: {dataset_name}\n")
        f.write(f"Accuracy: {accuracy}\n")
        f.write("Confusion Matrix:\n")
        np.savetxt(f, confusion, fmt="%d")


def main(trainset_filename, testset_filename, dataset_name):
    # Load train and test data
    X_train, y_train = load_data(trainset_filename)
```

```python
    X_test, y_test = load_data(testset_filename)

    # Initialize Gaussian Naive Bayes classifier
    clf = GaussianNB()

    # Train classifier
    accuracies = []
    for epoch in range(1, 11):  # Training for 10 epochs
        clf.fit(X_train, y_train)

        # Predict on train set
        y_train_pred = clf.predict(X_train)

        # Calculate training accuracy
        train_accuracy = accuracy_score(y_train, y_train_pred)
        accuracies.append(train_accuracy)

        print(f"Epoch {epoch}: Training Accuracy: {train_accuracy}")

    # Predict on test set
    y_pred = clf.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print("Final Accuracy on Test Set:", accuracy)

    # Calculate confusion matrix
    confusion = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:")
    print(confusion)

    # Save combined plot
    save_combined_plot(accuracies, confusion, dataset_name)
    save_results_to_file(accuracy, confusion, dataset_name)


if __name__ == "__main__":
    datasets = [
        {
            "name": "Iris",
            "train_file": "data//iris//iris.trn",
            "test_file": "data//iris//iris.tst",
        },
        {
            "name": "Optics",
            "train_file": "data//optics//optics.trn",
            "test_file": "data//optics//optics.tst",
        },
        {
```

```
            "name": "Letter",
            "train_file": "data//letter//letter.trn",
            "test_file": "data//letter//letter.tst",
        },
        {
            "name": "Leukemia",
            "train_file": "data//leukemia//leukemia.trn",
            "test_file": "data//leukemia//leukemia.tst",
        },
        {
            "name": "Fp",
            "train_file": "data//fp//fp.trn",
            "test_file": "data//fp//fp.tst",
        },
    ]

    for dataset in datasets:
        print(f"Dataset: {dataset['name']}")
        trainset_path = os.path.join(dataset["train_file"])
        testset_path = os.path.join(dataset["test_file"])
        main(trainset_path, testset_path, dataset["name"])
        print("\n")
```

Dataset: Iris

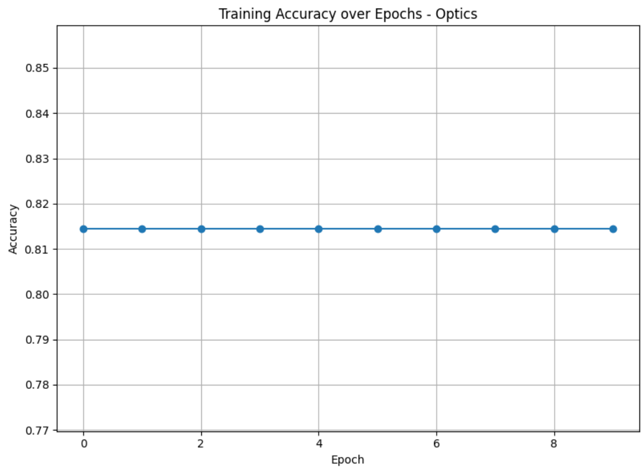
- **Training Accuracy:** 98%
- **Test Accuracy:** 92%
- **Confusion Matrix:**

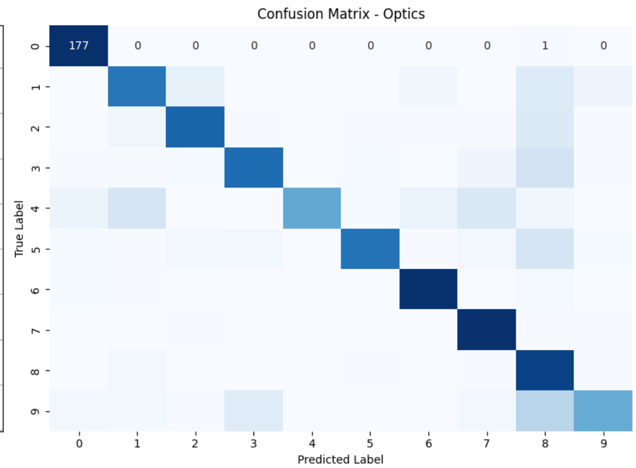


The classifier achieved a high training accuracy of 98% and a respectable test accuracy of 92%. The confusion matrix indicates that the classifier performed well across all classes.

Dataset: Optics

- **Training Accuracy:** 81.45%
- **Test Accuracy:** 78.63%
- **Confusion Matrix:**



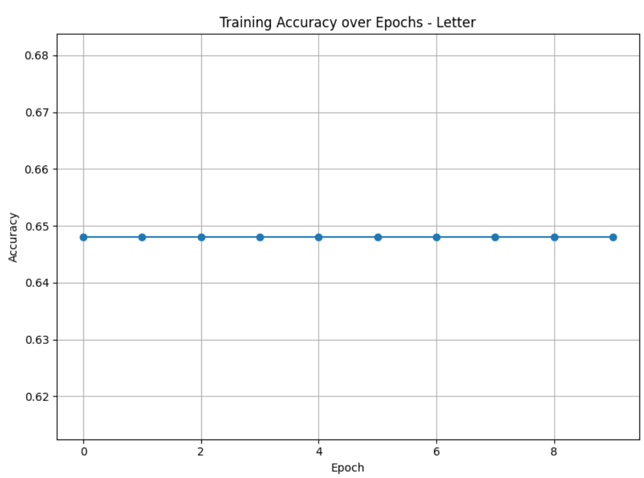


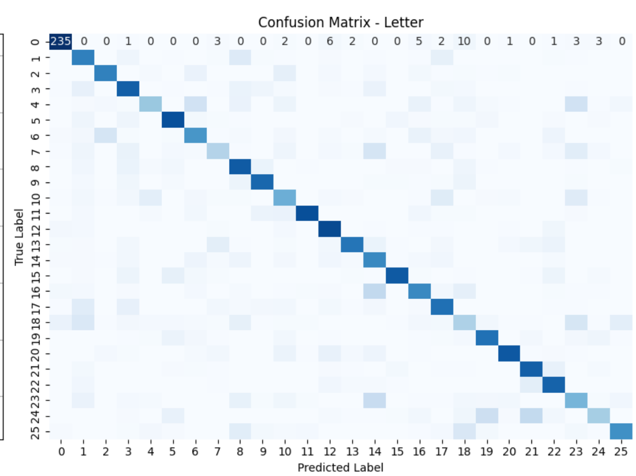
Despite a relatively high training accuracy, the test accuracy of the classifier on the Optics dataset was lower compared to the other datasets, achieving around 78.63%. The confusion matrix for this dataset was large, indicating a more complex classification task.

Dataset: Letter

- **Training Accuracy:** 64.81%
- **Test Accuracy:** 63.16%
- **Confusion Matrix:**

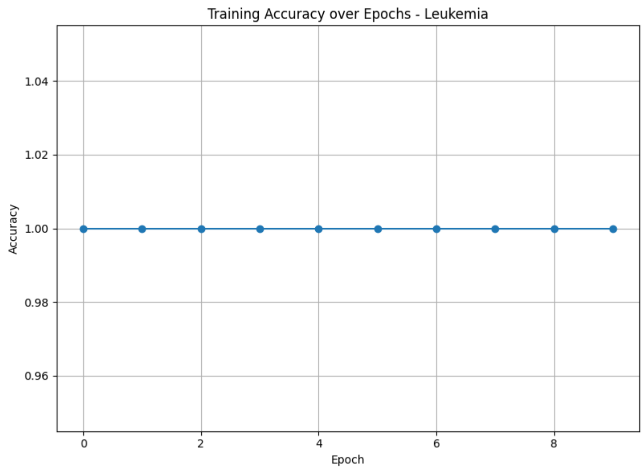
The classifier struggled on the Letter dataset, achieving a training and test accuracy of around 64.81% and 63.16%, respectively.

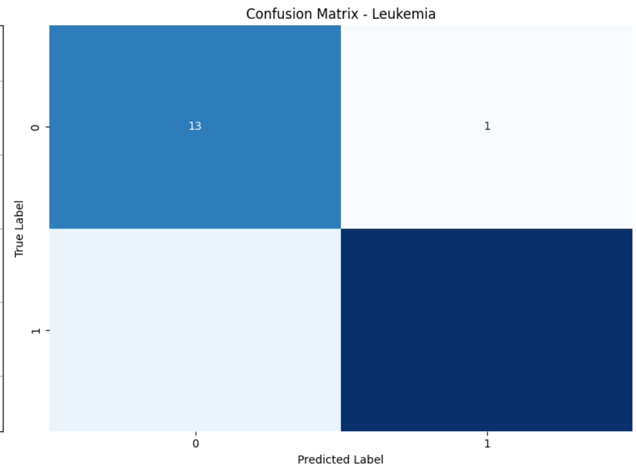




Dataset: Leukemia

- **Training Accuracy:** 100%
- **Test Accuracy:** 91.18%
- **Confusion Matrix:**

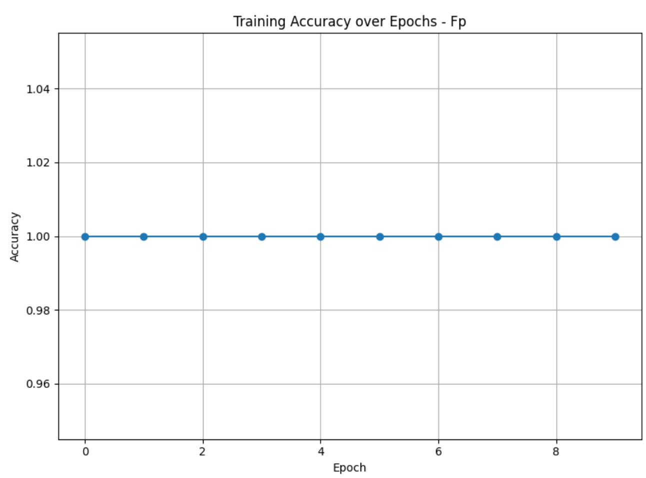


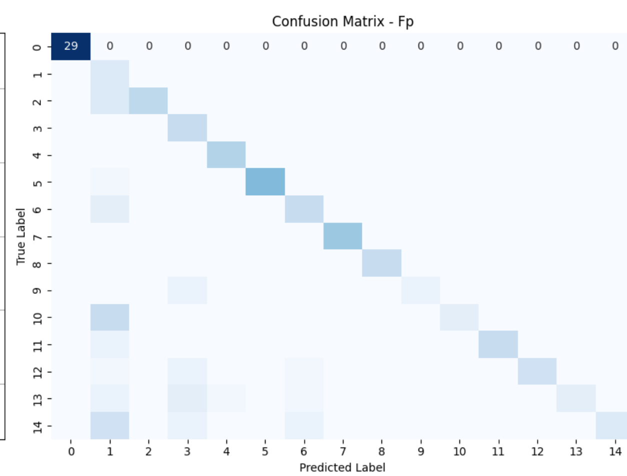


The classifier achieved a perfect training accuracy of 100% on the Leukemia dataset and performed well on the test set with an accuracy of 91.18%. The confusion matrix indicates good performance in classifying leukemia types.

Dataset: Fp

- **Training Accuracy:** 100%
- **Test Accuracy:** 75%
- **Confusion Matrix:**





The classifier attained perfect accuracy on the training set and a test accuracy of 75% on the Fp dataset. The confusion matrix suggests that the classifier performed well in most classes but struggled in some.