Naïve Bayes

**Nguyen Thanh Phat**

B2005853

Contents

[1. Naive Bayes: Dataset Golf 1](#_Toc157230359)

[Each case of Probability: 2](#_Toc157230360)

[Using likelihood for each case: 2](#_Toc157230361)

[Result 2](#_Toc157230362)

[2. Naive Bayes Numerical features: Dataset Golf 2](#_Toc157230363)

[Each case of Probability: 3](#_Toc157230364)

[Using likelihood for each case: 3](#_Toc157230365)

[Result 4](#_Toc157230366)

[3. Implement the program using **GaussianNB** in **scikit-learn** library. 4](#_Toc157230367)

[Source code: 4](#_Toc157230368)

[Dataset: Iris 7](#_Toc157230369)

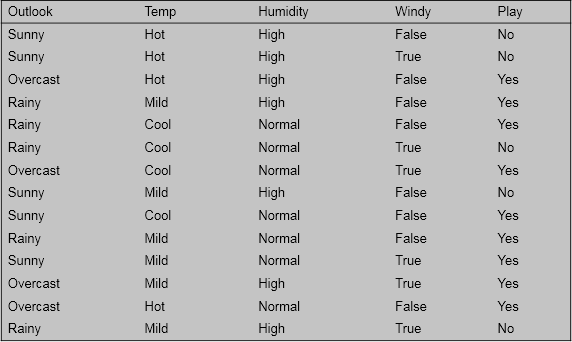
[Dataset: Optics 8](#_Toc157230370)

[Dataset: Letter 8](#_Toc157230371)

[Dataset: Leukemia 9](#_Toc157230372)

[Dataset: Fp 9](#_Toc157230373)

## Naive Bayes: Dataset Golf

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** |
| **Overcast** | 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 | **Normal** | 2/3 | 1/5 | **FALSE** | 2/3 | 2/5 | Yes | 0.0141093 | 0.0068571 |
| **???** | 1 | 1 | **Hot** | 2/9 | 2/5 | **Normal** | 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 |

## 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 |
| Overcast | 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 |
| Overcast | 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:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Outlook** | **Yes** | **No** | **Temp** | **Yes** | **No** | **Humidity** | **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 |
| **Sunny** | 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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Outlook** | **Temp** | **Humidity** | **Windy** | **Play** |
| Overcast | 66 | 80 | False | Yes |
| Rainy | 73 | 90 | False | Yes |
| Sunny | 80 | 85 | False | No |
| ??? | 90 | 85 | ??? | No |

## 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:

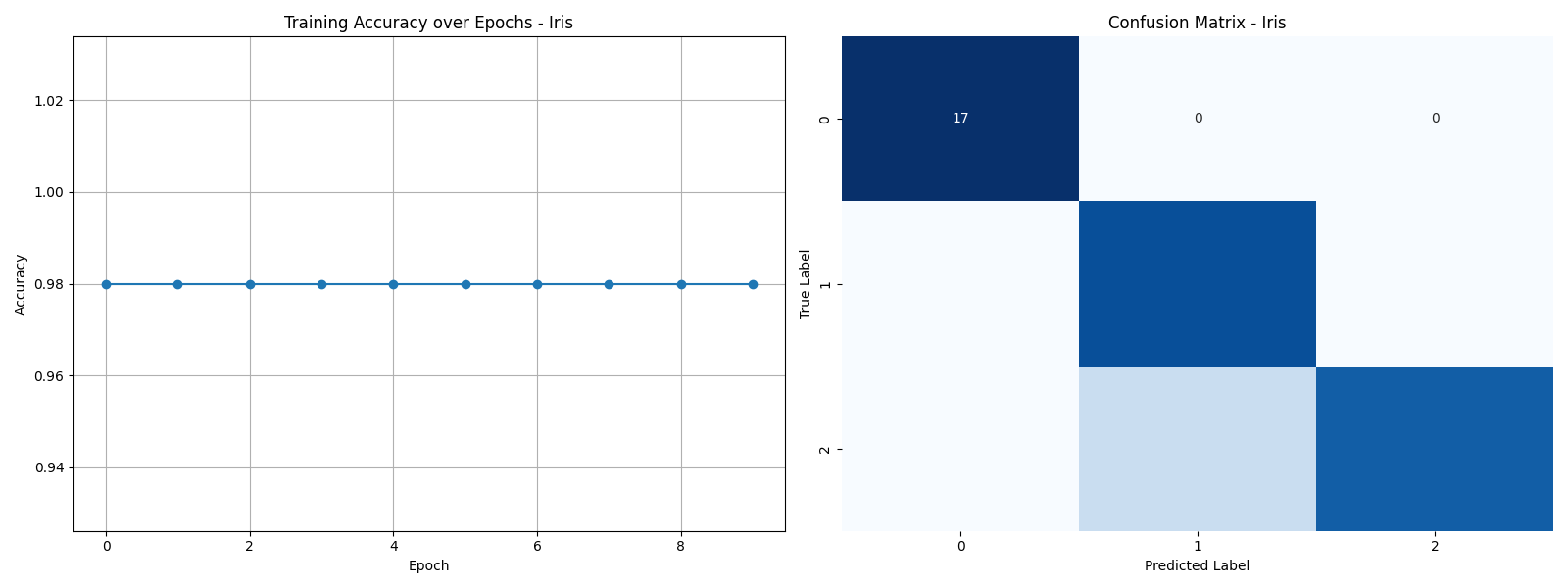
The directory for Part 3:

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
| 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)      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

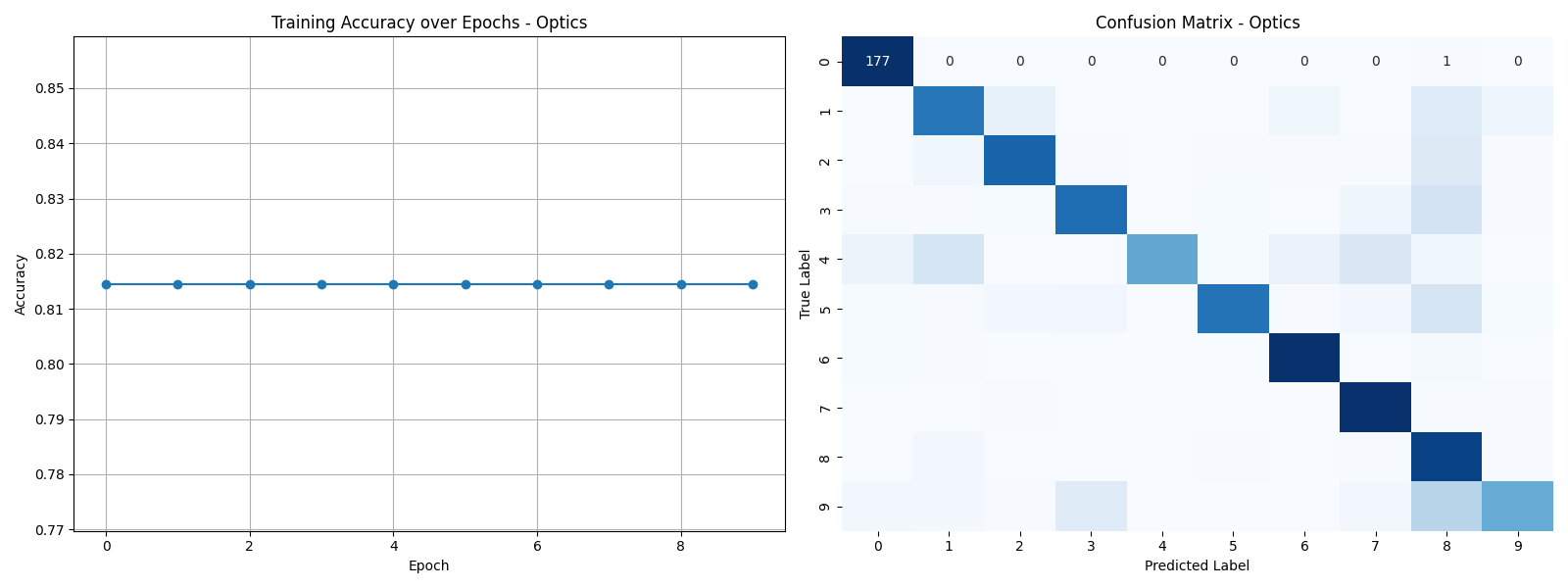
* **Training Accuracy:** 98%
* **Test Accuracy:** 92%
* **Confusion Matrix:**

****

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:**

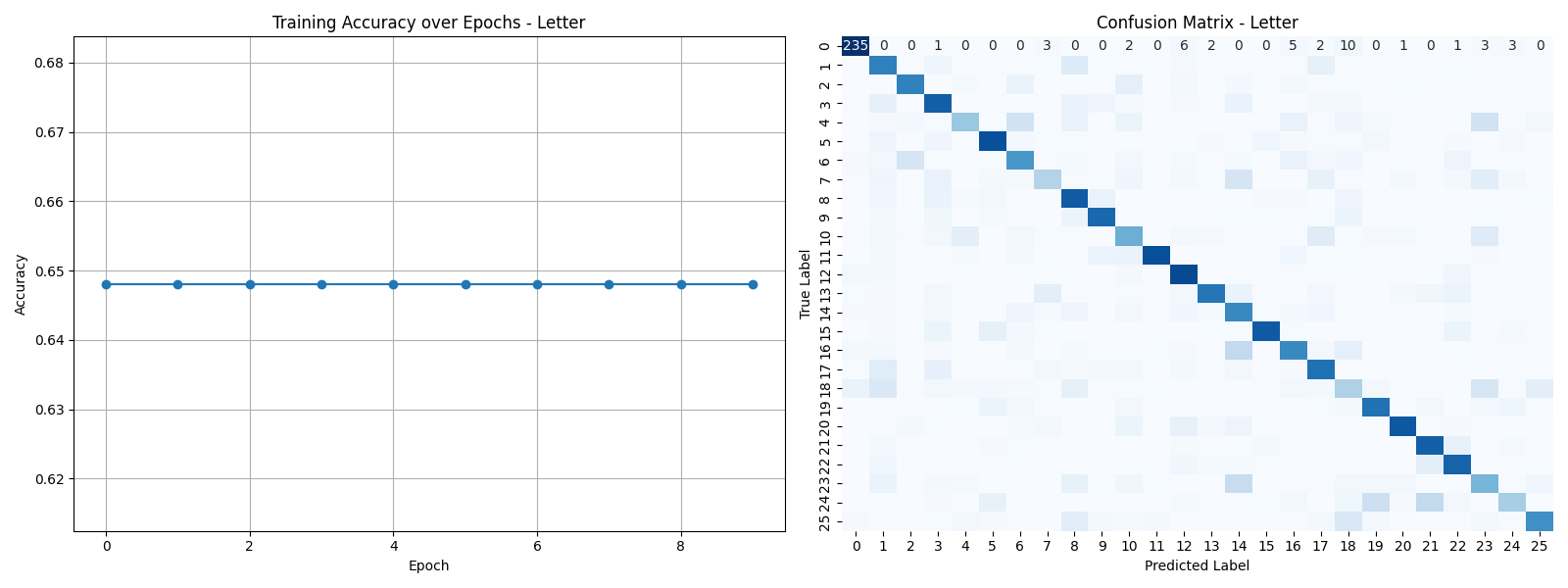


Despite a relatively high training accuracy, the test accuracy of the classifier on the Optics dataset was lower compared to the other datasets, achieving around 78.63%. The confusion matrix for this dataset was large, indicating a more complex classification task.

### Dataset: Letter

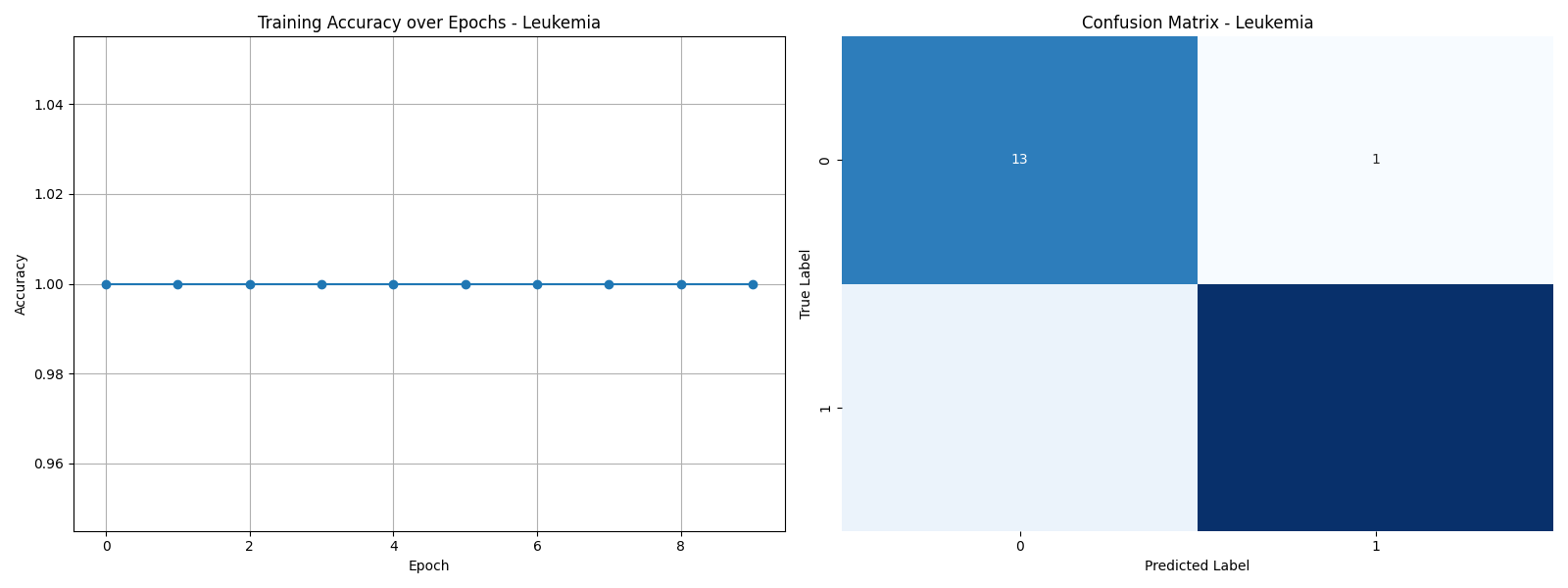
* **Training Accuracy:** 64.81%
* **Test Accuracy:** 63.16%
* **Confusion Matrix:**

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



### Dataset: Leukemia

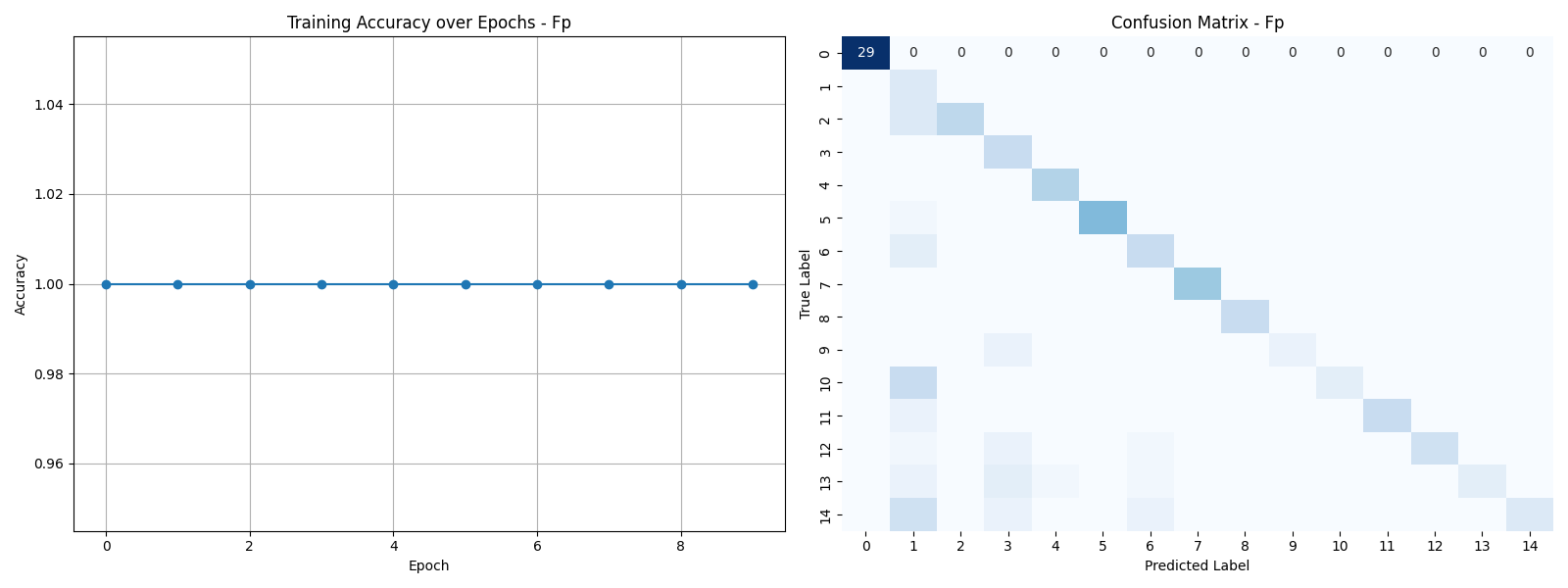
* **Training Accuracy:** 100%
* **Test Accuracy:** 91.18%
* **Confusion Matrix:**



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.