Naïve Bayes

**Nguyen Thanh Phat**

B2005853

Contents

[1. Naive Bayes: Dataset Golf 1](#_Toc999603107)

[Each case of Probability: 1](#_Toc242598057)

[Using likelihood for each case: 1](#_Toc1063679391)

[Result 1](#_Toc525550038)

[2. Naive Bayes Numerical features: Dataset Golf 2](#_Toc326551366)

[Each case of Probability: 2](#_Toc1360370751)

[Using likelihood for each case: 2](#_Toc1654550182)

[Result 3](#_Toc1340268422)

[3. Implement the program using GaussianNB in scikit-learn library. 3](#_Toc308032695)

[Source code: 3](#_Toc1563140167)

[Dataset: Iris 7](#_Toc2134167780)

[Dataset: Optics 7](#_Toc2066918988)

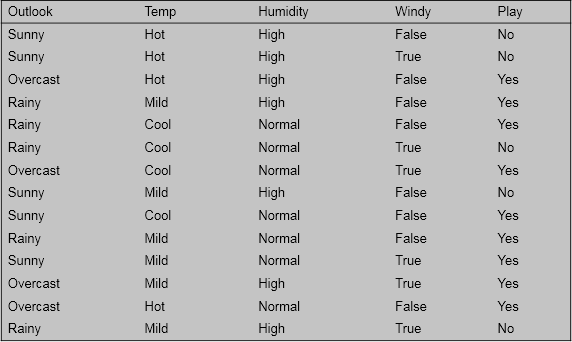
[Dataset: Letter 8](#_Toc1430477858)

[Dataset: Leukemia 8](#_Toc1828492199)

[Dataset: Fp 9](#_Toc1040782781)

[Dataset: Fp107 10](#_Toc1331284794)

## 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 6 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:

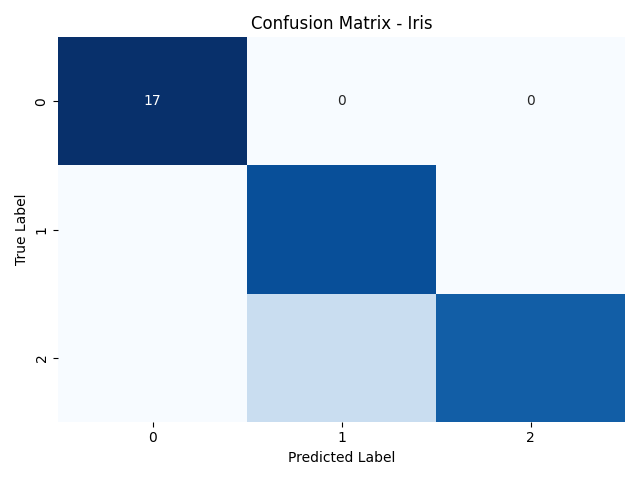
The directory for Part 3:

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

|  |
| --- |
| 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 export\_confusion\_matrix(confusion, dataset\_name):  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 print\_confusion\_matrix(confusion):  for row in confusion:  print(row)  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 test\_model(clf, X\_test, y\_test):  y\_pred = clf.predict(X\_test)  # Calculate accuracy  accuracy = accuracy\_score(y\_test, y\_pred)  return accuracy, y\_pred  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  clf.fit(X\_train, y\_train)  # Predict on test  train\_accuracy, \_ = test\_model(clf, X\_train, y\_train)  accuracy, y\_pred = test\_model(clf, X\_test, y\_test)  print("Train Accuracy:", train\_accuracy)  print("Test Accuracy:", accuracy)  # Calculate confusion matrix  confusion = confusion\_matrix(y\_test, y\_pred)  print("Confusion Matrix:")  print\_confusion\_matrix(confusion)  # Save into file  export\_confusion\_matrix(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",  },  {  "name": "Fp017",  "train\_file": "data//fp107//fp107.trn",  "test\_file": "data//fp107//fp107.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") |

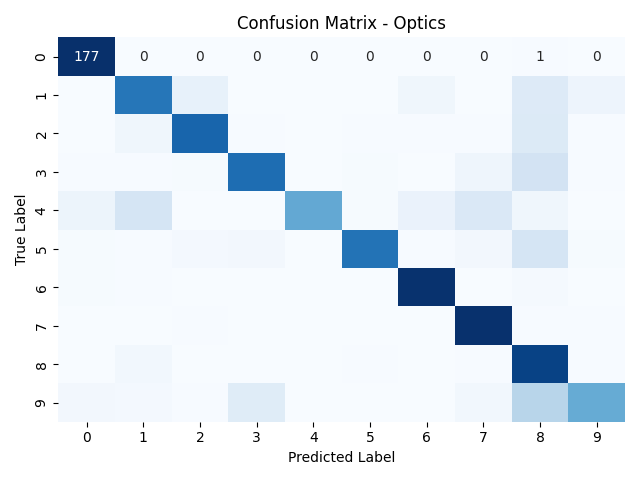
### Dataset: Iris

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



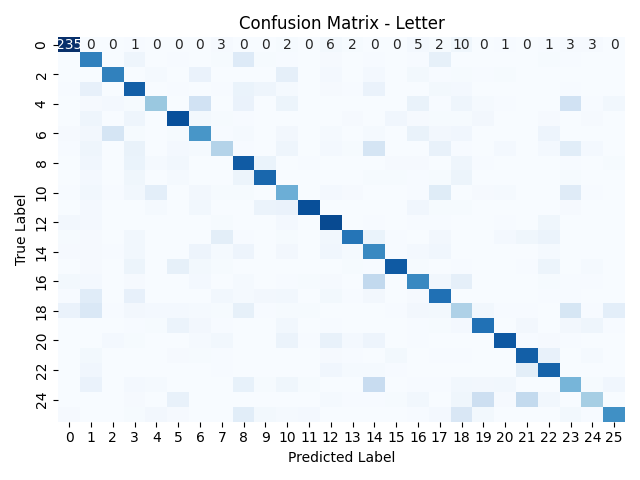
### Dataset: Optics

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



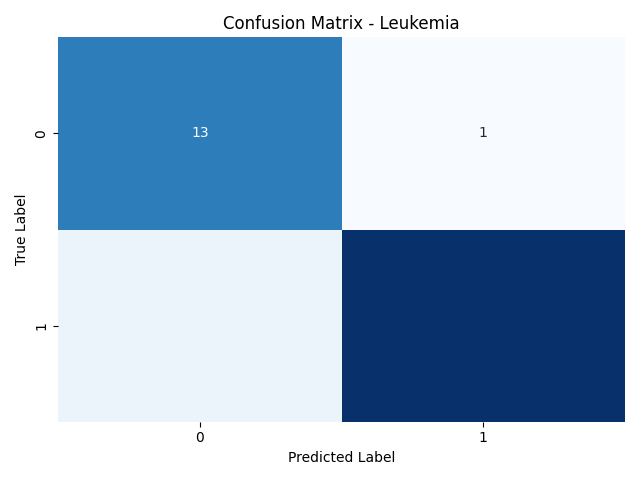
### Dataset: Letter

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



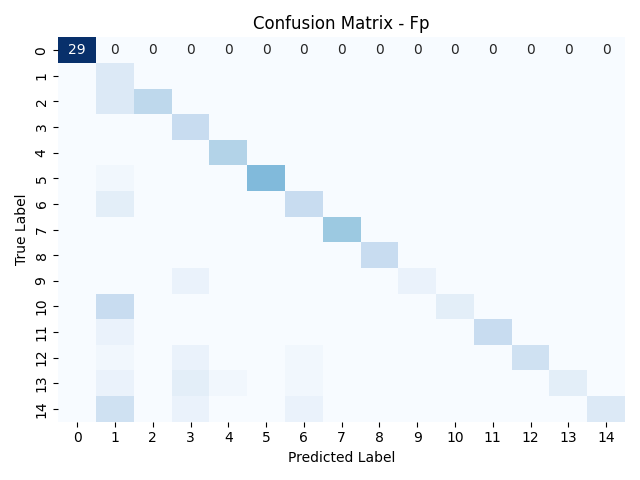
### Dataset: Leukemia

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



### Dataset: Fp

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



### Dataset: Fp107

* **Training Accuracy:** 100%
* **Test Accuracy:** 96.816%
* **Confusion Matrix:**

