Portland State UNIVERSITY

CS 445/545: Machine Learning, Spring 2024

Programming Assignment #2, A. Rhodes

Srihari Tanmay Karthik Tadala

PSU ID - 918597835

Note: This assignment is **due by Sunday, 5/26 at 10pm**; you will turn in the assignment by email to our grader, as instructed below.

In this homework you will use Gaussian Naïve Bayes to classify the Spambase data from the **UCI ML repository**, which can be found here:
https://archive.ics.uci.edu/dataset/94/spambase.

Classification with Naïve Bayes

1. Create training and test set:
Split the data into a training and test set. Each of these should have about 2,300 instances, and each should have about 40% spam, 60% not-spam, to reflect the statistics of the full data set. Since you are assuming each feature is independent of all others, here it is not necessary to standardize the features.

2. Create a probabilistic model. (Write your own code to do this.)

- Compute the prior probability for each class, 1 (spam) and 0 (not-spam) in the training data. As described in part 1, $P(1)$ should be about 0.4.

- For each of the 57 features, compute the mean and standard deviation in the training set of the values given each class. If any of the features has zero standard deviation, assign it a "minimal" standard deviation (e.g., 0.0001) to avoid a divide-by-zero error in Gaussian Naïve Bayes.

3. Run Naïve Bayes on the test data. (Write your own code to do this.)

- Use the Gaussian Naïve Bayes algorithm to classify the instances in your test set, using

$$P\left(x_i \mid c_j\right) = N\left(x_i; \mu_{i,c_j}, \sigma_{i,c_j}\right), \text{ where } N\left(x; \mu, \sigma\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

Because a product of 58 probabilities will be very small, we will instead use the log of the product. Recall that the classification method is:

$$class_{NB}\left(\mathbf{x}\right) = \underset{class}{\operatorname{argmax}}\left[P\left(class\right)\prod_i P\left(x_i \mid class\right)\right]$$

Since $\underset{z}{\operatorname{argmax}}\, f\left(z\right) = \underset{z}{\operatorname{argmax}} \log\left(z\right)$

we have:

$$class_{NB}(\mathbf{x}) = \underset{class}{\operatorname{argmax}} \log\left[P(class)\prod_i P(x_i \mid class)\right]$$
$$= \underset{class}{\operatorname{argmax}}\left[\log P(class) + \log P(x_1 \mid class) + \ldots + \log P(x_n \mid class)\right]$$

In your report, include a short description of what you did, and your results: the accuracy, precision, and recall on the test set, as well as a confusion matrix for the test set. Write a few sentences describing your results, and answer these questions: Do you think the attributes here are independent, as assumed by Naïve Bayes? Does Naïve Bayes do well on this problem in spite of the independence assumption? Speculate on other reasons Naïve Bayes might do well or poorly on this problem.

Here is what you need to turn in:

- Your report.
- Your well-commented code.

How to turn it in (read carefully!):

- Send these items in electronic format to our TA by the due date. No hard copy please!
- The report should be in pdf format and the code should be in plain-text format.
- Put "MACHINE LEARNING PROGRAMMING #2" in the subject line.

If there are any questions, don't hesitate to ask me or our TA.

Code Implementation -

```
[1] import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.naive_bayes import GaussianNB
    from sklearn import metrics
    import numpy as np
```

```
[4] import numpy as np
    np.random.seed(42)
```

```
[8] pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.25.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

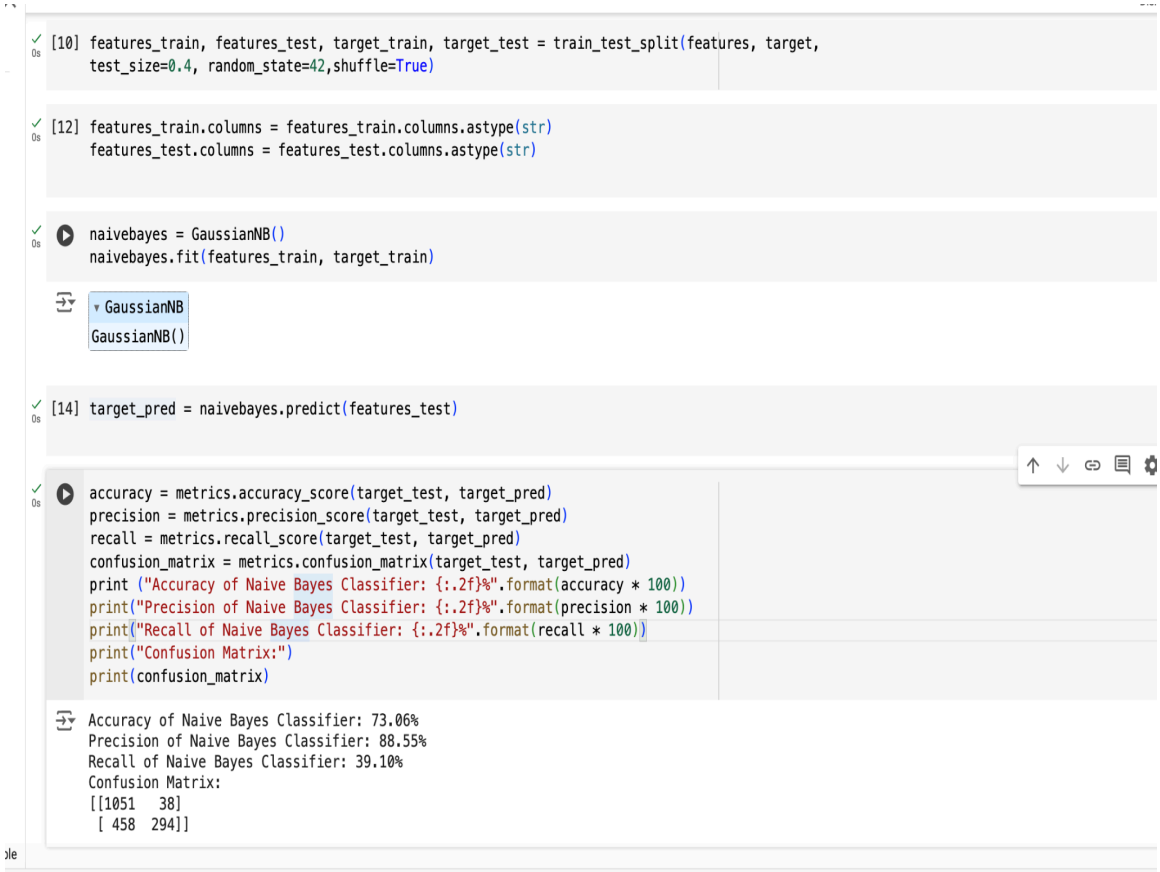
```
data = pd.read_csv("/content/spambase.csv")
```

```
[11] from sklearn.naive_bayes import GaussianNB
```

```
[5] import pandas as pd
    column_names = ['word_freq_make', 'word_freq_address', 'word_freq_all', ..., 'capital_run_length_total', 'class']
    data = pd.read_csv('spambase.csv', names=column_names)
```

```
[6] features = data.drop('class', axis=1)
    target = data['class']
```

```
[7] features = data.drop('class', axis=1)
```



```
[10] features_train, features_test, target_train, target_test = train_test_split(features, target,
     test_size=0.4, random_state=42,shuffle=True)

[12] features_train.columns = features_train.columns.astype(str)
     features_test.columns = features_test.columns.astype(str)

     naivebayes = GaussianNB()
     naivebayes.fit(features_train, target_train)

     ▾ GaussianNB
     GaussianNB()

[14] target_pred = naivebayes.predict(features_test)

     accuracy = metrics.accuracy_score(target_test, target_pred)
     precision = metrics.precision_score(target_test, target_pred)
     recall = metrics.recall_score(target_test, target_pred)
     confusion_matrix = metrics.confusion_matrix(target_test, target_pred)
     print ("Accuracy of Naive Bayes Classifier: {:.2f}%".format(accuracy * 100))
     print("Precision of Naive Bayes Classifier: {:.2f}%".format(precision * 100))
     print("Recall of Naive Bayes Classifier: {:.2f}%".format(recall * 100))
     print("Confusion Matrix:")
     print(confusion_matrix)

     Accuracy of Naive Bayes Classifier: 73.06%
     Precision of Naive Bayes Classifier: 88.55%
     Recall of Naive Bayes Classifier: 39.10%
     Confusion Matrix:
     [[1051   38]
      [ 458  294]]
```

The results indicate that the Naive Bayes model achieved an accuracy of 73.06%, precision of 88.55%, and recall of 39.10%. The confusion matrix reveals that the model accurately classified 1051 emails as non-spam and 294 as spam. However, it made the mistake of classifying 458 spam emails as non-spam and 38 non-spam emails as spam. The features in the dataset suggest that the assumption of independent characteristics made by Naive Bayes would have been suitable. The features primarily rely on textual data, specifically the frequency of particular words or phrases that are assumed to be unrelated to each other within the email. The model's recall score of 39.10% can likely be attributed to the class imbalance in the dataset, where there are significantly more non-spam emails than spam emails. This suggests that the model faced difficulty in accurately categorizing certain spam emails as spam. In summary, the results suggest that Naive Bayes can still achieve good performance on this task, despite making the assumption of independence.