Spam Email Classification

NAIVE BAYES CLASSIFIER

Data analysis

By:

Rawan Alharthi	**444001029**
Mashael Abdali	**444001062**
Shaimaa Alghamdi	**444000746**

Objective

The objective of this project is to build and evaluate a Naive Bayes classifier for predicting spam emails. The classifier uses the Gaussian Naive Bayes approach to calculate the probability of an email being spam or not based on the features in the dataset.

Data Processing

Loading the Dataset:

The dataset was loaded from the (Spambase) dataset.

Splitting Features and Target:

- **Features (X): All columns except the last one.**
- **Target (y): The label (spam or not spam) in the last column.**

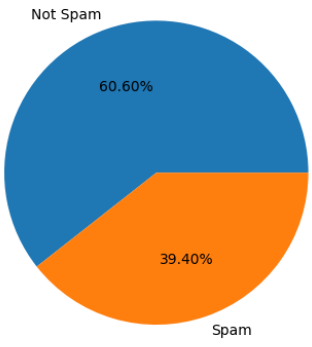
Label Encoding:

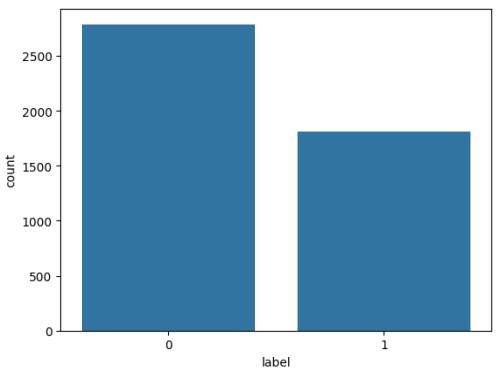
Since the label is already binary (0 for non-spam and 1 for spam), no major transformations were needed.

```
[ ]  # 0 = Not Spam , 1 = Spam
     labels = {0 : "Not Spam", 1 : "Spam"}
     label_counts = data['label'].value_counts()
     print(label_counts)
```

```
label
0    2788
1    1813
Name: count, dtype: int64
```

Data Visualization:





Removing Duplicate Emails:

This process involves identifying and deleting duplicate email addresses from a dataset to ensure each email is unique.

```
[ ]  # Removing Duplicate Emails
     print(f"Number of duplicated  : {np.sum(data.duplicated())}")

     data.drop_duplicates(inplace = True)

     print(f"Number after removing duplicates : {np.sum(data.duplicated())}")
     print(data.shape)
```

```
Number of duplicated  : 391
Number after removing duplicates : 0
(4210, 58)
```

```
[ ]  # After Removing Duplicate Emails
     label_counts = data['label'].value_counts()
     print(label_counts)
```

```
label
0    2531
1    1679
Name: count, dtype: int64
```

Data Splitting:

(The dataset was split into a training set (80%) and a test set (20%).

Naive Bayes Model Building

Prior Calculation:

Calculated the prior probabilities for each class (spam or not spam) based on the proportion of labels in the training data.

Gaussian Likelihood Calculation:

Estimated the mean and variance for each feature in each class to compute the Gaussian likelihood.

```
[ ]  # Naive Bayes Model Building
     classes = np.unique(y)

     n_samples, n_features = X.shape
     n_classes = len(classes)

     # Prior Probabilities
     priors = {}
     for class_label in classes:
         priors[class_label] = np.mean(y == class_label)
```

```
[ ]  means = {}
     variances = {}

     for class_label in classes:
         X_class = X[y == class_label]
         means[class_label] = np.mean(X_class, axis=0)
         variances[class_label] = np.var(X_class, axis=0) + 1e-9

     def gaussian_likelihood(x, mean, var):
         return (1 / np.sqrt(2 * np.pi * var)) * np.exp(- (x - mean) ** 2 / (2 * var))
```

Posterior Calculation:

Used Bayes' theorem to compute posterior probabilities for each class and selected the class with the highest posterior.

Performance Evaluation

Accuracy:

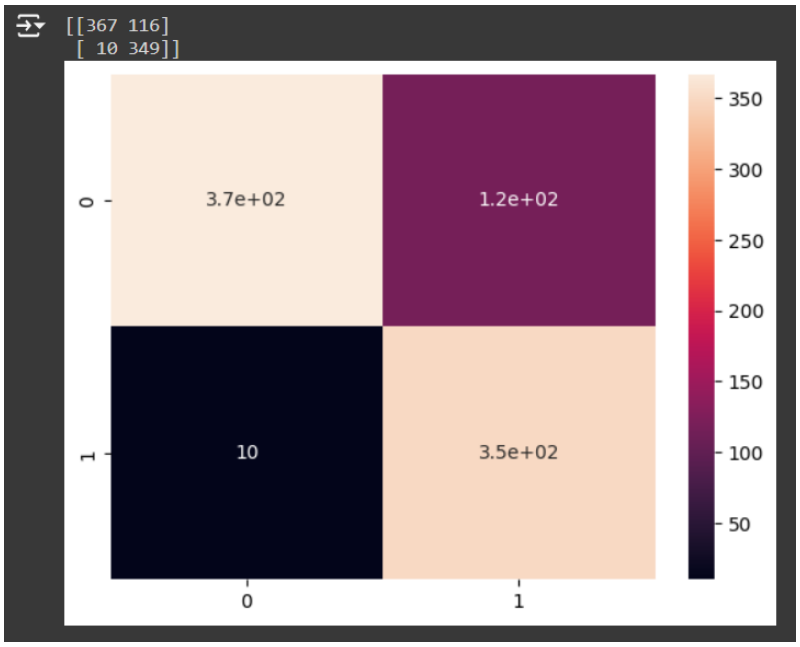
The model achieved an accuracy of (0.85035) which means that out of 921 predictions, approximately 85% were correct.

```
[ ]  # Accuracy
     accuracy = accuracy_score(y_test, predictions)
     print("Accuracy  : ", accuracy)

     Accuracy  :  0.850356294536817
```

Confusion Matrix:

- **True Negatives (367): The model correctly predicted 367 emails as not spam (class 0).**
- **False Positives (116): The model incorrectly predicted 116 emails as spam when they were actually not spam.**
- **False Negatives (10): The model incorrectly predicted 10 emails as not spam when they were actually spam.**
- **True Positives (349): The model correctly predicted 349 emails as spam (class 1).**



Classification Report:

Precision:

- **Class 0 (Not Spam): Precision of 97% indicates that when the model predicted not spam, it was correct 97% of the time.**
- **Class 1 (Spam): Precision of 75% indicates that when the model predicted spam, it was correct 75% of the time.**

Recall:

- **Class 0 (Not Spam): A recall of 76% indicates that out of all actual non-spam emails, the model correctly identified 76%.**
- **Class 1 (Spam): A recall of 97% shows that the model was very good at detecting spam emails, correctly identifying 97% of actual spam emails.**

F1-score:

- **Class 0 (Not Spam): F1-score of 83%, which balances precision and recall for non-spam.**
- **Class 1 (Spam): F1-score of 82%, indicating strong performance for spam detection.**

```
[ ]  clf_report = metrics.classification_report(y_test, predictions)
     print(clf_report)
```

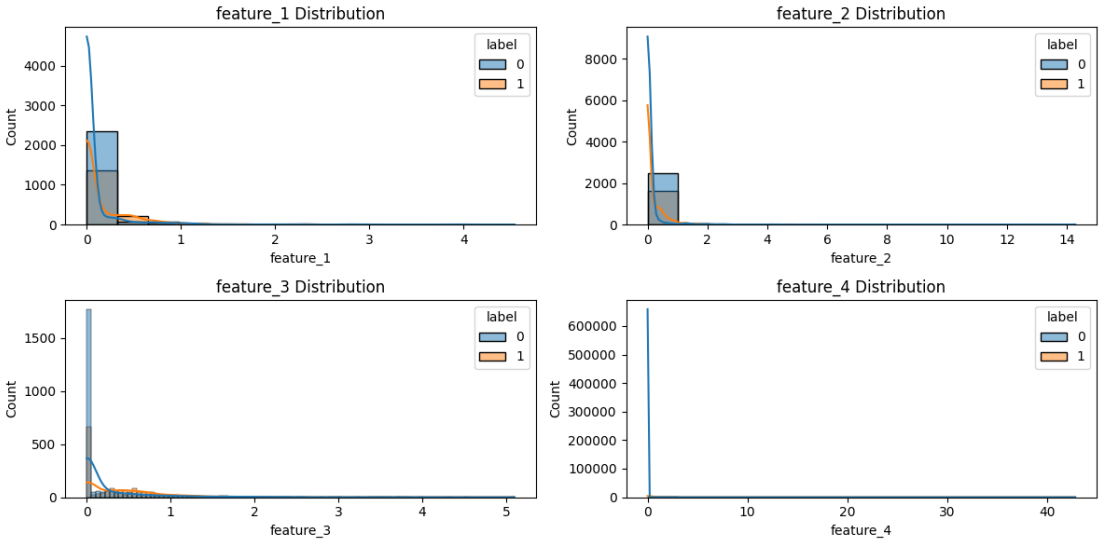
```
              precision    recall  f1-score   support

           0       0.97      0.76      0.85       483
           1       0.75      0.97      0.85       359

    accuracy                           0.85       842
   macro avg       0.86      0.87      0.85       842
weighted avg       0.88      0.85      0.85       842
```

Visualizing Feature Distribution Across Classes:

Creates a 2x2 grid of histograms to visualize the distribution of the first four features in the dataset, segmented by the label column. Each plot includes a density curve, making it easier to compare how these features vary across different classes (spam vs. not spam).



Conclusion

• We were able to build a model capable of classifying emails as 'spam' or 'not spam' with an accuracy of 85%.

• The data was prepared and cleaned by removing duplicate entries, followed by splitting the data into a training set and a test set.

• The results showed that the model was generally effective, especially in classifying non-spam messages.