RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CLASSIFYING SMS SPAM USING NAIVE BAYES CLASSIFIER

CS19411-PYTHON PROGRAMMING FOR MACHINE LEARNING

J JEBISHA JOSE

220901035

3RD YEAR EEE-A

ABSTRACT

The surge in SMS usage due to the expanding mobile user base has led to a rise in unsolicited messages, commonly known as spam. This increase in spam messages, aimed at promoting businesses or extracting sensitive information like credit card details, has underscored the importance of combating spam. To address this issue, various machine learning techniques have been employed to detect and filter out SMS spam. These methods leverage comprehensive datasets, such as those provided by the University of California, Irvine (UCI), to train and validate models capable of distinguishing between legitimate messages and spam. To classify the messages as spam or not spam we are using machine learning (the multinomial naïve Bayes classifier algorithm) and Count Vectorizer by Scikit-learn library in python programming .First, we collect the datasets and convert them into numerical data (matrix) by Count Vectorizer and then we apply the Naïve Bayes algorithm on datasets for classification purposes.

Overview

This project focuses on leveraging artificial intelligence (AI) and machine learning (ML) to address the pervasive issue of SMS spam. With smartphones being essential for communication, online shopping, and social media, the reliance on SIM cards with internet connectivity has increased. However, this connectivity also exposes users to numerous unwanted and potentially harmful text messages that attempt to steal confidential information or deceive with false promises of prizes.

The SMS industry significantly impacts the economy, contributing 11 to 13 percent of the Gross National Income (GNI) in developing countries as of 2013, with these figures continuing to rise. Despite the availability of various communication platforms, many people prefer SMS due to its minimal cost. Alarmingly, over 30 percent of spam messages originate from scammers in Asian countries, leading to many individuals falling victim to these deceptive practices.

This project will explore the current landscape of SMS spam and examine the methods used by scammers. By integrating AI and ML, we aim to enhance existing anti-spam technologies and develop innovative strategies to reduce spam messages. These advanced technologies will help in accurately identifying and filtering out spam, thereby protecting users from potential fraud and enhancing their overall communication experience.

PROGRAM AND OUTPUT

importing modules

```
!pip install scikit-plot
```

```
Collecting scikit-plot
  Downloading scikit_plot-0.3.7-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: matplotlib>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (3.8.0)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (1.5.2)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (1.13.1)
Requirement already satisfied: joblib>=0.10 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.4.7)
Requirement already satisfied: numpy<2,>=1.21 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (2.8.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18->scikit-plot) (3.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=1.4.0->scikit-plot) (1.16.0)
Downloading scikit_plot-0.3.7-py3-none-any.whl (33 kB)
Installing collected packages: scikit-plot
Successfully installed scikit-plot-0.3.7
```

```
!pip install scipy==1.7.3
!pip install scikit-plot --upgrade
```

```
Collecting scipy==1.7.3
  Downloading scipy-1.7.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.2 kB)
Collecting numpy<1.23.0,>=1.16.5 (from scipy==1.7.3)
  Downloading numpy-1.22.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.0 kB)
Downloading scipy-1.7.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (39.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 39.9/39.9 MB 17.2 MB/s eta 0:00:00
Downloading numpy-1.22.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.8/16.8 MB 70.5 MB/s eta 0:00:00
Installing collected packages: numpy, scipy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.26.4
    Uninstalling numpy-1.26.4:
      Successfully uninstalled numpy-1.26.4
  Attempting uninstall: scipy
    Found existing installation: scipy 1.13.1
    Uninstalling scipy-1.13.1:
      Successfully uninstalled scipy-1.13.1
```

```
[ ]  import pandas as pd
     import matplotlib.pyplot as plt
     import scikitplot as skplt

     #import scikitplot as skplt # This should import successfully now
     from sklearn.model_selection import train_test_split as tts
     from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.naive_bayes import MultinomialNB
     from sklearn.metrics import accuracy_score, precision_score, recall_score , confusion_matrix, classification_report
```

creating the data frame

```
[ ]  df = pd.read_csv('/content/SMSSpamCollection.unknown', sep='\t', names=['label', 'messages'])
```

```
df
```

	label	messages
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

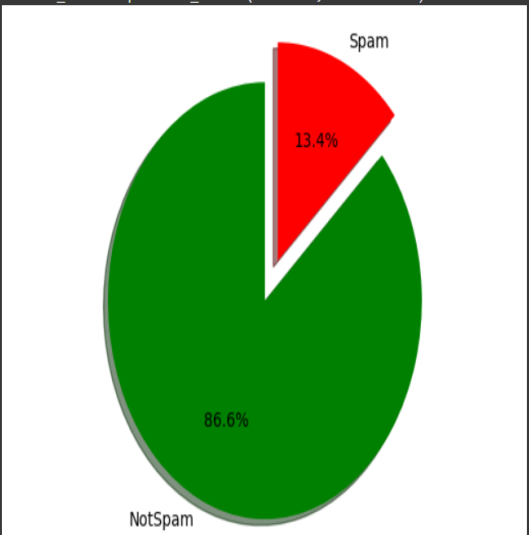
```
[ ]  df['label'] = df.label.map({'ham' : 0, 'spam' : 1})
```

```
[ ]  count_Class = pd.value_counts(df.label, sort = True)

     # Data to Plot
     labels = 'NotSpam', 'Spam'
     sizes = [count_Class[0], count_Class[1]]
     colors = ['green', 'red']
     explode = (0.1, 0.1)

     # Plot
     plt.pie(sizes, explode = explode, labels = labels, colors = colors, autopct = '%1.1f%%', shadow = True, startangle = 90)
     plt.axis('equal')
     plt.show()
```

```
<ipython-input-7-d35143b577c7>:1: FutureWarning: pandas.value_counts is deprecated and will be removed in a future version. Use pd.Series(obj).value_counts() instead.
  count_Class = pd.value_counts(df.label, sort = True)
```



Spliting the data into test and train sets

```
[ ] X_train, X_test, y_train, y_test = tts(df['messages'], df['label'], test_size=0.2, random_state=1)
```

vectorizing the data

```
[ ] count_vector = CountVectorizer()
    train_data = count_vector.fit_transform(X_train)#both learns the vocabulary from the training data and transforms the text data into a matrix of token counts.g
    test_data = count_vector.transform(X_test)
```

Naïve Bayes classifier for multinomial models

1.7.2 Multinomial Naïve Bayes

- MultinomialNB implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification.
- $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$

(y→each class, n→number of features, θ_{yi}→P(x_i |y)).

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

($N_{yi} = \sum_{x \in T} x_i$, $N_y = \sum_{i=1}^{|T|} N_{yi}$, $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. $\alpha = 1$→Laplace smoothing, $\alpha < 1$→ Lidstone smoothing.

```
Mnb = MultinomialNB()
Mnb.fit(train_data, y_train)
```

```
MultinomialNB()
```

Prediction

```
[ ] MnbPredicts = Mnb.predict(test_data)
```

Inspecting our trained model's accuracy, precision, and recall using sklearn.metrics.

```
[ ] from sklearn.metrics import f1_score

    print("The accuracy of our Naïve Bayes multinomial model is {} %".format(accuracy_score(y_test, MnbPredicts) * 100))
    print("The Precision of our Naïve Bayes multinomial model is {} %". format(precision_score(y_test, MnbPredicts)* 100))
    print("The Recall of our Naïve Bayes multinomial model is {} %" . format(recall_score(y_test, MnbPredicts)* 100))
    print("The F1 Score our Naïve Bayes multinomial model is {} %" . format(f1_score(y_test, MnbPredicts)* 100))
```

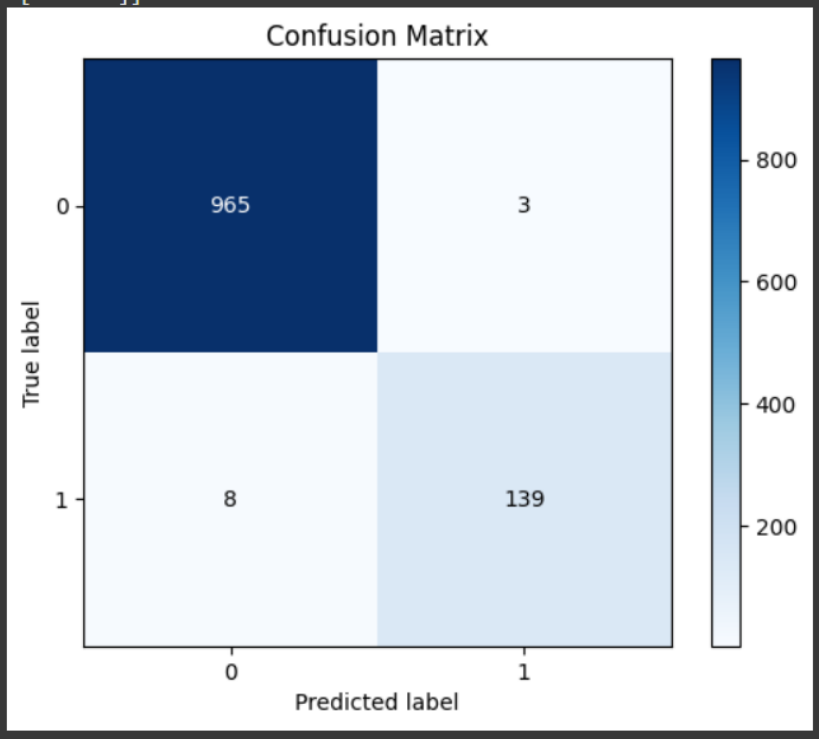
```
The accuracy of our Naïve Bayes multinomial model is 99.01345291479821 %
The Precision of our Naïve Bayes multinomial model is 97.88732394366197 %
The Recall of our Naïve Bayes multinomial model is 94.557823192517 %
The F1 Score our Naïve Bayes multinomial model is 96.19377162629758 %
```

Using **confusion matrix** to observe the performance of model.

```
confusionmatrix = confusion_matrix(y_test, MnbPredicts)
print("The accuracy of Naive Bayes clasifier is {}%".format(accuracy_score(y_test, MnbPredicts) * 100))
print("\n", confusionmatrix)
skplt.metrics.plot_confusion_matrix(y_test, MnbPredicts)
plt.show()
```

```
The accuracy of Naive Bayes clasifier is 99.01345291479821%

 [[965   3]
 [  8 139]]
```



Testing the model with a new Sms/Email massage

+ Code + Text

If the output is 0, it means that the input message is not spam, and if it is 1, it means that message has been spam.

```
[ ] new_test_sample_ham = ["Hi, I'm Mohammad Nabizadeh and I am glad to share the program that I've written with everyone."]
```

```
[ ] new_test_sample_spam= ["hi you got 10 lakh scholarship comment yes to claim"]
```

```
[ ] new_test_sample_ham_vectorized = count_vector.transform(new_test_sample_ham)
```

```
[ ] new_test_sample_spam_vectorized = count_vector.transform(new_test_sample_spam)
```

```
[ ] sample_predict = Mnb.predict(new_test_sample_ham_vectorized)
    sample_predict
```

```
array([0])
```

```
[ ] sample_predict = Mnb.predict(new_test_sample_spam_vectorized)
    sample_predict
```

```
array([1])
```

CONCLUSION

Accuracy : The model achieved an accuracy of 99%, indicating a high overall correctness in classifying messages.

$$\text{Accuracy} = (TP+TN)/(TP+TN+FP+FN)$$

Recall: The recall score was 94%, demonstrating the model's ability to identify actual spam messages.

$$\text{Recall} = TP/TP + FN$$

Precision: The precision score was 97%, suggesting that the majority of messages predicted as spam were indeed spam.

$$\text{Precision} = TP/TP + FP$$

F1 Score : The F1 score, which balances precision and recall, was 96%, indicating a robust performance in distinguishing between spam and non-spam messages.

$$\textbf{F1 Score} = (2*\text{Precision}*\text{Recall})/\text{Precision} + \text{Recall}$$