Experiment 4

Name	Pranay Singhvi
UID	2021300126

AIM: Perform document classification using NB(no use of library of NB classifier)

Copy abstract of papers and create labelled dataset.

Create word count vectors. Each row represents a document, and each column represents a word.

Read text from set of test documents and classify the unlabelled documents.

Generate confusion matrix, and calculate accuracy, precision, recall, F1 score.

Theory

Naive Bayes is a probabilistic algorithm based on Bayes' theorem, which calculates the probability of a hypothesis given observed evidence. In the context of text classification, Naive Bayes is often used to categorize documents into predefined classes or categories based on their content.

Here is a simplified explanation of the Naive Bayes text classification algorithm:

1. **Bayes' Theorem:** Bayes' theorem is the foundation of Naive Bayes. It states:

   $$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

   In the context of text classification:

   - P(A|B) is the probability of document belonging to class A given the observed words in the document.
   - P(B|A) is the probability of observing the words in the document given that the document belongs to class A.
   - P(A) is the prior probability of a document belonging to class A.
   - P(B) is the probability of observing the words in the document (regardless of class).
2. **Naive Assumption:** The "naive" assumption in Naive Bayes is that the features (words in the case of text classification) are conditionally independent given the class label. This means that the presence or absence of a particular word does not affect the presence or absence of any other word. Although this assumption simplifies the model, it may not hold true in reality.

3. **Text Classification:** In text classification, a document is represented as a bag-of-words, which disregards the order of words and focuses only on their frequency. Each word becomes a feature, and the model calculates the probability of a document belonging to each class based on the presence or absence of these words.

   The probability of a document D belonging to class C is calculated as:

   $$P(C|D) \propto P(C) \cdot \prod_{i=1}^{n} P(w_i|C)$$

   - P(C): Prior probability of class C.
   - P(w_i|C) : Probability of word w_i occurring in documents of class C.
4. **Smoothing:** To handle the issue of zero probabilities when a word is not present in the training data for a particular class, smoothing techniques like Laplace smoothing or add-one smoothing are often employed.

5. **Classification Decision:** The class with the highest posterior probability is assigned as the predicted class for the document.

Naive Bayes is computationally efficient, especially with large datasets, and it often performs well in text classification tasks, despite its simplifying assumptions. It's widely used in spam filtering, sentiment analysis, and document categorization.

In [ ]:
```python
import pandas as pd
import numpy as np
from collections import defaultdict
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
```

```
/var/folders/fc/b_3ntmtx3kv_43ckpdzjd47r0000gn/T/ipykernel_46145/2485052407.py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

In [ ]:
```python
data = pd.read_csv('df_file.csv')
```

In [ ]:
```
# Remove punctuation and convert text to lowercase
data['Text'] = data['Text'].str.replace('[^\w\s]', '').str.lower()
```

```
<>:2: SyntaxWarning: invalid escape sequence '\w'
<>:2: SyntaxWarning: invalid escape sequence '\w'
/var/folders/fc/b_3ntmtx3kv_43ckpdzjd47r0000gn/T/ipykernel_46145/2711725740.py:2: SyntaxWarning: invalid escape sequence '\w'
  data['Text'] = data['Text'].str.replace('[^\w\s]', '').str.lower()
```

In [ ]:
```
# Split dataset into training and testing sets
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)
```

In [ ]:
```
#  Implement Naive Bayes classifier
class NaiveBayesClassifier:
    def __init__(self):
        self.class_word_counts = defaultdict(lambda: defaultdict(int))
        self.class_counts = defaultdict(int)
        self.vocab = set()

    def train(self, X, y):
        for i in range(len(X)):
            text = X.iloc[i]
            label = y.iloc[i]
            self.class_counts[label] += 1
            words = text.split()
            for word in words:
                self.class_word_counts[label][word] += 1
                self.vocab.add(word)

    def predict(self, X):
        predictions = []
        probabilities = []
        for i in range(len(X)):
            text = X.iloc[i]
            max_score = float('-inf')
            best_class = None
            words = text.split()
            class_probs = {}
            for label in self.class_counts.keys():
                score = np.log(self.class_counts[label] / sum(self.class_counts.values()))
                for word in words:
                    count = self.class_word_counts[label][word] + 1
                    total_count = len(self.vocab) + sum(self.class_word_counts[label].values())
                    score += np.log(count / total_count)
                class_probs[label] = score
                if score > max_score:
                    max_score = score
                    best_class = label
            predictions.append(best_class)
            probabilities.append(class_probs)
        return predictions, probabilities
```

In [ ]:
```
classifier = NaiveBayesClassifier()
classifier.train(train_data['Text'], train_data['Label'])
```

In [ ]:
```
predictions, probabilities = classifier.predict(test_data['Text'])
```

In [ ]:
```
accuracy = accuracy_score(test_data['Label'], predictions)
precision = precision_score(test_data['Label'], predictions, average='macro')
recall = recall_score(test_data['Label'], predictions, average='macro')
f1 = f1_score(test_data['Label'], predictions, average='macro')
conf_matrix = confusion_matrix(test_data['Label'], predictions)

formatted_accuracy = "{:.2f}".format(accuracy*100)
formatted_precision = "{:.2f}".format(precision*100)
formatted_recall = "{:.2f}".format(recall*100)
formatted_f1 = "{:.2f}".format(f1*100)
print(f"Accuracy: {formatted_accuracy} %")

print(f"Precision: {formatted_precision} %")
print(f"Recall: {formatted_recall} %")
print(f"F1 Score: {formatted_f1} %")
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 96.85 %
Precision: 96.75 %
Recall: 96.81 %
F1 Score: 96.76 %
Confusion Matrix:
[[90  0  1  0  1]
 [ 1 97  0  0  0]
 [ 0  0 75  2  0]
 [ 2  1  1 72  0]
 [ 2  0  3  0 97]]
```

In [ ]:
```
def predict_single(sentence):
    sentence = sentence.lower().strip()
    prediction, probability = classifier.predict(pd.Series([sentence]))
    print("Probabilities:")
    category_mapping = {0: "Politics", 1: "Sport", 2: "Technology", 3: "Entertainment", 4: "Business", 5: "None"}
    for label, prob in probability[0].items():
        print(f"{category_mapping[label]}: {np.exp(prob)}")
    print("Predicted Class:", category_mapping[prediction[0]])

# Cell 10: Test prediction on a single input sentence
input_sentence = input("Enter a sentence: ")
print("Predicting...")
print(input_sentence)
predict_single(input_sentence)
```

```
Predicting...
I like to invest in stock market
Probabilities:
Entertainment: 3.277826610006645e-25
Business: 2.62501793062563e-22
Sport: 3.2523863576518463e-25
Politics: 1.3185633709796263e-24
Technology: 5.528931566700224e-24
Predicted Class: Business
```

Curiosity questions

1. What is the relation between accuracy and precision?

Accuracy and precision are two different metrics used to evaluate the performance of classification models, and they are related in the context of evaluating the correctness of predictions.

1. **Accuracy:**

   - Accuracy is a measure of the overall correctness of the model's predictions.
   - It is calculated as the ratio of correctly predicted instances to the total instances.
   - Formula:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}}$$

2. **Precision:**

   - Precision focuses on the accuracy of the positive predictions made by the model.
   - It is calculated as the ratio of correctly predicted positive instances to the total predicted positive instances.
   - Formula:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

The relationship between accuracy and precision can be understood in the following way:

- Precision is a component of accuracy that specifically considers the positive predictions. If a model has high precision, it means that when it predicts a positive instance, it is likely to be correct. However, a model can have high accuracy even if its precision is not perfect.

- If a model has high accuracy, it generally means that it is making correct predictions overall, both for positive and negative instances. However, a high accuracy does not guarantee a high precision.

In summary, precision is a more focused metric when evaluating the accuracy of positive predictions, while accuracy provides an overall measure of correctness. A model can have high accuracy with varying levels of precision, depending on how well it performs on positive predictions.

2. Give example where precision is significant compared to accuracy?

Precision becomes particularly significant in scenarios where the cost of false positives is high. Here's a common example:

Medical Diagnosis:

Consider a medical test for a rare disease. Let's say the disease occurs in only 1 out of 1000 people (0.1%). If a model predicts a person has the disease, it may be more crucial to ensure that the prediction is correct because the consequences of a false positive (incorrectly diagnosing a healthy person as having the disease) can be severe.

- **Scenario:**
  - Total instances (population): 100,000
  - Actual cases of the disease: 100 (0.1% of the population)
  - Model predicts 200 cases, out of which 150 are correct (True Positives) and 50 are incorrect (False Positives).
- **Metrics:** $$ Accuracy = \frac{150 + 99,850}{100,000} = 99.85

Precision = \frac{150}{150 + 50} = 75 $$ In this example, the accuracy is high (99.85%), but the precision is 75%. The precision indicates that out of all the predicted positive cases, only 75% are true positives. This matters because falsely diagnosing a person with a rare disease can lead to unnecessary stress, treatments, and costs.

In situations like medical diagnosis, fraud detection, or other contexts where the cost of false positives is high, precision becomes a more critical metric than accuracy. It helps to focus on the reliability of positive predictions, ensuring that when the model predicts a positive instance, it is more likely to be correct.

3. Give example where accuracy is significant compared to precision?

Accuracy becomes significant in scenarios where the cost of both false positives and false negatives is relatively balanced, and the goal is to evaluate overall correctness. Here's an example:

Email Spam Detection:

Consider a spam detection system for emails. In this scenario, false positives (classifying a legitimate email as spam) and false negatives (missing a spam email and classifying it as legitimate) both have consequences, but there may not be a strong preference for one over the other.

- **Scenario:**
  - Total emails: 10,000
  - Legitimate emails: 8,000
  - Spam emails: 2,000
  - Model predicts 9,000 emails correctly (True Positives + True Negatives), and 1,000 incorrectly (False Positives + False Negatives).
- **Metrics:**

$$Accuracy = \frac{9,000}{10,000} = 90$$

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

  In this example, accuracy is significant because it provides an overall measure of how well the model is performing. Achieving a high accuracy of 90% indicates that the model is correctly classifying emails, both spam and legitimate, with a high rate of success.

While precision is also important in spam detection to minimize false positives (classifying legitimate emails as spam), accuracy is valuable in situations where false positives and false negatives have comparable consequences. It helps to assess the model's correctness across both classes.

In summary, accuracy is more significant when the goal is to evaluate the model's overall correctness, especially in situations where the costs associated with false positives and false negatives are balanced.

CONCLUSION	In this experiment, I learned to create a Naive Bayes model for classifying specific text into particular classes.