Experiment 6 - Naive Bayes Classifier for Document Classification

April 30, 2023

1 Experiment Details

1.1 Submitted By

Desh Iyer, 500081889, Year III, AI/ML(H), B5

1.2 Problem Statement

Assuming a set of documents that need to be classified, use the naive Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.

1.3 Naive Bayes Classifier for Document Classification

The Naive Bayes Classifier is a probabilistic algorithm used for classification. It is based on Bayes' theorem and the assumption of independence between features. The formula for Bayes' theorem is:

P(A|B) = P(B|A) * P(A) / P(B)

where A and B are events, P(A|B) is the probability of A given B, P(B|A) is the probability of B given A, P(A) is the prior probability of A, and P(B) is the prior probability of B.

In the context of classification, A is the class variable (e.g. spam or not spam), and B is a set of features or attributes (e.g. words in an email). The Naive Bayes Classifier calculates the probability of each class given the features using Bayes' theorem and the assumption of independence between features:

P(class|features) = P(features|class) * P(class) / P(features)

where class is the class variable, features is a set of features, P(class|features) is the probability of the class given the features, P(features|class) is the probability of the features given the class, P(class) is the prior probability of the class, and P(features) is the prior probability of the features.

To classify a new document, the Naive Bayes Classifier calculates the probability of each class given the features and chooses the class with the highest probability.

1.4 Pseudocode

1. Import the required libraries.
2. Load the dataset using pandas `read_csv` method.
3. Create feature vectors using `CountVectorizer` to convert text data into numerical data.

4. Split the dataset into training and testing sets using train_test_split method from `sklearn.model_selection`.
5. Train the Naive Bayesian classifier using `MultinomialNB` method from `sklearn.naive_bayes` with training set.
6. Make predictions on the testing set using predict method of classifier.
7. Calculate accuracy, precision, and recall scores.
8. Print scores.

2 Import Libraries

[ ]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
```

3 Load the Dataset

[ ]:
```
# Load the dataset
data = pd.read_csv('./data/document-classification.txt')
```

4 Clean the Dataset

[ ]:
```
list_1 = []

for i in data['5485']:
    list_1.append(int(i[0]))

data['Target'] = list_1

data = data.rename({'5485': 'Text'}, axis=1)
```

[ ]:
```
data.head()
```

[ ]:

	Text	Target
0	1 champion products ch approves stock split ch…	1
1	2 computer terminal systems cpml completes sal…	2
2	1 cobanco inc cbco year net shr cts vs dlrs ne…	1
3	1 am international inc am nd qtr jan oper shr …	1
4	1 brown forman inc bfd th qtr net shr one dlr …	1

5 Create Vectorizer

[ ]:
```
# create feature vectors using bag of words model
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(data['Text'])
```

6 Train-test Split

[ ]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, data['Target'],␣
↪test_size=0.3, random_state=42)
```

7 Declare `MulitnomialNB` Classifier

[ ]:
```
# Train the classifier
classifier = MultinomialNB()
classifier.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = classifier.predict(X_test)
```

8 Calculate Classifier Scores

[ ]:
```
# Calculate accuracy, precision and recall
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
```

[ ]:
```
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
```

```
Accuracy: 0.9495747266099636
Precision: 0.9518234292669937
Recall: 0.9495747266099636
```
