Text Classification Model Report

M.Junaid 109(4B)

Table of Contents

1. Introduction 1

2. Libraries Used 1

3. Functions and Workflow 2

4. Code Execution and Output 2

5. Output 3

1. Introduction

This Python code implements a text classification model using the Naive Bayes algorithm.

It utilizes a synthetic dataset and processes the text using CountVectorizer for tokenization.

The goal of the model is to classify the input text data into predefined categories based on the content of the text.

The dataset is split into training and testing sets, and the model's performance is evaluated using accuracy and confusion matrix.

2. Libraries Used

- pandas: For reading and handling the dataset.
- numpy: For numerical operations (though not used directly here, it's a common dependency).
- sklearn.model_selection.train_test_split: To split the dataset into training and testing sets.
- sklearn.feature_extraction.text.CountVectorizer: To convert text into numerical feature vectors.
- sklearn.naive_bayes.MultinomialNB: The Naive Bayes model used for text classification.
- sklearn.metrics.accuracy_score, confusion_matrix: To evaluate the model's performance.

3. Functions and Workflow

The workflow of the code involves the following steps:

- Data Loading: The dataset is read using pandas' read_csv function.

```
data = pd.read_csv('synthetic_text_data.csv')
X = data['text']
y = data['label']
```

- Data Preprocessing: The 'text' and 'label' columns are extracted from the dataset for feature and target variables, respectively.
- Data Splitting: The data is split into training and testing sets using `train_test_split` with a test size of 20%.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Feature Extraction: CountVectorizer is used to convert text into a bag-of-words model (token count matrix).

```
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
```

- Model Training: A Multinomial Naive Bayes model is trained on the vectorized training data.

```
model = MultinomialNB()
model.fit(X_train_vectorized, y_train)
```

- Prediction and Evaluation: The model predicts the labels of the test data and the accuracy score is calculated. A confusion matrix is generated to further analyze the performance of the model.

4. Code Execution and Output

The model is executed using the following steps:

- The script first loads and preprocesses the dataset.
- It splits the data into training and testing sets, vectorizes the text, trains the Naive Bayes model, and evaluates it.
- The final output includes the model's accuracy and a confusion matrix to assess performance.

5. Output

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f'Accuracy: {accuracy *100}%')
```

```
Accuracy: 88.23529411764706%
```