Phase 3 project

In this part, you will begin building your project by loading and preprocessing the dataset. In this phase, start building the core components of our spam classifier. Use the Kaggle dataset which contains a collection of SMS messages labelled as spam or non-spam.

**Steps:**

1. Import necessary libraries:

2. Load the dataset:

You can download the dataset from Kaggle and read it into a pandas DataFrame. Assuming you have the dataset in a CSV file named 'spam.csv', you can load it as follows:

This code reads the CSV file and uses 'latin-1' encoding, as the dataset may contain special characters.

3. Explore the dataset:

It's essential to understand the structure of your dataset. You can use the following code to check the first few rows and the basic statistics of the dataset:

4. Data Preprocessing:

Preprocessing is a crucial step in building a spam classifier. You should perform the following tasks:

a. Remove unnecessary columns:

In most cases, you'll only need the 'v1' (label) and 'v2' (text message) columns. You can drop any other columns using the following code:

b. Convert labels to binary values:

Convert 'spam' to 1 and 'ham' (non-spam) to 0. You can do this using a simple mapping:

c. Text preprocessing:

You may want to perform text preprocessing tasks like removing punctuation, converting text to lowercase, and tokenizing the messages. Here's an example of how to do that:

5. Split the dataset into training and testing sets:

You should split your dataset into training and testing sets to evaluate the model's performance. You can use the following code:

6. Now you have a preprocessed dataset ready for building and training your spam classifier model. You can proceed with model selection and training using machine learning or deep learning techniques, depending on your preference.

This code demonstrates the initial steps to load and preprocess the SMS spam dataset using NumPy and pandas. After these preprocessing steps, you can move on to feature engineering and model building to create the spam classifier.

**Code:**

```
import numpy as np

import pandas as pd


# Step 2: Load the dataset

df = pd.read_csv('spam.csv', encoding='latin-1')


# Step 3: Explore the dataset

print(df.head())

print(df.info())
```

```
# Step 4: Data Preprocessing
# a. Remove unnecessary columns
df = df[['v1', 'v2']]

# b. Convert labels to binary values
df['v1'] = df['v1'].map({'spam': 1, 'ham': 0})

# c. Text preprocessing
df['v2'] = df['v2'].str.lower()  # Convert text to lowercase
df['v2'] = df['v2'].str.replace('[^\w\s]', '')  # Remove
punctuation

# Step 5: Split the dataset into training and testing sets
from sklearn.model_selection import train_test_split

X = df['v2']  # Message text
y = df['v1']  # Labels

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

# Now you have a preprocessed dataset ready for building
and training your spam classifier model.
```

# You can proceed with model selection and training using machine learning or deep learning techniques.

1. Import the necessary libraries:

   ```python
   import pandas as pd
   from sklearn.model_selection import train_test_split
   from sklearn.feature_extraction.text import CountVectorizer
   from sklearn.naive_bayes import MultinomialNB
   from sklearn.metrics import accuracy_score, classification_report
   ```

2. Load the dataset:

   Assuming you have downloaded the dataset as 'spam.csv' and it contains two columns: 'label' and 'message', you can load it using pandas:

   ```python
   df = pd.read_csv('spam.csv', encoding='latin-1')
   ```

3. Data exploration:

You can check the first few rows and some basic statistics of the dataset to understand its structure:

```python
print(df.head())
print(df.info())
```

4. Data preprocessing:

- Convert labels to binary values ('spam' as 1, 'ham' as 0):

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

- Text preprocessing (optional): You can perform text cleaning, like converting text to lowercase and removing punctuation, as needed.

5. Split the dataset into training and testing sets:

```python
X = df['message']  # Message text
y = df['label']    # Labels
```

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

6. Feature extraction:

You can use CountVectorizer to convert the text data into numerical features:

```python
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

7. Train the spam classifier (Multinomial Naive Bayes in this case):

```python
clf = MultinomialNB()
clf.fit(X_train, y_train)
```

8. Evaluate the model:

```python
y_pred = clf.predict(X_test)
```

```
   accuracy = accuracy_score(y_test, y_pred)
   report = classification_report(y_test, y_pred)
   print(f'Accuracy: {accuracy}')
   print('Classification Report:')
   print(report)
   ```

This code will load the dataset, preprocess it, convert text messages into numerical features, train a simple spam classifier using Multinomial Naive Bayes, and evaluate the model's performance. You can further fine-tune the model and explore more advanced techniques for better results.

Here's the code for loading and preprocessing the SMS spam dataset using pandas and scikit-learn in a single page:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score,
classification_report


# Step 1: Load the necessary libraries


# Step 2: Load the dataset

df = pd.read_csv('spam.csv', encoding='latin-1')


# Step 3: Data exploration

print(df.head())

print(df.info())


# Step 4: Data preprocessing

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

# You can perform text preprocessing here if needed


# Step 5: Split the dataset into training and testing sets

X = df['message']

y = df['label']
```

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

# Step 6: Feature extraction
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

# Step 7: Train the spam classifier
clf = MultinomialNB()
clf.fit(X_train, y_train)

# Step 8: Evaluate the model
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('Classification Report:')
print(report)
```

Make sure to replace 'spam.csv' with the actual file path to your dataset. This code loads the dataset, preprocesses it, converts text messages to numerical features, trains a Multinomial Naive Bayes spam classifier, and evaluates the model's performance.