Phase 4 project

Steps:

1. **Selecting a Machine Learning Algorithm**: In the code provided, we used the Multinomial Naive Bayes classifier as the machine learning algorithm. This is a common choice for text classification tasks like spam detection. You can experiment with other algorithms like Logistic Regression, Random Forest, or Support Vector Machines to see which one works best for your dataset.

2. **Training the Model**: We used the selected algorithm to train a spam classification model. The `fit` method is used to train the model on the training data.

3. **Evaluating the Model's Performance**: After training, we evaluated the model's performance using various metrics. In the code, we calculated the accuracy, which tells us the proportion of correctly classified messages. We also generated a classification report that provides more detailed metrics like precision, recall, and F1-score. Additionally, we created a confusion matrix to visualize true positives, true negatives, false positives, and false negatives.

The choice of algorithm and evaluation metrics can depend on the specific requirements of your project. You may want to balance precision and recall based on the consequences of false positives and false negatives in your application. The process involves experimenting with different algorithms, hyperparameters, and feature engineering to optimize the model's performance.

Code:

```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, confusion_matrix

# Step 1: Load the dataset
df = pd.read_csv('spam.csv', encoding='latin-1')

# Step 2: Data preprocessing
df['label'] = df['label'].map({'spam': 1, 'ham': 0})

# Step 3: 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 4: Feature extraction
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

# Step 5: Select and train a machine learning
algorithm (Multinomial Naive Bayes)
clf = MultinomialNB()
clf.fit(X_train, y_train)

# Step 6: Evaluate the model
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
```

```
print(f'Accuracy: {accuracy}')
print('Classification Report:')
print(report)
print('Confusion Matrix:')
print(confusion)
```

In this code, we've selected the Multinomial Naive Bayes algorithm. You can experiment with other machine learning algorithms like Logistic Regression, Random Forest, Support Vector Machines, or deep learning models as well. The evaluation includes accuracy, a classification report (including precision, recall, and F1-score), and a confusion matrix to assess the model's performance.

different algorithms and fine-tune your model for better results.