Imports and Helper Functions

In this cell, we import necessary libraries and define functions to read and process the email data.

```
import pandas as pd
import os
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

def read_spam(directory):
    return read_category('spam', directory)

def read_ham(directory):
    return read_category('ham', directory)

def read_category(category, directory):
    emails = []
    for filename in os.listdir(directory):
        if not filename.endswith(".txt"):
            continue
        with open(os.path.join(directory, filename), 'r',
encoding='utf-8', errors='ignore') as fp:
            try:
                content = fp.read()
                emails.append({'name': filename, 'content': content,
'category': category})
            except:
                print(f'skipped {filename}')
    return emails
```

Load Data

In this cell, we load the email data from the specified directories and create a combined DataFrame.

```
# Set directories
spam_directory = r'C:\Users\Kevin\Downloads\Chase 3\enron1\enron1\
spam'
ham_directory = r'C:\Users\Kevin\Downloads\Chase 3\enron1\enron1\ham'
```

```
# Load data
ham = read_ham(ham_directory)
spam = read_spam(spam_directory)

# Create DataFrame
df_ham = pd.DataFrame.from_records(ham)
df_spam = pd.DataFrame.from_records(spam)
df = pd.concat([df_ham, df_spam], ignore_index=True)
```

Data Preprocessing

We need to preprocess the email content to make it suitable for machine learning. This involves normalizing the text by removing non-alphabetic characters and converting it to lowercase.

```
# Data cleaning function
def preprocessor(e):
    e = re.sub(r'[^a-zA-Z\s]', ' ', e)  # Replace non-alphabet
characters with space
    e = e.lower()  # Convert to lowercase
    return e

# Apply preprocessing
df['content'] = df['content'].apply(preprocessor)
```

Train-Test Split and Vectorization

We will split the dataset into training and testing sets, and then use CountVectorizer to transform the text data into numerical features.

```
# Split data into training and testing sets
X = df['content']
y = df['category']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Initialize CountVectorizer with preprocessor
vectorizer = CountVectorizer(preprocessor=preprocessor)
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
```

Model Training and Evaluation

In this cell, we will train a Logistic Regression model using the training data and evaluate its performance on the test data.

```python
# Initialize and train Logistic Regression model
model = LogisticRegression()
model.fit(X_train_vectorized, y_train)

# Make predictions
y_pred = model.predict(X_test_vectorized)

# Evaluate model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

```
Accuracy: 0.9742268041237113
Confusion Matrix:
[[1093   27]
 [  13  419]]
Classification Report:
              precision    recall  f1-score   support

         ham       0.99      0.98      0.98      1120
        spam       0.94      0.97      0.95       432

    accuracy                           0.97      1552
   macro avg       0.96      0.97      0.97      1552
weighted avg       0.97      0.97      0.97      1552
```

Feature Analysis

We will analyze the features learned by the model, including the most significant words for predicting spam and ham emails.

```python
# Top features
features = vectorizer.get_feature_names_out()
coefficients = model.coef_.flatten()
feature_importance = dict(zip(features, coefficients))
```

```python
# Sort and get top features
top_positive_features = sorted(feature_importance.items(), key=lambda
x: x[1], reverse=True)[:10]
top_negative_features = sorted(feature_importance.items(), key=lambda
x: x[1])[:10]

print("\nTop positive coefficients (indicating spam):")
for feature, coef in top_positive_features:
    print(f"{feature}: {coef}")

print("\nTop negative coefficients (indicating ham):")
for feature, coef in top_negative_features:
    print(f"{feature}: {coef}")
```

```
Top positive coefficients (indicating spam):
no: 0.9868691705037805
prices: 0.7949593547093818
http: 0.7894854352272933
hello: 0.7841993001468335
more: 0.7782753722656328
online: 0.7124290238418107
here: 0.7071760852713657
pain: 0.6991255611812542
remove: 0.6745754312123181
paliourg: 0.644643152214869

Top negative coefficients (indicating ham):
attached: -1.5324846375857883
enron: -1.3764900998190654
daren: -1.3001222055677262
thanks: -1.2406397477726665
doc: -1.1946511791319099
deal: -1.1651409278549831
hpl: -1.1163173799897943
pictures: -1.04834692725795
neon: -1.028634449312889
meter: -1.0072682383774425
```

Conclusion

In this notebook, we developed a machine learning model to classify emails into spam and ham categories. We processed the data, trained a Logistic Regression model, and analyzed the most important features for classification. The model achieved high accuracy, indicating its effectiveness in distinguishing between spam and ham emails.