Spam Message Classifier Using Logistic Regression

1. Importing Required Libraries

```python
import pandas as pd
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
```

Explanation:
This section imports the necessary libraries. pandas is used for data manipulation, CountVectorizer is used to convert text data into numerical format, train_test_split is used to divide data into training and testing sets, LogisticRegression is the machine learning model used, and accuracy_score and confusion_matrix are used to evaluate the model.

---

2. Load and Prepare the Data

```python
data = pd.read_csv(r"D:\Uni Data\4th Semester\Programming AI Lab\Programming
data.rename(columns={'v1': 'label', 'v2': 'text'}, inplace=True)
data['label'] = data['label'].map({'ham': 0, 'spam': 1})
```

Explanation:
The dataset is read from a CSV file. The columns are renamed to 'label' for the target variable and 'text' for the message content. The labels are converted to numerical format where 'ham' becomes 0 and 'spam' becomes 1.

---

3. Text Vectorization

```python
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(data['text'])
y = data['label']
```

Explanation:
Text data is transformed into numerical vectors using `CountVectorizer`. `X` contains the vectorized form of the messages, while `y` contains the corresponding labels.

---

4. Splitting the Dataset

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Explanation:
The data is split into training and testing sets. 75% of the data is used for training, and 25% for testing. The `random_state` ensures consistent results each time the code is run.

---

5. Training the Logistic Regression Model

```python
model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)
```

Explanation:
A logistic regression model is created and trained on the training data using the `fit` function.

---

6. Model Evaluation

```python
y_pred = model.predict(X_test)
print("Accuracy_score" ,accuracy_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(f"[[{cm[0,0]} {cm[0,1]}]")
print(f" [{cm[1,0]} {cm[1,1]}]]")
```

Explanation:
The model is used to predict values for the test set. The accuracy score provides a measure of how well the model performed, while the confusion matrix gives detailed performance analysis including true positives, true negatives, false positives, and false negatives.

7. Manual Testing Function (Optional)

```python
def classify_message(message):
    message_vec = vectorizer.transform([message])
    prediction = model.predict(message_vec)[0]
    return "Spam" if prediction == 1 else "Ham"

# Example Usage
print(classify_message("You have won $1000 cash prize!"))
```

Explanation:
This function allows you to test new messages to determine whether they are spam or not. It vectorizes the input message and uses the trained model to make a prediction.