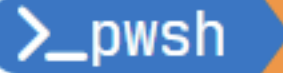
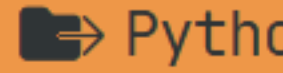


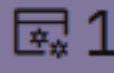


AI Lab\Exp_12\spamDetection.py

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.feature_extraction.text import CountVectorizer
4 from sklearn.naive_bayes import MultinomialNB
5 from sklearn.metrics import accuracy_score, confusion_matrix
6
7 # 1. Load dataset from URL [cite: 59, 60]
8 # The manual uses the SMS Spam Collection dataset from a specific GitHub URL [cite:
9 url = "https://raw.githubusercontent.com/justmarkham/pycon-2016-
10 tutorial/master/data/sms.tsv"
11
12 # 2. Encode labels: ham -> 0, spam -> 1 [cite: 62, 63]
13 data['label_num'] = data.label.map({'ham': 0, 'spam': 1})
14
15 # 3. Split the dataset [cite: 64]
16 # Splitting into training and testing sets (80% train, 20% test) [cite: 65]
17 X_train, X_test, y_train, y_test = train_test_split(
18     data['message'],
19     data['label_num'],
20     test_size=0.2,
21     random_state=42
22 )
23
24 # 4. Convert text messages into feature vectors
25 vectorizer = CountVectorizer()
26 X_train_counts = vectorizer.fit_transform(X_train)
27 X_test_counts = vectorizer.transform(X_test)
28
29 # 5. Train the Naïve Bayes classifier [cite: 68]
30 clf = MultinomialNB() # [cite: 70]
31 clf.fit(X_train_counts, y_train) # [cite: 71]
32
33 # 6. Make predictions on test data [cite: 72]
34 y_pred = clf.predict(X_test_counts) # [cite: 74]
35
36 # 7. Evaluate performance [cite: 75]
37 print("Accuracy:", accuracy_score(y_test, y_pred)) # [cite: 76]
38 print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred)) # [cite: 77]
```

 >_pwsh  Python_LocalVC  master ≡  ?7  1ms

>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\Exp_12\spamDetection.py"







Accuracy: 0.9919282511210762

Confusion Matrix:

```
[[966    0]
 [  9 140]]
```

AI Lab\Exp_13\KNN.py

```
1 from sklearn.datasets import load_iris
2 from sklearn.model_selection import train_test_split
3 from sklearn.preprocessing import StandardScaler
4 from sklearn.neighbors import KNeighborsClassifier
5 from sklearn.metrics import accuracy_score, confusion_matrix
6
7 # 1. Load the dataset
8 iris = load_iris()
9 X, y = iris.data, iris.target
10
11 # 2. Split the dataset
12 # Splitting into training (70%) and testing (30%) sets
13 X_train, X_test, y_train, y_test = train_test_split(
14     X, y, test_size=0.3, random_state=42
15 )
16
17 # 3. Scale features (Standardization)
18 # This ensures all features contribute equally to the distance calculation
19 scaler = StandardScaler()
20 X_train = scaler.fit_transform(X_train)
21 X_test = scaler.transform(X_test)
22
23 # 4. Initialize and Train the KNN Classifier
24 # Using K=3 neighbors
25 knn = KNeighborsClassifier(n_neighbors=3)
26 knn.fit(X_train, y_train)
27
28 # 5. Make predictions on test data
29 y_pred = knn.predict(X_test)
30
31 # 6. Evaluate performance
32 print("Accuracy:", accuracy_score(y_test, y_pred))
33 print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

  >_pwsh  Python_LocalVC  master  ?7  5s 659ms

```
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\Exp_13\KNN.py"
```

Accuracy: 1.0

Confusion Matrix:

```
[[19  0  0]
```

```
[ 0 13  0]
```

```
[ 0  0 13]]
```