AI Lab\Exp_12\spamDetection.py

```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, confusion_matrix

# 1. Load dataset from URL [cite: 59, 60]
# The manual uses the SMS Spam Collection dataset from a specific GitHub URL [cite: 61]
url = "https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/data/sms.tsv"
data = pd.read_csv(url, sep='\t', names=['label', 'message'])

# 2. Encode labels: ham -> 0, spam -> 1 [cite: 62, 63]
data['label_num'] = data.label.map({'ham': 0, 'spam': 1})

# 3. Split the dataset [cite: 64]
# Splitting into training and testing sets (80% train, 20% test) [cite: 65]
X_train, X_test, y_train, y_test = train_test_split(
    data['message'],
    data['label_num'],
    test_size=0.2,
    random_state=42
)

# 4. Convert text messages into feature vectors
vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)

# 5. Train the Naïve Bayes classifier [cite: 68]
clf = MultinomialNB() # [cite: 70]
clf.fit(X_train_counts, y_train) # [cite: 71]

# 6. Make predictions on test data [cite: 72]
y_pred = clf.predict(X_test_counts) # [cite: 74]

# 7. Evaluate performance [cite: 75]
print("Accuracy:", accuracy_score(y_test, y_pred)) # [cite: 76]
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

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# 1. Load the dataset
iris = load_iris()
X, y = iris.data, iris.target

# 2. Split the dataset
# Splitting into training (70%) and testing (30%) sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# 3. Scale features (Standardization)
# This ensures all features contribute equally to the distance calculation
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 4. Initialize and Train the KNN Classifier
# Using K=3 neighbors
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# 5. Make predictions on test data
y_pred = knn.predict(X_test)

# 6. Evaluate performance
print("Accuracy:", accuracy_score(y_test, y_pred))
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]]
```