STEP 1: Import Required Libraries

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_digits
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
```

STEP 2: Load Digits Dataset

```
digits = load_digits()

X = digits.data
y = digits.target

print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

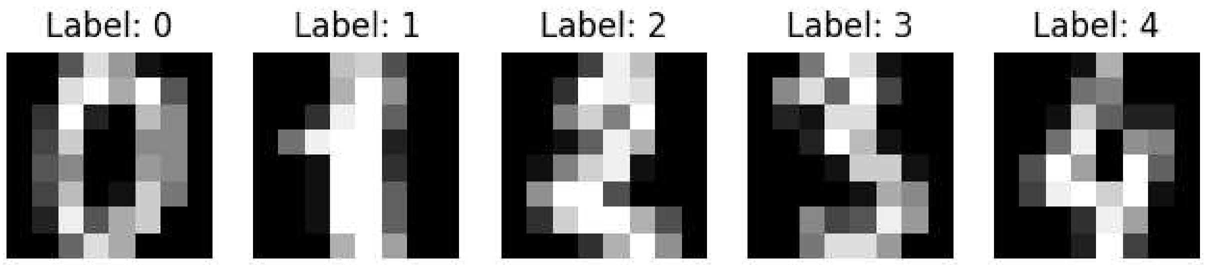
```
Shape of X: (1797, 64)
Shape of y: (1797,)
```

STEP 3: Visualize Few Digit Images

```
plt.figure(figsize=(8,4))

for i in range(5):
    plt.subplot(1,5,i+1)
    plt.imshow(digits.images[i], cmap='gray')
    plt.title("Label: " + str(y[i]))
    plt.axis("off")

plt.show()
```



STEP 4: Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

STEP 5: Feature Scaling (Important for KNN)

```
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

STEP 6: Train KNN with K = 3

```
knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

print("Accuracy (K=3):", accuracy_score(y_test, y_pred))
```

```
Accuracy (K=3): 0.9666666666666667
```

STEP 7: Try Multiple K Values

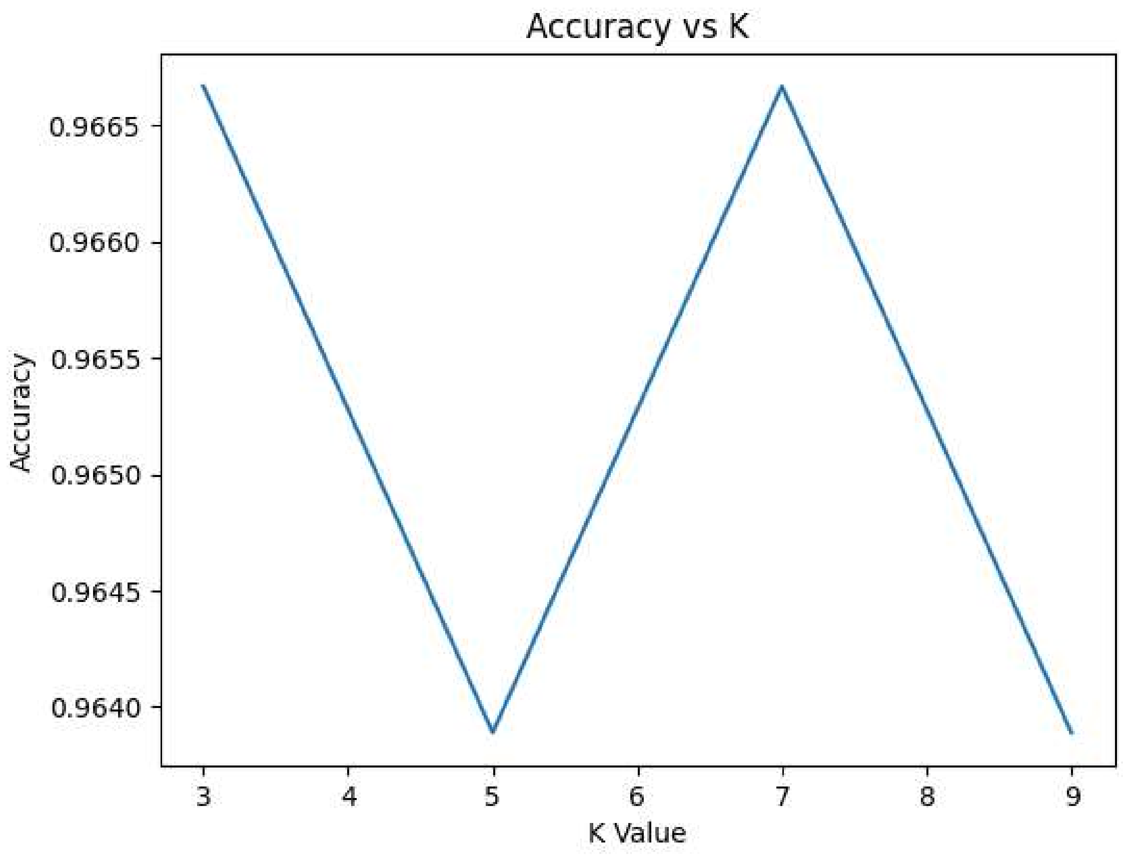
```
k_values = [3, 5, 7, 9]
accuracies = []

for k in k_values:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    acc = accuracy_score(y_test, preds)
    accuracies.append(acc)
    print(f"Accuracy for K={k}: {acc}")
```

```
Accuracy for K=3: 0.9666666666666667
Accuracy for K=5: 0.9638888888888889
Accuracy for K=7: 0.9666666666666667
Accuracy for K=9: 0.9638888888888889
```

STEP 8: Plot Accuracy vs K

```
plt.plot(k_values, accuracies)
plt.xlabel("K Value")
plt.ylabel("Accuracy")
plt.title("Accuracy vs K")
plt.show()
```



STEP 9: Confusion Matrix

```
best_k = k_values[np.argmax(accuracies)]
print("Best K:", best_k)

best_model = KNeighborsClassifier(n_neighbors=best_k)
best_model.fit(X_train, y_train)

y_best_pred = best_model.predict(X_test)

cm = confusion_matrix(y_test, y_best_pred)

print("Confusion Matrix:\n", cm)
```

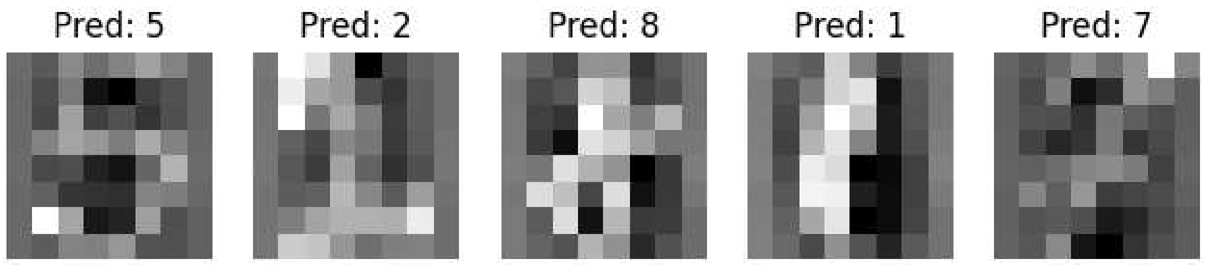
```
Best K: 3
Confusion Matrix:
 [[36  0  0  0  0  0  0  0  0  0]
 [ 0 35  1  0  0  0  0  0  0  0]
 [ 0  0 35  0  0  0  0  0  0  0]
 [ 0  0  1 36  0  0  0  0  0  0]
 [ 0  0  0  0 34  0  0  2  0  0]
 [ 0  0  0  0  0 37  0  0  0  0]
 [ 0  0  0  0  0  0 36  0  0  0]
 [ 0  0  1  0  0  0  0 35  0  0]
 [ 0  3  1  0  0  0  0  0 31  0]
 [ 0  0  0  0  1  0  1  0  1 33]]
```

STEP 10: Display 5 Test Images with Predictions

```
plt.figure(figsize=(8,4))

for i in range(5):
    plt.subplot(1,5,i+1)
    plt.imshow(X_test[i].reshape(8,8), cmap='gray')
    plt.title("Pred: " + str(y_best_pred[i]))
    plt.axis("off")

plt.show()
```


Pred: 5
Pred: 2
Pred: 8
Pred: 1
Pred: 7