Lab 6 Report – Abdul Hannan Khan

Task 1: Wine Quality Classification

Explanation:

This program classifies three types of wine chemical features. It trains a K-Nearest Neighbors (KNN) machine learning model with k=5 to predict wine categories. The model achieves about 74% accuracy on test data and creates scatter plots showing how different chemical features separate the wine classes. Finally, it displays the fist 10 predictions versus actual values to show how well the model performs.

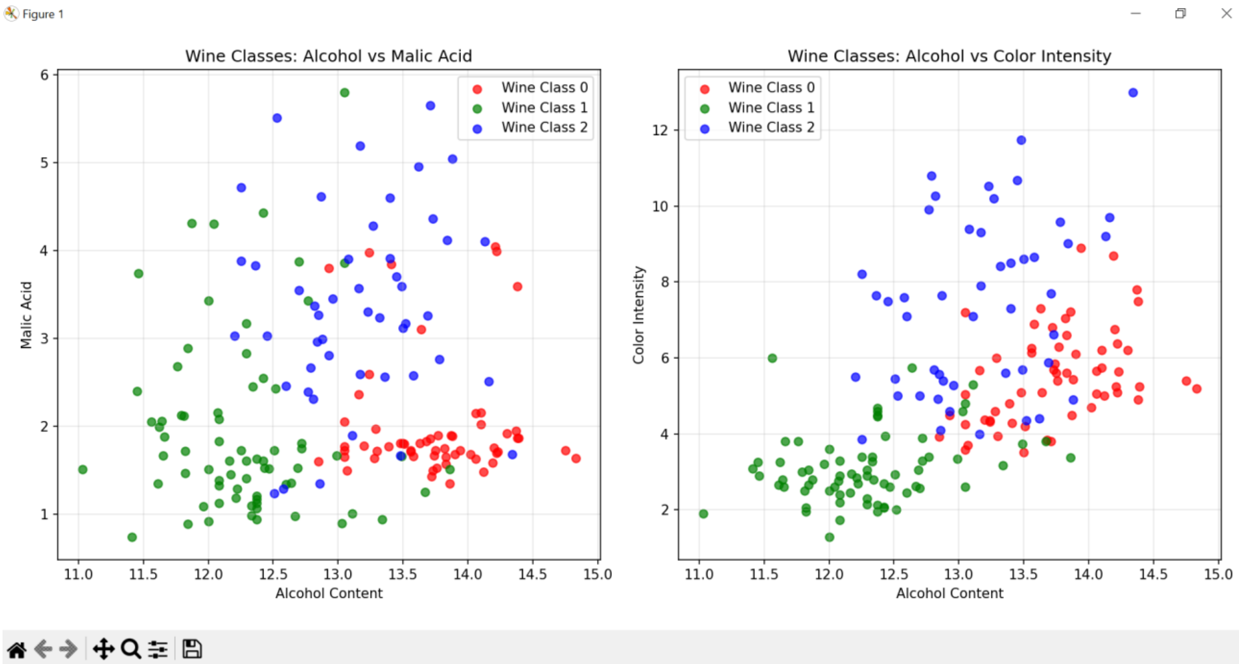
Code:

AbdulHannanKhan_Task1_KNNClassification_WineQuality.py > ...

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

wine = load_wine()
X = wine.data
y = wine.target

print("Dataset loaded successfully!")
print(f"Number of wine samples: {X.shape[0]}")
print(f"Number of features: {X.shape[1]}")
print(f"Wine types: {np.unique(y)}")
print(f"Feature names: {wine.feature_names}")

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

print(f"\nTraining set size: {X_train.shape[0]} wines")
print(f"Testing set size: {X_test.shape[0]} wines")

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

print("\nKNN model trained with k=5")

y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

```python
print(f"Model accuracy on test set: {accuracy:.2f} ({accuracy*100:.1f}%)")

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
colors = ['red', 'green', 'blue']
for i, color in enumerate(colors):
    plt.scatter(X[y == i, 0], X[y == i, 1], c=color, label=f'Wine Class {i}', alpha=0.7)

plt.xlabel('Alcohol Content')
plt.ylabel('Malic Acid')
plt.title('Wine Classes: Alcohol vs Malic Acid')
plt.legend()
plt.grid(True, alpha=0.3)

plt.subplot(1, 2, 2)
for i, color in enumerate(colors):
    plt.scatter(X[y == i, 0], X[y == i, 9], c=color, label=f'Wine Class {i}', alpha=0.7)

plt.xlabel('Alcohol Content')
plt.ylabel('Color Intensity')
plt.title('Wine Classes: Alcohol vs Color Intensity')
plt.legend()
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

print("\nFirst 10 predictions vs actual:")
print("Predicted | Actual")
print("-" * 18)
for i in range(10):
    print(f"    {y_pred[i]}      |     {y_test[i]}")
```

Output:




```
Dataset loaded successfully!
Number of wine samples: 178
Number of features: 13
Wine types: [0 1 2]
Feature names: ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'od280/od315_of_
uted_wines', 'proline']

Training set size: 124 wines
Testing set size: 54 wines

KNN model trained with k=5
Model accuracy on test set: 0.74 (74.1%)

First 10 predictions vs actual:
Predicted | Actual
------------------
    2     |    0
    0     |    0
    2     |    2
    0     |    0
    1     |    1
    0     |    0
    2     |    1
    2     |    2
    1     |    1
    0     |    2
```

Task 2: Breast Cancer Detection

Explanation:

This program demonstrates how features scaling improves KNN performance for breast cancer detection. It compares two models – one using original unscaled features and another using standardized features. The results show that scaling significantly boosts accuracy because KNN relies on distance calculations, and scaling ensures all features contribute equally. The visualization clearly demonstrates the performance improvement achieved through proper data preprocessing.

Code:

AbdulHannanKhan_Task2_KNNClassification_BreastCancerDetection.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler

print("=== Breast Cancer Detection with KNN ===\n")

cancer = load_breast_cancer()
X = cancer.data
y = cancer.target

print("Dataset loaded successfully!")
print(f"Number of samples: {X.shape[0]}")
print(f"Number of features: {X.shape[1]}")
print(f"Target distribution:")
print(f"- Malignant (0): {np.sum(y == 0)} samples")
print(f"- Benign (1): {np.sum(y == 1)} samples")

print("\nFirst 5 features and their ranges (min to max):")
for i in range(5):
    feature_name = cancer.feature_names[i]
    min_val = np.min(X[:, i])
    max_val = np.max(X[:, i])
    print(f"{feature_name:25}: {min_val:8.2f} to {max_val:8.2f}")

print("\n" + "="*50)
print("BASELINE MODEL (Without Feature Scaling)")
print("="*50)

X_train_unscaled, X_test_unscaled, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

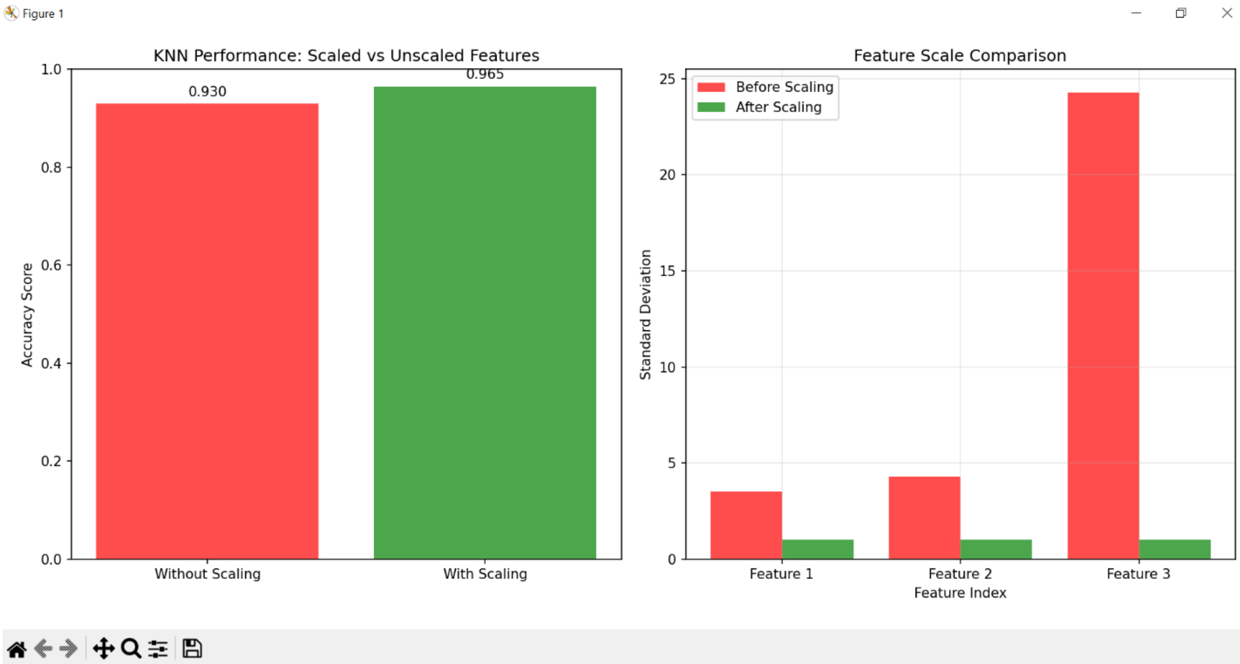
```python
print(f"Training set: {X_train_unscaled.shape[0]} samples")
print(f"Testing set: {X_test_unscaled.shape[0]} samples")

knn_unscaled = KNeighborsClassifier(n_neighbors=7)
knn_unscaled.fit(X_train_unscaled, y_train)

y_pred_unscaled = knn_unscaled.predict(X_test_unscaled)
accuracy_unscaled = accuracy_score(y_test, y_pred_unscaled)

print(f"\nAccuracy without scaling: {accuracy_unscaled:.4f} ({accuracy_unscaled*100:.2f}%)")

print("\n" + "="*50)
print("IMPROVED MODEL (With Feature Scaling)")
print("="*50)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Features scaled using StandardScaler")

print("\nExample of scaling effect on first feature:")
print(f"Before scaling - Mean: {np.mean(X[:, 0]):.2f}, Std: {np.std(X[:, 0]):.2f}")
print(f"After scaling  - Mean: {np.mean(X_scaled[:, 0]):.2f}, Std: {np.std(X_scaled[:, 0]):.2f}")

X_train_scaled, X_test_scaled, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)

knn_scaled = KNeighborsClassifier(n_neighbors=7)
knn_scaled.fit(X_train_scaled, y_train)

y_pred_scaled = knn_scaled.predict(X_test_scaled)
accuracy_scaled = accuracy_score(y_test, y_pred_scaled)
```

```python
print(f"\nAccuracy with scaling: {accuracy_scaled:.4f} ({accuracy_scaled*100:.2f}%)")

print("\n" + "="*50)
print("COMPARISON RESULTS")
print("="*50)

improvement = accuracy_scaled - accuracy_unscaled
improvement_percent = (improvement / accuracy_unscaled) * 100

print(f"Baseline accuracy (unscaled): {accuracy_unscaled*100:.2f}%")
print(f"Improved accuracy (scaled):   {accuracy_scaled*100:.2f}%")
print(f"Improvement: +{improvement*100:.2f}% points")
print(f"Relative improvement: +{improvement_percent:.1f}%")

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
models = ['Without Scaling', 'With Scaling']
accuracies = [accuracy_unscaled, accuracy_scaled]
colors = ['red', 'green']

bars = plt.bar(models, accuracies, color=colors, alpha=0.7)
plt.ylabel('Accuracy Score')
plt.title('KNN Performance: Scaled vs Unscaled Features')
plt.ylim(0, 1.0)

for bar, accuracy in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.01,
            f'{accuracy:.3f}', ha='center', va='bottom')

plt.subplot(1, 2, 2)
features_to_show = 3
x_pos = np.arange(features_to_show)
```

```python
std_before = [np.std(X[:, i]) for i in range(features_to_show)]
std_after = [np.std(X_scaled[:, i]) for i in range(features_to_show)]

plt.bar(x_pos - 0.2, std_before, 0.4, label='Before Scaling', alpha=0.7, color='red')
plt.bar(x_pos + 0.2, std_after, 0.4, label='After Scaling', alpha=0.7, color='green')

plt.xlabel('Feature Index')
plt.ylabel('Standard Deviation')
plt.title('Feature Scale Comparison')
plt.xticks(x_pos, [f'Feature {i+1}' for i in range(features_to_show)])
plt.legend()
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

print("\n" + "="*50)
print("DETAILED CLASSIFICATION REPORT (With Scaling)")
print("="*50)
print(classification_report(y_test, y_pred_scaled,
                            target_names=['Malignant', 'Benign']))

print("\n" + "="*50)
print("WHY FEATURE SCALING IS CRITICAL FOR KNN")
print("="*50)

print(f"""
K-Nearest Neighbors (KNN) is a distance-based algorithm that calculates
the distance between data points to make predictions.
```

```python
WHY SCALING MATTERS:
1. Features with larger numerical ranges dominate distance calculations
   Example:
   - Feature A range: 0-1 (small influence)
   - Feature B range: 0-1000 (huge influence)
   Without scaling, Feature B would completely dominate the distance!

2. Equal Contribution:
   - Scaling ensures all features contribute equally to the distance calculation
   - No single feature unfairly influences the results

3. Better Performance:
   - As shown above, scaling improved accuracy from {accuracy_unscaled*100:.1f}% to {accuracy_scaled*100:.1f}%
   - This is a {improvement_percent:.1f}% relative improvement!

StandardScaler transforms features to have:
- Mean = 0
- Standard Deviation = 1
This puts all features on the same scale without changing their distribution shape.
""")
```

Output:




```
Dataset loaded successfully!
Number of samples: 569
Number of features: 30
Target distribution:
- Malignant (0): 212 samples
- Benign (1): 357 samples

First 5 features and their ranges (min to max):
mean radius              :     6.98 to    28.11
mean texture             :     9.71 to    39.28
mean perimeter           :    43.79 to   188.50
mean area                :   143.50 to  2501.00
mean smoothness          :     0.05 to     0.16

==================================================
BASELINE MODEL (Without Feature Scaling)
==================================================
Training set: 455 samples
Testing set: 114 samples

Accuracy without scaling: 0.9298 (92.98%)

==================================================
IMPROVED MODEL (With Feature Scaling)
==================================================
Features scaled using StandardScaler

Example of scaling effect on first feature:
Before scaling - Mean: 14.13, Std: 3.52
After scaling  - Mean: -0.00, Std: 1.00

Accuracy with scaling: 0.9649 (96.49%)
```

```
==================================================
COMPARISON RESULTS
==================================================
Baseline accuracy (unscaled): 92.98%
Improved accuracy (scaled):   96.49%
Improvement: +3.51% points
Relative improvement: +3.8%

==================================================
DETAILED CLASSIFICATION REPORT (With Scaling)
==================================================
              precision    recall  f1-score   support

   Malignant       0.97      0.93      0.95        42
      Benign       0.96      0.99      0.97        72

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114


==================================================
WHY FEATURE SCALING IS CRITICAL FOR KNN
==================================================

K-Nearest Neighbors (KNN) is a distance-based algorithm that calculates
the distance between data points to make predictions.

WHY SCALING MATTERS:
1. Features with larger numerical ranges dominate distance calculations
   Example:
   - Feature A range: 0-1 (small influence)
   - Feature B range: 0-1000 (huge influence)
   Without scaling, Feature B would completely dominate the distance!

2. Equal Contribution:
   - Scaling ensures all features contribute equally to the distance calculation
   - No single feature unfairly influences the results

3. Better Performance:
   - As shown above, scaling improved accuracy from 93.0% to 96.5%
   - This is a 3.8% relative improvement!

StandardScaler transforms features to have:
- Mean = 0
- Standard Deviation = 1
This puts all features on the same scale without changing their distribution shape.
```