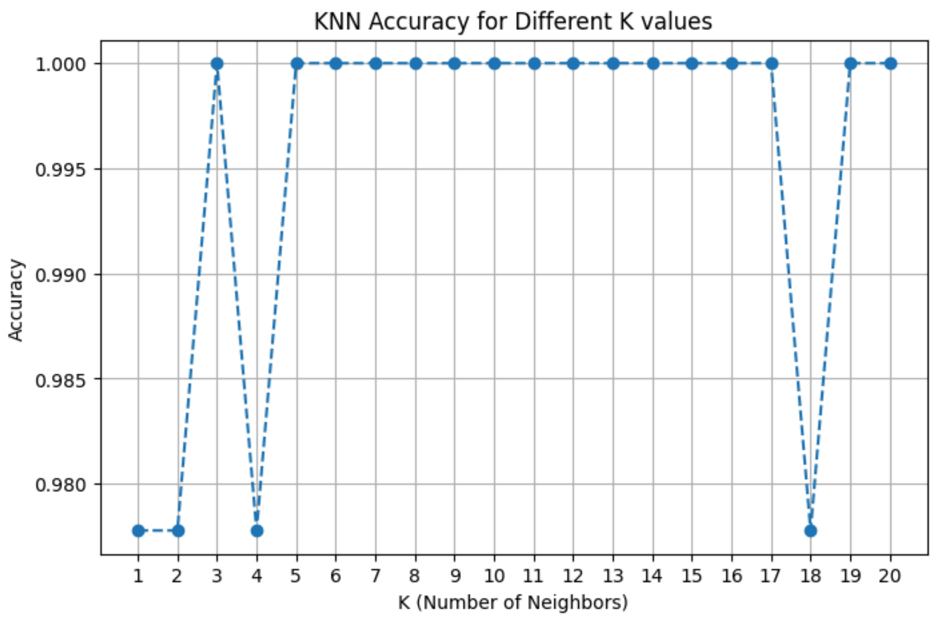
**MAHATI AKELLA 22070521027**

## K-Nearest Neighbors (KNN) Classifier

Problem Statement: Use the KNN algorithm to classify flowers in the Iris dataset. Tune the value of ‘K’ and evaluate accuracy using confusion matrix. Activities: • Implement KNN • Test multiple K values • Visualize decision boundaries

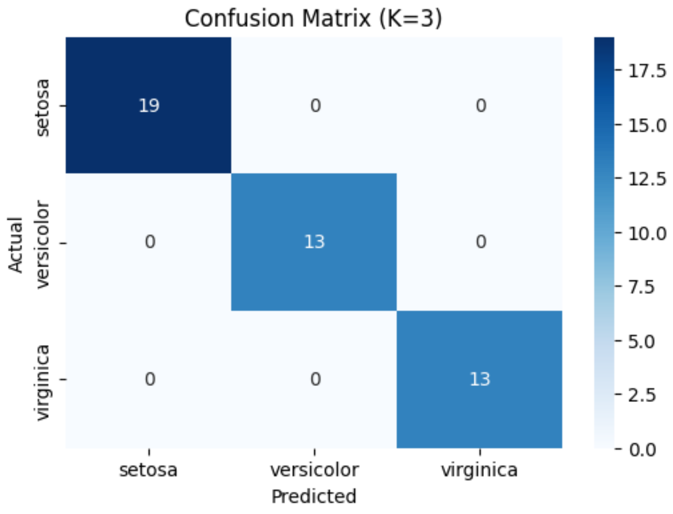
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix


iris = datasets.load_iris()
X, y = iris.data, iris.target


X_vis = X[:, :2]


# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# Feature scaling (important for distance-based algorithms)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


# Test multiple K values
k_values = range(1, 21)
accuracies = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)


# Plot accuracy vs K
plt.figure(figsize=(8, 5))
plt.plot(k_values, accuracies, marker='o', linestyle='--')
plt.title("KNN Accuracy for Different K values")
plt.xlabel("K (Number of Neighbors)")
plt.ylabel("Accuracy")
plt.xticks(k_values)
plt.grid(True)
plt.show()
```



```
# Best K
best_k = k_values[np.argmax(accuracies)]
print(f"Best K = {best_k}, Accuracy = {max(accuracies):.4f}")
```

```
Best K = 3, Accuracy = 1.0000
```

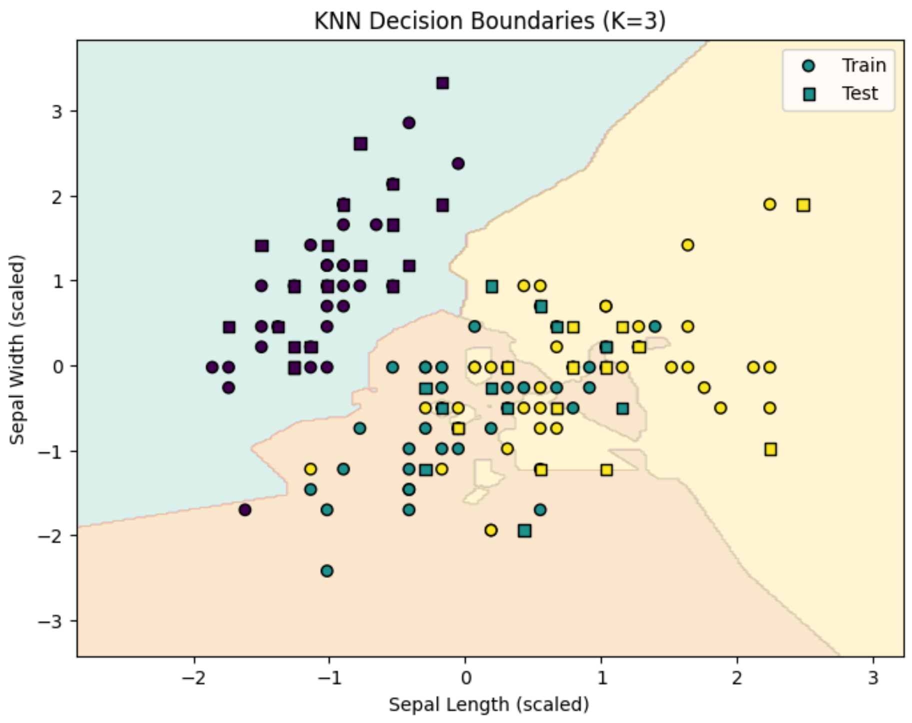
```
# Train final model with best K
knn_best = KNeighborsClassifier(n_neighbors=best_k)
knn_best.fit(X_train, y_train)
y_pred_best = knn_best.predict(X_test)


# Confusion Matrix
cm = confusion_matrix(y_test, y_pred_best)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=iris.target_names,
            yticklabels=iris.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title(f"Confusion Matrix (K={best_k})")
plt.show()
```



```
# ---- Decision Boundary Visualization ----
# Use only 2 features (for plotting)
X_vis = X[:, :2]
X_train_vis, X_test_vis, y_train_vis, y_test_vis = train_test_split(X_vis, y, test_size=0.3, random_state=42)
X_train_vis = scaler.fit_transform(X_train_vis)
X_test_vis = scaler.transform(X_test_vis)

knn_vis = KNeighborsClassifier(n_neighbors=best_k)
knn_vis.fit(X_train_vis, y_train_vis)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

```
# Create meshgrid
x_min, x_max = X_train_vis[:, 0].min() - 1, X_train_vis[:, 0].max() + 1
y_min, y_max = X_train_vis[:, 1].min() - 1, X_train_vis[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))


# Predict on meshgrid
Z = knn_vis.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)


# Plot decision boundary
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.Set3)
plt.scatter(X_train_vis[:, 0], X_train_vis[:, 1], c=y_train_vis, edgecolors='k', marker='o', label="Train")
plt.scatter(X_test_vis[:, 0], X_test_vis[:, 1], c=y_test_vis, edgecolors='k', marker='s', label="Test")
plt.xlabel("Sepal Length (scaled)")
plt.ylabel("Sepal Width (scaled)")
plt.title(f"KNN Decision Boundaries (K={best_k})")
plt.legend()
plt.show()
```



## Conclusions & Observations from KNN Classifier on Iris Dataset

**Effect of K on Accuracy**

The model's accuracy changes as we vary the value of K (number of neighbors).

For very small K (e.g., K=1), the classifier is too sensitive to noise → may overfit.

For very large K, the model becomes too generalized → may underfit.

The plot of Accuracy vs K helps us choose the optimal K where the model balances bias & variance.

**Best K Value**

From the accuracy curve, the best K (with maximum accuracy) is observed.

This shows that tuning hyperparameter K is critical for achieving good classification performance.

**Confusion Matrix Insights**

The confusion matrix shows how many samples of each Iris class (Setosa, Versicolor, Virginica) were correctly or incorrectly classified.

If most values lie on the diagonal, it means the classifier is performing well.

Misclassifications (if any) usually occur between Versicolor and Virginica since they are more similar compared to Setosa.

**Decision Boundaries**

The decision boundary visualization (using 2 features) shows how KNN separates classes in feature space.

Iris Setosa is very distinct and easy to classify, while Versicolor and Virginica overlap slightly.

**The shape of the decision boundaries changes with different K values:**

Small K → more complex, wiggly boundaries.

Large K → smoother, more generalized boundaries.

**General Understanding**

KNN is a distance-based, non-parametric algorithm. It works well for small datasets like Iris.

Feature scaling is important since distance is the basis of classification.

Choosing the right K is crucial: neither too small (overfitting) nor too large (underfitting)