LAB 5

KNN:

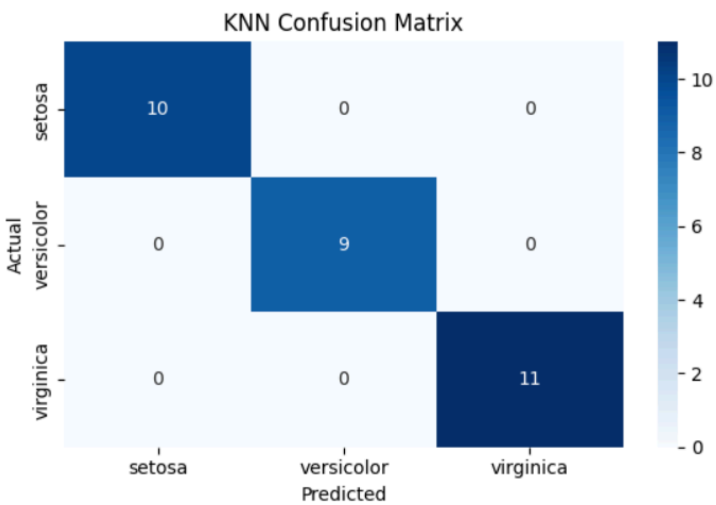
In [9]:

```
#Iris dataset
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from google.colab import files
# Upload the CSV file
uploaded = files.upload()
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(next(iter(uploaded)))  # Load the first uploaded file
print(df.head())
le = LabelEncoder()
df['species'] = le.fit_transform(df['species'])

# Split features and target
X = df.drop('species', axis=1)
y = df['species']

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Choose K using sqrt rule
k = int(math.sqrt(len(X_train)))
if k % 2 == 0:
    k += 1
```

```
# Train KNN model
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)

# Predict on test set
y_pred = knn.predict(X_test)

# Accuracy and Evaluation
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred, target_names=le.classes_)

# Print results
print(f"Chosen K: {k}")
print(f"Accuracy Score: {accuracy:.2f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)

# Plot confusion matrix
plt.figure(figsize=(6,4))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',
            xticklabels=le.classes_, yticklabels=le.classes_)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("KNN Confusion Matrix")
plt.tight_layout()
plt.show()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving iris (1).csv to iris (1) (6).csv
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
Chosen K: 11
Accuracy Score: 1.00

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```



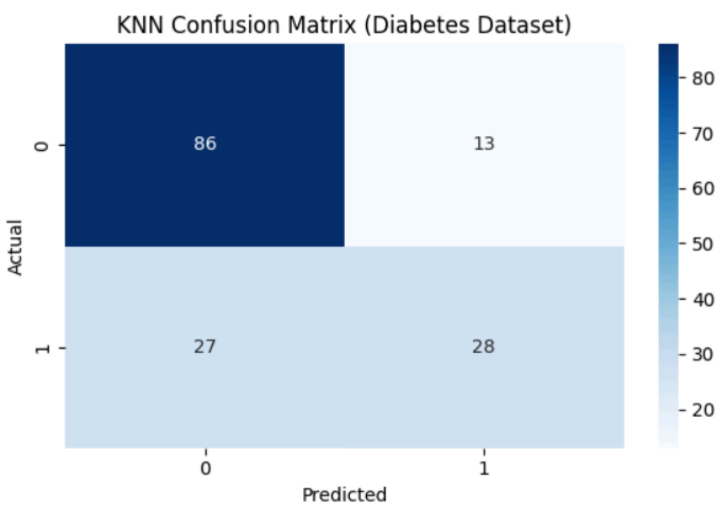
In [8]:

```
#diabetes dataset
import pandas as pd
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
import matplotlib.pyplot as plt
import seaborn as sns
import math
import pandas as pd
from google.colab import files
# Upload the CSV file
uploaded = files.upload()
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(next(iter(uploaded)))  # Load the first uploaded file
print(df.head())
columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
           'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']

# Split features and target
X = df.drop('Outcome', axis=1)
y = df['Outcome']

# Feature Scaling (Standardization)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Choose k value (sqrt(n) rule of thumb)
k = int(math.sqrt(len(X_train)))
if k % 2 == 0:
    k += 1  # Prefer odd k to avoid ties
```

```
# Evaluate model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Chosen K: {k}")
print(f"Accuracy Score: {accuracy:.2f}")
print("\nConfusion Matrix:")
print(conf_matrix)

# Plot Confusion Matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("KNN Confusion Matrix (Diabetes Dataset)")
plt.tight_layout()
plt.show()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving diabetes.csv to diabetes (1).csv
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
Chosen K: 25
Accuracy Score: 0.74

Confusion Matrix:
[[86 13]
 [27 28]]
```



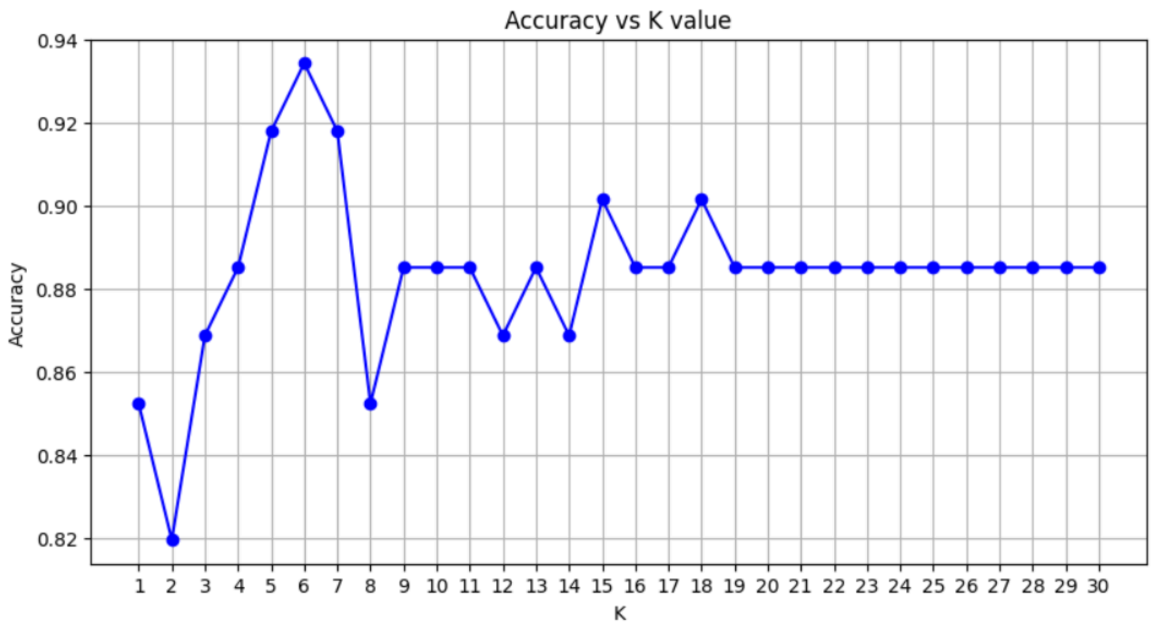
In [11]:

```python
#heart dataset
import pandas as pd
import numpy as np
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, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
import pandas as pd
from google.colab import files
# Upload the CSV file
uploaded = files.upload()
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(next(iter(uploaded)))  # Load the first uploaded file
print(df.head())

# Features and target
X = df.drop('target', axis=1)
y = df['target']

# Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Find the best K by testing accuracy for multiple values
accuracy_scores = []
k_range = range(1, 31)

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
```

```python
    accuracy_scores.append(acc)

# Plot Accuracy vs K
plt.figure(figsize=(10, 5))
plt.plot(k_range, accuracy_scores, marker='o', color='blue')
plt.title('Accuracy vs K value')
plt.xlabel('K')
plt.ylabel('Accuracy')
plt.xticks(k_range)
plt.grid(True)
plt.show()

# Get the best K
best_k = k_range[np.argmax(accuracy_scores)]
print(f"\n✅ Best K value: {best_k} with Accuracy: {max(accuracy_scores):.4f}")

# Retrain model using the best K
best_knn = KNeighborsClassifier(n_neighbors=best_k)
best_knn.fit(X_train, y_train)
y_pred_best = best_knn.predict(X_test)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred_best)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()

# Classification Report
report = classification_report(y_test, y_pred_best, target_names=["No Disease", "Disease"])
print("\n📊 Classification Report:")
print(report)
```

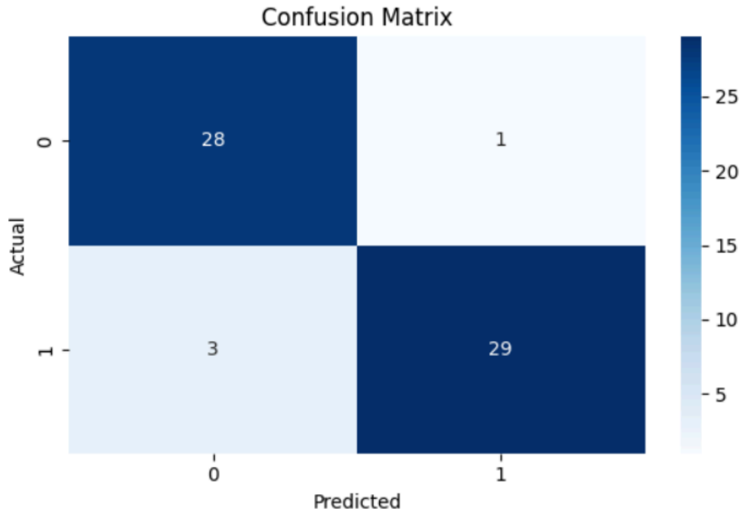
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving heart.csv to heart.csv
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   3       145   233    1        0      150      0      2.3      0
1   37    1   2       130   250    0        1      187      0      3.5      0
2   41    0   1       130   204    0        0      172      0      1.4      2
3   56    1   1       120   236    0        1      178      0      0.8      2
4   57    0   0       120   354    0        1      163      1      0.6      2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```




```
✅ Best K value: 6 with Accuracy: 0.9344
```

✅ Best K value: 6 with Accuracy: 0.9344



📊 Classification Report:

	precision	recall	f1-score	support
No Disease	0.90	0.97	0.93	29
Disease	0.97	0.91	0.94	32
accuracy			0.93	61
macro avg	0.93	0.94	0.93	61
weighted avg	0.94	0.93	0.93	61

(22)

7/4/25 monday

Lab - 5

KNN

1)

Age	Salary	Target	Distance
18	50	N	52.8
23	55	N	46.6
24	70	N	31.9
41	60	Y	40.4
43	70	Y	31.0
38	40	Y	60.1

$x = (35, 100)$

$d_1 = \sqrt{(18-35)^2 + (50-100)^2} = 52.8$

$d_2 = \sqrt{(23-35)^2 + (55-100)^2} = 46.6$

$d_3 = \sqrt{(24-35)^2 + (70-100)^2} = 31.9$

$d_4 = \sqrt{(41-35)^2 + (60-100)^2} = 40.4$

$d_5 = \sqrt{(43-35)^2 + (70-100)^2} = 31.0$

$d_6 = \sqrt{(38-35)^2 + (40-100)^2} = 60.1$

$K = 3$

closest neighbours are (43, 70), (24, 70) & (41, 60)
Y N Y

Majority votes are Y, hence (35, 100) target is Y

2) Iris dataset:

To choose K value, we calculate accuracy & error rate. we then plot Accuracy vs K & Error rate vs K, Choose K where accuracy is highest and error rate is lowest

$K = 11$

34 Diabetics dataset

Features having much larger range like Glucose are present with small range value like Diabetes pedigree Function, of which the larger values dominate the distance calculation. Hence, feature scaling is required. It is done by Z-score scaling scaling,

$$x_{scale} = \frac{x - \mu}{\sigma}$$

where, μ = mean, σ = Standard deviation

$K = 25$