```
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 (
    confusion_matrix, accuracy_score, precision_score,
    recall_score, f1_score
)
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("/content/diabetes.csv")   # <-- Change path if needed
print("✅ Dataset Loaded Successfully")
print("Shape of dataset:", df.shape)
print(df.head())
```

```
✅ Dataset Loaded Successfully
Shape of dataset: (768, 9)
   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

   Pedigree  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
```

```
print("\nMissing Values per column:\n", df.isnull().sum())
df = df.fillna(df.median())
```

```
Missing Values per column:
 Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
Pedigree         0
Age              0
Outcome          0
dtype: int64
```

```
X = df.drop("Outcome", axis=1)
y = df["Outcome"]

# Step 5: 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 6: Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

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

# Step 8: Predictions
y_pred = knn.predict(X_test)
```

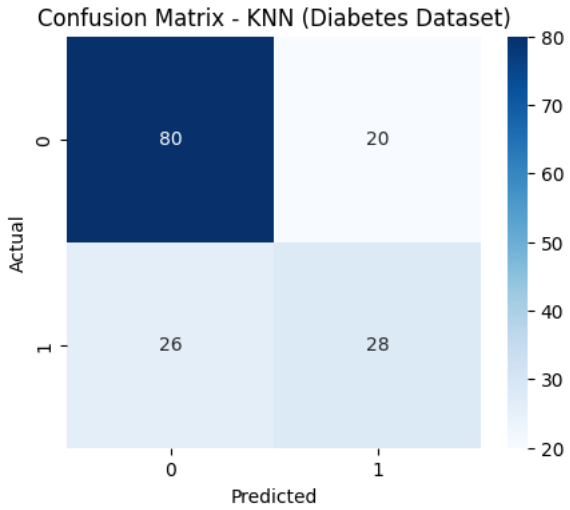
```
acc = accuracy_score(y_test, y_pred)
err = 1 - acc
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("\n📊 Model Performance Metrics:")
print(f"Accuracy     : {acc:.3f}")
print(f"Error Rate   : {err:.3f}")
print(f"Precision    : {prec:.3f}")
```

```
print(f"Recall       : {rec:.3f}")
print(f"F1 Score     : {f1:.3f}")
```

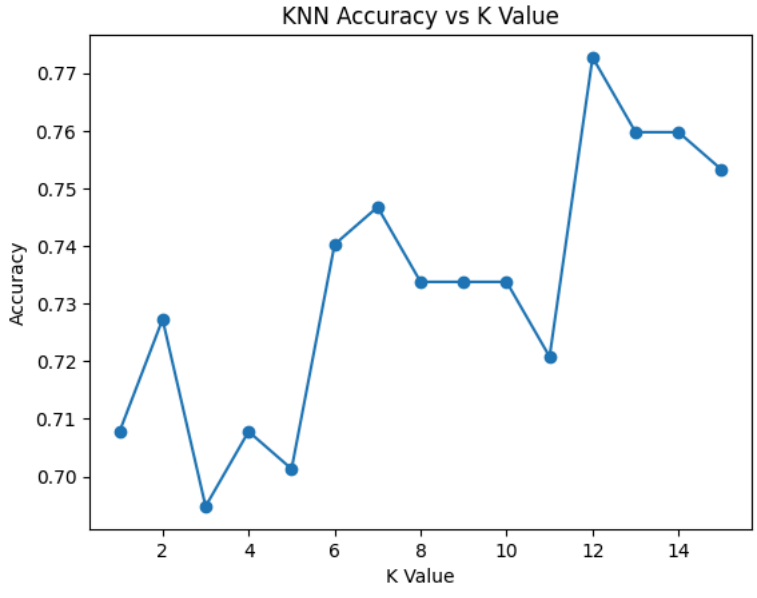
```
📊 Model Performance Metrics:
Accuracy     : 0.701
Error Rate   : 0.299
Precision    : 0.583
Recall       : 0.519
F1 Score     : 0.549
```

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.title("Confusion Matrix - KNN (Diabetes Dataset)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```




```
acc_values = []
for k in range(1, 16):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    acc_values.append(knn.score(X_test, y_test))
```

```
plt.plot(range(1, 16), acc_values, marker='o')
plt.xlabel("K Value")
plt.ylabel("Accuracy")
plt.title("KNN Accuracy vs K Value")
plt.show()
```



Start coding or generate with AI.