In [1]:

```
# Import necessary libraries
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 matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
data = pd.read_csv('diabetes.csv')
```

In [11]:

```
data.head()
```

Out[11]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Split the Data into Features and Target

In [3]:

```
# Define features (X) and target (y)
X = data.drop(columns='Outcome')
y = data['Outcome']
```

Split the Data into Training and Testing Sets We'll split the data into training and testing sets (80% training, 20% testing).

In [4]:

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

Feature Scaling (Standardization)

In [5]:

```
# Standardize the features (important for KNN)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Train the KNN Model

In [6]:

```
# Initialize the KNN classifier with k=5
knn = KNeighborsClassifier(n_neighbors=5)

# Train the model on the training data
knn.fit(X_train, y_train)
```

Out[6]:

```
▾ KNeighborsClassifier

KNeighborsClassifier()
```

In [7]:

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

Step 8: Evaluate the Model

In [8]:

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)

# Error Rate
error_rate = 1 - accuracy

# Precision
precision = precision_score(y_test, y_pred)

# Recall
recall = recall_score(y_test, y_pred)

# F1 Score (optional, to get a balanced measure)
```

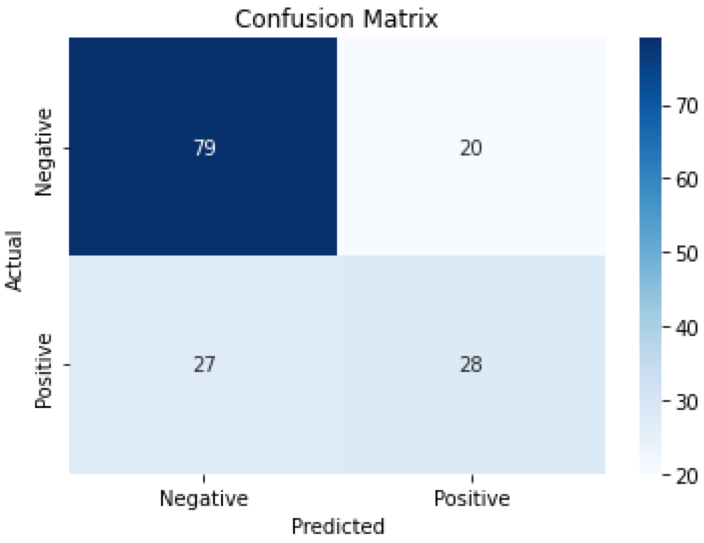
```
f1 = f1_score(y_test, y_pred)

# Display the results
print("Confusion Matrix:")
print(cm)
print(f"Accuracy: {accuracy:.4f}")
print(f"Error Rate: {error_rate:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

```
Confusion Matrix:
[[79 20]
 [27 28]]
Accuracy: 0.6948
Error Rate: 0.3052
Precision: 0.5833
Recall: 0.5091
F1 Score: 0.5437
```

In [9]:

```
# Visualize the confusion matrix using Seaborn heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Negative", "Positive"], yticklabels=["Negative", "
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
Actual
Negative
Positive
79
20
27
28
Negative
Positive
Predicted
70
60
50
40
30
20