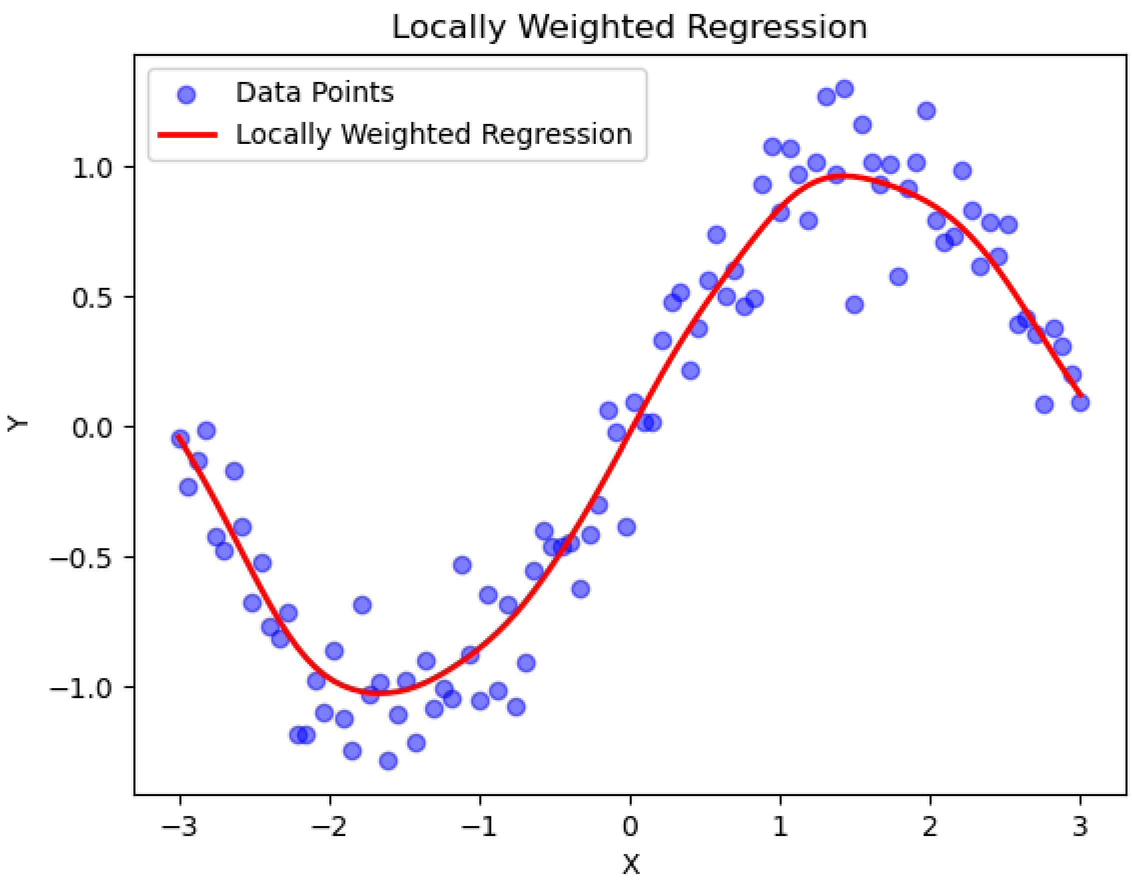
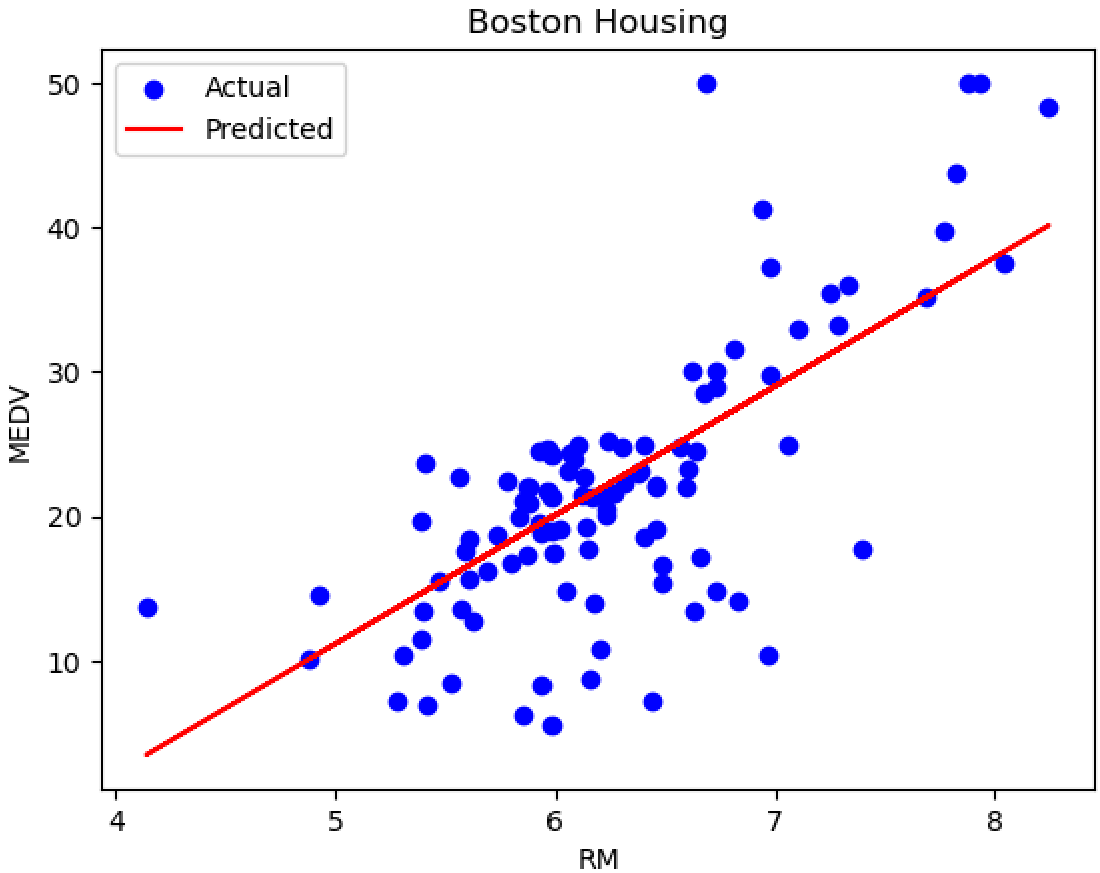
In [1]:
```
#6
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

np.random.seed(42)
X = np.linspace(-3, 3, 100)
Y = np.sin(X) + np.random.normal(0, 0.2, 100)

X = X.reshape(-1, 1)
Y = Y.reshape(-1, 1)

def weight_matrix(X, x_query, tau):
    return np.diag(np.exp(-(X - x_query)**2 / (2 * tau**2)).flatten())

def locally_weighted_regression(X, Y, x_query, tau=0.5):
    W = weight_matrix(X, x_query, tau)
    X_bias = np.hstack((np.ones_like(X), X))
    theta = np.linalg.inv(X_bias.T @ W @ X_bias) @ X_bias.T @ W @ Y
    return theta[0] + theta[1] * x_query

X_test = np.linspace(-3, 3, 100).reshape(-1, 1)
Y_pred = np.array([locally_weighted_regression(X, Y, x_query, tau=0.3) for x_query in X_test])

plt.scatter(X, Y, label="Data Points", color="blue", alpha=0.5)
plt.plot(X_test, Y_pred, label="Locally Weighted Regression", color="red", linewidth=2)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Locally Weighted Regression")
plt.legend()
plt.show()
```



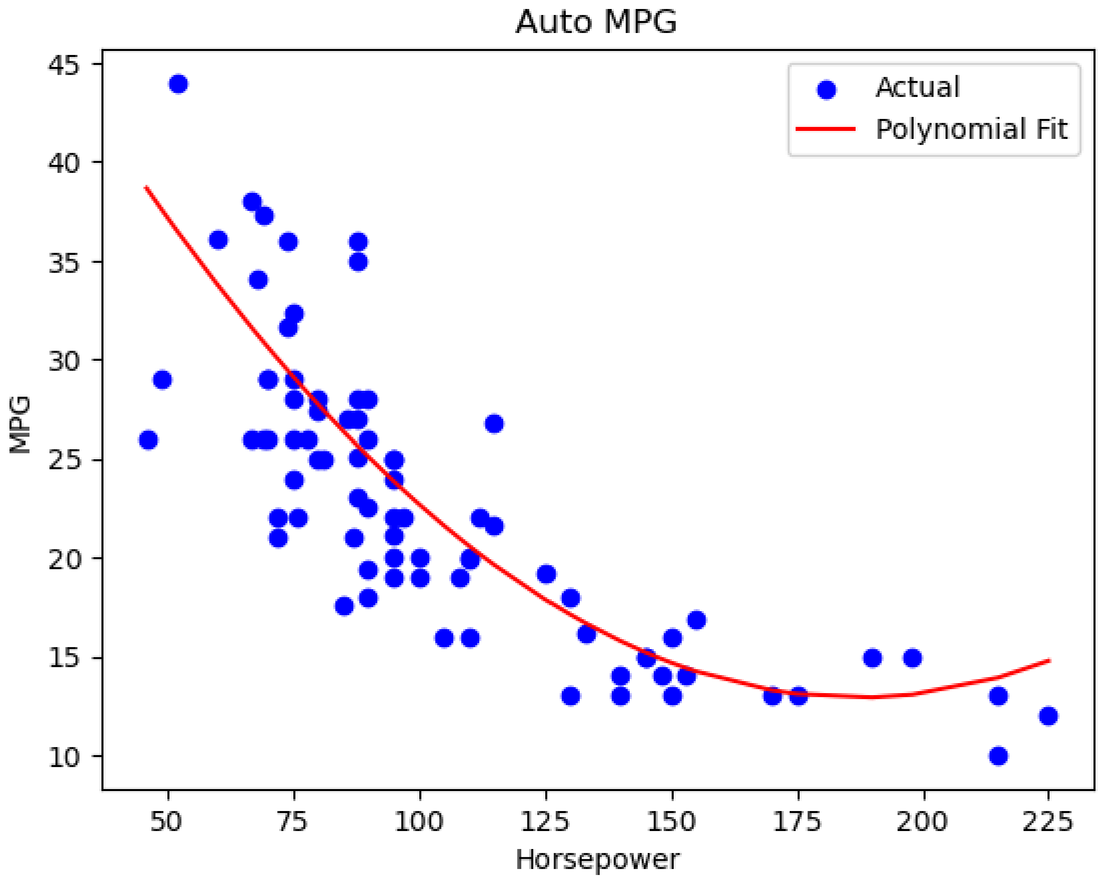
In [6]:
```
#7
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

def linear_regression_boston():
    df = pd.read_csv("BostonHousing.csv")
    X, y = df[['RM']], df['MEDV']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=43)
```

```
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    print("Linear Regression - Boston Housing")
    print(f"MSE: {mean_squared_error(y_test, y_pred):.2f}, R²: {r2_score(y_test, y_pred):.2f}")

    plt.scatter(X_test, y_test, color='blue', label='Actual')
    plt.plot(X_test, y_pred, color='red', label='Predicted')
    plt.xlabel('RM'), plt.ylabel('MEDV'), plt.title('Boston Housing')
    plt.legend(), plt.show()

def polynomial_regression_auto_mpg():
    url = "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
    cols = ['mpg', 'cyl', 'disp', 'hp', 'wt', 'acc', 'year', 'origin', 'name']
    df = pd.read_csv(url, names=cols, delim_whitespace=True, na_values='?').dropna()
    df['hp'] = df['hp'].astype(float)
    X, y = df[['hp']], df['mpg']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    poly = PolynomialFeatures(degree=2)
    X_train_poly, X_test_poly = poly.fit_transform(X_train), poly.transform(X_test)

    model = LinearRegression()
    model.fit(X_train_poly, y_train)
    y_pred = model.predict(X_test_poly)

    print("Polynomial Regression - Auto MPG")
    print(f"MSE: {mean_squared_error(y_test, y_pred):.2f}, R²: {r2_score(y_test, y_pred):.2f}")

    sorted_idx = X_test['hp'].argsort()
    plt.scatter(X_test, y_test, color='blue', label='Actual')
    plt.plot(X_test.iloc[sorted_idx], y_pred[sorted_idx], color='red', label='Polynomial Fit')
    plt.xlabel('Horsepower'), plt.ylabel('MPG'), plt.title('Auto MPG')
    plt.legend(), plt.show()

linear_regression_boston()
polynomial_regression_auto_mpg()
```

```
Linear Regression - Boston Housing
MSE: 44.84, R²: 0.51
```




```
C:\Users\Kusuma\AppData\Local\Temp\ipykernel_27376\385270490.py:32: FutureWarning: The 'delim_whitespace' keyword in pd.read_cs
v is deprecated and will be removed in a future version. Use ``sep='\s+'`` instead
  df = pd.read_csv(url, names=cols, delim_whitespace=True, na_values='?').dropna()
Polynomial Regression - Auto MPG
MSE: 18.42, R²: 0.64
```

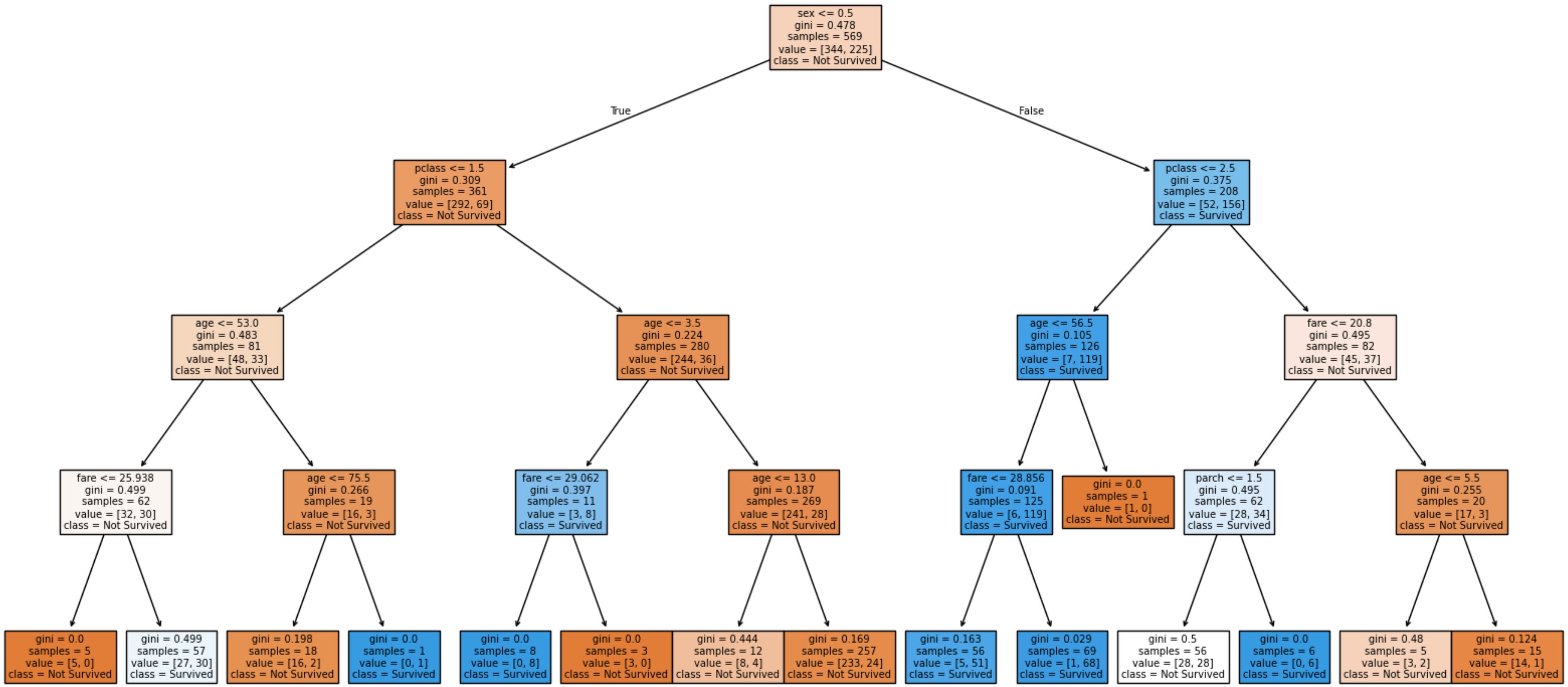


In [7]:

```
#8
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

df = sns.load_dataset("titanic")

features = ['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked']
df = df[features + ['survived']].dropna()
```

```
df['sex'] = df['sex'].map({'male': 0, 'female': 1})
df['embarked'] = df['embarked'].map({'S': 0, 'C': 1, 'Q': 2})

X = df[features]
y = df['survived']

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

clf = DecisionTreeClassifier(max_depth=4, random_state=43)
clf.fit(X_train, y_train)

plt.figure(figsize=(20, 10))
plot_tree(clf, feature_names=features, class_names=['Not Survived', 'Survived'], filled=True)
plt.title("Decision Tree - Titanic Survival Prediction")
plt.show()

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Classification Report")
print(classification_report(y_test, y_pred, target_names=["Not Survived", "Survived"]))
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

Decision Tree - Titanic Survival Prediction

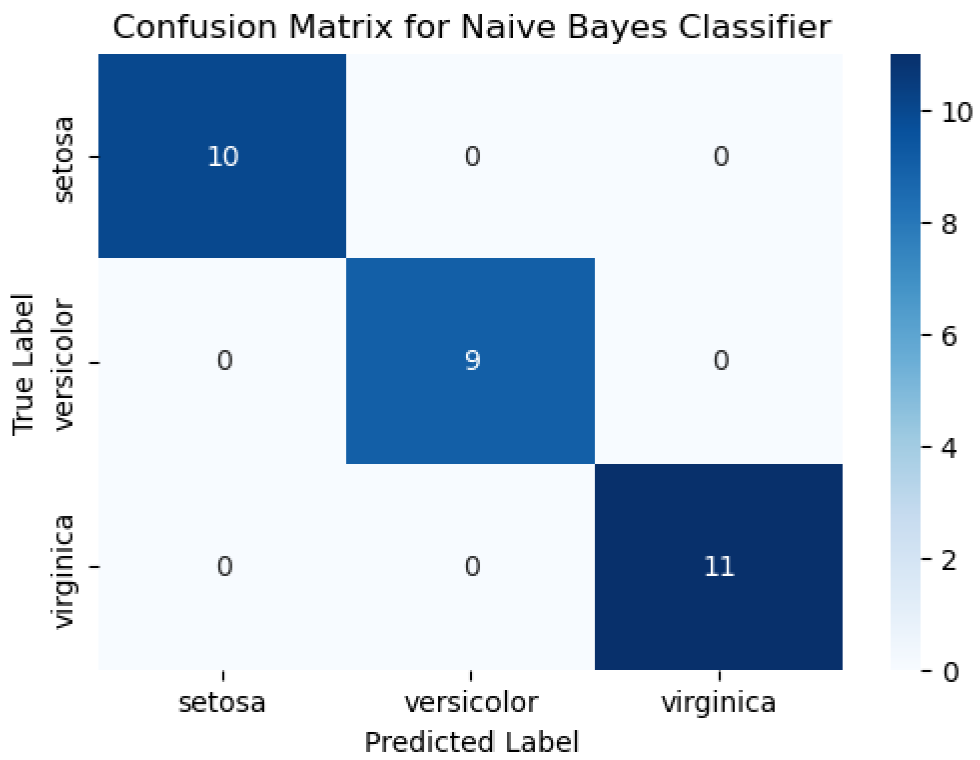


```
Classification Report
              precision    recall  f1-score   support

Not Survived       0.68      0.82      0.75        80
    Survived       0.70      0.51      0.59        63

    accuracy                           0.69       143
   macro avg       0.69      0.67      0.67       143
weighted avg       0.69      0.69      0.68       143

Accuracy: 0.69
Precision: 0.70
Recall: 0.51
F1 Score: 0.59
```

In [9]:
```python
#9
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_iris
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names

df = pd.DataFrame(X, columns=iris.feature_names)
df['species'] = y

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

nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred = nb_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
```

```
print("Classification Report")
print(classification_report(y_test, y_pred, target_names=target_names))
print(f"Accuracy of Naive Bayes Classifier: {accuracy:.2f}")

conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", xticklabels=target_names, yticklabels=target_names)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix for Naive Bayes Classifier")
plt.show()
```

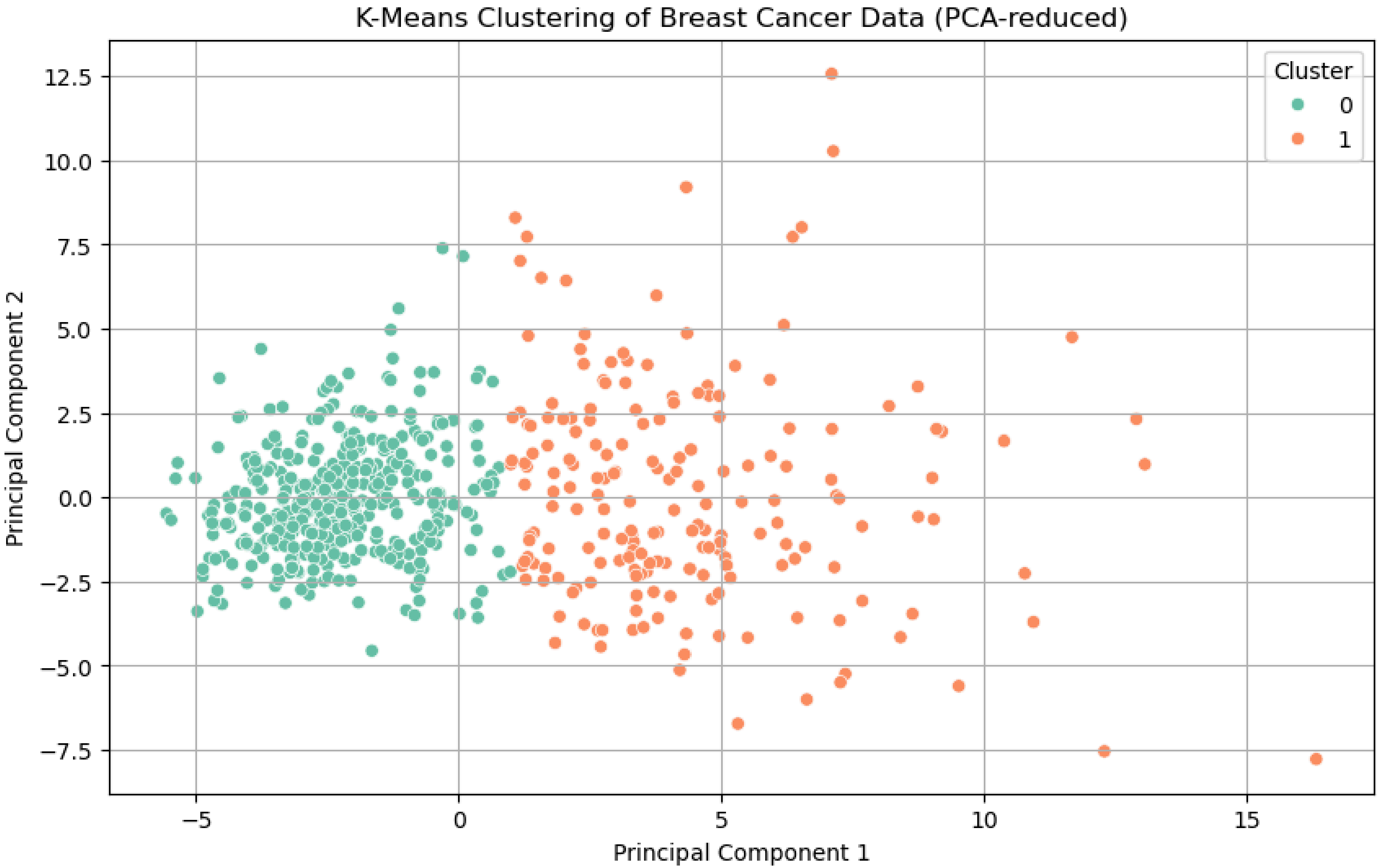
```
Classification Report
              precision    recall  f1-score   support

      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

Accuracy of Naive Bayes Classifier: 1.00
```



In [10]:
```
#10
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)

scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
```

```
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
kmeans.fit(scaled_data)
clusters = kmeans.labels_

df["Cluster"] = clusters

pca = PCA(n_components=2)
pca_components = pca.fit_transform(scaled_data)

df["PCA1"] = pca_components[:, 0]
df["PCA2"] = pca_components[:, 1]

plt.figure(figsize=(10, 6))
sns.scatterplot(x="PCA1", y="PCA2", hue="Cluster", palette="Set2", data=df)
plt.title("K-Means Clustering of Breast Cancer Data (PCA-reduced)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title="Cluster")
plt.grid(True)
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=3.
  warnings.warn(
```



K-Means Clustering of Breast Cancer Data (PCA-reduced)
Cluster
0
1
Principal Component 2
Principal Component 1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15

In [ ]: