Homework 4 Report
Name: Viprav Lipare
Student ID: 801288922
Homework Number: 4
Github Repository: https://github.com/vipravlipare/ECGR-5105-Intro-to-Machine-Learning

Problem 1

Source Code (1):

```
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

sample = datasets.load_breast_cancer()
X = sample.data
Y = sample.target

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=42, stratify=Y)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

kernels = ['linear', 'poly', 'rbf', 'sigmoid']
svm_results = {}
svm_cms = {}

for kernel in kernels:
    model = SVC(kernel=kernel, C=1.0)
    model.fit(X_train, Y_train)
    Y_pred = model.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)
    cm = confusion_matrix(Y_test, Y_pred)

    svm_results[kernel] = [accuracy, precision, recall, f1]
    svm_cms[kernel] = cm

    # Print performance results
    print("kernel:", kernel)
    print("Accuracy :" ,accuracy)
    print("Precision:" ,precision)
    print("Recall   :" ,recall)
    print("F1 Score :" ,f1)
    print(" ")

# Homework 3 Results
Hw3_accuracy = 0.9824561403508771
Hw3_precision = 0.9726027397260274
Hw3_recall = 1.0
Hw3_f1 = 0.9861111111111112

Hw3_cm = np.array([[41, 2],
                   [0, 71]])

# Print performance results
print("HW3 Results:")
print("Accuracy :" ,Hw3_accuracy)
print("Precision:" ,Hw3_precision)
print("Recall   :" ,Hw3_recall)
print("F1 Score :" ,Hw3_f1)
print(" ")

best_kernel = None
max_accuracy = 0

for kernel in svm_results:
    accuracy = svm_results[kernel][0]
    if accuracy > max_accuracy:
        max_accuracy = accuracy
        best_kernel = kernel
```

```
# Plot both confusion matrices side-by-side
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
ConfusionMatrixDisplay(confusion_matrix=svm_cms[best_kernel],
                       display_labels=sample.target_names).plot(ax=ax[0],
cmap='Blues')
ax[0].set_title(f"SVM ({best_kernel} kernel) Confusion Matrix")

ConfusionMatrixDisplay(confusion_matrix=Hw3_cm,
                       display_labels=sample.target_names).plot(ax=ax[1],
cmap='Reds')
ax[1].set_title("Logistic Regression (HW3) Confusion Matrix")

plt.tight_layout()
plt.show()
```

Figures / Results (1):

kernel: linear
Accuracy : 0.9736842105263158
Precision: 0.9859154929577465
Recall : 0.9722222222222222
F1 Score : 0.9790209790209791

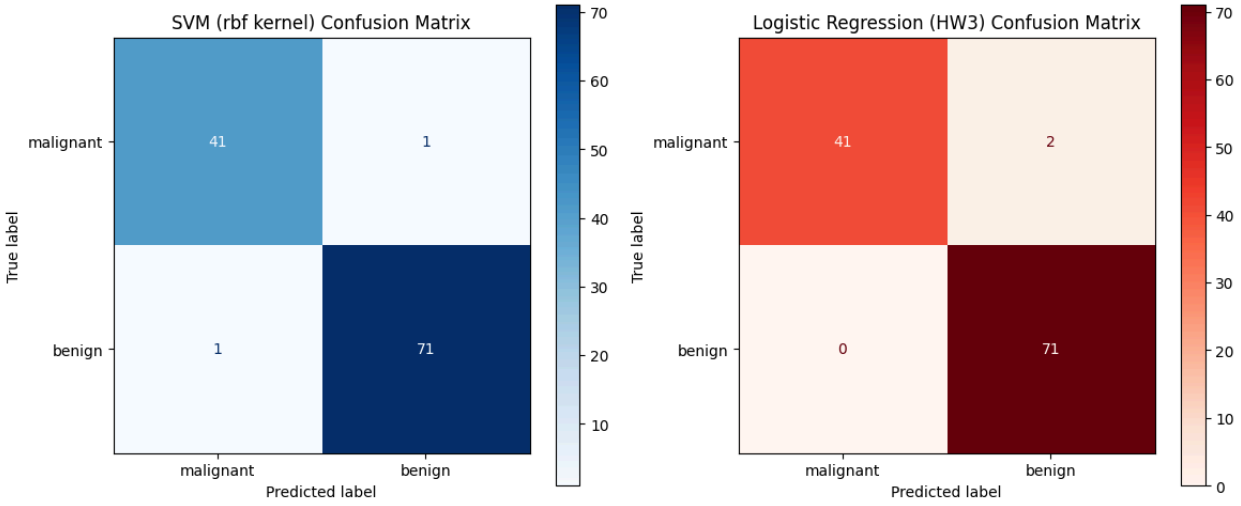
kernel: poly
Accuracy : 0.9122807017543859
Precision: 0.8780487804878049
Recall : 1.0
F1 Score : 0.935064935064935

kernel: rbf
Accuracy : 0.9824561403508771
Precision: 0.9861111111111112
Recall : 0.9861111111111112
F1 Score : 0.9861111111111112

kernel: sigmoid
Accuracy : 0.9298245614035088
Precision: 0.9571428571428572
Recall : 0.9305555555555556
F1 Score : 0.9436619718309859

HW3 Results:
Accuracy : 0.9824561403508771

Precision: 0.9726027397260274
Recall : 1.0
F1 Score : 0.9861111111111112



Problem 2

Source Code (2):

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVR
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
file_path = "Housing.csv"
sample = pd.read_csv(file_path)

inputs = ['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'parking',
'prefarea']

binary_cols = ['mainroad', 'guestroom', 'basement', 'hotwaterheating',
'airconditioning', 'prefarea']
for col in binary_cols:
    sample[col] = sample[col].map({'yes': 1, 'no': 0})
```

```
X = sample[inputs].values
Y = sample.values[:, 0]; #Price

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
scaler_X = StandardScaler()
X_train = scaler_X.fit_transform(X_train)
X_test = scaler_X.transform(X_test)

svr_rbf = SVR(kernel='rbf', C=100, gamma=0.1)
svr_lin = SVR(kernel='linear', C=5)
svr_poly = SVR(kernel='poly', C=100, degree=2)

svr_rbf.fit(X_train, y_train)
svr_lin.fit(X_train, y_train)
svr_poly.fit(X_train, y_train)

# Predict on test data
y_pred_rbf = svr_rbf.predict(X_test)
y_pred_lin = svr_lin.predict(X_test)
y_pred_poly = svr_poly.predict(X_test)

theta = np.array([
    1900563.24,   # X_0 (intercept)
    2796088.95,   # area
    786167.01,    # bedrooms
    2788723.16,   # bathrooms
    1242479.05,   # stories
    479021.23,    # mainroad
    291412.56,    # guestroom
    413403.10,    # basement
    697124.62,    # hotwaterheating
    843144.28,    # airconditioning
    861042.73,    # parking
    644538.66     # prefarea
])

X_test_with_bias = np.hstack((np.ones((X_test.shape[0], 1)), X_test))
y_pred_hw1 = X_test_with_bias.dot(theta)
```

```
# Plot predictions vs actual prices
lw = 2
plt.figure(figsize=(8, 6))

# Sort by actual price for smooth plotting
sorted_idx = np.argsort(y_test)
y_test_sorted = y_test[sorted_idx]
y_hw1_sorted = y_pred_hw1[sorted_idx]

plt.scatter(y_test, y_pred_rbf, color='darkorange', label='SVR (RBF)',
alpha=0.6)
plt.plot(y_test_sorted, np.sort(y_pred_rbf), color='navy', lw=lw,
label='RBF fit')
plt.plot(y_test_sorted, np.sort(y_pred_lin), color='darkred', lw=lw,
label='SVR (Linear)')
plt.plot(y_test_sorted, np.sort(y_pred_poly), color='red', lw=lw,
label='SVR (Polynomial)')

# Homework 1 predictions
plt.plot(y_test_sorted, y_hw1_sorted, color='lightgray', lw=lw, label='HW1
Linear Reg')

# Lock axes to fixed range
plt.xlim(0.1e7, 1.4e7)        # Actual Price range
plt.ylim(4.284e6, 4.3e6)      # Predicted Price range

plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('SVR vs Homework 2 Linear Regression - Housing Price
Prediction')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

Figures / Results (2):