Name: Vikas Vijaykumar Bastewad

Roll No.: 20CS10073

Question 1 : SVM Classifier

References codes:

In [ ]:
```
# from ucimlrepo import fetch_ucirepo
# import pandas as pd

# fetch dataset
# spambase = fetch_ucirepo(id=94)

# data (as pandas dataframes)
# X = spambase.data.features
# y = spambase.data.targets

# metadata
# print(spambase.metadata)

# variable information
# print(spambase.variables)

# # loading as dataframe
# x = spambase.data.features
# y = spambase.data.targets
```

In [ ]:
```
# # Import Scikit learn
# from sklearn import datasets
# from sklearn.model_selection import train_test_split
# from sklearn.svm import SVC
```

```
# Load a Dataset
# iris = datasets.load_iris()
# X = iris.data
# y = iris.target

# Printing X
# X

# Spliting the data into training and testing sets
# X_train, X_test, y_train, y_test = train_test_split(X, y,
# test_size=0.2, random_state=42)

# Train SVM Model
# svm_model = SVC(kernel='linear')
# svm_model.fit(X_train, y_train)
# Get predictions
# y_pred = svm_model.predict(X_test)
# print(y_pred)
# y_pred.shape
```

Part A: SVM Implementation

In [ ]:
```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from ucimlrepo import fetch_ucirepo
# Import Scikit learn
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

In [ ]:
```
# Load the dataset
spambase = fetch_ucirepo(id=94)
X = spambase.data.features
y = spambase.data.targets.values.ravel()

# Spliting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Defining the Regularisation parameter values
regularization_parameters = [0.001, 0.1, 1, 10, 100]

accuracy_scores = []  # List to store accuracy scores
```

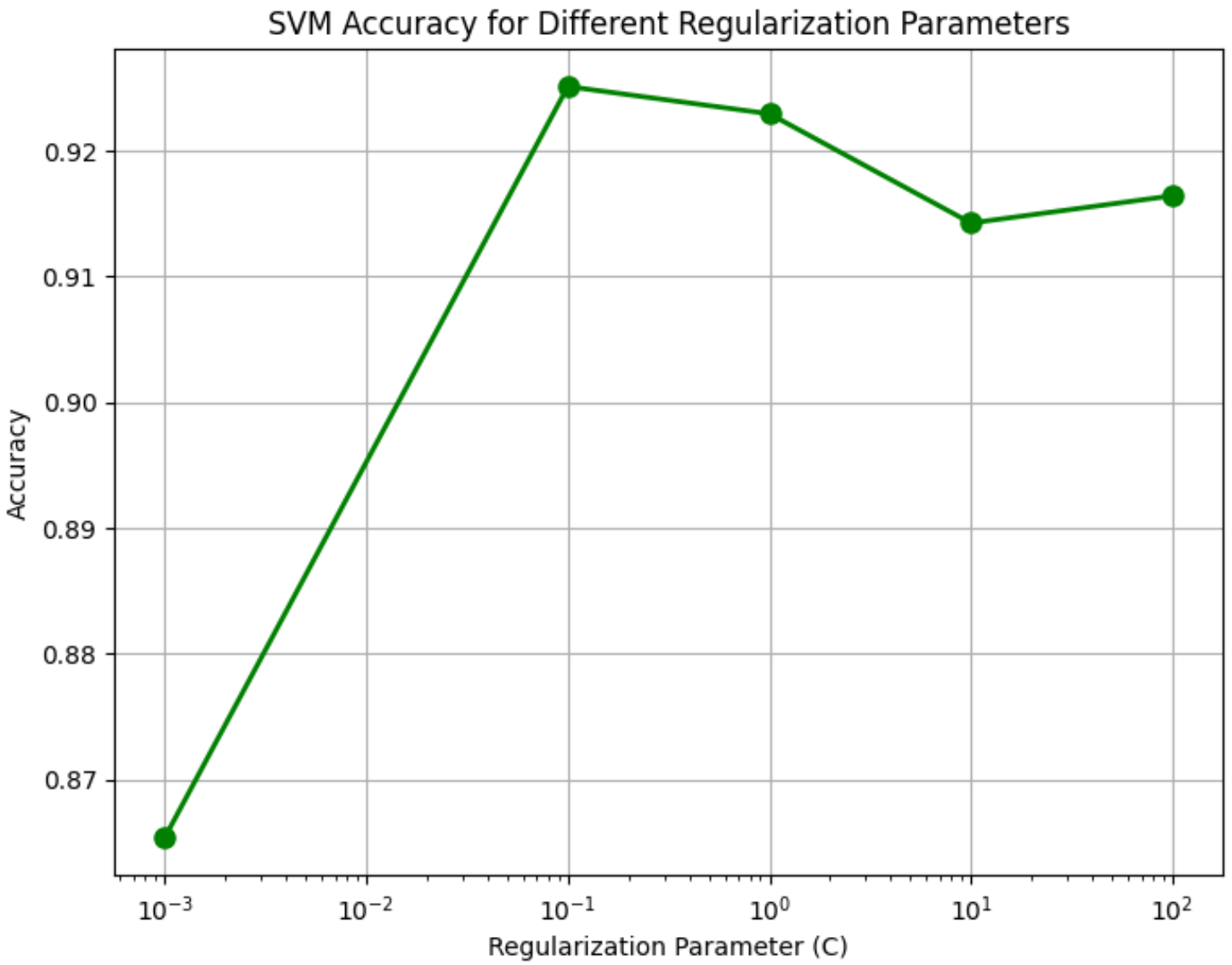
In [ ]:
```
# Implementing SVM with linear kernel for different regularization parameters
for reg_param in regularization_parameters:
    # Training SVM Model
    svm_model = SVC(kernel='linear', C=reg_param)
    svm_model.fit(X_train, y_train)

    # Geting predictions
    y_pred = svm_model.predict(X_test)

    # Calculating accuracy and appending to the accuracy_scores list
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

    # Printing the evaluation metrics
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    print(f"Regularization parameter: {reg_param}")
    print(f"Accuracy: {accuracy:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1-score: {f1:.2f}")
    print("-" * 40)
```

```
Regularization parameter: 0.001
Accuracy: 0.87
Precision: 0.91
Recall: 0.76
F1-score: 0.83
----------------------------------------
Regularization parameter: 0.1
Accuracy: 0.93
Precision: 0.93
Recall: 0.89
F1-score: 0.91
----------------------------------------
Regularization parameter: 1
Accuracy: 0.92
Precision: 0.93
Recall: 0.88
F1-score: 0.91
----------------------------------------
Regularization parameter: 10
Accuracy: 0.91
Precision: 0.90
Recall: 0.90
F1-score: 0.90
----------------------------------------
Regularization parameter: 100
Accuracy: 0.92
Precision: 0.90
Recall: 0.91
F1-score: 0.90
----------------------------------------
```

In [ ]:

```python
# Plotting the accuracy for different regularization parameters
plt.figure(figsize=(8, 6))
plt.plot(regularization_parameters, accuracy_scores, marker='o', color='g', linestyle='-', linewidth=
plt.xscale('log')
plt.xlabel('Regularization Parameter (C)')
plt.ylabel('Accuracy')
plt.title('SVM Accuracy for Different Regularization Parameters')
```

```
plt.grid(True)
plt.show()
```



Part B: Kernel Tricks

In [ ]:
```
# Kernels to be evaluated
kernels = ['poly', 'poly', 'sigmoid', 'rbf']  # Polynomial, Sigmoid, RBF
degrees = [2, 3, None, None] # Degree of polynomial kernel

# Evaluating SVM with different kernels
for i, kernel in enumerate(kernels):   # i is the index and kernel is the value
    degree = degrees[i]                # degree of polynomial kernel
    if kernel == 'poly':               # if kernel is polynomial
        svm_model = SVC(kernel=kernel, degree=degree, C=1)
    elif kernel == 'sigmoid':          # if kernel is sigmoid
        svm_model = SVC(kernel=kernel, C=1)
    elif kernel == 'rbf':              # if kernel is Radial Basis Function (RBF)
        svm_model = SVC(kernel=kernel, C=1, gamma='scale')
    else:
        print("Invalid kernel")
        continue

    # Training SVM Model
    svm_model.fit(X_train, y_train)

    # Geting the predictions
    y_pred = svm_model.predict(X_test)

    # Calculating and printing evaluation metrics
    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(f"Kernel: {kernel}, Degree: {degree}")
    print(f"Accuracy: {accuracy:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1-score: {f1:.2f}")
    print("-" * 40)
```

```
Kernel: poly, Degree: 2
Accuracy: 0.65
Precision: 0.88
Recall: 0.20
F1-score: 0.33
----------------------------------------
Kernel: poly, Degree: 3
Accuracy: 0.63
Precision: 0.87
Recall: 0.14
F1-score: 0.24
----------------------------------------
Kernel: sigmoid, Degree: None
Accuracy: 0.64
Precision: 0.57
Recall: 0.54
F1-score: 0.56
----------------------------------------
Kernel: rbf, Degree: None
Accuracy: 0.66
Precision: 0.66
Recall: 0.42
F1-score: 0.51
----------------------------------------
```

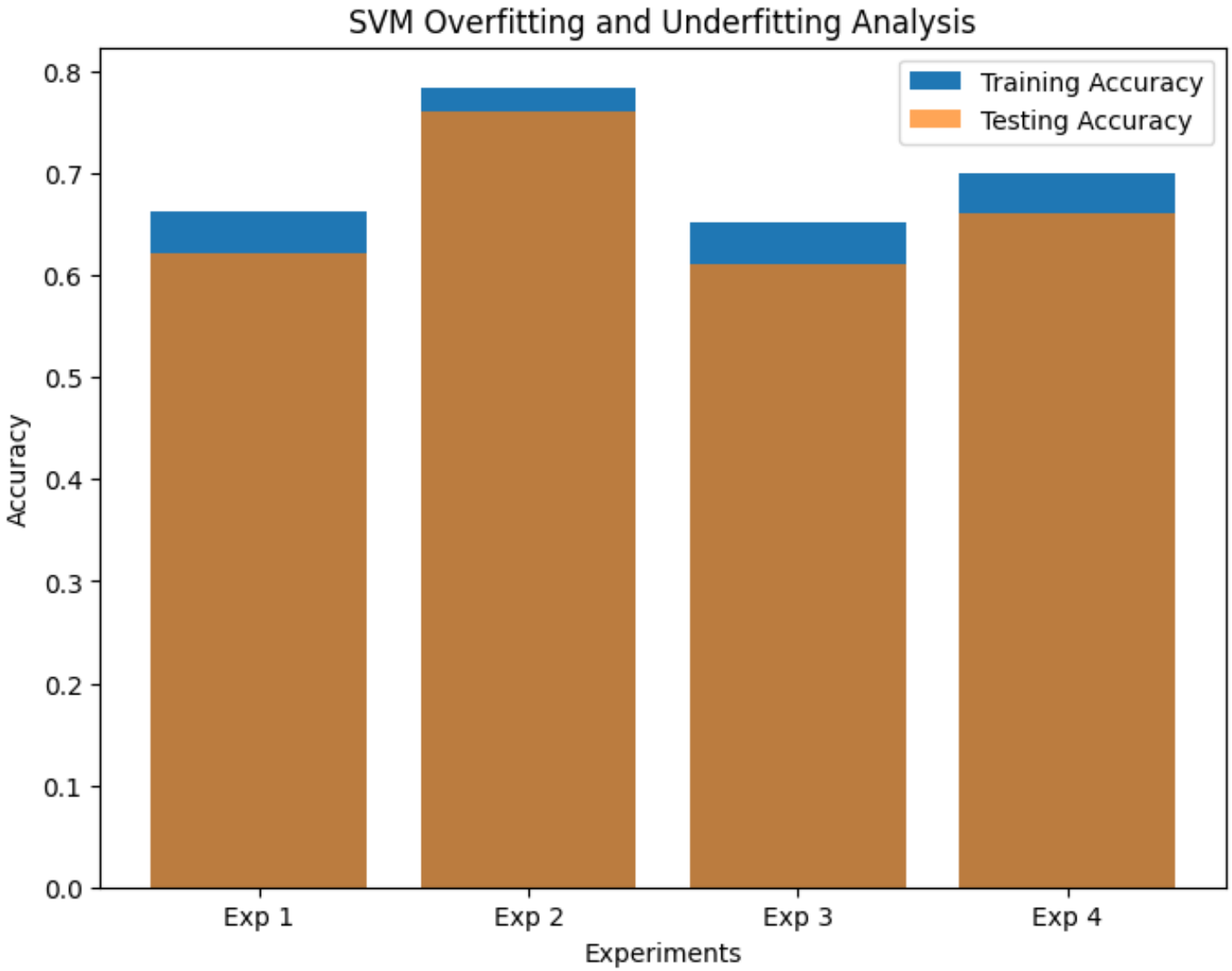
Part C: Overfitting & Underfitting Analysis

In [ ]:

```python
# Experiments with polynomial degree and regularization parameter 'C'
experiments = [(1, 0.01), (1, 100), (3, 0.01), (3, 100)]

# Training and evaluating SVM models for different experiments
train_accuracies = []  # For storing training accuracies
test_accuracies = []   # For storing testing accuracies

for exp_num, (degree, reg_param) in enumerate(experiments, start=1):
    # Training the SVM Model
```

```
    svm_model = SVC(kernel='poly', degree=degree, C=reg_param)
    svm_model.fit(X_train, y_train)

    # Geting predictions for training and testing data
    y_train_pred = svm_model.predict(X_train)
    y_test_pred = svm_model.predict(X_test)

    # Calculating accuracy for training and testing data
    train_accuracy = accuracy_score(y_train, y_train_pred)
    test_accuracy = accuracy_score(y_test, y_test_pred)

    # Storing accuracies for plotting
    train_accuracies.append(train_accuracy)
    test_accuracies.append(test_accuracy)

    # Printing and ploting the results
    print(f"Experiment {exp_num}: Polynomial Degree: {degree}, Regularization Parameter 'C': {reg_par
    print(f"Training Accuracy: {train_accuracy:.2f}")
    print(f"Testing Accuracy: {test_accuracy:.2f}")
    print("-" * 40)

# Plotting the results
experiments_labels = [f"Exp {i}" for i in range(1, len(experiments) + 1)]
plt.figure(figsize=(8, 6))
plt.bar(experiments_labels, train_accuracies, label='Training Accuracy')
plt.bar(experiments_labels, test_accuracies, label='Testing Accuracy', alpha=0.7)
plt.xlabel('Experiments')
plt.ylabel('Accuracy')
plt.title('SVM Overfitting and Underfitting Analysis')
plt.legend()
plt.show()
```

```
Experiment 1: Polynomial Degree: 1, Regularization Parameter 'C': 0.01
Training Accuracy: 0.66
Testing Accuracy: 0.62
----------------------------------------
Experiment 2: Polynomial Degree: 1, Regularization Parameter 'C': 100
Training Accuracy: 0.78
Testing Accuracy: 0.76
----------------------------------------
Experiment 3: Polynomial Degree: 3, Regularization Parameter 'C': 0.01
Training Accuracy: 0.65
Testing Accuracy: 0.61
----------------------------------------
Experiment 4: Polynomial Degree: 3, Regularization Parameter 'C': 100
Training Accuracy: 0.70
Testing Accuracy: 0.66
----------------------------------------
```


SVM Overfitting and Underfitting Analysis
Training Accuracy
Testing Accuracy
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Exp 1
Exp 2
Exp 3
Exp 4
Experiments