Library Imports

```
from ucimlrepo import fetch_ucirepo
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
```

Fetch Dataset

```
spambase = fetch_ucirepo(id=94)
```

Metadata

```
print(spambase.metadata)
```

```
{'uci_id': 94, 'name': 'Spambase', 'repository_url':
'https://archive.ics.uci.edu/dataset/94/spambase', 'data_url':
'https://archive.ics.uci.edu/static/public/94/data.csv', 'abstract':
'Classifying Email as Spam or Non-Spam', 'area': 'Computer Science',
'tasks': ['Classification'], 'characteristics': ['Multivariate'],
'num_instances': 4601, 'num_features': 57, 'feature_types':
['Integer', 'Real'], 'demographics': [], 'target_col': ['Class'],
'index_col': None, 'has_missing_values': 'no',
'missing_values_symbol': None, 'year_of_dataset_creation': 1999,
'last_updated': 'Mon Aug 28 2023', 'dataset_doi': '10.24432/C53G6X',
'creators': ['Mark Hopkins', 'Erik Reeber', 'George Forman', 'Jaap
Suermondt'], 'intro_paper': None, 'additional_info': {'summary': 'The
"spam" concept is diverse: advertisements for products/web sites, make
money fast schemes, chain letters, pornography...\n\nThe
classification task for this dataset is to determine whether a given
email is spam or not.\n\t\nOur collection of spam e-mails came from
our postmaster and individuals who had filed spam.  Our collection of
non-spam e-mails came from filed work and personal e-mails, and hence
the word \'george\' and the area code \'650\' are indicators of non-
spam.  These are useful when constructing a personalized spam filter.
One would either have to blind such non-spam indicators or get a very
wide collection of non-spam to generate a general purpose spam
filter.\n\nFor background on spam: Cranor, Lorrie F., LaMacchia, Brian
A.  Spam!, Communications of the ACM, 41(8):74-83, 1998.\n\nTypical
performance is around ~7% misclassification error. False positives
(marking good mail as spam) are very undesirable.If we insist on zero
false positives in the training/testing set, 20-25% of the spam passed
through the filter. See also Hewlett-Packard Internal-only Technical
Report. External version forthcoming. ', 'purpose': None, 'funded_by':
```

```
None, 'instances_represent': 'Emails', 'recommended_data_splits':
None, 'sensitive_data': None, 'preprocessing_description': None,
'variable_info': 'The last column of \'spambase.data\' denotes whether
the e-mail was considered spam (1) or not (0), i.e. unsolicited
commercial e-mail.  Most of the attributes indicate whether a
particular word or character was frequently occuring in the e-mail.
The run-length attributes (55-57) measure the length of sequences of
consecutive capital letters.  For the statistical measures of each
attribute, see the end of this file.  Here are the definitions of the
attributes:\r\n\r\n48 continuous real [0,100] attributes of type
word_freq_WORD \r\n= percentage of words in the e-mail that match
WORD, i.e. 100 * (number of times the WORD appears in the e-mail) /
total number of words in e-mail.  A "word" in this case is any string
of alphanumeric characters bounded by non-alphanumeric characters or
end-of-string.\r\n\r\n6 continuous real [0,100] attributes of type
char_freq_CHAR] \r\n= percentage of characters in the e-mail that
match CHAR, i.e. 100 * (number of CHAR occurences) / total characters
in e-mail\r\n\r\n1 continuous real [1,...] attribute of type
capital_run_length_average \r\n= average length of uninterrupted
sequences of capital letters\r\n\r\n1 continuous integer [1,...]
attribute of type capital_run_length_longest \r\n= length of longest
uninterrupted sequence of capital letters\r\n\r\n1 continuous integer
[1,...] attribute of type capital_run_length_total \r\n= sum of length
of uninterrupted sequences of capital letters \r\n= total number of
capital letters in the e-mail\r\n\r\n1 nominal {0,1} class attribute
of type spam\r\n= denotes whether the e-mail was considered spam (1)
or not (0), i.e. unsolicited commercial e-mail.  \r\n', 'citation':
None}}
```

Variable Information

```
print(spambase.variables)
```

```
                          name     role        type demographic  \
0               word_freq_make  Feature  Continuous        None
1            word_freq_address  Feature  Continuous        None
2                word_freq_all  Feature  Continuous        None
3                 word_freq_3d  Feature  Continuous        None
4                word_freq_our  Feature  Continuous        None
5               word_freq_over  Feature  Continuous        None
6             word_freq_remove  Feature  Continuous        None
7           word_freq_internet  Feature  Continuous        None
8              word_freq_order  Feature  Continuous        None
9               word_freq_mail  Feature  Continuous        None
10           word_freq_receive  Feature  Continuous        None
11              word_freq_will  Feature  Continuous        None
12            word_freq_people  Feature  Continuous        None
13            word_freq_report  Feature  Continuous        None
14         word_freq_addresses  Feature  Continuous        None
```

15	word_freq_free	Feature	Continuous	None	
16	word_freq_business	Feature	Continuous	None	
17	word_freq_email	Feature	Continuous	None	
18	word_freq_you	Feature	Continuous	None	
19	word_freq_credit	Feature	Continuous	None	
20	word_freq_your	Feature	Continuous	None	
21	word_freq_font	Feature	Continuous	None	
22	word_freq_000	Feature	Continuous	None	
23	word_freq_money	Feature	Continuous	None	
24	word_freq_hp	Feature	Continuous	None	
25	word_freq_hpl	Feature	Continuous	None	
26	word_freq_george	Feature	Continuous	None	
27	word_freq_650	Feature	Continuous	None	
28	word_freq_lab	Feature	Continuous	None	
29	word_freq_labs	Feature	Continuous	None	
30	word_freq_telnet	Feature	Continuous	None	
31	word_freq_857	Feature	Continuous	None	
32	word_freq_data	Feature	Continuous	None	
33	word_freq_415	Feature	Continuous	None	
34	word_freq_85	Feature	Continuous	None	
35	word_freq_technology	Feature	Continuous	None	
36	word_freq_1999	Feature	Continuous	None	
37	word_freq_parts	Feature	Continuous	None	
38	word_freq_pm	Feature	Continuous	None	
39	word_freq_direct	Feature	Continuous	None	
40	word_freq_cs	Feature	Continuous	None	
41	word_freq_meeting	Feature	Continuous	None	
42	word_freq_original	Feature	Continuous	None	
43	word_freq_project	Feature	Continuous	None	
44	word_freq_re	Feature	Continuous	None	
45	word_freq_edu	Feature	Continuous	None	
46	word_freq_table	Feature	Continuous	None	
47	word_freq_conference	Feature	Continuous	None	
48	char_freq_;	Feature	Continuous	None	
49	char_freq_(	Feature	Continuous	None	
50	char_freq_[	Feature	Continuous	None	
51	char_freq_!	Feature	Continuous	None	
52	char_freq_$	Feature	Continuous	None	
53	char_freq_#	Feature	Continuous	None	
54	capital_run_length_average	Feature	Continuous	None	
55	capital_run_length_longest	Feature	Continuous	None	
56	capital_run_length_total	Feature	Continuous	None	
57	Class	Target	Binary	None	

	description	units	missing_values
0	None	None	no
1	None	None	no
2	None	None	no
3	None	None	no

4	None	None	no
5	None	None	no
6	None	None	no
7	None	None	no
8	None	None	no
9	None	None	no
10	None	None	no
11	None	None	no
12	None	None	no
13	None	None	no
14	None	None	no
15	None	None	no
16	None	None	no
17	None	None	no
18	None	None	no
19	None	None	no
20	None	None	no
21	None	None	no
22	None	None	no
23	None	None	no
24	None	None	no
25	None	None	no
26	None	None	no
27	None	None	no
28	None	None	no
29	None	None	no
30	None	None	no
31	None	None	no
32	None	None	no
33	None	None	no
34	None	None	no
35	None	None	no
36	None	None	no
37	None	None	no
38	None	None	no
39	None	None	no
40	None	None	no
41	None	None	no
42	None	None	no
43	None	None	no
44	None	None	no
45	None	None	no
46	None	None	no
47	None	None	no
48	None	None	no
49	None	None	no
50	None	None	no
51	None	None	no
52	None	None	no

```
53                        None  None              no
54                        None  None              no
55                        None  None              no
56                        None  None              no
57  spam (1) or not spam (0)  None              no
```

Pandas Dataframes

```
x = spambase.data.features
y = spambase.data.targets
```

Part A: SVM Implementation

Train-Test-Split

```
x_train, x_test, y_train, y_test = train_test_split(x,
y,test_size=0.2, random_state=42)
```

Vectorize X_Train and Y_Train

```
X_Train=x_train.to_numpy()
Y_Train=y_train.to_numpy().T[0]
```

Vectorize X_Test and Y_Test

```
X_Test=x_test.to_numpy()
Y_Test=y_test.to_numpy().T[0]
```

Predict Function

```
def predict(model,data):
    Y_Pred=[]
    for x in data:
        Y_Pred.append(model.predict([x])[0])
    return np.array(Y_Pred)
```

Metrics Function

```
def metrics(Y_Pred,Y_Test):
    accuracy, precision, recall, F1score = (0,0,0,0)
    TP,TN,FP,FN = (0,0,0,0)
    N=Y_Pred.shape[0]
    for i in range(0,N):
        if(Y_Pred[i]==Y_Test[i]):
            if(Y_Pred[i]==1):
                TP+=1
            else:
                TN+=1
```

```
        elif(Y_Pred[i]==1):
            FP+=1
        else:
            FN+=1
    if(TP+FP+TN+FN!=0):
        accuracy= (TP+TN)/(TP+FP+TN+FN)
    if(TP+FP!=0):
        precision= TP/(TP+FP)
    if(TP+FN!=0):
        recall= TP/(TP+FN)
    if(precision+recall!=0):
        F1score= (2*precision*recall)/(precision+recall)
    return accuracy,precision,recall,F1score
```

Train Function

```
def train(model, X_Train, Y_Train, X_Test, Y_Test):
    model.fit(X_Train,Y_Train)
    Y_Pred=predict(model,X_Test)
    accuracy, precision, recall, F1score = metrics(Y_Pred,Y_Test)
    return accuracy, precision, recall, F1score
```

Linear-SVM model

```
svm_model = SVC(kernel='linear')
scaling = MinMaxScaler(feature_range=(-1,1)).fit(X_Train)
X_train = scaling.transform(X_Train)
X_test = scaling.transform(X_Test)
```

Prediction and Evaluation

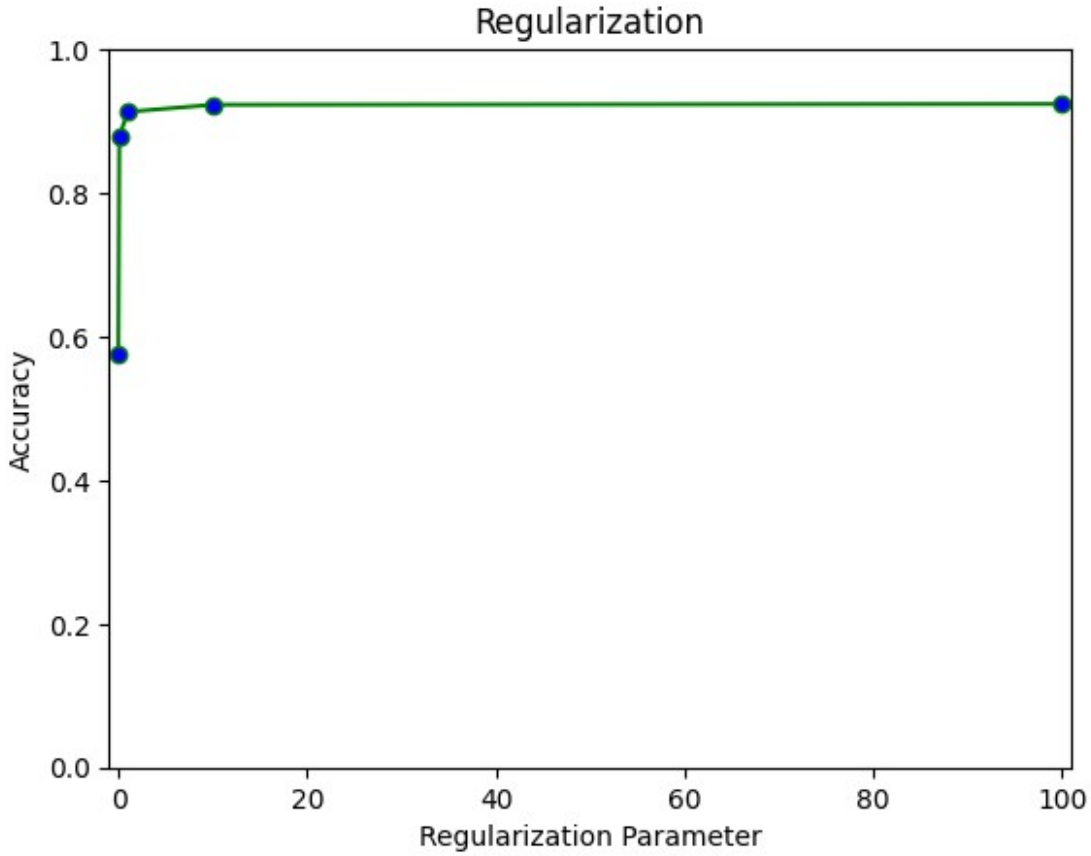
```
accuracy, precision, recall, F1score =
train(svm_model,X_train,Y_Train,X_test,Y_Test)

print(f"Accuracy: {accuracy}\nPrecision: {precision}\nRecall:
{recall}\nF1-Score: {F1score}")
```

```
Accuracy: 0.9131378935939196
Precision: 0.9329608938547486
Recall: 0.8564102564102564
F1-Score: 0.8930481283422459
```

Regularization

```
C=[0.001, 0.1, 1, 10, 100]
Accuracy=[]
for val in C:
    model = SVC(kernel='linear', C=val)
```

```
    accuracy, precision, recall, F1score =
train(model,X_train,Y_Train,X_test,Y_Test)
    Accuracy.append(accuracy)

Accuracy

[0.5765472312703583,
 0.8783930510314875,
 0.9131378935939196,
 0.9229098805646037,
 0.9239956568946797]

plt.plot(C, Accuracy,color='green',marker='o', markerfacecolor='blue')

plt.ylim(0,1)
plt.xlim(-1,101)
plt.xlabel('Regularization Parameter')
plt.ylabel('Accuracy')
plt.title('Regularization')
plt.show()
```



Part B: Kernel Tricks

Polynomial Kernel Degree 2

```
model = SVC(kernel='poly', degree=2)
accuracy, precision, recall, F1score =
train(model,X_train,Y_Train,X_test,Y_Test)
print(f"Accuracy: {accuracy}\nPrecision: {precision}\nRecall:
{recall}\nF1-Score: {F1score}")
```

```
Accuracy: 0.9370249728555917
Precision: 0.9391534391534392
Recall: 0.9102564102564102
F1-Score: 0.9244791666666666
```

Polynomial Kernel Degree 3

```
model = SVC(kernel='poly', degree=3)
accuracy, precision, recall, F1score =
train(model,X_train,Y_Train,X_test,Y_Test)
print(f"Accuracy: {accuracy}\nPrecision: {precision}\nRecall:
{recall}\nF1-Score: {F1score}")
```

```
Accuracy: 0.9348534201954397
Precision: 0.9411764705882353
Recall: 0.9025641025641026
F1-Score: 0.9214659685863874
```

Sigmoid

```
model = SVC(kernel='sigmoid')
accuracy, precision, recall, F1score =
train(model,X_train,Y_Train,X_test,Y_Test)
print(f"Accuracy: {accuracy}\nPrecision: {precision}\nRecall:
{recall}\nF1-Score: {F1score}")
```

```
Accuracy: 0.5765472312703583
Precision: 0
Recall: 0.0
F1-Score: 0
```

RBF

```
model = SVC(kernel='rbf')
accuracy, precision, recall, F1score =
train(model,X_train,Y_Train,X_test,Y_Test)
print(f"Accuracy: {accuracy}\nPrecision: {precision}\nRecall:
{recall}\nF1-Score: {F1score}")
```

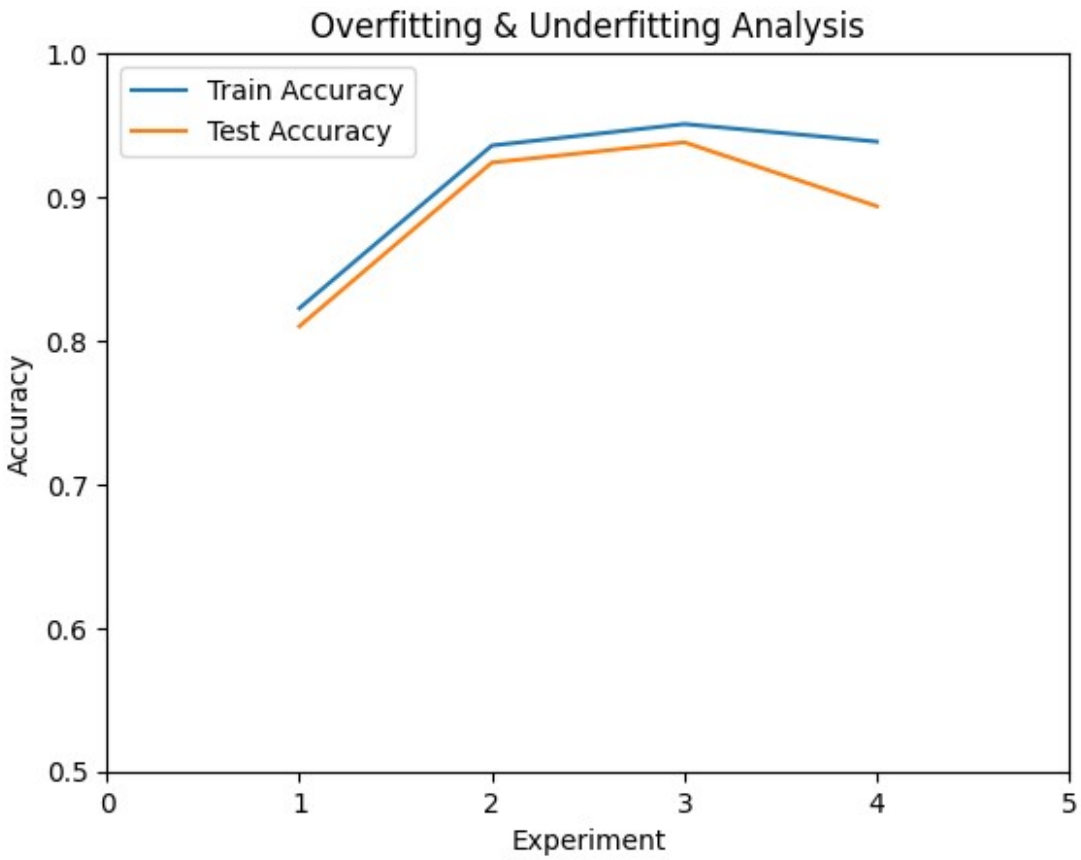
```
Accuracy: 0.9326818675352877
Precision: 0.9530386740331491
Recall: 0.8846153846153846
F1-Score: 0.9175531914893617
```

Part C: Overfitting & Underfitting Analysis

```
def train_metrics(model,x_train,y_train):
    y_pred=predict(model,x_train)
    accuracy, precision, recall, F1score = metrics(y_pred,y_train)
    return accuracy, precision, recall, F1score

D=[1,3]
C=[0.01,100]
pdegree=[]
cv=[]
train_accuracy=[]
test_accuracy=[]
for d in D:
    for c in C:
        pdegree.append(d)
        cv.append(c)
        model = SVC(kernel='poly', degree=d, C=c)
        accuracy, precision, recall, F1score =
train(model,X_train,Y_Train,X_test,Y_Test)
        test_accuracy.append(accuracy)
        accuracy, precision, recall, F1score =
train_metrics(model,X_train,Y_Train)
        train_accuracy.append(accuracy)
exp_result=pd.DataFrame({
    'Polynomial Degree': pdegree,
    'C Value': cv,
    'Training Accuracy': train_accuracy,
    'Test Accuracy': test_accuracy
})

print(exp_result)
```

	Polynomial Degree	C Value	Training Accuracy	Test Accuracy
0	1	0.01	0.822554	0.809989
1	1	100.00	0.935870	0.923996
2	3	0.01	0.950815	0.938111
3	3	100.00	0.938587	0.893594

```
N=len(D)*len(C)
X_Label=[i for i in range(1,N+1)]

plt.plot(X_Label, train_accuracy, label = "Train Accuracy")
plt.plot(X_Label, test_accuracy, label = "Test Accuracy")
plt.ylim(0.5,1)
plt.xlim(0,N+1)
plt.xlabel('Experiment')
plt.ylabel('Accuracy')
plt.title('Overfitting & Underfitting Analysis')
plt.legend()
plt.show()
```


Overfitting & Underfitting Analysis
Train Accuracy
Test Accuracy
Accuracy
Experiment
1.0
0.9
0.8
0.7
0.6
0.5
0
1
2
3
4
5