```
!pip install ucimlrepo
from ucimlrepo import fetch_ucirepo
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
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
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.3-py3-none-any.whl (7.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.3
```

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

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

```
# variable information
print(spambase.variables)
```

```
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
```

Resources ×

You are not subscribed. Learn more.
You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units here.
Manage sessions

Want more memory and disk space?
Upgrade to Colab Pro

Python 3 Google Compute Engine backend (GPU)
Showing resources from 11:00 AM to 12:35 PM

System RAM
1.4 / 12.7 GB

GPU RAM

Disk
27.0 / 78.2 GB

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

```
Accuracy=[]
regularization=[0,0.001,0.1,1,10,100]
```

```
print(y_train.shape)
y_train = y_train.to_numpy().ravel()
y_test = y_test.to_numpy().ravel()
```

```
(3680, 1)
```

```
def metrics(y_test,y_pred):
    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)

    # Calculate precision
    precision = precision_score(y_test, y_pred)

    # Calculate recall
    recall = recall_score(y_test, y_pred)

    # Calculate F1-score
    f1 = f1_score(y_test, y_pred)

    print("Accuracy:", accuracy)
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1-Score:", f1)
    Accuracy.append(accuracy)
```

Without Regularization

```
svm_classifier = SVC(kernel='linear', random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
```

```
Accuracy: 0.9457111834961998
Precision: 0.9393139841688655
Recall: 0.9295039164490861
F1-Score: 0.9343832020997375
```

Regularization 0.001

```
svm_classifier = SVC(C=0.001, kernel='linear', random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
metrics(y_test,y_pred)
```

```
Accuracy: 0.8849077090119435
Precision: 0.9037900874635568
Recall: 0.8093994778067886
F1-Score: 0.8539944903581267
```

Regularization 0.01

```
svm_classifier = SVC(C=0.1, kernel='linear', random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
metrics(y_test,y_pred)
```

```
Accuracy: 0.9446254071661238
Precision: 0.9414893617021277
Recall: 0.9242819843342036
F1-Score: 0.932806324110672
```

Regularization 1

```
svm_classifier = SVC(C=1, kernel='linear', random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
metrics(y_test,y_pred)
```

```
Accuracy: 0.9457111834961998
Precision: 0.9393139841688655
Recall: 0.9295039164490861
F1-Score: 0.9343832020997375
```

regularization 10

```
svm_classifier = SVC(C=10, kernel='linear', random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
metrics(y_test,y_pred)
```

```
Accuracy: 0.9424538545059717
Precision: 0.9319371727748691
Recall: 0.9295039164490861
F1-Score: 0.930718954248366
```

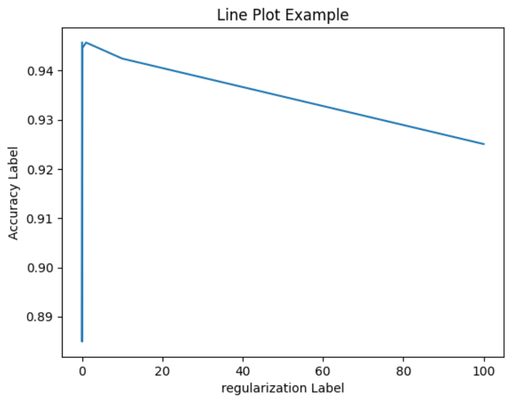
regularization 100

```
svm_classifier = SVC(C=100, kernel='linear', random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
metrics(y_test,y_pred)
```

```
Accuracy: 0.9250814332247557
Precision: 0.9005102040816326
Recall: 0.9216710182767625
F1-Score: 0.9109677419354839
```

```
plt.plot(regularization, Accuracy)
plt.xlabel('regularization Label')
plt.ylabel('Accuracy Label')
plt.title('Line Plot Example')
plt.show()
```



kernel Tricks

```
Accuracy=[]
x_labels=['poly degree 2','poly degree 3','sigmoid','RBF']
```

```
svm_classifier = SVC(C=1, kernel='poly',degree = 2, random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
```

```
Accuracy: 0.6579804560260586
Precision: 0.8695652173913043
Recall: 0.20887728459530025
F1-Score: 0.3368421052631579
```

```
svm_classifier = SVC(C=1, kernel='poly',degree = 3, random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
```

```
Accuracy: 0.6362649294245385
Precision: 0.9615384615384616
Recall: 0.13054830287206268
F1-Score: 0.2298850574712644
```

```
svm_classifier = SVC(C=1, kernel='sigmoid', gamma=0.1, coef0=0.5, random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
```

```
Accuracy: 0.5331161780673181
Precision: 0.02040816326530612
Recall: 0.0026109660574412533
F1-Score: 0.00462962962962963
```

```
svm_classifier = SVC(C=1, kernel='rbf', random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
```

```
Accuracy: 0.7176981541802389
Precision: 0.7530864197530864
Recall: 0.47780678851174935
F1-Score: 0.584664536741214
```

Part C

```
Accuracy=[]
training_accuracy=[]

x_labels=['poly degree 1 c=0.01','poly degree 1 c=100','poly degree 3 c=0.01','poly d
```

```
svm_classifier = SVC(C=0.01, kernel='poly',degree = 1, random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
y_pred = svm_classifier.predict(X_train)
training_accuracy.append(accuracy_score(y_train, y_pred))
```

```
Accuracy: 0.6471226927252985
Precision: 0.8152173913043478
Recall: 0.195822454308094
F1-Score: 0.3157894736842105
```

```
svm_classifier = SVC(C=100, kernel='poly',degree = 1, random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
y_pred = svm_classifier.predict(X_train)
training_accuracy.append(accuracy_score(y_train, y_pred))
```

```
Accuracy: 0.7861020629750272
Precision: 0.8690476190476191
Recall: 0.5718015665796344
F1-Score: 0.6897637795275591
```

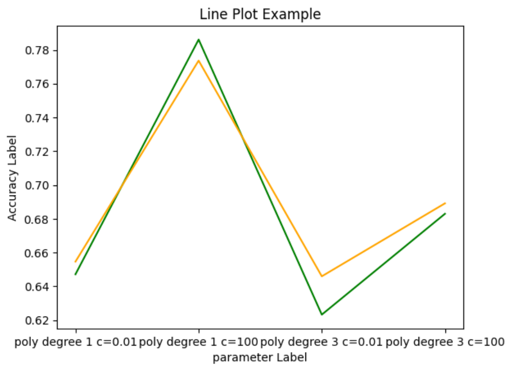
```
svm_classifier = SVC(C=0.01, kernel='poly',degree = 3, random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
y_pred = svm_classifier.predict(X_train)
training_accuracy.append(accuracy_score(y_train, y_pred))
```

```
     Accuracy: 0.6232356134636265
     Precision: 0.95
     Recall: 0.09921671018276762
     F1-Score: 0.1796690307328605
```

```
svm_classifier = SVC(C=100, kernel='poly',degree = 3, random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
y_pred = svm_classifier.predict(X_train)
training_accuracy.append(accuracy_score(y_train, y_pred))
```

```
     Accuracy: 0.6829533116178067
     Precision: 0.9690721649484536
     Recall: 0.2454308093994778
     F1-Score: 0.39166666666666666
```

```
plt.plot(x_labels, Accuracy,'green')
plt.plot(x_labels, training_accuracy,'orange')
plt.xlabel('parameter Label')
plt.ylabel('Accuracy Label')
plt.title('Line Plot Example')
plt.show()
```



Change runtime type