In [1]:
```
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
```

In [2]:
```
# fetch dataset
spambase = fetch_ucirepo(id=94)
```

In [3]:
```
# data (as pandas dataframes)
X = spambase.data.features
y = spambase.data.targets
```

In [4]:
```
# 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
```

In [42]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [57]:
```
Accuracy=[]
regularization=[0.001,0.1,1,10,100]
```

In [44]:
```
print(y_train.shape)
y_train = y_train.to_numpy().ravel()
y_test = y_test.to_numpy().ravel()
```

```
(3680, 1)
```

In [45]:
```
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)
```

Regularization 0.001

In [58]:
```
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)
```

In [59]:
```
metrics(y_test,y_pred)
```

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

Regularization 0.01

In [60]:
```
svm_classifier = SVC(C=0.1, kernel='linear', random_state=42,max_iter=5000)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\svm\_b
ase.py:255: ConvergenceWarning: Solver terminated early (max_iter=5000).  Consider
pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn('Solver terminated early (max_iter=%i).'
```

In [61]:
```
metrics(y_test,y_pred)
```

```
Accuracy: 0.8458197611292074
Precision: 0.7405189620758483
Recall: 0.9686684073107049
F1-Score: 0.8393665158371041
```

Regularization 1

In [62]:
```
svm_classifier = SVC(C=1, kernel='linear', random_state=42,max_iter=10000)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\svm\_b
ase.py:255: ConvergenceWarning: Solver terminated early (max_iter=10000).  Consider
pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn('Solver terminated early (max_iter=%i).'
```

In [63]:
```
metrics(y_test,y_pred)
```

```
Accuracy: 0.6764386536373507
Precision: 0.912621359223301
Recall: 0.2454308093994778
F1-Score: 0.38683127572016457
```

regularization 10

In [64]:
```
svm_classifier = SVC(C=10, kernel='linear', random_state=42,max_iter=3000)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\svm\_b
ase.py:255: ConvergenceWarning: Solver terminated early (max_iter=10000).  Consider
pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn('Solver terminated early (max_iter=%i).'
```

In [65]:
```
metrics(y_test,y_pred)
```

```
Accuracy: 0.38436482084690554
Precision: 0.383248730964467
Recall: 0.7885117493472585
F1-Score: 0.515798462852263
```

regularization 100

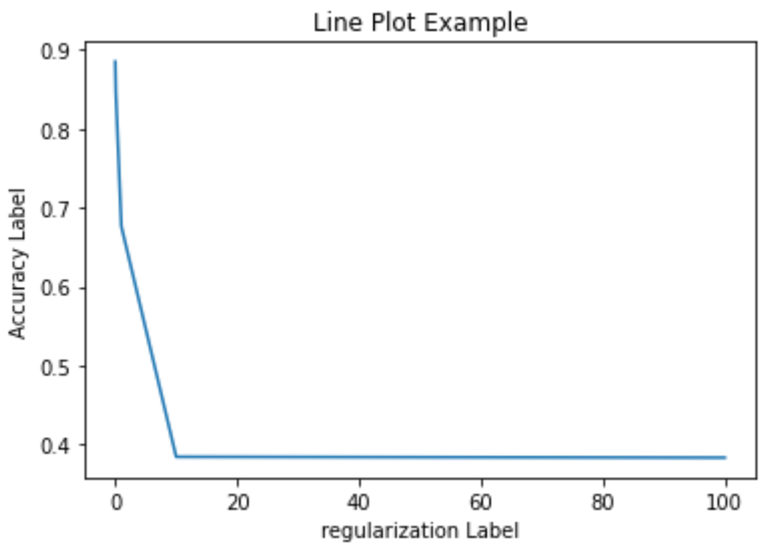
In [66]:
```
svm_classifier = SVC(C=100, kernel='linear', random_state=42,max_iter=10000)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\svm\_b
ase.py:255: ConvergenceWarning: Solver terminated early (max_iter=10000).  Consider
pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn('Solver terminated early (max_iter=%i).'
```

In [67]:
```
metrics(y_test,y_pred)
```

```
Accuracy: 0.38327904451682954
Precision: 0.3878787878787879
Recall: 0.835509138381201
F1-Score: 0.5298013245033113
```

In [68]:
```
plt.plot(regularization, Accuracy)
plt.xlabel('regularization Label')
plt.ylabel('Accuracy Label')
plt.title('Line Plot Example')
plt.show()
```



kernel Tricks

In [69]:
```
Accuracy=[]
x_labels=['poly degree 2','poly degree 3','sigmoid','RBF']
```

In [70]:
```
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
```

In [71]:
```
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
```

In [72]:
```
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
```

In [73]:
```
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

In [80]:
```
Accuracy=[]
training_accuracy=[]

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

In [81]:
```
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
```

In [82]:
```
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
```

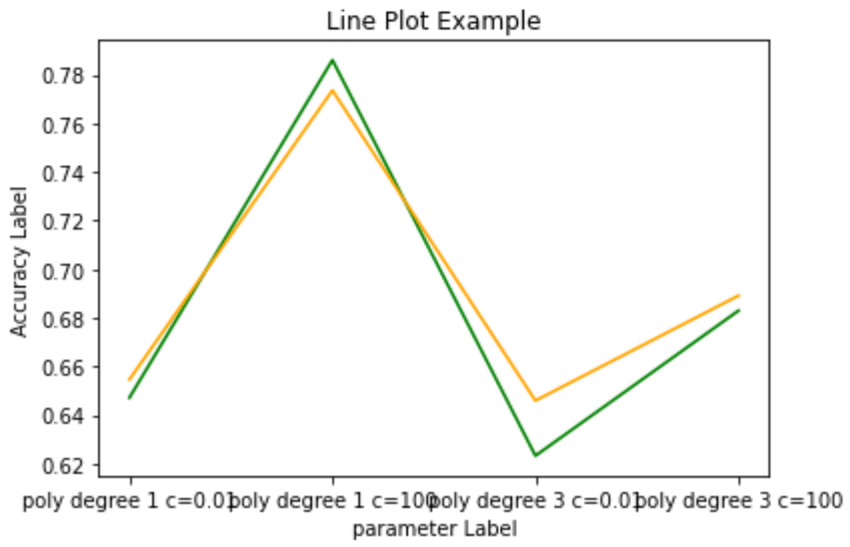
In [83]:
```
svm_classifier = SVC(C=0.01, kernel='poly',degree = 3, random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
metrics(y_test,y_pred)
y_pred = svm_classifier.predict(X_train)
training_accuracy.append(accuracy_score(y_train, y_pred))
```

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

In [84]:
```
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
```

In [86]:
```
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()
```


Line Plot Example
Accuracy Label
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
poly degree 1 c=0.01
poly degree 1 c=100
poly degree 3 c=0.01
poly degree 3 c=100
parameter Label

```
In [ ]:
```