Support Vector Machine (SVM)

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
%pylab inline
pylab.rcParams['figure.figsize'] = (10, 6)
```

```
%pylab is deprecated, use %matplotlib inline and import the required
libraries.
Populating the interactive namespace from numpy and matplotlib
```

Importing the dataset

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split  # <-- Make sure
to import this!
from sklearn.metrics import accuracy_score

url =
"https://gist.githubusercontent.com/dinukasaminda/59a5953ae00ca74485f2
52759073f959/raw/sample-dataset-for-clustering.csv"
df = pd.read_csv(url)

print(df.head())
```

```
    X   Y
0  48  35
1  48 -23
2  -6  42
3  17  40
4  -9  29
```

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=.3, random_state=0)

print('There are {} samples in the training set and {} samples in the
```

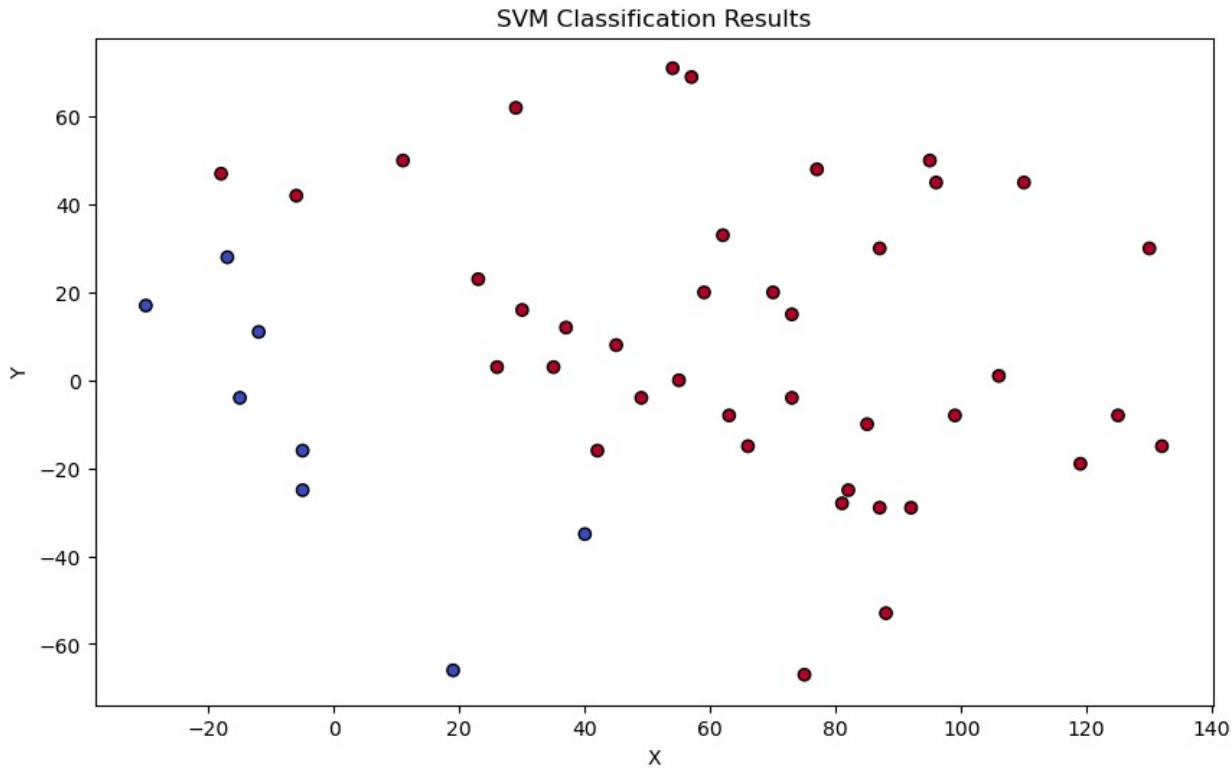
```
test set'.format(
X_train.shape[0], X_test.shape[0]))
```

```
There are 112 samples in the training set and 48 samples in the test
set
```

Plot the original Data

```
X = df[['X', 'Y']]
y = (df['X'] + df['Y'] > 10).astype(int)  # Create a simple binary
target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Train the SVM model
model = SVC(kernel='linear')
model.fit(X_train, y_train)

# Make predictions and evaluate
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))

# Visualize results
plt.scatter(X_test['X'], X_test['Y'], c=y_pred, cmap='coolwarm',
edgecolor='k')
plt.title("SVM Classification Results")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

```
Accuracy: 0.9583333333333334
```




```
from sklearn.svm import SVC

svm = SVC(kernel='rbf', random_state=0, gamma=.10, C=1.0)
svm.fit(X_train_std, y_train)
```

```
SVC(gamma=0.1, random_state=0)
```

```
print("Support Vector for model are :",svm.support_vectors_)
```

```
Support Vector for model are : [[-1.55024648 -0.68201676]
 [-1.94961992  0.82542825]
 [-2.12434581 -0.00626555]
 [ 0.12212983 -1.64366272]
 [-1.15087303  0.01972488]
 [-0.62669538 -0.83795935]
 [-0.30220446 -1.07187323]
 [-0.0775569  -1.69564358]
 [-1.22575555 -0.65602633]
 [-1.65008984 -0.26616986]
 [ 0.34677739 -1.79960531]
 [-0.65165623 -0.81196892]
 [-1.79985488 -0.34414116]
 [-1.77489404 -0.08423684]
 [ 0.79607252 -2.11149048]
 [-0.35212614 -0.99390194]
 [-0.12747858 -1.66965315]
```

```
 [-0.90126463 -0.75998806]
 [-0.40204782 -0.13621771]
 [-1.60016816  0.74745696]
 [-1.89969824  1.26726558]
 [ 0.34677739 -1.22781582]
 [-1.30063807  0.27962919]
 [-0.67661707 -0.24017943]
 [-0.52685202 -0.08423684]
 [-0.4519695  -0.57805504]
 [-0.77646043 -0.18819857]
 [-1.02606883 -0.00626555]
 [-1.20079471  0.30561962]
 [ 0.32181655 -1.20182539]
 [ 0.89591588 -1.30578711]
 [ 0.32181655 -0.70800719]
 [-0.87630379  0.17566747]
 [-0.17740026 -0.60404547]
 [ 0.32181655 -0.73399763]
 [-0.42700866 -0.70800719]]
```

```
print("Number of suppoort Vectors of each class 0 : -
",svm.n_support_[0])
print("Number of suppoort Vectors of each class 1 : -
",svm.n_support_[1])
print("Number of suppoort Vectors of each class 2 : -
",svm.n_support_[2])
```

```
Number of suppoort Vectors of each class 0 : -  18
Number of suppoort Vectors of each class 1 : -  18
```

```
---------------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
last)
Cell In[16], line 3
      1 print("Number of suppoort Vectors of each class 0 : -
",svm.n_support_[0])
      2 print("Number of suppoort Vectors of each class 1 : -
",svm.n_support_[1])
----> 3 print("Number of suppoort Vectors of each class 2 : -
",svm.n_support_[2])

IndexError: index 2 is out of bounds for axis 0 with size 2
```

```
print("Indices for support vectors are : ",svm.support_)
```

```
Indices for support vectors are :  [  3   6   7  17  31  41  42  51
52  61  62  63  73  77  79  83  87  97
   1  12  14  24  27  33  37  43  54  59  71  82  88  89  94  96 101
103]
```

```
print('The accuracy of the svm classifier on training data is {:.2f}
out of 1'.format(svm.score(X_train_std, y_train)))

print('The accuracy of the svm classifier on test data is {:.2f} out
of 1'.format(svm.score(X_test_std, y_test)))
```

```
The accuracy of the svm classifier on training data is 0.98 out of 1
The accuracy of the svm classifier on test data is 0.98 out of 1
```

```
from sklearn.metrics import confusion_matrix

# Assuming your SVM model is called svm and X_test_std is your test
data
cm = confusion_matrix(y_test, svm.predict(X_test_std))
print(cm)
```

```
[[ 7  1]
 [ 0 40]]
```

```
import sklearn.metrics as metrics

Accuracy = metrics.accuracy_score(y_test, svm.predict(X_test_std))
Precision = metrics.precision_score(y_test, svm.predict(X_test_std),
average='macro')
Sensitivity_recall = metrics.recall_score(y_test,
svm.predict(X_test_std), average='macro')
Specificity = metrics.recall_score(y_test, svm.predict(X_test_std),
pos_label=0, average='macro')
F1_score = metrics.f1_score(y_test, svm.predict(X_test_std),
average='macro')

print(f"Accuracy: {Accuracy}")
print(f"Precision: {Precision}")
print(f"Sensitivity (Recall): {Sensitivity_recall}")
print(f"Specificity: {Specificity}")
print(f"F1 Score: {F1_score}")
```

```
Accuracy: 0.9791666666666666
Precision: 0.9878048780487805
Sensitivity (Recall): 0.9375
Specificity: 0.9375
F1 Score: 0.9604938271604938
```

```
C:\Users\LENOVO\AppData\Roaming\Python\Python312\site-packages\
sklearn\metrics\_classification.py:1618: UserWarning: Note that
pos_label (set to 0) is ignored when average != 'binary' (got
'macro'). You may use labels=[pos_label] to specify a single positive
class.
  warnings.warn(
```