Machine Learning

Lab Sheet 6

Support Vector Machine

S Abhishek

AM.EN.U4CSE19147

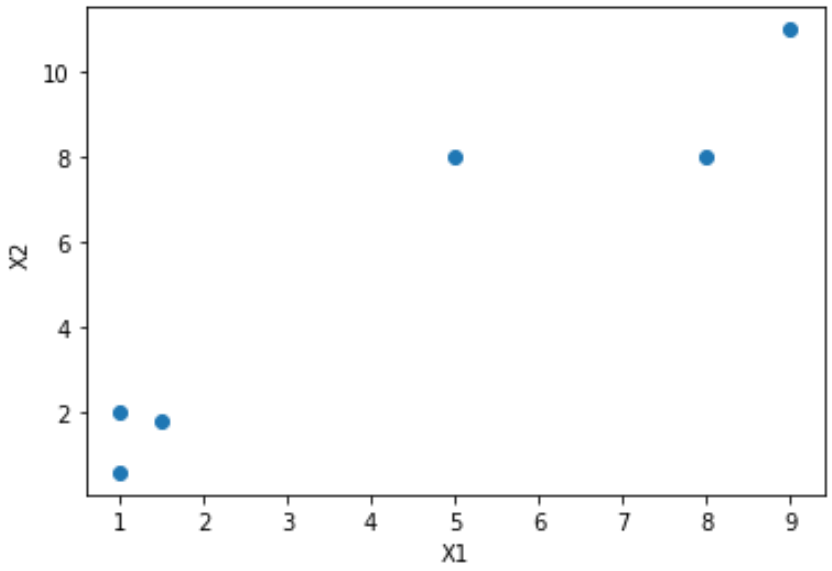
Colab Link

1 - Consider following two featured dataset

X1 = [1, 5, 1.5, 8, 1, 9]
X2 = [2, 8, 1.8, 8, 0.6, 11]

Plot the dataset using scatter plot and define SVM classifier

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

x1 = np.array([1, 5, 1.5, 8, 1, 9])
x2 = np.array([2, 8, 1.8, 8, 0.6, 11])
plt.scatter(x1,x2)
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()
```




```
y = np.array([0,1,0,1,0,1])

data = np.column_stack((x1,x2,y))

data
```

```
array([[ 1. ,  2. ,  0. ],
       [ 5. ,  8. ,  1. ],
       [ 1.5,  1.8,  0. ],
       [ 8. ,  8. ,  1. ],
       [ 1. ,  0.6,  0. ],
       [ 9. , 11. ,  1. ]])
```

```
X = data[:,0:2]
y = data[:,-1]
```

```
X
```

```
array([[ 1. ,  2. ],
       [ 5. ,  8. ],
       [ 1.5,  1.8],
       [ 8. ,  8. ],
       [ 1. ,  0.6],
       [ 9. , 11. ]])
```

```
y
```

```
array([0., 1., 0., 1., 0., 1.])
```

```
from sklearn import svm
clf = svm.SVC(kernel = 'linear',C=1.0)
clf.fit(X,y)

print(clf.predict([[0.58,0.76]]))
print(clf.predict([[10.58,10.76]]))
```

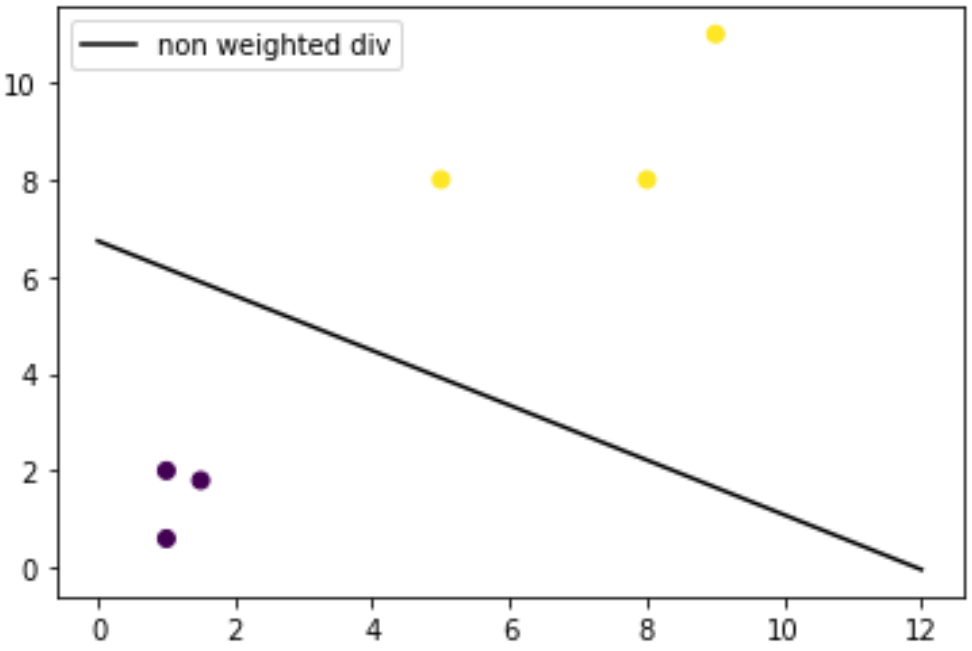
```
[0.]
[1.]
```

2 - Visualize the data

```
w = clf.coef_[0]

print(w)

a = -w[0] / w[1]

xx = np.linspace(0,12)

yy = a * xx - clf.intercept_[0] / w[1]
```

```
h0 = plt.plot(xx, yy, 'k-', label="non weighted div")
plt.scatter(X[:, 0], X[:, 1], c = y)
plt.legend()
plt.show()
```

```
[0.1380943  0.24462418]
```



3 - Support Vector Machine

```
from sklearn import datasets
from sklearn import metrics
from sklearn.svm import SVC

dataset = datasets.load_iris()
```

```
print("Features: ", dataset.feature_names)
print("Labels: ", dataset.target_names)
```

```
Features:  ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Labels:  ['setosa' 'versicolor' 'virginica']
```

```
model = SVC()
model.fit(dataset.data, dataset.target);
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
print("Accuracy:",metrics.accuracy_score(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.96      0.96      0.96        50
           2       0.96      0.96      0.96        50

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150


[[50  0  0]
 [ 0 48  2]
 [ 0  2 48]]

Accuracy: 0.9733333333333334
```

4 - Compare SVM with KNN and Logistic Regression

```
from sklearn import datasets
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
dataset = datasets.load_iris()
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(dataset.data, dataset.target)
expected = dataset.target
predicted = knn.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
print("\nAccuracy:",metrics.accuracy_score(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.94      0.94      0.94        50
           2       0.94      0.94      0.94        50

    accuracy                           0.96       150
   macro avg       0.96      0.96      0.96       150
weighted avg       0.96      0.96      0.96       150


[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]


Accuracy: 0.96
```

```
from sklearn import datasets
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
dataset = datasets.load_iris()
log_reg = LogisticRegression()
log_reg.fit(dataset.data, dataset.target);
expected = dataset.target
predicted = log_reg.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
print("\nAccuracy:",metrics.accuracy_score(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.98      0.94      0.96        50
           2       0.94      0.98      0.96        50

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150


[[50  0  0]
 [ 0 47  3]
 [ 0  1 49]]


Accuracy: 0.9733333333333334
```

5 - Kernel - Radial Basis Function (RBF) and Polynomial Kernel with degree 3.

```
from sklearn import datasets
from sklearn import metrics
from sklearn.svm import SVC
dataset = datasets.load_iris()
model = SVC(kernel='rbf', gamma=0.7, C=5.0)
model.fit(dataset.data, dataset.target)
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
print("\nAccuracy:",metrics.accuracy_score(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      0.92      0.96        50
           2       0.93      1.00      0.96        50

    accuracy                           0.97       150
   macro avg       0.98      0.97      0.97       150
weighted avg       0.98      0.97      0.97       150

[[50  0  0]
 [ 0 46  4]
 [ 0  0 50]]


Accuracy: 0.9733333333333334
```

```
from sklearn import datasets
from sklearn import metrics
from sklearn.svm import SVC
dataset = datasets.load_iris()
model = SVC(kernel='poly', degree=3, C=5.0)
model.fit(dataset.data, dataset.target)
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
print("\nAccuracy:",metrics.accuracy_score(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.98      0.96      0.97        50
           2       0.96      0.98      0.97        50

    accuracy                           0.98       150
   macro avg       0.98      0.98      0.98       150
weighted avg       0.98      0.98      0.98       150


[[50  0  0]
 [ 0 48  2]
 [ 0  1 49]]


Accuracy: 0.98
```

6 - Does the accuracy change if the kernel function changes?

- Yes, the accuracy changes if the kernel function changes.
- In our case, when the kernal is RBF we get the accuracy of 97% and when the kernal is Polynomial Kernel with degree 3, we get the accuracy of 98%.

Thankyou!!