svc_arcene

November 20, 2019

1 SVC

[20]:
```
import pandas as pd
import os
import matplotlib.pyplot as plt
import numpy as np
from sklearn.svm import SVC
from timeit import default_timer as timer
from sklearn import svm
from sklearn.metrics import␣
 ↪confusion_matrix,accuracy_score,recall_score,precision_score,f1_score,classification_report
from sklearn.svm import LinearSVC
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

def load_data(dataset):

    f = open(os.path.dirname("__file__") + '%s_train.data' % dataset)
    X = np.fromfile(f, dtype=np.float64, sep=' ')
    f.close()

    f = open(os.path.dirname("__file__") + '%s_train.labels' % dataset)
    y = np.fromfile(f, dtype=np.int32, sep=' ')
    f.close()

    f = open(os.path.dirname("__file__") + '%s_test.data' % dataset)
    test = np.fromfile(f, dtype=np.float64, sep=' ')
    f.close()

    f = open(os.path.dirname("__file__") + '%s_valid.data' % dataset)
    valid_data = np.fromfile(f, dtype=np.float64, sep=' ')
    f.close()

    f = open(os.path.dirname("__file__") + '%s_valid.labels' % dataset)
    valid_labels= np.fromfile(f, dtype=np.float64, sep=' ')
    f.close()
```

```
    X = X.reshape(-1, 10000)
    valid_data = T.reshape(-1, 10000)
    test = test.reshape(-1, 10000)


    return  X, y, test, valid_data, valid_labels

X, y, test, valid_data, valid_labels = load_data('arcene')

X
```

[20]:
```
array([[  0.,  71.,   0., ...,   0.,   0., 524.],
       [  0.,  41.,  82., ...,   0., 284., 423.],
       [  0.,   0.,   1., ...,   0.,  34., 508.],
       ...,
       [  2.,  15.,  48., ...,   0.,   0., 453.],
       [  8.,   0.,  38., ...,   0., 189., 403.],
       [  0.,   0.,   0., ...,   0.,  10., 365.]])
```

[13]:
```
y
```

[13]:
```
array([ 1, -1,  1,  1, -1, -1,  1, -1, -1, -1, -1,  1, -1,  1, -1,  1, -1,
       -1, -1, -1, -1, -1, -1,  1, -1, -1,  1, -1,  1, -1,  1,  1,  1, -1,
       -1,  1, -1, -1,  1, -1,  1, -1, -1,  1, -1, -1, -1, -1,  1,  1, -1,
        1, -1, -1,  1, -1,  1,  1,  1, -1,  1,  1, -1,  1, -1, -1, -1, -1,
        1,  1, -1,  1, -1, -1,  1, -1, -1,  1, -1,  1,  1,  1, -1,  1,  1,
       -1,  1,  1, -1, -1,  1, -1,  1,  1, -1, -1, -1,  1, -1,  1])
```

[14]:
```
valid_data
```

[14]:
```
array([[  0.,   0., 156., ...,   0.,   0., 465.],
       [  0.,   7.,   0., ...,   0.,  34., 199.],
       [  0.,  32.,   0., ...,   0.,  47., 219.],
       ...,
       [ 93.,  32., 137., ...,   0., 276., 312.],
       [119.,  12., 198., ...,   0.,   0., 350.],
       [112.,  19., 171., ...,   0.,   0., 367.]])
```

[15]:
```
valid_labels
```

[15]:
```
array([-1., -1., -1.,  1.,  1.,  1., -1.,  1., -1., -1.,  1., -1., -1.,
        1., -1., -1.,  1.,  1.,  1.,  1.,  1., -1.,  1., -1.,  1., -1.,
       -1., -1., -1., -1., -1., -1.,  1.,  1.,  1., -1., -1., -1.,  1.,
       -1., -1.,  1., -1., -1.,  1., -1., -1.,  1.,  1., -1., -1.,  1.,
        1., -1., -1., -1., -1.,  1.,  1.,  1., -1.,  1.,  1., -1., -1.,
       -1., -1.,  1., -1., -1.,  1., -1.,  1., -1., -1.,  1., -1., -1.,
        1.,  1.,  1., -1., -1.,  1.,  1., -1.,  1.,  1., -1., -1.,  1.,
       -1.,  1.,  1., -1., -1., -1.,  1.,  1., -1.])
```

[25]:
```
sc=SVC(kernel='poly', degree=2, gamma=8, probability=True)
sc.fit(X, y)
```

```
valid_pred = sc.predict(valid_data)
print("Accuracy score: {0}".format(accuracy_score(valid_labels,valid_pred)))

test_labels = sc.predict(test)

test_labels
```

```
Accuracy score: 0.84
```

[25]:
```
array([-1, -1, -1,  1, -1, -1, -1, -1,  1,  1, -1, -1, -1,  1,  1,  1, -1,
        1, -1,  1,  1,  1, -1, -1,  1,  1,  1,  1, -1, -1,  1, -1,  1,  1,
       -1,  1, -1, -1, -1,  1,  1, -1,  1,  1,  1, -1, -1, -1, -1,  1,  1,
       -1,  1, -1,  1,  1,  1,  1, -1,  1, -1, -1,  1, -1, -1,  1, -1,  1,
        1, -1, -1, -1, -1,  1,  1, -1, -1,  1, -1, -1,  1,  1,  1,  1, -1,
       -1, -1,  1, -1, -1, -1, -1,  1,  1, -1, -1,  1, -1, -1,  1, -1, -1,
       -1,  1, -1, -1, -1,  1, -1,  1, -1,  1,  1,  1,  1,  1, -1, -1, -1,
       -1, -1, -1,  1, -1, -1, -1,  1,  1,  1,  1, -1, -1,  1, -1, -1,  1,
        1,  1, -1,  1,  1, -1,  1, -1, -1,  1,  1,  1, -1,  1, -1, -1, -1,
        1,  1, -1, -1,  1, -1,  1, -1,  1,  1,  1, -1,  1, -1,  1, -1,  1,
       -1,  1, -1, -1,  1,  1, -1, -1, -1, -1, -1,  1, -1, -1, -1,  1, -1,
        1,  1, -1,  1, -1,  1,  1,  1,  1,  1, -1, -1,  1,  1, -1,  1, -1,
       -1,  1,  1, -1, -1, -1,  1,  1, -1, -1, -1,  1, -1, -1, -1, -1, -1,
       -1, -1,  1,  1,  1,  1,  1, -1, -1,  1, -1, -1, -1,  1, -1, -1,  1,
       -1, -1, -1,  1, -1, -1, -1, -1,  1, -1,  1, -1, -1,  1, -1, -1, -1,
       -1,  1, -1,  1, -1, -1, -1, -1, -1,  1, -1, -1,  1, -1, -1, -1,  1,
        1, -1, -1,  1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1,  1, -1, -1,
       -1, -1,  1,  1, -1, -1, -1,  1,  1,  1,  1,  1,  1, -1, -1,  1, -1,
        1, -1,  1,  1,  1,  1, -1, -1, -1,  1,  1,  1, -1,  1,  1,  1,  1,
        1,  1, -1, -1, -1, -1,  1,  1, -1, -1,  1,  1, -1, -1, -1, -1, -1,
       -1, -1,  1, -1,  1, -1, -1, -1, -1, -1, -1,  1,  1, -1, -1,  1,  1,
        1, -1, -1, -1,  1,  1,  1,  1, -1, -1,  1,  1,  1, -1, -1, -1, -1,
        1,  1, -1,  1, -1,  1, -1,  1,  1,  1,  1,  1,  1,  1, -1, -1, -1,
       -1,  1, -1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1, -1, -1,
        1, -1,  1, -1, -1, -1, -1, -1, -1,  1,  1, -1, -1, -1,  1,  1,  1,
       -1,  1,  1,  1, -1, -1, -1,  1,  1,  1,  1, -1, -1, -1,  1, -1, -1,
       -1,  1, -1,  1, -1, -1, -1,  1, -1, -1, -1,  1, -1, -1, -1, -1,  1,
       -1,  1,  1, -1,  1,  1, -1, -1,  1, -1, -1,  1,  1,  1, -1,  1,  1,
       -1,  1,  1,  1, -1,  1,  1, -1, -1,  1, -1, -1, -1,  1, -1,  1,  1,
        1,  1, -1,  1, -1,  1,  1, -1, -1, -1,  1, -1, -1, -1,  1, -1, -1,
        1,  1,  1, -1,  1,  1, -1, -1,  1,  1,  1, -1, -1, -1, -1,  1, -1,
       -1,  1,  1, -1,  1, -1, -1, -1,  1, -1,  1, -1, -1,  1,  1, -1, -1,
       -1, -1,  1,  1, -1, -1,  1, -1,  1, -1,  1,  1,  1,  1,  1, -1,  1,
       -1,  1, -1, -1,  1,  1, -1,  1, -1, -1,  1, -1, -1, -1, -1, -1, -1,
       -1,  1, -1, -1,  1,  1,  1, -1, -1,  1,  1,  1, -1,  1, -1,  1, -1,
       -1, -1, -1, -1, -1, -1,  1,  1, -1,  1, -1, -1,  1, -1,  1,  1,  1,
       -1,  1,  1,  1, -1,  1,  1, -1, -1,  1, -1, -1,  1, -1,  1,  1, -1,
```

```
 1, -1,  1,  1, -1,  1,  1, -1, -1,  1, -1, -1, -1, -1, -1,  1,  1,
-1, -1,  1, -1,  1, -1,  1, -1,  1,  1, -1, -1,  1, -1, -1, -1, -1,
-1,  1, -1, -1, -1,  1,  1, -1, -1, -1, -1, -1,  1,  1,  1, -1, -1,
-1,  1, -1,  1, -1,  1,  1, -1,  1,  1,  1,  1, -1, -1,  1, -1, -1,
-1,  1, -1])
```

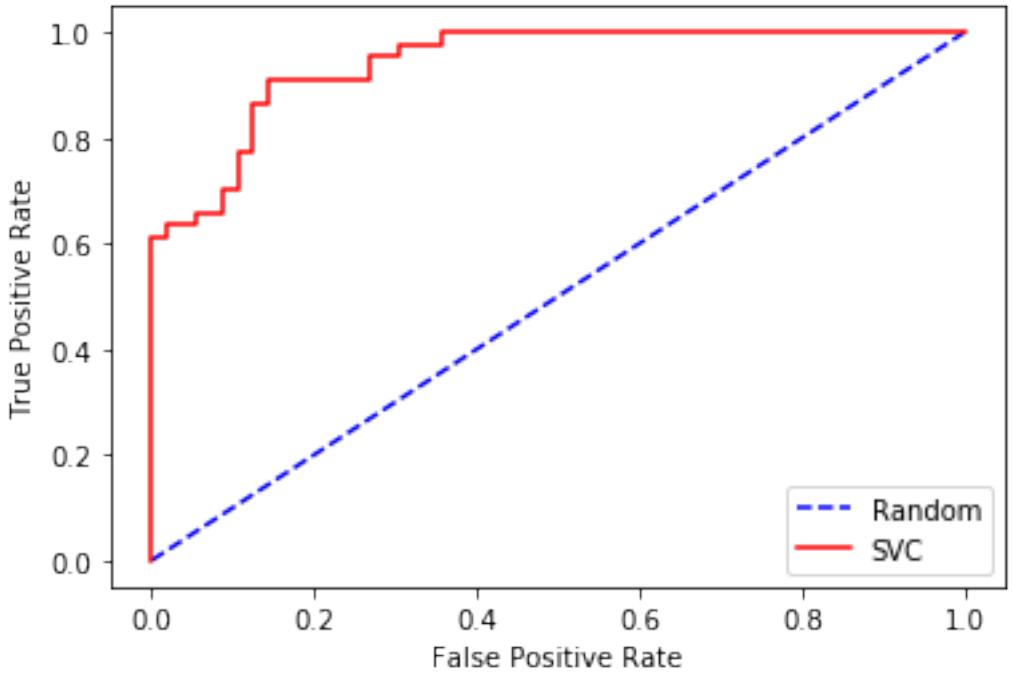
[26]:
```
cm = confusion_matrix(valid_labels, valid_pred)
print("Confusion matrix for SVC : \n")
print(cm)
```

```
Confusion matrix for SVC :

[[50  6]
 [10 34]]
```

[34]:
```
import sklearn.metrics as metrics
# Predict Probabilities of validation data using SVC model
y_pred_prob = sc.predict_proba(valid_data)

# Split the probability data into TPR, FPR and corresponding thresholds
svc_fpr, svc_tpr, _svc_thresholds = roc_curve(valid_labels, y_pred_prob[:,1])

# Probabilities for Random classifier
r_probs = [0 for _ in range(len(valid_labels))]

# Split the probability data into TPR, FPR and corresponding thresholds
r_fpr, r_tpr, _r = roc_curve(valid_labels, r_probs)

# Printing AUC
r_auc = roc_auc_score(valid_labels, r_probs)
lr_auc = roc_auc_score(valid_labels, y_pred_prob[:,1])
print("Area Under Curve (AUC) for RC  : {0}".format(r_auc))
print("Area Under Curve (AUC) for SVC : {0}".format(lr_auc))
```

```
Area Under Curve (AUC) for RC  : 0.5
Area Under Curve (AUC) for SVC : 0.9419642857142856
```

[35]:
```
# Plot the ROC
plt.plot(r_fpr, r_tpr, linestyle='--',color='blue', label='Random')
plt.plot(svc_fpr, svc_tpr,color='red', label='SVC')

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

plt.legend()
plt.show()
```


True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Random
SVC

[ ]: