Lab 5 - SVM

July 23, 2022

1 SVM implementation

```
[105]: # Import necessary modules
       import pandas as pd
       from sklearn.model_selection import train_test_split
       from sklearn.svm import SVC
       from sklearn import svm, datasets
```

```
[106]: # Download the fish dataset
       #dataset_url = "https://raw.githubusercontent.com/harika-bonthu/
        ↪SupportVectorClassifier/main/datasets_229906_491820_Fish.csv"
       #fish = pd.read_csv(dataset_url)
       #fish
       # import some data to play with
       #iris = datasets.load_iris()
       #X = iris.data[:, :2]  # we only take the first two features. We could
                             # avoid this ugly slicing by using a two-dim dataset
       #y = iris.target
       # load the dataset
       df = pd.read_csv("diabetes.csv")
       # See how our dataset is structured u
       df.head()
```

```
[106]:    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
       0            6      148             72             35        0  33.6
       1            1       85             66             29        0  26.6
       2            8      183             64              0        0  23.3
       3            1       89             66             23       94  28.1
       4            0      137             40             35      168  43.1

          DiabetesPedigreeFunction  Age  Outcome
       0                     0.627   50        1
       1                     0.351   31        0
       2                     0.672   32        1
       3                     0.167   21        0
       4                     2.288   33        1
```

[107]:
```
# Adding x and y variables
# The x variable will hold all the input columns, while the y variable will␣
 ↪hold the output column.
# In our case, our output column is the Output column. The remaining columns␣
 ↪will be used as model inputs.
X = df.drop("Outcome",axis="columns")
y = df.Outcome
```

[108]:
```
# Scaling data
#from sklearn.preprocessing import MinMaxScaler
#scaler = MinMaxScaler()
#X = scaler.fit_transform(X)

# Dataset scaling - Dataset scaling is transforming a dataset to fit within a␣
 ↪specific range.
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[109]:
```
# Define the Feature and the Target variables
#X = fish.drop(['Species'], axis = 'columns')
#y = fish.Species
```

[110]:
```
# Split the data into train, test sets using train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.
 ↪2,random_state = 0)
```

[111]:
```
print("shape of Training data X_train :"+str(X_train.shape))
print("shape of Test Data X_test :"+str(X_test.shape))
print("shape of Training Labels y_train :"+str(y_train.shape))
print("shape of Test Labels y_test :"+str(y_test.shape))
```

```
shape of Training data X_train :(614, 8)
shape of Test Data X_test :(154, 8)
shape of Training Labels y_train :(614,)
shape of Test Labels y_test :(154,)
```

[112]:
```
#Instantiate Linear SVC object

model = SVC(kernel = 'linear', C = 1)
```

[113]:
```
# Train the linear SVC classifier using the training data

model.fit(X_train, y_train)
```

[113]:
```
SVC(C=1, kernel='linear')
```

[114]:

```
#Make predictions
svm_pred = model.predict(X_test)
```

[115]:

```
# Check the accuracy of the model using the scoring method
accuracy = model.score(X_test, y_test)
accuracy
```

[115]:

```
0.8181818181818182
```

[ ]:

```
for this_C in [1,3,5,10,40,60,80,100]:
    clf = SVC(kernel='linear',C=this_C).fit(X_train,y_train)
    scoretrain = clf.score(X_train,y_train)
    scoretest  = clf.score(X_test,y_test)
    print("Linear SVM value of C:{}, training score :{:2f} , Test Score: {:2f}␣
  ↪\n".format(this_C,scoretrain,scoretest))
```

```
Linear SVM value of C:1, training score :0.765472 , Test Score: 0.818182

Linear SVM value of C:3, training score :0.768730 , Test Score: 0.818182

Linear SVM value of C:5, training score :0.765472 , Test Score: 0.818182

Linear SVM value of C:10, training score :0.763844 , Test Score: 0.818182

Linear SVM value of C:40, training score :0.767101 , Test Score: 0.792208

Linear SVM value of C:60, training score :0.771987 , Test Score: 0.785714
```

[ ]:

```
from sklearn.model_selection import cross_val_score,StratifiedKFold,LeaveOneOut
clf1 = SVC(kernel='linear',C=20).fit(X_train,y_train)
scores = cross_val_score(clf1,X_train,y_train,cv=5)
strat_scores =␣
  ↪cross_val_score(clf1,X_train,y_train,cv=StratifiedKFold(5,random_state=10,shuffle=True))
#Loo = LeaveOneOut()
#Loo_scores = cross_val_score(clf1,X_train,Y_train,cv=Loo)
print("The Cross Validation Score :"+str(scores))
print("The Average Cross Validation Score :"+str(scores.mean()))
print("The Stratified Cross Validation Score :"+str(strat_scores))
print("The Average Stratified Cross Validation Score :"+str(strat_scores.
  ↪mean()))
#print("The LeaveOneOut Cross Validation Score :"+str(Loo_scores))
#print("The Average LeaveOneOut Cross Validation Score :"+str(Loo_scores.
  ↪mean()))
```

[ ]:

```
# SMV with RBF KERNAL AND ONLY C PARAMETER
```

```
for this_C in [1,5,10,25,50,100]:
    clf3 = SVC(kernel='rbf',C=this_C).fit(X_train,y_train)
    clf3train = clf3.score(X_train,y_train)
    clf3test  = clf3.score(X_test,y_test)
    print("SVM for Non Linear \n C:{} Training Score : {:2f} Test Score : {:
 ↪2f}\n".format(this_C,clf3train,clf3test))
```

[ ]:

```
# SVM WITH RBF KERNAL, C AND GAMMA HYPERPARAMTER
for this_gamma in [.1,.5,.10,.25,.50,1]:
    for this_C in [1,5,7,10,15,25,50]:
        clf3 = SVC(kernel='rbf',C=this_C,gamma=this_gamma).fit(X_train,y_train)
        clf3train = clf3.score(X_train,y_train)
        clf3test  = clf3.score(X_test,y_test)
        print("SVM for Non Linear \n Gamma: {} C:{} Training Score : {:2f} Test␣
 ↪Score : {:2f}\n".format(this_gamma,this_C,clf3train,clf3test))
```

[ ]:

```
# Run SVM with sigmoid kernel and C=100.0
from sklearn.metrics import accuracy_score
# instantiate classifier with sigmoid kernel and C=100.0
sigmoid_svc100=SVC(kernel='sigmoid', C=100.0)


# fit classifier to training set
sigmoid_svc100.fit(X_train,y_train)


# make predictions on test set
y_pred=sigmoid_svc100.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with sigmoid kernel and C=100.0 : {0:0.4f}'.␣
 ↪format(accuracy_score(y_test, y_pred)))
```

[ ]:

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

[ ]:

```
# plot ROC Curve

from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_pred)

plt.figure(figsize=(6,4))

plt.plot(fpr, tpr, linewidth=2)

plt.plot([0,1], [0,1], 'k--' )

plt.rcParams['font.size'] = 12

plt.title('ROC curve for Predicting a Pulsar Star classifier')

plt.xlabel('False Positive Rate (1 - Specificity)')

plt.ylabel('True Positive Rate (Sensitivity)')

plt.show()
```

[ ]:

```
# compute ROC AUC

from sklearn.metrics import roc_auc_score

ROC_AUC = roc_auc_score(y_test, y_pred)

print('ROC AUC : {:.4f}'.format(ROC_AUC))
```