IE406 Machine Learning

Lab Assignment - 5

Group 14

201901466: Miti Purohit

202001430: Aryan Shah

[2]:
```
import warnings
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, precision_score
from sklearn.metrics import confusion_matrix, classification_report,␣
 ↪ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.datasets import load_iris
import seaborn as sns
from sklearn import svm
import numpy as np
import pandas as pd
import cvxpy as cp
import matplotlib.pyplot as plt
%matplotlib inline
warnings.filterwarnings('ignore')
```

Question 1

Perform SVM on iris dataset. (a) Use sklearn SVM classifier and perform classification on dataset. (b) normalize the data and then perform same experiment on normalized data. (c) use the given SVM kernels and perform svm classification. I. Linear II. poly III. bf IV. Sigmoid V. Precomputed

Answer

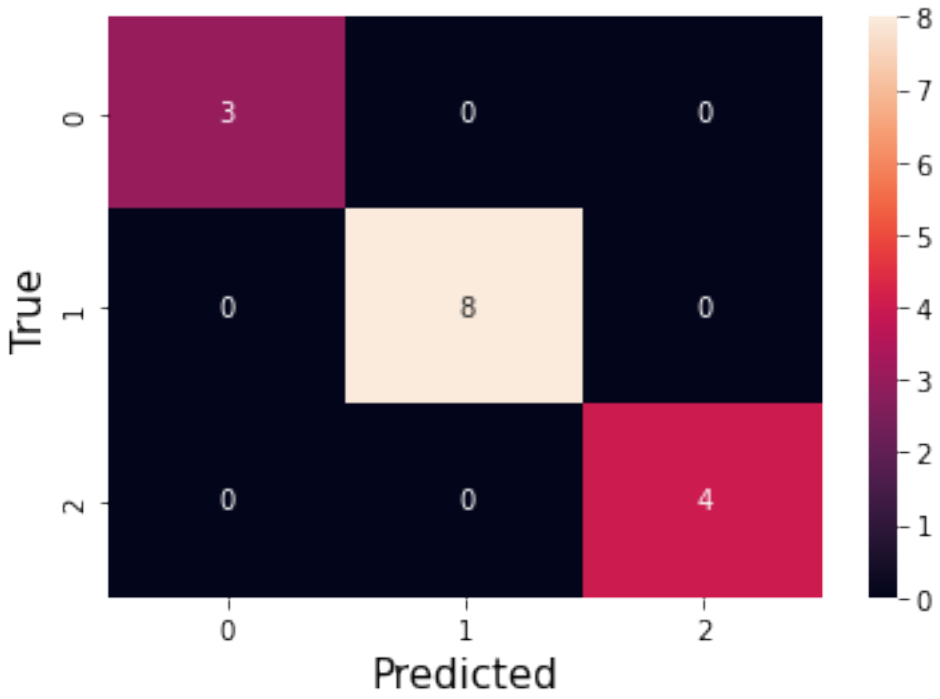
Code

[ ]:
```
target_names = data.target_names
# Split training and testing data
x_train, x_test, Y_train, Y_test = train_test_split(
    x, y, test_size=0.1, random_state=0)
y_train = np.array(Y_train)
y_test = np.array(Y_test)
x_train = np.array(x_train)
x_test = np.array(x_test)
```

```
['setosa' 'versicolor' 'virginica']
```

[ ]:
```
def plot_confusion_matrix(y_test, y_pred):

    confusionMatrix = confusion_matrix(y_test, y_pred)
    sns.heatmap(confusionMatrix, annot=True, fmt='.4g')
    plt.ylabel('True', fontsize=15)
    plt.xlabel('Predicted', fontsize=15)
    plt.show()
    print(classification_report(y_test, y_pred, target_names=target_names))


def svm_func(x_train, x_test, y_train, y_test):

    clf = svm.SVC()
    clf.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    plot_confusion_matrix(y_test, y_pred)
```

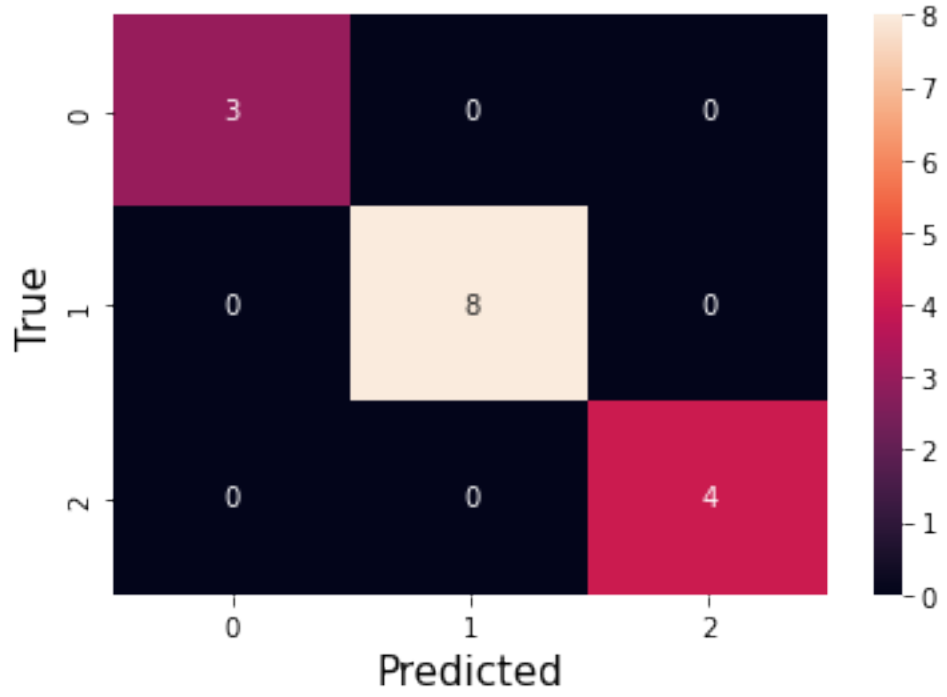
[ ]:
```
svm_func(x_train, x_test, y_train, y_test)
```




```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         3
  versicolor       1.00      1.00      1.00         8
```

virginica	1.00	1.00	1.00	4
accuracy			1.00	15
macro avg	1.00	1.00	1.00	15
weighted avg	1.00	1.00	1.00	15

[ ]:
```
sc = MinMaxScaler()
x_norm_train = sc.fit_transform(x_train)
x_norm_test = sc.fit_transform(x_test)
svm_func(x_norm_train, x_norm_test, y_train, y_test)
```

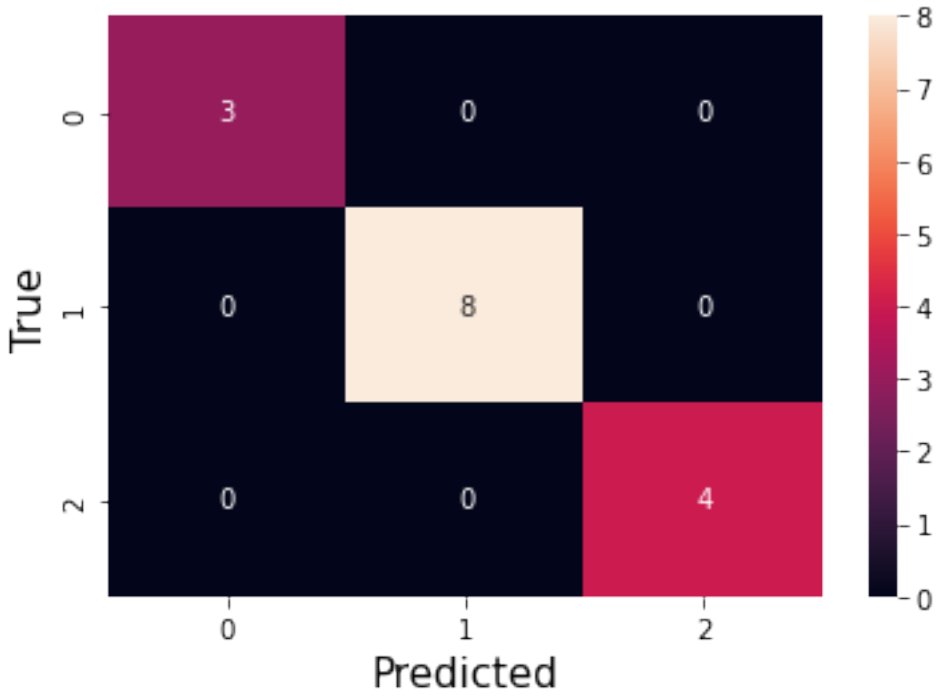


	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	3
versicolor	1.00	1.00	1.00	8
virginica	1.00	1.00	1.00	4
accuracy			1.00	15
macro avg	1.00	1.00	1.00	15
weighted avg	1.00	1.00	1.00	15

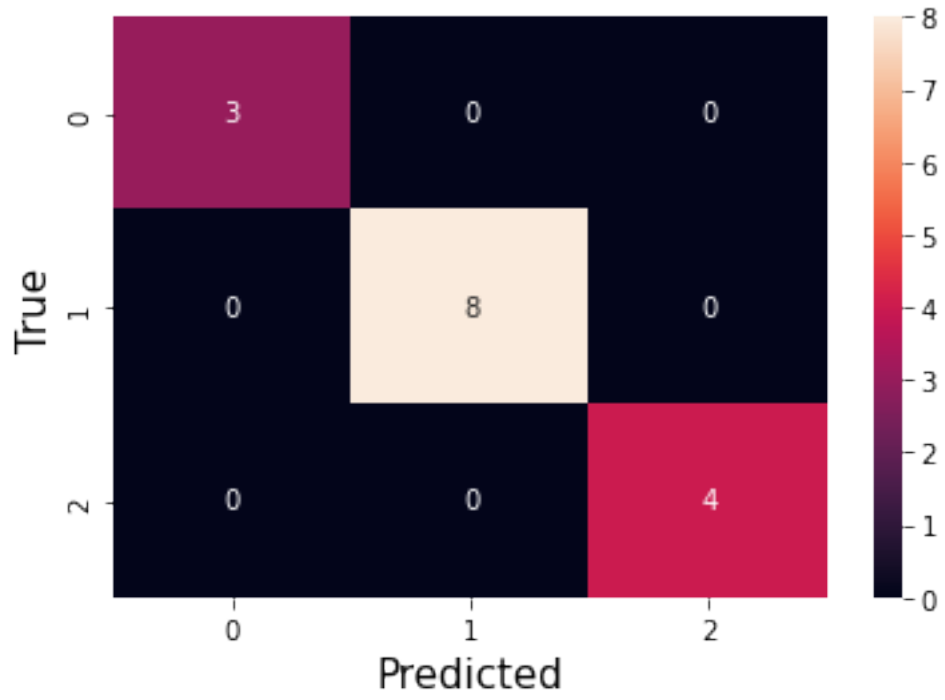
[ ]:
```
def svm_func_kernel(x_train, x_test, y_train, y_test):
    k = ['linear', 'poly', 'rbf', 'sigmoid']

    for i in range(len(k)):
        clf = svm.SVC(kernel=k[i])
        clf.fit(x_train, y_train)
        y_pred = clf.predict(x_test)
        print('kernel ' + k[i])
        plot_confusion_matrix(y_test, y_pred)
```

[ ]:
```
svm_func_kernel(x_train, x_test, y_train, y_test)
```

```
kernel linear
```

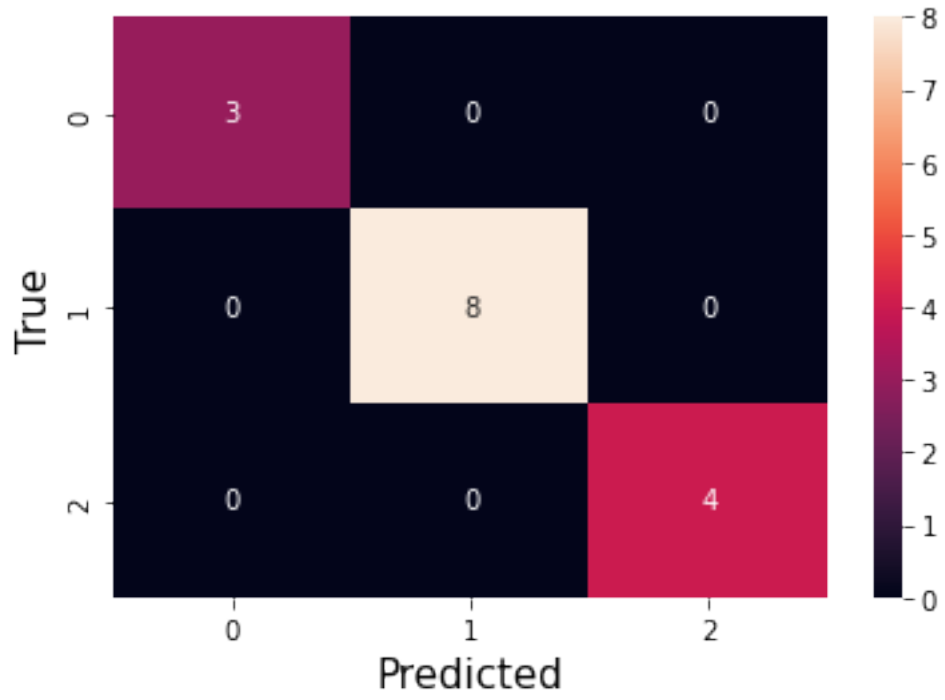



```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         3
  versicolor       1.00      1.00      1.00         8
   virginica       1.00      1.00      1.00         4

    accuracy                           1.00        15
   macro avg       1.00      1.00      1.00        15
weighted avg       1.00      1.00      1.00        15

kernel poly
```



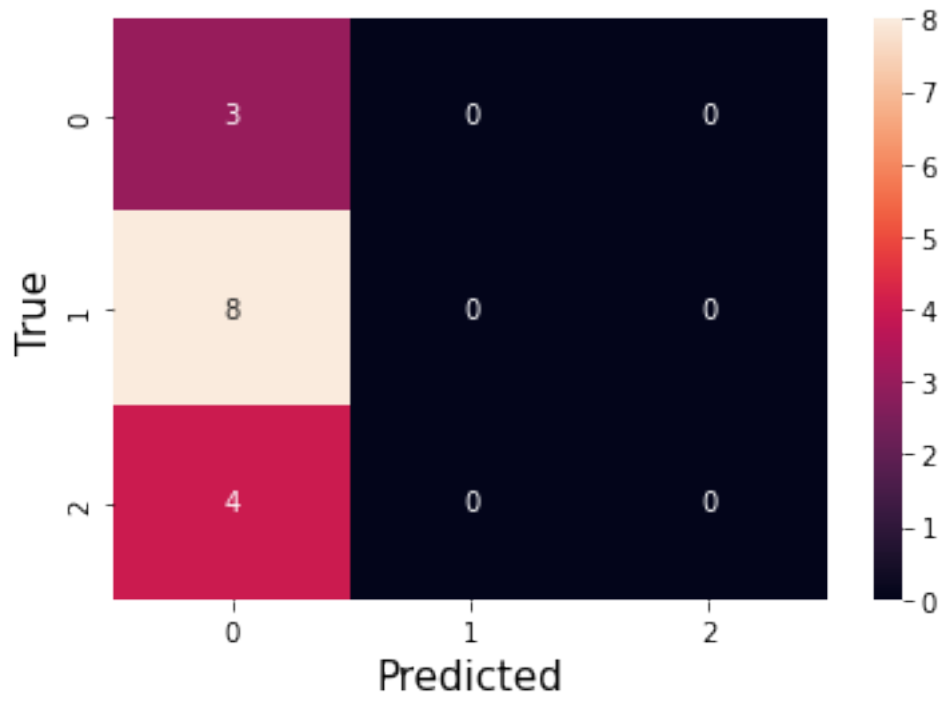
	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	3
versicolor	1.00	1.00	1.00	8
virginica	1.00	1.00	1.00	4
accuracy			1.00	15
macro avg	1.00	1.00	1.00	15
weighted avg	1.00	1.00	1.00	15

kernel rbf



	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	3
versicolor	1.00	1.00	1.00	8
virginica	1.00	1.00	1.00	4
accuracy			1.00	15
macro avg	1.00	1.00	1.00	15
weighted avg	1.00	1.00	1.00	15

kernel sigmoid



	precision	recall	f1-score	support
setosa	0.20	1.00	0.33	3
versicolor	0.00	0.00	0.00	8
virginica	0.00	0.00	0.00	4
accuracy			0.20	15
macro avg	0.07	0.33	0.11	15
weighted avg	0.04	0.20	0.07	15

Result

The resulting graphs are shown above.

Question 2

Perform SVM on https://drive.google.com/file/d/13nw-uRXPY8XIZQxKRNZ3yYlho-CYm_Qt (a) Use sklearn SVM classifier and perform classification on dataset. (b) normalize the data and then perform same experiment on normalized data (c) use the given SVM kernels and perform svm classification. VI. Linear VII. poly VIII. bf IX. Sigmoid X. precomputed

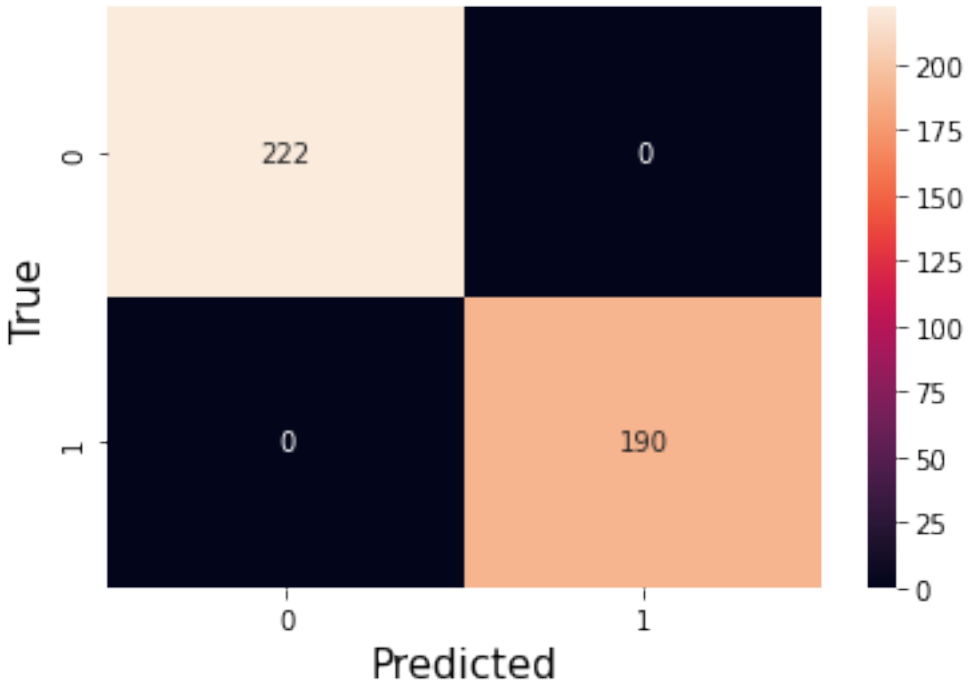
Answer

Code

[ ]:
```
data = pd.read_csv('bill_authentication.csv')

X = data.iloc[:,:-1]
Y = data.iloc[:,-1]

target_names = ('0','1')

from sklearn.model_selection import train_test_split
X_train , X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.
 ↪3,random_state=5)
```

```
('0', '1')
```

[ ]:
```
svm_func(X_train, X_test, Y_train, Y_test)
```

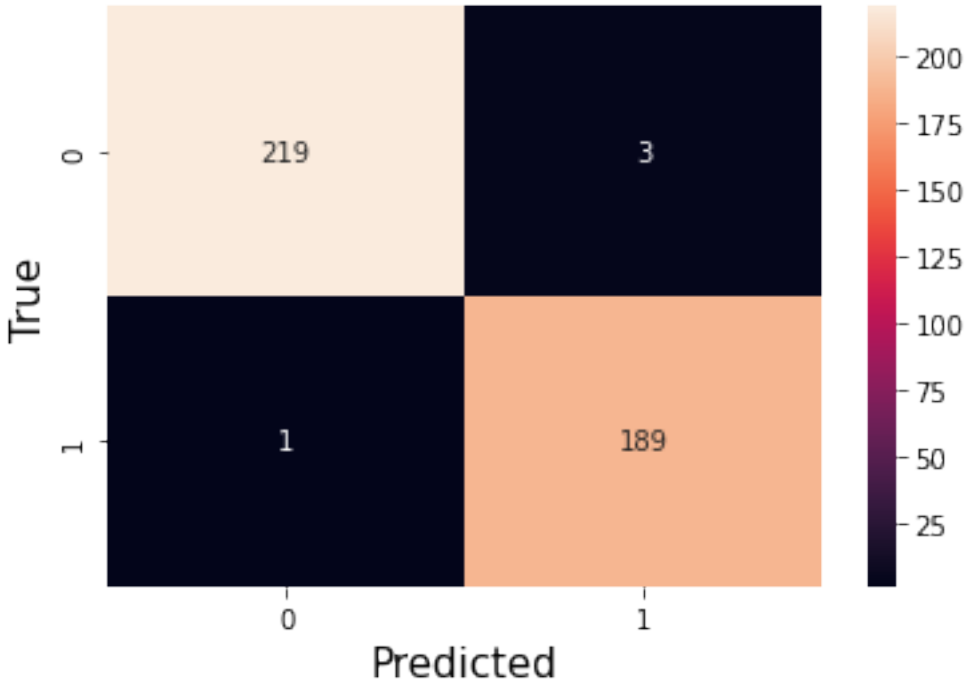


	precision	recall	f1-score	support
0	1.00	1.00	1.00	222
1	1.00	1.00	1.00	190
accuracy			1.00	412
macro avg	1.00	1.00	1.00	412

```
weighted avg       1.00      1.00      1.00       412
```

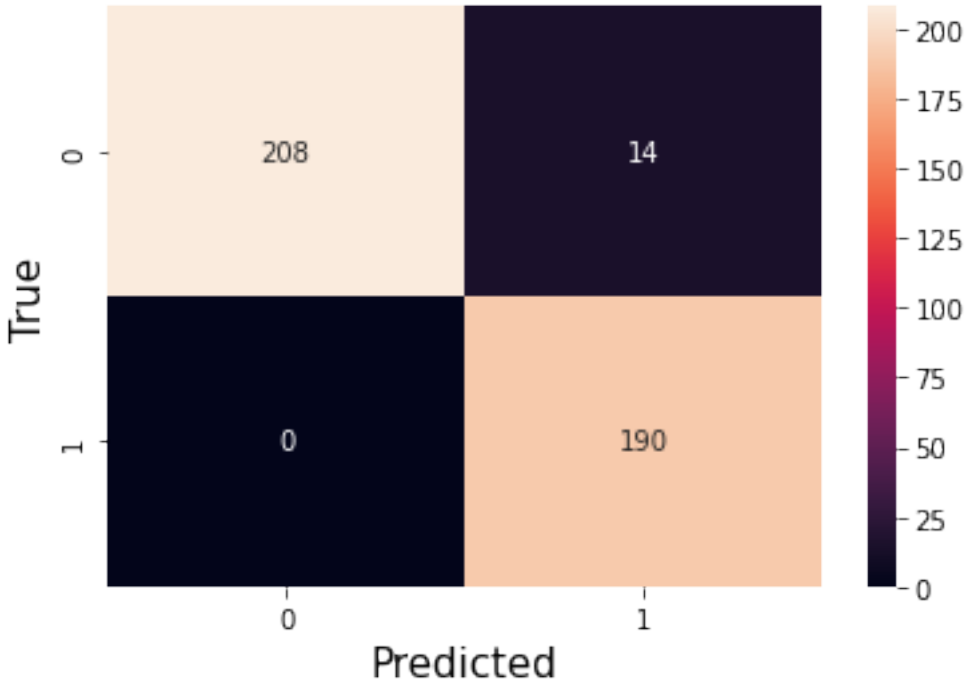
[ ]:
```
svm_func_kernel(X_train, X_test, Y_train, Y_test)
```

```
kernel linear
```



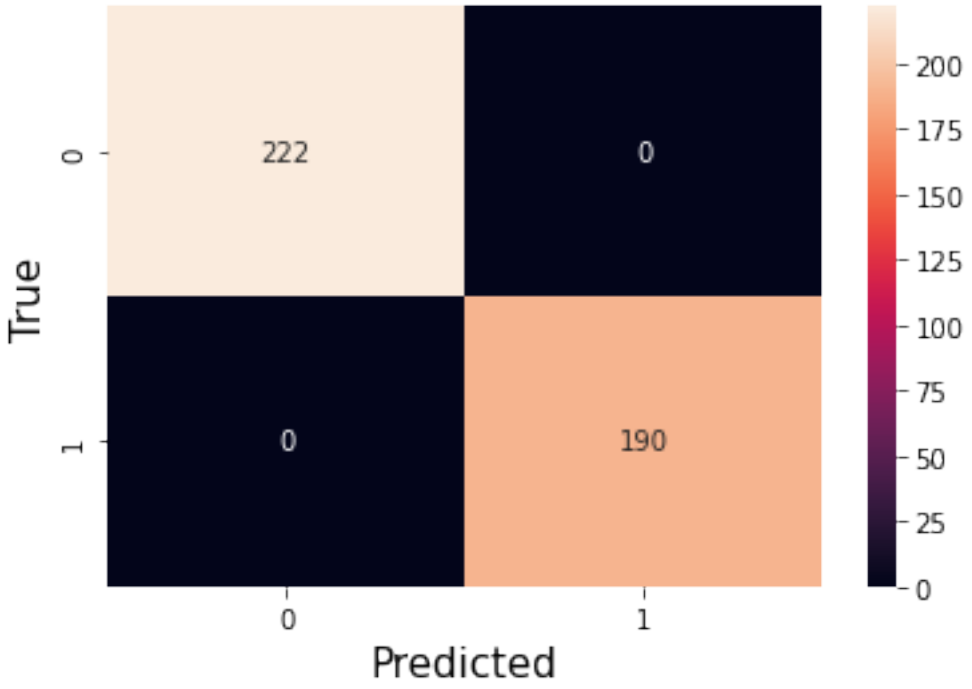
	precision	recall	f1-score	support
0	1.00	0.99	0.99	222
1	0.98	0.99	0.99	190
accuracy			0.99	412
macro avg	0.99	0.99	0.99	412
weighted avg	0.99	0.99	0.99	412

```
kernel poly
```



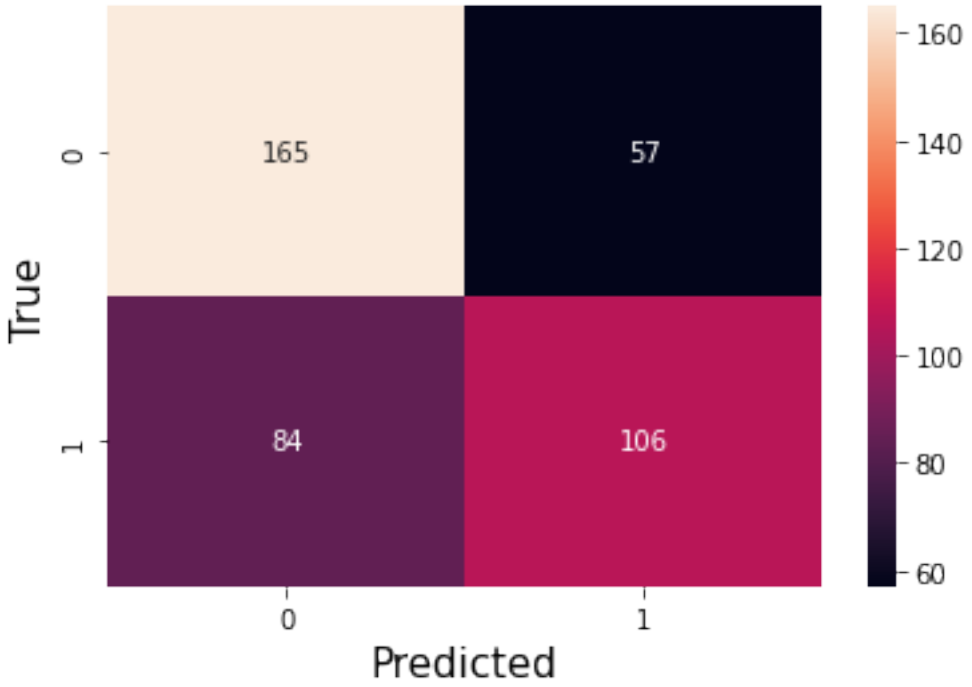
	precision	recall	f1-score	support
0	1.00	0.94	0.97	222
1	0.93	1.00	0.96	190
accuracy			0.97	412
macro avg	0.97	0.97	0.97	412
weighted avg	0.97	0.97	0.97	412

kernel rbf



	precision	recall	f1-score	support
0	1.00	1.00	1.00	222
1	1.00	1.00	1.00	190
accuracy			1.00	412
macro avg	1.00	1.00	1.00	412
weighted avg	1.00	1.00	1.00	412

kernel sigmoid



	precision	recall	f1-score	support
0	0.66	0.74	0.70	222
1	0.65	0.56	0.60	190
accuracy			0.66	412
macro avg	0.66	0.65	0.65	412
weighted avg	0.66	0.66	0.65	412

Result

The resulting graphs are shown above.

Question 3

You may find this tutorial on using CVXPY for Ridge regression very useful as a fully worked out example for the problems in this assignment. A. Implement a function for hard margin SVM in primal form using cvxpy. For keeping this task simple assume w is two dimensional, i.e. (x)= SIGN(w1 x1 + w2 x2 + b) where x and w are both two dimensional vectors. B. Show the usage of your implementation on the IRIS dataset. We will only be making use of sepal-length and petal-width as the two features. We have only two classes - Setosa and Not-Setosa. This problem is linearly separable. C. Plot the decision boundary (separating hyperplane) in dark black and the margins in dotted lines. Encircle the support vector points. D. Plot the decision boundary in dark black and the margins in dotted lines. This time use SKlearn's SVM with a linear kernel. Encircle

the support vector points. Do you get the same answer as when you use your own SVM? E. If you throw away all the points except the support vectors does your decision boundary remain the same? Why?

Answer

code

[3]:
```
plt.style.use('ggplot')
```

Part A

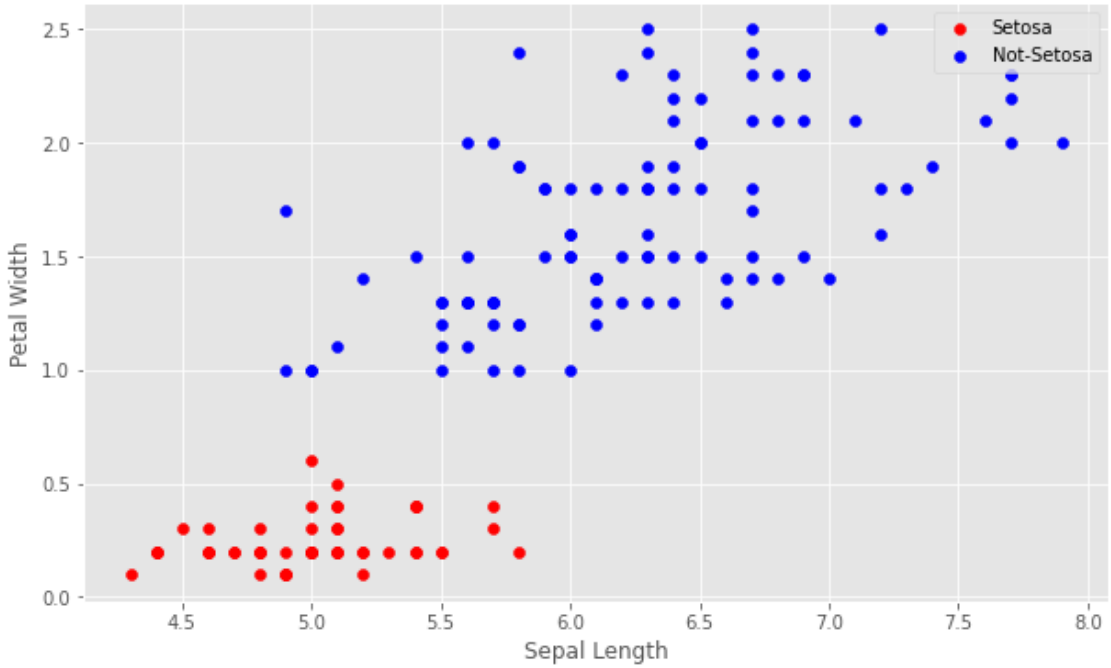
[6]:
```
df = pd.read_csv('/Iris.csv')
df.head()
```

[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Part B

[7]:
```
iris = df.drop(['SepalWidthCm', 'PetalLengthCm'], axis=1)
iris['Species'] = iris['Species'].apply(lambda x: 1 if x == 'Iris-setosa' else␣
 ↪-1)

X = np.array(iris.drop(['Id', 'Species'], axis=1))
Y = np.array(iris['Species'])

plt.figure(figsize=(10,6))
c1 = plt.scatter(X[Y==1, 0], X[Y==1, 1], c = 'r')
c2 = plt.scatter(X[Y==-1, 0], X[Y==-1, 1], c = 'b')
plt.legend((c1,c2),('Setosa','Not-Setosa'))
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')
plt.show()
```

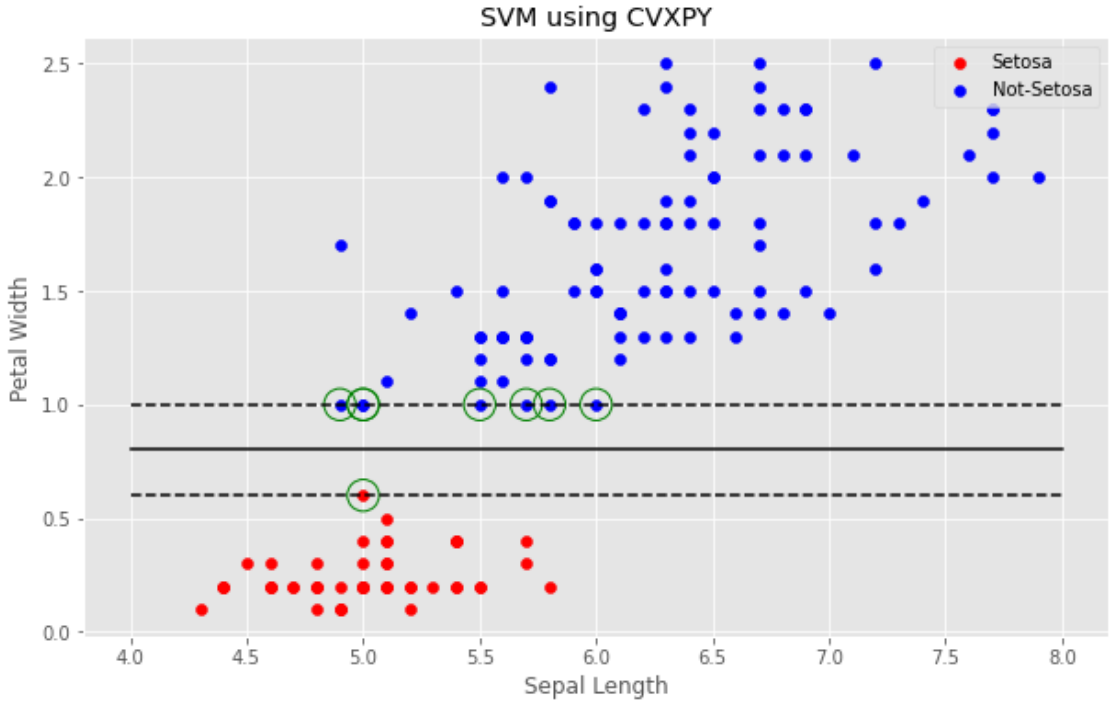


Part C

[8]:
```
theta = cp.Variable(2)
theta0 = cp.Variable(1)

obj_fun = cp.Minimize(0.5*cp.square(cp.norm(theta)))

constraints = []
for i in range(150):
  constraints.append(1 - Y[i]*(theta@X[i] + theta0)<=0)

prob = cp.Problem(obj_fun, constraints)
prob.solve()

#Optimal values of theta and theta0
Theta0 = theta0.value
Theta1 = theta.value[0]
Theta2 = theta.value[1]

print('Optimal value of \u03F4 = ', theta.value)
print('Optimal value of \u03F40 = ', theta0.value)


plt.figure(figsize=(10,6))
c1 = plt.scatter(X[Y==1, 0], X[Y==1, 1], c = 'r')
```

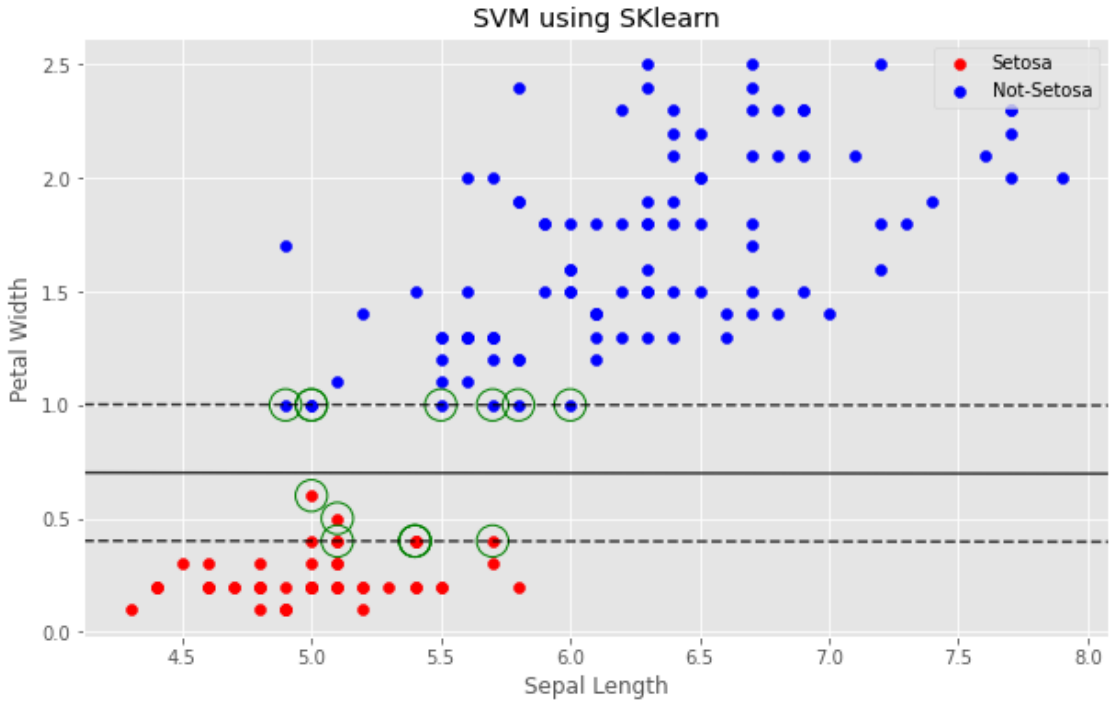
```
c2 = plt.scatter(X[Y==-1, 0], X[Y==-1, 1], c = 'b')
plt.legend((c1,c2), ('Setosa','Not-Setosa'))
plt.title('SVM using CVXPY')
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')

x_axis = np.linspace(4, 8, 1000)
hyperplane = -Theta0 - (Theta1*x_axis)

plt.plot(x_axis, hyperplane/Theta2,'k')

plt.plot(x_axis, (1 + hyperplane)/Theta2,'k--')
plt.plot(x_axis, (-1 + hyperplane)/Theta2,'k--')

for i in range(150):
  temp = Y[i]*((X[i]@theta.value) + theta0.value)
  if(np.round(temp, 2)==1):
    plt.scatter(X[i][0],X[i][1], facecolors='none', s = 300, linewidth=1, marker
 ↪= 'o',edgecolor = 'g')

plt.show()
```

```
Optimal value of  =  [-2.46457378e-10 -5.00000000e+00]
Optimal value of 0 =  [4.]
```



Part D

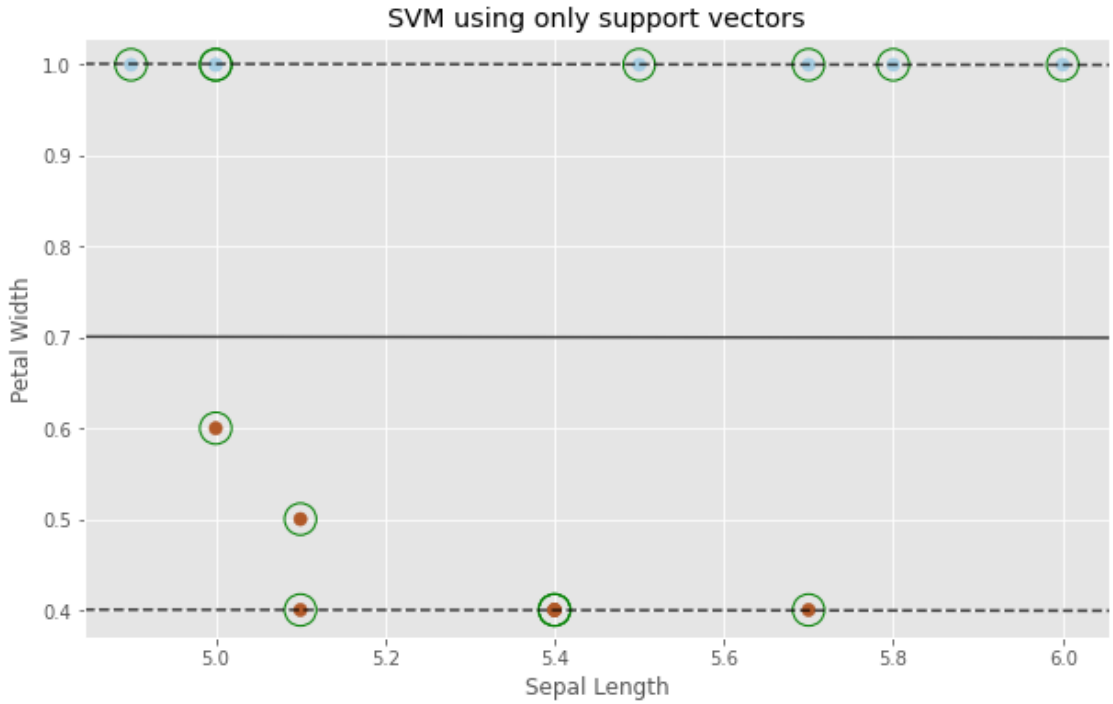
[10]:
```
from sklearn.svm import SVC
#Making Setosa = 1 and Non-Setosa = 0
Y = np.where(Y==1, 1, 0)
#Training SVM
SVM = SVC(kernel='linear')
SVM.fit(X, Y)
#Data visualisation
plt.figure(figsize=(10,6))
c1 = plt.scatter(X[Y==1, 0], X[Y==1, 1], c = 'r')
c2 = plt.scatter(X[Y==0, 0], X[Y==0, 1], c = 'b')
plt.legend((c1,c2), ('Setosa','Not-Setosa'))
plt.title('SVM using SKlearn')
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()
x = np.linspace(xlim[0], xlim[1], 30)
y = np.linspace(ylim[0], ylim[1], 30)
y1, x1 = np.meshgrid(y, x)
xy = np.vstack([x1.ravel(), y1.ravel()]).T
P = SVM.decision_function(xy).reshape(x1.shape)
#Plotting decision boundary and margins
ax.contour(x1, y1, P, colors='k',levels=[-1, 0, 1], alpha=0.75,linestyles=['--',␣
 ↪'-', '--'])
#Plotting support vectors
ax.scatter(SVM.support_vectors_[:, 0], SVM.support_vectors_[:, 1],␣
 ↪s=300,linewidth=1, facecolors='none', edgecolors='g')
ax.set_xlim(xlim)
ax.set_ylim(ylim)
plt.show()
```



Part E

[12]:
```
#Getting support vectors
X_sv = SVM.support_vectors_
Y_sv = []
for i in SVM.support_:
  Y_sv.append(Y[i])
SVM_sv = SVC(kernel='linear')
SVM_sv.fit(X_sv, Y_sv)
#Plotting support vectors
plt.figure(figsize=(10,6))
plt.scatter(X_sv[:, 0], X_sv[:, 1], c=Y_sv, s=50, cmap=plt.cm.Paired)
plt.title('SVM using only support vectors')
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()
xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
P_sv = SVM_sv.decision_function(xy).reshape(XX.shape)
#Plotting decision boundary and margins
```

```
ax.contour(XX, YY, P_sv, colors='k', levels=[-1, 0, 1], alpha=0.
 ↪75,linestyles=['--', '-', '--'])
ax.scatter(SVM_sv.support_vectors_[:, 0], SVM_sv.support_vectors_[:, 1],s=300,␣
 ↪linewidth=1, facecolors='none', edgecolors='g')
plt.show()
```



Result

The resulting graphs are shown above.

Observation/ Justification

E. If you throw away all the points except the support vectors does your decision boundary remain the same? Why?

Upon removing all points other than support vector, the decision boundary does not change since the support vectors are the data points closest to the decision surface. Therefore, the desicion boundary wont change if the support vectors remain the same.