15Z332 Ex7 - NaiveBayes and SVM

October 17, 2018

1 Exercise 7

1.1 NaiveBayes and SVM Classifiers

Perform classification using NaiveBayes and SVM on Iris dataset. Display the confusion matrix for the various classifiers. Analyze the performance of SVM for different kernel functions. Represent the results of the classification in terms of precision and recall using graphs.

1.1.1 Step 1: Import Iris dataset

In [81]:
```
import pandas as pd
df = pd.read_csv('./Iris.csv')
df = df.drop('Id',axis=1)
```

1.1.2 Step 2: Convert categorical data into numerical data.

Since the target attribute 'Species' is categorical, we convert it into numerical data using LabelEncoder

In [82]:
```
print(df.Species.dtype)
```

```
object
```

In [83]:
```
from sklearn import preprocessing

if df.Species.dtype == 'object':
    lbl = preprocessing.LabelEncoder()
    lbl.fit(list(df.Species.values))
    df.Species = lbl.transform(list(df.Species.values))

print(df.Species.dtype)
```

```
int64
```

1.1.3 Step 3: Split the dataset into train and test data

In [84]:
```
from sklearn.model_selection import train_test_split

Y = df.Species.values
X = df.drop(['Species'],axis=1).values

x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size= 0.20, random_state
```

1.1.4 Step 4: Fit the data into NaiveBayes classifier

In [85]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
gnb = GaussianNB()
NaiveBayesModel = gnb.fit(x_train, y_train)
y_pred = NaiveBayesModel.predict(x_test)
accuracy_score(y_test, y_pred)
```

Out[85]: 0.9

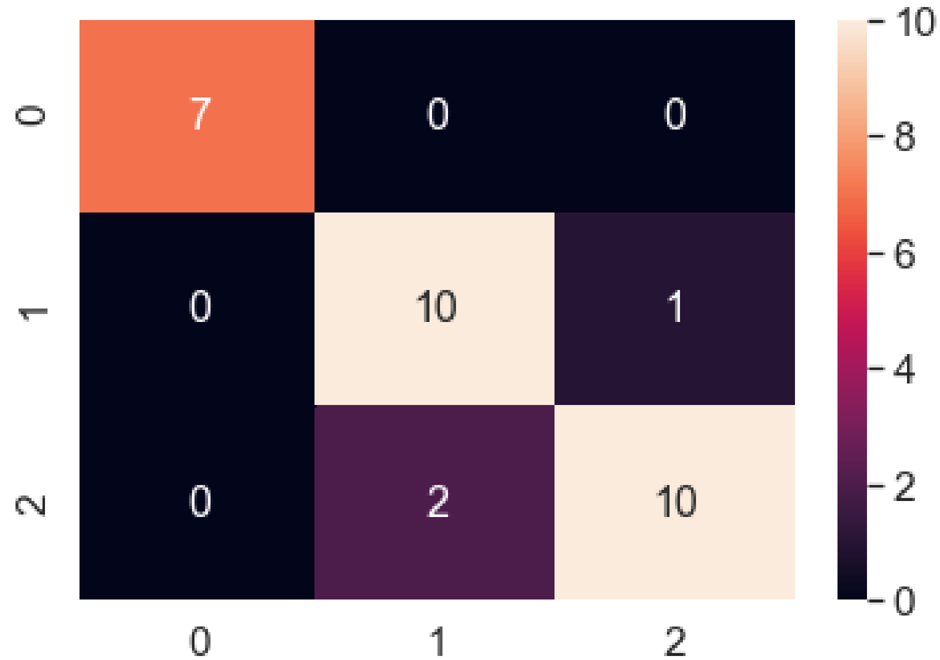
Draw the confusion matrix

In [86]:
```
from sklearn.metrics import confusion_matrix
con_mat = confusion_matrix(y_test,y_pred)
print(con_mat)
```

```
[[ 7  0  0]
 [ 0 10  1]
 [ 0  2 10]]
```

In [87]:
```
import seaborn as sn

sn.set(font_scale=1.4)
sn.heatmap(pd.DataFrame(con_mat), annot=True,annot_kws={"size": 16})
```

Out[87]: <matplotlib.axes._subplots.AxesSubplot at 0x11a8107f0>



1.1.5 Step 5: Fit the data into SVM classifier

```
In [88]: from sklearn import svm

         from sklearn.metrics import average_precision_score

         x_train, x_test, y_train, y_test = train_test_split(X[Y<2],Y[Y<2], test_size= 0.95)

         classifier = svm.LinearSVC(random_state=27)
         classifier.fit(x_train, y_train)
         y_score = classifier.decision_function(x_test)

         average_precision = average_precision_score(y_test, y_score)
         print('Average precision-recall score: {0:0.2f}'.format(average_precision))
```

```
Average precision-recall score: 1.00
```

Since the precision-recall score is 100%, we will try adding noise data and build the SVM model.

```
In [89]: import numpy as np
         # Add noisy features
         random_state = np.random.RandomState(0)
         n_samples, n_features = X.shape
```

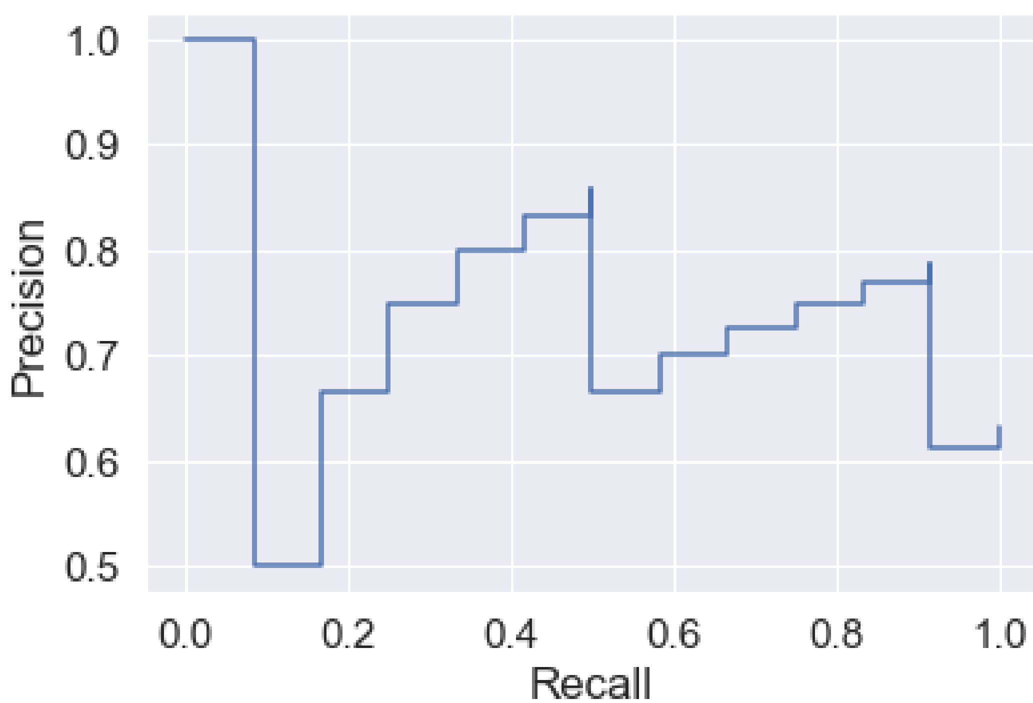
```
X = np.c_[X, random_state.randn(n_samples, 200 * n_features)]

x_train, x_test, y_train, y_test = train_test_split(X[Y<2],Y[Y<2], test_size= 0.20)
```

1) Using Linear kernel

```
In [90]: from sklearn.metrics import precision_recall_curve
         import matplotlib.pyplot as plt

         classifier = svm.SVC(kernel='linear', random_state=27)
         classifier.fit(x_train, y_train)
         y_score = classifier.decision_function(x_test)

         print('Average precision-recall score:',average_precision_score(y_test, y_score))
         precision, recall, _ = precision_recall_curve(y_test, y_score)

         plt.step(recall, precision, color='b')
         plt.xlabel('Recall')
         plt.ylabel('Precision')
```

```
Average precision-recall score: 0.7725782988940885
```

```
Out[90]: Text(0,0.5,'Precision')
```

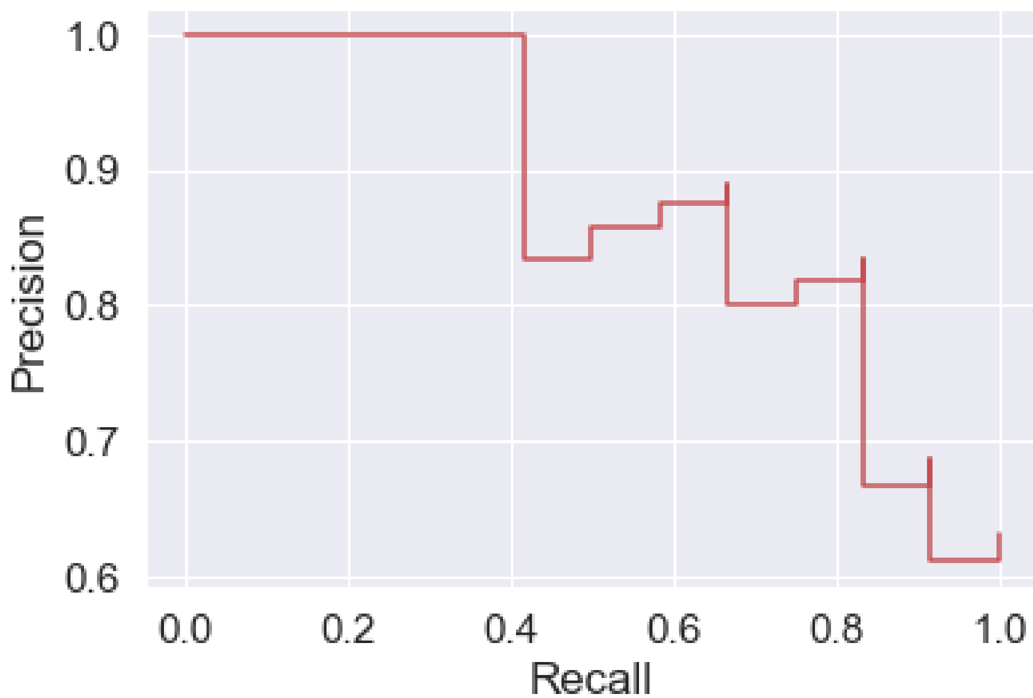


2) Using Polynomial kernel

```
In [91]: classifier = svm.SVC(kernel='poly', random_state=27)
         classifier.fit(x_train, y_train)
         y_score = classifier.decision_function(x_test)

         print('Average precision-recall score:',average_precision_score(y_test, y_score))
         precision, recall, _ = precision_recall_curve(y_test, y_score)

         plt.step(recall, precision, color='r')
         plt.xlabel('Recall')
         plt.ylabel('Precision')
```

```
Average precision-recall score: 0.8826354870762766
```

```
Out[91]: Text(0,0.5,'Precision')
```



3) Using RBF (Radial Basis Function) Kernel

```
In [92]: classifier = svm.SVC(kernel='rbf', random_state=27)
         classifier.fit(x_train, y_train)
         y_score = classifier.decision_function(x_test)
```

```
print('Average precision-recall score:',average_precision_score(y_test, y_score))
precision, recall, _ = precision_recall_curve(y_test, y_score)

plt.step(recall, precision, color='g')
plt.xlabel('Recall')
plt.ylabel('Precision')
```

```
Average precision-recall score: 0.7730399230399231
```

Out[92]: Text(0,0.5,'Precision')