9. Support Vector Machines (SVMs)

November 15, 2022

1 Support Vector Machines (SVMs)

[15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[16]:
```
bankdata = pd.read_csv("bill_authentication.csv")
bankdata.head()
```

[16]:

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

[17]:
```
#To see the rows and columns of the data, execute the following command:
bankdata.shape
```

[17]:
```
(1372, 5)
```

This means that the bank note dataset has 1372 rows and 5 columns

[4]:
```
X = bankdata.drop('Class', axis=1)
y = bankdata['Class']
```

[5]:
```
X
```

[5]:

	Variance	Skewness	Curtosis	Entropy
0	3.62160	8.66610	-2.8073	-0.44699
1	4.54590	8.16740	-2.4586	-1.46210
2	3.86600	-2.63830	1.9242	0.10645
3	3.45660	9.52280	-4.0112	-3.59440
4	0.32924	-4.45520	4.5718	-0.98880
…	…	…	…	…
1367	0.40614	1.34920	-1.4501	-0.55949
1368	-1.38870	-4.87730	6.4774	0.34179

```
1369  -3.75030 -13.45860   17.5932 -2.77710
1370  -3.56370  -8.38270   12.3930 -1.28230
1371  -2.54190  -0.65804    2.6842  1.19520

[1372 rows x 4 columns]
```

[6]:
```
y
```

[6]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1367    1
1368    1
1369    1
1370    1
1371    1
Name: Class, Length: 1372, dtype: int64
```

[7]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

[8]:
```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
```

[8]:
```
SVC(kernel='linear')
```

[9]:
```
y_pred = svclassifier.predict(X_test)
y_pred
```

[9]:
```
array([1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0])
```

[12]:
```
variance=float(input("enter variance: "))
skewness=float(input("enter skewness: "))
curtosis=float(input("enter curtosis: "))
entrophy=float(input("enter entrophy: "))

to_predict_list=[variance,skewness,curtosis,entrophy]
input_data= np.asarray(to_predict_list)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data.reshape(1,-1)

prediction = svclassifier.predict(input_data_reshaped)

if(prediction[0]==1):
    print("The bank currency note is authentic")
else:
    print("The bank currency note is Not authentic")
```

```
enter variance: 3.62160
enter skewness: 8.66610
enter curtosis: -2.8073
enter entrophy: -0.44699
The bank currency note is Not authentic
```

```
/home/samuel-adirala/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450:
UserWarning: X does not have valid feature names, but SVC was fitted with
feature names
  warnings.warn(
```

[13]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[137   1]
 [  2 135]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       138
           1       0.99      0.99      0.99       137

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```

[14]:
```
from sklearn.metrics import accuracy_score
accuracy=accuracy_score(y_test,y_pred)*100
print("accuracy score is",accuracy)
```

```
accuracy score is 98.9090909090909
```

2 Kernel methods in SVM

2.1 Polynomial Kernel

[18]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[19]:
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Assign colum names to the dataset
colnames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',␣
 ↪'Class']

# Read dataset to pandas dataframe
irisdata = pd.read_csv(url, names=colnames)
irisdata.head()
```

[19]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

[20]:
```
X = irisdata.drop('Class', axis=1)
y = irisdata['Class']
```

[21]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

[22]:
```
from sklearn.svm import SVC
svclassifier = SVC(kernel='poly', degree=8)
svclassifier.fit(X_train, y_train)
```

[22]:
```
SVC(degree=8, kernel='poly')
```

[23]:
```
y_pred = svclassifier.predict(X_test)
```

[24]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0  6  1]
```

```
 [ 0  2 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       0.75      0.86      0.80         7
 Iris-virginica       0.91      0.83      0.87        12

       accuracy                           0.90        30
      macro avg       0.89      0.90      0.89        30
   weighted avg       0.91      0.90      0.90        30
```

2.2 Gaussian Kernel

[25]:
```
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
```

[25]:
```
SVC()
```

[26]:
```
y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0  7  0]
 [ 0  1 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       0.88      1.00      0.93         7
 Iris-virginica       1.00      0.92      0.96        12

       accuracy                           0.97        30
      macro avg       0.96      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

2.3 Sigmoid Kernel

[27]:
```
from sklearn.svm import SVC
svclassifier = SVC(kernel='sigmoid')
svclassifier.fit(X_train, y_train)
```

[27]:
```
SVC(kernel='sigmoid')
```

[28]:
```python
y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[ 0 11  0]
 [ 0  7  0]
 [ 0 12  0]]
                 precision    recall  f1-score   support

    Iris-setosa       0.00      0.00      0.00        11
Iris-versicolor       0.23      1.00      0.38         7
 Iris-virginica       0.00      0.00      0.00        12

       accuracy                           0.23        30
      macro avg       0.08      0.33      0.13        30
   weighted avg       0.05      0.23      0.09        30

/home/samuel-adirala/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/samuel-adirala/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/samuel-adirala/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```