Assignment no 2 _ML

October 8, 2024

[36]:
```
#ASSIGNMENT NO 2
#Use K-Nearest Neighbors and Support Vector Machine for classification. Analyze␣
 ↪their performance.
#Dataset link: The emails.csv dataset on the Kaggle https://www.kaggle.com/
 ↪datasets/balaka18/email-spam-classification-dataset-csv
```

[37]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[38]:
```
df=pd.read_csv('/home/pc13/Documents/Email/emails.csv')
```

[51]:
```
df.head()  #it returns the first five rows of the DataFrame df
```

[51]:
```
  Email No.  the  to  ect  and  for  of    a  you  hou  …  connevey  jay  \
0   Email 1    0   0    1    0    0   0    2    0    0  …         0    0
1   Email 2    8  13   24    6    6   2  102    1   27  …         0    0
2   Email 3    0   0    1    0    0   0    8    0    0  …         0    0
3   Email 4    0   5   22    0    5   1   51    2   10  …         0    0
4   Email 5    7   6   17    1    5   2   57    0    9  …         0    0

   valued  lay  infrastructure  military  allowing  ff  dry  Prediction
0       0    0               0         0         0   0    0           0
1       0    0               0         0         0   1    0           0
2       0    0               0         0         0   0    0           0
3       0    0               0         0         0   0    0           0
4       0    0               0         0         0   1    0           0

[5 rows x 3002 columns]
```

[52]:
```
df.info()  #df.info() function in pandas provides a concise summary of a␣
 ↪DataFrame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3002 entries, Email No. to Prediction
```

```
dtypes: int64(3001), object(1)
memory usage: 118.5+ MB
```

[53]:

```
df.isnull().sum() #The df.isnull().sum() function in pandas is used to check␣
 ↪for missing (null) values in a DataFrame.
```

[53]:

```
Email No.     0
the           0
to            0
ect           0
and           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3002, dtype: int64
```

[54]:

```
X = df.iloc[:, 1:-1].values
y = df.iloc[:, -1].values
#X and y are being created from a pandas DataFrame df using the iloc method,␣
 ↪which is used for integer-location based indexing
#X typically represents the feature set (input data) used for training a␣
 ↪machine learning model.
#y usually represents the target variable (output data) that the model aims to␣
 ↪predict.
```

[55]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,␣
 ↪random_state=101)
#the train_test_split function from the sklearn.model_selection module is used␣
 ↪to split the dataset into training and testing sets
```

[56]:

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
#The StandardScaler from the sklearn.preprocessing module is used to␣
 ↪standardize the feature set
```

[57]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
#using the KNeighborsClassifier from the sklearn.neighbors module to create and␣
 ↪train a K-Nearest Neighbors (KNN) classifier.
```

[57]:
```
KNeighborsClassifier()
```

[58]:
```
#KNeighborsClassifier() is a class in the sklearn.neighbors module of the␣
 ↪scikit-learn library, which implements the K-Nearest Neighbors (KNN)␣
 ↪algorithm for classification tasks
```

[59]:
```
y_pred = classifier.predict(X_test)
#In this line of code, y_pred = classifier.predict(X_test), you're using the␣
 ↪trained K-Nearest Neighbors classifier to make predictions on the test set
```

[60]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
#using functions from sklearn.metrics to evaluate the performance of your␣
 ↪K-Nearest Neighbors classifier by generating a confusion matrix.
```

[61]:
```
cm
#The variable cm contains the confusion matrix generated by the␣
 ↪confusion_matrix function.
```

[61]:
```
array([[866, 248],
       [ 16, 422]])
```

[49]:
```
from sklearn.metrics import classification_report
cl_report=classification_report(y_test,y_pred)
print(cl_report)
#Generating a classification report using the classification_report function␣
 ↪from the sklearn.metrics module
```

```
              precision    recall  f1-score   support

           0       0.98      0.78      0.87      1114
           1       0.63      0.96      0.76       438

    accuracy                           0.83      1552
   macro avg       0.81      0.87      0.81      1552
weighted avg       0.88      0.83      0.84      1552
```

[50]:
```
print("Accuracy Score for KNN : ", accuracy_score(y_pred,y_test))
```

```
Accuracy Score for KNN :  0.8298969072164949
```

[62]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
#importing the Support Vector Classifier (SVC) from the sklearn.svm module and␣
 ↪the accuracy_score function from sklearn.metrics
```

[69]:

```
svc = SVC(C=1.0,kernel='rbf',gamma='auto')
svc.fit(X_train,y_train)
y_pred2 = svc.predict(X_test)
#you're creating and training a Support Vector Classifier (SVC) using the␣
 ↪Radial Basis Function (RBF) kernel
```

[64]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
#generating a confusion matrix for the predictions made by the Support Vector␣
 ↪Classifier (SVC
cm = confusion_matrix(y_test, y_pred2)
#Creating the Confusion Matrix
```

[70]:

```
cm
#The variable cm contains the confusion matrix generated from your SVC model's␣
 ↪predictions
```

[70]:

```
array([[1106,    8],
       [  95,  343]])
```

[67]:

```
print("Accuracy Score for SVC : ", accuracy_score(y_pred2,y_test))
```

```
Accuracy Score for SVC :  0.9336340206185567
```

[71]:

```
from sklearn.metrics import classification_report
cl_report=classification_report(y_test,y_pred2)
print(cl_report)
#generating a classification report for the predictions made by your Support␣
 ↪Vector Classifier (SVC)
```

```
              precision    recall  f1-score   support

           0       0.92      0.99      0.96      1114
           1       0.98      0.78      0.87       438

    accuracy                           0.93      1552
   macro avg       0.95      0.89      0.91      1552
weighted avg       0.94      0.93      0.93      1552
```

[ ]:

```

```