Assignment_5

October 9, 2024

1 CLASSIFY THE EMAIL USING THE BINARY CLASSIFICATION METHOD

1.1 Importing Libraries

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```python
df = pd.read_csv('emails.csv')
```

[3]:
```python
df
```

[3]:
```
        Email No.  the  to  ect  and  for  of    a  you  hou  …  connevey  \
0         Email 1    0   0    1    0    0   0    2    0    0  …         0
1         Email 2    8  13   24    6    6   2  102    1   27  …         0
2         Email 3    0   0    1    0    0   0    8    0    0  …         0
3         Email 4    0   5   22    0    5   1   51    2   10  …         0
4         Email 5    7   6   17    1    5   2   57    0    9  …         0
…             …   …  ..   …   …   …  ..   …   …   …   …         …
5167   Email 5168    2   2    2    3    0   0   32    0    0  …         0
5168   Email 5169   35  27   11    2    6   5  151    4    3  …         0
5169   Email 5170    0   0    1    1    0   0   11    0    0  …         0
5170   Email 5171    2   7    1    0    2   1   28    2    0  …         0
5171   Email 5172   22  24    5    1    6   5  148    8    2  …         0

      jay  valued  lay  infrastructure  military  allowing  ff  dry  \
0       0       0    0               0         0         0   0    0
1       0       0    0               0         0         0   1    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         0   1    0
…     …       …    …               …         …         …  ..  …
5167    0       0    0               0         0         0   0    0
5168    0       0    0               0         0         0   1    0
5169    0       0    0               0         0         0   0    0
5170    0       0    0               0         0         0   1    0
5171    0       0    0               0         0         0   0    0
```

```
      Prediction
0              0
1              0
2              0
3              0
4              0
…            …
5167           0
5168           0
5169           1
5170           1
5171           0

[5172 rows x 3002 columns]
```

1.2 Checking Null Values

[4]:
```
df.isnull().sum()
```

[4]:
```
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
```

[5]:
```
df = df.drop(columns=['Email No.'])
```

[6]:
```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

[7]:
```
X
```

[7]:
```
      the  to  ect  and  for  of    a  you  hou  in  …  enhancements  \
0       0   0    1    0    0   0    2    0    0   0  …             0
1       8  13   24    6    6   2  102    1   27  18  …             0
2       0   0    1    0    0   0    8    0    0   4  …             0
3       0   5   22    0    5   1   51    2   10   1  …             0
4       7   6   17    1    5   2   57    0    9   3  …             0
…     …  ..  …  …  …  ..  …  …  …  ..  …            …
```

```
5167     2   2    2    3    0   0   32    0    0   5  …            0
5168    35  27   11    2    6   5  151    4    3  23  …            0
5169     0   0    1    1    0   0   11    0    0   1  …            0
5170     2   7    1    0    2   1   28    2    0   8  …            0
5171    22  24    5    1    6   5  148    8    2  23  …            0

      connevey  jay  valued  lay  infrastructure  military  allowing  ff  dry
0            0    0       0    0               0         0         0   0    0
1            0    0       0    0               0         0         0   1    0
2            0    0       0    0               0         0         0   0    0
3            0    0       0    0               0         0         0   0    0
4            0    0       0    0               0         0         0   1    0
…          …  …      …  …             …        …        …  ..  …
5167         0    0       0    0               0         0         0   0    0
5168         0    0       0    0               0         0         0   1    0
5169         0    0       0    0               0         0         0   0    0
5170         0    0       0    0               0         0         0   1    0
5171         0    0       0    0               0         0         0   0    0

[5172 rows x 3000 columns]
```

[ ]:

[8]:
```
y
```

[8]:
```
0       0
1       0
2       0
3       0
4       0
       ..
5167    0
5168    0
5169    1
5170    1
5171    0
Name: Prediction, Length: 5172, dtype: int64
```

[47]:
```
X_train.shape
y_train.shape
```

[47]:
```
(4137,)
```

1.3 Scaling the data

```
[10]: from sklearn.preprocessing import StandardScaler
      scale = StandardScaler()
```

```
[16]: columns_df = X.columns
```

```
[17]: columns_df
```

```
[17]: Index(['the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou', 'in',
              …
              'enhancements', 'connevey', 'jay', 'valued', 'lay', 'infrastructure',
              'military', 'allowing', 'ff', 'dry'],
             dtype='object', length=3000)
```

```
[19]: X_scaled = scale.fit_transform(X)
```

1.4 Spliting the Dataset

```
[20]: from sklearn.model_selection import train_test_split
```

```
[21]: X_train,X_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.
        ↪2,random_state=42)
```

1.5 Importing Model(KNN)

```
[22]: from sklearn.neighbors import KNeighborsClassifier
```

```
[42]: model = KNeighborsClassifier(n_neighbors=2)
```

```
[43]: model.fit(X_train,y_train)
```

```
[43]: KNeighborsClassifier(n_neighbors=2)
```

```
[44]: y_pred = model.predict(X_test)
```

1.6 Evaluating Model

```
[50]: from sklearn.metrics import␣
        ↪accuracy_score,confusion_matrix,classification_report
```

```
[46]: accuracy_score(y_test,y_pred)
```

```
[46]: 0.9043478260869565
```

```
[49]: confusion_matrix(y_test,y_pred)
```

[49]:
```
array([[685,  54],
       [ 45, 251]])
```

[52]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.93      0.93       739
           1       0.82      0.85      0.84       296

    accuracy                           0.90      1035
   macro avg       0.88      0.89      0.88      1035
weighted avg       0.91      0.90      0.90      1035
```

1.7 Trying Different K values

[56]:
```
for i in range(1,10):

    model = KNeighborsClassifier(n_neighbors=i)
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    print(f"Accuracy for {i} th no. of neighbors is␣
 ↪{accuracy_score(y_test,y_pred)}")
```

```
Accuracy for 1 th no. of neighbors is 0.8966183574879227
Accuracy for 2 th no. of neighbors is 0.9043478260869565
Accuracy for 3 th no. of neighbors is 0.8618357487922705
Accuracy for 4 th no. of neighbors is 0.8705314009661835
Accuracy for 5 th no. of neighbors is 0.8338164251207729
Accuracy for 6 th no. of neighbors is 0.8444444444444444
Accuracy for 7 th no. of neighbors is 0.8077294685990338
Accuracy for 8 th no. of neighbors is 0.81256038647343
Accuracy for 9 th no. of neighbors is 0.7806763285024154
Accuracy for 10 th no. of neighbors is 0.7816425120772947
Accuracy for 11 th no. of neighbors is 0.7574879227053141
Accuracy for 12 th no. of neighbors is 0.7603864734299517
Accuracy for 13 th no. of neighbors is 0.7304347826086957
Accuracy for 14 th no. of neighbors is 0.7323671497584541
Accuracy for 15 th no. of neighbors is 0.7120772946859903
Accuracy for 16 th no. of neighbors is 0.7111111111111111
Accuracy for 17 th no. of neighbors is 0.693719806763285
Accuracy for 18 th no. of neighbors is 0.6975845410628019
Accuracy for 19 th no. of neighbors is 0.6879227053140097
```

1.8 Importing Model (SVM)

```
[72]: from sklearn.svm import SVC
      SVCClf = SVC(kernel = 'sigmoid',C=2)
      SVCClf.fit(X_train, y_train)
```

```
[72]: SVC(C=2, kernel='sigmoid')
```

```
[73]: y_pred = SVCClf.predict(X_test)
```

```
[74]: accuracy_score(y_test,y_pred)
```

```
[74]: 0.9169082125603865
```

1.9 Trying Various Kernels and C-Values

```
[79]: for i in range(1,10):

          SVCClf = SVC(kernel = 'sigmoid',C = i)
          SVCClf.fit(X_train,y_train)
          y_pred = SVCClf.predict(X_test)
          print(f"Accuracy for {i} th no. of c is {accuracy_score(y_test,y_pred)}")
```

```
Accuracy for 1 th no. of c is 0.9217391304347826
Accuracy for 2 th no. of c is 0.9169082125603865
Accuracy for 3 th no. of c is 0.9101449275362319
Accuracy for 4 th no. of c is 0.9082125603864735
Accuracy for 5 th no. of c is 0.9101449275362319
Accuracy for 6 th no. of c is 0.9062801932367149
Accuracy for 7 th no. of c is 0.9033816425120773
Accuracy for 8 th no. of c is 0.9014492753623189
Accuracy for 9 th no. of c is 0.9014492753623189
```

```
[80]: for i in range(1,10):

          SVCClf = SVC(kernel = 'linear',C = i)
          SVCClf.fit(X_train,y_train)
          y_pred = SVCClf.predict(X_test)
          print(f"Accuracy for {i} th no. of c is {accuracy_score(y_test,y_pred)}")
```

```
Accuracy for 1 th no. of c is 0.9449275362318841
Accuracy for 2 th no. of c is 0.9439613526570049
Accuracy for 3 th no. of c is 0.9429951690821256
Accuracy for 4 th no. of c is 0.9381642512077295
Accuracy for 5 th no. of c is 0.9352657004830918
Accuracy for 6 th no. of c is 0.936231884057971
Accuracy for 7 th no. of c is 0.936231884057971
Accuracy for 8 th no. of c is 0.936231884057971
Accuracy for 9 th no. of c is 0.936231884057971
```

[81]:
```
for i in range(1,10):

    SVCClf = SVC(kernel = 'rbf',C = i)
    SVCClf.fit(X_train,y_train)
    y_pred = SVCClf.predict(X_test)
    print(f"Accuracy for {i} th no. of c is {accuracy_score(y_test,y_pred)}")
```

```
Accuracy for 1 th no. of c is 0.9468599033816425
Accuracy for 2 th no. of c is 0.9623188405797102
Accuracy for 3 th no. of c is 0.9642512077294686
Accuracy for 4 th no. of c is 0.966183574879227
Accuracy for 5 th no. of c is 0.9652173913043478
Accuracy for 6 th no. of c is 0.9652173913043478
Accuracy for 7 th no. of c is 0.9652173913043478
Accuracy for 8 th no. of c is 0.9632850241545894
Accuracy for 9 th no. of c is 0.961352657004831
```

[ ]:
```

```