email_spam_filtering

November 7, 2023

[1]:
```
import pandas as pd
import numpy as np
```

[3]:
```
df = pd.read_csv(r"C:\Users\lalit\OneDrive\Desktop\practicals\machine␣
↪learning\practical2\emails.csv")
```

[4]:
```
df.head()
```

[4]:

	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]

[5]:
```
df.info()
```

```
<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
```

[6]:
```
df.shape
```

[6]: (5172, 3002)

[7]:
```
df.dtypes
```

[7]:
```
Email No.     object
the            int64
to             int64
ect            int64
and            int64
               …
military       int64
allowing       int64
ff             int64
dry            int64
Prediction     int64
Length: 3002, dtype: object
```

[8]:
```
#Removing the null values
df.isnull().sum()
```

[8]:
```
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
```

[16]:
```
from sklearn.model_selection import train_test_split

# Define X (Features)
X = df.drop(columns=['Email No.', 'Prediction'])

# Define y (Target)
y = df['Prediction']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[18]:
```
# Encode the target labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
```

[21]:
```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, roc_auc_score

# Create a K-NN model
knn_model = KNeighborsClassifier(n_neighbors=5)  # You can adjust the number of␣
 ↪neighbors
```

[22]:
```
# Train the K-NN model on the training data
knn_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred_knn = knn_model.predict(X_test)
```

```
D:\Anaconda\lib\site-packages\sklearn\neighbors\_classification.py:228:
FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the
default behavior of `mode` typically preserves the axis it acts along. In SciPy
1.11.0, this behavior will change: the default value of `keepdims` will become
False, the `axis` over which the statistic is taken will be eliminated, and the
value None will no longer be accepted. Set `keepdims` to True or False to avoid
this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

[26]:
```
# Evaluate the K-NN model's performance
print("K-Nearest Neighbors (K-NN) Performance:")
```

```
K-Nearest Neighbors (K-NN) Performance:
```

[27]:
```
accuracy_knn = accuracy_score(y_test, y_pred_knn)

print(f"Accuracy: {accuracy_knn:.2f}")
```

```
Accuracy: 0.86
```

[28]:
```
precision_knn = precision_score(y_test, y_pred_knn)
print(f"Precision: {precision_knn:.2f}")
```

```
Precision: 0.73
```

[29]:
```
recall_knn = recall_score(y_test, y_pred_knn)
print(f"Recall: {recall_knn:.2f}")
```

```
Recall: 0.84
```

[30]:
```
f1_score_knn = f1_score(y_test, y_pred_knn)
print(f"F1 Score: {f1_score_knn:.2f}")
```

```
F1 Score: 0.78
```

[32]:
```
roc_auc_knn = roc_auc_score(y_test, y_pred_knn)
print(f"ROC-AUC: {roc_auc_knn:.2f}")
```

```
ROC-AUC: 0.86
```

[33]:
```python
from sklearn.svm import SVC

# Create an SVM model
svm_model = SVC(kernel='linear', C=1)  # You can adjust the kernel and C␣
 ↪parameter

# Train the SVM model on the training data
svm_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred_svm = svm_model.predict(X_test)
```

[35]:
```python
# Evaluate the SVM model's performance
accuracy_svm = accuracy_score(y_test, y_pred_svm)
precision_svm = precision_score(y_test, y_pred_svm)
recall_svm = recall_score(y_test, y_pred_svm)
f1_score_svm = f1_score(y_test, y_pred_svm)
roc_auc_svm = roc_auc_score(y_test, y_pred_svm)

# Print the SVM model's evaluation metrics
print("\nSupport Vector Machine (SVM) Performance:")
print(f"Accuracy: {accuracy_svm:.2f}")
print(f"Precision: {precision_svm:.2f}")
print(f"Recall: {recall_svm:.2f}")
print(f"F1 Score: {f1_score_svm:.2f}")
print(f"ROC-AUC: {roc_auc_svm:.2f}")
```

```
Support Vector Machine (SVM) Performance:
Accuracy: 0.96
Precision: 0.92
Recall: 0.94
F1 Score: 0.93
ROC-AUC: 0.95
```

[ ]: