ML_2

November 10, 2025

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

[3]:
```
df = pd.read_csv(r"C:
 ↪\Users\suraj\OneDrive\Desktop\LP3-master\ML\datasets\emails.csv")
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]
```

[5]:
```
df.shape
```

[5]:
```
(5172, 3002)
```

[9]:
```
X= df['Email No.']
y= df['Prediction']
X.dtypes
```

[9]:
```
dtype('O')
```

[11]:
```
print(df.isnull().sum())
```

```
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
```

[13]:
```
# Step 4: Convert Text into Numeric Form using TF-IDF
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report

vectorizer = TfidfVectorizer(stop_words='english', max_features=3000)
X = vectorizer.fit_transform(X)
```

[17]:
```
#step5 : split the data into training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

```
#step6: Train KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

[17]:
```
KNeighborsClassifier()
```

[19]:
```
#step7: evaluate the model
y_pred = knn.predict(X_test)

print("KNN Accuracy score:",accuracy_score(y_test, y_pred))
print("KNN Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("KNN Classification Report:\n", classification_report(y_test, y_pred,␣
 ↪zero_division=0))
```

```
KNN Accuracy score: 0.7140096618357488
KNN Confusion matrix:
 [[739   0]
 [296   0]]
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.71      1.00      0.83       739
           1       0.00      0.00      0.00       296

    accuracy                           0.71      1035
   macro avg       0.36      0.50      0.42      1035
weighted avg       0.51      0.71      0.59      1035
```

[21]:
```
#MODEL 2
from sklearn.svm import SVC

svm = SVC(kernel='linear')
svm.fit(X_train, y_train)

y_pred_svm= svm.predict(X_test)

#evaluate
print("\n---- SVM Results ----")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred_svm))
print("Classification Report:\n", classification_report(y_test, y_pred_svm,␣
 ↪zero_division=0))
```

```
---- SVM Results ----
Accuracy: 0.7140096618357488
```

```
Confusion matrix:
 [[739   0]
 [296   0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.71      1.00      0.83       739
           1       0.00      0.00      0.00       296

    accuracy                           0.71      1035
   macro avg       0.36      0.50      0.42      1035
weighted avg       0.51      0.71      0.59      1035
```

[23]:
```
knn_acc = accuracy_score(y_test, y_pred)
svm_acc = accuracy_score(y_test, y_pred_svm)

print(" Model Comparison:")
print(f"KNN Accuracy: {knn_acc:.4f}")
print(f"SVM Accuracy: {svm_acc:.4f}")
```

```
 Model Comparison:
KNN Accuracy: 0.7140
SVM Accuracy: 0.7140
```

[ ]:
```

```