In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("C:/Users/Adamin/OneDrive/Desktop/emails.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	i
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	

5 rows × 3002 columns

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

Out[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
```

In [5]:
```
X = df.iloc[:,1:3001]  # word frequency features
X
```

Out[5]:

	the	to	ect	and	for	of	a	you	hou	in	...	enhancements	connevey	jay
0	0	0	1	0	0	0	2	0	0	0	...	0	0	(
1	8	13	24	6	6	2	102	1	27	18	...	0	0	(
2	0	0	1	0	0	0	8	0	0	4	...	0	0	(
3	0	5	22	0	5	1	51	2	10	1	...	0	0	(
4	7	6	17	1	5	2	57	0	9	3	...	0	0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
5167	2	2	2	3	0	0	32	0	0	5	...	0	0	(
5168	35	27	11	2	6	5	151	4	3	23	...	0	0	(
5169	0	0	1	1	0	0	11	0	0	1	...	0	0	(
5170	2	7	1	0	2	1	28	2	0	8	...	0	0	(
5171	22	24	5	1	6	5	148	8	2	23	...	0	0	(

5172 rows × 3000 columns

In [6]:

```
Y = df.iloc[:,-1].values # 1 = spam, 0 = not spam
Y
```

Out[6]:

```
array([0, 0, 0, ..., 1, 1, 0], shape=(5172,))
```

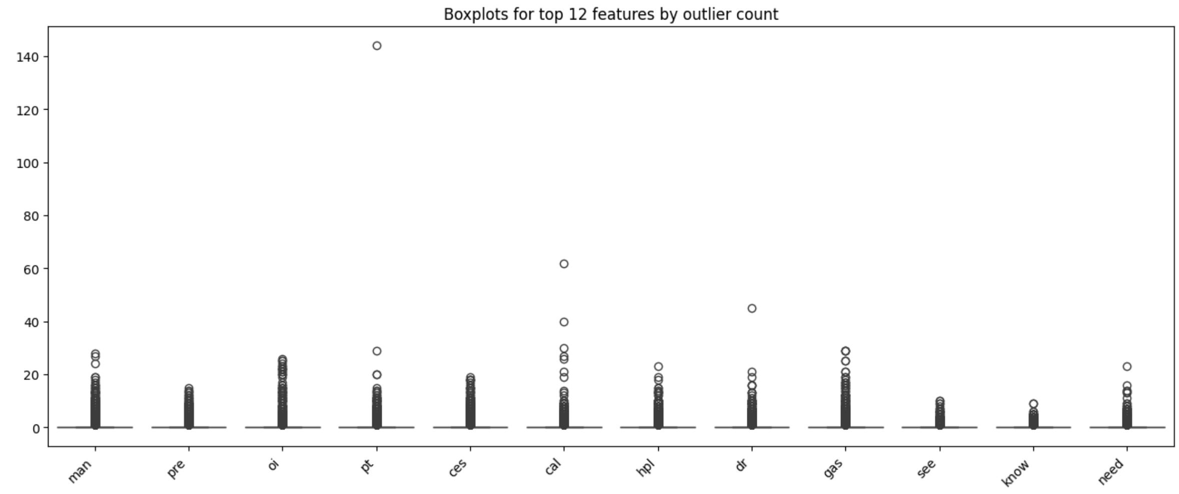
In [8]:

```
df_numeric = df[numeric_cols]
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
# Visualize outliers
import matplotlib.pyplot as plt
import seaborn as sns

# compute IQR outlier counts (you already had this)
Q1 = df_numeric.quantile(0.25)
Q3 = df_numeric.quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR
outlier_mask = ((df_numeric < lower) | (df_numeric > upper))
outlier_counts = outlier_mask.sum().sort_values(ascending=False)

# pick top N features
topN = 12
top_features = outlier_counts.head(topN).index.tolist()

plt.figure(figsize=(16,6))
sns.boxplot(data=df_numeric[top_features])
plt.title(f"Boxplots for top {topN} features by outlier count")
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [9]:
```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random
```

In [10]:
```
from sklearn.metrics import classification_report, confusion_matrix

# -------- Support Vector Machine --------
svc = SVC(C=1.0, kernel='rbf', gamma='auto')
svc.fit(X_train, y_train)
svc_pred = svc.predict(X_test)
```

In [11]:
```
print("SVM Accuracy:", accuracy_score(y_test, svc_pred))
print("SVM Classification Report:\n", classification_report(y_test, svc_pred))
print("SVM Confusion Matrix:\n", confusion_matrix(y_test, svc_pred))
```

```
SVM Accuracy: 0.8932714617169374
SVM Classification Report:
               precision    recall  f1-score   support

           0       0.90      0.96      0.93       913
           1       0.87      0.74      0.80       380

    accuracy                           0.89      1293
   macro avg       0.89      0.85      0.87      1293
weighted avg       0.89      0.89      0.89      1293

SVM Confusion Matrix:
 [[872  41]
 [ 97 283]]
```

In [12]:
```
# -------- K-Nearest Neighbors --------
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_test)
```

In [13]:
```
print("KNN Accuracy:", knn.score(X_test, y_test))
print("KNN Classification Report:\n", classification_report(y_test, knn_pred))
print("KNN Confusion Matrix:\n", confusion_matrix(y_test, knn_pred))
```

```
KNN Accuracy: 0.8662026295436969
KNN Classification Report:
               precision    recall  f1-score   support

           0       0.93      0.87      0.90       913
           1       0.74      0.85      0.79       380

    accuracy                           0.87      1293
   macro avg       0.83      0.86      0.85      1293
weighted avg       0.87      0.87      0.87      1293

KNN Confusion Matrix:
 [[798 115]
 [ 58 322]]
```

In [14]:

```
scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

ks = [1, 3, 5]

results = {}
for k in ks:
    knn = KNeighborsClassifier(n_neighbors=k, n_jobs=-1)
    knn.fit(X_train_s, y_train)              # X_train_s must be scaled features
    y_pred = knn.predict(X_test_s)          # X_test_s must be scaled features

    acc = accuracy_score(y_test, y_pred)
    cm  = confusion_matrix(y_test, y_pred)
    report = classification_report(y_test, y_pred, zero_division=0)

    results[k] = acc

    print(f"\nK = {k}:")
    print(f"  Accuracy = {acc:.4f}")
    print("  Confusion Matrix:")
    print(cm)
    print("  Classification Report:")
    print(report)
```

```
C:\Users\Adamin\AppData\Roaming\Python\Python313\site-packages\joblib\externals\l
oky\backend\context.py:136: UserWarning: Could not find the number of physical co
res for the following reason:
[WinError 2] The system cannot find the file specified
Returning the number of logical cores instead. You can silence this warning by se
tting LOKY_MAX_CPU_COUNT to the number of cores you want to use.
  warnings.warn(
  File "C:\Users\Adamin\AppData\Roaming\Python\Python313\site-packages\joblib\ext
ernals\loky\backend\context.py", line 257, in _count_physical_cores
    cpu_info = subprocess.run(
        "wmic CPU Get NumberOfCores /Format:csv".split(),
        capture_output=True,
        text=True,
    )
  File "C:\Program Files\Python313\Lib\subprocess.py", line 554, in run
    with Popen(*popenargs, **kwargs) as process:
         ~~~~~^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Program Files\Python313\Lib\subprocess.py", line 1039, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
    ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                        pass_fds, cwd, env,
                        ^^^^^^^^^^^^^^^^^^^
    ...<5 lines>...
                        gid, gids, uid, umask,
                        ^^^^^^^^^^^^^^^^^^^^^^
                        start_new_session, process_group)
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Program Files\Python313\Lib\subprocess.py", line 1554, in _execute_chi
ld
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
                       ~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^
                             # no special security
                             ^^^^^^^^^^^^^^^^^^^^^
    ...<4 lines>...
                             cwd,
                             ^^^^
                             startupinfo)
                             ^^^^^^^^^^^^
```

```
K = 1:
  Accuracy = 0.9026
  Confusion Matrix:
[[819  94]
 [ 32 348]]
  Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.90      0.93       913
           1       0.79      0.92      0.85       380

    accuracy                           0.90      1293
   macro avg       0.87      0.91      0.89      1293
weighted avg       0.91      0.90      0.90      1293


K = 3:
  Accuracy = 0.8639
  Confusion Matrix:
[[758 155]
 [ 21 359]]
  Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.83      0.90       913
           1       0.70      0.94      0.80       380

    accuracy                           0.86      1293
   macro avg       0.84      0.89      0.85      1293
weighted avg       0.89      0.86      0.87      1293


K = 5:
  Accuracy = 0.8345
  Confusion Matrix:
[[716 197]
 [ 17 363]]
  Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.78      0.87       913
           1       0.65      0.96      0.77       380

    accuracy                           0.83      1293
   macro avg       0.81      0.87      0.82      1293
weighted avg       0.88      0.83      0.84      1293
```

In [ ]: