```
import numpy as mp
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
df = pd.read_csv('emails.csv', on_bad_lines='skip')
```

```
print(df)
```

```
      Email No.  the  to  ect  and  for  of    a  you  hou  ...  connevey  \
0        Email 1    0   0    1    0    0   0    2    0    0  ...       0.0
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
...          ...  ...  ..  ...  ...  ...  ..  ...  ...  ...  ...       ...
4123  Email 5168    2   2    2    3    0   0   32    0    0  ...       0.0
4124  Email 5169   35  27   11    2    6   5  151    4    3  ...       0.0
4125  Email 5170    0   0    1    1    0   0   11    0    0  ...       0.0
4126  Email 5171    2   7    1    0    2   1   28    2    0  ...       0.0
4127  Email 5172   22  24    5    1    6   5  148    8    2  ...       0.0

      jay  valued  lay  infrastructure  military  allowing   ff  dry  \
0     0.0     0.0  0.0             0.0       0.0       0.0  0.0  0.0
1     0.0     0.0  0.0             0.0       0.0       0.0  1.0  0.0
2     0.0     0.0  0.0             0.0       0.0       0.0  0.0  0.0
3     0.0     0.0  0.0             0.0       0.0       0.0  0.0  0.0
4     0.0     0.0  0.0             0.0       0.0       0.0  1.0  0.0
...   ...     ...  ...             ...       ...       ...  ...  ...
4123  0.0     0.0  0.0             0.0       0.0       0.0  0.0  0.0
4124  0.0     0.0  0.0             0.0       0.0       0.0  1.0  0.0
4125  0.0     0.0  0.0             0.0       0.0       0.0  0.0  0.0
4126  0.0     0.0  0.0             0.0       0.0       0.0  1.0  0.0
4127  0.0     0.0  0.0             0.0       0.0       0.0  0.0  0.0

      Prediction
0            0.0
1            0.0
2            0.0
3            0.0
4            0.0
...          ...
4123         0.0
4124         0.0
4125         1.0
4126         1.0
4127         0.0

[4128 rows x 3002 columns]
```

```
data = df.where((pd.notnull(df)), '')
```

```
data.head(12)
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff
0	Email 1	0	0	1	0	0	0	2	0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	Email 2	8	13	24	6	6	2	102	1	27	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	Email 3	0	0	1	0	0	0	8	0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	Email 4	0	5	22	0	5	1	51	2	10	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	Email 5	7	6	17	1	5	2	57	0	9	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
5	Email 6	4	5	1	4	2	3	45	1	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	Email 7	5	3	1	3	2	1	37	0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	Email 8	0	2	2	3	1	2	21	6	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4128 entries, 0 to 4127
Columns: 3002 entries, Email No. to Prediction
dtypes: int64(1380), object(1622)
memory usage: 94.5+ MB
```

```
data.shape
```

```
(4128, 3002)
```

```
data['Prediction'] = data['Prediction'].replace('', 0)
data['Category'] = data['Prediction'].astype(int)
```

```
/tmp/ipython-input-13341190.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
  data['Prediction'] = data['Prediction'].replace('', 0)
/tmp/ipython-input-13341190.py:2: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling
  data['Category'] = data['Prediction'].astype(int)
```

```
X = data.drop(['Email No.', 'Category'], axis=1)
Y = data['Category']
```

```
print(X)
```

```
      the  to  ect  and  for  of    a  you  hou  in  ...  connevey  jay  \
0       0   0    1    0    0   0    2    0    0   0  ...       0.0  0.0
1       8  13   24    6    6   2  102    1   27  18  ...       0.0  0.0
2       0   0    1    0    0   0    8    0    0   4  ...       0.0  0.0
3       0   5   22    0    5   1   51    2   10   1  ...       0.0  0.0
4       7   6   17    1    5   2   57    0    9   3  ...       0.0  0.0
...   ...  ..  ...  ...  ...  ..  ...  ...  ...  ..  ...       ...  ...
4123    2   2    2    3    0   0   32    0    0   5  ...       0.0  0.0
4124   35  27   11    2    6   5  151    4    3  23  ...       0.0  0.0
4125    0   0    1    1    0   0   11    0    0   1  ...       0.0  0.0
4126    2   7    1    0    2   1   28    2    0   8  ...       0.0  0.0
4127   22  24    5    1    6   5  148    8    2  23  ...       0.0  0.0

      valued  lay  infrastructure  military  allowing   ff  dry  Prediction
0        0.0  0.0             0.0       0.0       0.0  0.0  0.0         0.0
1        0.0  0.0             0.0       0.0       0.0  1.0  0.0         0.0
2        0.0  0.0             0.0       0.0       0.0  0.0  0.0         0.0
3        0.0  0.0             0.0       0.0       0.0  0.0  0.0         0.0
4        0.0  0.0             0.0       0.0       0.0  1.0  0.0         0.0
...      ...  ...             ...       ...       ...  ...  ...         ...
4123     0.0  0.0             0.0       0.0       0.0  0.0  0.0         0.0
4124     0.0  0.0             0.0       0.0       0.0  1.0  0.0         0.0
4125     0.0  0.0             0.0       0.0       0.0  0.0  0.0         1.0
4126     0.0  0.0             0.0       0.0       0.0  1.0  0.0         1.0
4127     0.0  0.0             0.0       0.0       0.0  0.0  0.0         0.0

[4128 rows x 3001 columns]
```

```
print(Y)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
4123    0
4124    0
4125    1
4126    1
4127    0
Name: Category, Length: 4128, dtype: int64
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state = 3)
```

```
print(X.shape)
print(X_train.shape)
print(X_test.shape)
```

```
(4128, 3001)
(3302, 3001)
(826, 3001)
```

```
print(Y.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(4128,)
(3302,)
(826,)
```

```
# Convert all relevant columns in X_train and X_test to numeric,
# coercing errors (empty strings become NaN) and filling NaN values with 0.
for col in X_train.columns:
    X_train[col] = pd.to_numeric(X_train[col], errors='coerce').fillna(0)
for col in X_test.columns:
    X_test[col] = pd.to_numeric(X_test[col], errors='coerce').fillna(0)

Y_train = Y_train.astype('int')
Y_test = Y_test.astype('int')
```

```
print(X_train)
```

```
      the  to  ect  and  for  of    a  you  hou  in  ...  connevey  jay  \
2294    0   0    1    0    1   0    4    0    0   0  ...       0.0  0.0
3461    7   8    3    1    1   2   39    0    0  12  ...       0.0  0.0
2562    1   2    1    0    2   0   14    2    0   1  ...       0.0  0.0
2093    0   0    1    0    1   0    2    0    0   0  ...       0.0  0.0
1348    0   3    2    0    3   2   43    0    1   9  ...       0.0  0.0
...   ...  ..  ...  ...  ...  ..  ...  ...  ...  ..  ...       ...  ...
789     3  11    3    5    6   0   43    0    0  10  ...       0.0  0.0
968     2   1    2    0    0   1   10    0    0   5  ...       0.0  0.0
1667    8   5    8    1    3   2   50    2    5   4  ...       0.0  0.0
3321   31  25   13   22    5  22  194    6    2  48  ...       0.0  0.0
1688    1   8    2    1    1   0   71    5    0  16  ...       0.0  0.0

      valued  lay  infrastructure  military  allowing   ff  dry  Prediction
2294     0.0  0.0             0.0       0.0       0.0  0.0  0.0         0.0
3461     0.0  0.0             0.0       0.0       0.0  2.0  0.0         0.0
2562     0.0  0.0             0.0       0.0       0.0  0.0  0.0         1.0
2093     0.0  0.0             0.0       0.0       0.0  0.0  0.0         0.0
1348     0.0  0.0             0.0       0.0       0.0  2.0  0.0         0.0
...      ...  ...             ...       ...       ...  ...  ...         ...
789      0.0  0.0             0.0       0.0       0.0  1.0  0.0         0.0
968      0.0  0.0             0.0       0.0       0.0  1.0  0.0         0.0
1667     0.0  0.0             0.0       0.0       0.0  1.0  0.0         0.0
3321     0.0  1.0             0.0       0.0       0.0  2.0  0.0         1.0
1688     0.0  0.0             0.0       0.0       0.0  0.0  0.0         1.0

[3302 rows x 3001 columns]
```

```
print(X_train_features)
```

```
<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 2739 stored elements and shape (3001, 2739)>
  Coords        Values
  (2, 743)      1.0
  (8, 1152)     1.0
  (13, 793)     1.0
  (27, 438)     1.0
  (30, 1001)    1.0
  (34, 601)     1.0
  (36, 1556)    1.0
  (37, 1159)    1.0
  (43, 527)     1.0
  (49, 1343)    1.0
  (50, 1639)    1.0
  (52, 968)     1.0
  (53, 1651)    1.0
  (58, 1591)    1.0
  (67, 1915)    1.0
  (68, 583)     1.0
  (71, 455)     1.0
  (74, 1389)    1.0
  (79, 2721)    1.0
  (80, 893)     1.0
  (81, 171)     1.0
  (83, 1234)    1.0
  (85, 1551)    1.0
  (86, 598)     1.0
  (87, 2486)    1.0
  :     :
  (2976, 2299)  1.0
  (2977, 1198)  1.0
  (2978, 887)   1.0
  (2979, 899)   1.0
  (2980, 1095)  1.0
  (2981, 918)   1.0
  (2982, 2019)  1.0
  (2983, 606)   1.0
  (2984, 2049)  1.0
  (2985, 183)   1.0
  (2986, 1851)  1.0
```

```
  (2987, 1957)  1.0
  (2988, 2072)  1.0
  (2989, 1190)  1.0
  (2990, 790)   1.0
  (2991, 487)   1.0
  (2992, 1293)  1.0
  (2993, 2607)  1.0
  (2994, 1371)  1.0
  (2995, 1236)  1.0
  (2996, 1572)  1.0
  (2997, 96)    1.0
  (2998, 914)   1.0
  (2999, 715)   1.0
  (3000, 1895)  1.0
```

```
model = LogisticRegression()
```

```
model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression

```
LogisticRegression()
```

```
prediction_on_training_data = model.predict(X_train)
accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)
```

```
print('Acc on training data :', accuracy_on_training_data)
```

```
Acc on training data : 0.9954572986069049
```

```
prediction_on_test_data = model.predict(X_test)
accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)
```

```
print('Accuracy on test data:', accuracy_on_test_data)
```

```
Accuracy on test data: 0.9806295399515739
```

```
import re
from collections import Counter

input_your_mail = ["""Conversation opened. 1 unread message.\n\nSkip to content\nUsing Gmail with screen readers\nin:spam\r

# Get the list of features (words) from the training data
model_features = X_train.columns.tolist()

# Preprocess the input email
# Convert to lowercase and tokenize words
processed_mail = re.findall(r'\\b\\w+\\b', input_your_mail[0].lower())
word_counts = Counter(processed_mail)

# Create a dictionary for the new email's features, initialized to 0
new_email_features_dict = {feature: 0 for feature in model_features}

# Fill in the counts for words present in the model's features
for word, count in word_counts.items():
    if word in new_email_features_dict:
        new_email_features_dict[word] = count

# Convert the dictionary to a DataFrame, ensuring column order matches X_train
input_data_features = pd.DataFrame([new_email_features_dict])

prediction = model.predict(input_data_features)
```

```
print("Prediction:", "Spam" if prediction[0] == 1 else "Not Spam")
```

```
Prediction: Not Spam
```