In [15]:
```
#!/usr/bin/env python

import pandas as pd
import numpy as np
import random
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

In [6]:
```
# Reading data from csv file
data = pd.read_csv("data.csv")
data.head()
```

Out[6]:

	url	label
0	diaryofagameaddict.com	bad
1	espdesign.com.au	bad
2	iamagameaddict.com	bad
3	kalantzis.net	bad
4	slightlyoffcenter.net	bad

In [7]:
```
# Labels
y = data["label"]

# Features
url_list = data["url"]
```

In [8]:
```
# Using Tokenizer
vectorizer = TfidfVectorizer()

# Store vectors into X variable as Our XFeatures
X = vectorizer.fit_transform(url_list)
```

In [23]:
```
# Split into training and testing dataset 80:20 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
url_train, url_test = train_test_split(url_list, test_size=0.2, random_state=42)
```

In [10]:
```
# Model Building using Logistic regression
logit = LogisticRegression()
logit.fit(X_train, y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460:
ConvergenceWarning: lbfgs failed to converge (status=1):
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(
```

Out[10]:
```
▾ LogisticRegression
LogisticRegression()
```

In [12]:
```
# Predictions
y_pred = logit.predict(X_test)
```

In [13]:
```
# Accuracy of Our Model
print("Accuracy of our model is: ", logit.score(X_test, y_test))
```

```
Accuracy of our model is:  0.9645987180859287
```

In [16]:
```
# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

         bad       0.97      0.83      0.89     14964
        good       0.96      0.99      0.98     69129

    accuracy                           0.96     84093
   macro avg       0.97      0.91      0.94     84093
weighted avg       0.96      0.96      0.96     84093
```

In [19]:
```
# Sample size (we will just take the first 5 URLs for this demonstration)
sample_size = 5
```

In [24]:
```
# Create a DataFrame showing the first 5 samples of actual vs predicted values
result_df = pd.DataFrame({
    'URL': url_test[:sample_size],  # Extract the first few URLs
    'Actual Label': y_test[:sample_size].values,  # Actual labels
    'Predicted Label': y_pred[:sample_size]  # Predicted labels
})
```

In [25]:
```
print("\nSample Actual vs Predicted Results:")
print(result_df)
```

```
Sample Actual vs Predicted Results:
                                                      URL Actual Label  \
153858  trib.com/lifestyles/announcements/community/ar...         good
268846   jcbsolutions.com/english/realisations_domtar.htm         good
119204                              morgenbrookehomes.com/         good
225271                    empireonline.com/100britishfilms/         good
147290                          steugenecatholicchurch.org/         good

       Predicted Label
153858            good
268846            good
119204            good
225271            good
147290            good
```

In [26]:
```
# Count the number of good and bad URLs
counts = pd.Series(y_pred).value_counts()
print("\nCount of Good and Bad URLs in the Predictions:")
print(counts)
```

```
Count of Good and Bad URLs in the Predictions:
good    71334
bad     12759
Name: count, dtype: int64
```

In [28]:
```
# Create DataFrames for "good" and "bad" URLs
good_urls_df = pd.DataFrame({
```

```
    'URL': url_test[y_pred == 'good'],  # Filter URLs where the prediction is 'good
    'Actual Label': y_test[y_pred == 'good'].values,  # Corresponding actual labels
    'Predicted Label': y_pred[y_pred == 'good']  # Predicted labels
})
```

In [29]:

```
bad_urls_df = pd.DataFrame({
    'URL': url_test[y_pred == 'bad'],  # Filter URLs where the prediction is 'bad'
    'Actual Label': y_test[y_pred == 'bad'].values,  # Corresponding actual labels
    'Predicted Label': y_pred[y_pred == 'bad']  # Predicted labels
})
```

In [30]:

```
# Display the tables
print("\nTable of Good URLs:")
print(good_urls_df)
```

```
Table of Good URLs:
                                                      URL Actual Label  \
153858  trib.com/lifestyles/announcements/community/ar...         good
268846   jcbsolutions.com/english/realisations_domtar.htm         good
119204                              morgenbrookehomes.com/         good
225271                     empireonline.com/100britishfilms/         good
147290                          steugenecatholicchurch.org/         good
...                                                   ...          ...
342695  starsmaster.com/g/genevieve_bujold_01/topceleb...         good
242466                               first-unitarian-pgh.org/         good
326876                    reverbnation.com/kerrypatrickclark         good
348490                         thefreedictionary.com/cleric         good
91572   filetram.com/download/mediafire/1045786576/all...         good

       Predicted Label
153858            good
268846            good
119204            good
225271            good
147290            good
...                ...
342695            good
242466            good
326876            good
348490            good
91572             good

[71334 rows x 3 columns]
```

In [31]:

```
print("\nTable of Bad URLs:")
print(bad_urls_df)
```

```
Table of Bad URLs:
                                                   URL Actual Label  \
418900                       222.186.21.204:8880/lsxm.7          bad   
414553                                  minee.cm/xyjwc          bad   
406691                                  coinpack.info/          bad   
404242                          club.konjiki.jp/n7mm85          bad   
410284                             keshamrit.com/7fg3g          bad   
...                                                ...          ...   
10515   pulse-entertainment.ca/blog/wp-content/1f3a101...          bad   
52698                                benindiary.info/         good   
411527                www.ralrab.com/rar/wrar531nl.exe          bad   
410181                   atronis.com/images/gallery.php          bad   
29249   surubiproducciones.com/wp/wp-includes/1/nel/ar...          bad   

       Predicted Label  
418900             bad  
414553             bad  
406691             bad  
404242             bad  
410284             bad  
...                ...  
10515              bad  
52698              bad  
411527             bad  
410181             bad  
29249              bad  

[12759 rows x 3 columns]
```