Project Notebook

Before you start:

- **Make your own copy** of this notebook
  - Select 'File' --> 'Save a copy in Drive...'
  - Add your name in front of the title of the notebook by clicking on the file name above.

Webpage Phishing Detection

Authorship and Resources Used

- Jacob Flores
- I used Gemini to help me solve syntax problems and improve visualizations

---

Data Description and Source

- The provided dataset includes 11430 URLs with 87 extracted features. The dataset is designed to be used as benchmarks for machine learning-based phishing detection systems. Features are from three different classes: 56 extracted from the structure and syntax of URLs, 24 extracted from the content of their correspondent pages, and 7 are extracted by querying external services.
- The dataset is balanced, it contains exactly 50% phishing and 50% legitimate URLs.
- This data was extracted from Kaggle and contains 11430 rows and 87 columns
- https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset/data

---

Research Question

- Can we predict whether a webpage is phishing or legitimate based on its URL and other website characteristics?
- Motivation: Phishing is a serious threat, and being able to automatically detect phishing websites could protect users from scams and data breaches.
- Relevance: This is a relevant problem because phishing attacks are becoming increasingly sophisticated, making it difficult for users to identify them manually.

Import Libraries and Set Preferences for Visualization

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
from sklearn.preprocessing import LabelEncoder, StandardScaler

# add any additional libraries or plot preferences to this block
import os

# kaggle data name (poster's username followed by data name)
kaggle_data_name = 'shashwatwork/web-page-phishing-detection-dataset' # <--edit

# kaggle user name (your username on Kaggle)
kaggle_user = 'jacobbb593'   # <--edit this

# kaggle key (your key)
kaggle_key = 'd717ba8bf5f04970837ca3d8973d941c' # <--edit this

#  create the local file name with path
zip_name = f'/content/{kaggle_data_name.split("/")[1]}.zip'

# check if the file already exists in the local file structure
if not os.path.exists(zip_name):

  # install kaggle api
  !pip install -q kaggle

  # authentication with Kaggle
  os.environ['KAGGLE_USERNAME'] = kaggle_user
  os.environ['KAGGLE_KEY'] = kaggle_key

  # download data from Kaggle
  !kaggle datasets download -d $kaggle_data_name
```

```

  # unzip the downloaded data
  !unzip -u $zip_name

else:
  print('Data already downloaded!')

# Create a list of local files
files = [f for f in os.listdir('.') if os.path.isfile(f) and not f.endswith('zip

# print the paths of the local files
if len(files) == 0:
  print('\nNo local files are available.')
else:
  print('\nThe following file(s) are available:')
  for f in files:
    print(f'/content/{f}')
```

```
Dataset URL: https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-dete
License(s): Attribution 4.0 International (CC BY 4.0)
Downloading web-page-phishing-detection-dataset.zip to /content
  0% 0.00/1.01M [00:00<?, ?B/s]
100% 1.01M/1.01M [00:00<00:00, 17.7MB/s]
Archive:  /content/web-page-phishing-detection-dataset.zip
  inflating: dataset_phishing.csv

The following file(s) are available:
/content/dataset_phishing.csv
```

Read and Verify Data

```
# Load the dataset
df = pd.read_csv("/content/dataset_phishing.csv")
df
```

	url	length_url	length_hostname	ip	n
0	http://www.crestonwood.com/router.php	37	19	0	
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	
2	https://support-appleld.com.secureupdate.duila...	126	50	1	
3	http://rgipt.ac.in	18	11	0	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	
...	...	...	...	...	
11425	http://www.fontspace.com/category/blackletter	45	17	0	
11426	http://www.budgetbots.com/server.php/Server%20...	84	18	0	
11427	https://www.facebook.com/Interactive-Televisio...	105	16	1	
11428	http://www.mypublicdomainpictures.com/	38	30	0	
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	477	14	1	

11430 rows × 89 columns

```
1 df['status'].value_counts(normalize=True)
```

status	proportion
legitimate	0.5
phishing	0.5

dtype: float64

```
1 print(df.columns)
```

```
Index(['url', 'length_url', 'length_hostname', 'ip', 'nb_dots', 'nb_hyphens',
       'nb_at', 'nb_qm', 'nb_and', 'nb_or', 'nb_eq', 'nb_underscore',
       'nb_tilde', 'nb_percent', 'nb_slash', 'nb_star', 'nb_colon', 'nb_comma',
       'nb_semicolumn', 'nb_dollar', 'nb_space', 'nb_www', 'nb_com',
       'nb_dslash', 'http_in_path', 'https_token', 'ratio_digits_url',
       'ratio_digits_host', 'punycode', 'port', 'tld_in_path',
       'tld_in_subdomain', 'abnormal_subdomain', 'nb_subdomains',
       'prefix_suffix', 'random_domain', 'shortening_service',
       'path_extension', 'nb_redirection', 'nb_external_redirection',
       'length_words_raw', 'char_repeat', 'shortest_words_raw',
       'shortest_word_host', 'shortest_word_path', 'longest_words_raw',
       'longest_word_host', 'longest_word_path', 'avg_words_raw',
       'avg_word_host', 'avg_word_path', 'phish_hints', 'domain_in_brand',
       'brand_in_subdomain', 'brand_in_path', 'suspecious_tld',
```

```
       'statistical_report', 'nb_hyperlinks', 'ratio_intHyperlinks',
       'ratio_extHyperlinks', 'ratio_nullHyperlinks', 'nb_extCSS',
       'ratio_intRedirection', 'ratio_extRedirection', 'ratio_intErrors',
       'ratio_extErrors', 'login_form', 'external_favicon', 'links_in_tags',
       'submit_email', 'ratio_intMedia', 'ratio_extMedia', 'sfh', 'iframe',
       'popup_window', 'safe_anchor', 'onmouseover', 'right_clic',
       'empty_title', 'domain_in_title', 'domain_with_copyright',
       'whois_registered_domain', 'domain_registration_length', 'domain_age',
       'web_traffic', 'dns_record', 'google_index', 'page_rank', 'status'],
      dtype='object')
```

Analyses and Visualizations

- (Delete this instruction) Include code comments in each step to clearly articulate what you are doing and why you are doing it.

```
# Handling missing values
df = df.dropna()

# Encoding categorical variables
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    df[column] = label_encoders[column].fit_transform(df[column])

# Standardizing continuous variables
scaler = StandardScaler()
df[df.select_dtypes(include=['float64', 'int64']).columns] = scaler.fit_transfor
```

```
# Summary statistics
print(df.describe())
```

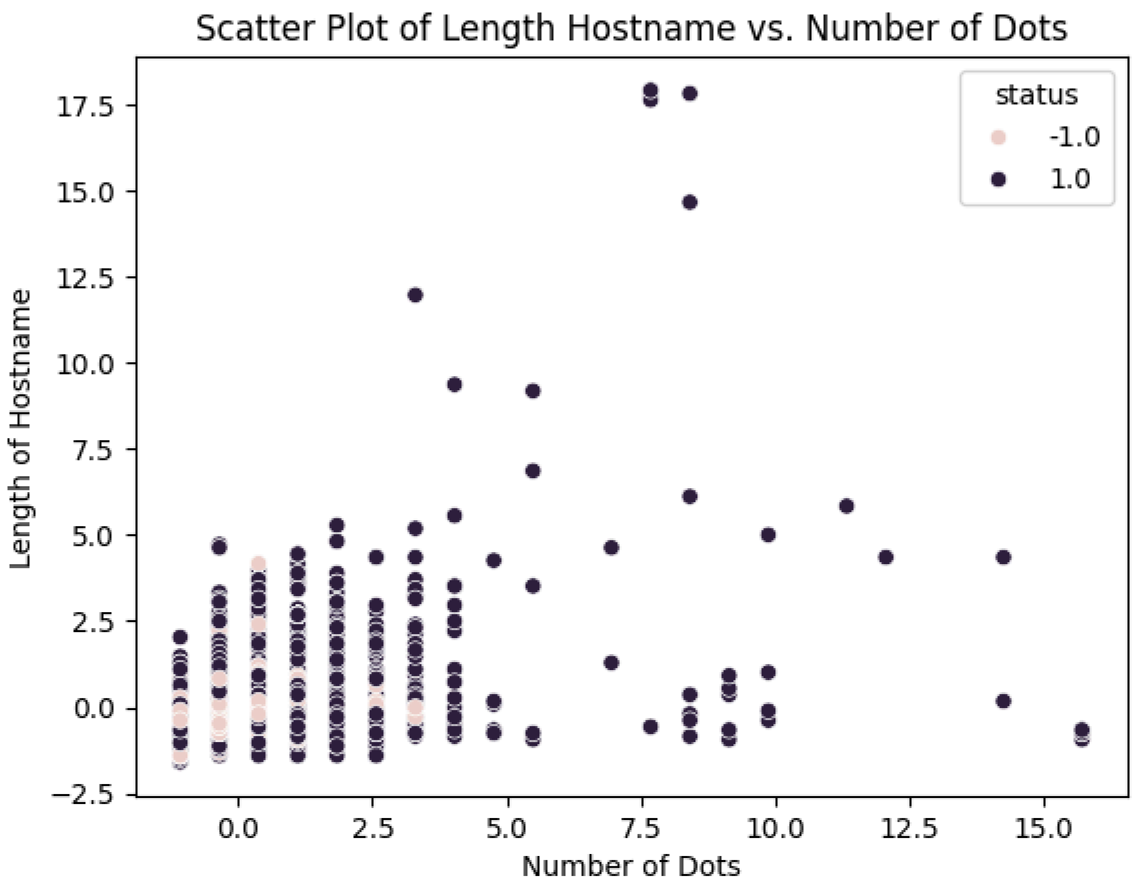
```
                url    length_url  length_hostname            ip  \
count  1.143000e+04  1.143000e+04     1.143000e+04  1.143000e+04
mean  -9.138214e-17 -5.221836e-17    -1.367624e-16  5.284001e-18
std    1.000044e+00  1.000044e+00     1.000044e+00  1.000044e+00
min   -1.731701e+00 -8.884488e-01    -1.585856e+00 -4.210204e-01
25%   -8.660163e-01 -5.086669e-01    -5.651349e-01 -4.210204e-01
50%   -2.826667e-05 -2.554790e-01    -1.939637e-01 -4.210204e-01
75%    8.659598e-01  1.785574e-01     2.700002e-01 -4.210204e-01
max    1.731948e+00  2.857177e+01     1.790063e+01  2.375182e+00

            nb_dots    nb_hyphens         nb_at         nb_qm        nb_and  \
count  1.143000e+04  1.143000e+04  1.143000e+04  1.143000e+04  1.143000e+04
mean   4.693436e-17  5.284001e-18  1.181130e-17 -1.554118e-17  1.989271e-17
std    1.000044e+00  1.000044e+00  1.000044e+00  1.000044e+00  1.000044e+00
min   -1.081136e+00 -4.779840e-01 -1.429146e-01 -3.874641e-01 -1.976037e-01
25%   -3.510099e-01 -4.779840e-01 -1.429146e-01 -3.874641e-01 -1.976037e-01
50%   -3.510099e-01 -4.779840e-01 -1.429146e-01 -3.874641e-01 -1.976037e-01
```

```
75%    3.791163e-01  1.173790e-03 -1.429146e-01 -3.874641e-01 -1.976037e-01
max    1.571177e+01  2.012580e+01  2.558171e+01  7.844347e+00  2.293641e+01

         nb_or  ...  domain_in_title  domain_with_copyright  \
count  11430.0  ...     1.143000e+04           1.143000e+04   
mean       0.0  ...     7.335437e-17          -5.843484e-17   
std        0.0  ...     1.000044e+00           1.000044e+00   
min        0.0  ...    -1.860473e+00          -8.855872e-01   
25%        0.0  ...     5.374978e-01          -8.855872e-01   
50%        0.0  ...     5.374978e-01          -8.855872e-01   
75%        0.0  ...     5.374978e-01           1.129194e+00   
max        0.0  ...     5.374978e-01           1.129194e+00   

       whois_registered_domain  domain_registration_length    domain_age  \
count             1.143000e+04                1.143000e+04  1.143000e+04   
mean              6.216472e-19                2.237930e-17 -1.243294e-18   
std               1.000044e+00                1.000044e+00  1.000044e+00   
min              -2.803697e-01               -6.057589e-01 -1.311134e+00   
25%              -2.803697e-01               -5.014303e-01 -9.944154e-01   
50%              -2.803697e-01               -3.075019e-01 -2.237825e-02   
75%              -2.803697e-01               -5.343119e-02  9.538421e-01   
max               3.566720e+00                3.600743e+01  2.835409e+00   

        web_traffic    dns_record  google_index     page_rank        status
count  1.143000e+04  1.143000e+04  1.143000e+04  1.143000e+04  11430.000000
mean  -1.429789e-17  1.150047e-17 -9.138214e-17 -8.703061e-18      0.000000
std    1.000044e+00  1.000044e+00  1.000044e+00  1.000044e+00      1.000044
min   -4.293403e-01 -1.433029e-01 -1.070361e+00 -1.255788e+00     -1.000000
25%   -4.293403e-01 -1.433029e-01 -1.070361e+00 -8.615977e-01     -1.000000
50%   -4.285130e-01 -1.433029e-01  9.342641e-01 -7.321666e-02      0.000000
75%   -2.419978e-01 -1.433029e-01  9.342641e-01  7.151644e-01      1.000000
max    4.966743e+00  6.978227e+00  9.342641e-01  2.686117e+00      1.000000

[8 rows x 89 columns]
```

```
# Countplot of Phishing Status
plt.figure(figsize=(6, 4))
sns.countplot(x='status', data=df)
plt.title('Distribution of Phishing Status')
plt.xlabel('Status (0: Legitimate, 1: Phishing)')
plt.ylabel('Count')
plt.show()
```

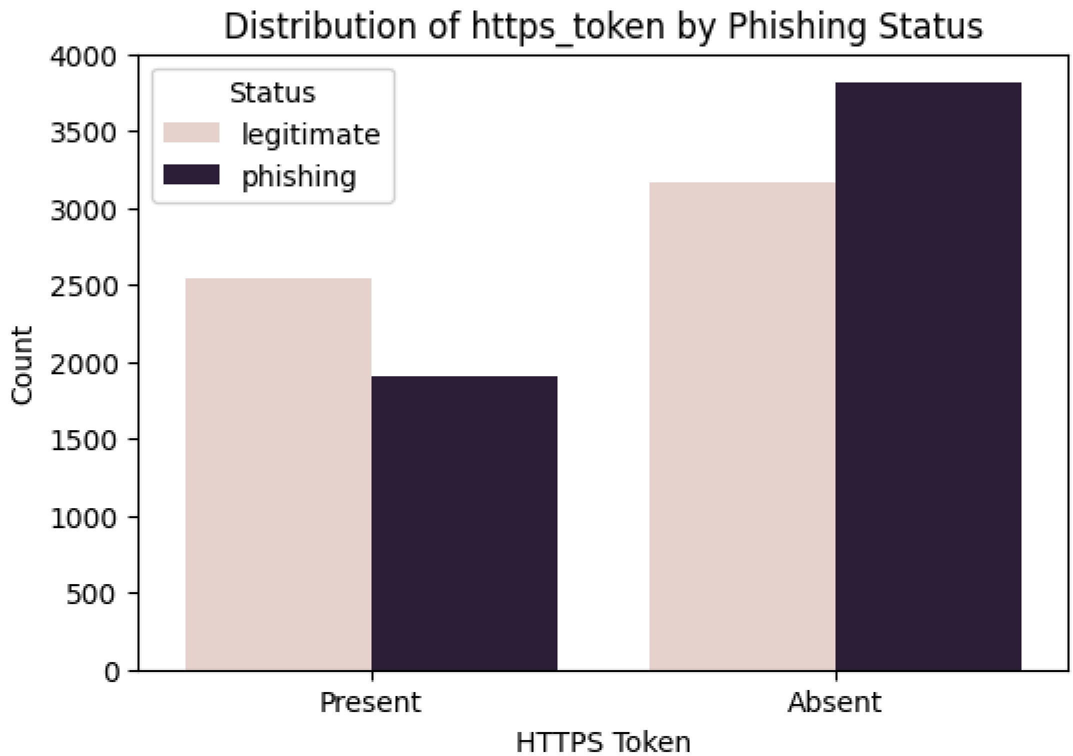

```
# Multiple features for box plots
features_for_boxplots = ['length_url', 'nb_dots', 'nb_slash']
y_axis_labels = ['URL Length', 'Number of Dots', 'Number of slashes']

# Subplots for each feature
fig, axes = plt.subplots(1, len(features_for_boxplots), figsize=(15, 5))

# Box plots for each feature
for i, (feature, label) in enumerate(zip(features_for_boxplots, y_axis_labels)):
    sns.boxplot(x='status', y=feature, data=df, ax=axes[i])
    axes[i].set_title(f'Box Plot of {feature} by Status')
    axes[i].set_xlabel('Status (0: Legitimate, 1: Phishing)')
    axes[i].set_ylabel(label)

plt.tight_layout()  # Adjust spacing between subplots
plt.show()
```


```python
# Scatter plot
sns.scatterplot(x='nb_dots', y='length_hostname', hue='status', data=df)
plt.title('Scatter Plot of Length Hostname vs. Number of Dots')
plt.ylabel('Length of Hostname')
plt.xlabel('Number of Dots')
plt.show()
```

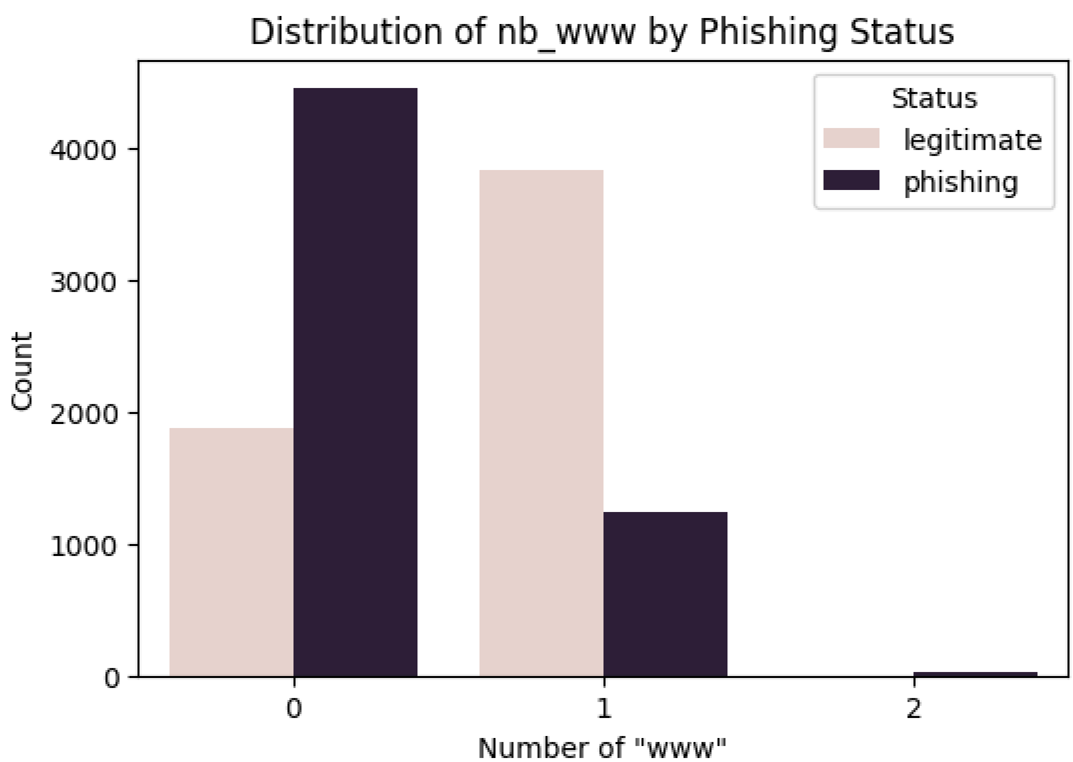



```python
categorical_features = ['https_token', 'nb_www']

for feature in categorical_features:
    plt.figure(figsize=(6, 4))
    ax = sns.countplot(x=feature, data=df, hue='status')
    plt.title(f'Distribution of {feature} by Phishing Status')
    plt.ylabel('Count')

    # Customizing x-axis labels
    if feature == 'https_token':
        ax.set_xticklabels(['Present', 'Absent'])
        plt.xlabel('HTTPS Token')
    elif feature == 'nb_www':
        ax.set_xticklabels(['0', '1', '2'])
        plt.xlabel('Number of "www"')

    # Customizing legend labels
    handles, labels = ax.get_legend_handles_labels()
    ax.legend(handles, ['legitimate', 'phishing'], title='Status')

    plt.show()
```

```
<ipython-input-10-d38cc637e9da>:11: UserWarning: set_ticklabels() should only be
  ax.set_xticklabels(['Present', 'Absent'])
```



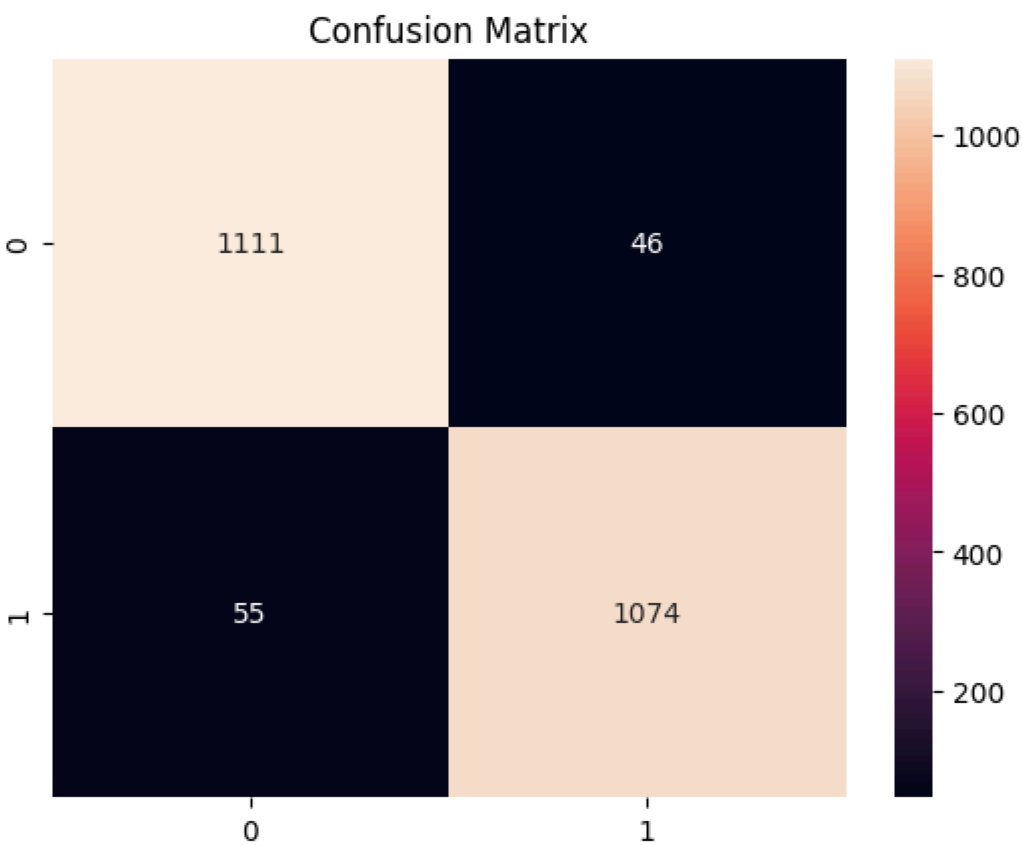

```
<ipython-input-10-d38cc637e9da>:14: UserWarning: set_ticklabels() should only be
  ax.set_xticklabels(['0', '1', '2'])
```




```
1 # Splitting data
2 X = df.drop('status', axis=1)
3 y = df['status']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Logistic Regression Model
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
y_pred = lr_model.predict(X_test)

# Model evaluation
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(f'Precision: {precision_score(y_test, y_pred)}')
print(f'Recall: {recall_score(y_test, y_pred)}')
print(f'F1 Score: {f1_score(y_test, y_pred)}')
print(f'ROC-AUC Score: {roc_auc_score(y_test, y_pred)}')

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.9558180227471567
Precision: 0.9589285714285715
Recall: 0.9512843224092117
F1 Score: 0.9550911516229436
ROC-AUC Score: 0.9557631637975184
```




```
# Cross-validation
cv_scores = cross_val_score(lr_model, X, y, cv=5, scoring='roc_auc'
```

```
print(f'Cross-validated ROC-AUC scores: {cv_scores}')
print(f'Mean ROC-AUC score: {np.mean(cv_scores)}')
```

```
Cross-validated ROC-AUC scores: [0.98325308 0.98789007 0.98553637 0.98541696 0.9
Mean ROC-AUC score: 0.9850804738646515
```

```
# Model Coefficients
coefficients = pd.DataFrame({"Feature": X.columns, "Coefficient": lr_model.coef_
coefficients = coefficients.sort_values(by="Coefficient", ascending=False)
print(coefficients)
```

```
              Feature  Coefficient
45  longest_words_raw     3.566386
86       google_index     1.490370
18      nb_semicolumn     1.468793
51        phish_hints     1.412147
2     length_hostname     0.835549
..                ...          ...
5          nb_hyphens    -0.882293
57      nb_hyperlinks    -0.993297
49      avg_word_host    -1.095541
21             nb_www    -1.107017
87          page_rank    -1.530128

[88 rows x 2 columns]
```

Conclusions

In conclusion, my project on webpage phishing detection shows that we can use URL and website characteristics to predict phishing sites effectively. By analyzing and visualizing the data, I found that features like URL length, the number of dots, and the presence of HTTPS tokens are important in distinguishing phishing from legitimate webpages. The logistic regression model I trained performed well, with an accuracy of 95.58% and an F1 score of 95.51%. These results show that the model is very good at identifying phishing sites with few mistakes.

Completed the exercise?