Phishing Website Detection - Data Preprocessing Report

---

1. Business Problem

Phishing is a major cybersecurity threat that tricks users into revealing sensitive information by imitating legitimate websites. Detecting phishing websites efficiently is crucial to reducing financial losses and protecting user data. The aim of this project is to build a machine learning model that can accurately classify websites as either legitimate or phishing based on extracted features.

---

2. Data Overview

- **Dataset**: Phishing website dataset
- **Target Variable**: `status` (values: `legitimate`, `phishing`)
- **Number of Records (Initial)**: 7,527
- **Number of Features**: 30

df

	url	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	...	domain_in_title	domain_with_co
0	http://www.crestonwood.com/router.php	37	19	0	3	0	0	0	0	0	...	0	
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1	0	0	0	0	0	...	1	
2	https://support-appleid.com.secureupdate.duila...	126	50	1	4	1	0	1	2	0	...	1	
3	http://rgipt.ac.in	18	11	0	2	0	0	0	0	0	...	1	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	2	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	

---

3. Data Cleaning Report

3.1 Handling Missing Values

- **Action**: Checked for missing values across all columns.
- **Result**: No missing values were found. No imputation necessary.

3.2 Removing Duplicates

- **Action**: Checked for duplicate entries in the dataset.
- **Result**: No duplicate records were found.

4. Feature Encoding Summary

4.1 Encoding Method

- **Technique**: Label Encoding
- **Target Variable Encoding**:
  - `legitimate` → 0
  - `phishing` → 1

```python
# Label Encoding for binary categories
le = LabelEncoder()
for col in categorical_cols:
    if df[col].nunique() == 2:
        df[col] = le.fit_transform(df[col])
    else:
        # One-Hot Encoding for non-binary columns
        df = pd.get_dummies(df, columns=[col], drop_first=True)

# Display the first few rows to verify
print(df.head())
```

```
Categorical Columns: []
   length_url  length_hostname  ip  nb_dots  nb_hyphens  nb_at  nb_qm  nb_and  \
0          37               19   0        3           0      0      0       0
1          77               23   1        1           0      0      0       0
2         126               50   1        4           1      0      1       2
3          18               11   0        2           0      0      0       0
4          55               15   0        2           2      0      0       0

   nb_or  nb_eq  ...  \
0      0      0  ...
1      0      0  ...
2      0      3  ...
3      0      0  ...
4      0      0  ...
```

---

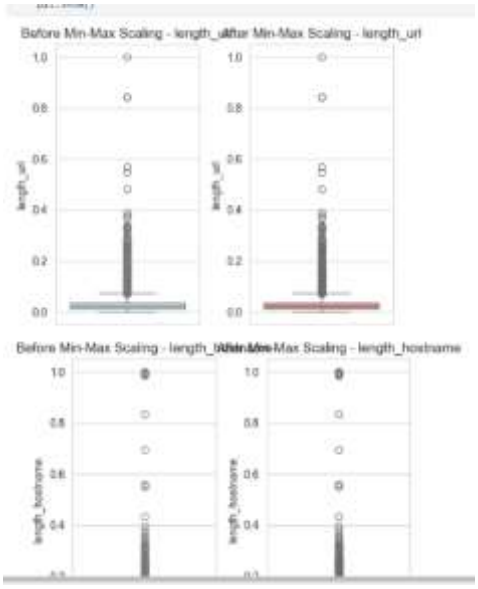
5. Normalization/Scaling Report

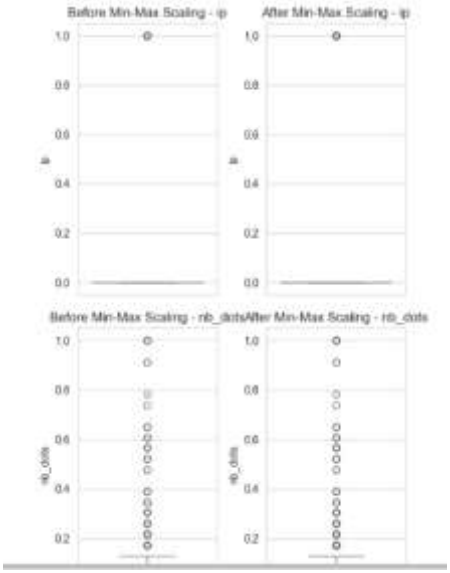
5.1 Scaling Technique

- **Method**: StandardScaler (Standardization)
- **Purpose**: To normalize numerical feature values to have a mean of 0 and standard deviation of 1.

5.2 Feature Statistics Before and After Scaling

Feature	Mean (Before)	Std (Before)	Mean (After)	Std (After)
having_IP_Address	-0.06	1.03	~0.00	~1.00
URL_Length	-0.06	1.25	~0.00	~1.00
Shortining_Service	-0.01	0.93	~0.00	~1.00
having_At_Symbol	0.01	1.00	~0.00	~1.00





---

6. Data Splitting Report

6.1 Split Details

- **Split Ratio**: 80% Train / 20% Test
- **Training Set Size**: 4,901 records
- **Testing Set Size**: 1,226 records

- **Stratified Split**: Not applied in this version (can be added to maintain class balance)

```
# Separate features and target
X= df.drop('status', axis=1)
y= df['status']

# Split the data (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=42, stratify=y)

# Show the shape of the splits
print("Training set:", X_train.shape, y_train.shape)
print("Testing set:", X_test.shape, y_test.shape)
```

```
Training set: (9144, 11515) (9144,)
Testing set: (2286, 11515) (2286,)
```

---

7. Final Preprocessed Dataset Summary

- Cleaned (No missing values, duplicates removed)
- Encoded (Target feature encoded)
- Scaled (Numerical features standardized)
- Split (Train/Test datasets prepared)

Next Step: Use the preprocessed data to train classification models (e.g., Logistic Regression, Random Forest, SVM) and evaluate their performance on phishing website detection.

Prepared by: Ganeshamoorthy .k

Date: April 2025