Project Mid Point Submission

Artificial Intelligence 2024W (DSMM Group 2)

Group 4

Aishwarya Karki (c0903073)

Bikash Thapa Magar (c0907642)

Nirmala Regmi (c0903616)

Samar Fathima (c0908466)

Sijal Shrestha (c0910639)

Sujata Gurung (c0903143)

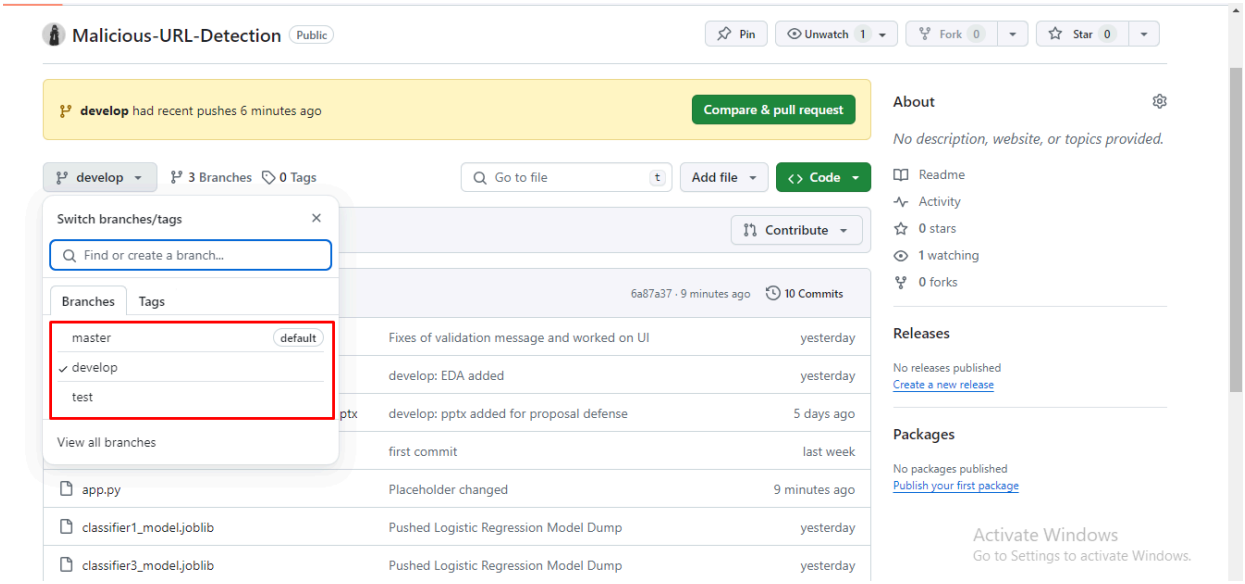
1. Git Repository Walkthrough

a. Branching Strategy

i. The **master** branch represents the production-ready codebase.

ii. The **develop** branch serves as the integration branch for ongoing development work.

iii. The **test** branch is used for testing purposes, separate from both development and production environments



b. Workflow and best practices

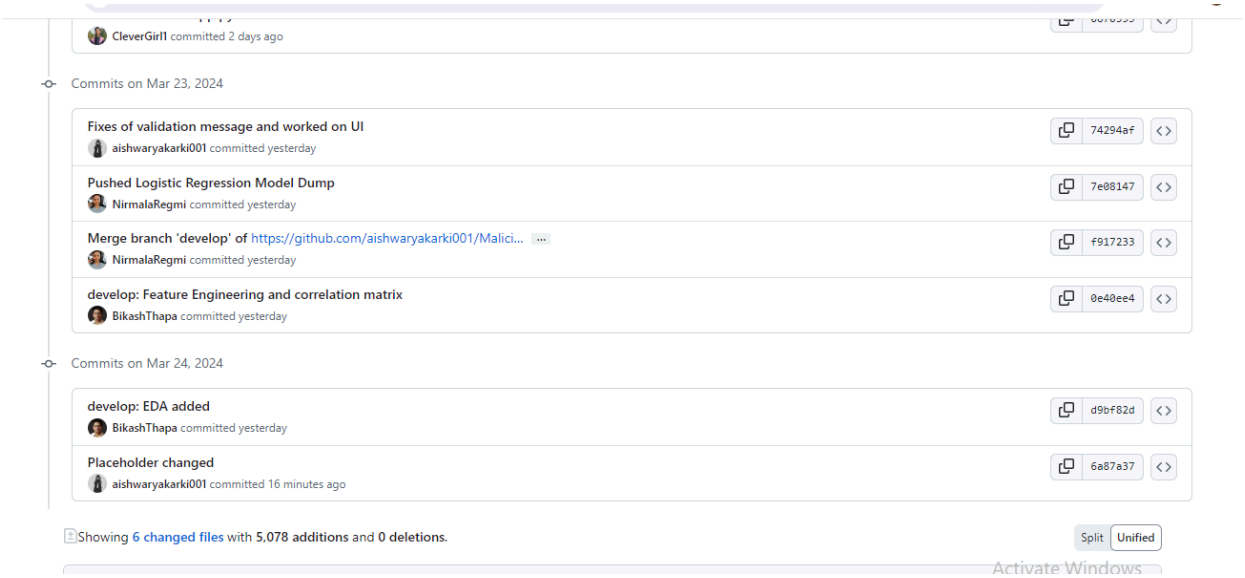
i. Performed apt pull requests

ii. Worked on different test branches and performed merges directly on Develop branch.

iii. Discussed team activities and recent code changes before making any commits.

iv. Team members made Github commits with clear comments dictating code changes.

v. Followed Scrum board closely while creating subtasks for required activities.



2. Data Preprocessing and Feature Engineering

a. Data understanding and Cleaning

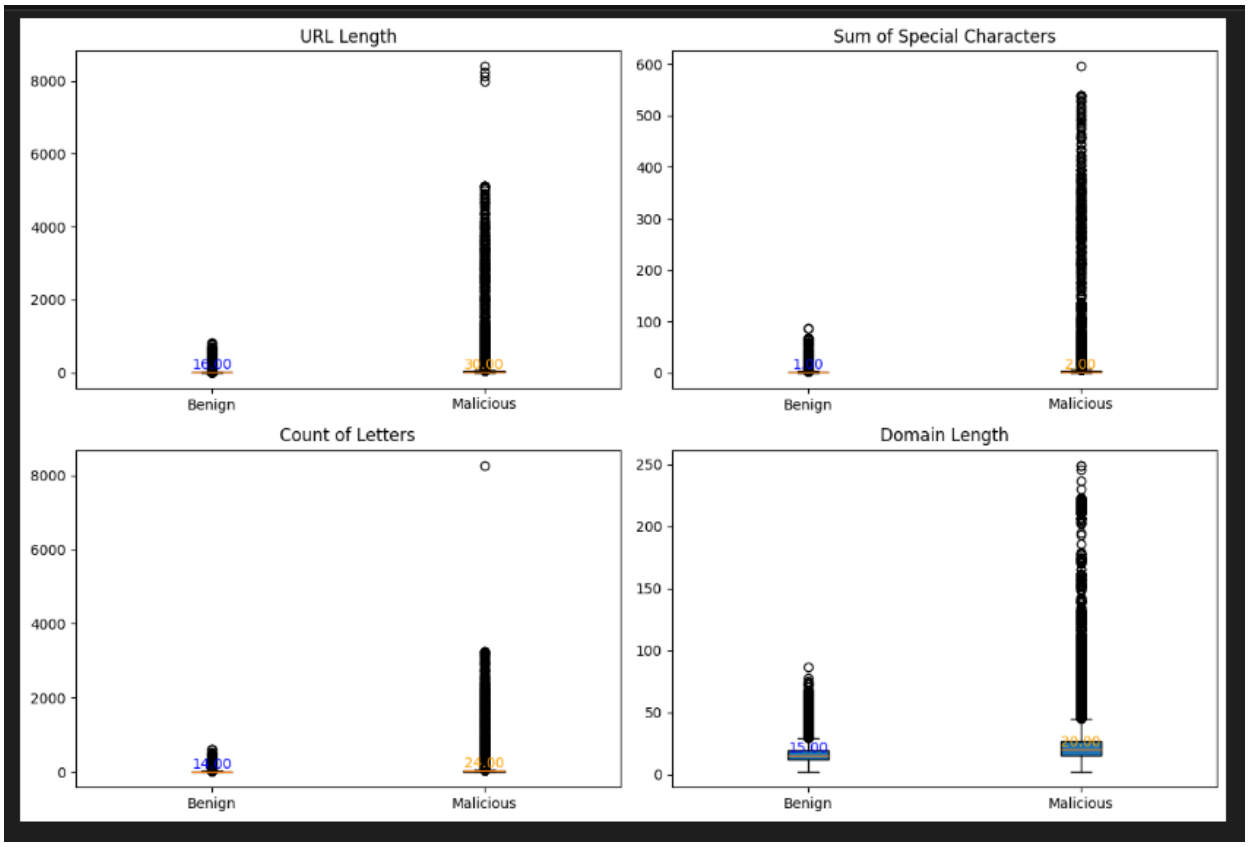
i. The dataset is loaded by concatenating two CSV files: train_dataset.csv and test_dataset.csv.

ii. Columns from index 2 to 60 are dropped from the dataset.

b. Exploratory Data Analysis(EDA)

i. Summary statistics are calculated for various numerical features such as URL length, count of special characters, count of letters, and domain length, both for benign and malicious URLs.

ii. Box plots are generated to visualize the distribution of these features between benign and malicious URLs.



iii. A pair plot is created to visualize relationships between pairs of features, with different colors representing benign and malicious URLs.

c. Feature Engineering

i. Various features are engineered from the URL data:

ii. url_len: Length of the URL.

iii. domain: Top-level domain extracted from the URL.

iv. abnormal_url: Binary feature indicating whether the hostname appears in the URL.

v. use_of_ip_address: Binary feature indicating whether an IP address is present in the URL.

vi. Counts of special characters and specific features like '@', '?', '-', etc., in the URL.

vii. HTTPS: Binary feature indicating whether the URL uses HTTPS protocol.

viii. Counts of digits and letters in the URL.

ix. Shortining_Service: Binary feature indicating whether the URL is from a URL shortening service.
x. google_index: Binary feature indicating whether the URL is indexed by Google.

3. Model Building and Evaluation

a. Model Selection
   i. Five machine learning algorithms were considered for malicious URL detection: Logistic Regression, Random Forest, K-Nearest Neighbors (KNN), Naive Bayes, and Decision Tree.
   ii. These algorithms were chosen based on their suitability for binary classification tasks, interpretability, and computational efficiency.

b. Model Training and Evaluation
   i. Logistic Regression: Achieved an accuracy of **82.84%.**
   ii. K-Nearest Neighbors (KNN): Achieved an accuracy of **86.79%.**
   iii. Naive Bayes: Achieved an accuracy of **82.71%**.
   iv. Decision Tree: Achieved an accuracy of **87.67%.**
   v. Random Forest: Achieved the highest accuracy of **87.82%.**

c. Model Selection:

   Random Forest was selected based on its highest accuracy as well as its nature of detecting and dropping the validity of Outliers. Random Forest generalizes well to the majority of the data, rather than fitting exactly to each data point with reduces our need to downsample our dataset due to its significantly higher ratio of Malicious URL compared to Benign URLS.

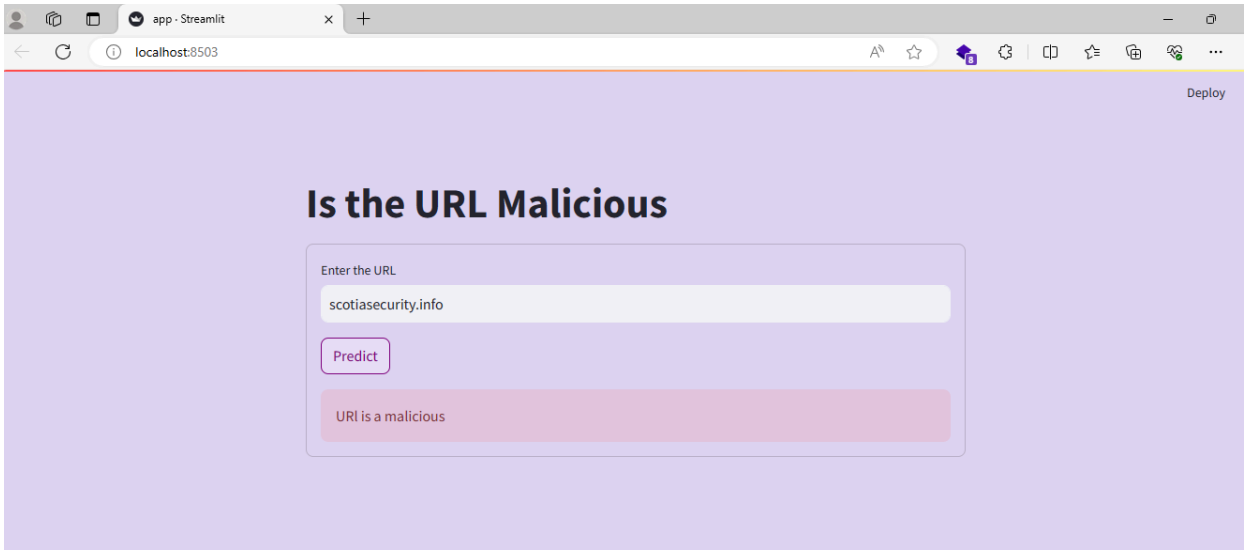
4. Data Insights

a. Accuracy of the Model: Currently we found significant high accuracy across the board for our models.

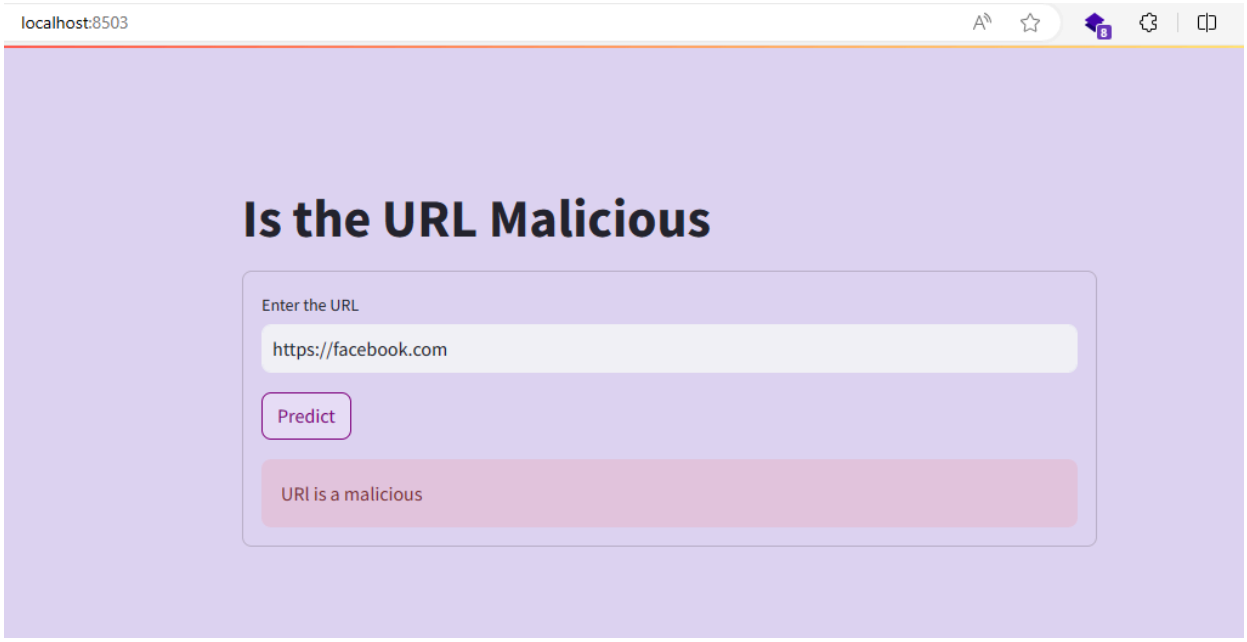
```
#Comparing accuracy
data = {
    'Model': ['LogisticRegression', 'KNeighborsClassifier', 'GaussianNB', 'DecisionTreeClassifier', 'RandomForestClassifier'],
    'Accuracy': [accuracy1, accuracy2, accuracy3, accuracy4, accuracy5]
}

accuracy_table = pd.DataFrame(data)
print(accuracy_table)
```

```
                 Model  Accuracy
0   LogisticRegression  0.828396
1   KNeighborsClassifier  0.867873
2           GaussianNB  0.827094
3   DecisionTreeClassifier  0.876680
4   RandomForestClassifier  0.878206
```

b. Screenshots of model predictive URLS as Malicious or Not: This is the current state of the deployed model in terms of predictions for whether a URL is malicious or not.



c. Fault in terms of lack of HTTPS and HTTP detection in URL: Currently, the model's predictive model detects any URL with HTTPS inherently in it as malicious regardless of the actual result.



d. Low number of Non Malicious URLs used for dataset results in Model Bias: We predict that the significantly higher count of Malicious URL for the training dataset has resulted in significant model bias. In terms of countering this, we're considering multiple solutions such as combining the training dataset with a whitelist URL CSV full of benign URLs.