A PROJECT REPORT ON

Data Labeling of URL Using Active Learning

Thesis to be submitted in partial fulfillment of the requirements

for the Machine learning (CS-584)

Submitted by :

Shivani Shrivastav (A20553589)

Ankan Mazumdar (A20541357)

Group No: 36

Under the Guidance of Professor

Binghui Wang

ILLINOIS TECH

ILLINOIS INSTITUTE OF TECHNOLOGY

Declaration of Authorship

We solemnly affirm that the thesis entitled "**Data Labeling of URL using Active Learning**" is a genuine work undertaken by the undersigned. This document has been developed in compliance with the established academic norms and principles of ethical conduct. Furthermore, in adherence to these standards, we ensure that all sources and materials not original to this work have been appropriately acknowledged and cited.

Name: **Shivani Shrivastav (A20553589)**

Ankan Mazumdar (A20541357)

Thesis Title: Data Labeling of URL Using Active Learning

ACKNOWLEDGEMENT

Every project big or small is successful largely due to the effort of a number of wonderful people who have always given their valuable advice or lent a helping hand. I sincerely appreciate the inspiration; support and guidance of all those people who have been instrumental in making this project a success. We, Ankan Mazumdar & Shivani Shrivastav, are extremely grateful to our professor for the confidence bestowed in us and entrusting my project entitled **“Data Labeling of URL using Active Learning**” with special reference.

At this juncture, I express my sincere thanks to Professor: Binghui Wang of the Computer Science Department for making the resources available at the right time and providing valuable insights leading to the successful completion of our project who even assisted us in completing the project.

In our project, Shivani has played a vital role, handling the drafting the abstract, conducting a thorough literature review, clearly defining the problem statement.She has also contributed significantly to the exploratory data analysis and initial preprocessing, played a key role in the model building and training.

Similarly, Ankan has been instrumental in our project. He worked on detailing the problem statement , and methodology and approaches. He has done aso the data visualization part actively participated in the EDA and preprocessing finalizing the top feature selection. Ankan has also managed the project's requirements and contributed significantly to the model building and testing. His collaboration in user interface development.

Both of us will take joint responsibility for writing the final report, ensuring that our research findings are effectively and clearly communicated.

Together, our combined efforts have significantly enhanced the depth and breadth of our research.

Contents

Abstract:

Data labeling can be a very time- and/or money-consuming procedure. For this, a domain specialist is occasionally required. Active Learning is an approach that uses fewer training data to achieve better optimization by iteratively training a model. We can use a classification model to perform the majority of the labeling using active learning, requiring us to label samples only when absolutely necessary. Active learning seeks to address this issue by asking to annotate only the most informative data from the unlabeled set.If given the freedom to

choose which data to label, the active learning methodology has the potential to significantly reduce the amount of labeled data needed to train a model with improved accuracy. The data that the model is most unsure of is prioritized through active learning, and labels are only requested for those. As a result, the model picks up new information more quickly.

This is a very important problem ,the reason being, in real world situations, we might want to go ahead to obtain manual labels for new data, but we have constraints on how much labeled data we can actually obtain within a certain time & limited budget.

In this project we will use active learning techniques to smartly label large sets of text data. This would help us reduce the manual effort in data labeling by automating it as much as possible.
Active learning is a supervised learning technique that involves an iterative process of selecting the most informative data points for the learner to label. The learner actively queries the user/domain expert for labels on these selected data points, focusing its learning efforts on the most valuable information. This approach is particularly useful when labeling data is expensive or time-consuming.Reducing labeling effort, improving model performance.

Literature Review:

Managing and streamlining a data annotation process is difficult. Businesses face a number of internal and external challenges that make the work of annotating documents inefficient and ineffective. Therefore, the only way to overcome these difficulties is to get to the root of them, comprehend them, and then resolve them appropriately. Let's begin.

Having trouble managing a large workforce-

Data-hungry ML and AI models need a huge amount of labeled data to learn. Businesses employ a sizable staff to manually tag datasets in order to provide the vast amount of labeled data needed to feed the computer models. Additionally, processing Big Data and labeling them with the best quality is crucial to achieving a high level of accuracy.

A lack of access to innovative tools and technologies-

Having a large and skilled workforce does not guarantee the production of high-quality tagged datasets. To carry out the accurate data annotation process, the right equipment and technology are needed. Different technologies and methods are used to tag datasets for deep learning depending on the type of data. It is therefore crucial to deploy the right technology that guarantees the greatest quality at a reasonable price.But businesses frequently fall short in creating the infrastructure necessary for the greatest possible data annotation. Because of the high cost of the tools and a lack of professional process expertise, organizations struggle to choose the best technology to implement.

Lack of reliable and high-quality data tagging-

A precise data annotation model necessitates the highest caliber dataset tagging. There is no room for error at all. Small mistakes can have a major impact on the company's bottom line. The ML model will pick up on incorrect information from your data samples in the same way. As a result, AI won't be able to correctly forecast it or recognise it.Furthermore, obtaining consistent high-quality data is the true challenge; assuring it is not the primary criterion. It is essential for organizations to keep a steady flow of high-quality, labeled datasets for machine learning (ML) training and accurate AI prediction.

Not an inexpensive project-

The process of labeling data for annotation is time-consuming. As a result, businesses find it difficult to determine how much money they need to create an ML & AI training project.

Businesses occasionally have to move backwards as a result of paying a large workforce a significant wage for a longer length of time and investing in expensive technologies.

Additionally, setting up a sizable, ergonomic office space with all the required amenities is difficult for businesses to handle.

Failure to abide by data security rules-

Because of a severe lack of process understanding, data annotation organizations struggle to adhere to international data security standards. As Big Data becomes more and more widespread, the standards governing data privacy compliance are becoming stricter.When it comes to raw data, it contains incredibly private information such as reading texts, identifying faces, etc. Therefore, tagging misinformation or any little mistakes can have huge repercussions. Besides, the data leak is the most important factor to be addressed here.

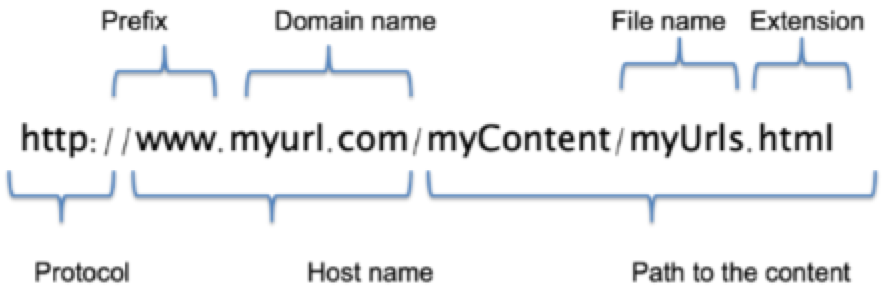
Hence, data labeling companies are sometimes failing to comply with these compliances with privacy and internal data security standards.

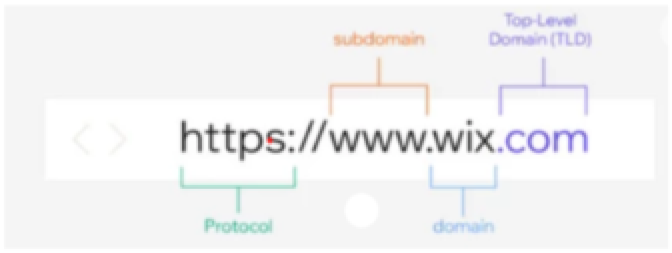
Problem Statement:

In the rapidly evolving landscape of machine learning applications for businesses, the crucial bottleneck lies in the accurate and efficient annotation of vast datasets. As organizations strive to develop robust machine learning algorithms to enhance decision-making processes, they encounter challenges in managing annotation projects. These challenges encompass diverse aspects such as the need for specialized tools, user interface (UI) design considerations, the integration of annotation processes into existing workflows, and the inevitable turnover of annotation personnel.

The problem at hand is to establish a seamless and scalable data annotation framework that aligns with business goals, accommodates various project requirements, integrates smoothly with machine learning algorithms, provides an intuitive user interface for annotators, employs efficient annotation tools, and addresses the issues arising from workforce turnover. Balancing the intricacies of these components is essential to ensure the consistent generation of high-quality annotated data sets, which in turn is pivotal for the success of machine learning initiatives within the organization.

Parts of URL :





Pandas, Numpy, sklearn, LIME etc libraries will be used to process the data.

Approach-Methodology :

A.Theoretical Approach:

1.Label a small portion of data. Lets say Labeled data as Dl & unlabeled data as Du where Du>> Dl means Du is significantly larger than Dl.

2.Lets train the initial model M0 with Dl & use M0 to predict class labels.

3. From the unlabeled dataset Du,we‘ll smartly pick /sample a small subset of data s1 which could be hardly 1% of Du & obtain labels for this s1. When we smartly sample data, it means our target is to enhance the performance of our model M0.

4.we’ll take our existing model M0 which is trained on previously labeled data Dl and we’ll add this newly labeled data s1, and we can either retrain the whole model from scratch or can retrain model M0 to obtain M1.

5.Repeat step 3 & step 4 until we obtain all labels correctly.

6.The stopping criteria would be when there is no significant improvement in model M1’s performance compared to M0.

You would have to determine the type of scenario your would like to use (that is, Membership Query Synthesis, Stream-Based Selective Sampling or Pool-Based sampling)

Query strategies :

Algorithms for determining which data points should be labeled can be organized into a number of different categories, based upon their purpose-

Expected error reduction: label those points that would most reduce the model's generalization error.

Uncertainty sampling: Predict class probabilities for all unlabeled data using the trained classifier. If a classification has a probability higher than a predetermined threshold, it is assumed to be accurate. Include these samples in the training set along with their expected classes. label those points for which the current model is least certain as to what the correct output should be.

When a Supervised Machine Learning model makes a prediction, it often gives a confidence in that prediction. If the model is uncertain (low confidence), then human feedback can help.

Getting human feedback when a model is uncertain is a type of Active Learning known as Uncertainty Sampling.

Among types of Uncertainty Sampling, we can use-

Least Confidence: difference between the most confident prediction and 100% confidence

OR

Entropy: difference between all predictions, as defined by information theory

B.Empirical/Experimental approach :

We have not yet finalized on the approach of the experiment. Most probably, the sampling strategy we will use is a combination of uncertainty sampling and pool-based sampling.

An activation layer is used to pass through scores from the previous layer. The score is

converted into probability values by the activation layer. The data is finally classified to the class with the highest probability value.Activation function could be Softmax or sigmoid activation function in the final layer, but we are yet to finalize it.

Each class's final score should be independent from the others. Most likely, sigmoid should be used because, unlike softmax, which converts each score of the final node between 0 and 1 independently of the previous scores, Sigmoid converts each score between 0 and 1 probabilities considering other’s score.

For Productionisation, We would use separate systems for model training(compute cluster) & serving web application/web API.There would be data store (which is also storing feedback that users are giving , needs to be run periodically in regular intervals) and the model get retrained (either with entire combined dataset or only with recent data or only those incorrectly labeled ones corrected by the user as per the business needs)

Retrained model is pushed to the Model registry/repository with its version.It's redeployed to the API.Use of pre-trained weights from the previous layer/model to new layer for new dataset and so on, & updating the model as new data arrives.

The majority of ML algorithms demand that any input or output variables be numbers. Any categorical data must therefore be converted to integers.

Empirical/Experimental approach:

1. Dataset Processing:

We extract the columns from the dataset and analyze the most important features.

2. Dataset Partitioning:

We will start with a labeled dataset and partition it into three sets: training, testing, and validation. This division helps evaluate model performance on unseen data and fine-tune it iteratively.

3. Baseline Model Training:

We will train baseline models using the training dataset. These models serve as the initial models to make predictions.

4. Initial Prediction:

We will use the trained models to make predictions on the test dataset. We will evaluate the accuracy of the predictions.

5. Accuracy Threshold Check:

We will check if the accuracy of the predictions meets a predefined threshold. If the accuracy is below the threshold, it indicates that the model needs improvement.

6. Label Correction:

We will correct the labels of the misclassified instances in the test dataset. This step involves human intervention to correct the ground truth labels based on the model's predictions.

7. Data Augmentation:

We will use the corrected test dataset as additional training data. This is a form of data augmentation where we iteratively incorporate new labeled data into the training set.

8. Retraining:

We will retrain the models using the augmented training dataset. This step helps the models adapt to the corrected labels and potentially improve their performance.

9.Iterative Process:

We will repeat the process iteratively: make predictions, check accuracy, correct labels, augment data, and retrain models. This iterative approach allows the models to learn from the corrected labels and improve over successive iterations.

Data Description:

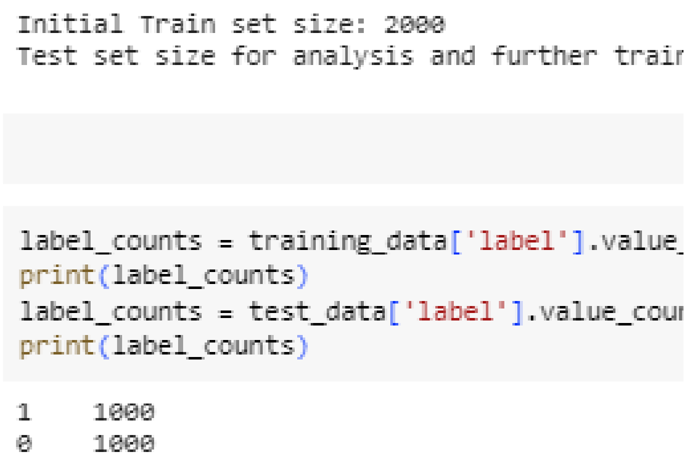
Link- https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls?resource=download

The dataset was acquired from various sources such as PhisTank etc.

Pandas,Numpy,sklearn etc libraries will be used to process the data.

Labeled dataset is chosen in order to pass labels to the data whenever is required in order to help the model to enhance performance & to estimate the accuracy of the predicted labels by the model.

URLdata open source dataset will be used. It has 450176 URL website names with 2 classes (**Benign , Malicious)**, with 345738 URL records as benign classand 104438 as malicious.Initially, only 2000 records are to be picked as training data for baseline model & then we will evaluate the accuracy and will keep on passing further data for self training.Labeled dataset is chosen in order to pass labels to the data whenever is required in order to help the model to enhance performance & to estimate the accuracy of the predicted labels by the model.



```
Initial Train set size: 2000
Test set size for analysis and further trair
```

```
label_counts = training_data['label'].value_
print(label_counts)
label_counts = test_data['label'].value_cour
print(label_counts)
```

```
1    1000
0    1000
```

We are initially splitting the data into training and testing sets, balancing the number of records from each class in the training set by picking just 1000 records from each class means 2000 URL records for first time training.and then the training and test data sets are further shuffled for randomness. This process is done to ensure a representative distribution of classes in both the training and testing sets, and shuffling helps prevent any bias that might arise from the original order of the data.

Initially, we will do feature engineering to obtain data records with features.

The following features will be extracted from the URL for classification.

Length Features:

- ☐ Length Of Url
- ☐ Length of Hostname
- ☐ Length Of Path

☐ Length Of First Directory
☐ Length Of Top Level Domain
☐

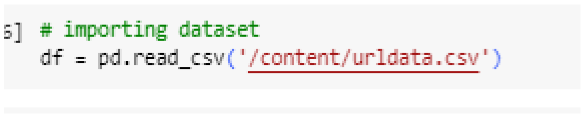
Count Features:

☐ Count Of '-'
☐ Count Of '@'
☐ Count Of '?'
☐ Count Of '%'
☐ Count Of '.'
☐ Count Of '='
☐ Count Of 'http'
☐ Count of 'https'
☐ Count Of 'www'
☐ Count Of Digits
☐ Count Of Letters
☐ Count Of Number Of Directories
☐
☐

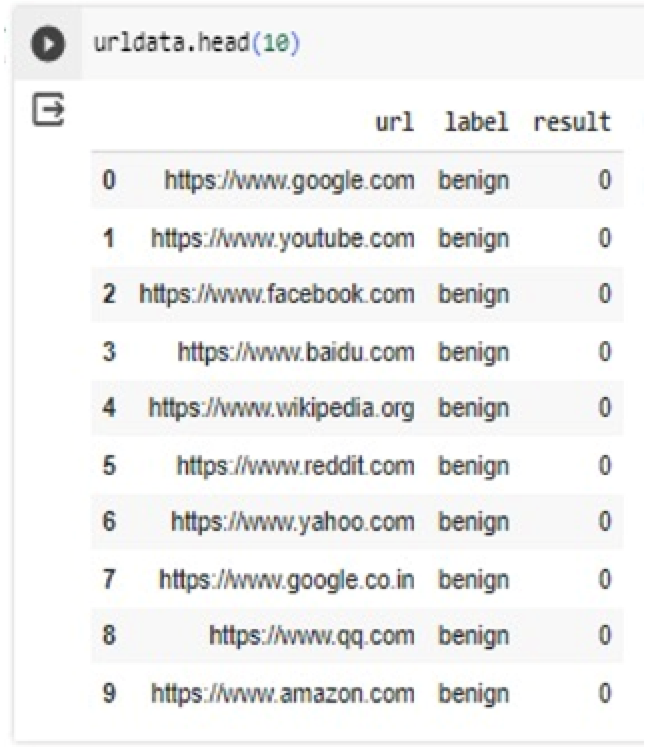
Binary Features

☐ ‘use_of_IP’
☐ ‘use_of_shortening_url_service’

Loading the dataset: Using Pandas we can load the dataset it has 3 columns and no null values-

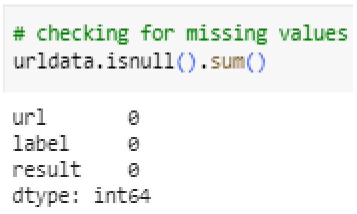


```
5] # importing dataset
   df = pd.read_csv('/content/urldata.csv')
```



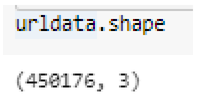
```
urldata.head(10)
```

	url	label	result
0	https://www.google.com	benign	0
1	https://www.youtube.com	benign	0
2	https://www.facebook.com	benign	0
3	https://www.baidu.com	benign	0
4	https://www.wikipedia.org	benign	0
5	https://www.reddit.com	benign	0
6	https://www.yahoo.com	benign	0
7	https://www.google.co.in	benign	0
8	https://www.qq.com	benign	0
9	https://www.amazon.com	benign	0



```
# checking for missing values
urldata.isnull().sum()
```

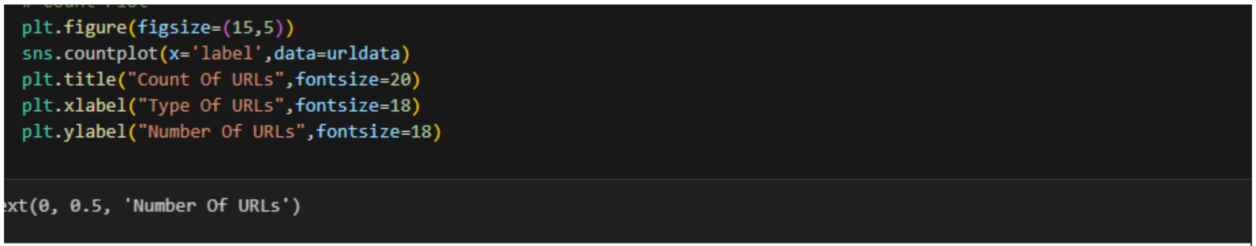
```
url       0
label     0
result    0
dtype: int64
```



```
urldata.shape
```

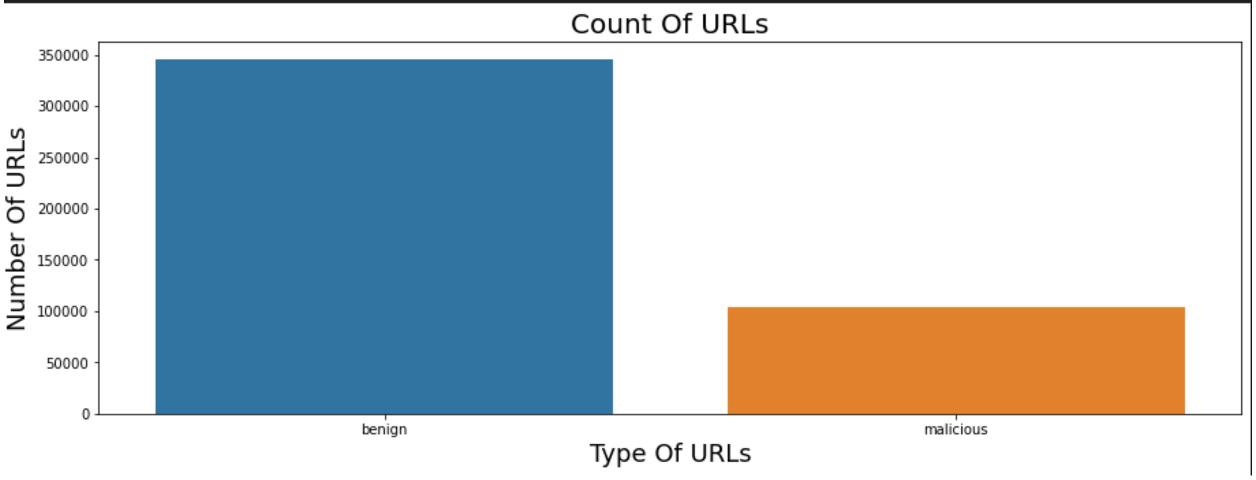
```
(450176, 3)
```

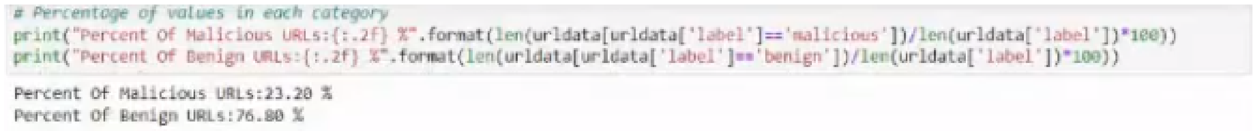
Imbalanced dataset with 76% as benign and 24% as malicious.



```
plt.figure(figsize=(15,5))
sns.countplot(x='label',data=urldata)
plt.title("Count Of URLs",fontsize=20)
plt.xlabel("Type Of URLs",fontsize=18)
plt.ylabel("Number Of URLs",fontsize=18)
```

```
xt(0, 0.5, 'Number Of URLs')
```





```
# Percentage of values in each category
print("Percent Of Malicious URLs:{:.2f} %".format(len(urldata[urldata['label']=='malicious'])/len(urldata['label'])*100))
print("Percent Of Benign URLs:{:.2f} %".format(len(urldata[urldata['label']=='benign'])/len(urldata['label'])*100))
```

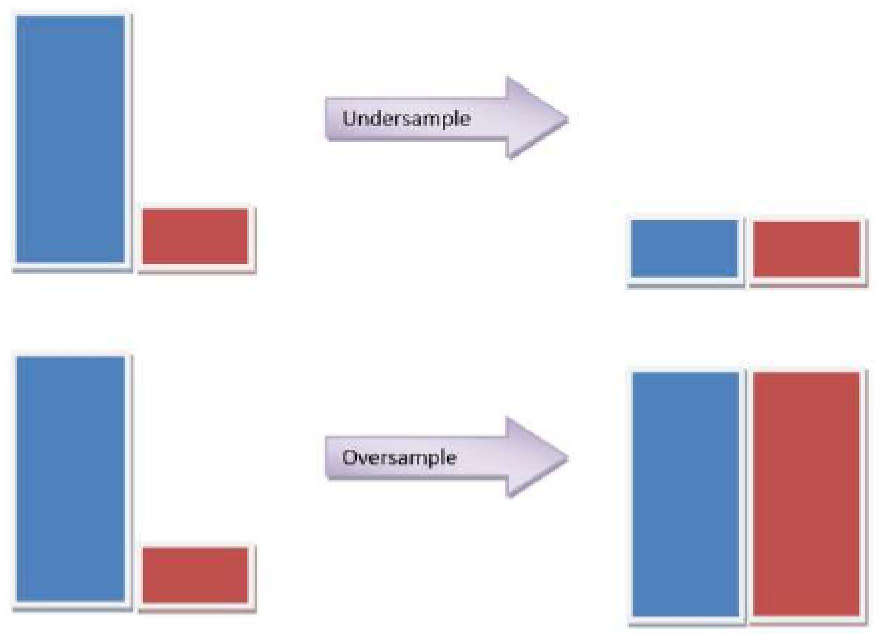
```
Percent Of Malicious URLs:23.20 %
Percent Of Benign URLs:76.80 %
```

Approaches to deal with the imbalance dataset problem-

The dataset was found to be imbalanced with 77% of data labeled as “benign” and 23% of data labeled as “malicious”. Most machine learning algorithms work best when the number of instances of each class is roughly equal.

a. **Resampling:**

This technique is used to upsample or downsample the minority or majority class. When we are using an imbalanced dataset, we can oversample the minority class using replacement. This technique is called oversampling. Similarly, we can randomly delete rows from the majority class to match them with the minority class which is called undersampling. After sampling the data we can get a balanced dataset for both majority and minority classes. So, when both classes have a similar number of records present in the dataset, we can assume that the classifier will give equal importance to both classes.



b. SMOTE (Synthetic Minority Oversampling Technique):

Synthetic Minority Oversampling Technique or SMOTE is another technique to oversample the minority class. Simply adding duplicate records of minority classes often doesn't add any new information to the model. In SMOTE new instances are synthesized from the existing data. If we explain it in simple words, SMOTE looks into minority class instances and uses k nearest neighbor to select a random nearest neighbor, and a synthetic instance is created randomly in feature space. This method is being used in the project. We have performed oversampling on the minority class with SMOTE using the imblearn library.

c. Breaking the dataset into small chunks

We will break the dataset into small datasets having few data points. Once we have these small chunks then we can train the model on one chunk and make predictions on the other chunk after which we can correct the incorrect predictions and retrain the model on the correct dataset. This process will be repeated iteratively until the entire dataset is labeled. So we have followed steps similar to this approach-

1. Separating data by class: We first separate the data into two classes based on the "label" column. As it is a binary classification task where data points belong to Class 0 or Class 1.

2. Creating training and testing sets: For each class, we took 1000 data points for training and kept the rest for testing. This creates a balanced training set with 50% data from each class.

3. Shuffling the data: Then both the training and testing datasets are shuffled to ensure no specific order or pattern biases the model.

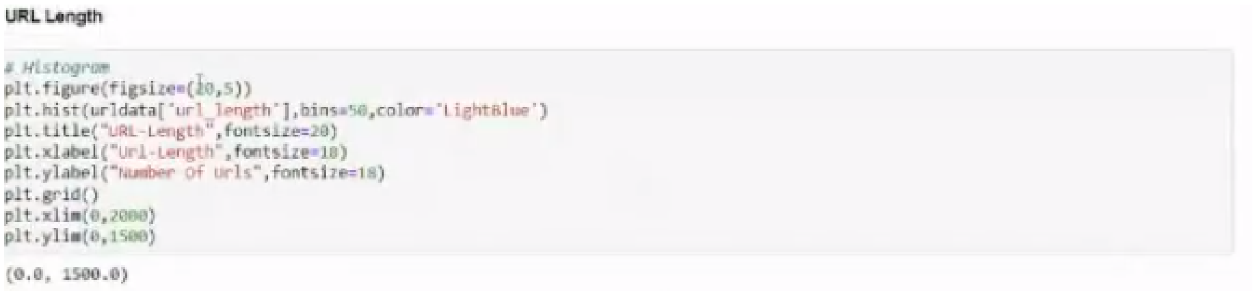
4. Creating a balanced set for further training: Additionally, we concatenate all data points and shuffle them again. This creates a larger dataset containing 10,000 data points from each class for further training and experimentation.

```python
from sklearn.model_selection import train_test_split

# Separate data into Class 0 and Class 1
class_0_data = urldata[urldata['label'] == 0]
class_1_data = urldata[urldata['label'] == 1]
train_class_0 = class_0_data.head(1000)
train_class_1 = class_1_data.head(1000)

# The remaining records will be used for testing
test_class_0 = class_0_data.tail(len(class_0_data) - 1000)
test_class_1 = class_1_data.tail(len(class_1_data) - 1000)

# print("class 0 set size:", len(class_0_data))
# print("class 1 set size:", len(class_1_data))

# Concatenate the training and testing sets for both classes
training_data = pd.concat([train_class_0, train_class_1])
test_data = pd.concat([test_class_0, test_class_1])

# Shuffle the data to mix both classes
training_data = training_data.sample(frac=1, random_state=42).reset_index(drop=True)
test_data = test_data.sample(frac=1, random_state=42).reset_index(drop=True)

# Now, 'train_data' contains 50% of data from both classes for training
# 'test_data' contains the remaining data for testing later
print("Initial Train set size:", len(training_data))
print("Test set size for analysis and further training:", len(test_data))

# Concatenate the training and testing sets for both classes
train_data = pd.concat([train_class_0, train_class_1])
test_data = pd.concat([test_class_0, test_class_1])

# Shuffle the data to mix both classes
train_data = train_data.sample(frac=1, random_state=42).reset_index(drop=True)
test_data = test_data.sample(frac=1, random_state=42).reset_index(drop=True)
```

```
Initial Train set size: 2000
Test set size for analysis and further training: 448176
```

EDA and Initial Preprocessing:

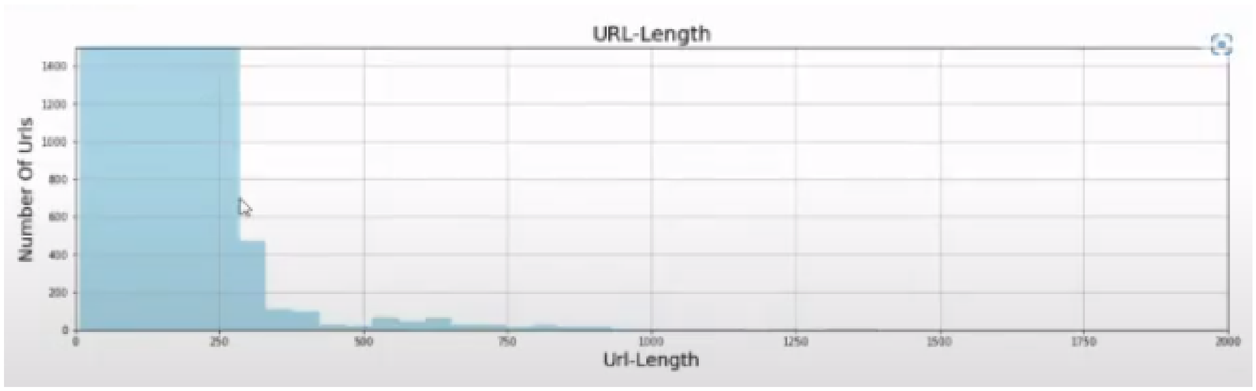
Length Features of URL-

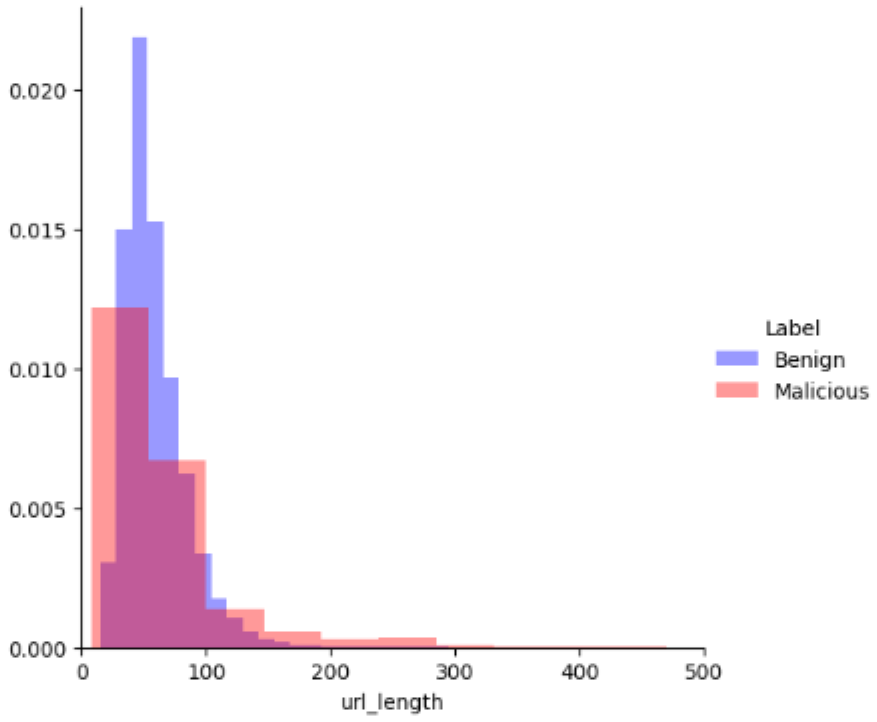


URL Length

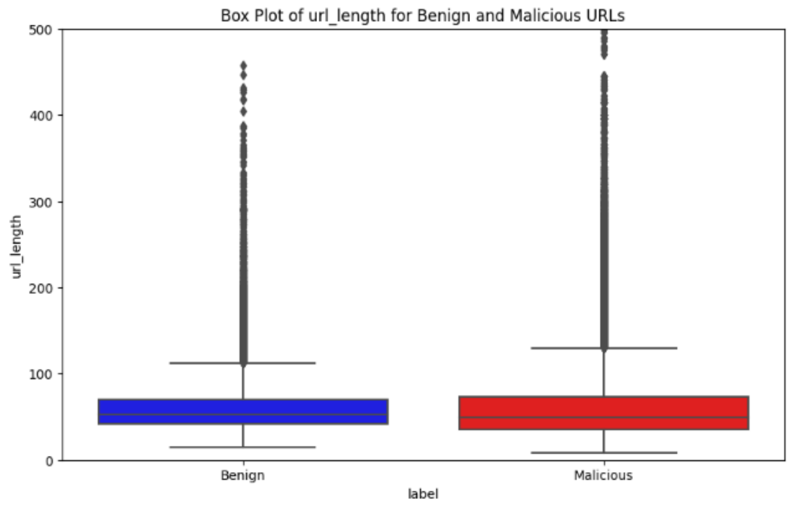
```
# Histogram
plt.figure(figsize=(20,5))
plt.hist(urldata['url_length'],bins=50,color='LightBlue')
plt.title("URL-Length",fontsize=20)
plt.xlabel("Url-Length",fontsize=18)
plt.ylabel("Number Of Urls",fontsize=18)
plt.grid()
plt.xlim(0,2000)
plt.ylim(0,1500)
```

```
(0.0, 1500.0)
```





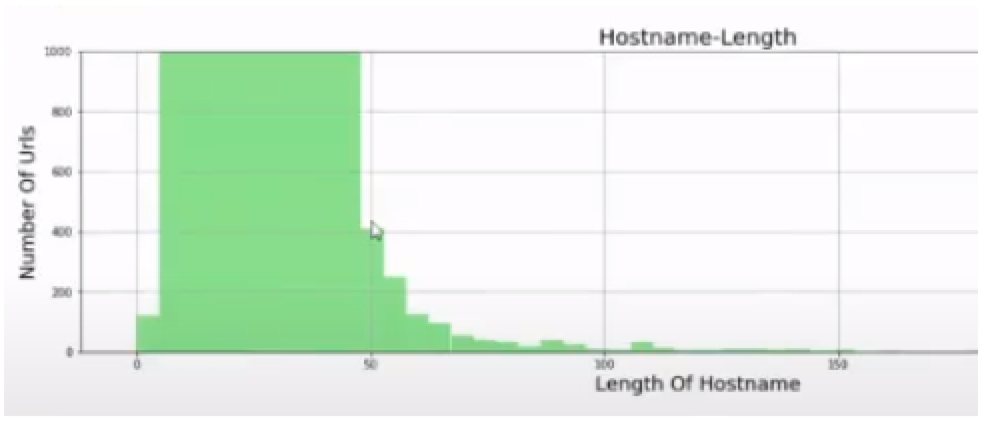
Observation- URL Lengths are mostly between 0 to 300 and categories are overlapping.



Insights:

- Malware URLs tend to be longer than benign URLs.
- Malicious URLs exhibit greater variance in url_length compared to benign URLs.
- Outliers exist in both distributions, emphasizing the need for additional features for accurate predictions.
- A url_length of 50 characters can serve as a rough threshold, but exceptions exist.

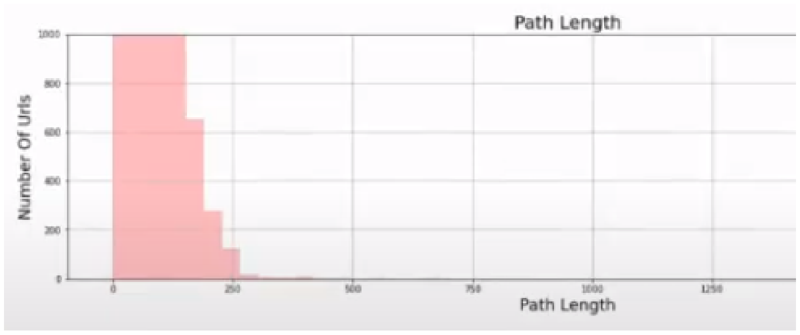
Hostnames:

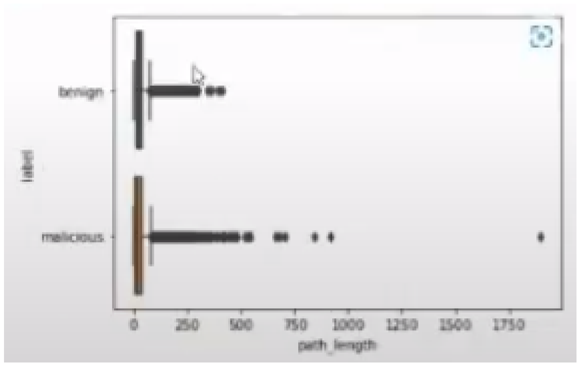


Observation - Most of the URLs has length between 0 to 50 characters.

There is an overlap between among the densities of Malicious and Benign URLs for hostname length lesser than 50 characters

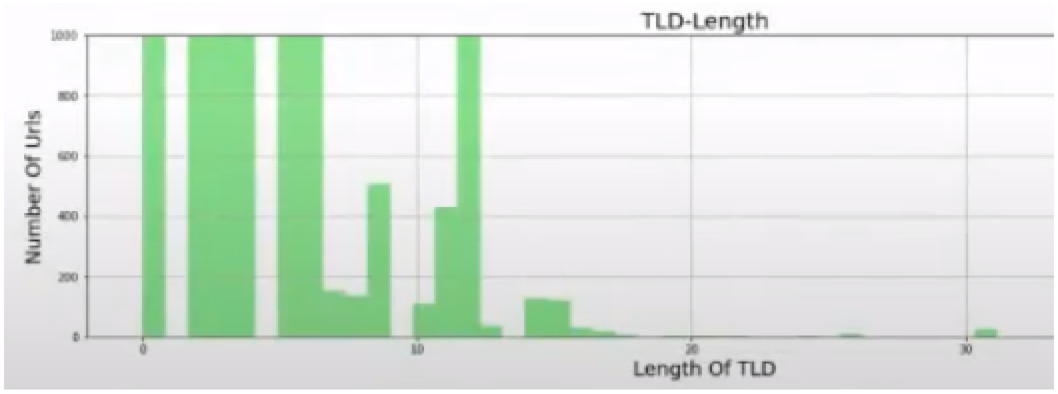
Path Length-

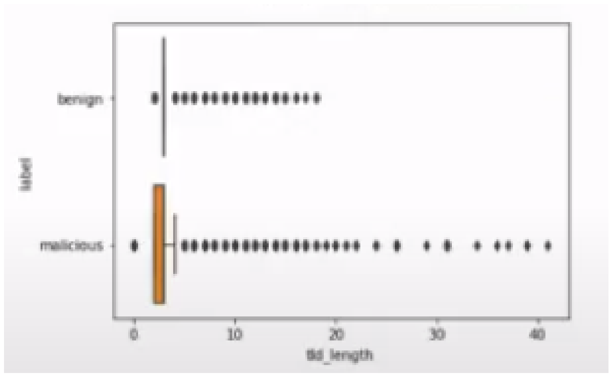




Observation - Path lengths are in the range 0 to 250

TLD-

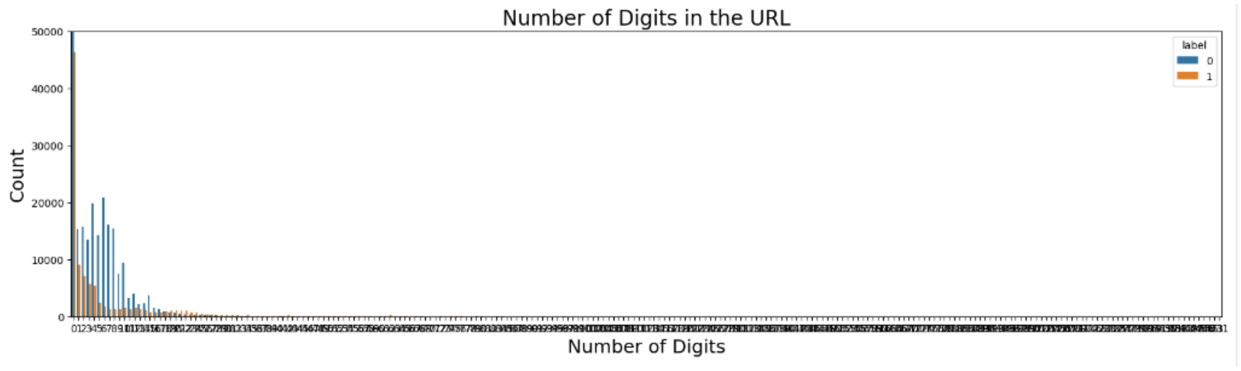




Observations- With the help of the TLD feature , the classes are separated better than the other features.

Count features:

Number of digits in URL:

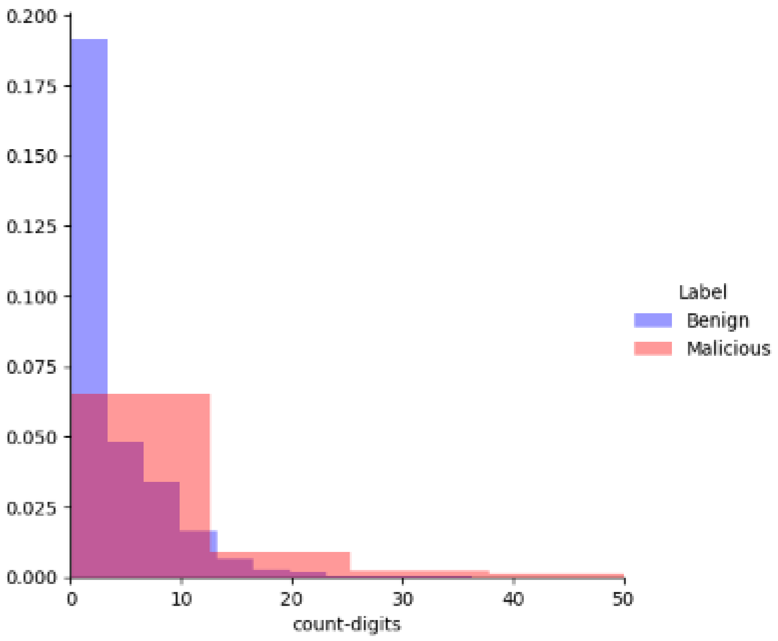


Observation-

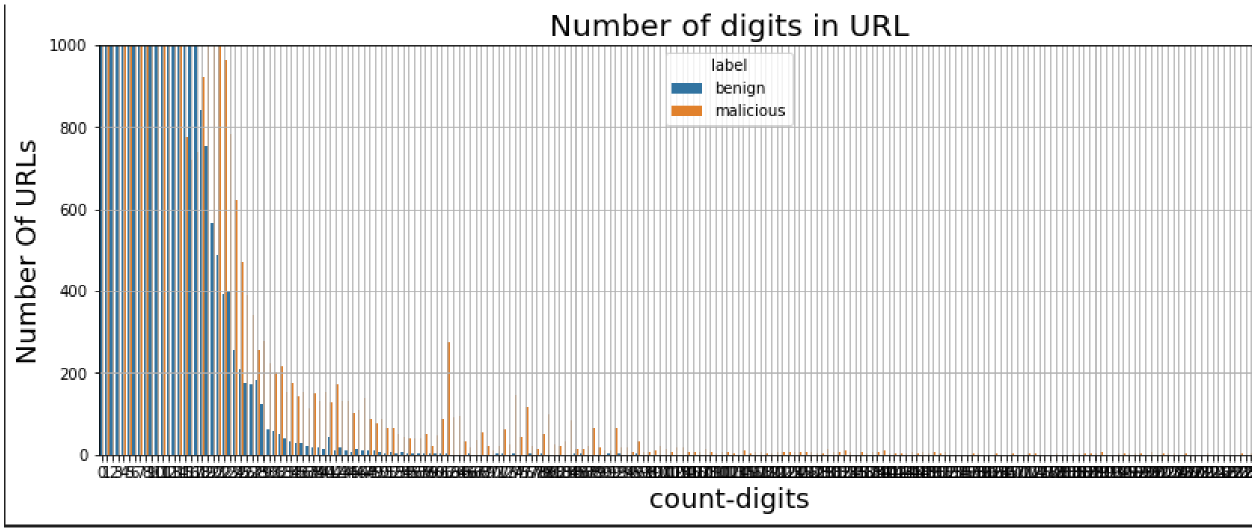
- Number of digits is mostly less than 100 for both Malicious and Benign URLs.

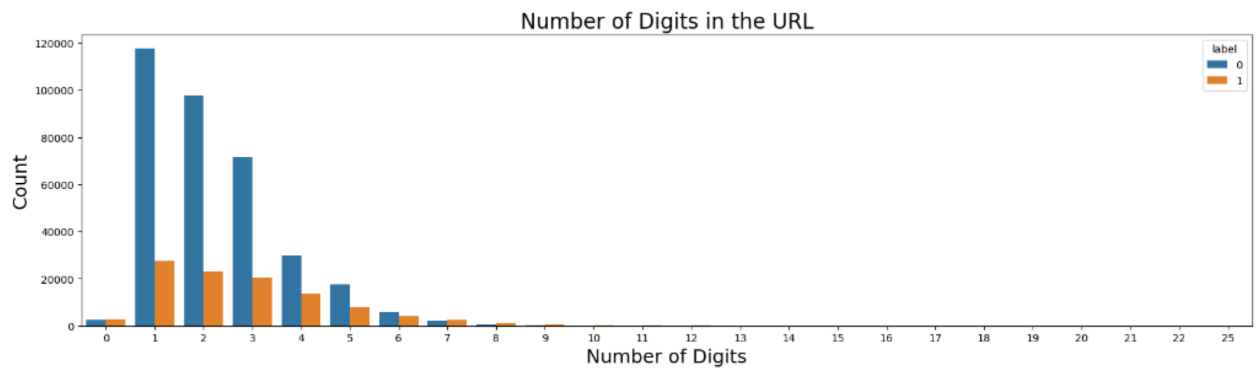
- Malicious URLs have more digits than Benign URLs.

Count of Digits-

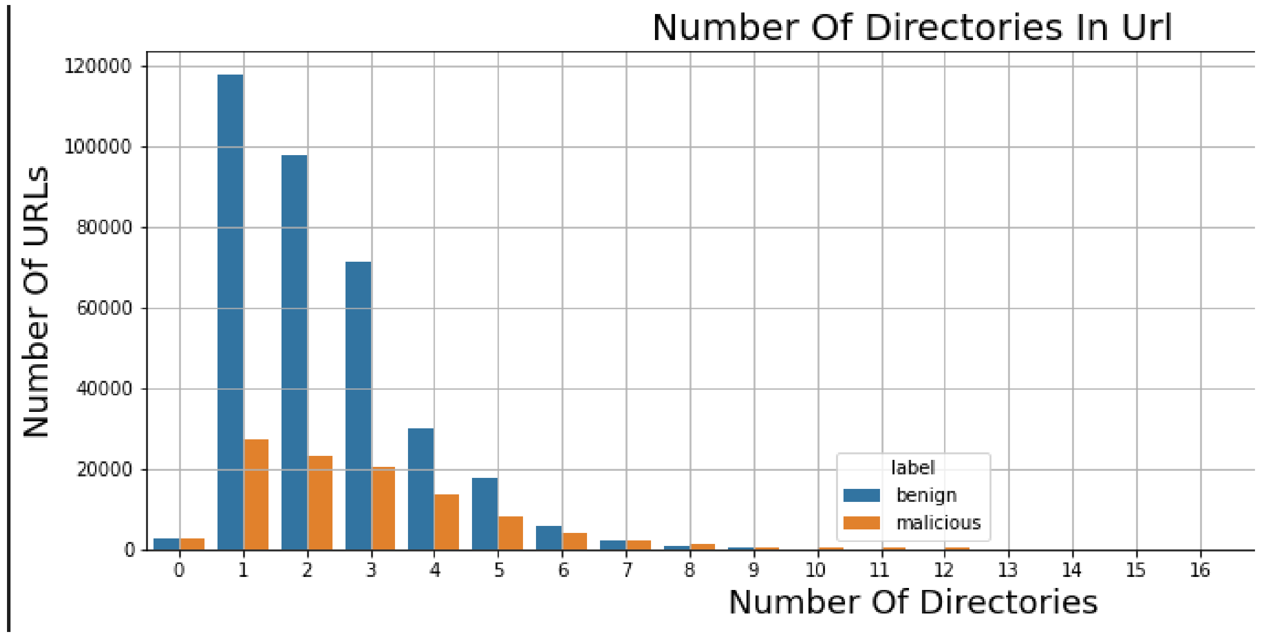


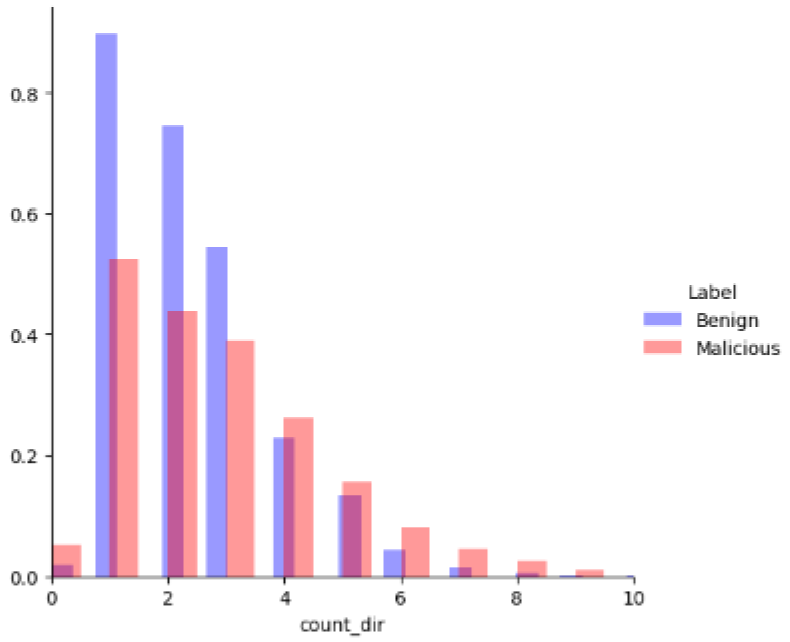
Number of Digits in URL-





Count of directories:



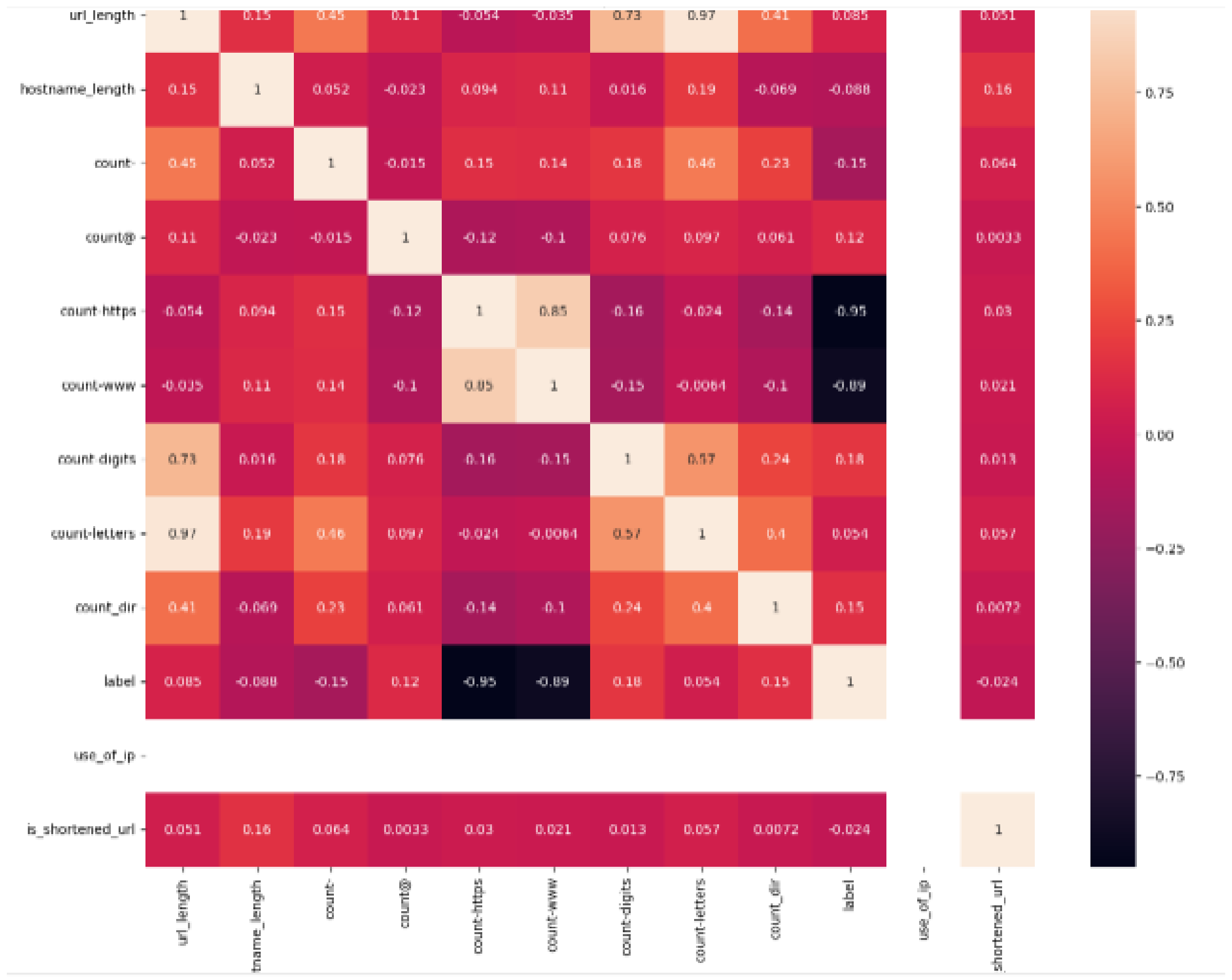


Observations-

- There is an overlap among both the categories.
- The number of directories is mostly less than 10 for both Malicious and Benign URLs.

Multivariate Analysis :

We initially thought of plotting the pair plot , but that wouldn’t be visually effective and wouldn't be able to make much sense of it. Hence, we plotted the heat map over here-



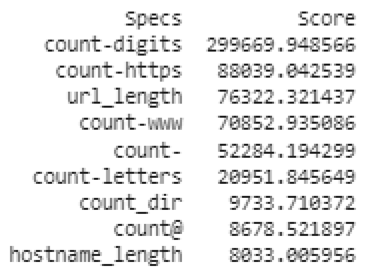
We found that our results Count_https and count_www are heavily related to the result.

Selecting Top features for model Training using slearn SelectKBest, Chi-Squared and RFE:

Chi-Squared Test:

The chi-squared test is a statistical test that measures the association between two categorical variables. In this case, one variable is the feature (e.g., URL length) and the other variable is the target variable (malicious or not). The chi-squared test is a good choice for feature selection because it is simple to calculate and we can handle a large number of features.

Here is the result we get after applying these :



```
          Specs          Score
   count-digits  299669.948566
    count-https   88039.042539
     url_length   76322.321437
      count-www   70852.935086
         count-   52284.194299
  count-letters   20951.845649
      count_dir    9733.710372
         count@    8678.521897
hostname_length    8033.005956
```

The second column, labeled "Score", represents the chi-squared test statistic for each feature. This statistic measures the strength of the relationship between each feature and the target variable (whether the URL is malicious or not). Higher scores indicate a stronger relationship.

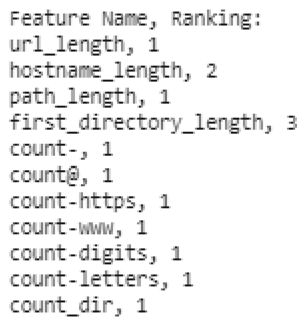
Count digits feature has the highest chi-squared score (299669.95), indicating a strong association with the target variable followed by count-https, url_length,...upto hostname_length.

Chi-squared test purpose:

The chi-squared test is a suitable choice for feature selection in this case because it is specifically designed to assess the association between categorical variables. In this scenario, the target variable (malicious or not) is categorical, and the features of interest (URL characteristics) are also categorical or can be treated as categorical for the purpose of feature selection. The chi-squared test effectively measures the strength of the relationship between these categorical variables, allowing us to identify features that are most indicative of the target variable.

RFE:

Alternatively, we have also used Recursive Feature Elimination (RFE), which is a feature selection technique which we iteratively remove the least important features until a desired number of features remains.We are getting similar results from RFE as well-



```
Feature Name, Ranking:
url_length, 1
hostname_length, 2
path_length, 1
first_directory_length, 3
count-, 1
count@, 1
count-https, 1
count-www, 1
count-digits, 1
count-letters, 1
count_dir, 1
```

Conclusion-

- Multiple URL features are created based on length, count and binary features.
- On visualization, it is found that count_https and count_www are heavily related to the result.
- The top 10 features are count-digits, count-https, url_length, count-www, count-, count-letters, count=, count_dir, count@ and hostname_length.

Key Performance metric (KPI) :

In the real world, there are many different types of metrics. We'll examine at a few of the mostly useful used KPIs that we might utilize in this multi class classification project-

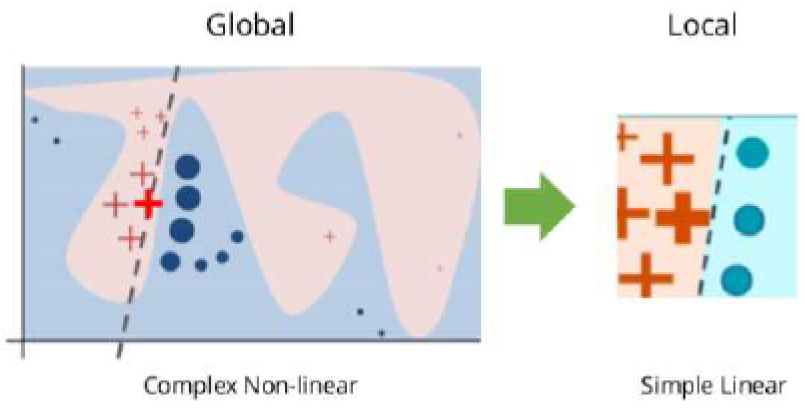
- Accuracy
- Confidence score

Interpretability:

Models are interpretable when humans can readily understand the reasoning behind predictions and decisions made by the model. The more interpretable the models are, the easier it is for someone to comprehend and trust the model. We have used LIME to showcase our model interpretability.

LIME (Local Interpretable Model-Agnostic Explanations)

LIME, the acronym for local interpretable model-agnostic explanations, is a technique that approximates any black box machine learning model with a local, interpretable model to explain each individual prediction. It is relatively faster than SHAP in many cases.



If the user wants to understand how the prediction is being made then they can click on expand and find out the LIME chart.

Requirements:

Functional Requirements:

Purpose

To build a system that will predict the URL Data labels correctly, basically an automated annotation system. The system will need to retrain the model redundantly until a certain threshold accuracy is reached.

Inputs

A CSV File containing a list of URLs is provided as the input to the system.

Output

The system interprets & predicts the URL Data labels as Malicious and Benign.

Usability

It reduces manual efforts, saves time, has higher accuracy, scalability, streamlined process, on-time delivery, and reduces cost.

Software Requirements

Notepad++ or any Code Editor

Anaconda Navigator, Jupyter Notebook, VS code & Streamlit.

Python 3

Python libraries like pandas, NumPy etc.

scikit-learn

Client environment may be Windows or Linux

Web Browser

Model Building and training:

Exploring Various Models and Choosing Logistic Regression

During our experimentation phase, we explored a variety of machine learning models, including Linear Regression (LR), Stochastic Gradient Descent (SGD), Support Vector Machine (SVM), and Random Forest. Overall the initial accuracy is almost in the range 26- 30% for all algos but the one given by logistic regression is the highest, we have proceeded with it.

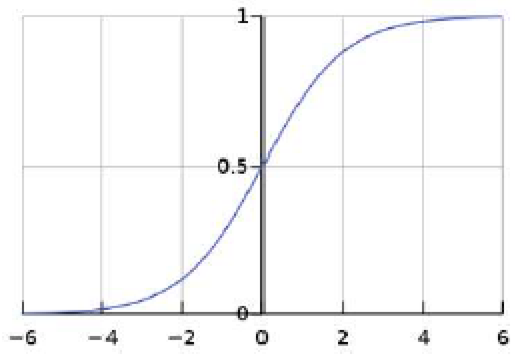
Employing GridSearchCV for cross-validation, we discovered that **Logistic Regression** outperformed the other three models on our relatively small dataset. **Logistic Regression** generally requires less hyperparameter tuning and practically performs well binary classification on text data.

Logistic Regression modeling-

The logistic function is a sigmoid function, which takes any real and outputs a value between zero and one. For the logit, this is interpreted as taking input log odds and having output probability. The standard logistic function is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

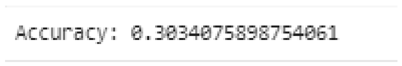
A graph of the logistic function on the t-interval (−6,6) is shown below.



We will follow the brute force method to understand how our model performs at each iteration. We are not taking any user input to correct the incorrect predictions but we are using the actual dataset as the ground truth for now.

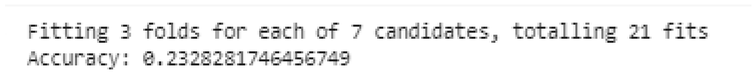
We opted for Logistic Regression as our baseline model. Initially, we trained the model on a subset of 2000 records, utilizing the warm_start=True parameter to facilitate incremental learning. This technique allowed the model to retain the weights of previously trained records, eliminating the need to start training from scratch each time new data was introduced.

1.The initial accuracy of the model was around 23%.



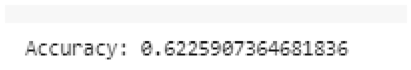
```
Accuracy: 0.3034075898754061
```

2. Next we transferred 10 data points from the test to the train dataset and applied Grid Search CV which is used here for cross-validating and tuning hyperparameters with 3-fold CV and L2 regularization. The accuracy was reduced to 23%



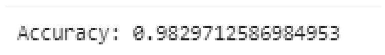
```
Fitting 3 folds for each of 7 candidates, totalling 21 fits
Accuracy: 0.2328281746456749
```

3. As we progressively introduced additional sets of data, the accuracy significantly improved. With the second set of data, the accuracy climbed to 62%



```
Accuracy: 0.6225907364681836
```

4. With the third chunk of data., the model performance reached an impressive 98%.



```
Accuracy: 0.9829712586984953
```

We have achieved very good accuracy in just three iterations. Even the basic Logistic Regression model is working well.

Continuous Learning and User Feedback Integration aspect-

In the production environment, the model will be tasked with predicting labels for user-provided test datasets. In instances where the model produces incorrect predictions, users will have the opportunity to correct those labels. The corrected labels will then be fed back to the model, enabling it to continuously learn and refine its predictions.

This iterative process of prediction, user feedback correction, and retraining will ensure that the LR model remains adaptable and up-to-date, continuously enhancing its ability to provide accurate predictions in real-time. The user's active involvement in the correction process will further strengthen the model's performance and improve the overall user experience.

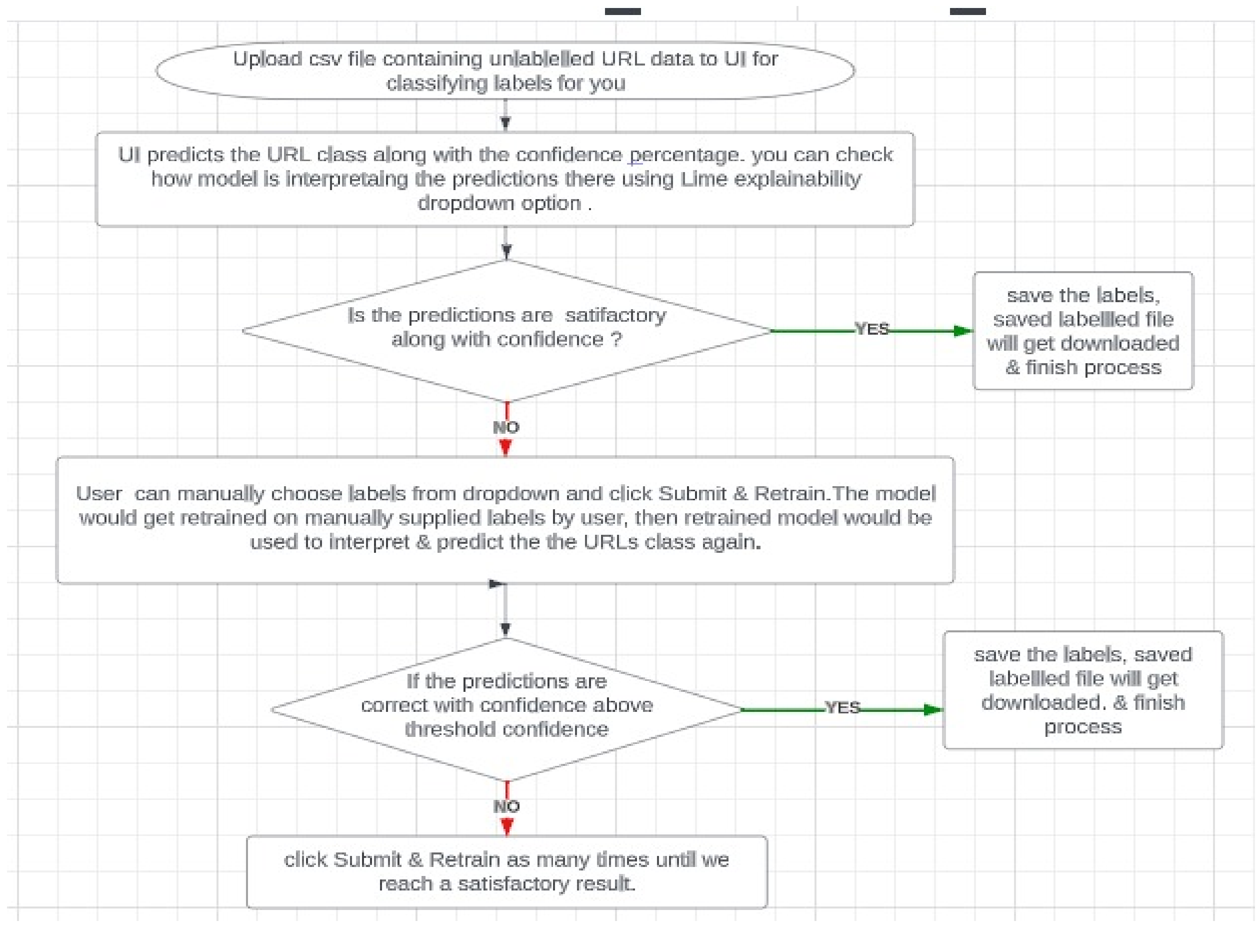
Here we are deploying the whole machine learning pipeline into a production system, into a real-time scenario. In a real-world production scenario, the model will be deployed where it takes continuous raw input from the real world and predicts the output. But for our prototype case, we just kept a single user interface providing excel csv files as input.

User interface :

User interface is built on a Streamlined application which is an open source platform.

The code is written in stream_app.py file which will use a baseline model & retrained model to showcase how active learning improves prediction accuracy & confidence scores on the predicted labels.

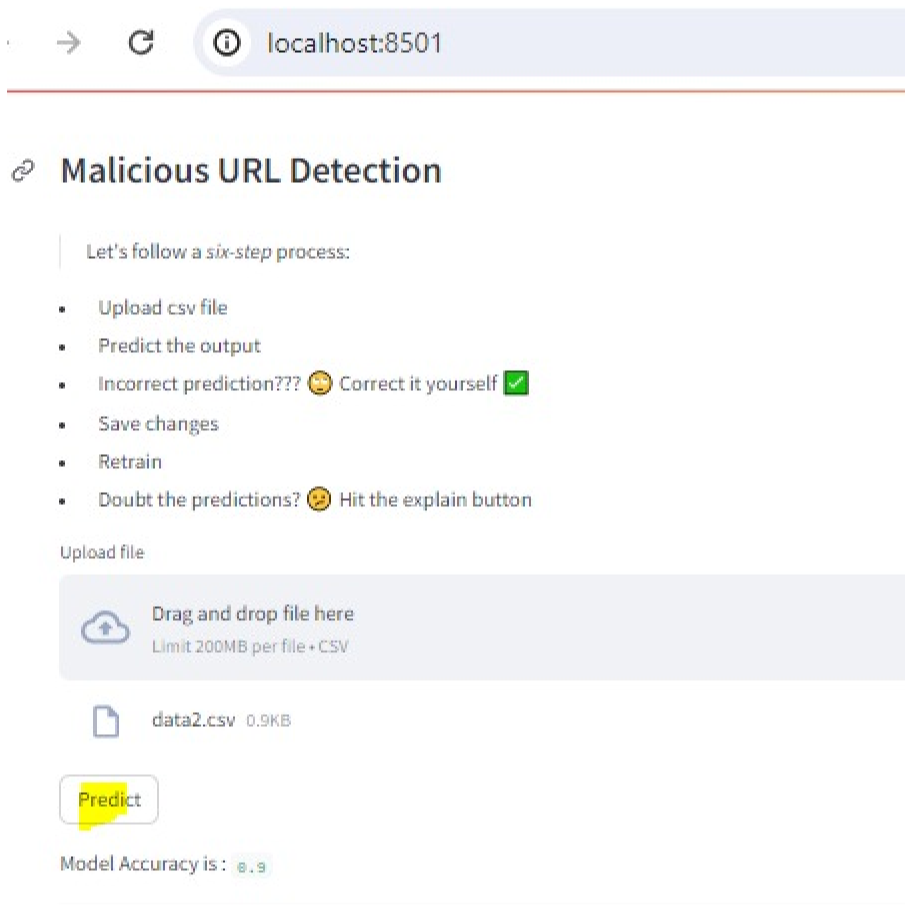
Flowchart of UI :


Upload csv file containing unlablelled URL data to UI for classifying labels for you
UI predicts the URL class along with the confidence percentage. you can check how model is interpretaing the predictions there using Lime explainability dropdown option .
Is the predictions are satifactory along with confidence ?
YES
save the labels, saved labellled file will get downloaded & finish process
NO
User can manually choose labels from dropdown and click Submit & Retrain.The model would get retrained on manually supplied labels by user, then retrained model would be used to interpret & predict the the URLs class again.
If the predictions are correct with confidence above threshold confidence
YES
save the labels, saved labellled file will get downloaded. & finish process
NO
click Submit & Retrain as many times until we reach a satisfactory result.

FUNCTIONING OF UI labeling tool:

On launching the URL in the browser, the initial page shows the instructions to operate the page.

1.Just browse and select a csv file having URL data and click on the predict button to get baseline model predictions.


localhost:8501

Malicious URL Detection

Let's follow a *six-step* process:

- Upload csv file
- Predict the output
- Incorrect prediction??? 😳 Correct it yourself ✅
- Save changes
- Retrain
- Doubt the predictions? 😕 Hit the explain button

Upload file

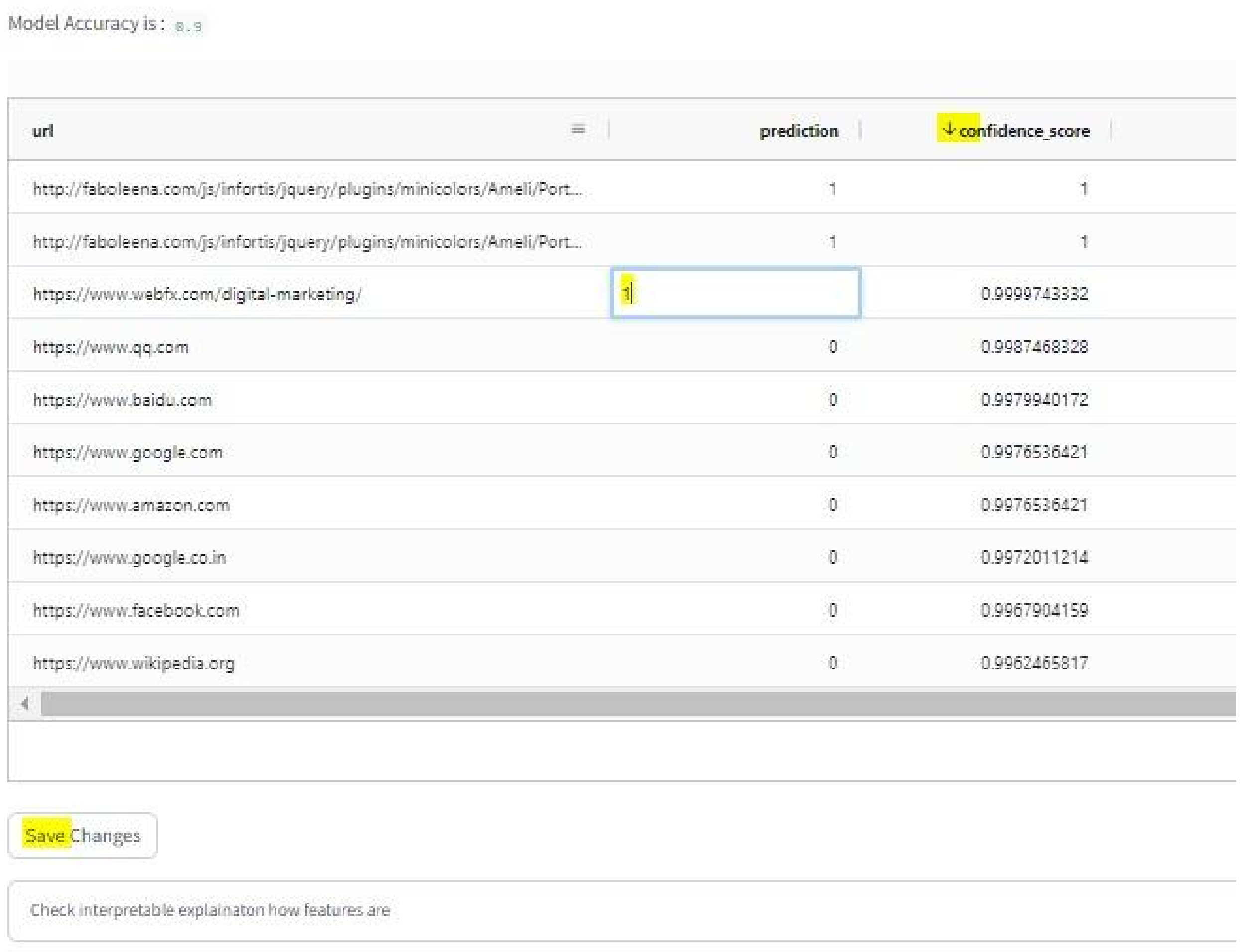
Drag and drop file here
Limit 200MB per file • CSV

data2.csv 0.9KB

Predict

Model Accuracy is : `0.9`

2. On predicting, we will get the data along with predictions and confidence scores against each prediction. User /Doman expert has to just validate if the predictions are correct or not. In case of any incorrect predictions, user has to correct the predicted label and click on save changes. We can also sort the predictions on confidence scores.



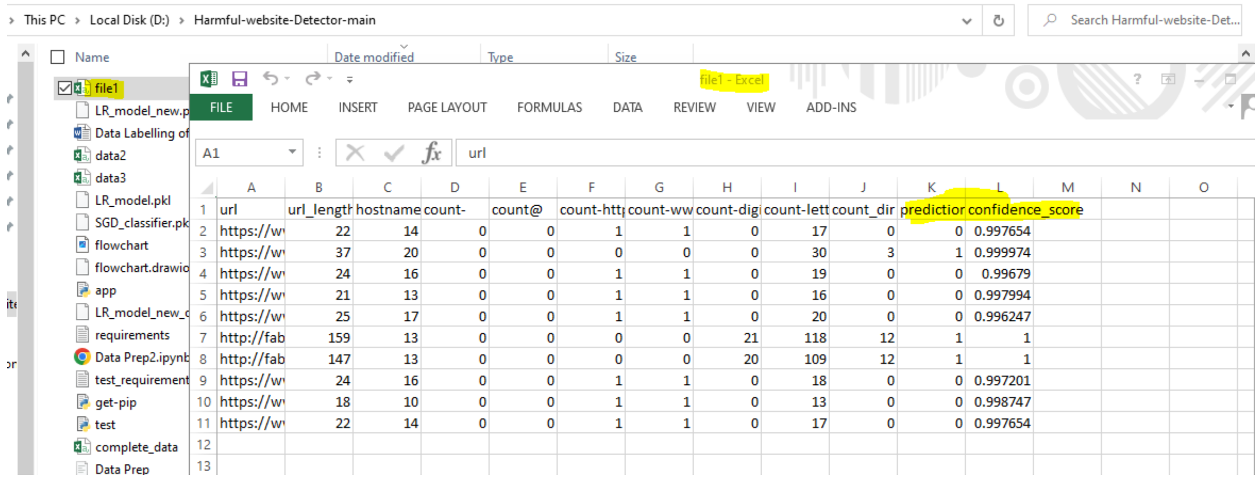
Model Accuracy is : 0.9

url	prediction	confidence_score
http://faboleena.com/js/infortis/jquery/plugins/minicolors/Ameli/Port...	1	1
http://faboleena.com/js/infortis/jquery/plugins/minicolors/Ameli/Port...	1	1
https://www.webfx.com/digital-marketing/	1	0.9999743332
https://www.qq.com	0	0.9987468328
https://www.baidu.com	0	0.9979940172
https://www.google.com	0	0.9976536421
https://www.amazon.com	0	0.9976536421
https://www.google.co.in	0	0.9972011214
https://www.facebook.com	0	0.9967904159
https://www.wikipedia.org	0	0.9962465817

Save Changes

Check interpretable explainaton how features are

3. On clicking the save changes button a data frame will appear with the modified values and it will also get saved in our backend/ local folder.

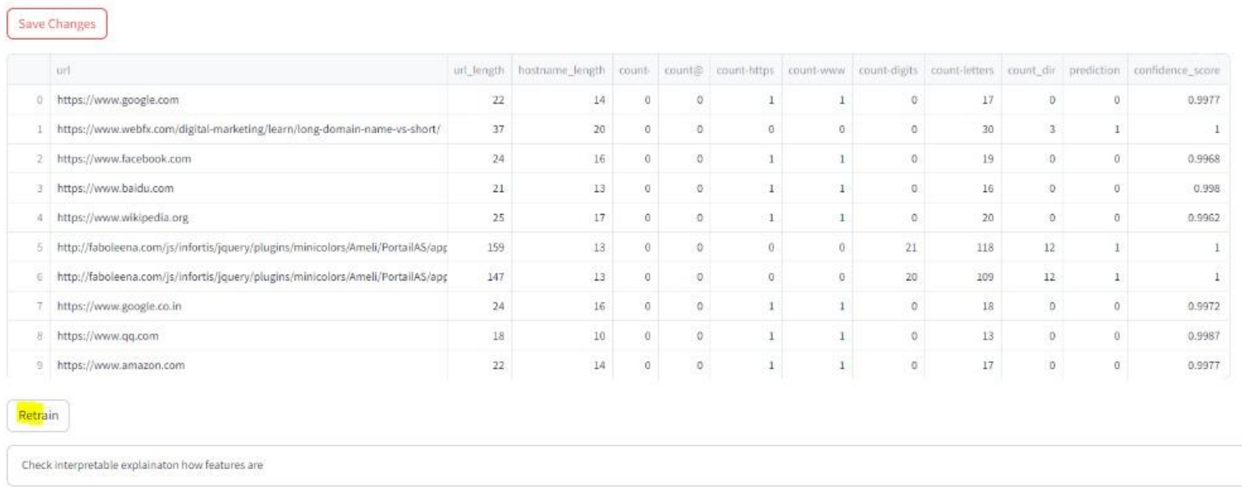


This PC › Local Disk (D:) › Harmful-website-Detector-main

file1 - Excel

	A	B	C	D	E	F	G	H	I	J	K	L
1	url	url_length	hostname	count-	count@	count-http	count-ww	count-digi	count-lett	count_dir	predictior	confidence_score
2	https://w	22	14	0	0	1	1	0	17	0	0	0.997654
3	https://w	37	20	0	0	0	0	0	30	3	1	0.999974
4	https://w	24	16	0	0	1	1	0	19	0	0	0.99679
5	https://w	21	13	0	0	1	1	0	16	0	0	0.997994
6	https://w	25	17	0	0	1	1	0	20	0	0	0.996247
7	http://fab	159	13	0	0	0	0	21	118	12	1	1
8	http://fab	147	13	0	0	0	0	20	109	12	1	1
9	https://w	24	16	0	0	1	1	0	18	0	0	0.997201
10	https://w	18	10	0	0	1	1	0	13	0	0	0.998747
11	https://w	22	14	0	0	1	1	0	17	0	0	0.997654

4. Once we have our modified data saved, we can retrain the baseline model using the new data. We have used Logistic Regression with a warm start as True which lets us train the model from where we left at and not retraining from scratch. This helps to reduce computational cost and is fast as well.

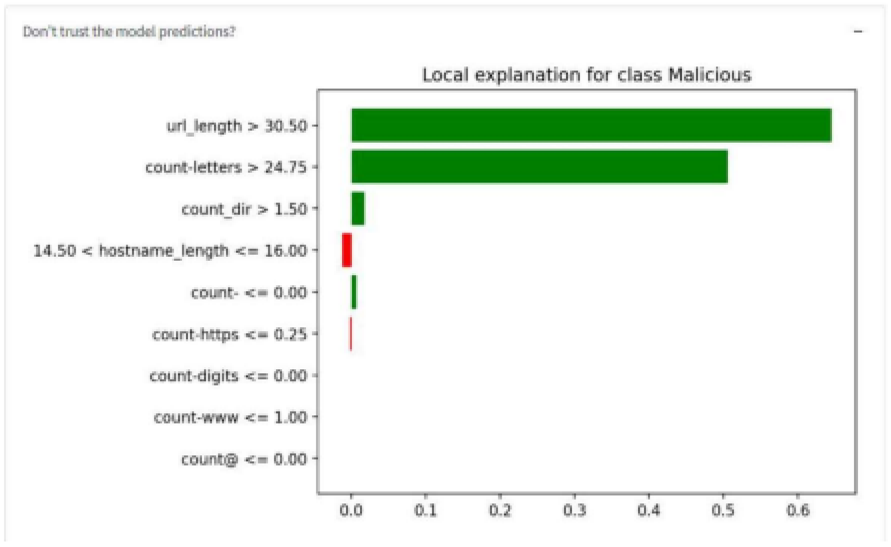


	url	url_length	hostname_length	count-	count@	count-https	count-www	count-digits	count-letters	count_dir	prediction	confidence_score
0	https://www.google.com	22	14	0	0	1	1	0	17	0	0	0.9977
1	https://www.webfx.com/digital-marketing/learn/long-domain-name-vs-short/	37	20	0	0	0	0	0	30	3	1	1
2	https://www.facebook.com	24	16	0	0	1	1	0	19	0	0	0.9968
3	https://www.baidu.com	21	13	0	0	1	1	0	16	0	0	0.998
4	https://www.wikipedia.org	25	17	0	0	1	1	0	20	0	0	0.9962
5	http://faboleena.com/js/infortis/jquery/plugins/minicolors/Ameli/PortailAS/apr	159	13	0	0	0	0	21	118	12	1	1
6	http://faboleena.com/js/infortis/jquery/plugins/minicolors/Ameli/PortailAS/apr	147	13	0	0	0	0	20	109	12	1	1
7	https://www.google.co.in	24	16	0	0	1	1	0	18	0	0	0.9972
8	https://www.qq.com	18	10	0	0	1	1	0	13	0	0	0.9987
9	https://www.amazon.com	22	14	0	0	1	1	0	17	0	0	0.9977

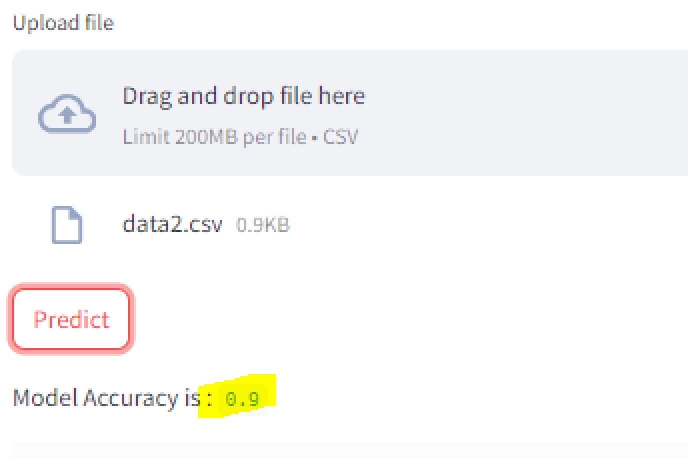
Check interpretable explainaton how features are

5. After retraining, our baseline model will be modified with the newly trained model and over the next iterations, our model's performance will keep on improving.

6. Additionally, we have used LIME to showcase our model interpretability. The green bars indicates those features which contributes towards malicious prediction. Here's the demo video link of the UI we developed- https://drive.google.com/file/d/1PYy24U0zoSbbzDOYaw4Y3vRhTzF6xmr9/view?usp=sharing



Result Analysis:

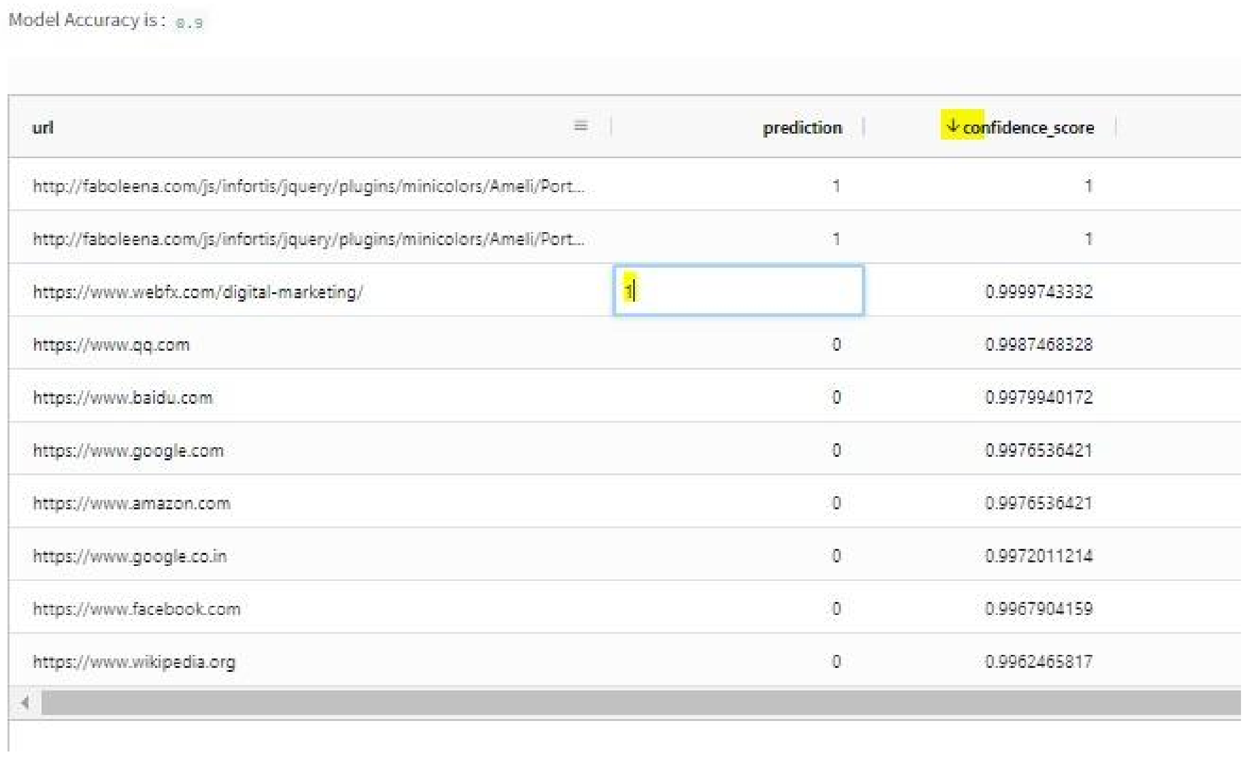


Upload file

Drag and drop file here
Limit 200MB per file • CSV

data2.csv 0.9KB

Model Accuracy is : 0.9



Model Accuracy is : 0.9

url	prediction	↓confidence_score
http://faboleena.com/js/infortis/jquery/plugins/minicolors/Ameli/Port...	1	1
http://faboleena.com/js/infortis/jquery/plugins/minicolors/Ameli/Port...	1	1
https://www.webfx.com/digital-marketing/	1	0.9999743332
https://www.qq.com	0	0.9987468328
https://www.baidu.com	0	0.9979940172
https://www.google.com	0	0.9976536421
https://www.amazon.com	0	0.9976536421
https://www.google.co.in	0	0.9972011214
https://www.facebook.com	0	0.9967904159
https://www.wikipedia.org	0	0.9962465817

The model achieved at least 90% accuracy and a confidence score of over 99% on testing for the chunks of data, making it a strong candidate for deployment in a production setup.

Generally in practice, these scenarios are very rare with complex datasets where the model achieved such high accuracy and proposes a continuous improvement process in which the model is upgraded by retraining on the manually corrected data labels or the ones having the highest confidence score and keep repeating until a certain threshold is achieved. This iterative and incremental learning approach aligns with best practices for maintaining and improving model performance over time.

References:

- https://docs.aws.amazon.com/sagemaker/latest/dg/sms-automated-labeling.html
- https://medium.com/mindboard/active-learning-for-fast-data-set-labeling-890d4080d750
- https://www.knime.com/blog/labeling-with-active-learning
- https://towardsdatascience.com/active-learning-overview-strategies-and-uncertainty-measures-521565e0b0b
- https://arxiv.org/abs/2307.02719
- http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://stackoverflow.com/questions/45651096/warm-start-parameter-and-its-impact-on-computational-time
- http://www.chioka.in/class-imbalance-problem/
- https://en.wikipedia.org/wiki/Logistic_regression#Definition_of_the_logistic_function
- https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-usingsmote-techniques/
- https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanceddata-for-a-classification-problem/#:~:text=Imbalanced%20data%20refers%20to%20those,understand%20it%20with%20an%20example
- https://docs.streamlit.io/
- https://python-data-science.readthedocs.io/en/latest/activelearning.html
- https://paperswithcode.com/task/active-learning/latest
- https://www.diva-portal.org/smash/get/diva2:1586990/FULLTEXT01.pdf
- https://www.datacamp.com/tutorial/active-learning
- https://en.wikipedia.org/wiki/Active_learning_(machine_learning)