CMPE-255 Project Report
Phishing Website Detection

Section 1: Introduction

Motivation :

- ➔ Phishing websites disguise themselves as trustworthy websites in order to gain the trust of their victims, and malicious parties use them to obtain sensitive information from their victims.
- ➔ Attackers change the subdomain and file path(if it contains in the URL) or they make a typo mistake to look like a legitimate website. So to identify phishing websites we need to break down the URL to look at what each part of the URL contains.

Objective:

The objective of the project is to detect phishing websites by using website URL features like top level domain, filepath , domain activation time etc.

Approach :

- ➔ We broke down the URL into multiple parts: protocol, domain name, directory name, file name and other URL parameters, then found the relation of the features to the target variable.
- ➔ We have chosen important features by visualizing the features using correlation heat map, using mutual information to the target variable.
- ➔ After choosing the important features from the dataset we have applied various ML models like Regression, Support Vector Machine, KNN, Random Forest, Stacking and XGboost.
- ➔ We have compared the accuracy, precision and recall of the above mentioned models along with the time complexity of the model.

Literature/Market review :

There have been several different approaches to detect phishing websites. One notable research article,[1] *Know Your Phish: Novel Techniques for Detecting Phishing Sites and their Targets,* discusses using machine learning models to classify websites to be legitimate or not. In this article, there was a lot of focus in data collection and preparation. The authors collected data from the internet from the well known phishtank website, and used Python scripts to extract the key features like HTML source code, Starting URL , Landing URL and preprocess the data.

To use the features extracted from the python script for discriminating phishing from legitimate they used supervised machine learning algorithms and for that, they chose classification Gradient Boosting model. They used 5-fold cross validation and got accuracy of around 95%.

Section 2: System Design and implementation

Algorithms considered/ selected:

➔ Regression, Support Vector Machine, KNN, Random Forest, Stacking and XGboost. We considered Decision Tree Algorithm, and did not select it because

Technologies & tools used (and why) :

➔ Seaborn, Matplotlib, Numpy, Panda, Scala, Yellow brick
➔ Jupyter Notebook

System (and subsystems if needed) design/architecture/data flow (you may use diagrams with some supportive text):

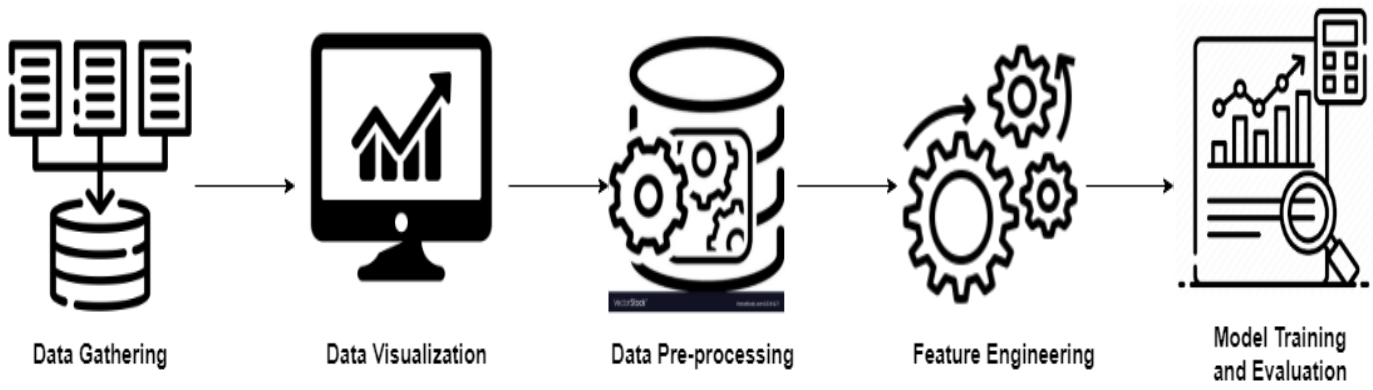

Fig. 1 Data Flow diagram

Section 3: Experiments / Proof of Concept Evaluation:

Dataset(s) used :

➔ We took the dataset from mendeley data named “Phishing website Dataset”.
➔ There are a total of 111 features from which 4 features namely domain_in_ip, server_client_domain, email_in_url , domain_spf are categorical and rest is numerical. Total number of rows: 88647
➔ Data cleaning done with features having inappropriate and constant values.

Methodology followed:

➔ We have used 10 stratified cross validation. In the total no of rows (i.e. 88647), training size is 80% of the total, and testing is 20% of the total.

Feature selection using correlation heat map

➔ We divide the features into the subgroup for better visualization of the correlation heatmap.

➔ **URL - based features** :

Insight : quantity of equal in the URL and quantity of and in the URL is highly correlated and quality of slash URL is highly correlated with target variable.

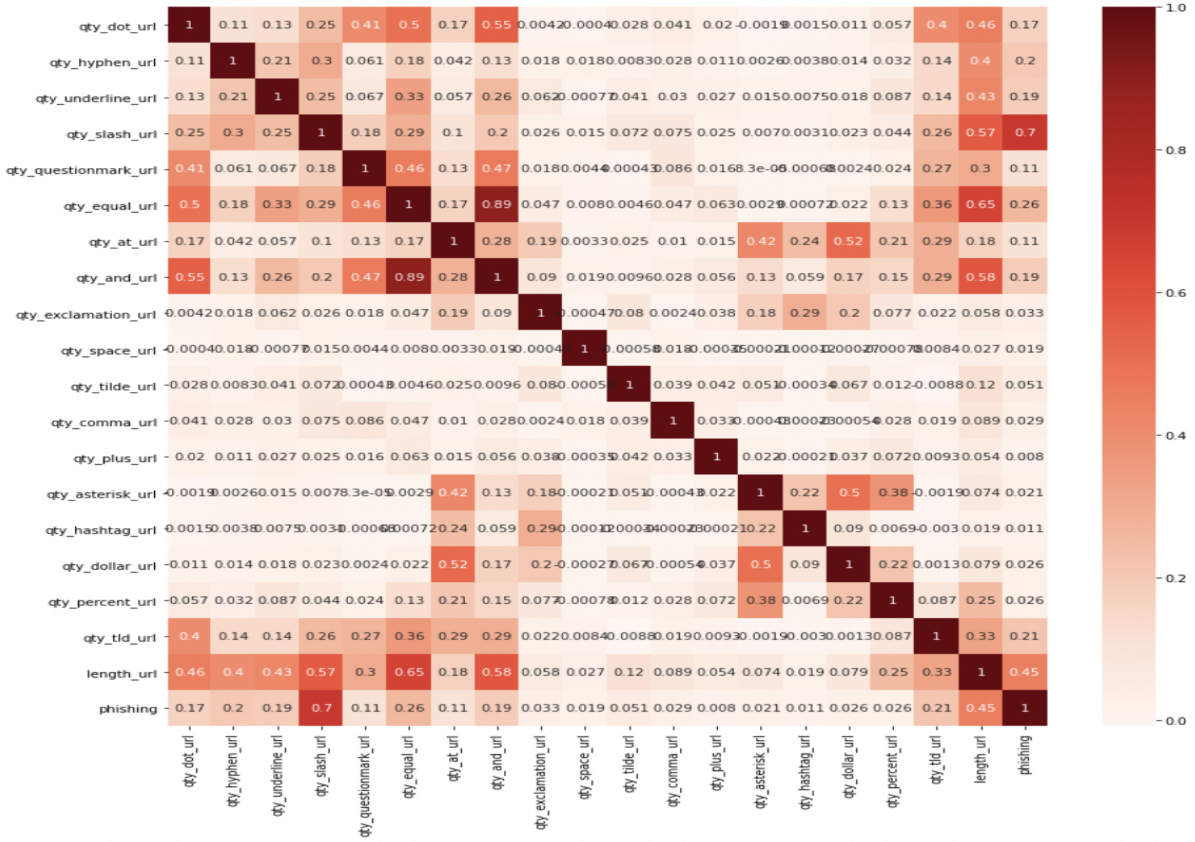

Fig. 2 URL based feature correlation heatmap

➔ **Domain based features**

Insights : There are no domain features which are highly correlated with the target variable but quantity of the vowels in the domain and domain length is positively correlated.

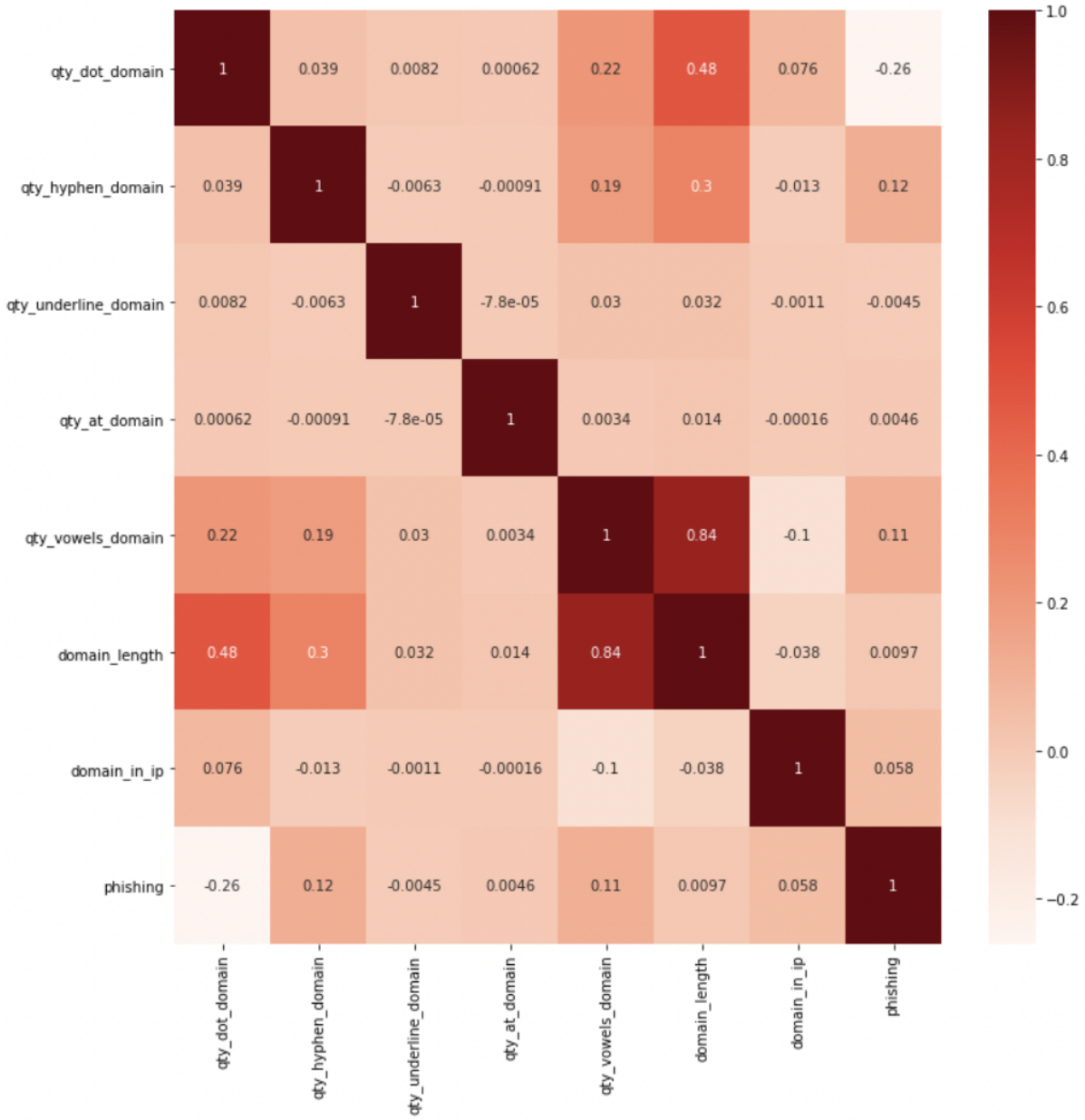

Fig. 3 Domain based feature correlation heatmap

➔ **External service based feature.**

Insight : Time domain activation is inversely proportional to the target variable. having google indexed of the url and domain is correlated with each other.

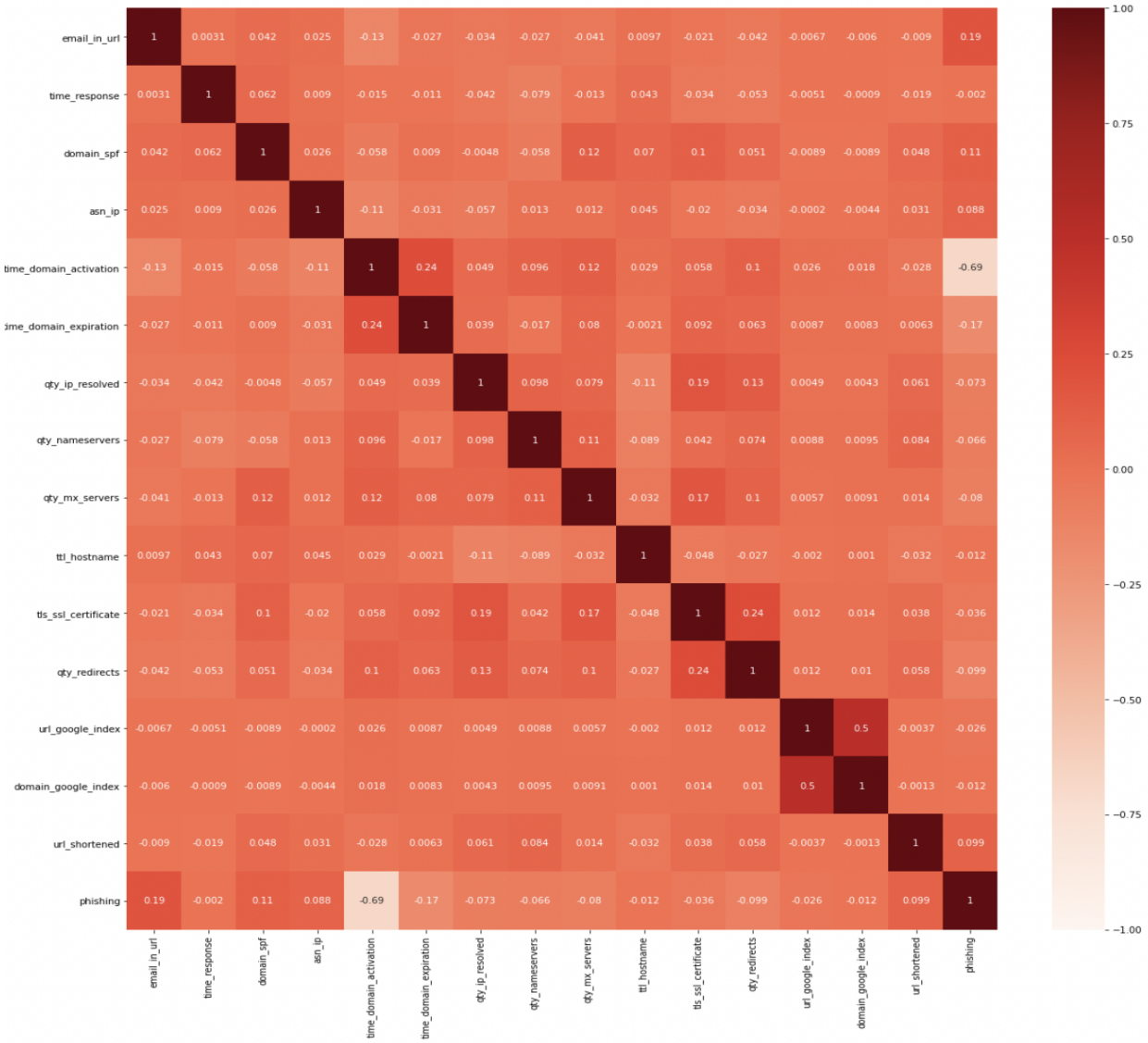

Fig. 4 External feature based correlation heatmap

Feature selection using mutual information to the target variable.

Feature importance of the different categories to the target variable.

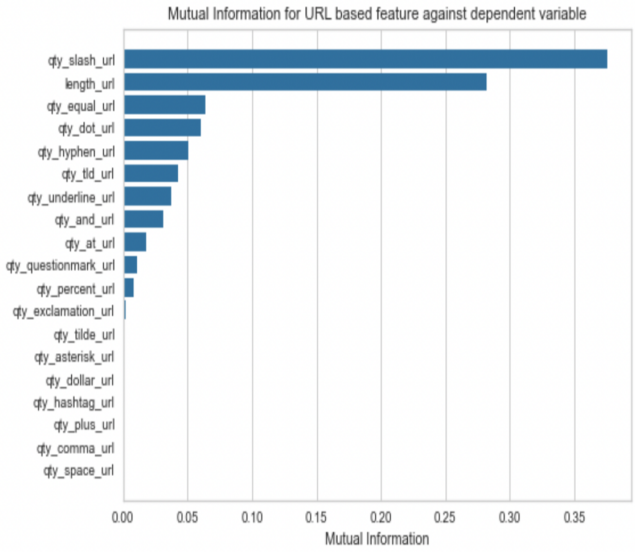

Fig. 5 Feature importance graph for URL based feature

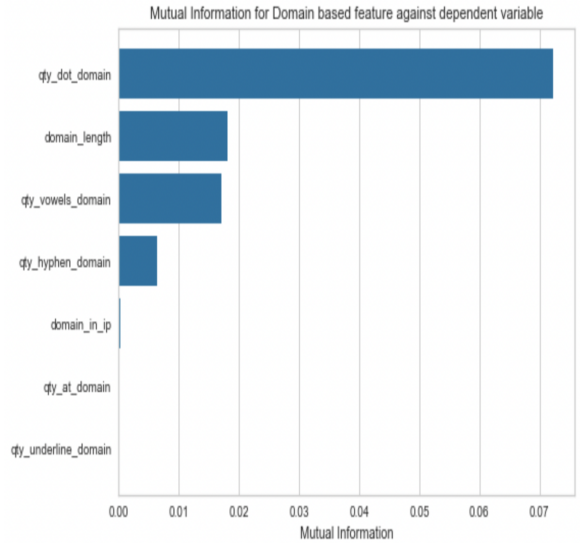

Fig. 6 Feature importance for Domain based feature

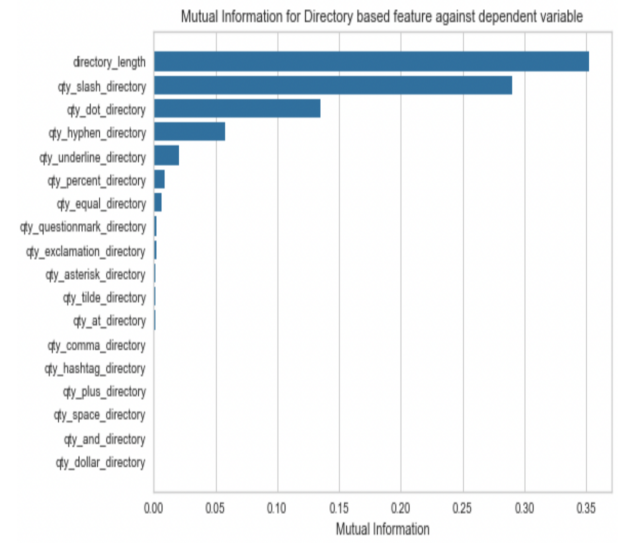

Fig. 7 Feature importance graph for Directory based feature

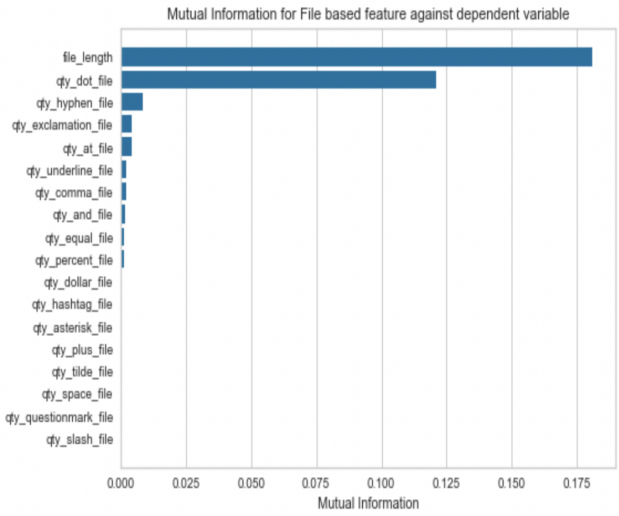

Fig. 8 Feature importance of File based features

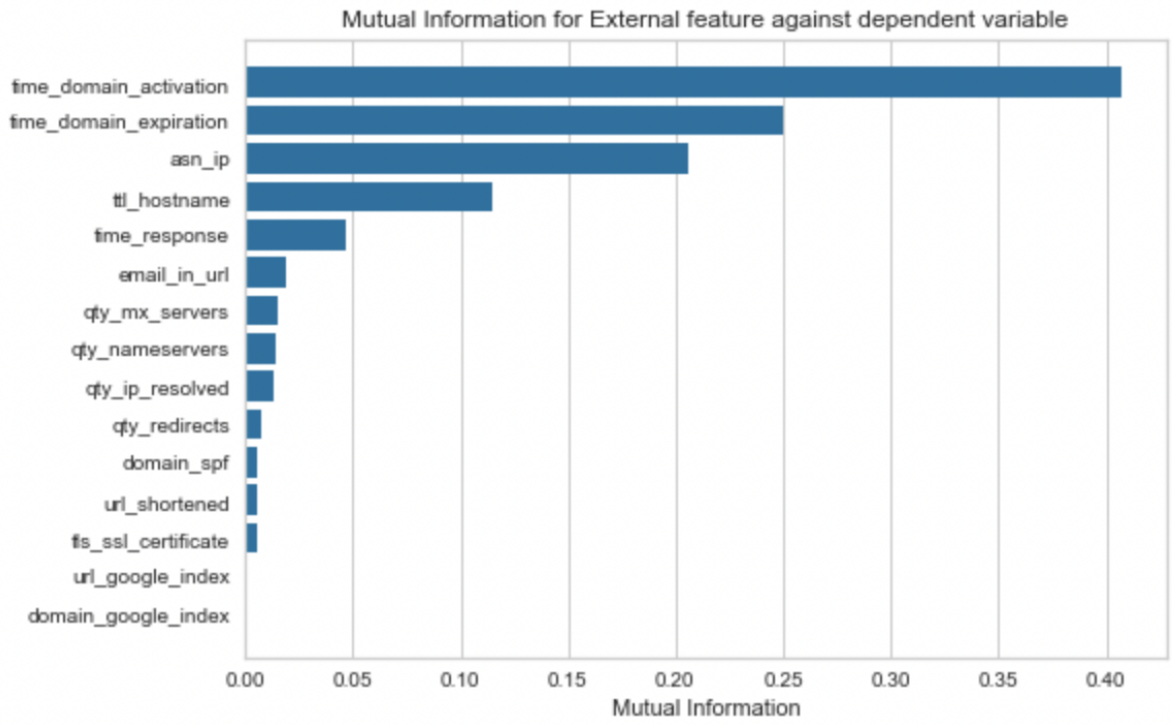

Fig. 9 Feature importance graph for External features

Insights **:**

- Fig 5 : quantity of the slash in the URL and length of the URL are more important features to predict whether the website is legitimate or not.
- Fig 6 : No Domain related feature has any information to decide whether the URL is scam or not.
- Fig 7 : quantity of the slash in the Directory and length of the Directory are more important features to predict whether the website is legitimate or not.
- Fig 8 : No File related feature has any information to decide whether the URL is scam or not.
- Fig 9: Activation and expiration time of the URL domain are more important external features to predict whether the website is legitimate or not.

Graphs showing different parameters/algorithms evaluated in a comparative manner, along with some supportive text (as applicable):

Model	Best Hyperparameter	Accuracy	Precision	Recall	Time taken for algorithm convergence
Logistic Regression	'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'	93%	95%	95%	47s
KNN	'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'	97.85%	98%	98%	45 min
SVM	'C': 50, 'gamma': 'scale', 'kernel': 'rbf'	97.4%	98%	98%	1 hr 15 min
Random Forest	'max_features': 'log2', 'n_estimators': 1000	98.5%	99%	99%	30 min
Naive Bayes Classifier	'var_smoothing': 0.053	87%	86%	97%	36s
Stacking	Use Best Hyperparameter for the LR, SVM,Random Forest , KNN, NB	98.4%	99%	99%	400s
XGBoost	**'learning_rate': 1, 'max_depth': 5, 'max_features': 2, 'n_estimators': 150**	**98.5%**	**99%**	**99%**	**48s**

Fig. 10 Comparision of applied machine learning models

Analysis of results:

➔ By comparing different models, we found that XGBoost worked the best with an accuracy of 98.5% on testing data, precision and recall of 99 %. The time taken for algorithm convergence was 48seconds which was the best out of all models.

```
Time taken to find best parameters for GridSearchCV  is  45.0  seconds.
MSE :  0.016187253243090807
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     11612
           1       0.97      0.98      0.98      6118

    accuracy                           0.98     17730
   macro avg       0.98      0.98      0.98     17730
weighted avg       0.98      0.98      0.98     17730

Accuracy:  0.9838127467569092
```

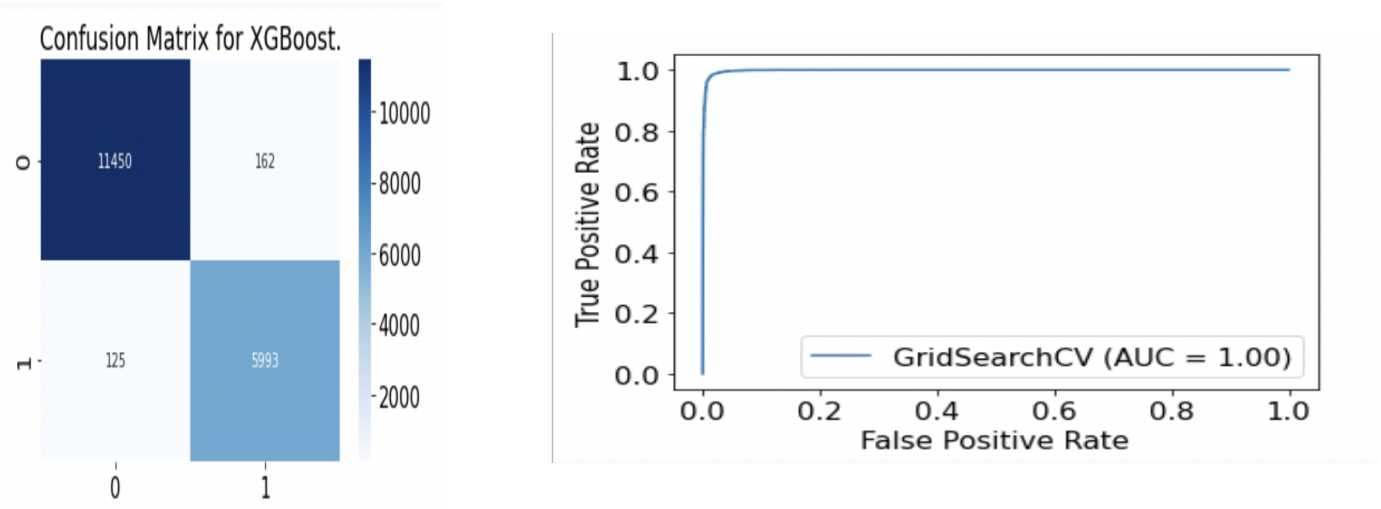

Fig. 11 Performance of XGBoost

Section 4: Discussion and Conclusions:

Decisions made:

- ➔ We divided the features into various categories which made our job easy to do visualization and feature selection.
- ➔ We decided to use feature selection because when we fed 111 features to our model we got the accuracy of 93%, which was a lot less after doing feature selection.

Difficulties faced:

- ➔ After doing the data visualization we got to know about many inappropriate values in our features, those values were not NaN values, so we had to make our functions to replace those values using mean, median or mode.

Things that worked well:

- ➔ After feature selection, our accuracy of the models boosted.
- ➔ We have various visualization graphs for our data which helped us to better select the relevant features. Dividing the features into categories went well, it helped us to better visualize the graph

Things that didn't work well:

- ➔ Naive Bayes, and Knn didn't work well on the given features.

Future work identified based on your experience with this project (problems that would be interesting to explore in the future):

- ➔ We can extend this project to incorporate the content of the website and from the content we can identify whether the website is legitimate or not using NLP models.
- ➔ Create the UI, to allow users to enter the website and get the result of the website being legitimate or not.

Conclusion:

- ➔ As we have a lot of features, Random forest and XG Boost worked well as this type of algorithm internally does feature importance.
- ➔ Not having SSLcertificate increases the chances of the website being not legitimate.
- ➔ If the domain has lesser activation time then the website tends to be an illegitimate one.
- ➔ Website is legitimate or not majorly depends on the length of the url, file path and the number of slash in the url. These insights were found after data visualization for the given dataset.

Section 5: Project Plan / Task Distribution:

Who was assigned to what task:

Shilpi Soni: Selecting the domain, gathering domain knowledge, finding the dataset, feature distribution.

Urja Naik: Data visualization, extracting key insights.

Vishwa Shah: Feature selection and ML model training.

Github URL :https://github.com/VishwaShah7e7/Phishing-Website-dataset-using-ML

References

[1] S. Marchal, K. Saari, N. Singh, N. Asokan, "*Know Your Phish: Novel Techniques for Detecting Phishing Sites and their Targets*