# CLASSIFICATION PROJECT REPORT

**Is It Phishing?**

**ABSTRACT**
 A SDAIA T5 Data scince boot camp fourth project aim to propose a model that clasify URLs and verefy whether it is Phishing or Begnin

**Norah Alqahtani, Albatoul Alosaimi, and Shroaq Almutairi**

## Introduction

Malicious URL, a.k.a. malicious website, is a common and serious threat to cybersecurity. Malicious URLs host unsolicited content (spam, phishing, drive-by downloads, etc.) and lure unsuspecting users to become victims of scams (monetary loss, theft of private information, and malware installation), and cause losses of billions of dollars every year. It is imperative to detect and act on such threats in a timely manner. Traditionally, this detection is done mostly through the usage of blacklists. However, blacklists cannot be exhaustive, and lack the ability to detect newly generated malicious URLs. To improve the generality of malicious URL detectors, machine learning techniques have been explored with increasing attention in recent years. So, we aim in this Project to take advantage of Classification techniques to predict whether a specific URL malicious od benign.

## Design And Data Description

Our Data Sources consist of tow Parts, First one is group of URLs that has been checked via **whois** command to verify whether it is *benign* or *Phishing* URL. while second part is features of these URLs which have been fetched via **whois** command too. the features were: whois_regDate, whois_expDate, whois_updatedDate, dot_count, url_len, digit_count, special_count, hyphen_count, double_slash, single_slash, at_the_rate, protocol, and protocol_count. Then we merged these Two Datasets to have **data_df** Dataframe

| | whois_regDate | whois_expDate | whois_updatedDate | dot_count | url_len | digit_count | special_count | hyphen_count | double_slash | single_slash | at_the_rate | p |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 403 | 326 | 23 | 6 | 225 | 58 | 12 | 4 | 0 | 10 | 0 | |
| 1 | 2727 | 194 | 168 | 7 | 177 | 47 | 0 | 1 | 0 | 11 | 0 | |
| 2 | 5431 | 46 | 317 | 6 | 60 | 0 | 0 | 0 | 0 | 2 | 0 | |
| 3 | 3643 | 374 | 5 | 1 | 116 | 21 | 1 | 1 | 1 | 10 | 0 | |
| 4 | -1 | -1 | -1 | 3 | 36 | 0 | 0 | 0 | 0 | 1 | 0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 9995 | 8450 | 316 | 473 | 2 | 23 | 0 | 0 | 0 | 0 | 2 | 0 | |
| 9996 | 8719 | 776 | 1109 | 3 | 28 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 9997 | 7481 | 553 | 112 | 3 | 43 | 1 | 0 | 0 | 0 | 3 | 0 | |
| 9998 | 7658 | 376 | 3 | 3 | 38 | 0 | 0 | 1 | 0 | 1 | 0 | |
| 9999 | 8443 | 689 | 824 | 3 | 32 | 0 | 0 | 0 | 0 | 1 | 0 | |

10000 rows × 14 columns

## Algorithms

- We build Decision Tree, KNN, Random Forest, XGBoost, SVM, Naive Bayes, and Logistic Regression in python, and techniques such as regularization and polynomial features, adding interaction terms and dummy variables have been used.
- Rigorous model selection and evaluation has been used to select model between Decision Tree, KNN, Random Forest, and Logistic regression Regression
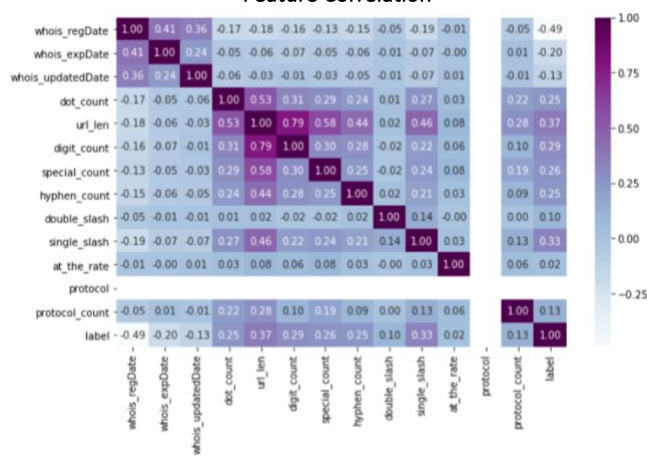
## Tools

- Pandas and Numpy (Exploring the data)
- Matplotlib and Seaborn (Visualizing the data)
- Sci-kit Learn (linear Regression model and other models)
- Xgboost
- Whois and tldextract
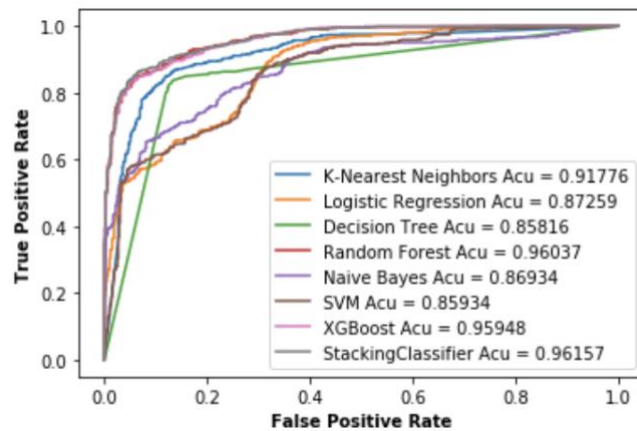- FLASK API

# Communication

### Data Balance

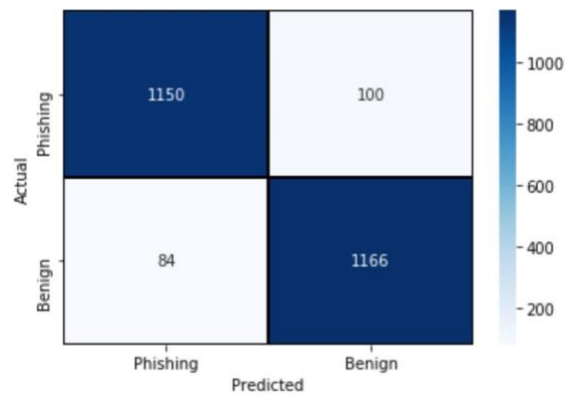<matplotlib.axes._subplots.AxesSubplot at 0x7ff639e355d0>
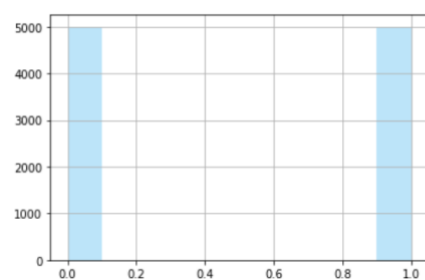


### Feature Correlation



## Results of all tested model

|   | Model | Accuracy | Recall | Precision | F1 score |
|---|---|---|---|---|---|
| 0 | KNN | 0.860833 | 0.873355 | 0.855072 | 0.864117 |
| 1 | Logistic Regression | 0.755833 | 0.789474 | 0.744186 | 0.766161 |
| 2 | Decision Tree | 0.851667 | 0.837171 | 0.865646 | 0.851171 |
| 3 | Random Forest | 0.884167 | 0.881579 | 0.888889 | 0.885219 |
| 4 | Naive Bayes | 0.744167 | 0.523026 | 0.949254 | 0.674443 |
| 5 | SVM | 0.775833 | 0.856908 | 0.741110 | 0.794813 |
| 6 | XGBoost | 0.877500 | 0.875000 | 0.882255 | 0.878613 |
| 7 | StackingClassifier | 0.889167 | 0.891447 | 0.889984 | 0.890715 |

ROC Curve



- K-Nearest Neighbors Acu = 0.91776
- Logistic Regression Acu = 0.87259
- Decision Tree Acu = 0.85816
- Random Forest Acu = 0.96037
- Naive Bayes Acu = 0.86934
- SVM Acu = 0.85934
- XGBoost Acu = 0.95948
- StackingClassifier Acu = 0.96157

confusion mtrix of Random Forest model



Final results

```
In [52]:    1  print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       0.93      0.92      0.93      1250
           1       0.92      0.93      0.93      1250

    accuracy                           0.93      2500
   macro avg       0.93      0.93      0.93      2500
weighted avg       0.93      0.93      0.93      2500
```

## Conclusion

- Our model solve the problem of classifing URLs with accuricy(0.93) and F1(0.93)
- By comparing our results with the results of the previous work; our accuracy is better where theirs is 0.87

## future work

- Publish a website
- Improving the Model until F1= 0.96
- Create ourown Dataset