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

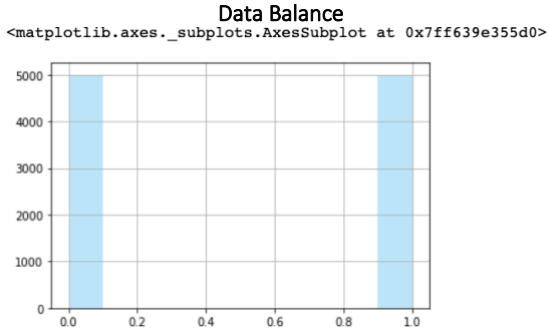
- We build Decision Tree, KNN, Random Forest, XGBoost, SVM, Naive Bayes, and Logistic Regression in python, and techniques such as regularization and polynomial features, adding interaction terms and dummy variables have been used.
- Rigorous model selection and evaluation has been used to select model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

Tools

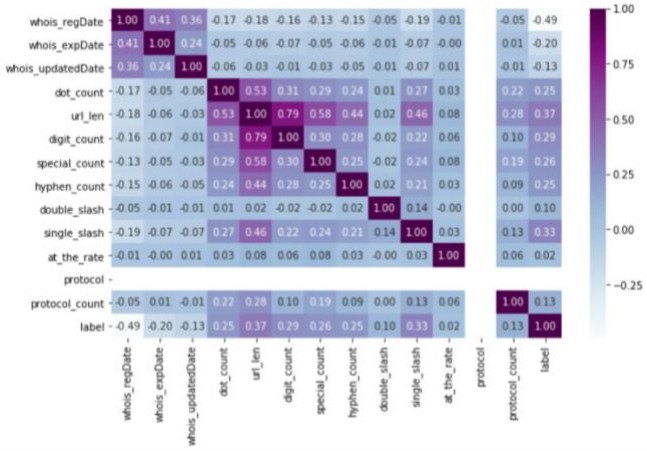
- Pandas and Numpy (Exploring the data)
- Matplotlib and Seaborn (Visualizing the data)
- Sci-kit Learn (linear Regression model and other models)
- Xgboost
- Whois and tldextract
- FLASK API

Communication

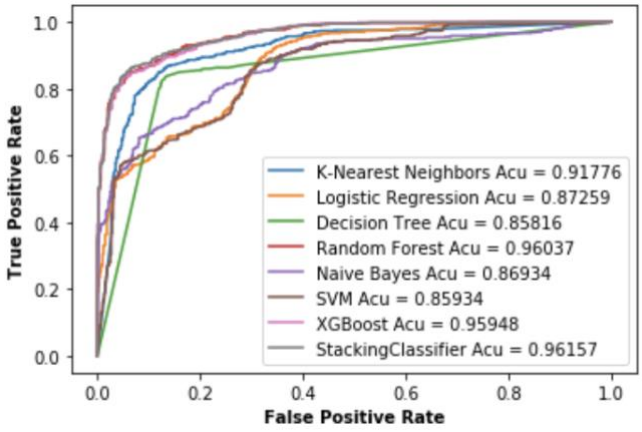
Data Balance



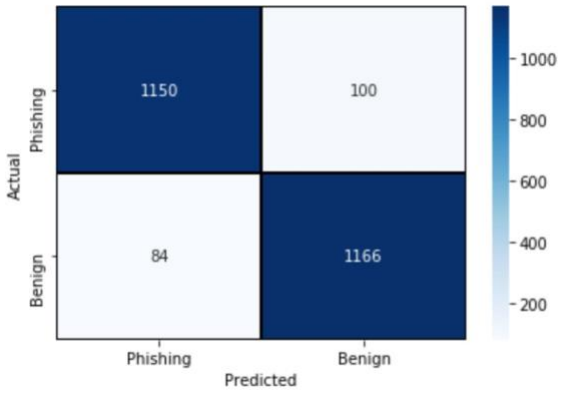
Feature Correlation



ROC Curve



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

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