Phishing Detection

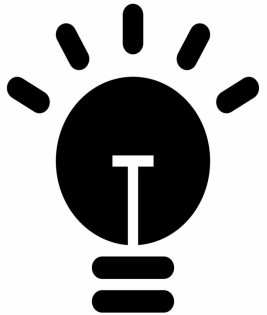
Using Ensemble methods of Machine Learning

Discussion Points

1. Introduction
2. How it works?
3. Problem statement
4. Schema
5. Data Pre-Processing
6. Feature Selection
7. Model Implementation
8. Results
9. Conclusion

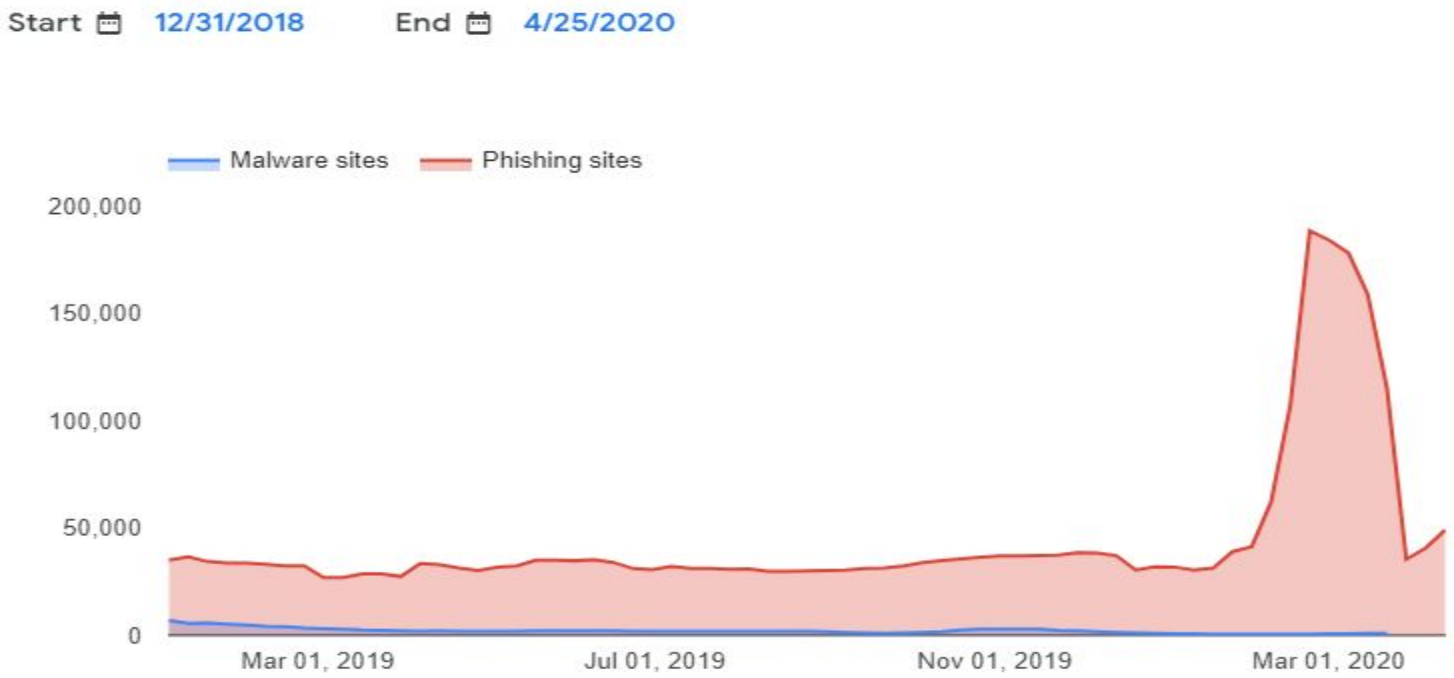
Introduction

- **Phishing** is a fraudulent practice that attempts to acquire sensitive information like passwords, and credit card details through deceptive emails and websites. It is **sophisticated and a old form of cybercrime**, since 1990s.
- It is **called ‘Phishing’** because it is similar to a **fisher throwing a baited hook** to catch the fishes, which here is the attacker spamming deceptive emails to gain confidential data.
- Phishing accounts for **90% of the data breaches** and 30% of phishing messages gets opened. **Facebook and Google,** were scammed through a **fake invoice** for over $100 million.
- Preventive measures: using **multi-factor authentication**, using **security software** and frequent **software updates.**

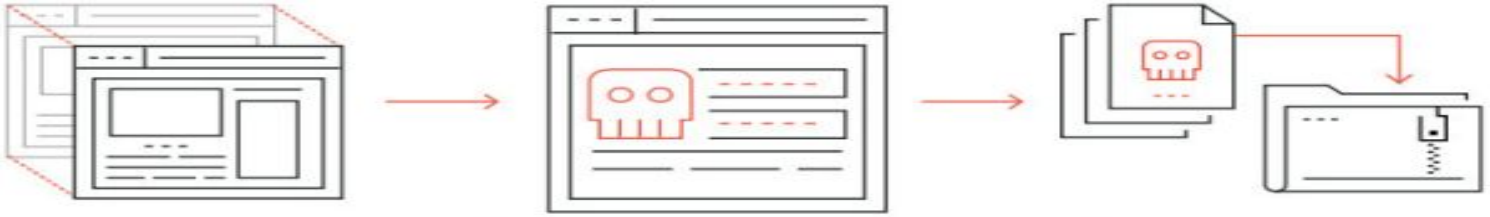


Increase in Phishing attacks due to Covid-19:

As of March since the beginning of the year, the number of **phishing attacks have increased by 350%** according to Google Transparency Report, as the usage of online services has increased.



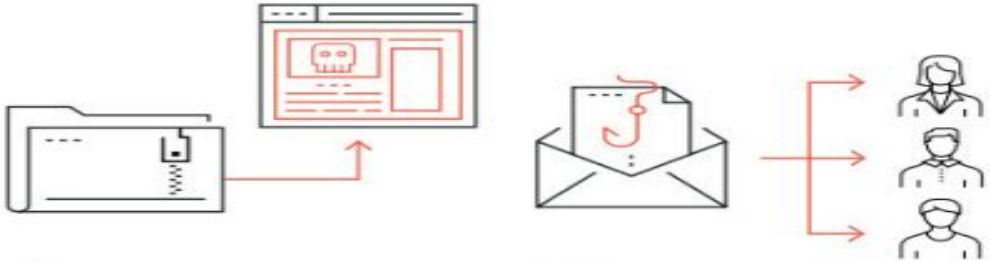
How it works?



1.
The legitimate website is cloned

2.
The login page is changed to point to a credential-stealing script

3.
The modified files are bundled into a zip file to make a phishing kit



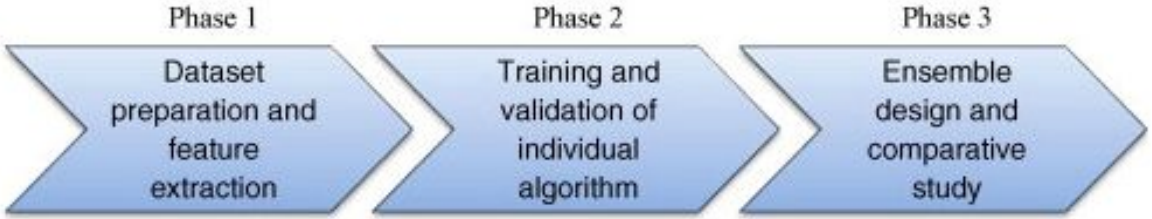
4.
The phishing kit is uploaded to the hacked website, files are unzipped

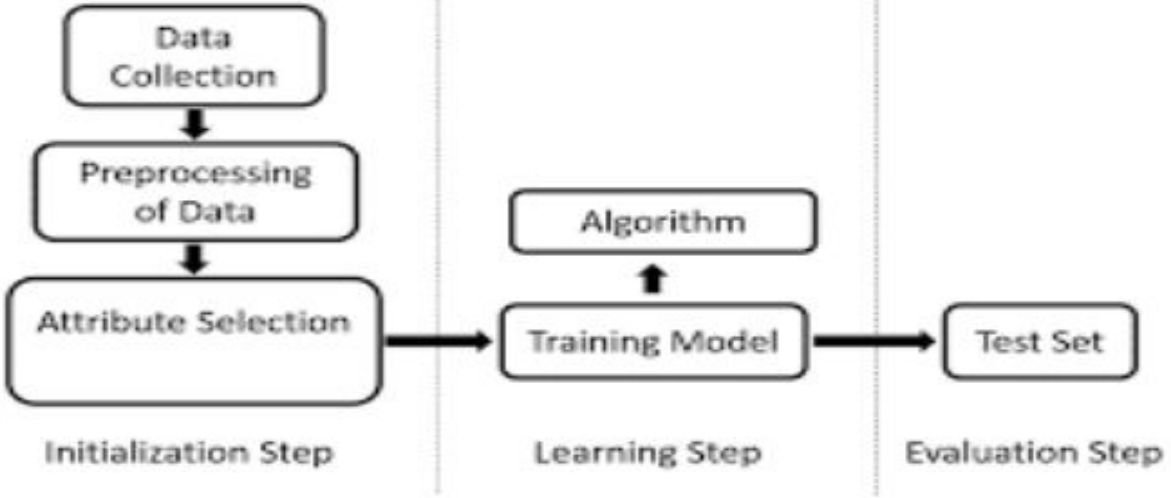
5.
Emails are sent with links pointing to the new spoofed website

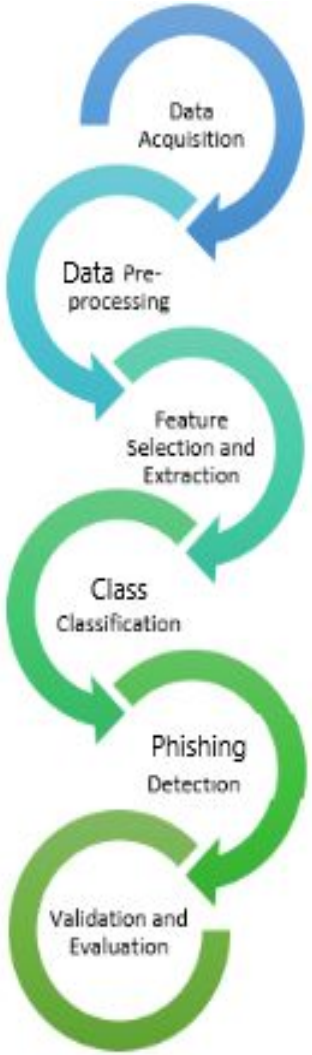
Problem Statement

The aim of our project is to investigate the efficacy of Machine Learning algorithms using Ensemble methods and select a combination of features that would increase the accuracy in detecting the phishing URLs as **‘Legitimate’** and **‘Phishing’**

Schema


Phase 1
Dataset preparation and feature extraction
Phase 2
Training and validation of individual algorithm
Phase 3
Ensemble design and comparative study


Data Collection
Preprocessing of Data
Attribute Selection
Initialization Step
Algorithm
Training Model
Learning Step
Test Set
Evaluation Step

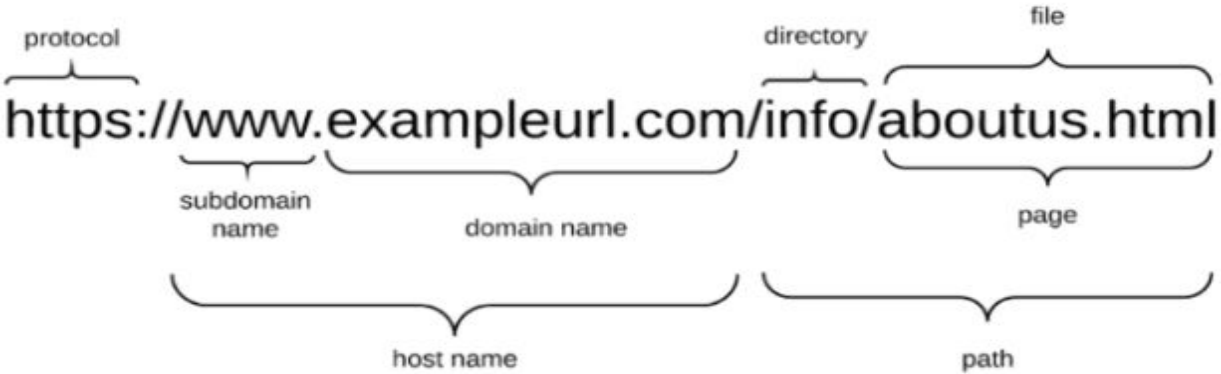

Data Acquisition
Data Pre-processing
Feature Selection and Extraction
Class Classification
Phishing Detection
Validation and Evaluation

About the Data

- We will be using the ‘Phishing website dataset’ from Kaggle, this dataset consists of legitimate and phishing URLs.
- The legitimate websites are taken from **‘Yahoo’** and **‘starting point directory’(Whitelists)** and the phishing websites are collected from **‘Phishtank data archive’(Blacklists)**, where suspicious websites are submitted and verified.
- Our dataset consists of **11055 URLs and 32 features**. There are 6157 legitimate, 4898 phishing websites.

Characteristics of Phishing URLs:

A URL is the **address of the website** on the internet.



- The **protocol** shows how the messages are formatted and transmitted.
- The **domain name** has to be registered at the domain name registrar and can be set only once. The **file and directory** portion gives us the path of the URL, which can be **changed by the attacker**.
- The **subdomain name and the path** are controllable by the attacker and are together called '**Free URL**'.

	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Domain_registeration_length	Request_URL	URL_of_Anchor	SFH	age_of_domain	Links_in_tags
0	2	2	2	2	1	2	2	2	1
1	2	0	1	2	1	0	2	2	2
2	2	2	2	2	1	0	2	1	2
3	2	2	2	1	2	0	2	2	0
4	2	1	1	2	1	0	2	2	0

- **Prefix_Suffix:** prefixes or suffixes are added separated by (-) to the domain name to create a sense of legitimate website. http://www.Confirme-paypal.com/
- **SSLfinal_State:** Presence of HTTPS indicates the legitimacy of a website, but this is not enough, checking the certificate assigned, the certificate issuer, and the certificate age is important(2 years).
- **Domain_registeration_length:** Phishing websites are short lived, they exist for maximum one year.
- **Request_URL:** External objects such as images and videos should be of the same domain.
- **Age_of_domain:** Minimum age of the legitimate domain is 6 months.
- **Links_in_tags:** Legitimate websites use tags to offer metadata about the HTML document.
- **URL_of_Anchor:** It examines if the tags are from a different domain
- **Having_sub_domain:** sub_domain name acts like a domain name.
- **Google_Index:** This feature examines whether a website is in Google's index or not.

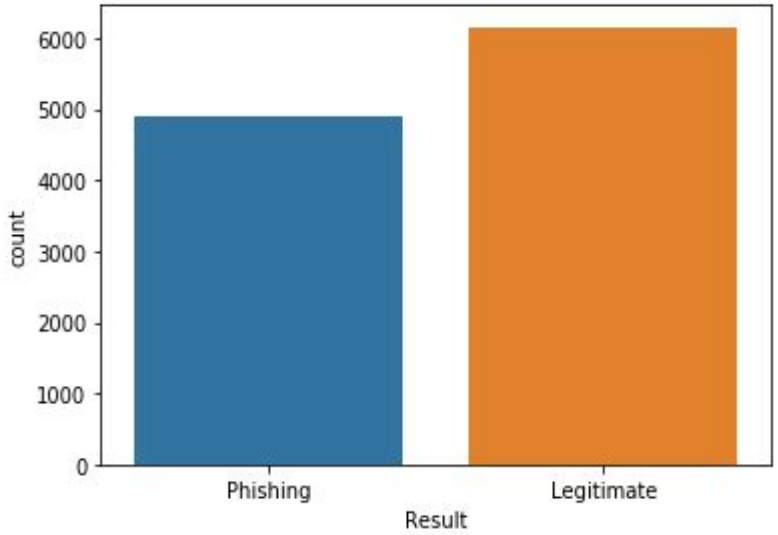
Phase 1
Data Pre-processing

- Checked for **Null values**, there are no missing values in our dataset

```
In [62]: (new_data.isnull()).any()

Out[62]: Prefix_Suffix                 False
         having_Sub_Domain             False
         SSLfinal_State                False
         Domain_registeration_length   False
         Request_URL                   False
         SFH                           False
         age_of_domain                 False
         URL_of_Anchor                 False
         Links_in_tags                 False
         Links_pointing_to_page        False
         Result                        False
         dtype: bool
```

- Checked for **inappropriate values**, since all our features are categorized into 1,-1 and 1,0 and -1, any value other than those will be considered inappropriate. No inappropriate values in our dataset.

- Checked whether the **dataset** classified as legitimate and phishing is **balanced, to avoid bias in predictions.**



- Then we divide the dataset into **training and test sets.**
- Relevant features are from a pool of 32 features selected using feature selection technique.

Feature Selection

- Feature selection is a process of selecting the features or attributes which contributes the most to our variable of interest.
- Irrelevant features can have a negative impact on the performance of the model.
- Feature selection **reduces overfitting**, **increases accuracy** and **reduces training time**.
- Chi-Square test for feature selection:
  - This test finds the features our target variable is highly dependent on.
  - The closer the observed value and the expected value, the smaller the chi-square value and vice-versa.

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where:

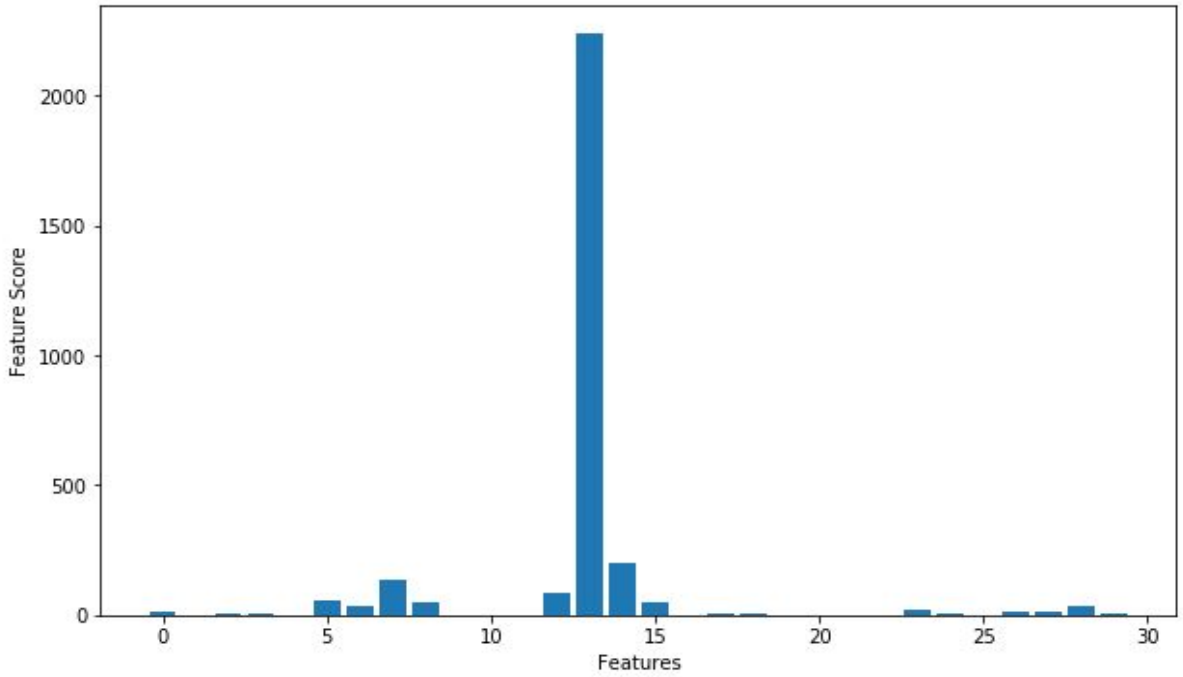
c = degrees of freedom

O = observed value(s)

E = expected value(s)

Feature Selection

- The features with **high chi-square values are selected**.
- Here we have selected 10 features with the highest scores and created a new dataset from the original dataset using only the selected features.
- SSL_Finalstate,URL_ofAnchor and Google_index have high scores



ENSEMBLE METHODS

BAGGING

- Bagging is a parallel ensemble learning technique that aims to improve accuracy, reduce variance and avoids overfitting.
- Given a dataset, first, create multiple bootstrap samples that act as independent datasets.
- Then fit a weak learner (algorithm) to each of these datasets and gather their outputs.
- These outputs are then averaged to reduce the variance.
- For regression algorithms, averaging can be the simple average of the outputs of all the models used.
- For classification algorithms, averaging is done by voting.

ENSEMBLE METHODS

VOTING:

- *Hard voting*- A voting technique where each output class is considered as a vote and the class with the most votes is returned by the ensemble method.
- *Soft voting*- Gather the probability that a class is returned as the output by each model and average it.

ENSEMBLE METHODS

BOOSTING

- Boosting is a sequential ensemble technique which is similar to bagging where it fits weak learners with datasets to make the model better.
- The only difference is that the datasets are fitted one at a time where each weak learner focuses more on the observations that were poorly handled by the previous weak learning algorithm.
- In this way, the final model significantly reduces bias.

IMPLEMENTATION

PHASE 2: **Training and Validation of individual algorithms**

1. Logistic Regression

- Uses a statistical model that uses a logistic function to determine a binary dependent variable.

```
In [24]: from sklearn.linear_model import LogisticRegression

In [25]: logmodel = LogisticRegression()
         classifier = logmodel.fit(X_train,y_train)

In [26]: pred = logmodel.predict(X_test)
```

IMPLEMENTATION

- Determining Precision, Recall and Accuracy

In [27]:

```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,pred))
print(confusion_matrix(y_test,pred))
```

```
              precision    recall  f1-score   support

  Legitimate       0.82      0.86      0.84      1818
    Phishing       0.82      0.77      0.80      1499

    accuracy                           0.82      3317
   macro avg       0.82      0.82      0.82      3317
weighted avg       0.82      0.82      0.82      3317

[[1572  246]
 [ 346 1153]]
```

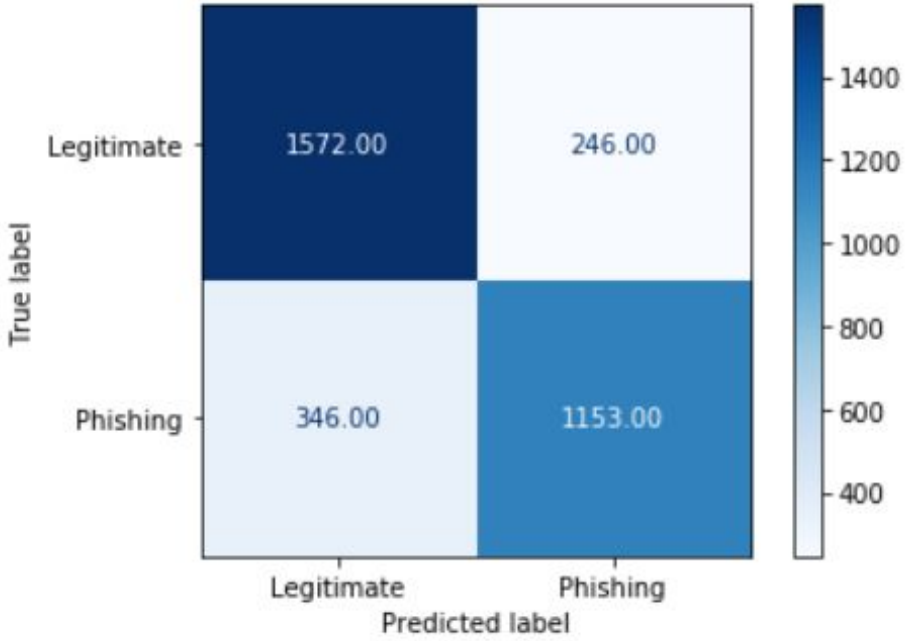
In [47]:

```
from sklearn.metrics import accuracy_score

print(accuracy_score(y_test, pred, normalize=True))
```

```
0.8215254748266506
```

IMPLEMENTATION

- Confusion Matrix



IMPLEMENTATION

PHASE 3: **Ensemble design and comparative study**

2. Random Forest (Bagging Algorithm)

- Operates by constructing multiple decision trees during training and outputs the mean of the classes predicted by each decision tree.

In [31]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

rf = RandomForestClassifier()
```

In [32]:
```
#number of trees
params_rf = {'n_estimators': [50, 100, 200]}
rf_gs = GridSearchCV(rf, params_rf, cv=5)
```

In [ ]:
```
rf_gs.fit(X_train, y_train)
```

In [34]:
```
#best estimator
rf_best = rf_gs.best_estimator_
print(rf_gs.best_params_)
```

```
{'n_estimators': 50}
```

In [35]:
```
rpred = rf_gs.predict(X_test)
```

IMPLEMENTATION

- Determining Precision, Recall and Accuracy

In [36]:
```
print(classification_report(y_test,rpred))
print(confusion_matrix(y_test,rpred))
```

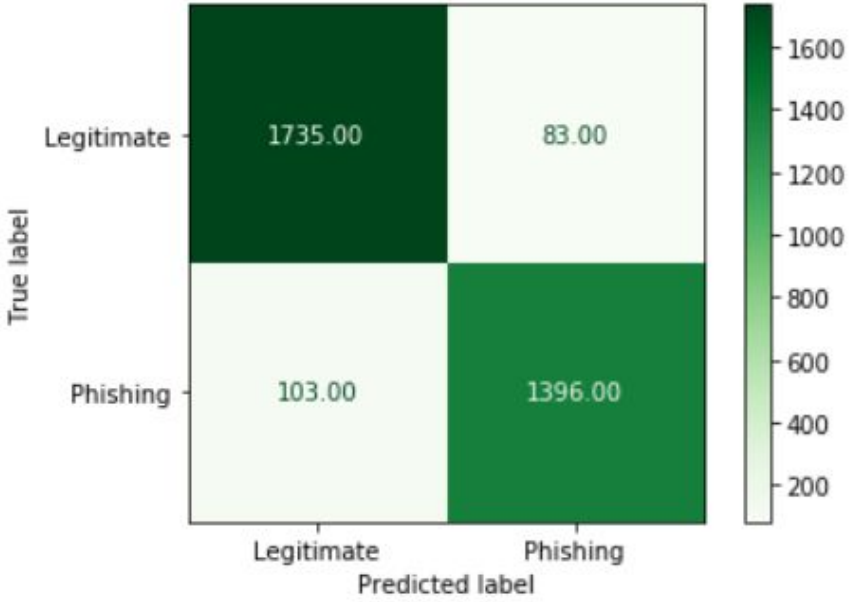
```
              precision    recall  f1-score   support

  Legitimate       0.94      0.95      0.95      1818
    Phishing       0.94      0.93      0.94      1499

    accuracy                           0.94      3317
   macro avg       0.94      0.94      0.94      3317
weighted avg       0.94      0.94      0.94      3317

[[1735   83]
 [ 103 1396]]
```

In [48]:
```
print(accuracy_score(y_test, rpred, normalize=True))
```

```
0.9439252336448598
```

IMPLEMENTATION

- Confusion Matrix



IMPLEMENTATION

3. AdaBoost Classifier (Boosting Algorithm)

- Combines multiple ‘weak classifiers’ into one single ‘strong classifier’.
- Default base classifiers are Decision Trees.

In [42]:
```
from sklearn.ensemble import AdaBoostClassifier
model2 = AdaBoostClassifier(n_estimators = 30, random_state= 100)
```

In [43]:
```
model2.fit(X_train, y_train)
model2_pred= model2.predict(X_test)
```

IMPLEMENTATION

- Determining Precision, Recall and Accuracy

In [44]:
```
print(classification_report(y_test,model2_pred))
print(confusion_matrix(y_test,model2_pred))
```

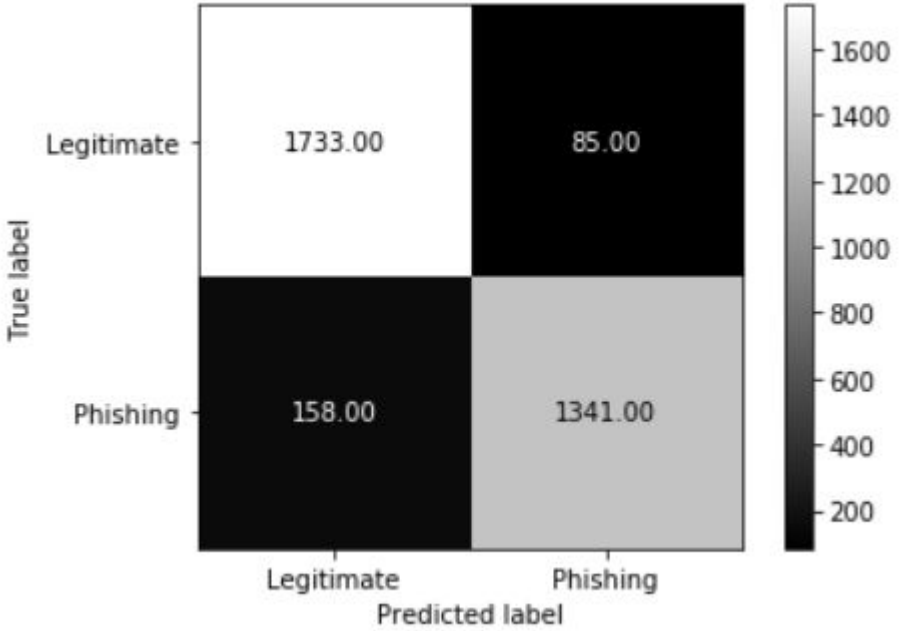
```
              precision    recall  f1-score   support

  Legitimate       0.92      0.95      0.93      1818
    Phishing       0.94      0.89      0.92      1499

    accuracy                           0.93      3317
   macro avg       0.93      0.92      0.93      3317
weighted avg       0.93      0.93      0.93      3317

[[1733   85]
 [ 158 1341]]
```

In [50]:
```
print(accuracy_score(y_test, model2_pred, normalize=True))
```

```
0.9267410310521556
```

IMPLEMENTATION

- Confusion Matrix



IMPLEMENTATION

4. Extreme Gradient Boosting- XGB (Boosting Algorithm)

- Implementation of Gradient boosted Decision Trees designed for speed and performance.

In [38]:
```
from xgboost import XGBClassifier
model = XGBClassifier()
```

In [39]:
```
model.fit(X_train, y_train)
model_pred = model.predict(X_test)
```

IMPLEMENTATION

- Determining Precision, Recall and Accuracy

In [40]:
```
print(classification_report(y_test,model_pred))
print(confusion_matrix(y_test,model_pred))
```

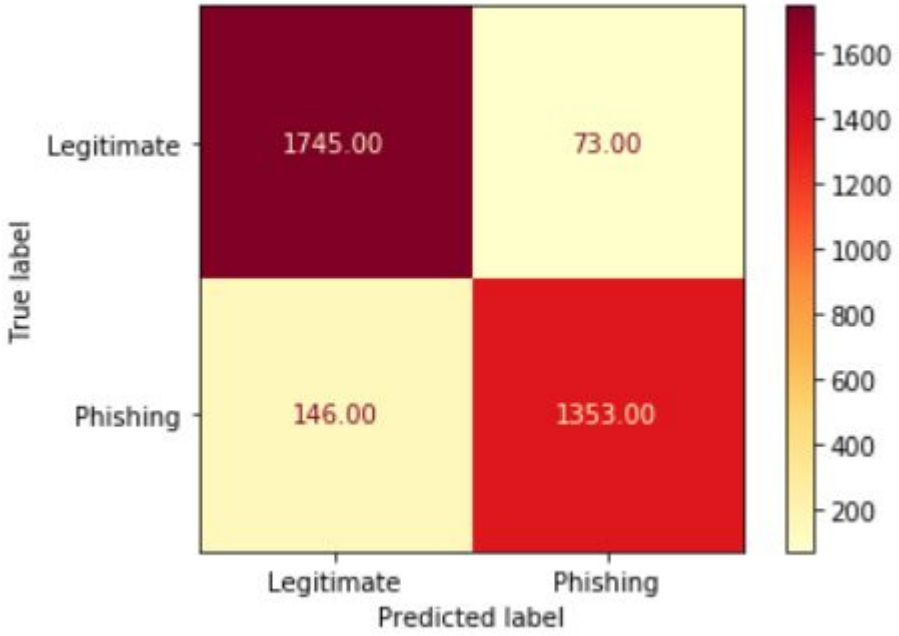
```
              precision    recall  f1-score   support

  Legitimate       0.92      0.96      0.94      1818
    Phishing       0.95      0.90      0.93      1499

    accuracy                           0.93      3317
   macro avg       0.94      0.93      0.93      3317
weighted avg       0.93      0.93      0.93      3317

[[1745   73]
 [ 146 1353]]
```

In [49]:
```
print(accuracy_score(y_test, model_pred, normalize=True))
```

```
0.9339764847753994
```

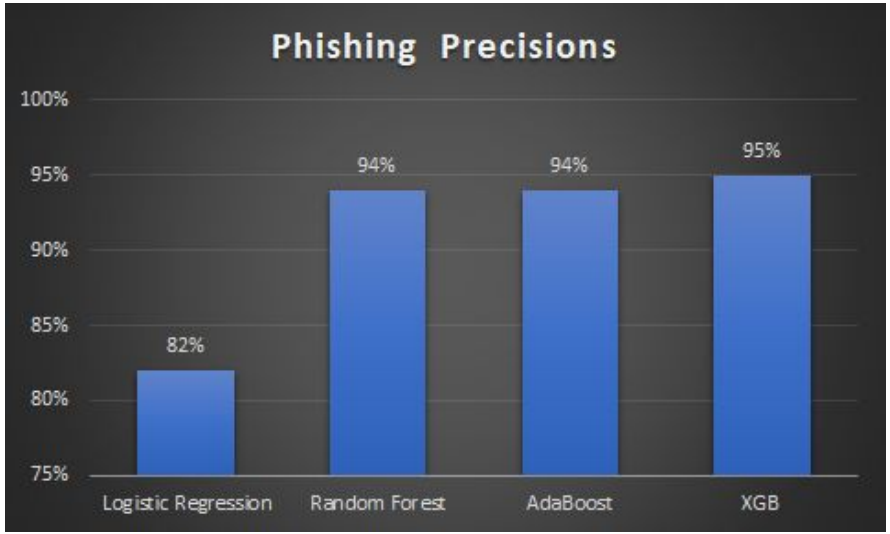
IMPLEMENTATION

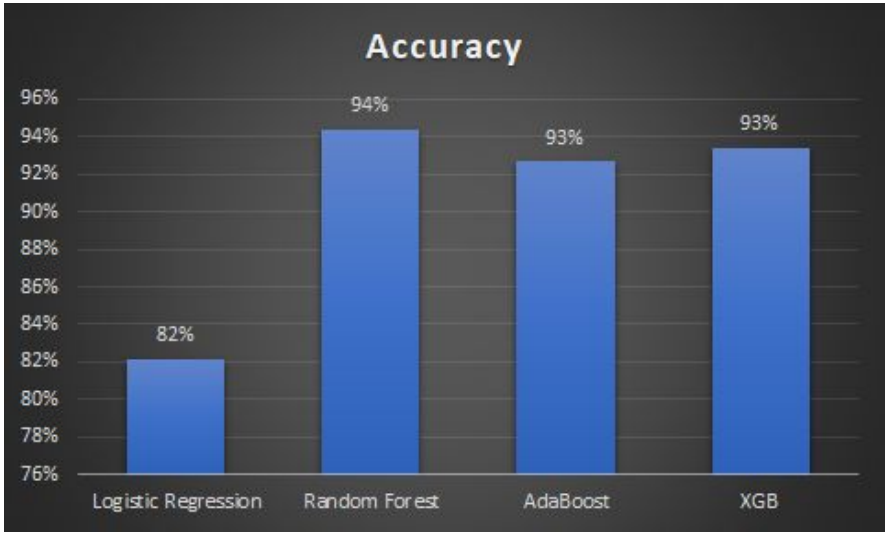
- Confusion Matrix



CONCLUSION

- Comparing the accuracy of the logistic regression model with the accuracies of the ensemble learning algorithms show that the ensemble method drastically improved the accuracy.
- The precision with which the ensemble models detected a phishing website considerably increased as well.





Any Questions?