Phishing Site Analysis ML

Kiet, Axel, Jenna

Introduction

Background	Phishing websites pose a significant threat to internet users by deceiving them into providing sensitive information
Problem Statement	Develop an solution for detecting phishing websites to enhance cybersecurity
Approach	Leverage machine learning algorithms and data visualization techniques to analyze and predict phishing websites

Dataset Overview

We used a Kaggle dataset that includes over 11429 URLs with 87 extracted features. Features are split from three different classes: 56 extracted from the structure/syntax of URLs, 24 from the content of their correspondent pages (ie. web_traffic), and 7 are extracted by querying external services.

The output variable of interest is status (legitimate or phishing), which we re-coded as a binary measure of 0 or 1 for our analysis, and the dataset is precisely balanced between 50% fishing and 50% legitimate URLs.

url	# length_url	# length_hostname	# ip
11429 unique values	12 – 1641	4 – 214	0
http://www.crestonwood.com/router.php	37	19	0
http://shadetreetechnology.com/V4/validation/a111aedc8ae390eabcfa130e041a10a4	77	23	1
https://support-appleld.com.secureupdate.duilawyeryork.com/ap/89e6a3b4b063b8d/?cmd=_update&dispatch=...	126	50	1
http://rgipt.ac.in	18	11	0
http://www.iracing.com/tracks/gateway-motorsports-park/	55	15	0

Steps

Clean and preprocess the dataset for analysis

Data Preprocessing

Use Python libraries (e.g., Matplotlib) to visualize the extracted features and their relationships

Data Visualization

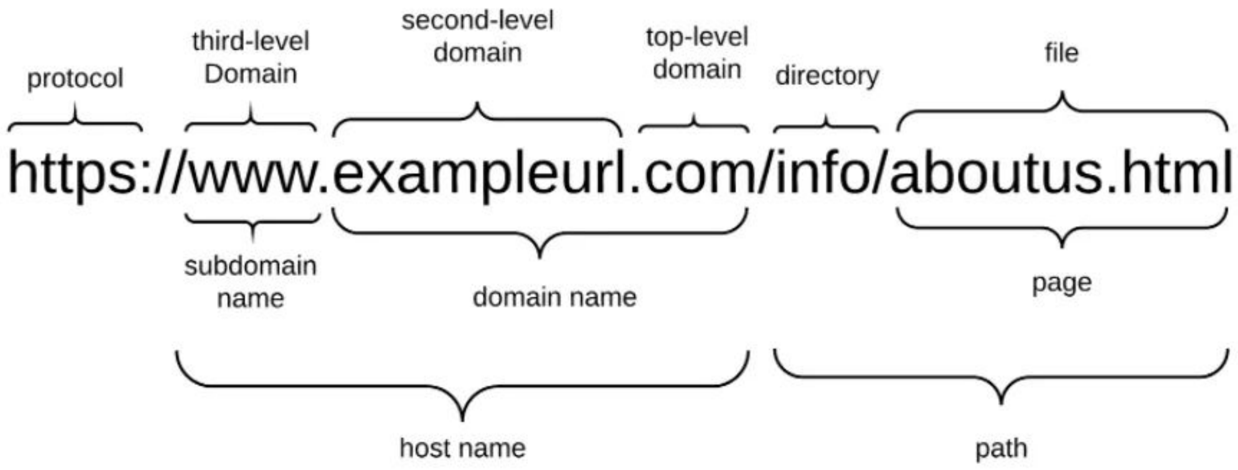
Train ML models such as Logistic Regression, Random Forest, and a Feed Forward neural Network

Model Development

Evaluate model performance using metrics such as accuracy, precision, recall, and F1-score

Model Evaluation

URL Overview



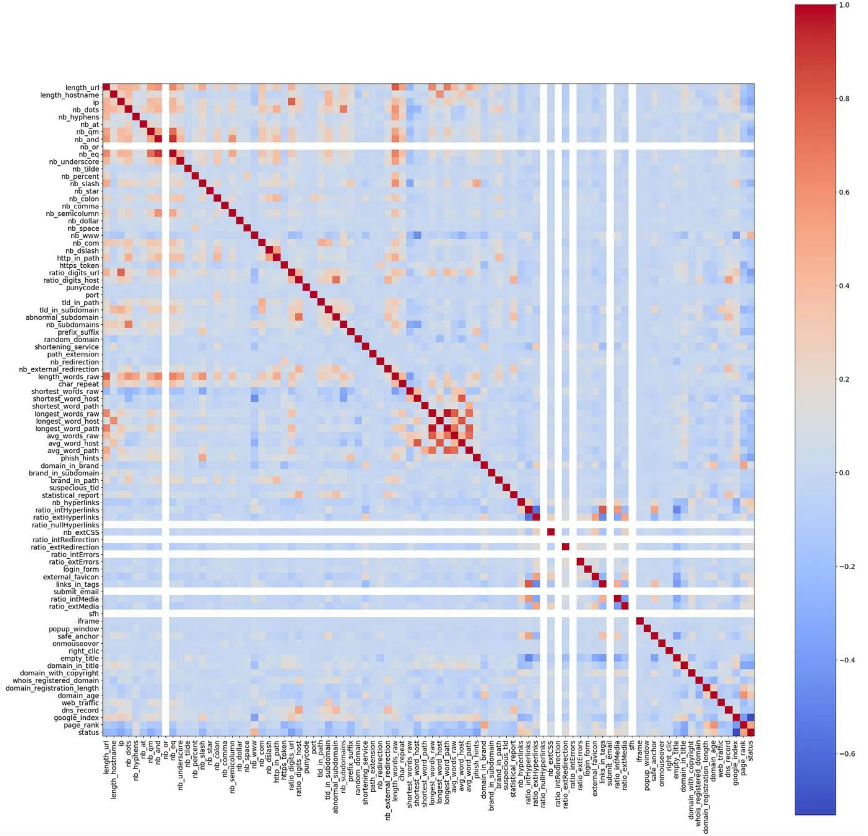
The subdomain name and **path** are fully controlled by the website creator and can be anything they want, even if it’s misleading or deceptive

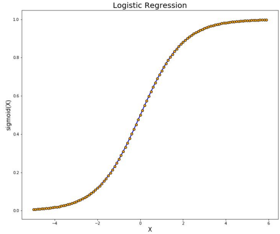
For example, a phisher could create a URL with a subdomain name like “secure.chase.com” to make it appear legitimate and trick users into providing sensitive information

Data Visualization

We plotted a **heat map** of all relevant variables so see which features most strongly correlated with “status” (our output variable)

The **correlation matrix** is useful for identifying which variables are strongly related to the target variable (in this case, the ‘status’ column), as well as for identifying multicollinearity between independent variables.





Logistic Regression model

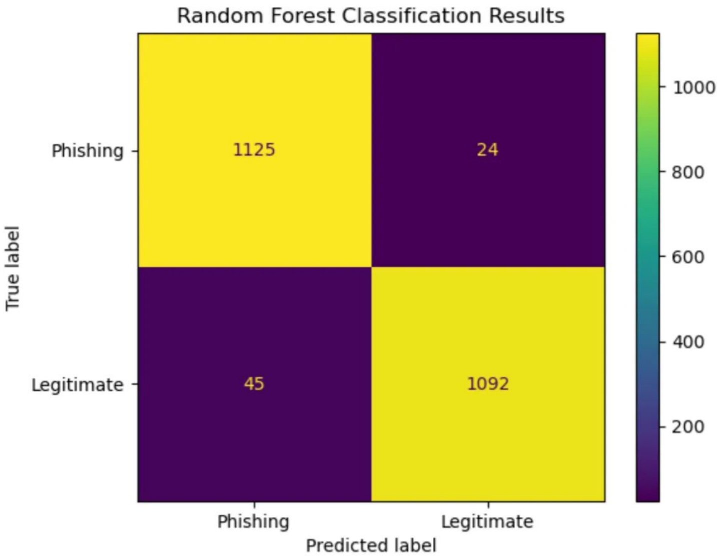
The Logistic Regression model works by modeling the relationship between the independent variables and the log-odds of the dependent variable. For this project, it is served as a baseline model. If more complex models like random forests and neural networks do not perform significantly better than logistic regression, it can indicate that the additional complexity is not needed.

```
Accuracy Score: 0.8009623797025371
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.80      0.80      1149
           1       0.80      0.80      0.80      1137

    accuracy                           0.80      2286
   macro avg       0.80      0.80      0.80      2286
weighted avg       0.80      0.80      0.80      2286

Confusion Matrix:
 [[919 230]
 [225 912]]
```

Looking at the confusion matrix, it misclassified 230 legitimate websites as phishing (false positives) and 225 phishing websites as legitimate (false negatives). Overall, the baseline model performed well with balanced precision and recall scores for both classes.

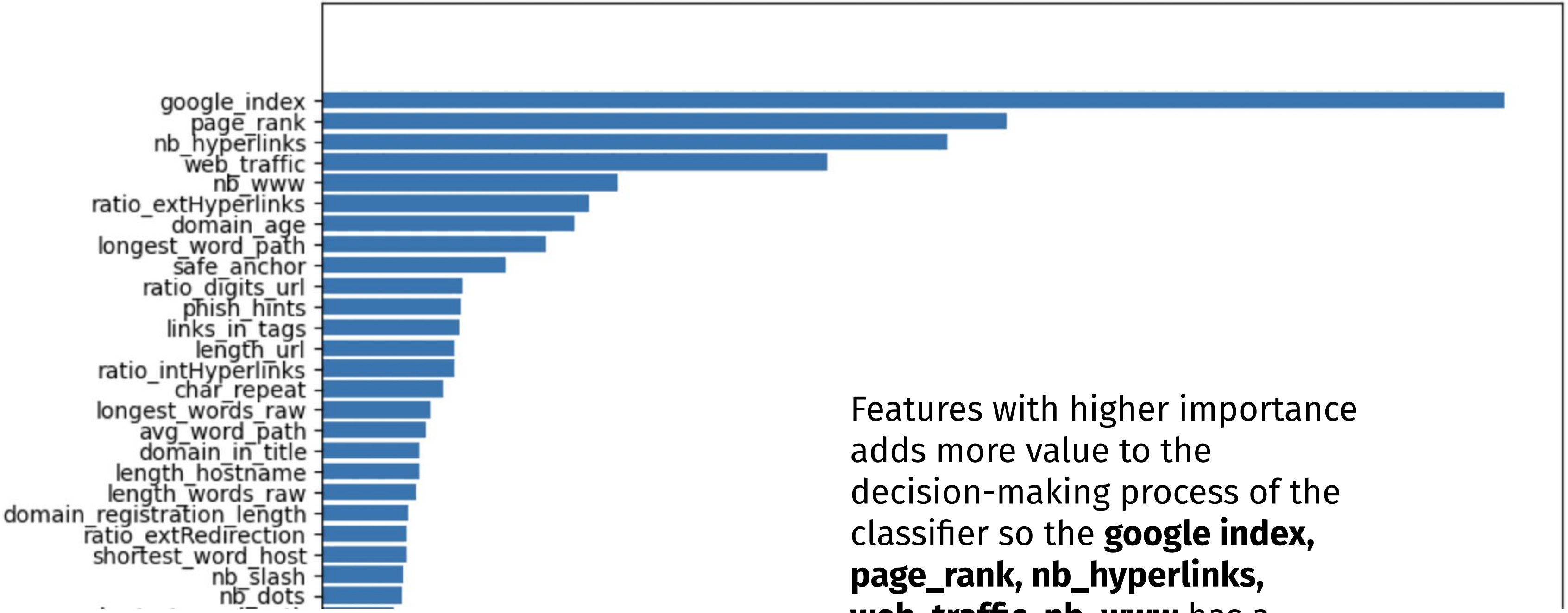
Random Forest Classifier Model

The Random Forest Classifier model is an ensemble learning algorithm that works by constructing and combining multiple decision trees to improve the accuracy and stability of the predictions. The model is a powerful and effective way for complex classification problems, as it is known for its robustness against overfitting but also the ability to capture non-linear relationships between the independent and dependent variables.




```
Accuracy:  0.9698162729658792
Precision:  0.978494623655914
Recall:  0.9604221635883905
F1-score:  0.9693741677762983
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      1149
           1       0.98      0.96      0.97      1137

    accuracy                           0.97      2286
   macro avg       0.97      0.97      0.97      2286
weighted avg       0.97      0.97      0.97      2286
```

Feature Importance

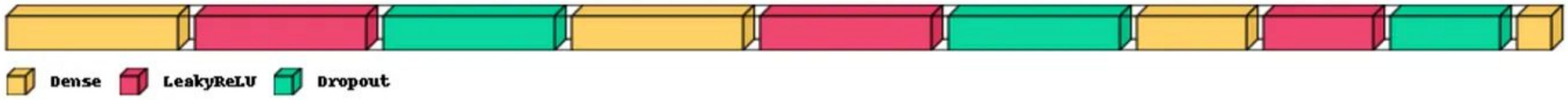


Features with higher importance adds more value to the decision-making process of the classifier so the **google index, page_rank, nb_hyperlinks, web_traffic, nb_www** has a stronger influence on the final output

Feed Forward Neural Network

The Feedforward Neural Network is a powerful artificial neural network that can learn complex patterns in the data and is capable of capturing non-linear relationships between the independent and dependent variables. It works by passing information forward through multiple layers of interconnected nodes

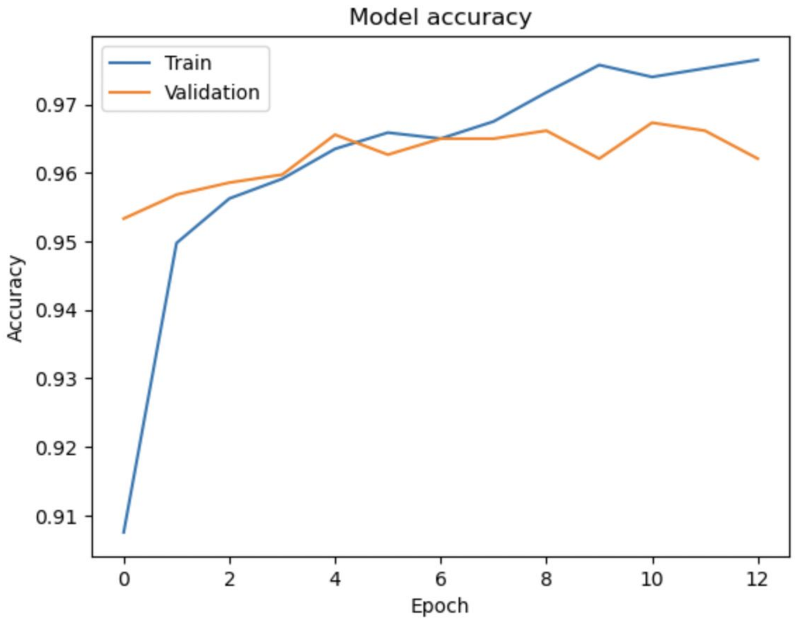
```
model = Sequential([
    # Input layer
    Dense(256, input_shape=(X_train.shape[1],)),
    LeakyReLU(alpha=0.01),
    Dropout(0.2),

    # Hidden layers
    Dense(128),
    LeakyReLU(alpha=0.01),
    Dropout(0.2),

    Dense(64),
    LeakyReLU(alpha=0.01),
    Dropout(0.2),

    # Output layer
    Dense(1, activation='sigmoid')
])
```

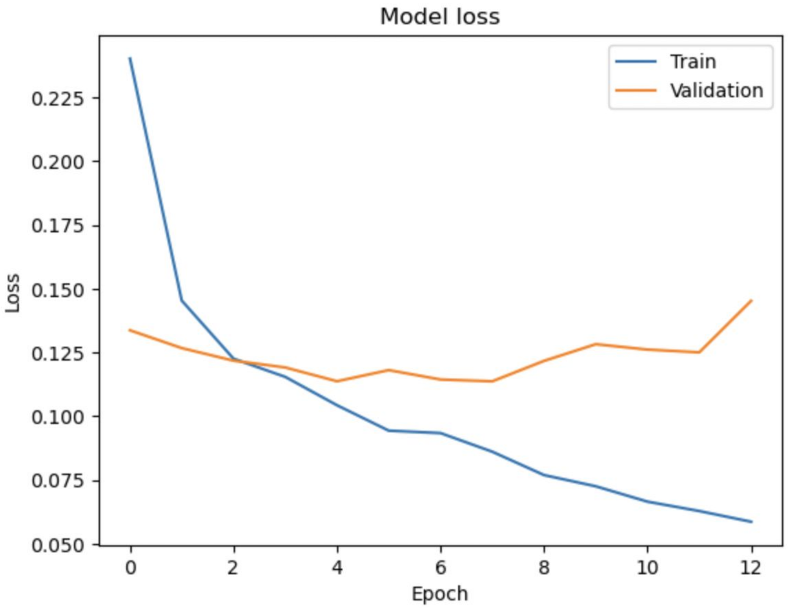


Feed Forward Neural Network

The Feedforward model shows high accuracy, precision, recall, and F1-scores (96%+). This means the model has a good balance between making false positive and false negative predictions!

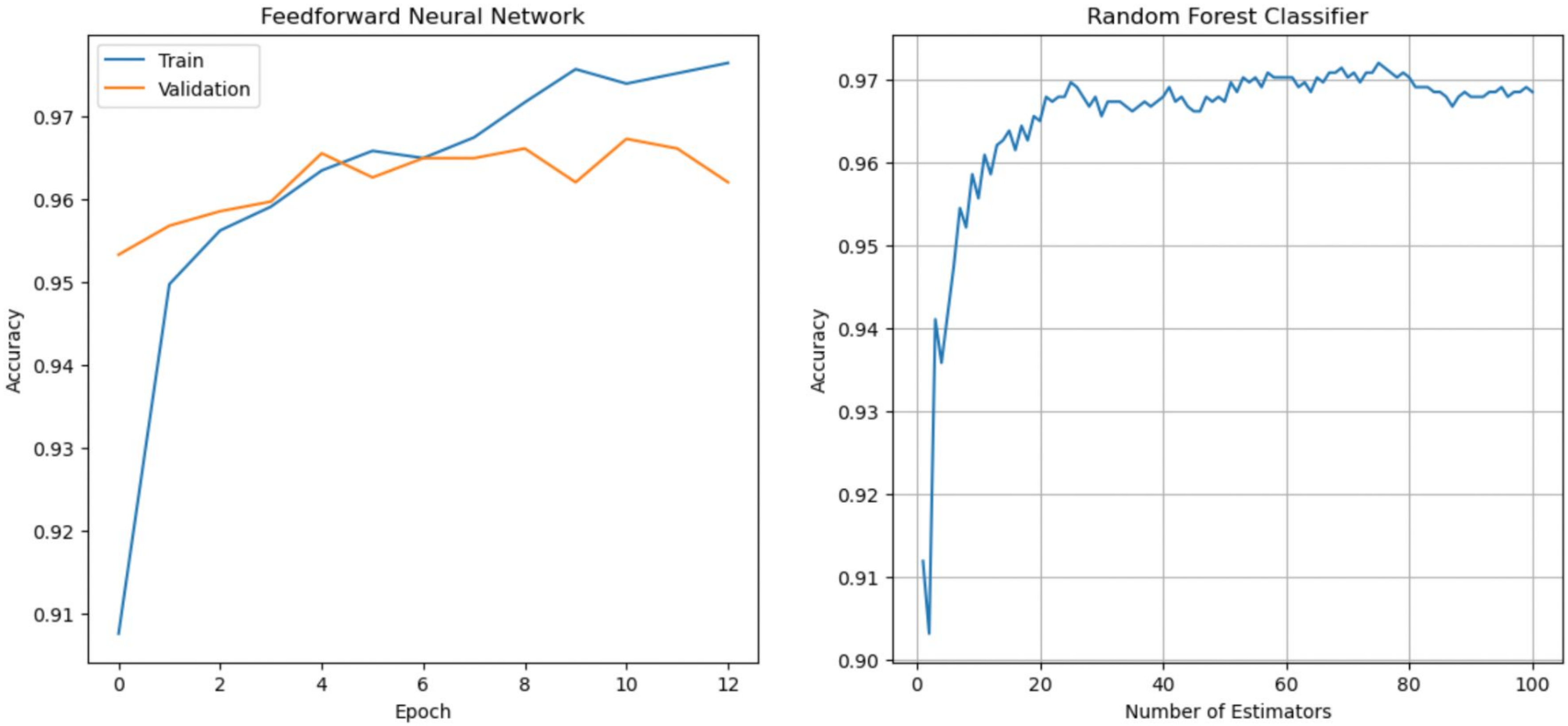
```
54/54 [==============================] - 0s 1ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       857
           1       0.97      0.96      0.96       857

    accuracy                           0.96      1714
   macro avg       0.97      0.96      0.96      1714
weighted avg       0.97      0.96      0.96      1714

Confusion Matrix:
 [[830  27]
 [ 33 824]]
Accuracy Score: 0.9649941656942824
```





Both graphs stop at **epoch 12**, it means that the training process stopped early due to the **EarlyStopping** callback. **EarlyStopping** is a regularization technique used to prevent overfitting during the training process.

The fact that the training stopped at **epoch 12** indicates that the validation loss stopped improving after 7 epochs and continued to stagnate or degrade for the next 5 epochs. Once the 5-epoch patience threshold was reached, **the training stopped**, and the model's weights were restored to the best performing iteration (as specified by restore_best_weights=True).

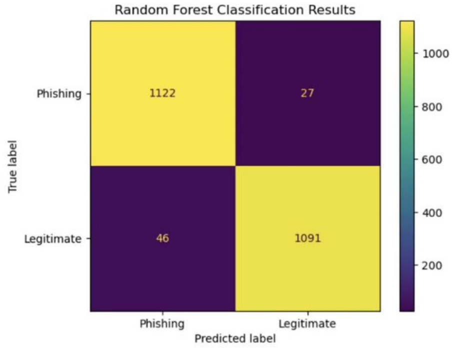

Feedforward Neural Network
Train
Validation
Accuracy
Epoch
0.91
0.92
0.93
0.94
0.95
0.96
0.97
0
2
4
6
8
10
12
Random Forest Classifier
Accuracy
Number of Estimators
0.90
0.91
0.92
0.93
0.94
0.95
0.96
0.97
0
20
40
60
80
100

More Work

Using the random forest model, we selected only the features that have importance higher than the median from both the training and testing sets to recalculate predictions using a random forest classifier to see if the results differ.

```
from sklearn.ensemble import RandomForestClassifier
import numpy as np

RF2 = RandomForestClassifier(n_estimators=100, random_state=0)

importance = None
mask = None

RF2.fit(X_train, y_train)
importance = RF2.feature_importances_

median = np.median(importance)
mask = importance > median
```

```
X_train_selected = X_train[:,mask]
X_test_selected = X_test[:,mask]

RF3 = RandomForestClassifier(n_estimators=100, random_state=0)
RF3.fit(X_train_selected, y_train)

y_pred3 = RF3.predict(X_test_selected)

cm3 = confusion_matrix(y_test, y_pred3, labels=RF3.classes_)
disp3 = ConfusionMatrixDisplay(confusion_matrix=cm3,display_labels=["Phishing", "Legitimate"])
disp3.plot()
plt.title("Random Forest Classification Results")
plt.show()
```




```
Accuracy: 0.968066491688539
Precision: 0.9758497316636852
Recall: 0.9595426561125769
F1-score: 0.9676274944567629
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      1149
           1       0.98      0.96      0.97      1137

    accuracy                           0.97      2286
   macro avg       0.97      0.97      0.97      2286
weighted avg       0.97      0.97      0.97      2286
```

We concluded that by selecting only the most important features, we still achieve a very similar performance compared to using all features.

Model Evaluations

Logistic Regression	80% Accuracy	80% Precision
Random Forest	97% Accuracy	98% Precision
Important Feature RF	97% Accuracy	98% Precision
Feed Forward	96% Accuracy	97% Precision

Future work

```
# Splitting the dataset into train, validation, and test sets
from sklearn.model_selection import train_test_split

X = df[feature_columns]
y = df['status']

# First, split into train and (validation + test) sets
X_train, X_val_test, y_train, y_val_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Then, split the (validation + test) set into validation and test sets
X_val, X_test, y_val, y_test = train_test_split(X_val_test, y_val_test, test_size=0.5, random_state=42)
```

Spilt up the validation data and train the models with that data and graph it out

After thoughts

These models have the potential to provide a powerful tool in the fight against cybercrime.

The use of machine learning models in detecting and classifying phishing websites is a promising area of research and development.

And as the threat of cyber attacks continues to grow, the development of more advanced and accurate models can help protect individuals and organizations from harm.

Check out the medium blog & code!

Axeltfm
May 1 · 6 min read · Listen

Phishing Site Detection ML

The incorporation of machine learning to detect phishing websites

by Kiet Ha, Jenna Kampe, Axel Motulsky

https://medium.com/phishing-site-detection/phishing-site-detection-353df9c3fe40

In [2]:
```
import pandas as pd

df = pd.read_csv("dataset_phishing.csv")
copy = df.copy()
df.head()
```

Out[2]:

	url	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	...	domain_in_title
0	http://www.crestonwood.com/router.php	37	19	0	3	0	0	0	0	0	...	0
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1	0	0	0	0	0	...	1
2	https://support-appleld.com.secureupdate.duila...	126	50	1	4	1	0	1	2	0	...	1
3	http://rgipt.ac.in	18	11	0	2	0	0	0	0	0	...	1
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	2	0	0	0	0	...	0

5 rows × 89 columns

In [3]:
```
# Encoding non-numeric features

encoding1 = {'legitimate': 1, 'phishing': 0}
df['status'] = df['status'].replace(encoding1)

# Remove url from df

df = df.drop('url', axis = 1)

df.head()
```

Out[3]:

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	nb_eq	...	domain_in_title	domain_with_copyright	whois_registe
0	37	19	0	3	0	0	0	0	0	0	...	0	1	
1	77	23	1	1	0	0	0	0	0	0	...	1	0	
2	126	50	1	4	1	0	1	2	0	3	...	1	0	
3	18	11	0	2	0	0	0	0	0	0	...	1	0	
4	55	15	0	2	2	0	0	0	0	0	...	0	1	

5 rows × 88 columns

https://github.com/jennakampe/Big-Data-Final/blob/main/Final.ipynb