UNIVERSITI MALAYA

Presented by Group 6

SIT3017

PHISHING

DECISION TREE & RANDOM FOREST CLASSIFICATION

Lecturer: PROFESOR DR. IBRAHIM BIN MOHAMED

No.	Name	Matric No.
1	NUR NAJLA NABILA BT AZMAN	U2001004
2	MUHAMMAD IMRAN BIN MOHD ISA	U2000717
3	CHIN YEE WEI	U2103394
4	KOO HOONG KHEN	U2103676
5	PANG WILSON	S2132422

Intro

BACKGROUND

Phishing is a type of cyberattack that involves pretending to be a legitimate company website in order to obtain personal data from users, including passwords, social security numbers, and usernames. A typical phishing assault starts with the victims receiving an email that appears to be from a legitimate company. These emails request that recipients update their details by clicking on a URL link in the message. We get the data from https://www.kaggle.com/datasets/ahmednour/website-phishing-data-set/data.

DATASET

Intro

	A	B	C	D	E	F	G	H	I	J
1	SFH	popUpWidnow	SSLfinal_State	Request_URL	URL_of_Anchor	web_traffic	URL_Length	age_of_domain	having_IP_Address	Result
2	1	-1	1	-1	-1	1	1	1	0	0
3	-1	-1	-1	-1	-1	0	1	1	1	1
4	1	-1	0	0	-1	0	-1	1	0	1
5	1	0	1	-1	-1	0	1	1	0	0
6	-1	-1	1	-1	0	0	-1	1	0	1
7	-1	-1	1	-1	-1	1	0	-1	0	1
8	1	-1	0	1	-1	0	0	1	0	-1
9	1	0	1	1	0	0	0	1	1	-1
10	-1	-1	0	-1	-1	-1	-1	1	0	0
11	-1	0	-1	-1	1	1	0	-1	0	1
12	-1	-1	0	-1	-1	1	-1	-1	0	1

ATTRIBUTES

SFH	popUpWindow	SSLfinal_State	Request_URL	URL_of_Anchor
Server From Handler (SFH) SFH is an empty string or the website terms "about: blank"	A popup is a window that automatically appears ("pops up") on a website while the user is browsing, often by itself.	a type of cyberattack where attackers impersonate a trusted website that uses the HTTPS protocol to deceive victims into providing sensitive information	consists of an HTTP method, a base URL, and a resource URI. The request header also includes parameters such as the content type and authorization information	a web page element that links to another location on the same page

web_traffic	URL_Length	age_of_domain	having_IP_Address	Result
Website traffic refers to web users who visit a website	URL length measures the number of characters that a URL consists of.	the time of a domain's existence, which is counted from the moment of its first registration	An Internet Protocol (IP) address is the unique identifying number assigned to every device connected to the internet	"1": Legitimate "0": Suspicious "-1": Phishy

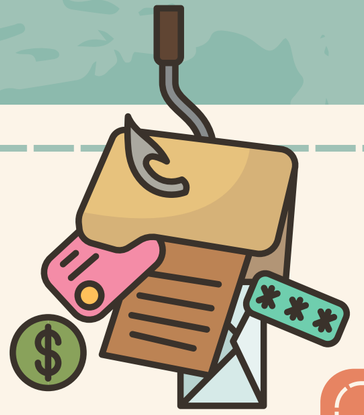
RULES OF CLASSIFICATION

	Legit (“1”)	Suspicious (“0”)	Phishy (“-1”)
SFH	else	redirects to different domain	“about : blank” or empty
popUpWindow	else	right-click showing alert	right-click disabled
SSLfinal_State	use https & trusted issuer & age at least 2 years	use https & issuer is not trusted	else
Request_URL	less than 22%	between 22% and 61%	greater than 61%
URL_of_Anchor	less than 31%	between 31% and 67%	greater than 67%

RULES OF CLASSIFICATION

	Legit (“1”)	Suspicious (“0”)	Phishy (“-1”)
web_traffic	rank less than 150,000	rank greater than 150,000	else
URL_Length	less than 54 characters	between 54 and 75 characters	more than 75 characters
age_of_domain	else	-	less than 6 months
having_IP_Address	else	-	IP Address exists in URL

IMPORTING PACKAGES



Packages

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn import metrics, tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
 recall_score, ConfusionMatrixDisplay
```

LOADING DATA

data = pd.read_csv("**C:/Users/acer/OneDrive - Universiti Malaya/ Desktop/ Phishing.csv**")

ftnames = ["SFH", "popUpWindow", "SSLfinal_State", "Request_URL", "URL_of_Anchor", "web_traffic", "URL_Length", "age_of_domain", "having_IP_Address","Result"]

classnames = ["-1","0","1"]

print(data.head())

	SFH	popUpWindow	SSLfinal_State	...	age_of_domain	having_IP_Address	Result
0	1	-1	1	...	1	0	0
1	-1	-1	-1	...	1	1	1
2	1	-1	0	...	1	0	1
3	1	0	1	...	1	0	0
4	-1	-1	1	...	1	0	1

[5 rows x 10 columns]

EXPLORATORY DATA ANALYSIS

EDA

EDA

1 Examine the data

```
a=len(data[data.Result==0])
b=len(data[data.Result==-1])
c=len(data[data.Result==1])
print("Count of Suspicious Websites = ", a)
print("Count of Phishy Websites = ", b)
print("Count of Legitimate Websites = ", c)
```


```
Count of Suspicious Websites =  103
Count of Phishy Websites =  702
Count of Legitimate Websites =  548
```

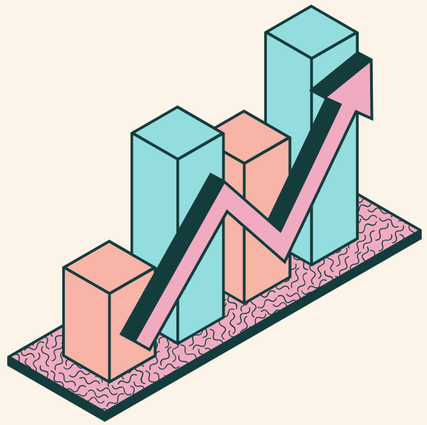
EDA

2 Barplot

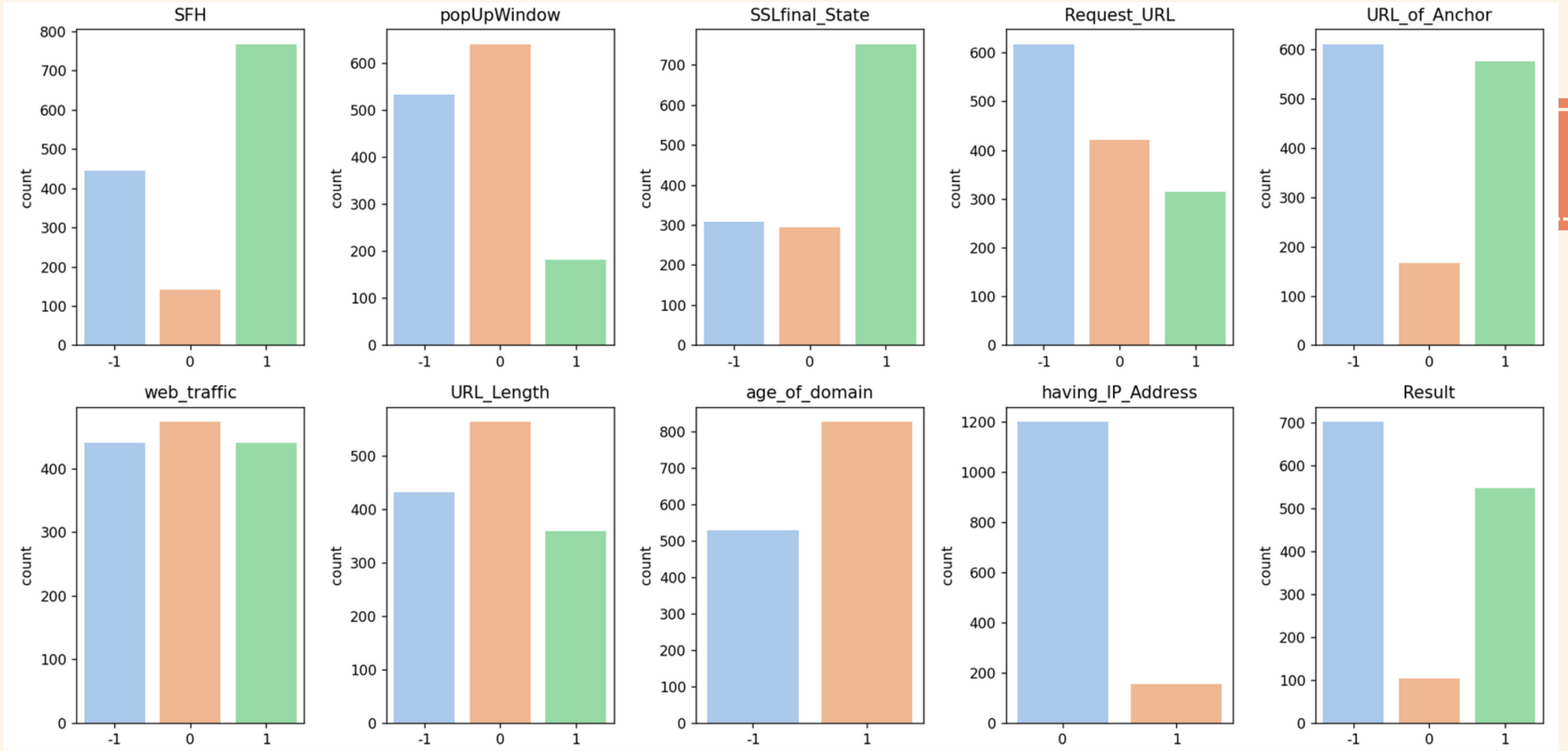
```
fig = plt.figure(figsize = (15,10))
for i in range(1,11):
    axi = fig.add_subplot(2,5,i)
    sns.countplot(data = data, x = ftnames[i-1], ax = axi, palette = 'pastel')
    axi.set_title(ftnames[i-1])
    axi.set_xlabel(" ")
plt.tight_layout()
plt.show()
```

3 Correlation Matrix

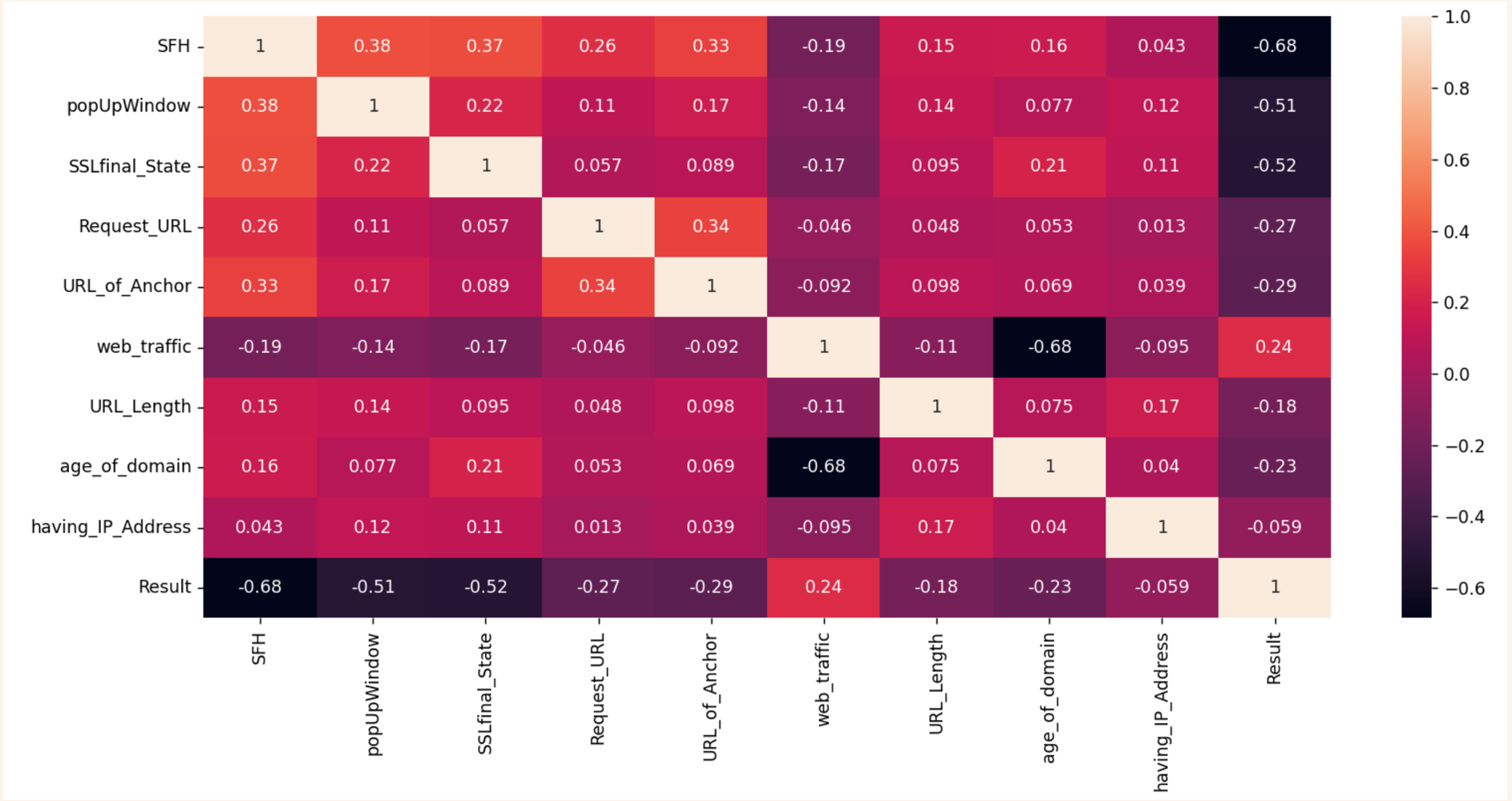
```
sns.heatmap(data.corr(), annot = True)
plt.tight_layout()
plt.show()
```

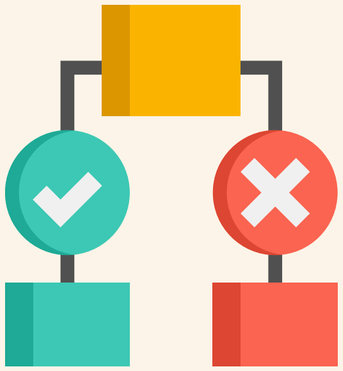


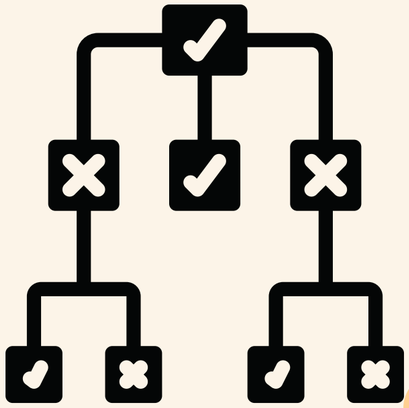
EDA


SFH
popUpWindow
SSLfinal_State
Request_URL
URL_of_Anchor
web_traffic
URL_Length
age_of_domain
having_IP_Address
Result
count

EDA


SFH 1 0.38 0.37 0.26 0.33 -0.19 0.15 0.16 0.043 -0.68
popUpWindow 0.38 1 0.22 0.11 0.17 -0.14 0.14 0.077 0.12 -0.51
SSLfinal_State 0.37 0.22 1 0.057 0.089 -0.17 0.095 0.21 0.11 -0.52
Request_URL 0.26 0.11 0.057 1 0.34 -0.046 0.048 0.053 0.013 -0.27
URL_of_Anchor 0.33 0.17 0.089 0.34 1 -0.092 0.098 0.069 0.039 -0.29
web_traffic -0.19 -0.14 -0.17 -0.046 -0.092 1 -0.11 -0.68 -0.095 0.24
URL_Length 0.15 0.14 0.095 0.048 0.098 -0.11 1 0.075 0.17 -0.18
age_of_domain 0.16 0.077 0.21 0.053 0.069 -0.68 0.075 1 0.04 -0.23
having_IP_Address 0.043 0.12 0.11 0.013 0.039 -0.095 0.17 0.04 1 -0.059
Result -0.68 -0.51 -0.52 -0.27 -0.29 0.24 -0.18 -0.23 -0.059 1
SFH
popUpWindow
SSLfinal_State
Request_URL
URL_of_Anchor
web_traffic
URL_Length
age_of_domain
having_IP_Address
Result
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6





DECISION TREE

Decision Tree

1 Split data set in features and target variable

```
x = data.drop('Result',axis=1).values
y = data['Result'].values
```

2 Split data set into training set and test set

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random
_state=6)
print("Training set has {} samples.".format(x_train.shape[0]))
print("Testing set has {} samples.".format(x_test.shape[0]))
```

```
Training set has 1082 samples.
Testing set has 271 samples.
```

3 **Create Decision Tree Classifer object**

```
clf = DecisionTreeClassifier(random_state=6)
```

4 **Train Decision Tree Classifer**

```
clf = clf.fit(x_train,y_train)
```

5 **Predict the response for test dataset**

```
y_pred = clf.predict(x_test)
```

6 **Model Accuracy**

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8745387453874539
```

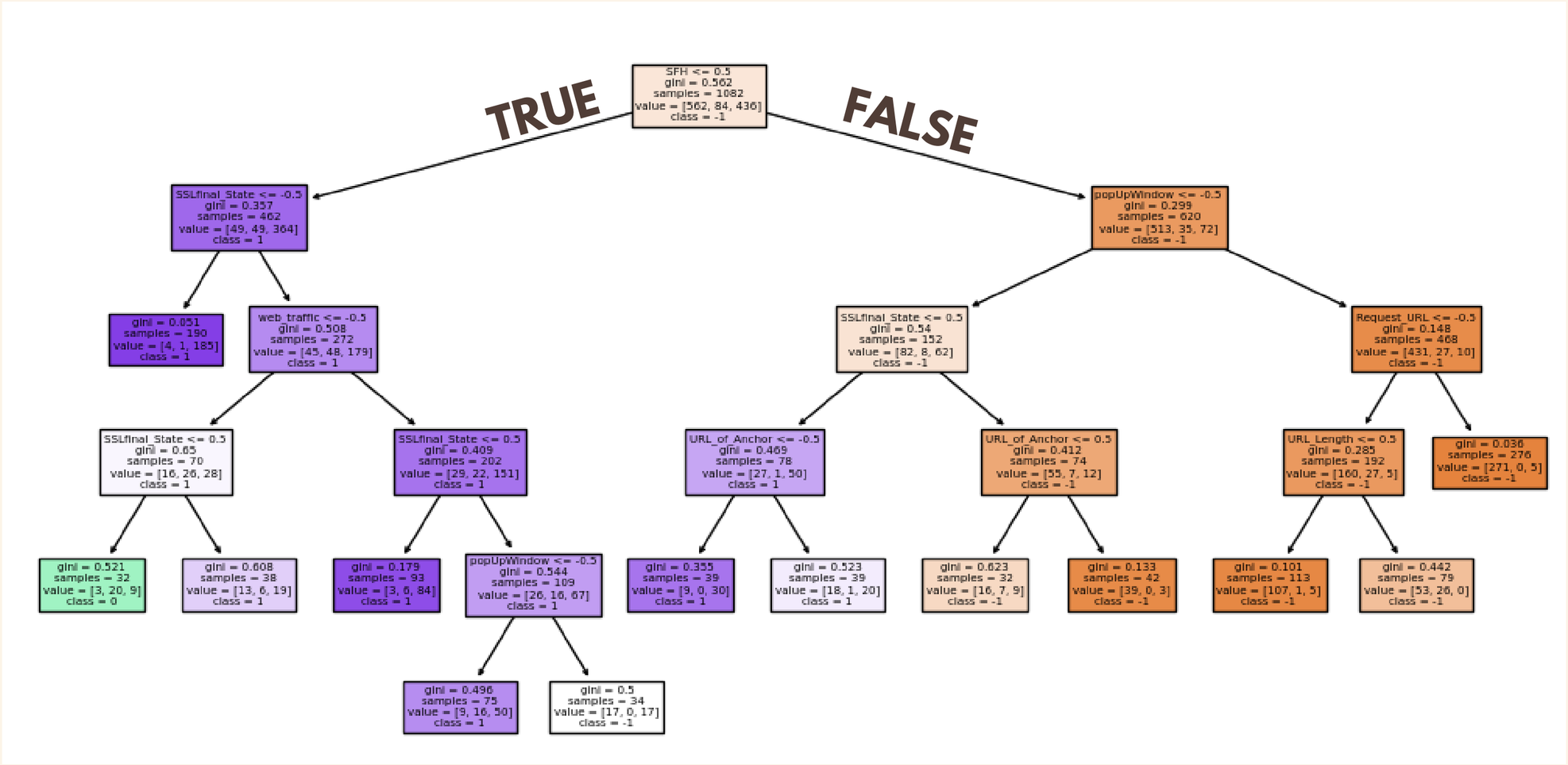
7 Prune Decision Tree

```
dt_param_dist = {'max_leaf_nodes': range(2,100),
                 'max_depth': range(1,20),
                 'min_samples_leaf': list((30,100,300,int(0.05*(a+b+c))))}
dt_rand_search = RandomizedSearchCV(clf, param_distributions = dt_param_dist,
n_iter=5, cv=5, random_state=6)
dt_rand_search.fit(x_train, y_train)
best_dt = dt_rand_search.best_estimator_
print('Best decision tree hyperparameters:',  dt_rand_search.best_params_)
```

```
Best decision tree hyperparameters: {'min_samples_leaf': 30, 'max_leaf_nodes': 13, 'max_depth': 6}
```

8 Visualizing the best decision tree

```
best_dt.fit(x_train,y_train)
fig = plt.figure(figsize=(12,30))
a = tree.plot_tree(best_dt, feature_names = ftnames, class_names = classnames,
filled = True)
plt.show()
```


TRUE
FALSE
SFH <= 0.5
gini = 0.562
samples = 1082
value = [562, 84, 436]
class = -1
SSLfinal_State <= -0.5
gini = 0.357
samples = 462
value = [49, 49, 364]
class = 1
popUpWindow <= -0.5
gini = 0.299
samples = 620
value = [513, 35, 72]
class = -1
gini = 0.051
samples = 190
value = [4, 1, 185]
class = 1
web_traffic <= -0.5
gini = 0.508
samples = 272
value = [45, 48, 179]
class = 1
SSLfinal_State <= 0.5
gini = 0.54
samples = 152
value = [82, 8, 62]
class = -1
Request_URL <= -0.5
gini = 0.148
samples = 468
value = [431, 27, 10]
class = -1
SSLfinal_State <= 0.5
gini = 0.65
samples = 70
value = [16, 26, 28]
class = 1
SSLfinal_State <= 0.5
gini = 0.409
samples = 202
value = [29, 22, 151]
class = 1
URL_of_Anchor <= -0.5
gini = 0.469
samples = 78
value = [27, 1, 50]
class = 1
URL_of_Anchor <= 0.5
gini = 0.412
samples = 74
value = [55, 7, 12]
class = -1
URL_Length <= 0.5
gini = 0.285
samples = 192
value = [160, 27, 5]
class = -1
gini = 0.036
samples = 276
value = [271, 0, 5]
class = -1
gini = 0.521
samples = 32
value = [3, 20, 9]
class = 0
gini = 0.608
samples = 38
value = [13, 6, 19]
class = 1
gini = 0.179
samples = 93
value = [3, 6, 84]
class = 1
popUpWindow <= -0.5
gini = 0.544
samples = 109
value = [26, 16, 67]
class = 1
gini = 0.355
samples = 39
value = [9, 0, 30]
class = 1
gini = 0.523
samples = 39
value = [18, 1, 20]
class = 1
gini = 0.623
samples = 32
value = [16, 7, 9]
class = -1
gini = 0.133
samples = 42
value = [39, 0, 3]
class = -1
gini = 0.101
samples = 113
value = [107, 1, 5]
class = -1
gini = 0.442
samples = 79
value = [53, 26, 0]
class = -1
gini = 0.496
samples = 75
value = [9, 16, 50]
class = 1
gini = 0.5
samples = 34
value = [17, 0, 17]
class = -1

9

Model Accuracy after Pruning

```
y_pred_best = best_dt.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test,y_pred_best))
```

```
Accuracy: 0.8154981549815498
```

DISCUSSION

- The aim of the decision tree is to identify if the website is phishy, legitimate or suspicous.
- The model accuracy after pruning appears to be lower than before.
- This might happen due to the dataset requires a **deeper tree structure** to capture patterns which will make the decision tree too complex.
- From the decision tree diagram, we can detect on whether the website is phishy or not based on 7 factors.

- The brighter the colours of each nodes indicates that it is more pure.
- The paler it gets, means that the nodes is impure.
- From the root node, if the condition SFH <= 0.5 is true then the decision tree continues to SSLfinal_State <= -0.5, when it is true, the leaf nodes makes a final decision that the website is legit.
- However, we are expecting it to be phishy since the two condition satisfy a phishy website.

DISCUSSION

- The same thing we can observed when the terminal node is false followed by popupwindow > -0.5 and request_URL > -0.5 we are expecting the terminal node to predict a legit website but we are getting a phising instead.

Random Forest

DEFAULT MODEL

BUILT RANDOM FOREST CLASSIFIER WITH 50 DECISION TREES

```
> rf_model = RandomForestClassifier(n_estimators = 50, random_state=6)
```

FIT THE MODEL INTO TRAINING SET

```
> rf_model.fit(x_train, y_train)
```

PREDICT THE RESPONSE FOR THE TEST SET

```
> y_pred = rf_model.predict(x_test)
```

THE MODEL ACCURACY, PRECISION & RECALL

THE CODING

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average="weighted")
recall = recall_score(y_test, y_pred, average="weighted")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
```

THE OUTPUT

```
Accuracy: 0.8856088560885609
Precision: 0.8885367557514432
Recall: 0.8856088560885609
```

The average=”weighted” indicates that the precision and recall is calculated for each class and been divided by the number of true instances for each class

BEST RANDOM FOREST

Random Forest

TUNING THE HYPERPARAMETERS

THE CODING

```
rf_param_dist = {'n_estimators': range(50,2000),
                 'max_depth': range(1,30)}

rf_rand_search = RandomizedSearchCV(rf_model,
param_distributions = rf_param_dist, n_iter=5, cv=5,
random_state=6)

rf_rand_search.fit(x_train, y_train)

best_rf = rf_rand_search.best_estimator_

print('Best random forest hyperparameters:',

rf_rand_search.best_params_)
```

- **rf_param_dist is defined, specifying the number of decision tree and the maximum depth of each decision tree**
- **RandomizedSearchCV is used to perform a randomized search over the specified hyperparameter space**

THE OUTPUT

```
Best random forest hyperparameters: {'n_estimators': 1263, 'max_depth': 27}
```

FIT THE BEST RANDOM FOREST MODEL

```
> best_rf.fit(x_train, y_train)
```

PREDICT & PLOT THE CONFUSION MATRIX

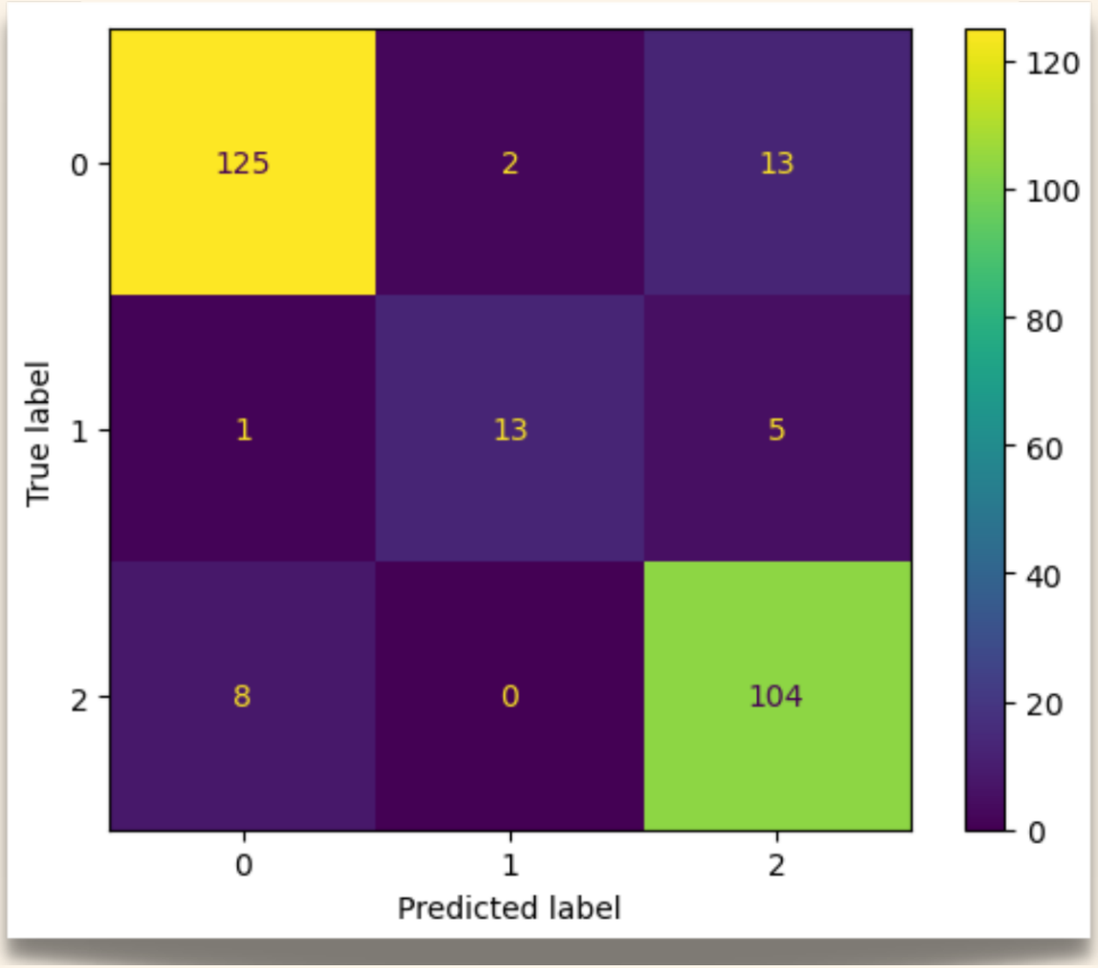
```
> y_pred_best = best_rf.predict(x_test)
>> cm=confusion_matrix(y_test,y_pred_best)
>> ConfusionMatrixDisplay(confusion_matrix=cm).plot( )
> plt.show( )
```

THE CONFUSION MATRIX

Class 0 :
Phishy Websites

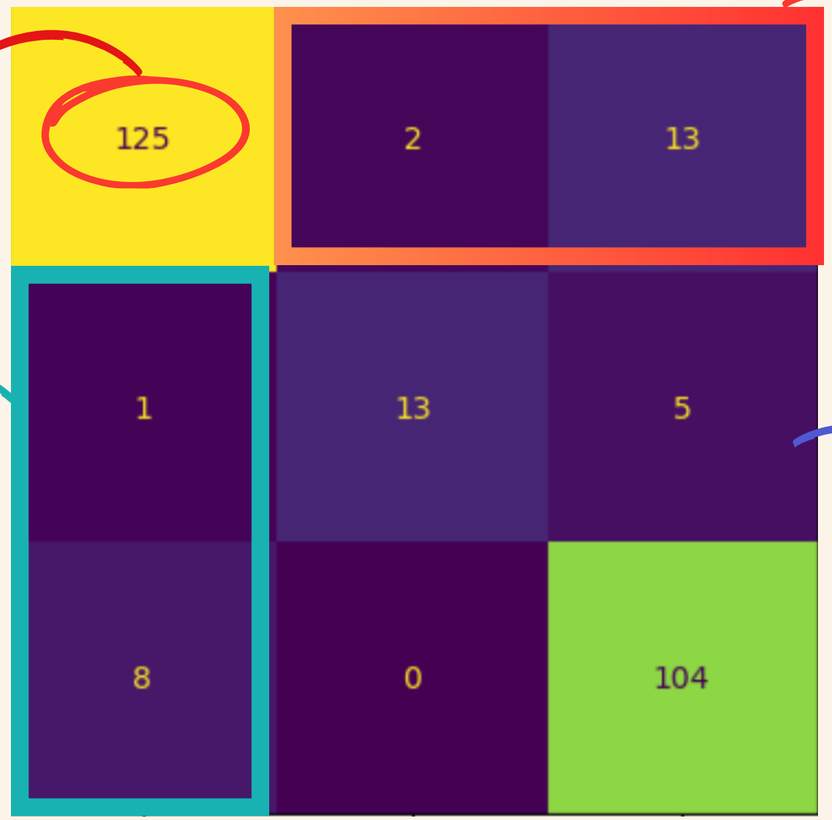
Class 1 :
Suspicious Websites

Class 2 :
Legitimate Websites



CLASS 0 : PHISHY WEBSITES

TP : Actual=Predicted
TP = 125 (The model correctly predict phishy websites)



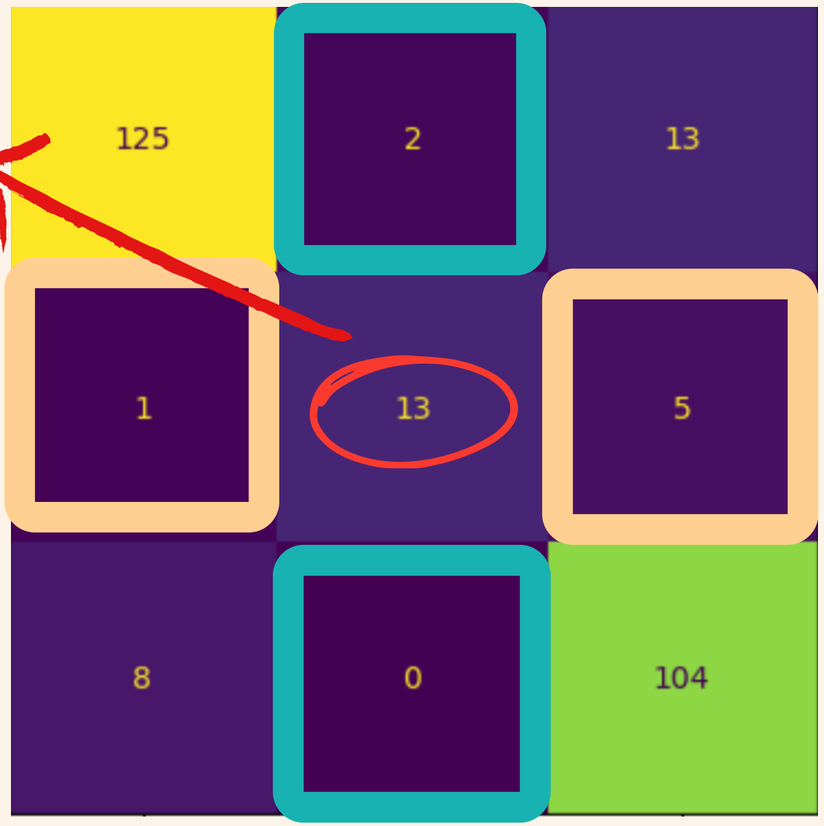
FN : Actual is (+ : PHISHY) but model predict it as (–)
FN = 2+13 =15 (the sum of values for phishy row except TP)

FP : Actual is (–) but model predict it as (+ : PHISHY)
FP = 1 +8 =9 (the sum of values for phishy column except TP)

TN : Sum of all values except for values in phishy class
TN = 13+5+0+104 =122

CLASS 1 : SUSPICIOUS(SUS) WEBSITES

TP : Actual=Predicted
TP = 13 (The model correctly predict sus websites)



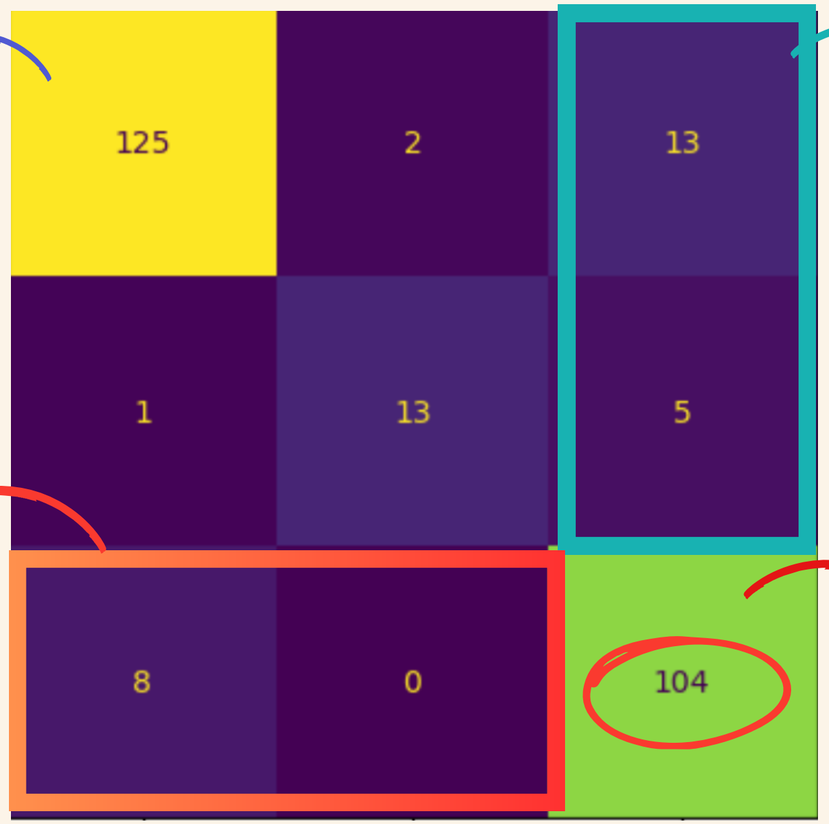
FN : Actual is (+ : sus) but model predict it as (–)
FN = 1+5 =6 (the sum of values for sus row except TP)

FP : Actual is (–) but model predict it as (+ : SUS)
FP = 2 +0 =2 (the sum of values for sus column except TP)

TN : Sum of all values except for values in sus class
TN = 125+13+8+104 = 250

CLASS 2 : LEGITIMATE(LEGIT) WEBSITES

TN : Sum of all values except for values in legit class
TN = 125+2+1+13 =141



FP : Actual is (–) but model predict it as (+ : LEGIT)
FP = 13+5 =18 (the sum of values for legit column except TP)

FN : Actual is (+ : LEGIT) but model predict it as (–)
FN = 8 +0 =8 (the sum of values for legit row except TP)

TP : Actual=Predicted
TP = 104 (The model correctly predict legit websites)

CALCULATING ACCURACY

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}}$$

Sum of all TP values for each class

Sum of all values. In this case, it is 271

THE MODEL ACCURACY, PRECISION & RECALL (BEST MODEL)

THE CODING

```
accuracy = accuracy_score(y_test, y_pred_best)
precision = precision_score(y_test, y_pred_best, average="weighted")
recall = recall_score(y_test, y_pred_best, average="weighted")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
```

THE OUTPUT

```
Best Accuracy: 0.8929889298892989
Best Precision: 0.8949781971516804
Best Recall: 0.8929889298892989
```

The accuracy, precision and recall has been improved from the last model

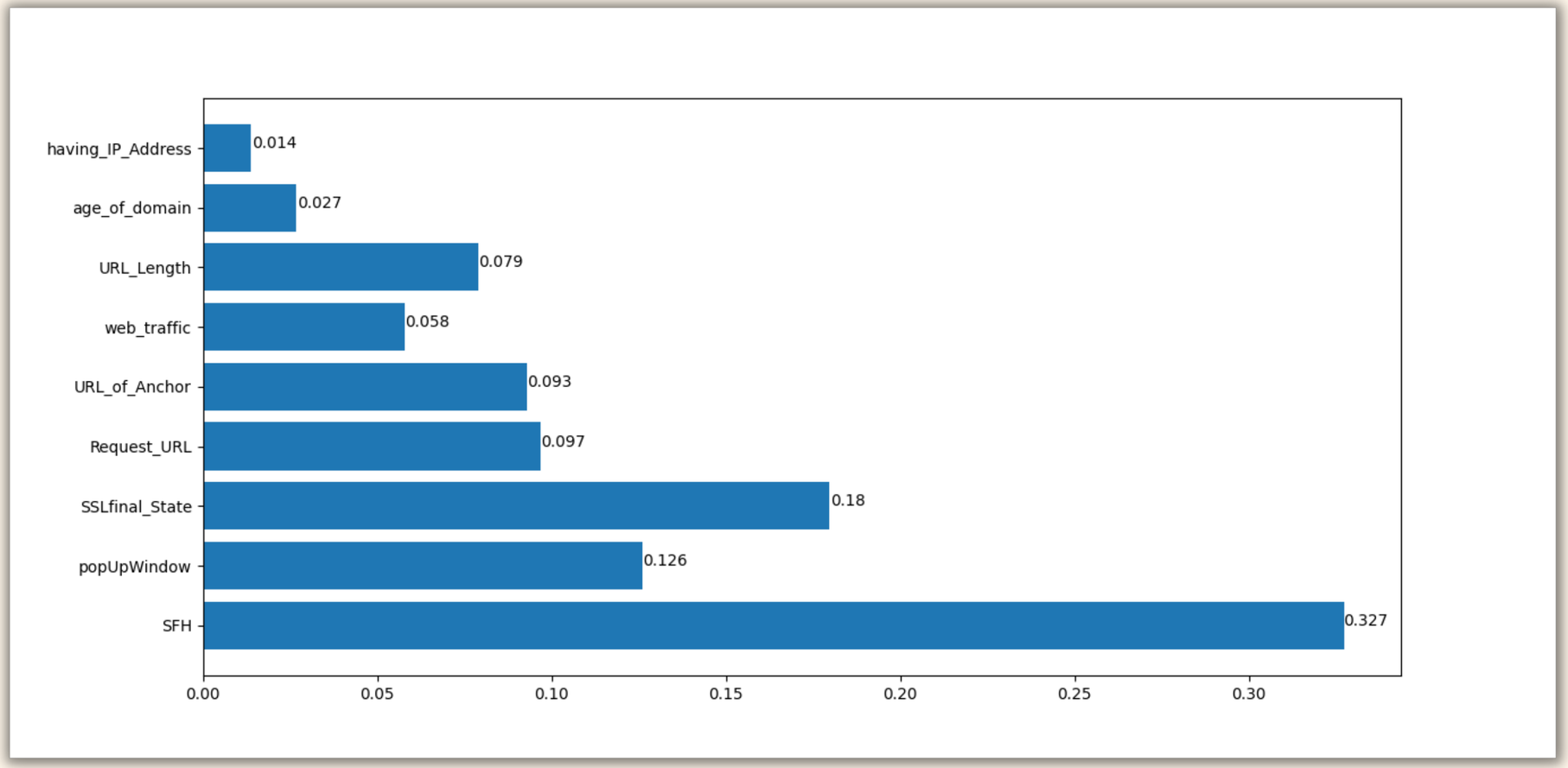
IMPORTANCE OF FEATURE

```
best_rf.feature_importances_
```

BAR PLOT OF FEATURE IMPORTANCE

```
rounded_best_feature = [np.round(x,3) for x in best_rf.feature_importances_]
features = ftnames[:]
features.pop()
plt.barh(features, best_rf.feature_importances_)
for index, value in enumerate(rounded_best_feature):
    plt.text(value, index,
             str(value))
plt.show( )
```

THE BARPLOT OF FEATURE IMPORTANCE



DISCUSSION

- The model accuracy of the best random forest shows an improvement on the accuracy but the best decision tree does not.
- This might happen because by aggregating prediction from multiple tree, the random forest might mitigate the overfitting.
- The barplot demonstrates the significance of SFH in this model by indicating that if the domain name in SFHs differs from the webpage's domain name, it indicates that the webpage is dubious since external domains are handling the submission.

- The reason SSL_finalstate is the second most crucial feature is that without it, consumer data may be read by unauthorised individuals, which increases website phishing.

CONCLUSION

- The Random Forest, with hyperparameter tuning, showcased enhanced predictive accuracy by 89% over the baseline.
- Decision Tree and Random Forest models, when appropriately tuned and optimized, present valuable tools for website phishing classification.
- Hence, these models contribute to the ongoing efforts to bolster cybersecurity measures providing reliable methods for distinguishing between legitimate and potentially malicious websites.

DATA SOURCE

- ThAbdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based Associative Classification data mining. Expert Systems with Applications, 41(13), 5948–5959. https://doi.org/10.1016/j.eswa.2014.03.019

QNA

Thank You

By Group 6