Swarm Intelligence Approaches

```
Parameter Setting of Deep Learning Neural Network: Acase Study of Website
Classification.
```

1. Introduction

```
The problem of detecting phishing websites hasbeen addressed many times using
various methodologies from conventional classifiers to more complex hybrid methods.
Recentadvancements in deep learning approaches suggested that the classification of
phishing websites using deep learning neural networksshould outperform the
traditional machine learning algorithms.However, the results of utilizing deep
neural networks heavily depend on the setting of different learning parameters. In
this paper,we propose a swarm intelligence based approach to parameter setting of
deep learning neural network.
Therefore we import the libraries and tabular Dataset that we intend to use in
developing physhing website classifier.
```

In [8]:

```
#importing data for basic analysis-preprocessing.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import random
%matplotlib inline
#importing the libraries for data preprocessing
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score,KFold,Stratifi
from sklearn.metrics import confusion_matrix,accuracy_score,f1_score,average_precisi
from sklearn.preprocessing import RobustScaler,StandardScaler,LabelEncoder,MinMaxSca

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoo
from sklearn.feature_selection import SelectKBest,chi2

from keras.models import Sequential
from keras.layers import Activation,BatchNormalization
from keras.layers.core import Dense,Dropout
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy
from keras.callbacks import ReduceLROnPlateau,EarlyStopping
Data=pd.read_csv("//Users//nelsonotumaongaya//Desktop//Phishing.csv",header="infer")
```

```
Using TensorFlow backend.
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorflow/python/framework/dtypes.py:516: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorflow/python/framework/dtypes.py:517: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorflow/python/framework/dtypes.py:518: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorflow/python/framework/dtypes.py:519: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorflow/python/framework/dtypes.py:520: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorflow/python/framework/dtypes.py:525: FutureWarning: Passing (typ
e, 1) or '1type' as a synonym of type is deprecated; in a future vers
```

```
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning: Passin
g (type, 1) or '1type' as a synonym of type is deprecated; in a futur
e version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning: Passin
g (type, 1) or '1type' as a synonym of type is deprecated; in a futur
e version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorboard/compat/tensorflow_stub/dtypes.py:543: FutureWarning: Passin
g (type, 1) or '1type' as a synonym of type is deprecated; in a futur
e version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorboard/compat/tensorflow_stub/dtypes.py:544: FutureWarning: Passin
g (type, 1) or '1type' as a synonym of type is deprecated; in a futur
e version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorboard/compat/tensorflow_stub/dtypes.py:545: FutureWarning: Passin
g (type, 1) or '1type' as a synonym of type is deprecated; in a futur
e version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/te
nsorboard/compat/tensorflow_stub/dtypes.py:550: FutureWarning: Passin
g (type, 1) or '1type' as a synonym of type is deprecated; in a futur
e version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [9]:

```
Data
```

Out[9]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redir
0	-1	1	1	1	
1	1	1	1	1	
2	1	0	1	1	
3	1	0	1	1	
4	1	0	-1	1	
...	...	...	...	...	
11050	1	-1	1	-1	
11051	-1	1	1	-1	
11052	1	-1	1	1	
11053	-1	-1	1	1	
11054	-1	-1	1	1	

11055 rows × 31 columns

Understanding the Data Sets We explore the data sets further.

We need to understand the data set indetail first. We develop a brief understanding of the dataset with which we will be working with. For example how many features are there in the dataset, how man unique label. How are they distributed or how are the labels distributed, different data types and quantities.

In [3]:

```
Data.head()# we explore the headers on the datasets to understand the features.
```

Out[3]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirectin
0	-1	1	1	1	-
1	1	1	1	1	
2	1	0	1	1	
3	1	0	1	1	
4	1	0	-1	1	

5 rows × 31 columns

In [4]:

```
Data.tail()# we explore the tail on the datasets to understand it. Shows the bottom
```

Out[4]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redir
11050	1	-1	1	-1	
11051	-1	1	1	-1	
11052	1	-1	1	1	
11053	-1	-1	1	1	
11054	-1	-1	1	1	

5 rows × 31 columns

In [5]:

```
Data.tail().T# We explore the data sets to check on the headers conformity and unde
```

Out[5]:

	11050	11051	11052	11053	11054
having_IP_Address	1	-1	1	-1	-1
URL_Length	-1	1	-1	-1	-1
Shortining_Service	1	1	1	1	1
having_At_Symbol	-1	-1	1	1	1
double_slash_redirecting	1	-1	1	1	1
Prefix_Suffix	1	-1	-1	-1	-1
having_Sub_Domain	1	1	1	-1	-1
SSLfinal_State	1	-1	-1	-1	-1
Domain_registeration_length	-1	-1	-1	1	1
Favicon	-1	-1	1	-1	1
port	-1	-1	1	1	1
HTTPS_token	1	1	1	1	1
Request_URL	1	1	1	-1	-1
URL_of_Anchor	1	-1	0	-1	-1
Links_in_tags	1	-1	-1	1	0
SFH	-1	0	-1	-1	-1
Submitting_to_email	-1	-1	1	1	1
Abnormal_URL	1	-1	1	1	1
Redirect	0	1	0	0	0
on_mouseover	-1	-1	1	-1	1
RightClick	-1	1	1	1	1
popUpWidnow	-1	-1	1	-1	1
Iframe	-1	1	1	1	1
age_of_domain	1	1	1	1	-1
DNSRecord	1	1	1	1	1
web_traffic	-1	1	1	1	-1
Page_Rank	-1	1	-1	-1	-1
Google_Index	1	1	1	1	-1
Links_pointing_to_page	1	-1	0	1	1
Statistical_report	1	1	1	1	-1
Result	1	-1	-1	-1	-1

In [6]:

```
Data.sample(10)
```

Out[6]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redir
5680	-1	1	1	-1	
3981	1	-1	1	1	
6881	-1	-1	1	1	
10988	-1	-1	1	1	
8116	-1	-1	1	1	
5211	1	-1	1	-1	
414	1	-1	1	1	
6511	-1	-1	1	1	
5013	1	-1	1	1	
5175	-1	-1	-1	-1	

10 rows × 31 columns

In [7]:

```
Data.T
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	11045	11046	11047	110
having_IP_Address	-1	1	1	1	1	-1	1	1	1	1	...	1	-1	-1	
URL_Length	1	1	0	0	0	0	0	0	0	1	...	-1	-1	-1	
Shortining_Service	1	1	1	1	-1	-1	-1	1	-1	-1	...	1	1	1	
having_At_Symbol	1	1	1	1	1	1	1	1	1	1	...	1	1	1	
double_slash_redirecting	-1	1	1	1	1	-1	1	1	1	1	...	1	1	1	
Prefix_Suffix	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	...	-1	-1	-1	
having_Sub_Domain	-1	0	-1	-1	1	1	-1	-1	1	-1	...	1	1	1	
SSLfinal_State	-1	1	-1	-1	1	1	-1	-1	1	1	...	-1	1	-1	
Domain_registeration_length	-1	-1	-1	1	-1	-1	1	1	-1	-1	...	-1	-1	-1	
Favicon	1	1	1	1	1	1	1	1	1	1	...	1	1	1	
port	1	1	1	1	1	1	1	1	1	1	...	1	1	1	
HTTPS_token	-1	-1	-1	-1	1	-1	1	-1	-1	1	...	1	1	-1	
Request_URL	1	1	1	-1	1	1	-1	-1	1	1	...	1	-1	1	
URL_of_Anchor	-1	0	0	0	0	0	-1	0	0	0	...	0	0	-1	
Links_in_tags	1	-1	-1	0	0	0	0	-1	1	1	...	1	1	-1	
SFH	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	...	-1	1	-1	
Submitting_to_email	-1	1	-1	1	1	-1	-1	1	1	1	...	1	1	-1	
Abnormal_URL	-1	1	-1	1	1	-1	-1	1	1	1	...	1	1	1	
Redirect	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
on_mouseover	1	1	1	1	-1	1	1	1	1	1	...	1	1	1	
RightClick	1	1	1	1	1	1	1	1	1	1	...	1	1	1	
popUpWidnow	1	1	1	1	-1	1	1	1	1	1	...	1	1	1	
Iframe	1	1	1	1	1	1	1	1	1	1	...	1	1	1	
age_of_domain	-1	-1	1	-1	-1	1	1	-1	1	1	...	1	1	1	
DNSRecord	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	...	1	1	1	
web_traffic	-1	0	1	1	0	1	-1	0	1	0	...	0	0	1	
Page_Rank	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	...	-1	-1	-1	
Google_Index	1	1	1	1	1	1	1	1	1	1	...	1	1	1	
Links_pointing_to_page	1	1	0	-1	1	-1	0	0	0	0	...	0	1	0	
Statistical_report	-1	1	-1	1	1	-1	-1	1	1	1	...	1	1	1	
Result	-1	-1	-1	-1	1	1	-1	-1	1	-1	...	1	1	-1	

31 rows × 11055 columns

In [8]:

```
Data.isnull().sum()#Check if there are any missing values
```

Out[8]:

```
having_IP_Address              0
URL_Length                     0
Shortining_Service             0
having_At_Symbol               0
double_slash_redirecting       0
Prefix_Suffix                  0
having_Sub_Domain              0
SSLfinal_State                 0
Domain_registeration_length    0
Favicon                        0
port                           0
HTTPS_token                    0
Request_URL                    0
URL_of_Anchor                  0
Links_in_tags                  0
SFH                            0
Submitting_to_email            0
Abnormal_URL                   0
Redirect                       0
on_mouseover                   0
RightClick                     0
popUpWidnow                    0
Iframe                         0
age_of_domain                  0
DNSRecord                      0
web_traffic                    0
Page_Rank                      0
Google_Index                   0
Links_pointing_to_page         0
Statistical_report             0
Result                         0
dtype: int64
```

2. We now Querry the data sets to obtain the reports.

In [9]:

```
len(Data)#Shows how much data the Dataset contains:
```

Out[9]:

```
11055
```

In [10]:

```
Data.head() #Shows the top Data in the set that you are exploring.
```

Out[10]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirectin
0	-1	1	1	1	-
1	1	1	1	1	
2	1	0	1	1	
3	1	0	1	1	
4	1	0	-1	1	

5 rows × 31 columns

In [11]:

```
Data.info() #This displays all columns and their data types,
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 31 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   having_IP_Address            11055 non-null  int64
 1   URL_Length                   11055 non-null  int64
 2   Shortining_Service           11055 non-null  int64
 3   having_At_Symbol             11055 non-null  int64
 4   double_slash_redirecting     11055 non-null  int64
 5   Prefix_Suffix                11055 non-null  int64
 6   having_Sub_Domain            11055 non-null  int64
 7   SSLfinal_State               11055 non-null  int64
 8   Domain_registeration_length  11055 non-null  int64
 9   Favicon                      11055 non-null  int64
 10  port                         11055 non-null  int64
 11  HTTPS_token                  11055 non-null  int64
 12  Request_URL                  11055 non-null  int64
 13  URL_of_Anchor                11055 non-null  int64
 14  Links_in_tags                11055 non-null  int64
 15  SFH                          11055 non-null  int64
 16  Submitting_to_email          11055 non-null  int64
 17  Abnormal_URL                 11055 non-null  int64
 18  Redirect                     11055 non-null  int64
 19  on_mouseover                 11055 non-null  int64
 20  RightClick                   11055 non-null  int64
 21  popUpWidnow                  11055 non-null  int64
 22  Iframe                       11055 non-null  int64
 23  age_of_domain                11055 non-null  int64
 24  DNSRecord                    11055 non-null  int64
 25  web_traffic                  11055 non-null  int64
 26  Page_Rank                    11055 non-null  int64
 27  Google_Index                 11055 non-null  int64
 28  Links_pointing_to_page       11055 non-null  int64
 29  Statistical_report           11055 non-null  int64
 30  Result                       11055 non-null  int64
dtypes: int64(31)
memory usage: 2.6 MB
```

In [12]:

```
Data.describe()# This shows you some basic descriptive statistics for all numeric co
```

Out[12]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redi
count	11055.000000	11055.000000	11055.000000	11055.000000	11055.
mean	0.313795	-0.633198	0.738761	0.700588	0.
std	0.949534	0.766095	0.673998	0.713598	0.
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.
25%	-1.000000	-1.000000	1.000000	1.000000	1.
50%	1.000000	-1.000000	1.000000	1.000000	1.
75%	1.000000	-1.000000	1.000000	1.000000	1.
max	1.000000	1.000000	1.000000	1.000000	1.

8 rows × 31 columns

Cleaning the dataset accordingly so that it is well suited for a Machine Learning Model.

Let us examine the feature "Sub-domain and multi sub-domain". A technique used by phishers to scam users is by adding a sub-domain to the URL so users may believe they are dealing with an authentic website.

In [13]:

```
x=Data.iloc [:,1:-1]
x=x.values
y=Data.iloc[:,-1].values
```

In [14]:

```
x
```

Out[14]:

```
array([[ 1,  1,  1, ...,  1,  1, -1],
       [ 1,  1,  1, ...,  1,  1,  1],
       [ 0,  1,  1, ...,  1,  0, -1],
       ...,
       [-1,  1,  1, ...,  1,  0,  1],
       [-1,  1,  1, ...,  1,  1,  1],
       [-1,  1,  1, ..., -1,  1, -1]])
```

In [15]:

```
y
```

Out[15]:

```
array([-1, -1, -1, ..., -1, -1, -1])
```

2. Building The Model Using Decision Tree

In [16]:

```
from sklearn import preprocessing
x1= preprocessing.normalize(x)
#We normalize the data: refers to rescaling real valued numeric attributes into the
#It is useful to scale the input attributes for a model that relies on the magnitude
```

In [17]:

```
x1
```

Out[17]:

```
array([[ 0.18898224,  0.18898224,  0.18898224, ...,  0.18898224,
         0.18898224, -0.18898224],
       [ 0.2       ,  0.2       ,  0.2       , ...,  0.2       ,
         0.2       ,  0.2       ],
       [ 0.        ,  0.2       ,  0.2       , ...,  0.2       ,
         0.        , -0.2       ],
       ...,
       [-0.19611614,  0.19611614,  0.19611614, ...,  0.19611614,
         0.        ,  0.19611614],
       [-0.18898224,  0.18898224,  0.18898224, ...,  0.18898224,
         0.18898224,  0.18898224],
       [-0.19245009,  0.19245009,  0.19245009, ..., -0.19245009,
         0.19245009, -0.19245009]])
```

We spilt the data and create training and test data sets 80% Training set and 20% is the Testing Set.

In [18]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x1,y,test_size=0.20,random_state=42)
```

In [19]:

```
print(x_train.shape, y_train.shape,x_test.shape, y_test.shape)
```

```
(8844, 29) (8844,) (2211, 29) (2211,)
```

In [20]:

```
#We now fit the model to determine the accuracy using Decision Trees
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
clf_gini=DecisionTreeClassifier(criterion="gini", random_state=100,max_depth=3, min_
clf_gini.fit(x_train, y_train)
Y_pred=clf_gini.predict(x_test)
from sklearn import metrics
metrics.accuracy_score(y_test,Y_pred)*100
```

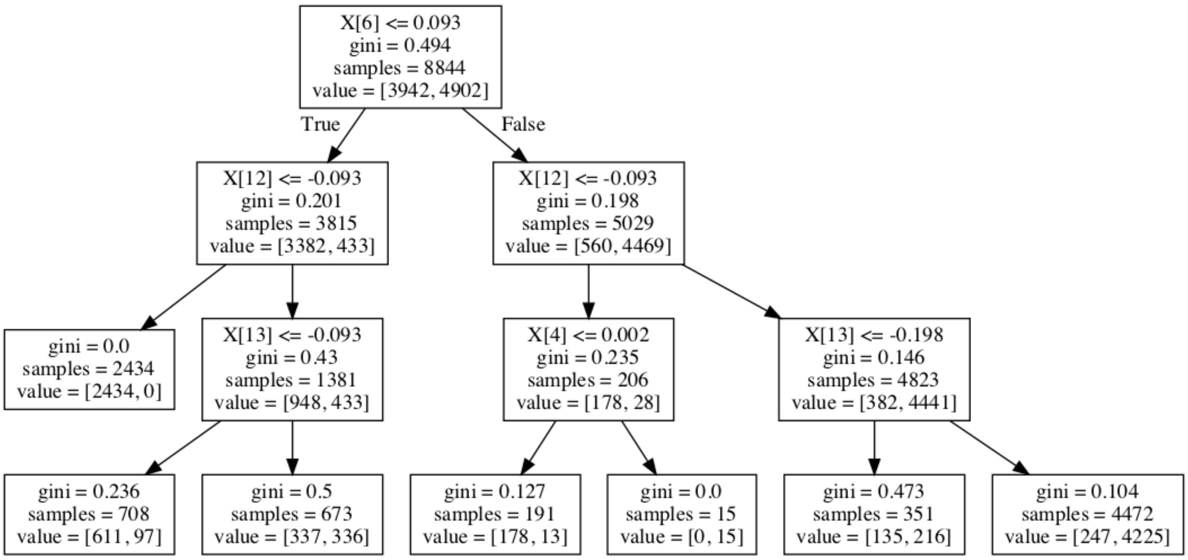
Out[20]:

```
91.40660334690185
```

In [21]:

```
import pydotplus
from sklearn.tree import export_graphviz
from IPython.display import Image
dot_data=tree.export_graphviz(clf_gini)
graph=pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Out[21]:



Plotting the Data Graphically

if the value in the column of result is less than 1 then it is legit, if its more than >>1 then its phishy "Sub-domain and multi sub-domain". A technique used by phishers to scam users is by adding a sub-domain to the URL so users may believe they are dealing with an authentic website. therefore the dots in the domain part should be less than 1,if it is more than 1 or equals to zero then its not legitimate, hence suspicious else phishy

In [22]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x1,y,test_size=0.20,random_state=42)
from sklearn.metrics import accuracy_score
```

In [23]:

```
maxdepths=[3,4,5,6,7,8,9,10,11,15,20,25,30,35,40,45,50,60]# alist contains
trainAcc=np.zeros(len(maxdepths))
testAcc=np.zeros(len(maxdepths))
```

In [24]:

```
trainAcc
```

Out[24]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0.])
```

In [25]:

```
testAcc
```

Out[25]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0.])
```

In [26]:

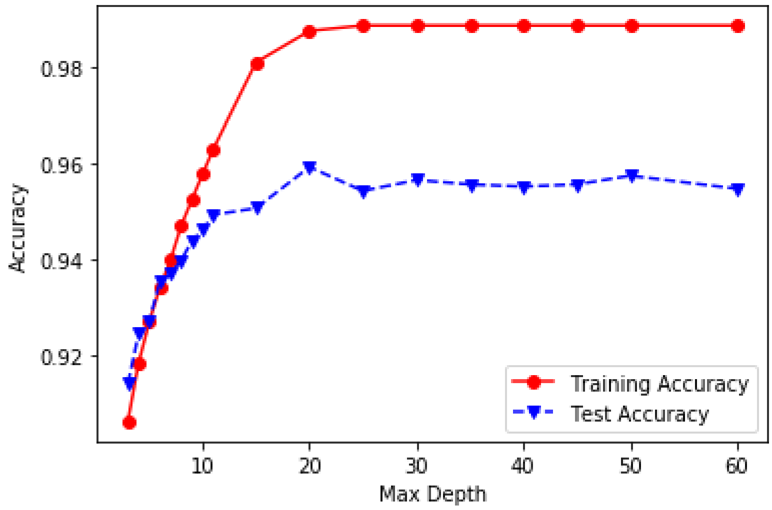
```
index=0
for depth in maxdepths:
    clf=tree.DecisionTreeClassifier(max_depth=depth)
    clf=clf.fit(x_train,y_train)
    y_predTrain=clf.predict(x_train)
    y_predTest=clf.predict(x_test)
    trainAcc[index]=accuracy_score(y_train,y_predTrain)
    testAcc[index]=accuracy_score(y_test,y_predTest)
    index +=1
```

In [27]:

```
plt.plot(maxdepths,trainAcc,'ro-',maxdepths,testAcc,'bv--')
plt.legend(['Training Accuracy','Test Accuracy'])
plt.xlabel('Max Depth')
plt.ylabel('Accuracy')
```

Out[27]:

```
Text(0, 0.5, 'Accuracy')
```



Conclusion

The plot shows that the accuracy will continue to improve as the maximum depth of the tree increases,i.e the model becomes more complex.

(II)Building the Model using Multi Linear Regression (Swarm Intelligence Approach)

Linear Regression is one of the most commonly used predictive modelling techniques. The aim of the modelling technique is to find a Mathematical equation for continuous response variable. The equation can be generalised as:y=B1+B2x+x. The coefficients are called regression coefficient(B1 and B2). We import the libraries for our data sets, the libraries will assist us in exploring the data sets.

In [28]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#importing the libraries for our data sets.
```

In [29]:

```
#we now load the dataset from the folder called Phishing
Data=pd.read_csv("//Users//nelsonotumaongaya//Desktop//Phishing.csv",header="infer")
```

In [30]:

```
Data# we now visualize our dataset
```

Out[30]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redir
0	-1	1	1	1	
1	1	1	1	1	
2	1	0	1	1	
3	1	0	1	1	
4	1	0	-1	1	
...	...	...	...	...	
11050	1	-1	1	-1	
11051	-1	1	1	-1	
11052	1	-1	1	1	
11053	-1	-1	1	1	
11054	-1	-1	1	1	

11055 rows × 31 columns

In [31]:

```
#We now check for our dataset information.
Data.info
```

Out[31]:

```
<bound method DataFrame.info of        having_IP_Address  URL_Length
Shortining_Service  having_At_Symbol  \
0                     -1           1                  1
1
1                      1           1                  1
1
2                      1           0                  1
1
3                      1           0                  1
1
4                      1           0                 -1
1
...                  ...         ...                ...
...
11050                  1          -1                  1
-1
11051                 -1           1                  1
-1
11052                  1          -1                  1
1
11053                 -1          -1                  1
1
11054                 -1          -1                  1
1

       double_slash_redirecting  Prefix_Suffix  having_Sub_Domain  \
0                            -1             -1                 -1
1                             1             -1                  0
2                             1             -1                 -1
3                             1             -1                 -1
4                             1             -1                  1
...                         ...            ...                ...
11050                         1              1                  1
11051                        -1             -1                  1
11052                         1             -1                  1
11053                         1             -1                 -1
11054                         1             -1                 -1

       SSLfinal_State  Domain_registeration_length  Favicon  ...  pop
UpWidnow  \
0                  -1                           -1        1  ...
1
1                   1                           -1        1  ...
1
2                  -1                           -1        1  ...
1
3                  -1                            1        1  ...
1
4                   1                           -1        1  ...
-1
...               ...                          ...      ...  ...
...
11050               1                           -1       -1  ...
-1
```

```
11051               -1                               -1        -1  ...
-1
11052               -1                               -1         1  ...
1
11053               -1                                1        -1  ...
-1
11054               -1                                1         1  ...
1

       Iframe  age_of_domain  DNSRecord  web_traffic  Page_Rank  Goog
le_Index  \
0           1             -1         -1           -1         -1
1
1           1             -1         -1            0         -1
1
2           1              1         -1            1         -1
1
3           1             -1         -1            1         -1
1
4           1             -1         -1            0         -1
1
...       ...            ...        ...          ...        ...
...
11050      -1              1          1           -1         -1
1
11051       1              1          1            1          1
1
11052       1              1          1            1         -1
1
11053       1              1          1            1         -1
1
11054       1             -1          1           -1         -1
-1

       Links_pointing_to_page  Statistical_report  Result
0                           1                  -1      -1
1                           1                   1      -1
2                           0                  -1      -1
3                          -1                   1      -1
4                           1                   1       1
...                       ...                 ...     ...
11050                       1                   1       1
11051                      -1                   1      -1
11052                       0                   1      -1
11053                       1                   1      -1
11054                       1                  -1      -1

[11055 rows x 31 columns]>
```

In [32]:

```
Data
```

Out[32]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redir
0	-1	1	1	1	
1	1	1	1	1	
2	1	0	1	1	
3	1	0	1	1	
4	1	0	-1	1	
...	...	...	...	...	
11050	1	-1	1	-1	
11051	-1	1	1	-1	
11052	1	-1	1	1	
11053	-1	-1	1	1	
11054	-1	-1	1	1	

11055 rows × 31 columns

In [33]:

```
#We now check for number of columns in our dataset
Data.columns.values
```

Out[33]:

```
array(['having_IP_Address', 'URL_Length', 'Shortining_Service',
       'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffi
x',
       'having_Sub_Domain', 'SSLfinal_State',
       'Domain_registeration_length', 'Favicon', 'port', 'HTTPS_toke
n',
       'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH',
       'Submitting_to_email', 'Abnormal_URL', 'Redirect', 'on_mouseove
r',
       'RightClick', 'popUpWidnow', 'Iframe', 'age_of_domain',
       'DNSRecord', 'web_traffic', 'Page_Rank', 'Google_Index',
       'Links_pointing_to_page', 'Statistical_report', 'Result'],
      dtype=object)
```

Lets Get the Independent and Dependent Variables

In [34]:

```
#Get dependent and independent variables.
# enables us select all rows / columns
# -1 is the index of last column in python
x = Data.iloc[:,:-1].values #independent variables
y = Data.iloc[:,-1].values #dependent variable
```

In [35]:

```
#We now Display all values in x - independent variables
print(x)
```

```
[[-1  1  1 ...  1  1 -1]
 [ 1  1  1 ...  1  1  1]
 [ 1  0  1 ...  1  0 -1]
 ...
 [ 1 -1  1 ...  1  0  1]
 [-1 -1  1 ...  1  1  1]
 [-1 -1  1 ... -1  1 -1]]
```

In [36]:

```
#Display all values in y - dependent variables
print(y)
```

```
[-1 -1 -1 ... -1 -1 -1]
```

Lets Check for missing values in my dataset. True shows that the respective column have missing values

In [37]:

```
Data.isnull()
```

Out[37]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redir
0	False	False	False	False	
1	False	False	False	False	
2	False	False	False	False	
3	False	False	False	False	
4	False	False	False	False	
...	...	...	...	...	
11050	False	False	False	False	
11051	False	False	False	False	
11052	False	False	False	False	
11053	False	False	False	False	
11054	False	False	False	False	

11055 rows × 31 columns

Lets Check for the sum of missing values in our dataset.

In [38]:

```
Data.isnull().sum()
```

Out[38]:

```
having_IP_Address             0
URL_Length                    0
Shortining_Service            0
having_At_Symbol              0
double_slash_redirecting      0
Prefix_Suffix                 0
having_Sub_Domain             0
SSLfinal_State                0
Domain_registeration_length   0
Favicon                       0
port                          0
HTTPS_token                   0
Request_URL                   0
URL_of_Anchor                 0
Links_in_tags                 0
SFH                           0
Submitting_to_email           0
Abnormal_URL                  0
Redirect                      0
on_mouseover                  0
RightClick                    0
popUpWidnow                   0
Iframe                        0
age_of_domain                 0
DNSRecord                     0
web_traffic                   0
Page_Rank                     0
Google_Index                  0
Links_pointing_to_page        0
Statistical_report            0
Result                        0
dtype: int64
```

From the results, the dataset is clean, there are no missing values

Splitting the Dataset into Train Set and Test Set

In [39]:

```
#We now split the dataset into train and
# 80% observation in Train and 20% in Test - since we have 50 observation
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, random_sta
```

In [40]:

```
print (x_train)
```

```
[[ 1 -1  1 ...  1  1  1]
 [ 1  1  1 ...  1  0  1]
 [-1 -1  1 ...  1  1 -1]
 ...
 [-1  1  1 ...  1  1  1]
 [-1 -1  1 ...  1  1  1]
 [ 1 -1  1 ...  1  1  1]]
```

In [41]:

```
print (y_train)
```

```
[ 1 -1 -1 ...  1 -1 -1]
```

In [42]:

```
#Display.max function in pandas enables us to display all values in our dataset.
pd.options.display.max_columns=None
print (x_test)
```

```
[[ 1 -1  1 ...  1  0  1]
 [ 1 -1  1 ...  1  0  1]
 [ 1 -1  1 ...  1  1  1]
 ...
 [-1 -1  1 ...  1  1  1]
 [ 1 -1  1 ...  1  0  1]
 [ 1 -1  1 ...  1  1  1]]
```

In [43]:

```
print (y_test)
```

```
[-1 -1 -1 ...  1  1  1]
```

Fitting Multi Linear Regression to the Training Set.

In [44]:

```
#We will import Linear Regression function from sklearn package
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[44]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normali
ze=False)
```

In [45]:

```
print(x_train)
```

```
[[ 1 -1  1 ...  1  1  1]
 [ 1  1  1 ...  1  0  1]
 [-1 -1  1 ...  1  1 -1]
 ...
 [-1  1  1 ...  1  1  1]
 [-1 -1  1 ...  1  1  1]
 [ 1 -1  1 ...  1  1  1]]
```

In [46]:

```
print(y_test)
```

```
[-1 -1 -1 ...  1  1  1]
```

Predicting the Test Set Results

In [47]:

```
#Predicting Test Results
y_pred = regressor.predict(x_test)
```

In [48]:

```
print(y_pred)
```

```
[-0.36458813 -0.94940467  0.47266869 ... -0.27618963 -0.26077198
  0.94651359]
```

In [49]:

```
#Compare Predicted and Actuals
Data = pd.DataFrame({'Actual':y_test.flatten(),'Predicted':y_pred.flatten()})
```

In [50]:

```
Data
```

Out[50]:

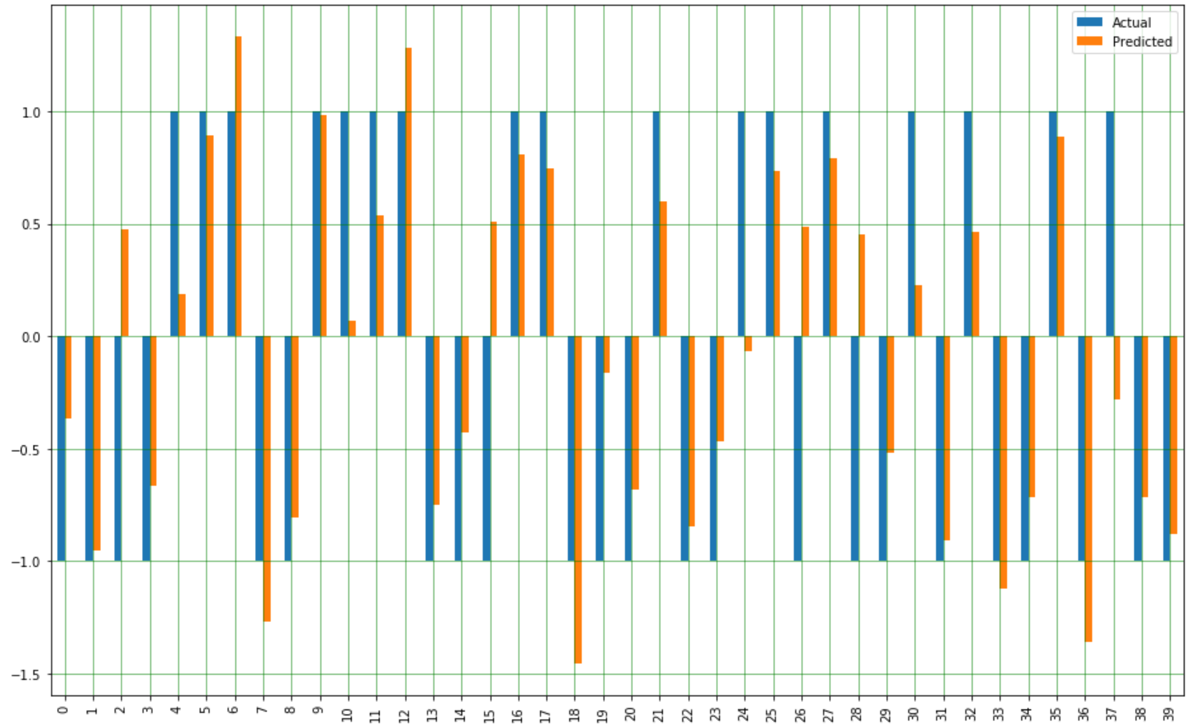
	Actual	Predicted
0	-1	-0.364588
1	-1	-0.949405
2	-1	0.472669
3	-1	-0.665583
4	1	0.186053
...	...	...
2206	1	0.926493
2207	-1	-1.109687
2208	1	-0.276190
2209	1	-0.260772
2210	1	0.946514

2211 rows × 2 columns

We Now Visualize The Results

In [51]:

```
#Visualize Actuals Vs Predicted
df1 = Data.head(40)
df1.plot(kind='bar',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```



In [52]:

```
#Import Metrics from sklearn
from sklearn import metrics
```

In [53]:

```
#Evaluate the Performance of the algorithm
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)
```

```
Mean Absolute Error: 0.4190841065745994
Mean Squared Error: 0.307601513936276
Root Mean Squared Error: 0.5546183498012629
```

Conclusion

Therefore the accuracy of the model is 94%- This shows that the model is fit for data analytics.since the predicted results against the actual indicate close relationship of the data.

3. K-Nearest Neighbour Classifier (Swam Intelligence Approach)

Introduction

In this approach the class label of a test instance is predicted based on the majority class of its k closest training instances. The number of nearest neighbors, k , is a hyperparameter that must be provided by the user, along with the distance metric. By default we can use Euclidean distance (equivalent to Minkoswki distance with an exponent factor equals to p=2)

In [54]:

```
#we import the libraries for our data preprocessing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import random
%matplotlib inline
```

In [55]:

```
#we now load the dataset from the folder called Phishing
Data=pd.read_csv("//Users//nelsonotumaongaya//Desktop//Phishing.csv",header="infer")
```

In [56]:

```
Data
```

Out[56]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redir
0	-1	1	1	1	
1	1	1	1	1	
2	1	0	1	1	
3	1	0	1	1	
4	1	0	-1	1	
...	...	...	...	...	
11050	1	-1	1	-1	
11051	-1	1	1	-1	
11052	1	-1	1	1	
11053	-1	-1	1	1	
11054	-1	-1	1	1	

11055 rows × 31 columns

2 Building The Model Using K-Nearest Neighbor Classifier

We have already explored the same data in Decision tree and Regression Analysis therefore the data is clean-

In [57]:

```
#We first split the dataset into training set comprising 80% and test set,comprising
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2,random_stat
```

In [58]:

```
x_train
```

Out[58]:

```
array([[ 1, -1,  1, ...,  1,  0, -1],
       [-1, -1,  1, ...,  1,  1,  1],
       [-1, -1,  1, ...,  1,  1,  1],
       ...,
       [ 1, -1,  1, ...,  1,  1,  1],
       [-1,  1,  1, ...,  1,  0,  1],
       [ 1, -1,  1, ...,  1,  0,  1]])
```

In [59]:

```
y_train
```

Out[59]:

```
array([-1, -1,  1, ...,  1,  1,  1])
```

In [60]:

```
#Display.max function in pandas enables us to display all values in our dataset.
pd.options.display.max_columns=None
x_test
```

Out[60]:

```
array([[-1, -1,  1, ...,  1,  1,  1],
       [ 1, -1,  1, ...,  1,  0,  1],
       [-1, -1, -1, ..., -1, -1,  1],
       ...,
       [ 1, -1,  1, ...,  1,  0,  1],
       [ 1, -1,  1, ...,  1,  0,  1],
       [ 1,  1,  1, ...,  1,  1, -1]])
```

In [61]:

```
y_test
```

Out[61]:

```
array([-1,  1,  1, ..., -1,  1,  1])
```

In [62]:

```
#We import KNeighborsClassifier from sklearn
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
%matplotlib inline
numNeighbours=[1,5,10,15,20,25,30]
trainAcc=[]
testAcc=[]
```

In [63]:

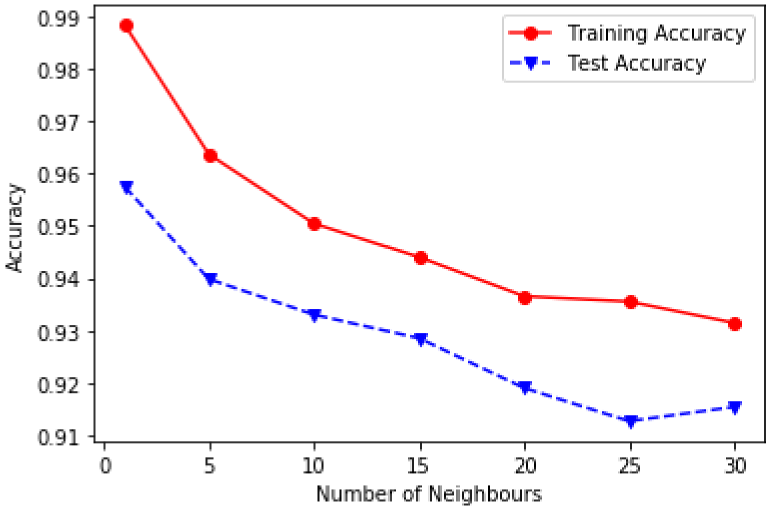
```
#We fit the KNearest Neighbor model to the training set
for k in numNeighbours:
    clf=KNeighborsClassifier(n_neighbors=k, metric='minkowski',p=2)
    clf.fit(x_train,y_train)
    y_predTrain=clf.predict(x_train)
    y_predTest=clf.predict(x_test)
    trainAcc.append(accuracy_score(y_train,y_predTrain))
    testAcc.append(accuracy_score(y_test,y_predTest))
```

In [64]:

```
plt.plot(numNeighbours,trainAcc,'ro-',numNeighbours,testAcc,'bv--')
plt.legend(['Training Accuracy','Test Accuracy'])
plt.xlabel('Number of Neighbours')
plt.ylabel('Accuracy')
```

Out[64]:

```
Text(0, 0.5, 'Accuracy')
```



3 Conclusion

Based on the training accuracy graph, it decreases with increase in depth. The same happens with test accuracy graph. It decreases sharply then increases slightly then it drops. From the test accuracy graph, the model performs best when it has a depth of 1 hence that is when it is best to deploy it (with this depth its accuracy level in detecting phishing sites is 96%).

4. Polynomial Regression

Explore the Dataset Again to Understand it.

In [65]:

```
#library import section
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np # linear algebra
import matplotlib.pyplot as plt # plotting
import os # accessing directory structure
from mpl_toolkits.mplot3d import Axes3D
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression
```

In [66]:

```
# Load the training data from the CSV file
Data=pd.read_csv("//Users//nelsonotumaongaya//Desktop//Phishing.csv",header="infer")
Data.dataframeName = 'Phishing.csv'
#determine data size
nRow, nCol = Data.shape
print(f'There are {nRow} rows and {nCol} columns')
print("\nLet's take a quick glance at what the data looks like:")
Data.head(5)
```

```
There are 11055 rows and 31 columns

Let's take a quick glance at what the data looks like:
```

Out[66]:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slash_redirectin
0	-1	1	1	1	-
1	1	1	1	1	
2	1	0	1	1	
3	1	0	1	1	
4	1	0	-1	1	

Each row has records/inputs(the features/indepedent variables) collected for a particular webite; and the end result(output/dependent variable): if the website was used for phising or not

In [67]:

```
print("\nKey statistical values:")
print(Data.describe())
```

```
Key statistical values:
       having_IP_Address    URL_Length  Shortining_Service  having_At_
Symbol  \
count       11055.000000  11055.000000        11055.000000      11055.
000000
mean            0.313795     -0.633198            0.738761          0.
700588
std             0.949534      0.766095            0.673998          0.
713598
min            -1.000000     -1.000000           -1.000000         -1.
000000
25%            -1.000000     -1.000000            1.000000          1.
000000
50%             1.000000     -1.000000            1.000000          1.
000000
75%             1.000000     -1.000000            1.000000          1.
000000
max             1.000000      1.000000            1.000000          1.
000000

       double_slash_redirecting  Prefix_Suffix  having_Sub_Domain  \
count              11055.000000   11055.000000       11055.000000
mean                   0.741474      -0.734962           0.063953
std                    0.671011       0.678139           0.817518
min                   -1.000000      -1.000000          -1.000000
25%                    1.000000      -1.000000          -1.000000
50%                    1.000000      -1.000000           0.000000
75%                    1.000000      -1.000000           1.000000
max                    1.000000       1.000000           1.000000

       SSLfinal_State  Domain_registeration_length       Favicon  \
count    11055.000000                 11055.000000  11055.000000
mean         0.250927                    -0.336771      0.628584
std          0.911892                     0.941629      0.777777
min         -1.000000                    -1.000000     -1.000000
25%         -1.000000                    -1.000000      1.000000
50%          1.000000                    -1.000000      1.000000
75%          1.000000                     1.000000      1.000000
max          1.000000                     1.000000      1.000000

               port   HTTPS_token   Request_URL  URL_of_Anchor  Links_
in_tags  \
count  11055.000000  11055.000000  11055.000000   11055.000000    1105
5.000000
mean       0.728268      0.675079      0.186793      -0.076526      -
0.118137
std        0.685324      0.737779      0.982444       0.715138
0.763973
min       -1.000000     -1.000000     -1.000000      -1.000000      -
1.000000
25%        1.000000      1.000000     -1.000000      -1.000000      -
1.000000
50%        1.000000      1.000000      1.000000       0.000000
0.000000
75%        1.000000      1.000000      1.000000       0.000000
```

```
0.000000
max         1.000000      1.000000      1.000000       1.000000
1.000000

                SFH  Submitting_to_email  Abnormal_URL      Redirect
\
count  11055.000000         11055.000000  11055.000000  11055.000000
mean      -0.595749             0.635640      0.705292      0.115694
std        0.759143             0.772021      0.708949      0.319872
min       -1.000000            -1.000000     -1.000000      0.000000
25%       -1.000000             1.000000      1.000000      0.000000
50%       -1.000000             1.000000      1.000000      0.000000
75%       -1.000000             1.000000      1.000000      0.000000
max        1.000000             1.000000      1.000000      1.000000

       on_mouseover    RightClick   popUpWidnow        Iframe  age_of_
domain  \
count  11055.000000  11055.000000  11055.000000  11055.000000   11055.
000000
mean       0.762099      0.913885      0.613388      0.816915       0.
061239
std        0.647490      0.405991      0.789818      0.576784       0.
998168
min       -1.000000     -1.000000     -1.000000     -1.000000      -1.
000000
25%        1.000000      1.000000      1.000000      1.000000      -1.
000000
50%        1.000000      1.000000      1.000000      1.000000       1.
000000
75%        1.000000      1.000000      1.000000      1.000000       1.
000000
max        1.000000      1.000000      1.000000      1.000000       1.
000000

          DNSRecord   web_traffic     Page_Rank  Google_Index  \
count  11055.000000  11055.000000  11055.000000  11055.000000
mean       0.377114      0.287291     -0.483673      0.721574
std        0.926209      0.827733      0.875289      0.692369
min       -1.000000     -1.000000     -1.000000     -1.000000
25%       -1.000000      0.000000     -1.000000      1.000000
50%        1.000000      1.000000     -1.000000      1.000000
75%        1.000000      1.000000      1.000000      1.000000
max        1.000000      1.000000      1.000000      1.000000

       Links_pointing_to_page  Statistical_report        Result
count            11055.000000        11055.000000  11055.000000
mean                 0.344007            0.719584      0.113885
std                  0.569944            0.694437      0.993539
min                 -1.000000           -1.000000     -1.000000
25%                  0.000000            1.000000     -1.000000
50%                  0.000000            1.000000      1.000000
75%                  1.000000            1.000000      1.000000
max                  1.000000            1.000000      1.000000
```

Distribution graphs (histogram/bar graph) of sampled columns:

1.Loading and Understanging the Phishing Websites

Dataset"

"We need to understand the data set indetail first.\n", "We develop a brief understanding of the dataset with which we will be working with. For example how many features are there in the dataset, how man unique label. How are they distributed or how are the labels distributed, different data types and quantities. "

In [68]:

```
#importing data for basic analysis-preprocessing.\n",
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import random
%matplotlib inline
```

Getting useful information from the dataset"

"Now, lets quickly find out how many classes in the dataset and how they the distributed in the dataset."

In [69]:

```
Data=pd.read_csv("//Users//nelsonotumaongaya//Desktop//Phishing.csv",header="infer")
from collections import Counter
```

In [70]:

```
classes =Counter(Data['Result'].values)
```

In [71]:

```
classes.most_common()
```

Out[71]:

```
[(1, 6157), (-1, 4898)]
```

This information can be presented using a DataFrame which will produce a very good table

In [72]:

```
class_dist=pd.DataFrame(classes.most_common(),columns=['Class','Num_Observation'])
```

In [73]:

```
class_dist
```

Out[73]:

	Class	Num_Observation
0	1	6157
1	-1	4898

We could also use a plot to convey the information as well

In [74]:

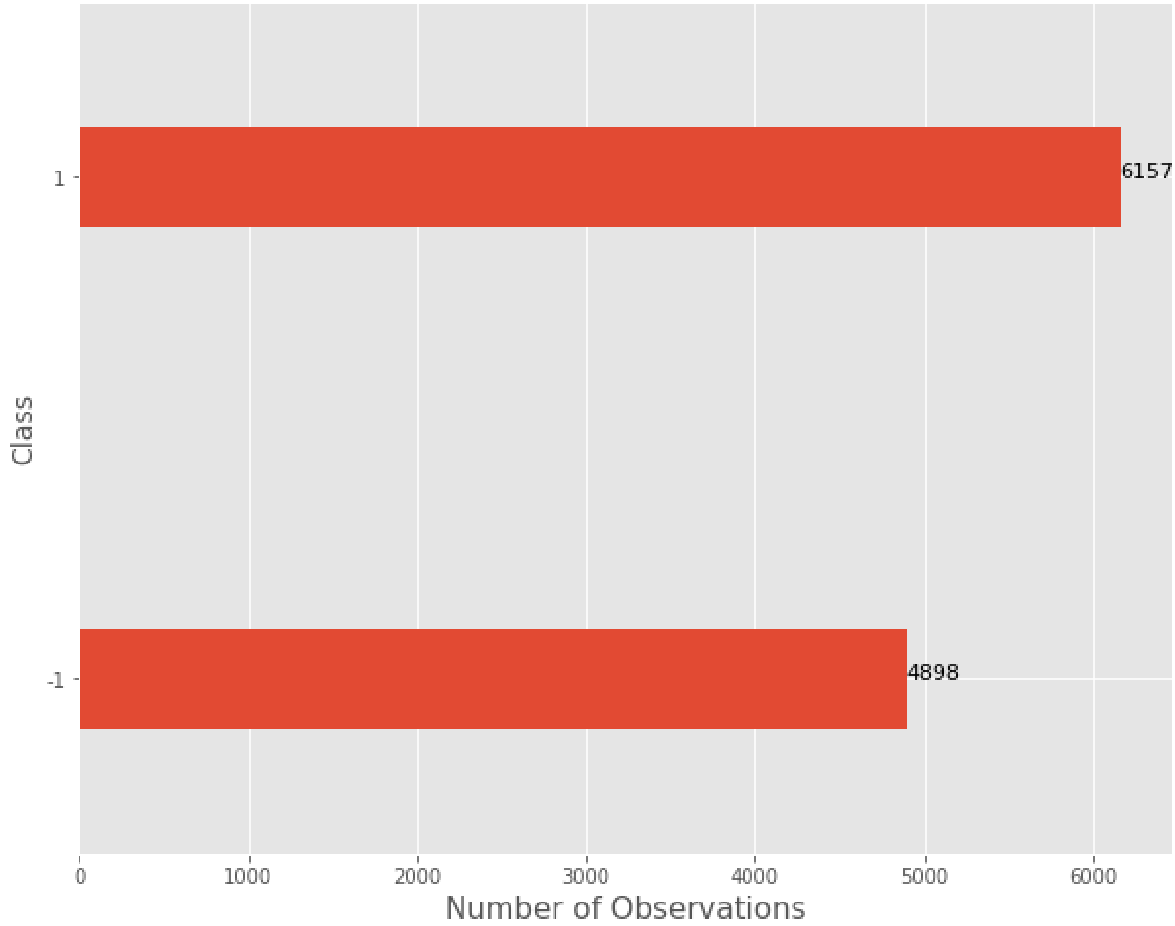
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [75]:

```
plt.style.use('ggplot')
```

In [76]:

```
subplot=class_dist.groupby("Class")["Num_Observation"].sum().plot(kind="barh",width=
subplot.set_xlabel("Number of Observations",fontsize=15)
subplot.set_ylabel("Class",fontsize=14)
for i in subplot.patches:
    subplot.text(i.get_width()+0.1,i.get_y()+0.1, str(i.get_width()),fontsize=11)
```



In [ ]:

"What is the range of the values present in the different columns, what are the unique values present in them and so on pandas we can use describe."

In [77]:

```
Data.describe().T# We describe the data. further description of the dataset
```

Out[77]:

	count	mean	std	min	25%	50%	75%	max
having_IP_Address	11055.0	0.313795	0.949534	-1.0	-1.0	1.0	1.0	1.0
URL_Length	11055.0	-0.633198	0.766095	-1.0	-1.0	-1.0	-1.0	1.0
Shortining_Service	11055.0	0.738761	0.673998	-1.0	1.0	1.0	1.0	1.0
having_At_Symbol	11055.0	0.700588	0.713598	-1.0	1.0	1.0	1.0	1.0
double_slash_redirecting	11055.0	0.741474	0.671011	-1.0	1.0	1.0	1.0	1.0
Prefix_Suffix	11055.0	-0.734962	0.678139	-1.0	-1.0	-1.0	-1.0	1.0
having_Sub_Domain	11055.0	0.063953	0.817518	-1.0	-1.0	0.0	1.0	1.0
SSLfinal_State	11055.0	0.250927	0.911892	-1.0	-1.0	1.0	1.0	1.0
Domain_registeration_length	11055.0	-0.336771	0.941629	-1.0	-1.0	-1.0	1.0	1.0
Favicon	11055.0	0.628584	0.777777	-1.0	1.0	1.0	1.0	1.0
port	11055.0	0.728268	0.685324	-1.0	1.0	1.0	1.0	1.0
HTTPS_token	11055.0	0.675079	0.737779	-1.0	1.0	1.0	1.0	1.0
Request_URL	11055.0	0.186793	0.982444	-1.0	-1.0	1.0	1.0	1.0
URL_of_Anchor	11055.0	-0.076526	0.715138	-1.0	-1.0	0.0	0.0	1.0
Links_in_tags	11055.0	-0.118137	0.763973	-1.0	-1.0	0.0	0.0	1.0
SFH	11055.0	-0.595749	0.759143	-1.0	-1.0	-1.0	-1.0	1.0
Submitting_to_email	11055.0	0.635640	0.772021	-1.0	1.0	1.0	1.0	1.0
Abnormal_URL	11055.0	0.705292	0.708949	-1.0	1.0	1.0	1.0	1.0
Redirect	11055.0	0.115694	0.319872	0.0	0.0	0.0	0.0	1.0
on_mouseover	11055.0	0.762099	0.647490	-1.0	1.0	1.0	1.0	1.0
RightClick	11055.0	0.913885	0.405991	-1.0	1.0	1.0	1.0	1.0
popUpWidnow	11055.0	0.613388	0.789818	-1.0	1.0	1.0	1.0	1.0
Iframe	11055.0	0.816915	0.576784	-1.0	1.0	1.0	1.0	1.0
age_of_domain	11055.0	0.061239	0.998168	-1.0	-1.0	1.0	1.0	1.0
DNSRecord	11055.0	0.377114	0.926209	-1.0	-1.0	1.0	1.0	1.0
web_traffic	11055.0	0.287291	0.827733	-1.0	0.0	1.0	1.0	1.0
Page_Rank	11055.0	-0.483673	0.875289	-1.0	-1.0	-1.0	1.0	1.0
Google_Index	11055.0	0.721574	0.692369	-1.0	1.0	1.0	1.0	1.0
Links_pointing_to_page	11055.0	0.344007	0.569944	-1.0	0.0	0.0	1.0	1.0
Statistical_report	11055.0	0.719584	0.694437	-1.0	1.0	1.0	1.0	1.0
Result	11055.0	0.113885	0.993539	-1.0	-1.0	1.0	1.0	1.0

In [78]:

```
Data.info()# Gives more information about the data as null
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 31 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   having_IP_Address            11055 non-null  int64
 1   URL_Length                   11055 non-null  int64
 2   Shortining_Service           11055 non-null  int64
 3   having_At_Symbol             11055 non-null  int64
 4   double_slash_redirecting     11055 non-null  int64
 5   Prefix_Suffix                11055 non-null  int64
 6   having_Sub_Domain            11055 non-null  int64
 7   SSLfinal_State               11055 non-null  int64
 8   Domain_registeration_length  11055 non-null  int64
 9   Favicon                      11055 non-null  int64
 10  port                         11055 non-null  int64
 11  HTTPS_token                  11055 non-null  int64
 12  Request_URL                  11055 non-null  int64
 13  URL_of_Anchor                11055 non-null  int64
 14  Links_in_tags                11055 non-null  int64
 15  SFH                          11055 non-null  int64
 16  Submitting_to_email          11055 non-null  int64
 17  Abnormal_URL                 11055 non-null  int64
 18  Redirect                     11055 non-null  int64
 19  on_mouseover                 11055 non-null  int64
 20  RightClick                   11055 non-null  int64
 21  popUpWidnow                  11055 non-null  int64
 22  Iframe                       11055 non-null  int64
 23  age_of_domain                11055 non-null  int64
 24  DNSRecord                    11055 non-null  int64
 25  web_traffic                  11055 non-null  int64
 26  Page_Rank                    11055 non-null  int64
 27  Google_Index                 11055 non-null  int64
 28  Links_pointing_to_page       11055 non-null  int64
 29  Statistical_report           11055 non-null  int64
 30  Result                       11055 non-null  int64
dtypes: int64(31)
memory usage: 2.6 MB
```

3. Cleaning the Class Labels and Inspecting for Missing Vaues

The aim id clean the data and split it into two parts. Training and Testing

Introduction

It is not good practice to create Machine Learning models using the labels with negative values. it affects the performance of the model hence we need to change the value -1 to be 0"

In [79]:

```
Data.rename(columns={"Result":"Class"},inplace=True)
Data["Class"]=Data["Class"].map({-1:0,1:1})
Data["Class"].unique()
```

Out[79]:

```
array([0, 1])
```

We now split the dataset into 80:20 ratio

In [80]:

```
from sklearn.model_selection import train_test_split
```

In [11]:

```
x=Data.iloc[:,0:30].values.astype(int)
y=Data.iloc[:,30].values.astype(int)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=np.rar
```

In [12]:

```
x
```

Out[12]:

```
array([[-1,  1,  1, ...,  1,  1, -1],
       [ 1,  1,  1, ...,  1,  1,  1],
       [ 1,  0,  1, ...,  1,  0, -1],
       ...,
       [ 1, -1,  1, ...,  1,  0,  1],
       [-1, -1,  1, ...,  1,  1,  1],
       [-1, -1,  1, ..., -1,  1, -1]])
```

In [13]:

```
y
```

Out[13]:

```
array([-1, -1, -1, ..., -1, -1, -1])
```

In [14]:

```
len(y_train)
```

Out[14]:

```
8844
```

In [15]:

```
8844+2211
```

Out[15]:

```
11055
```

In [16]:

```
2211/11055*100# this how we have picked our training set
```

Out[16]:

```
20.0
```

Let's Serialize the splits as well. Remember that our splits are now nothing but an array of valus and can be serialized"

In [17]:

```
# our data now does not have missing values
x_train
```

Out[17]:

```
array([[-1, -1,  1, ...,  1,  1,  1],
       [-1,  1,  1, ...,  1,  0,  1],
       [ 1, -1,  1, ...,  1,  0, -1],
       ...,
       [-1, -1,  1, ...,  1,  1,  1],
       [ 1, -1,  1, ...,  1,  1,  1],
       [-1, -1, -1, ...,  1,  1,  1]])
```

In [18]:

```
x_train
```

Out[18]:

```
array([[-1, -1,  1, ...,  1,  1,  1],
       [-1,  1,  1, ...,  1,  0,  1],
       [ 1, -1,  1, ...,  1,  0, -1],
       ...,
       [-1, -1,  1, ...,  1,  1,  1],
       [ 1, -1,  1, ...,  1,  1,  1],
       [-1, -1, -1, ...,  1,  1,  1]])
```

In [19]:

```
y_train
```

Out[19]:

```
array([ 1, -1, -1, ...,  1,  1, -1])
```

4. Training Logistics Regression Model"¶

We instatiate Logistic Regression Model and fit it to the training data

In [20]:

```
from sklearn.metrics import accuracy_score, precision_recall_fscore_support, classif
from sklearn.linear_model import LogisticRegression
import wandb
import time
```

We define some utility function for training machine learning model, with code to measure its training time performance

In [21]:

```
def train_eval_pipeline(model, train_data,test_data,name):
    #initialize weights and biases
    wandb.init(project="Phishing-websites-detection",name=name)
    #segragate the datasets
    (x_train, y_train)=train_data
    (x_test,y_test)=test_data
    #Train the model and keep the log of all the necessary metrics
    start=time.time()
    model.fit(x_train,y_train)
    end=time.time()-start
    prediction=model.predict(x_test)
    wandb.log({"accuracy": accuracy_score(y_test,prediction)*100.0,"precision": prec
    print("Accuracy score of the Logistic Regression classifier with default hyperpa
    print("\n")
    print("---Classificatin Report of the Logistic Regression classifier with defaul
    print("\n")
    print(classification_report(y_test,prediction,target_names=["Phishing Websites",
logreg=LogisticRegression()
train_eval_pipeline(logreg,(x_train,y_train),(x_test,y_test),"logistic_regression")
```

Logging results to [Weights & Biases (https://wandb.com)](https://wandb.com) [(Documentation) (https://docs.wandb.com/integrations/jupyter.html)](https://docs.wandb.com/integrations/jupyter.html).
Project page: [https://app.wandb.ai/python2020-/Phishing-websites-detection (https://app.wandb.ai/python2020-/Phishing-websites-detection)](https://app.wandb.ai/python2020-/Phishing-websites-detection)
Run page: [https://app.wandb.ai/python2020-/Phishing-websites-detection/runs/1abodntm (https://app.wandb.ai/python2020-/Phishing-websites-detection/runs/1abodntm)](https://app.wandb.ai/python2020-/Phishing-websites-detection/runs/1abodntm)

```
Accuracy score of the Logistic Regression classifier with default hype
rparameter values 93.71%


---Classificatin Report of the Logistic Regression classifier with def
ault hyperparameter values ----


                   precision    recall  f1-score   support

Phishing Websites       0.94      0.92      0.93       974
  Normal Websites       0.94      0.95      0.94      1237

         accuracy                           0.94      2211
        macro avg       0.94      0.94      0.94      2211
     weighted avg       0.94      0.94      0.94      2211
```

Improving the Model

Can we improve this model? Agood way to start approaching this idea is tune the hyperparameters of the model. We want to look at which is the best parameter for our model. We define the grid of values for the hyperparameter we would like to tune. In this case we use random search for hyperparameters tuning.

In [22]:

```
#import GridSearchCV if something goes outside the region we penalize it
from sklearn.model_selection import RandomizedSearchCV
```

In [23]:

```
#We define the grid
penalty=["l1","l2"]
C=[0.8,0.9,1.0]
tol=[0.01,0.001,0.0001]#what we can tolerate-tolerant values
max_iter=[100,150,200,250]# maximum iteration
```

In [26]:

```
#we create key value dist
param_grid=dict(penalty=penalty,C=C,tol=tol,max_iter=max_iter)
```

Now with the grid, we work to find the best set of values of hyperparameters values.

In [28]:

```
#Instanstiate RandomizedSearchCV with the required parameters.
param_grid=dict(penalty=penalty,C=C,tol=tol,max_iter=max_iter)
random_model=RandomizedSearchCV(estimator=logreg,param_distributions=param_grid, cv=
```

In [29]:

```
#Instanstiate RandomizedSearchCV with the required parameters.
random_model=RandomizedSearchCV(estimator=logreg,param_distributions=param_grid, cv=
random_model_result=random_model.fit(x_train,y_train)
```

```
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/skl
earn/model_selection/_validation.py:536: FitFailedWarning: Estimator f
it failed. The score on this train-test partition for these parameters
will be set to nan. Details:
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l
1 penalty.

  FitFailedWarning)
/Users/nelsonotumaongaya/opt/anaconda3/lib/python3.7/site-packages/skl
earn/model_selection/_validation.py:536: FitFailedWarning: Estimator f
it failed. The score on this train-test partition for these parameters
will be set to nan. Details:
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l
1 penalty.

  FitFailedWarning)
```

In [30]:

```
#summary of the results
best_score, best_params=random_model_result.best_score_,random_model_result.best_par
```

In [31]:

```
#summary of the results
best_score, best_params=random_model_result.best_score_,random_model_result.best_par
print("Best Score: %.2f using %s" %(best_score*100, best_params))
```

```
Best Score: 92.41 using {'tol': 0.01, 'penalty': 'l2', 'max_iter': 10
0, 'C': 0.9}
```

Random search did not help much in boosting up the accuracy score. Just to ensure that lets take the hyperparameter values and train another logistic regression model with the same values.

In [32]:

```
#log the hyperparameter values with which we are going to train our model.
config=wandb.config
config.tol=0.01
config.penalty="l2"
config.C=1.0
```

In [33]:

```
#Train the model
logreg=LogisticRegression(tol=config.tol,penalty=config.penalty,max_iter=250,C=confi
train_eval_pipeline(logreg,(x_train,y_train),(x_test,y_test),'Logistic-regression -r
```

Logging results to [Weights & Biases (https://wandb.com)](https://wandb.com) [(Documentation) (https://docs.wandb.com/integrations/jupyter.html)](https://docs.wandb.com/integrations/jupyter.html).
Project page: [https://app.wandb.ai/python2020-/Phishing-websites-detection (https://app.wandb.ai/python2020-/Phishing-websites-detection)](https://app.wandb.ai/python2020-/Phishing-websites-detection)
Run page: [https://app.wandb.ai/python2020-/Phishing-websites-detection/runs/1ntr66vp (https://app.wandb.ai/python2020-/Phishing-websites-detection/runs/1ntr66vp)](https://app.wandb.ai/python2020-/Phishing-websites-detection/runs/1ntr66vp)

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
 last)
<ipython-input-33-ef11bdd6b5c7> in <module>
      1 #Train the model
      2 logreg=LogisticRegression(tol=config.tol,penalty=config.penalt
y,max_iter=250,C=config.C)
----> 3 train_eval_pipeline(logreg,(x_train,y_train),(x_test,y_test),
'Logistic-regression -random-search')

<ipython-input-21-0998d3a485b0> in train_eval_pipeline(model, train_da
ta, test_data, name)
      7     #Train the model and keep the log of all the necessary met
rics
      8     start=time.time()
----> 9     model.fit(x_train,y_train)
     10     end=time.time()-start
     11     prediction=model.predict(x_test)

~/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/_logi
stic.py in fit(self, X, y, sample_weight)
   1486         The SAGA solver supports both float64 and float32 bit
 arrays.
   1487         """
-> 1488         solver = _check_solver(self.solver, self.penalty, self
.dual)
   1489 
   1490         if not isinstance(self.C, numbers.Number) or self.C <
0:

~/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/_logi
stic.py in _check_solver(solver, penalty, dual)
    439     if penalty not in all_penalties:
    440         raise ValueError("Logistic Regression supports only pe
nalties in %s,"
--> 441                          " got %s." % (all_penalties, penalt
y))
    442 
    443     if solver not in ['liblinear', 'saga'] and penalty not in
('l2', 'none'):

ValueError: Logistic Regression supports only penalties in ['l1', 'l
2', 'elasticnet', 'none'], got 12.
```

8. Random Forest Classifier-RFC

In [34]:

```
#print("Random Forest Classifier")
#forest_params = {"max_depth": list(range(10,50,1)),"n_estimators" : [350,400,450]}
#forest = GridSearchCV(RandomForestClassifier(), forest_params,n_jobs=-1,cv=10,scori
#forest.fit(x_train, y_train)
#random_forest = forest.best_estimator_
#print("Best Estimator")
#print(random_forest)
```

In [35]:

```
#Parameters have been choosing based on GridSearchCV
random_forest = RandomForestClassifier(max_depth=10,n_estimators=350)
random_forest.fit(x_train,y_train)
```

Out[35]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=Non
e,
                       criterion='gini', max_depth=10, max_features='a
uto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=N
one,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=350,
                       n_jobs=None, oob_score=False, random_state=Non
e,
                       verbose=0, warm_start=False)
```

In [37]:

```
forest_score = cross_val_score(random_forest, x_train, y_train, cv=10,scoring='roc_a
forest_score_teste = cross_val_score(random_forest, x_test, y_test, cv=10,scoring='r
print('Score RFC Train: ', round(forest_score.mean() * 100, 2).astype(str) + '%')
print('Score RFC Test: ', round(forest_score_teste.mean() * 100, 2).astype(str) + '%
```

```
Score RFC Train:  99.25%
Score RFC Test:  99.16%
```

In [38]:

```
y_pred_rf = random_forest.predict(x_test)
```

In [39]:

```
cm_rf = confusion_matrix(y_test,y_pred_rf)
```

In [51]:

```
acc_score_rf = accuracy_score(y_test,y_pred_rf)
f1_score_rf = f1_score(y_test,y_pred_rf)
pred_rf = average_precision_score(y_test,y_pred_rf)
recall_rf = recall_score(y_test,y_pred_rf)
roc_rf = roc_auc_score(y_test,y_pred_rf,multi_class='ovo')
print('Accuracy Random Forest ',round(acc_score_rf*100,2).astype(str)+'%')
print('Pred media Random Forest ',round(pred_rf*100,2).astype(str)+'%')
print('F1 Random Forest ',round(f1_score_rf*100,2).astype(str)+'%')
print('Recall Random Forest ',round(recall_rf*100,2).astype(str)+'%')
print('ROC Random Forest ',round(roc_rf*100,2).astype(str)+'%')
```

```
Accuracy Random Forest  96.38%
Pred media Random Forest  94.93%
F1 Random Forest  96.8%
Recall Random Forest  97.82%
ROC Random Forest  96.19%
```

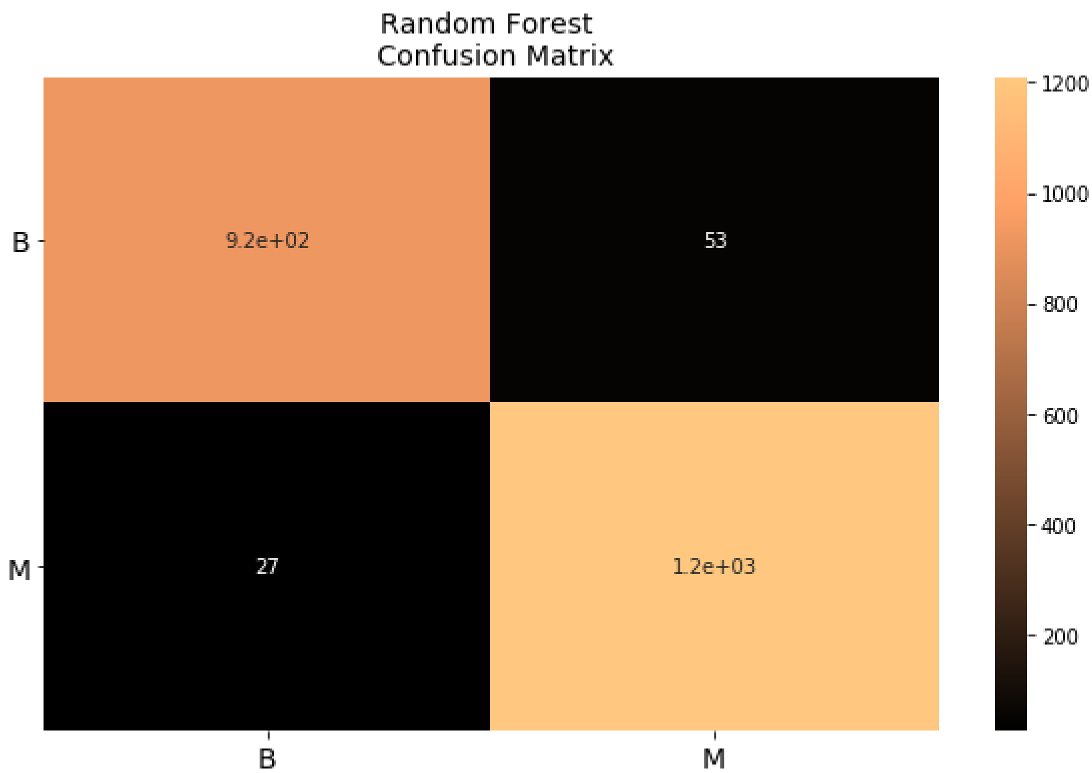
The accuracy in Random Forest for Website Phishing is 96.38%

In [55]:

```
fig, ax = plt.subplots(figsize=(10,6))
sns.heatmap(cm_rf, ax=ax, annot=True, cmap=plt.cm.copper)
ax.set_title("Random Forest \n Confusion Matrix", fontsize=14)
ax.set_xticklabels(['B', 'M'], fontsize=14, rotation=0)
ax.set_yticklabels(['B', 'M'], fontsize=14, rotation=360)
```

Out[55]:

```
[Text(0, 0.5, 'B'), Text(0, 1.5, 'M')]
```



Ada Boost Classifier

In [56]:

```
#print("Ada Boost Classifier")
#ada_params = {'n_estimators' : list(range(100,200))}
#grid_ada = GridSearchCV(AdaBoostClassifier(), ada_params,n_jobs=8,cv=10,scoring='rc
#grid_ada.fit(X_train, y_train)
#ada = grid_ada.best_estimator_
#print("Best Estimator")
#print(ada)
```

In [57]:

```
#Parameters have been choosing based on GridSearchCV
ada = AdaBoostClassifier(n_estimators=102)
ada.fit(x_train,y_train)
```

Out[57]:

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_
rate=1.0,
                   n_estimators=102, random_state=None)
```

In [58]:

```
ada_score = cross_val_score(ada, x_train, y_train, cv=10,scoring='roc_auc_ovo')
ada_score_teste = cross_val_score(ada, x_test, y_test, cv=10,scoring='roc_auc_ovo')
print('Score AdaBoost Train: ', round(ada_score.mean() * 100, 2).astype(str) + '%')
print('Score AdaBoost Test: ', round(ada_score_teste.mean() * 100, 2).astype(str) +
```

```
Score AdaBoost Train:  98.61%
Score AdaBoost Test:  98.81%
```

In [59]:

```
y_pred_ada = ada.predict(x_test)
```

In [60]:

```
cm_ada = confusion_matrix(y_test,y_pred_ada)
```

In [64]:

```
acc_score_ada = accuracy_score(y_test,y_pred_ada)
f1_score_ada = f1_score(y_test,y_pred_ada)
precisao_ada = average_precision_score(y_test,y_pred_ada)
recall_ada = recall_score(y_test,y_pred_ada)
roc_ada = roc_auc_score(y_test,y_pred_ada,multi_class='ovo')
print('Acuracy ADA Boost ',round(acc_score_ada*100,2).astype(str)+'%')
print('Pred media Ada Boost ',round(precisao_ada*100,2).astype(str)+'%')
print('F1 Ada Boost ',round(f1_score_ada*100,2).astype(str)+'%')
print('Recall Ada Boost ',round(recall_ada*100,2).astype(str)+'%')
print('ROC Ada Boost ',round(roc_ada*100,2).astype(str)+'%')
```

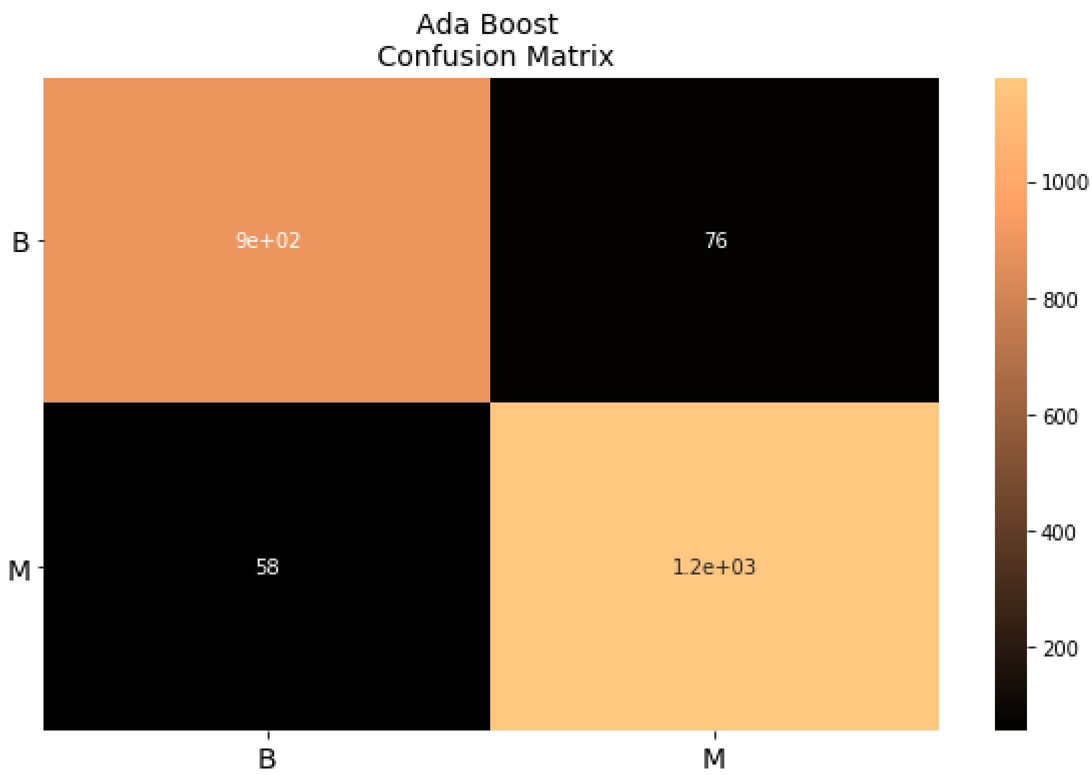
```
Acuracy ADA Boost  93.94%
Pred media Ada Boost  92.16%
F1 Ada Boost  94.62%
Recall Ada Boost  95.31%
ROC Ada Boost  93.75%
```

In [66]:

```
fig, ax = plt.subplots(figsize=(10,6))
sns.heatmap(cm_ada, ax=ax, annot=True, cmap=plt.cm.copper)
ax.set_title("Ada Boost \n Confusion Matrix", fontsize=14)
ax.set_xticklabels(['B', 'M'], fontsize=14, rotation=0)
ax.set_yticklabels(['B', 'M'], fontsize=14, rotation=360)
```

Out[66]:

```
[Text(0, 0.5, 'B'), Text(0, 1.5, 'M')]
```



9. Gradient Boost Classifier- GBC

In [67]:

```
#print("Gradient Boost Classifier")
#grad_params = {'n_estimators' : [50,55,60,65,70,75,80,85,90],'max_depth' : list(ran
#grad = GridSearchCV(GradientBoostingClassifier(), grad_params,n_jobs=-1,cv=10,scori
#grad.fit(X_train, y_train)
#grad_boost = grad.best_estimator_
#print("Best Estimator")
#print(grad_boost)
```

In [68]:

```
#Parameters have been choosing based on GridSearchCV
grad_boost = GradientBoostingClassifier(n_estimators=65,max_depth=4)
grad_boost.fit(x_train, y_train)
```

Out[68]:

```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', in
it=None,
                           learning_rate=0.1, loss='deviance', max_dep
th=4,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_spl
it=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=
65,
                           n_iter_no_change=None, presort='deprecate
d',
                           random_state=None, subsample=1.0, tol=0.000
1,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

In [69]:

```
grad_score = cross_val_score(grad_boost, x_train, y_train, cv=10,scoring='roc_auc_ov
grad_score_teste = cross_val_score(grad_boost, x_test, y_test, cv=10,scoring='roc_au
print('Score GradBoost Train: ', round(grad_score.mean() * 100, 2).astype(str) + '%'
print('Score GradBoost Test: ', round(grad_score_teste.mean() * 100, 2).astype(str)
```

```
Score GradBoost Train:  99.0%
Score GradBoost Test:  99.02%
```

In [70]:

```
y_pred_gb = grad_boost.predict(x_test)
```

In [71]:

```
cm_gb = confusion_matrix(y_test,y_pred_gb)
```

In [73]:

```
acc_score_gb = accuracy_score(y_test,y_pred_gb)
f1_score_gb = f1_score(y_test,y_pred_gb)
pred_gb = average_precision_score(y_test,y_pred_gb)
recall_gb = recall_score(y_test,y_pred_gb)
roc_gb = roc_auc_score(y_test,y_pred_gb,multi_class='ovo')
print('Acuracy Gradient Boosting ',round(acc_score_gb*100,2).astype(str)+'%')
print('Pred media Gradient Boosting  ',round(pred_gb*100,2).astype(str)+'%')
print('F1 Gradient Boosting  ',round(f1_score_gb*100,2).astype(str)+'%')
print('Recall Gradient Boosting  ',round(recall_gb*100,2).astype(str)+'%')
print('ROC Gradient Boosting ',round(roc_gb*100,2).astype(str)+'%')
```

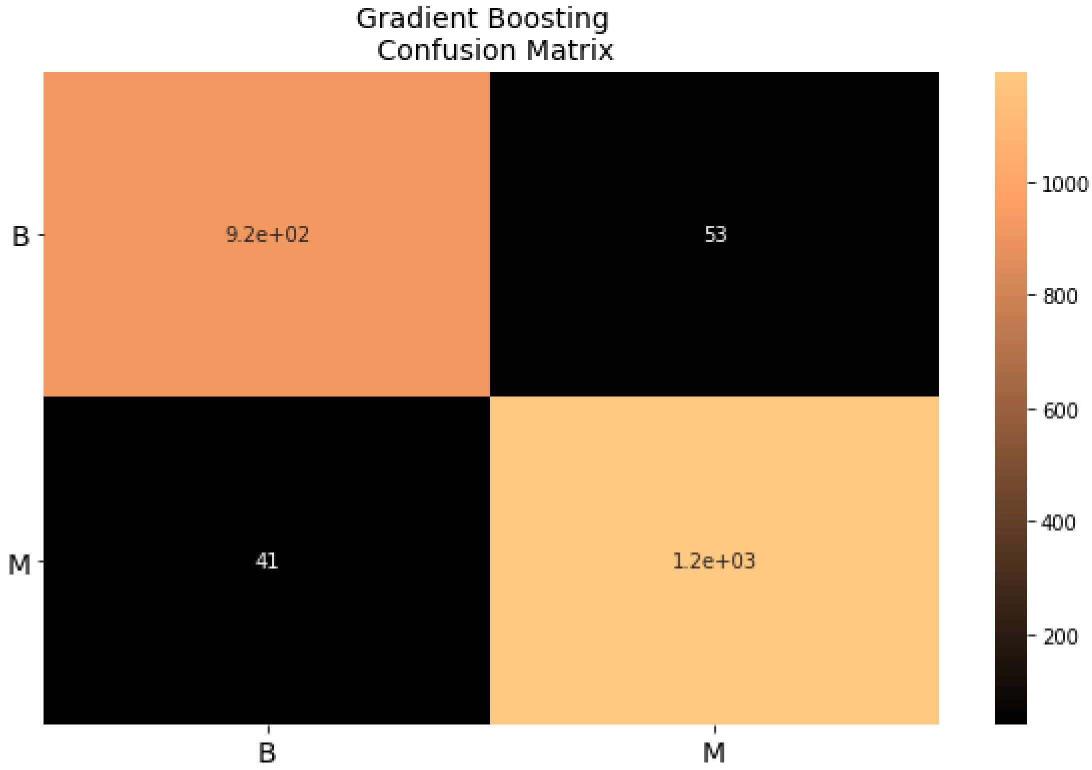
```
Acuracy Gradient Boosting  95.75%
Pred media Gradient Boosting   94.44%
F1 Gradient Boosting   96.22%
Recall Gradient Boosting   96.69%
ROC Gradient Boosting  95.62%
```

In [74]:

```
fig, ax = plt.subplots(figsize=(10,6))
sns.heatmap(cm_gb, ax=ax, annot=True, cmap=plt.cm.copper)
ax.set_title("Gradient Boosting  \n Confusion Matrix", fontsize=14)
ax.set_xticklabels(['B', 'M'], fontsize=14, rotation=0)
ax.set_yticklabels(['B', 'M'], fontsize=14, rotation=360)
```

Out[74]:

```
[Text(0, 0.5, 'B'), Text(0, 1.5, 'M')]
```

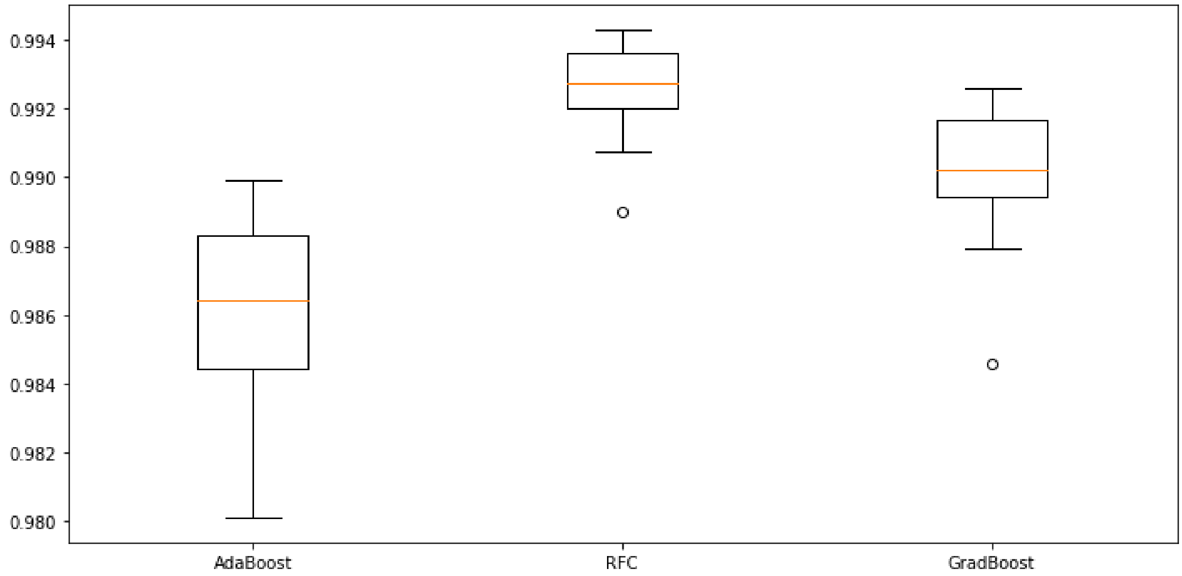


In [80]:

```
results = [ada_score,forest_score,grad_score]
results_test = [ada_score_teste,forest_score_teste,grad_score_teste]
name_model = ["AdaBoost","RFC","GradBoost"]
```

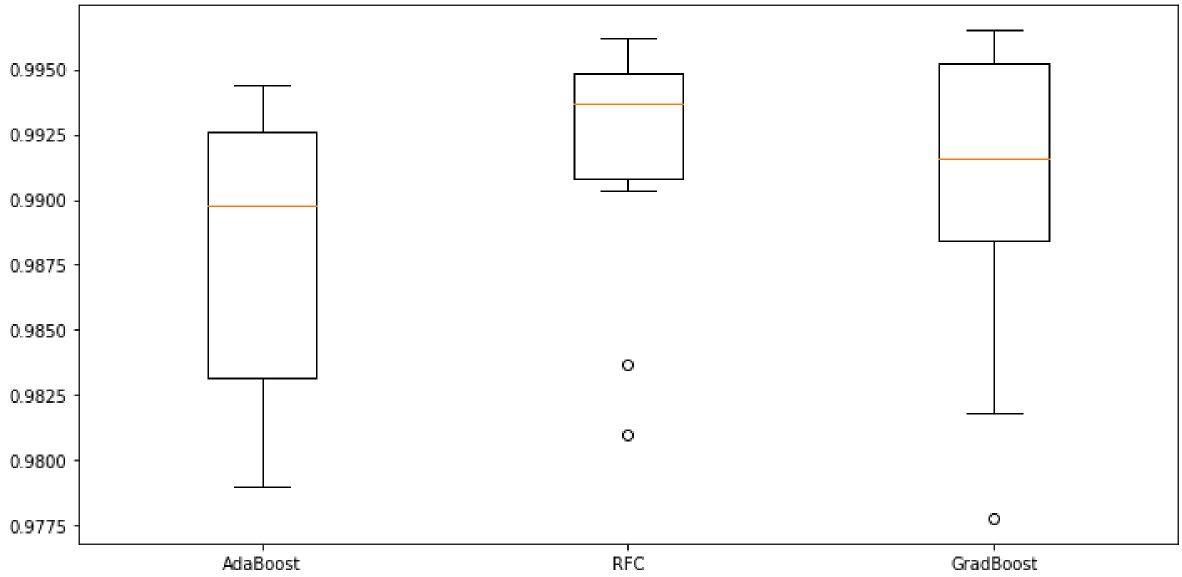
In [81]:

```
fig,ax=plt.subplots(figsize=(10,5))
ax.boxplot(results)
ax.set_xticklabels(name_model)
plt.tight_layout()
```



In [82]:

```
fig,ax=plt.subplots(figsize=(10,5))
ax.boxplot(results_test)
ax.set_xticklabels(name_model)
plt.tight_layout()
```



In [ ]:

```

```