Bot

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\bot.csv")
df
```

			against natural majori...							15:29:50
1	289683	hinesstephanie	Authority research natural life material staff...	55	5	9617	True	0	Sanderston	2022-11-26 05:18:10
2	779715	roberttran	Manage whose quickly especially foot none to g...	6	2	4363	True	0	Harrisonfurt	2022-08-08 03:16:54
3	696168	pmason	Just cover eight opportunity strong policy which.	54	5	2242	True	1	Martinezberg	2021-08-14 22:27:05

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   User ID         50000 non-null  int64 
 1   Username        50000 non-null  object
 2   Tweet           50000 non-null  object
 3   Retweet Count   50000 non-null  int64 
 4   Mention Count   50000 non-null  int64 
 5   Follower Count  50000 non-null  int64 
 6   Verified        50000 non-null  bool  
 7   Bot Label       50000 non-null  int64 
 8   Location        50000 non-null  object
 9   Created At      50000 non-null  object
 10  Hashtags        41659 non-null  object
dtypes: bool(1), int64(5), object(5)
memory usage: 3.9+ MB
```

In [5]:
```
df1=df[['Retweet Count','Mention Count','Follower Count','Bot Label']][0:50]
```

In [6]:
```
df1.fillna(value=1)
```

Out[6]:

	Retweet Count	Mention Count	Follower Count	Bot Label
0	85	1	2353	1
1	55	5	9617	0
2	6	2	4363	0
3	54	5	2242	1
4	26	3	8438	1
5	41	4	3792	1
6	54	0	10	0
7	64	0	1442	1
8	25	2	836	0
9	67	3	6523	1
10	57	4	8694	1
11	29	1	5986	1
12	60	2	6779	0
13	61	0	6073	0
14	21	2	4846	0
15	78	1	2342	0
16	64	5	6947	1
17	43	4	7945	0
18	39	0	8305	1
19	8	2	1256	0
20	26	4	653	1
21	84	4	5466	0
22	86	1	5131	1
23	55	0	5278	1
24	56	1	3347	1
25	43	2	4851	0
26	49	5	7600	1
27	7	0	706	1
28	75	1	7313	1
29	39	0	337	1
30	77	1	7006	0
31	40	2	697	0
32	25	5	4517	0
33	15	2	1785	0
34	85	0	4057	1
35	13	0	7925	0
36	63	4	2804	1
37	75	4	3544	1
38	58	1	2063	0
39	34	5	466	0
40	66	1	2852	1

	Retweet Count	Mention Count	Follower Count	Bot Label
41	18	0	3782	0
42	0	3	4581	0
43	21	3	6979	0
44	7	3	7523	0
45	39	0	3755	0
46	34	4	6933	0
47	72	0	8386	1
48	24	5	4096	1
49	77	0	7967	1

In [7]:
```
feature_matrix = df1.iloc[:,0:3]
target_vector = df1.iloc[:,-1]
```

In [8]:
```
feature_matrix.shape
```

Out[8]: (50, 3)

In [9]:
```
target_vector.shape
```

Out[9]: (50,)

In [10]:
```
from sklearn.preprocessing import StandardScaler
```

In [11]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [12]:
```
logr=LogisticRegression()
```

In [13]:
```
logr.fit(fs,target_vector)
```

Out[13]: LogisticRegression()

In [14]:
```
observation=[[3,4,4]]
```

In [15]:
```
prediction = logr.predict(observation)
print(prediction)
```

```
[1]
```

In [16]:
```
logr.classes_
```

Out[16]: array([0, 1], dtype=int64)

In [17]:
```
logr.predict_proba(observation)[0][0]
```

Out[17]: 0.08362498004517882

In [18]:
```
logr.predict_proba(observation)[0][1]
```

Out[18]: 0.9163750199548212

Logistic Regression-2

In [19]:
```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

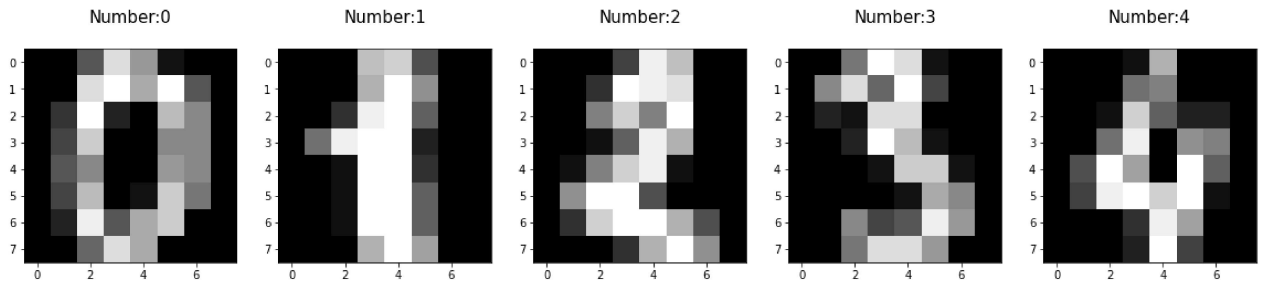
In [20]:
```
digits=load_digits()
digits
```

```
array([[[ 0.,  0.,  1., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ...,  2.,  1.,  0.],
        [ 0.,  0., 16., ..., 16.,  5.,  0.],
        ...,
        [ 0.,  0., 16., ..., 15.,  0.,  0.],
        [ 0.,  0., 15., ..., 16.,  0.,  0.],
        [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

       [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
        [ 0.,  0., 14., ..., 15.,  1.,  0.],
        [ 0.,  4., 16., ..., 16.,  7.,  0.],
        ...,
        [ 0.,  0.,  0., ..., 16.,  2.,  0.],
        [ 0.,  0.,  4., ..., 16.,  2.,  0.],
        [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

       [[ 0.,  0., 10., ...,  1.,  0.,  0.],
        [ 0.,  2., 16., ...,  1.,  0.,  0.],
        [ 0.,  0., 15., ..., 15.,  0.,  0.],
        ...,
```

In [21]:
```
plt.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:5])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title("Number:%i\n"%label,fontsize=15)
```



In [22]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30)
```

In [23]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1257, 64)
(540, 64)
(1257,)
(540,)
```

In [24]:
```
logr=LogisticRegression(max_iter=10000)
```

In [25]:
```
logr.fit(x_train,y_train)
```

Out[25]:
```
LogisticRegression(max_iter=10000)
```

In [26]:
```
print(logr.predict(x_test))
```

```
[2 5 0 5 8 5 1 1 0 4 5 1 2 9 5 6 4 4 6 0 2 2 8 2 7 9 4 4 2 6 7 5 2 3 7 4 6
 5 3 7 9 0 5 6 4 0 6 9 5 1 5 5 4 9 9 7 0 5 1 7 5 1 0 7 6 2 4 4 2 5 6 2 0 8
 3 2 2 9 1 2 6 3 9 0 8 6 6 3 3 9 3 0 9 6 8 6 5 4 8 2 2 9 9 5 1 6 5 6 0 5 2
 0 2 7 2 4 3 8 2 9 9 1 0 6 1 4 2 7 9 0 0 0 6 3 4 1 4 5 2 8 2 3 5 2 6 6 3 1
 1 1 7 4 0 4 6 1 4 2 4 5 1 9 5 3 6 9 0 3 2 2 2 0 6 6 3 6 2 1 5 6 7 7 0 1 7
 4 5 4 8 9 9 9 4 3 2 7 8 6 8 8 6 2 7 4 2 7 2 1 3 1 1 9 0 9 9 4 1 2 3 1 0 0
 5 3 1 7 3 1 0 4 6 7 5 3 6 2 8 8 8 0 9 5 4 6 8 8 7 0 9 2 8 5 2 4 7 5 1 8 6
 5 7 2 6 1 2 0 2 4 8 8 6 4 1 3 9 6 7 6 3 7 0 8 9 4 1 8 4 7 2 0 5 0 8 0 9 2
 4 9 7 9 3 0 7 1 7 1 6 0 0 9 6 9 9 3 0 3 1 1 2 4 5 6 8 7 1 3 6 7 7 8 2 3 2
 8 2 3 4 0 9 2 1 4 6 1 3 1 1 8 4 0 9 1 4 0 4 8 7 6 3 7 2 9 1 3 3 7 2 4 5 1
 5 1 4 9 2 3 2 8 1 5 7 9 0 8 8 3 5 7 8 4 3 6 1 8 1 7 5 8 7 5 0 7 5 3 6 6 6
 8 4 3 0 9 4 5 3 7 6 4 3 8 5 9 4 0 7 1 2 2 2 2 0 0 0 2 3 8 5 1 6 3 0 9 5 5
 7 3 4 9 7 8 0 5 7 4 7 4 0 3 3 8 4 6 3 6 0 0 6 7 7 4 1 0 3 4 5 0 5 4 8 7 4
 7 1 8 0 1 2 7 3 1 7 2 8 0 5 6 4 1 2 4 5 4 7 6 0 5 4 8 5 0 6 7 7 1 6 2 9 3
 3 6 6 6 9 6 4 4 8 7 3 0 5 1 8 3 5 5 7 0 0 0]
```

In [27]:
```
print(logr.score(x_test,y_test))
```

```
0.9666666666666667
```

Random Forest

In [28]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [43]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\ionosphere.csv")
df
```

Out[43]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171	0.41078	-0.461(
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.184(
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.221(
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.000(
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.532(
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03240	0.092(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0.001(
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0.049(
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0.025(
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0.077(
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0.048(

350 rows × 35 columns

In [49]:
```
df['g'].value_counts()
```

Out[49]:
```
g    224
b    126
Name: g, dtype: int64
```

In [50]:
```
x=df.drop('g',axis=1)
y=df['g']
```

In [51]:
```
g1={"g":{1:"g",2:"b"}}
df=df.replace(Bot_Label1)
print(df)
```

```
     1  0  0.99539  -0.05889  0.85243  0.02306  0.83398  -0.37708      1.1  \
0    1  0  1.00000  -0.18829  0.93035 -0.36156 -0.10868  -0.93597  1.00000
1    1  0  1.00000  -0.03365  1.00000  0.00485  1.00000  -0.12062  0.88965
2    1  0  1.00000  -0.45161  1.00000  1.00000  0.71216  -1.00000  0.00000
3    1  0  1.00000  -0.02401  0.94140  0.06531  0.92106  -0.23255  0.77152
4    1  0  0.02337  -0.00592 -0.09924 -0.11949 -0.00763  -0.11824  0.14706
..  .. ..      ...       ...      ...      ...      ...       ...      ...
345  1  0  0.83508   0.08298  0.73739 -0.14706  0.84349  -0.05567  0.90441
346  1  0  0.95113   0.00419  0.95183 -0.02723  0.93438  -0.01920  0.94590
347  1  0  0.94701  -0.00034  0.93207 -0.03227  0.95177  -0.03431  0.95584
348  1  0  0.90608  -0.01657  0.98122 -0.01989  0.95691  -0.03646  0.85746
349  1  0  0.84710   0.13533  0.73638 -0.06151  0.87873   0.08260  0.88928

     0.03760  ...  -0.51171  0.41078  -0.46168  0.21266  -0.34090  0.42267  \
0   -0.04549  ...  -0.26569 -0.20468  -0.18401 -0.19040  -0.11593 -0.16626
1    0.01198  ...  -0.40220  0.58984  -0.22145  0.43100  -0.17365  0.60436
2    0.00000  ...   0.90695  0.51613   1.00000  1.00000  -0.20099  0.25682
3   -0.16399  ...  -0.65158  0.13290  -0.53206  0.02431  -0.62197 -0.05707
4    0.06637  ...  -0.01535 -0.03240   0.09223 -0.07859   0.00732  0.00000
..       ...  ...       ...      ...       ...      ...       ...      ...
345 -0.04622  ...  -0.04202  0.83479   0.00123  1.00000   0.12815  0.86660
346  0.01606  ...   0.01361  0.93522   0.04925  0.93159   0.08168  0.94066
347  0.02446  ...   0.03193  0.92489   0.02542  0.92120   0.02242  0.92459
348  0.00110  ...  -0.02099  0.89147  -0.07760  0.82983  -0.17238  0.96022
349 -0.09139  ...  -0.15114  0.81147  -0.04822  0.78207  -0.00703  0.75747

     -0.54487  0.18641  -0.45300  g
0    -0.06288 -0.13738  -0.02447  b
1    -0.24180  0.56045  -0.38238  g
2     1.00000 -0.32382   1.00000  b
3    -0.59573 -0.04608  -0.65697  g
4     0.00000 -0.00039   0.12011  b
..        ...      ...       ... ..
345  -0.10714  0.90546  -0.04307  g
346  -0.00035  0.91483   0.04712  g
347   0.00442  0.92697  -0.00577  g
348  -0.03757  0.87403  -0.16243  g
349  -0.06678  0.85764  -0.06151  g

[350 rows x 35 columns]
```

In [52]:
```
from sklearn.model_selection import train_test_split
```

In [53]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

In [54]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [55]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[55]:
```
RandomForestClassifier()
```

In [56]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [57]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[57]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [58]:
```
grid_search.best_score_
```

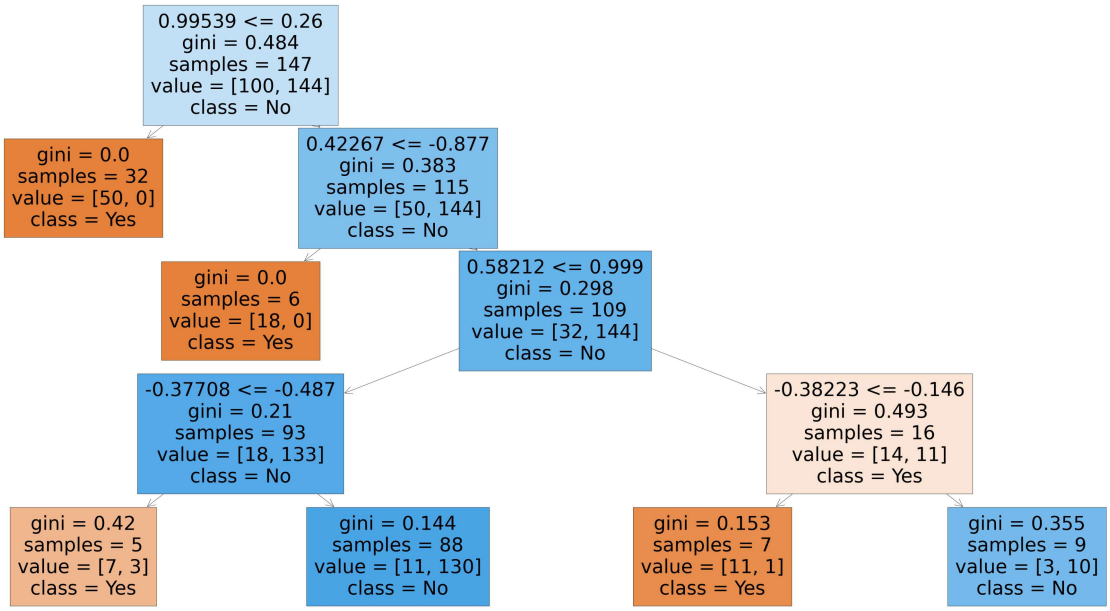
Out[58]:
```
0.9262295081967213
```

In [59]:
```
from sklearn.tree import plot_tree
```

In [60]:
```
rfc_best=grid_search.best_estimator_
```

In [61]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True
```

Out[61]:
```
[Text(1116.0, 1956.96, '0.99539 <= 0.26\ngini = 0.484\nsamples = 147\nvalue = [100, 144]\ncla
ss = No'),
 Text(558.0, 1522.0800000000002, 'gini = 0.0\nsamples = 32\nvalue = [50, 0]\nclass = Yes'),
 Text(1674.0, 1522.0800000000002, '0.42267 <= -0.877\ngini = 0.383\nsamples = 115\nvalue = [5
0, 144]\nclass = No'),
 Text(1116.0, 1087.2, 'gini = 0.0\nsamples = 6\nvalue = [18, 0]\nclass = Yes'),
 Text(2232.0, 1087.2, '0.58212 <= 0.999\ngini = 0.298\nsamples = 109\nvalue = [32, 144]\nclas
s = No'),
 Text(1116.0, 652.3200000000002, '-0.37708 <= -0.487\ngini = 0.21\nsamples = 93\nvalue = [18,
133]\nclass = No'),
 Text(558.0, 217.44000000000005, 'gini = 0.42\nsamples = 5\nvalue = [7, 3]\nclass = Yes'),
 Text(1674.0, 217.44000000000005, 'gini = 0.144\nsamples = 88\nvalue = [11, 130]\nclass = N
o'),
 Text(3348.0, 652.3200000000002, '-0.38223 <= -0.146\ngini = 0.493\nsamples = 16\nvalue = [1
4, 11]\nclass = Yes'),
 Text(2790.0, 217.44000000000005, 'gini = 0.153\nsamples = 7\nvalue = [11, 1]\nclass = Yes'),
 Text(3906.0, 217.44000000000005, 'gini = 0.355\nsamples = 9\nvalue = [3, 10]\nclass = No')]
```



In [ ]:
```
```