Vijay 02-09-2023

In [195]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [196]:
```
from sklearn.linear_model import LogisticRegression
a=pd.read_csv(r"C:\USERS\user\Downloads\C9_Data.csv")
a
```

0	0	18	2022-07-29 09:08:54	7
1	1	18	2022-07-29 09:09:54	9
2	2	18	2022-07-29 09:09:54	9
3	3	18	2022-07-29 09:10:06	5
4	4	18	2022-07-29 09:10:08	5
...	...	...	...	...
37513	37513	6	2022-12-31 20:38:56	11
37514	37514	6	2022-12-31 20:39:22	6
37515	37515	6	2022-12-31 20:39:23	6
37516	37516	6	2022-12-31 20:39:31	9
37517	37517	6	2022-12-31 20:39:31	9

37518 rows × 4 columns

In [197]:

```
1 a=a.head(60)
2 a
```

Out[197]:

	row_id	user_id	timestamp	gate_id
0	0	18	2022-07-29 09:08:54	7
1	1	18	2022-07-29 09:09:54	9
2	2	18	2022-07-29 09:09:54	9
3	3	18	2022-07-29 09:10:06	5
4	4	18	2022-07-29 09:10:08	5
5	5	18	2022-07-29 09:10:34	10
6	6	18	2022-07-29 09:32:47	11
7	7	18	2022-07-29 09:33:12	4
8	8	18	2022-07-29 09:33:13	4
9	9	1	2022-07-29 09:33:16	7
10	10	18	2022-07-29 09:33:23	9
11	11	18	2022-07-29 09:33:23	9
12	12	18	2022-07-29 09:33:41	5
13	13	18	2022-07-29 09:33:42	5
14	14	18	2022-07-29 09:34:04	10
15	15	1	2022-07-29 09:34:18	9
16	16	1	2022-07-29 09:34:18	9
17	17	1	2022-07-29 09:34:32	5
18	18	1	2022-07-29 09:34:33	5
19	19	1	2022-07-29 09:35:00	10
20	20	3	2022-07-29 09:40:40	7
21	21	3	2022-07-29 09:42:49	9
22	22	3	2022-07-29 09:42:49	9
23	23	3	2022-07-29 09:43:01	5
24	24	3	2022-07-29 09:43:03	5
25	25	3	2022-07-29 09:43:29	10
26	26	6	2022-07-29 09:53:22	7
27	27	29	2022-07-29 09:53:44	7
28	28	29	2022-07-29 09:53:46	7
29	29	6	2022-07-29 09:54:25	9
30	30	6	2022-07-29 09:54:25	9
31	31	6	2022-07-29 09:54:35	5
32	32	6	2022-07-29 09:54:37	5
33	33	6	2022-07-29 09:54:57	10
34	34	29	2022-07-29 09:56:04	9
35	35	29	2022-07-29 09:56:04	9
36	36	29	2022-07-29 09:56:12	5
37	37	29	2022-07-29 09:56:14	5
38	38	18	2022-07-29 09:56:31	12

	row_id	user_id	timestamp	gate_id
39	39	18	2022-07-29 09:56:33	12
40	40	29	2022-07-29 09:56:41	10
41	41	55	2022-07-29 10:09:23	7
42	42	55	2022-07-29 10:10:28	3
43	43	55	2022-07-29 10:10:30	3
44	44	55	2022-07-29 10:10:50	10
45	45	24	2022-07-29 10:12:52	15
46	46	24	2022-07-29 10:15:52	3
47	47	24	2022-07-29 10:15:55	3
48	48	24	2022-07-29 10:16:19	10
49	49	24	2022-07-29 10:17:48	11
50	50	24	2022-07-29 10:18:13	4
51	51	24	2022-07-29 10:18:14	4
52	52	24	2022-07-29 10:18:25	9
53	53	24	2022-07-29 10:18:25	9
54	54	24	2022-07-29 10:18:42	5
55	55	24	2022-07-29 10:18:44	5
56	56	24	2022-07-29 10:19:05	10
57	57	39	2022-07-29 10:20:52	9
58	58	39	2022-07-29 10:20:52	9
59	59	39	2022-07-29 10:21:18	5

In [198]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [199]:
```
1 a.columns
```

Out[199]:
```
Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')
```

In [200]:

```
1 b=a[['row_id', 'user_id', 'gate_id']]
2 b
```

Out[200]:

	row_id	user_id	gate_id
0	0	18	7
1	1	18	9
2	2	18	9
3	3	18	5
4	4	18	5
5	5	18	10
6	6	18	11
7	7	18	4
8	8	18	4
9	9	1	7
10	10	18	9
11	11	18	9
12	12	18	5
13	13	18	5
14	14	18	10
15	15	1	9
16	16	1	9
17	17	1	5
18	18	1	5
19	19	1	10
20	20	3	7
21	21	3	9
22	22	3	9
23	23	3	5
24	24	3	5
25	25	3	10
26	26	6	7
27	27	29	7
28	28	29	7
29	29	6	9
30	30	6	9
31	31	6	5
32	32	6	5
33	33	6	10
34	34	29	9
35	35	29	9
36	36	29	5
37	37	29	5
38	38	18	12

	row_id	user_id	gate_id
39	39	18	12
40	40	29	10
41	41	55	7
42	42	55	3
43	43	55	3
44	44	55	10
45	45	24	15
46	46	24	3
47	47	24	3
48	48	24	10
49	49	24	11
50	50	24	4
51	51	24	4
52	52	24	9
53	53	24	9
54	54	24	5
55	55	24	5
56	56	24	10
57	57	39	9
58	58	39	9
59	59	39	5

In [201]:
```
1 c=b.iloc[:,0:3]
2 d=b.iloc[:,-1]
```

In [202]:
```
1 c.shape
```

Out[202]: (60, 3)

In [203]:
```
1 d.shape
```

Out[203]: (60,)

In [204]:

```
from sklearn.preprocessing import StandardScaler
fs=StandardScaler().fit_transform(c)
fs
```

Out[204]:
```
array([[-1.7034199 , -0.11374344, -0.1663911 ],
       [-1.64567685, -0.11374344,  0.57312489],
       [-1.5879338 , -0.11374344,  0.57312489],
       [-1.53019075, -0.11374344, -0.90590709],
       [-1.47244771, -0.11374344, -0.90590709],
       [-1.41470466, -0.11374344,  0.94288289],
       [-1.35696161, -0.11374344,  1.31264088],
       [-1.29921857, -0.11374344, -1.27566508],
       [-1.24147552, -0.11374344, -1.27566508],
       [-1.18373247, -1.32226746, -0.1663911 ],
       [-1.12598942, -0.11374344,  0.57312489],
       [-1.06824638, -0.11374344,  0.57312489],
       [-1.01050333, -0.11374344, -0.90590709],
       [-0.95276028, -0.11374344, -0.90590709],
       [-0.89501723, -0.11374344,  0.94288289],
       [-0.83727419, -1.32226746,  0.57312489],
       [-0.77953114, -1.32226746,  0.57312489],
       [-0.72178809, -1.32226746, -0.90590709],
       [-0.66404504, -1.32226746, -0.90590709],
       [-0.606302  , -1.32226746,  0.94288289],
       [-0.54855895, -1.18008816, -0.1663911 ],
       [-0.4908159 , -1.18008816,  0.57312489],
       [-0.43307286, -1.18008816,  0.57312489],
       [-0.37532981, -1.18008816, -0.90590709],
       [-0.31758676, -1.18008816, -0.90590709],
       [-0.25984371, -1.18008816,  0.94288289],
       [-0.20210067, -0.96681922, -0.1663911 ],
       [-0.14435762,  0.66824269, -0.1663911 ],
       [-0.08661457,  0.66824269, -0.1663911 ],
       [-0.02887152, -0.96681922,  0.57312489],
       [ 0.02887152, -0.96681922,  0.57312489],
       [ 0.08661457, -0.96681922, -0.90590709],
       [ 0.14435762, -0.96681922, -0.90590709],
       [ 0.20210067, -0.96681922,  0.94288289],
       [ 0.25984371,  0.66824269,  0.57312489],
       [ 0.31758676,  0.66824269,  0.57312489],
       [ 0.37532981,  0.66824269, -0.90590709],
       [ 0.43307286,  0.66824269, -0.90590709],
       [ 0.4908159 , -0.11374344,  1.68239888],
       [ 0.54855895, -0.11374344,  1.68239888],
       [ 0.606302  ,  0.66824269,  0.94288289],
       [ 0.66404504,  2.51657355, -0.1663911 ],
       [ 0.72178809,  2.51657355, -1.64542308],
       [ 0.77953114,  2.51657355, -1.64542308],
       [ 0.83727419,  2.51657355,  0.94288289],
       [ 0.89501723,  0.31279445,  2.79167287],
       [ 0.95276028,  0.31279445, -1.64542308],
       [ 1.01050333,  0.31279445, -1.64542308],
       [ 1.06824638,  0.31279445,  0.94288289],
       [ 1.12598942,  0.31279445,  1.31264088],
       [ 1.18373247,  0.31279445, -1.27566508],
       [ 1.24147552,  0.31279445, -1.27566508],
       [ 1.29921857,  0.31279445,  0.57312489],
       [ 1.35696161,  0.31279445,  0.57312489],
       [ 1.41470466,  0.31279445, -0.90590709],
       [ 1.47244771,  0.31279445, -0.90590709],
       [ 1.53019075,  0.31279445,  0.94288289],
       [ 1.5879338 ,  1.37913918,  0.57312489],
       [ 1.64567685,  1.37913918,  0.57312489],
       [ 1.7034199 ,  1.37913918, -0.90590709]])
```

In [205]:
```
logr=LogisticRegression()
logr.fit(fs,d)
```

Out[205]:
```
LogisticRegression()
```

In [206]:
```
e=[[2,5,77]]
```

In [207]:
```
prediction=logr.predict(e)
prediction
```

Out[207]:
```
array([15], dtype=int64)
```

In [208]:
```
logr.classes_
```

Out[208]:
```
array([ 3,  4,  5,  7,  9, 10, 11, 12, 15], dtype=int64)
```

In [209]:
```
logr.predict_proba(e)[0][0]
```

Out[209]:
```
1.2310640195501703e-123
```

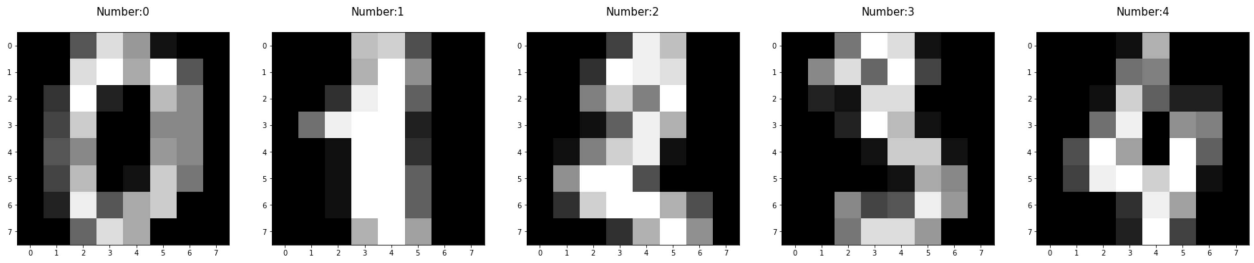
In [210]:
```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [211]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [212]:
```
digits=load_digits()
digits
```

```
        [ 0.,  8., 16., ..., 16.,  8.,  0.],
        [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digits dataset\n-
--------------------------------------------------\n\n**Data Set Characteristics:**\n\n
:Number of Instances: 1797\n    :Number of Attributes: 64\n    :Attribute Information:
8x8 image of integer pixels in the range 0..16.\n    :Missing Attribute Values: None\n
:Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)\n    :Date: July; 1998\n\nThis is a c
opy of the test set of the UCI ML hand-written digits datasets\nhttps://archive.ics.uc
i.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits\n\nThe data set contains i
mages of hand-written digits: 10 classes where\neach class refers to a digit.\n\nPrepr
ocessing programs made available by NIST were used to extract\nnormalized bitmaps of h
andwritten digits from a preprinted form. From a\ntotal of 43 people, 30 contributed t
o the training set and different 13\nto the test set. 32x32 bitmaps are divided into n
onoverlapping blocks of\n4x4 and the number of on pixels are counted in each block. Th
is generates\nan input matrix of 8x8 where each element is an integer in the range\n
0..16. This reduces dimensionality and gives invariance to small\ndistortions.\n\nFor
info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.\nT. Candela, D.
L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.\nL. Wilson, NIST Form-Based H
andprint Recognition System, NISTIR 5469,\n1994.\n\n.. topic:: References\n\n  - C. Ka
```

In [213]:
```
plt.figure(figsize=(50,25))
for index,(image,label) in enumerate(zip(digits.data[0:8],digits.target[0:5])):
    plt.subplot(1,8,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title('Number:%i\n'%label,fontsize=15)
```



In [214]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.
```

In [215]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(539, 64)
(1258, 64)
(539,)
(1258,)
```

In [216]:
```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)

```

Out[216]:
```
LogisticRegression(max_iter=10000)
```

In [217]:
```
print(logre.predict(x_test))
```

```
[8 6 7 ... 6 1 2]
```

In [218]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [219]:
```
a=pd.read_csv(r"C:\USERS\user\Downloads\C9_Data.csv")
```

In [279]:
```
a=a.head(10)
a
```

Out[279]:

	row_id	user_id	timestamp	gate_id
0	0	18	2022-07-29 09:08:54	7
1	1	18	2022-07-29 09:09:54	9
2	2	18	2022-07-29 09:09:54	9
3	3	18	2022-07-29 09:10:06	5
4	4	18	2022-07-29 09:10:08	5
5	5	18	2022-07-29 09:10:34	10
6	6	18	2022-07-29 09:32:47	11
7	7	18	2022-07-29 09:33:12	4
8	8	18	2022-07-29 09:33:13	4
9	9	1	2022-07-29 09:33:16	7

In [280]:
```
b=a[['row_id', 'user_id', 'gate_id']]
b
```

Out[280]:

	row_id	user_id	gate_id
0	0	18	7
1	1	18	9
2	2	18	9
3	3	18	5
4	4	18	5
5	5	18	10
6	6	18	11
7	7	18	4
8	8	18	4
9	9	1	7

In [281]:
```
b['gate_id'].value_counts()
```

Out[281]:
```
4     2
5     2
7     2
9     2
10    1
11    1
Name: gate_id, dtype: int64
```

In [282]:
```
x=b.drop('gate_id',axis=1)
y=b['gate_id']
print(b)
```

```
   row_id  user_id  gate_id
0       0       18        7
1       1       18        9
2       2       18        9
3       3       18        5
4       4       18        5
5       5       18       10
6       6       18       11
7       7       18        4
8       8       18        4
9       9        1        7
```

In [283]:
```
g1={"gate_id":{'g1':1,'b':2}}
a=a.replace(g1)
print(a)
```

```
   row_id  user_id            timestamp  gate_id
0       0       18  2022-07-29 09:08:54        7
1       1       18  2022-07-29 09:09:54        9
2       2       18  2022-07-29 09:09:54        9
3       3       18  2022-07-29 09:10:06        5
4       4       18  2022-07-29 09:10:08        5
5       5       18  2022-07-29 09:10:34       10
6       6       18  2022-07-29 09:32:47       11
7       7       18  2022-07-29 09:33:12        4
8       8       18  2022-07-29 09:33:13        4
9       9        1  2022-07-29 09:33:16        7
```

In [284]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

In [285]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [286]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[286]:
```
RandomForestClassifier()
```

In [287]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [288]:
```
from sklearn.model_selection import GridSearchCV
```

In [289]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:666: UserWa
rning: The least populated class in y has only 1 members, which is less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
```

Out[289]:
```
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 [290]:
```
grid_search.best_score_
```

Out[290]:
```
0.29166666666666663
```

In [291]:
```
rfc_best=grid_search.best_estimator_
```

In [292]:
```
from sklearn.tree import plot_tree
```

In [293]:
```
plt.figure(figsize=(20,10))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],fi
```

Out[293]:
```
[Text(558.0, 271.8, 'gini = 0.735\nsamples = 4\nvalue = [2, 1, 2, 2, 0]\nclass = Yes')]
```