In [2]:

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

In [3]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C9_Data.csv")
a
```

Out[3]:

	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
...	...	...	...	...
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 [9]:

```
a=a[['row_id','user_id','gate_id']]
a
```

Out[9]:

	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 [10]:

```
from sklearn.linear_model import LogisticRegression
```

In [11]:

```
a=a.head(10)
a
```

Out[11]:

	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 [12]:

```
c=a.iloc[:,0:9]
d=a.iloc[:,-1]
```

In [13]:

```
c.shape
```

Out[13]:

```
(10, 3)
```

In [14]:

```
d.shape
```

Out[14]:

```
(10,)
```

In [15]:

```
from sklearn.preprocessing import StandardScaler
```

In [16]:

```
fs=StandardScaler().fit_transform(c)
```

In [17]:

```
logr=LogisticRegression()
logr.fit(fs,d)
```

Out[17]:

```
LogisticRegression()
```

In [21]:

```
e=[[2,5,77]]
```

In [22]:

```
prediction=logr.predict(e)
prediction
```

Out[22]:

```
array([11], dtype=int64)
```

In [23]:

```
logr.classes_
```

Out[23]:

```
array([ 4,  5,  7,  9, 10, 11], dtype=int64)
```

In [24]:

```
logr.predict_proba(e)[0][0]
```

Out[24]:

```
1.1739426061051213e-52
```

In [25]:

```
logr.predict_proba(e)[0][1]
```

Out[25]:

```
1.1014934484995237e-52
```

In [26]:

```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [27]:

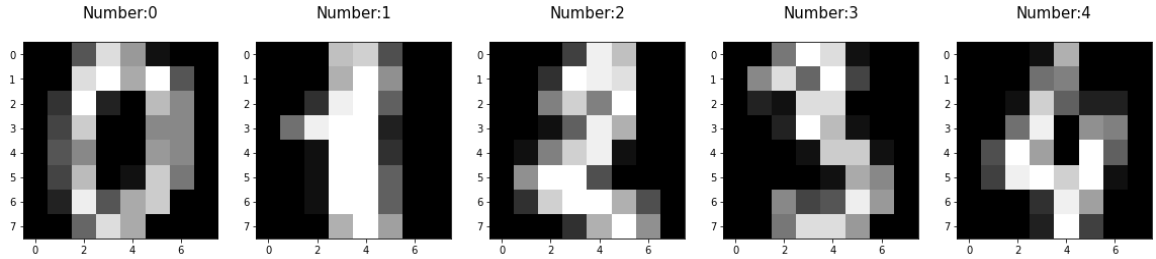
```
digits=load_digits()
digits
```

Out[27]:

```
{'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'frame': None,
 'feature_names': ['pixel_0_0',
  'pixel_0_1',
  'pixel_0_2',
  'pixel_0_3',
  'pixel_0_4',
  'pixel_0_5',
  'pixel_0_6',
  'pixel_0_7',
  'pixel_1_0',
```

In [28]:

```
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 [29]:

```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30)
```

In [30]:

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

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

In [31]:

```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[31]:

```
LogisticRegression(max_iter=10000)
```

In [32]:

```
logre.predict(x_test)
```

Out[32]:

```
array([6, 2, 4, 6, 1, 9, 5, 1, 5, 8, 3, 1, 9, 1, 3, 5, 5, 6, 4, 2, 2, 1,
       9, 2, 9, 3, 3, 3, 8, 4, 3, 6, 0, 9, 4, 4, 4, 3, 1, 7, 6, 6, 1, 8,
       0, 2, 6, 3, 3, 0, 0, 0, 1, 7, 5, 3, 5, 8, 3, 2, 5, 6, 5, 7, 4, 2,
       9, 4, 8, 7, 5, 5, 5, 0, 0, 2, 8, 8, 5, 7, 1, 6, 3, 4, 9, 0, 7, 1,
       8, 2, 5, 5, 1, 0, 6, 9, 2, 0, 9, 3, 6, 5, 7, 5, 6, 9, 7, 9, 7, 9,
       7, 9, 7, 3, 2, 5, 6, 0, 9, 6, 5, 9, 4, 4, 7, 7, 6, 1, 2, 6, 0, 1,
       1, 2, 3, 3, 4, 5, 4, 0, 9, 0, 1, 1, 2, 4, 9, 6, 0, 8, 5, 9, 2, 8,
       4, 5, 5, 1, 0, 5, 1, 7, 8, 0, 2, 2, 3, 3, 2, 6, 3, 5, 1, 2, 3, 2,
       0, 1, 6, 4, 0, 8, 4, 4, 3, 3, 7, 3, 8, 3, 3, 8, 7, 5, 3, 0, 1, 5,
       0, 8, 1, 2, 8, 5, 7, 9, 3, 2, 2, 7, 2, 8, 7, 0, 6, 2, 9, 8, 3, 1,
       4, 5, 6, 6, 8, 7, 6, 0, 1, 5, 9, 0, 3, 7, 1, 0, 9, 1, 6, 7, 1, 7,
       4, 3, 2, 7, 4, 1, 6, 6, 5, 5, 5, 3, 9, 1, 8, 2, 7, 6, 4, 7, 7, 5,
       0, 8, 4, 9, 8, 7, 6, 5, 0, 5, 3, 0, 0, 0, 6, 6, 1, 9, 2, 8, 6, 1,
       7, 0, 9, 3, 1, 9, 3, 5, 8, 0, 3, 5, 2, 0, 5, 2, 0, 4, 1, 8, 5, 2,
       6, 4, 0, 0, 1, 9, 3, 1, 8, 7, 8, 1, 1, 1, 4, 1, 2, 4, 8, 3, 7, 5,
       8, 9, 5, 7, 3, 7, 4, 1, 5, 0, 3, 2, 5, 0, 3, 5, 0, 1, 6, 7, 9, 8,
       8, 3, 9, 6, 8, 3, 2, 5, 7, 1, 6, 7, 4, 2, 7, 2, 6, 2, 3, 1, 1, 2,
       8, 1, 0, 8, 4, 7, 2, 5, 8, 9, 3, 2, 3, 8, 4, 5, 4, 8, 7, 5, 1, 9,
       2, 1, 9, 3, 1, 8, 7, 2, 2, 9, 3, 2, 2, 1, 1, 2, 3, 1, 2, 4, 2, 7,
       5, 6, 3, 0, 9, 4, 0, 0, 0, 4, 5, 3, 5, 6, 7, 0, 4, 5, 1, 1, 0, 7,
       9, 4, 1, 2, 3, 6, 7, 3, 0, 2, 2, 7, 3, 7, 7, 9, 7, 1, 2, 0, 5, 6,
       3, 2, 4, 0, 8, 4, 8, 5, 2, 9, 1, 3, 7, 2, 5, 3, 7, 1, 8, 4, 5, 1,
       8, 3, 4, 2, 7, 8, 4, 2, 5, 7, 9, 1, 0, 0, 4, 3, 5, 5, 1, 5, 6, 9,
       8, 3, 8, 3, 9, 3, 7, 4, 8, 6, 8, 4, 1, 7, 6, 0, 9, 2, 3, 2, 6, 0,
       8, 8, 9, 7, 4, 7, 8, 8, 9, 1, 2, 1])
```

In [33]:

```
logre.score(x_test,y_test)
```

Out[33]:

```
0.9611111111111111
```

In [34]:

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

In [35]:

```
a=a.head(10)
a
```

Out[35]:

	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 [37]:

```
a['gate_id'].value_counts()
```

Out[37]:

```
4     2
5     2
7     2
9     2
10    1
11    1
Name: gate_id, dtype: int64
```

In [38]:

```
x=a.drop('gate_id',axis=1)
y=a['gate_id']
```

In [39]:

```
g1={"g":{'g':1,'b':2}}
a=a.replace(g1)
print(a)
```

```
   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 [40]:

```
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 [41]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [42]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[42]:

```
RandomForestClassifier()
```

In [43]:

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

In [44]:

```
from sklearn.model_selection import GridSearchCV
```

In [45]:

```
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: UserWarning: The least populated class in y has only 1 members, wh
ich is less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
```

Out[45]:

```
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 [46]:

```
grid_search.best_score_
```

Out[46]:

```
0.29166666666666663
```

In [47]:

```
rfc_best=grid_search.best_estimator_
```

In [48]:

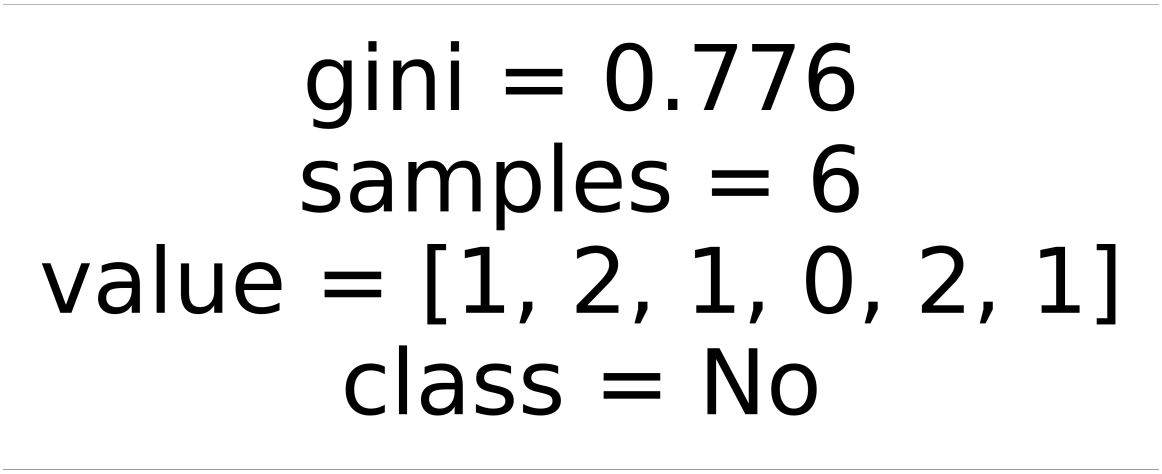
```
from sklearn.tree import plot_tree
```

In [49]:

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

Out[49]:

```
[Text(2232.0, 1087.2, 'gini = 0.776\nsamples = 6\nvalue = [1, 2, 1, 0, 2,
1]\nclass = No')]
```



In [ ]:

```
```