mk 02-09-2023

In [342]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [343]:
```
from sklearn.linear_model import LogisticRegression
a=pd.read_csv(r"C:\USERS\user\Downloads\C10_loan1.csv")
a
```

Out[343]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	Yes	Single	125	No
1	No	Married	100	No
2	No	Single	70	No
3	Yes	Married	120	No
4	No	Divorced	95	Yes
5	No	Married	60	No
6	Yes	Divorced	220	No
7	No	Single	85	Yes
8	No	Married	75	No
9	No	Single	90	Yes

In [344]:
```
a=a.head(60)
a
```

Out[344]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	Yes	Single	125	No
1	No	Married	100	No
2	No	Single	70	No
3	Yes	Married	120	No
4	No	Divorced	95	Yes
5	No	Married	60	No
6	Yes	Divorced	220	No
7	No	Single	85	Yes
8	No	Married	75	No
9	No	Single	90	Yes

In [345]:
```
from sklearn.linear_model import LogisticRegression
```

In [346]:
```
a.columns
```

Out[346]:
```
Index(['Home Owner', 'Marital Status', 'Annual Income', 'Defaulted Borrower'], dtype='obje
ct')
```

In [347]:
```
b=a[['Annual Income']]
b
```

Out[347]:

	Annual Income
0	125
1	100
2	70
3	120
4	95
5	60
6	220
7	85
8	75
9	90

In [348]:
```
c=b.iloc[:,0:3]
d=b.iloc[:,-1]
```

In [349]:
```
c.shape
```

Out[349]:
```
(10, 1)
```

In [350]:
```
d.shape
```

Out[350]:
```
(10,)
```

In [351]:
```
from sklearn.preprocessing import StandardScaler
fs=StandardScaler().fit_transform(c)
fs
```

Out[351]:
```
array([[ 0.4851036 ],
       [-0.09240069],
       [-0.78540584],
       [ 0.36960275],
       [-0.20790154],
       [-1.01640755],
       [ 2.67961991],
       [-0.43890326],
       [-0.66990498],
       [-0.3234024 ]])
```

In [352]:
```
logr=LogisticRegression()
logr.fit(fs,d)
```

Out[352]:
```
LogisticRegression()
```

In [353]:
```
e=[[777]]
```

In [354]:
```
prediction=logr.predict(e)
prediction
```

Out[354]:
```
array([220], dtype=int64)
```

In [355]:
```
logr.classes_
```
Out[355]:
```
array([ 60,  70,  75,  85,  90,  95, 100, 120, 125, 220], dtype=int64)
```

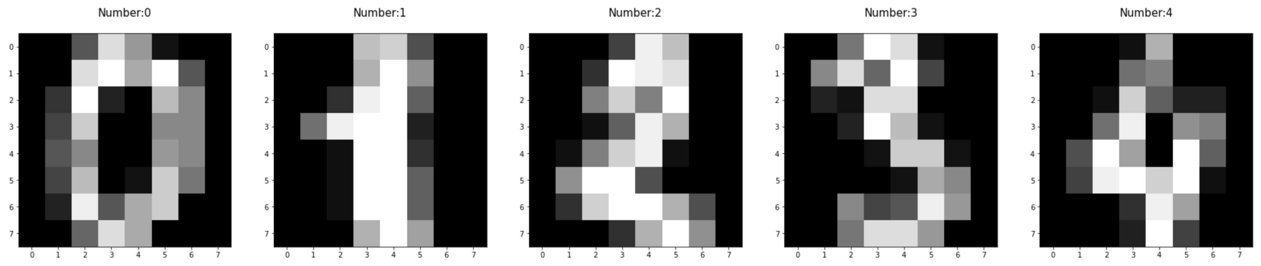
In [356]:
```
logr.predict_proba(e)[0][0]
```
Out[356]:
```
0.0
```

In [357]:
```
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 [358]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [359]:
```
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 co
py of the test set of the UCI ML hand-written digits datasets\nhttps://archive.ics.uci.
edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits\n\nThe data set contains imag
es of hand-written digits: 10 classes where\neach class refers to a digit.\n\nPreproces
sing programs made available by NIST were used to extract\nnormalized bitmaps of handwr
itten digits from a preprinted form. From a\ntotal of 43 people, 30 contributed to the
training set and different 13\nto the test set. 32x32 bitmaps are divided into nonoverl
apping blocks of\n4x4 and the number of on pixels are counted in each block. This gener
ates\nan input matrix of 8x8 where each element is an integer in the range\n0..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 Handprint Recogni
tion System, NISTIR 5469,\n1994.\n\n.. topic:: References\n\n  - C. Kaynak (1995) Metho
```

In [360]:
```
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 [361]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.7
```

In [362]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(377, 64)
(1420, 64)
(377,)
(1420,)
```

In [363]:
```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)

```

Out[363]:
```
LogisticRegression(max_iter=10000)
```

In [364]:
```
print(logre.predict(x_test))
```

```
[2 8 5 ... 7 0 3]
```

In [365]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [366]:
```
a=pd.read_csv(r"C:\USERS\user\Downloads\C10_loan1.csv")
```

In [367]:
```
a=a.head(60)
a
```

Out[367]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	Yes	Single	125	No
1	No	Married	100	No
2	No	Single	70	No
3	Yes	Married	120	No
4	No	Divorced	95	Yes
5	No	Married	60	No
6	Yes	Divorced	220	No
7	No	Single	85	Yes
8	No	Married	75	No
9	No	Single	90	Yes

In [368]:
```
b=a[['Home Owner', 'Annual Income']]
b
```

Out[368]:

	Home Owner	Annual Income
0	Yes	125
1	No	100
2	No	70
3	Yes	120
4	No	95
5	No	60
6	Yes	220
7	No	85
8	No	75
9	No	90

In [369]:
```
b['Home Owner'].value_counts()
```

Out[369]:
```
No     7
Yes    3
Name: Home Owner, dtype: int64
```

In [370]:
```
x=b.drop('Home Owner',axis=1)
y=b['Home Owner']
print(b)
```

```
  Home Owner  Annual Income
0        Yes            125
1         No            100
2         No             70
3        Yes            120
4         No             95
5         No             60
6        Yes            220
7         No             85
8         No             75
9         No             90
```

In [371]:
```
g1={"Home Owner":{'g1':1}}
a=a.replace(g1)
print(a)
```

```
  Home Owner Marital Status  Annual Income Defaulted Borrower
0        Yes         Single            125                 No
1         No        Married            100                 No
2         No         Single             70                 No
3        Yes        Married            120                 No
4         No       Divorced             95                Yes
5         No        Married             60                 No
6        Yes       Divorced            220                 No
7         No         Single             85                Yes
8         No        Married             75                 No
9         No         Single             90                Yes
```

In [372]:
```
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 [373]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [374]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[374]:
```
RandomForestClassifier()
```

In [375]:
```
parameters={'max_depth':[1,2,3,4,5],
           'min_samples_leaf':[5,10,15,20,25],
           'n_estimators':[10,20,30,40,50]}
```

In [376]:
```
from sklearn.model_selection import GridSearchCV
```

In [377]:
```
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: UserWarn
ing: 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[377]:
```
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 [378]:
```
grid_search.best_score_
```

Out[378]:
```
0.875
```

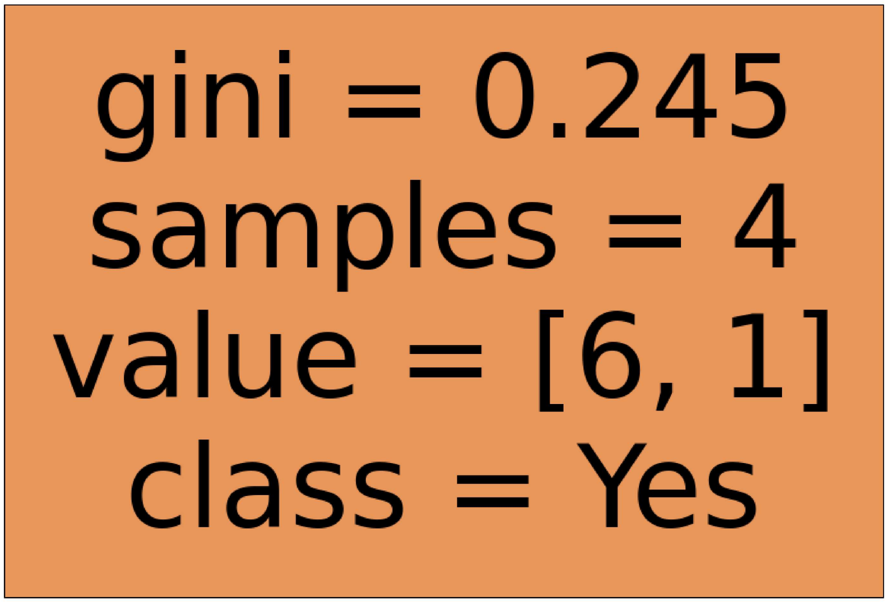
In [379]:
```
rfc_best=grid_search.best_estimator_
```

In [380]:
```
from sklearn.tree import plot_tree
```

In [381]:
```
plt.figure(figsize=(20,10))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],fil

```

Out[381]:
```
[Text(558.0, 271.8, 'gini = 0.245\nsamples = 4\nvalue = [6, 1]\nclass = Yes')]
```


gini = 0.245
samples = 4
value = [6, 1]
class = Yes