mk (02-09-2023)

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

In [153]:
```
1 from sklearn.linear_model import LogisticRegression
2 a=pd.read_csv(r"C:\USERS\user\Downloads\C4_framingham.csv")
3 a
```

Out[153]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentH
0	1	39	4.0	0	0.0	0.0	0	
1	0	46	2.0	0	0.0	0.0	0	
2	1	48	1.0	1	20.0	0.0	0	
3	0	61	3.0	1	30.0	0.0	0	
4	0	46	3.0	1	23.0	0.0	0	
...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	
4234	1	51	3.0	1	43.0	0.0	0	
4235	0	48	2.0	1	20.0	NaN	0	
4236	0	44	1.0	1	15.0	0.0	0	
4237	0	52	2.0	0	0.0	0.0	0	

In [154]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [200]:
```
a=a.head(100)
a
```

Out[200]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	di
0	1	39	4.0	0	0.0	0.0	0	0	
1	0	46	2.0	0	0.0	0.0	0	0	
2	1	48	1.0	1	20.0	0.0	0	0	
3	0	61	3.0	1	30.0	0.0	0	1	
4	0	46	3.0	1	23.0	0.0	0	0	
5	0	43	2.0	0	0.0	0.0	0	1	
6	0	63	1.0	0	0.0	0.0	0	0	
7	0	45	2.0	1	20.0	0.0	0	0	
8	1	52	1.0	0	0.0	0.0	0	1	
9	1	43	1.0	1	30.0	0.0	0	1	

In [201]:
```
a.columns
```

Out[201]:
```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

In [202]:
```
b=a[['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD']]
b
```

Out[202]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	di
0	1	39	4.0	0	0.0	0.0	0	0	
1	0	46	2.0	0	0.0	0.0	0	0	
2	1	48	1.0	1	20.0	0.0	0	0	
3	0	61	3.0	1	30.0	0.0	0	1	
4	0	46	3.0	1	23.0	0.0	0	0	
5	0	43	2.0	0	0.0	0.0	0	1	
6	0	63	1.0	0	0.0	0.0	0	0	
7	0	45	2.0	1	20.0	0.0	0	0	
8	1	52	1.0	0	0.0	0.0	0	1	
9	1	43	1.0	1	30.0	0.0	0	1	

In [203]:
```
c=b.iloc[:,0:15]
d=b.iloc[:,-1]
```

In [204]:
```
c.shape
```

Out[204]: (10, 15)

In [205]:
```
d.shape
```

Out[205]: (10,)

In [ ]:
```

```

In [206]:
```
from sklearn.preprocessing import StandardScaler
fs=StandardScaler().fit_transform(c)
fs
```

Out[206]:
```
array([[ 1.22474487, -1.28931674,  2.        , -1.        , -0.96754461,
         0.        ,  0.        , -0.81649658,  0.        , -1.41062997,
        -1.27734618, -1.15306967,  0.05752806,  0.14484136, -0.70928138],
       [-0.81649658, -0.34918995,  0.        , -1.        , -0.96754461,
         0.        ,  0.        , -0.81649658,  0.        ,  0.20235648,
        -0.63004237, -0.35029965,  0.58486866,  1.59325501, -0.8106073 ],
       [ 1.22474487, -0.0805823 , -1.        ,  1.        ,  0.60569866,
         0.        ,  0.        , -0.81649658,  0.        ,  0.05572135,
        -0.34954405, -0.42327874, -0.43086123, -0.33796318, -1.41856277],
       [-0.81649658,  1.66536746,  1.        ,  1.        ,  1.39232029,
         0.        ,  0.        ,  1.22474487,  0.        , -0.53081918,
         0.62141165,  0.67140766,  0.53992486, -1.30357228,  1.92519233],
       [-0.81649658, -0.34918995,  1.        ,  1.        ,  0.84168515,
         0.        ,  0.        , -0.81649658,  0.        ,  1.22880241,
        -0.24166009, -0.13136237, -1.10202199,  0.62764591,  0.10132591],
       [-0.81649658, -0.75210143,  0.        , -1.        , -0.96754461,
         0.        ,  0.        ,  1.22474487,  0.        , -0.4428381 ,
         1.91601926,  1.76609405,  1.05528044, -0.14484136,  1.51988868],
       [-0.81649658,  1.93397511, -1.        , -1.        , -0.96754461,
         0.        ,  0.        , -0.81649658,  0.        , -1.11735971,
         0.10356861, -1.08009058,  1.89722763, -1.78637683,  0.10132591],
       [-0.81649658, -0.48349378,  0.        ,  1.        ,  0.60569866,
         0.        ,  0.        , -0.81649658,  0.        ,  2.04995915,
        -1.5362677 , -1.08009058, -1.52748997,  0.04828045, -0.60795547],
       [ 1.22474487,  0.45663301, -1.        , -1.        , -0.96754461,
         0.        ,  0.        ,  1.22474487,  0.        ,  0.49562675,
         0.25460616,  0.2335331 , -0.12524339, -0.24140227, -0.50662956],
       [ 1.22474487, -0.75210143, -1.        ,  1.        ,  1.39232029,
         0.        ,  0.        ,  1.22474487,  0.        , -0.53081918,
         1.1392547 ,  1.54715677, -0.94921307,  1.40013319,  0.40530365]])
```

In [207]:
```
logr=LogisticRegression()
logr.fit(fs,d)
```

Out[207]: LogisticRegression()

In [208]:
```
e=[[2,5,77,8,5,2.3,5.2,1,1.2,16,56,52,45,25,65]]
```

In [209]:
```
prediction=logr.predict(e)
prediction
```

Out[209]:
```
array([1], dtype=int64)
```

In [210]:
```
logr.classes_
```

Out[210]:
```
array([0, 1], dtype=int64)
```

In [211]:
```
logr.predict_proba(e)[0][0]
```

Out[211]:
```
1.9817379515174594e-08
```

In [212]:
```
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 [213]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [214]:
```
digits=load_digits()
digits
```

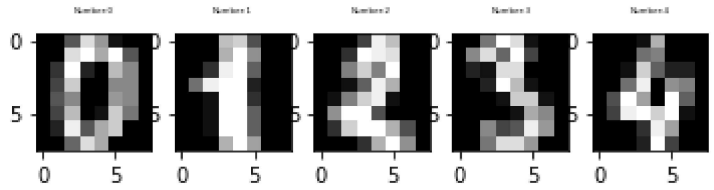
```
        [ 0.,  8., 16., ..., 16.,  8.,  0.],
        [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digits
dataset\n--------------------------------------------------\n\n**Data Set C
haracteristics:**\n\n    :Number of Instances: 1797\n    :Number of Attribu
tes: 64\n    :Attribute Information: 8x8 image of integer pixels in the ran
ge 0..16.\n    :Missing Attribute Values: None\n    :Creator: E. Alpaydin
(alpaydin '@' boun.edu.tr)\n    :Date: July; 1998\n\nThis is a copy 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 images of hand-written digits: 10 classes where\neach class refers
to a digit.\n\nPreprocessing programs made available by NIST were used to e
xtract\nnormalized bitmaps of handwritten digits from a preprinted form. Fr
om a\ntotal of 43 people, 30 contributed to the training set and different
13\nto the test set. 32x32 bitmaps are divided into nonoverlapping blocks o
f\n4x4 and the number of on pixels are counted in each block. This generate
s\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\ndistortio
ns.\n\nFor info on NIST preprocessing routines, see M. D. Garris, J. L. Blu
```

In [215]:
```
plt.figure(figsize=(20,4))
```

Out[215]:
```
<Figure size 1440x288 with 0 Axes>
```

```
<Figure size 1440x288 with 0 Axes>
```

In [216]:
```
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8
                                 ,8)),cmap=plt.cm.gray)
    plt.title('Number:%i\n'%label,fontsize=4)
```



In [217]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,t
```

In [218]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [219]:
```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)

```

Out[219]:
```
LogisticRegression(max_iter=10000)
```

In [220]:
```
print(logre.predict(x_test))
```

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

In [221]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

In [270]:
```
1 a=pd.read_csv(r"C:\USERS\user\Downloads\C4_framingham.csv")
```

In [271]:
```
1 a['male'].value_counts()
```

Out[271]:
```
0    2419
1    1819
Name: male, dtype: int64
```

In [272]:
```
1 x=b.drop('male',axis=1)
2 y=b['male']
3 print(b)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-272-1b5f8ccb0541> in <module>
----> 1 x=b.drop('male',axis=1)
      2 y=b['male']
      3 print(b)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self,
labels, axis, index, columns, level, inplace, errors)
   4306                 weight  1.0     0.8
   4307         """
-> 4308         return super().drop(
   4309             labels=labels,
   4310             axis=axis,

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(sel
f, labels, axis, index, columns, level, inplace, errors)
   4151         for axis, labels in axes.items():
   4152             if labels is not None:
-> 4153                 obj = obj._drop_axis(labels, axis, level=level, error
s=errors)
   4154
   4155         if inplace:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in _drop_ax
is(self, labels, axis, level, errors)
   4186                 new_axis = axis.drop(labels, level=level, errors=erro
rs)
   4187             else:
-> 4188                 new_axis = axis.drop(labels, errors=errors)
   4189             result = self.reindex(**{axis_name: new_axis})
   4190

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in dro
p(self, labels, errors)
   5589         if mask.any():
   5590             if errors != "ignore":
-> 5591                 raise KeyError(f"{labels[mask]} not found in axis")
   5592             indexer = indexer[~mask]
   5593         return self.delete(indexer)

KeyError: "['male'] not found in axis"
```

In [273]:

```
g1={"Mention Count":{'Mention Count':1,'b':2}}
df=df.replace(g1)
print(df)
```

```
       User ID         Username  \
0       132131            flong
1       289683   hinesstephanie
2       779715       roberttran
3       696168            pmason
4       704441            noah87
...        ...               ...
49995   491196            uberg
49996   739297     jessicamunoz
49997   674475   lynncunningham
49998   167081  richardthompson
49999   311204         daniel29

                                                   Tweet  Retweet Count  \
0      Station activity person against natural majori...             85
1      Authority research natural life material staff...             55
2      Manage whose quickly especially foot none to g...              6
3      Just cover eight opportunity strong policy which.             54
4                          Animal sign six data good or.             26
...                                                  ...            ...
49995  Want but put card direction know miss former h...             64
49996  Provide whole maybe agree church respond most ...             18
49997  Bring different everyone international capital...             43
49998  Than about single generation itself seek sell ...             45
49999  Here morning class various room human true bec...             91

       Mention Count  Follower Count  Verified  Bot Label            Location
\
0                  1            2353     False          1           Adkinston
1                  5            9617      True          0          Sanderston
2                  2            4363      True          0        Harrisonfurt
3                  5            2242      True          1        Martinezberg
4                  3            8438     False          1        Camachoville
...              ...             ...       ...        ...                 ...
49995              0            9911      True          1  Lake Kimberlyburgh
49996              5            9900     False          1           Greenbury
49997              3            6313      True          1         Deborahfort
49998              1            6343     False          0         Stephenside
49999              4            4006     False          0           Novakberg

               Created At                         Hashtags
0      2020-05-11 15:29:50                              NaN
1      2022-11-26 05:18:10                       both live
2      2022-08-08 03:16:54                     phone ahead
3      2021-08-14 22:27:05                ever quickly new I
4      2020-04-13 21:24:21                   foreign mention
...                    ...                              ...
49995  2023-04-20 11:06:26  teach quality ten education any
49996  2022-10-18 03:57:35            add walk among believe
49997  2020-07-08 03:54:08          onto admit artist first
49998  2022-03-22 12:13:44                             star
49999  2022-12-03 06:11:07                             home

[50000 rows x 11 columns]
```

In [274]:
```
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 [275]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [276]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[276]:
```
RandomForestClassifier()
```

In [277]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [278]:
```
from sklearn.model_selection import GridSearchCV
```

In [279]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring=
grid_search.fit(x_train,y_train)
```

Out[279]:
```
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 [280]:
```
grid_search.best_score_
```

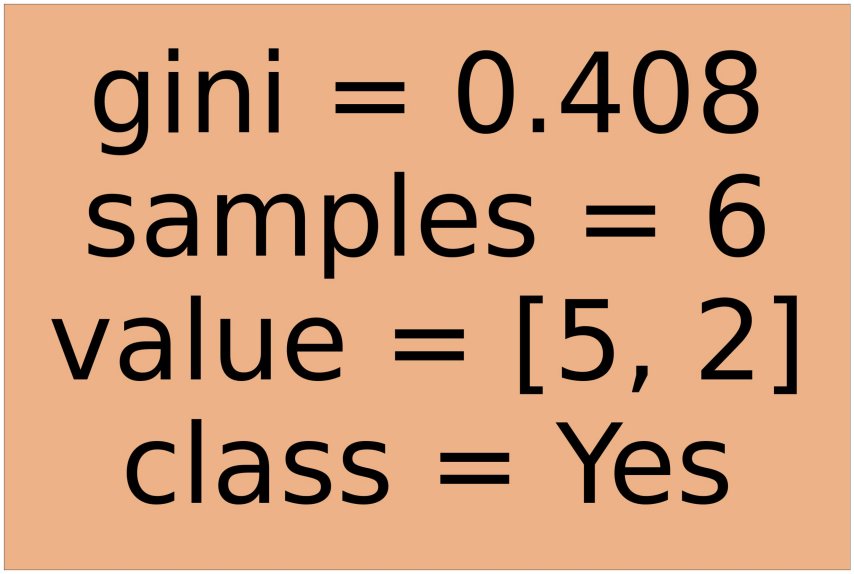
Out[280]:
```
0.5833333333333333
```

In [281]:
```
rfc_best=grid_search.best_estimator_
```

In [282]:
```
from sklearn.tree import plot_tree
```

In [283]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Ye
```

Out[283]:
```
[Text(2232.0, 1087.2, 'gini = 0.408\nsamples = 6\nvalue = [5, 2]\nclass = Ye
s')]
```



In [ ]:
```

```