mk 02-09-2023

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

In [238]:
```
from sklearn.linear_model import LogisticRegression
a=pd.read_csv(r"C:\USERS\user\Downloads\C5_health care diabetes.csv")
a
```

Out[238]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0
764	2	122	70	27	0	36.8	0
765	5	121	72	23	112	26.2	0
766	1	126	60	0	0	30.1	0
767	1	93	70	31	0	30.4	0

In [303]:
```
a=a.head(10)
a
```

Out[303]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
5	5	116	74	0	0	25.6	0.201
6	3	78	50	32	88	31.0	0.248
7	10	115	0	0	0	35.3	0.134
8	2	197	70	45	543	30.5	0.158
9	8	125	96	0	0	0.0	0.232

In [304]:
```
from sklearn.linear_model import LogisticRegression
```

In [305]:
```
a.columns
```

Out[305]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [306]:
```
b=a[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin'
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']]
b
```

Out[306]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
5	5	116	74	0	0	25.6	0.201
6	3	78	50	32	88	31.0	0.248
7	10	115	0	0	0	35.3	0.134
8	2	197	70	45	543	30.5	0.158
9	8	125	96	0	0	0.0	0.232

In [307]:
```
c=b.iloc[:,0:15]
d=b.iloc[:,-1]
```

In [308]:
```
c.shape
```

Out[308]:
```
(10, 9)
```

In [309]:
```
d.shape
```

Out[309]:
```
(10,)
```

In [ ]:
```

```

In [310]:
```
from sklearn.preprocessing import StandardScaler
fs=StandardScaler().fit_transform(c)
fs
```

Out[310]:
```
array([[ 0.48154341,  0.54447108,  0.50126452,  0.88534503, -0.55391299,
         0.55143407,  0.19208855,  1.25171488,  0.81649658],
       [-1.02327975, -1.11261482,  0.25474099,  0.5335523 , -0.55391299,
        -0.10392051, -0.25268024, -0.43499311, -1.22474487],
       [ 1.08347268,  1.46507436,  0.17256648, -1.16677921, -0.55391299,
        -0.41287339,  0.2646052 , -0.34621901,  0.81649658],
       [-1.02327975, -1.00740302,  0.25474099,  0.18175957,  0.02915332,
         0.03651261, -0.54919276, -1.32273416, -1.22474487],
       [-1.32424438,  0.25513862, -0.81352767,  0.88534503,  0.48816296,
         1.44084387,  2.86875866, -0.2574449 ,  0.81649658],
       [ 0.18057878, -0.29722334,  0.58343903, -1.16677921, -0.55391299,
        -0.1975426 , -0.4944024 , -0.52376722, -1.22474487],
       [-0.42135049, -1.29673547, -0.40265511,  0.70944866, -0.00806368,
         0.30801666, -0.41866279, -0.87886364,  0.81649658],
       [ 1.68540194, -0.32352629, -2.45701791, -1.16677921, -0.55391299,
         0.71059162, -0.60237164, -0.61254132, -1.22474487],
       [-0.72231512,  1.83331567,  0.41909001,  1.47166623,  2.81422536,
         0.26120561, -0.56369609,  1.51803719,  0.81649658],
       [ 1.08347268, -0.06049679,  1.48735867, -1.16677921, -0.55391299,
        -2.59426794, -0.44444649,  1.6068113 ,  0.81649658]])
```

In [311]:
```
logr=LogisticRegression()
logr.fit(fs,d)
```

Out[311]:
```
LogisticRegression()
```

In [312]:
```
e=[[2,5,77,8,56,52,45,25,65]]
```

In [313]:
```
prediction=logr.predict(e)
prediction
```

Out[313]:
```
array([1], dtype=int64)
```

In [314]:
```
logr.classes_
```

Out[314]:
```
array([0, 1], dtype=int64)
```

In [315]:
```
logr.predict_proba(e)[0][0]
```

Out[315]:
```
0.0
```

In [316]:
```
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 [317]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [318]:
```
digits=load_digits()
digits
```

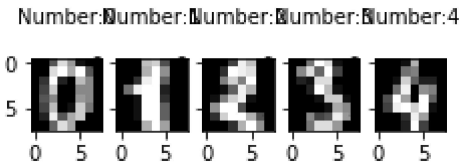
```
        [ 0.,  8., 16., ..., 16.,  8.,  0.],
        [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digits
dataset\n--------------------------------------------------\n\n**Data Set C
haracteristics:**\n\n    :Number of Instances: 1797\n    :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 [319]:
```
plt.figure(figsize=(20,4))
```

Out[319]:
```
<Figure size 1440x288 with 0 Axes>
```

<Figure size 1440x288 with 0 Axes>

In [320]:
```
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:
    plt.subplot(1,8,index+1)
    plt.imshow(np.reshape(image,(8
                                 ,8)),cmap=plt.cm.gray)
    plt.title('Number:%i\n'%label,fontsize=10)
```



In [321]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,t
```

In [322]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [323]:
```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)

```

Out[323]:
```
LogisticRegression(max_iter=10000)
```

In [324]:
```
print(logre.predict(x_test))
```

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

In [325]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [326]:
```
a=pd.read_csv(r"C:\USERS\user\Downloads\C5_health care diabetes.csv")
```

In [329]:
```
a=a.head(10)
a
```

Out[329]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
5	5	116	74	0	0	25.6	0.201
6	3	78	50	32	88	31.0	0.248
7	10	115	0	0	0	35.3	0.134
8	2	197	70	45	543	30.5	0.158
9	8	125	96	0	0	0.0	0.232

In [343]:
```
a['Outcome'].value_counts()
```

Out[343]:
```
1    6
0    4
Name: Outcome, dtype: int64
```

In [358]:
```
x=b.drop('Outcome',axis=1)
y=b['Outcome']
print(b)
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1
5            5      116             74              0        0  25.6
6            3       78             50             32       88  31.0
7           10      115              0              0        0  35.3
8            2      197             70             45      543  30.5
9            8      125             96              0        0   0.0

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
5                     0.201   30        0
6                     0.248   26        1
7                     0.134   29        0
8                     0.158   53        1
9                     0.232   54        1
```

In [359]:
```
g1={"Outcome":{'Outcome':1,'b':2}}
a=a.replace(g1)
print(a)
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1
5            5      116             74              0        0  25.6
6            3       78             50             32       88  31.0
7           10      115              0              0        0  35.3
8            2      197             70             45      543  30.5
9            8      125             96              0        0   0.0

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
5                     0.201   30        0
6                     0.248   26        1
7                     0.134   29        0
8                     0.158   53        1
9                     0.232   54        1
```

In [360]:
```
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 [361]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [362]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[362]:
```
RandomForestClassifier()
```

In [363]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [364]:
```
from sklearn.model_selection import GridSearchCV
```

In [365]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring=
grid_search.fit(x_train,y_train)
```

Out[365]:
```
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 [366]:
```
1 grid_search.best_score_
```

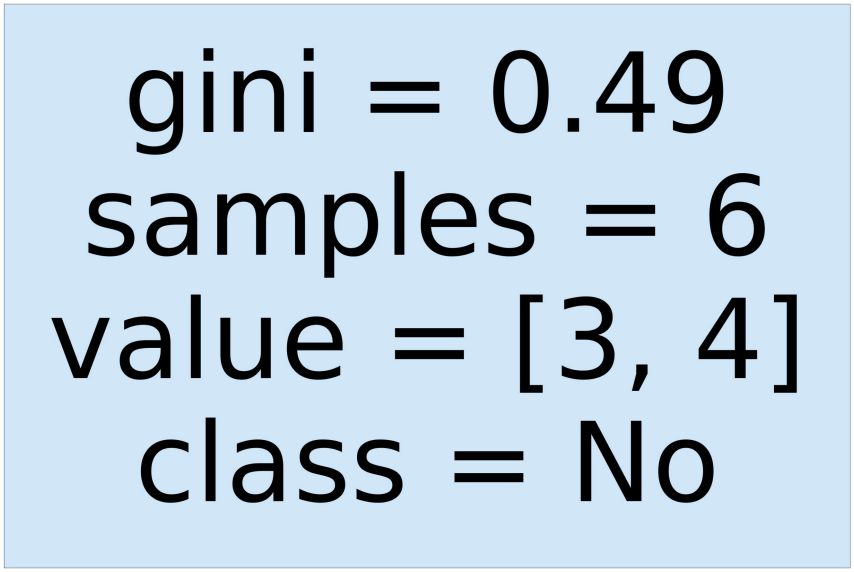
Out[366]:
```
0.5833333333333333
```

In [367]:
```
1 rfc_best=grid_search.best_estimator_
```

In [368]:
```
1 from sklearn.tree import plot_tree
```

In [369]:
```
1 plt.figure(figsize=(80,40))
2 plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Ye
3
```

Out[369]:
```
[Text(2232.0, 1087.2, 'gini = 0.49\nsamples = 6\nvalue = [3, 4]\nclass = N
o')]
```



In [ ]:
```
1
```