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

In [2]:
```
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
df=pd.read_csv("C6_bmi.csv")
df
```

Out[3]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3
...	...	...	...	...
495	Female	150	153	5
496	Female	184	121	4
497	Female	141	136	5
498	Male	150	95	5
499	Male	173	131	5

500 rows × 4 columns

In [4]:
```
df=df.dropna()
df
```

Out[4]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3
...	...	...	...	...
495	Female	150	153	5
496	Female	184	121	4
497	Female	141	136	5

	Gender	Height	Weight	Index
498	Male	150	95	5
499	Male	173	131	5

500 rows × 4 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Gender  500 non-null    object
 1   Height  500 non-null    int64 
 2   Weight  500 non-null    int64 
 3   Index   500 non-null    int64 
dtypes: int64(3), object(1)
memory usage: 19.5+ KB
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Gender', 'Height', 'Weight', 'Index'], dtype='object')
```

In [7]:
```
feature_matrix=df[['Height', 'Weight']]
target_vector=df[ 'Index']
```

In [8]:
```
feature_matrix.shape
```

Out[8]:
```
(500, 2)
```

In [9]:
```
target_vector.shape
```

Out[9]:
```
(500,)
```

In [10]:
```
from sklearn.preprocessing import StandardScaler
```

In [11]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [12]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

Out[12]:
```
LogisticRegression()
```

In [13]:
```
observation=[[1,2]]
```

In [14]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[5]
```

In [15]:
```
logr.classes_
```

Out[15]:
```
array([0, 1, 2, 3, 4, 5], dtype=int64)
```

In [16]:
```
logr.predict_proba(observation)[0][0]
```

Out[16]:
```
5.5956697582538237e-11
```

In [17]:
```
logr.predict_proba(observation)
```

Out[17]:
```
array([[5.59566976e-11, 6.05990036e-10, 1.19071465e-07, 4.99471797e-05,
        2.03791363e-02, 9.79570797e-01]])
```

In [18]:
```
df['Index'].value_counts()
```

Out[18]:
```
5    198
4    130
2     69
3     68
1     22
0     13
Name: Index, dtype: int64
```

In [19]:
```
x=df[['Height', 'Weight']]
y=df['Index']
```

In [20]:
```
#g1={ 'TenYearCHD':{'True':1,'False':2}}
#df=df.replace(g1)
#df
```

In [21]:
```
from sklearn.model_selection import train_test_split
```

In [22]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

In [23]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [24]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[24]:
```
RandomForestClassifier()
```

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

In [26]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[26]:
```
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 [27]:
```
grid_search.best_score_
```

Out[27]:
```
0.7714285714285715
```

In [28]:
```
rfc_best=grid_search.best_estimator_
```

In [35]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d',
```

Out[35]:
```
[Text(2547.391304347826, 1993.2, 'Weight <= 129.0\ngini = 0.723\nsamples = 222\nvalue =
[4, 16, 41, 45, 111, 133]\nclass = f'),
 Text(1407.1304347826087, 1630.8000000000002, 'Weight <= 65.5\ngini = 0.75\nsamples = 14
9\nvalue = [4, 16, 41, 45, 91, 36]\nclass = e'),
 Text(582.2608695652174, 1268.4, 'Weight <= 61.5\ngini = 0.635\nsamples = 31\nvalue =
[4, 16, 22, 4, 0, 0]\nclass = c'),
 Text(388.17391304347825, 906.0, 'Height <= 171.5\ngini = 0.624\nsamples = 23\nvalue =
[4, 14, 14, 1, 0, 0]\nclass = b'),
 Text(194.08695652173913, 543.5999999999999, 'gini = 0.436\nsamples = 10\nvalue = [2, 1,
11, 1, 0, 0]\nclass = c'),
 Text(582.2608695652174, 543.5999999999999, 'Weight <= 58.5\ngini = 0.438\nsamples = 13
\nvalue = [2, 13, 3, 0, 0, 0]\nclass = b'),
 Text(388.17391304347825, 181.19999999999982, 'gini = 0.298\nsamples = 8\nvalue = [2, 9,
0, 0, 0, 0]\nclass = b'),
 Text(776.3478260869565, 181.19999999999982, 'gini = 0.49\nsamples = 5\nvalue = [0, 4,
3, 0, 0, 0]\nclass = b'),
 Text(776.3478260869565, 906.0, 'gini = 0.544\nsamples = 8\nvalue = [0, 2, 8, 3, 0, 0]\n
class = c'),
 Text(2232.0, 1268.4, 'Weight <= 105.5\ngini = 0.668\nsamples = 118\nvalue = [0, 0, 19,
41, 91, 36]\nclass = e'),
 Text(1746.782608695652, 906.0, 'Weight <= 84.5\ngini = 0.709\nsamples = 75\nvalue = [0,
0, 19, 40, 40, 15]\nclass = d'),
 Text(1358.608695652174, 543.5999999999999, 'Height <= 179.5\ngini = 0.656\nsamples = 36
\nvalue = [0, 0, 16, 12, 23, 1]\nclass = e'),
 Text(1164.5217391304348, 181.19999999999982, 'gini = 0.53\nsamples = 26\nvalue = [0, 0,
2, 12, 23, 1]\nclass = e'),
 Text(1552.695652173913, 181.19999999999982, 'gini = 0.0\nsamples = 10\nvalue = [0, 0, 1
4, 0, 0, 0]\nclass = c'),
 Text(2134.9565217391305, 543.5999999999999, 'Weight <= 103.5\ngini = 0.668\nsamples = 3
9\nvalue = [0, 0, 3, 28, 17, 14]\nclass = d'),
```

```
 Text(1940.8695652173913, 181.19999999999982, 'gini = 0.691\nsamples = 33\nvalue = [0,
0, 3, 20, 15, 13]\nclass = d'),
 Text(2329.0434782608695, 181.19999999999982, 'gini = 0.43\nsamples = 6\nvalue = [0, 0,
0, 8, 2, 1]\nclass = d'),
 Text(2717.217391304348, 906.0, 'Height <= 165.0\ngini = 0.429\nsamples = 43\nvalue =
[0, 0, 0, 1, 51, 21]\nclass = e'),
 Text(2523.1304347826085, 543.5999999999999, 'gini = 0.0\nsamples = 15\nvalue = [0, 0,
0, 0, 0, 21]\nclass = f'),
 Text(2911.304347826087, 543.5999999999999, 'Weight <= 115.5\ngini = 0.038\nsamples = 28
\nvalue = [0, 0, 0, 1, 51, 0]\nclass = e'),
 Text(2717.217391304348, 181.19999999999982, 'gini = 0.1\nsamples = 11\nvalue = [0, 0,
0, 1, 18, 0]\nclass = e'),
 Text(3105.391304347826, 181.19999999999982, 'gini = 0.0\nsamples = 17\nvalue = [0, 0,
0, 0, 33, 0]\nclass = e'),
 Text(3687.6521739130435, 1630.8000000000002, 'Weight <= 138.5\ngini = 0.283\nsamples =
73\nvalue = [0, 0, 0, 0, 20, 97]\nclass = f'),
 Text(3299.478260869565, 1268.4, 'Height <= 181.0\ngini = 0.436\nsamples = 19\nvalue =
[0, 0, 0, 0, 9, 19]\nclass = f'),
 Text(3105.391304347826, 906.0, 'gini = 0.0\nsamples = 13\nvalue = [0, 0, 0, 0, 0, 19]\n
class = f'),
 Text(3493.565217391304, 906.0, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 0, 0, 9, 0]\ncl
ass = e'),
 Text(4075.8260869565215, 1268.4, 'Height <= 191.5\ngini = 0.217\nsamples = 54\nvalue =
[0, 0, 0, 0, 11, 78]\nclass = f'),
 Text(3881.7391304347825, 906.0, 'gini = 0.0\nsamples = 47\nvalue = [0, 0, 0, 0, 0, 76]
\nclass = f'),
 Text(4269.913043478261, 906.0, 'gini = 0.26\nsamples = 7\nvalue = [0, 0, 0, 0, 11, 2]\n
class = e')]
```