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

In [2]:

In [3]:
```
df = pd.read_csv("C4_framingham.csv")
```

Out[3]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp
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
...	...	...	...	...	...	...	...	...
4233	1	50	1.0	1	1.0	0.0	0	1
4234	1	51	3.0	1	43.0	0.0	0	0
4235	0	48	2.0	1	20.0	NaN	0	0
4236	0	44	1.0	1	15.0	0.0	0	0
4237	0	52	2.0	0	0.0	0.0	0	0

4238 rows × 16 columns

In [4]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4238 non-null   int64  
 1   age              4238 non-null   int64  
 2   education        4133 non-null   float64
 3   currentSmoker    4238 non-null   int64  
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   int64  
 7   prevalentHyp     4238 non-null   int64  
 8   diabetes         4238 non-null   int64  
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
```

In [5]:

In [6]:

Out[6]:
```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

In [7]:
```
f_m=df[['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose']]
```

In [8]:

Out[8]:
```
(3656, 15)
```

In [9]:

Out[9]:
```
(3656,)
```

In [10]:

In [11]:

In [12]:
```
logr=LogisticRegression()
```

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

In [13]:

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

Out[14]:
```
array([1], dtype=int64)
```

In [15]:

Out[15]:
```
array([0, 1], dtype=int64)
```

In [16]:

Out[16]:
```
0.0002214783507201723
```

In [17]:

Out[17]:
```
0.9997785216492798
```

RANDOM FOREST

In [18]:

```
df['TenYearCHD'].value_counts()
```

Out[18]:

```
0    3099
1     557
Name: TenYearCHD, dtype: int64
```

In [19]:

```
x=df[['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose']]
y=df['TenYearCHD']
```

In [20]:

```
#g1={"Verified":{'True':1,'False':2}}
#df=df.replace(g1)
```

In [21]:

```
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 [22]:

```
rfc=RandomForestClassifier()
```

Out[22]:

```
RandomForestClassifier()
```

In [23]:

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

In [24]:

```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="ac
```

Out[24]:

```
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 [25]:

```
grid_search.best_score_
```

Out[25]:

```
0.8468154441946834
```

In [26]:

```
rfc_best=grid_search.best_estimator_
```

In [27]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','
```

Out[27]:
```
[Text(2232.0, 1902.6000000000001, 'diaBP <= 99.25\ngini = 0.273\nsamples = 16
37\nvalue = [2142, 417]\nclass = Yes'),
 Text(1116.0, 1359.0, 'sysBP <= 137.75\ngini = 0.238\nsamples = 1477\nvalue =
[1977, 316]\nclass = Yes'),
 Text(558.0, 815.4000000000001, 'sysBP <= 108.75\ngini = 0.188\nsamples = 108
0\nvalue = [1488, 175]\nclass = Yes'),
 Text(279.0, 271.79999999999995, 'gini = 0.078\nsamples = 165\nvalue = [236,
10]\nclass = Yes'),
 Text(837.0, 271.79999999999995, 'gini = 0.206\nsamples = 915\nvalue = [1252,
165]\nclass = Yes'),
 Text(1674.0, 815.4000000000001, 'BMI <= 37.065\ngini = 0.347\nsamples = 397\
nvalue = [489, 141]\nclass = Yes'),
 Text(1395.0, 271.79999999999995, 'gini = 0.337\nsamples = 384\nvalue = [480,
131]\nclass = Yes'),
 Text(1953.0, 271.79999999999995, 'gini = 0.499\nsamples = 13\nvalue = [9, 1
0]\nclass = No'),
 Text(3348.0, 1359.0, 'totChol <= 257.5\ngini = 0.471\nsamples = 160\nvalue =
[165, 101]\nclass = Yes'),
 Text(2790.0, 815.4000000000001, 'sysBP <= 182.5\ngini = 0.434\nsamples = 10
6\nvalue = [118, 55]\nclass = Yes'),
 Text(2511.0, 271.79999999999995, 'gini = 0.373\nsamples = 82\nvalue = [103,
34]\nclass = Yes'),
 Text(3069.0, 271.79999999999995, 'gini = 0.486\nsamples = 24\nvalue = [15, 2
1]\nclass = No'),
 Text(3906.0, 815.4000000000001, 'BMI <= 28.075\ngini = 0.5\nsamples = 54\nva
lue = [47, 46]\nclass = Yes'),
 Text(3627.0, 271.79999999999995, 'gini = 0.44\nsamples = 34\nvalue = [35, 1
7]\nclass = Yes'),
 Text(4185.0, 271.79999999999995, 'gini = 0.414\nsamples = 20\nvalue = [12, 2
9]\nclass = No')]
```