In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

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

In [3]:
```
df=pd.read_csv(r"E:\154\C4_framingham - C4_framingham.csv").dropna()
df
```

Out[3]:

e	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	t
1	39	4.0	0	0.0	0.0	0	0	0	
0	46	2.0	0	0.0	0.0	0	0	0	
1	48	1.0	1	20.0	0.0	0	0	0	
0	61	3.0	1	30.0	0.0	0	1	0	
0	46	3.0	1	23.0	0.0	0	0	0	
..	...	...	...	...	...	...	...	...	
1	58	3.0	0	0.0	0.0	0	1	0	
1	68	1.0	0	0.0	0.0	0	1	0	
1	50	1.0	1	1.0	0.0	0	1	0	
1	51	3.0	1	43.0	0.0	0	0	0	
0	52	2.0	0	0.0	0.0	0	0	0	

× 16 columns

In [4]:
```
df.head()
```

Out[4]:

	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	

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3656 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             3656 non-null   int64  
 1   age              3656 non-null   int64  
 2   education        3656 non-null   float64
 3   currentSmoker    3656 non-null   int64  
 4   cigsPerDay       3656 non-null   float64
 5   BPMeds           3656 non-null   float64
 6   prevalentStroke  3656 non-null   int64  
 7   prevalentHyp     3656 non-null   int64  
 8   diabetes         3656 non-null   int64  
 9   totChol          3656 non-null   float64
 10  sysBP            3656 non-null   float64
 11  diaBP            3656 non-null   float64
 12  BMI              3656 non-null   float64
 13  heartRate        3656 non-null   float64
 14  glucose          3656 non-null   float64
 15  TenYearCHD       3656 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 485.6 KB
```

In [6]:
```
df.describe()
```

Out[6]:

igsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	
56.000000	3656.000000	3656.000000	3656.000000	3656.000000	3656.000000	3656.000000	365
9.022155	0.030361	0.005744	0.311543	0.027079	236.873085	132.368025	8
11.918869	0.171602	0.075581	0.463187	0.162335	44.096223	22.092444	1
0.000000	0.000000	0.000000	0.000000	0.000000	113.000000	83.500000	4
0.000000	0.000000	0.000000	0.000000	0.000000	206.000000	117.000000	7
0.000000	0.000000	0.000000	0.000000	0.000000	234.000000	128.000000	8
20.000000	0.000000	0.000000	1.000000	0.000000	263.250000	144.000000	9
70.000000	1.000000	1.000000	1.000000	1.000000	600.000000	295.000000	14

In [7]:
```
df.columns
```

Out[7]:
```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

In [8]:

```
feature_matrix = df.iloc[:,0:15]
target_vector = df.iloc[:,-1]
```

In [9]:

```
fs=StandardScaler().fit_transform(feature_matrix)
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

Out[9]:

```
LogisticRegression()
```

In [10]:

```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]]
```

In [11]:

```
prediction=logr.predict(observation)
print(prediction)
```

```
[1]
```

In [12]:

```

logr.classes_


```

Out[12]:

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

In [13]:

```
logr.predict_proba(observation)[0][0]


```

Out[13]:

```
0.0002214783507201723
```

In [14]:

```
logr.predict_proba(observation)[0][1]

```

Out[14]:

```
0.9997785216492798
```

Random Forest

In [15]:

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

Out[15]:

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

In [16]:

```
x=df[['TenYearCHD']]
y=df['TenYearCHD']
```

In [18]:

```
g1={'TenYearCHD':{'0':1, "1":2}}
df=df.replace(g1)
df
```

Out[18]:

	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
...	...	...	...	...	...	...	...	...
4231	1	58	3.0	0	0.0	0.0	0	1
4232	1	68	1.0	0	0.0	0.0	0	1
4233	1	50	1.0	1	1.0	0.0	0	1
4234	1	51	3.0	1	43.0	0.0	0	0
4237	0	52	2.0	0	0.0	0.0	0	0

3656 rows × 16 columns

In [24]:

```
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 [25]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[25]:

```
RandomForestClassifier()
```

In [26]:

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

In [27]:

```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="a
grid_search.fit(x_train,y_train)
```

Out[27]:

```
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 [28]:

```
grid_search.best_score_
```

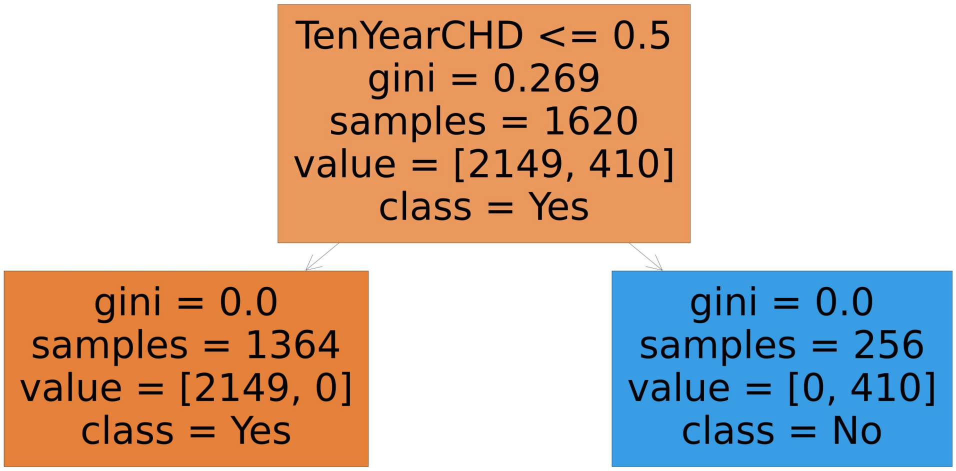
Out[28]:

```
1.0
```

In [29]:
```
rfc_best = grid_search.best_estimator_
```

In [31]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(95,40))
plot_tree(rfc_best.estimators_[5], feature_names=x.columns, class_names=['Yes'
```

Out[31]:
```
[Text(2650.5, 1630.8000000000002, 'TenYearCHD <= 0.5\ngini = 0.269\nsamples =
1620\nvalue = [2149, 410]\nclass = Yes'),
 Text(1325.25, 543.5999999999999, 'gini = 0.0\nsamples = 1364\nvalue = [2149,
0]\nclass = Yes'),
 Text(3975.75, 543.5999999999999, 'gini = 0.0\nsamples = 256\nvalue = [0, 41
0]\nclass = No')]
```



In [ ]:
```
```