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("C5_health care diabetes.csv")
df
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

768 rows × 9 columns

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

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

768 rows × 9 columns

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 60.0 KB
```

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

Out[6]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [7]:
```
feature_matrix=df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin
       'BMI', 'DiabetesPedigreeFunction', 'Age']]
target_vector=df[ 'Outcome']
```

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

Out[8]:
```
(768, 8)
```

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

Out[9]:
```
(768,)
```

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=[[1,2,3,4,5,6,7,8]]
```

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

```
[1]
```

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

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

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

Out[16]:
```
0.00029236948687560993
```

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

Out[17]:
```
array([[2.92369487e-04, 9.99707631e-01]])
```

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

Out[18]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [19]:
```
x=df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']]
y=df['Outcome']
```

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]:
```
0.7877087610275759
```

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

In [29]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],fill
```

Out[29]:
```
[Text(2232.0, 1993.2, 'Age <= 28.5\ngini = 0.446\nsamples = 350\nvalue = [357, 180]\ncla
ss = Yes'),
 Text(1395.0, 1630.8000000000002, 'Pregnancies <= 2.5\ngini = 0.286\nsamples = 170\nvalu
e = [206, 43]\nclass = Yes'),
 Text(837.0, 1268.4, 'Glucose <= 132.5\ngini = 0.232\nsamples = 125\nvalue = [162, 25]\n
class = Yes'),
 Text(558.0, 906.0, 'Age <= 23.5\ngini = 0.039\nsamples = 99\nvalue = [147, 3]\nclass =
Yes'),
 Text(279.0, 543.5999999999999, 'gini = 0.0\nsamples = 59\nvalue = [91, 0]\nclass = Ye
s'),
 Text(837.0, 543.5999999999999, 'Glucose <= 100.5\ngini = 0.097\nsamples = 40\nvalue =
[56, 3]\nclass = Yes'),
 Text(558.0, 181.19999999999982, 'gini = 0.0\nsamples = 20\nvalue = [31, 0]\nclass = Ye
s'),
 Text(1116.0, 181.19999999999982, 'gini = 0.191\nsamples = 20\nvalue = [25, 3]\nclass =
Yes'),
 Text(1116.0, 906.0, 'gini = 0.482\nsamples = 26\nvalue = [15, 22]\nclass = No'),
 Text(1953.0, 1268.4, 'SkinThickness <= 18.5\ngini = 0.412\nsamples = 45\nvalue = [44, 1
8]\nclass = Yes'),
 Text(1674.0, 906.0, 'gini = 0.452\nsamples = 20\nvalue = [19, 10]\nclass = Yes'),
 Text(2232.0, 906.0, 'gini = 0.367\nsamples = 25\nvalue = [25, 8]\nclass = Yes'),
 Text(3069.0, 1630.8000000000002, 'DiabetesPedigreeFunction <= 0.204\ngini = 0.499\nsamp
les = 180\nvalue = [151, 137]\nclass = Yes'),
```

```
 Text(2790.0, 1268.4, 'gini = 0.375\nsamples = 33\nvalue = [42, 14]\nclass = Yes'),
 Text(3348.0, 1268.4, 'Glucose <= 139.5\ngini = 0.498\nsamples = 147\nvalue = [109, 123]
\nclass = No'),
 Text(2790.0, 906.0, 'BMI <= 27.55\ngini = 0.474\nsamples = 98\nvalue = [94, 59]\nclass
= Yes'),
 Text(2511.0, 543.5999999999999, 'gini = 0.268\nsamples = 23\nvalue = [37, 7]\nclass = Y
es'),
 Text(3069.0, 543.5999999999999, 'DiabetesPedigreeFunction <= 0.514\ngini = 0.499\nsampl
es = 75\nvalue = [57, 52]\nclass = Yes'),
 Text(2790.0, 181.19999999999982, 'gini = 0.484\nsamples = 53\nvalue = [43, 30]\nclass =
Yes'),
 Text(3348.0, 181.19999999999982, 'gini = 0.475\nsamples = 22\nvalue = [14, 22]\nclass =
No'),
 Text(3906.0, 906.0, 'SkinThickness <= 23.5\ngini = 0.308\nsamples = 49\nvalue = [15, 6
4]\nclass = No'),
 Text(3627.0, 543.5999999999999, 'gini = 0.202\nsamples = 23\nvalue = [4, 31]\nclass = N
o'),
 Text(4185.0, 543.5999999999999, 'gini = 0.375\nsamples = 26\nvalue = [11, 33]\nclass =
No')]
```