In [2]:
```
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
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.tree import plot_tree
```

In [3]:
```
df=pd.read_csv("C5_health care diabetes - C5_health care diabetes.csv")
df
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
...	...	...	...	...	...	...	.
763	10	101	76	48	180	32.9	0.17
764	2	122	70	27	0	36.8	0.34
765	5	121	72	23	112	26.2	0.24
766	1	126	60	0	0	30.1	0.34
767	1	93	70	31	0	30.4	0.31

768 rows × 9 columns

In [4]:
```
df1=df.fillna(value=0)
df1
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
...	...	...	...	...	...	...	.
763	10	101	76	48	180	32.9	0.17
764	2	122	70	27	0	36.8	0.34
765	5	121	72	23	112	26.2	0.24
766	1	126	60	0	0	30.1	0.34
767	1	93	70	31	0	30.4	0.31

768 rows × 9 columns

In [5]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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: 54.1 KB
```

In [6]:
```
df1.columns
```

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

In [7]:
```
df2=df1[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']]
df2
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
...	...	...	...	...	...	...	.
763	10	101	76	48	180	32.9	0.17
764	2	122	70	27	0	36.8	0.34
765	5	121	72	23	112	26.2	0.24
766	1	126	60	0	0	30.1	0.34
767	1	93	70	31	0	30.4	0.31

768 rows × 9 columns

In [9]:
```
df2['Outcome'].value_counts()
```

Out[9]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [10]:
```
x=df2.drop('Outcome',axis=1)
y=df2['Outcome']
```

In [11]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.70)
```

In [12]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

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

In [13]:
```
parameters = {'max_depth':[1,2,3,4,5],
              'min_samples_leaf':[5,10,15,20,25],
               'n_estimators':[10,20,30,40,50]}
```

In [14]:
```
grid_search = GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring='ac
grid_search.fit(x_train,y_train)
```

Out[14]:
```
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 [15]:
```
grid_search.best_score_
```

Out[15]:
```
0.7608695652173914
```

In [16]:
```
rfc_best =grid_search.best_estimator_
```

In [17]:
```
plt.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','N
```

```
s'),
 Text(1488.0, 226.5, 'gini = 0.459\nsamples = 23\nvalue = [27, 15]\nclass =
Yes'),
 Text(2604.0, 679.5, 'Pregnancies <= 6.5\ngini = 0.499\nsamples = 44\nvalue
= [29, 31]\nclass = No'),
 Text(2232.0, 226.5, 'gini = 0.494\nsamples = 34\nvalue = [25, 20]\nclass =
Yes'),
 Text(2976.0, 226.5, 'gini = 0.391\nsamples = 10\nvalue = [4, 11]\nclass =
No'),
 Text(3720.0, 1132.5, 'SkinThickness <= 14.5\ngini = 0.401\nsamples = 13\nv
alue = [5, 13]\nclass = No'),
 Text(3348.0, 679.5, 'gini = 0.18\nsamples = 8\nvalue = [1, 9]\nclass = N
o'),
 Text(4092.0, 679.5, 'gini = 0.5\nsamples = 5\nvalue = [4, 4]\nclass = Ye
s'),
 Text(2511.0, 2038.5, 'gini = 0.117\nsamples = 9\nvalue = [1, 15]\nclass =
No')]
```