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

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

In [3]:
```
df=pd.read_csv("C5 health.csv").dropna()

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]:
```
df.dropna(inplace=True)
```

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]:
```
feature_matrix = df[['Pregnancies','Glucose','BloodPressure','SkinThickness','
target_vector = df['Outcome']
```

In [7]:
```
feature_matrix.shape
```

Out[7]:
```
(768, 8)
```

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

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

In [9]:
```
from sklearn.preprocessing import StandardScaler
```

In [10]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [11]:
```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

Out[11]:
```
LogisticRegression()
```

In [12]:
```
feature_matrix.shape
```

Out[12]:
```
(768, 8)
```

In [13]:
```
target_vector.shape
```

Out[13]:
```
(768,)
```

In [14]:
```
from sklearn.preprocessing import StandardScaler
```

In [15]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

Loading [MathJax]/extensions/Safe.js

In [16]:
```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```
Out[16]:
```
LogisticRegression()
```

In [17]:
```
observation=df[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insul
```

In [18]:
```
prediction = logr.predict(observation)
prediction
```
Out[18]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

In [19]:
```
logr.classes_
```
Out[19]:
```
array([0, 1], dtype=int64)
```

In [20]:
```
logr.predict_proba(observation)[0][1]
```
Out[20]:
```
1.0
```

Random Forest

In [21]:
```
df['Outcome'].value_counts()
```
Out[21]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [22]:
```
x=df[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI',
y=df['Outcome']
```

Loading [MathJax]/extensions/Safe.js

In [23]:
```
g1={'Outcome':{"1":1, "0":2}}
df=df.replace(g1)
df
```

Out[23]:

	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 [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]:
```
0.7821256172668257
```

Loading [MathJax]/extensions/Safe.js

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

Loading [MathJax]/extensions/Safe.js

In [30]:
```
from sklearn.tree import plot_tree
plt.figure(figsize = (80,40,))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','
```

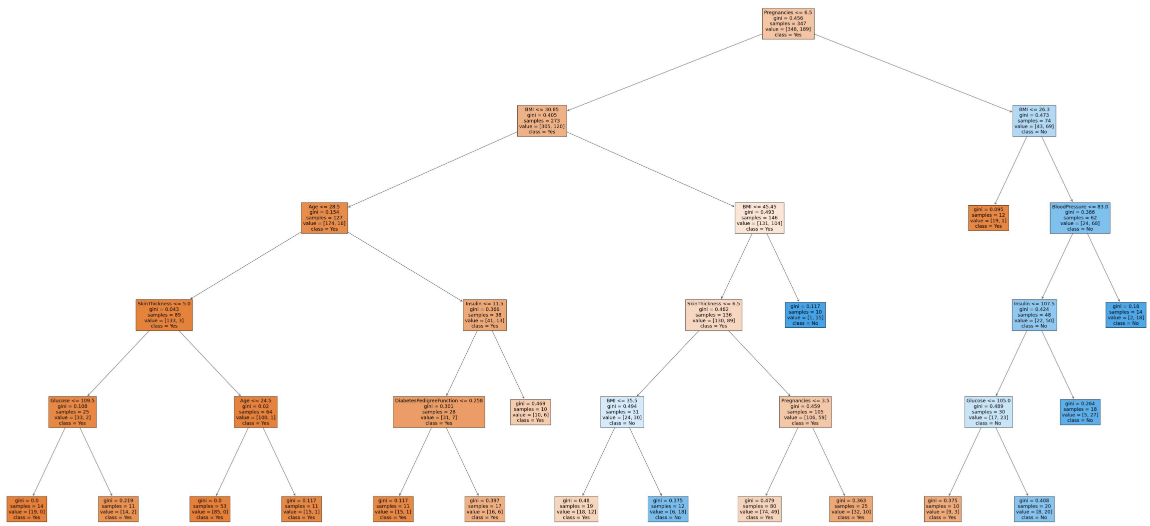
Loading [MathJax]/extensions/Safe.js

Out[30]:

```
[Text(3026.076923076923, 1993.2, 'Pregnancies <= 6.5\ngini = 0.456\nsamples =
347\nvalue = [348, 189]\nclass = Yes'),
 Text(2103.230769230769, 1630.8000000000002, 'BMI <= 30.85\ngini = 0.405\nsam
ples = 273\nvalue = [305, 120]\nclass = Yes'),
 Text(1287.6923076923076, 1268.4, 'Age <= 28.5\ngini = 0.154\nsamples = 127\n
value = [174, 16]\nclass = Yes'),
 Text(686.7692307692307, 906.0, 'SkinThickness <= 5.0\ngini = 0.043\nsamples
= 89\nvalue = [133, 3]\nclass = Yes'),
 Text(343.38461538461536, 543.5999999999999, 'Glucose <= 109.5\ngini = 0.108
\nsamples = 25\nvalue = [33, 2]\nclass = Yes'),
 Text(171.69230769230768, 181.19999999999982, 'gini = 0.0\nsamples = 14\nvalu
e = [19, 0]\nclass = Yes'),
 Text(515.0769230769231, 181.19999999999982, 'gini = 0.219\nsamples = 11\nval
ue = [14, 2]\nclass = Yes'),
 Text(1030.1538461538462, 543.5999999999999, 'Age <= 24.5\ngini = 0.02\nsampl
es = 64\nvalue = [100, 1]\nclass = Yes'),
 Text(858.4615384615383, 181.19999999999982, 'gini = 0.0\nsamples = 53\nvalue
= [85, 0]\nclass = Yes'),
 Text(1201.8461538461538, 181.19999999999982, 'gini = 0.117\nsamples = 11\nva
lue = [15, 1]\nclass = Yes'),
 Text(1888.6153846153845, 906.0, 'Insulin <= 11.5\ngini = 0.366\nsamples = 38
\nvalue = [41, 13]\nclass = Yes'),
 Text(1716.9230769230767, 543.5999999999999, 'DiabetesPedigreeFunction <= 0.2
58\ngini = 0.301\nsamples = 28\nvalue = [31, 7]\nclass = Yes'),
 Text(1545.230769230769, 181.19999999999982, 'gini = 0.117\nsamples = 11\nval
ue = [15, 1]\nclass = Yes'),
 Text(1888.6153846153845, 181.19999999999982, 'gini = 0.397\nsamples = 17\nva
lue = [16, 6]\nclass = Yes'),
 Text(2060.3076923076924, 543.5999999999999, 'gini = 0.469\nsamples = 10\nval
ue = [10, 6]\nclass = Yes'),
 Text(2918.7692307692305, 1268.4, 'BMI <= 45.45\ngini = 0.493\nsamples = 146
\nvalue = [131, 104]\nclass = Yes'),
 Text(2747.076923076923, 906.0, 'SkinThickness <= 6.5\ngini = 0.482\nsamples
= 136\nvalue = [130, 89]\nclass = Yes'),
 Text(2403.6923076923076, 543.5999999999999, 'BMI <= 35.5\ngini = 0.494\nsamp
les = 31\nvalue = [24, 30]\nclass = No'),
 Text(2232.0, 181.19999999999982, 'gini = 0.48\nsamples = 19\nvalue = [18, 1
2]\nclass = Yes'),
 Text(2575.3846153846152, 181.19999999999982, 'gini = 0.375\nsamples = 12\nva
lue = [6, 18]\nclass = No'),
 Text(3090.461538461538, 543.5999999999999, 'Pregnancies <= 3.5\ngini = 0.459
\nsamples = 105\nvalue = [106, 59]\nclass = Yes'),
 Text(2918.7692307692305, 181.19999999999982, 'gini = 0.479\nsamples = 80\nva
lue = [74, 49]\nclass = Yes'),
 Text(3262.1538461538457, 181.19999999999982, 'gini = 0.363\nsamples = 25\nva
lue = [32, 10]\nclass = Yes'),
 Text(3090.461538461538, 906.0, 'gini = 0.117\nsamples = 10\nvalue = [1, 15]
\nclass = No'),
 Text(3948.9230769230767, 1630.8000000000002, 'BMI <= 26.3\ngini = 0.473\nsam
ples = 74\nvalue = [43, 69]\nclass = No'),
 Text(3777.230769230769, 1268.4, 'gini = 0.095\nsamples = 12\nvalue = [19, 1]
\nclass = Yes'),
 Text(4120.615384615385, 1268.4, 'BloodPressure <= 83.0\ngini = 0.386\nsample
s = 62\nvalue = [24, 68]\nclass = No'),
 Text(3948.9230769230767, 906.0, 'Insulin <= 107.5\ngini = 0.424\nsamples = 4
8\nvalue = [22, 50]\nclass = No'),
 Text(3777.230769230769, 543.5999999999999, 'Glucose <= 105.0\ngini = 0.489\n
```

Loading [MathJax]/extensions/Safe.js

```
samples = 30\nvalue = [17, 23]\nclass = No'),
 Text(3605.5384615384614, 181.19999999999982, 'gini = 0.375\nsamples = 10\nva
lue = [9, 3]\nclass = Yes'),
 Text(3948.9230769230767, 181.19999999999982, 'gini = 0.408\nsamples = 20\nva
lue = [8, 20]\nclass = No'),
 Text(4120.615384615385, 543.5999999999999, 'gini = 0.264\nsamples = 18\nvalu
e = [5, 27]\nclass = No'),
 Text(4292.307692307692, 906.0, 'gini = 0.18\nsamples = 14\nvalue = [2, 18]\n
class = No')]
```



Loading [MathJax]/extensions/Safe.js