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\C5_health care diabetes - C5_health care diabetes.csv")
df
```

Out[3]:

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

768 rows × 9 columns

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

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

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

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.00(
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.24(
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.76(
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.00(
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.00(
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.00(
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.00(
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.00(

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

Out[7]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [8]:
```
feature_matrix = df.iloc[:,0:8]
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]]
```

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]:
```
0.00029236948687560993
```

In [14]:
```
logr.predict_proba(observation)[0][1]
```

Out[14]:
```
0.9997076305131244
```

Random Forest

In [15]:
```
df['Outcome'].value_counts()
```

Out[15]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [16]:
```
x=df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']]
y=df['Outcome']
```

In [17]:
```
g1={'Outcome':{'True':1, "False":2}}
df=df.replace(g1)
df
```

Out[17]:

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

768 rows × 9 columns

In [18]:
```
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 [19]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[19]:
```
RandomForestClassifier()
```

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

In [21]:
```
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[21]:
```
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')
```

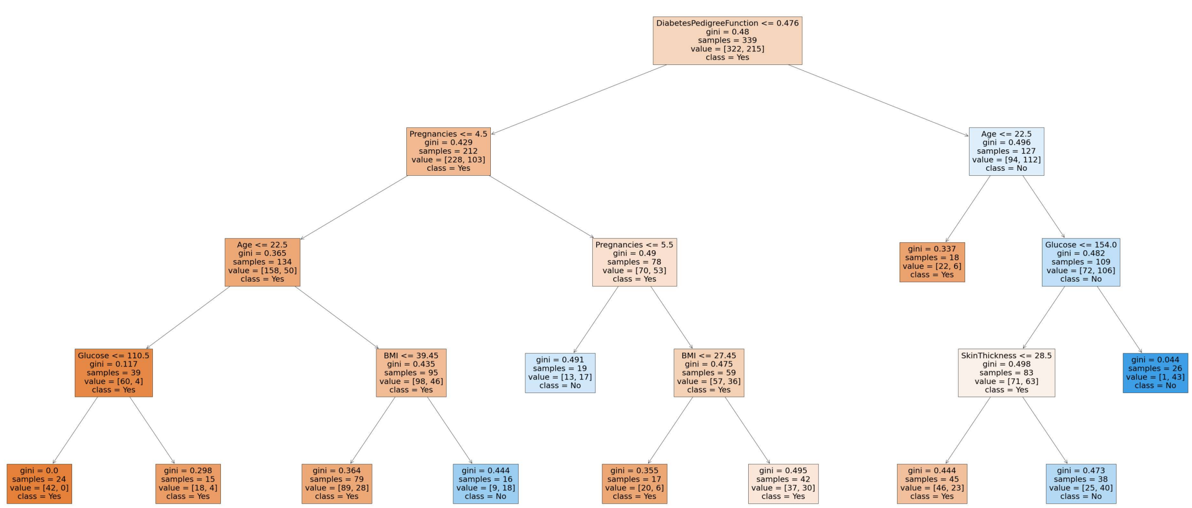
In [22]:
```
grid_search.best_score_
```

Out[22]:
```
0.7616448427009932
```

In [23]:
```
rfc_best = grid_search.best_estimator_
```

In [24]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(89,40))
plot_tree(rfc_best.estimators_[5], feature_names=x.columns, class_names=['Yes', 'No'], filled=True
```

Out[24]:
```
[Text(2994.326470588235, 1956.96, 'DiabetesPedigreeFunction <= 0.476\ngini = 0.48\nsamples = 339
\nvalue = [322, 215]\nclass = Yes'),
 Text(1898.8411764705882, 1522.0800000000002, 'Pregnancies <= 4.5\ngini = 0.429\nsamples = 212\nv
alue = [228, 103]\nclass = Yes'),
 Text(1168.5176470588235, 1087.2, 'Age <= 22.5\ngini = 0.365\nsamples = 134\nvalue = [158, 50]\nc
lass = Yes'),
 Text(584.2588235294118, 652.3200000000002, 'Glucose <= 110.5\ngini = 0.117\nsamples = 39\nvalue
= [60, 4]\nclass = Yes'),
 Text(292.1294117647059, 217.44000000000005, 'gini = 0.0\nsamples = 24\nvalue = [42, 0]\nclass =
Yes'),
 Text(876.3882352941176, 217.44000000000005, 'gini = 0.298\nsamples = 15\nvalue = [18, 4]\nclass
= Yes'),
 Text(1752.7764705882353, 652.3200000000002, 'BMI <= 39.45\ngini = 0.435\nsamples = 95\nvalue =
[98, 46]\nclass = Yes'),
 Text(1460.6470588235293, 217.44000000000005, 'gini = 0.364\nsamples = 79\nvalue = [89, 28]\nclas
s = Yes'),
 Text(2044.9058823529413, 217.44000000000005, 'gini = 0.444\nsamples = 16\nvalue = [9, 18]\nclass
= No'),
 Text(2629.164705882353, 1087.2, 'Pregnancies <= 5.5\ngini = 0.49\nsamples = 78\nvalue = [70, 53]
\nclass = Yes'),
 Text(2337.035294117647, 652.3200000000002, 'gini = 0.491\nsamples = 19\nvalue = [13, 17]\nclass
= No'),
 Text(2921.2941176470586, 652.3200000000002, 'BMI <= 27.45\ngini = 0.475\nsamples = 59\nvalue =
[57, 36]\nclass = Yes'),
 Text(2629.164705882353, 217.44000000000005, 'gini = 0.355\nsamples = 17\nvalue = [20, 6]\nclass
= Yes'),
 Text(3213.423529411765, 217.44000000000005, 'gini = 0.495\nsamples = 42\nvalue = [37, 30]\nclass
= Yes'),
 Text(4089.8117647058825, 1522.0800000000002, 'Age <= 22.5\ngini = 0.496\nsamples = 127\nvalue =
[94, 112]\nclass = No'),
 Text(3797.6823529411763, 1087.2, 'gini = 0.337\nsamples = 18\nvalue = [22, 6]\nclass = Yes'),
 Text(4381.941176470588, 1087.2, 'Glucose <= 154.0\ngini = 0.482\nsamples = 109\nvalue = [72, 10
6]\nclass = No'),
 Text(4089.8117647058825, 652.3200000000002, 'SkinThickness <= 28.5\ngini = 0.498\nsamples = 83\n
value = [71, 63]\nclass = Yes'),
 Text(3797.6823529411763, 217.44000000000005, 'gini = 0.444\nsamples = 45\nvalue = [46, 23]\nclas
s = Yes'),
 Text(4381.941176470588, 217.44000000000005, 'gini = 0.473\nsamples = 38\nvalue = [25, 40]\nclass
= No'),
 Text(4674.070588235294, 652.3200000000002, 'gini = 0.044\nsamples = 26\nvalue = [1, 43]\nclass =
No')]
```



In [ ]: