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\C6_bmi - C6_bmi.csv").dropna()
df
```

Out[3]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3
...	...	...	...	...
495	Female	150	153	5
496	Female	184	121	4
497	Female	141	136	5
498	Male	150	95	5
499	Male	173	131	5

500 rows × 4 columns

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

Out[4]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Gender  500 non-null    object
 1   Height  500 non-null    int64 
 2   Weight  500 non-null    int64 
 3   Index   500 non-null    int64 
dtypes: int64(3), object(1)
memory usage: 19.5+ KB
```

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

Out[6]:

	Height	Weight	Index
count	500.000000	500.000000	500.000000
mean	169.944000	106.000000	3.748000
std	16.375261	32.382607	1.355053
min	140.000000	50.000000	0.000000
25%	156.000000	80.000000	3.000000
50%	170.500000	106.000000	4.000000
75%	184.000000	136.000000	5.000000
max	199.000000	160.000000	5.000000

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

Out[7]:
```
Index(['Gender', 'Height', 'Weight', 'Index'], dtype='object')
```

In [8]:
```
feature_matrix = df[['Height','Weight']]
target_vector = df[['Index']]
```

In [9]:
```
fs=StandardScaler().fit_transform(feature_matrix)
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: Da
taConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

Out[9]:
```
LogisticRegression()
```

In [10]:
```
observation=[[1,2]]
```

In [11]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[5]
```

In [12]:
```
logr.classes_
```

Out[12]:
```
array([0, 1, 2, 3, 4, 5], dtype=int64)
```

In [13]:
```
logr.predict_proba(observation)[0][0]
```

Out[13]:
```
5.5956697582538237e-11
```

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

Out[14]:
```
6.059900360819463e-10
```

Random Forest

In [37]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [38]:
```
df=pd.read_csv(r"E:\154\C6_bmi - C6_bmi.csv")
```

In [39]:
```
df['Gender'].value_counts()
```

Out[39]:
```
Female    255
Male      245
Name: Gender, dtype: int64
```

In [40]:
```
x=df[['Height','Weight']]
y=df['Gender']
```

In [41]:
```
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 [42]:
```
from sklearn.ensemble import RandomForestClassifier
rfc =RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[42]:
```
RandomForestClassifier()
```

In [43]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
           }
```

In [44]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="ac
grid_search.fit(x_train,y_train)
```

Out[44]:
```
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')
```

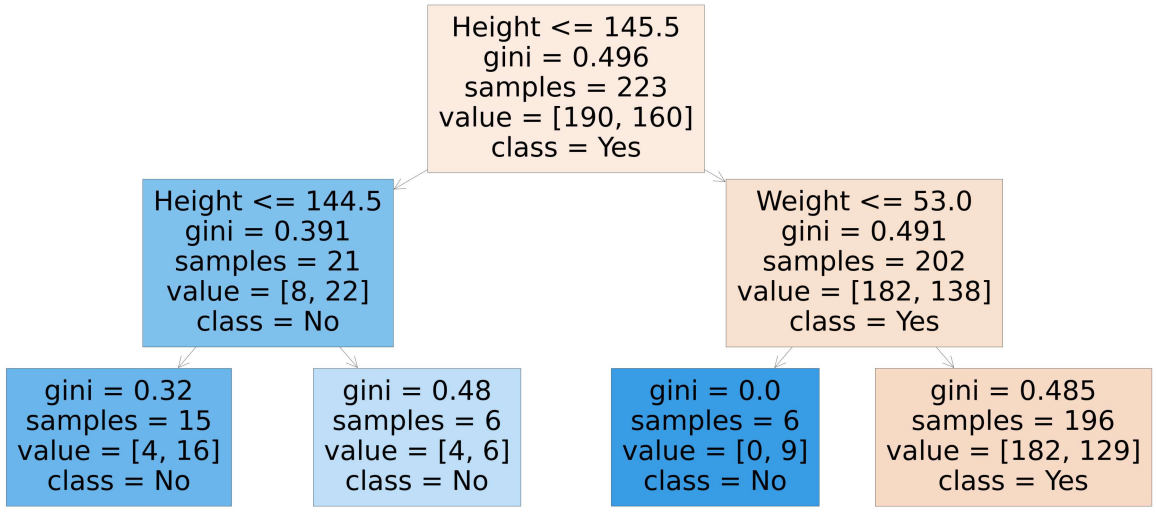
In [45]:
```
grid_search.best_score_
```

Out[45]:
```
0.5314285714285714
```

In [46]:
```
rfc_best=grid_search.best_estimator_
```

In [47]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(89,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns, class_names=["Yes",
          )
```

Out[47]:
```
[Text(2483.1, 1812.0, 'Height <= 145.5\ngini = 0.496\nsamples = 223\nvalue =
[190, 160]\nclass = Yes'),
 Text(1241.55, 1087.2, 'Height <= 144.5\ngini = 0.391\nsamples = 21\nvalue =
[8, 22]\nclass = No'),
 Text(620.775, 362.39999999999986, 'gini = 0.32\nsamples = 15\nvalue = [4, 1
6]\nclass = No'),
 Text(1862.3249999999998, 362.39999999999986, 'gini = 0.48\nsamples = 6\nvalu
e = [4, 6]\nclass = No'),
 Text(3724.6499999999996, 1087.2, 'Weight <= 53.0\ngini = 0.491\nsamples = 20
2\nvalue = [182, 138]\nclass = Yes'),
 Text(3103.875, 362.39999999999986, 'gini = 0.0\nsamples = 6\nvalue = [0, 9]
\nclass = No'),
 Text(4345.425, 362.39999999999986, 'gini = 0.485\nsamples = 196\nvalue = [18
2, 129]\nclass = Yes')]
```



In [ ]: