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
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 [2]:
```
df=pd.read_csv("C10_loan1 - C10_loan1.csv")
df
```

Out[2]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	Yes	Single	125	No
1	No	Married	100	No
2	No	Single	70	No
3	Yes	Married	120	No
4	No	Divorced	95	Yes
5	No	Married	60	No
6	Yes	Divorced	220	No
7	No	Single	85	Yes
8	No	Married	75	No
9	No	Single	90	Yes

In [3]:
```
df1=df.fillna(value=0)
df1
```

Out[3]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	Yes	Single	125	No
1	No	Married	100	No
2	No	Single	70	No
3	Yes	Married	120	No
4	No	Divorced	95	Yes
5	No	Married	60	No
6	Yes	Divorced	220	No
7	No	Single	85	Yes
8	No	Married	75	No
9	No	Single	90	Yes

In [4]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Home Owner          10 non-null     object
 1   Marital Status      10 non-null     object
 2   Annual Income       10 non-null     int64 
 3   Defaulted Borrower  10 non-null     object
dtypes: int64(1), object(3)
memory usage: 448.0+ bytes
```

In [5]:
```
df1.columns
```

Out[5]:
```
Index(['Home Owner', 'Marital Status', 'Annual Income', 'Defaulted Borrowe
r'], dtype='object')
```

In [6]:
```
df2=df1[['Annual Income', 'Defaulted Borrower']]
df2
```

Out[6]:

	Annual Income	Defaulted Borrower
0	125	No
1	100	No
2	70	No
3	120	No
4	95	Yes
5	60	No
6	220	No
7	85	Yes
8	75	No
9	90	Yes

In [7]:
```
df2['Defaulted Borrower'].value_counts()
```

Out[7]:
```
No     7
Yes    3
Name: Defaulted Borrower, dtype: int64
```

In [8]:
```
x=df2.drop('Defaulted Borrower',axis=1)
y=df2['Defaulted Borrower']
```

In [9]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.70)
```

In [10]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[10]:
```
RandomForestClassifier()
```

In [11]:
```
parameters = {'max_depth':[1,2,3,4,5],
              'min_samples_leaf':[5,10,15,20,25],
               'n_estimators':[10,20,30,40,50]}
```

In [12]:
```
grid_search = GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring='ac
grid_search.fit(x_train,y_train)
```

Out[12]:
```
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')
```

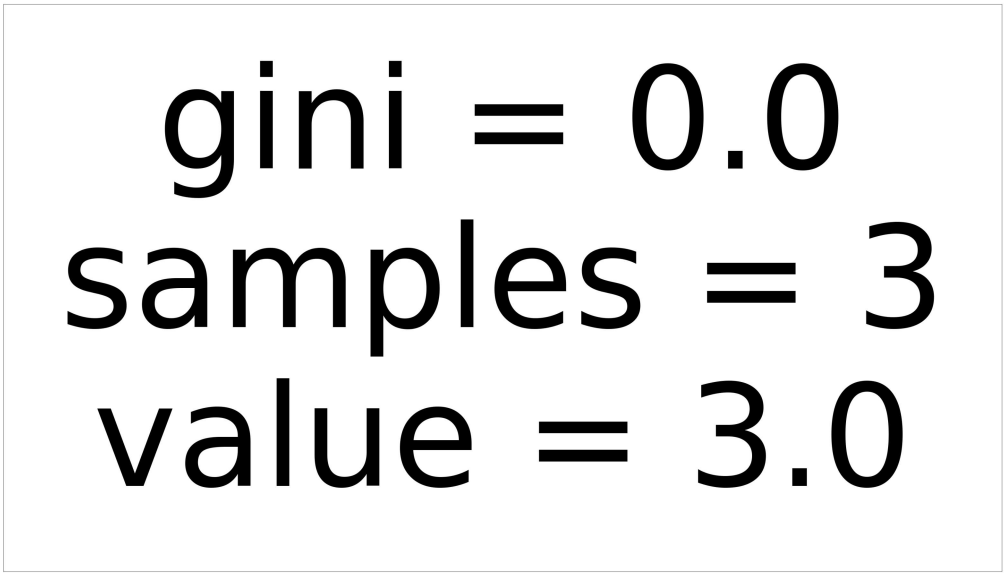
In [13]:
```
grid_search.best_score_
```

Out[13]:
```
1.0
```

In [14]:
```
rfc_best =grid_search.best_estimator_
```

In [16]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','N
```

Out[16]:
```
[Text(2232.0, 1087.2, 'gini = 0.0\nsamples = 3\nvalue = 3.0')]
```



In [ ]:

In [ ]: