```python
In [64]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
```

```python
In [65]: df=pd.read_csv(r"C10_loan1.csv")
         df
```

Out[65]:

|   | 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 |

```python
In [66]: df.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
```

```python
In [67]: df.isnull().sum()
```

```
Out[67]: Home Owner          0
         Marital Status      0
         Annual Income       0
         Defaulted Borrower  0
         dtype: int64
```

In [61]: `df.describe()`

Out[61]:

|       | row_id       | user_id      | gate_id      |
|-------|--------------|--------------|--------------|
| count | 37518.000000 | 37518.000000 | 37518.000000 |
| mean  | 18758.500000 | 28.219015    | 6.819607     |
| std   | 10830.658036 | 17.854464    | 3.197746     |
| min   | 0.000000     | 0.000000     | -1.000000    |
| 25%   | 9379.250000  | 12.000000    | 4.000000     |
| 50%   | 18758.500000 | 29.000000    | 6.000000     |
| 75%   | 28137.750000 | 47.000000    | 10.000000    |
| max   | 37517.000000 | 57.000000    | 16.000000    |

In [62]: `df.columns`

Out[62]: `Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')`

In [63]: 
```python
df['user_id'].value_counts()
```

```
Out[63]:  37    2262
          55    2238
          6     2013
          12    1953
          19    1793
          15    1756
          18    1578
          47    1341
          53    1311
          1     1299
          33    1285
          11    1281
          49    1275
          0     1250
          39    1144
          32    1076
          54    1070
          9     1034
          50     994
          29     990
          3      989
          48     743
          14     696
          17     677
          27     603
          35     601
          46     502
          57     497
          24     416
          42     359
          26     316
          34     284
          23     261
          25     247
          40     242
          31     191
          56     137
          41     124
          43     124
          20     115
          22      96
          28      64
          45      57
          7       49
          36      48
          2       39
          8       29
          10      17
          38      13
          5       10
          30      10
          52       5
          21       5
          44       4
          51       3
```

```
    4          2
Name: user_id, dtype: int64
```

In [ ]:
```python
g1={"gate_id":{'6':1,'5':4}}
df=df.replace(g1)
print(df)
```

In [ ]:
```python
x=df.drop("row_id",axis=1)
y=df["row_id"]
```

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

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

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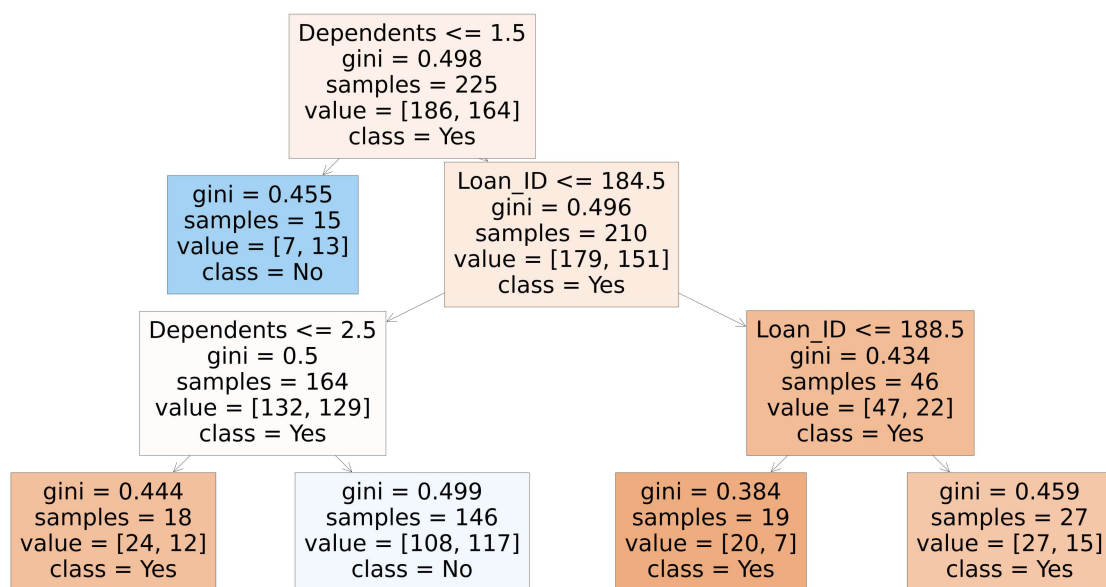
In [ ]:
```python
grid_search.best_score_
```

In [41]:
```python
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [42]:
```python
rfc_best=grid_search.best_estimator_
```

In [43]:
```python
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','N
```

Out[43]: [Text(0.375, 0.875, 'Dependents <= 1.5\ngini = 0.498\nsamples = 225\nvalue =
[186, 164]\nclass = Yes'),
 Text(0.25, 0.625, 'gini = 0.455\nsamples = 15\nvalue = [7, 13]\nclass = N
o'),
 Text(0.5, 0.625, 'Loan_ID <= 184.5\ngini = 0.496\nsamples = 210\nvalue = [17
9, 151]\nclass = Yes'),
 Text(0.25, 0.375, 'Dependents <= 2.5\ngini = 0.5\nsamples = 164\nvalue = [13
2, 129]\nclass = Yes'),
 Text(0.125, 0.125, 'gini = 0.444\nsamples = 18\nvalue = [24, 12]\nclass = Ye
s'),
 Text(0.375, 0.125, 'gini = 0.499\nsamples = 146\nvalue = [108, 117]\nclass =
No'),
 Text(0.75, 0.375, 'Loan_ID <= 188.5\ngini = 0.434\nsamples = 46\nvalue = [4
7, 22]\nclass = Yes'),
 Text(0.625, 0.125, 'gini = 0.384\nsamples = 19\nvalue = [20, 7]\nclass = Ye
s'),
 Text(0.875, 0.125, 'gini = 0.459\nsamples = 27\nvalue = [27, 15]\nclass = Ye
s')]



In [ ]: