RANDOM FOREST - 8

In [29]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [30]:
```
df=pd.read_csv(r"C:\Users\BHOOMISH\Downloads\C8_loan-test.csv")
df
```

Out[30]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amo
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	
...	...	...	...	...	...	...	...	...	...	
362	LP002971	Male	Yes	3+	Not Graduate	Yes	4009	1777	113.0	
363	LP002975	Male	Yes	0	Graduate	No	4158	709	115.0	
364	LP002980	Male	No	0	Graduate	No	3250	1993	126.0	
365	LP002986	Male	Yes	0	Graduate	No	5000	2393	158.0	
366	LP002989	Male	No	0	Graduate	Yes	9200	0	98.0	

367 rows × 12 columns

In [31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            367 non-null    object 
 1   Gender             356 non-null    object 
 2   Married            367 non-null    object 
 3   Dependents         357 non-null    object 
 4   Education          367 non-null    object 
 5   Self_Employed      344 non-null    object 
 6   ApplicantIncome    367 non-null    int64  
 7   CoapplicantIncome  367 non-null    int64  
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

In [32]:
```
df=df.dropna()
```

In [33]:
```
df.isnull().sum()
```

Out[33]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
dtype: int64
```

In [34]:
```
df.describe()
```

Out[34]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	289.000000	289.000000	289.000000	289.000000	289.000000
mean	4637.352941	1528.262976	136.792388	342.671280	0.840830
std	4790.683934	2377.599209	59.699582	65.655503	0.366469
min	0.000000	0.000000	28.000000	6.000000	0.000000
25%	2875.000000	0.000000	102.000000	360.000000	1.000000
50%	3833.000000	879.000000	126.000000	360.000000	1.000000
75%	5000.000000	2400.000000	158.000000	360.000000	1.000000
max	72529.000000	24000.000000	460.000000	480.000000	1.000000

In [35]:
```
df.columns
```

Out[35]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

In [36]:
```
df['Gender'].value_counts()
```

Out[36]:
```
Male      230
Female     59
Name: Gender, dtype: int64
```

In [37]:

```
g1={"Gender":{'Female':1,'Male':2}}
df=df.replace(g1)
print(df)
```

```
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
0    LP001015       2     Yes          0      Graduate            No
1    LP001022       2     Yes          1      Graduate            No
2    LP001031       2     Yes          2      Graduate            No
4    LP001051       2      No          0  Not Graduate            No
5    LP001054       2     Yes          0  Not Graduate           Yes
..        ...     ...     ...        ...           ...           ...
361  LP002969       2     Yes          1      Graduate            No
362  LP002971       2     Yes         3+  Not Graduate           Yes
363  LP002975       2     Yes          0      Graduate            No
365  LP002986       2     Yes          0      Graduate            No
366  LP002989       2      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5720                  0       110.0             360.0
1               3076               1500       126.0             360.0
2               5000               1800       208.0             360.0
4               3276                  0        78.0             360.0
5               2165               3422       152.0             360.0
..               ...                ...         ...               ...
361             2269               2167        99.0             360.0
362             4009               1777       113.0             360.0
363             4158                709       115.0             360.0
365             5000               2393       158.0             360.0
366             9200                  0        98.0             180.0

     Credit_History Property_Area
0               1.0         Urban
1               1.0         Urban
2               1.0         Urban
4               1.0         Urban
5               1.0         Urban
..              ...           ...
361             1.0     Semiurban
362             1.0         Urban
363             1.0         Urban
365             1.0         Rural
366             1.0         Rural

[289 rows x 12 columns]
```

In [38]:
```
x=df.drop("Gender",axis=1)
y=df["Gender"]
```

In [39]:
```
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 [40]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[40], line 3
      1 from sklearn.ensemble import RandomForestClassifier
      2 rfc=RandomForestClassifier()
----> 3 rfc.fit(x_train,y_train)

File ~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:345, in BaseForest.fit(self, X, y, sample_wei
ght)
    343 if issparse(y):
    344     raise ValueError("sparse multilabel-indicator for y is not supported.")
--> 345 X, y = self._validate_data(
    346     X, y, multi_output=True, accept_sparse="csc", dtype=DTYPE
    347 )
    348 if sample_weight is not None:
    349     sample_weight = _check_sample_weight(sample_weight, X)

File ~\anaconda3\lib\site-packages\sklearn\base.py:565, in BaseEstimator._validate_data(self, X, y, reset, v
alidate_separately, **check_params)
    563         y = check_array(y, input_name="y", **check_y_params)
    564     else:
--> 565         X, y = check_X_y(X, y, **check_params)
    566     out = X, y
    568 if not no_val_X and check_params.get("ensure_2d", True):

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:1106, in check_X_y(X, y, accept_sparse, accep
t_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples,
ensure_min_features, y_numeric, estimator)
   1101         estimator_name = _check_estimator_name(estimator)
   1102     raise ValueError(
   1103         f"{estimator_name} requires y to be passed, but the target y is None"
   1104     )
-> 1106 X = check_array(
   1107     X,
   1108     accept_sparse=accept_sparse,
   1109     accept_large_sparse=accept_large_sparse,
   1110     dtype=dtype,
   1111     order=order,
   1112     copy=copy,
   1113     force_all_finite=force_all_finite,
   1114     ensure_2d=ensure_2d,
   1115     allow_nd=allow_nd,
   1116     ensure_min_samples=ensure_min_samples,
```

```
   1117     ensure_min_features=ensure_min_features,
   1118     estimator=estimator,
   1119     input_name="X",
   1120 )
   1122 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, estimator=estimator)
   1124 check_consistent_length(X, y)
```

```
File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:879, in check_array(array, accept_sparse, acc
ept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_
features, estimator, input_name)
    877         array = xp.astype(array, dtype, copy=False)
    878     else:
--> 879         array = _asarray_with_order(array, order=order, dtype=dtype, xp=xp)
    880 except ComplexWarning as complex_warning:
    881     raise ValueError(
    882         "Complex data not supported\n{}\n".format(array)
    883     ) from complex_warning
```

```
File ~\anaconda3\lib\site-packages\sklearn\utils\_array_api.py:185, in _asarray_with_order(array, dtype, ord
er, copy, xp)
    182     xp, _ = get_namespace(array)
    183 if xp.__name__ in {"numpy", "numpy.array_api"}:
    184     # Use NumPy API to support order
--> 185     array = numpy.asarray(array, order=order, dtype=dtype)
    186     return xp.asarray(array, copy=copy)
    187 else:
```

```
File ~\anaconda3\lib\site-packages\pandas\core\generic.py:2070, in NDFrame.__array__(self, dtype)
   2069 def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2070     return np.asarray(self._values, dtype=dtype)
```

```
ValueError: could not convert string to float: 'LP002027'
```

In [ ]:

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

In [ ]:
```
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)
```

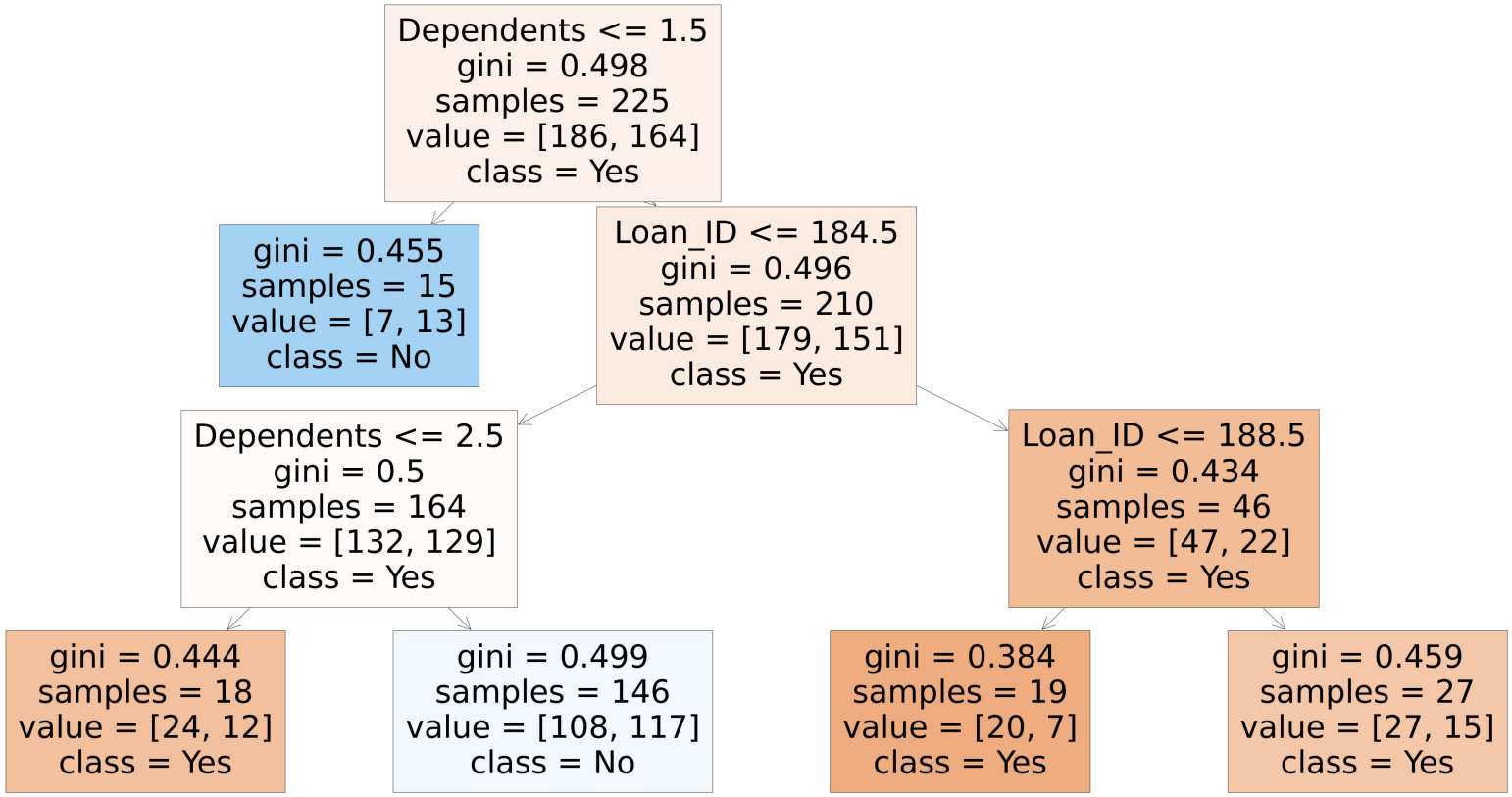
In [ ]:
```
grid_search.best_score_
```

In [41]:
```
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]:
```
rfc_best=grid_search.best_estimator_
```

In [43]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True)
```

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 = No'),
 Text(0.5, 0.625, 'Loan_ID <= 184.5\ngini = 0.496\nsamples = 210\nvalue = [179, 151]\nclass = Yes'),
 Text(0.25, 0.375, 'Dependents <= 2.5\ngini = 0.5\nsamples = 164\nvalue = [132, 129]\nclass = Yes'),
 Text(0.125, 0.125, 'gini = 0.444\nsamples = 18\nvalue = [24, 12]\nclass = Yes'),
 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 = [47, 22]\nclass = Yes'),
 Text(0.625, 0.125, 'gini = 0.384\nsamples = 19\nvalue = [20, 7]\nclass = Yes'),
 Text(0.875, 0.125, 'gini = 0.459\nsamples = 27\nvalue = [27, 15]\nclass = Yes')]
```



In [ ]: