In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.tree import plot_tree
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn import metrics
```

In [2]:

```
data=pd.read_csv("loan_dataset_final.csv")
data
```

Out[2]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	installment	em
0	1000007	1225615	5150	5150	5150.0	60 months	132.58	tom
1	1000030	1225638	20000	20000	20000.0	36 months	635.07	T Thre
2	1000033	1225642	12800	12800	12750.0	60 months	316.54	T-
3	1000045	1225655	14000	14000	14000.0	60 months	349.98	Trad
4	1000067	1225680	15000	15000	14975.0	60 months	370.94	Tru Engi
...	...	...	...	...	...	...	...	
42530	999250	1224611	10000	10000	10000.0	36 months	339.31	Co Ber
42531	999295	1224664	9600	9600	9600.0	36 months	341.17	CU
42532	999634	1225015	1200	1200	1200.0	36 months	39.70	The
42533	999763	1225141	8000	8000	8000.0	60 months	179.66	Lav of Bet
42534	999816	1225200	16000	16000	15950.0	60 months	382.92	cha coun

42535 rows × 47 columns

In [3]:

```
data.isnull().mean().sort_values(ascending=True)
```

Out[3]:

```
id                          0.000000
race_name                   0.000000
```

```
revol_bal                     0.000000
out_prncp                     0.000000
out_prncp_inv                 0.000000
total_pymnt                   0.000000
total_pymnt_inv               0.000000
dti                           0.000000
total_rec_prncp               0.000000
total_rec_late_fee            0.000000
recoveries                    0.000000
collection_recovery_fee       0.000000
interest_rate                 0.000000
number_bc_gt_75               0.000000
fico_score                    0.000000
total_rec_int                 0.000000
addr_state                    0.000000
gender                        0.000000
purpose                       0.000000
loan_amnt                     0.000000
funded_amnt_inv               0.000000
term                          0.000000
installment                   0.000000
member_id                     0.000000
home_ownership                0.000000
funded_amnt                   0.000000
issue_d                       0.000000
verification_status           0.000000
loan_status                   0.000000
pymnt_plan                    0.000000
annual_inc                    0.000094
lti                           0.000094
title                         0.000306
pub_rec                       0.000682
open_acc                      0.000682
inq_last_6mths                0.000682
delinq_2yrs                   0.000682
month_since_oldest_tl         0.000682
total_acc                     0.000682
revol_utilization             0.002116
collections_12_mths_ex_med    0.003409
emp_length                    0.026143
pub_rec_bankruptcies          0.032091
emp_title                     0.061737
desc                          0.317832
mths_since_last_delinq        0.633032
mths_since_last_record        0.914165
dtype: float64
```

In [4]:
```
data.drop(['mths_since_last_delinq','mths_since_last_record','collections_12_mths_ex
```

In [ ]:
```
```

In [5]:
```
def fill_missing(data,variable):
    random_sample=data[variable].dropna().sample(data[variable].isnull().sum(),rando
    random_sample.index=data[data[variable].isnull()].index
    data.loc[data[variable].isnull(),variable]=random_sample
```

In [6]:
```
fill_missing(data,"annual_inc")
```

In [7]:
```
fill_missing(data,"lti")
```

In [8]:
```
```

```
fill_missing(data,"title")
```

In [9]:
```
fill_missing(data,"pub_rec")
```

In [10]:
```
fill_missing(data,"open_acc")
```

In [11]:
```
fill_missing(data,"inq_last_6mths")
```

In [12]:
```
fill_missing(data,"delinq_2yrs")
```

In [13]:
```
fill_missing(data,"month_since_oldest_tl")
```

In [14]:
```
fill_missing(data,"total_acc")
```

In [15]:
```
fill_missing(data,"revol_utilization")
```

In [16]:
```
fill_missing(data,"emp_length")
```

In [17]:
```
fill_missing(data,"pub_rec_bankruptcies")
```

In [18]:
```
fill_missing(data,"emp_title")
```

In [19]:
```
fill_missing(data,"desc")
```

In [20]:
```
data.isnull().mean().sort_values(ascending=True)
```

Out[20]:
```
id                          0.0
revol_bal                   0.0
total_acc                   0.0
out_prncp                   0.0
out_prncp_inv               0.0
total_pymnt                 0.0
total_pymnt_inv             0.0
total_rec_prncp             0.0
total_rec_int               0.0
pub_rec                     0.0
total_rec_late_fee          0.0
collection_recovery_fee     0.0
pub_rec_bankruptcies        0.0
interest_rate               0.0
revol_utilization           0.0
number_bc_gt_75             0.0
fico_score                  0.0
lti                         0.0
month_since_oldest_tl       0.0
recoveries                  0.0
open_acc                    0.0
inq_last_6mths              0.0
delinq_2yrs                 0.0
```

```
member_id                 0.0
loan_amnt                 0.0
funded_amnt               0.0
funded_amnt_inv           0.0
term                      0.0
installment               0.0
emp_title                 0.0
emp_length                0.0
home_ownership            0.0
annual_inc                0.0
verification_status       0.0
issue_d                   0.0
loan_status               0.0
pymnt_plan                0.0
desc                      0.0
purpose                   0.0
title                     0.0
addr_state                0.0
dti                       0.0
race_name                 0.0
gender                    0.0
dtype: float64
```

In [ ]:

In [21]:
```
data['loan_status'].replace('Fully Paid',1,inplace=True)
```

In [22]:
```
data['loan_status'].replace('Charged Off',0,inplace=True)
```

In [23]:
```
data['loan_status'].replace('Does not meet the credit policy. Status:Fully Paid',0,i
```

In [24]:
```
data['loan_status'].replace('Does not meet the credit policy. Status:Charged Off',0,
```

In [25]:
```
data['loan_status'].replace('Current',0,inplace=True)
```

In [26]:
```
data['loan_status'].replace('In Grace Period',0,inplace=True)
```

In [27]:
```
data['loan_status'].replace('Late (31-120 days)',0,inplace=True)
```

In [28]:
```
data['loan_status'].replace('Late (16-30 days)',0,inplace=True)
```

In [29]:
```
data['loan_status'].replace('Default',0,inplace=True)
```

In [30]:
```
data['loan_status'].unique()
```

Out[30]:
```
array([0, 1], dtype=int64)
```

In [32]:
```
data['loan_status'].value_counts()
```

Out[32]:
```
1    33586
```

```
0     8949
Name: loan_status, dtype: int64
```

In [34]:
```
data.shape
```

Out[34]:
```
(42535, 44)
```

In [35]:
```
data.isnull().sum()
```

Out[35]:
```
id                          0
member_id                   0
loan_amnt                   0
funded_amnt                 0
funded_amnt_inv             0
term                        0
installment                 0
emp_title                   0
emp_length                  0
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
pymnt_plan                  0
desc                        0
purpose                     0
title                       0
addr_state                  0
dti                         0
delinq_2yrs                 0
inq_last_6mths              0
open_acc                    0
pub_rec                     0
revol_bal                   0
total_acc                   0
out_prncp                   0
out_prncp_inv               0
total_pymnt                 0
total_pymnt_inv             0
total_rec_prncp             0
total_rec_int               0
total_rec_late_fee          0
recoveries                  0
collection_recovery_fee     0
pub_rec_bankruptcies        0
interest_rate               0
revol_utilization           0
number_bc_gt_75             0
fico_score                  0
lti                         0
month_since_oldest_tl       0
race_name                   0
gender                      0
dtype: int64
```

In [36]:
```
b=data.select_dtypes(include=['object'])
b
```

Out[36]:

	term	emp_title	emp_length	home_ownership	verification_status	issue_d	pymnt_plan

	term	emp_title	emp_length	home_ownership	verification_status	issue_d	pymnt_plan
0	60 months	atlantic tomorrows office	1 year	RENT	Source Verified	Nov-11	n
1	36 months	The Red Threads Inc.	6 years	RENT	Verified	Oct-11	n
2	60 months	T-Mobile USA Inc	9 years	MORTGAGE	Source Verified	Nov-11	n
3	60 months	Trader Joe's	9 years	MORTGAGE	Verified	Oct-11	n
4	60 months	Truevance Engineering	< 1 year	RENT	Verified	Nov-11	n
...	...	...	...	...	...	...	...
42530	36 months	County of San Bernardino	3 years	MORTGAGE	Not Verified	Oct-11	n
42531	36 months	LA CURACAO	5 years	RENT	Source Verified	Nov-11	n
42532	36 months	The Home Depot	3 years	RENT	Source Verified	Nov-11	n

	term	emp_title	emp_length	home_ownership	verification_status	issue_d	pymnt_plan
42533	60 months	Law Office of Melissa Betancourt	4 years	RENT	Verified	Oct-11	n
42534	60 months	champlain coummunity services	1 year	MORTGAGE	Not Verified	Nov-11	n

42535 rows × 13 columns

In [37]:
```
b.dtypes
```

Out[37]:
```
term                  object
emp_title             object
emp_length            object
home_ownership        object
verification_status   object
issue_d               object
pymnt_plan            object
desc                  object
purpose               object
title                 object
addr_state            object
race_name             object
gender                object
dtype: object
```

In [38]:
```
from sklearn.preprocessing import OrdinalEncoder
```

In [39]:
```
enc=OrdinalEncoder()
```

In [40]:
```
k=['term','emp_title','emp_length','home_ownership','verification_status','issue_d',
```

In [41]:
```
enc.fit(data[k])
```

Out[41]:
```
OrdinalEncoder()
```

In [42]:
```
data[k]=enc.transform(data[k])
```

In [43]:
```
data.head(10)
```

Out[43]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	installment	emp_title	e
0	1000007	1225615	5150	5150	5150.0	1.0	132.58	27034.0	
1	1000030	1225638	20000	20000	20000.0	0.0	635.07	23472.0	

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	installment	emp_title	e
2	1000033	1225642	12800	12800	12750.0	1.0	316.54	22633.0	
3	1000045	1225655	14000	14000	14000.0	1.0	349.98	23926.0	
4	1000067	1225680	15000	15000	14975.0	1.0	370.94	24096.0	
5	1000095	1225913	12000	12000	12000.0	0.0	365.23	20632.0	
6	1000115	1225934	30000	30000	29925.0	0.0	938.71	7677.0	
7	1000124	1225944	2800	2800	2725.0	1.0	59.37	23979.0	
8	1000138	1225960	2700	2700	2650.0	0.0	91.62	2480.0	
9	1000142	1225965	6600	6600	6350.0	0.0	206.52	20185.0	

10 rows × 44 columns

In [44]:
```
data.dtypes
```

Out[44]:
```
id                        int64
member_id                 int64
loan_amnt                 int64
funded_amnt               int64
funded_amnt_inv         float64
term                    float64
installment             float64
emp_title               float64
emp_length              float64
home_ownership          float64
annual_inc              float64
verification_status     float64
issue_d                 float64
loan_status               int64
pymnt_plan              float64
desc                    float64
purpose                 float64
title                   float64
addr_state              float64
dti                     float64
delinq_2yrs             float64
inq_last_6mths          float64
open_acc                float64
pub_rec                 float64
revol_bal                 int64
total_acc               float64
out_prncp               float64
out_prncp_inv           float64
total_pymnt             float64
total_pymnt_inv         float64
total_rec_prncp         float64
total_rec_int           float64
total_rec_late_fee      float64
recoveries              float64
collection_recovery_fee float64
pub_rec_bankruptcies    float64
interest_rate           float64
revol_utilization       float64
number_bc_gt_75           int64
fico_score                int64
lti                     float64
month_since_oldest_tl   float64
race_name               float64
```

```
gender                    float64
dtype: object
```

In [45]:
```
df=data[['inq_last_6mths','loan_amnt','number_bc_gt_75','revol_bal','emp_length','ve
df
```

Out[45]:

	inq_last_6mths	loan_amnt	number_bc_gt_75	revol_bal	emp_length	verification_status	hom
0	3.0	5150	2	15187	0.0	1.0	
1	2.0	20000	0	15637	6.0	2.0	
2	1.0	12800	0	905	9.0	1.0	
3	1.0	14000	0	9218	9.0	2.0	
4	0.0	15000	0	10891	10.0	2.0	
...	...	...	...	...	...	...	
42530	2.0	10000	0	2227	3.0	0.0	
42531	5.0	9600	0	5131	5.0	1.0	
42532	3.0	1200	2	1376	3.0	1.0	
42533	1.0	8000	0	2007	4.0	2.0	
42534	5.0	16000	1	10547	0.0	0.0	

42535 rows × 19 columns

In [46]:
```
df.corr()
```

Out[46]:

	inq_last_6mths	loan_amnt	number_bc_gt_75	revol_bal	emp_length	verificati
inq_last_6mths	1.000000	-0.013952	0.084017	0.012169	0.001607	
loan_amnt	-0.013952	1.000000	0.070236	0.254293	-0.055575	
number_bc_gt_75	0.084017	0.070236	1.000000	0.144233	-0.005139	
revol_bal	0.012169	0.254293	0.144233	1.000000	-0.043035	
emp_length	0.001607	-0.055575	-0.005139	-0.043035	1.000000	
verification_status	-0.024393	0.411117	0.047826	0.100553	-0.034941	
home_ownership	-0.048468	-0.179251	0.018446	-0.196312	0.069409	
dti	0.025530	0.065112	0.184431	0.190727	-0.018178	
pub_rec	0.070058	-0.051719	0.049640	-0.047969	-0.011900	
revol_utilization	-0.010177	0.056465	0.609345	0.193796	0.000670	
open_acc	0.082372	0.176270	0.006831	0.257532	-0.039724	
delinq_2yrs	0.025464	-0.032657	-0.025999	-0.042985	-0.021881	
annual_inc	0.012749	0.276133	0.016135	0.283604	-0.045545	
interest_rate	0.221281	0.292346	0.267940	0.081883	-0.009960	
installment	-0.003991	0.930869	0.079882	0.264837	-0.046341	

	inq_last_6mths	loan_amnt	number_bc_gt_75	revol_bal	emp_length	verificati
month_since_oldest_tl	-0.144883	0.111075	-0.086800	0.138316	-0.061669	
fico_score	-0.260250	0.092033	-0.356491	-0.037425	-0.015109	
loan_status	-0.529179	-0.032195	-0.237932	-0.088581	-0.000956	
gender	-0.009149	0.002800	-0.007050	-0.002862	0.000509	

In [47]:
```
df.head(10)
```

Out[47]:

	inq_last_6mths	loan_amnt	number_bc_gt_75	revol_bal	emp_length	verification_status	home_ow
0	3.0	5150	2	15187	0.0	1.0	
1	2.0	20000	0	15637	6.0	2.0	
2	1.0	12800	0	905	9.0	1.0	
3	1.0	14000	0	9218	9.0	2.0	
4	0.0	15000	0	10891	10.0	2.0	
5	0.0	12000	0	22385	8.0	0.0	
6	3.0	30000	0	2174	1.0	2.0	
7	1.0	2800	0	812	5.0	1.0	
8	0.0	2700	1	884	2.0	2.0	
9	0.0	6600	1	7362	5.0	0.0	

In [48]:
```
df.shape
```

Out[48]:
```
(42535, 19)
```

In [ ]:
```

```

In [38]:
```
z=pd.DataFrame(df)
```

In [39]:
```
z['loan_amnt']=z['loan_amnt'].astype(float)
```

In [40]:
```
z['number_bc_gt_75']=z['number_bc_gt_75'].astype(float)
```

In [41]:
```
z['revol_bal']=z['revol_bal'].astype(float)
```

In [49]:
```
df
```

Out[49]:

	inq_last_6mths	loan_amnt	number_bc_gt_75	revol_bal	emp_length	verification_status	hom

	inq_last_6mths	loan_amnt	number_bc_gt_75	revol_bal	emp_length	verification_status	hom
0	3.0	5150	2	15187	0.0	1.0	
1	2.0	20000	0	15637	6.0	2.0	
2	1.0	12800	0	905	9.0	1.0	
3	1.0	14000	0	9218	9.0	2.0	
4	0.0	15000	0	10891	10.0	2.0	
...	...	...	...	...	...	...	
42530	2.0	10000	0	2227	3.0	0.0	
42531	5.0	9600	0	5131	5.0	1.0	
42532	3.0	1200	2	1376	3.0	1.0	
42533	1.0	8000	0	2007	4.0	2.0	
42534	5.0	16000	1	10547	0.0	0.0	

42535 rows × 19 columns

In [50]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42535 entries, 0 to 42534
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   inq_last_6mths         42535 non-null  float64
 1   loan_amnt              42535 non-null  int64  
 2   number_bc_gt_75        42535 non-null  int64  
 3   revol_bal              42535 non-null  int64  
 4   emp_length             42535 non-null  float64
 5   verification_status    42535 non-null  float64
 6   home_ownership         42535 non-null  float64
 7   dti                    42535 non-null  float64
 8   pub_rec                42535 non-null  float64
 9   revol_utilization      42535 non-null  float64
 10  open_acc               42535 non-null  float64
 11  delinq_2yrs            42535 non-null  float64
 12  annual_inc             42535 non-null  float64
 13  interest_rate          42535 non-null  float64
 14  installment            42535 non-null  float64
 15  month_since_oldest_tl  42535 non-null  float64
 16  fico_score             42535 non-null  int64  
 17  loan_status            42535 non-null  int64  
 18  gender                 42535 non-null  float64
dtypes: float64(14), int64(5)
memory usage: 6.2 MB
```

In [51]:
```
x=df.drop(labels=['loan_status'],axis=1)
y=df['loan_status']
```

In [52]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

In [53]:
```
x_train.shape,x_test.shape
```

Out[53]: ((29774, 18), (12761, 18))

In [54]:
```
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn import metrics
```

In [55]:
```
clf=DecisionTreeClassifier(criterion="entropy",max_depth=3)
```

In [56]:
```
clf=clf.fit(x_train,y_train)
```

In [57]:
```
y_pred=clf.predict(x_test)
y_pred
```

Out[57]: array([1, 0, 1, ..., 1, 1, 1], dtype=int64)

In [58]:
```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.9242222396363922

In [59]:
```
x.head()
```

Out[59]:

	inq_last_6mths	loan_amnt	number_bc_gt_75	revol_bal	emp_length	verification_status	home_ow
0	3.0	5150	2	15187	0.0	1.0	
1	2.0	20000	0	15637	6.0	2.0	
2	1.0	12800	0	905	9.0	1.0	
3	1.0	14000	0	9218	9.0	2.0	
4	0.0	15000	0	10891	10.0	2.0	

In [60]:
```
y.head()
```

Out[60]:
```
0    0
1    1
2    1
3    1
4    0
Name: loan_status, dtype: int64
```

In [61]:
```
x.dtypes
```

Out[61]:
```
inq_last_6mths          float64
loan_amnt                 int64
number_bc_gt_75           int64
revol_bal                 int64
emp_length              float64
verification_status     float64
```

```
home_ownership           float64
dti                      float64
pub_rec                  float64
revol_utilization        float64
open_acc                 float64
delinq_2yrs              float64
annual_inc               float64
interest_rate            float64
installment              float64
month_since_oldest_tl    float64
fico_score                 int64
gender                   float64
dtype: object
```

In [62]:
```
clf=DecisionTreeClassifier(random_state=42)
model=clf.fit(x,y)
```

In [63]:
```
from sklearn import tree
from sklearn.tree import *
```

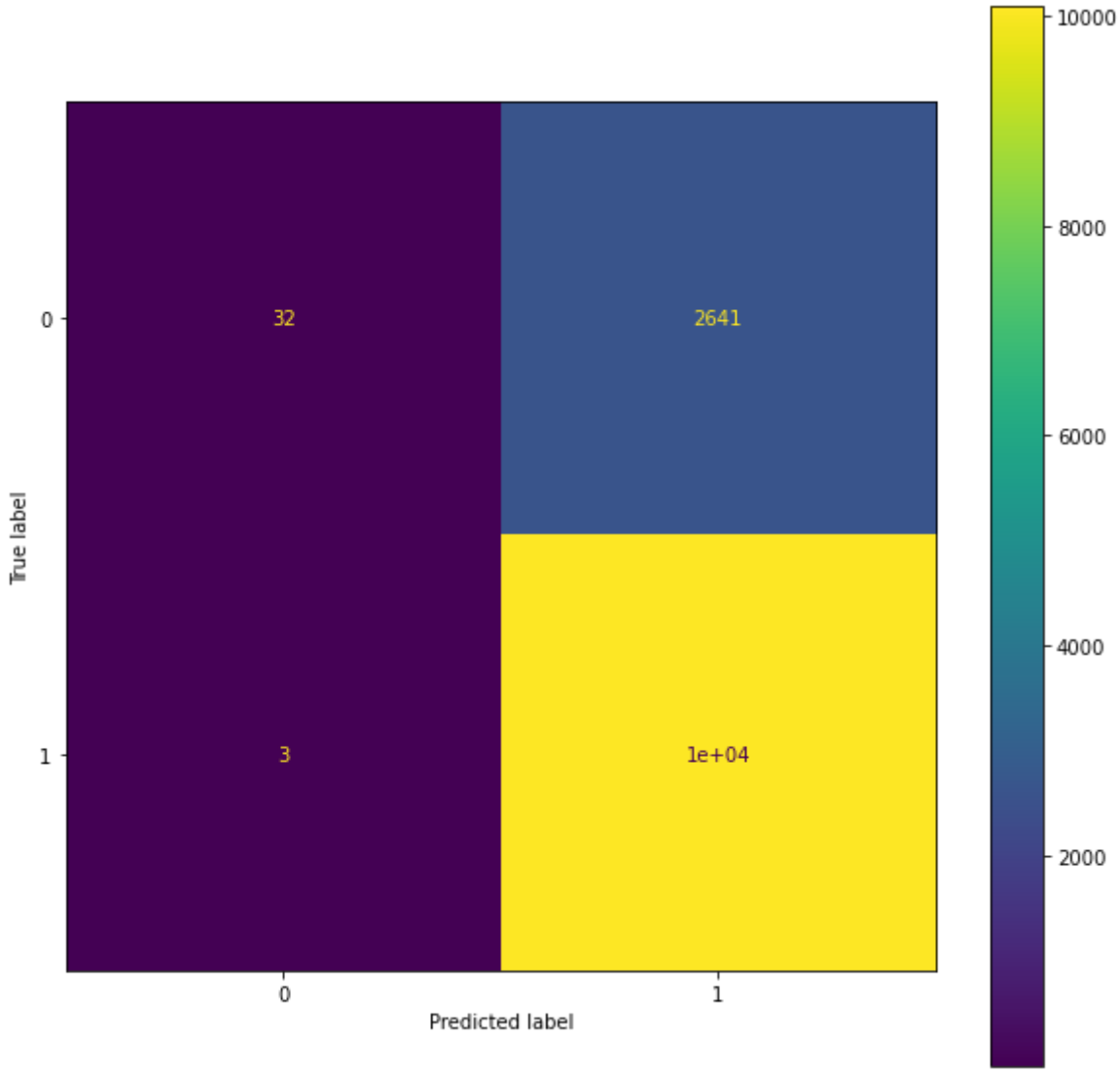
In [ ]:
```
text_representation=tree.export_text(clf)
text_representation
```

In [ ]:
```
plt.figure(figsize=(40,40))
_=tree.plot_tree(clf,feature_names=x_test.columns,filled=True)
```

In [ ]:
```
plt.figure(figsize=(30,25))
_=plot_tree(clf,
          filled=True,
          rounded=True,
          feature_names=x_train.columns)
```

In [111…
```
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

In [116…
```
clf = SVC(random_state=0)
clf.fit(x_train, y_train)
SVC(random_state=0)
plot_confusion_matrix(clf, x_test, y_test)
plt.show()
```



In [66]:
```
y
```

Out[66]:
```
0        0
1        1
2        1
3        1
4        0
        ..
42530    0
42531    0
42532    0
42533    0
42534    0
Name: loan_status, Length: 42535, dtype: int64
```

In [58]:
```
x_encoded=pd.get_dummies(x,columns=['emp_length','verification_status','home_ownersh
x_encoded.head()
```

Out[58]:

	inq_last_6mths	loan_amnt	number_bc_gt_75	revol_bal	dti	pub_rec	revol_utilization	open_ac
0	3.0	5150	2	15187	17.59	1.0	0.873	15.
1	2.0	20000	0	15637	7.80	0.0	0.354	6.
2	1.0	12800	0	905	14.22	0.0	0.754	7.
3	1.0	14000	0	9218	20.35	0.0	0.357	16.

	inq_last_6mths	loan_amnt	number_bc_gt_75	revol_bal	dti	pub_rec	revol_utilization	open_ac
4	0.0	15000	0	10891	20.50	0.0	0.369	12.

5 rows × 34 columns

In [ ]:
```
y_not_zero_index=y>0
y[y_not_zero_index]=1
y.unique()
```

In [57]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_encoded,y,random_state=42)
```

In [68]:
```
x_train.shape,x_test.shape
```

Out[68]:
```
((29774, 18), (12761, 18))
```

In [69]:
```
clf_dt=DecisionTreeClassifier(random_state=42)
clf_dt=clf_dt.fit(x_train,y_train)
```

In [70]:
```
from sklearn.tree import plot_tree
```

In [71]:
```
import xgboost as xgb
from sklearn.metrics import mean_squared_error
```

In [72]:
```
data_dmatrix=xgb.DMatrix(data=x,label=y)
```

In [73]:
```
xg_reg=xgb.XGBRegressor(objective='reg:linear',colsample_bytree=0.3,learning_rate=0.
```

In [74]:
```
xg_reg.fit(x_train,y_train)
```

```
[16:30:21] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/sr
c/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squa
rederror.
```

Out[74]:
```
XGBRegressor(alpha=10, base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.3,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.1, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=5, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=10, n_jobs=0,
             num_parallel_tree=1, objective='reg:linear', predictor='auto',
             random_state=0, ...)
```

In [75]:
```
preds=xg_reg.predict(x_test)
```

In [76]:
```
rmse=np.sqrt(mean_squared_error(y_test,preds))
print('RMSE:%f'%(rmse))
```

```
RMSE:0.342453
```

In [77]:
```
params = {"objective":"reg:linear",'colsample_bytree': 0.3,'learning_rate': 0.1,
                'max_depth': 5, 'alpha': 10}
```

In [78]:
```
cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=3,
                    num_boost_round=50,early_stopping_rounds=10,metrics="rmse", as_p
```

```
[16:30:30] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/sr
c/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squa
rederror.
[16:30:30] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/sr
c/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squa
rederror.
[16:30:30] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/sr
c/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squa
rederror.
```

In [79]:
```
cv_results.head()
```

Out[79]:

	train-rmse-mean	train-rmse-std	test-rmse-mean	test-rmse-std
0	0.470059	0.000165	0.470096	0.000134
1	0.451301	0.005368	0.451301	0.005383
2	0.429229	0.004847	0.429470	0.004719
3	0.413025	0.010414	0.413388	0.010128
4	0.397361	0.013318	0.397652	0.013210

In [80]:
```
print((cv_results["test-rmse-mean"]).tail(1))
```

```
49    0.232196
Name: test-rmse-mean, dtype: float64
```

In [81]:
```
xg_reg = xgb.train(params=params, dtrain=data_dmatrix, num_boost_round=10)
```

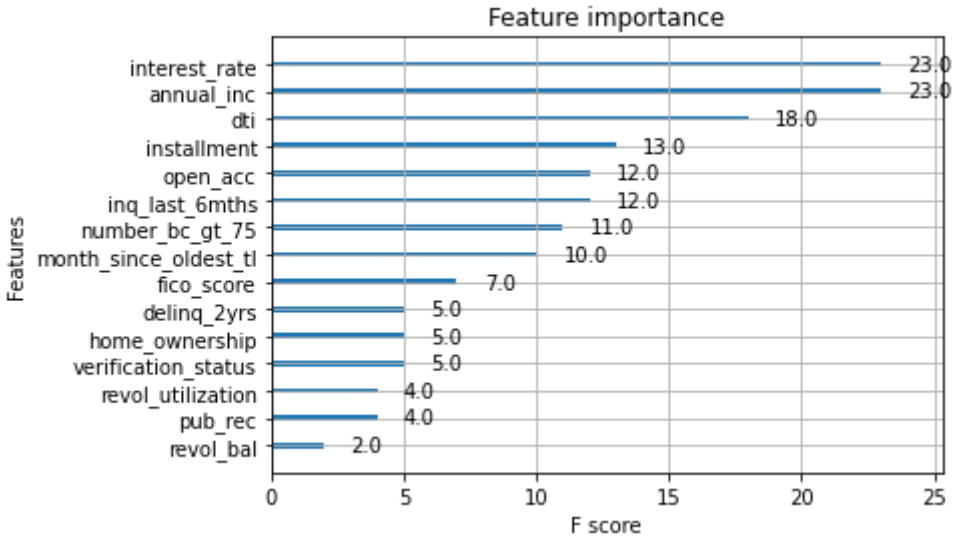
```
[16:30:37] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/sr
c/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squa
rederror.
```

In [82]:
```
import graphviz
```

In [ ]:
```
import matplotlib.pyplot as plt

xgb.plot_tree(xg_reg,num_trees=0)
plt.rcParams['figure.figsize'] = [50, 10]
plt.show()
```

In [84]:
```
xgb.plot_importance(xg_reg)
plt.rcParams['figure.figsize'] = [10, 10]
plt.show()
```



In [85]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [86]:
```
classifier_rf = RandomForestClassifier(random_state=42, n_jobs=-1, max_depth=5,
                                       n_estimators=100, oob_score=True)
```

In [89]:
```
%time
classifier_rf.fit(x_train, y_train)
```

```
Wall time: 0 ns
```

Out[89]:
```
RandomForestClassifier(max_depth=5, n_jobs=-1, oob_score=True, random_state=42)
```

In [90]:
```
# checking the oob score
classifier_rf.oob_score_
```

Out[90]:
```
0.9359844159333647
```

In [91]:
```
rf = RandomForestClassifier(random_state=42, n_jobs=-1)
```

In [92]:
```
params = {
    'max_depth': [2,3,5,10,20],
    'min_samples_leaf': [5,10,20,50,100,200],
    'n_estimators': [10,25,30,50,100,200]
}
```

In [93]:
```
from sklearn.model_selection import GridSearchCV
```

In [94]:
```
# Instantiate the grid search model
grid_search = GridSearchCV(estimator=rf,
                           param_grid=params,
                           cv = 4,
                           n_jobs=-1, verbose=1, scoring="accuracy")
```

In [96]:
```
%%time
grid_search.fit(x_train, y_train)
```

```
Fitting 4 folds for each of 180 candidates, totalling 720 fits
Wall time: 5min 45s
```

Out[96]:
```
GridSearchCV(cv=4, estimator=RandomForestClassifier(n_jobs=-1, random_state=42),
             n_jobs=-1,
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy', verbose=1)
```

In [97]:
```
grid_search.best_score_
```

Out[97]:
```
0.9544233018513694
```

In [98]:
```
rf_best = grid_search.best_estimator_
rf_best
```

Out[98]:
```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=30,
                       n_jobs=-1, random_state=42)
```

In [ ]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5], feature_names = x.columns,class_names=['good loan'
```

In [102…
```
#importing and fitting KNN
from sklearn.neighbors import KNeighborsClassifier
```

In [104…
```
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
```

Out[104…
```
KNeighborsClassifier(n_neighbors=3)
```

In [105…
```
#predicting result using test dataset
pred=knn.predict(x_test)
```

In [106…
```
from sklearn.metrics import accuracy_score
```

In [107…
```
accuracy_score(pred,y_test)
```

Out[107…
```
0.7388919363686232
```

In [110…
```
confusion_matrix(y_test,pred)
```

Out[110…
```
array([[ 403, 2270],
       [1062, 9026]], dtype=int64)
```

In [ ]:
```

```