In [2]:
```
import numpy as np
import pandas as pd
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
dat = pd.read_csv("C:\\Users\\ymx19\\Desktop\\test\\Lending_Club_v2.csv")
```

```
C:\Users\ymx19\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3071: DtypeWarning: Columns (47) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [3]:
```
# 1. EDA
dat.shape
dat.head(10)
dat.info()
dat.describe()
dat.loan_amnt.value_counts()
```

Out[3]:
```
(42535, 144)
```

Out[3]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	orig_projected_a
0	1	NaN	5000	5000	4975.0	36 months	0.11	162.87	B	B2	...	
1	2	NaN	2500	2500	2500.0	60 months	0.15	59.83	C	C4	...	
2	3	NaN	2400	2400	2400.0	36 months	0.16	84.33	C	C5	...	
3	4	NaN	10000	10000	10000.0	36 months	0.13	339.31	C	C1	...	
4	5	NaN	3000	3000	3000.0	60 months	0.13	67.79	B	B5	...	
5	6	NaN	5000	5000	5000.0	36 months	0.08	156.46	A	A4	...	
6	7	NaN	7000	7000	7000.0	60 months	0.16	170.08	C	C5	...	
7	8	NaN	3000	3000	3000.0	36 months	0.19	109.43	E	E1	...	
8	9	NaN	5600	5600	5600.0	60 months	0.21	152.39	F	F2	...	
9	10	NaN	5375	5375	5350.0	60 months	0.13	121.45	B	B5	...	

10 rows × 144 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42535 entries, 0 to 42534
Columns: 144 entries, id to settlement_term
dtypes: float64(111), int64(7), object(26)
memory usage: 46.7+ MB
```

Out[3]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	url	
count	42535.000000	0.0	42535.000000	42535.000000	42535.000000	42535.000000	42535.000000	4.253100e+04	0.0	4
mean	21267.000000	NaN	11089.722581	10821.585753	10139.938785	0.121564	322.623903	6.913656e+04	NaN	
std	12278.941187	NaN	7410.938391	7146.914675	7131.598014	0.037024	208.927216	6.409635e+04	NaN	
min	1.000000	NaN	500.000000	500.000000	0.000000	0.050000	15.670000	1.896000e+03	NaN	
25%	10634.500000	NaN	5200.000000	5000.000000	4950.000000	0.100000	165.520000	4.000000e+04	NaN	
50%	21268.000000	NaN	9700.000000	9600.000000	8500.000000	0.120000	277.690000	5.900000e+04	NaN	
75%	31901.500000	NaN	15000.000000	15000.000000	14000.000000	0.150000	428.180000	8.250000e+04	NaN	
max	42535.000000	NaN	35000.000000	35000.000000	35000.000000	0.250000	1305.190000	6.000000e+06	NaN	

8 rows × 118 columns

Out[3]:
```
10000    3016
12000    2439
5000     2260
6000     2037
15000    2012
         ...
30500       1
10350       1
12525       1
16675       1
28750       1
Name: loan_amnt, Length: 898, dtype: int64
```

In [4]:
```
# 2 feature reduction
data = dat.dropna(axis = 'columns', how='all')
data.head(3)
```

Out[4]:

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	...	pub_rec_bankruptc
0	1	5000	5000	4975.0	36 months	0.11	162.87	B	B2	NaN	...	
1	2	2500	2500	2500.0	60 months	0.15	59.83	C	C4	Ryder	...	
2	3	2400	2400	2400.0	36 months	0.16	84.33	C	C5	NaN	...	

3 rows × 63 columns

In [5]:
```
# 3 Transform any characteristics or categorical variables into numeric
x = data[['term','home_ownership','sub_grade','emp_length','verification_status','loan_amnt','funded_amnt','funded_amnt_inv','int_rate','installment','annual_inc','revol_bal','revol_util','total_pymnt','total_rec_int']]
X = pd.get_dummies(x)
X = X.rename(columns = {'emp_length_< 1 year': 'emp_length less than a year'}, inplace = False)
```

In [15]:
```
# 4  Feature selection
y = data[['loan_status']]
y = y.replace(to_replace={'Fully Paid':'Does not meet the credit policy. Status:Fully Paid','Charged Off':'Does not meet the credit policy. Status:Charged Off'}, value=[0,0,1,1])
```

In [7]:
```
# 5
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2,random_state = 12)
```

In [8]:
```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [100, 200, 300, 400],
    'learning_rate': [0.2, 0.4, 0.6, 0.8, 1, 1.2]
}
clf_ada = XGBClassifier()
rf_Grid = GridSearchCV(clf_ada, param_grid, cv = 5, scoring = 'roc_auc',refit = True, n_jobs=-1,
verbose = 5)
rf_Grid.fit(X_train, y_train.values.ravel())
sorted(rf_Grid.cv_results_.keys())
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done  40 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 114 out of 120 | elapsed:  3.3min remaining:   10.4s
[Parallel(n_jobs=-1)]: Done 120 out of 120 | elapsed:  3.4min finished
C:\Users\ymx19\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[09:45:38] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[8]:
```
GridSearchCV(cv=5,
             estimator=XGBClassifier(base_score=None, booster=None,
                                     colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None, gamma=None,
                                     gpu_id=None, importance_type='gain',
                                     interaction_constraints=None,
                                     learning_rate=None, max_delta_step=None,
                                     max_depth=None, min_child_weight=None,
                                     missing=nan, monotone_constraints=None,
                                     n_estimators=100, n_jobs=None,
                                     num_parallel_tree=None, random_state=None,
                                     reg_alpha=None, reg_lambda=None,
                                     scale_pos_weight=None, subsample=None,
                                     tree_method=None, validate_parameters=None,
                                     verbosity=None),
             n_jobs=-1,
             param_grid={'learning_rate': [0.2, 0.4, 0.6, 0.8, 1, 1.2],
                         'n_estimators': [100, 200, 300, 400]},
             scoring='roc_auc', verbose=5)
```

Out[8]:
```
['mean_fit_time',
 'mean_score_time',
 'mean_test_score',
 'param_learning_rate',
 'param_n_estimators',
 'params',
 'rank_test_score',
 'split0_test_score',
 'split1_test_score',
 'split2_test_score',
 'split3_test_score',
 'split4_test_score',
 'std_fit_time',
 'std_score_time',
 'std_test_score']
```

In [9]:
```
best_estimator_model = rf_Grid.best_estimator_
rf_Grid_prob_pred = best_estimator_model.predict_proba(X_test)[:, 1]
rf_Grid_prob_pred

rf_Grid.predict = best_estimator_model.predict(X_test)
rf_Grid.predict
```

```
C:\Users\ymx19\anaconda3\lib\site-packages\xgboost\data.py:112: UserWarning: Use subset (sliced data) of np.ndarray is not recommended because it will generate extra copies and increase memory consumption
  warnings.warn(
```

Out[9]:
```
array([0.00694919, 0.99975973, 0.006777  , ..., 0.00327836, 0.99990547,
       0.00236576], dtype=float32)
```

Out[9]:
```
array([0, 1, 0, ..., 0, 1, 0], dtype=int64)
```

In [12]:
```
# 6
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# confusion matrix (test model)
confusion_matrix(y_test, rf_Grid.predict)
# classification report (test model)
target_names = ['0', '1']
print(classification_report(y_test, rf_Grid.predict, target_names=target_names))
# roc_auc_score
from sklearn.metrics import roc_auc_score
print('test model roc_auc_score : ''',roc_auc_score(y_test,rf_Grid_prob_pred))
```

Out[12]:
```
array([[7203,    0],
       [ 125, 1179]], dtype=int64)
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      7203
           1       1.00      0.90      0.95      1304

    accuracy                           0.99      8507
   macro avg       0.99      0.95      0.97      8507
weighted avg       0.99      0.99      0.98      8507

test model roc_auc_score :  0.9828451037357474
```

In [13]:
```
rf_Grid_prob_pred_train = best_estimator_model.predict_proba(X_train)[:, 1]
rf_Grid_prob_pred_train

rf_Grid.predict_train = best_estimator_model.predict(X_train)
rf_Grid.predict_train
# confusion matrix (train)
confusion_matrix(y_train, rf_Grid.predict_train)
# classification report (train)
target_names = ['0', '1']
print(classification_report(y_train, rf_Grid.predict_train, target_names=target_names))
# roc_auc_score
from sklearn.metrics import roc_auc_score
print('train roc_auc_score : ''',roc_auc_score(y_train,rf_Grid_prob_pred_train))
```

```
C:\Users\ymx19\anaconda3\lib\site-packages\xgboost\data.py:112: UserWarning: Use subset (sliced data) of np.ndarray is not recommended because it will generate extra copies and increase memory consumption
  warnings.warn(
```

Out[13]:
```
array([0.00304389, 0.00722173, 0.00115316, ..., 0.00362942, 0.00497114,
       0.0137087 ], dtype=float32)
```

Out[13]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

Out[13]:
```
array([[28901,     0],
       [  181,  4946]], dtype=int64)
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     28901
           1       1.00      0.96      0.98      5127

    accuracy                           0.99     34028
   macro avg       1.00      0.98      0.99     34028
weighted avg       0.99      0.99      0.99     34028

train roc_auc_score :  0.9998871405310068
```

In [ ]:
```

```