Plan for Data Cleaning/Pre-processing

1. Removed the features which has more than 75% null values

```
In [44]:
#Removing features which has got null values above 75%
nulls = pd.DataFrame(round(df.isnull().sum()/len(df.index)*100,2),columns=['null_percent'])
drop_cols = nulls[nulls['null_percent']>75.0].index
df.drop(drop_cols, axis=1, inplace=True)
```

2. Check missing value count and percent

```
In [48]:
# Check missing values count and percent
total= df.isnull().sum().sort_values(ascending=False)
percent= (df.isnull().sum()/df.isnull().count()).sort_values(ascending=False)*100
missing_data= pd.concat([total, percent],axis=1, keys=["Total", "Percent"])
missing_data.head(54)
```

Out[48]:

	Total	Percent
mths_since_last_delinq	454312	51.197065
next_pymnt_d	252971	28.507661
tot_cur_bal	70276	7.919502
total_rev_hi_lim	70276	7.919502
tot_coll_amt	70276	7.919502
emp_title	51462	5.799326
emp_length	44825	5.051393
last_pymnt_d	17659	1.990018
revol_util	502	0.056571
title	152	0.017129
collections_12_mths_ex_med	145	0.016340
last_credit_pull_d	53	0.005973
total_acc	29	0.003268
delinq_2yrs	29	0.003268

3. Calculating the NaN values for all the columns

```
In [56]:
#Let's see the data shape and NaN values
print(df.shape)
print(type(df))
print(df.head())
print(df.isnull().sum())
print(df.isnull().sum().value_counts())
```

```
(887379, 16)
<class 'pandas.core.frame.DataFrame'>
   loan_amnt  funded_amnt_inv        term  int_rate sub_grade home_ownership  \
0     5000.0           4975.0   36 months     10.65        B2           RENT
1     2500.0           2500.0   60 months     15.27        C4           RENT
2     2400.0           2400.0   36 months     15.96        C5           RENT
3    10000.0          10000.0   36 months     13.49        C1           RENT
4     3000.0           3000.0   60 months     12.69        B5           RENT

   annual_inc  loan_status         purpose addr_state    dti  delinq_2yrs  \
0     24000.0   Fully Paid     credit_card         AZ  27.65          0.0
1     30000.0  Charged Off             car         GA   1.00          0.0
2     12252.0   Fully Paid  small_business         IL   8.72          0.0
3     49200.0   Fully Paid           other         CA  20.00          0.0
4     80000.0      Current           other         OR  17.94          0.0

   inq_last_6mths  total_acc   total_pymnt  total_pymnt_inv
0             1.0        9.0   5861.071414          5831.78
1             5.0        4.0   1008.710000          1008.71
2             2.0       10.0   3003.653644          3003.65
3             1.0       37.0  12226.302212         12226.30
4             0.0       38.0   3242.170000          3242.17
loan_amnt          0
funded_amnt_inv    0
```

4. Filling NaN values with Mean

In [57]:

```
#Filling the numeric columns with missing values by mean
for cols in df.columns:
    if df[cols].isnull().sum() != 0:
        df[cols].fillna((df[cols].mean()), inplace=True)

df.isnull().sum()
```

Out[57]:

```
loan_amnt          0
funded_amnt_inv    0
term               0
int_rate           0
sub_grade          0
home_ownership     0
annual_inc         0
loan_status        0
purpose            0
addr_state         0
dti                0
delinq_2yrs        0
inq_last_6mths     0
total_acc          0
total_pymnt        0
total_pymnt_inv    0
dtype: int64
```

Feature Engineering

Feature Engineering can simply be defined as the process of creating new features from the existing features in a dataset.

The performance of a predictive model is heavily dependent on the quality of the features in the dataset used to train that model. If you are able to create new features which help in providing more information to the model about the target variable, it's performance will go up. Hence, when we don't have enough quality features in our dataset, we have to lean on feature engineering.

Manual Feature Engineering

Manual feature engineering can be a tedious process (which is why we use automated feature engineering with featuretools!) and often relies on domain expertise. Since we have limited domain knowledge of loans and what makes a person likely to default, we will instead concentrate of getting as much info as possible into the final training dataframe. The idea is that the model will then pick up on which features are important rather than us having to decide that. Basically, our approach is to make as many features as possible and then give them all to the model to use! Later, we can perform feature reduction using the feature importances from the model.

The process of manual feature engineering will involve plenty of Pandas code, a little patience, and a lot of great practice manipulation data. Even though automated feature engineering tools are starting to be made available, feature engineering will still have to be done using plenty of data wrangling for a little longer while.

We initially had 74 columns in the Lending Club dataset, we narrowed it down first to 34 columns by eliminating the ones with more than 75% null values, then we understood the dataset properly with the help of LCDataDictionary and then removed further columns manually on the basis of our understanding.

Feature Tools

Featuretools is an open source library for performing automated feature engineering. It is a great tool designed to fast-forward the feature generation process, thereby giving more time to focus on other aspects of machine learning model building. In other words, it makes your data "machine learning ready".

Before taking Featuretools for a spin, there are three major components of the package that we should be aware of:

- Entities

  Creating entity set for feature tools

```
In [67]: es = ft.EntitySet()
         es = es.entity_from_dataframe(entity_id="loandata",
                                       dataframe=loandata,
                                       index="id")
```

```
In [68]: es["loandata"].variables
```

```
Out[68]: [<Variable: id (dtype = index)>,
          <Variable: member_id (dtype = numeric)>,
          <Variable: loan_amnt (dtype = numeric)>,
          <Variable: funded_amnt_inv (dtype = numeric)>,
          <Variable: term (dtype = categorical)>,
          <Variable: int_rate (dtype = numeric)>,
          <Variable: installment (dtype = numeric)>,
          <Variable: grade (dtype = categorical)>,
          <Variable: sub_grade (dtype = categorical)>,
          <Variable: home_ownership (dtype = categorical)>,
          <Variable: annual_inc (dtype = numeric)>,
          <Variable: verification_status (dtype = categorical)>,
          <Variable: issue_d (dtype: datetime, format: None)>,
          <Variable: loan_status (dtype = categorical)>,
          <Variable: purpose (dtype = categorical)>,
          <Variable: addr_state (dtype = categorical)>,
          <Variable: dti (dtype = numeric)>,
          <Variable: delinq_2yrs (dtype = numeric)>,
          <Variable: inq_last_6mths (dtype = numeric)>,
          <Variable: total_acc (dtype = numeric)>,
          <Variable: initial_list_status (dtype = categorical)>,
          <Variable: out_prncp (dtype = numeric)>,
          <Variable: out_prncp_inv (dtype = numeric)>,
          <Variable: total_pymnt (dtype = numeric)>,
```

- Deep Feature Synthesis (DFS)

  Creating feature matrix

```
In [69]: feature_matrix, feature_names = ft.dfs(entityset=es,
             target_entity = 'loandata',
             max_depth = 3,
             verbose = 3,
             n_jobs = 1)
```

```
Built 36 features
Elapsed: 00:07 | Remaining: 00:00 | Progress: 100%|██████████| Calculated: 11/11 chunks
```

```
In [70]: feature_matrix.head()
```

Out[70]:

id	member_id	loan_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	home_ownership	annual_inc	...	recoveries	collection_
54734	80364	25000.0	19080.057198	36 months	11.89	829.10	B	B4	RENT	85000.0	...	0.00	
55521	107577	1000.0	0.000000	36 months	16.08	35.20	F	F2	RENT	30000.0	...	0.00	
55742	114426	7000.0	672.803839	36 months	10.71	228.22	B	B5	RENT	65000.0	...	0.00	
56413	129814	7000.0	0.007494	36 months	16.08	246.38	F	F2	MORTGAGE	189500.0	...	0.25	
56705	133361	11000.0	11000.000000	36 months	9.99	354.89	B	B3	MORTGAGE	33500.0	...	0.00	

5 rows × 36 columns

- Feature primitives