Data Preparation

Import Libraries

In [ ]:
```
import numpy as np
import pandas as pd
```

Import Data

(A) The dataset contains all available data for more than 800,000 consumer loans issued from 2007 to 2015 (B) We build Expected Loss models (C) we built the Probability of Default (PD) model

In [ ]:
```
loan_data_backup = pd.read_csv('loan_data_2007_2014.csv')
```

In [ ]:
```
loan_data = loan_data_backup.copy()
```

Explore Data

In [ ]:
```
loan_data
```

In [ ]:
```
pd.options.display.max_columns = None
```

In [ ]:
```
loan_data
```

In [ ]:
```
loan_data.head()
```

In [ ]:
```
loan_data.tail()
```

In [ ]:
```
loan_data.columns.values
```

In [ ]:
```
loan_data.info()
```

General Preprocessing

Preprocessing few continuous variables

In [ ]:
```
loan_data['emp_length'].unique()
```

In [ ]:
```
loan_data['emp_length_int'] = loan_data['emp_length'].str.replace('\+ years',
loan_data['emp_length_int'] = loan_data['emp_length_int'].str.replace('< 1 yea
loan_data['emp_length_int'] = loan_data['emp_length_int'].str.replace('n/a',
```

```
loan_data['emp_length_int'] = loan_data['emp_length_int'].str.replace(' years'
loan_data['emp_length_int'] = loan_data['emp_length_int'].str.replace(' year',
```

In [ ]:
```
type(loan_data['emp_length_int'][0])
```

In [ ]:
```
loan_data['emp_length_int'] = pd.to_numeric(loan_data['emp_length_int'])
```

In [ ]:
```
type(loan_data['emp_length_int'][0])
```

In [ ]:
```
loan_data['earliest_cr_line']
```

In [ ]:
```
loan_data['earliest_cr_line_date'] = pd.to_datetime(loan_data['earliest_cr_lin
```

In [ ]:
```
type(loan_data['earliest_cr_line_date'][0])
```

In [ ]:
```
pd.to_datetime('2017-12-01') - loan_data['earliest_cr_line_date']
```

In [ ]:
```
# Assume we are now in December 2017
loan_data['mths_since_earliest_cr_line'] = round(pd.to_numeric((pd.to_datetime
# We calculate the difference between two dates in months, turn it to numeric (
# We save the result in a new variable.
```

In [ ]:
```
loan_data['mths_since_earliest_cr_line'].describe()
# Dates from 1969 and before are not being converted well, i.e., they have bec
```

In [ ]:
```
loan_data.loc[: , ['earliest_cr_line', 'earliest_cr_line_date', 'mths_since_ea
```

In [ ]:
```
loan_data['mths_since_earliest_cr_line'][loan_data['mths_since_earliest_cr_lin
```

In [ ]:
```
min(loan_data['mths_since_earliest_cr_line'])
```

same operation for term column

In [ ]:
```
loan_data['term']
```

In [ ]:
```
loan_data['term'].describe()
```

In [ ]:
```
loan_data['term_int'] = loan_data['term'].str.replace(' months', '')
```

In [ ]:
```
loan_data['term_int']
```

In [ ]:
```
type(loan_data['term_int'][25])
```

In [ ]:
```
loan_data['term_int'] = pd.to_numeric(loan_data['term'].str.replace(' months',
loan_data['term_int']
```

In [ ]:
```
type(loan_data['term_int'][0])
```

In [ ]:
```
loan_data['issue_d']
```

In [ ]:
```
loan_data['issue_d_date'] = pd.to_datetime(loan_data['issue_d'], format = '%b-%
loan_data['mths_since_issue_d'] = round(pd.to_numeric((pd.to_datetime('2017-12-
loan_data['mths_since_issue_d'].describe()
```

Preprocessing few discrete variables

In [ ]:
```
loan_data.info()
```

In [ ]:
```
pd.get_dummies(loan_data['grade'], prefix = 'grade', prefix_sep = ':')
# Create dummy variables from a variable.
```

In [ ]:
```
loan_data_dummies = [pd.get_dummies(loan_data['grade'], prefix = 'grade', prefi
                     pd.get_dummies(loan_data['sub_grade'], prefix = 'sub_grad
                     pd.get_dummies(loan_data['home_ownership'], prefix = 'hom
                     pd.get_dummies(loan_data['verification_status'], prefix =
                     pd.get_dummies(loan_data['loan_status'], prefix = 'loan_s
                     pd.get_dummies(loan_data['purpose'], prefix = 'purpose', |
                     pd.get_dummies(loan_data['addr_state'], prefix = 'addr_st
                     pd.get_dummies(loan_data['initial_list_status'], prefix =
```

In [ ]:
```
loan_data_dummies = pd.concat(loan_data_dummies, axis = 1)
```

In [ ]:
```
type(loan_data_dummies)
```

In [ ]:
```
loan_data = pd.concat([loan_data, loan_data_dummies], axis = 1)
```

In [ ]:
```
loan_data.columns.values
```

Check for missing values and clean

In [ ]:
```
loan_data.isnull()
```

In [ ]:
```
pd.options.display.max_rows = None
loan_data.isnull().sum()
```

In [ ]:
```
pd.options.display.max_rows = 100
```

In [ ]:
```
loan_data['total_rev_hi_lim'].fillna(loan_data['funded_amnt'], inplace=True)
```

In [ ]:
```
loan_data['total_rev_hi_lim'].isnull().sum()
```

In [ ]:
```
loan_data['annual_inc'].fillna(loan_data['annual_inc'].mean(), inplace=True)
# We fill the missing values with the mean value of the non-missing values.
```

In [ ]:
```
loan_data['mths_since_earliest_cr_line'].fillna(0, inplace=True)
loan_data['acc_now_delinq'].fillna(0, inplace=True)
loan_data['total_acc'].fillna(0, inplace=True)
loan_data['pub_rec'].fillna(0, inplace=True)
loan_data['open_acc'].fillna(0, inplace=True)
loan_data['inq_last_6mths'].fillna(0, inplace=True)
loan_data['delinq_2yrs'].fillna(0, inplace=True)
```

```
loan_data['emp_length_int'].fillna(0, inplace=True)
# We fill the missing values with zeroes.
```

PD model

In [ ]:
```
loan_data['loan_status'].value_counts() / loan_data['loan_status'].count()
```

In [ ]:
```
loan_data['good_bad'] = np.where(loan_data['loan_status'].isin(['Charged Off',
                                                                'Does not meet the credi
                                                                'Late (31-120 days)']),
# We create a new variable that has the value of '0' if a condition is met, and
```

In [ ]:
```
loan_data['good_bad']
```

Splitting Data

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
train_test_split(loan_data.drop('good_bad', axis = 1), loan_data['good_bad'])
```

In [ ]:
```
loan_data_inputs_train, loan_data_inputs_test, loan_data_targets_train, loan_da
```

In [ ]:
```
loan_data_inputs_train.shape
```

In [ ]:
```
loan_data_targets_train.shape
```

In [ ]:
```
loan_data_inputs_test.shape
```

In [ ]:
```
loan_data_targets_test.shape
```

In [ ]:
```
loan_data_inputs_train, loan_data_inputs_test, loan_data_targets_train, loan_da
```

In [ ]:
```
loan_data_inputs_train.shape
```

In [ ]:
```
loan_data_targets_train.shape
```

In [ ]:
```
loan_data_inputs_test.shape
```

In [ ]:
```
loan_data_targets_test.shape
```

Data Preparation: calculation of Weight of evidence and Information Value

In [ ]:
```
#####
#df_inputs_prepr = loan_data_inputs_train
#df_targets_prepr = loan_data_targets_train
#####
df_inputs_prepr = loan_data_inputs_test
df_targets_prepr = loan_data_targets_test
```

In [ ]:
```
df_inputs_prepr['grade'].unique()
```

In [ ]:
```
df1 = pd.concat([df_inputs_prepr['grade'], df_targets_prepr], axis = 1)
df1.head()
```

In [ ]:
```
df1.groupby(df1.columns.values[0], as_index = False)[df1.columns.values[1]].co
```

In [ ]:
```
df1.groupby(df1.columns.values[0], as_index = False)[df1.columns.values[1]].me
```

In [ ]:
```
df1 = pd.concat([df1.groupby(df1.columns.values[0], as_index = False)[df1.colu
                 df1.groupby(df1.columns.values[0], as_index = False)[df1.colum
```

In [ ]:
```
df1
```

In [ ]:
```
df1 = df1.iloc[:, [0, 1, 3]]
df1
```

In [ ]:
```
df1.columns = [df1.columns.values[0], 'n_obs', 'prop_good']
df1
```

In [ ]:
```
df1['prop_n_obs'] = df1['n_obs'] / df1['n_obs'].sum()
df1
```

In [ ]:
```
df1['n_good'] = df1['prop_good'] * df1['n_obs']
df1['n_bad'] = (1 - df1['prop_good']) * df1['n_obs']
df1
```

In [ ]:
```
df1['prop_n_good'] = df1['n_good'] / df1['n_good'].sum()
df1['prop_n_bad'] = df1['n_bad'] / df1['n_bad'].sum()
df1
```

In [ ]:
```
df1['WoE'] = np.log(df1['prop_n_good'] / df1['prop_n_bad'])
df1
```

In [ ]:
```
df1 = df1.sort_values(['WoE'])
df1 = df1.reset_index(drop = True)
df1
```

In [ ]:
```
df1['diff_prop_good'] = df1['prop_good'].diff().abs()
df1['diff_WoE'] = df1['WoE'].diff().abs()
df1
```

In [ ]:
```
df1['IV'] = (df1['prop_n_good'] - df1['prop_n_bad']) * df1['WoE']
df1['IV'] = df1['IV'].sum()
df1
```

Preprocessing Discrete Variables: Automating Calculaions

In [ ]:
```
def woe_discrete(df, discrete_variabe_name, good_bad_variable_df):
    df = pd.concat([df[discrete_variabe_name], good_bad_variable_df], axis = 1
    df = pd.concat([df.groupby(df.columns.values[0], as_index = False)[df.colu
                    df.groupby(df.columns.values[0], as_index = False)[df.colu
    df = df.iloc[:, [0, 1, 3]]
    df.columns = [df.columns.values[0], 'n_obs', 'prop_good']
    df['prop_n_obs'] = df['n_obs'] / df['n_obs'].sum()
    df['n_good'] = df['prop_good'] * df['n_obs']
    df['n_bad'] = (1 - df['prop_good']) * df['n_obs']
    df['prop_n_good'] = df['n_good'] / df['n_good'].sum()
    df['prop_n_bad'] = df['n_bad'] / df['n_bad'].sum()
    df['WoE'] = np.log(df['prop_n_good'] / df['prop_n_bad'])
    df = df.sort_values(['WoE'])
    df = df.reset_index(drop = True)
    df['diff_prop_good'] = df['prop_good'].diff().abs()
    df['diff_WoE'] = df['WoE'].diff().abs()
    df['IV'] = (df['prop_n_good'] - df['prop_n_bad']) * df['WoE']
    df['IV'] = df['IV'].sum()
    return df
```

In [ ]:
```
df_temp = woe_discrete(df_inputs_prepr, 'grade', df_targets_prepr)
df_temp
```

Preprocessing Discrete Variables: Visualizing Results

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

In [ ]:
```
def plot_by_woe(df_WoE, rotation_of_x_axis_labels = 0):
    x = np.array(df_WoE.iloc[:, 0].apply(str))
    y = df_WoE['WoE']
    plt.figure(figsize=(18, 6))
    plt.plot(x, y, marker = 'o', linestyle = '--', color = 'k')
    plt.xlabel(df_WoE.columns[0])
    plt.ylabel('Weight of Evidence')
    plt.title(str('Weight of Evidence by ' + df_WoE.columns[0]))
    plt.xticks(rotation = rotation_of_x_axis_labels)
```

In [ ]:
```
plot_by_woe(df_temp)
```

Preprocessing Discrete Variables: Creating Dummy Variables, Part 1

In [ ]:
```
df_temp = woe_discrete(df_inputs_prepr, 'home_ownership', df_targets_prepr)
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

In [ ]:
```
df_inputs_prepr['home_ownership:RENT_OTHER_NONE_ANY'] = sum([df_inputs_prepr['
                                                        df_inputs_prepr['home_ow
```

Preprocessing Discrete Variables: Creating Dummy Variables, Part 2

In [ ]:
```
df_inputs_prepr['addr_state'].unique()
```

In [ ]:
```
df_temp = woe_discrete(df_inputs_prepr, 'addr_state', df_targets_prepr)
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

In [ ]:
```
if ['addr_state:ND'] in df_inputs_prepr.columns.values:
    pass
else:
    df_inputs_prepr['addr_state:ND'] = 0
```

In [ ]:
```
plot_by_woe(df_temp.iloc[2: -2, : ])
```

In [ ]:
```
plot_by_woe(df_temp.iloc[6: -6, : ])
```

In [ ]:
```
df_inputs_prepr['addr_state:ND_NE_IA_NV_FL_HI_AL'] = sum([df_inputs_prepr['add
                                                    df_inputs_prepr['addr_state:IA']
                                                    df_inputs_prepr['addr_state:FL']
                                                                df_inputs_prepr['add

df_inputs_prepr['addr_state:NM_VA'] = sum([df_inputs_prepr['addr_state:NM'], d

df_inputs_prepr['addr_state:OK_TN_MO_LA_MD_NC'] = sum([df_inputs_prepr['addr_s
                                                    df_inputs_prepr['addr_state:MO']
                                                    df_inputs_prepr['addr_state:MD']

df_inputs_prepr['addr_state:UT_KY_AZ_NJ'] = sum([df_inputs_prepr['addr_state:U
                                                    df_inputs_prepr['addr_state:AZ']

df_inputs_prepr['addr_state:AR_MI_PA_OH_MN'] = sum([df_inputs_prepr['addr_stat
                                                    df_inputs_prepr['addr_state:PA']
                                                    df_inputs_prepr['addr_state:MN']]

df_inputs_prepr['addr_state:RI_MA_DE_SD_IN'] = sum([df_inputs_prepr['addr_stat
                                                    df_inputs_prepr['addr_state:DE']
                                                    df_inputs_prepr['addr_state:IN']]

df_inputs_prepr['addr_state:GA_WA_OR'] = sum([df_inputs_prepr['addr_state:GA']
                                                    df_inputs_prepr['addr_state:OR']]

df_inputs_prepr['addr_state:WI_MT'] = sum([df_inputs_prepr['addr_state:WI'], d

df_inputs_prepr['addr_state:IL_CT'] = sum([df_inputs_prepr['addr_state:IL'], d

df_inputs_prepr['addr_state:KS_SC_CO_VT_AK_MS'] = sum([df_inputs_prepr['addr_s
                                                    df_inputs_prepr['addr_state:CO']
                                                    df_inputs_prepr['addr_state:AK']
```

```
df_inputs_prepr['addr_state:WV_NH_WY_DC_ME_ID'] = sum([df_inputs_prepr['addr_s
                                                  df_inputs_prepr['addr_state:WY']
                                                  df_inputs_prepr['addr_state:ME']
```

Preprocessing Discrete Variables

In [ ]:
```
df_temp = woe_discrete(df_inputs_prepr, 'verification_status', df_targets_prep
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

In [ ]:
```
df_temp = woe_discrete(df_inputs_prepr, 'purpose', df_targets_prepr)
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
df_inputs_prepr['purpose:educ__sm_b__wedd__ren_en__mov__house'] = sum([df_inpu
                                                                  df_inputs_prep
                                                                  df_inputs_prep
df_inputs_prepr['purpose:oth__med__vacation'] = sum([df_inputs_prepr['purpose:
                                                df_inputs_prepr['purpose:vacation
df_inputs_prepr['purpose:major_purch__car__home_impr'] = sum([df_inputs_prepr[
                                                       df_inputs_prepr['purpos
```

In [ ]:
```
df_temp = woe_discrete(df_inputs_prepr, 'initial_list_status', df_targets_prep
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

Preprocessing Continuous Variables: Automating Calculations and Visualizing Results

In [ ]:
```
def woe_ordered_continuous(df, discrete_variabe_name, good_bad_variable_df):
    df = pd.concat([df[discrete_variabe_name], good_bad_variable_df], axis = 1
    df = pd.concat([df.groupby(df.columns.values[0], as_index = False)[df.colu
                    df.groupby(df.columns.values[0], as_index = False)[df.colu
    df = df.iloc[:, [0, 1, 3]]
    df.columns = [df.columns.values[0], 'n_obs', 'prop_good']
    df['prop_n_obs'] = df['n_obs'] / df['n_obs'].sum()
    df['n_good'] = df['prop_good'] * df['n_obs']
    df['n_bad'] = (1 - df['prop_good']) * df['n_obs']
    df['prop_n_good'] = df['n_good'] / df['n_good'].sum()
    df['prop_n_bad'] = df['n_bad'] / df['n_bad'].sum()
    df['WoE'] = np.log(df['prop_n_good'] / df['prop_n_bad'])
    #df = df.sort_values(['WoE'])
    #df = df.reset_index(drop = True)
    df['diff_prop_good'] = df['prop_good'].diff().abs()
    df['diff_WoE'] = df['WoE'].diff().abs()
    df['IV'] = (df['prop_n_good'] - df['prop_n_bad']) * df['WoE']
    df['IV'] = df['IV'].sum()
    return df
```

Preprocessing Continuous Variables: Creating Dummy Variables, Part 1

In [ ]:
```
df_inputs_prepr['term_int'].unique()
```

In [ ]:
```
df_temp = woe_ordered_continuous(df_inputs_prepr, 'term_int', df_targets_prepr
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

In [ ]:
```
df_inputs_prepr['term:36'] = np.where((df_inputs_prepr['term_int'] == 36), 1, (
df_inputs_prepr['term:60'] = np.where((df_inputs_prepr['term_int'] == 60), 1, (
```

In [ ]:
```
df_inputs_prepr['emp_length_int'].unique()
```

In [ ]:
```
df_temp = woe_ordered_continuous(df_inputs_prepr, 'emp_length_int', df_targets_
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

In [ ]:
```
df_inputs_prepr['emp_length:0'] = np.where(df_inputs_prepr['emp_length_int'].is
df_inputs_prepr['emp_length:1'] = np.where(df_inputs_prepr['emp_length_int'].is
df_inputs_prepr['emp_length:2-4'] = np.where(df_inputs_prepr['emp_length_int']
df_inputs_prepr['emp_length:5-6'] = np.where(df_inputs_prepr['emp_length_int']
df_inputs_prepr['emp_length:7-9'] = np.where(df_inputs_prepr['emp_length_int']
df_inputs_prepr['emp_length:10'] = np.where(df_inputs_prepr['emp_length_int'].
```

Preprocessing Continuous Variables: Creating Dummy Variables, Part 2

In [ ]:
```
df_inputs_prepr['mths_since_issue_d'].unique()
```

In [ ]:
```
df_inputs_prepr['mths_since_issue_d_factor'] = pd.cut(df_inputs_prepr['mths_sin
```

In [ ]:
```
df_inputs_prepr['mths_since_issue_d_factor']
```

In [ ]:
```
df_temp = woe_ordered_continuous(df_inputs_prepr, 'mths_since_issue_d_factor',
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
plot_by_woe(df_temp.iloc[3: , : ], 90)
```

In [ ]:
```
df_inputs_prepr['mths_since_issue_d:<38'] = np.where(df_inputs_prepr['mths_sin
df_inputs_prepr['mths_since_issue_d:38-39'] = np.where(df_inputs_prepr['mths_s
df_inputs_prepr['mths_since_issue_d:40-41'] = np.where(df_inputs_prepr['mths_s
df_inputs_prepr['mths_since_issue_d:42-48'] = np.where(df_inputs_prepr['mths_s
df_inputs_prepr['mths_since_issue_d:49-52'] = np.where(df_inputs_prepr['mths_s
df_inputs_prepr['mths_since_issue_d:53-64'] = np.where(df_inputs_prepr['mths_s
```

```
df_inputs_prepr['mths_since_issue_d:65-84'] = np.where(df_inputs_prepr['mths_s
df_inputs_prepr['mths_since_issue_d:>84'] = np.where(df_inputs_prepr['mths_sin
```

In [ ]:
```
df_inputs_prepr['int_rate_factor'] = pd.cut(df_inputs_prepr['int_rate'], 50)
```

In [ ]:
```
df_temp = woe_ordered_continuous(df_inputs_prepr, 'int_rate_factor', df_target
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
# '< 9.548', '9.548 - 12.025', '12.025 - 15.74', '15.74 - 20.281', '> 20.281'
```

In [ ]:
```
df_inputs_prepr['int_rate:<9.548'] = np.where((df_inputs_prepr['int_rate'] <= 
df_inputs_prepr['int_rate:9.548-12.025'] = np.where((df_inputs_prepr['int_rate
df_inputs_prepr['int_rate:12.025-15.74'] = np.where((df_inputs_prepr['int_rate
df_inputs_prepr['int_rate:15.74-20.281'] = np.where((df_inputs_prepr['int_rate
df_inputs_prepr['int_rate:>20.281'] = np.where((df_inputs_prepr['int_rate'] > 
```

In [ ]:
```
df_inputs_prepr['funded_amnt_factor'] = pd.cut(df_inputs_prepr['funded_amnt'],
df_temp = woe_ordered_continuous(df_inputs_prepr, 'funded_amnt_factor', df_tar
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

Data Preparation: Continuous Variables

In [ ]:
```
df_inputs_prepr['mths_since_earliest_cr_line_factor'] = pd.cut(df_inputs_prepr
df_temp = woe_ordered_continuous(df_inputs_prepr, 'mths_since_earliest_cr_line
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
plot_by_woe(df_temp.iloc[6: , : ], 90)
```

In [ ]:
```
df_inputs_prepr['mths_since_earliest_cr_line:<140'] = np.where(df_inputs_prepr
df_inputs_prepr['mths_since_earliest_cr_line:141-164'] = np.where(df_inputs_pr
df_inputs_prepr['mths_since_earliest_cr_line:165-247'] = np.where(df_inputs_pr
df_inputs_prepr['mths_since_earliest_cr_line:248-270'] = np.where(df_inputs_pr
df_inputs_prepr['mths_since_earliest_cr_line:271-352'] = np.where(df_inputs_pr
df_inputs_prepr['mths_since_earliest_cr_line:>352'] = np.where(df_inputs_prepr
```

In [ ]:
```
df_temp = woe_ordered_continuous(df_inputs_prepr, 'delinq_2yrs', df_targets_pr
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

In [ ]:
```
df_inputs_prepr['delinq_2yrs:0'] = np.where((df_inputs_prepr['delinq_2yrs'] ==
df_inputs_prepr['delinq_2yrs:1-3'] = np.where((df_inputs_prepr['delinq_2yrs'] 
df_inputs_prepr['delinq_2yrs:>=4'] = np.where((df_inputs_prepr['delinq_2yrs'] 
```

In [ ]:
```
df_temp = woe_ordered_continuous(df_inputs_prepr, 'inq_last_6mths', df_targets
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

In [ ]:
```
df_inputs_prepr['inq_last_6mths:0'] = np.where((df_inputs_prepr['inq_last_6mth
df_inputs_prepr['inq_last_6mths:1-2'] = np.where((df_inputs_prepr['inq_last_6m
df_inputs_prepr['inq_last_6mths:3-6'] = np.where((df_inputs_prepr['inq_last_6m
df_inputs_prepr['inq_last_6mths:>6'] = np.where((df_inputs_prepr['inq_last_6mt
```

In [ ]:
```
df_temp = woe_ordered_continuous(df_inputs_prepr, 'open_acc', df_targets_prepr
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
plot_by_woe(df_temp.iloc[ : 40, :], 90)
```

In [ ]:
```
df_inputs_prepr['open_acc:0'] = np.where((df_inputs_prepr['open_acc'] == 0), 1
df_inputs_prepr['open_acc:1-3'] = np.where((df_inputs_prepr['open_acc'] >= 1) &
df_inputs_prepr['open_acc:4-12'] = np.where((df_inputs_prepr['open_acc'] >= 4)
df_inputs_prepr['open_acc:13-17'] = np.where((df_inputs_prepr['open_acc'] >= 1
df_inputs_prepr['open_acc:18-22'] = np.where((df_inputs_prepr['open_acc'] >= 1
df_inputs_prepr['open_acc:23-25'] = np.where((df_inputs_prepr['open_acc'] >= 2
df_inputs_prepr['open_acc:26-30'] = np.where((df_inputs_prepr['open_acc'] >= 2
df_inputs_prepr['open_acc:>=31'] = np.where((df_inputs_prepr['open_acc'] >= 31
```

In [ ]:
```
df_temp = woe_ordered_continuous(df_inputs_prepr, 'pub_rec', df_targets_prepr)
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
df_inputs_prepr['pub_rec:0-2'] = np.where((df_inputs_prepr['pub_rec'] >= 0) &
df_inputs_prepr['pub_rec:3-4'] = np.where((df_inputs_prepr['pub_rec'] >= 3) &
df_inputs_prepr['pub_rec:>=5'] = np.where((df_inputs_prepr['pub_rec'] >= 5), 1
```

In [ ]:
```
df_inputs_prepr['total_acc_factor'] = pd.cut(df_inputs_prepr['total_acc'], 50)
df_temp = woe_ordered_continuous(df_inputs_prepr, 'total_acc_factor', df_targe
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
# Categories: '<=27', '28-51', '>51'
df_inputs_prepr['total_acc:<=27'] = np.where((df_inputs_prepr['total_acc'] <= 
df_inputs_prepr['total_acc:28-51'] = np.where((df_inputs_prepr['total_acc'] >=
df_inputs_prepr['total_acc:>=52'] = np.where((df_inputs_prepr['total_acc'] >= 
```

In [ ]:
```
df_temp = woe_ordered_continuous(df_inputs_prepr, 'acc_now_delinq', df_targets_
df_temp
```

In [ ]:
```
plot_by_woe(df_temp)
```

In [ ]:
```
# Categories: '0', '>=1'
df_inputs_prepr['acc_now_delinq:0'] = np.where((df_inputs_prepr['acc_now_delin
df_inputs_prepr['acc_now_delinq:>=1'] = np.where((df_inputs_prepr['acc_now_del
```

In [ ]:
```
df_inputs_prepr['total_rev_hi_lim_factor'] = pd.cut(df_inputs_prepr['total_rev_
df_temp = woe_ordered_continuous(df_inputs_prepr, 'total_rev_hi_lim_factor', d
df_temp
```

In [ ]:
```
plot_by_woe(df_temp.iloc[: 50, : ], 90)
```

In [ ]:
```
# Categories
# '<=5K', '5K-10K', '10K-20K', '20K-30K', '30K-40K', '40K-55K', '55K-95K', '>95
df_inputs_prepr['total_rev_hi_lim:<=5K'] = np.where((df_inputs_prepr['total_rev
df_inputs_prepr['total_rev_hi_lim:5K-10K'] = np.where((df_inputs_prepr['total_
df_inputs_prepr['total_rev_hi_lim:10K-20K'] = np.where((df_inputs_prepr['total_
df_inputs_prepr['total_rev_hi_lim:20K-30K'] = np.where((df_inputs_prepr['total_
df_inputs_prepr['total_rev_hi_lim:30K-40K'] = np.where((df_inputs_prepr['total_
df_inputs_prepr['total_rev_hi_lim:40K-55K'] = np.where((df_inputs_prepr['total_
df_inputs_prepr['total_rev_hi_lim:55K-95K'] = np.where((df_inputs_prepr['total_
df_inputs_prepr['total_rev_hi_lim:>95K'] = np.where((df_inputs_prepr['total_rev
```

In [ ]:
```
df_inputs_prepr['installment_factor'] = pd.cut(df_inputs_prepr['installment'],
df_temp = woe_ordered_continuous(df_inputs_prepr, 'installment_factor', df_targ
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

Preprocessing Continuous Variables: Creating Dummy Variables, Part 3

In [ ]:
```
df_inputs_prepr['annual_inc_factor'] = pd.cut(df_inputs_prepr['annual_inc'], 50
df_temp = woe_ordered_continuous(df_inputs_prepr, 'annual_inc_factor', df_targe
df_temp
```

In [ ]:
```
df_inputs_prepr['annual_inc_factor'] = pd.cut(df_inputs_prepr['annual_inc'], 10
df_temp = woe_ordered_continuous(df_inputs_prepr, 'annual_inc_factor', df_targe
df_temp
```

In [ ]:
```
df_inputs_prepr_temp = df_inputs_prepr.loc[df_inputs_prepr['annual_inc'] <= 140
#loan_data_temp = loan_data_temp.reset_index(drop = True)
#df_inputs_prepr_temp
```

In [ ]:
```
df_inputs_prepr_temp["annual_inc_factor"] = pd.cut(df_inputs_prepr_temp['annua
# Here we do fine-classing: using the 'cut' method, we split the variable into
df_temp = woe_ordered_continuous(df_inputs_prepr_temp, 'annual_inc_factor', df_
# We calculate weight of evidence.
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
# We plot the weight of evidence values.
```

In [ ]:
```
# WoE is monotonically decreasing with income, so we split income in 10 equal c
df_inputs_prepr['annual_inc:<20K'] = np.where((df_inputs_prepr['annual_inc'] <=
df_inputs_prepr['annual_inc:20K-30K'] = np.where((df_inputs_prepr['annual_inc']
df_inputs_prepr['annual_inc:30K-40K'] = np.where((df_inputs_prepr['annual_inc']
df_inputs_prepr['annual_inc:40K-50K'] = np.where((df_inputs_prepr['annual_inc']
df_inputs_prepr['annual_inc:50K-60K'] = np.where((df_inputs_prepr['annual_inc']
df_inputs_prepr['annual_inc:60K-70K'] = np.where((df_inputs_prepr['annual_inc']
```

```
df_inputs_prepr['annual_inc:70K-80K'] = np.where((df_inputs_prepr['annual_inc'
df_inputs_prepr['annual_inc:80K-90K'] = np.where((df_inputs_prepr['annual_inc'
df_inputs_prepr['annual_inc:90K-100K'] = np.where((df_inputs_prepr['annual_inc
df_inputs_prepr['annual_inc:100K-120K'] = np.where((df_inputs_prepr['annual_in
df_inputs_prepr['annual_inc:120K-140K'] = np.where((df_inputs_prepr['annual_in
df_inputs_prepr['annual_inc:>140K'] = np.where((df_inputs_prepr['annual_inc'] >
```

In [ ]:
```
df_inputs_prepr_temp = df_inputs_prepr[pd.notnull(df_inputs_prepr['mths_since_
df_inputs_prepr_temp['mths_since_last_delinq_factor'] = pd.cut(df_inputs_prepr_
df_temp = woe_ordered_continuous(df_inputs_prepr_temp, 'mths_since_last_delinq_
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
# We plot the weight of evidence values.
```

In [ ]:
```
# Categories: Missing, 0-3, 4-30, 31-56, >=57
df_inputs_prepr['mths_since_last_delinq:Missing'] = np.where((df_inputs_prepr[
df_inputs_prepr['mths_since_last_delinq:0-3'] = np.where((df_inputs_prepr['mths
df_inputs_prepr['mths_since_last_delinq:4-30'] = np.where((df_inputs_prepr['mt
df_inputs_prepr['mths_since_last_delinq:31-56'] = np.where((df_inputs_prepr['m
df_inputs_prepr['mths_since_last_delinq:>=57'] = np.where((df_inputs_prepr['mt
```

Preprocessing Continuous Variables: Creating Dummy Variables

In [ ]:
```
df_inputs_prepr['dti_factor'] = pd.cut(df_inputs_prepr['dti'], 100)
df_temp = woe_ordered_continuous(df_inputs_prepr, 'dti_factor', df_targets_prep
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
df_inputs_prepr_temp = df_inputs_prepr.loc[df_inputs_prepr['dti'] <= 35, : ]
```

In [ ]:
```
df_inputs_prepr_temp['dti_factor'] = pd.cut(df_inputs_prepr_temp['dti'], 50)
df_temp = woe_ordered_continuous(df_inputs_prepr_temp, 'dti_factor', df_target
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
# Categories:
df_inputs_prepr['dti:<=1.4'] = np.where((df_inputs_prepr['dti'] <= 1.4), 1, 0)
df_inputs_prepr['dti:1.4-3.5'] = np.where((df_inputs_prepr['dti'] > 1.4) & (df
df_inputs_prepr['dti:3.5-7.7'] = np.where((df_inputs_prepr['dti'] > 3.5) & (df
df_inputs_prepr['dti:7.7-10.5'] = np.where((df_inputs_prepr['dti'] > 7.7) & (d
df_inputs_prepr['dti:10.5-16.1'] = np.where((df_inputs_prepr['dti'] > 10.5) &
df_inputs_prepr['dti:16.1-20.3'] = np.where((df_inputs_prepr['dti'] > 16.1) &
df_inputs_prepr['dti:20.3-21.7'] = np.where((df_inputs_prepr['dti'] > 20.3) &
df_inputs_prepr['dti:21.7-22.4'] = np.where((df_inputs_prepr['dti'] > 21.7) &
df_inputs_prepr['dti:22.4-35'] = np.where((df_inputs_prepr['dti'] > 22.4) & (d
df_inputs_prepr['dti:>35'] = np.where((df_inputs_prepr['dti'] > 35), 1, 0)
```

In [ ]:
```
df_inputs_prepr_temp = df_inputs_prepr[pd.notnull(df_inputs_prepr['mths_since_
df_inputs_prepr_temp['mths_since_last_record_factor'] = pd.cut(df_inputs_prepr_
df_temp = woe_ordered_continuous(df_inputs_prepr_temp, 'mths_since_last_record_
df_temp
```

In [ ]:
```
plot_by_woe(df_temp, 90)
```

In [ ]:
```
df_inputs_prepr['mths_since_last_record:Missing'] = np.where((df_inputs_prepr[
df_inputs_prepr['mths_since_last_record:0-2'] = np.where((df_inputs_prepr['mths
df_inputs_prepr['mths_since_last_record:3-20'] = np.where((df_inputs_prepr['mth
df_inputs_prepr['mths_since_last_record:21-31'] = np.where((df_inputs_prepr['m
df_inputs_prepr['mths_since_last_record:32-80'] = np.where((df_inputs_prepr['m
df_inputs_prepr['mths_since_last_record:81-86'] = np.where((df_inputs_prepr['m
df_inputs_prepr['mths_since_last_record:>86'] = np.where((df_inputs_prepr['mths
```

Preprocessing the Test Dataset

In [ ]:
```
#####
#loan_data_inputs_train = df_inputs_prepr
#####
loan_data_inputs_test = df_inputs_prepr
```

In [ ]:
```
loan_data_inputs_train.to_csv('loan_data_inputs_train.csv')
loan_data_targets_train.to_csv('loan_data_targets_train.csv')
loan_data_inputs_test.to_csv('loan_data_inputs_test.csv')
loan_data_targets_test.to_csv('loan_data_targets_test.csv')
```