In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
df = pd.read_csv('application_train.csv')
```

Exploratory data analysis

In [3]:
```python
df.head()
```

Out[3]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN
0	157876	0	Cash loans	F	N	
1	157878	0	Cash loans	M	Y	
2	157879	0	Revolving loans	M	N	
3	157880	0	Cash loans	F	N	
4	157881	0	Cash loans	F	N	

5 rows × 122 columns

In [4]:
```python
df.columns.shape
```

Out[4]:
```
(122,)
```

In [5]:
```python
len(df)
```

Out[5]:
```
257512
```

In [6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 257512 entries, 0 to 257511
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(64), int64(42), object(16)
memory usage: 239.7+ MB
```

In [7]:
```
df.describe()
```

Out[7]:

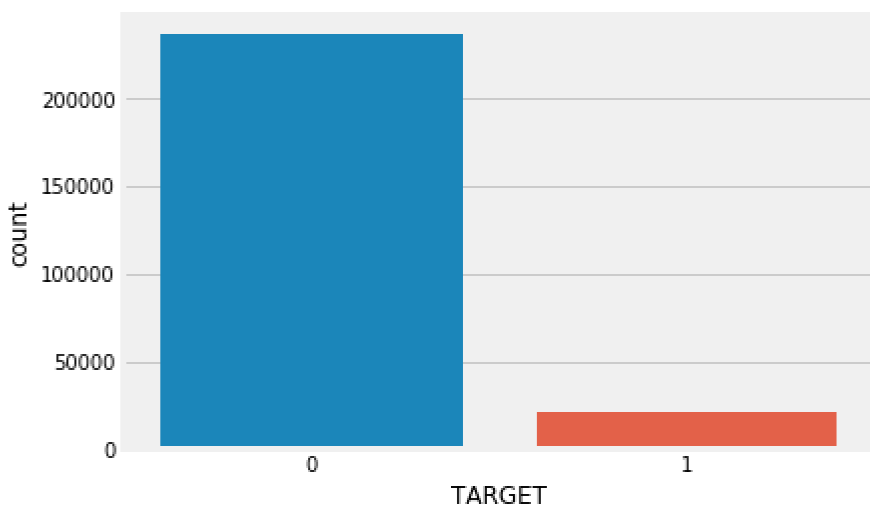
	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_A
count	257512.000000	257512.000000	257512.000000	2.575120e+05	2.575120e+05	257501
mean	307143.115397	0.080769	0.416509	1.684155e+05	5.988950e+05	27108
std	86047.050997	0.272481	0.721749	1.105872e+05	4.025061e+05	14480
min	157876.000000	0.000000	0.000000	2.610000e+04	4.500000e+04	1615
25%	232638.750000	0.000000	0.000000	1.125000e+05	2.700000e+05	16542
50%	307140.500000	0.000000	0.000000	1.476000e+05	5.135310e+05	24903
75%	381476.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596
max	456255.000000	1.000000	19.000000	1.800009e+07	4.050000e+06	230161

8 rows × 106 columns

In [8]:
```
df.TARGET.value_counts()
```

Out[8]:
```
0    236713
1     20799
Name: TARGET, dtype: int64
```

In [9]:
```
sns.countplot(x='TARGET', data=df);
```



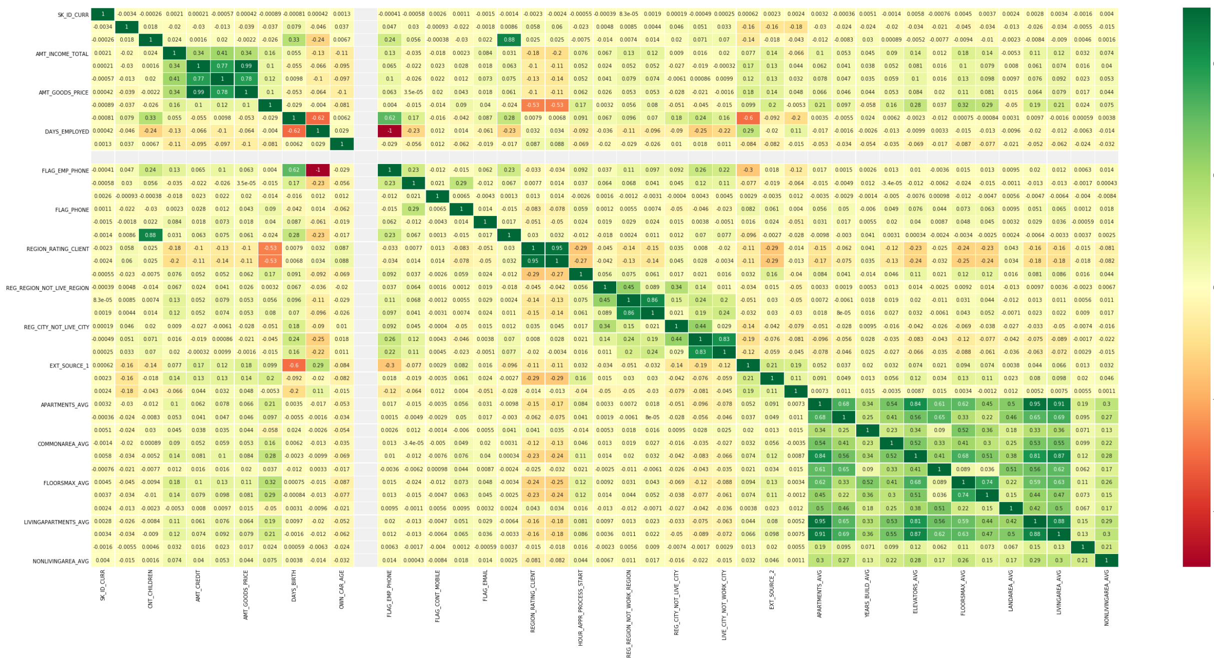
Observation : Imbalance data set

In [10]:

```
df = df.drop(
['DAYS_REGISTRATION',
'DAYS_ID_PUBLISH',
'YEARS_BEGINEXPLUATATION_AVG',
'APARTMENTS_MODE',
'BASEMENTAREA_MODE',
'YEARS_BEGINEXPLUATATION_MODE',
'YEARS_BUILD_MODE',
'COMMONAREA_MODE',
'ELEVATORS_MODE',
'ENTRANCES_MODE',
'FLOORSMAX_MODE',
'FLOORSMIN_MODE',
'LANDAREA_MODE',
'LIVINGAPARTMENTS_MODE',
'LIVINGAREA_MODE',
'NONLIVINGAPARTMENTS_MODE',
'NONLIVINGAREA_MODE',
'APARTMENTS_MEDI',
'BASEMENTAREA_MEDI',
'YEARS_BEGINEXPLUATATION_MEDI',
'YEARS_BUILD_MEDI',
'COMMONAREA_MEDI',
'ELEVATORS_MEDI',
'ENTRANCES_MEDI',
'FLOORSMAX_MEDI',
'FLOORSMIN_MEDI',
'LANDAREA_MEDI',
'LIVINGAPARTMENTS_MEDI',
'LIVINGAREA_MEDI',
'NONLIVINGAPARTMENTS_MEDI',
'NONLIVINGAREA_MEDI',
'FONDKAPREMONT_MODE',
'HOUSETYPE_MODE',
'TOTALAREA_MODE',
'WALLSMATERIAL_MODE',
'EMERGENCYSTATE_MODE',
'OBS_30_CNT_SOCIAL_CIRCLE',
'DEF_30_CNT_SOCIAL_CIRCLE',
'OBS_60_CNT_SOCIAL_CIRCLE',
'DEF_60_CNT_SOCIAL_CIRCLE',
'DAYS_LAST_PHONE_CHANGE',
'FLAG_DOCUMENT_2',
'FLAG_DOCUMENT_3',
'FLAG_DOCUMENT_4',
'FLAG_DOCUMENT_5',
'FLAG_DOCUMENT_6',
'FLAG_DOCUMENT_7',
'FLAG_DOCUMENT_8',
'FLAG_DOCUMENT_9',
'FLAG_DOCUMENT_10',
'FLAG_DOCUMENT_11',
'FLAG_DOCUMENT_12',
'FLAG_DOCUMENT_13',
'FLAG_DOCUMENT_14',
'FLAG_DOCUMENT_15',
```

```
'FLAG_DOCUMENT_16',
'FLAG_DOCUMENT_17',
'FLAG_DOCUMENT_18',
'FLAG_DOCUMENT_19',
'FLAG_DOCUMENT_20',
'FLAG_DOCUMENT_21',
'AMT_REQ_CREDIT_BUREAU_HOUR',
'AMT_REQ_CREDIT_BUREAU_DAY',
'AMT_REQ_CREDIT_BUREAU_WEEK',
'AMT_REQ_CREDIT_BUREAU_MON',
'AMT_REQ_CREDIT_BUREAU_QRT',
'AMT_REQ_CREDIT_BUREAU_YEAR'],axis = 1)
```

In [11]:
```
sns.heatmap(df.corr(),annot=True,cmap='RdYlGn',linewidths=0.2)
fig=plt.gcf()
fig.set_size_inches(40,20)
plt.show()
```



Observation:

AMT_GOODS_PRICE is highly correlated with AMT_CREDIT (ρ = 0.98697)
LIVINGAPARTMENTS_AVG is highly correlated with APARTMENTS_AVG (ρ = 0.94558)
LIVINGAREA_AVG is highly correlated with APARTMENTS_AVG (ρ = 0.91463)
REGION_RATING_CLIENT_W_CITY is highly correlated with REGION_RATING_CLIENT (ρ = 0.95087)

FLAG_MOBIL is having constant value so removing this...

In [12]:
```
df = df.drop(['AMT_GOODS_PRICE',
              'FLAG_MOBIL',
              'LIVINGAPARTMENTS_AVG',
              'LIVINGAREA_AVG',
              'REGION_RATING_CLIENT_W_CITY'],axis = 1)
```

Checking for missing values

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
SK_ID_CURR                       0
TARGET                           0
NAME_CONTRACT_TYPE               0
CODE_GENDER                      0
FLAG_OWN_CAR                     0
FLAG_OWN_REALTY                  0
CNT_CHILDREN                     0
AMT_INCOME_TOTAL                 0
AMT_CREDIT                       0
AMT_ANNUITY                     11
NAME_TYPE_SUITE               1100
NAME_INCOME_TYPE                 0
NAME_EDUCATION_TYPE              0
NAME_FAMILY_STATUS               0
NAME_HOUSING_TYPE                0
REGION_POPULATION_RELATIVE       0
DAYS_BIRTH                       0
DAYS_EMPLOYED                    0
OWN_CAR_AGE                 169979
FLAG_EMP_PHONE                   0
FLAG_WORK_PHONE                  0
FLAG_CONT_MOBILE                 0
FLAG_PHONE                       0
FLAG_EMAIL                       0
OCCUPATION_TYPE              80737
CNT_FAM_MEMBERS                  1
REGION_RATING_CLIENT             0
WEEKDAY_APPR_PROCESS_START       0
HOUR_APPR_PROCESS_START          0
REG_REGION_NOT_LIVE_REGION       0
REG_REGION_NOT_WORK_REGION       0
LIVE_REGION_NOT_WORK_REGION      0
REG_CITY_NOT_LIVE_CITY           0
REG_CITY_NOT_WORK_CITY           0
LIVE_CITY_NOT_WORK_CITY          0
ORGANIZATION_TYPE                0
EXT_SOURCE_1                145206
EXT_SOURCE_2                   534
EXT_SOURCE_3                 51021
APARTMENTS_AVG              130676
BASEMENTAREA_AVG            150744
YEARS_BUILD_AVG             171249
COMMONAREA_AVG              179905
ELEVATORS_AVG               137240
ENTRANCES_AVG               129633
FLOORSMAX_AVG               128145
FLOORSMIN_AVG               174748
LANDAREA_AVG                152869
NONLIVINGAPARTMENTS_AVG     178800
NONLIVINGAREA_AVG           142110
dtype: int64
```

Fill missing values

In [14]:

```
df['AMT_ANNUITY'].fillna(df['AMT_ANNUITY'].mean(), inplace=True)
df['APARTMENTS_AVG'].fillna(df['APARTMENTS_AVG'].mean(), inplace=True)
df['BASEMENTAREA_AVG'].fillna(df['BASEMENTAREA_AVG'].mean(), inplace=True)
df['COMMONAREA_AVG'].fillna(df['COMMONAREA_AVG'].mean(), inplace=True)

df['ELEVATORS_AVG'].fillna(df['ELEVATORS_AVG'].mean(), inplace=True)

df['ENTRANCES_AVG'].fillna(df['ENTRANCES_AVG'].mean(), inplace=True)
df['EXT_SOURCE_1'].fillna(df['EXT_SOURCE_1'].mean(), inplace=True)
df['EXT_SOURCE_2'].fillna(df['EXT_SOURCE_2'].mean(), inplace = True)
df['EXT_SOURCE_3'].fillna(df['EXT_SOURCE_3'].mean(), inplace=True)
df['FLOORSMAX_AVG'].fillna(df['FLOORSMAX_AVG'].mean(), inplace=True)
df['FLOORSMIN_AVG'].fillna(df['FLOORSMIN_AVG'].mean(), inplace=True)


df['LANDAREA_AVG'].fillna(df['LANDAREA_AVG'].mean(), inplace = True)
df['NONLIVINGAPARTMENTS_AVG'].fillna(df['NONLIVINGAPARTMENTS_AVG'].mean(), i
df['NONLIVINGAREA_AVG'].fillna(df['NONLIVINGAREA_AVG'].mean(), inplace=True)

df['OCCUPATION_TYPE'].fillna(df['OCCUPATION_TYPE'].mode().values[0], inplace
df['OWN_CAR_AGE'].fillna(df['OWN_CAR_AGE'].mean(), inplace=True)


df['CNT_FAM_MEMBERS'].fillna(1, inplace=True)
df['AMT_ANNUITY'].fillna(df['AMT_ANNUITY'].mean(), inplace=True)
df['NAME_TYPE_SUITE'].fillna(df['NAME_TYPE_SUITE'].mode().values[0], inplace
df['YEARS_BUILD_AVG'].fillna(df['YEARS_BUILD_AVG'].mean(), inplace=True)
```

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
SK_ID_CURR                    0
TARGET                        0
NAME_CONTRACT_TYPE            0
CODE_GENDER                   0
FLAG_OWN_CAR                  0
FLAG_OWN_REALTY               0
CNT_CHILDREN                  0
AMT_INCOME_TOTAL              0
AMT_CREDIT                    0
AMT_ANNUITY                   0
NAME_TYPE_SUITE               0
NAME_INCOME_TYPE              0
NAME_EDUCATION_TYPE           0
NAME_FAMILY_STATUS            0
NAME_HOUSING_TYPE             0
REGION_POPULATION_RELATIVE    0
DAYS_BIRTH                    0
DAYS_EMPLOYED                 0
OWN_CAR_AGE                   0
FLAG_EMP_PHONE                0
FLAG_WORK_PHONE               0
FLAG_CONT_MOBILE              0
FLAG_PHONE                    0
FLAG_EMAIL                    0
OCCUPATION_TYPE               0
CNT_FAM_MEMBERS               0
REGION_RATING_CLIENT          0
WEEKDAY_APPR_PROCESS_START    0
HOUR_APPR_PROCESS_START       0
REG_REGION_NOT_LIVE_REGION    0
REG_REGION_NOT_WORK_REGION    0
LIVE_REGION_NOT_WORK_REGION   0
REG_CITY_NOT_LIVE_CITY        0
REG_CITY_NOT_WORK_CITY        0
LIVE_CITY_NOT_WORK_CITY       0
ORGANIZATION_TYPE             0
EXT_SOURCE_1                  0
EXT_SOURCE_2                  0
EXT_SOURCE_3                  0
APARTMENTS_AVG                0
BASEMENTAREA_AVG              0
YEARS_BUILD_AVG               0
COMMONAREA_AVG                0
ELEVATORS_AVG                 0
ENTRANCES_AVG                 0
FLOORSMAX_AVG                 0
FLOORSMIN_AVG                 0
LANDAREA_AVG                  0
NONLIVINGAPARTMENTS_AVG       0
NONLIVINGAREA_AVG             0
dtype: int64
```

Handle categorical variables

In [16]:
```
# Mark all the organization types to appropriate category.
def f(x):
  if (x['ORGANIZATION_TYPE'] != 'Business Entity Type 3' and
     x['ORGANIZATION_TYPE'] != 'XNA' and
     x['ORGANIZATION_TYPE'] != 'Self-employed' and
     x['ORGANIZATION_TYPE'] != 'Medicine' and
     x['ORGANIZATION_TYPE'] != 'Government' and
     x['ORGANIZATION_TYPE'] != 'School' and
     x['ORGANIZATION_TYPE'] != 'Trade: type 7' and
     x['ORGANIZATION_TYPE'] != 'Kindergarten') : return "Other"
  else:
    return x['ORGANIZATION_TYPE']

df['ORGANIZATION_TYPE'] = df.apply(f, axis=1)
```

one hot encoding

In [17]:
```
df = pd.get_dummies(df, columns=['CODE_GENDER','FLAG_OWN_CAR', 'FLAG_OWN_REA
                                                   'NAME_CONTRACT_TYPE','NAM
                                                   'NAME_FAMILY_STATUS', 'NA
                                                   'NAME_INCOME_TYPE','NAME_
                                                   'OCCUPATION_TYPE','ORGANI
                                                   'WEEKDAY_APPR_PROCESS_STA
df.shape
```

Out[17]: (257512, 101)

Apply pandas profiling to see if anymore colinearity is there or not ?

In [18]:
```
#Need not to run again and again. I will update the output file,
#import pandas_profiling
#profDf = pandas_profiling.ProfileReport(df)
#profDf.to_file(outputfile="EDAs.html")
```

Observation:

NAME_INCOME_TYPE_Pensioner and ORGANIZATION_TYPE_XNA need to be deleted since they are causing colinearity

In [19]:
```
df = df.drop(['NAME_INCOME_TYPE_Pensioner','ORGANIZATION_TYPE_XNA'],axis = 1
```

Check skewness if any

In [20]:
```
df.skew(axis = 0, skipna = True)
```

Out[20]:
```
SK_ID_CURR                              -0.000818
TARGET                                   3.077167
CNT_CHILDREN                             1.993603
AMT_INCOME_TOTAL                        29.963418
AMT_CREDIT                               1.236943
AMT_ANNUITY                              1.558330
REGION_POPULATION_RELATIVE               1.488825
DAYS_BIRTH                              -0.115056
DAYS_EMPLOYED                            1.661637
OWN_CAR_AGE                              4.697084
FLAG_EMP_PHONE                          -1.662176
FLAG_WORK_PHONE                          1.504792
FLAG_CONT_MOBILE                       -23.268670
FLAG_PHONE                               0.970517
FLAG_EMAIL                               3.824572
CNT_FAM_MEMBERS                          0.994939
REGION_RATING_CLIENT                     0.087365
HOUR_APPR_PROCESS_START                 -0.028260
REG_REGION_NOT_LIVE_REGION               7.932626
REG_REGION_NOT_WORK_REGION               4.085007
LIVE_REGION_NOT_WORK_REGION              4.638841
REG_CITY_NOT_LIVE_CITY                   3.151788
REG_CITY_NOT_WORK_CITY                   1.282365
LIVE_CITY_NOT_WORK_CITY                  1.670053
EXT_SOURCE_1                            -0.104457
EXT_SOURCE_2                            -0.794435
EXT_SOURCE_3                            -0.456820
APARTMENTS_AVG                           3.783937
BASEMENTAREA_AVG                         5.578418
YEARS_BUILD_AVG                         -1.700178
                                           ...
OCCUPATION_TYPE_Cleaning staff           7.925194
OCCUPATION_TYPE_Cooking staff            6.978424
OCCUPATION_TYPE_Core staff               2.868653
OCCUPATION_TYPE_Drivers                  3.689870
OCCUPATION_TYPE_HR staff                23.545563
OCCUPATION_TYPE_High skill tech staff    4.905667
OCCUPATION_TYPE_IT staff                23.966426
OCCUPATION_TYPE_Laborers                 0.027730
OCCUPATION_TYPE_Low-skill Laborers      12.055907
OCCUPATION_TYPE_Managers                 3.387534
OCCUPATION_TYPE_Medicine staff           5.755460
OCCUPATION_TYPE_Private service staff   10.667495
OCCUPATION_TYPE_Realty agents           20.175682
OCCUPATION_TYPE_Sales staff              2.583591
OCCUPATION_TYPE_Secretaries             15.251494
OCCUPATION_TYPE_Security staff           6.569893
OCCUPATION_TYPE_Waiters/barmen staff    15.064144
ORGANIZATION_TYPE_Government             5.164803
ORGANIZATION_TYPE_Kindergarten           6.441955
ORGANIZATION_TYPE_Medicine               4.950059
ORGANIZATION_TYPE_Other                  0.739920
ORGANIZATION_TYPE_School                 5.623879
ORGANIZATION_TYPE_Self-employed          2.268713
ORGANIZATION_TYPE_Trade: type 7          5.993332
```

```
WEEKDAY_APPR_PROCESS_START_MONDAY        1.811169
WEEKDAY_APPR_PROCESS_START_SATURDAY      2.489191
WEEKDAY_APPR_PROCESS_START_SUNDAY        4.004916
WEEKDAY_APPR_PROCESS_START_THURSDAY      1.806866
WEEKDAY_APPR_PROCESS_START_TUESDAY       1.707312
WEEKDAY_APPR_PROCESS_START_WEDNESDAY     1.764320
Length: 99, dtype: float64
```

Find numerical values and plot them to see the distribution

In [21]:

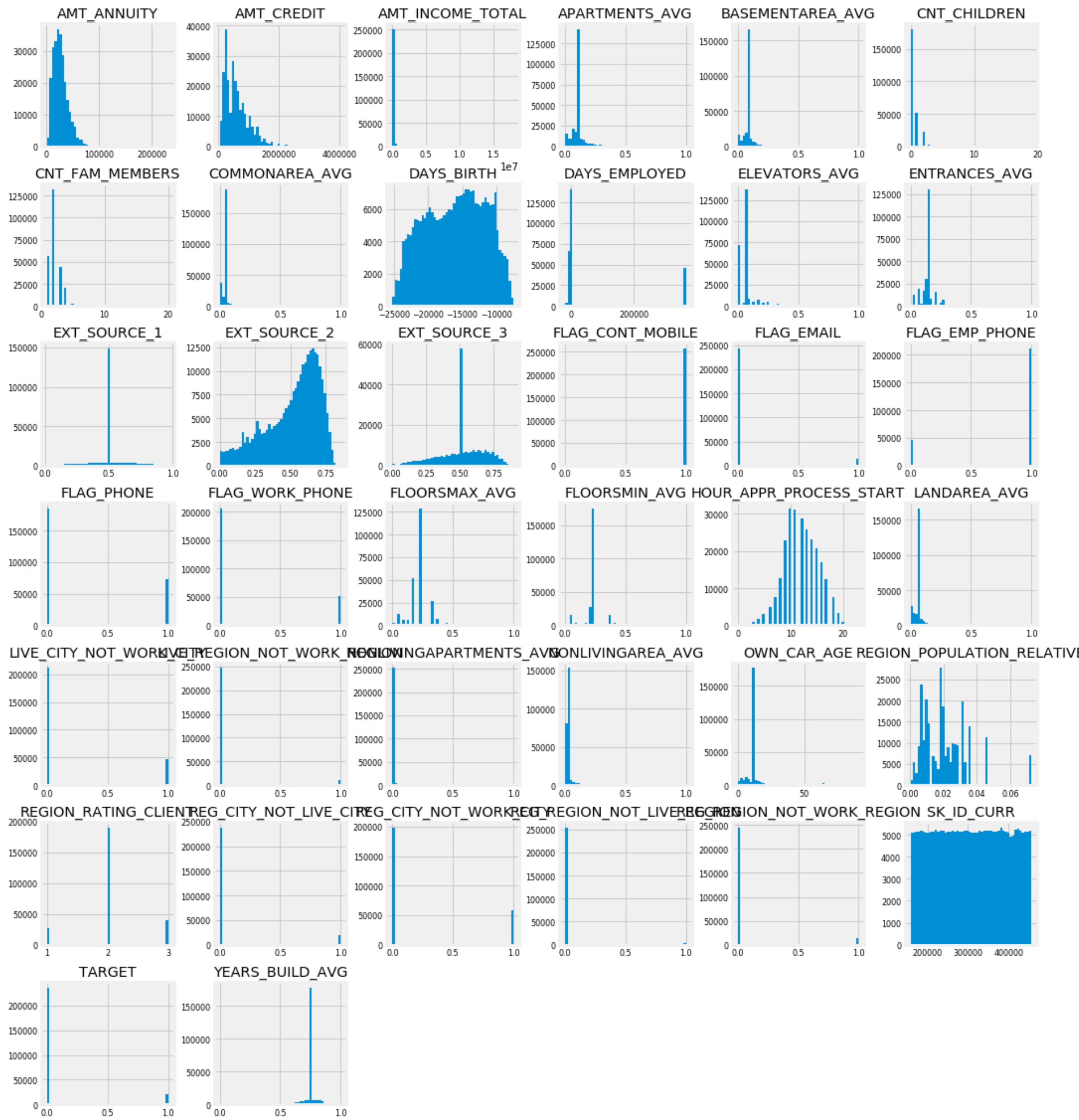
```
df_num = df.select_dtypes(include = ['float64', 'int64'])
df_num.head()
```

Out[21]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	RI
0	157876	0	0	67500.0	343800.0	16155.0	
1	157878	0	2	247500.0	945000.0	40167.0	
2	157879	0	2	180000.0	540000.0	27000.0	
3	157880	0	0	112500.0	295168.5	16011.0	
4	157881	0	0	63000.0	298512.0	17266.5	

5 rows × 38 columns

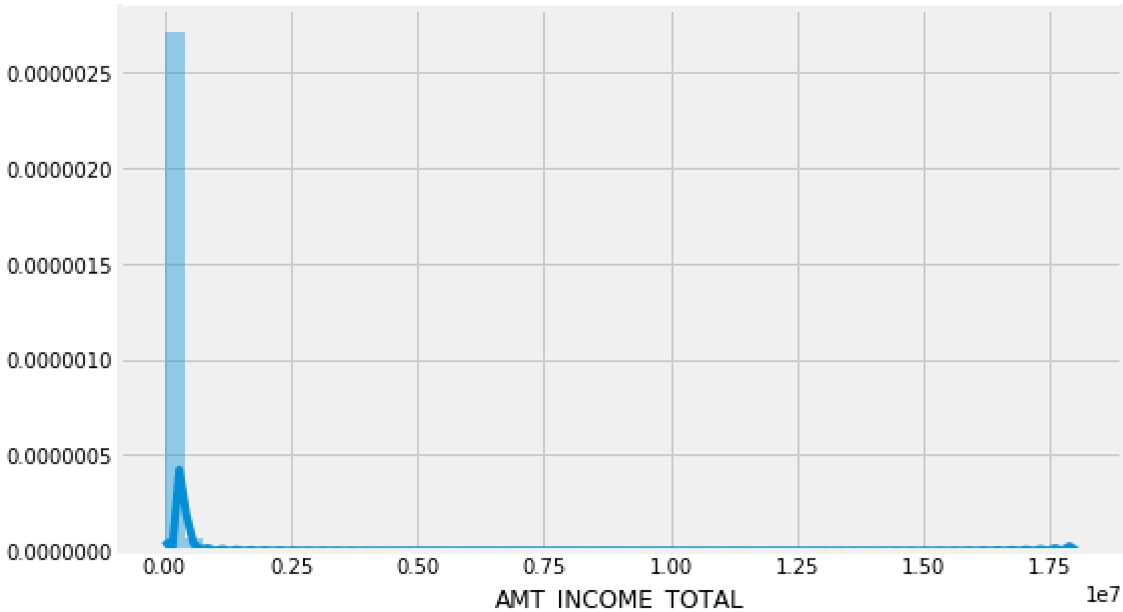
In [22]:
```
df_num.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=8); # ; avoi

```



Observation : AMT_INCOME_TOTAL and NONLIVINGAPARTMENTS_AVG are highly skewed. Lets normalize them using log normalization.

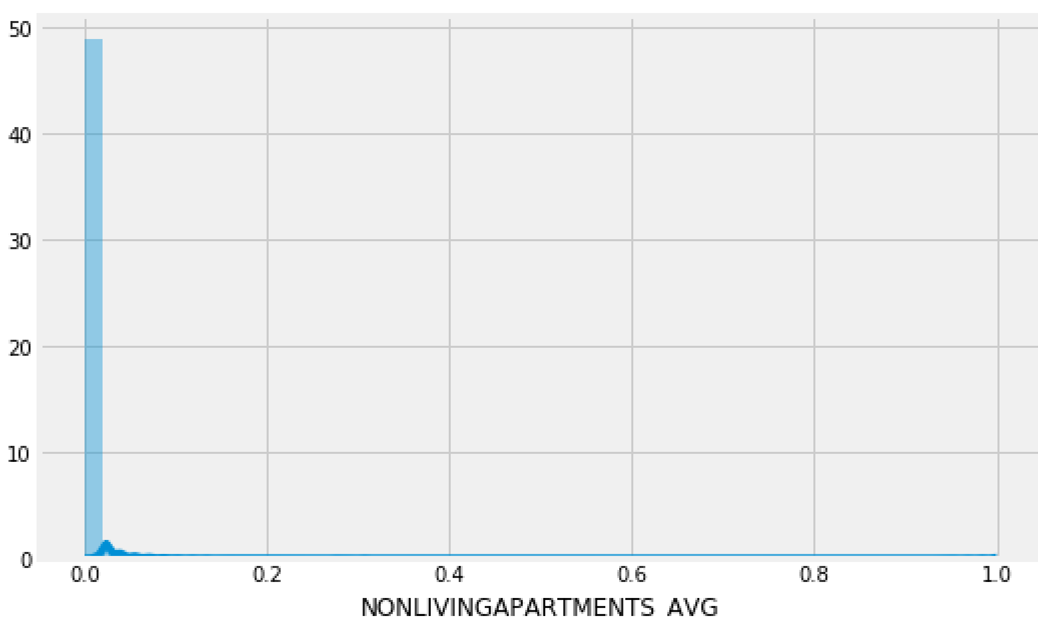
In [23]:
```
plt.figure(figsize=(8,5))

sns.distplot(df['AMT_INCOME_TOTAL'])
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1e0904a8>



In [24]:
```
plt.figure(figsize=(8,5))

sns.distplot(df['NONLIVINGAPARTMENTS_AVG'])
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x1eb59588>



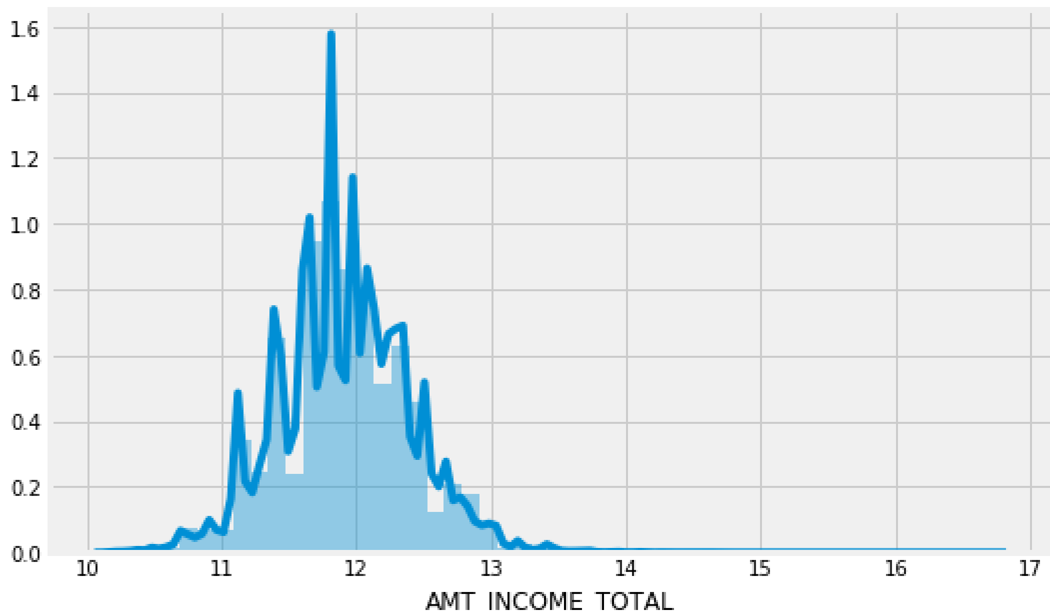
Normalizing the skewed data columns using Log normalization

In [25]:
```
temp_df = pd.DataFrame(df['AMT_INCOME_TOTAL'])

df_log = np.log(temp_df.AMT_INCOME_TOTAL)
df_log.describe()

df = df.drop('AMT_INCOME_TOTAL',axis=1)

df = df.join(df_log)
```

In [26]:
```
temp_df = pd.DataFrame(df['NONLIVINGAPARTMENTS_AVG'])

df_log = np.log(temp_df.NONLIVINGAPARTMENTS_AVG + 1)
df_log.describe()

df = df.drop('NONLIVINGAPARTMENTS_AVG',axis=1)

df = df.join(df_log)
```

In [27]:
```
plt.figure(figsize=(8,5))

sns.distplot(df['AMT_INCOME_TOTAL'])
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1e3bbba8>



Observation : After normalization the values follow normal distribution

Modeling

In [28]:
```
#Perform grid search with lesser number of data points
sampledf = df.sample(frac=0.1, replace=True, random_state=1)
```

In [29]:
```
sy = sampledf['TARGET']
sX = sampledf.drop(['TARGET','SK_ID_CURR'],axis=1)
```

In [30]:
```
from sklearn.model_selection import train_test_split
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings("ignore")
```

Apply grid search CV for hyperparamete tuning

In [23]:
```
X_tr, X_t, y_tr, y_t = train_test_split(sX, sy, test_size=0.3)
```

In [24]:
```
X_tr, X_cv, y_tr, y_cv = train_test_split(X_tr, y_tr, test_size=0.3)
```

In [29]:
```
tuned_parameters = [{'max_depth': [1, 5, 10, 50, 100],'n_estimators': [10,50
```

In [30]:
```
model = GridSearchCV(XGBClassifier(), tuned_parameters, scoring = 'roc_auc',
model.fit(X_tr, y_tr)

print(model.best_estimator_)
print(model.score(X_cv, y_cv))
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=1, min_child_weight=1, missing=None,
       n_estimators=200, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
0.748401301996768
```

In [31]:
```
y = df['TARGET']

X = df.drop(['TARGET','SK_ID_CURR'],axis=1)
```

In [32]:
```
from sklearn.model_selection import train_test_split
def split_data():
    return train_test_split(X, y, test_size=0.20, random_state=1)
X_train, X_test, y_train, y_test = split_data()
```

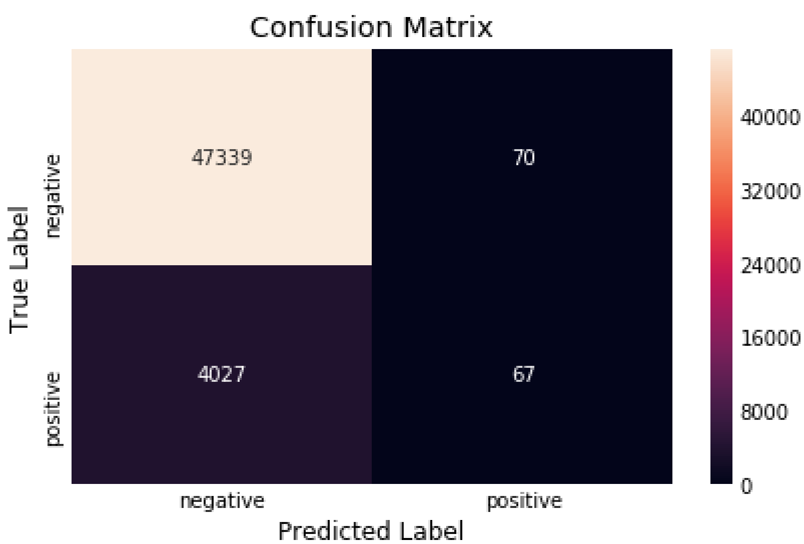
In [33]:
```
import xgboost as xgb
```

In [34]:
```
gbm = xgb.XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=None, n_estimators=200, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1).fit(X_train, y_train)
predictions = gbm.predict(X_test)
```

In [35]:
```
def showHeatMap(con_mat):
    class_label = ["negative", "positive"]
    df_cm = pd.DataFrame(con_mat, index = class_label, columns = class_label
    sns.heatmap(df_cm, annot = True, fmt = "d")
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.show()
```

In [36]:
```
from sklearn.metrics import confusion_matrix
```

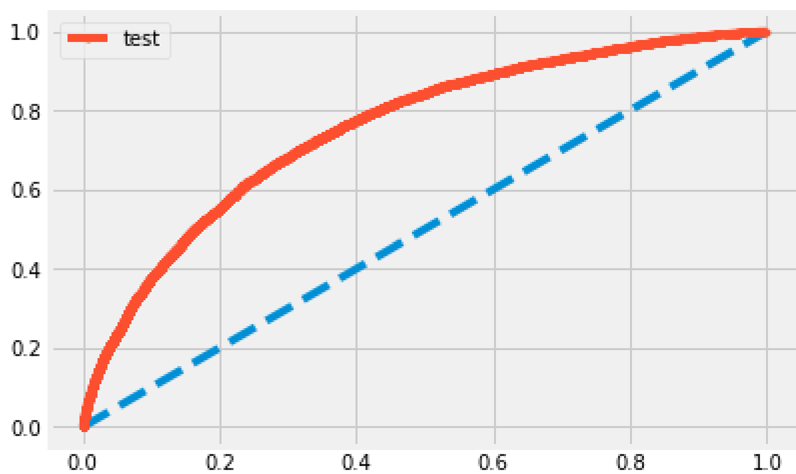
In [37]:
```
showHeatMap(confusion_matrix(y_test, predictions, [0, 1]))
```



Observation : My model predicted 4027 + 70 points wrongly.

In [38]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
acc = accuracy_score(y_test, predictions) * 100
print('\nThe accuracy of the RF classifier for %f%%' % (acc))
probs = gbm.predict_proba(X_test)
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(y_test, probs)
print('AUC: %.4f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y_test, probs)

# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.',label='test')
#plt.plot(fpr1, tpr1, marker='*',label='train')
plt.legend()
# show the plot
plt.show()
```

```
The accuracy of the RF classifier for 92.045124%
AUC: 0.7559
```



Observation : Model is predicted with AUC: .7559 better than the Dumb model

Preparing test

In [39]:
```
test_df = pd.read_csv('application_test.csv')
```

In [40]:

```
rm_cols = ['DAYS_REGISTRATION',
 'DAYS_ID_PUBLISH',
 'YEARS_BEGINEXPLUATATION_AVG',
 'APARTMENTS_MODE',
 'BASEMENTAREA_MODE',
 'YEARS_BEGINEXPLUATATION_MODE',
 'YEARS_BUILD_MODE',
 'COMMONAREA_MODE',
 'ELEVATORS_MODE',
 'ENTRANCES_MODE',
 'FLOORSMAX_MODE',
 'FLOORSMIN_MODE',
 'LANDAREA_MODE',
 'LIVINGAPARTMENTS_MODE',
 'LIVINGAREA_MODE',
 'NONLIVINGAPARTMENTS_MODE',
 'NONLIVINGAREA_MODE',
 'APARTMENTS_MEDI',
 'BASEMENTAREA_MEDI',
 'YEARS_BEGINEXPLUATATION_MEDI',
 'YEARS_BUILD_MEDI',
 'COMMONAREA_MEDI',
 'ELEVATORS_MEDI',
 'ENTRANCES_MEDI',
 'FLOORSMAX_MEDI',
 'FLOORSMIN_MEDI',
 'LANDAREA_MEDI',
 'LIVINGAPARTMENTS_MEDI',
 'LIVINGAREA_MEDI',
 'NONLIVINGAPARTMENTS_MEDI',
 'NONLIVINGAREA_MEDI',
 'FONDKAPREMONT_MODE',
 'HOUSETYPE_MODE',
 'TOTALAREA_MODE',
 'WALLSMATERIAL_MODE',
 'EMERGENCYSTATE_MODE',
 'OBS_30_CNT_SOCIAL_CIRCLE',
 'DEF_30_CNT_SOCIAL_CIRCLE',
 'OBS_60_CNT_SOCIAL_CIRCLE',
 'DEF_60_CNT_SOCIAL_CIRCLE',
 'DAYS_LAST_PHONE_CHANGE',
 'FLAG_DOCUMENT_2',
 'FLAG_DOCUMENT_3',
 'FLAG_DOCUMENT_4',
 'FLAG_DOCUMENT_5',
 'FLAG_DOCUMENT_6',
 'FLAG_DOCUMENT_7',
 'FLAG_DOCUMENT_8',
 'FLAG_DOCUMENT_9',
 'FLAG_DOCUMENT_10',
 'FLAG_DOCUMENT_11',
 'FLAG_DOCUMENT_12',
 'FLAG_DOCUMENT_13',
 'FLAG_DOCUMENT_14',
 'FLAG_DOCUMENT_15',
 'FLAG_DOCUMENT_16',
```

```
 'FLAG_DOCUMENT_17',
 'FLAG_DOCUMENT_18',
 'FLAG_DOCUMENT_19',
 'FLAG_DOCUMENT_20',
 'FLAG_DOCUMENT_21',
 'AMT_REQ_CREDIT_BUREAU_HOUR',
 'AMT_REQ_CREDIT_BUREAU_DAY',
 'AMT_REQ_CREDIT_BUREAU_WEEK',
 'AMT_REQ_CREDIT_BUREAU_MON',
 'AMT_REQ_CREDIT_BUREAU_QRT',
 'AMT_REQ_CREDIT_BUREAU_YEAR',
 'AMT_GOODS_PRICE',
 'FLAG_MOBIL',
 'LIVINGAPARTMENTS_AVG',
 'LIVINGAREA_AVG',
 'REGION_RATING_CLIENT_W_CITY']

test_df = test_df.drop(rm_cols,axis=1)
```

In [41]:

```
test_df['AMT_ANNUITY'].fillna(test_df['AMT_ANNUITY'].mean(), inplace=True)
test_df['APARTMENTS_AVG'].fillna(test_df['APARTMENTS_AVG'].mean(), inplace=T
test_df['BASEMENTAREA_AVG'].fillna(test_df['BASEMENTAREA_AVG'].mean(), inpla
test_df['COMMONAREA_AVG'].fillna(test_df['COMMONAREA_AVG'].mean(), inplace=T

test_df['ELEVATORS_AVG'].fillna(test_df['ELEVATORS_AVG'].mean(), inplace=Tru

test_df['ENTRANCES_AVG'].fillna(test_df['ENTRANCES_AVG'].mean(), inplace=Tru
test_df['EXT_SOURCE_1'].fillna(test_df['EXT_SOURCE_1'].mean(), inplace=True)
test_df['EXT_SOURCE_2'].fillna(test_df['EXT_SOURCE_2'].mean(), inplace = Tru
test_df['EXT_SOURCE_3'].fillna(test_df['EXT_SOURCE_3'].mean(), inplace=True)
test_df['FLOORSMAX_AVG'].fillna(test_df['FLOORSMAX_AVG'].mean(), inplace=Tru
test_df['FLOORSMIN_AVG'].fillna(test_df['FLOORSMIN_AVG'].mean(), inplace=Tru


test_df['LANDAREA_AVG'].fillna(test_df['LANDAREA_AVG'].mean(), inplace = Tru
test_df['NONLIVINGAPARTMENTS_AVG'].fillna(test_df['NONLIVINGAPARTMENTS_AVG']
test_df['NONLIVINGAREA_AVG'].fillna(test_df['NONLIVINGAREA_AVG'].mean(), inp

test_df['OCCUPATION_TYPE'].fillna(test_df['OCCUPATION_TYPE'].mode().values[0
test_df['OWN_CAR_AGE'].fillna(test_df['OWN_CAR_AGE'].mean(), inplace=True)


test_df['CNT_FAM_MEMBERS'].fillna(1, inplace=True)
test_df['AMT_ANNUITY'].fillna(test_df['AMT_ANNUITY'].mean(), inplace=True)
test_df['NAME_TYPE_SUITE'].fillna(test_df['NAME_TYPE_SUITE'].mode().values[0
test_df['YEARS_BUILD_AVG'].fillna(test_df['YEARS_BUILD_AVG'].mean(), inplace
```

In [42]:
```
def f(x):
 if (x['ORGANIZATION_TYPE'] != 'Business Entity Type 3' and
    x['ORGANIZATION_TYPE'] != 'XNA' and
    x['ORGANIZATION_TYPE'] != 'Self-employed' and
    x['ORGANIZATION_TYPE'] != 'Medicine' and
    x['ORGANIZATION_TYPE'] != 'Government' and
    x['ORGANIZATION_TYPE'] != 'School' and
    x['ORGANIZATION_TYPE'] != 'Trade: type 7' and
    x['ORGANIZATION_TYPE'] != 'Kindergarten') : return "Other"
 else:
   return x['ORGANIZATION_TYPE']
test_df['ORGANIZATION_TYPE'] = test_df.apply(f, axis=1)
```

In [43]:
```
test_df= pd.get_dummies(test_df, columns=['CODE_GENDER','FLAG_OWN_CAR', 'FLA
                                            'NAME_CONTRACT_TYPE','NAM
                                            'NAME_FAMILY_STATUS', 'NA
                                            'NAME_INCOME_TYPE','NAME_
                                            'OCCUPATION_TYPE','ORGANI
                                            'WEEKDAY_APPR_PROCESS_STA
```

In [44]:
```
test_df = test_df.drop(['NAME_INCOME_TYPE_Pensioner','ORGANIZATION_TYPE_XNA'
```

In [45]:
```
temp_df = pd.DataFrame(test_df['AMT_INCOME_TOTAL'])

df_log = np.log(temp_df.AMT_INCOME_TOTAL+ 1)
df_log.describe()

test_df = test_df.drop('AMT_INCOME_TOTAL',axis=1)

test_df = test_df.join(df_log)

temp_df = pd.DataFrame(test_df['NONLIVINGAPARTMENTS_AVG'])

df_log = np.log(temp_df.NONLIVINGAPARTMENTS_AVG + 1)
df_log.describe()

test_df = test_df.drop('NONLIVINGAPARTMENTS_AVG',axis=1)

test_df = test_df.join(df_log)
```

In [46]:
```
Xtest = test_df.drop(['SK_ID_CURR'],axis=1)
```

In [47]:

```
test_pred = gbm.predict(Xtest)

print(type(test_pred))

id = test_df['SK_ID_CURR']
id_arr = id.as_matrix()
tar = test_pred.reshape(-1)

saved_df = pd.DataFrame({'SK_ID_CURR':id_arr,'TARGET':tar})

saved_df.to_csv("output_final.csv",index=False)
```

```
<class 'numpy.ndarray'>
```

In [ ]:

```

```