In [3]:
```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Moz
```

```
--2020-02-10 16:29:58--  https://storage.googleapis.com/kaggle-competition
s-data/kaggle-v2/9120/860599/bundle/archive.zip?GoogleAccessId=web-data@ka
ggle-161607.iam.gserviceaccount.com&Expires=1581611378&Signature=XRXJdQkSs
wHY5iMTnjVSXiaJaaVY7zrXqaOliII2EaggoXI3pDpfrY5DyIJhiYaob1s1ZE6PVCR0H3%2BzS
fb6ryuWmEO2n9fqsAGN7wRxGn6u7xW8Y7YC258RVjvTVKLvp2Ls5ODf4XexGqMTqteAhZ9nZIO
PIeUTCiqN3sqAPr65JRgghDoYBx4eAnvXqMi7vVUx%2B5wMk86qQV1Qv64H4%2BVBpn3%2BZ2B
DJOxcrc%2FK2W7lxs6VuSppzjbMH9MSJ9pqZMF2mI%2FfuBqDwf1ZwcaawK2GG7Gt20XZrCyua
VctcRT2GGShvIWpbjtj4q4G9Txo6Ry2o2N4YTosVuuP1pdROA%3D%3D&response-content-d
isposition=attachment%3B+filename%3Dhome-credit-default-risk.zip (https://
storage.googleapis.com/kaggle-competitions-data/kaggle-v2/9120/860599/bund
le/archive.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.c
om&Expires=1581611378&Signature=XRXJdQkSswHY5iMTnjVSXiaJaaVY7zrXqaOliII2Ea
ggoXI3pDpfrY5DyIJhiYaob1s1ZE6PVCR0H3%2BzSfb6ryuWmEO2n9fqsAGN7wRxGn6u7xW8Y7
YC258RVjvTVKLvp2Ls5ODf4XexGqMTqteAhZ9nZIOPIeUTCiqN3sqAPr65JRgghDoYBx4eAnvX
qMi7vVUx%2B5wMk86qQV1Qv64H4%2BVBpn3%2BZ2BDJOxcrc%2FK2W7lxs6VuSppzjbMH9MSJ9
pqZMF2mI%2FfuBqDwf1ZwcaawK2GG7Gt20XZrCyuaVctcRT2GGShvIWpbjtj4q4G9Txo6Ry2o2
N4YTosVuuP1pdROA%3D%3D&response-content-disposition=attachment%3B+filenam
e%3Dhome-credit-default-risk.zip)
Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.203.1
28, 2607:f8b0:400c:c04::80
Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.203.
128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 721616255 (688M) [application/zip]
Saving to: ‘home-credit-default-risk.zip’

home-credit-default 100%[===================>] 688.19M   143MB/s    in 5.0
s

2020-02-10 16:30:03 (138 MB/s) - ‘home-credit-default-risk.zip’ saved [721
616255/721616255]
```

In [4]:
```
!unzip 'home-credit-default-risk.zip'
```

```
Archive:  home-credit-default-risk.zip
  inflating: HomeCredit_columns_description.csv
  inflating: POS_CASH_balance.csv
  inflating: application_test.csv
  inflating: application_train.csv
  inflating: bureau.csv
  inflating: bureau_balance.csv
  inflating: credit_card_balance.csv
  inflating: installments_payments.csv
  inflating: previous_application.csv
  inflating: sample_submission.csv
```

In [5]:
```
!pip install lightgbm
```

```
Requirement already satisfied: lightgbm in /usr/local/lib/python3.6/dist-p
ackages (2.2.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/di
st-packages (from lightgbm) (0.22.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-pack
ages (from lightgbm) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-pack
ages (from lightgbm) (1.17.5)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/di
st-packages (from scikit-learn->lightgbm) (0.14.1)
```

In [0]:
```
import pandas as pd

```

In [0]:

```
import gc
import pickle
import numpy as np
from lightgbm import LGBMClassifier
from sklearn.metrics import roc_auc_score, roc_curve,auc
from sklearn.model_selection import KFold, StratifiedKFold
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

def final_fun_1(t_df):
  # code taken from https://www.kaggle.com/gemartin/load-data-reduce-m
  def reduce_mem_usage(df):
    """ iterate through all the columns of a dataframe and modify the
        to reduce memory usage.
    """
    start_mem = df.memory_usage().sum() / 1024**2
    #print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iin
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iin
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iin
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.fin
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.f
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)

    end_mem = df.memory_usage().sum() / 1024**2
    #print('Memory usage after optimization is: {:.2f} MB'.format(end_
    #print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) /

    return df



  def one_hot(df):
    original_col = list(df.columns)
    c = [c for c in df.columns if df[c].dtype == 'object']
    df = pd.get_dummies(df, columns= c, dummy_na= True)
    new_col = [c for c in df.columns if c not in original_col]
    return df, new_col

  df = reduce_mem_usage(pd.read_csv('application_train.csv'))
  test_df = reduce_mem_usage(t_df)
  df = df.append(test_df).reset_index()

  df = df[df['CODE_GENDER'] != 'XNA']
  df = df[df['AMT_GOODS_PRICE'].notnull()]
  df = df[df['NAME_INCOME_TYPE'] != 'Maternity leave']
  df = df[df['DAYS_LAST_PHONE_CHANGE'].notnull()]
  df['DAYS_EMPLOYED'].replace(365243, np.nan, inplace= True)

  df['DAYS BIRTH / DAYS EMPLOYED'] = df['DAYS BIRTH'] / df['DAYS EMPLO
```

In [0]:

```
import gc
import pickle
import numpy as np
from lightgbm import LGBMClassifier
from sklearn.metrics import roc_auc_score, roc_curve,auc
from sklearn.model_selection import KFold, StratifiedKFold
import seaborn as sns
import warnings
import pandas as pd
warnings.simplefilter(action='ignore', category=FutureWarning)

def final_fun_2(t_df,y):
  # code taken from https://www.kaggle.com/gemartin/load-data-reduce-m
  def reduce_mem_usage(df):
    """ iterate through all the columns of a dataframe and modify the
        to reduce memory usage.
    """
    start_mem = df.memory_usage().sum() / 1024**2
    #print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iin
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iin
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iin
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.fin
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.f
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)

    end_mem = df.memory_usage().sum() / 1024**2
    #print('Memory usage after optimization is: {:.2f} MB'.format(end_
    #print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) /

    return df



  def one_hot(df):
    original_col = list(df.columns)
    c = [c for c in df.columns if df[c].dtype == 'object']
    df = pd.get_dummies(df, columns= c, dummy_na= True)
    new_col = [c for c in df.columns if c not in original_col]
    return df, new_col

  #df = reduce_mem_usage(pd.read_csv('application_train.csv'))
  df = reduce_mem_usage(t_df)
  #df = df.reset_index()

  df = df[df['CODE_GENDER'] != 'XNA']
  df = df[df['AMT_GOODS_PRICE'].notnull()]
  df = df[df['NAME_INCOME_TYPE'] != 'Maternity leave']
  df = df[df['DAYS_LAST_PHONE_CHANGE'].notnull()]
  df['DAYS_EMPLOYED'].replace(365243, np.nan, inplace= True)

```

In [32]:
```
1 test_df=pd.read_csv('application_test.csv')
2 test_df=test_df[:1]
3 test=final_fun_1(test_df)
4 test[['SK_ID_CURR', 'TARGET']]
```

Out[32]:

	SK_ID_CURR	TARGET
307511	100001	0.035038

In [33]:
```
1 train_df=pd.read_csv('application_train.csv')
2 train_df=train_df.reset_index()
3 y=train_df['TARGET']
4 print(final_fun_2(train_df,y))
```

```
0.7836583223570655
```

In [0]:
```
1
```