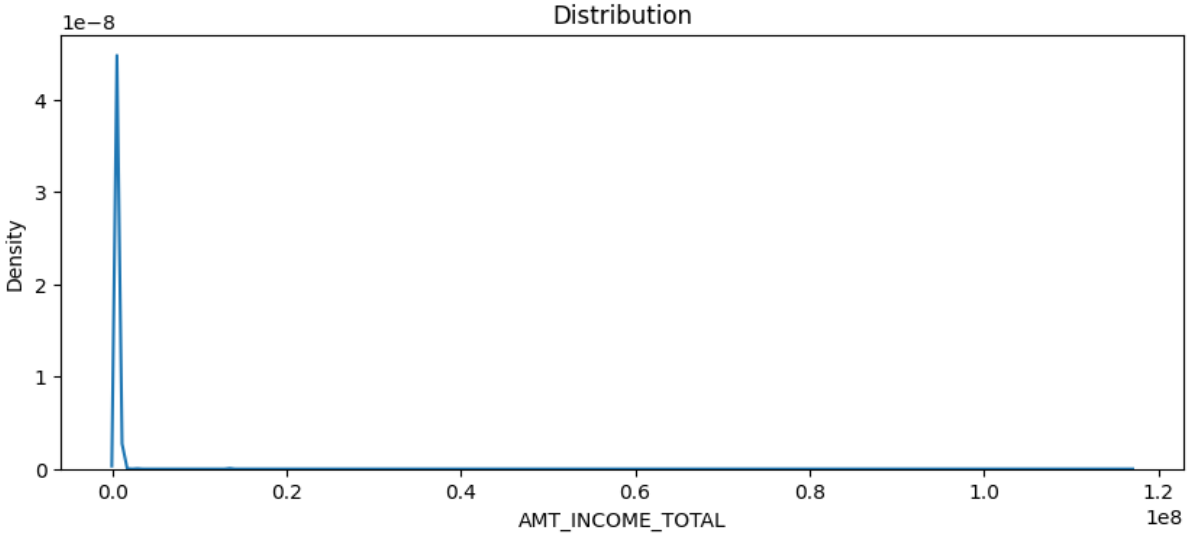
Báo cáo tổng kết

1. Income by Organization,Occupation

```
other_col.select_dtypes('object').apply(pd.Series.nunique, axis = 0)
```
✓ 0.2s

NAME_CONTRACT_TYPE	2
CODE_GENDER	3
FLAG_OWN_CAR	2
FLAG_OWN_REALTY	2
NAME_TYPE_SUITE	7
NAME_INCOME_TYPE	8
NAME_EDUCATION_TYPE	5
NAME_FAMILY_STATUS	6
NAME_HOUSING_TYPE	6
OCCUPATION_TYPE	18
WEEKDAY_APPR_PROCESS_START	7
ORGANIZATION_TYPE	58
FONDKAPREMONT_MODE	4
HOUSETYPE_MODE	3
WALLSMATERIAL_MODE	7
EMERGENCYSTATE_MODE	2

dtype: int64




```
other_col[['AMT_INCOME_TOTAL', 'OCCUPATION_TYPE']].groupby('OCCUPATION_TYPE').median()
```
✓ 0.8s

OCCUPATION_TYPE	AMT_INCOME_TOTAL
Accountants	178650.0
Cleaning staff	112500.0
Cooking staff	126000.0
Core staff	157500.0
Drivers	180000.0
HR staff	158400.0
High skill tech staff	157500.0
IT staff	180000.0

```
other_col[['AMT_INCOME_TOTAL', 'ORGANIZATION_TYPE']].groupby('ORGANIZATION_TYPE').median()
```

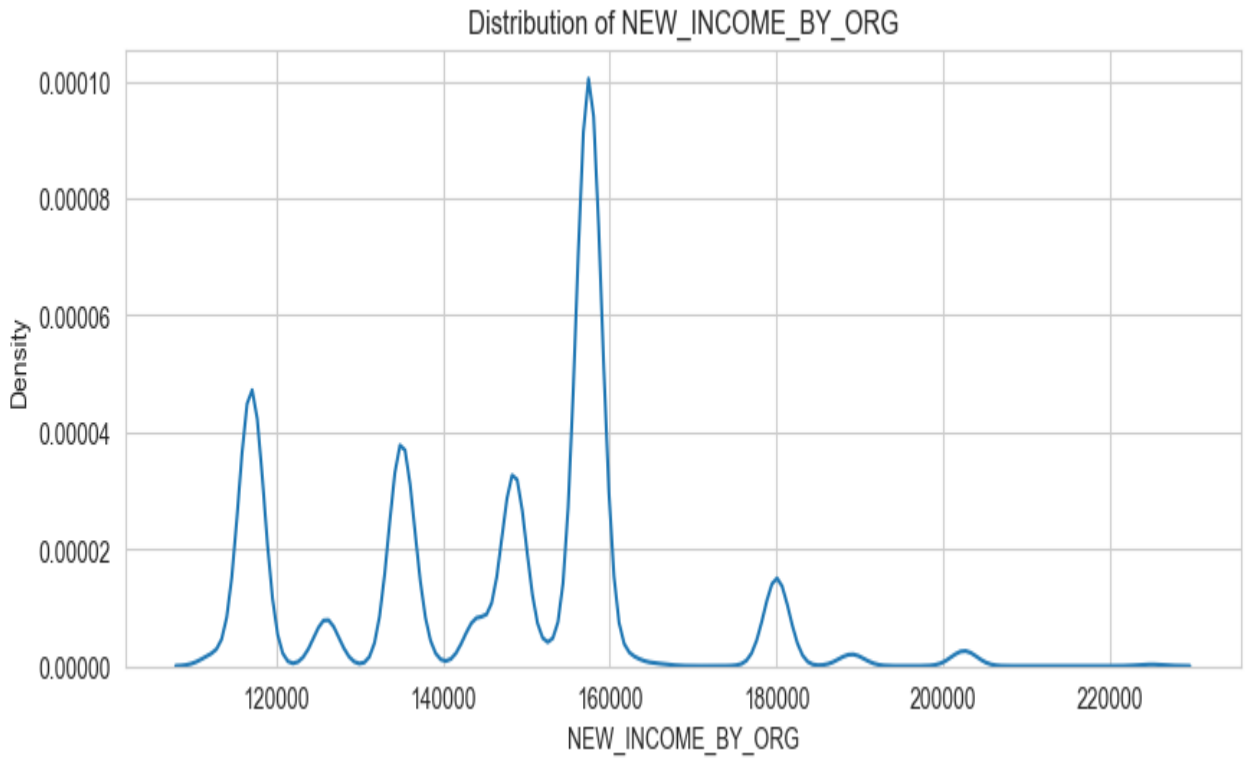
ORGANIZATION_TYPE	AMT_INCOME_TOTAL
Advertising	165600.0
Agriculture	126000.0
Bank	157500.0
Business Entity Type 1	157500.0
Business Entity Type 2	157500.0
Business Entity Type 3	157500.0
Cleaning	135000.0
Construction	180000.0
Culture	157500.0
Electricity	157500.0
Emergency	162000.0
Government	135000.0
Hotel	135000.0
Housing	135000.0

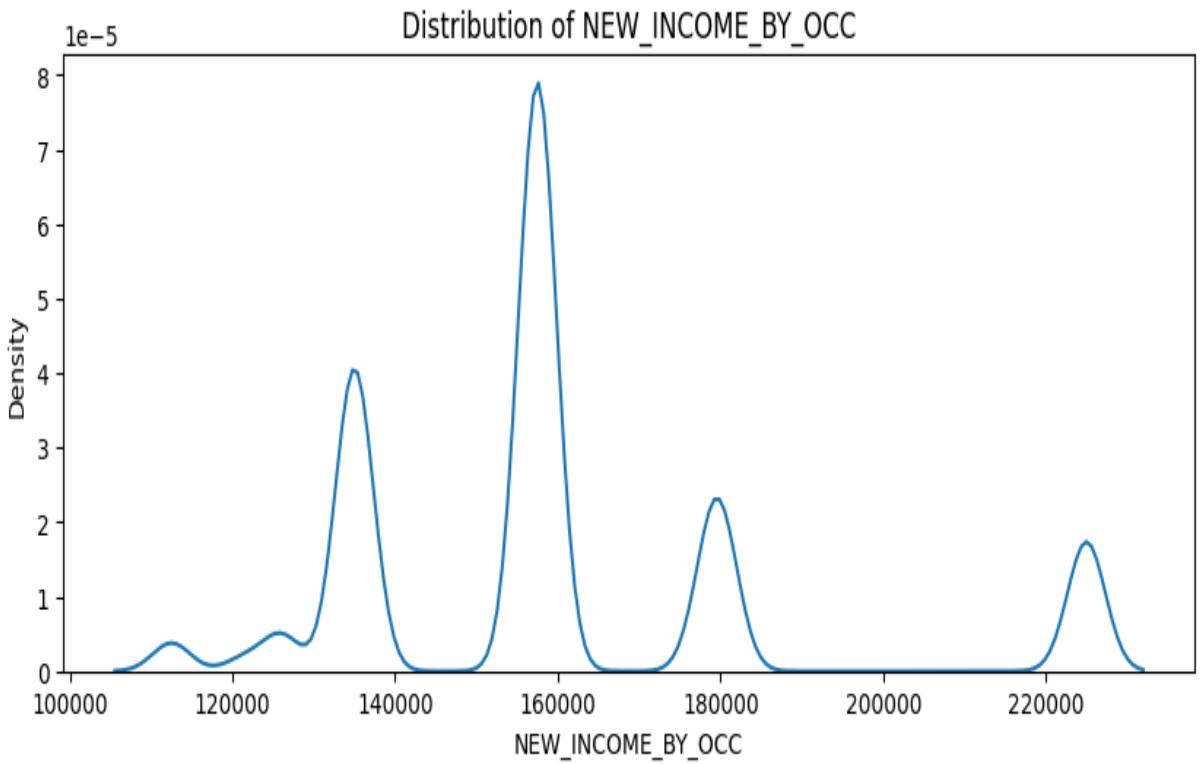
```python
income_by_organi_train = other_col[['AMT_INCOME_TOTAL', 'ORGANIZATION_TYPE']].groupby('ORGANIZATION_TYPE').median()['AMT_INCOME_TOTAL']
income_by_organi_test = application_test[['AMT_INCOME_TOTAL', 'ORGANIZATION_TYPE']].groupby('ORGANIZATION_TYPE').median()['AMT_INCOME_TOTAL']
income_by_occupa_train = other_col[['AMT_INCOME_TOTAL', 'OCCUPATION_TYPE']].groupby('OCCUPATION_TYPE').median()['AMT_INCOME_TOTAL']
income_by_occupa_test = application_test[['AMT_INCOME_TOTAL', 'OCCUPATION_TYPE']].groupby('OCCUPATION_TYPE').median()['AMT_INCOME_TOTAL']
```

Python

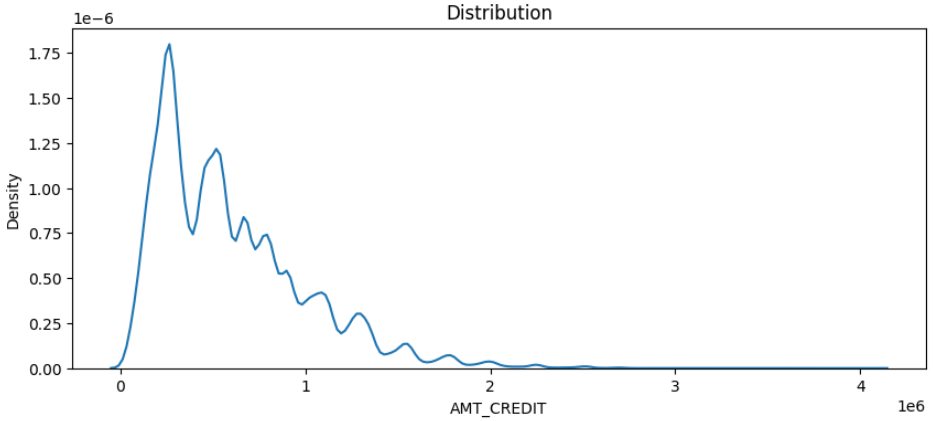
```python
other_col['NEW_INCOME_BY_ORG'] = other_col['ORGANIZATION_TYPE'].map(income_by_organi_train)
application_test['NEW_INCOME_BY_ORG'] = application_test['ORGANIZATION_TYPE'].map(income_by_organi_test)
other_col['NEW_INCOME_BY_OCC'] = other_col['OCCUPATION_TYPE'].map(income_by_occupa_train)
application_test['NEW_INCOME_BY_OCC'] = application_test['OCCUPATION_TYPE'].map(income_by_occupa_test)
```

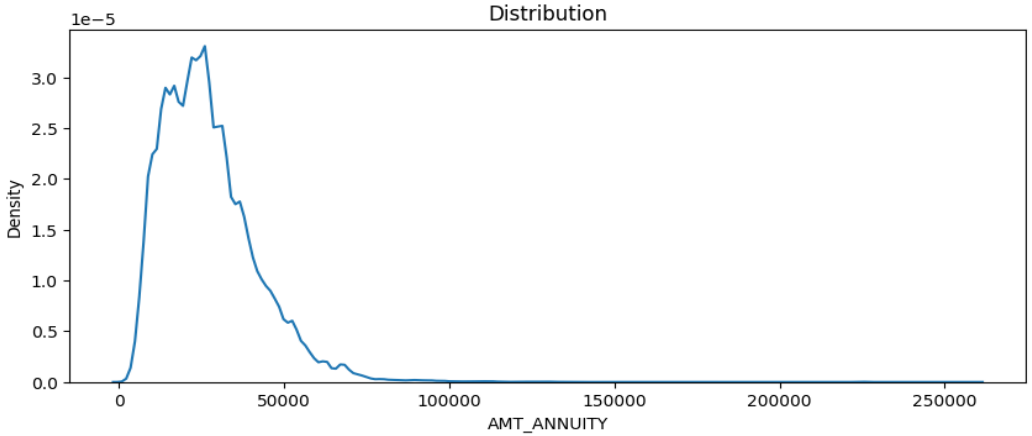
Python

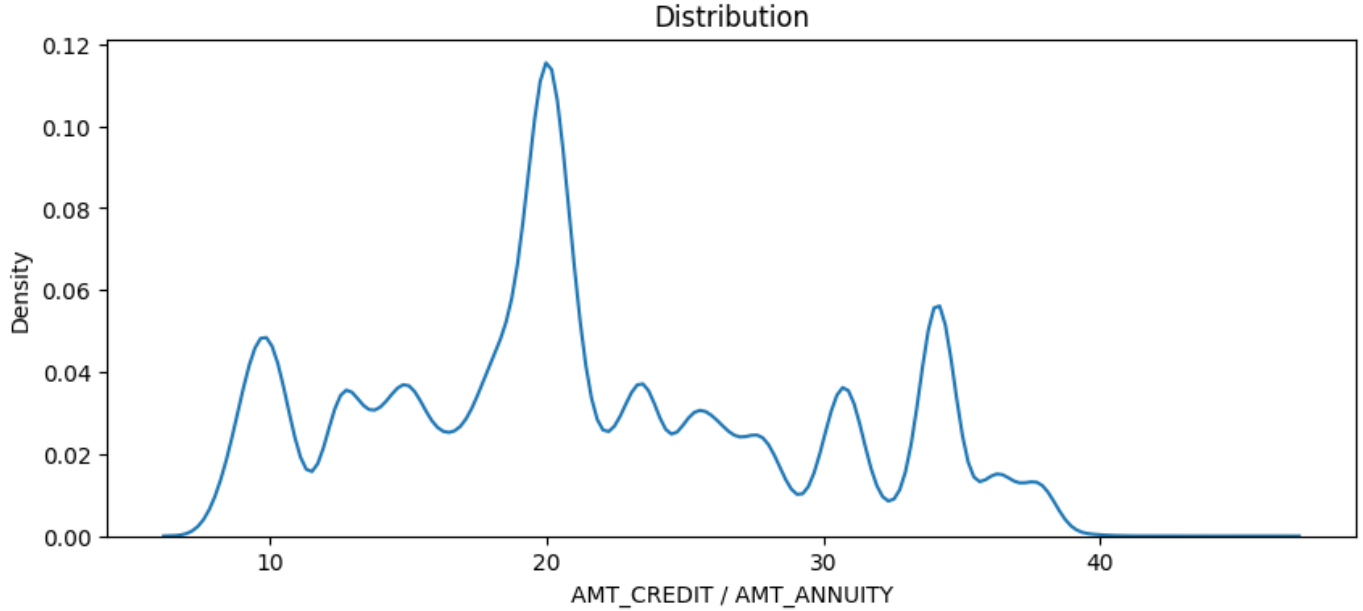




2.AMT_CREDIT/AMT_ANNUITY

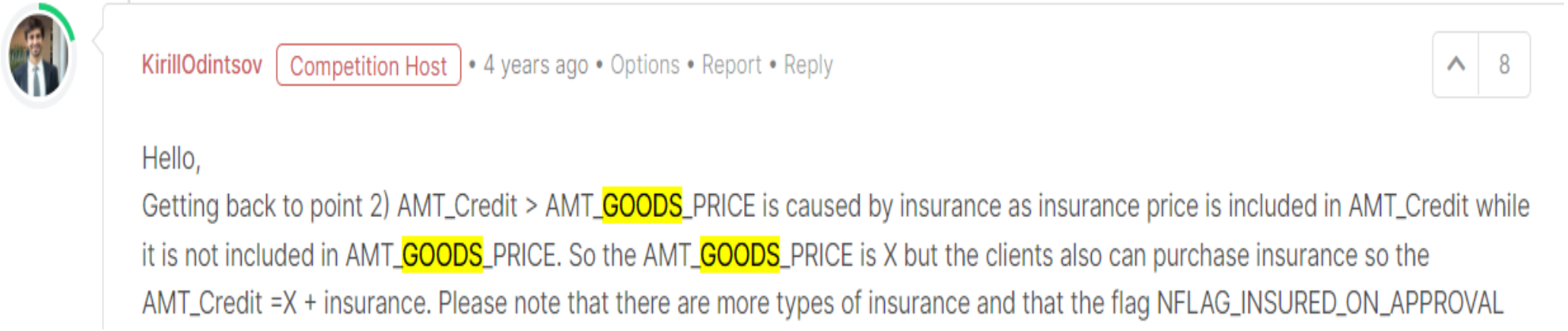




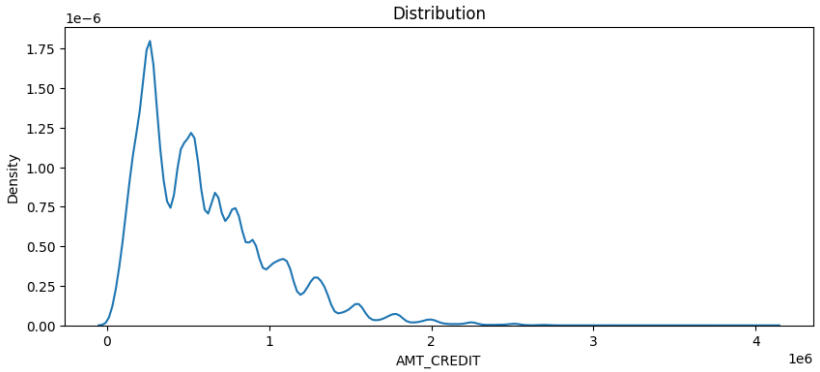


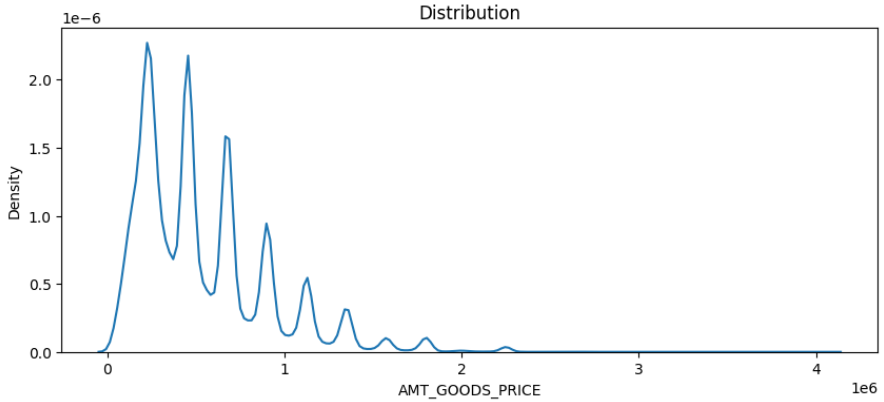
3.AMT_CREDIT/AMT_GOODS_PRICE

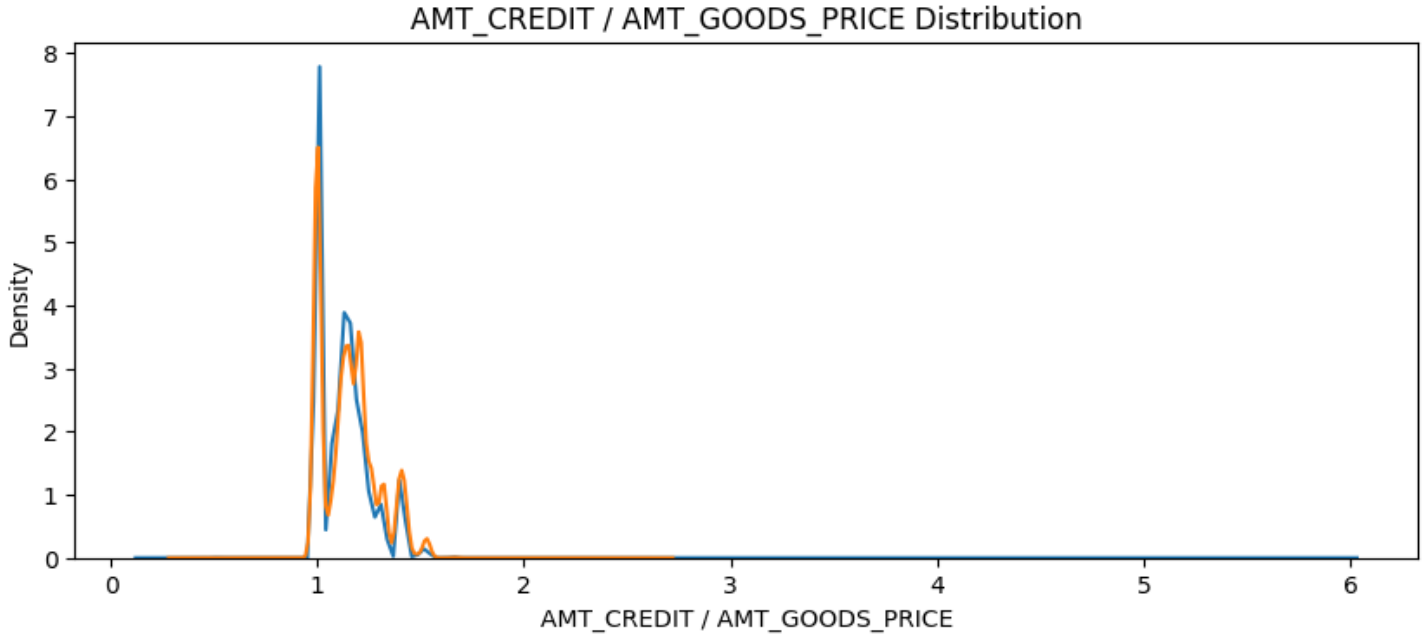
- Đây là **tỷ lệ cho vay trên giá trị(Loan-To-Value ratio)** là một **đánh giá về rủi ro cho vay** mà các tổ chức tài chính và các tổ chức cho vay khác kiểm tra trước khi chấp thuận một khoản thế chấp.
- Thông thường, các đánh giá cho vay có tỷ lệ LTV cao được coi là các khoản vay có rủi ro cao hơn. Do đó, nếu thế chấp được chấp thuận, khoản vay có lãi suất cao hơn.



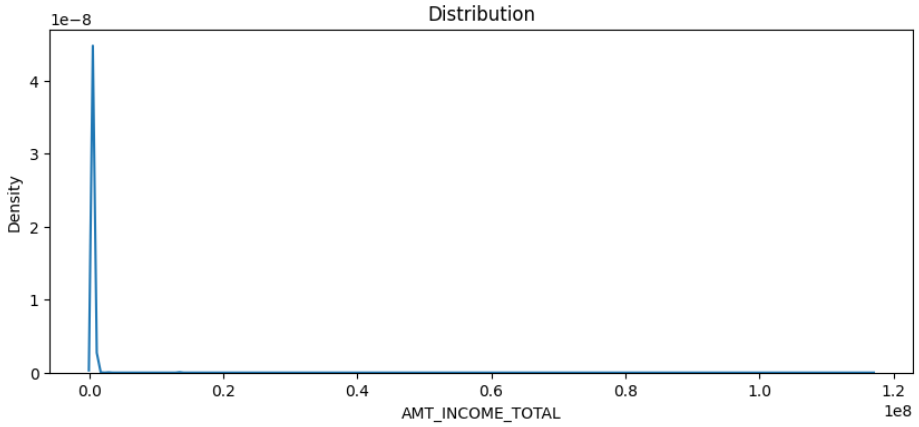
3.AMT_CREDIT/AMT_GOODS_PRICE

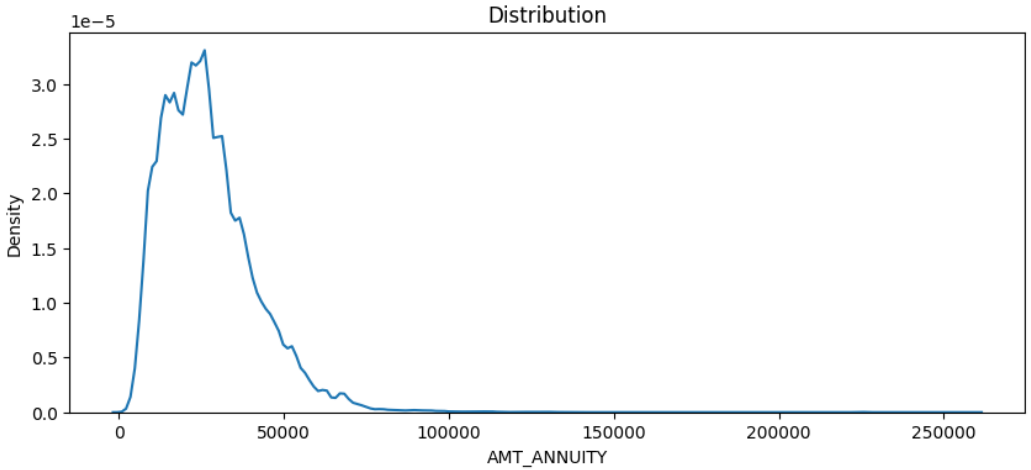


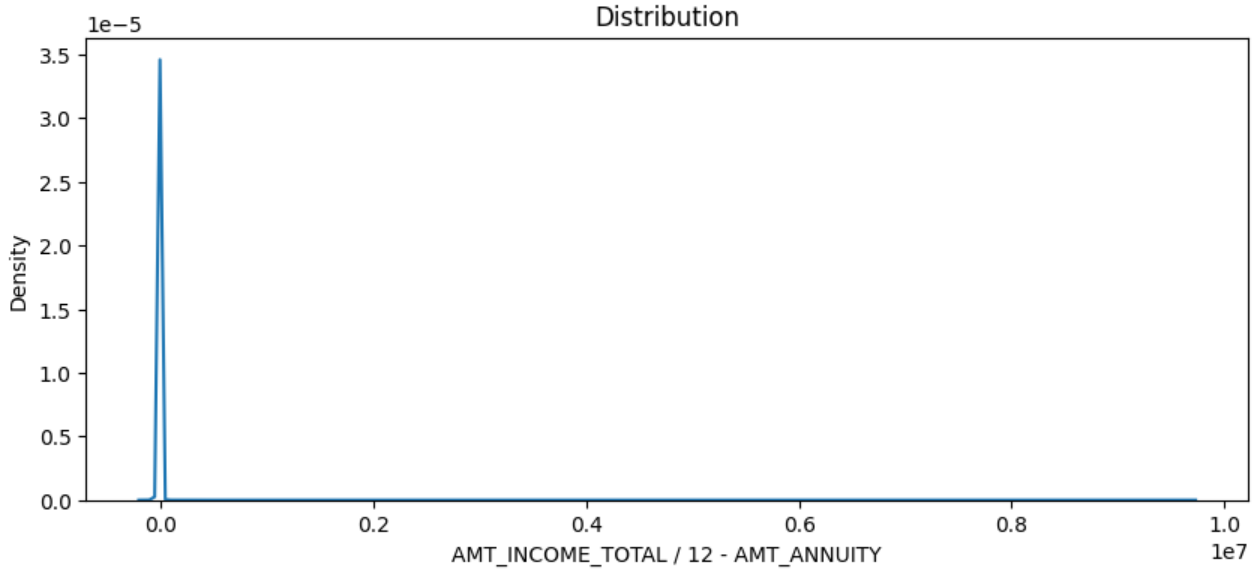




4.AMT_INCOME_TOTAL/12-AMT_ANNUITY







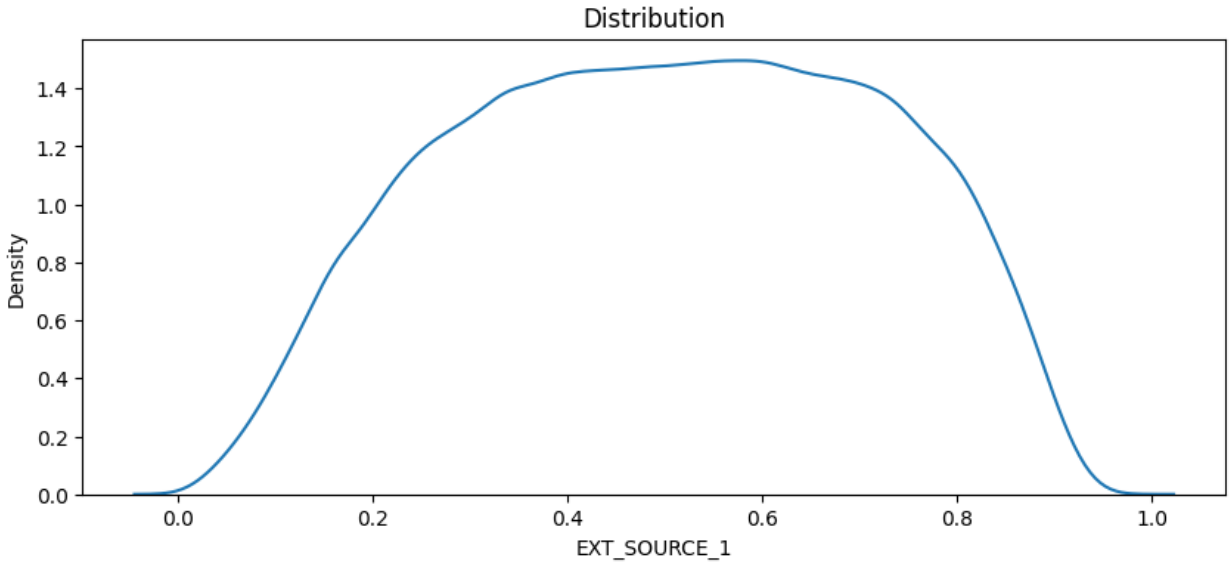

```
other_col['AMT_INCOME_TOTAL / 12 - AMT_ANNUITY'].describe()
✓ 0.1s
count    3.074990e+05
mean    -1.304214e+04
std      2.215326e+04
min     -2.062500e+05
25%     -1.987800e+04
50%     -1.171800e+04
75%     -4.638000e+03
max      9.723806e+06
Name: AMT_INCOME_TOTAL / 12 - AMT_ANNUITY, dtype: float64
```

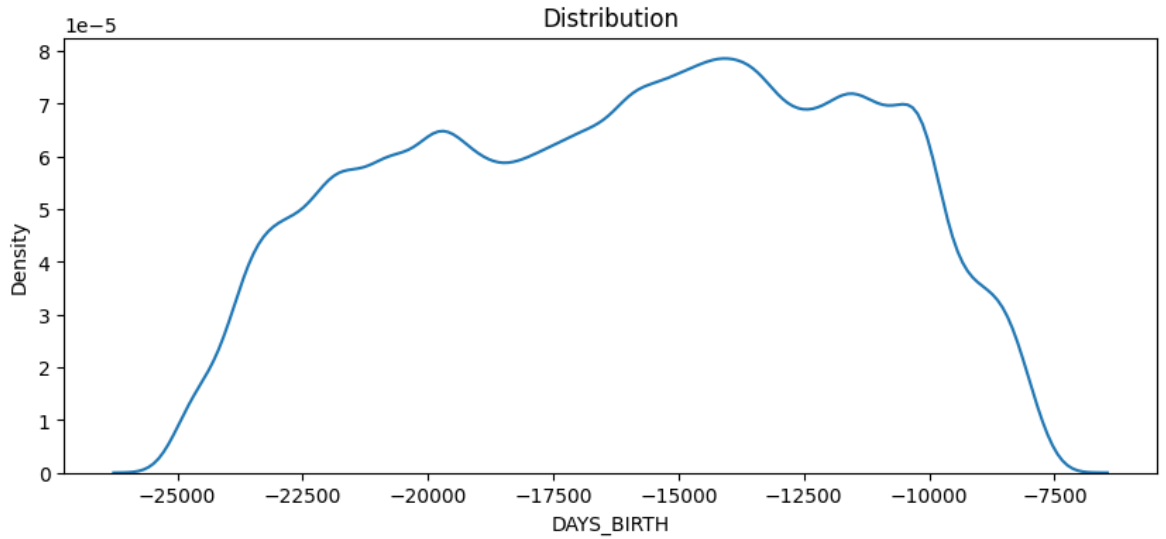
5.EXT_SOURCE_1/DAYS_BIRTH

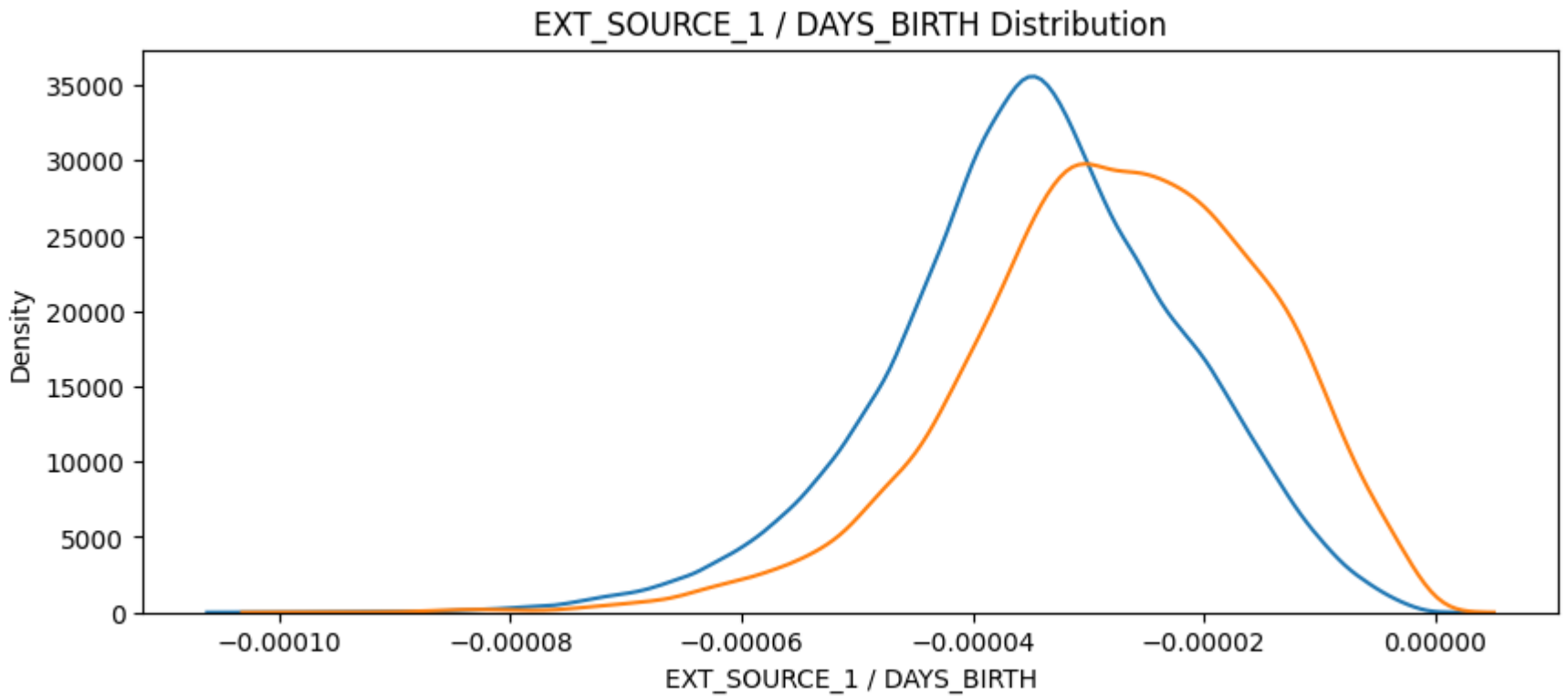
```
corr_matrix = other_col.corr().abs()
corr_matrix_ext = corr_matrix.loc[corr_matrix['EXT_SOURCE_1'] > 0.5]['EXT_SOURCE_1']
corr_matrix_ext
#corr_matrix_ext.style.background_gradient(cmap = 'coolwarm')
```

✓ 7.9s

```
DAYS_BIRTH                  0.600610
EXT_SOURCE_1                1.000000
```





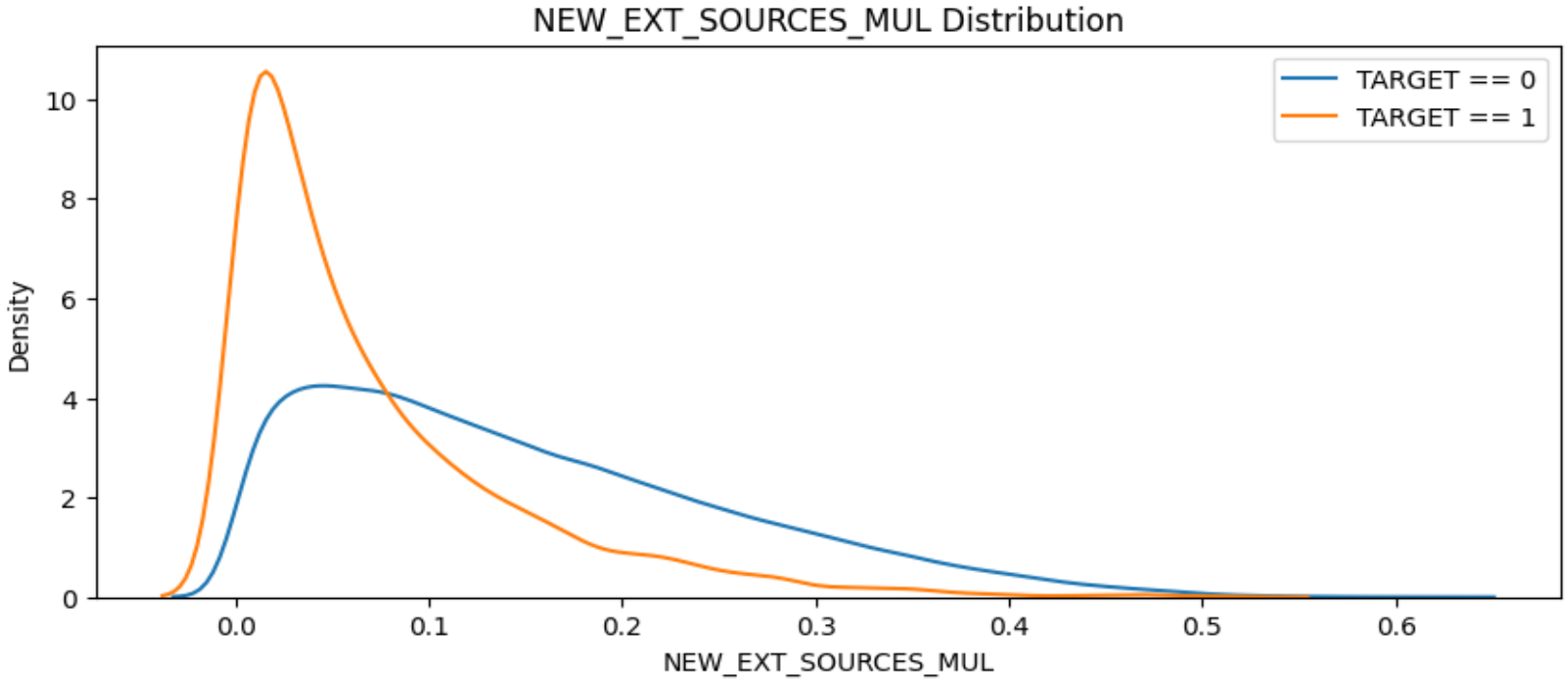

EXT_SOURCE_1 / DAYS_BIRTH Distribution
Density
35000
30000
25000
20000
15000
10000
5000
0
−0.00010
−0.00008
−0.00006
−0.00004
−0.00002
0.00000
EXT_SOURCE_1 / DAYS_BIRTH

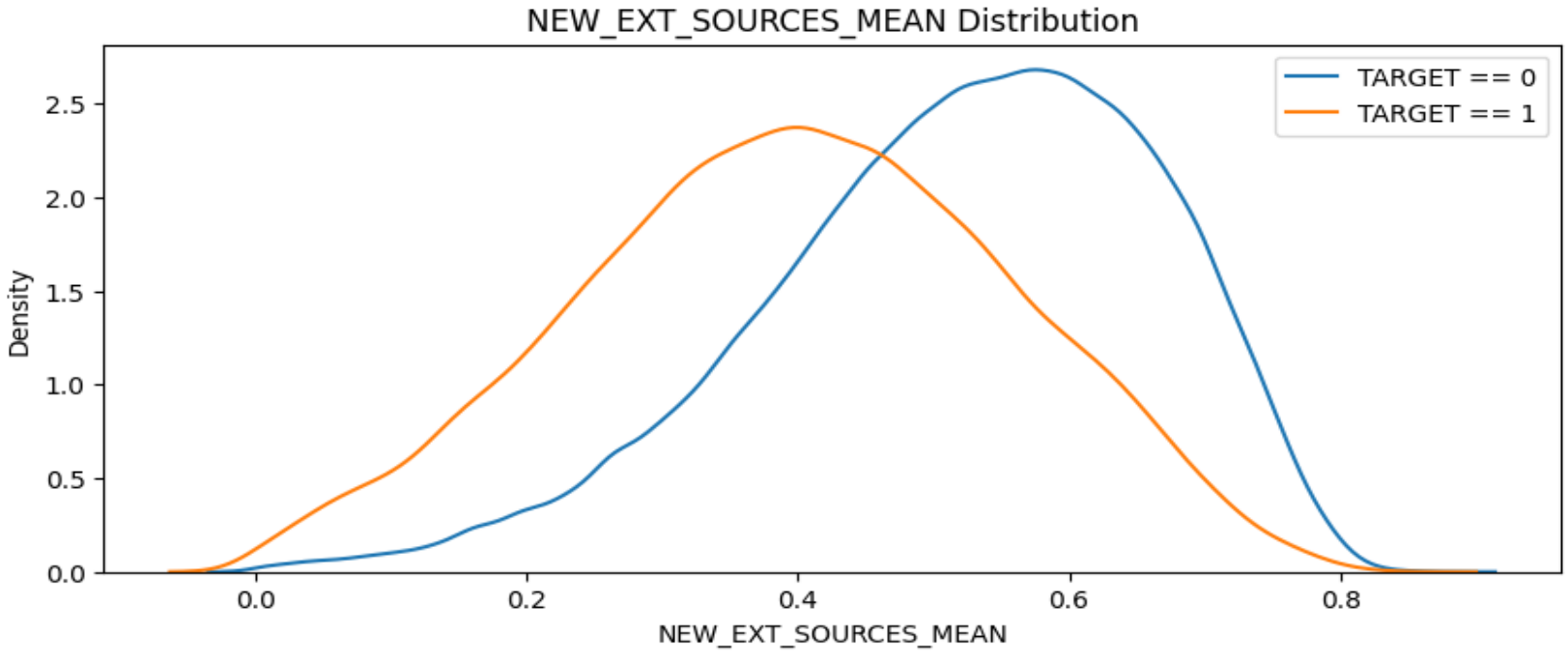
6.Feature liên quan đến EXT_SOURCE

```
other_col['NEW_EXT_SOURCES_MUL'] = other_col['EXT_SOURCE_1'] * other_col['EXT_SOURCE_2'] * other_col['EXT_SOURCE_3']
other_col['NEW_EXT_SOURCES_MEAN'] = other_col[['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3']].mean(axis=1)
```

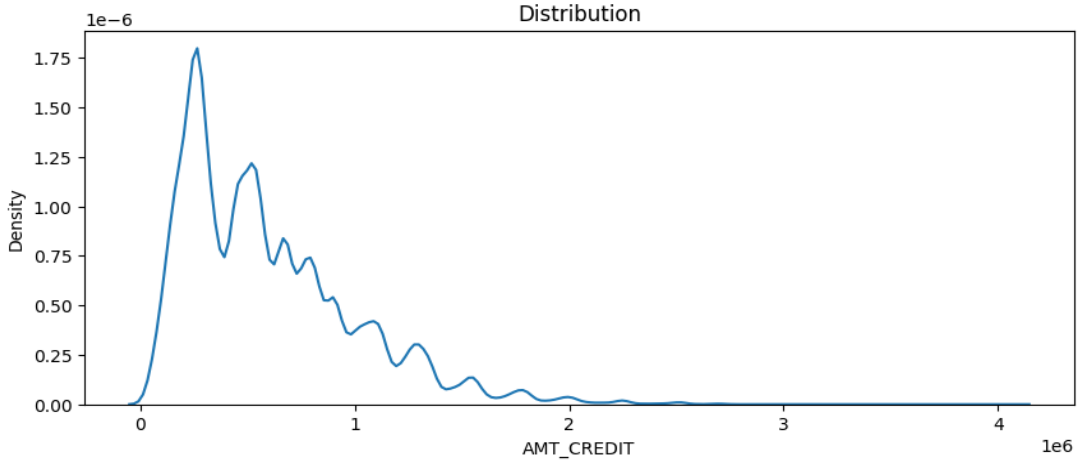
✓ 0.3s

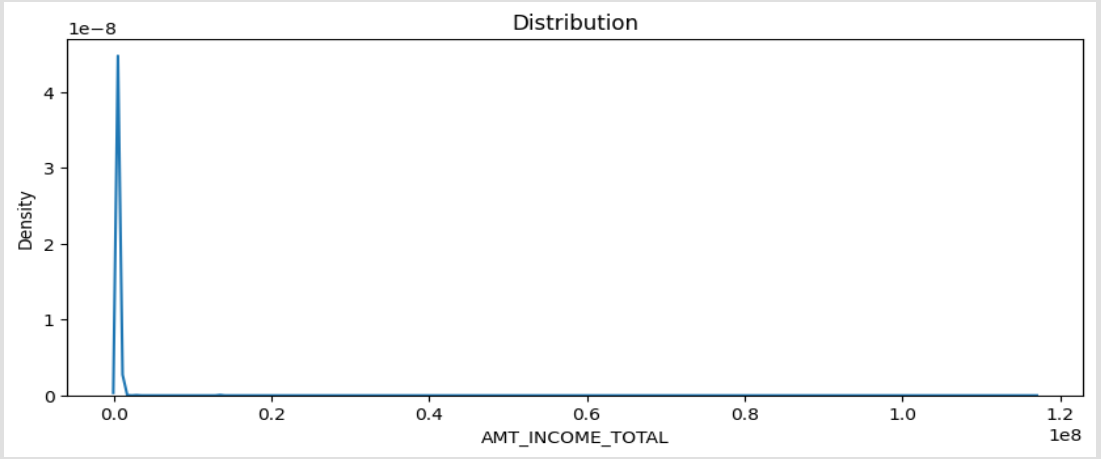
	TARGET	EXT_SOURCE_1	EXT_SOURCE_2	EXT_SOURCE_3	NEW_EXT_SOURCES_MUL	NEW_EXT_SOURCES_MEAN
TARGET	1.000000	-0.155317	-0.160472	-0.178919	-0.188552	-0.222052
EXT_SOURCE_1	-0.155317	1.000000	0.213982	0.186846	0.681550	0.741583
EXT_SOURCE_2	-0.160472	0.213982	1.000000	0.109167	0.561639	0.749328
EXT_SOURCE_3	-0.178919	0.186846	0.109167	1.000000	0.618632	0.704012
NEW_EXT_SOURCES_MUL	-0.188552	0.681550	0.561639	0.618632	1.000000	0.931005
NEW_EXT_SOURCES_MEAN	-0.222052	0.741583	0.749328	0.704012	0.931005	1.000000

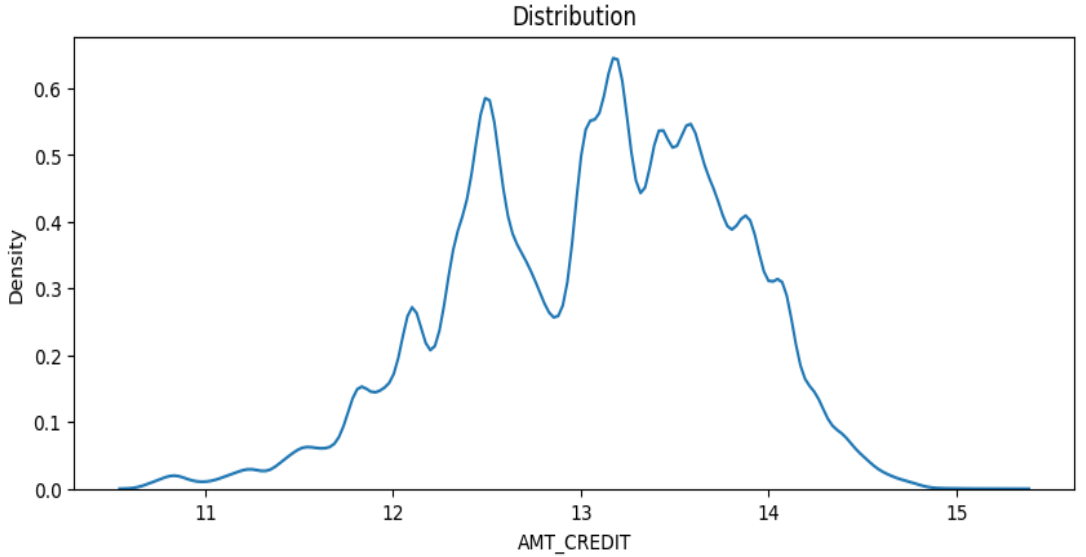

NEW_EXT_SOURCES_MUL Distribution
TARGET == 0
TARGET == 1
Density
NEW_EXT_SOURCES_MUL
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6

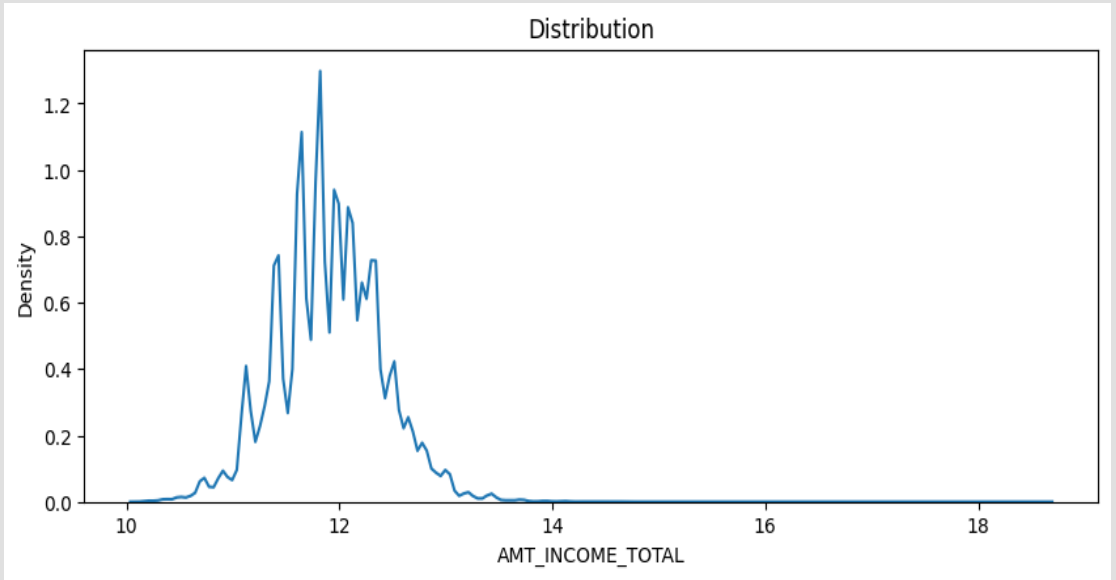

NEW_EXT_SOURCES_MEAN Distribution
TARGET == 0
TARGET == 1
Density
NEW_EXT_SOURCES_MEAN
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8

7.Log scale các feature phân phối lệch về một phía









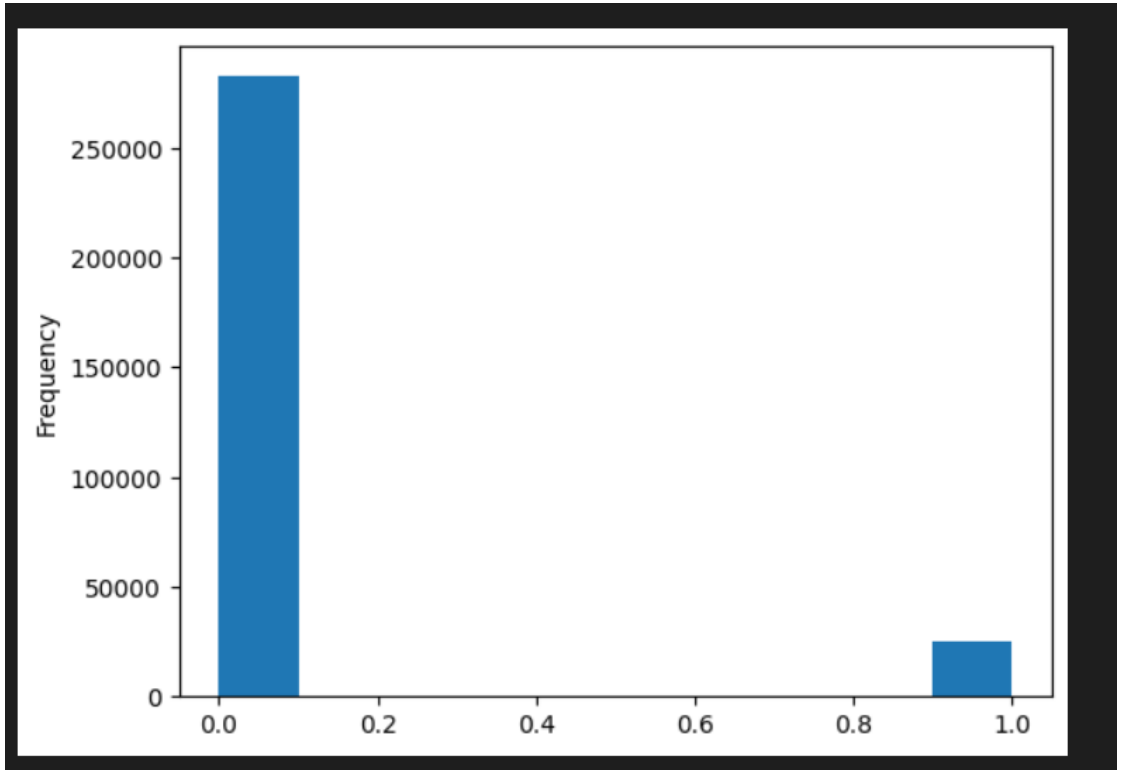
II.Quá trình thực hiện

1.Phân tích train/test data

Kích thước của các tập data

```
Training datashape: (307511, 122)
float64    65
int64      41
object     16
dtype: int64
Training datashape: (48744, 121)
float64    65
int64      40
object     16
dtype: int64
```

Phân bố của cột TARGET



2.Xử lý train/test data

Xoá các cột mã hoá căn hộ của khách hàng

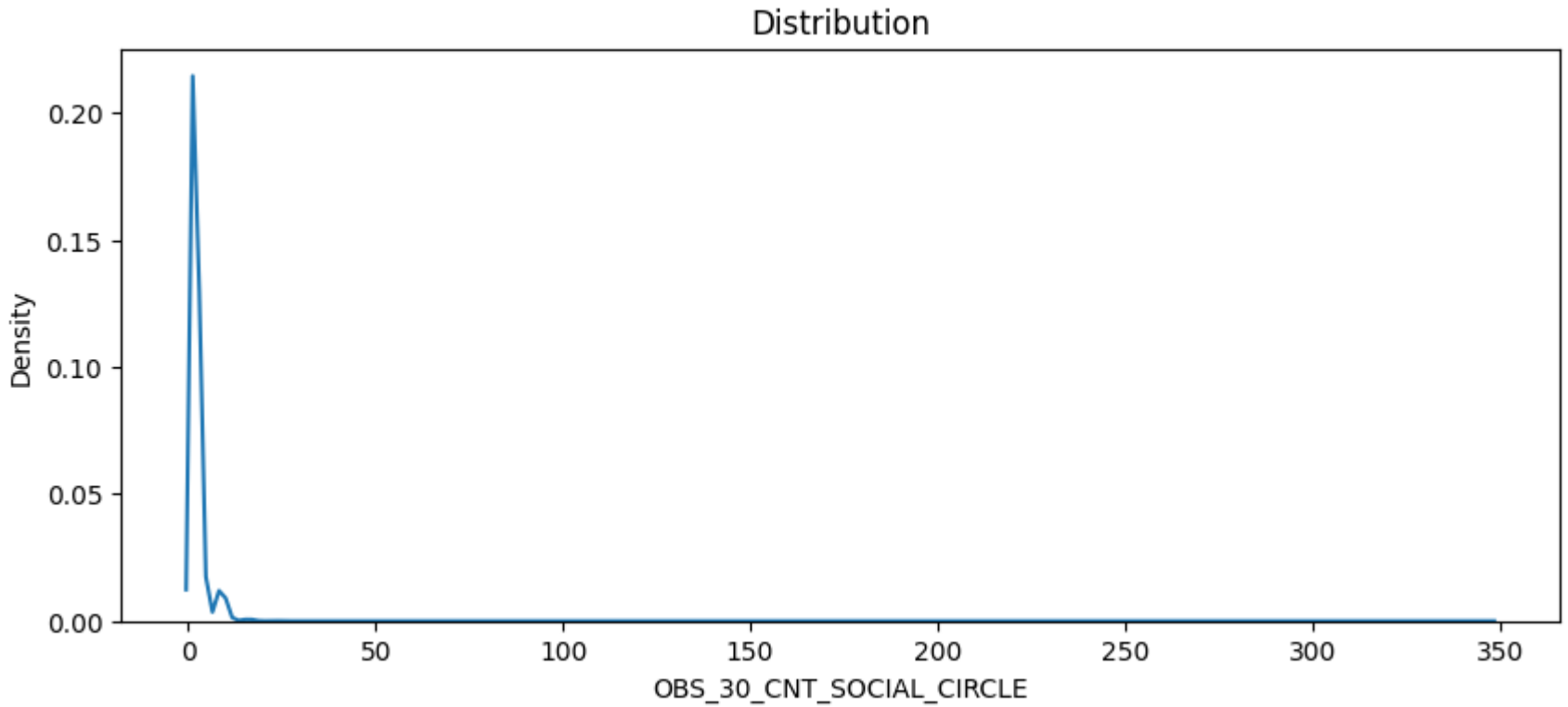
```
del_col = []
for i in application_train.columns:
    if i.find('_AVG')!=-1:
        del_col.append(i)
    if i.find('_MODE')!=-1:
        del_col.append(i)
    if i.find('_MEDI')!=-1:
        del_col.append(i)
application_train = application_train.drop(columns=del_col)
application_test = application_test.drop(columns=del_col)
```

Xoá các giá trị xuất hiện trên tập train mà không có trên tập test, xử lý một vài outlier

```
#Remove some rows with values not present in test set

application_train.drop(application_train[application_train['CODE_GENDER'] == 'XNA'].index, inplace=True)
application_train.drop(application_train[application_train['NAME_INCOME_TYPE'] == 'Maternity leave'].index, inplace=True)
application_train.drop(application_train[application_train['NAME_FAMILY_STATUS'] == 'Unknown'].index, inplace=True)

#Replace some outliers
application_train['DAYS_EMPLOYED'].replace(365243, np.nan, inplace=True)
application_train.loc[application_train['OWN_CAR_AGE'] > 80, 'OWN_CAR_AGE'] = np.nan
application_train.loc[application_train['REGION_RATING_CLIENT_W_CITY'] < 0, 'REGION_RATING_CLIENT_W_CITY'] = np.nan
application_train.loc[application_train['AMT_INCOME_TOTAL'] > 1e8, 'AMT_INCOME_TOTAL'] = np.nan
application_train.loc[application_train['AMT_REQ_CREDIT_BUREAU_QRT'] > 80, 'AMT_REQ_CREDIT_BUREAU_QRT'] = np.nan
application_train.loc[application_train['OBS_30_CNT_SOCIAL_CIRCLE'] > 80, 'OBS_30_CNT_SOCIAL_CIRCLE'] = np.nan
```




```
application_train['OBS_30_CNT_SOCIAL_CIRCLE'].sort_values(ascending = False)
```

✓ 0.6s

```
148403    348.0
77497      47.0
280641     30.0
```

2.Xử lý train/test data

- Feature engineering

```
application_train['age'] = application_train['DAYS_BIRTH'] / -365
application_train.loc[application_train['DAYS_EMPLOYED'] == 365243, 'DAYS_EMPLOYED'] = np.nan
application_train['years_employed'] = application_train['DAYS_EMPLOYED'] / -365
application_train['AMT_CREDIT / AMT_ANNUITY'] = application_train['AMT_CREDIT'] / application_train['AMT_ANNUITY']
application_train['EXT_SOURCE mean'] = application_train[['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3']].mean(axis = 1)
application_train['AMT_INCOME_TOTAL / 12 - AMT_ANNUITY'] = application_train['AMT_INCOME_TOTAL'] / 12. - application_train['AMT_ANNUITY']
application_train['AMT_INCOME_TOTAL / AMT_ANNUITY'] = application_train['AMT_INCOME_TOTAL'] / application_train['AMT_ANNUITY']
application_train['AMT_INCOME_TOTAL - AMT_GOODS_PRICE'] = application_train['AMT_INCOME_TOTAL'] - application_train['AMT_GOODS_PRICE']
application_train['DAYS_EMPLOYED_PERC'] = application_train['DAYS_EMPLOYED'] / application_train['DAYS_BIRTH']
application_train['INCOME_CREDIT_PERC'] = application_train['AMT_INCOME_TOTAL'] / application_train['AMT_CREDIT']
application_train['INCOME_PER_PERSON'] = application_train['AMT_INCOME_TOTAL'] / application_train['CNT_FAM_MEMBERS']
application_train['ANNUITY_INCOME_PERC'] = application_train['AMT_ANNUITY'] / application_train['AMT_INCOME_TOTAL']
application_train['PAYMENT_RATE'] = application_train['AMT_ANNUITY'] / application_train['AMT_CREDIT']
application_train['NEW_INCOME_BY_ORG'] = application_train['ORGANIZATION_TYPE'].map(income_by_organi_train)
application_train['NEW_INCOME_BY_OCC'] = application_train['ORGANIZATION_TYPE'].map(income_by_occupa_train)
application_train['NEW_EXT_SOURCES_MUL'] = application_train['EXT_SOURCE_1'] * application_train['EXT_SOURCE_2'] * application_train['EXT_SOURCE_3']
application_train['NEW_EXT_SOURCES_MEAN'] = application_train[['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3']].mean(axis=1)
application_train['AMT_INCOME_TOTAL'] = np.log1p(application_train['AMT_INCOME_TOTAL'])
application_train['AMT_CREDIT'] = np.log1p(application_train['AMT_CREDIT'])
```

2.Xử lý train/test data

- In ra số class ở mỗi categorical feature

```
other_col.select_dtypes('object').apply(pd.Series.nunique, axis = 0)
```

✓ 0.1s

```
NAME_CONTRACT_TYPE            2
CODE_GENDER                   2
FLAG_OWN_CAR                  2
FLAG_OWN_REALTY               2
NAME_TYPE_SUITE               7
NAME_INCOME_TYPE              7
NAME_EDUCATION_TYPE           5
NAME_FAMILY_STATUS            5
NAME_HOUSING_TYPE             6
WEEKDAY_APPR_PROCESS_START    7
dtype: int64
```

- Dùng label encoding cho các feature có số trường <=2

```
# Create a label encoder object
le = LabelEncoder()
le_count = 0

# Iterate through the columns
for col in other_col:
    if other_col[col].dtype == 'object':
        # If 2 or fewer unique categories
        if len(list(other_col[col].unique())) <= 2:
            # Train on the training data
            le.fit(other_col[col])
            # Transform both training and testing data
            other_col[col] = le.transform(other_col[col])
            application_test[col] = le.transform(application_test[col])

            # Keep track of how many columns were label encoded
            le_count += 1

print('%d columns were label encoded.' % le_count)
```

✓ 0.3s

```
4 columns were label encoded.
```

- One-hot encoding cho các feature còn lại

```
categorical_features = [col for col in other_col.columns if other_col[col].dtype == 'object']

one_hot_df = pd.concat([other_col,application_test])
one_hot_df = pd.get_dummies(one_hot_df, columns=categorical_features)

other_col = one_hot_df.iloc[:other_col.shape[0],:]
application_test = one_hot_df.iloc[other_col.shape[0]:,]
```

✓ 0.6s

- Phân tích missing data sau khi chọn feature xong:

```
total = application_train.isnull().sum().sort_values(ascending= False)
percent = (application_train.isnull().sum()/application_train.isnull().count()*100).sort_values(ascending= False)
missing_application_train_data = pd.concat([total,percent],axis=1,keys=['Total','Percent'])
missing_application_train_data.head(60)
```

✓ 0.8s

	Total	Percent
OWN_CAR_AGE	202923	65.991220
NEW_EXT_SOURCES_MUL	197913	64.361951
EXT_SOURCE_1	173374	56.381789
NEW_INCOME_BY_OCC	96388	31.345691
EXT_SOURCE_3	60959	19.824065
DAYS_EMPLOYED	55374	18.007805
DAYS_EMPLOYED_PERC	55374	18.007805
years_employed	55374	18.007805
AMT_REQ_CREDIT_BUREAU_QRT	41515	13.500813
AMT_REQ_CREDIT_BUREAU_WEEK	41514	13.500488
AMT_REQ_CREDIT_BUREAU_HOUR	41514	13.500488

- Xử lý bằng cách impute bằng mean

```
from sklearn.impute import SimpleImputer
Imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
application_train[numerical_list] = Imputer.fit_transform(application_train[numerical_list])
application_test[numerical_list] = Imputer.fit_transform(application_test[numerical_list])
```

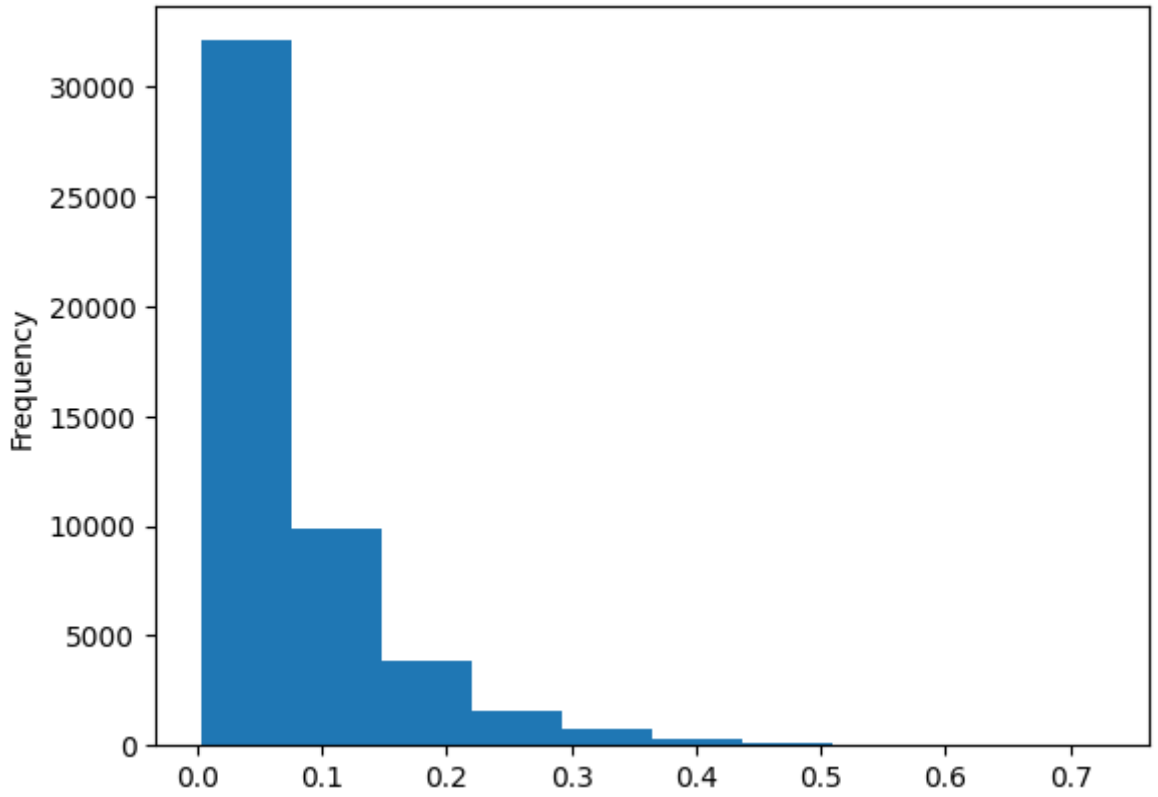
Model LightGBM

```
model = lgb.LGBMClassifier(nthread=4,
    n_estimators=10000,
    learning_rate=0.02,
    num_leaves=34,
    colsample_bytree=0.9497036,
    subsample=0.8715623,
    max_depth=8,
    reg_alpha=0.041545473,
    reg_lambda=0.0735294,
    min_split_gain=0.0222415,
    min_child_weight=39.3259775,
    silent=-1,
    verbose=-1,)

# Train the model
model.fit(train_features, train_labels, eval_metric = 'auc',
          eval_set = [(valid_features, valid_labels), (train_features, train_labels)],
          eval_names = ['valid', 'train'], categorical_feature = cat_indices,
          early_stopping_rounds = 200, verbose = 200)
```

Submission and Description	Private Score	Public Score	Use for Final Score
selected_features_submission.csv just now by Đức Trường 120 feature	0.76223	0.76361	☐

Histogram


Frequency
30000
25000
20000
15000
10000
5000
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

Những kiến thức đã học được

- I.KFold và StratifiedKFold

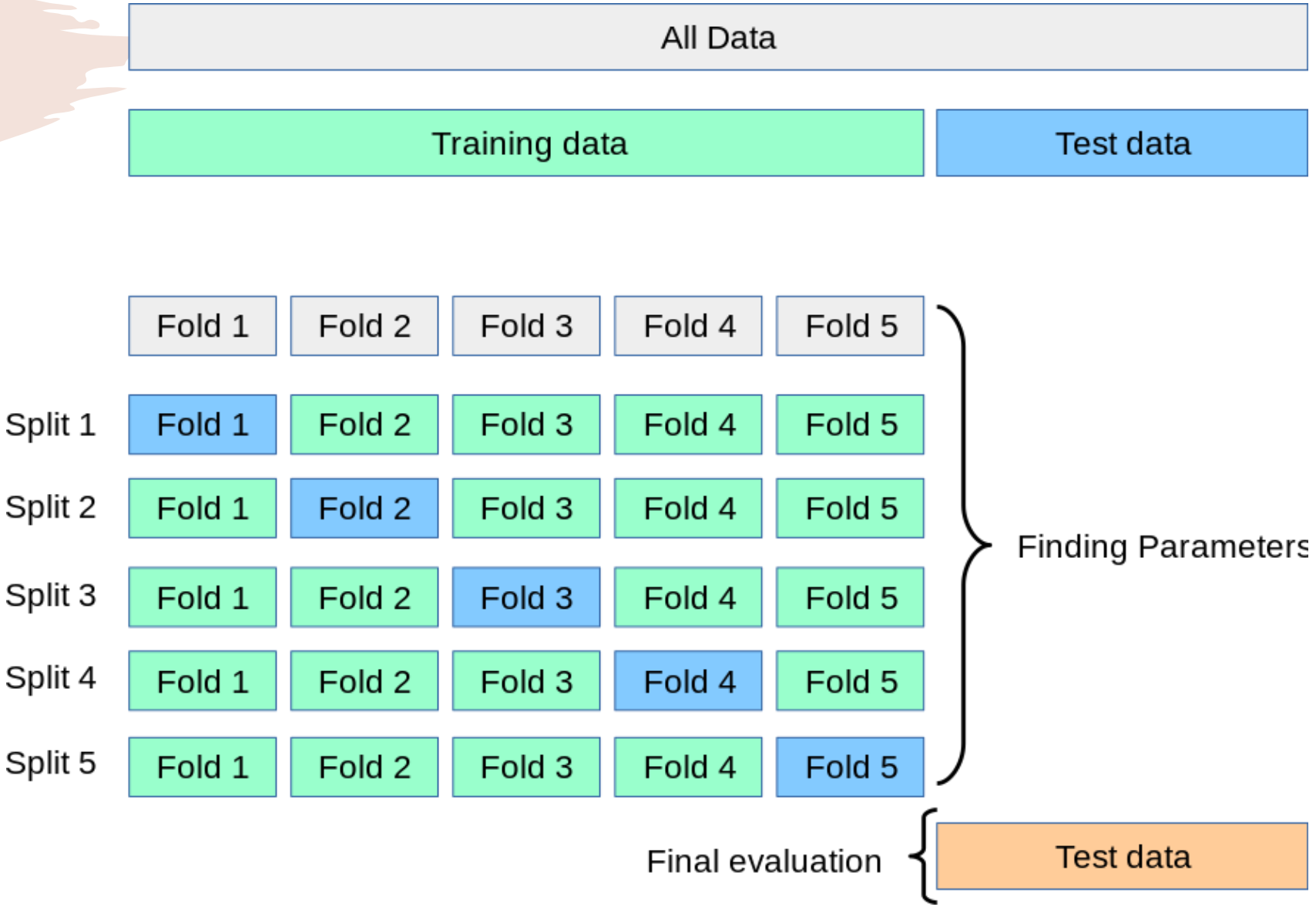
1.Tại sao phải dùng KFold?

-Khi có tập dữ liệu không đủ lớn thì việc chia tập train/val theo tỉ lệ 80/20 có thể dẫn đến việc model hoạt động cực kém.Vì có thể có một số điểm dữ liệu có ích trong quá trình train có thể bị đưa vào tập val dẫn đến model không có cơ hội học điểm dữ liệu đó.Đôi khi có một vài class chỉ có trong val hoặc test mà không có trong train dẫn đến kết quả không tốt khi test.

-> Vậy nên chúng ta cần đến KFold Cross Validation.

2.KFold là gì?

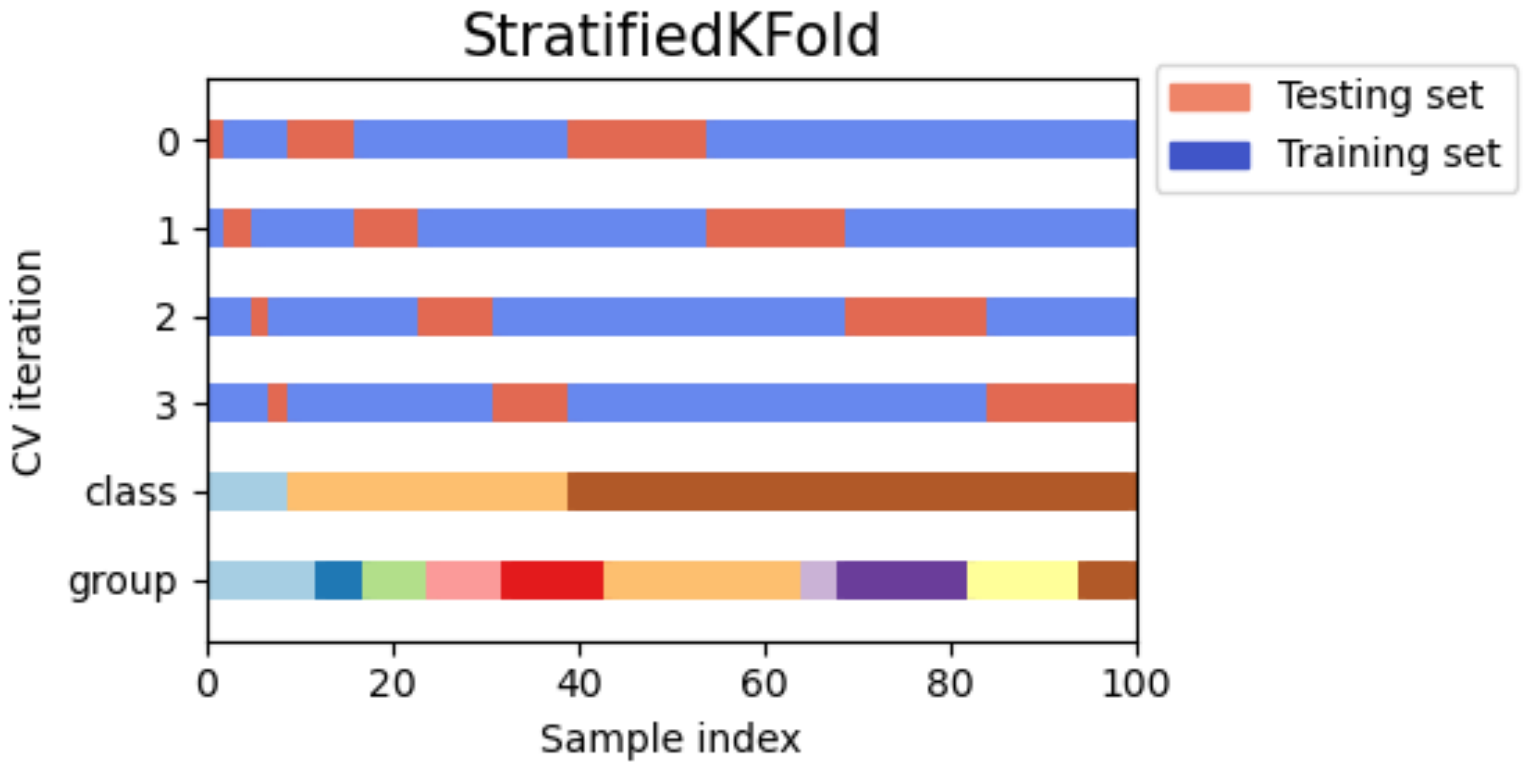
Phần dữ liệu Training thì sẽ được chia ngẫu nhiên thành K phần. Sau đó train model K lần, mỗi lần train sẽ chọn 1 phần làm dữ liệu validation và K-1 phần còn lại làm dữ liệu training. Kết quả đánh giá model cuối cùng sẽ là trung bình cộng kết quả đánh giá của K lần train. Đó chính là lý do vì sao ta đánh giá khách quan và chính xác hơn.


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

2. StratifiedKFold

-Khác với KFold thì Stratified KFold lấy mẫu theo cách chia sao cho tỷ lệ các class trong các fold là tương tự như nhau còn KFold sẽ lấy mẫu ngẫu nhiên

-Stratified KFold hữu ích hơn trong trường hợp các bài toán phân loại(Classification)



II.Integer encoding và One-hot encoding

1.Label encoding

-Đây là cách đưa giá trị dạng category về số,map mỗi category với một số nguyên

-Tuy nhiên điều này đã gán các category này một giá trị làm dẫn đến việc thiếu chính xác khi tính toán mô hình

2.One-hot encoding

-Là một trong những cách encode phổ biến.

-Mỗi giá trị category sẽ tương ứng với một one-hot vector với k phần tử khác nhau, với mỗi một giá trị khác nhau của categorical feature, chúng ta sẽ tạo ra một feature mới.

-Mỗi category mới này sẽ được gán cho một giá trị là 0 hoặc 1. Nếu giá trị thuộc category nào thì giá trị ở đó sẽ là 1.

Data Visualization-các dạng biểu đồ thường gặp

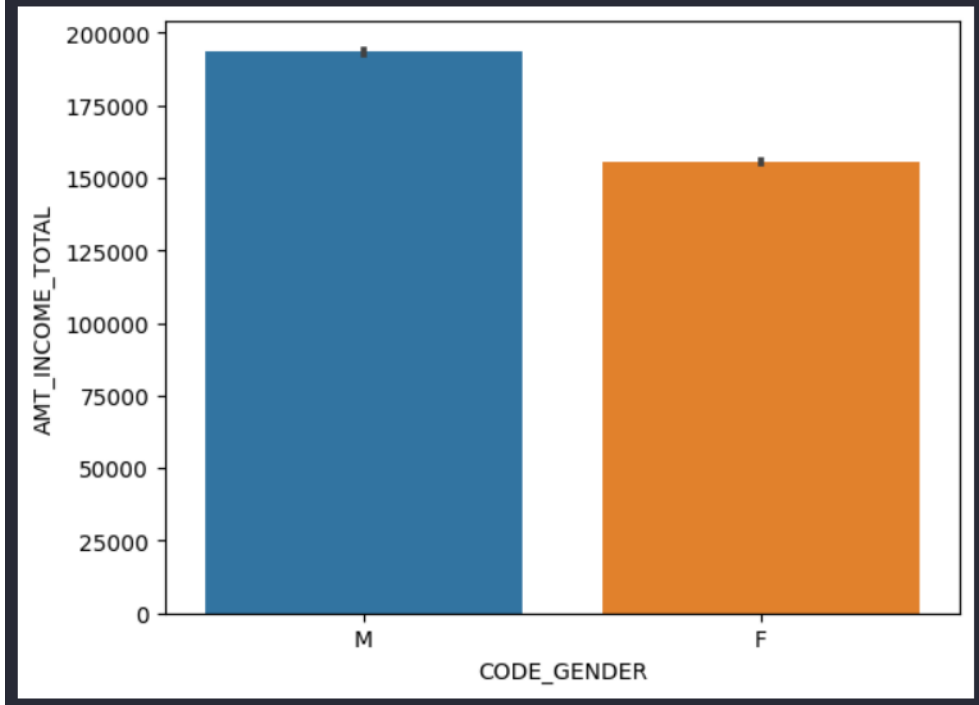
1.Bar Chart (Biểu đồ cột)

-Thường được dùng để so sánh số liệu giữa các categories

```
sns.barplot(application_train, x='CODE_GENDER',y='AMT_INCOME_TOTAL',estimator=np.mean)
```

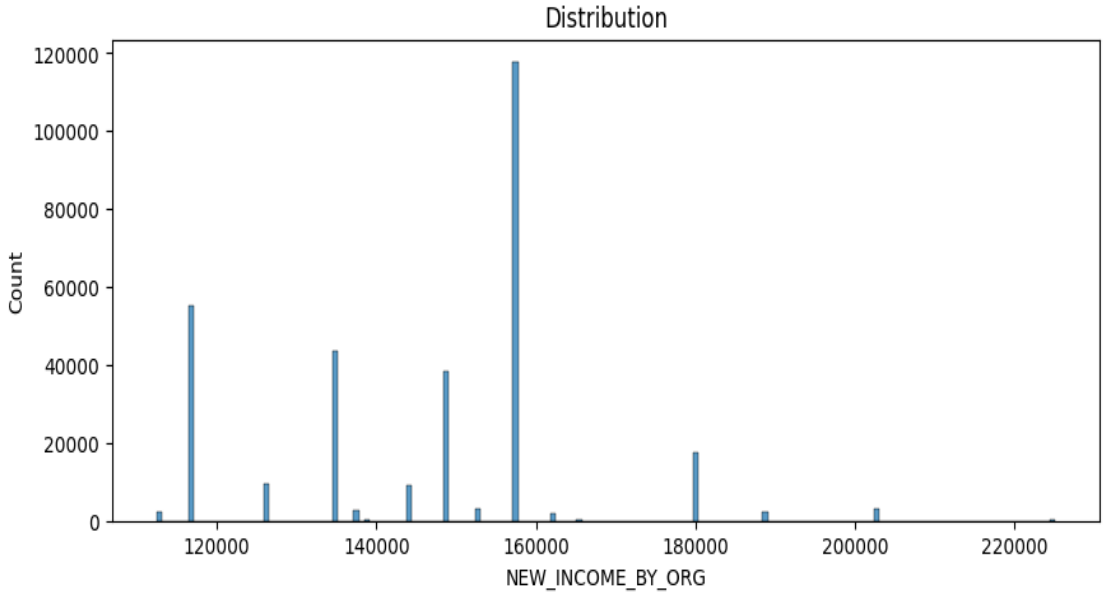
✓ 3.1s

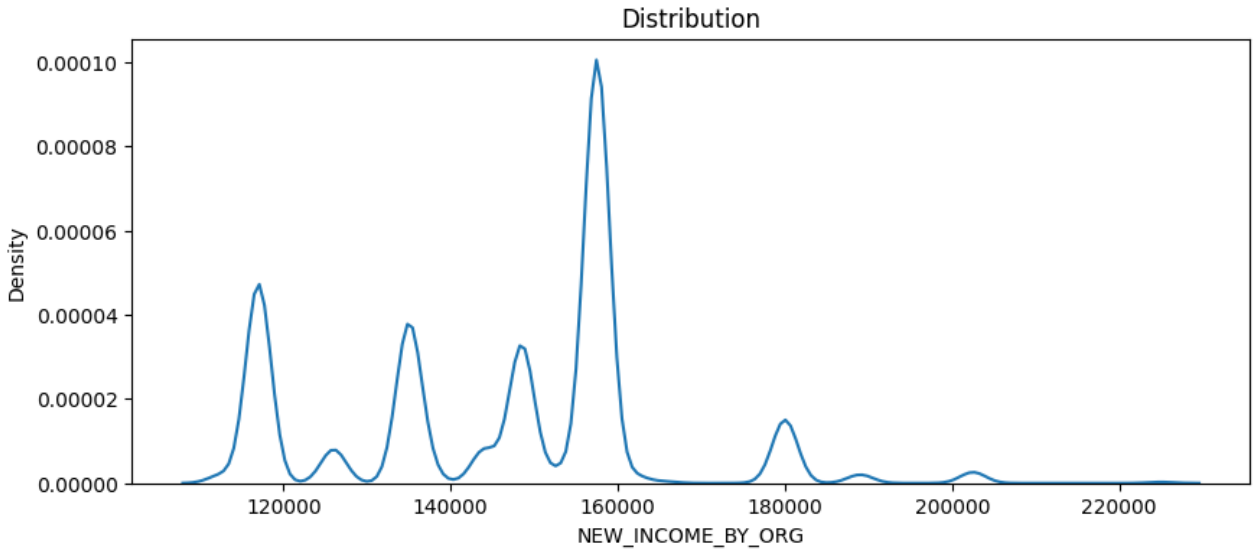
```
<AxesSubplot: xlabel='CODE_GENDER', ylabel='AMT_INCOME_TOTAL'>
```



2.Histogram – KDE plot

-Thường được sử dụng để biểu diễn sự phân bố của một mẫu dữ liệu





3.Box Plot

-Thường được sử dụng để biểu diễn tóm tắt sự phân bố của các mẫu dữ liệu

-Trong đó hình hộp là thể hiện cho khoảng 50% giá trị của mẫu, bắt đầu từ điểm 25% và kết thúc ở điểm 75%

```
sns.boxplot(application_train, x='CODE_GENDER',y='AMT_CREDIT')
```

✓ 0.2s

```
<AxesSubplot: xlabel='CODE_GENDER', ylabel='AMT_CREDIT'>
```