



application_{train|test}.csv

- Main tables – our train and test samples
- Target (binary)
- Info about loan and loan applicant at application time

bureau.csv

- Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
- One row per client's loan in Credit Bureau

SK_ID_BUREAU

bureau_balance.csv

- Monthly balance of credits in Credit Bureau
- Behavioral data

previous_application.csv

- Application data of client's previous loans in Home Credit
- Info about the previous loan parameters and client info at time of previous application
- One row per previous application

SK_ID_PREV

SK_ID_PREV

POS_CASH_balance.csv

- Monthly balance of client's previous loans in Home Credit
- Behavioral data

instalments_payments.csv

- Past payment data for each installments of previous credits in Home Credit related to loans in our sample
- Behavioral data

credit_card_balance.csv

- Monthly balance of client's previous credit card loans in Home Credit
- Behavioral data

SK_ID_CURR

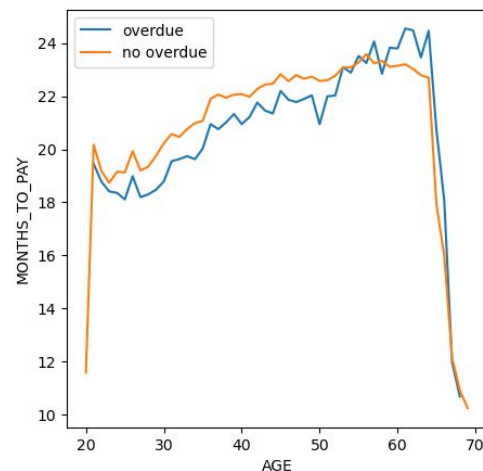
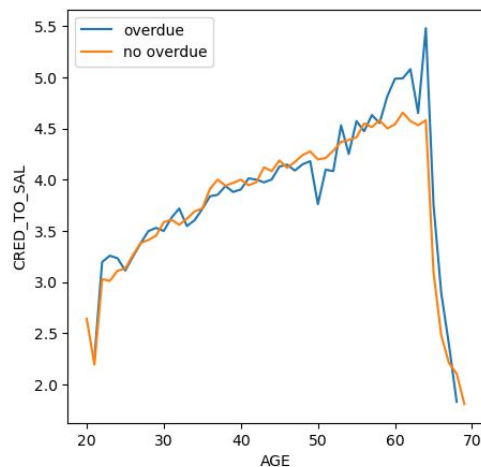
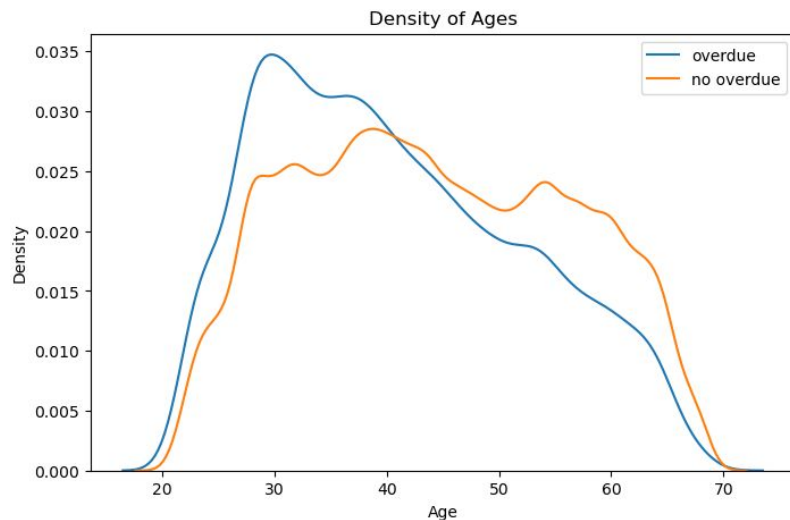
SK_ID_CURR

SK_ID_CURR

SK_ID_CURR

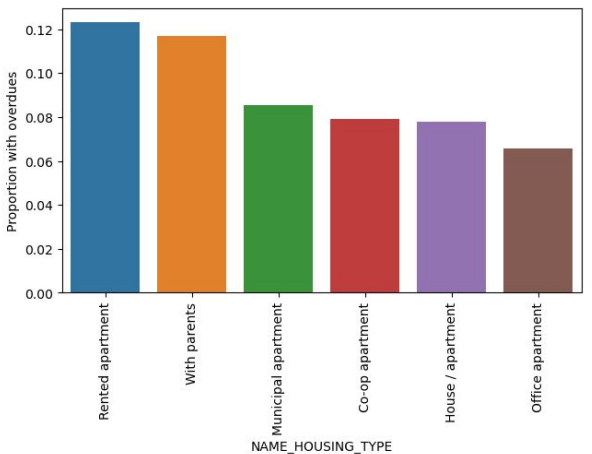
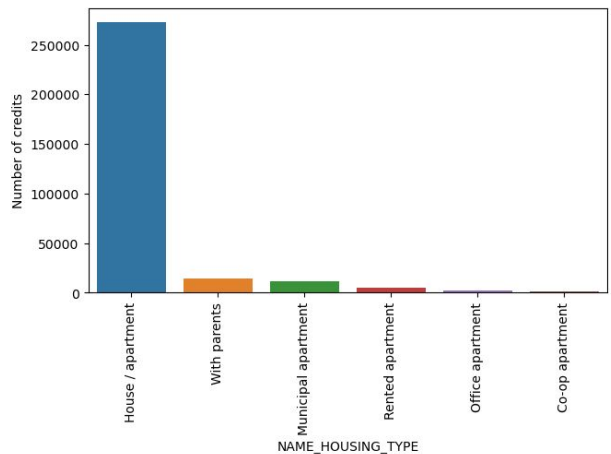
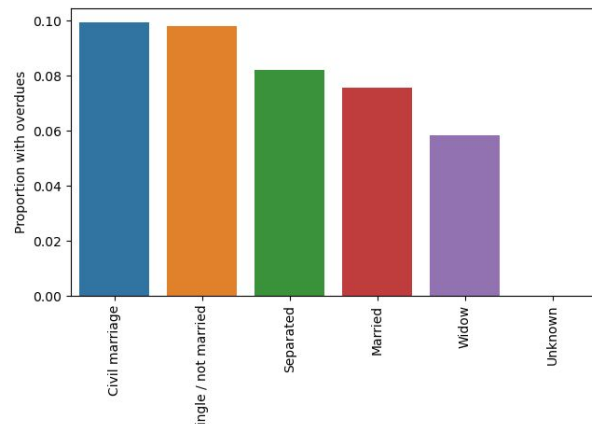
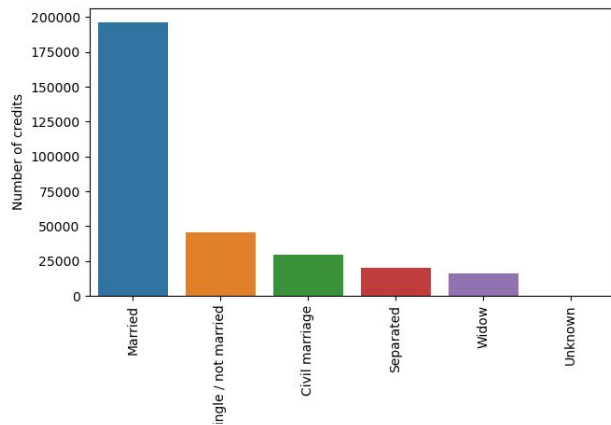
Предварительный анализ (возраст)

- Как можно увидеть, среди молодых людей чаще встречается просрочка по кредиту
- На втором графике представлено отношение размера кредита к зарплате. Можно заметить всплеск в районе 65 лет.
- Такой же всплеск и для количества месяцев на выплату кредита. Это может быть связано с преклонным возрастом и риском внезапной смерти кредитора.



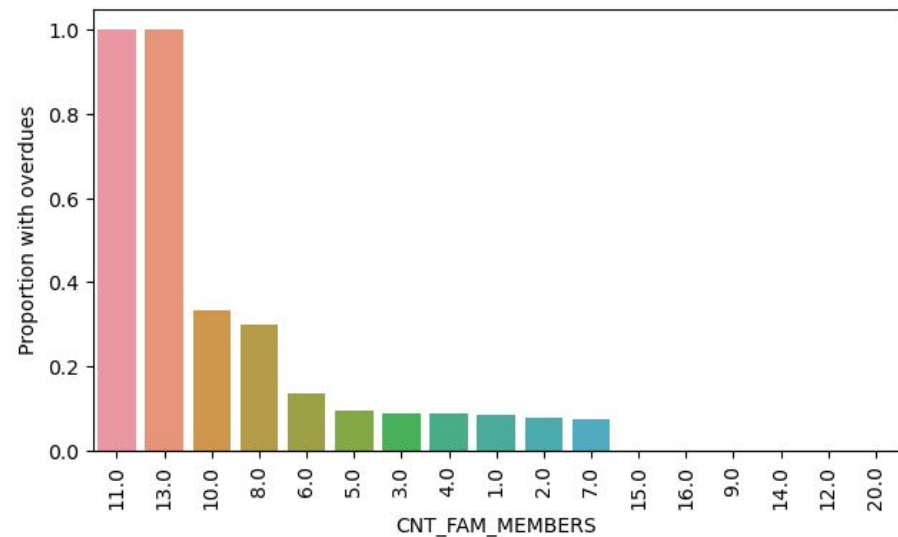
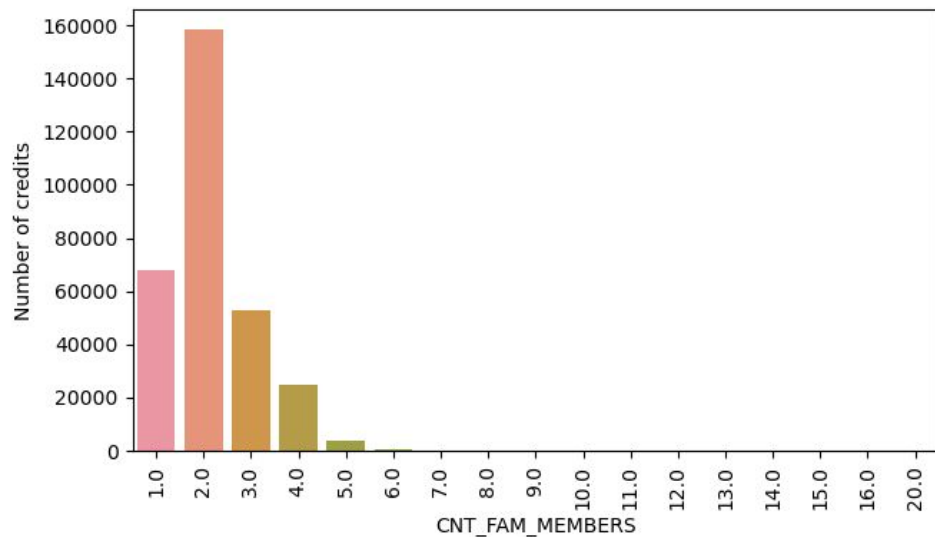
Предварительный анализ(семейный статус, тип жилья)

- Вдовы берут меньше кредитов, чем остальные группы, но при этом просрочивают на уровне других групп
- Люди, которые живут в со-ор берут меньше всего кредитов, но при этом просрочивают их на таком же уровне



Предварительный анализ (размер семьи)

- Люди с большими семьями берут мало кредитов, но при этом просрочивают выплаты по ним чаще



Обработка application_train / application_test / bureau

Поиск None значений:

```
result_train = pd.concat([application_train.isnull().sum(), round(application_train.isnull().mean() * 100, 3)], axis=1)
result_train = result_train.rename(index=str, columns={0: 'total missing', 1: 'proportion'})
result_train = result_train.apply(lambda x: x[x > 0])
result_train = result_train.sort_values('proportion')

pd.set_option('display.max_rows', None)
result_train
```

✓ 0.2s

	total missing	proportion
CNT_FAM_MEMBERS	2	0.001
AMT_ANNUITY	12	0.004
AMT_GOODS_PRICE	278	0.090
EXT_SOURCE_2	660	0.215
OBS_60_CNT_SOCIAL_CIRCLE	1021	0.332
DEF_30_CNT_SOCIAL_CIRCLE	1021	0.332
DEF_60_CNT_SOCIAL_CIRCLE	1021	0.332
OBS_30_CNT_SOCIAL_CIRCLE	1021	0.332
NAME_TYPE_SUITE	1292	0.420
AMT_REQ_CREDIT_BUREAU_WEEK	41519	13.502
AMT_REQ_CREDIT_BUREAU_QRT	41519	13.502
AMT_REQ_CREDIT_BUREAU_MON	41519	13.502
AMT_REQ_CREDIT_BUREAU_HOUR	41519	13.502
AMT_REQ_CREDIT_BUREAU_DAY	41519	13.502
AMT_REQ_CREDIT_BUREAU_YEAR	41519	13.502
EXT_SOURCE_3	60965	19.825
OCCUPATION_TYPE	96391	31.346
EMERGENCYSTATE_MODE	145755	47.398
TOTALAREA_MODE	148431	48.269
YEARS_BEGINEXPLUATATION_AVG	150007	48.781
YEARS_BEGINEXPLUATATION_MEDI	150007	48.781
YEARS_BEGINEXPLUATATION_MODE	150007	48.781

Работа с категориальными фичами

```
✓ pd.set_option('display.max_rows', None)
  for column in application_train.select_dtypes(include='object').columns:
    print(column, " ", application_train[column].unique(), "\n")
✓ 0.1s
```

NAME_CONTRACT_TYPE ['Cash loans' 'Revolving loans']

CODE_GENDER ['M' 'F' 'XNA']

FLAG_OWN_CAR ['N' 'Y']

FLAG_OWN_REALTY ['Y' 'N']

NAME_TYPE_SUITE ['Unaccompanied' 'Family' 'Spouse, partner' 'Children' 'Other_A' nan
'Other_B' 'Group of people']

NAME_INCOME_TYPE ['Working' 'State servant' 'Commercial associate' 'Pensioner' 'Unemployed'
'Student' 'Businessman' 'Maternity leave']

NAME_EDUCATION_TYPE ['Secondary / secondary special' 'Higher education' 'Incomplete higher'
'Lower secondary' 'Academic degree']

NAME_FAMILY_STATUS ['Single / not married' 'Married' 'Civil marriage' 'Widow' 'Separated'
'Unknown']

NAME_HOUSING_TYPE ['House / apartment' 'Rented apartment' 'With parents'
'Municipal apartment' 'Office apartment' 'Co-op apartment']

OCCUPATION_TYPE ['Laborers' 'Core staff' 'Accountants' 'Managers' nan 'Drivers'
'Sales staff' 'Cleaning staff' 'Cooking staff' 'Private service staff']

...

'Trade: type 1' 'Industry: type 5' 'Industry: type 10' 'Legal Services'
'Advertising' 'Trade: type 5' 'Cleaning' 'Industry: type 13'
'Trade: type 4' 'Telecom' 'Industry: type 8' 'Realtor' 'Industry: type 6']

то, что имеет значения да/нет, заменим на 0/1

```
application_train['NAME_CONTRACT_TYPE'] = application_train['NAME_CONTRACT_TYPE'].apply(lambda x: int(1) if x == 'Y' else int(0))
application_train['FLAG_OWN_CAR'] = application_train['FLAG_OWN_CAR'].apply(lambda x: int(1) if x == 'Y' else int(0))
application_train['FLAG_OWN_REALTY'] = application_train['FLAG_OWN_REALTY'].apply(lambda x: int(1) if x == 'Y' else int(0))
application_train['CODE_GENDER'] = application_train['CODE_GENDER'].apply(lambda x: int(1) if x == 'M' else (int(0) if x == 'F' else int(np.random.rand(1) < 0.5)))

application_test['NAME_CONTRACT_TYPE'] = application_test['NAME_CONTRACT_TYPE'].apply(lambda x: int(1) if x == 'Y' else int(0))
application_test['FLAG_OWN_CAR'] = application_test['FLAG_OWN_CAR'].apply(lambda x: int(1) if x == 'Y' else int(0))
application_test['FLAG_OWN_REALTY'] = application_test['FLAG_OWN_REALTY'].apply(lambda x: int(1) if x == 'Y' else int(0))
application_test['CODE_GENDER'] = application_test['CODE_GENDER'].apply(lambda x: int(1) if x == 'M' else (int(0) if x == 'F' else int(np.random.rand(1) < 0.5)))
```

[53]

✓ 0.3s

Py

mean_target_encoding

```
def add_noise(series, noise_level):
    return series * (1 + noise_level * np.random.randn(len(series)))

def target_encode(trn_series=None,
                  tst_series=None,
                  target=None,
                  min_samples_leaf=1,
                  smoothing=1,
                  noise_level=0):
    """
    Smoothing is computed like in the following paper by Daniele Micci-Barreca
    https://kaggle2.blob.core.windows.net/forum-message-attachments/225952/7441/high%20cardinality%20categoricals.pdf
    trn_series : training categorical feature as a pd.Series
    tst_series : test categorical feature as a pd.Series
    target : target data as a pd.Series
    min_samples_leaf (int) : minimum samples to take category average into account
    smoothing (int) : smoothing effect to balance categorical average vs prior
    """
    assert len(trn_series) == len(target)
    assert trn_series.name == tst_series.name
    temp = pd.concat([trn_series, target], axis=1)
    # Compute target mean
    averages = temp.groupby(by=trn_series.name)[target.name].agg(["mean", "count"])
    # Compute smoothing
    smoothing = 1 / (1 + np.exp(-(averages["count"] - min_samples_leaf) / smoothing))
    # Apply average function to all target data
    prior = target.mean()
    # The bigger the count the less full_avg is taken into account
    averages[target.name] = prior * (1 - smoothing) + averages["mean"] * smoothing
    averages.drop(["mean", "count"], axis=1, inplace=True)
    # Apply averages to trn and tst series
    ft_trn_series = pd.merge(
        trn_series.to_frame(trn_series.name),
        averages.reset_index().rename(columns={'index': target.name, target.name: 'average'}),
        on=trn_series.name,
        how='left')['average'].rename(trn_series.name + '_mean').fillna(prior)
    # pd.merge does not keep the index so restore it
    ft_trn_series.index = trn_series.index
    ft_tst_series = pd.merge(
        tst_series.to_frame(tst_series.name),
        averages.reset_index().rename(columns={'index': target.name, target.name: 'average'}),
        on=tst_series.name,
        how='left')['average'].rename(trn_series.name + '_mean').fillna(prior)
    # pd.merge does not keep the index so restore it
    ft_tst_series.index = tst_series.index
    return add_noise(ft_trn_series, noise_level), add_noise(ft_tst_series, noise_level)
```

Заполним None средним

```
result_train = pd.concat([application_train.isnull().sum(), round(application_train.isnull().mean() * 100, 3)], axis=1)
result_train = result_train.rename(index=str, columns={0: 'total missing', 1: 'proportion'})
result_train = result_train.apply(lambda x: x[x > 0])
result_train = result_train.sort_values('proportion')
```

```
pd.set_option('display.max_rows', None)
result_train
```

✓ 0.0s

	total missing	proportion
CNT_FAM_MEMBERS	2	0.001
AMT_ANNUITY	12	0.004
AMT_GOODS_PRICE	278	0.090
EXT_SOURCE_2	660	0.215
OBS_30_CNT_SOCIAL_CIRCLE	1021	0.332
DEF_30_CNT_SOCIAL_CIRCLE	1021	0.332
OBS_60_CNT_SOCIAL_CIRCLE	1021	0.332
DEF_60_CNT_SOCIAL_CIRCLE	1021	0.332
AMT_REQ_CREDIT_BUREAU_HOUR	41519	13.502
AMT_REQ_CREDIT_BUREAU_DAY	41519	13.502
AMT_REQ_CREDIT_BUREAU_WEEK	41519	13.502
AMT_REQ_CREDIT_BUREAU_MON	41519	13.502
AMT_REQ_CREDIT_BUREAU_QRT	41519	13.502
AMT_REQ_CREDIT_BUREAU_YEAR	41519	13.502
EXT_SOURCE_3	60965	19.825

Таблица bureau

bureau.head()

[59]

✓ 0.0s

Python

...

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_DAY_OVERDUE	DAYS_CREDIT_ENDDATE	DAYS_ENDDATE_FACT	AMT_CREDIT_MAX_OVERDUE	CNT_CREDIT_PROLONG	AMT_CREDIT_SUM	AMT_CREDIT_SUM_DEB
0	215354	5714462	Closed	currency 1	-497	0	-153.0	-153.0	NaN	0	91323.0	0.
1	215354	5714463	Active	currency 1	-208	0	1075.0	NaN	NaN	0	225000.0	171342.
2	215354	5714464	Active	currency 1	-203	0	528.0	NaN	NaN	0	464323.5	Na
3	215354	5714465	Active	currency 1	-203	0	NaN	NaN	NaN	0	90000.0	Na
4	215354	5714466	Active	currency 1	-629	0	1197.0	NaN	77674.5	0	2700000.0	Na

bureau.shape

[60]

✓ 0.0s

Python

...

(1716428, 17)

Фичи (bureau) -> добавляем в train и test (aplication)

Создадим таблицу, где будет ID, а также 2 столбца Active и Closed, что соответствует количеству открытых кредитов и погашенных. Так же 2 столбца со среденй суммой закрытых кредитов и средней открытых + максимальный закрытй + максимальный открытый + минимальный открытый + минимальный закрытый.

Начнем с конца:

```
helper = features_from_bureau.groupby(by=['SK_ID_CURR', 'CREDIT_ACTIVE'])['AMT_CREDIT_SUM'].agg(['mean', 'min', 'max'])

mean_active_col = helper.xs('Active', level='CREDIT_ACTIVE')['mean']
mean_closed_col = helper.xs('Closed', level='CREDIT_ACTIVE')['mean']

min_active_col = helper.xs('Active', level='CREDIT_ACTIVE')['min']
max_active_col = helper.xs('Active', level='CREDIT_ACTIVE')['max']

min_closed_col = helper.xs('Closed', level='CREDIT_ACTIVE')['min']
max_closed_col = helper.xs('Closed', level='CREDIT_ACTIVE')['max']

# mean_active_col.shape, mean_closed_col.shape, min_active_col.shape, max_active_col.shape, min_closed_col.shape, max_c'
helper.head()
```

		mean	min	max
SK_ID_CURR	CREDIT_ACTIVE			
100001	Active	294675.000	168345.0	378000.0
	Closed	142335.000	85500.0	279720.0
100004	Closed	94518.900	94500.0	94537.8
100011	Closed	108807.075	54000.0	145242.0

Фичи (bureau) -> добавляем в train и test (aplication)

```
test = pd.merge(mean_closed_col, mean_active_col, on='SK_ID_CURR', how='left')
test = pd.merge(test, min_active_col, on='SK_ID_CURR', how='left')
test = pd.merge(test, max_active_col, on='SK_ID_CURR', how='left')
test = pd.merge(test, min_closed_col, on='SK_ID_CURR', how='left')
test = pd.merge(test, max_closed_col, on='SK_ID_CURR', how='left')

rename_dict = {'mean_x': 'mean_Closed', 'mean_y': 'mean_Active',
               'min_x': 'min_Active', 'max_x': 'max_Active',
               'min_y': 'min_Closed', 'max_y': 'max_Closed',
               }

test = test.rename(columns=rename_dict)
test = test.fillna(0)
test.head()
```



mean_Closed mean_Active min_Active max_Active min_Closed max_Closed

SK_ID_CURR

100001	142335.0	294675.0000	168345.000	378000.0	85500.0	279720.0
100002	63844.5	240994.2825	31988.565	450000.0	0.0	135000.0
100003	69133.5	810000.0000	810000.000	810000.0	22248.0	112500.0
100004	94518.9	0.0000	0.000	0.0	94500.0	94537.8
100005	58500.0	299313.0000	29826.000	568800.0	58500.0	58500.0

Обработка корреляции

Обработка корреляции

```
corrmat = application_train.corr()  
fig, ax = plt.subplots()  
fig.set_size_inches(70, 70)  
sns.heatmap(corrmat, cmap="YlGnBu", linewidths=.5, annot=True)
```

[67] ✓ 3.9s

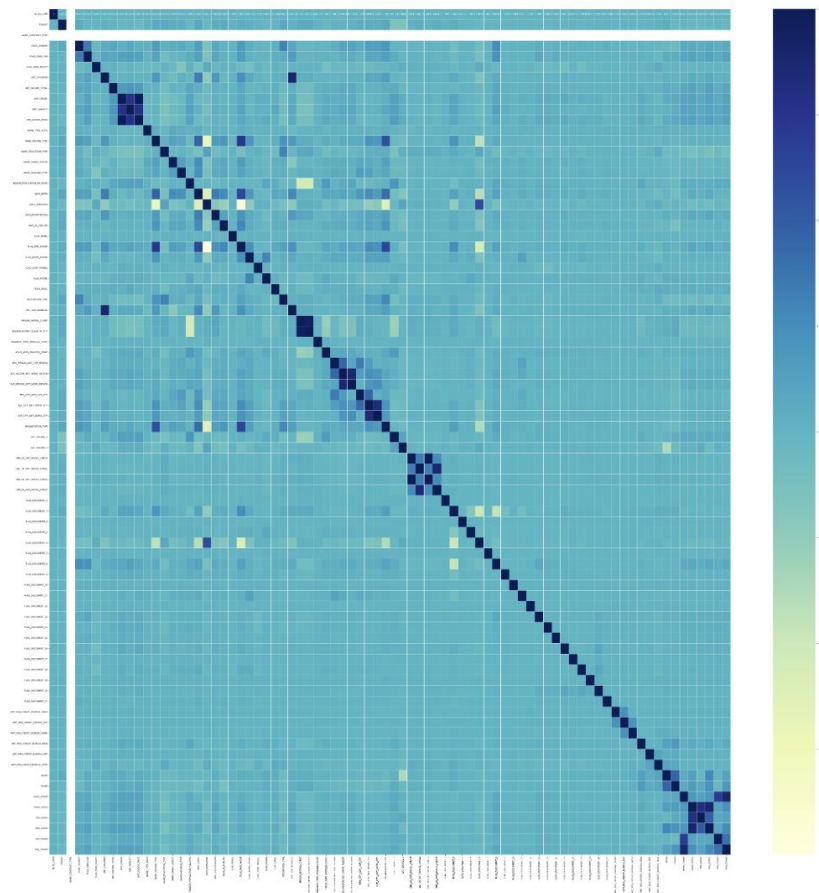
Python

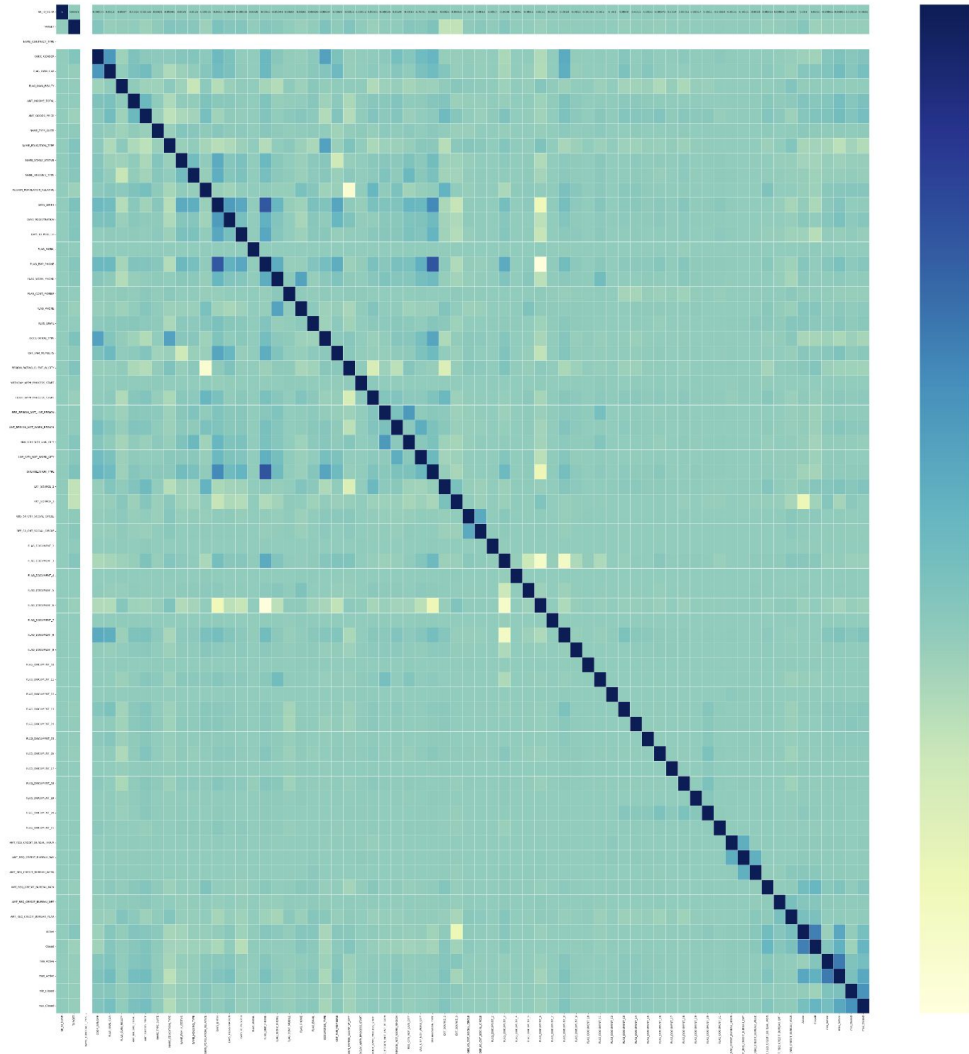
Outputs are collapsed ...

```
corr_matrix = application_train.corr().abs()  
  
# порог корреляции  
threshold = 0.7  
  
# создание маски для отбрасывания коррелирующих признаков  
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))  
corr_matrix_masked = corr_matrix.mask(mask)  
  
# отброс признаков с модулем корреляции больше порога  
to_drop = [column for column in corr_matrix_masked.columns if any(corr_matrix_masked[column] > threshold)]  
application_train = application_train.drop(to_drop, axis=1)
```

[68] ✓ 2.4s

Python





Фичи (installments_payments)

```
ins['NEW_DAYS_PAID_EARLIER'] = ins['DAYS_INSTALMENT']-ins['DAYS_ENTRY_PAYMENT']

ins['NEW_NUM_PAID_LATER'] = ins['NEW_DAYS_PAID_EARLIER'].map(lambda x: 1 if x<0 else 0)

agg_list = {'NUM_INSTALMENT_VERSION': ['nunique'],
            'NUM_INSTALMENT_NUMBER': ['max'],
            'DAYS_INSTALMENT': ['min', 'max'],
            'DAYS_ENTRY_PAYMENT': ['min', 'max'],
            'AMT_INSTALMENT': ['min', 'max', 'sum', 'mean'],
            'AMT_PAYMENT': ['min', 'max', 'sum', 'mean'],
            'NEW_DAYS_PAID_EARLIER': 'mean',
            'NEW_NUM_PAID_LATER': 'sum'}

ins_agg = ins.groupby('SK_ID_PREV').agg(agg_list)

ins_agg.columns = pd.Index(['INS_' + e[0] + '_' + e[1].upper() for e in ins_agg.columns.tolist()])

ins_agg.drop(['INS_DAYS_INSTALMENT_MIN',
              'INS_DAYS_INSTALMENT_MAX',
              'INS_DAYS_ENTRY_PAYMENT_MIN',
              'INS_DAYS_ENTRY_PAYMENT_MAX'],axis=1,inplace=True)

ins_agg['INS_NEW_PAYMENT_PERC'] = ins_agg['INS_AMT_PAYMENT_SUM'] / ins_agg['INS_AMT_INSTALMENT_SUM']
ins_agg['INS_NEW_PAYMENT_DIFF'] = ins_agg['INS_AMT_INSTALMENT_SUM'] - ins_agg['INS_AMT_PAYMENT_SUM']

agg_list_previous_application = {}

for col in ins_agg.columns:
    agg_list_previous_application[col] = ['mean', "min", "max", "sum"]

ins_agg.reset_index(inplace = True)

return agg_list_previous_application, ins_agg
```

- installments_payments.csv - содержит историю выплат по ранее выданным кредитам в Home Credit
- NEW_DAYS_PAID_EARLIER - показывает, насколько раньше был сделан платеж.
- NEW_NUM_PAID_LATER - 1, если платеж был сделан позже, и 0 иначе.
- Фичи агрегируются по SK_ID_PREV с по 'nunique', 'max', 'min', 'sum', 'mean'.
- Удаляется лишнее
- INS_NEW_PAYMENT_PERC - отношение суммы платежей к сумме взносов
- INS_NEW_PAYMENT_DIFF - разница между суммой взносов и суммой платежей.

Фичи (POS_CASH_balance)

```
pos = pd.read_csv('POS_CASH_balance.csv')
pos = pd.get_dummies(pos, columns=['NAME_CONTRACT_STATUS'], dummy_na = True)
agg_list = {'MONTHS_BALANCE': ['min', 'max'],
            'CNT_INSTALLMENT': ['min', 'max'],
            'CNT_INSTALLMENT_FUTURE': ['min', 'max'],
            'SK_DPD': ['max', 'mean'],
            'SK_DPD_DEF': ['max', 'mean'],
            'NAME_CONTRACT_STATUS_Active': 'sum',
            'NAME_CONTRACT_STATUS_Amortized debt': 'sum',
            'NAME_CONTRACT_STATUS_Approved': 'sum',
            'NAME_CONTRACT_STATUS_Canceled': 'sum',
            'NAME_CONTRACT_STATUS_Completed': 'sum',
            'NAME_CONTRACT_STATUS_Demand': 'sum',
            'NAME_CONTRACT_STATUS_Returned to the store': 'sum',
            'NAME_CONTRACT_STATUS_Signed': 'sum',
            'NAME_CONTRACT_STATUS_XNA': 'sum',
            'NAME_CONTRACT_STATUS_nan': 'sum'
            }

pos_agg = pos.groupby('SK_ID_PREV').agg(agg_list)

pos_agg.columns= pd.Index(['POS_' + e[0] + '_' + e[1].upper() for e in pos_agg.columns.tolist()])

pos_agg['POS_NEW_IS_CREDIT_NOT_COMPLETED_ON_TIME'] = (pos_agg['POS_CNT_INSTALLMENT_FUTURE_MIN']==0) & (pos_agg['POS_NAME_CONTRACT_STATUS_Completed_SUM']!=0)

pos_agg['POS_NEW_IS_CREDIT_NOT_COMPLETED_ON_TIME']=pos_agg['POS_NEW_IS_CREDIT_NOT_COMPLETED_ON_TIME'].astype(int)

pos_agg.drop(['POS_NAME_CONTRACT_STATUS_Approved_SUM',
              'POS_NAME_CONTRACT_STATUS_Amortized debt_SUM',
              'POS_NAME_CONTRACT_STATUS_Canceled_SUM',
              'POS_NAME_CONTRACT_STATUS_Returned to the store_SUM',
              'POS_NAME_CONTRACT_STATUS_Signed_SUM',
              'POS_NAME_CONTRACT_STATUS_XNA_SUM',
```

- POS_CASH_balance.csv - содержит месячные снэпшоты баланса предыдущих POS (оплат) и наличных кредитов, которые заявитель имел с Home Credit.
- Фичи агрегируются по SK_ID_PREV по 'max', 'min', 'sum', 'mean'.
- POS_NEW_IS_CREDIT_NOT_COMPLETE D_ON_TIME - указывает, был ли кредит завершен вовремя.
- Удаляется лишнее

Фичи (previous_application)

```
def previous_application(agg_list_previous_application):  
  
    df_prev = pd.read_csv('previous_application.csv')  
  
    df_prev["WEEKDAY_APPR_PROCESS_START"] = df_prev["WEEKDAY_APPR_PROCESS_START"].replace(['MONDAY', 'TUESDAY', 'WEDNESDAY', 'THURSDAY', 'FRIDAY'], 'WEEK_DAY')  
    df_prev["WEEKDAY_APPR_PROCESS_START"] = df_prev["WEEKDAY_APPR_PROCESS_START"].replace(['SATURDAY', 'SUNDAY'], 'WEEKEND')  
  
    a = [8,9,10,11,12,13,14,15,16,17]  
    df_prev["HOUR_APPR_PROCESS_START"] = df_prev["HOUR_APPR_PROCESS_START"].replace(a, 'working_hours')  
  
    b = [18,19,20,21,22,23,0,1,2,3,4,5,6,7]  
    df_prev["HOUR_APPR_PROCESS_START"] = df_prev["HOUR_APPR_PROCESS_START"].replace(b, 'off_hours')  
  
    df_prev["DAYS_DECISION"] = [1 if abs(i/(12*30)) <= 1 else 0 for i in df_prev.DAYS_DECISION]  
  
    df_prev["NAME_TYPE_SUITE"] = df_prev["NAME_TYPE_SUITE"].replace('Unaccompanied', 'alone')  
  
    b = ['Family', 'Spouse, partner', 'Children', 'Other_B', 'Other_A', 'Group of people']  
    df_prev["NAME_TYPE_SUITE"] = df_prev["NAME_TYPE_SUITE"].replace(b, 'not_alone')  
  
    a = ['Auto Accessories', 'Jewelry', 'Homewares', 'Medical Supplies', 'Vehicles', 'Sport and Leisure',  
        'Gardening', 'Other', 'Office Appliances', 'Tourism', 'Medicine', 'Direct Sales', 'Fitness', 'Additional Service',  
        'Education', 'Weapon', 'Insurance', 'House Construction', 'Animals']  
    df_prev["NAME_GOODS_CATEGORY"] = df_prev["NAME_GOODS_CATEGORY"].replace(a, 'others')  
  
    a = ['Auto technology', 'Jewelry', 'MLM partners', 'Tourism']  
    df_prev["NAME_SELLER_INDUSTRY"] = df_prev["NAME_SELLER_INDUSTRY"].replace(a, 'others')  
  
    df_prev["LOAN_RATE"] = df_prev.AMT_APPLICATION/df_prev.AMT_CREDIT
```

- previous_application.csv - содержит предыдущие заявки на кредиты.
- Дни недели, когда была начата процедура одобрения, группируются на рабочие /выходные + часы на рабочие/нерабочие.
- DAYS_DECISION - равен 1, если решение было принято в течение последнего года, и 0 в противном случае.
- Признаки NAME_TYPE_SUITE, NAME_GOODS_CATEGORY, NAME_SELLER_INDUSTRY упрощаются
- новый LOAN_RATE - отношение суммы заявки к сумме кредита
- NEW_INSURANCE - равен 1, если разница больше 0 (что указывает на наличие страховки), и 0, если разница меньше или равна 0. Если данные отсутствуют, устанавливается np.nan. Разница между суммой кредита (AMT_CREDIT) и стоимостью товара (AMT_GOODS_PRICE)
- Фичи агрегируются по SK_ID_CURR по 'max', 'min', 'sum', 'mean'.

Далее они мержаются по SK_ID_CURR + отсечение по порогу корреляции

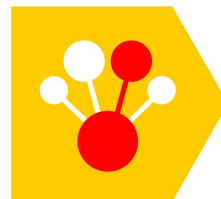
Модель

Рассматривался бустинг, в том числе CatBoost, Lightgbm, XGBoost

Перебирались следующие параметры:

1. learning rate
2. n_iterations
3. параметр регуляризации l2

XGBoost







LightGBM

Модель

В итоге используем CatBoostClassifier на 10 тыс итераций

Причины выбора:

- Легко собрать и запустить на GPU
- Много tutorиалов
- Качество у всех бустингов сравнимое

Submission and Description	Private Score ⓘ	Public Score ⓘ	Selected
 res11.csv Complete (after deadline) · 13h ago	0.73254	0.73733	<input type="checkbox"/>
 res10.csv Complete (after deadline) · 13h ago	0.7326	0.73749	<input type="checkbox"/>
 res9.csv Complete (after deadline) · 13h ago	0.76337	0.76584	<input type="checkbox"/>
 res8.csv Complete (after deadline) · 2d ago	0.76455	0.76628	<input type="checkbox"/>