Прогнозирование остатка на счете

Команда «Программистики»

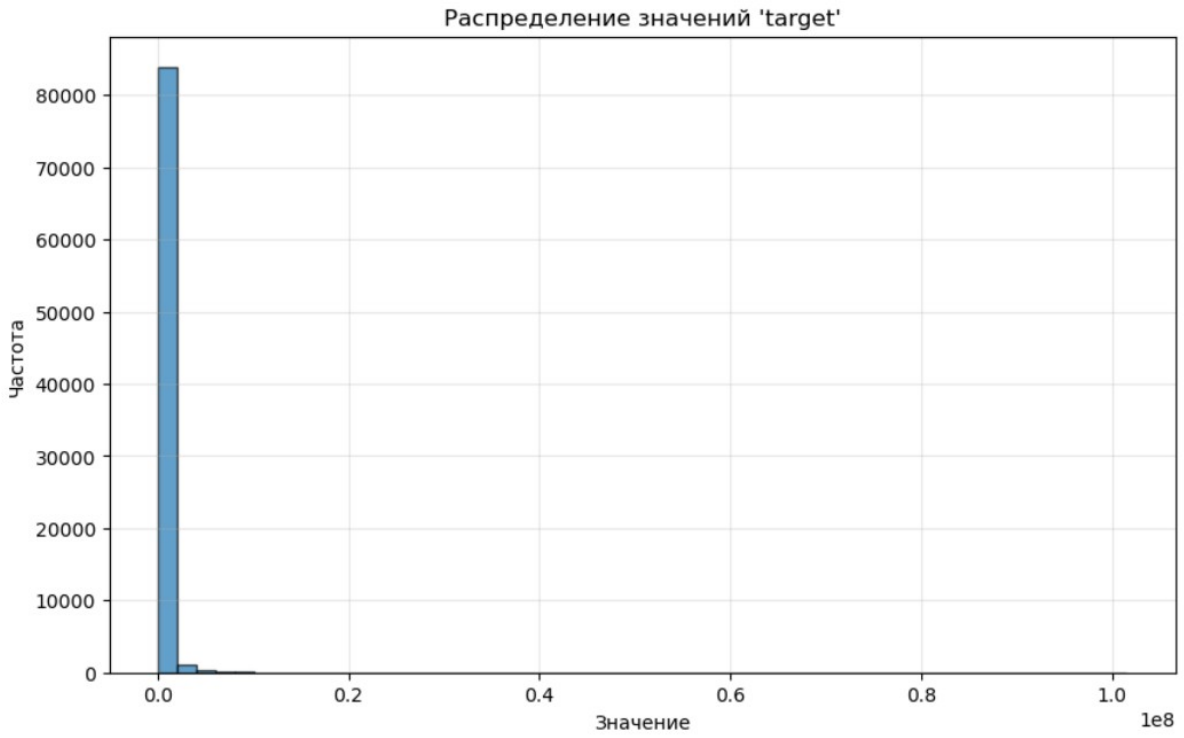
Состав: Армбристер Никита, Бурковская Вероника

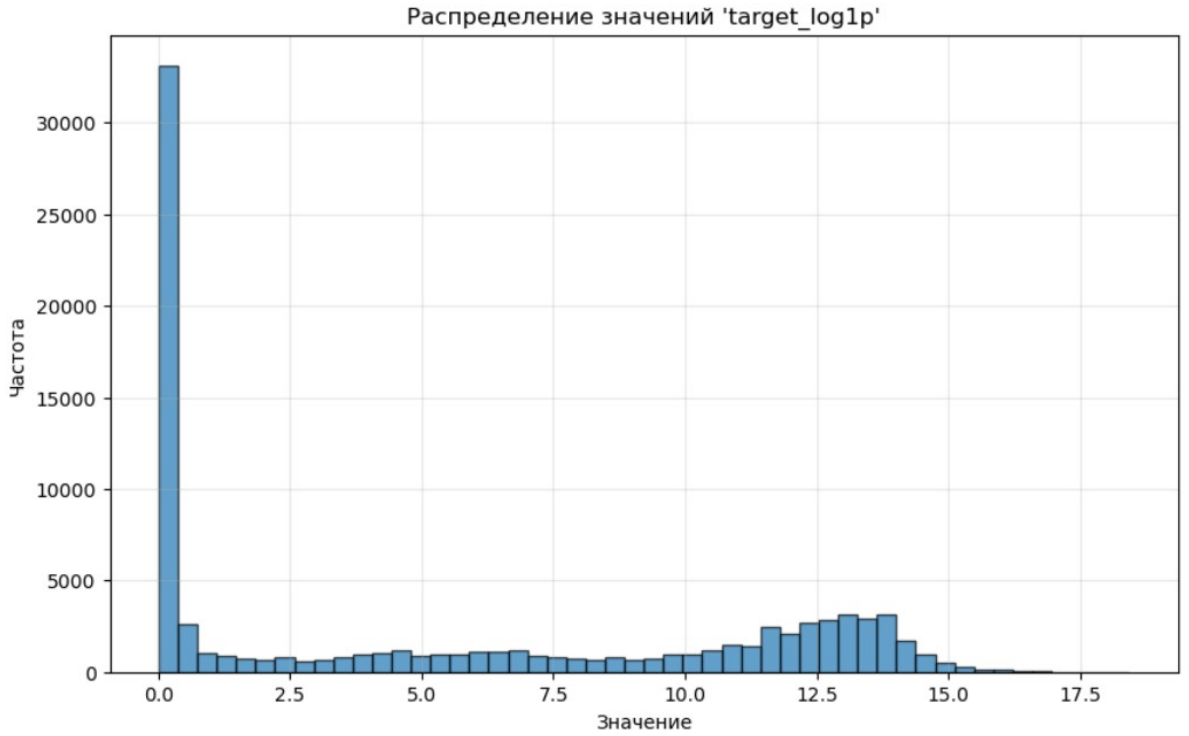
Куратор: Зелинский Илья

План:

1. target и id
2. Анализ данных (EDA)
3. Модели
4. Безрезультатные действия
5. Возможности для развития
6. Инференс

Распределения данных target





Повторения id в датасетах

```
# Было выяснено, что некоторые id в датасете повторяются. Необходимо продумать этот момент
train_target['id'].value_counts()
```

```
id
12039976    2
30775013    2
48717440    2
45299472    2
39341940    2
           ..
73859230    1
4020746     1
49174163    1
74793930    1
84261968    1
Name: count, Length: 85557, dtype: int64
```

```
train_target['index'] = train_target.index
train_main_df['index'] = train_main_df.index
train_card_spending_df['index'] = train_card_spending_df.index
train_mcc_operations_df['index'] = train_mcc_operations_df.index
train_mcc_preferences_df['index'] = train_mcc_preferences_df.index

# соединяем train_target, train_main_df
train_full_info = pd.merge(train_target, train_main_df, on=['id', 'index'])
```

```
train_main_df[train_main_df['id'] == 12039976].dropna(axis=1).T
```

	10893	55025
avg_dep_avg_balance_12month_amt	7784.149414	3788234.0
avg_dep_avg_balance_12month_amt_term_savings	0.0	3801863.75
avg_dep_avg_balance_1month_amt	4533.266113	2984567.75
avg_dep_avg_balance_1month_amt_term_savings	0.0	2981169.0
avg_dep_avg_balance_3month_amt	2528.572998	2965722.0
...	...	...
savings_safe_acc_flg	1	1
savings_broker_flg	0	0
savings_oms_flg	0	0
id	12039976	12039976
index	10893	55025

Категориальные признаки

1. Отношение порядка

```
'Массовый': 0,
'Привилегия': 1,
'VIP': 2
```

```
'0': 0,
'0-3м': 1,
'3-6м': 2,
'6-12м': 3,
'12м+': 4
```

2. Частота встречаемости категорий

Пример на марках машин:

0 - самая частая ,

1 - менее частая,

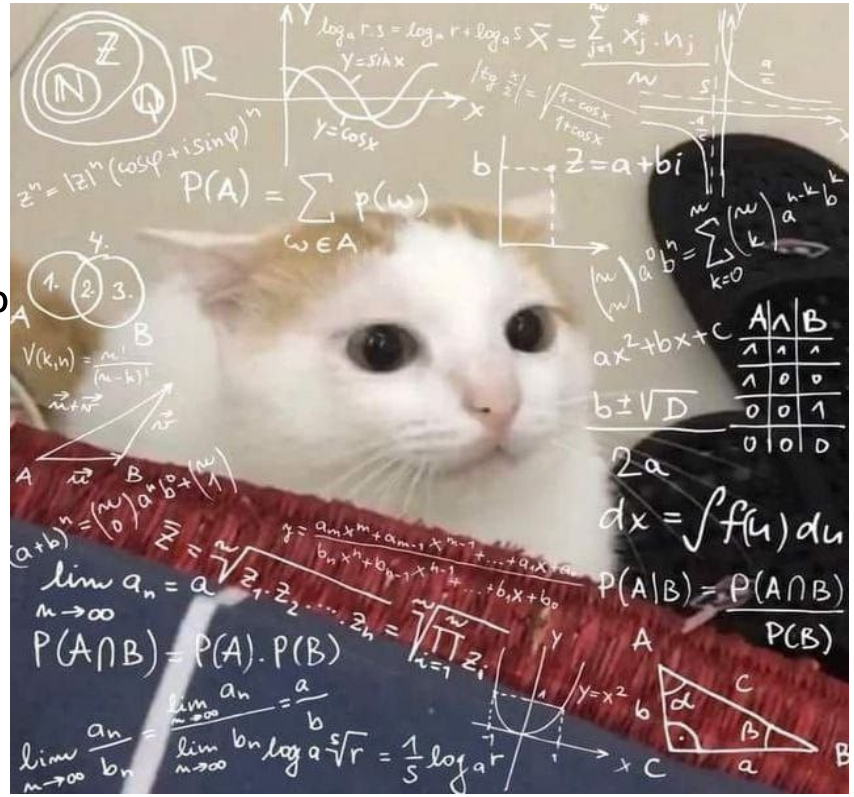
...

n - самая редкая марка в df.

Отсеянные числовые признаки

1. Пропуски >70%
2. Сильная корреляция между собой > 0.95%
3. Корреляция с таргетом <0.1
4. Дисперсия < 0.02

Осталось: 470 признаков



Модели

Линейная регрессия

Дерево Решений

Рандомный лес

CatBoost

Catboost
Ассиметричное распределение

```
modelcat = CatBoostRegressor(
    bootstrap_type='MVS',        # Лучший бутстрап для длинных хвостов
    grow_policy='Lossguide',     # Лучше ловит выбросы
    #loss_function='Poisson',    # для пуассоновских данных но в нашем случае не эффективно
    max_leaves=64,               # Контроль сложности
    l2_leaf_reg=5,               # Сильная регуляризация
    min_data_in_leaf=20,         # Защита от переобучения
    has_time=False,              # Если данные не временные
    boosting_type='Plain',       # Лучше для регрессии
    score_function='L2',         # Для регрессии
    random_state=42,
    verbose=20,
    rsm = 0.7,
    iterations=400,
    learning_rate=0.07,
    max_depth=6
)
```

Идея заполнения колонок по маске

```
# проверка корреляции между похожими колонками
mask_avg_dep = train_full_info.columns.str.startswith('avg_dep_avg_balance_fact_')
train_full_info.loc[:, mask_avg_dep].corr(method='spearman')
```

	avg_dep_avg_balance_fact_12month_amt	avg_dep_avg_balance_fact_12month_amt_term
avg_dep_avg_balance_fact_12month_amt	1.000000	0.859138
avg_dep_avg_balance_fact_12month_amt_term	0.859138	1.000000
avg_dep_avg_balance_fact_12month_amt_term_savings	0.945686	0.874632
avg_dep_avg_balance_fact_1month_amt	0.901775	0.776486
avg_dep_avg_balance_fact_1month_amt_term	0.865747	0.959196


```
# идея заполнения однотипных колонок по маске
mask_avg_dep = train_full_info.columns.str.startswith('avg_dep_avg_balance_fact_')
mask_zp = train_full_info.columns.str.startswith('zp_')
mask_agg = train_full_info.columns.str.startswith(('max_', 'min_', 'avg_', 'sum_'))
mask_dep = train_full_info.columns.str.startswith('dep_')
mask_income = train_full_info.columns.str.startswith('income_')
mask_cnt = train_full_info.columns.str.startswith('cnt_')

# заполнение медианой колонок из списка avg_dep_avg_balance_fact_...
cols_to_fill = train_full_info.columns[mask_avg_dep]
train_full_info[cols_to_fill] = train_full_info[cols_to_fill].fillna(train_full_info[cols_to_fill].median())
```

Связь income_колонок

```
income_columns = ['app_income_app', 'income_verified', 'income_verified_primary_job', 'income_unverified']
train_full_info[income_columns].dropna(thresh=4).head(10)
```

	app_income_app	income_verified	income_verified_primary_job	income_unverified
2	105372.960938	106076.906250	104224.382812	159.007492
3	0.000000	631.288635	0.000000	23.733812
5	26904.433594	26173.880859	25919.302734	0.000000
6	98499.656250	99265.968750	99286.156250	0.000000
7	51601.230469	48454.519531	51466.089844	88.390167
8	80143.890625	78442.468750	78873.171875	0.000000
11	235907.078125	233963.281250	233864.937500	0.000000
12	62818.480469	61697.394531	61851.847656	157.446381
13	81700.992188	54101.441406	54836.464844	25041.867188
16	177145.187500	145401.171875	144743.187500	30058.400391


```
# Вывод: app_income_app = mean(income_verified, income_verified_primary_job) + income_unverified
func = lambda x: (x['income_verified'] + x['income_verified_primary_job']) / 2 + x['income_unverified']
train_full_info['app_income_app'] = train_full_info.apply(func, axis=1)
```

Возможности для развития

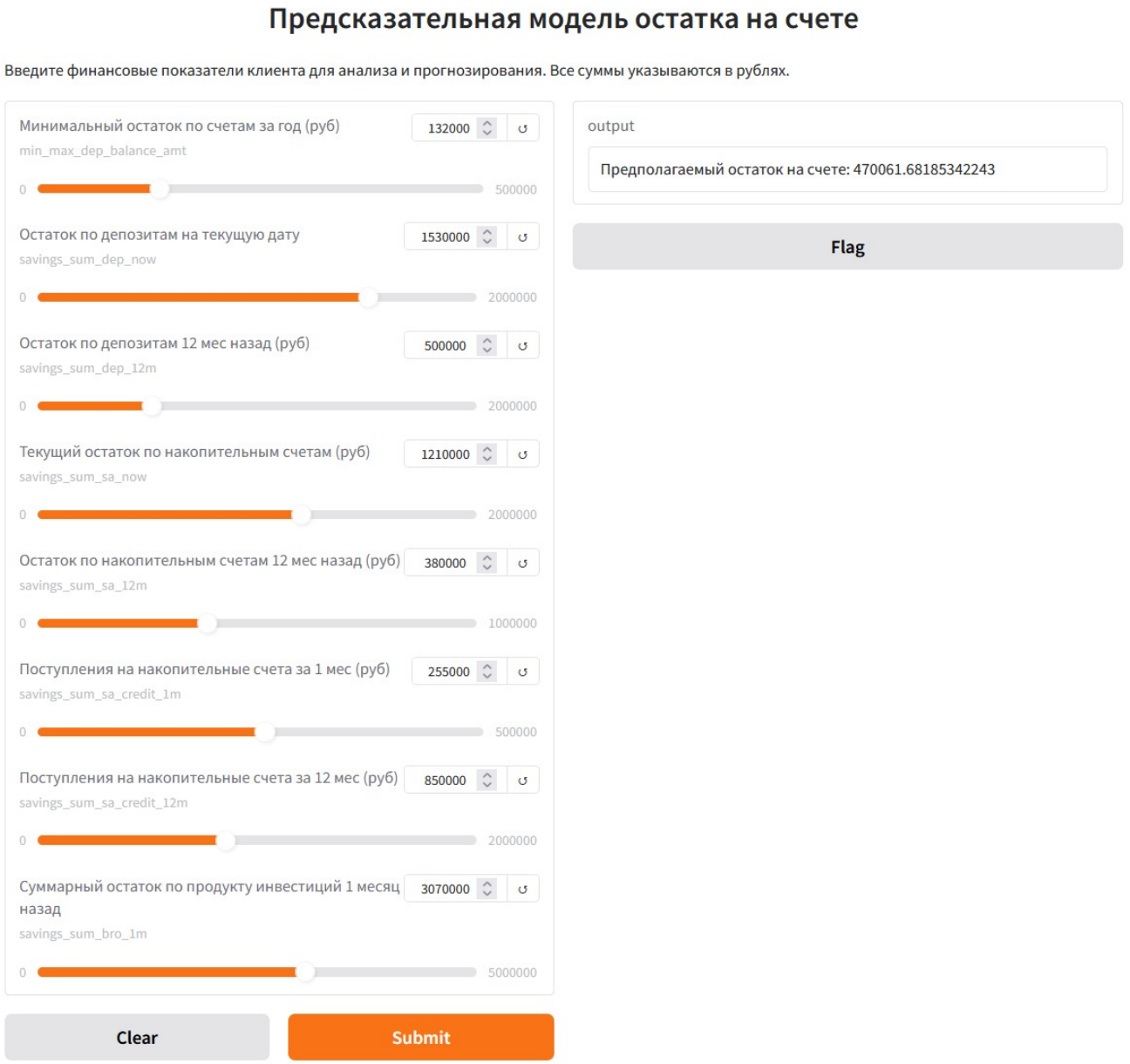
- Работа с выбросами
- Формирование новых признаков
- Boruta
- SHAP

Функции глобальной очистки

- **filter_by_variance** - удаляет колонки с дисперсией ниже заданного порога
- **filter_by_low_correlation** - оставляет только колонки с корреляцией (по модулю) выше порога с целевой переменной
- **remove_temporal_duplicates** - удаляет временные варианты колонок
- **preprocess_data** - удаляет колонки с долей пропусков (NaN) выше заданного порога
- **remove_correlated_features** - удаляет сильно коррелирующие между собой признаки

Инференс

- min_max_dep_balance_amt
- savings_sum_dep_now
- savings_sum_dep_12m
- savings_sum_sa_now
- savings_sum_sa_12m
- savings_sum_sa_credit_1m
- savings_sum_sa_credit_12m
- savings_sum_bro_1m



ШИФТ
Экосистема

Спасибо за внимание!