ПОСТРОЕНИЕ ЗАЯВОЧНОЙ МОДЕЛИ ПО ФИЗИЧЕСКИМ ЛИЦАМ С УЧЕТОМ ОТКАЗАННЫХ ЗАЯВОК (APPLICATION SCORING MODEL WITH REJECT INFERENCE)

СТАШЕВСКИ ЭМИЛЬ
КЕММЕР АНАСТАСИЯ
ИЗОСЕНКОВ АЛЕКСАНДР

СТРУКТУРА ДОКЛАДА

- ОПИСАНИЕ ЗАДАЧИ
- ДАННЫЕ И ПРИЗНАКИ
- ПРИМЕНЕНИЕ МОДЕЛЕЙ
- ВЫВОДЫ

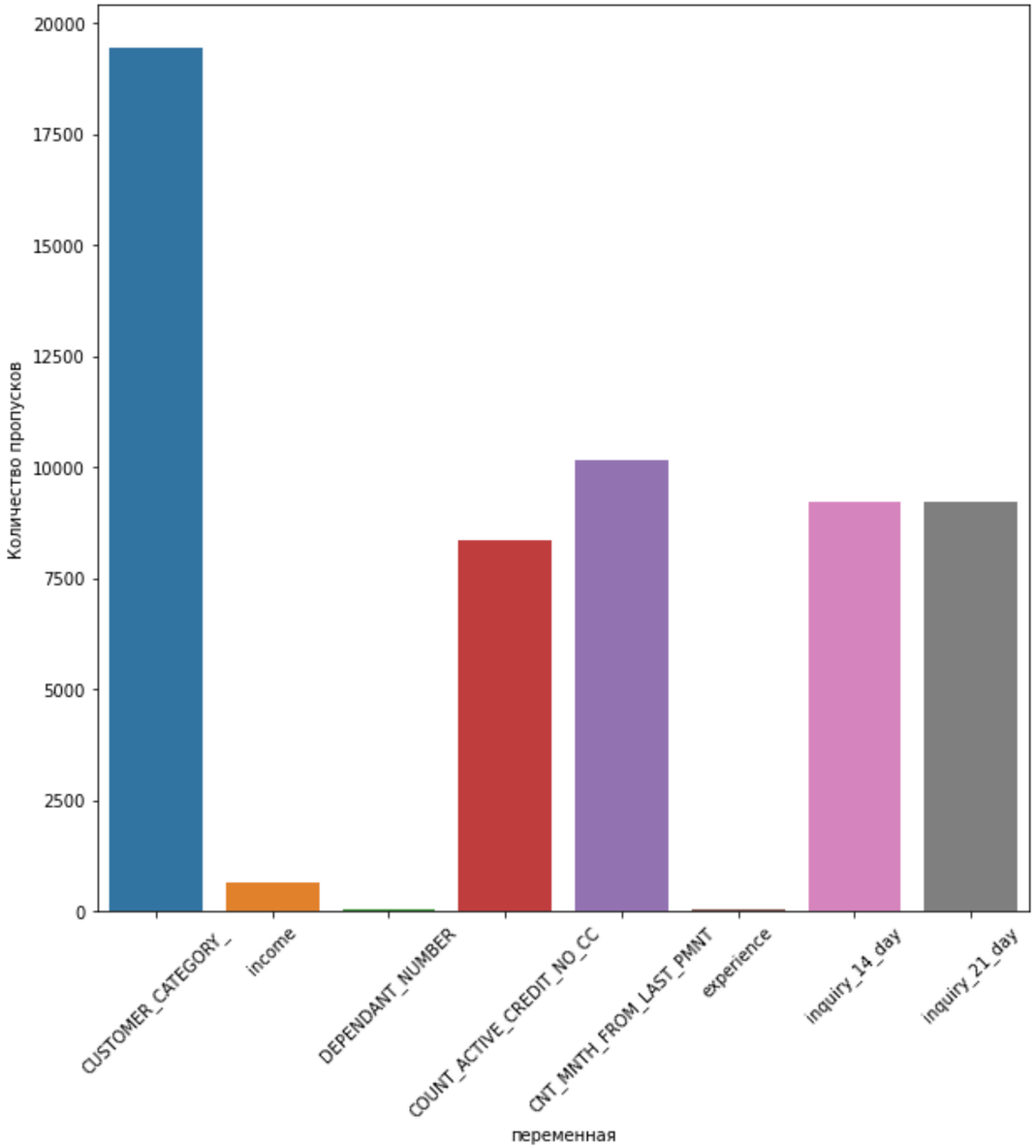
ПОСТАНОВКА ЗАДАЧИ

ЗАДАЧА: СОЗДАТЬ ПРАВИЛА ОТБОРА КРЕДИТНЫХ ЗАЯВОК, ОСНОВЫВАЯСЬ НА ИСТОРИЧЕСКИХ ДАННЫХ ПО КЛИЕНТАМ БАНКА.

СТРУКТУРА ДОКЛАДА

- ОПИСАНИЕ ЗАДАЧИ
- ДАННЫЕ И ПРИЗНАКИ
- ПРИМЕНЕНИЕ МОДЕЛЕЙ
- ВЫВОДЫ

ПРОПУСКИ В ДАННЫХ

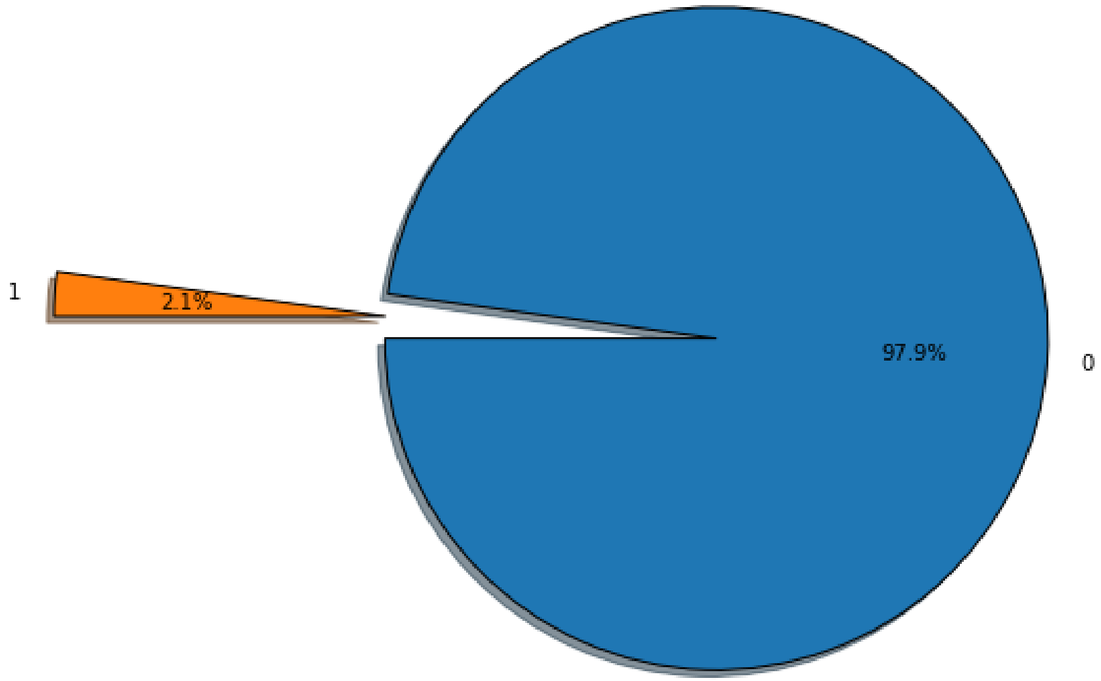


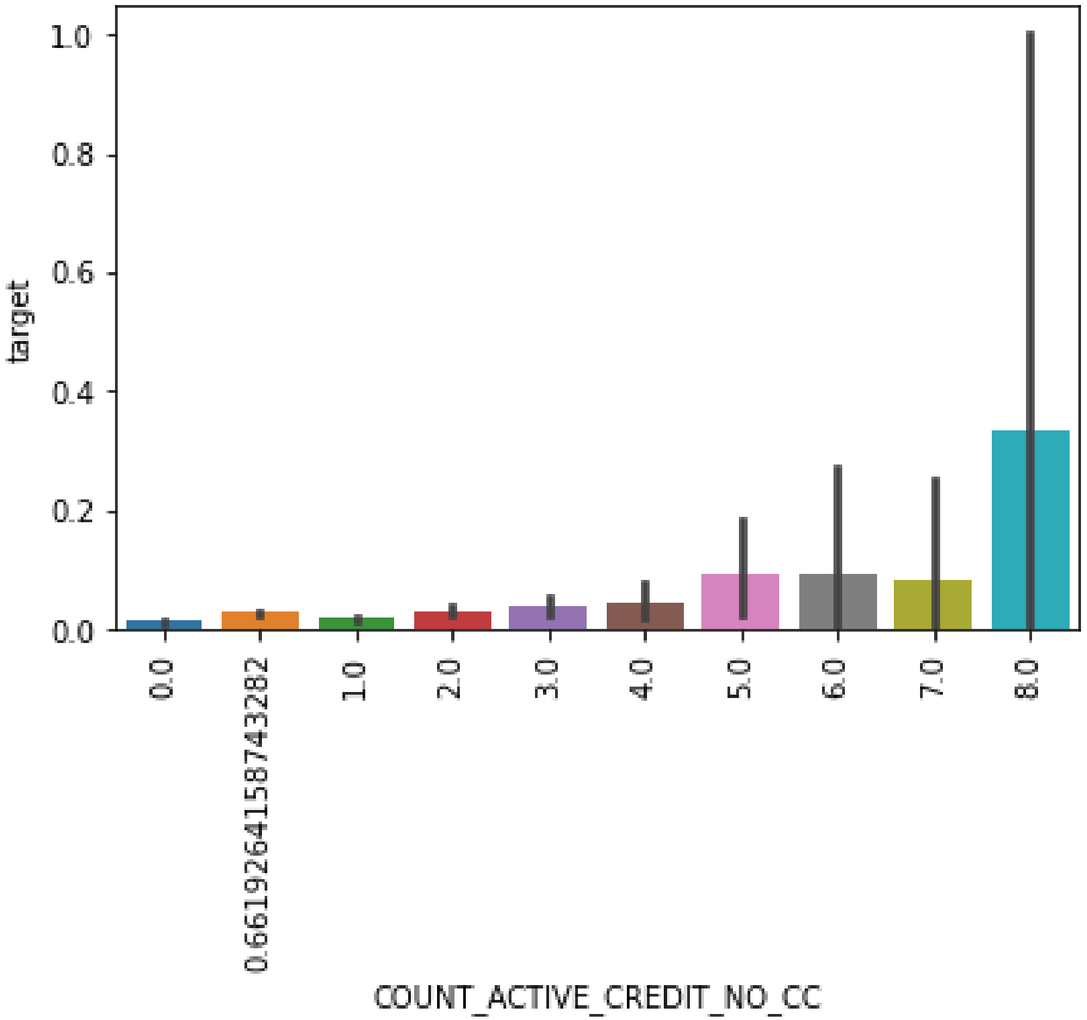
ОБРАБОТКА ПРОПУСКОВ

```
reject['CUSTOMER_CATEGORY_']=reject['CUSTOMER_CATEGORY_'].fillna('basic_category')

reject['count_mnth_act_passport']=np.where(reject['count_mnth_act_passport'] <0, np.NaN,reject['count_mnth_act_passport'] )

reject['experience']=np.where((reject['experience'] <0) & ( reject['experience'] >60), np.NaN,reject['experience'])
reject['CNT_MNTH_FROM_LAST_PMNT']=np.where(reject['CNT_MNTH_FROM_LAST_PMNT'] <0, np.NaN, reject['CNT_MNTH_FROM_LAST_PMNT'])

reject['count_mnth_act_passport'] = reject['count_mnth_act_passport'].fillna(reject['count_mnth_act_passport'].mean())
reject['CNT_MNTH_FROM_LAST_PMNT'] = reject['CNT_MNTH_FROM_LAST_PMNT'].fillna(reject['CNT_MNTH_FROM_LAST_PMNT'].mean())
reject['COUNT_ACTIVE_CREDIT_NO_CC'] = reject['COUNT_ACTIVE_CREDIT_NO_CC'].fillna(reject['COUNT_ACTIVE_CREDIT_NO_CC'].mean())
reject['inquiry_21_day'] = reject['inquiry_21_day'].fillna(reject['inquiry_21_day'].mean())
reject['inquiry_14_day'] = reject['inquiry_14_day'].fillna(reject['inquiry_14_day'].mean())
reject['DEPENDANT_NUMBER'] = reject['DEPENDANT_NUMBER'].fillna(reject['DEPENDANT_NUMBER'].mean())
reject['income'] = reject['income'].fillna(reject['income'].mean())
reject['CNT_MNTH_FROM_LAST_PMNT'] = reject['CNT_MNTH_FROM_LAST_PMNT'].fillna(reject['CNT_MNTH_FROM_LAST_PMNT'].mean())


X=reject[['inquiry_14_day','inquiry_21_day']].fillna(0)


reject['inquiry']= PCA_model.fit_transform(X)

reject =  reject.drop(['inquiry_14_day','inquiry_21_day'], axis=1)

reject['ratio_experience'] = reject['experience']/reject['age']

reject = pd.get_dummies(reject, columns=['INCOME_TYPE', 'EDUCATION_', 'CUSTOMER_CATEGORY_'])
```

```
for col in reject.columns:
    if reject[col].dtype == 'object':
        reject[col].fillna(reject[col].mode()[0], inplace = True)
    else:
        reject[col].fillna(reject[col].median(), inplace = True)
```

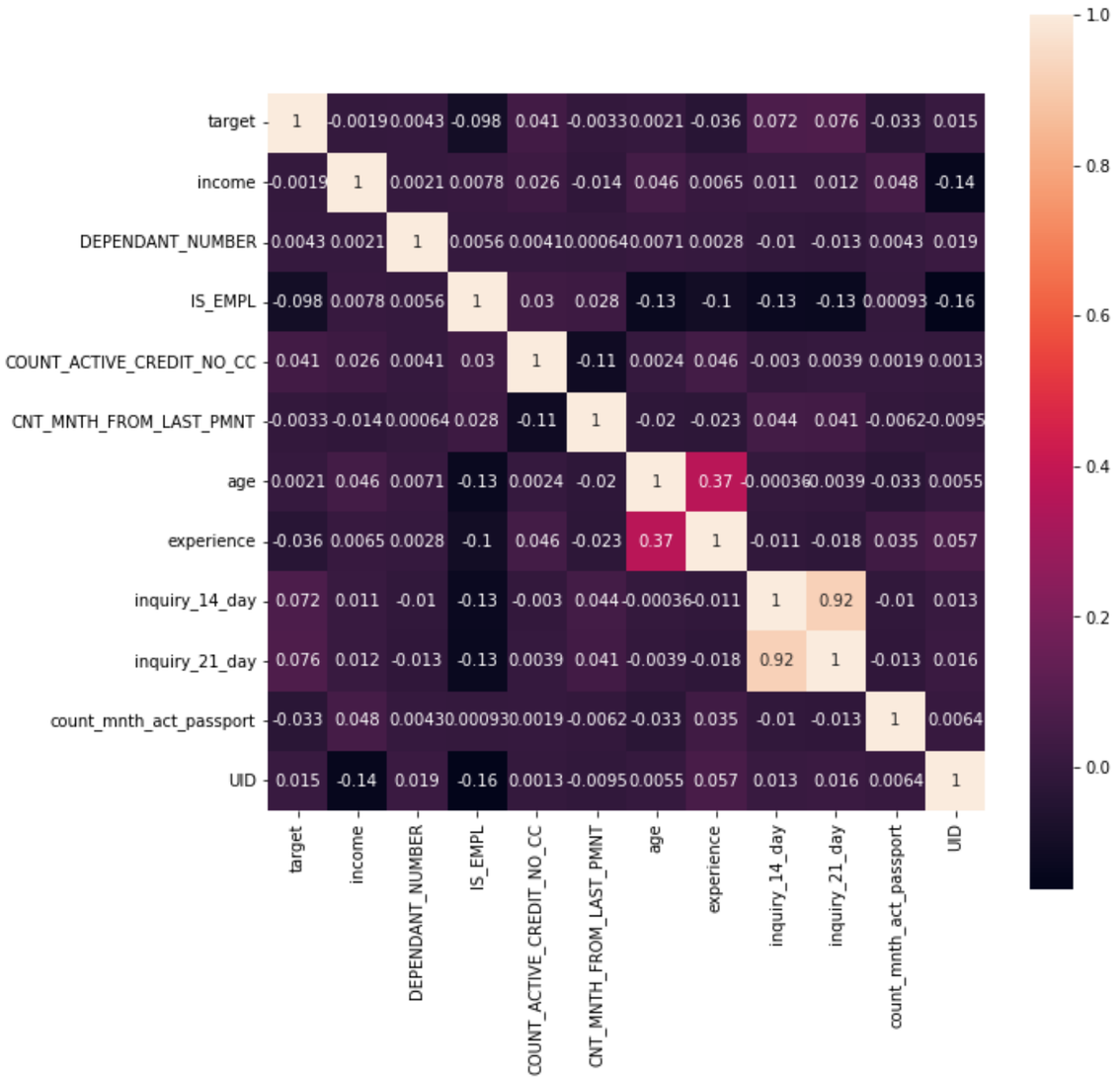
РАСПРЕДЕЛЕНИЕ ЦЕЛЕВОЙ ПЕРЕМЕННОЙ





КОЛИЧЕСТВО ОТКРЫТЫХ КРЕДИТОВ ПОЛОЖИТЕЛЬНО ВЛИЯЮТ НА ВЕРОТЯНОСТЬ ОТКАЗА

КОРРЕЛЯЦИИ



- INQUIRY_14_DAY И INQUIRY_21_DAY СИЛЬНО КОРРЕЛИРУЮТ (КОР ПИРСОНА = 0.922379), ВОЗРАСТ И ОПЫТ РАБОТЫ (КОР ПИРСОНА = 0.37)

- НАИБОЛЕЕ СИЛЬНО СКОРРЕЛИРОВАН С ЦЕЛЕВОЙ ПЕРЕМЕННОЙ СТОЛБЕЦ IS_EMPL - СТАТУС СОТРУДНИКА БАНКА УМЕНЬШАЕТ ШАНС ОДОБРЕНИЯ. ТАКЖЕ ЭТА ПЕРЕМЕННАЯ ИМЕЕТ ОТРИЦАТЕЛЬНУЮ КОРРЕЛЯЦИЮ С ОПЫТОМ

СХЛОПЫВАНИЕ СИЛЬНО СКОРРЕЛИРОВАННХЫ ПЕРЕМЕННЫХ

```
X=accept[['inquiry_14_day','inquiry_21_day']].fillna(0)

from sklearn import decomposition
pca = decomposition.PCA(n_components=1)
pca.fit(X)
print(pca.explained_variance_ratio_)
```

```
[0.96171079]
```

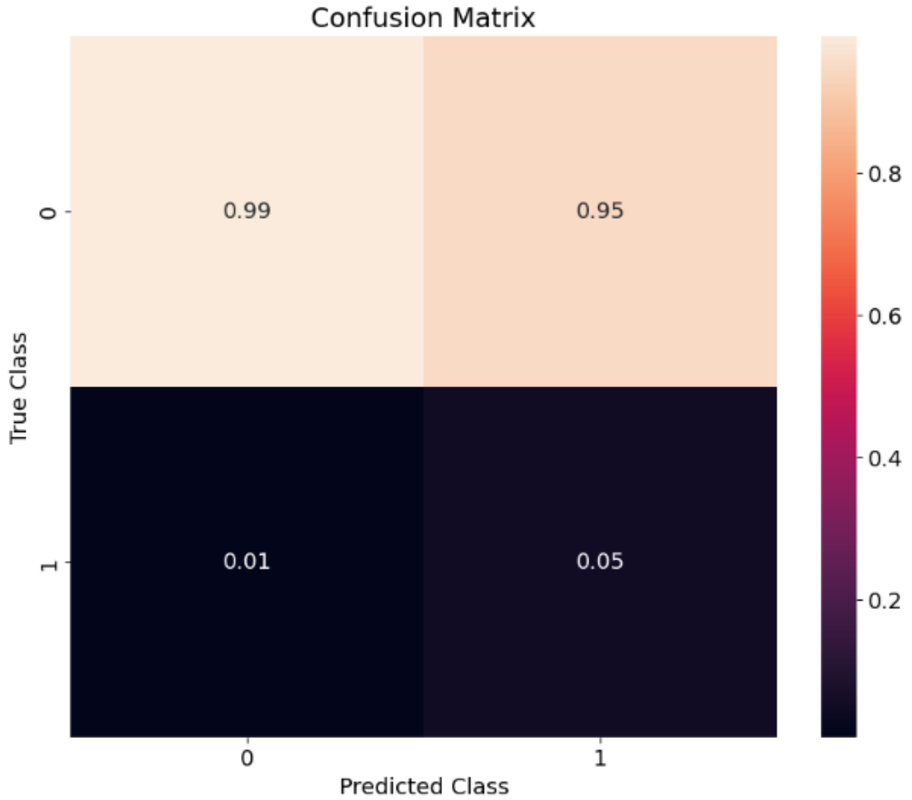
Первая компонента сожержит более 96% вариации двух переменных, потеря вариации незначительна, можем брать

```
from sklearn.decomposition import PCA
PCA_model = PCA(n_components=1,random_state=42) # We reduce the dimensionality to two dimensions and set the
                                                # random state to 42
accept['inquiry']= PCA_model.fit_transform(X)

accept =  accept.drop(['inquiry_14_day','inquiry_21_day'], axis=1)
```

СТРУКТУРА ДОКЛАДА

- ОПИСАНИЕ ЗАДАЧИ
- ДАННЫЕ И ПРИЗНАКИ
- ПРИМЕНЕНИЕ МОДЕЛЕЙ
- ВЫВОДЫ

МОДЕЛЬ: ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ




```
[ ]   1 from sklearn.metrics import precision_score, recall_score
      2 print('precision:', precision_score(test['predictions_01'], y_test))
      3 print('recall:', recall_score(test['predictions_01'], y_test))
```

```
precision: 0.7769784172661871
recall: 0.05227492739593417
```

СКОРКАРТА

```
[ ]  1 scorecard['score'] = (-1)*(scorecard['coef']*scorecard['WoE'] +
     2 scorecard['score'] = scorecard['score'].round(0)
     3 scorecard['WoE'] = scorecard['WoE'].round(2)
     4 scorecard['coef'] = scorecard['coef'].round(2)
     5 scorecard
     6
```

	feature	Value	WoE	coef	score
0	income	(56411.35, 61331.771]	-0.89	-0.64	30.0
1	income	(46185.15, 50000.0]	-0.43	-0.64	39.0
2	income	(39223.0, 42281.0]	-0.33	-0.64	41.0
3	income	(24972.0, 27000.0]	-0.29	-0.64	41.0
4	income	(79957.0, 92849.05]	-0.16	-0.64	44.0
...	...	...	...	...	...
86	DEPENDANT_NUMBER	2.0	-0.56	-0.90	32.0
87	DEPENDANT_NUMBER	3.0	0.00	-0.90	47.0
88	DEPENDANT_NUMBER	4.0	0.00	-0.90	47.0
89	DEPENDANT_NUMBER	0.0	0.05	-0.90	48.0
90	DEPENDANT_NUMBER	1.0	0.13	-0.90	50.0

91 rows × 5 columns

```
from catboost import CatBoostClassifier
```

```
cls = CatBoostClassifier(depth=10)
```

```
cls.fit(X_train, y_train)
```

```
Learning rate set to 0.041441
0:      learn: 0.6141397        total: 98.8ms   remaining: 1m 38s
1:      learn: 0.5508941        total: 112ms    remaining: 55.8s
2:      learn: 0.4942153        total: 130ms    remaining: 43.3s
3:      learn: 0.4439200        total: 148ms    remaining: 36.9s
4:      learn: 0.4005256        total: 166ms    remaining: 33.1s
5:      learn: 0.3606899        total: 209ms    remaining: 34.6s
6:      learn: 0.3293282        total: 220ms    remaining: 31.3s
7:      learn: 0.3008490        total: 229ms    remaining: 28.4s
8:      learn: 0.2729805        total: 238ms    remaining: 26.2s
9:      learn: 0.2509479        total: 251ms    remaining: 24.8s
10:     learn: 0.2299393        total: 276ms    remaining: 24.8s
11:     learn: 0.2136254        total: 291ms    remaining: 24s
12:     learn: 0.1993223        total: 314ms    remaining: 23.9s
13:     learn: 0.1842685        total: 329ms    remaining: 23.2s
14:     learn: 0.1715944        total: 352ms    remaining: 23.1s
15:     learn: 0.1625036        total: 363ms    remaining: 22.3s
16:     learn: 0.1546448        total: 374ms    remaining: 21.6s
17:     learn: 0.1469932        total: 384ms    remaining: 21s
```

```
pr = cls.predict(X_test)
```

```
from sklearn.metrics import accuracy_score, precision_recall_display
```

```
accuracy_score(pr, y_test)
```

```
0.979419444017816
```

```
precision_recall_curve(pr, y_test).di
```

```
(array([0.00322531, 0.09352518, 1.        ]),
 array([1.        , 0.61904762, 0.        ]),
 array([0, 1]))
```

```
from sklearn.metrics import precision_score, recall_score
print('precision:', precision_score(pr, y_test))
print('recall:', recall_score(pr, y_test))
```

```
precision: 0.09352517985611511
recall: 0.6190476190476191
```

БУСТИНГ НА ДАННЫХ ПО ОДОБРЕННЫМ ЗАЯВКАМ ЖЕРТВУЕТ ТОЧНОСТЬЮ РАДИ ПОЛНОТЫ

```
[ ]   1 print('precision:', precision_score(pr, y_test))
      2 print('recall:', recall_score(pr, y_test))
```

```
precision: 0.09352517985611511
recall: 0.6190476190476191
```

REJECT INFERENCE

Reject inference

```
reject['target'] = 1
reject.loc[
  ((reject['age'] > 23) & (reject['income'] > 30000) & (reject['COUNT_ACTIVE_CREDIT_NO_CC'] <= 1)),
  'target'
] = 0
```

```
reject['target'].value_counts()
```

```
1    6202
0    4491
Name: target, dtype: int64
```

```
newdata = pd.concat([accept2, reject])
assert len(newdata) == len(accept2) + len(reject)
```

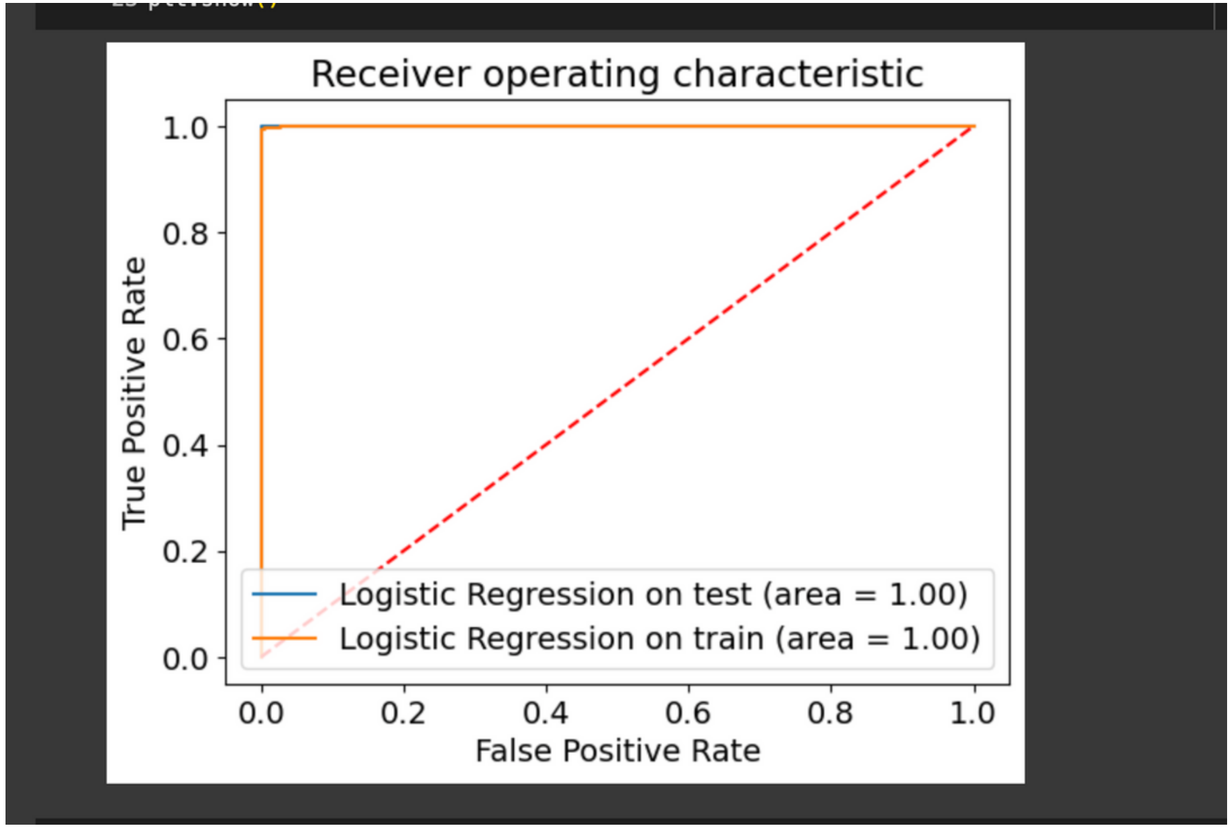
```
interval_features = newdata.select_dtypes(include=['float64', 'int64']).columns.tolist()
interval_features.remove('target')
interval_features.remove('UID')
interval_features.remove('IS_EMPL')
interval_features.remove('DEPENDANT_NUMBER')
interval_features
```

```
['income',
 'COUNT_ACTIVE_CREDIT_NO_CC',
 'CNT_MNTH_FROM_LAST_PMNT',
 'age',
 'experience',
 'count_mnth_act_passport',
 'inquiry',
 'ratio_experience',
 'INCOME_TYPE_2NDFL',
 'INCOME_TYPE_OTHER',
 'EDUCATION__advanced',
 'EDUCATION__beginner',
 'EDUCATION__elementary',
 'EDUCATION__high',
 'CUSTOMER_CATEGORY__Corporate',
 'CUSTOMER_CATEGORY__VIP',
 'CUSTOMER_CATEGORY__basic_category',
 'inquiry_14_day',
 'inquiry_21_day']
```

ОБУЧАЕМ МОДЕЛЬ НА БОЛЕЕ СБАЛАНСИРОВАННОЙ ВЫБОРКЕ

ЧАСТИ ОТКЛОНЕННЫХ ЗАЯВОК МЕНЯЕМ КЛАСС НА УДОБРЕННУЮ

(ДЛЯ КЛИЕНТОВ СТАРШЕ 23 С ДОХОДОМ БОЛЬШЕ 30К И КОЛИЧЕСТВОМ ТЕКУЩИХ КРЕДИТОВ НЕ БОЛЬШЕ 1)



МОДЕЛЬ: ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ НА СБАЛАНСИРОВАННОЙ ВЫБОРКЕ

```
[ ]  1 print('precision', precision_score(y_pred, y_test))
     2 print('recall   ', accuracy_score(y_pred, y_test))
```

```
precision 0.9946686976389947
recall    0.9990751445086705
```

РЕЗУЛЬТАТ: СКОРКАРТА

	feature	Value	WoE	coef	score
0	income	(-0.001, 15000.0]	-1.32	1.51	107.0
1	income	(15000.0, 18000.0]	-1.26	1.51	104.0
2	income	(20388.9, 22938.2]	-1.18	1.51	100.0
3	income	(18000.0, 20388.9]	-1.13	1.51	99.0
4	income	(25000.0, 27350.8]	-1.11	1.51	98.0
...	...	...	...	...	...
131	DEPENDANT_NUMBER	4.0	0.00	-2.03	49.0
132	DEPENDANT_NUMBER	3.0	0.00	-2.03	49.0
133	DEPENDANT_NUMBER	0.0	0.00	-2.03	49.0
134	DEPENDANT_NUMBER	1.0	0.31	-2.03	67.0
135	DEPENDANT_NUMBER	2.0	0.95	-2.03	105.0

136 rows × 5 columns

СТРУКТУРА ДОКЛАДА

- ОПИСАНИЕ ЗАДАЧИ
- ДАННЫЕ И ПРИЗНАКИ
- ПРИМЕНЕНИЕ МОДЕЛЕЙ
- ВЫВОДЫ

ВЫВОДЫ

- СБАЛАНСИРОВАННАЯ ВЫБОРКА КРАЙНЕ ВАЖНА ДЛЯ РЕШЕНИЯ ЗАДАЧИ КРЕДИТНОГО СКОРИНГА
- ЛИНЕЙНЫЕ МОДЕЛИ МОГУТ ПОКАЗЫВАТЬ ХОРОШЕЕ КАЧЕСТВО ПРИ ПРАВИЛЬНОЙ ПОДГОТОВКЕ ДАННЫХ
- WOE ПЕРЕМЕННЫЕ ПОМОГАЮТ ОТЛИЧНО РЕШАТЬ ЗАДАЧИ КРЕДИТНОГО СКОРИНГА