# Ход Решения

Загружаем данные

data = pd.read\_excel('Test.xlsx', header = 0)

Очищаем от нулевых значений

data = data.dropna()

Убираем столбец даты (по условия задания)

data = data.drop('dt', axis=1)

Проверяем форматы:

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 fpd5\_flg 14073 non-null int64

1 gender 14073 non-null int64

2 device\_quaility\_index 14073 non-null float64

3 behavioral\_index 14073 non-null float64

4 type\_of\_document 14073 non-null object

5 age 14073 non-null int64

6 email\_domain\_formatted 14073 non-null object

7 social\_status 14073 non-null object

8 salary 14073 non-null float64

9 disbursment\_type 14073 non-null object

10 os 14073 non-null object

11 email\_index 14073 non-null float64

Проверяем выпадающие значения. Видимо по столбцу ‘Salary’:

**1540 1500020000**

**8811 2387700**

**5082 900000**

**6826 800000**

**13821 780000**

5684 550000

4527 508000

5949 480009

12650 450000

8313 450000

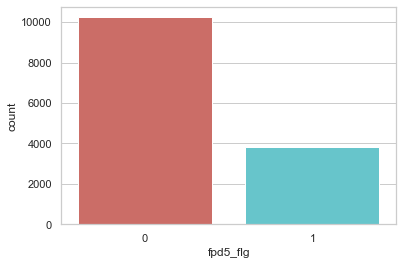
3421 400000

6781 380000

Убираем людей с чрезвычайно высокой ‘Salary’

data=data.drop([1540, 8811, 5082, 13821, 6826])

Проверяем сбалансированность искомого значения



Процент без дефолта 72.80352573215809

Процент дефолта 27.196474267841907

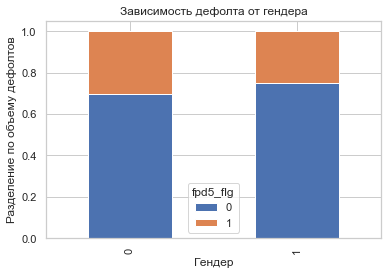
Сбалансированность слабая. Вопрос балансировки будет решаться позже.

Проверяем, имеет ли влияние на дефолт показатели числовых факторов:

| **fpd5\_flg** | **gender** | **device\_quaility\_index** | **behavioral\_index** | **age** | **salary** | **email\_index** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.64 | 2.42 | 4.02 | 35.35 | 24295.64 | 1.48 |
| **1** | 0.58 | 2.27 | 4.11 | 34.27 | 21658.91 | 1.66 |

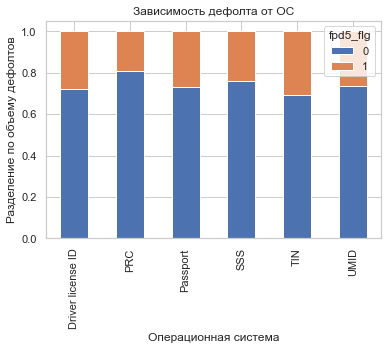
Из результата можно судить, что все показатели в той или иной степени могут влиять на показатель вероятности дефолта, хотя какого-то очевидно наглядного влияние не видно.

Проверяем показатель гендера:



Явной зависимости от гендера не наблюдается.

Проверяем по типу предоставляемого документа:

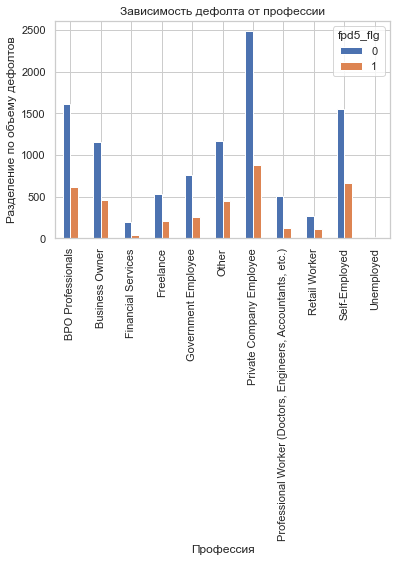


Можно наблюдать, что хоть разница и не велика, но все же определенная зависимость между предоставляемым документом и дефолтом есть.

Проверим, связано ли это долей и частотой предоставления того или иного документа.

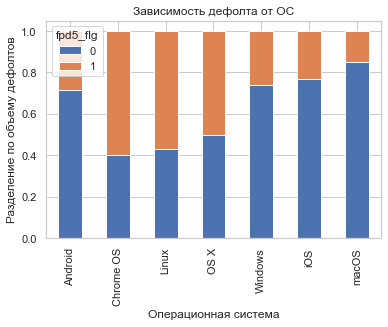
Можно предположить, что наличие того или иного документа так же отражает статус Клиента, его национальность и особенности его финансового поведения (так, например, человек с правами имеет автомобиль, что сильно влияет на его финансовое поведение).

Проверяем зависимость от рода деятельности:



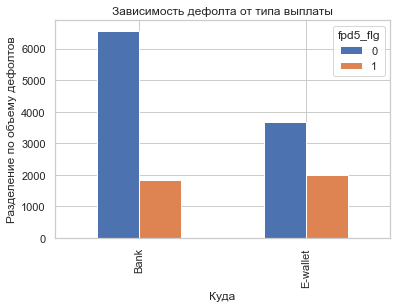
Тут зависимость очевидная и прямая. У людей из финансовой сферы дефолт происходит в среднем в два раза реже чем у людей из ретейла. Самый частый дефолт у безработных.

Смотрим зависимость от ОС:



Можно наблюдать, что относительно все три основные операционные системы примерно равны по объему дефолта. Однако это так же может быть показателем статуса - у человека есть средства на мобильный телефон под управлением apple/android.

Зависимость от того, куда направляются средства:



Зависимость явная.

Смотрим распределение почтовых сервисов:

GMAIL.COM 0.785826

YAHOO.COM 0.172590

YAHOO.COM.PH 0.011871

DEPED.GOV.PH 0.006824

ICLOUD.COM 0.005971

YMAIL.COM 0.004478

ROCKETMAIL.COM 0.002488

HOTMAIL.COM 0.001493

OUTLOOK.COM 0.001208

EMAIL.COM 0.000569

GMAIL и YAHOO имеют доминирующий перевес. Не думаю, что в данном случае имеется какая-то корреляция между сервисом почты и дефолтом. Думаю данным показателем можно пренебречь.

Далее вещественные столбцы преобразуем в числовые. Получаем следующие столбцы:

array(['fpd5\_flg', 'gender', 'device\_quaility\_index', 'behavioral\_index',

'age', 'salary', 'email\_index',

'type\_of\_document\_Driver license ID', 'type\_of\_document\_PRC',

'type\_of\_document\_Passport', 'type\_of\_document\_SSS',

'type\_of\_document\_TIN', 'type\_of\_document\_UMID',

'social\_status\_BPO Professionals', 'social\_status\_Business Owner',

'social\_status\_Financial Services', 'social\_status\_Freelance',

'social\_status\_Government Employee', 'social\_status\_Other',

'social\_status\_Private Company Employee',

'social\_status\_Professional Worker (Doctors, Engineers, Accountants, etc.)',

'social\_status\_Retail Worker', 'social\_status\_Self-Employed',

'social\_status\_Unemployed', 'disbursment\_type\_Bank',

'disbursment\_type\_E-wallet', 'os\_Android', 'os\_Chrome OS',

'os\_Linux', 'os\_OS X', 'os\_Windows', 'os\_iOS', 'os\_macOS'],

Проверяем корреляцию между элементами

data\_final.corr()

Сильная корреляция между элементами отсутствует.

Производим балансировку с помощью библиотеки SMOTE:

Размер cбалансированного файла 14368

Cбалансированое число без дефолта 7184

Число с дефолтом 7184

Доля без дефолта 0.5

Доля с дефолтом 0.5

Теперь у нас есть идеально сбалансированные данные. Over-sampling был выполнен только для обучающих данных, поскольку в таком случае информация из тестовых данных не используется для создания синтетических сведений, и, следовательно, не произойдёт её утечки в обучающую модель.

Производим рекурсивное устранение признаков. Выбрано количество элементов, при которых скоринг модели наивысший.

Выбираем следующие столбцы:

['type\_of\_document\_Driver license ID',

'type\_of\_document\_PRC',

'type\_of\_document\_Passport',

'type\_of\_document\_SSS',

'type\_of\_document\_TIN',

'type\_of\_document\_UMID',

'social\_status\_BPO Professionals',

'social\_status\_Business Owner',

'social\_status\_Financial Services',

'social\_status\_Freelance',

'social\_status\_Government Employee',

'social\_status\_Other',

'social\_status\_Private Company Employee',

'social\_status\_Professional Worker (Doctors, Engineers, Accountants, etc.)',

'social\_status\_Retail Worker',

'social\_status\_Self-Employed',

'social\_status\_Unemployed',

'disbursment\_type\_Bank',

'disbursment\_type\_E-wallet',

'os\_iOS',

'os\_macOS']

Производим обучение:

from sklearn.linear\_model import LogisticRegression  
from sklearn import metrics

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)  
logreg = LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True, intercept\_scaling=1, max\_iter=100, multi\_class='ovr', n\_jobs=1, penalty='l2', random\_state=None, solver='liblinear', tol=0.0001, verbose=0, warm\_start=False)  
logreg.fit(X\_train, y\_train)

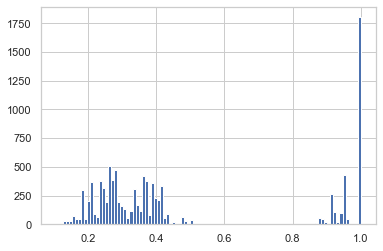
y\_pred = logreg.predict(X\_test)

Точность классификатора логистической регрессии для тестового набора: 0.79

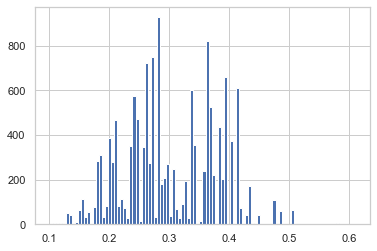
Расчёт ROC для тестового набора:



Распределение гистограммы для тестового набора:



Распределение для итогового набора:



Вероятность дефолта по каждому клиенту сохраняем в отдельном файле:

'output.xlsx'