

# Хакатон от компании «Интерфакс».

## Подтема

Определение признаков, важных, для  
предсказания дефолта российских эмитентов





## Дмитрий Яковлев

Закончил курсы повышения квалификации в МГТУ им. Баумана.

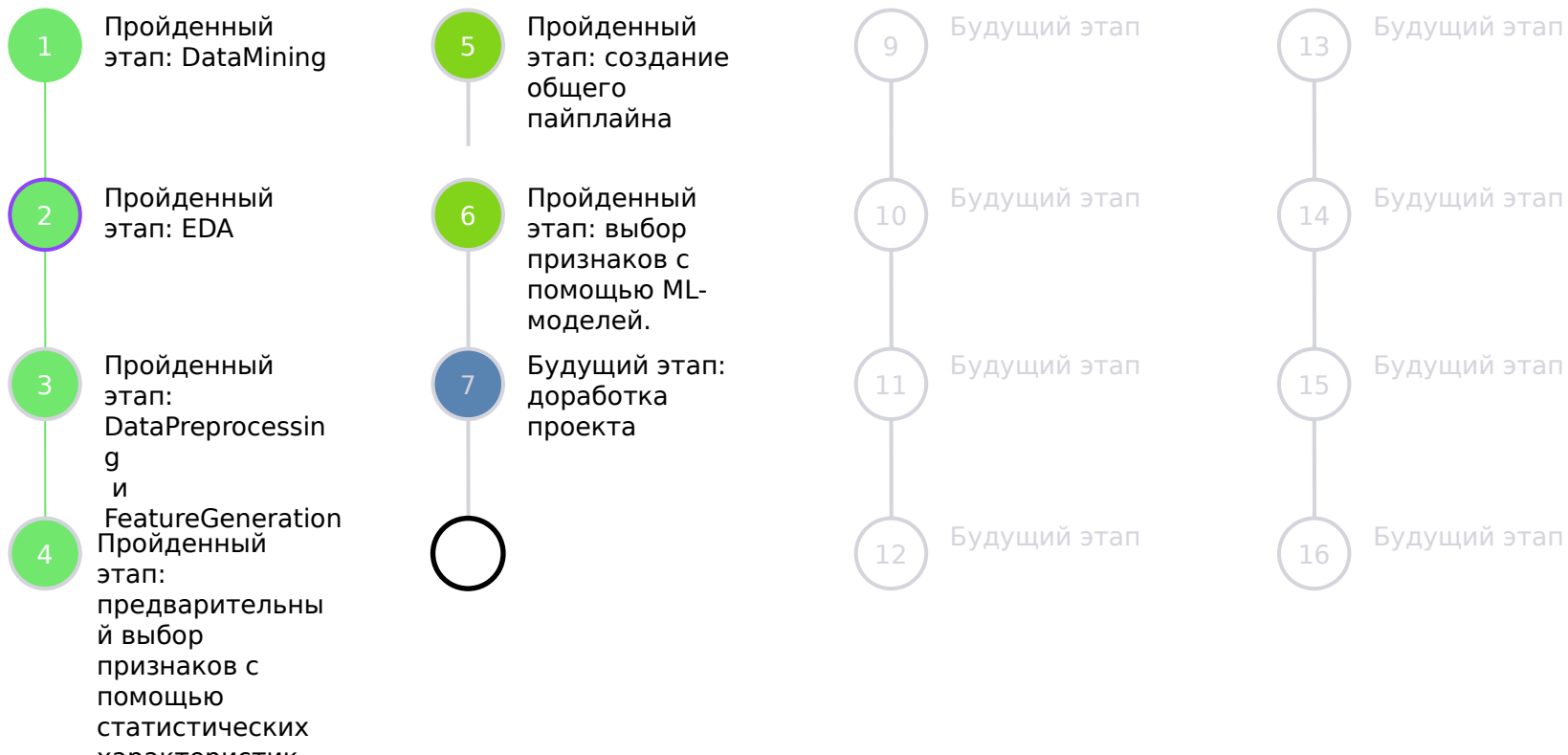
Закончил факультет искусственного интеллекта GeekBrains..

Закончил Сибирскую аэрокосмическую академию по специальности «Системы управления ракетно-космическими объектами и комплексами летательных аппаратов»

Инженер по работе с клиентами и проектировщиками на предприятии, производящем высоковольтное оборудование.



# План проекта





## Достигнутые цели

Из столь скудного датасета удалось выжать максимальные метрики качества. Понятно, что не снят вопрос с переобучением, но убрать переобучение легче, чем пытаться поднять неудовлетворительные метрики.

Решение еще очень сырое, но уже есть уверенность что все получится.

Кроме того рекомендую посмотреть мое решение похожего (предсказание дефолта «физиков») соревнования на Kaggle, где я смог получить максимальную для этого соревнования целевую метрику:

[https://github.com/DAYa66/Credit\\_Default\\_Prediction](https://github.com/DAYa66/Credit_Default_Prediction)



## Решение задачи / План работы

- Исследование данных на платформах InterFax.
- Датамайнинг данных и создание общего датасета, предварительная предобработка датасета, простейший анализ (ноутбук Assembling\_dataset).
- Исследование признаков, провел простейший анализ. Первый этап выявления самых бесполезных признаков. Преобразование временных признаков. Поиск закономерностей и проблем. Проведение анализа пропущенных значений и генерация признака, который четко делит датасет на дефолтные и недефолтные наблюдения. Заполнение пропущенных значений в датасете. Сжатие количественные признаки методом PCA. Подготовка датасета к предсказанию. (ноутбук EDA\_and\_FeatureIngeneering)
- Генерация новых категориальных признаков, вспомогательных к количественным. Исследование взаимной информации между признаками и таргетом. Удаление бесполезных признаков на основе этой информации. Обработка датасета с целью его приведения к возможности использования методов машинного обучения. Выбор классификаторов. Использование мощных инструментов машинного обучения для выявления важности признаков. (ноутбук Prediction)



## Трудности

Очень маленькое количество наблюдений с целевым событием, в несколько раз меньшее, чем общее количество признаков. Очень значимое количество пропусков в признаках. Попытка исследования пропусков привела к созданию признака, четко разделяющего эмитентов на дефолтных и недефолтных. Использование ML-моделей для заполнения пропусков не возможно из-за очень маленького количества событий.

Также в датасете имеется значительный дисбаланс классов по целевому признаку, который удалось исправить методом oversampling. Если сравнивать с одним из игрушечных датасетов Diabetes, то там при схожем количестве наблюдений почти 35% наблюдений с целевым событием и всего 9 признаков. В моем случае на 490 наблюдений 570 признаков, и всего 11% наблюдений с целевым событием. Т.е. за право предсказать какое-то событие, должны драться 10,5 признаков при условии что один признак предсказывает только одно событие, а это далеко не так. Такое явление приводит к такому понятию как «проклятие размерности».

Отсутствие в REST API данных по бухгалтерской отчетности эмитентов, движения капиталов и других важных во времени признаков.






## Assembling dataset

В виду отсутствия времени (через REST API предлагалось скачивать максимум по 20 облигаций за раз, причем питоновский список API не принимал, поэтому цикл по скачиванию облигаций сделать не получилось, а Excel у меня на Линуксе отсутствует) собрал основные датасеты через сайт и проверил полноту дефолтных облигаций через REST API. Количество дефолтных организаций совпало.



# Assembling dataset

Проверил из этого списка Гос. рег. номера на этом сайте: действительно у этих облигаций нет ISIN. Но, блин, потерять 32 из 158 дефолтных облигаций в наш таких облигаций не получится, если алгоритм на них не обучится. А это 20% от всех дефолтных облигаций.

 [https://www.isin.ru/ru/ru\\_isin/db/#search\\_rezult](https://www.isin.ru/ru/ru_isin/db/#search_rezult)  80% 

Наименование	<input type="text"/>
Категория ценных бумаг	<input type="text"/>
Тип ценных бумаг	<input type="text"/>
Регистрационный номер*	<input type="text" value="RU25069TMS0"/>

Поиск по коду:

☒ ISIN\*  ☐ CFI\*

Состояние кода

Результаты поиска. Найдено записей: 0

По Вашему запросу данных не найдено. Задайте другие параметры поиска.

Попытался найти пропущенные значения в датасете из других источников. Но быстро это сделать не получилось. Например, вот так пытался найти пропущенные ISIN.





# Assembling dataset

```
pd.DataFrame(df.groupby('Эмитент')['Дата начала размещения'].max()).merge(df,  
                                how='left', on=['Эмитент', 'Дата начала размещения'])['target'].value_counts()
```

```
0    519  
1     88  
Name: target, dtype: int64
```

```
cnt_bonds = pd.DataFrame(df.groupby('Эмитент')['Cratyc'].count()).rename(columns={'Cratyc': "cnt_bonds"}, errors="raise")  
cnt_bonds
```

cnt_bonds	
Эмитент	
2M	2
4K	1
А Девелопмент	1
А7 Агро	3
АБЗ-1	3
...	...
ЮниСервис Капитал	1
ЯМАЛЫ-АЛТЫН	1
аташ	0

В найденном датасете дефолтных облигаций по одному эмитенту. Значит, чтобы сравнивать их с недефолтными, недефолтные нельзя агрегировать по эмитенту (там есть несколько облигаций на одного эмитента). Оставлю по эмитентам самые свежие облигации. Но для сохранения информации создам признак: сколько у эмитента облигаций.



## Assembling dataset

```
# Поиск эмитента по списку кодов (fininstId, ИНН, ОГРН, shortname)
```

```
def Find_Emitent(token):
```

```
    url = api_url+'/Emitent/Find'
```

```
    body = {
```

```
        "codes": [
```

```
            '2М',
```

```
            '4К',
```

```
            'А Девелопмент',
```

```
            'А7 Агро',
```

```
            'АБЗ-1',
```

```
            'АВТОБАН-Финанс',
```

```
            'АВТОДОМ',
```

```
            'АК БАРС Банк',
```

```
            'АО ИИ Т.Б. "Сбербанк"
```

Выгруженные с сайта датасеты не имели fininstId, ИНН, ОГРН. Чтобы пользоваться API выгрузил эти данные этим методом.



## Assembling dataset

```
# Возвращает данные об обременениях имущества компаний
def Encumbrance(token):
    url = api_url+'/Emitent/EncumbranceList'
    body = {
        "actualOnDate": "2023-03-05T04:00:32.435Z",
        "pageNum": 1,
        "pageSize": 10000
    }

    return doPostRequest(url, body, token)

df_Encumbrance = pd.DataFrame(Encumbrance(token))

print (f'Число обременений:{df_Encumbrance.shape[0]}')
df_Encumbrance.tail(10)
```

Этим методом собрал данные есть ли обременения у эмитентов. Данные скорингов и листингов выгружать не стал, т.к. они могут занулить важность собранных признаков. Решил ориентироваться на первоисточники.



## EDA and FeatureEngineering

```
to_big_to_fail = ['Авангард АКБ', 'АК БАРС Банк', 'Альфа-Банк', 'Аэрофлот', 'ББР Банк', 'БКС Банк',  
    'Балтика АКБ', 'Банк Аверс', 'Банк ВБРР', 'Банк ВТБ', 'Банк ГПБ', 'Банк ДОМ.РФ', 'Банк Зенит', 'Банк НационалСтандарт',  
    'ВТБ СырьевТовары Фин', 'ВЭБ-лизинг', 'Вертолеты России', 'Внешпромбанк', 'Держава АКБ', 'Интерпромбанк КБ',  
    'МРСК Урала', 'МСП БАНК', 'МТС-Банк', 'МеталлИнвестБанк', 'Курская Обл Адм', 'МеталлИнвестБанк',  
    'МинУправлФинансСамарскОбл', 'Минфин Башкортостана', 'Минфин Калининград Обл', 'Минфин Камчатского края',  
    'Минфин Карелии', 'Минфин Кировской Обл', 'Минфин Коми', 'Минфин КраснодарскогоКрая', 'Минфин Красноярского Края',  
    'Минфин Кузбасса', 'Минфин Магаданск Обл', 'Минфин Марий Эл', 'Минфин НижегородскОбл', 'Минфин Новосибирской Обл',  
    'Минфин Омской Обл', 'Минфин Оренбургской Обл', 'Минфин РФ', 'Минфин Саратовской Обл', 'Минфин Саха (Якутия)',  
    'Минфин СвердлОбласти', 'Минфин СтавропольКрая', 'Минфин Тамбов Обл', 'Минфин Ульяновской Обл',  
    'Минфин Хабаровского Края', 'Минфин Хакасии', 'Минфин Челябинской Обл', 'Московский Кредитный Банк',  
    'Новосибирск Мария', 'Открытие Брокер', 'Открытие Холдинг', 'Промсвязьбанк', 'РМБ Банк', 'РН Банк',  
    'РОСНАНО', 'РСГ-Финанс', 'РСХБ', 'Ренессанс Кредит КБ', 'РосДорБанк', 'Россети Кубань', 'Россети Центр',  
    'Ростелеком', 'Росэксимбанк', 'СМП Банк', 'Сбербанк России', 'Совкомбанк', 'Солидарность КБ', 'ТГК-1',  
    'ТКБ БАНК ПАО', 'Татфондбанк', 'Тинькофф Банк', 'Удмуртия Прав', 'УправлениеФинансЛипецкОбл', 'УралКапиталБанк',  
    'ХКФ Банк', 'Экспобанк', 'Юго-Западная ТЭЦ']
```

Первым делом убрал организации, которые будут спасать регуляторы. Ну и пришлось убрать банки и госучреждения, поскольку их в дефолтных не осталось. Обучать их не на чем. По банкам у ЦБ РФ своя политика, отличная от политики по другим организациям.



## EDA and FeatureEngineering

```
# if тупо сплюсовать эти колонки получается каша, можно еще попробовать отдать преимущество АКРА
df['Рейтинг агрег'] = df.apply(lambda x: x['Рейтинг агрег. Эксперт РА'] if\
    x['Рейтинг агрег. Эксперт РА'] != 'NONE' else x['Рейтинг агрег. АКРА'], axis=1)
```

```
df['Рейтинг эмиссии'] = df.apply(lambda x: x['Рейтинг эмиссии Эксперт РА'] if\
    x['Рейтинг эмиссии Эксперт РА'] != 'NONE' else x['Рейтинг эмиссии АКРА'], axis=1)
```

```
df.drop(columns=['Рейтинг агрег. Эксперт РА', 'Рейтинг агрег. АКРА',
    'Рейтинг эмиссии Эксперт РА', 'Рейтинг эмиссии АКРА'], inplace=True)
```

```
# Возвращаю обратно пропуски
df['Рейтинг агрег'] = df['Рейтинг агрег'].apply(lambda x: None if x=="NONE" else x)
df['Рейтинг эмиссии'] = df['Рейтинг эмиссии'].apply(lambda x: None if x=="NONE" else x)
```

В связи с тем что колонки с рейтингами имеют повышенное количество пропусков, объединил эти признаки создав сначала из четырех признаков два.

Потом выяснилось что оба оставшихся рейтинга друг друга дублируют, за исключением того, что немного различается само обозначение, но в "Рейтинг эмиссии" пропусков больше. Его удаляю.



## EDA and FeatureEngineering

```
def fixing_nan(df, cnt_nan = 40):  
    """ Функция создает колонки выявляющие признаки,  
    в которых количество None больше чем cnt_nan """  
    for col in df.columns:  
        if df[col].isna().sum():  
            df[f"{col}_nan"] = 1  
            df[f"{col}_nan"] = df[f"{col}_nan"] * df[col].isna()  
            if len(df.loc[df[f"{col}_nan"] == 1]) < cnt_nan:  
                df.drop(columns=[f"{col}_nan"], inplace=True)
```

С помощью этой функции создал признаки характеризующие пропуски в соответствующих признаках с пропущенными значениями.

Выяснилось, что пропущенные значения в признаке «**Дата начала 1-го купона**» с точностью 100% выявляют дефолтных эмитентов. У нескольких остальных созданных таким способом признаков тоже прослеживаются источники интересной информации. Для возможности использования такой информации требуется экспертное мнение, поэтому в своем решении я заполнял пропущенные значения медианой, т.к. более продвинутые способы машинного обучения на столь малом датасете не доступны. Только в признаке рейтингов заполнил пропущенные значения новой категорией.



## EDA and FeatureEngineering

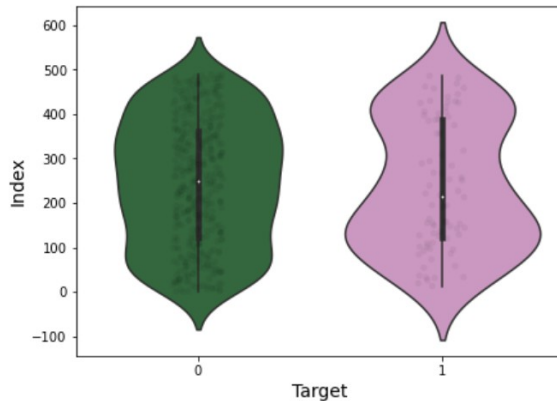
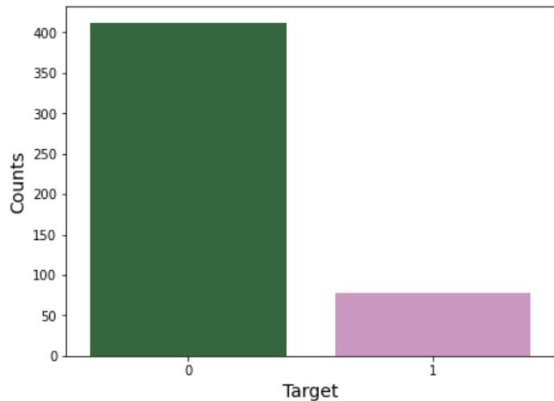
```
def create_datetime_features(X, time_feature, year, month, day, copy = True):  
    """ Создание datetime-признаков  
    Parameters  
    _____  
    X: pd.DataFrame - обучающая выборка  
    copy: bool = True - булевый флаг использования копии выборки  
    time_feature - временная фича  
    year - год начальной даты  
    month - месяц начальной даты  
    day - день начальной даты  
    """  
  
    if copy:  
        X = X.copy()  
  
    base = pd.datetime(year, month, day)  
    X[time_feature] = X[time_feature].apply(lambda x: datetime.timedelta(seconds = x))  
    X[time_feature] = X[time_feature].apply(lambda x: base + x)  
  
    X[f"{time_feature} year"] = X[time_feature].dt.year
```

С помощью этой функции преобразовал временные признаки. Поскольку стояла задача создать MVP, не стал глубоко прорабатывать временные признаки. Только преобразовал их в признаки года, месяца и дня событий. И дальнейшим анализом не занимался.



# EDA and FeatureEngineering

Target Distribution Analysis



Простейший анализ целевой переменной показал, что имеется большой дисбаланс классов. Распределения этого признака на дефолтных и не дефолтных данных разное.





# EDA and FeatureEngineering

```
df_numerical.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Номинал	490.0	6.386991e+05	9.709639e+06	1.0000	1.000000e+03	1.000000e+03	1.000000e+03	1.902000e+08
Кол-во сделок за 2 дня	306.0	1.564085e+02	3.020809e+02	0.0000	3.000000e+00	6.350000e+01	1.875000e+02	2.781000e+03
Объем торгов за 2 дня	306.0	6.542237e+06	3.152375e+07	0.0000	6.916450e+03	7.373867e+05	2.965249e+06	4.718941e+08
Цена, %	254.0	9.559351e+01	1.782850e+01	3.1000	9.734500e+01	9.983500e+01	1.012350e+02	1.156200e+02
НКД, в вал. ном.	252.0	4.739253e+01	2.394465e+02	0.0000	5.737500e+00	1.292000e+01	2.491750e+01	2.416440e+03
Кол-во сделок за 10 дней	306.0	7.334216e+02	1.327927e+03	0.0000	1.950000e+01	2.935000e+02	8.330000e+02	1.122700e+04
cnt_bonds	490.0	2.416327e+00	3.966390e+00	1.0000	1.000000e+00	1.000000e+00	3.000000e+00	6.100000e+01
Дневной оборот на МБ	248.0	5.142249e+06	3.306459e+07	85.0000	1.004188e+05	4.980170e+05	1.563704e+06	4.717593e+08
Кол-во дней со сделками за месяц	306.0	1.408170e+01	7.817374e+00	0.0000	9.250000e+00	1.900000e+01	1.900000e+01	1.900000e+01
Дюрация модиф., лет	252.0	1.484598e+00	1.343950e+00	0.0000	7.870250e-01	1.276900e+00	1.871025e+00	1.295280e+01
Кол-во сделок за месяц	306.0	1.478376e+03	2.538408e+03	0.0000	3.700000e+01	6.035000e+02	1.711250e+03	1.906300e+04
Доходность эффект., %	289.0	1.389346e+04	1.170152e+05	-23.9695	1.053210e+01	1.319470e+01	1.652170e+01	9.999000e+05
Объем торгов за 5 дней	306.0	1.260190e+07	4.125739e+07	0.0000	3.307648e+04	2.272031e+06	8.975937e+06	4.720750e+08
Кол-во сделок	306.0	7.491830e+01	1.496427e+02	0.0000	2.500000e+01	2.800000e+01	8.375000e+01	1.380000e+03
Объем торгов за 10 дней	306.0	2.177415e+07	6.668183e+07	0.0000	7.553379e+04	4.294826e+06	1.715413e+07	6.868190e+08
Объем торгов за месяц	306.0	5.624503e+07	1.828671e+08	0.0000	1.768115e+05	9.006507e+06	3.840840e+07	2.066102e+09
Средний дневной оборот за мес.	304.0	4.108354e+06	2.025729e+07	0.0000	1.079625e+04	4.916630e+05	2.025437e+06	3.047435e+08
Ставка купона, %	490.0	8.571892e+00	6.047770e+00	0.0000	4.010989e-01	9.800000e+00	1.300000e+01	2.400000e+01
Кол-во сделок за 5 дней	306.0	3.832516e+02	7.106403e+02	0.0000	8.000000e+00	1.605000e+02	4.537500e+02	6.657000e+03
Z-спред	270.0	2.245385e-01	7.607982e-01	0.0000	0.000000e+00	0.000000e+00	0.000000e+00	5.650300e+00
Дюрация, лет	341.0	1.867818e+00	1.993688e+00	0.0027	7.726000e-01	1.474000e+00	2.268500e+00	1.751230e+01
Доходность средневзв., %	248.0	1.478683e+04	1.169511e+05	-23.9695	1.036085e+01	1.246840e+01	1.542892e+01	9.999000e+05
Заявл. объем эмиссии, в вал. ном.	490.0	4.780342e+09	1.821142e+10	510.0000	1.127095e+08	6.882015e+08	4.000000e+09	3.403700e+11
Объем эмиссии, шт.	490.0	4.186015e+06	1.852272e+07	2.0000	1.000000e+05	5.000825e+05	3.162064e+06	3.800000e+08
Объем торгов	306.0	4.701670e+06	3.036681e+07	0.0000	2.130000e+01	3.010594e+05	1.289819e+06	4.717593e+08

Почистил количественные признаки. Анализ показал их сильную неоднородность и большое количество выбросов. Но для нелинейной модели классификации это не представляет большой проблемы, если их правильно обработать.

В связи с немерено большим для такого количества наблюдений количества признаков принял решение все эти количественные признаки сжать методом PCA до трех признаков. Тем более классификаторы, использующие статистические методы, не терпят корреляции между признаками.

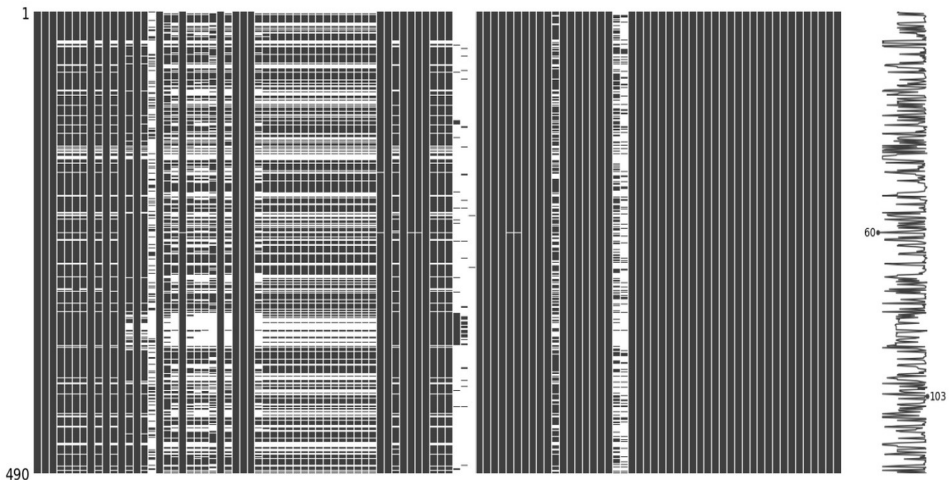
После завершения работы с категориальными признаками можно будет эти разжать обратно и выявить значимые признаки уже из количественных признаков.



# EDA and Feature Engineering

```
msno.matrix(df)
```

```
df.isnull().sum()
```



Провел анализ количественных признаков, получился слишком большой объем чтобы его описать в презентации. Хочу обратить внимание, что имеется достаточно много наблюдений, в которых количественные признаки вообще отсутствуют, что несомненно уменьшает их значимость.

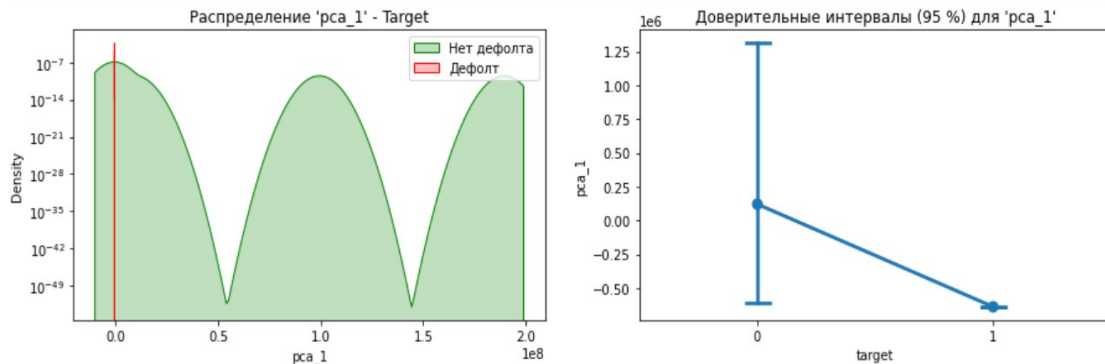
В задачах классификации в мощных алгоритмах выбросы имеют низкое значение, и необходимо, чтобы алгоритм на них тоже обучился. Многие мощные алгоритмы не умеют экстраполировать данные, и чтобы он мог предсказывать на выбросах, необходимо его на выбросах обучить.

В анализе категориальных признаков никаких экстраординарных моментов не выявлено.

Последующий анализ признаков будет уже первым этапом предсказания. Поэтому буду проводить его в ноутбуке Prediction.



## Prediction



```
Параметрический критерий Шапиро-Уилка pvalue = 0.0000.  
Распределение не нормальное.  
Непараметрический критерий Манна-Уитни pvalue = 0.0000  
Согласно значению p-value, гипотеза о равенстве мат. ожиданий отвергается.
```

Наличие взаимной информации между количественными признаками и таргетом проверял в первую очередь на доверительных интервалах. Ну и сверялся с критериями Шапиро-Уилка и U-критерия Манна – Уитни. Но статистическим критериям я доверяю в последнюю очередь. Когда данных в выборке мало, всегда принимается нулевая гипотеза. Когда слишком много – нулевая гипотеза всегда отвергается. А критерии мало-много в статистике слишком размытые. Именно по этой причине появилось машинное обучение.



# Prediction

```
print(f"target=1_min: {df.loc[df['target']==1]['pca_2'].min()}")
print(f"target=0_min: {df.loc[df['target']==0]['pca_2'].min()}")
print(f"target=1_max: {df.loc[df['target']==1]['pca_2'].max()}")
print(f"target=0_max: {df.loc[df['target']==0]['pca_2'].max()}")
```

```
target=1_min: -2019.8420512077305
target=0_min: -2029.3219590587664
target=1_max: 256242.55089907823
target=0_max: 563.3020203467081
```

```
# доверительный интервал 95% для единиц
norm.interval(alpha=0.05,
              loc=np.mean(df.loc[df['target']==1]['pca_2']),
              scale=sem(df.loc[df['target']==1]['pca_2']))
```

```
(-1555.0671724168897, 22761.538455782327)
```

```
# доверительный интервал 95% для нулей
norm.interval(alpha=0.05,
              loc=np.mean(df.loc[df['target']==0]['pca_2']),
              scale=sem(df.loc[df['target']==0]['pca_2']))
```

```
(-2021.5813922964448, -1993.235986593129)
```

```
# Создам признак, отделяющий единицы по таргету
df['pca_2_1_max'] = None
df['pca_2_1_max'] = df.apply(lambda x: 1 if x['pca_2'] > -1000.235986593129 else 0, axis=1)
df['pca_2_1_max'].value_counts()
```

```
0    473
1     17
Name: pca_2_1_max, dtype: int64
```

Из распределений видно, что для предсказаний эти признаки можно попробовать дополнить категориальными. Например, вот таким образом. Гиперпараметр «доверительный интервал» не настраивал.



## Prediction

```
def chi_2_box(df, feature, TARGET_NAME):  
    """Функция предсказания наличия взаимной информации с таргетом у категориальных признаков"""  
  
    df_for_table = df[['id', feature, TARGET_NAME]]  
    table = df_for_table.pivot_table(values='id', index = feature,  
                                     columns= TARGET_NAME, aggfunc='count')  
  
    chi2, p, _, _ = chi2_contingency(table)  
    print(f"p_value chi_2: {p}")  
    print(f"Распределение внутри признака: {df[feature].value_counts()}")  
  
    fig = px.box(df, color = TARGET_NAME, y = feature, points="all", title = feature)  
    fig.show()
```

```
categorical_features = df.select_dtypes(exclude=[np.number])  
categorical_features.T
```

С помощью этой функции провел поиск взаимной информации между категориальными признаками и таргетом.



# Prediction

Бесполезные: 'Кол-во дней со сделками за 2 дня', 'Амортизируемость', 'Закрытая подписка', 'not\_isin', 'Encumbrance', 'Дата гос. рег-ции/присвоения\_dow', 'Дата ближ. купона\_dow', 'Состояние бумаги\_eng', 'Срок обращения', 'Вид облигации', 'Способ размещения', 'Индексируемость номинала',

Отсекают не дефолтных: 'Ломбардный список БР', 'Кол-во дней со сделками за 5 дней', 'Кол-во дней со сделками за 10 дней', 'Конвертация', 'Дата начала размещения\_dow', 'Дата погашения\_dow', 'Дата окончания размещения\_dow', 'Дата ближ. купона\_year', 'Дата начала размещения\_day', 'Дата ближ. купона\_day', 'Дата ближ. возм. погаш.\_month', 'Дата ближ. купона\_month', 'Дата гос. рег-ции/присвоения\_month', 'Дата гос. рег-ции/присвоения\_year', 'Дата начала размещения\_month', 'Дата окончания размещения\_month', 'Дата погашения\_month', 'Страна заемщика', 'Сектор заемщика', 'Сектор эмитента', 'Тип купона', 'Валюта номинала', 'Тип облигации', 'Для некавал. инвесторов', 'Форма выпуска\_eng', 'Тип по классиф. БР', 'Вид залога', 'Структурированность', 'Гос. гарантия', 'Вид обеспечения', 'Старшинство секьюритизации', 'Тип бумаги (МСП, Корп., Гос.)', 'Сектор повышенного инвест. риска', 'Сектор роста',

Отсекают дефолтных: 'Кол-во купонов в год', 'CFI', 'Регион эмитента', 'Рейтинг\_agree', 'Дата ближ. возм. погаш.\_year', 'Дата погашения\_year',

Разные распределения, но интервалы одинаковые: 'Дата ближ. возм. погаш.\_dow', 'Дата регистрации\_dow',

Разные распределения и интервалы: 'Дата ближ. возм. погаш.\_day', 'Дата погашения\_day', 'Дата окончания размещения\_day', 'Дата регистрации\_day', 'Дата гос. рег-ции/присвоения\_day', 'Дата начала размещения\_year', 'Дата окончания размещения\_year', 'Дата регистрации\_month', 'Дата регистрации\_year',

Удалить 'Статус'

На основе полученных данных провел анализ, и разделил категориальные признаки по таким категориям. Бесполезные, для уменьшения «проклятия размерности» удалил.



## Prediction

```
# OneHotEncoder увеличит кол-во признаков в несколько раз
encoder_list = [ce.backward_difference.BackwardDifferenceEncoder,
                ce.basen.BaseNEncoder,
                ce.binary.BinaryEncoder,
                ce.cat_boost.CatBoostEncoder,
                #ce.hashing.HashingEncoder,
                ce.helmert.HelmertEncoder,
                ce.james_stein.JamesSteinEncoder,
                #ce.one_hot.OneHotEncoder,
                ce.leave_one_out.LeaveOneOutEncoder,
                ce.m_estimate.MEstimateEncoder,
                ce.ordinal.OrdinalEncoder,
                #ce.polynomial.PolynomialEncoder,
                ce.sum_coding.SumEncoder,
                ce.target_encoder.TargetEncoder,
                ce.woe.WOEEncoder
                ]
```

Далее провел выбор кодировки категориальных признаков из следующих энкодеров на байесовском наивном классификаторе. Одним из лучших оказался энкодер Катбуста, его и выбрал.

В связи с неоднозначным датасетом, далее метрики считаю строго на кроссвалидации. Поскольку у выбранного классификатора гиперпараметры отсутствуют, тестовая выборка не требуется.

Целевой метрикой выбрал ROC\_AUC (далее «метрика») в связи с тем, что для нее не требуется вычисление трешхолда, требующееся для вычисления большинства основных метрик классификации.



## Prediction

```
cv_scores = cross_val_score(model_Baies, X_catb, y, cv=3, scoring='roc_auc', n_jobs=-1)
cv_score = np.mean(cv_scores)
print('CV score is {}'.format(cv_score))
cv_std = np.std(cv_scores) * 2
print('CV var is {}'.format(cv_std))
```

```
CV score is 0.3800760577974161
CV var is 0.06242304072961585
```

```
model_QDA = QuadraticDiscriminantAnalysis()

#запустим кросс-валидацию
kfold_cv = KFold(n_splits=3, shuffle=True, random_state=0)
cv_scores = cross_val_score(model_QDA, X_catb, y, cv=kfold_cv, scoring='roc_auc', n_jobs=-1)
cv_score = np.mean(cv_scores)
print('CV score is {}'.format(cv_score))
cv_std = np.std(cv_scores) * 2
print('CV var is {}'.format(cv_std))
```

```
CV score is 0.7847477519733617
CV var is 0.06871587816709768
```

На полученном датасете вычислил метрику с использованием Наивного Байеса и родственном ему нелинейном классификаторе QDA. QDA показал допустимую метрику, но слишком большой разброс метрики по фолдам кроссвалидации.





## Prediction

```
def balance_df_by_target(df, target_name, method='over'):
    """ Функция для исправления дисбаланса классов """

    assert method in ['over'], 'Неверный метод сэмплирования'

    np.random.seed = 1

    target_counts = df[target_name].value_counts()

    major_class_name = target_counts.argmax()
    minor_class_name = target_counts.argmin()

    disbalance_coeff = int(target_counts[major_class_name] / target_counts[minor_class_name]) - 1

    if method == 'over':
        for i in range(disbalance_coeff):
            sample = df[df[target_name] == minor_class_name].sample(target_counts[minor_class_name])
            df = df.append(sample, ignore_index=True)

    return df.sample(frac=1)
```

С помощью этой функции провел балансировку датасета по таргету методом oversampling. Это единственно доступный в нашем случае метод, поскольку синтетические методы на таких малых датасетах не работают.

В результате количество наблюдений целевого события в датасете увеличилось до 390.



# Prediction

На полученном датасете у Наивного Байеса по прежнему метрика не удовлетворительная. А модели LDA и QDA показали максимальную метрику с практически нулевым разбросом метрики по фолдам кроссвалидации.

```
cv_scores = cross_val_score(model_Baies, X_over, y_over, cv=5, scoring='roc_auc', n_jobs=-1)
cv_score = np.mean(cv_scores)
print('CV score is {}'.format(cv_score))
cv_std = np.std(cv_scores) * 2
print('CV var is {}'.format(cv_std))
```

```
CV score is 0.40801719437866696
CV var is 0.1022928142069396
```

```
model_LDA = LinearDiscriminantAnalysis(solver='svd')

#запустим кросс-валидацию
cv_scores = cross_val_score(model_LDA, X_over, y_over, cv=5, scoring='roc_auc', n_jobs=-1)
cv_score = np.mean(cv_scores)
print('CV score is {}'.format(cv_score))
cv_std = np.std(cv_scores) * 2
print('CV var is {}'.format(cv_std))
```

```
CV score is 0.996024010925343
CV var is 0.0031279471675714563
```

```
model_QDA = QuadraticDiscriminantAnalysis()

#запустим кросс-валидацию
cv_scores = cross_val_score(model_QDA, X_over, y_over, cv=5, scoring='roc_auc', n_jobs=-1)
cv_score = np.mean(cv_scores)
print('CV score is {}'.format(cv_score))
cv_std = np.std(cv_scores) * 2
print('CV var is {}'.format(cv_std))
```

```
CV score is 1.0
CV var is 0.0
```



## Prediction

	model_name	Precision	Recall	F_Score	Threshold	CV_AUC	CV_STD
0	CatBoostClassifier	1.000000	1.000000	1.000000	0.505686	1	0.001000
1	LogisticRegression	1.000000	1.000000	1.000000	0.459545	0.99026	0.008002
2	LinearSVC	0.871287	1.000000	0.931217	0.335394	0.801487	0.440114
3	XGBClassifier	1.000000	0.977273	0.988506	0.513841	0.994822	0.007622
4	LGBMClassifier	0.880435	0.920455	0.900000	0.457784	0.5	0.000000
5	RandomForestClassifier	0.546584	1.000000	0.706827	0.497931	0.5	0.000000
6	ExtraTreesClassifier	0.546584	1.000000	0.706827	0.500000	0.5	0.000000

Также полученный датасет прогнал через свой пайплайн выбора модели классификации. Гиперпараметры бустингов настраивал с помощью BayesianOptimization. Гиперпараметры бэггингов настраивал с помощью Huperopt. Линейные модели настраивал с помощью GridSearchCV. Выбрать не из чего :-). Линейные модели не найдут нелинейные связи между признаками. Нелинейные дают либо слишком сильный скор либо очень плохой. Идеальный результат у Катбуста, но бустинги склонны к переобучению. А у QuadraticDiscriminantAnalysis тоже все идеально, и к тому же это нелинейная статистическая модель близкая к байесовскому классификатору, и в отличие от Катбуста не требует настройки гиперпараметров, а значит не требуется валидационная выборка, что особенно важно на столь маленьком датасете. Выбираю QuadraticDiscriminantAnalysis.



## Prediction

```
)  
importance_scores = importance_scores.sort_values(  
    by="importance-mean", ascending=False  
)  
importance_scores = importance_scores.reset_index(drop=True)  
decrease_scores = importance_scores[importance_scores["importance-mean"] <= 0]  
decrease_scores = decrease_scores.reset_index(drop=True)  
decrease_scores
```

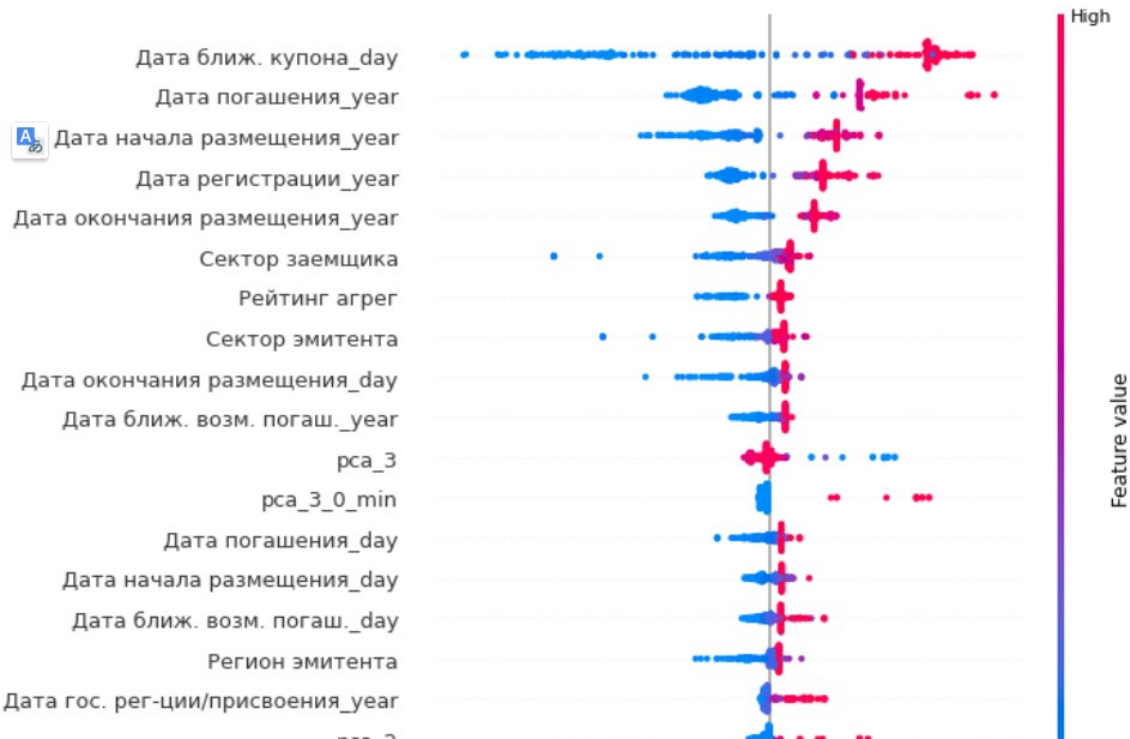
С помощью **Permutation Importance** нашел бесполезные признаки, которые удалил. Важность для Permutation Importance признаков специально не вывожу, поскольку в этом плане следующий метод точнее.

	features	importance-mean	importance-std
0	Конвертация	0.0	0.0
1	Тип бумаги (МСП, Корп., Гос.)	0.0	0.0
2	Страна заемщика	0.0	0.0
3	id	0.0	0.0



## Prediction

```
# summarize the effects of all the features  
shap.summary_plot(shap_values, x_valid)
```



Дальнейшее исследование признаков провожу в одном из самых мощных инструментов машинного обучения – SHAP.

К сожалению SHAP с QDA не работает, поэтому обучил под SHAP CatBoostClassifier.

На мой взгляд CatBoostClassifier переобучился на временных признаках, но требуется дальнейшее изучение датасета методами SHAP. Два из которых я привел в ноутбуке.



## Воспроизводимость ноутбуков

```
!pip freeze > pred_requirements.txt
```

```
tzlocal==4.2
```

```
ujson @ file:///tmp/build/80754af9/ujson_1611259522456/work
```

```
unicodcsv==0.14.1
```

```
updater==0.3.2
```

```
uritemplate==4.1.1
```

```
urllib3==1.26.11
```

```
watchdog==2.1.9
```

```
wcwidth @ file:///Users/ktietz/demo/mc3/conda-bld/wcwidth_1629357192024/work
```

```
webencodings==0.5.1
```

```
: 0%
```

```
werkzeug==2.2.1
```

Для каждого ноутбука я создал файл `..._requirements.txt`. Рекомендую перед запуском ноутбука создать его виртуальное окружение, например, в Anaconda, и загрузить требуемые в `requirements` библиотеки. Только я, к сожалению, торопился и отдельное виртуальное окружение для ноутбуков не создавал. Извините.

Лайфхак: в Google Collab все ноутбуки точно запустятся.



## Предложения по проекту

Для существующего решения:

В первую очередь более качественно сделать датамайнинг и проработать предобработку данных. Распаковать из PCA количественные признаки и проработать их важность. На базе существующих признаков создать новые признаки и их исследовать. В результате важность существующих признаков может существенно измениться. Намайнить новые данные и на их основе добавить новые признаки. Например, из метода API /Emitent/Profile найти описания и проработать предсказание дефолта на основе этой информации, например, методами как в этой моей работе: [https://](https://github.com/DAYa66/Huggingface_models_fine_tuning/blob/main/RuBert_NER_ru.ipynb)

[github.com/DAYa66/Huggingface\\_models\\_fine\\_tuning/blob/main/RuBert\\_NER\\_ru.ipynb](https://github.com/DAYa66/Huggingface_models_fine_tuning/blob/main/RuBert_NER_ru.ipynb)

Это самое мощное решение на базе трансформеров, хотя в NLP есть куча разных нюансов, которые здесь не описать. Для стабилизации классификации проработать различные способы ансамблирования классификационных моделей. В первую очередь найти решения с помощью нейронных сетей, т.к. их предсказания будут менее всего коррелированы с уже найденными качественными классификаторами. Но для них скорее всего нужна более сложная предобработка данных. Т.к. датасет нестабильный, использовать для проверки гипотез более точные статистические методы выбора гипотез. Ну и существуют различные полезные лайфхаки по работе с классификаторами. Попробовать методы обучения без учителя в помощь классификации.



## Предложения по проекту

Для нового решения:

На мой взгляд более качественным будет решение, основанное на анализе признаков во времени. В первую очередь бухгалтерских, финансовых, макроэкономических, признаках, основанных на классификации аналитических статей. Будет очень здорово, если получится получить сильные предсказания нового датасета на основе рекуррентных нейросетей, трансформеров.