

# Машинное обучение в финансах

## Лекция 8: Подготовка данных. Feature engineering

Роман В. Литвинов\*

\*CRO  
Финансовая Группа БКС

Высшая школа экономики, Апрель 2021

# Содержание

- 1 Процесс решения задачи с помощью методов ML
- 2 Загрузка данных и обзор
- 3 Чистка и подготовка данных
- 4 Feature engineering
- 5 Работа с высококоррелированными предикторами

# Содержание

- 1 Процесс решения задачи с помощью методов ML
- 2 Загрузка данных и обзор
- 3 Чистка и подготовка данных
- 4 Feature engineering
- 5 Работа с высококоррелированными предикторами

# Процесс решения задачи с помощью методов ML

Верхнеуровнного процесс решения какой-либо задачи с помощью методов машинного обучения можно разбить на следующие этапы:

- ① поиск данных, необходимых для решения нашей проблемы;
- ② загрузка и очистка данных (импорт, заполнение пропусков, зачистка выбросов, удаление некорректных данных);
- ③ feature engineering (построение на базе очищенных данных характеристик/факторов, с которыми будет работать модель);
- ④ обучение модели/моделей ML. Выбор оптимальной модели;
- ⑤ имплементация модели в 'промышленной' среде (пром);
- ⑥ мониторинг поведения модели, донастройка, управление рисками.

# Процесс решения задачи с помощью методов ML

На сегодняшней лекции (и практическом занятии) мы сосредоточимся на этапах 2 и 3.

По моим скромным оценкам, этапы, связанные с конструированием и поддержанием качественного дата сета, могут занимать до 60 процентов времени отведенного на проект. Возможно больше.

Данные это фундамент любого проекта, связанного с ML (*garbage in/garbage out*).

В рамках наших задач - предсказания форвардной доходности акции в качестве 'сырых' данных могут выступать: котировки акций компаний/ других компаний (*cross-correlation*), балансовые показатели, макроэкономические показатели, данные твиттер и тп.

На основе подобных данных мы будем конструировать факторы, о которых мы говорили на лекции 6.

# Содержание

- 1 Процесс решения задачи с помощью методов ML
- 2 Загрузка данных и обзор
- 3 Чистка и подготовка данных
- 4 Feature engineering
- 5 Работа с высококоррелированными предикторами

# Загрузка данных и обзор

Мы уже сталкивались с вопросами импорта данных на лекции о Pandas (Лекция 5).

Сегодня мы будем работать с Excel файлом, содержащим данные с котировками акций и фундаментальными показателями компании Apple. Файл содержит данные за период с начала 2000 по середину 2019 гг.

Как и раньше, первое, что от нас требуется, это импортировать данные с помощью Pandas:

```
filename = 'apple_data_v2.xlsx'  
data = pd.read_excel(filename,  
                     sheet_names='APPLE_data', index_col=0)
```

Теперь с помощью команды `data.info()` проверим, успешно ли загрузились данные, и что они из себя представляют.

# Загрузка данных и обзор

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 7115 entries, 2000-01-01 to 2019-06-24
Data columns (total 27 columns):
PX_LAST                               7115 non-null float64
BEST_EPS                               973 non-null float64
BS_LT_BORROW                           7115 non-null int64
BS_ST_BORROW                           7115 non-null int64
BS_TOT_ASSET                            7115 non-null int64
CF_CASH_FROM_OPER                      7115 non-null int64
EBITDA                                 7115 non-null int64
EQY_DVD_YLD_IND_NET                   2527 non-null float64
EQY_SH_OUT                             7115 non-null float64
EPS_Annualized                         7115 non-null float64
IS_Diluted_EPS                          7115 non-null float64
Net_income                             7115 non-null int64
IS_EPS                                 7115 non-null float64
IS_INC_BEF_X0_ITEM                     7115 non-null int64
SALES_REV_TURN                          7115 non-null int64
PX_TO_BOOK_RATIO                        7115 non-null float64
PE_RATIO                               7115 non-null float64
PX_TO_CASH_FLOW                         7115 non-null float64
CUR_MKT_CAP                            7115 non-null float64
Current_EV_to_t12m_ebitda              7115 non-null float64
PX_VOLUME                              7115 non-null int64
CASH_AND_MARKETABLE_SECURITIES        7115 non-null int64
GROSS_PROFIT                           7115 non-null int64
CF_FREE_CASH_FLOW                      7115 non-null int64
ENTERPRISE_VALUE                        7115 non-null float64
TOT_COMMON_EQY                          7115 non-null int64
BOOK_VAL_PER_SH                         7115 non-null float64
dtypes: float64(14), int64(13)
memory usage: 1.5 MB
```

## Загрузка данных и обзор

Из данных видно, что не все метрики покрывают анализируемый период. Например, по BEST-EPS есть только 973 наблюдения (из 7 115) и по EQY-DVD-YLD-IND-NET только 2 527 наблюдений.

Учитывая столь низкое покрытие выборки этими метриками, логичнее будет не думать о том, как нам заполнить эти пропуски, а просто отбросить эти столбцы.

```
data.dropna(axis='columns', inplace=True)
```

После этого, с помощью метода `data.head()`, посмотреть как выглядят верхние строки получившегося дата сета.

Dates	PX_LAST	BEST_EPS	BS_LT_BORROW	BS_ST_BORROW	BS_TOT_ASSET	CF_CASH_FROM_OPER	EBITDA
2000-01-01	3.6719	NaN	300	0	7586	373	120
2000-01-02	3.6719	NaN	300	0	7586	373	120
2000-01-03	3.9978	NaN	300	0	7586	373	120
2000-01-04	3.6607	NaN	300	0	7586	373	120

# Загрузка данных и обзор

Далее целесообразно просмотреть статистические характеристики нашего набора данных с помощью команды

```
data.describe().round(2)
```

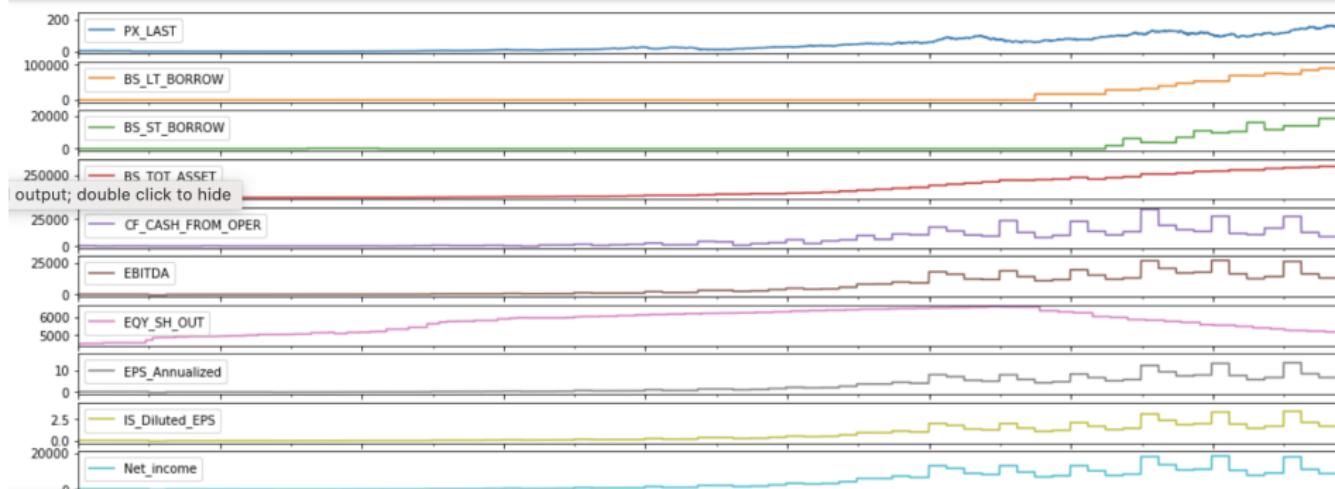
	PX_LAST	BEST_EPS	BS_LT_BORROW	BS_ST_BORROW	BS_TOT_ASSET	CF_CASH_FROM_OPER	EBITDA	EPS
count	7115.00	973.00	7115.00	7115.00	7115.00	7115.00	7115.00	7115.00
mean	54.79	2.29	19140.05	3444.04	121712.54	7434.82	7827.51	1.97
std	58.82	0.16	33219.54	6640.25	130958.48	8487.48	8480.54	0.00
min	0.94	1.97	0.00	0.00	5986.00	-125.00	-396.00	0.00
25%	4.57	2.12	0.00	0.00	8050.00	283.00	193.00	0.00
50%	27.20	2.27	0.00	0.00	48140.00	3938.00	3887.00	0.00
75%	95.24	2.43	28987.00	2010.00	225184.00	12523.00	15277.00	0.00
max	232.07	2.59	103922.00	22429.00	406794.00	33722.00	29019.00	0.00

8 rows × 27 columns

# Загрузка данных и обзор

Следующий шаг, который я обычно делаю, это визуальная инспекция данных с помощью команды

```
data.plot(figsize=(20, 20), subplots=True);
```



Как вы думаете о чем говорят эти 'пилы' на графике?

# Содержание

- 1 Процесс решения задачи с помощью методов ML
- 2 Загрузка данных и обзор
- 3 Чистка и подготовка данных
- 4 Feature engineering
- 5 Работа с высококоррелированными предикторами

# Чистка и подготовка данных

Частая проблема, с которой приходится сталкиваться, это пропуски в данных. Их необходимо заполнить подходящими 'синтетическими' значениями.

Решить эту проблему можно следующими основными способами:

- считать, что отсутствующее значение переменной равно предыдущему не пустому;
- считать, что отсутствующее значение переменной равно некому среднему (недельному, месячному, среднему между предшествующим и последующим);
- заполнить пропущенные данные с помощью методов машинного обучения (например, KNN).

## Чистка и подготовка данных

Другая распространенная проблема - это борьба с выбросами (outliers) - аномальными нереалистичными значениями переменных, которые (скорее всего) носят ошибочный характер.

Наша задача идентифицировать такие аномальные значения (человек с ростом 1000м, дом с 3-мя тыс комнат или доходность - 3 млн процентов).

Самое простое решение - исключить такие наблюдения из выборки/ зачистить данные.

Следующая очень серьезная ошибка связана непосредственно со сферой финансов, встречается довольно часто, порой даже в серьезных публикациях. Я искренне надеюсь, что вы будете о ней всегда помнить и никогда не наступать на подобного рода 'грабли'.

## Чистка и подготовка данных

В большинстве данных, полученных от сторонних провайдеров (хотя, такие ошибки встречаются и при самостоятельной 'ручной' обработке данных), датой появления новых балансовых показателей компании будет являться дата, на которую составлена бухгалтерская отчетность.

В реальности эта информация появляется в публичных источниках и следовательно 'отражается' в котировках акций компании позже - в дату раскрытия отчетности. Такое раскрытие может происходить на ме или несколько мес позже даты, на которую составлена отчетность.

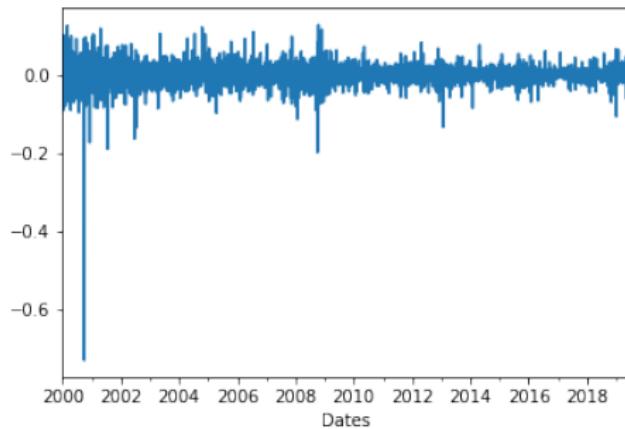
Что будет, если этот дефект не устраниТЬ? Не трудно догадаться. Машина будет 'видеть' данные, которых в реальности не существовало в указанный день (*back to future*) и строить прогноз на этих данных.

Что нужно сделать, чтобы устранить это? Правильное решение - 'сдвинуть' данные базируясь на официальных датах раскрытия отчетности (веб-скраппер, вручную и проч). Упрощенное - на некий фиксированный промежуток времени (минимум 1 мес).

## Чистка и подготовка данных

Следующий важный этап - подготовка данных, которые мы собираемся предсказывать - доходностей актива. Мы будем работать с лог-доходностями. Рассчитать однодневную лог-доходность можно с помощью следующей формулы:

```
data['1d_rets']=np.log(data['PX_LAST'] /  
                      data['PX_LAST'].shift(1))  
data['1d_rets'].plot() #визуализация результата
```



## Чистка и подготовка данных

Теперь перейдем непосредственно к тем переменным, которые мы будем прогнозировать с помощью методов ML. Как мы условились в начале курса, нашей основной задачей будет являться предсказание однومесячных форвардных доходностей актива.

Мы можем сконструировать ряд таких переменных следующим образом:

```
data['1m_forward_rets']=(np.log(data['PX_LAST']) /  
data['PX_LAST'].shift(30)).shift(30)
```

Мы с вами понимаем, что этот ряд подходит для задач связанных с регрессией.

Что можно придумать, если мы хотим также использовать методы классификации?

# Чистка и подготовка данных

Самая простая идея, которая приходит в голову, это заставить машину предсказывать положительную/отрицательную доходность (-/+)- бинарная классификация (0 или 1).

Мы же сконструируем более интересный пример, связанный с мультиномиальной классификацией. Машина будет генерировать сигнал, 'сила' которого зависит от того, насколько существенное (в терминах стандартных отклонений) нас ждет падение/ рост доходности акции (bullish vs bearish).

Signal description	Condition in terms of st.dev of one-month forward log-returns ( $r$ )
Extreme positive returns	$r > +2 \text{ st.dev}$
Very bullish	$+2 \text{ st.dev} \geq r > +1 \text{ st.dev}$
Bullish	$+1 \text{ st.dev} \geq r > +0.01^* \text{ st.dev}$
Neutral	$+0.01^* \text{ st.dev} \geq r > -0.01^* \text{ st.dev}$
Bearish	$-0.01^* \text{ st.dev} \geq r > -1 \text{ st.dev}$
Very bearish	$-1 \text{ st.dev} \geq r > -2 \text{ st.dev}$
Extreme negative returns	$r < -2 \text{ st.dev}$

# Содержание

- 1 Процесс решения задачи с помощью методов ML
- 2 Загрузка данных и обзор
- 3 Чистка и подготовка данных
- 4 Feature engineering
- 5 Работа с высококоррелированными предикторами

# Feature engineering

Теперь, на основании подготовленных данных, нам необходимо расчитать набор факторов, которые будут являться предикторами для наших моделей.

Мы будем рассчитывать полный набор (и даже больше) факторов, о которых мы говорили на 6-й лекции. Все это очень удобно реализовано в Pandas. Иногда нам понадобится инструментарий NumPy.

Ниже, для иллюстрации, приведен пример расчета фактора из группы 'моментум'.

```
data['momentum_3m']=np.log(data['PX_LAST'] /  
                           data['PX_LAST'].shift(90))
```

Всего, после завершения работы, у нас с вами получится порядка 36 факторов. Их мы будем использовать для обучения машины. Но пока с ними нужно еще поработать.

# Feature engineering

После первичной обработки данных нам может потребоваться их дополнительная трансформация. Можно выделить следующие основные методы трансформации численных данных:

Масштабирование (rescalling):

Иногда требуется скалировать фактороры таким образом, чтобы они имели значение, например, от 0 до 1 ли от -1 до 1. Распространенная техника называется мини-макс масштабирование (mi-max scaling). При мини-максе мы используем minimum и maximum предиктора, чтобы скалировать его следующим образом от 0 до 1:

$$x'_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

где  $x$  - вектор содержащий скалируемые данные.

# Feature engineering

Стандартизация (standardizing):

Такое масштабирование, при котором мы трансформируем ряд в стандартно нормально распределенный (имеющий среднее ноль и стандартное отклонение равное единице).

Делается это следующим образом:

$$x'_i = \frac{x_i - \text{mean}(x)}{\text{st.dev}(x)}$$

Другими распространенными техниками является полиномиальное преобразование предикторов (напр, возведение в квадрат или куб), дискретизация (разбиение непрерывных данных на дискретные наборы корзин/бинов).

# Feature engineering

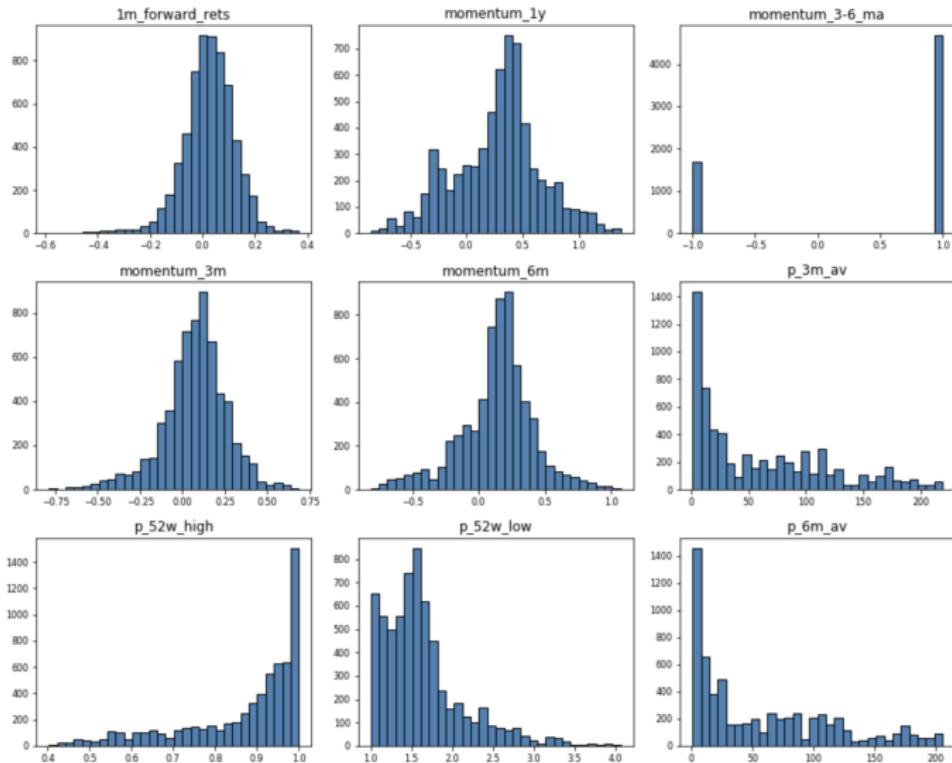
Для того, чтобы понять какие методы трансформаций имеет смысл применять, а также диагностировать проблемы с данными на ранней стадии имеет смысл визуализировать факторы и подробно проанализировать результат:

С чего я обычно начинаю, это построение набора диаграмм для того, чтобы проанализировать распределения факторов их нормальность, асимметричность, дискретность и тп.

Для начала нужно сгруппировать несколько факторов и затем с помощью средств Pandas вывести блок соответствующих гистограмм.

Ниже пример для группы моментум-факторов. Давайте его обсудим.  
Что здесь можно отметить?

# Feature engineering



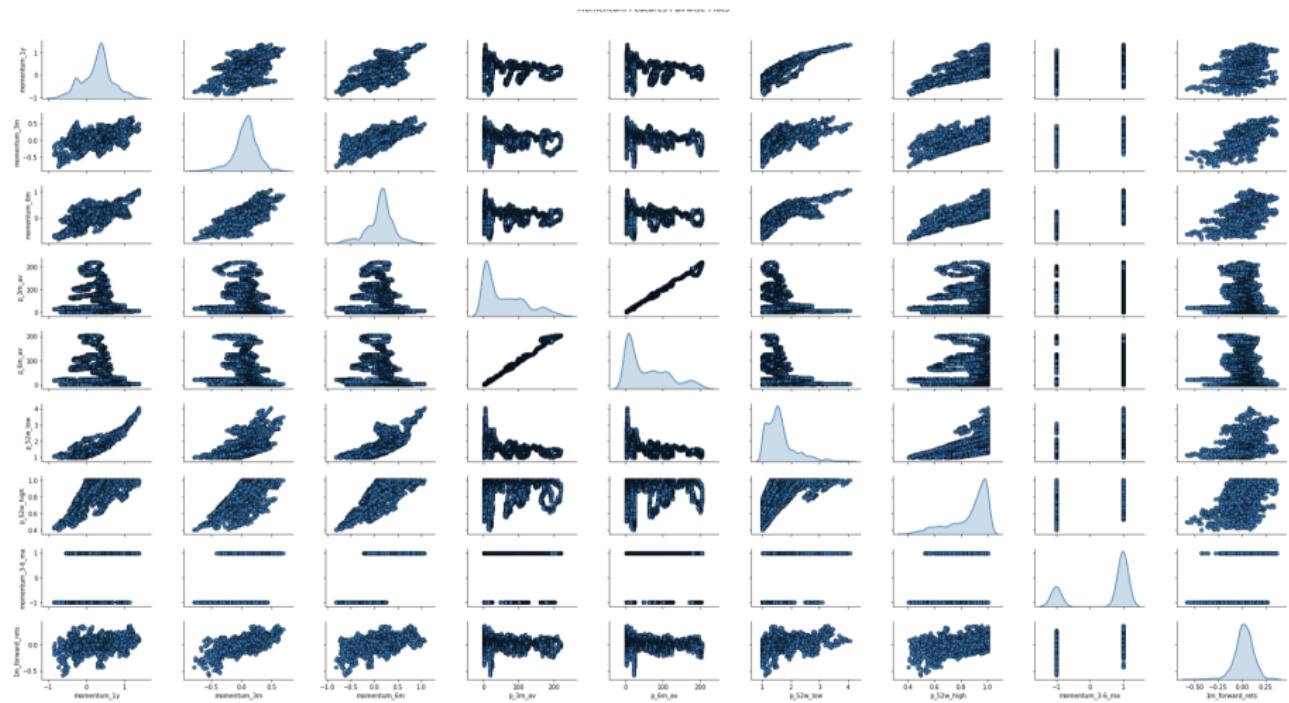
# Feature engineering

Следующим этап это визуальный анализ факторов/предикторов на предмет формы функциональной зависимости между ними и предсказываемой переменной. Линейна или нелинейна эта зависимость? Монотонна ли она? и тп.

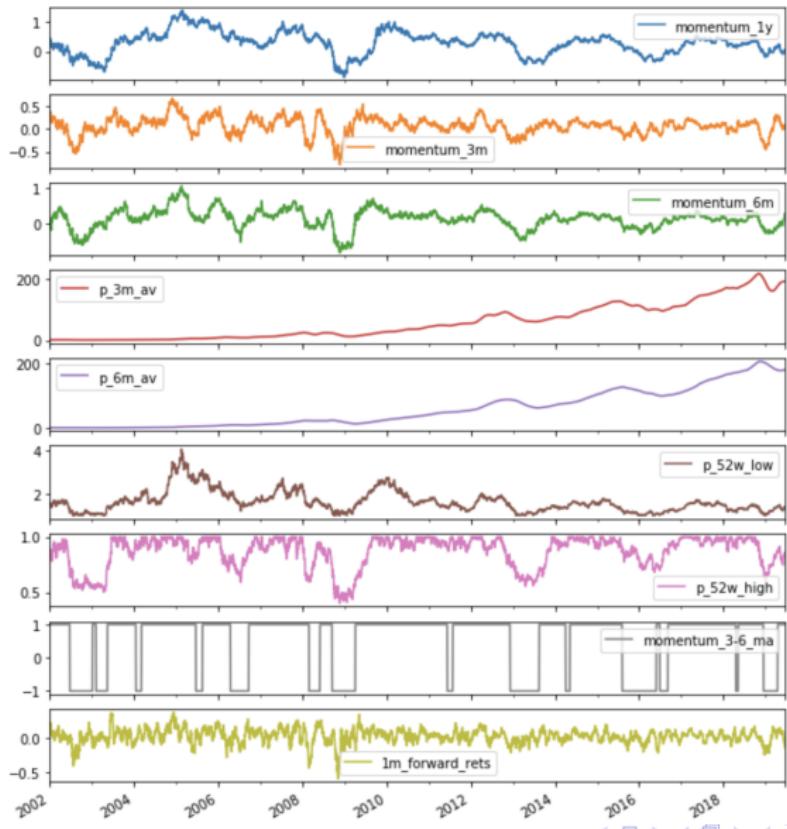
Сделать это можно с помощью пакета визуализации Seaborn, о котором я уже упоминал вам ранее.

После этого, есть смысл визуализировать каждый фактор, как соответствующий временной ряд, стандартными средствами Pandas, чтобы просмотреть итоговый результат.

# Feature engineering



# Feature engineering



# Содержание

- 1 Процесс решения задачи с помощью методов ML
- 2 Загрузка данных и обзор
- 3 Чистка и подготовка данных
- 4 Feature engineering
- 5 Работа с высококоррелированными предикторами

# Работа с высококоррелированными предикторами

Финальный этап, после того, как мы почистили, рассчитали и трансформировали факторы, - это удаление высококоррелированных факторов.

Есть несколько причин того, почему сильно коррелированные предикторы это плохо:

- такие предикторы добавляют больше 'сложности' (complexity) в модель, нежели пользы;
- меньшее количество некоррелированных предикторов лучше с точки зрения затрат (времени, вычислительных и тп);
- в отдельных моделях (линейная регрессия, например) использование высококоррелированных факторов будет приводить к нестабильности, численным ошибкам, и как результат - худшей производительности.

Давайте посмотрим как выглядит стартовая корреляционная матрица нашей выборки.

## Работа с высококоррелированными предикторами

# Работа с высококоррелированными предикторами

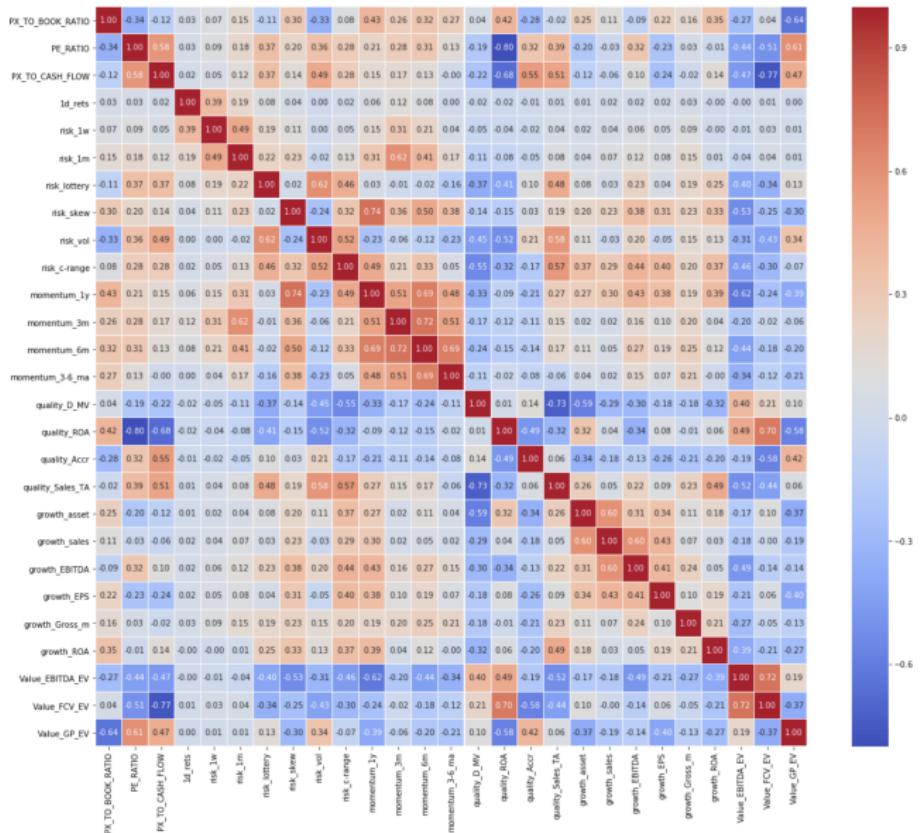
Обычно стандартной схемой работы с высококоррелированными факторами, является 'слепое' удаление всех факторов с корреляцией выше 0.75 по модулю.

Мы с вами будем использовать более продвинутую схему (Kuhn, Johnson, 2013). Вкратце, алгоритм выглядит следующим образом:

- ① рассчитывается корреляционная матрица предикторов;
- ② выбираются два предиктора А и В с наибольшей взаимной корреляцией (по модулю);
- ③ рассчитывается средняя корреляция между А и другими факторами. Тоже вычисляется для В;
- ④ Отбрасывается фактор с наибольшей средней корреляцией;
- ⑤ После этого шаги 1-4 повторяются до тех пор, пока в матрице не останется корреляций выше 0.75.

После подобной 'зачистки' у нас с вами останется 27 факторов, с которыми мы и будем работать.

# Работа с высококоррелированными предикторами



# Литература

Дополнительная литература к сегодняшней лекции:

- Albon, C. (2018) Machine learning with Python cookbook (O'Reilly).
- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An introduction to statistical learning. Springer. (есть на русском)
- Kuhn, M., Johnson, K., (2016). Applied predictive modeling. Springer. (есть на русском)
- Zheng, A. (2018). Feature engineering for Machine learning (O'Reilly).