Реестр тестов и метрик

Core package (базовый функционал). Анализ данных

Точечные оценки

cd_1_1_Mean

- **Техническое название**: cd_1_1_Mean
- **Описание**: Mean. Среднее значение для выбранного столбца
- **Теги**: core, data, scalar
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Отношение суммы всех значений в столбце к числу значений. ** Только для столбцов с числовыми данными. * Если в столбце есть пропуски, они исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_column: название столбца для расчета (column)

higher_is_better: Признак для настройки светофора: большее значение метрики - лучше (True) / хуже (False) (bool)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо, т.к. результат - скалярное выражение (число).

Output (long): Отсутствует

Output (short): Число:среднее значение

Output example (picture): не применимо, т.к. результат - скалярное выражение (число).

cd_1_2_Median

- **Техническое название**: cd_1_2_Median
- **Описание**: Median. Медиана для выбранного столбца
- **Теги**: core, data, scalar
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Число, которое делит упорядоченный набор значений выбранного столбца на две равные части. * *Только для столбцов с числовыми данными.*

* *Если в столбце есть пропуски, они исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_column: название столбца для расчета (column)

higher_is_better: Признак для настройки светофора: большее значение метрики - лучше (True) / хуже (False) (bool)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо.

Output (long): Отсутствует

Output (short): Число: значение медианы

Output example (picture): не применимо.

cd_1_3_Mode

- **Техническое название**: cd_1_3_Mode
- **Описание**: Mode. Мода и N популярных значений столбца
- **Теги**: core, data
- **requirements**: `typing`, `pandas`, , `plotly.graph_objects`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Наиболее часто встречающееся значение в выбранном столбце. * *Для столбцов с произвольным типом данных*. * *Если в столбце есть пропуски, они исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_column: название столбца для расчета (column)

top_col_number: Число отображаемых популярных значений столбца

РЕЗУЛЬТАТЫ

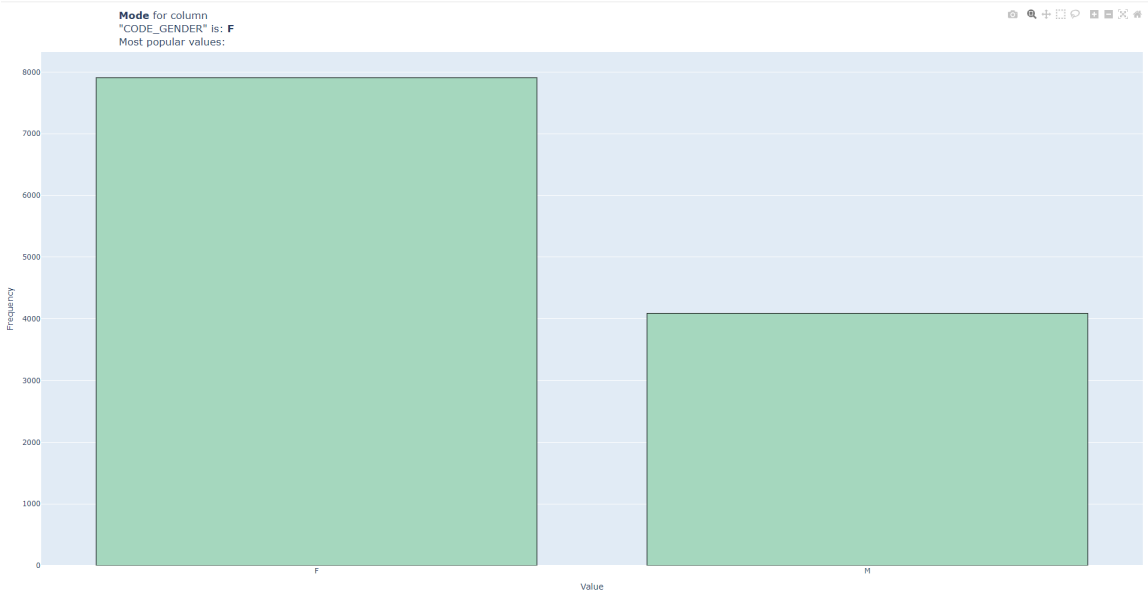
Движок отрисовки графика: plotly.js

Output (long): Barchart

- *xaxis:* значения (N наиболее часто встречающихся значений столбца)
- *yaxis:* частота встречаемости значения

Output (short): Отсутствует

Output example (picture):



cd_1_4_Min

- **Техническое название**: cd_1_4_Min
- **Описание**: Min. Минимальное значение в выбранном столбце
- **Теги**: core, data, scalar
- **requirements**: `typing`, `pandas`

- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Минимальное значение в выбранном столбце.

* *Только для столбцов с числовыми данными.*

* *Если в столбце есть пропуски, они исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_column: название столбца для расчета (column)

higher_is_better: Признак для настройки светофора: большее значение метрики - лучше (True) / хуже (False) (bool)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):Число: минимальное значение в столбце

Output example (picture): не применимо.

> ***cd_1_5_Max***

- **Техническое название**: cd_1_5_Max
- **Описание**: Max. Максимальное значение в выбранном столбце
- **Теги**: core, data, scalar
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Максимальное значение в выбранном столбце.

* *Только для столбцов с числовыми данными.*

* *Если в столбце есть пропуски, они исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_column: название столбца для расчета (column)

higher_is_better: Признак для настройки светофора: большее значение метрики - лучше (True) / хуже (False) (bool)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):Число: максимальное значение в столбце

Output example (picture): не применимо.

> ***cd_1_6_Weighted_Prob***

- **Техническое название**: cd_1_6_Weighted_Prob
- **Описание**: Weighted Probability. Взвешенная вероятность по предсказаниям модели
- **Теги**: core, data, scalar
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Для каждой записи (строки) вычисляем \(score*weight\), суммируем полученные числа.

* *На выходе одно число - значение WP.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

predict_column: название столбца со скором модели (column)

weight_column: название столбца с весами (column)

higher_is_better: Признак для настройки светофора: большее значение метрики - лучше (True) / хуже (False) (bool)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short): Число: значение WP

Output example (picture): не применимо.

cd_1_7_Null_count

- **Техническое название**: cd_1_7_Null_count
- **Описание**: Null count. Количество Null значений в выбранном столбце
- **Теги**: core, data, scalar
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Количество Null значений в выбранном столбце.

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_column: название столбца для расчета (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short): Число Null-значений в выбранном столбце

Output example (picture): не применимо.

Базовые оценки по датасету

cd_2_1_Df_Stats

- **Техническое название**: cd_2_1_Df_Stats

- **Описание**: Df Statistics Table. Базовые статистики по датафрейму
- **Теги**: core, data
- **requirements**: `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Выводятся:

- число строк и число столбцов df
- число пропусков в каждом из столбцов
- для каждого из столбцов с числовыми данными: число записей, среднее значение, стандартное отклонение, min, max, перцентили: 25, 50 и 75.

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Таблица с базовыми статистиками по столбцам

Output (short):

- Число строк в датафрейме
- Число столбцов в датафрейме

Output example (picture):

col_name	count	mean	std	min	25%	50%	75%	max
client_id	11628	5250.7752	3158.7814	0	2441.75	5348.5	8255.25	9999
y_pred	11628	0.4739	0.4993	0	0	0	1	1
y_fact	11628	0.4878	0.4999	0	0	0	1	1
total_sum	11628	448.4191	501.8431	2.1204	108.2134	220.0095	640.0333	3537.6186
total_sum_	11628	437.2705	566.3495	0.916	92.4079	202.5112	562.1943	5989.9508
total_cnt	11628	0.0187	0.0069	0.0004	0.0143	0.0183	0.0228	0.0576

col_name	null_count
client_id	0
y_pred	0
y_fact	0
total_sum	0
total_sum_pred	0
total_cnt	0
report_dttm	0
date	0

Число строк:
11.63k

Число столбцов:
8

cd_2_2_Df_Stats_features

- **Техническое название**: cd_2_2_Df_Stats_features

- **Описание**: Statistics Table for Selected Columns. Статистики для выбранных столбцов
- **Теги**: core, data
- **requirements**: `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Выводятся: count, mean, std, min, max, перцентили: 25, 50 и 75 для выбранных столбцов

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

field_columns: Список названий столбцов для исследования через запятую (multi-column)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Таблица с базовыми статистиками по столбцам

Output (short): Отсутствует

Output example (picture):

col_name	count	mean	std	min	25%	50%	75%	max
EVENT_PROBABILITY	11999	0.0794	0.0689	0.0072	0.0353	0.0582	0.0983	0.7166
TARGET	11999	0.0769	0.2665	0	0	0	0	1

cd_2_3_Density_Distr

- **Техническое название**: cd_2_3_Density_Distr
- **Описание**: Density Distribution. Плотность распределения для всех полей датасета
- **Теги**: core, data
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Цикл по всем столбцам df:

- если данные в столбце ***не числовые***, пропускаем его;
- если данные ***категориальные*** (меньше или равно *categorial_threshold* различных значений), строится гистограмма;

- если данные ***непрерывные***, строится линейный график плотности распределения.

Полученные гистограммы и графики выводятся в общее поле или на отдельные поля, в зависимости от значения *split_charts*. При выводе на общее поле нажатием на легенду можно ***активировать нужный элемент***. Масштаб автоматически подстраивается под значения метрики.

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

categorial_threshold: граница числа уникальных объектов в категориальной колонке датасета (int)

split_charts: флаг разделения графиков для разных колонок по разным карточкам (bool)

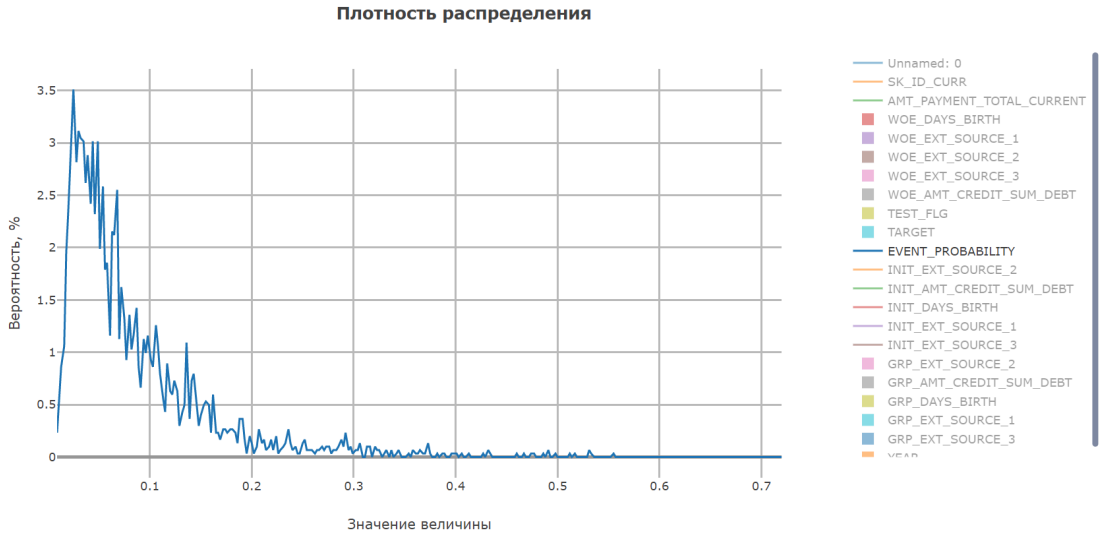
РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

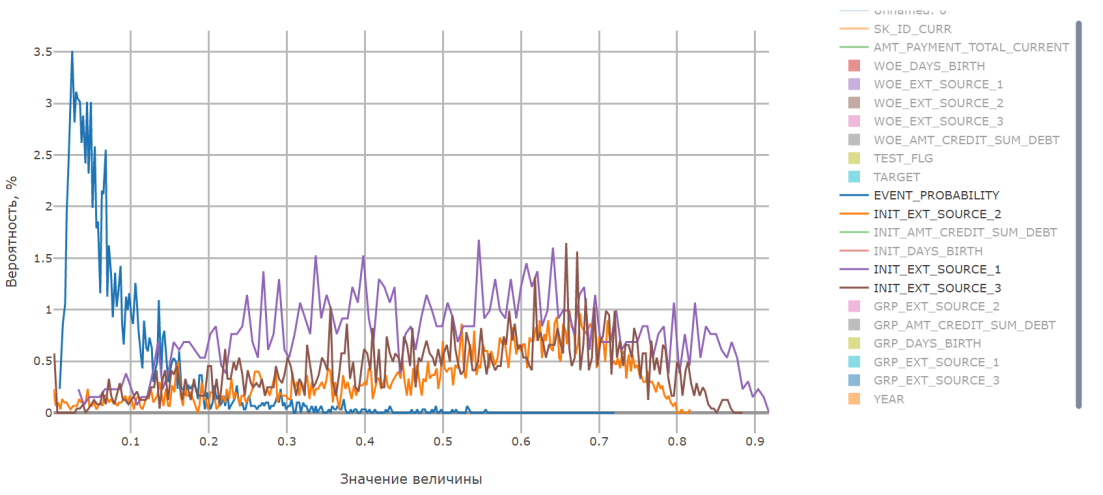
Output (long): Массив графиков

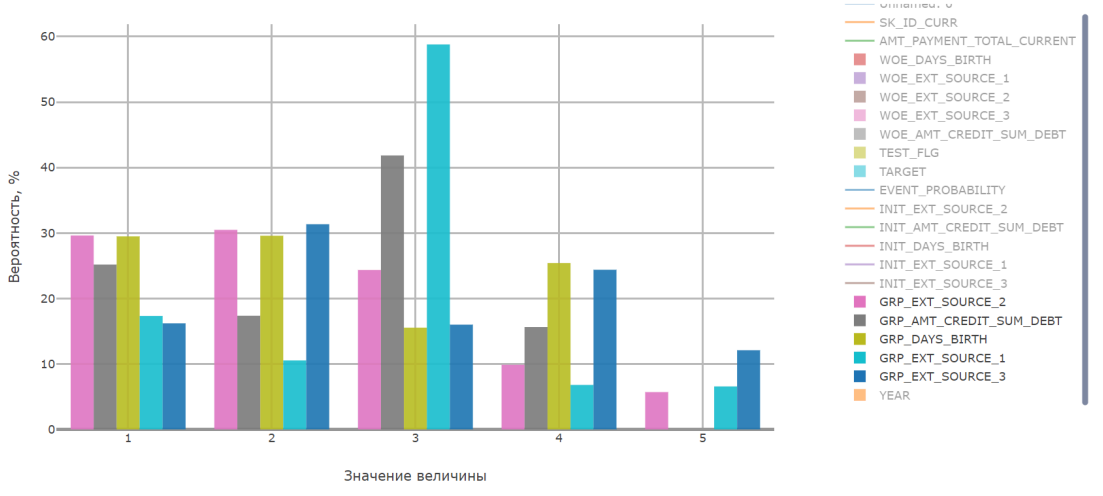
Output (short): Отсутствует

Output example (picture):




Плотность распределения
Вероятность, %
Значение величины
Unnamed: 0
SK_ID_CURR
AMT_PAYMENT_TOTAL_CURRENT
WOE_DAYS_BIRTH
WOE_EXT_SOURCE_1
WOE_EXT_SOURCE_2
WOE_EXT_SOURCE_3
WOE_AMT_CREDIT_SUM_DEBT
TEST_FLG
TARGET
EVENT_PROBABILITY
INIT_EXT_SOURCE_2
INIT_AMT_CREDIT_SUM_DEBT
INIT_DAYS_BIRTH
INIT_EXT_SOURCE_1
INIT_EXT_SOURCE_3
GRP_EXT_SOURCE_2
GRP_AMT_CREDIT_SUM_DEBT
GRP_DAYS_BIRTH
GRP_EXT_SOURCE_1
GRP_EXT_SOURCE_3


Плотность распределения
Вероятность, %
Значение величины
SK_ID_CURR
AMT_PAYMENT_TOTAL_CURRENT
WOE_DAYS_BIRTH
WOE_EXT_SOURCE_1
WOE_EXT_SOURCE_2
WOE_EXT_SOURCE_3
WOE_AMT_CREDIT_SUM_DEBT
TEST_FLG
TARGET
EVENT_PROBABILITY
INIT_EXT_SOURCE_2
INIT_AMT_CREDIT_SUM_DEBT
INIT_DAYS_BIRTH
INIT_EXT_SOURCE_1
INIT_EXT_SOURCE_3
GRP_EXT_SOURCE_2
GRP_AMT_CREDIT_SUM_DEBT
GRP_DAYS_BIRTH
GRP_EXT_SOURCE_1
GRP_EXT_SOURCE_3
YEAR


Плотность распределения
Вероятность, %
Значение величины
SK_ID_CURR
AMT_PAYMENT_TOTAL_CURRENT
WOE_DAYS_BIRTH
WOE_EXT_SOURCE_1
WOE_EXT_SOURCE_2
WOE_EXT_SOURCE_3
WOE_AMT_CREDIT_SUM_DEBT
TEST_FLG
TARGET
EVENT_PROBABILITY
INIT_EXT_SOURCE_2
INIT_AMT_CREDIT_SUM_DEBT
INIT_DAYS_BIRTH
INIT_EXT_SOURCE_1
INIT_EXT_SOURCE_3
GRP_EXT_SOURCE_2
GRP_AMT_CREDIT_SUM_DEBT
GRP_DAYS_BIRTH
GRP_EXT_SOURCE_1
GRP_EXT_SOURCE_3
YEAR

> ***cd_2_4_Density_Distr_features***

- **Техническое название**: cd_2_4_Density_Distr_features
- **Описание**: Density Distribution for Selected Columns. Плотность распределения для выбранных полей
- **Теги**: core, data
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Цикл по выбранным столбцам df:

- если данные в столбце ***не числовые***, пропускаем его;
- если данные ***категориальные*** (меньше или равно *categorial_threshold* различных значений), строится гистограмма;
- если данные ***непрерывные***, строится линейный график плотности распределения.

Полученные гистограммы и графики выводятся в общее поле или на отдельные поля, в зависимости от значения *split_charts*.

При выводе на общее поле нажатием на легенду можно ***активировать нужный элемент***. Масштаб автоматически подстраивается под значения метрики.

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

categorial_threshold: граница числа уникальных объектов в категориальной колонке датасета (int)

field_columns: Список названий столбцов для исследования (multi-column)

split_charts: флаг разделения графиков для разных колонок по разным карточкам (bool)

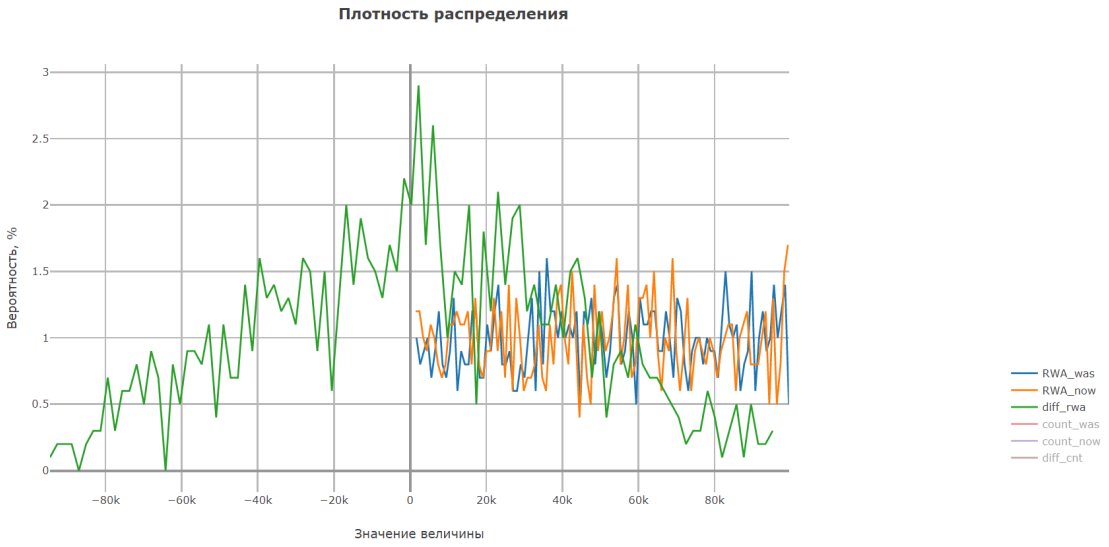
РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Массив графиков

Output (short): Отсутствует

Output example (picture):



cd_2_5_Pivot_table_filtered

- **Техническое название**: cd_2_5_Pivot_table_filtered
- **Описание**: Filtered Pivot Table. Сводная таблица, значения фильтруются кнопками
- **Теги**: core, data
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Сводная таблица.

Кнопки фильтруют входной набор данных для расчета по значениям filter_column.

Для каждого столбца из ***value_columns*** применяются все функции из ***agg_funcs***.

ВНИМАНИЕ: применение математических агрегацией вызовет ошибку, если в ***value_columns*** есть хоть один столбец с нечисловыми данными.

ВХОДНЫЕ ПАРАМЕТРЫ

df: Датасет для исследования (dataframe)

filter_column: Столбец, по значениям которого фильтруются данные (column)

index_column: Столбец, значения которого образуют индекс pivot-таблицы (column)

value_columns: Список столбцов для вычисления показателей (sum) по группам (multi-column)

agg_funcs: Список агрегирующих функций, используемых над *value_columns*. Default= ['sum'].

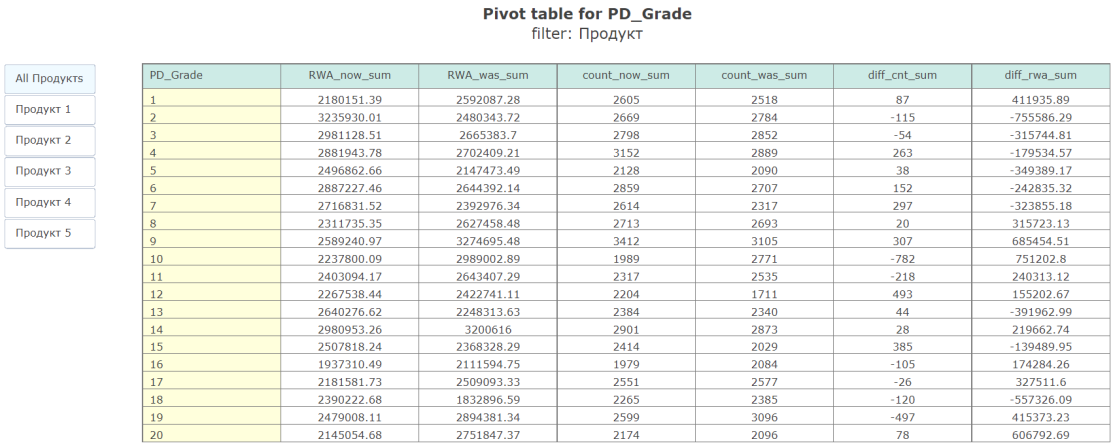
РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Сводная таблица

Output (short): Отсутствует

Output example (picture):



Pivot table for PD_Grade
filter: Продукт

All Продукты
Продукт 1
Продукт 2
Продукт 3
Продукт 4
Продукт 5

PD_Grade	RWA_now_sum	RWA_was_sum	count_now_sum	count_was_sum	diff_cnt_sum	diff_rwa_sum
1	2180151.39	2592087.28	2605	2518	87	411935.89
2	3235930.01	2480343.72	2669	2784	-115	-755586.29
3	2981128.51	2665383.7	2798	2852	-54	-315744.81
4	2881943.78	2702409.21	3152	2889	263	-179534.57
5	2496862.66	2147473.49	2128	2090	38	-349389.17
6	2887227.46	2644392.14	2859	2707	152	-242835.32
7	2716831.52	2392976.34	2614	2317	297	-323855.18
8	2311735.35	2627458.48	2713	2693	20	315723.13
9	2589240.97	3274695.48	3412	3105	307	685454.51
10	2237800.09	2989002.89	1989	2771	-782	751202.8
11	2403094.17	2643407.29	2317	2535	-218	240313.12
12	2267538.44	2422741.11	2204	1711	493	155202.67
13	2640276.62	2248313.63	2384	2340	44	-391962.99
14	2980953.26	3200616	2901	2873	28	219662.74
15	2507818.24	2368328.29	2414	2029	385	-139489.95
16	1937310.49	2111594.75	1979	2084	-105	174284.26
17	2181581.73	2509093.33	2551	2577	-26	327511.6
18	2390222.68	1832896.59	2265	2385	-120	-557326.09
19	2479008.11	2894381.34	2599	3096	-497	415373.23
20	2145054.68	2751847.37	2174	2096	78	606792.69

cd_2_6_Barchart_filtered

- **Техническое название**: cd_2_6_Barchart_filtered
- **Описание**: Filtered Barchart. Столбчатая диаграмма, значения фильтруются кнопками
- **Теги**: core, data
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Кнопки фильтруют входной набор данных для расчета по значениям filter_column.

Для столбца ***value_column*** применяется функция ***agg_funcs***.

ВНИМАНИЕ: применение математических агрегацией вызовет ошибку, если ***value_column*** является столбцом с нечисловыми данными

ВХОДНЫЕ ПАРАМЕТРЫ

df: Датасет для исследования (dataframe)

filter_column: Столбец, по значениям которого фильтруются данные (column)

index_column: Столбец для группировки (column)

value_column: Столбец для вычисления показателей (sum) по группам (column)

agg_func: Список агрегирующих функций, используемых над *value_columns*. Default= ['sum'].

РЕЗУЛЬТАТЫ

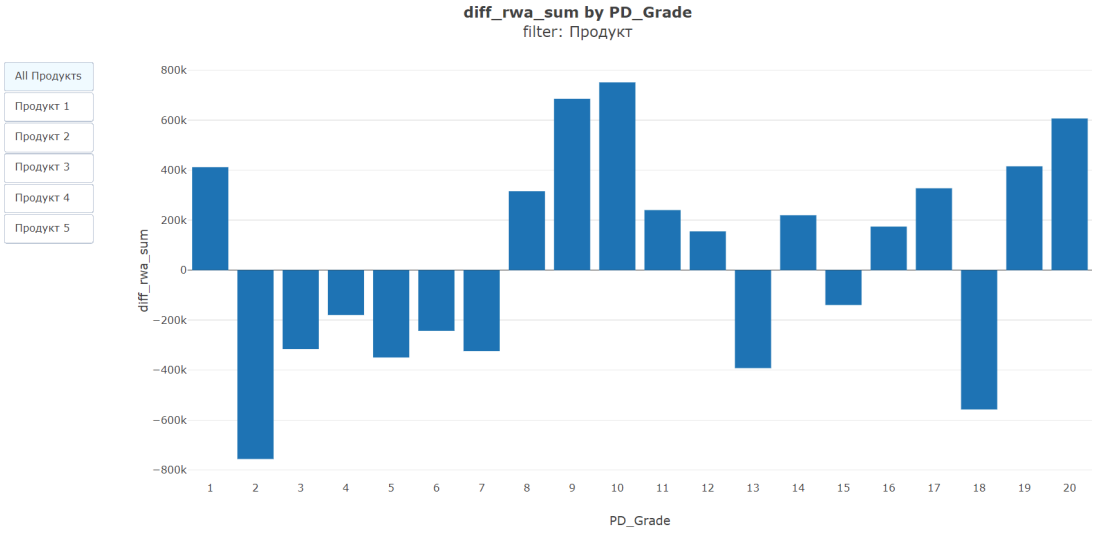
Движок отрисовки графика: plotly.js

Output (long): Barchart

- *xaxis*: значения index_column
- *yaxis*: суммы значений value_column по группам

Output (short): Отсутствует

Output example (picture):



cd_2_7_Fill_Pct

- **Техническое название**: cd_2_7_Fill_Pct
- **Описание**: Fill Percent for Features. Barchart с процентом не-null значений в выбранных столбцах
- **Теги**: core, data
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Для каждого из выбранных столбцов датафрейма вычисляется отношение числа заполненных (не Null) полей к общему числу полей

ВХОДНЫЕ ПАРАМЕТРЫ

df : Объект данных для расчета (dataframe)

field_columns: Фичи для расчета (multi-column)

РЕЗУЛЬТАТЫ

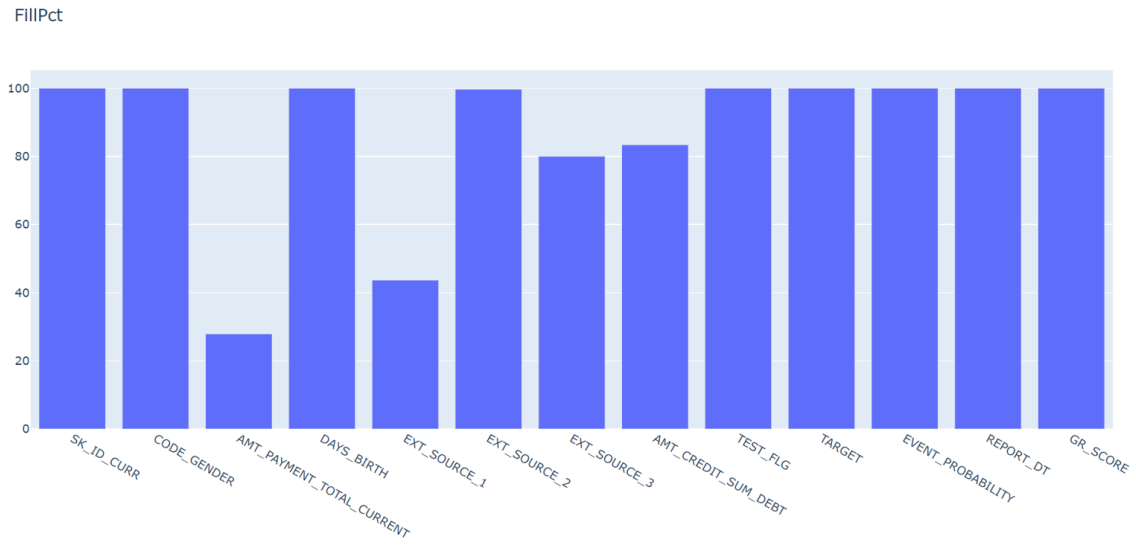
Движок отрисовки графика: plotly.js

Output (long): Barchart

- *xaxis*: названия столбцов, для которых производился расчет
- *yaxis*: процент заполненных значений

Output (short): Отсутствует

Output example (picture):



Анализ распределений

cd_3_1_Histogram

- **Техническое название**: cd_3_1_Histogram
- **Описание**: Histogram. Гистограмма для выбранного столбца
- **Теги**: core, data
- **requirements**: `typing` , `pandas`

- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Дает картину распределения значений в выбранном столбце

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

field_column: Столбец для расчета (column)

nbins: Максимальное число столбцов в гистограмме (int value; по умолчанию 30)

РЕЗУЛЬТАТЫ

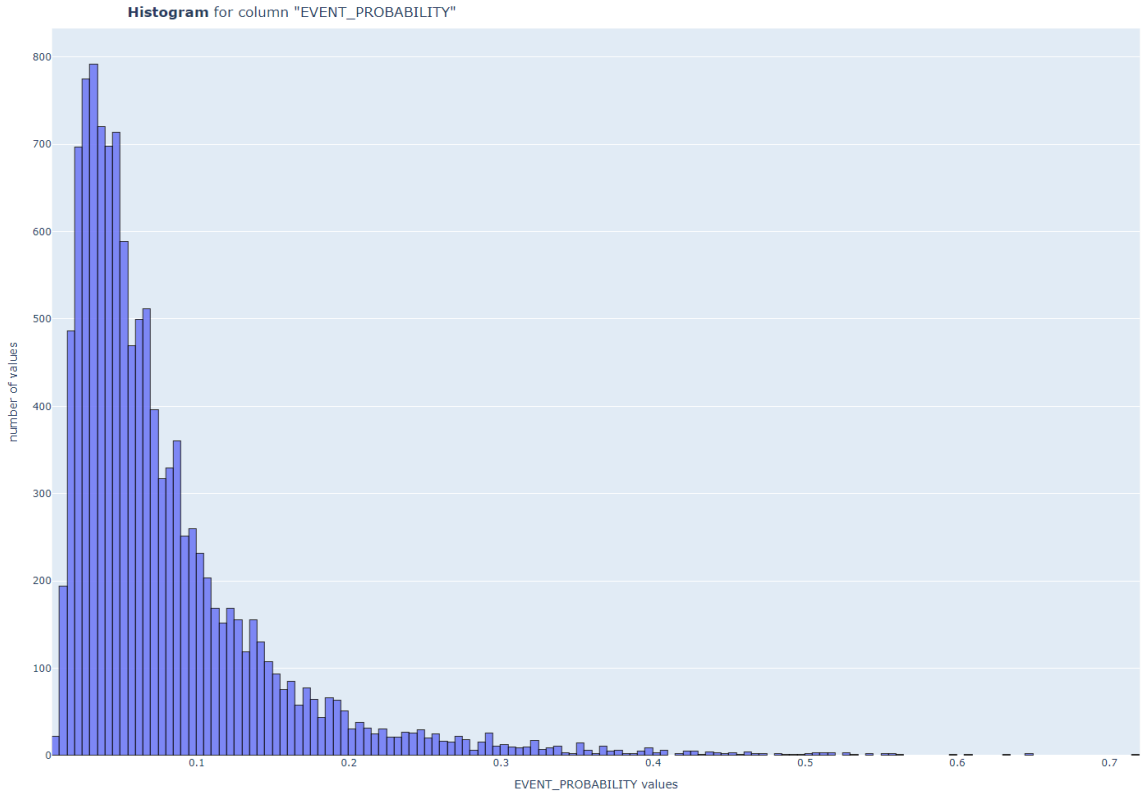
Движок отрисовки графика: plotly.js

Output (long): Гистограмма:

- *xaxis*: значения из рассматриваемого столбца
- *yaxis*: частоты встречаемости значений

Output (short): Отсутствует

Output example (picture):



cd_3_2_Target_Variables_Rates

- **Техническое название**: cd_3_2_Target_Variables_Rates
- **Описание**: Target Variables Rates. Доля целевой переменной по сегментам
- **Теги**: core, data
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Данные разбиваются на группы по значениям ***cat_field_column***.

* *Число групп = числу уникальных значений cat_field_column*

Для каждой группы определяется:

- доля наблюдений с ***target_column***==1 среди всех наблюдений данной группы
- средний ***predict_column*** по данной группе

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: Целевая переменная модели (column)

predict_column: Скор модели (column)

cat_field_column: Столбец с категориальной переменной, по значениям которой проводим группировку (column)

РЕЗУЛЬТАТЫ

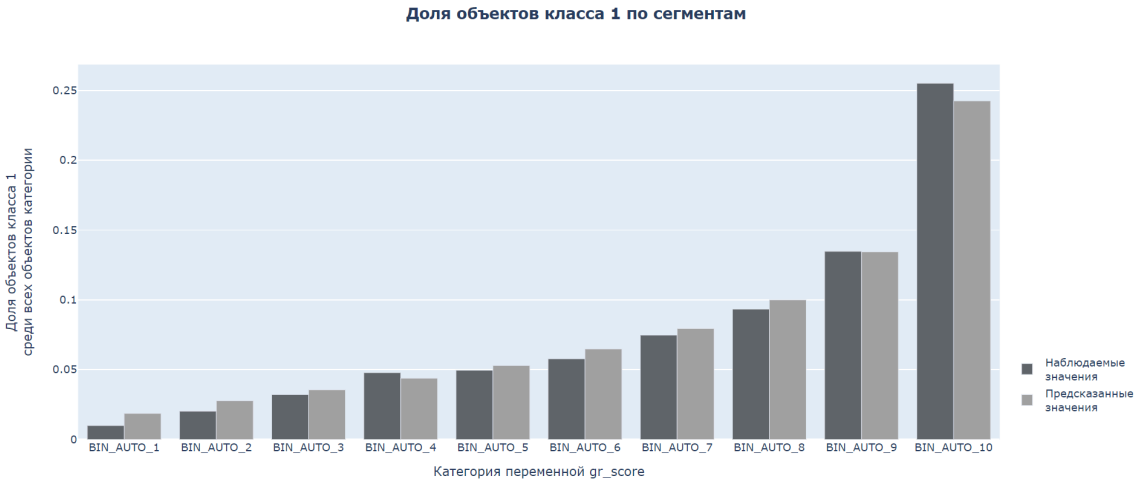
Движок отрисовки графика: plotly.js

Output (long): Barchart

- *xaxis*: группы (значения cat-field)
- *yaxis*: средние значения target и score по группам

Output (short): Отсутствует

Output example (picture):



cd_3_3_Shapiro_Wilk_Test

- **Техническое название**: cd_3_3_Shapiro_Wilk_Test
- **Описание**: Shapiro-Wilk Test. Тест Шапиро-Уилка (нормальность распределения данных)
- **Теги**: core, data, scalar
- **requirements**: `typing`, `pandas`, `scipy.stats`
- **Примечания**: nan

ЛОГИКА ИСПОЛНЕНИЯ

Тест на нормальность распределения данных.

\(H_0\) : Данная выборка - из нормального распределения.

Вычисляем \(P\text{-}value\) с помощью *scipy.stats.shapiro*, по нему настраиваем светофор.

* *Если в столбце есть пропущенные значения, они исключаются из рассмотрения.*

Пример выставления *signal bounds* (светофора):

\(P\text{-}value ≤ 1(\%)\) - красный,

\(1(\%) < P\text{-}value ≤ 5(\%)\) - желтый,

\(P\text{-}value > 5(\%)\) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

field_column: Столбец для расчета (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short): Число: значение \(P\text{-}value\)

Output example (picture): не применимо.

cd_3_4_Percentiles

- **Техническое название**: cd_3_4_Percentiles
- **Описание**: Percentiles. Линейный график в осях номер-значение перцентиля для выбранного столбца
- **Теги**: core, data
- **requirements**: `typing`, `pandas`, `numpy`, `plotly.graph_objects`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

**Если в столбце есть пропуски, они исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_column: название столбца для расчета (column)

РЕЗУЛЬТАТЫ

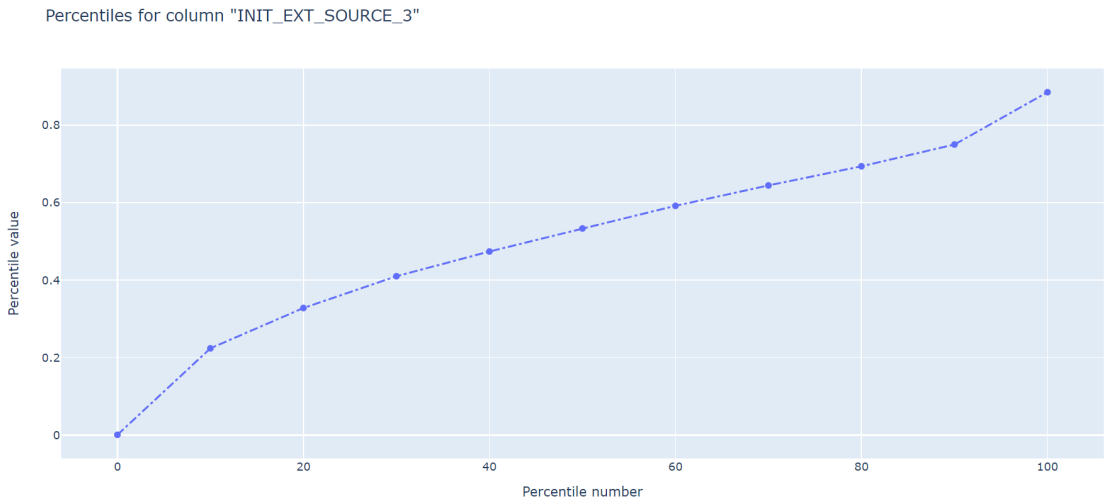
Движок отрисовки графика: plotly.js

Output (long): Линейный график

- *xaxis*: номер перцентиля (10, 20, ... 90)
- *yaxis*: значение перцентиля

Output (short): Отсутствует

Output example (picture):



Поиск зависимостей в данных, отбор признаков

cd_4_1_Pearson_Correlations

- **Техническое название**: cd_4_1_Pearson_Correlations
- **Описание**: Pearson Correlations. Корреляции Пирсона (в %)
- **Теги**: core, data, scalar
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

* *Только для линейных моделей.*

Коэффициент корреляции Пирсона характеризует существование линейной связи между двумя признаками (столбцами *df*).

Пропущенные поля исключаем из рассмотрения.

Пример выставления *signal bounds*:

\(max(Corr) > 75\) - красный,

\(60 < max(Corr) ≤ 75\) - желтый,

\(max(Corr) ≤ 60\) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

field_columns: Фичи для расчета (multi-column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

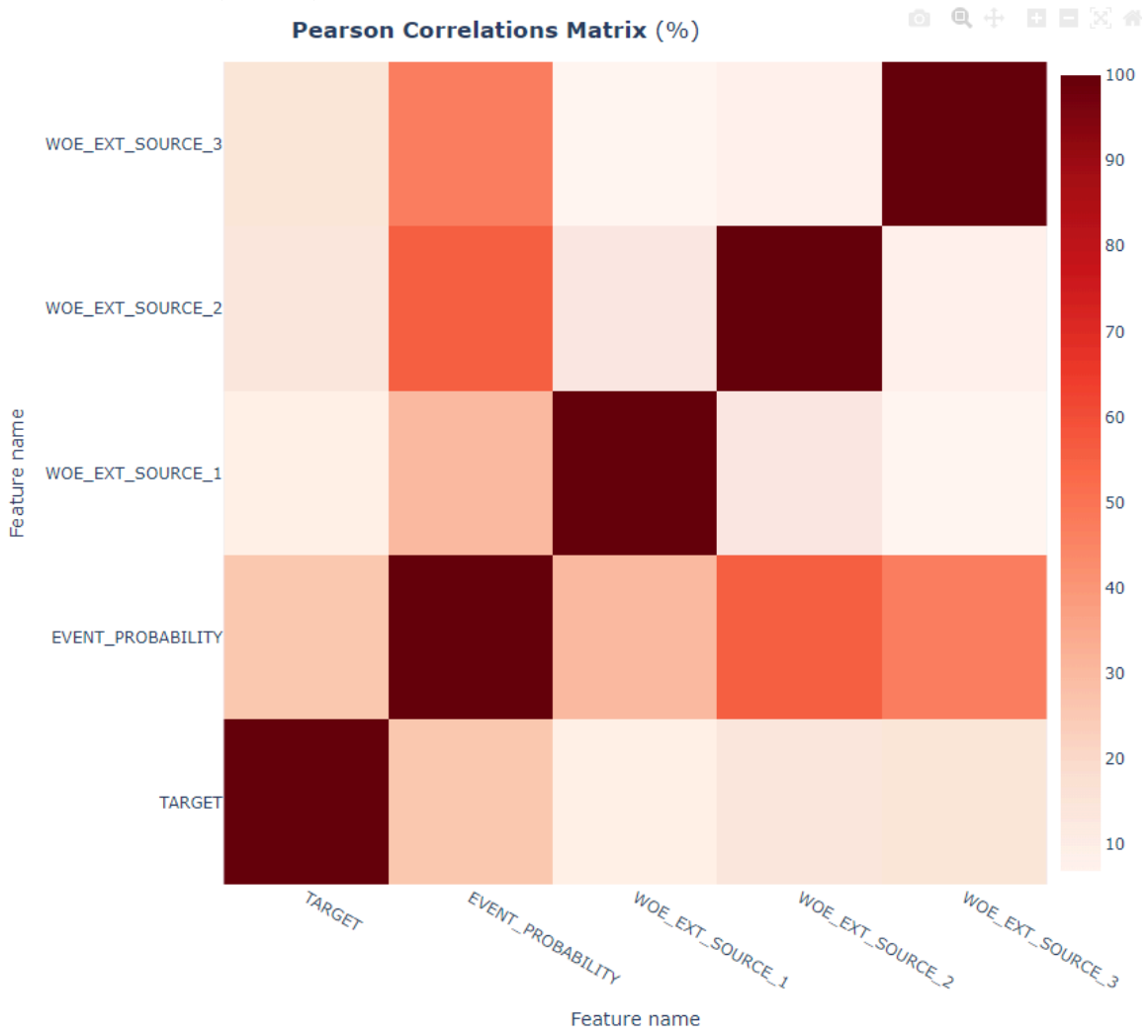
Output (long): Heatmap

Матрица попарных корреляций выбранных столбцов

Output (short):

- Число - максимальное из значений парной корреляции для выбранных столбцов (в %)
- Светофор

Output example (picture):



> ***cd_4_2_Gini_features***

- **Техническое название**: cd_4_2_Gini_features
- **Описание**: Gini Index (%) for Features. Индекс Джини (%) в разрезе отдельных факторов
- **Теги**: core, data, risk
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

В цикле для каждого *field* из *field_columns*:

1. Расчет \(ROC AUC\) по *target_column*, *field*
2. \([Gini index] = 2 * [ROC AUC] - 1\)

** Если в признаке (field) или target_column присутствует пропущенное значение, такая строка исключается из рассмотрения. Для разных признаков исключаются из рассмотрения разные строки.*

Индекс Джини для фактора служит оценкой способности фактора разделять наблюдения разных классов.

Чем больше Джини, тем сильнее по заданному фактору разделяются классы *target_column*.

Тест используется при отборе факторов для модели (целесообразно выбрать факторы с наибольшими Джини).

При валидации: проверка того, что факторы, используемые в модели, действительно информативные.

Пример выставления *signal bounds*:

\(Gini ≤ 5(\%)\) - красный,

\(5(\%) < Gini ≤ 15(\%)\) - желтый,

\(Gini > 15(\%)\) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

field_columns: массив названий столбцов, по которым считаем тест (multi-column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

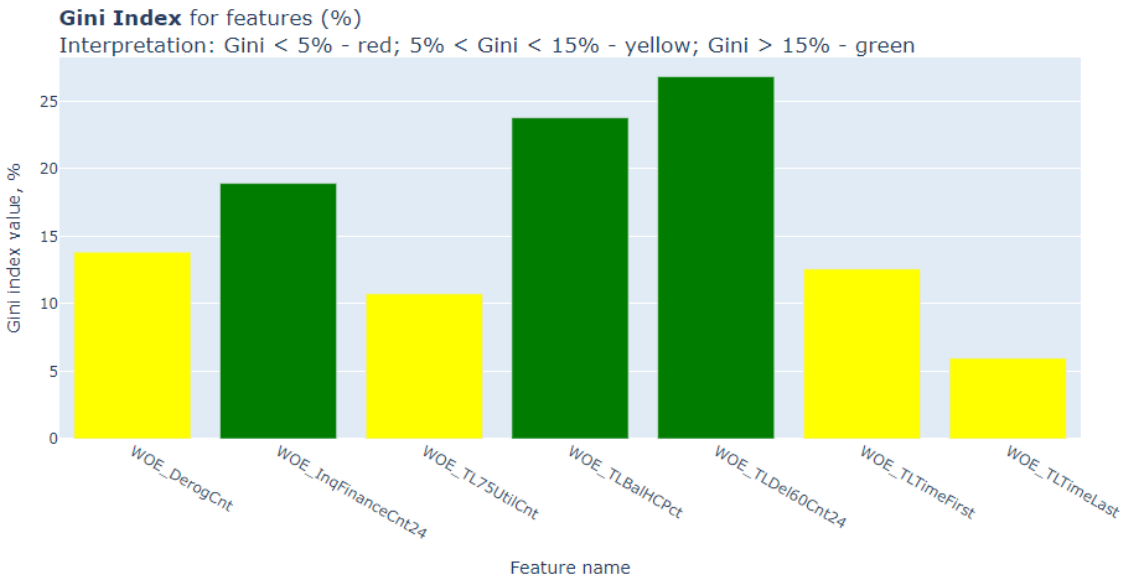
Barchart:

- *xaxis*: названия столбцов, для которых производился расчет
- *yaxis*: значения коэффициента Джини (в %)

Светофор: столбцы окрашиваются в цвет светофора для соответствующего признака

Output (short): Отсутствует

Output example (picture):



cd_4_3_Chi_Square_features

- **Техническое название**: cd_4_3_Chi_Square_features
- **Описание**: Chi Square Test for features Pct. Хи квадрат (проверка схожести 2-х категориальных распределений) в процентах
- **Теги**: core, data, scalar
- **requirements**: `typing`, `pandas`, `scipy.stats`
- **Примечания**:

Применение данной реализации критерия согласия хи-квадрат:

1) сравнение 2-х категориальных признаков

2) сравнение распределений target и категориального predict по классам (например, при калибровке модели)

ЛОГИКА ИСПОЛНЕНИЯ

В общем случае: тест хи-квадрат проверяет нулевую гипотезу о том, что категориальные данные имеют заданное распределение по группам.

В реализации: \(H_0\): Распределения первой и второй категориальных переменных одинаковы

Выполнение \(H_0\) - желаемый исход => чем больше \(P\text{-}value\), тем лучше

Пример выставления светофора: \(P\text{-}value ≤ 1(\%)\) - красный,

\(1(\%) < P\text{-}value ≤ 5(\%)\)- желтый,

\(P\text{-}value > 5(\%)\) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

cat_feature_1_column: Первая категориальная переменная (column)

cat_feature_2_column: Вторая категориальная переменная (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short): \(P\text{-}value\) критерия хи-квадрат,

светофор

Output example (picture): не применимо.

> ***cd_4_4_VIF (r_3_2_VIF)***

- **Техническое название**: cd_4_4_VIF (r_3_2_VIF)
- **Описание**: Variance Inflation Factor. Коэффициент инфляции дисперсии. Используется для обнаружения мультиколлинеарности.

- **Теги**: core, data, risk
- **requirements**: `typing`, `pandas`, `statsmodels.stats.outliers_influence`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

В цикле для каждого категориального *field* из *fields_to_test* считаем statsmodels.stats.outliers_influence.variance_inflation_factor

\(VIF\) позволяет оценить степень мультиколлинеарности в рамках анализа взаимозависимости факторов модели методом наименьших квадратов. Индекс показывает насколько велика зависимость фактора от других факторов модели.

(В реализации - зависимость выбранного фактора от других факторов из *fields_to_test*).

\(VIF\) принимает значения от 1 до +∞. Чем больше значение, тем сильнее свидетельство присутствия мультиколлинеарности.
Значение, равное 1, свидетельствует об ортогональности (независимости) независимой переменной остальным переменным модели.

* *Используется для линейных моделей*

Пример выставления *signal bounds*:

\(VIF ≥ 5\) - красный,

\(3 ≤ VIF < 5\) - желтый,

\(VIF < 3\) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_columns: массив названий столбцов, по которым считаем тест (multi-column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

Barchart:

- *xaxis*: названия столбцов, для которых производился расчет

- *yaxis*: значения \(VIF\)

Светофор: столбцы окрашиваются в цвет светофора для соответствующего признака

Output (short): Отсутствует

Output example (picture):

Core package (базовый функционал). Метрики качества регрессии

rvc_1_MAE

- **Техническое название**: rvc_1_MAE
- **Описание**: Mean Absolute Error (MAE). Средняя абсолютная ошибка
- **Теги**: core, regression, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: Применять только на датасетах с небинарным таргетом

ЛОГИКА ИСПОЛНЕНИЯ

Средняя абсолютная ошибка

(Средняя абсолютная разность между предсказаниями модели и целевыми значениями)

** Строки с пропусками в target_column или predict_column исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: НЕБИНАРНАЯ
целевая переменная модели (column)

predict_column: Предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Числовое значение средней абсолютной ошибки,

светофор

Output example (picture): не применимо.

> ***rvc_2_RMSE***

- **Техническое название**: rvc_2_RMSE
- **Описание**: Root Mean Square Error (RMSE). Среднеквадратичная ошибка
- **Теги**: core, regression, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Корень из среднеквадратичной ошибки

(Квадратный корень из отношения суммы квадратов отклонений предсказаний модели от истинных значений к количеству наблюдений)

* *Строки с пропусками в target или score исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: Целевая переменная модели (column)

predict_column: Предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Числовое значение среднеквадратичной ошибки,

светофор

Output example (picture): не применимо.

> ***rvc_3_MAPE***

- **Техническое название**: rvc_3_MAPE
- **Описание**: Mean Absolute Percentage Error (MAPE). Средняя абсолютная ошибка в процентах
- **Теги**: core, regression, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: Применять только на датасетах с небинарным таргетом

ЛОГИКА ИСПОЛНЕНИЯ

Среднее отношение абсолютной ошибки предсказания к величине предсказанного значения.

* *Строки с пропусками в target_column или predict_column исключаются из рассмотрения.*

* *Также исключаются из рассмотрения строки с target==0, поэтому применение метрики корректно только на датасетах для регрессии (с небинарным таргетом)*

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: НЕБИНАРНАЯ

целевая переменная модели (column)

predict_column: Предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Числовое значение MAPE,

светофор

Output example (picture): не применимо.

rvc_4_MASE

- **Техническое название**: rvc_4_MASE
- **Описание**: Mean absolute scaled error. Средняя абсолютная масштабированная ошибка
- **Теги**: core, regression, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`, `numpy`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Средняя абсолютная масштабированная ошибка

В прогнозировании временных рядов MASE дает представление об эффективности алгоритма прогнозирования по отношению к наивному прогнозу. Ее значение больше единицы (1) указывает на то, что алгоритм работает плохо по сравнению с наивным прогнозом.

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: Целевая переменная модели (column)

predict_column: Предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо **Output (long)**: Отсутствует **Output (short)**: Число: значение средней абсолютной масштабированной ошибки

Output example (picture): не применимо.

> ***rvc_5_WAPE***

- **Техническое название**: rvc_5_WAPE
- **Описание**: Weighted Absolute Percentage Error (WAPE). Взвешенная абсолютная ошибка в процентах
- **Теги**: core, regression, scalar
- **requirements**: -
- **Примечания**: Применять только на датасетах с небинарным таргетом

ЛОГИКА ИСПОЛНЕНИЯ

\(\frac{\text{(df[target] - df[predict]).abs().sum()}}{\text{df[target].abs().sum()}}\)

** Строки с пропусками в target_column или predict_column исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: НЕБИНАРНАЯ
целевая переменная модели (column)

predict_column: Предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Отсутствует

Output (short):

Числовое значение WAPE

светофор

Output example (picture): не применимо.

> ***rvc_6_Weighted_MAPE***

- **Техническое название**: rvc_6_Weighted_MAPE
- **Описание**: Weighted MAPE (WMAPE). Взвешенная ошибка MAPE
- **Теги**: core, regression, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: Применять только на датасетах с небинарным таргетом

ЛОГИКА ИСПОЛНЕНИЯ

Значения MAPE, взвешенные по столбцу weight_column

** Строки с пропусками в target_column, predict_column или weight_column исключаются из рассмотрения.*

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: НЕБИНАРНАЯ

целевая переменная модели (column)

predict_column: Предсказание модели (column)

weight_column: Столбец используемый в качестве веса (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Отсутствует

Output (short):

Число: значение взвешенной ошибки,

светофор

Output example (picture): не применимо.

> ***rvc_7_Average_Bias***

- **Техническое название**: rvc_7_Average_Bias
- **Описание**: Average Bias. Среднее смещение для Squared Loss
- **Теги**: core, regression, scalar

- **requirements**: `typing`, `pandas`, `numpy`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Опр: Матожидание разности между истинным ответом и ответом, выданным алгоритмом

В реализации:

среднее смещение рассчитывается по формуле:

\(mean(abs(target - mean(score)))\)

Разложение ошибки на смещение (bias) и разброс (variance): bias_variance_decomp

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: Целевая переменная модели (column)

predict_column: Предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Числовое значение среднего смещения,

светофор

Output example (picture): не применимо.

> ***rvc_8_R_Squared_Score***

- **Техническое название**: rvc_8_R_Squared_Score
- **Описание**: R2 score. Коэффициент детерминации
- **Теги**: core, regression, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: Применять только на датасетах с небинарным таргетом

ЛОГИКА ИСПОЛНЕНИЯ

R2 используется для оценки производительности модели машинного обучения на основе регрессии. Суть его работы заключается в измерении количества отклонений в прогнозах, объясненных набором данных (разница между выборками в наборе данных и прогнозами, сделанными моделью).

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: НЕБИНАРНАЯ

целевая переменная модели (column)

predict_column: Предсказание модели (column)

weight_column: Столбец используемый в качестве веса (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Числовое значение коэффициента детерминации,

светофор

Output example (picture): не применимо.

> ***rvc_9_Regression_Performance***

- **Техническое название**: rvc_9_Regression_Performance
- **Описание**: Regression Performance. Распределение ошибок регрессии и диаграммы рассеяния в осях актуальные-предсказанные значения для обучающих и текущих данных
- **Теги**: core, regression
- **requirements**: `typing`, `pandas`
- **Примечания**: Для больших датасетов получается очень объемный json графика (что сказывается на скорости отображения результата в UI). Подумать как выводить не все точки, при этом верно отображая вид зависимостей.

ЛОГИКА ИСПОЛНЕНИЯ

1) Гистограммы распределения ошибки предсказания (для 2-х датафреймов);

2) Scatter plots в координатах Actual value - Predicted value (для 2-х датафреймов).

ВХОДНЫЕ ПАРАМЕТРЫ

df_reference: датасет с данными на базовый период, либо за предыдущий период (dataframe)

df_current: датасет с данными на текущий период (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца с предсказаниями модели (column)

(! названия столбцов с *target_column* и *predict_column* должны совпадать в *df_reference* и *df_current*)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

Распределение ошибок регрессии:

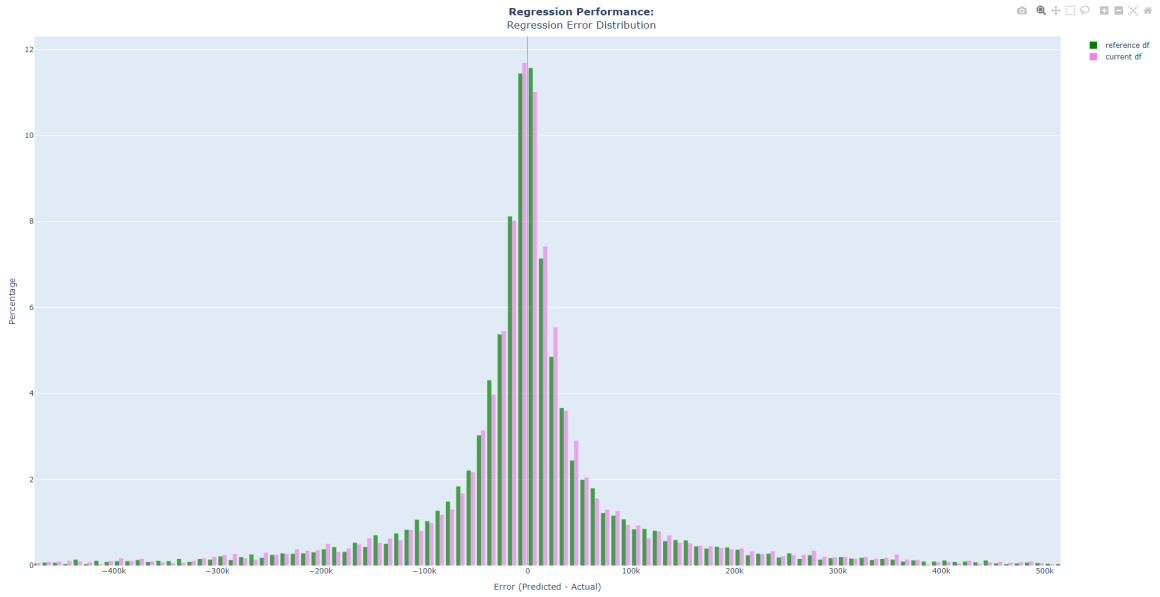
1. Гистограмма ошибок на df_reference
2. Гистограмма ошибок на df_current

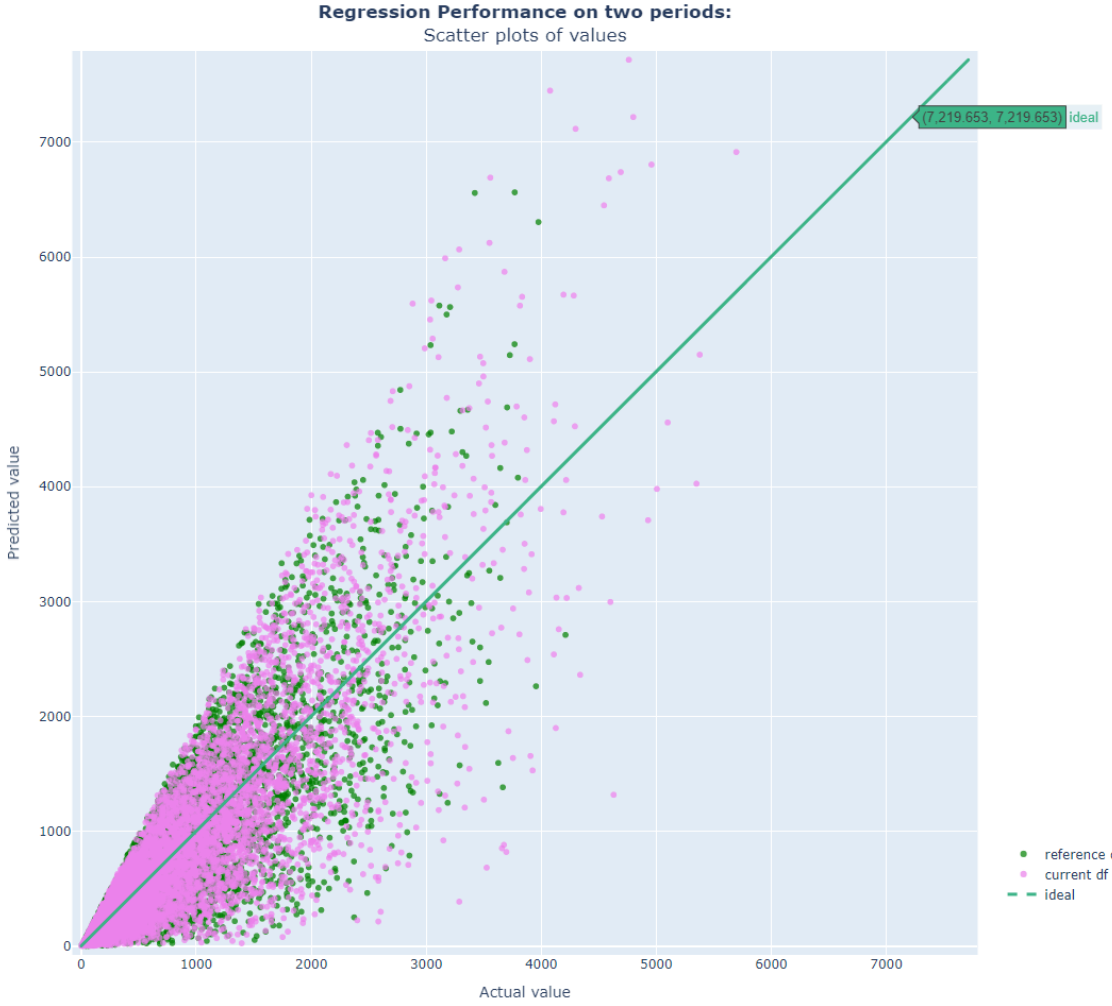
Диаграммы рассеяния в осях актуальные-предсказанные значения:

1. Диаграмма значений на df_reference
2. Диаграмма значений на df_current

Output (short): Отсутствует

Output example (picture):





rvc_10_Reg_Error_Analysis

- **Техническое название**: rvc_10_Reg_Error_Analysis
- **Описание**: Regression Error Analysis. Распределение ошибок регрессии и диаграммы рассеяния в осях актуальные-предсказанные значения для одного датасета
- **Теги**: core, regression
- **requirements**: `typing`, `pandas`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

1) Гистограмма распределения ошибки предсказания;

2) Scatter plot в координатах Actual value - Predicted value.

Метрика аналогична п.9, но предназначена для одного датасета

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца с предсказаниями модели (column)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

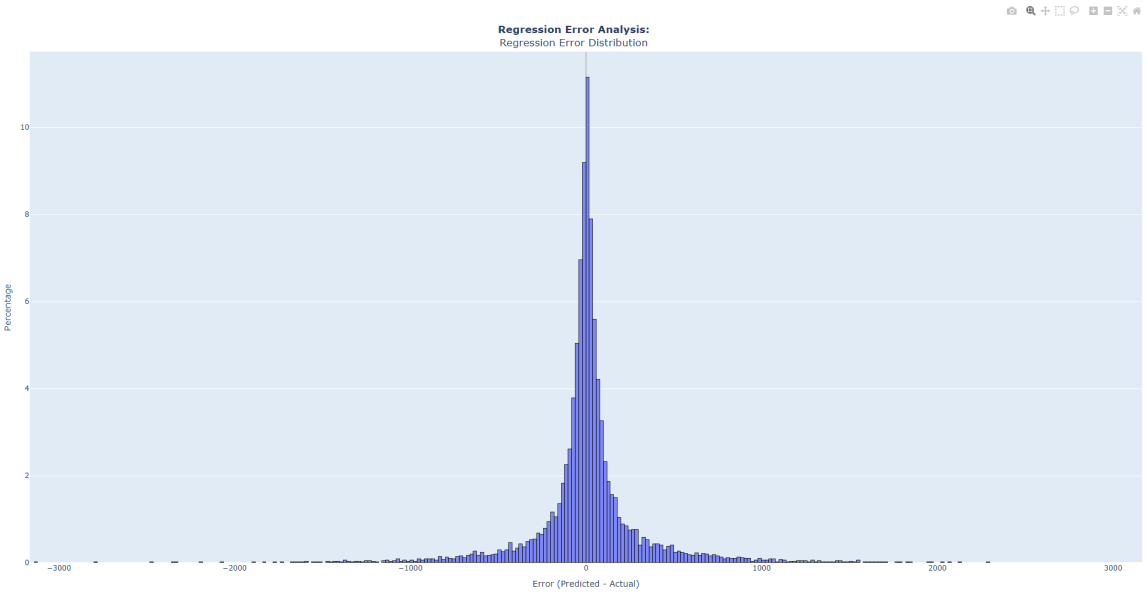
Output (long):

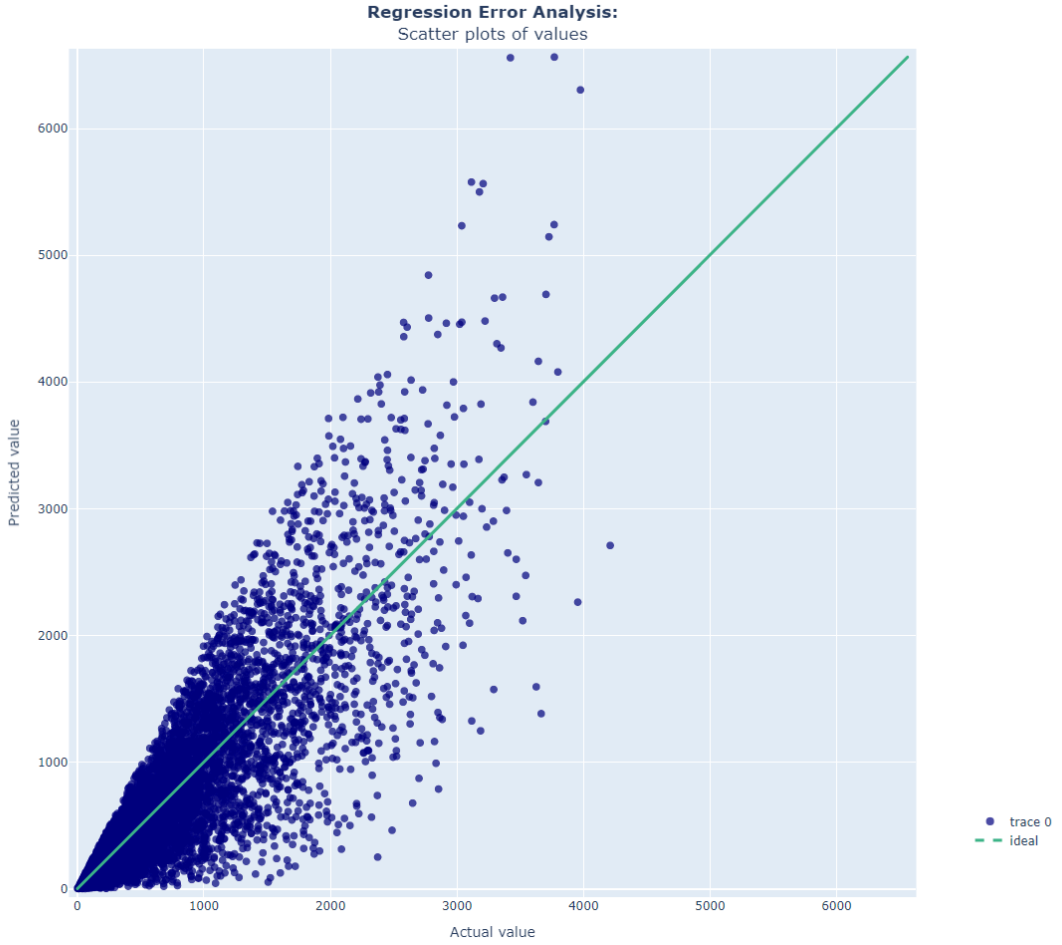
Распределение ошибок регрессии в виде гистограммы

Диаграммы рассеяния в осях актуальные-предсказанные значения

Output (short): Отсутствует

Output example (picture):





Core package (базовый функционал). Метрики качества классификации

Оценка качества бинарной классификации

cvc_1_1_F1_score

- **Техническое название**: cvc_1_1_F1_score
- **Описание**: F1 Score. Оценка F1
- **Теги**: core, classification, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: Реализация может быть доработана для мультиклассовой классификации

ЛОГИКА ИСПОЛНЕНИЯ

Показатель, используемый для измерения точности модели бинарной классификации. Вычисляется по формуле:

\(F_1 = \frac {2 * (precision * recall)}{(precision + recall)}\)

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: БИНАРНАЯ

целевая переменная модели (column)

predict_column: БИНАРНОЕ предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо **Output (long)**: Отсутствует

Output (short): Числовое значение оценки \(F_1\),

светофор

Output example (picture): не применимо.

cvc_1_2_Precision

- **Техническое название**: cvc_1_2_Precision
- **Описание**: Precision score. Оценка точности
- **Теги**: core, classification, scalar

- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: Реализация может быть доработана для мультиклассовой классификации

ЛОГИКА ИСПОЛНЕНИЯ

Показатель, используемый для измерения точности модели бинарной классификации.

Вычисляется как доля истинных положительных предсказаний среди всех предсказанных положительных случаев.

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: БИНАРНАЯ

целевая переменная модели (column)

predict_column: БИНАРНОЕ предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Числовое значение точности,

светофор

Output example (picture): не применимо.

> ***cvc_1_3_Recall***

- **Техническое название**: cvc_1_3_Recall
- **Описание**: Recall score. Оценка полноты
- **Теги**: core, classification, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: Реализация может быть доработана для мультиклассовой классификации

ЛОГИКА ИСПОЛНЕНИЯ

Показатель, используемый для измерения полноты модели бинарной классификации.

Вычисляется по формуле:

Вычисляется как доля истинных положительных предсказаний среди всех фактических положительных случаев.

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: БИНАРНАЯ

целевая переменная модели (column)

predict_column: БИНАРНОЕ предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Числовое значение полноты,

светофор

Output example (picture): не применимо.

> ***cvc_1_4_Chi_Square_Binary***

- **Техническое название**: cvc_1_4_Chi_Square_Binary
- **Описание**: Chi Square Test for binary model. Хи квадрат для модели бинарной классификации
- **Теги**: core, classification, scalar
- **requirements**: `typing`, `pandas`, `scipy.stats`
- **Примечания**: Реализация может быть доработана для мультиклассовой классификации

ЛОГИКА ИСПОЛНЕНИЯ

В общем случае: тест хи-квадрат проверяет нулевую гипотезу о том, что категориальные данные имеют заданное распределение по группам.

В реализации:

\(H_0\) : Распределения *target* и *predict* по классам (категориям *target*) одинаковы.

Если в переменную *predict_column* подается скор (а не бинарный предикт), предикт вычисляется путем сравнения скора с заданным пользователем значением *class_threshold*.

Выполнение \(H_0\) - желаемый исход => чем больше \(P\text{-}value\), тем лучше

Пример выставления *signal bounds*:

\(P\text{-}value ≤ 1(\%)\) - красный,

\(1(\%) < P\text{-}value ≤ 5(\%)\)- желтый,

\(P\text{-}value > 5(\%)\) - зеленый

Применение критерия согласия хи-квадрат:

1) сравнение распределения средних значений *target* и *predict* по группам (например, разрядам рейтинговой шкалы)

2) сравнение распределений *target* и *predict* по классам (например, при калибровке модели)

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: БИНАРНАЯ
целевая переменная модели (column)

predict_column: Предсказания модели - в виде вероятностей или бинарные (column)

class_threshold: Порог отсечения классов (float-value, 0.5 по умолчанию)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

\(P\text{-}value\) критерия хи-квадрат,

светофор

Output example (picture): не применимо.

> ***cvc_1_5_Confusion_Matrix***

- **Техническое название**: cvc_1_5_Confusion_Matrix
- **Описание**: Confusion Matrix. Матрица ошибок
- **Теги**: core, classification
- **requirements**: `typing`, `pandas`
- **Примечания**: Реализация может быть доработана для мультиклассовой классификации

ЛОГИКА ИСПОЛНЕНИЯ

Используется для оценки точности модели в задаче классификации.

Цветом отображается количество:

False Negative (FN), True Positive (TP),

True Negative (TN), False Positive (FP),

решений алгоритма.

Если в переменную score подается именно скор (а не бинарный предикт), предикт вычисляется путем сравнения скора с заданным пользователем значением class_threshold.

(Пример вычисления оптимального порога отсечения в зависимости от выборки см. напр в коде метрики r_4_6 Lift_dynamic)

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: БИНАРНАЯ
целевая переменная модели (column)

predict_column: Предсказания модели - в виде вероятностей или бинарные (column)

class_threshold: Порог отсечения классов (float-value, 0.5 по умолчанию)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Heatmap

Всего 4 поля:

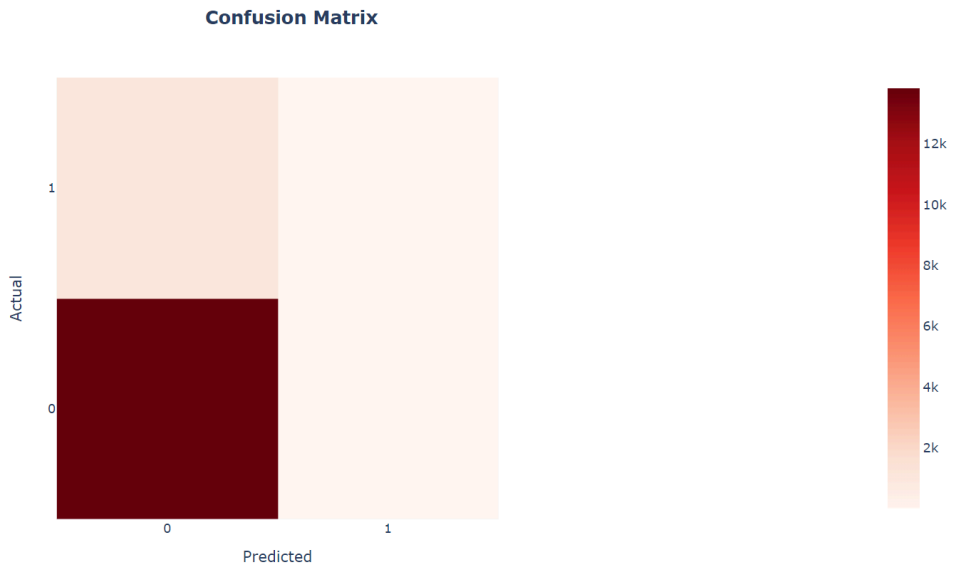
1 строка - FN, TP

2 строка - TN, FP

** При наведении мышки отображается точное количество объектов в каждой категории.*

Output (short): Отсутствует

Output example (picture):



Оценка качества вероятностной классификации

cvc_2_1_ROC_AUC

- **Техническое название**: cvc_2_1_ROC_AUC
- **Описание**: График ROC Curve, значение ROC-AUC
- **Теги**: core, classification, scalar
- **requirements**: `typing`, `pandas`, `numpy`, `sklearn.metrics`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

ROC-кривая используется для оценки качества бинарной классификации, а также вероятностной классификации, для которой производится автоматическое определение порогового значения. Ось X данного графика представляет собой FPR, а ось Y — TPR.

FPR (False Positive Rate) - доля объектов, ошибочно отнесенных алгоритмом к классу 1, среди всех объектов класса 0

TPR (True Positive Rate) - доля верно классифицированных алгоритмом объектов класса 1 среди всех объектов класса 1

AUC = Area Under Curve.

Величина ***ROC AUC*** (площадь под ROC-кривой) характеризует качество классификатора. Чем выше (ближе к 1) показатель AUC, тем качественнее классификатор. Значение 0,5 соответствует случайному классификатору - т.е. модели с ***ROC AUC*** < 0.5 бесполезны.

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: Целевая переменная модели (column)

predict_column: Предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): 2 линейных графика:

ROC curve для модели,

ROC curve для случайного классификатора

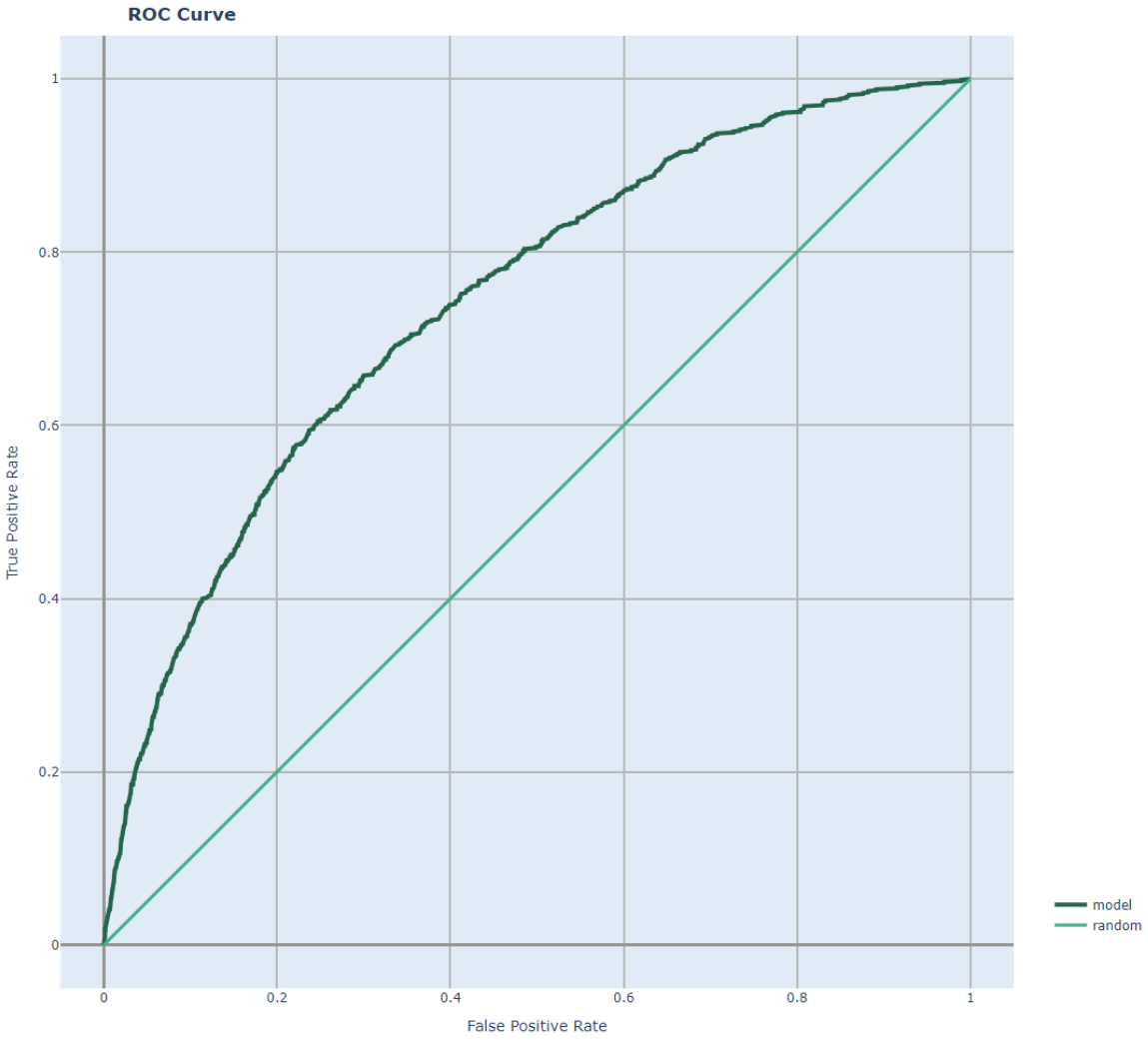
- *xaxis*: FPR
- *yaxis*: TPR

Output (short):

Числовое значение ROC_AUC,

светофор

Output example (picture):



cvc_2_2_Gain_Curve

- **Техническое название**: cvc_2_2_Gain_Curve
- **Описание**: График Cumulative Gain Curve
- **Теги**: core, classification
- **requirements**: `typing`, `pandas`, `numpy`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Оценка модели классификации, рассчитанная путем соотношения результатов, полученных с моделью и без нее. Ось X данного графика представляет собой Percentage of sample, а ось Y — Percentage of positive target.

Percentage of sample - доля пройденного семпла с предсказаниями, отсортированного от самых высоких предсказаний алгоритма

Percentage of positive target - доля объектов, действительно относящихся к целевому классу, в данной доле семпла

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: Целевая переменная модели (column)

predict_column: Предсказание модели (column)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

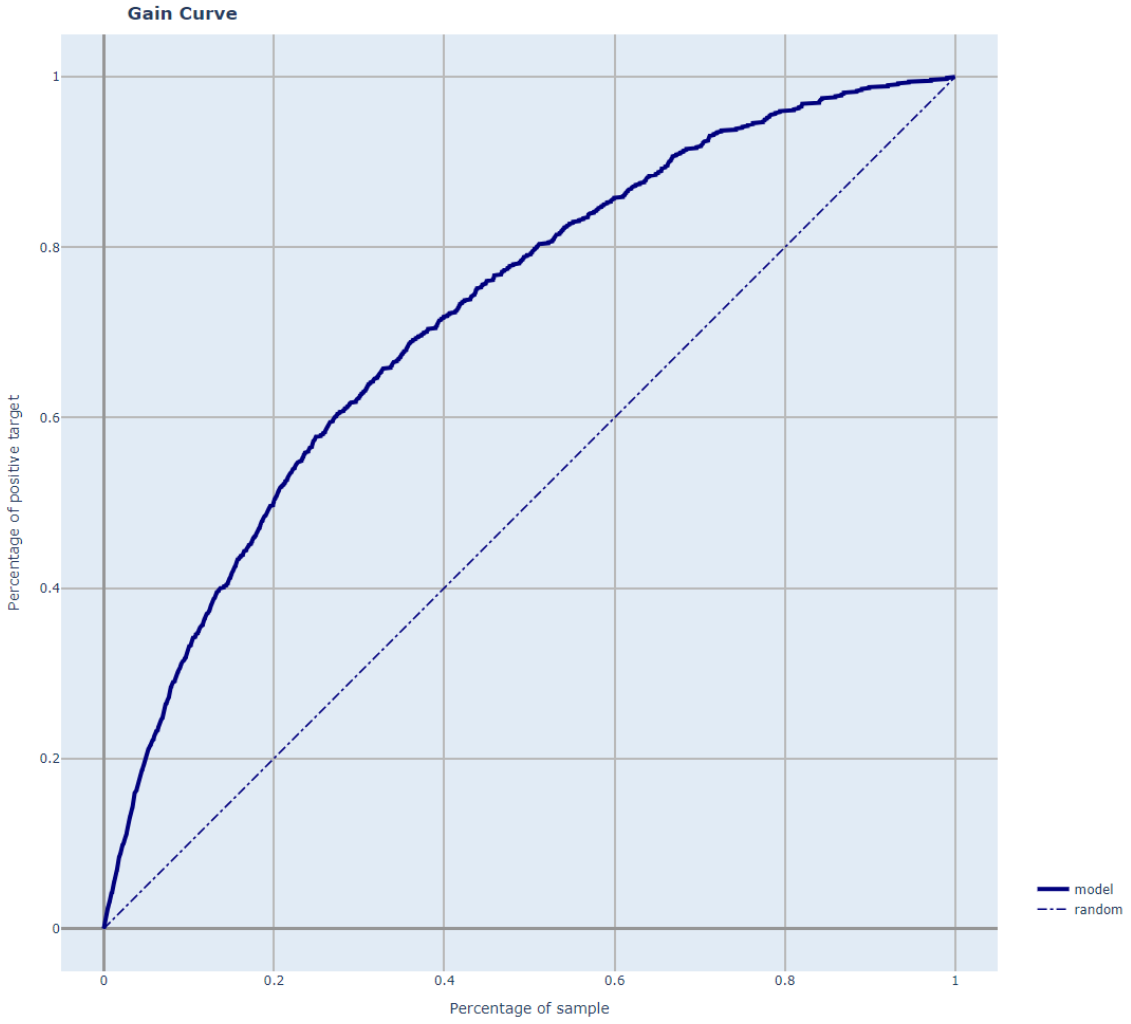
Output (long):

Линейный график

- *xaxis*: Percentage of sample
- *yaxis*: Percentage of positive target

Output (short): Отсутствует

Output example (picture):



cvc_2_3_Lift_Curve_Cumulative

- **Техническое название**: cvc_2_3_Lift_Curve_Cumulative
- **Описание**: График Lift Curve и значение Cumulative Lift для топ n% наблюдений
- **Теги**: core, classification, scalar
- **requirements**: `typing`, `pandas`, `numpy`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

График Lift Curve - это отношение высот Gain-кривой и диагонали. Ось X данного графика представляет собой Percentage of sample (или PS), а ось Y — Lift.

$\text{lift} =\frac{\text{накопленная доля от наблюдений с target_field=1}}{\text{накопленная доля от наблюдений}}$

$\text{lift} = \frac{\text{TPR}}{\text{PS}}$

Пояснения на примере:

predict_column (отсортированный по убыванию): 0.8; 0.7; 0.6; 0.5; 0.4; 0.2

target_column (соотв.): 1; 1; 0; 1; 0; 0

В момент, когда мы прошли первые три объекта выборки:

-- *накопленная доля от наблюдений с target_field=1:*

- всего во всей выборке объектов с target_field=1 - 3,
- в нашей точке (прошли 3 объекта) - 2.

Получается искомая доля = 2\3

-- *накопленная доля наблюдений:*

- всего наблюдений - 6,
- мы прошли - 3.

Получается искомая доля = 3\6

В итоге: lift ~ 0,67\0,5

TPR (True Positive Rate) - доля верно классифицированных алгоритмом объектов класса 1 среди всех "истинных" объектов класса 1 (target_failed = 1). Или другими словами: это доля целевых событий семпла относительно всех целевых событий массива.

PS (Percentage of sample) - доля пройденного семпла с предсказаниями, отсортированного от самых высоких предсказаний алгоритма, относительно всей длины массива с предсказаниями.

Интерпретация "топ n% наблюдений":

- n% объектов с наивысшими оценками алгоритма мы относим к классу 1 (т.е. при PS=n/100)
- Cumulative Lift топ n% = max(Lift для PS>=n/100)

Общие сведения о Cumulative Lift:

- Обычно Cumulative Lift рассчитывают для топ 10% или топ 30% наблюдений.
- Cumulative Lift убывает с ростом n.
- Чем больше значение Cumulative Lift, тем лучше модель.

Пример выставления signal bounds (светофора) для n=10:

$\text{Cumul Lift} \leq 3$ - красный,

$3< \text{Cumul Lift} \leq 3.5$ - желтый,

$\text{Cumul Lift} > 3.5$ - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: Целевая переменная модели (column)

predict_column: Предсказание модели (column)

n_perc: Процент наблюдений для расчета Cumulative Lift (int value; по умолчанию 10)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

Линейный график:

- *xaxis*: PS
- *yaxis*: lift = TPR/PS

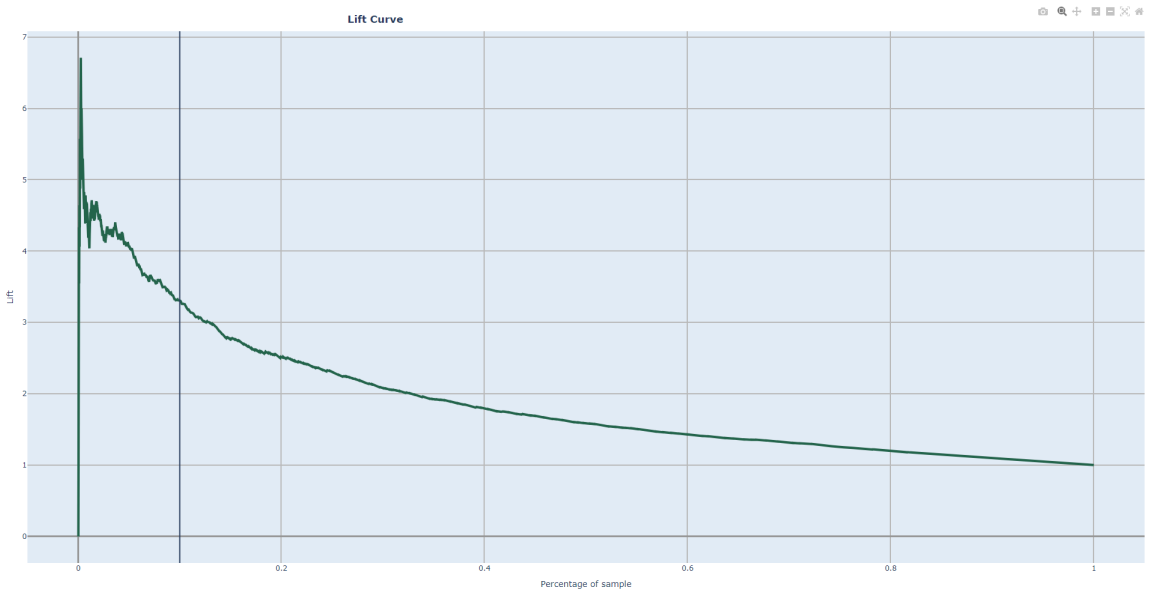
Вертикальная линия:

x = *n_perc*/100

Output (short): Значение *Cumulative Lift* для топ *n*% наблюдений,

светофор

Output example (picture):



cvc_2_4_CAP_Curve_accuracy_rate

- **Техническое название**: cvc_2_4_CAP_Curve_accuracy_rate
- **Описание**: *CAP Curve* and *AR score*. График *Cumulative Accuracy Profile* и значение *accuracy rate* (%)
- **Теги**: core, classification, scalar
- **requirements**: `typing`, `pandas`, `numpy`, `sklearn.metrics`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

CAP curve - это аналог *Gain curve*

Оценка эффективности модели классификации, рассчитанная путем соотношения результатов, полученных с моделью и без нее.

В реализации: вместо долей отображается кол-во объектов из выборки.

Пояснение на примере модели рассчитывающей вероятность дефолта(PD):

CAP-кривая представляет собой кривую, построенную в осях долей наблюдений (ось абсцисс) и долей дефолтов по наблюдениям (ось ординат).

**Наблюдения перед построением упорядочиваются по убыванию PD (вероятности дефолта) контрагентов.*

Совершенная модель довольно быстро достигает 1, т. е. проходит все дефолты.

Произвольная модель равноудалена от обеих осей.

Искомая модель должна работать более точно, чем «бросание монетки», поэтому ее CAP-кривая должна проходить выше, чем у произвольной модели.

Расчет accuracy rate:

Пусть *aP* — площадь между CAP-кривыми совершенной и произвольной моделей,

а *aR* — между CAP-кривыми рассматриваемой модели и произвольной моделью соответственно.

Тогда искомый индекс точности (*accuracy rate*) определяется как \(\frac{\text{aR}}{\text{aP}}*100\%\).

Сво-ва accuracy rate:

- Максимум данного соотношения — 1 достигается, в случае, если модель является совершенной моделью.
- Положительные ненулевые значения индекса говорят о том, что модель работает лучше, чем «бросание монетки».

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: Целевая переменная модели (column)

predict_column: Предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Линейный график

- *xaxis*:

Накопленное кол-во наблюдений (отсортированных по убыванию predict_column) *(*аналог PR)*

- *yaxis*:

Накопленное кол-во объектов класса 1 (target_filed=1 в массиве df, строки которого остортированы по убыванию predict_column).

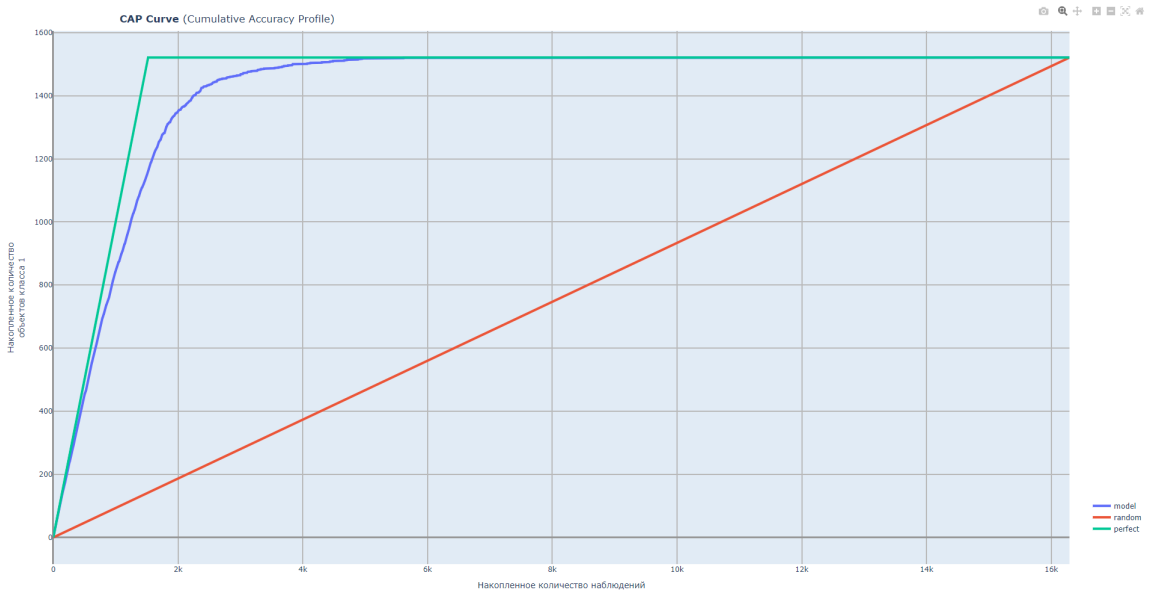
*(*аналог TPR)*

Output (short):

Число: значение *accuracy rate* на основе *CAP* по модели (в %)

светофор

Output example (picture):



cvc_2_5_PR_Curve_PRC_AUC

- **Техническое название**: cvc_2_5_PR_Curve_PRC_AUC
- **Описание**: PR Curve and PRC-AUC. График Precision-Recall Curve и Значение PRC-AUC
- **Теги**: core, classification, scalar
- **requirements**: `typing`, `pandas`, `numpy`, `sklearn.metrics`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

PR Curve - это график отношения recall(полноты) и precision(точности) с разными границами. Ось X данного графика представляет собой Recall, а ось Y — Precision. Каждой точке на этой кривой будет соответствовать классификатор с некоторым значением границы.

** В случае идеального классификатора, то есть если существует такая граница(порог), что и точность, и полнота рав- ны 100%, кривая будет проходить через точку (1,1). Таким образом, чем ближе кривая пройдет к этой точке, тем лучше оценки.*

Площадь под этой кривой называется ***PRC-AUC***, или площадь под PR-кривой.

$\text{Precision} = \frac{TP}{TP + FP}$, доля верно классифицированных алгоритмом объектов класса 1 среди всех объектов, классифицированных алгоритмом классом 1.

$\text{Recall} = \frac{TP}{TP + FN}$, доля верно классифицированных алгоритмом объектов класса 1 среди всех объектов класса 1.

Граница (порог) - это значение, по сравнению с которым определяется, к какому классу будет отнесён объект данных

ВХОДНЫЕ ПАРАМЕТРЫ

df: Объект данных для расчета (dataframe)

target_column: Целевая переменная модели (column)

predict_column: Предсказание модели (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Линейный график:

1) PR Curve:

- *xaxis*: recall (полнота)
- *yaxis*: precision (точность)

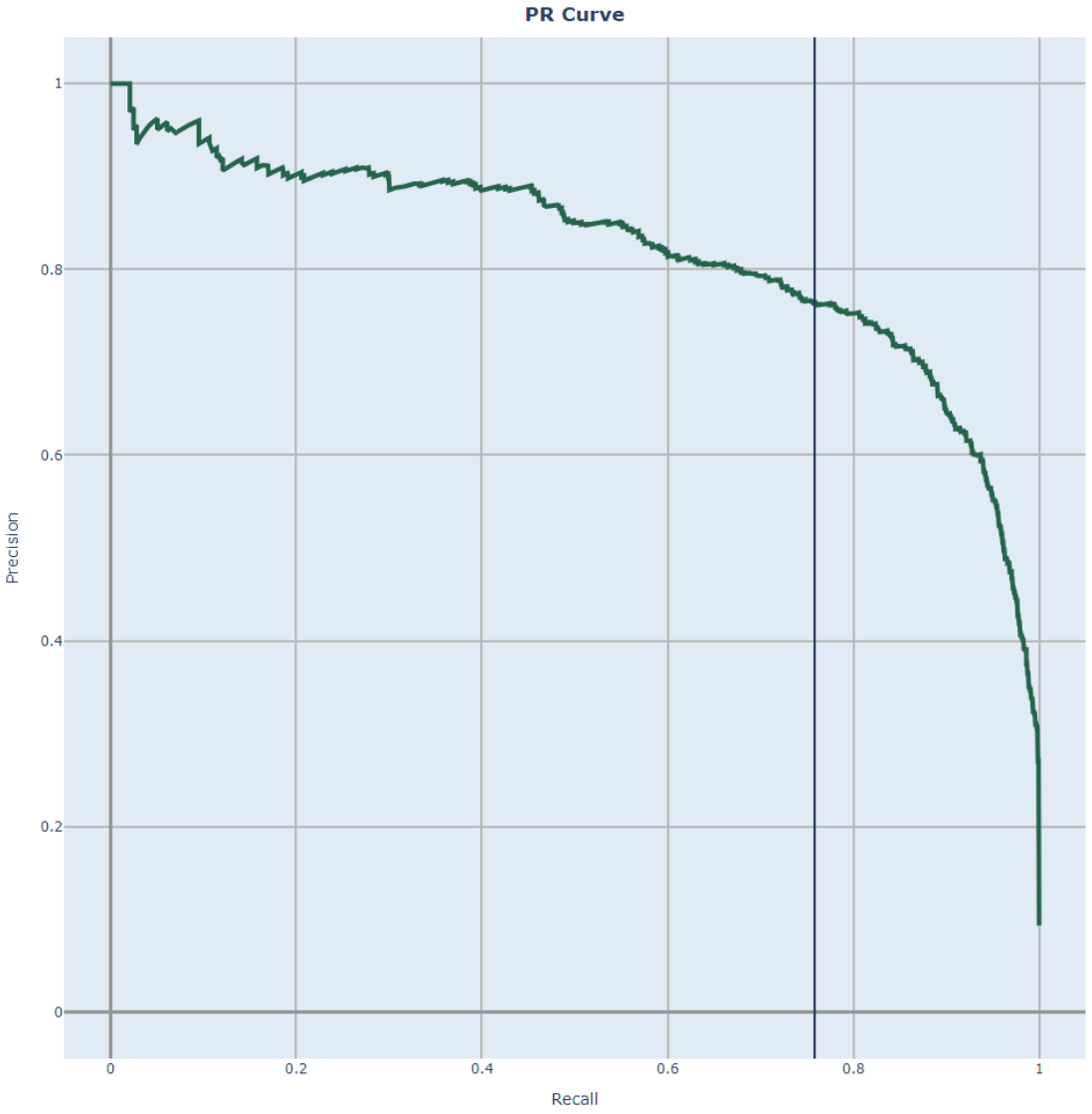
2) Вертикальная линия: x = оптимальная граница для значений predict_column

Output (short):

Числовое значение *PRC-AUC*,

светофор

Output example (picture):



cvc_2_6_AUCs_Dynamic

- **Техническое название**: cvc_2_6_AUCs_Dynamic
- **Описание**: AUCs Dynamic. Динамика *ROC AUC* и *PRC AUC* модели на исторических данных
- **Теги**: core, classification
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Наблюдения из *df* разбиваются на группы по значениям столбца с датой (*report_dt*).

В одну группу попадают все наблюдения за period (месяцы, квартал или год).

Для наблюдений каждой группы рассчитываются *ROC AUC* и *PRC AUC*.

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

report_dt_column: название столбца с датой (column)

period: гранулярность данных: *dropdown*:

- одно из значений типа str: 'month', 'quarter', 'year';
- по умолчанию -'quarter'

РЕЗУЛЬТАТЫ

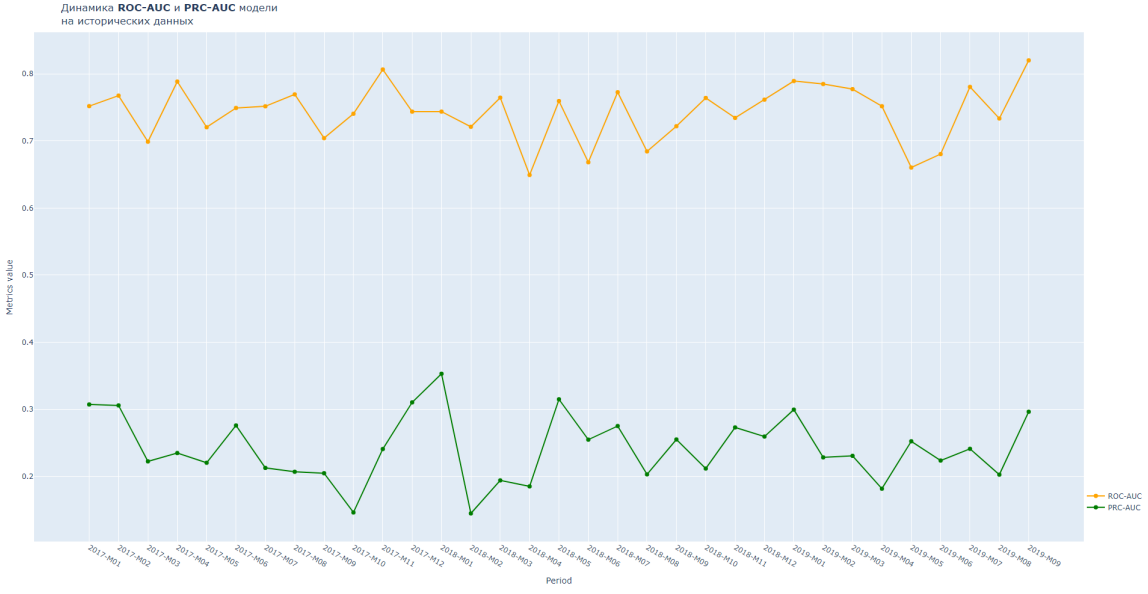
Движок отрисовки графика: plotly.js

Output (long): Массив графиков:

- *xaxis*: период
- *yaxis*: значения *ROC AUC* и *PRC AUC*, рассчитанные на наблюдениях за соответствующий период.

Output (short): Отсутствует

Output example (picture):



cvc_2_7_Gini_model

- **Техническое название**: cvc_2_7_Gini_model
- **Описание**: *Gini Index (%) for Model.* Индекс Джини (%) по модели
- **Теги**: core, classification, risk, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

1. Расчет ROC AUC по target_field, score_field
2. [Gini index] = 2 * [ROC AUC] - 1

Индекс Джини позволяет оценить качество предсказательной способности модели относительно случайного классификатора.

Чем выше индекс Джини, тем лучше модель справляется с разделением классов.

Джини от 0 до 1 (до 100%) - модель лучше случайного классификатора

Джини меньше 0 - модель хуже случайного классификатора

Пример выставления signal bounds:

$\text{Gini} \leq 40(\%)$ - красный,

$40(\%) < \text{Gini} \leq 60(\%)$ - желтый,

$\text{Gini} > 60(\%)$ - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Число: значение индекса Джини (в %),

светофор

Output example (picture): не применимо.

> ***cvc_2_8_KS_Test_Predicts***

- **Техническое название**: cvc_2_8_KS_Test_Predicts
- **Описание**: Kolmogorov-Smirnov Test. Тест Колмогорова-Смирнова. Сравнивает распределения score для "хороших"/"плохих" клиентов
- **Теги**: core, classification, scalar
- **requirements**: `typing`, `pandas`, `numpy`, `scipy.stats`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Сравниваются распределения score для "хороших"(target==0) и "плохих"(target==1) клиентов.

H_0: распределения одинаковы.

Желательным является наличие статистически значимых различий между группами (это означ, что модель хорошо разделяет классы) => чем меньше \(P\text{-}value\) тем лучше

Реализация: Двухвыборочный критерий Колмогорова-Смирнова ks_2samp из scipy.stats

(проверка гипотезы о принадлежности значений двух независимых выборок к одному и тому же закону распределения)

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long): Массив графиков

Output (short):

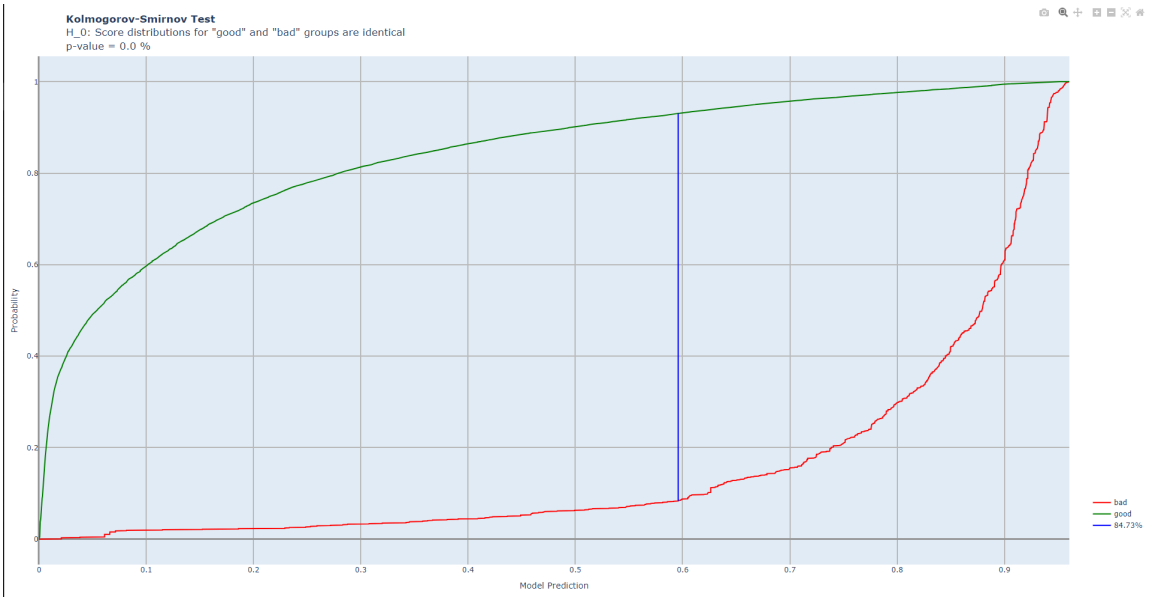
Числовые значения:

1) Статистики K-C, % (скаляр)

2) \(P\text{-}value\) теста K-C, % (на графике)

светофор на Статистики K-C, %

Output example (picture):



cvc_2_9_Barometers_by_bins

- **Техническое название**: cvc_2_9_Barometers_by_bins
- **Описание**: Table of barometers by bins. Таблица показателей по бакетам. Показатели по бакетам: точность, полнота, лифт, max, min и средний скор-балл, и др
- **Теги**: -
- **requirements**: `typing`, `pandas`, `numpy`, `sklearn.metrics`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

df разбивается на заданное число (nbins) бакетов по квантилям score.

Для каждого бакета (бина) выводятся:

- номер бина
- число записей в бине
- min score в бине
- max score в бине

- средний score в бине
- к-во записей с target==1 в бине
- cumsum (target) по агрегированной таблице
- cumsum (числа записей в бине) по агрегированной таблице
- recall (полнота)
- precision (точность)
  - lift

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

nbins: число бакетов (int value; по умолчанию 20)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

Таблица 11 столбцов (по числу рассчитываемых показателей), число строк = числу бакетов

Output (short): -

Output example (picture):

Значения показателей по бакетам с шагом 7 %

Номер бакета	Всего записей	Min score	Max score	Средний score	Записей с target=1	Cumsum записей с target=1	Cumsum числа записей	Полнота (recall)	Точность (precision)	Lift
15	800	0.1889	0.7166	0.2751	225	225	800	1	0.2812	3.6562
14	800	0.139	0.1888	0.16	140	365	1600	0.9857	0.1767	2.2971
13	800	0.1125	0.139	0.1256	99	464	2400	0	0	1
12	799	0.0946	0.1125	0.1029	85	549	3199	0	0	1
11	793	0.0821	0.0946	0.088	59	608	3992	0	0	1
10	807	0.0705	0.082	0.0759	56	664	4799	0	0	1
9	801	0.0625	0.0705	0.0666	45	709	5600	0	0	1
8	799	0.0545	0.0625	0.0583	43	752	6399	0	0	1
7	800	0.0483	0.0545	0.0512	40	792	7199	0	0	1
6	784	0.0422	0.0483	0.0452	40	832	7983	0	0	1
5	816	0.0367	0.0422	0.0395	34	866	8799	0	0	1
4	800	0.0314	0.0367	0.0341	21	887	9599	0	0	1
3	800	0.0263	0.0314	0.0289	17	904	10399	0	0	1
2	799	0.0209	0.0263	0.0236	12	916	11198	0	0	1
1	801	0.0072	0.0209	0.0167	7	923	11999	0	0	1

Risk validation package

Качество данных

r_1_1_PSI_field

- **Техническое название**: r_1_1_PSI_field
- **Описание**: PSI Features. Population stability index (PSI) for features
- **Теги**: risk
- **requirements**: `typing`, `pandas`, `numpy`, `plotly.graph_objects`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

PSI - мера удаленности двух распределений

Для каждого из выбранных факторов определяется степень различия распределений этого фактора на выборках для обучения и тестирования

В цикле для каждого field из fields_to_test:

1. Если признак категориальный, то перейти к шагу 2. Если непрерывный, то нужно провести автоматический биннинг (10-20 бакетов).
2. Для каждой категории отдельно на train и test рассчитываем % наблюдений, принадлежащий этой категории.
3. temp_i = (%test - %train) * ln(%test / %train)
4. PSI = сумма temp_i по всем категориям

Возможный диапазон значений PSI: (0; +inf).

Чем ниже PSI, тем меньше отличаются распределения признака на train и test => Чем ниже PSI для фактора, тем лучше (тем стабильнее) этот фактор

Пример выставления signal bounds: PSI < 0.1 - зеленый, 0.1 < PSI < 0.25 - желтый, PSI > 0.25 - красный

ВХОДНЫЕ ПАРАМЕТРЫ

df_train: датасет с данными для обучения (dataframe)

df_test: датасет с данными для теста (dataframe)

field_columns: массив названий атрибутов, по которым считаем PSI (multi-column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора (! названия столбцов из field_columns должны совпадать в df_train и df_test)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

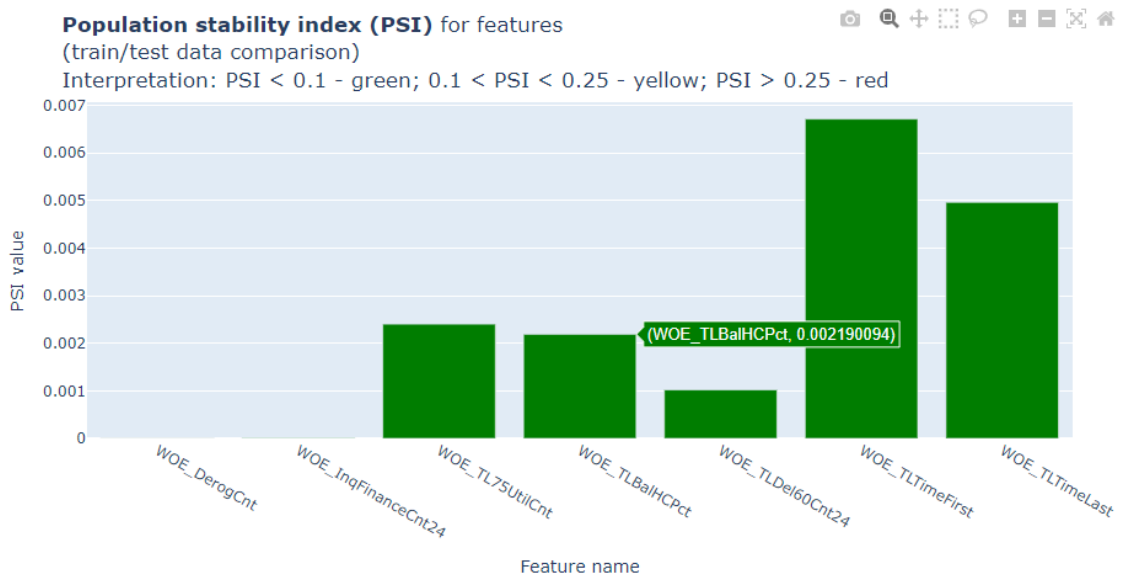
Barchart xaxis: Столбцы соответствуют признакам, для которых производился расчет yaxis : Высота столбца - значение PSI для признака

светофор: столбцы окрашиваются в цвет светофора для соответствующего признака

Output (short):

Отсутствует

Output example (picture):



r_1_2_Default_rate_dynamic

- **Техническое название**: r_1_2_Default_rate_dynamic
- **Описание**: Default Rate Dynamic. Уровень дефолта на исторических данных:
  - доля (%) наблюдений с дефолтом по периодам,
  - общее к-во наблюдений по периодам.
- **Теги**: risk
- **requirements**: `typing`, `pandas`, `plotly.graph_objects`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

Группируем данные с гранулярностью period.

Для каждой группы (периода) отрисовываем на комбинированном графике:

- количество наблюдений
- % наблюдений с дефолтом = (к-во наблюдений с [target=1] за период) / (к-во наблюдений за период) * 100%

Тест обычно используется для первичной валидации модели. (Например, для оценки соотношения классов в выборках, поиска периодичности во времени, отсечения слишком старых и неактуальных данных)

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

report_dt_column: название столбца с датой (column)

period: гранулярность данных
(dropdown - одно из значений типа str: 'month', 'quarter', 'year'; по умолчанию -'quarter')

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

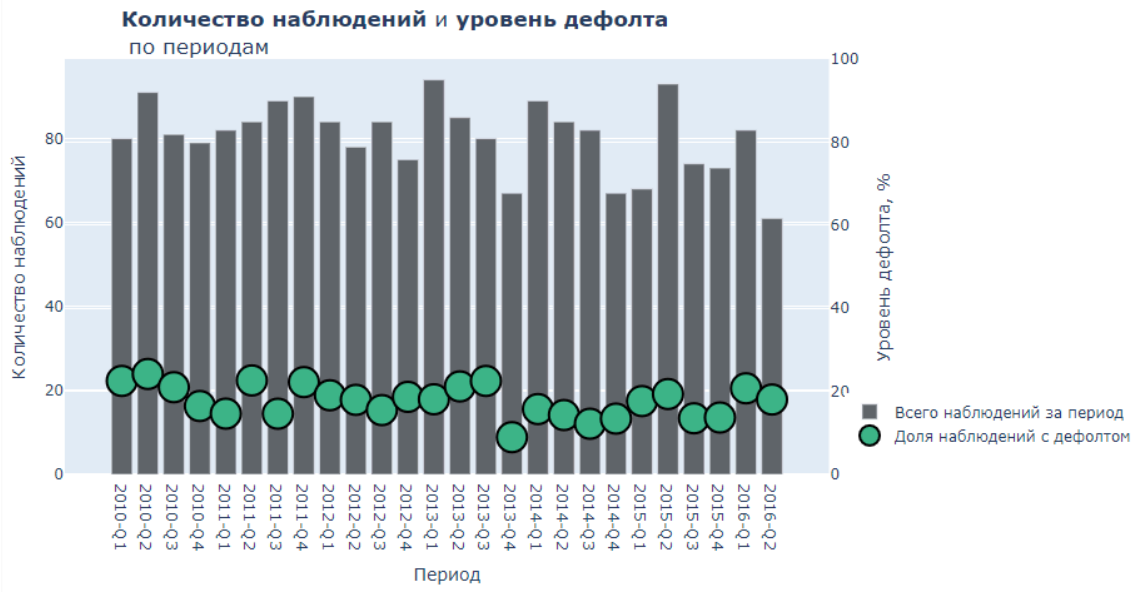
Output (long):

Barchart xaxis: Столбцы соответствуют периодам yaxis : Высота столбца (шкала слева) - число наблюдений в датасете, относящихся к выбранному периоду Точка (шкала справа) - доля (%) наблюдений с дефолтом в выбранный период

Output (short):

Отсутствует

Output example (picture):



Ранжирующая способность

cvc_2_7_Gini_model (r_2_1_Gini_model)

- **Техническое название**: cvc_2_7_Gini_model (r_2_1_Gini_model)
- **Описание**: Gini Index (%) for Model. Индекс Джини (%) по модели
- **Теги**: -
- **requirements**: -

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

См. раздел "Core package/Метрики качества классификации", пункт 2.7

ВХОДНЫЕ ПАРАМЕТРЫ

-

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Отсутствует

Output example (picture): не применимо.

cd_4_2_Gini_features (r_2_2_Gini_features)

- **Техническое название**: cd_4_2_Gini_features (r_2_2_Gini_features)
- **Описание**: Gini Index (%) for Features. Индекс Джини (%) в разрезе отдельных факторов
- **Теги**: -
- **requirements**: -

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

См. раздел "Core package/Анализ данных", пункт 4.2

ВХОДНЫЕ ПАРАМЕТРЫ

-

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Отсутствует

Output example (picture): не применимо.

r_2_3_Gini_bootstrap_model

- **Техническое название**: r_2_3_Gini_bootstrap_model
- **Описание**: Bootstrapped Gini Index (%) for Model. Индекс Джини (%) по модели на bootstrap-подвыборках. Интервальная версия точечной оценки Джини (см. 2.1). Используется при наличии дисбаланса классов в выборке.
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`, `joblib`, `numpy`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

1. Собираем выборку из bootstrap_n значений Gini, рассчитанных на bootstrap-подвыборках
2. Выбираем из полученной выборки перцентиль perc

При несбалансированной выборке это позволяет избежать смещения точечной оценки. По умолчанию берется 2.5 перцентиль распределения Gini.

Чем больше значение Bootstrapped Gini, тем лучше предсказательная способность модели.

Пример выставления signal bounds: Bootstrapped Gini ≤ 30(%) - красный,
30(%) < Bootstrapped Gini ≤ 45(%) - желтый,
Bootstrapped Gini > 45(%) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

bootstrap_n: количество bootstrap-подвыборок (int value; по умолчанию 1000)

perc: перцентиль, по которому проводим итоговую оценку (float value; по умолчанию 2.5)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Число: значение индекса Джини (в %)
(2.5 перцентиль распределения Джини на бутстраппированных подвыборках из исходного df),

светофор

Output example (picture): не применимо.

r_2_4_Gini_bootstrap_field

- **Техническое название**: r_2_4_Gini_bootstrap_field

- **Описание**: Bootstrapped Gini Index (%) for Features. Индекс Джини (%) в разрезе отдельных факторов на bootstrap-подвыборках. Интервальная версия оценки Джини для факторов (см. 2.2). Используется при наличии дисбаланса классов в выборке.
- **Теги**: risk
- **requirements**: `typing`, `pandas`, `sklearn.metrics`, `joblib`, `numpy`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

В цикле для каждого field из fields:

1. Собираем выборку из bootstrap_n значений Gini, рассчитанных на bootstrap-подвыборках
2. Выбираем из полученной выборки перцентиль perc

При несбалансированной выборке это позволяет избежать смещения точечной оценки. По умолчанию берется 2.5 перцентиль распределения Gini.

Если в признаке (field) или target присутствует пропущенное значение, такая строка исключается из рассмотрения. Для разных признаков исключаются из рассмотрения разные строки. Чем больше значение Bootstrapped Gini для фактора, тем лучше ранжирующая способность этого фактора.

Пример выставления signal bounds: Bootstrapped Gini ≤ 3(%) - красный,
3(%) < Bootstrapped Gini ≤ 5(%) - желтый,
Bootstrapped Gini > 5(%) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

field_columns: массив названий столбцов, по которым считаем тест (multi-column)

bootstrap_n: количество bootstrap-подвыборок (int value; по умолчанию 1000)

perc: перцентиль, по которому ставим итоговую оценку (float value; по умолчанию 2.5)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

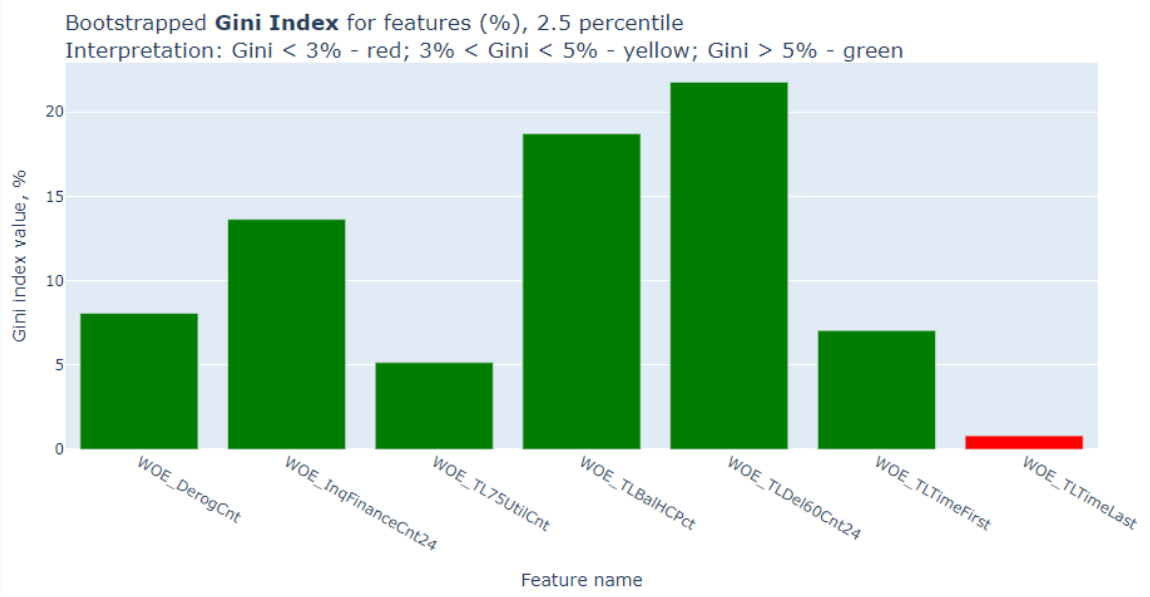
Barchart: xaxis : названия столбцов, для которых производился расчет yaxis : значения коэффициента Джини (в %)
(2.5 перцентиль распределения Джини для фактора на бутстраппированных подвыборках из

светофор: столбцы окрашиваются в цвет светофора для соответствующего признака

Output (short):

Отсутствует

Output example (picture):



r_2_5_KS_on_scale

- **Техническое название**: r_2_5_KS_on_scale
- **Описание**: KS-test on scale. Тест Колмогорова-Смирнова. Проверяет насколько отличаются 2 распределения и показывает насколько хорошо score модели отделяет хороших клиентов от плохих в разрезе рейтинговой шкалы.
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `numpy`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

1. Собираем 2 кумулятивных распределения:
    a. Good - доля клиентов с [target_field = 0] с группировкой по scale_field
    b. Bad- доля клиентов с [target_field = 1] с группировкой по scale_field
2. Считаем значение статистики max(\Good-Bad) по всем разрядам шкалы. Чем больше, чем лучше.

Альтернатива cvc_2_8_KS_Test_Predicts с использованием scale шкалы. Обычно разряды рейтинговой шкалы присваиваются на основании score модели.

Формирование scale: по предсказанию модели (score) каждому из наблюдений присваивается разряд рейтинговой шкалы => получаем категориальную переменную

Интерпретация графика: для хорошей модели кривые bad и good д.б. удалены друг от друга.

H_0 теста K-C: выборки взяты из одного распределения.

Нам нужно, чтобы bad и good максимально отличались =>
Чем больше значение статистики K-C, тем лучше.

Пример выставления signal bounds: KC ≤ 30(%) - красный,
30(%) < KC ≤ 40(%) - желтый,
KC > 40(%) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

scale_column: название столбца с присвоенным разрядом рейтинговой шкалы (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

Массив графиков:

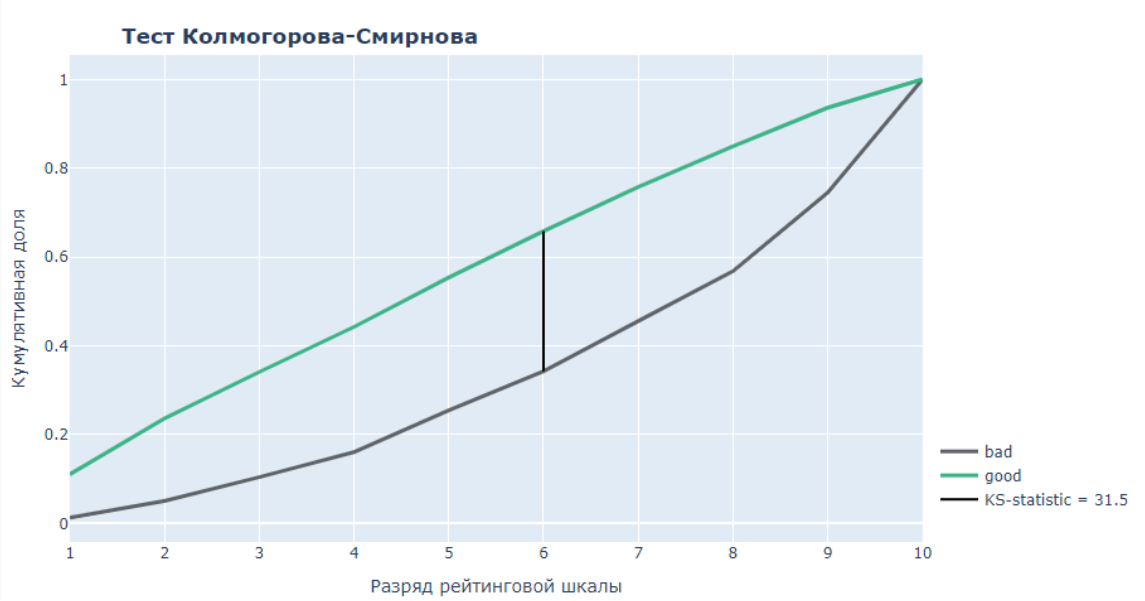
- xaxis : разряды рейтинговой шкалы
- yaxis : кумулятивная вероятность,
- Значение статистики K-C (в %) - в легенде

Output (short):

Число: Значение статистики К-С (в %)

светофор

Output example (picture):



r_2_6_IV_model

- **Техническое название**: r_2_6_IV_model
- **Описание**: Information Value для модели. Показывает степень отличия распределений predict для хороших и плохих клиентов
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `numpy`, `scipy.stats.stats`, `sklearn`, +, Преобразование WOE, созданное отдельным классом в том же файле метрики

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

1. Провести автоматический биннинг score_field (10-20 бакетов).
2. Для каждой категории рассчитываем:
    a. %good - процент наблюдений с [target=0]
    b. %bad- процент наблюдений с [target=1]
3. temp_i = (%good- %bad) * ln(%good/ %bad)
4. IV = сумма temp_i по всем категориям

Формула аналогична PSI, но если PSI сравнивает распр-е переменной на разных временных срезах, то IV сравнивает распределение score для хороших и плохих клиентов (на "хороших" и "плохох" разбиваем по знач-ю target)

Чем сильнее отлич-ся эти распр-я, тем лучше =>
чем больше IV, тем лучше.

Пример выставления signal bounds: IV ≤ 10(%) - красный,
10(%) < IV ≤ 30(%) - желтый,
IV > 30(%) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Число: значение Information Value для модели (в %),

светофор

Output example (picture): не применимо.

r_2_7_IV_field

- **Техническое название**: r_2_7_IV_field
- **Описание**: Information value в разрезе отдельных факторов. Показывает степень отличия распределений фактора для хороших и плохих клиентов
- **Теги**: risk
- **requirements**: `typing`, `pandas`, `numpy`, `scipy.stats.stats`, `sklearn`, +, Преобразование WOE, созданное отдельным классом в том же файле метрики

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

В цикле для каждого field из fields:

1. Если field категориальный, то перейти к п.2, иначе провести автоматический биннинг (10-20 бакетов).
2. Для каждой категории рассчитываем:
    a. %good - процент наблюдений с [target=0]
    b. %bad- процент наблюдений с [target=1]
3. temp_i = (%good- %bad) * ln(%good/ %bad)
4. IV = сумма temp_i по всем категориям

Чем больше IV для признака, тем лучше (тем информативнее этот признак).

Тест используется для отбора признаков модели. (См. статью о WOE и IV)

Пример выставления signal bounds: IV ≤ 10(%) - красный,
10(%) < IV ≤ 30(%)- желтый,
IV > 30(%) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

field_columns: массив названий столбцов, по которым считаем тест (multi-column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

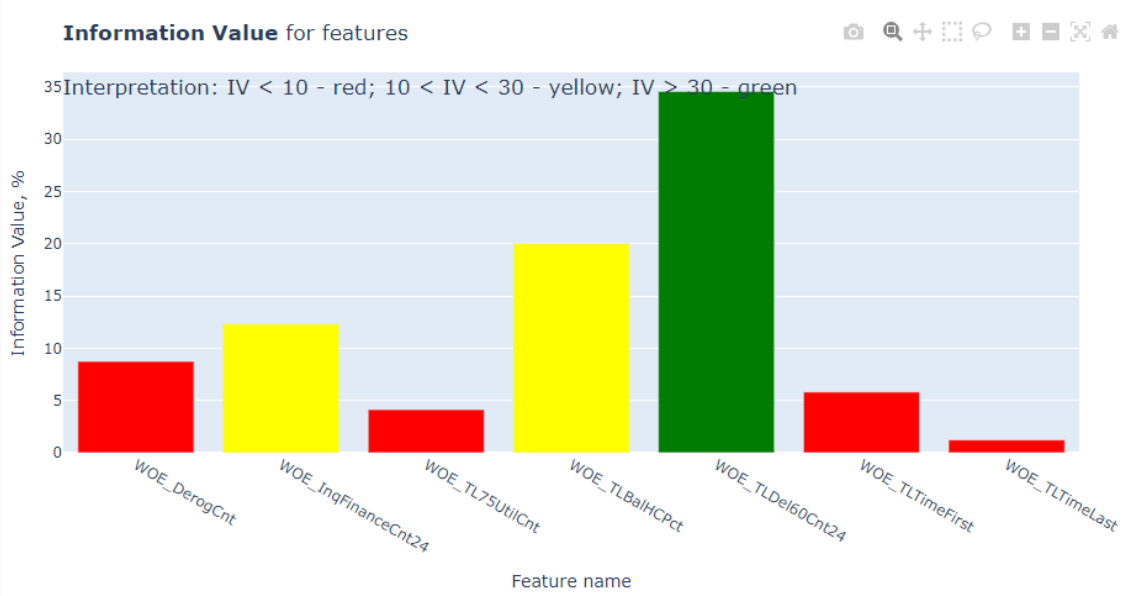
Barchart:

- xaxis : названия столбцов, для которых производился расчет
- yaxis : значения Information Value (в %)
- светофор: столбцы окрашиваются в цвет светофора для соответствующего признака

Output (short):

Отсутствует

Output example (picture):



r_2_8_HL_test

- **Техническое название**: r_2_8_HL_test
- **Описание**: Hosmer–Lemeshow Test. Тест Хосмера-Лемешова (хи-квадрат). Проверяет схожесть распределений среднего фактического и среднего прогнозного уровня дефолта по бакетам
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `scipy.stats`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

1. Для i-го разряда шкалы рассчитываются параметры:
    a. \(N_i\) - количество наблюдений в бакете
    b. \(PD_i\) - среднее значение score_field по бакету (прогнозный уровень дефолта)
    c. \(DR_i\) - среднее значение target_field по бакету (фактический уровень дефолта)

2. Рассчитывается статистика Хосмера-Лемешова по формуле

$$T_n = \sum_{i=0}^{n} \frac{(N_i PD_i - DR_i)^2}{N_i PD_i (1 - PD_i)},$$

3. \(P\text{-}value\) определяется по распределению хи-квадрат (scipy.stats.chi2)

H_0: распределения по группам одинаковы.

Мы стремимся к тому, чтобы распределения по бакетам для target и score были как можно более схожи => Чем выше \(P\text{-}value\), тем лучше

Пример выставления signal bounds:
\(P\text{-}value ≤ 1(\%)\) - красный,
$1(\%) < \(P\text{-}value ≤ 5(\%)\)- желтый,
\(P\text{-}value > 5(\%)\) - зеленый

Используется также для калибровки модели.

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

scale_column: название столбца с присвоенным разрядом рейтинговой шкалы (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Число: \(P\text{-}value\) теста Хосмера-Лемешова (в %),

светофор

Output example (picture): не применимо.

> ***r_2_9_MW_test***

- **Техническое название**: r_2_9_MW_test
- **Описание**: Mann-Whitney Test. Тест Манна-Уитни (левосторонний). Проверка схожести распределений score и target по бакетам (ранговый метод)
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `scipy.stats`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

1. Разбиваем score на nbins бакетов по квантилям --> получим категориальную переменную с nbins значений.
2. Группируем наблюдения по новой категориальной переменной. В каждой группе рассчитываем mean(score) и mean(target) --> получим две выборки: средние score по бакетам и средние target по бакетам
3. С помощью непараметрического критерия Манна-Уитни (scipy.stats.mannwhitneyu) проверяем гипотезу о принадлежности этих выборок к одной и той же генеральной совокупности.
4. Выводим \(P\text{-}value\).

Мы стремимся чтобы распределения по бакетам для score и target были как можно более схожи => Чем выше \(P\text{-}value\), тем лучше

Используется для: калибровки модели, проверки стабильности.

Пример выставления signal bounds:
\(P\text{-}value ≤ 1(\%)\) - красный,
\(1(\%) < P\text{-}value ≤ 5(\%)\)- желтый,
\(P\text{-}value > 5(\%)\) - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

nbins: число бакетов, на которое score разбивается по квантилям
(int value; по умолчанию 50)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Число: \(P\text{-}value\) теста Манна-Уитни (в %),

светофор

Output example (picture): не применимо.

Спецификация (только для линейных модели)

r_3_1_Monotony_field

- **Техническое название**: r_3_1_Monotony_field
- **Описание**: Monotony Field. Монотонность фактора. Доля дефолтов и количество наблюдений по категориям фактора на train/test
- **Теги**: risk
- **requirements**: `typing`, `pandas`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

Разбиваем наблюдения на группы в соответствии со значениями категориальной переменной field. Число групп = числу уникальных значений field.

Для категориального field отрисовываем комбинированный график:

1. Barchart: % наблюдений в каждой категории field на train и test = (наблюдений в выбранной категории) / (всего наблюдений в выборке) * 100
2. Линия: % дефолтов в каждой категории field на train и test = (наблюдений с target==1 в выбранной категории) / (всего наблюдений в выбранной категории) * 100

Тест применяется если в модели использовались биннинги или WOE (Weight of Evidence). Для упорядоченных WOE доля дефолтов должна строго возрастать

ВХОДНЫЕ ПАРАМЕТРЫ

df_train: датасет с данными для обучения (dataframe)

df_test: датасет с данными для теста (dataframe)

target_column: название столбца с таргетом (column)

cat_field_column: название столбца с категориальной переменной по значениям которой разбиваем наблюдения на группы (column) (! названия столбцов с target_column и cat_field_column должны совпадать в df_train и df_test)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

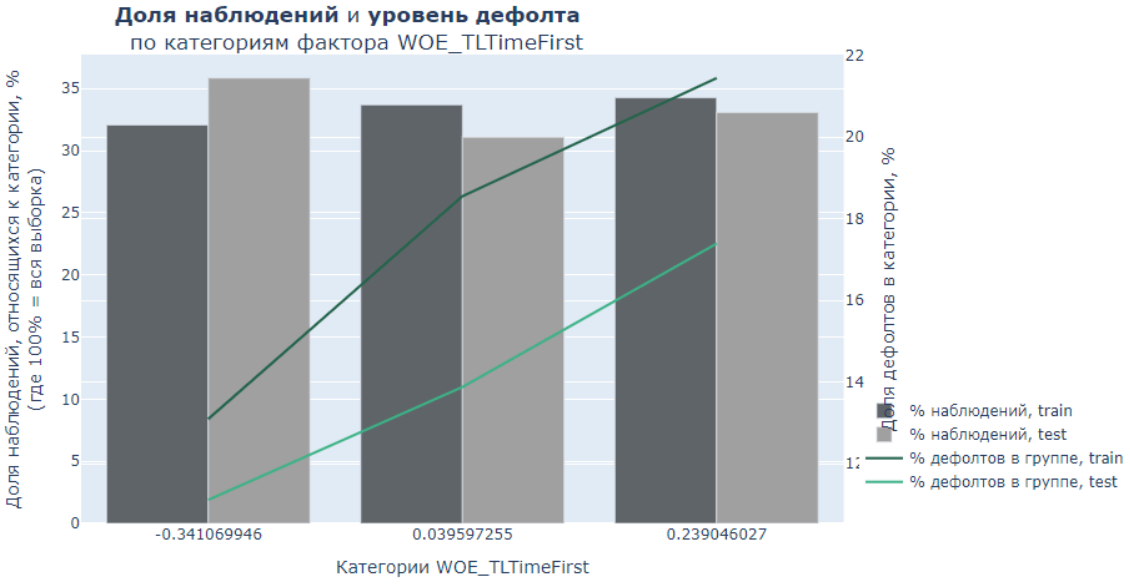
Output (long):

Массив графиков Barchart: xaxis: каждая пара столбцов соответствует одному значению категориальной переменной field на train/test yaxis (шкала слева) : высота столбца = доля всех наблюдений, относящихся к выбранной категории Линии: yaxis (шкала справа): доля дефолтов в выбранной категории

Output (short):

Отсутствует

Output example (picture):



cd_4_4_VIF (r_3_2_VIF)

- **Техническое название**: cd_4_4_VIF (r_3_2_VIF)
- **Описание**: Variance Inflation Factor. Коэффициент инфляции дисперсии (используется для обнаружения мультиколлинеарности)

- **Теги**: -
- **requirements**: -

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

См. раздел "Core package/Анализ данных", пункт 4.4

ВХОДНЫЕ ПАРАМЕТРЫ

-

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Отсутствует

Output example (picture): не применимо.

r_3_3_Spearman_Correlation

- **Техническое название**: r_3_3_Spearman_Correlation
- **Описание**: Spearman Correlations. Матрица попарных ранговых корреляций Спирмена и максимальное значение корреляции (%)
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

Отрисовываем heatmap с матрицей парных корреляций всех признаков из fields_to_test. Пропущенные поля исключаем из рассмотрения.

Корреляция Спирмена - непараметрическая мера статистической зависимости между двумя переменными. Она оценивает, насколько хорошо можно описать зависимость между переменными монотонной функцией.

При отсутствии повторяющихся значений коэфф Спирмена +1 или -1 свидетельствует о том, что одна переменная является строго монотонной функцией другой (зависимость не обязательно линейная).

Коэфф корреляции Спирмена менее чувствителен к выбросам, чем коэфф корреляции Пирсона.

Пример выставления signal bounds:
max(Corr) ≥ 50- красный,
40 ≤ max(Corr) < 50 - желтый,
max(Corr) < 40 - зеленый

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_columns: массив названий столбцов, по которым считаем тест (multi-column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

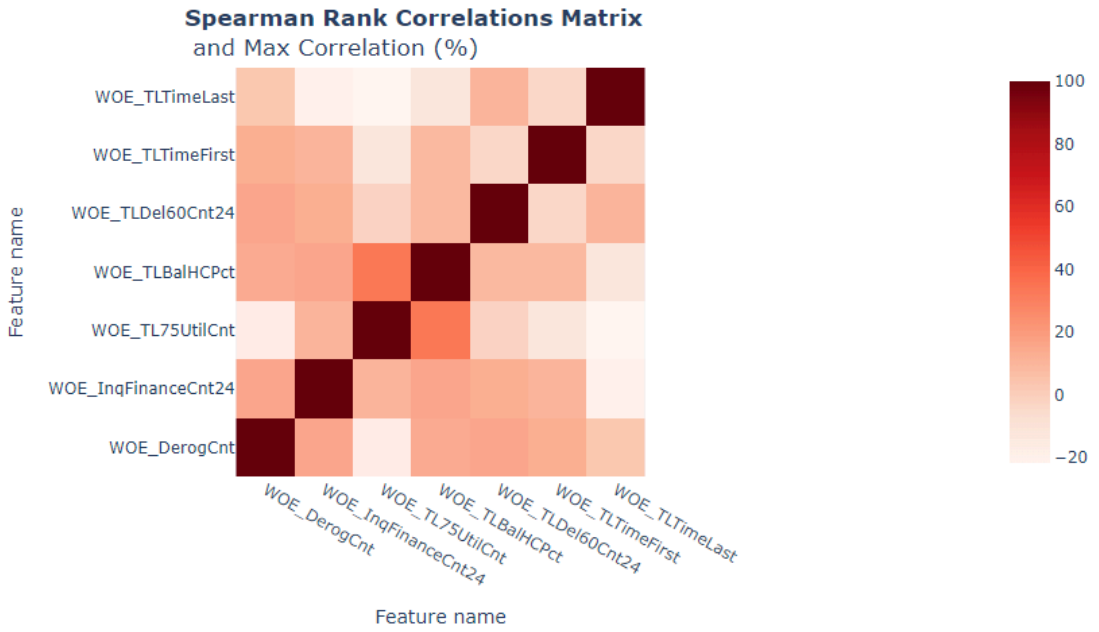
Heatmap: Матрица попарных корреляций выбранных столбцов

Output (short):

Число: максимальное из значений парной корреляции для выбранных столбцов, в %

светофор

Output example (picture):



Стабильность

r_4_1_Gini_diff_model

- **Техническое название**: r_4_1_Gini_diff_model
- **Описание**: Abs Gini Difference (%) train/test. Абсолютное значение изменения Gini Index модели (в %) на train/test. Показывает, насколько отличается предсказательная способность модели на разных выборках
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

abs([Gini index_test] - [Gini index_train])

Чем меньше значение разницы Gini. тем лучше.

Если Gini на тесте значительно меньше, чем на train, то произошло переобучение.

Пример выставления signal bounds:
\Δ Gini\ < 5% - зеленый,
5% ≤ \Δ Gini\ < 10% - желтый,
\Δ Gini\ ≥ 10% - красный

ВХОДНЫЕ ПАРАМЕТРЫ

df_train: датасет с данными для обучения (dataframe)

df_test: датасет с данными для теста (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора (! названия столбцов с target_column и predict_column должны совпадать в df_train и df_test)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

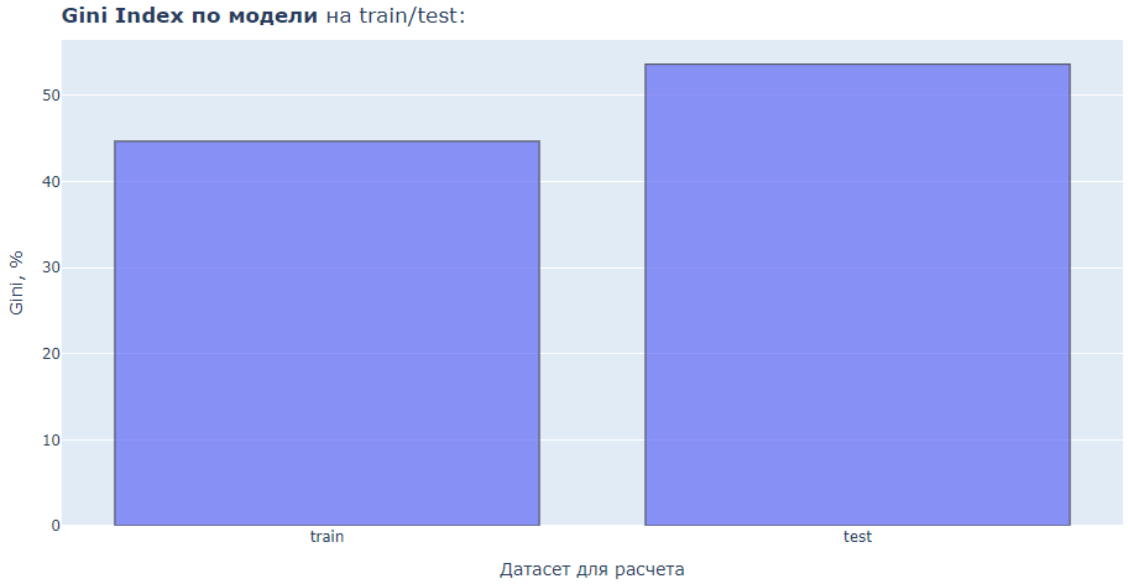
Output (long):

Barchart: 2 столбца - соответствуют выборкам train и test. Высота столбца - значение Gini по модели на указанной выборке.

Output (short):

Число: разница Gini на train и test в % (выводится в составе заголовка)

Output example (picture):



r_4_2_Gini_reltv_diff_model

- **Техническое название**: r_4_2_Gini_reltv_diff_model

- **Описание**: Gini Ind model - train/test reltv diff (%). Относительное изменение Gini index модели (в %) на train/test. Показывает насколько улучшилась/ухудшилась предсказательная способность модели на test по отношению к предсказательной способности на train
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

([Gini index_test] - [Gini index_train]) / [Gini index_train]

Чем меньше модуль показателя, тем лучше

Пример выставления signal bounds:
\Δ Gini\ < 10% - зеленый,
10% ≤ \Δ Gini\ < 20% - желтый,
\Δ Gini\ ≥ 20% - красный

ВХОДНЫЕ ПАРАМЕТРЫ

df_train: датасет с данными для обучения (dataframe)

df_test: датасет с данными для теста (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора (! названия столбцов с target_column и predict_column должны совпадать в df_train и df_test)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Число: относительное изменение Gini по модели, в %,

светофор

Output example (picture): не применимо.

> ***r_4_3_Gini_diff_features***

- **Техническое название**: r_4_3_Gini_diff_features
- **Описание**: Gini Ind features - train/test diff (%). Относительное изменение Gini index на train/test в разрезе отдельных факторов. Показывает, насколько улучшилась/ухудшилась информативность каждого из факторов на test по отношению к информативности этого фактора на train
- **Теги**: risk
- **requirements**: `typing`, `pandas`, `sklearn.metrics`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

В цикле для каждого категориального field из fields считаем: abs([Gini index_test] - [Gini index_train])

Если в признаке (field) или target присутствует пропущенное значение, такая строка исключается из рассмотрения.

Для разных признаков исключаются из рассмотрения разные строки.

Чем меньше модули показателей, тем лучше

Пример выставления signal bounds на относительное изменение индекса Джини по факторам между двумя соседними срезами данных:
\Δ Gini\ < 10% - зеленый,
10% ≤ \Δ Gini\ < 15% - желтый,
\Δ Gini\ ≥ 15% - красный

ВХОДНЫЕ ПАРАМЕТРЫ

df_train: датасет с данными для обучения (dataframe)

df_test: датасет с данными для теста (dataframe)

target_column: название столбца с таргетом (column)

field_columns: массив названий столбцов, по которым считаем тест (multi-column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора (! названия столбцов с target_column и field_columns должны совпадать в df_train и df_test)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

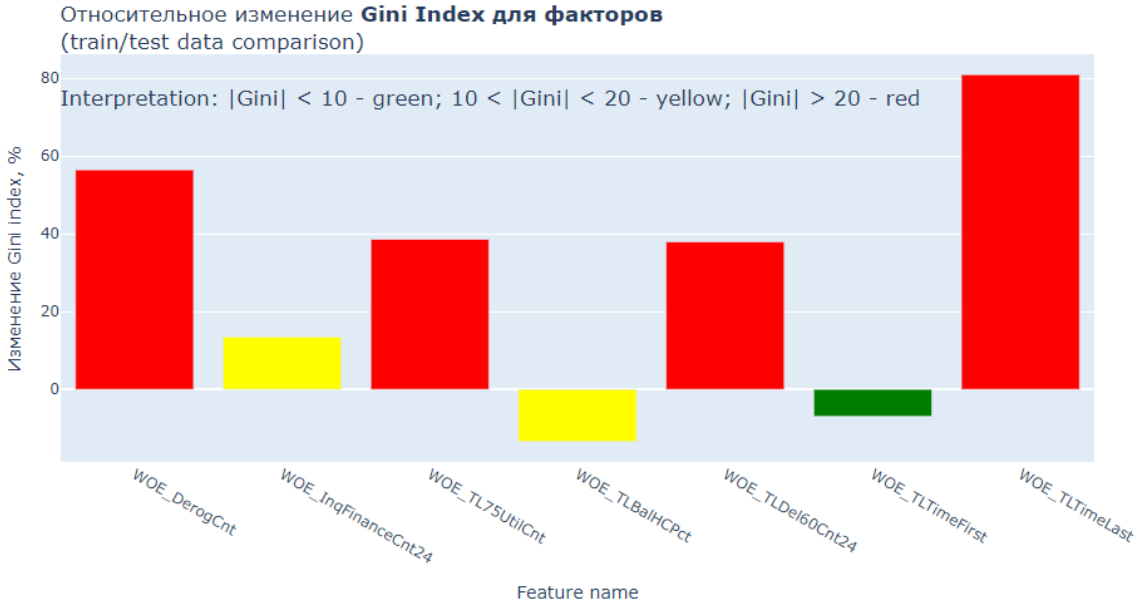
Barchart: Столбцы соответствуют признакам Высота столбца - отличие Gini для соответствующего признака на train и test.

светофор: столбцы окрашиваются в цвет светофора для соответствующего признака

Output (short):

Отсутствует

Output example (picture):



r_4_4_Gini_dynamic_model

- **Техническое название**: r_4_4_Gini_dynamic_model
- **Описание**: Gini Index (model) Dynamic. Динамика индекса Джини модели. Bar-chart со значениями Джини по модели в разные периоды времени
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`, `numpy`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

Наблюдения из df разбиваются на группы по значениям столбца с датой (report_dt). В одну группу попадают все наблюдения за period (месяцы, квартал или год).

Для наблюдений каждой группы рассчитывается индекс Джини по модели.

Чем больше Джини, тем лучше модель. Джини меньше 0 означает, что результаты модели хуже случайного угадывания. В динамике: чем стабильнее Джини, тем лучше.

Пример выставления signal bounds на абсолютное изменение индекса Джини по модели между двумя соседними срезами данных: \Δ Gini\ < 5% - зеленый,
5% ≤ \Δ Gini\ < 10% - желтый,
\Δ Gini\ ≥ 10% - красный

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

report_dt_column: название столбца с датой (column)

period: гранулярность данных
(dropdown - одно из значений типа str: 'month', 'quarter', 'year'; по умолчанию -'year')

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

Barchart: Столбцы соответствуют периодам Высота столбца - значение индекса Gini по модели, рассчитанного на наблюдениях за соответствующий период Точное значение Gini отображается при наведении мышки на соответствующий столбец

Output (short):

Число: максимальное изменение Gini

Output example (picture):

r_4_5_PSI_model

- **Техническое название**: r_4_5_PSI_model
- **Описание**: PSI Model. Population stability index (PSI) по модели
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `numpy`

Примечание:

Реализовано автоматическое разбиение score на заданное число бакетов. В большинстве моделей допустимо автоматическое разбиение на заданное число бакетов, без дополнительной шкалы.

Альтернативой может выступать разделение по мастер-шкале в отдельном столбце. Мастер-шкала составляется банком. В рисковых моделях обычно используют разбиение через мастер-шкалу.

ЛОГИКА ИСПОЛНЕНИЯ

Проводится проверка различия распределений предсказаний модели (score) на выборках для обучения и тестирования

1. Для каждого бакета (разбиение по значениям score) отдельно на train и test рассчитываем % наблюдений, принадлежащий этому бакету
2. temp_i = (%test - %train) * ln(%test / %train)
3. PSI = сумма temp_i по всем категориям

Корреляция с качеством модели - отрицательная. Чем ниже PSI, тем лучше модель

Пример выставления signal bounds: PSI < 0.1 - зеленый,
0.1 < PSI < 0.25 - желтый,
PSI > 0.25 - красный

ВХОДНЫЕ ПАРАМЕТРЫ

df_train: датасет с данными для обучения (dataframe)

df_test: датасет с данными для теста (dataframe)

predict_column: название столбца со скором (column)

nbuckets: число бакетов, на которое разбивается score
(int value; по умолчанию 30)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора (! названия столбца со predict_column должно совпадать в df_train и df_test)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: не применимо

Output (long): Отсутствует

Output (short):

Число: значение PSI по модели между train и test

светофор

Output example (picture): не применимо.

r_4_6_Lift_dynamic

- **Техническое название**: r_4_6_Lift_dynamic
- **Описание**: Lift Dynamic. Lift в динамике
- **Теги**: risk
- **requirements**: `typing`, `pandas`, `sklearn.metrics`, `numpy`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

Значения Lift по периодам

Группируем данные с гранулярностью period. Для каждой группы отрисовываем на Barcharte значения lift за каждый период:

- lift = [доля наблюдений с predict_field=1] / [доля наблюдений с target_field=1]
- lift = TPR/PR

В динамике: чем стабильнее Lift, тем лучше

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

report_dt_column: название столбца с датой (column)

period: гранулярность данных
(dropdown - одно из значений типа str: 'month', 'quarter', 'year'; по умолчанию -'year')

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

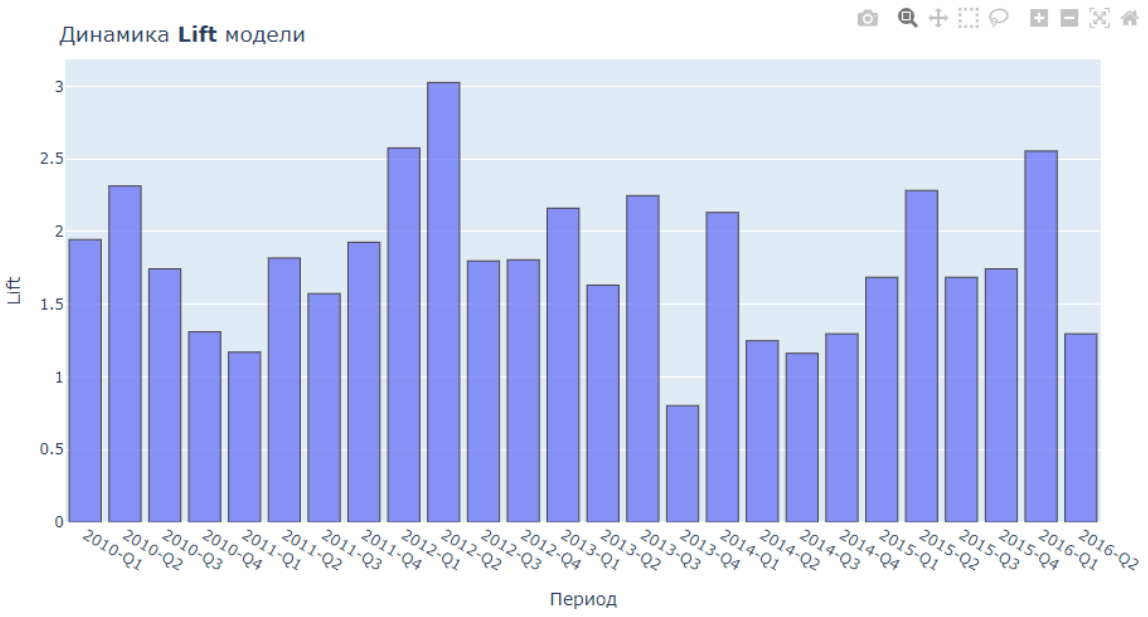
Output (long):

Barchart: Столбцы соответствуют периодам Высота столбца - значение Lift, рассчитанное на наблюдениях за соответствующий период Точное значение Lift отображается при наведении мышки на соответствующий столбец

Output (short):

Отсутствует

Output example (picture):



r_4_7_reltv_PR_Curve

- **Техническое название**: r_4_7_reltv_PR_Curve
- **Описание**: PR Curves train/test. Графики Precision-Reccall Curves (кривые точности-полноты) для train/test и относительное изменение PRC AUC
- **Теги**: risk, scalar
- **requirements**: `typing`, `pandas`, `sklearn.metrics`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

([PR AUC_test] - [PR AUC_train]) / [PR AUC_train]

Характеризует качество модели классификации. Чем больше (ближе к 1) значение PRC AUC, тем лучше модель. В динамике: чем стабильнее PRC AUC, тем лучше.

PRC полезна в задачах , где объектов класса 1 мало, но их необходимо выявлять.

ВХОДНЫЕ ПАРАМЕТРЫ

df_train: датасет с данными для обучения (dataframe)

df_test: датасет с данными для теста (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора (! названия столбцов с target_column и predict_column должны совпадать в df_train и df_test)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

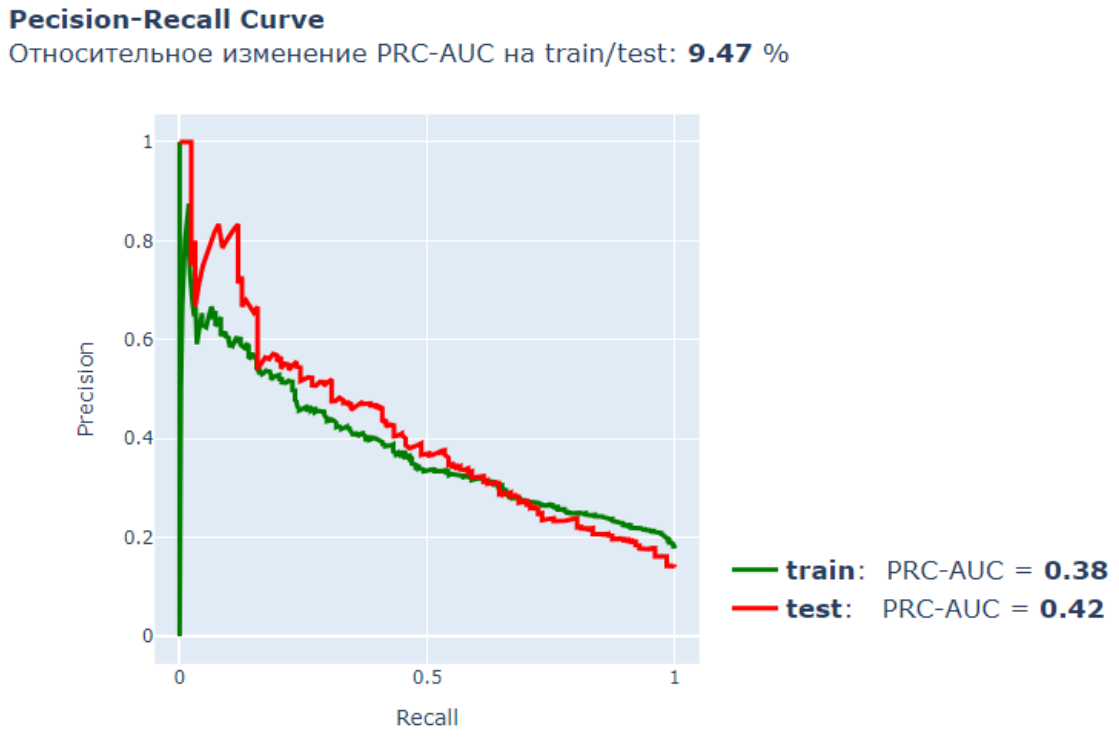
Output (long):

PR curves для обучающей и тестовой выборок

Output (short):

Число: относительное изменение PR AUC на train/test

Output example (picture):



Концентрация

r_5_1_HHI

- **Техническое название**: r_5_1_HHI
- **Описание**: Herfindahl-Hirschman Index (HHI). Индекс Херфиндаля-Хиршмана и Barchart с распределением наблюдений по разрядам рейтинговой шкалы
- **Теги**: risk, scalar

- **requirements**: `typing`, `pandas`

Примечание: -

ЛОГИКА ИСПОЛНЕНИЯ

Индекс Херфиндаля-Хиршмана описывает равномерность распределения по разрядам рейтинговой шкалы

1. Для i-го разряда рейтинговой шкалы рассчитываем N_i - количество наблюдений
2. $HI = \sum_i \left(\frac{N_i}{N}\right)^2$, где N - общее число наблюдений в датасете

Чем меньше индекс XX, тем лучше (тем равномернее распределение по группам)

Пример выставления signal bounds:
HHI < 20% - зеленый,
20% ≤ HHI < 30% - желтый,
HHI ≥ 30% - красный

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

scale_column: название столбца с присвоенным разрядом рейтинговой шкалы (column)

threshold_yellow: желтая граница светофора

threshold_red: красная граница светофора

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

Barchart: xaxis: Разряды рейтинговой шкалы yaxis:
Количество наблюдений, относящихся к соответствующему разряду

Output (short):

Число: значение индекса Херфиндаля-Хиршмана для выбранного столбца, в %

светофор

Output example (picture):

r_5_2_Unique_clients

- **Техническое название**: r_5_2_Unique_clients
- **Описание**: Unique Clients by Bins. Гистограммы распределения по разрядам рейтинговой шкалы для:
  - количества значений,
  - количества уникальных значений выбранного столбца (например, столбца с id клиентов)
- **Теги**: risk
- **requirements**: `typing`, `pandas`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

Отрисовываем комбинированный Barchart с:

- общим числом наблюдений в каждом бакете рейтинговой шкалы
- числом уникальных значений в каждом бакете рейтинговой шкалы

Пример применения метрики: исследование распределения уникальных клиентов по рейтинговой шкале (в переменную field подаем название столбца с id клиентов. Получим: всего клиентов по группам, уникальных клиентов по группам)

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

scale_column: название столбца с присвоенным разрядом рейтинговой шкалы (column)

field_column: название столбца, для которого проверяем распределение значений по разрядам (column)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

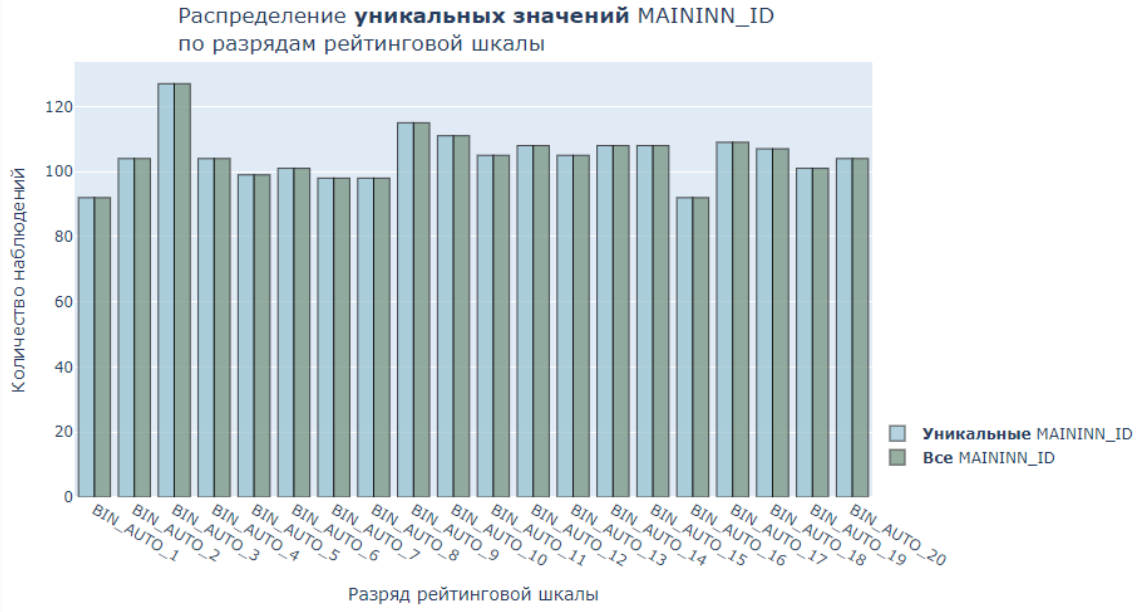
Output (long):

Barchart: xaxis: Пара столбцов соответствует разряду рейтинговой шкалы. yaxis: Высота столбцов: левого - число уникальных значений в группе, правого - общее число наблюдений в группе

Output (short):

Отсутствует

Output example (picture):



r_5_3_Derivative_Score_Distribution

- **Техническое название**: r_5_3_Derivative_Score_Distribution
- **Описание**: Derivative Score Distribution. Распределение производных оценок
- **Теги**: risk
- **requirements**: `typing`, `pandas`, `numpy`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

DSD обычно описывает распределение производных оценок, что может помочь в анализе чувствительности модели к изменениям входных данных.

Отрисовываем линейный график, представляющий собой график зависимости производной функции доли наблюдений от скора, начиная со значения, соответствующего перцентилю perc_threshold

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

predict_column: название столбца со скором (column)

perc_threshold: граница перцентиля для отсечения значений predict_column (float)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

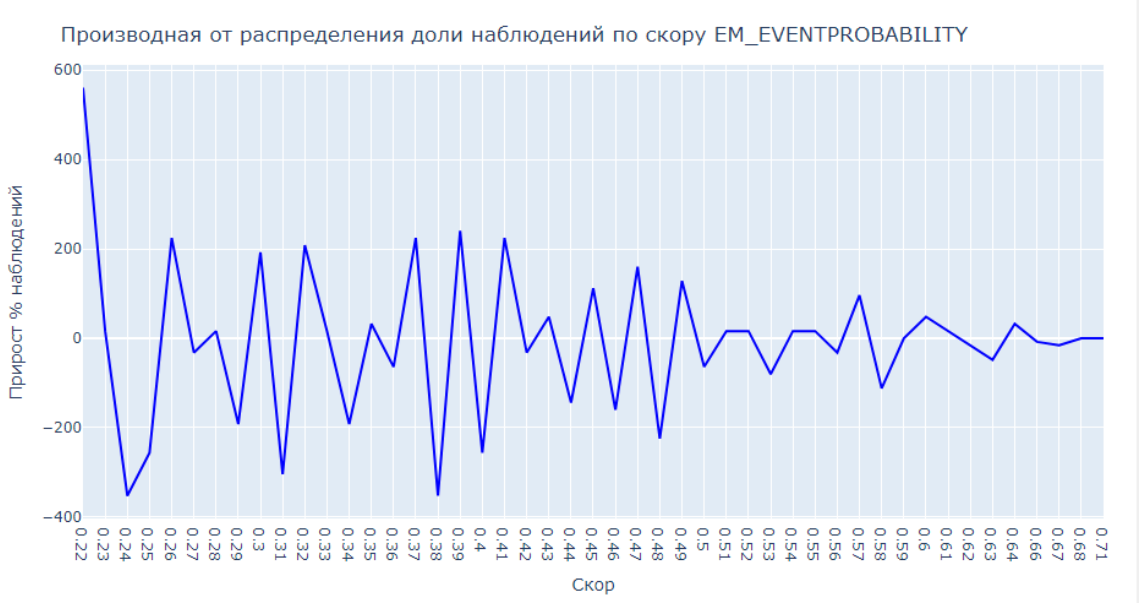
Output (long):

Линия: xaxis: Значения предсказаний модели yaxis: Прирост процента наблюдений значений предсказаний модели в датасете

Output (short):

Отсутствует

Output example (picture):



Калибровка

r_6_1_Model_default_rate

- **Техническое название**: r_6_1_Model_default_rate
- **Описание**: Model Default Rate. Модельный и фактический уровень дефолта по бакетам рейтинговой шкалы
- **Теги**: risk
- **requirements**: `typing`, `pandas`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

Прогнозный и фактический default rate находим как средние значения score и target по разрядам рейтинговой шкалы.

Отрисовываем линейные графики:

1. Среднее значение target_field по i-ому разряду рейтинговой шкалы
2. Среднее значение score_field по i-ому разряду рейтинговой шкалы

Чем ближе др. к др. графики (прогнозный и реальный default rate), тем лучше

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

scale_column: название столбца с присвоенным разрядом рейтинговой шкалы (column)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

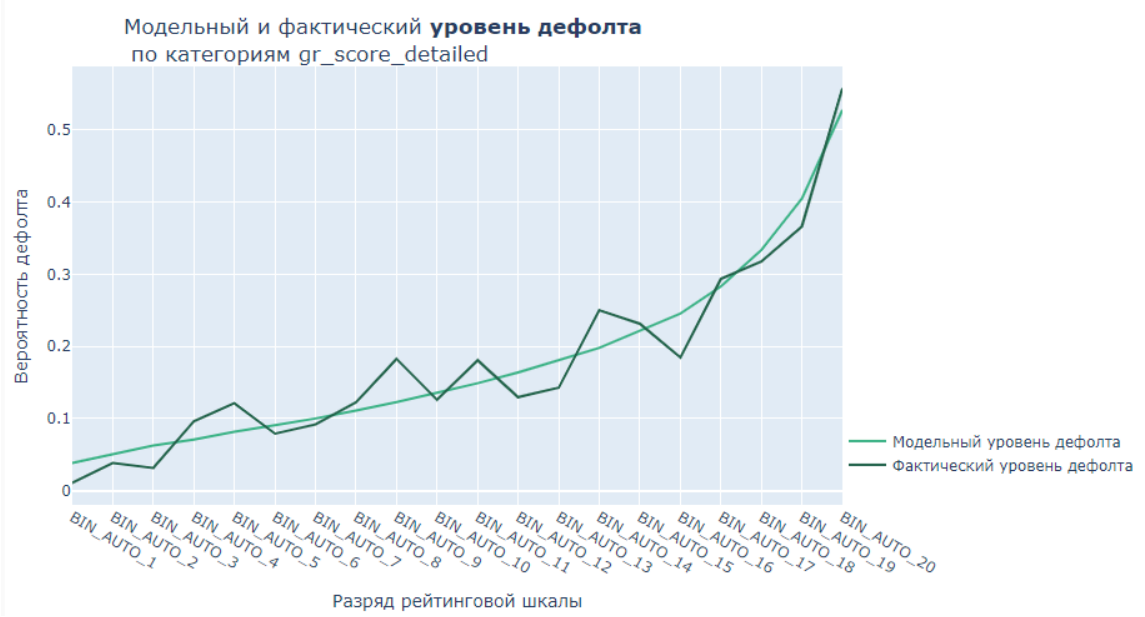
Output (long):

Массив графиков: xaxis: разряд рейтинговой шкалы yaxis: вероятность дефолта Графики соответствуют модельному и фактическому уровню дефолта

Output (short):

Отсутствует

Output example (picture):



r_6_2_Binomial_test

- **Техническое название**: r_6_2_Binomial_test
- **Описание**: Binomial Test. Биномиальный тест. Для каждой группы рейтинговой шкалы проверяется попадание фактического значения дефолта (среднего target по группе) в доверительный интервал для среднего значения предсказания (score) по группе
- **Теги**: risk
- **requirements**: `typing`, `pandas`, `scipy.stats`, `numpy`

Примечание:

-

ЛОГИКА ИСПОЛНЕНИЯ

1. Для i-го бакета рейтинговой шкалы на основе распределения score_field строим доверительный интервал по формуле:

$$PD_i - \varphi^{-1}(CI)\sqrt{PD_i * (1 - PD_i)/n_i}; PD_i + \varphi^{-1}(CI)\sqrt{PD_i * (1 - PD_i)/n_i}, \text{ где}$$

$\varphi^{-1}(CI)\sqrt{PD_i * (1 - PD_i)/n_i}$ – критическое значение уровня дефолтов, аппроксимированное с помощью нормального распределения,

φ^{-1} – обратная функция нормального распределения,

CI – уровень доверия,

\(PD_i\) - средний score_field по бакету

2. Для успешного прохождения теста средний target_field должен попадать в доверительный интервал по для каждого бакета рейтинговой шкалы

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

target_column: название столбца с таргетом (column)

predict_column: название столбца со скором (column)

scale_column: название столбца с присвоенным разрядом рейтинговой шкалы (column)

confidence_level: уровень доверия (float value; по умолчанию 0.99)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

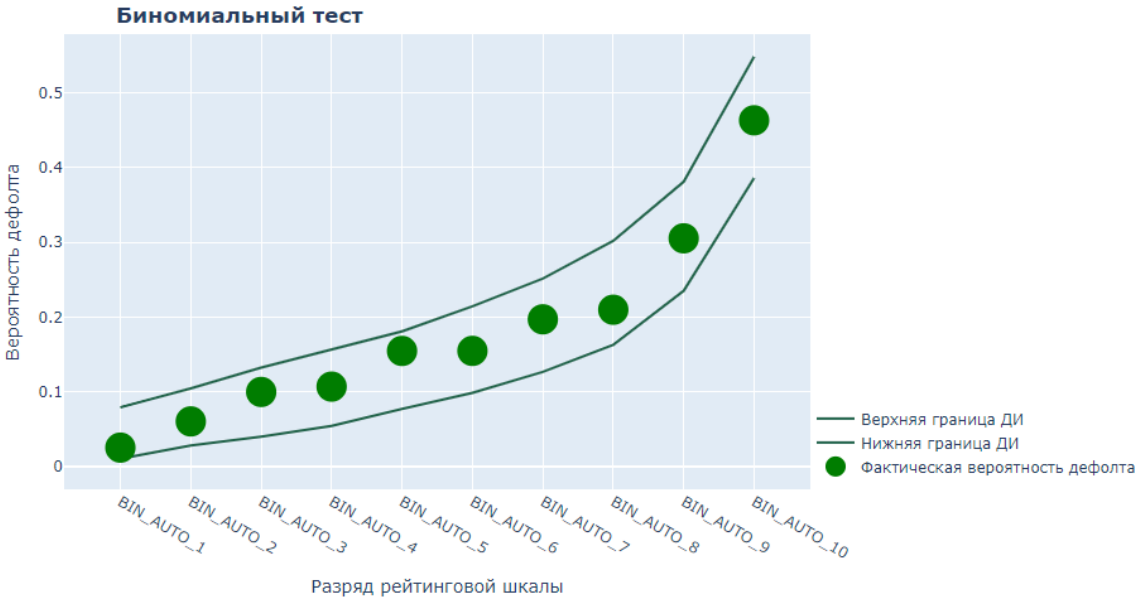
Output (long):

Массив графиков xaxis: разряд рейтинговой шкалы yaxis: вероятность дефолта

Output (short):

Отсутствует

Output example (picture):



Data drift package

dd_1_ADWIN

- **Техническое название**: dd_1_ADWIN
- **Описание**: Расчет Data Drift методом ADWIN (Adaptive Windowing)
- **Теги**: data_drift
- **requirements**: `typing`, `pandas`, `river`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Алгоритм выстраивает данные по временному ряду. Далее алгоритм проходит по данным оконным методом и вычисляет средние значения на этих окнах. Если обнаруживается значимое различие между близлежащими окнами - регистрируется событие дрейфа данных

Более подробно о методе ADWIN

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_column: название столбца для расчета (column)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

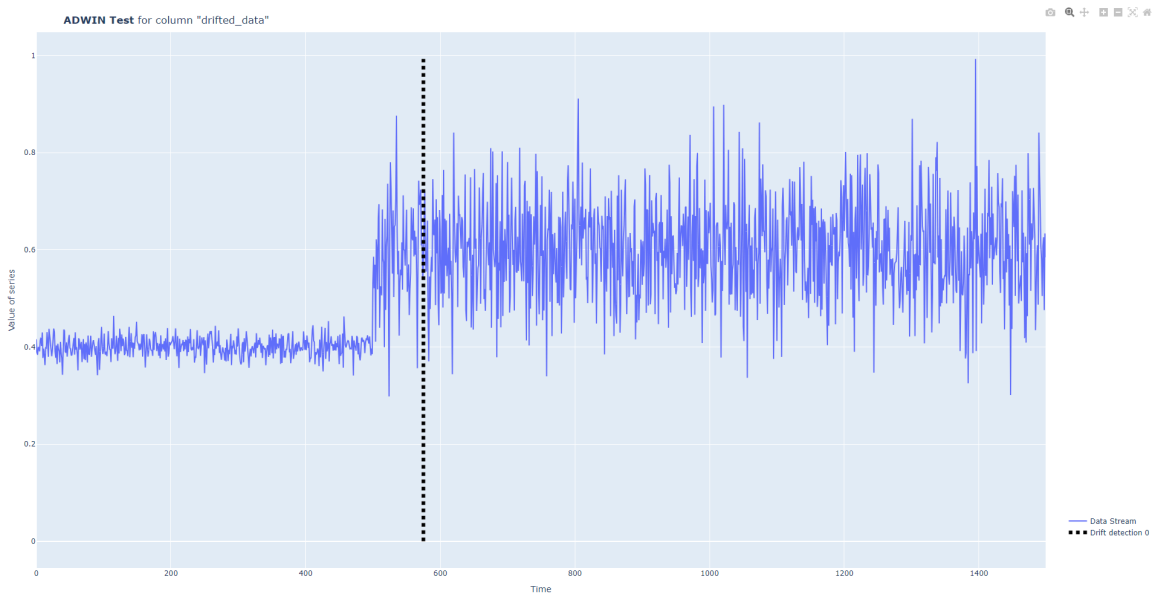
Массив графиков:

1. График значений столбца в виде линейного графика
2. Вертикальные границы, отмечающие события дрейфа данных, найденные алгоритмом

Output (short):

Отсутствует

Output example (picture):



dd_2_PageHinkleyTest

- **Техническое название**: dd_2_PageHinkleyTest
- **Описание**: Расчет Data Drift методом Page-Hinkley
- **Теги**: data_drift
- **requirements**: `typing`, `pandas`, `river`
- **Примечания**: -

ЛОГИКА ИСПОЛНЕНИЯ

Базово алгоритм похож на ADWIN (п. 1)

Более подробно о методе Page Hinkley Test

ВХОДНЫЕ ПАРАМЕТРЫ

df: датасет с данными для исследования (dataframe)

field_column: название столбца для расчета (column)

РЕЗУЛЬТАТЫ

Движок отрисовки графика: plotly.js

Output (long):

Массив графиков:

1. График значений столбца в виде линейного графика
2. Вертикальные границы, отмечающие события дрейфа данных, найденные алгоритмом

Output (short):

Отсутствует

Output example (picture):