Матвеева Валерия Владимировна

Описание решения кейса GPN CUP 2023: направление Data Science

1. Предобработка исходных данных

1.1 Данные о транзакциях

Исходные данные о транзакциях приведены в формате почасовых продаж. Т.к. в дальнейшем мы будем предсказывать на дни вперёд, изменим формат дат. Изменённый датафрейм:

	product	price	amount	place	date
0	Целебные травы	3.90	1.242125	Анор Лондо	2216-01-02
1	Целебные травы	3.90	-0.079689	Анор Лондо	2216-01-02
2	Целебные травы	3.90	0.882450	Анор Лондо	2216-01-02
3	Целебные травы	3.90	0.621377	Анор Лондо	2216-01-02
4	Целебные травы	3.90	1.367161	Анор Лондо	2216-01-02
...	...	...	...	...	...
875031	Эстус	9.28	1.079104	Фалькония	2218-09-27
875032	Эстус	9.28	1.580617	Фалькония	2218-09-27
875033	Эстус	9.28	1.738492	Фалькония	2218-09-27
875034	Эстус	9.28	0.430340	Фалькония	2218-09-27
875035	Эстус	9.28	0.090248	Фалькония	2218-09-27

1) Обработка пропусков

Краткая информация о датафрейме свидетельствует о том, что в нем имеются пропуски в столбце с городами:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 875036 entries, 0 to 875035
Data columns (total 5 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   product  875036 non-null  object
 1   price    875036 non-null  float64
 2   amount   875036 non-null  float64
 3   place    874604 non-null  object
 4   date     875036 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(2), object(2)
memory usage: 33.4+ MB
```

Исследуем его подробнее. Небольшая часть с пропуском:

	product	price	amount	place	date
1410	Целебные травы	-898.04	0.419002	Анор Лондо	2216-01-10
1411	Целебные травы	3.90	0.834557	None	2216-01-10
1412	Целебные травы	-7999.82	1.824032	Анор Лондо	2216-01-10
1413	Целебные травы	-9054.33	0.339604	Анор Лондо	2216-01-10
1414	Целебные травы	3.90	-0.023644	Анор Лондо	2216-01-10

После исследования становится понятно, что, вероятно, пропуски образовались в процессе ошибки устройства, которое не зафиксировало

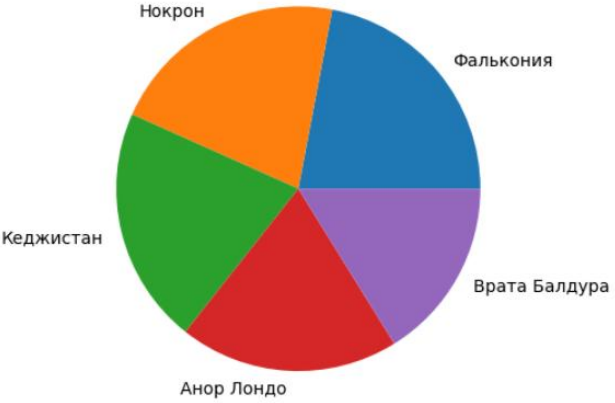
название города, но зафиксировало все остальное. Т.к. данные отсортированные (по городу и продукту и по всем датам по порядку, пропуски «окаймляются» одним и тем же городом), можно заполнить пропуски тем городом, который встретился перед ним.

На всякий случай можно посмотреть распределение городов в датафрейме: сколько каждый город встречался в процентном соотношении. Заодно выводится и количество пропусков.

```
place
Фалькония        22.002180
Нокрон           21.293524
Кеджистан        21.023021
Анор Лондо       19.472342
Врата Балдура    16.159564
None              0.049369
Name: proportion, dtype: float64
```

После заполнения пропусков предшествующим значением сильно распределение не изменилось, заполнения примерно равномерно распределились между городами.

```
place
Фалькония        22.013380
Нокрон           21.303923
Кеджистан        21.034677
Анор Лондо       19.481598
Врата Балдура    16.166421
Name: proportion, dtype: float64
```



2) Аномальные значения и ошибки

Далее рассмотрим количественные данные: цену и продажи. Описательные статистики соответствующих столбцов:

	count	mean	std	min	25%	50%	75%	max
price	875036.0	-299.886482	1407.438729	-9998.99	6.850000	9.730000	13.800000	31.200000
amount	875036.0	0.739259	0.717872	-0.50	0.119508	0.733257	1.356674	1.999999

Можно увидеть, что присутствуют аномальные значения: минимумы цены и продаж - отрицательные значения. И это наблюдается в 25% данных, 6% из которых в ценах, остальные 19% в продажах.

Сначала обработаем цены. Отдельно описательные статистики для цен:

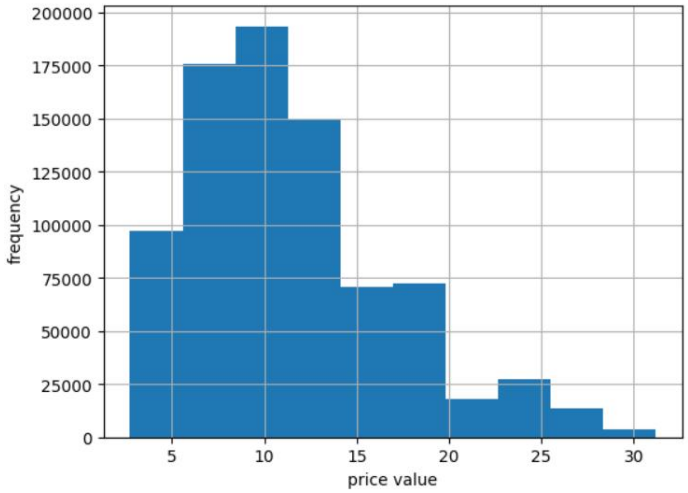
	count	mean	std	min	25%	50%	75%	max
price	875036.0	-299.886482	1407.438729	-9998.99	6.85	9.73	13.8	31.2

По ценам наблюдаются очень большие отклонения. Исследуем поподробнее, часть датафрейма с отрицательными ценами:

	product	price	amount	place	date
1410	Целебные травы	-898.04	0.419002	Анор Лондо	2216-01-10
1411	Целебные травы	3.90	0.834557	Анор Лондо	2216-01-10
1412	Целебные травы	-7999.82	1.824032	Анор Лондо	2216-01-10
1413	Целебные травы	-9054.33	0.339604	Анор Лондо	2216-01-10
1414	Целебные травы	3.90	-0.023644	Анор Лондо	2216-01-10

Это аномальные значения, но можно заметить, что это ошибки - они встречаются между которые положительными значениями цен, наблюдаемыми в течение дня. Возможно, произошла ошибка фиксации и автомат ввёл аномальное значение. Таким образом, ошибки в ценах можно заменить на интерполированные значения, перед этим заменив отрицательные значения на nan’ы.

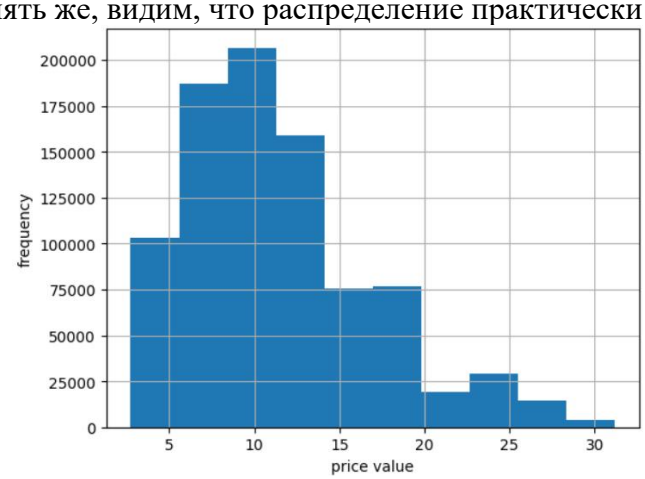
На всякий случай посмотрим, как распределены положительные цены, похоже на логнормальное или гамма-распределение:



Часть датафрейма с исправленными ошибками:

	product	price	amount	place	date
1410	Целебные травы	3.9	0.419002	Анор Лондо	2216-01-10
1411	Целебные травы	3.9	0.834557	Анор Лондо	2216-01-10
1412	Целебные травы	3.9	1.824032	Анор Лондо	2216-01-10
1413	Целебные травы	3.9	0.339604	Анор Лондо	2216-01-10
1414	Целебные травы	3.9	-0.023644	Анор Лондо	2216-01-10

Опять же, видим, что распределение практически не изменилось:



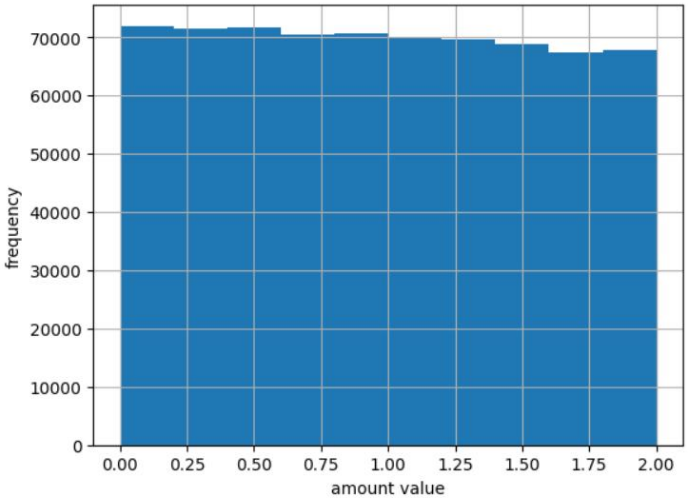
Теперь обработаем продажи. Описательные статистики для продаж:

	count	mean	std	min	25%	50%	75%	max
amount	875036.0	0.739259	0.717872	-0.5	0.119508	0.733257	1.356674	1.999999

По продажам отклонение меньше, и данные будто подходят нам по значениям, если бы мы брали абсолютное значение. Но это, скорее всего, догадки. Вероятнее, в этот момент сломался аппарат, фиксирующий продажи, и подал неправильно значение, поэтому заменим отрицательные значения на 0, т.к. в дальнейшем мы будем агрегировать данные по дням.

	product	price	amount	place	date
30	Целебные травы	3.9	0.959618	Анор Лондо	2216-01-02
31	Целебные травы	3.9	0.258301	Анор Лондо	2216-01-02
32	Целебные травы	3.9	1.396313	Анор Лондо	2216-01-02
33	Целебные травы	3.9	-0.379642	Анор Лондо	2216-01-02
34	Целебные травы	3.9	-0.424848	Анор Лондо	2216-01-02
35	Целебные травы	3.9	1.115369	Анор Лондо	2216-01-02

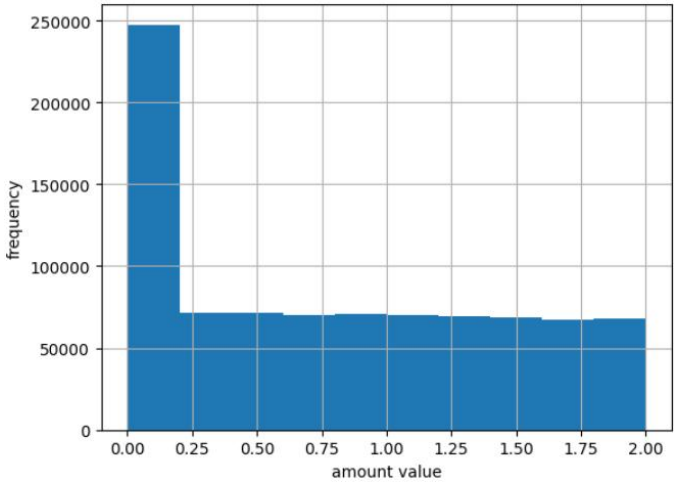
Посмотрим на распределение значений продаж, оно равномерное:



Часть датафрейма с исправлениями:

	product	price	amount	place	date
30	Целебные травы	3.9	0.959618	Анор Лондо	2216-01-02
31	Целебные травы	3.9	0.258301	Анор Лондо	2216-01-02
32	Целебные травы	3.9	1.396313	Анор Лондо	2216-01-02
33	Целебные травы	3.9	0.000000	Анор Лондо	2216-01-02
34	Целебные травы	3.9	0.000000	Анор Лондо	2216-01-02
35	Целебные травы	3.9	1.115369	Анор Лондо	2216-01-02

Распределение, конечно же, поменяется, ведь мы, по сути, добавили нулей. Но при агрегировании по дням эти нули уйдут.



Итоговые описательные статистики без отрицательных значений выглядят следующим образом:

	price	amount
count	875036.000000	875036.000000
mean	11.409971	0.789397
std	5.403449	0.649595
min	2.730000	0.000000
25%	7.490000	0.119508
50%	10.010000	0.733257
75%	14.140000	1.356674
max	31.200000	1.999999

Изменились отклонения: теперь они в пределах разумного, разброс значений не очень большой, минимальные значения положительны.

1.2 Данные о конкурентах

Исходный датафрейм:

	place	product	competitor	price_c	date
0	Анор Лондо	Целебные травы	Артериас&Со	3.78	2216-01-04
1	Анор Лондо	Целебные травы	Артериас&Со	3.78	2216-01-05
2	Анор Лондо	Целебные травы	Артериас&Со	3.78	2216-01-06
3	Анор Лондо	Целебные травы	Артериас&Со	3.78	2216-01-09
5	Анор Лондо	Целебные травы	Артериас&Со	3.78	2216-01-11
...	...	...	...	...	...
39451	Фалькония	Эстус	Светлые Души	14.99	2218-09-17
39453	Фалькония	Эстус	Светлые Души	15.36	2218-09-20
39454	Фалькония	Эстус	Светлые Души	15.36	2218-09-22
39455	Фалькония	Эстус	Светлые Души	15.36	2218-09-23
39456	Фалькония	Эстус	Светлые Души	15.36	2218-09-24

Информация о датафрейме:

```
<class 'pandas.core.frame.DataFrame'>
Index: 31799 entries, 0 to 39456
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   place       31799 non-null  object
 1   product     31799 non-null  object
 2   competitor  31799 non-null  object
 3   price_c     31799 non-null  float64
 4   date        31799 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(1), object(3)
memory usage: 1.5+ MB
```

Описательные статистики:

	count	mean	std	min	25%	50%	75%	max
price_c	31799.0	11.89481	5.574935	2.85	7.78	11.05	15.03	31.94

Таблица с конкурентами и их ценами не содержит пропусков и аномальных значений.

1.3 Данные о погоде

Исходный датафрейм:

	place	hot	rain	snow	date
0	Фалькония	0	0	0	2216-01-02
1	Анор Лондо	1	0	0	2216-01-02
2	Врата Балдура	0	0	1	2216-01-02
3	Нокрон	0	0	0	2216-01-02
4	Кеджистан	0	0	1	2216-01-02
...	...	...	...	...	...
4995	Фалькония	0	0	0	2218-09-27
4996	Анор Лондо	0	0	1	2218-09-27
4997	Врата Балдура	1	0	0	2218-09-27
4998	Нокрон	0	0	0	2218-09-27
4999	Кеджистан	0	0	0	2218-09-27

Данные предоставлены в формате бинарных признаков, отвечающих на вопрос, была ли в данный день жара, дождь или снег.

Информация о датафрейме:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   place   5000 non-null   object
 1   hot     5000 non-null   int64
 2   rain    5000 non-null   int64
 3   snow    5000 non-null   int64
 4   date    5000 non-null   datetime64[ns]
dtypes: datetime64[ns](1), int64(3), object(1)
memory usage: 195.4+ KB
```

Таблица с погодой и их ценами также не содержит пропусков и аномальных значений.

Проверено, что в каждый из дней была либо жара, либо дождь, либо снег. Нет дней, в которые были погодные изменения.

```
weather_df[(weather_df.snow == 1) & (weather_df.rain == 1)]
```

place	hot	rain	snow	date

```
weather_df[(weather_df.hot == 1) & (weather_df.rain == 1)]
```

place	hot	rain	snow	date

```
weather_df[(weather_df.hot == 1) & (weather_df.snow == 1)]
```

place	hot	rain	snow	date

1.4 Данные о себестоимостях

Исходный датафрейм:

	place	product	cost	date
0	Анор Лондо	Целебные травы	2.07	2216-01-02
1	Анор Лондо	Целебные травы	3.07	2216-01-26
2	Анор Лондо	Целебные травы	3.30	2216-02-12
3	Анор Лондо	Целебные травы	4.04	2216-03-06
4	Анор Лондо	Целебные травы	3.88	2216-03-27
...	...	...	...	...
1112	Фалькония	Эстус	3.10	2218-07-29
1113	Фалькония	Эстус	7.20	2218-08-10
1114	Фалькония	Эстус	3.31	2218-08-27
1115	Фалькония	Эстус	3.26	2218-09-16
1116	Фалькония	Эстус	7.29	2218-09-24

Информация о датафрейме:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1117 entries, 0 to 1116
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   place    1117 non-null   object
 1   product  1117 non-null   object
 2   cost     1117 non-null   float64
 3   date     1117 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 35.0+ KB
```

Описательные статистики:

	count	mean	std	min	25%	50%	75%	max
cost	1117.0	6.859955	3.856876	1.18	3.91	5.93	8.86	24.52

Таблица с себестоимостями не содержит пропусков и аномальных значений, в чем можно убедиться опять же, проанализировав датафрейм и статистики количественных признаков.Отклонение в пределах нормального.

2. Объединение и группировка таблиц

Перед тем, как брать временные ряды, анализировать их компоненты и прогнозировать их, необходимо объединить данные для сопоставления и сгруппировать по городу, продукту и дням.

2.1 Объединение таблиц с транзакциями и с ценами конкурентов

В некоторые дни нам неизвестны данные о конкурентах и их ценах, поэтому при объединении таблиц появляются пропуски. Их можно заполнить из следующих соображений: возможно, в этот день у нас действительно не было конкурентов, но если они были, и мы этого не учтём (допустим, нам просто не удалось собрать информацию по конкурентам за день) - потеряем полезную информацию. Поэтому мы будем считать, что цена конкурентов в день, в который по каким-то причинам у нас не было информации о ценах конкурентов, примерно такая же, что и в смежные дни. Т.е. мы будем интерполировать цены конкурентов. Можно было взять минимальную цену в данном городе на данный продукт, либо медианную или среднюю, либо взять последнюю замеченную (что схоже с вариантом, который выбрали мы). В зависимости от задачи и предпочтений.

Еще одно замечание: нам не важно какие конкуренты у нас в тот или иной день. Нам важно какая на рынке средняя цена у конкурента: больше нашей или меньше? Нам не важно к какому из конкурентов пойдут, нам просто важно, чтобы шли не к ним - а к нам. Так что в целом нам просто надо знать ситуацию на рынке. По этой причине в качестве цен конкурентов (чтобы уменьшить уровни группировки и размеры таблиц после агрегации) были взяты средние цены. Можно, опять же, взять минимальные, медианные и т.д. - в зависимости от задачи можно брать разную статистику.

Итоговый объединённый и сгруппированный датафрейм на шаге 1 выглядит следующим образом:

			amount	price	price_c
place	product	date			
Анор Лондо	Целебные травы	2216-01-02	41.494198	3.90	4.210000
		2216-01-03	46.984980	3.90	4.046667
		2216-01-04	39.587422	3.90	3.980000
		2216-01-05	39.769653	3.90	3.903333
		2216-01-06	49.805077	3.90	3.980000
...	...	...	...	...	...
Фалькония	Эстус	2218-09-23	59.140366	9.28	14.690000
		2218-09-24	51.237969	9.28	15.016667
		2218-09-25	61.007146	9.28	14.756667
		2218-09-26	57.613643	9.28	14.845000
		2218-09-27	50.422484	9.28	20.940000

2.2 Добавление погоды

Следующий шаг - добавление в датафрейм данных о погоде, т.к. она тоже нужна для дальнейшего понимания, как она влияет на продажи, и прогнозирования продаж. В датафрейме по погоде 5000 строк, т.е. 1000 строк для каждого города (в чем тоже можно убедиться), поэтому пропусков при объединении не возникает.

Итоговый объединённый и сгруппированный датафрейм на шаге 2 выглядит следующим образом:

			amount	price	price_c	hot	rain	snow
place	product	date						
Анор Лондо	Целебные травы	2216-01-02	41.494198	3.90	4.210000	1	0	0
		2216-01-03	46.984980	3.90	4.046667	1	0	0
		2216-01-04	39.587422	3.90	3.980000	0	0	0
		2216-01-05	39.769653	3.90	3.903333	0	0	0
		2216-01-06	49.805077	3.90	3.980000	0	0	1
...	...	...	...	...	...	...	...	...
Фалькония	Эстус	2218-09-23	59.140366	9.28	14.690000	0	1	0
		2218-09-24	51.237969	9.28	15.016667	0	0	0
		2218-09-25	61.007146	9.28	14.756667	0	0	1
		2218-09-26	57.613643	9.28	14.845000	0	0	1
		2218-09-27	50.422484	9.28	20.940000	0	0	0

2.3 Добавление себестоимостей

Насчёт себестоимостей нельзя сказать того же самого, что и про погоду - в датафрейме себестоимостей всего 1117 строк. Цена производства зависит от города, продукта и дня производства.

Мы будем считать, что если продукт $product$ в городе $place$ был произведён в день t по цене p, то он был доставлен целой партией в этот день t в магазин и дальше уже продавался там. Т.к. мы не знаем запасы, которые у нас лежат, будем считать, что перевозки организованы оптимально, т.е. поставки совершаются в дни, когда товар у нас

закончился, все распродано. Другими словами, себестоимость при продаже одной партии продукта в течение длительного времени не меняется: нам поставляют партию, произведённую по некоторой цене, а мы дальше её продаём, до следующей поставки партии, произведённой уже по какой-то другой цене.

Таким образом, образовавшиеся пропуски при объединении можно заменить на последнее замеченное значение себестоимости.

Итоговый объединённый и сгруппированный датафрейм на последнем шаге выглядит следующим образом:

			amount	price	price_c	hot	rain	snow	cost
place	**product**	**date**							
Анор Лондо	**Целебные травы**	**2216-01-02**	41.494198	3.90	4.210000	1	0	0	2.07
		2216-01-03	46.984980	3.90	4.046667	1	0	0	2.07
		2216-01-04	39.587422	3.90	3.980000	0	0	0	2.07
		2216-01-05	39.769653	3.90	3.903333	0	0	0	2.07
		2216-01-06	49.805077	3.90	3.980000	0	0	1	2.07
...	**...**	**...**	...	...	...	...	...	...	...
Фалькония	**Эстус**	**2218-09-23**	59.140366	9.28	14.690000	0	1	0	3.26
		2218-09-24	51.237969	9.28	15.016667	0	0	0	7.29
		2218-09-25	61.007146	9.28	14.756667	0	0	1	7.29
		2218-09-26	57.613643	9.28	14.845000	0	0	1	7.29
		2218-09-27	50.422484	9.28	20.940000	0	0	0	7.29

Таким образом, итоговый датафрейм содержит данные о:

1. продажах;
2. цене, по которой они были проданы;
3. цене, предлагаемой конкурентами;
4. погодных условиях;
5. себестоимостях.

Данные предоставлены в каждом из городов по каждому продукту в каждый из дней.

Замечание: было обнаружено, что в данном датафрейме транзакций нет данных по продажам в несколько дней в одном из городов (Врата Балдура) по одному из продуктов(Эльфийская пыльца) (пропущены даты 2217-04-03 и 2217-07-01), что тоже следует обработать, т.к. в дальнейшем анализе временных рядов это может привести к возможным ошибкам.

	place	product	amount	price	price_c	hot	rain	snow	cost
date									
2217-04-02	Врата Балдура	Эльфийская пыльца	17.897556	20.820000	18.45	0	1	0	9.55
2217-04-04	Врата Балдура	Эльфийская пыльца	13.293795	20.820000	19.66	0	1	0	9.55
2217-04-05	Врата Балдура	Эльфийская пыльца	18.992042	20.820000	19.98	0	0	1	9.55

2217-06-30	Врата Балдура	Эльфийская пыльца	5.748985	22.080000	21.11	0	0	0	7.69
2217-07-02	Врата Балдура	Эльфийская пыльца	18.715556	22.172759	19.58	0	0	1	7.69
2217-07-03	Врата Балдура	Эльфийская пыльца	12.729953	22.350000	21.11	0	1	0	7.69

Возникшие пропуски можно заменить следующим образом: изначально в целом добавить строку с пропущенной датой в датафрейм, продажи поставить нулевые, цену и себестоимость интерполировать ценами и себестоимостями в соседние дни, цену конкурента можно заполнить исходя из датафрема *competitors_df*, взяв в нем данные по нужному городу, продукту и дате и посчитать среднюю цену конкурентов, погоду мы также можем узнать из датафрейма *weather_df* по нужному пропущенному дню.

В итоге получим ряд без пропусков:

	place	product	amount	price	price_c	hot	rain	snow	cost
date									
2217-06-30	Врата Балдура	Эльфийская пыльца	5.748985	22.080000	21.110	0.0	0.0	0.0	7.69
2217-07-01	Врата Балдура	Эльфийская пыльца	0.000000	22.126379	20.305	0.0	0.0	0.0	7.69
2217-07-02	Врата Балдура	Эльфийская пыльца	18.715556	22.172759	19.580	0.0	0.0	1.0	7.69

	place	product	amount	price	price_c	hot	rain	snow	cost
date									
2217-04-02	Врата Балдура	Эльфийская пыльца	17.897556	20.82	18.450	0.0	1.0	0.0	9.55
2217-04-03	Врата Балдура	Эльфийская пыльца	0.000000	20.82	18.745	0.0	0.0	0.0	9.55
2217-04-04	Врата Балдура	Эльфийская пыльца	13.293795	20.82	19.660	0.0	1.0	0.0	9.55

Сразу для дальнейшего прогноза добавим в датафрейм строки для последних 90 дней для каждого города и продукта. Погоду спрогнозировать в формате бинарных переменных не так просто, поэтому мы будем считать, что в эти месяцы погода будет такая же, как в прошлом году в эти же месяцы, что более-менее логично.

Итоговый датафрейм для прогноза:

	date	place	product	amount	price	price_c	hot	rain	snow	cost
0	2216-01-02	Анор Лондо	Целебные травы	41.494198	3.90	4.210000	1.0	0.0	0.0	2.07
1	2216-01-03	Анор Лондо	Целебные травы	46.984980	3.90	4.046667	1.0	0.0	0.0	2.07
2	2216-01-04	Анор Лондо	Целебные травы	39.587422	3.90	3.980000	0.0	0.0	0.0	2.07
3	2216-01-05	Анор Лондо	Целебные травы	39.769653	3.90	3.903333	0.0	0.0	0.0	2.07
4	2216-01-06	Анор Лондо	Целебные травы	49.805077	3.90	3.980000	0.0	0.0	1.0	2.07
...	...	...	...	...	...	...	...	...	...	...
16345	2218-12-22	Фалькония	Эстус	NaN	9.28	NaN	0.0	0.0	0.0	NaN
16346	2218-12-23	Фалькония	Эстус	NaN	9.28	NaN	0.0	0.0	0.0	NaN
16347	2218-12-24	Фалькония	Эстус	NaN	9.28	NaN	0.0	0.0	0.0	NaN
16348	2218-12-25	Фалькония	Эстус	NaN	9.28	NaN	0.0	0.0	1.0	NaN
16349	2218-12-26	Фалькония	Эстус	NaN	9.28	NaN	0.0	0.0	0.0	NaN

3. Прогноз спроса и себестоимости

Цены зависят от себестоимости и спроса. Соответственно, чтобы оптимизировать цены нам нужно знать, какими примерно будут себестоимость и спрос в 90 дней.

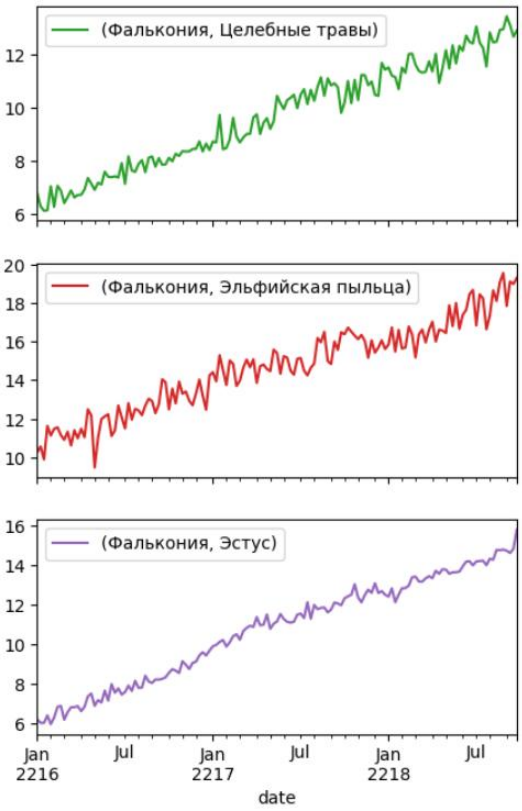
Для того, чтобы спрогнозировать спрос, нам необходимы факторы, от которых он зависит. В данном случае это цена конкурента и погода. Погоду мы уже «спрогнозировали», осталась цена конкурента.

4.1 Анализ рядов цены конкурента, себестоимостей и построение прогнозов

Необходимо анализировать цены конкурента и себестоимости в каждом городе по каждому продукту отдельно, т.к. в дальнейшем мы будем строить прогноз продаж в каждом городе на каждый продукт по отдельности, для чего и создавался сгруппированный датафрейм.

1) Ряды цен конкурентов

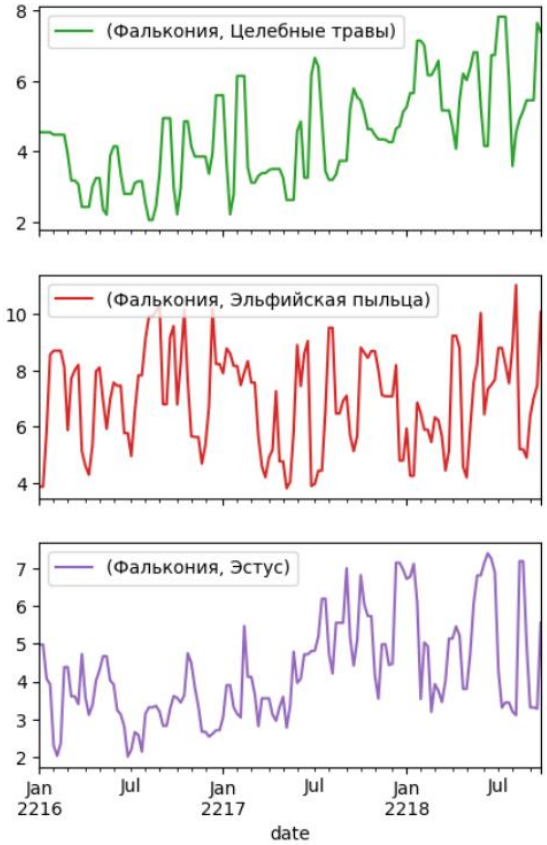
Исходные графики цен конкурентов имеют очень сильные колебания и похожи на шум, т.к. данные каждодневные. В них вообще трудно что-то выявить кроме тренда, поэтому можно агрегировать данные цены конкурентов по неделям (взять данные за всю неделю и усреднить), построить графики по усредненным данным и попробовать выявить компоненты в них. Несколько таких графиков:



Посмотрев на график средних цен за неделю, можно увидеть возрастающий, скорее всего линейный тренд. Всё-таки довольно сложно уловить сезонность, но можно предположить, что есть недельная сезонность (исходя из спроса и логики формирования цен), а также годовая сезонность (и, возможно, квартальная, но её мы не будем включать в дальнейшее построение моделей и прогнозов). Сезонность на некоторых графиках аддитивная, а на некоторых - мультипликативная, можно заметить как амплитуда колебаний увеличивается с течением времени.

2) Ряды себестоимостей

Данные по себестоимостям тоже можно агрегировать по неделям для удобства. Несколько графиков усреднённых по неделям себестоимостей:



Можно сделать примерно такие же выводы, что и по ценам конкурента. Тренд тоже присутствует, но более слабо выраженный.

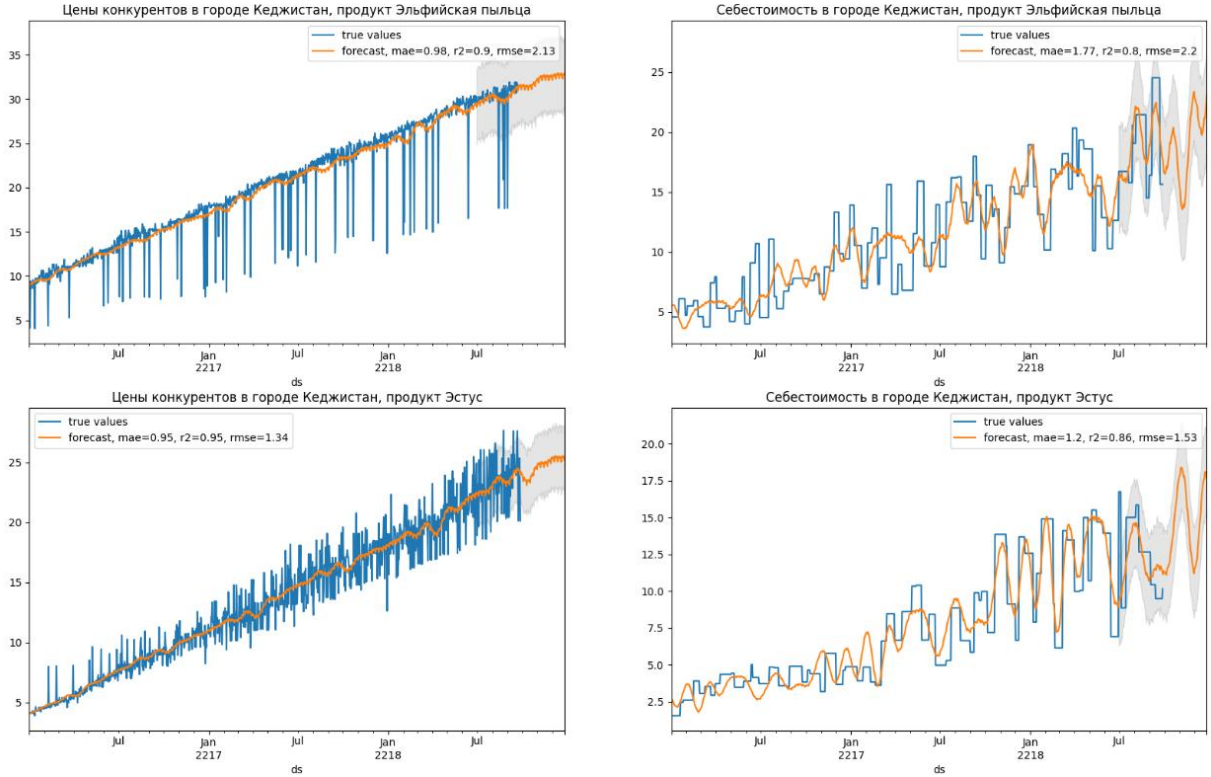
3) Построение прогнозов

Построение моделей и прогноз цен конкурентов и себестоимостей осуществлялся с помощью библиотеки $Prophet$, созданной Facebook. Идея, лежащая в основе построения моделей и прогноза, - разложение временного ряда на основные составляющие:

$$y(t) = g(t) + s(t) + h(t) + f(t) + e_t,$$

где $g(t)$ – компонента, описывающая тренд;
$s(t)$ – компонента, описывающая сезонные колебания;
$h(t)$ – компонента, отвечающая за праздники;
$f(t)$ – компонента, учитывающая экзогенные факторы;
e_t – ошибка прогноза.

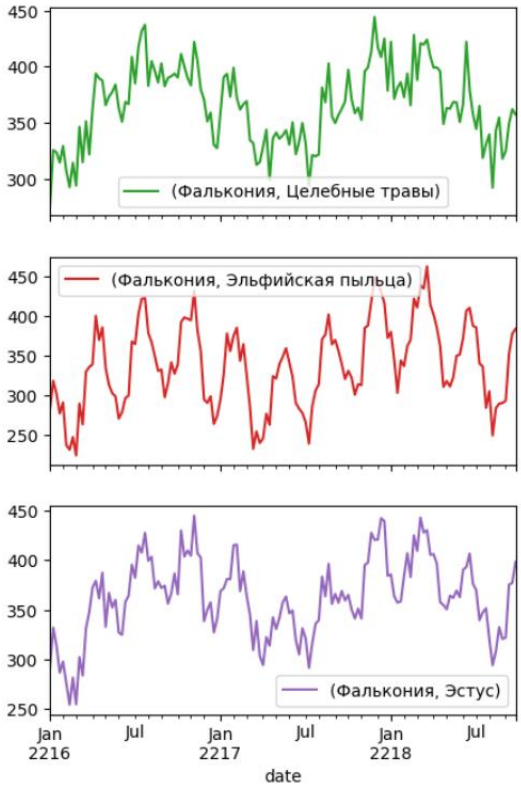
Результаты получились следующими:



Прогноз можно считать хорошим, т.к. полученные метрики (MAE и RMSE, R^2) дают довольно хорошие результаты.

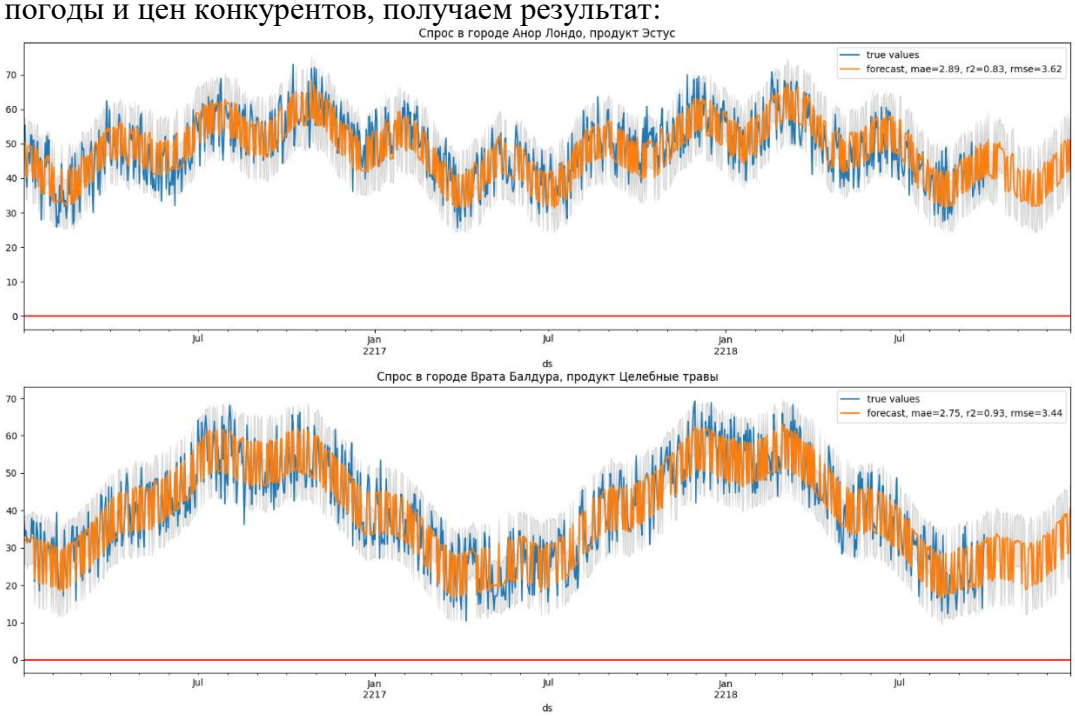
4.2 Прогноз продаж

Для анализа компонент рядов продаж тоже агрегируем их по неделям, возьмём сумму (на самом деле можно и среднее). Посмотрим на графики:



Анализ графиков продаж даёт гораздо больше информации: видно, что тренда, вообще говоря, никакого нет, отчётливо видна сезонность около 16-ти месяцев, а также ежеквартальная. Опять же, исходя из логических соображений, в дальнейшем добавим в модель недельную сезонность.

Обучив модель $Prophet$ с добавленными регрессорами в виде погоды и цен конкурентов, получаем результат:



В среднем $\mathrm{MAE} = 2.77, \mathrm{RMSE} = 3.46, \mathrm{R}^2 = 0.85$. На графиках видно, что модель очень хорошо улавливает сезонности и траекторию продаж.

Добавив полученные прогнозы в раннее созданный датафейм, получим:

date	place	product	amount	price	price_c	hot	rain	snow	cost	amount_pred
2216-01-02	Анор Лондо	Целебные травы	41.494198	3.90	4.210000	1.0	0.0	0.0	2.070000	44.277035
2216-01-03	Анор Лондо	Целебные травы	46.984980	3.90	4.046667	1.0	0.0	0.0	2.070000	44.080445
2216-01-04	Анор Лондо	Целебные травы	39.587422	3.90	3.980000	0.0	0.0	0.0	2.070000	29.801964
2216-01-05	Анор Лондо	Целебные травы	39.769653	3.90	3.903333	0.0	0.0	0.0	2.070000	29.963666
2216-01-06	Анор Лондо	Целебные травы	49.805077	3.90	3.980000	0.0	0.0	1.0	2.070000	44.246446
...	...	...	...	...	...	...	...	...	...	...
2218-12-22	Фалькония	Эстус	48.573931	9.28	15.923121	0.0	0.0	0.0	5.900058	48.573931
2218-12-23	Фалькония	Эстус	48.719230	9.28	15.766782	0.0	0.0	0.0	5.843973	48.719230
2218-12-24	Фалькония	Эстус	49.266531	9.28	15.688450	0.0	0.0	0.0	5.862870	49.266531
2218-12-25	Фалькония	Эстус	58.322015	9.28	15.857842	0.0	0.0	1.0	5.849556	58.322015
2218-12-26	Фалькония	Эстус	49.939101	9.28	15.810387	0.0	0.0	0.0	5.878643	49.939101

4. Нахождение оптимальных цен на 90 дней

Сам кейс - объединение задачи прогнозирования и задачи оптимизации. Прогнозируем спрос и себестоимость, оптимизируем цены.

В этой части задания необходимо составить точную математическую постановку задачи нахождения оптимальной цены с ограничениями при максимизации прибыли. Т.е. целевая функция - прибыль, которую мы максимизируем, неизвестные - цены в каждый момент времени, а ограничения:

1. цена должна держаться более 3-х дней;
2. цена не должна меняться за раз более, чем на 1 золотой;
3. цена не должна быть выше на 20%, чем у конкурента.

4.1 Формирование зависимости спроса и цены

Спрос зависит от цены, а цена (будущая) зависит от спроса. Но мы спрогнозировали спрос так, что от цены он не зависит. Это неправильно, потому что если мы установим цену в какой-то из дней очень большой, например, 100, а спронозированный спрос на этот день был равен 50, то и спрос должен изменится - из-за слишком большой цены он должен упасть. И наоборот. Эту зависимость можно добавить в модель посредством построения модели линейной регрессии.

Идея заключается в том, чтобы взять реальный спрос и аппроксимацию спроса (до тех 90 дней, на которые мы хотим спрогнозировать). Их разница - ошибка модели. Пусть

$$\widehat{am}_t - am_t = \alpha \times p_{t-1} + \beta,$$

т.е. ошибка модели состоит из α, умноженной на вчерашнюю цену (вчерашнюю, т.к. сегодняшнюю мы ещё не знаем, мы хотим её определить), и β.

Таким образом, в дальнейшем мы сможем формировать спрос, исходя из спрогнозированного спроса $\widehat{am}_t$ и поправки за счёт цены $\alpha \times p_{t-1} + \beta$: как раз в таком случае, если цена будет очень большой, спрос будет уменьшаться, а если маленькой, увеличиваться.

Коэффициенты α и β представлены в таблице ниже (*coef* и *intercept*):

	place	product	coef	intercept
0	Анор Лондо	Целебные травы	-0.097185	0.851656
1	Анор Лондо	Эльфийская пыльца	-0.145837	2.109102
2	Анор Лондо	Эстус	-0.225626	1.553611
3	Врата Балдура	Целебные травы	0.024044	-0.290652
4	Врата Балдура	Эльфийская пыльца	-0.023678	0.50238
5	Врата Балдура	Эстус	-0.027498	0.376547
6	Кеджистан	Целебные травы	-0.051128	0.566171
7	Кеджистан	Эльфийская пыльца	0.00957	-0.19713
8	Кеджистан	Эстус	-0.003566	0.056148
9	Нокрон	Целебные травы	-0.070096	0.410607
10	Нокрон	Эльфийская пыльца	-0.049146	0.633771
11	Нокрон	Эстус	-0.005352	0.039114
12	Фалькония	Целебные травы	0.006573	-0.049694
13	Фалькония	Эльфийская пыльца	-0.054288	0.695389
14	Фалькония	Эстус	-0.15052	1.245275

Видно, что в некоторых местах получилось $\alpha > 0$, но это компенсируется за счёт отрицательного коэффициента β.

4.3 Формирование экземпляров городов с продаваемыми продуктами

Для данной задачи удобно использовать *dataclass* для создания экземпляра для каждого города и для каждого продукта в совокупности. В каждом экземпляре города и продукта помимо их названий хранится:

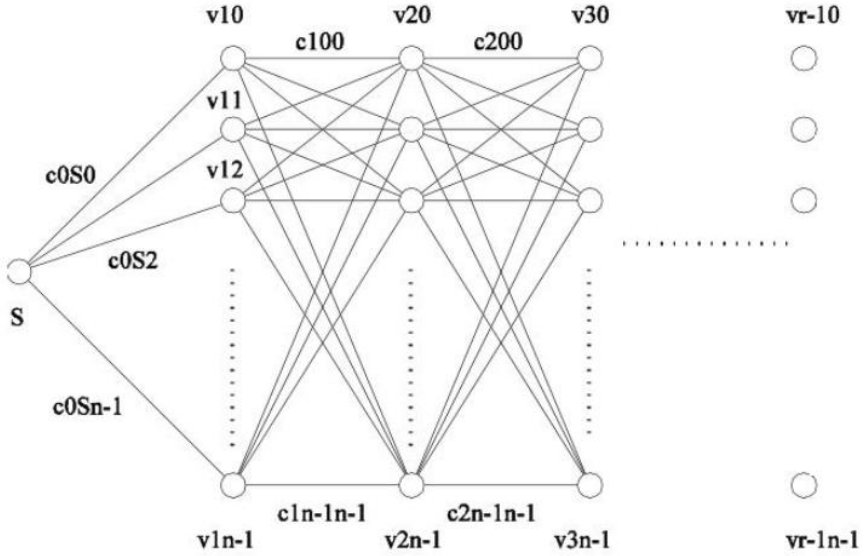
1. горизонт планирования (90 дней);
2. цены, которые мы можем установить в данном городе (берётся 200 вещественных цен от минимальной спрогнозированной стоимости до максимальной наблюдаемой цены конкурента в заданном городе на заданный продукт), умноженной на 1.2 (по сути - верхняя граница нашей цены по постановке);
3. коэффициенты α и β;
4. спрогнозированная на 90 дней себестоимость;
5. спрогнозированная на 90 дней цена конкурента;
6. спрогнозированный на 90 дней спрос;
7. цены в последние 4 дня;
8. *delta* (1 золотой), больше которого мы не можем изменять цену за раз.

Т.е. имеем 15 экземпляров городов с продуктами (5 городов, 3 продукта продаётся в каждом).

```
@dataclass
class PlaceAndProductData:
    place: str
    product: str
    days: int # горизонт планирования
    feasible_prices: np.array # допустимые цены
    alpha: np.array # массив коэффициентов
    costs: np.array # себестоимость на прогнозируемые дни
    prices_c: np.array # цена конкурента в прогнозируемые дни
    am_old_p: np.array # спрос в прогнозируемые дни
    last_4_prices: np.array # последние данные 4 цены
    delta: float # = 1, больше этого значения не можем изменить цену
```

4.3 Алгоритм динамического ценообразования

Идея выбранного алгоритма динамического ценообразования: мы представляем все в виде послойного графа, где каждая вершина представлена в виде $(day, price)$, где day - это конкретный день, а $price$ - какая-то цена из заранее равномерно нарезанных цен. На рисунке ниже представлен пример послойного графа:



При поиске максимальной прибыли, на этом графе по сути мы будем решать задачу ***Longest Path Problem*** . Она является NP-трудной, но так как граф ациклический (и ещё и сразу топологически отсортированный), то это задача становится разрешимой за полиномиальное время. Это осуществляется с помощью динамического программирования.

Для каждой вершины v мы просматриваем все предыдущие вершины u и среди них выбираем аргумент максимума величины $dp[v] = dp[u] + w[u, v]$. При этом вес формируется как прибыль, зависящая от спроса с поправкой на цену, $w[u, v] = (p[v] - c_t)(am_t + \alpha \times p[u] + \beta)$. Также отсекаются ребра, по которым нельзя ходить (по условиям исходной задачи (не можем менять цену за раз более, чем на 1 золотой и допустимость цены с точки зрения цены конкурента)). В итоге получаем допустимое оптимальное решение данной задачи нахождения оптимальных цен при максимизации прибыли.

Сам алгоритм с пояснёнными строками:

```
def dynamic_pricing_algorithm(inst):
    n = inst.feasible_prices.shape[0]

    # ключи - вершины вида (day, p_idx), где p_idx - индекс цены в массиве цен,
    # значения - кортеж вида (profit, p_idxes), где profit - суммарная прибыль к дню day при цене с индексом p_idx
    # p_idxes - предыдущие 4 вершины (индексы цен) перед вершиной p_idx
    dp = dict()

    # вершины - индексы последних 4х цен
    v0 = [0 for _ in inst.last_4_prices]
     # ищем место каждой цены в массиве допустимых цен
    for i in range(len(v0)):
        best_error = 1000
        for v in range(n):
            cur_error = abs(inst.feasible_prices[v] - inst.last_4_prices[i])
            if cur_error < best_error:
              best_error = cur_error
              v0[i] = v
```

```
    # шаг инициализации dp: что после последних 4х цен?
    # если 4 последних наблюдаемых дня была одинаковая цена
    if v0[0] == v0[1] == v0[2] == v0[3]:
        for v in range(n):
            # проверяю 2 условия для каждой допустимой цены: беру ее только в том случае, если
            # 1. между ней и последней наблюдаемой ценой разница не более 1 золотого
            # 2. она не выше цены конкурента на 20% в данный день
            if inst.feasible_prices[v] - inst.feasible_prices[v0[-1]] <= inst.delta and inst.feasible_prices[v] <= 1.2 * inst.prices_c[0]:
                # это условие нужно, если формировать допустимые цены таким образом, что они могут быть меньше себестоимости
                if inst.costs[0] <= inst.feasible_prices[v]:
                    # спрос с поправкой на цену
                    fixed_b = inst.am_old_p[0] + inst.feasible_prices[v0[-1]] * inst.alpha[0] + inst.alpha[1]
                    profit = (inst.feasible_prices[v] - inst.costs[0]) * fixed_b

                    # добавляем в словарь по вершине текущую суммарную оптимальную прибыль
                    # и предыдущие 4 вершины, соответствующие предыдущим 4м ценам,
                    # т.е. добавляем информацию о том, что мы можем поставить данную цену после последней и какой при этом будет прибыль
                    dp[(0, v)] = (profit, v0)
    # если 4 последних наблюдаемых дня цена была разная
    # то у нас есть только 1 выход - поставить последнюю наблюдаемую цену, т.к. любая цена должна держаться > 3 дней
    else:
        fixed_b = inst.am_old_p[0] + inst.feasible_prices[v0[-1]] * inst.alpha[0] + inst.alpha[1]
        profit = (inst.feasible_prices[v0[-1]] - inst.costs[0]) * fixed_b
        dp[(0, v0[-1])] = (profit, v0)

    # шаги заполнения dp: добавляем остальные вершины в dp
    for t in range(1, inst.days):
        for v in range(n):
            for u in range(n):
                # проверяю уже 3 условия для каждой цены: 1. она не выше цены конкурента на 20%
                #                                         2. существует ли ребро ((t-1, u), (t, v))
                #                                         3. между ней и предыдущей разница больше 1 золотого
                if inst.feasible_prices[v] <= 1.2 * inst.prices_c[t] and (t - 1, u) in dp and inst.feasible_prices[v] - inst.feasible_prices[u] <= inst.d
                    # предыдущие вершины для и
                    pr = dp[t - 1, u][1]
                    # проверяем, держалась ли цена > 3 дней
                    if (pr[0] == pr[1] == pr[2] == pr[3] and u == v) or (pr[1] == pr[2] == pr[3] == u) or (pr[3] == u == v):
                        if inst.costs[t] <= inst.feasible_prices[v]:
                            # спрос с поправкой на цену
                            fixed_b = inst.am_old_p[t] + inst.feasible_prices[u] * inst.alpha[0] + inst.alpha[1]
                            # суммируемая прибыль
                            profit = (inst.feasible_prices[v] - inst.costs[t]) * fixed_b + dp[(t - 1, u)][0]
                            # проверяем, надо ли обновить вершину (вершина не была определена, либо прибыль больше прежде зафиксированной)
                            if ((t, v) not in dp) or dp[(t, v)][0] <= profit:
                                # добавляем в словарь по вершине текущую суммарную оптимальную прибыль
                                # и предыдущие 4 вершины, соответствующие предыдущим 4м ценам
                                dp[(t, v)] = profit, pr[1:] + [u]
    # на последнем слое у нас как раз хранится информация о совокупных прибылях, полученных по определенным траекториям цен
    last_layer = [(j, dp[i, j][0]) for i, j in dp if i == inst.days - 1]

    # ищем максимальную прибыль
    argmx = sorted(last_layer, key=lambda x: x[1], reverse=True)[0]
    # сохраняем индекс вершины, в которую пришли и прибыль
    last_v, total_profit = argmx
    last_t = inst.days - 1
    res = [last_v]
    while last_t >= 1:
        last_v = dp[last_t, last_v][1][-1] # последовательно продвигаемся по вершинам назад
        res.append(last_v) # и формируем путь
        last_t -= 1

    return total_profit, inst.feasible_prices[res[::-1]]
```

В итоге можно сформировать датафрейм с 1350 строками, где хранится информация об оптимальных с точки зрения прибыли ценах, удовлетворяющих всем поставленным ограничениям, которые необходимо установить на последующие 90 дней в каждом городе на каждый продукт. Также, просто для сравнения туда добавлены базовые цены (бэйзлайн), цены конкурента и верхняя граница для устанавливаемой цены. Описанный датафрейм:

	date	product	place	price	base_price	price_c	UB_price_c
0	2218-09-28	Целебные травы	Анор Лондо	11.003371	13.63	9.464237	11.357084
1	2218-09-29	Целебные травы	Анор Лондо	11.003371	13.63	9.721946	11.666335
2	2218-09-30	Целебные травы	Анор Лондо	11.003371	13.63	9.572652	11.487183
3	2218-10-01	Целебные травы	Анор Лондо	11.003371	13.63	9.191660	11.029993
4	2218-10-02	Целебные травы	Анор Лондо	11.241528	13.63	9.577265	11.492718
...	...	...	...	...	...	...	...
85	2218-12-22	Эстус	Фалькония	18.896187	9.28	15.923121	19.107746
86	2218-12-23	Эстус	Фалькония	18.896187	9.28	15.766782	18.920138
87	2218-12-24	Эстус	Фалькония	18.790563	9.28	15.688450	18.826140
88	2218-12-25	Эстус	Фалькония	18.790563	9.28	15.857842	19.029410
89	2218-12-26	Эстус	Фалькония	18.790563	9.28	15.810387	18.972465

1350 rows × 7 columns

По нему можно посмотреть допустима ли цена с точки зрения верхней границы и т.д. По среднему, к слову, можно легко проверить допустимость наших цен, т.к. если каждая наша цена была меньше верхней границы в каждый день, то и среднее будет меньше среднего верхней границы.

		price	base_price	price_c	UB_price_c
place	**product**				
Анор Лондо	**Целебные травы**	11.393243	13.630000	9.731401	11.677681
	Эльфийская пыльца	22.956929	18.950000	19.751019	23.701223
	Эстус	13.757135	10.680000	11.776833	14.132200
Врата Балдура	**Целебные травы**	20.063126	16.110000	17.116455	20.539746
	Эльфийская пыльца	31.802719	27.470000	26.962862	32.355434
	Эстус	24.383254	20.330000	20.618274	24.741928
Кеджистан	**Целебные травы**	21.138758	19.090000	17.986001	21.583201
	Эльфийская пыльца	37.480392	31.200000	32.189196	38.627035
	Эстус	28.926706	24.245081	24.670115	29.604138
Нокрон	**Целебные травы**	10.885396	9.480000	9.213866	11.056639
	Эльфийская пыльца	19.339778	18.110000	16.329045	19.594854
	Эстус	12.848164	9.970000	10.910342	13.092411
Фалькония	**Целебные травы**	15.109747	10.970000	12.938453	15.526143
	Эльфийская пыльца	21.204531	15.249648	18.846938	22.616325
	Эстус	16.864687	9.280000	15.650364	18.780437

Если взять в итоговом датафрейме столбцы с городами и продуктами, номерами дней, в которые мы выставляем цену, и ценами, округлёнными до сотых (т.е. который мы должны предоставить в формате .parquet), получим:

	day_number	place	product	price
0	1	Анор Лондо	Целебные травы	11.00
1	2	Анор Лондо	Целебные травы	11.00
2	3	Анор Лондо	Целебные травы	11.00
3	4	Анор Лондо	Целебные травы	11.00
4	5	Анор Лондо	Целебные травы	11.24
...	...	...	...	...
85	86	Фалькония	Эстус	18.90
86	87	Фалькония	Эстус	18.90
87	88	Фалькония	Эстус	18.79
88	89	Фалькония	Эстус	18.79
89	90	Фалькония	Эстус	18.79

1350 rows × 4 columns

Также была сформирована таблица с прибылью в каждом городе по каждому продукту. В неё добавлены столбцы базовой прибыли (по спрогнозированному спросу с учётом поправки по цене и базовой цене), допустимо ли с точки зрения цены конкурента базовое решение и прибавка по прибыли при динамическом ценообразовании относительно использования постоянной цены.

	product	place	opt_profit	baseline_profit	is_baseline_feasible	increase
0	Целебные травы	Анор Лондо	11402.021353	18568.384989	False	-7166.363636
1	Эльфийская пыльца	Анор Лондо	41343.306979	30875.079048	True	10468.227931
2	Эстус	Анор Лондо	29616.798322	18961.599183	True	10655.19914
3	Целебные травы	Врата Балдура	25638.962189	15028.503302	True	10610.458887
4	Эльфийская пыльца	Врата Балдура	28353.520504	21299.324027	True	7054.196477
5	Эстус	Врата Балдура	26537.266414	17876.680107	True	8660.586308
6	Целебные травы	Кеджистан	34130.161158	25949.620629	False	8180.540528
7	Эльфийская пыльца	Кеджистан	71598.56733	47971.473976	True	23627.093353
8	Эстус	Кеджистан	56940.331776	38540.143618	True	18400.188158
9	Целебные травы	Нокрон	25208.057614	19364.250278	True	5843.807336
10	Эльфийская пыльца	Нокрон	31334.57729	26797.492862	True	4537.084429
11	Эстус	Нокрон	29938.82952	18134.568951	True	11804.260569
12	Целебные травы	Фалькония	35602.561359	17898.397186	True	17704.164173
13	Эльфийская пыльца	Фалькония	52796.503999	29512.969802	True	23283.534197
14	Эстус	Фалькония	48476.834223	17425.01288	True	31051.821342

По последнему столбцу можно убедиться, что в целом мы заработаем примерно на 185 тысяч больше при использовании динамических цен, а в среднем будем зарабатывать больше на 12 тысяч.

Таблицу также можно использовать для дальнейшего анализа, в каком из городов было больше прибыли от определённого продукта, в целом в каком городе было больше всего прибыли и т.д.