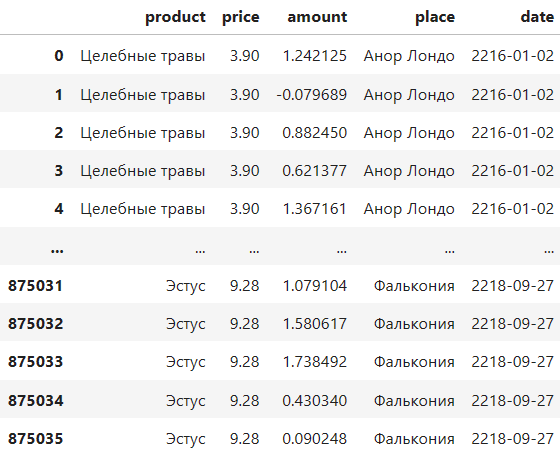
**Матвеева Валерия Владимировна**

**Описание решения кейса GPN CUP 2023:**

**направление Data Science**

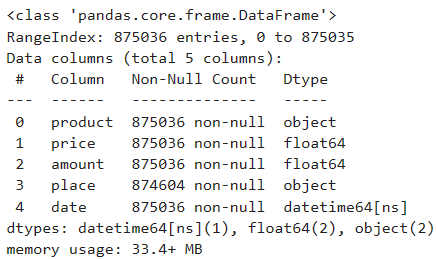
1. **Предобработка исходных данных**
   1. **Данные о транзакциях**

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

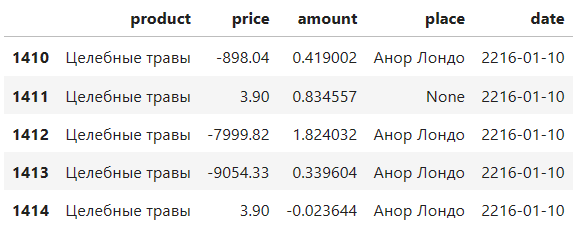


1. **Обработка пропусков**

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



Исследуем его подробнее. Небольшая часть с пропуском:



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.

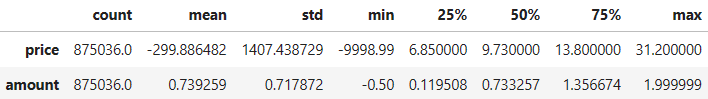


После заполнения пропусков предшествующим значением сильно распределение не изменилось, заполнения примерно равномерно распределились между городами.



1. **Аномальные значения и ошибки**

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



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:

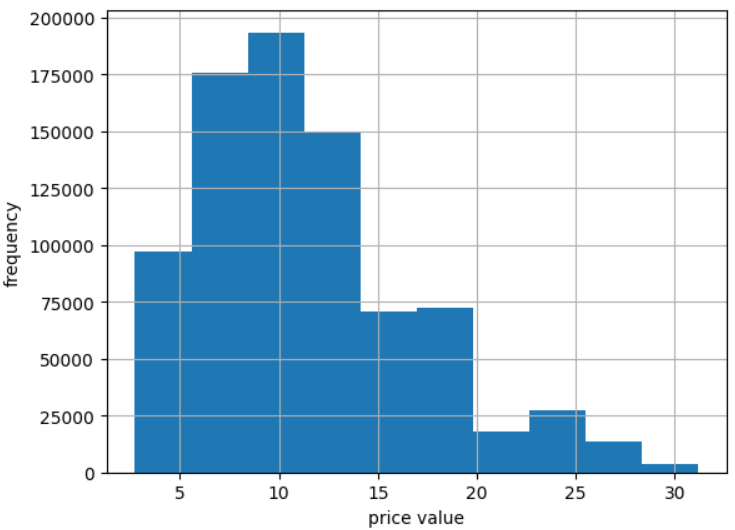


По ценам наблюдаются очень большие отклонения. Исследуем поподробнее, часть датафрейма с отрицательными ценами:



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.

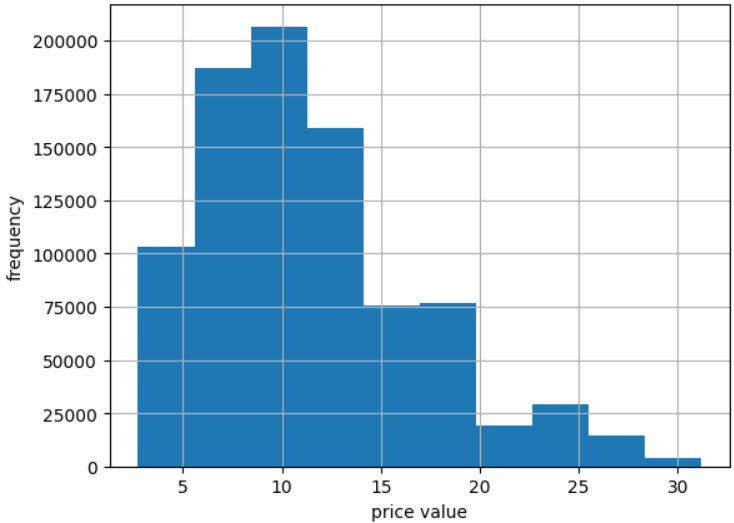
На всякий случай посмотрим, как распределены положительные цены, похоже на логнормальное или гамма-распределение:



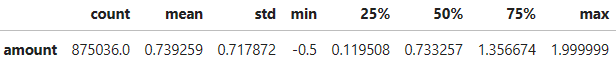
Часть датафрейма с исправленными ошибками:



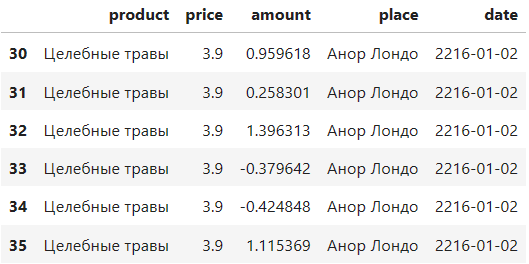
Опять же, видим, что распределение практически не изменилось:



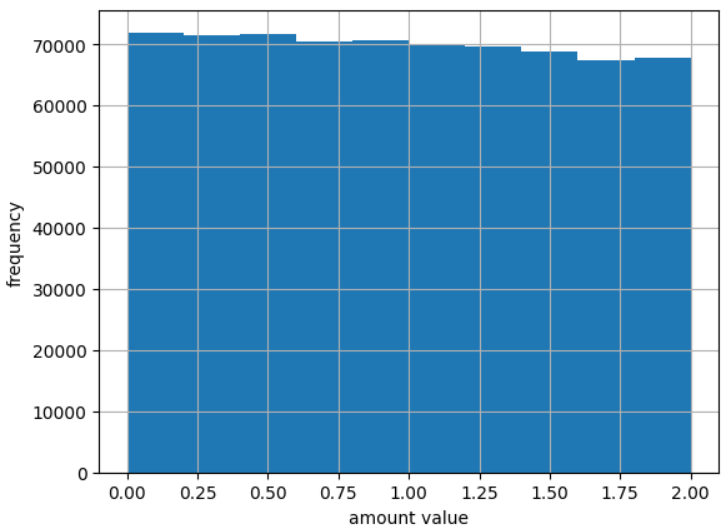
Теперь обработаем продажи. Описательные статистики для продаж:



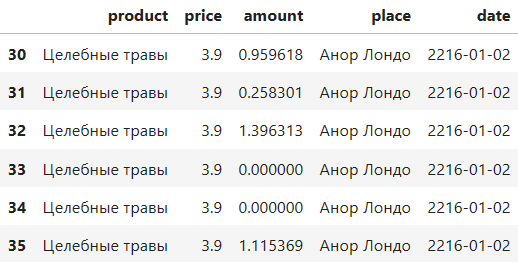
По продажам отклонение меньше, и данные будто подходят нам по значениям, если бы мы брали абсолютное значение. Но это, скорее всего, догадки. Вероятнее, в этот момент сломался аппарат, фиксирующий продажи, и подал неправильно значение, поэтому заменим отрицательные значения на 0, т.к. в дальнейшем мы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



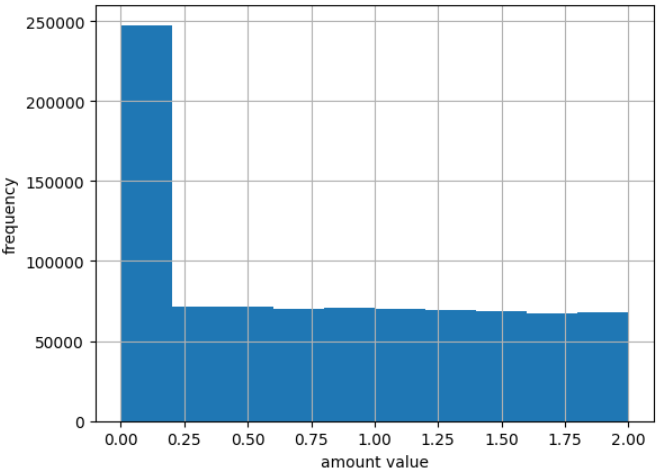
Посмотрим на распределение значений продаж, оно равномерное:



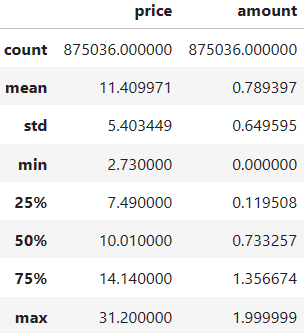
Часть датафрейма с исправлениями:



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:



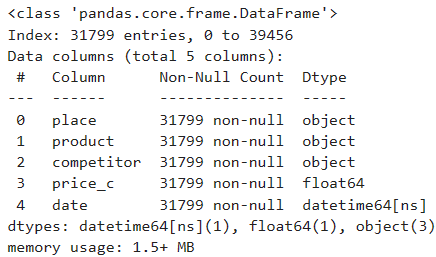
Изменились отклонения: теперь они в пределах разумного, разброс значений не очень большой, минимальные значения положительны.

* 1. **Данные о конкурентах**

Исходный датафрейм:



Информация о датафрейме:



Описательные статистики:

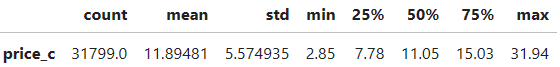
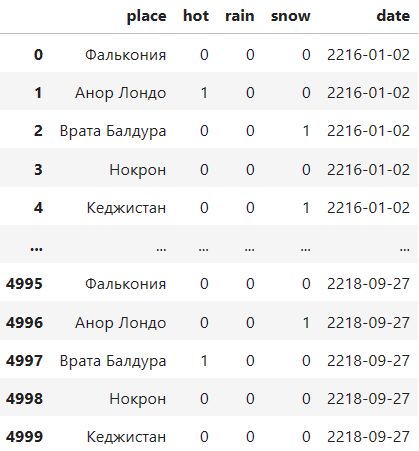


Таблица с конкурентами и их ценами не содержит пропусков и аномальных значений.

* 1. **Данные о погоде**

Исходный датафрейм:



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

Информация о датафрейме:

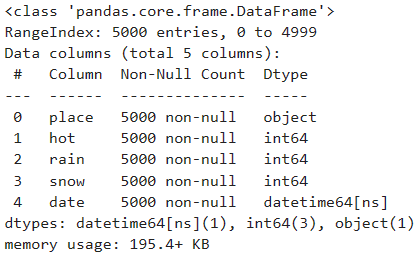
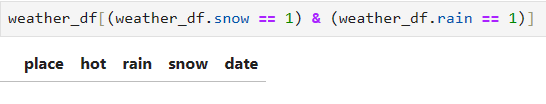
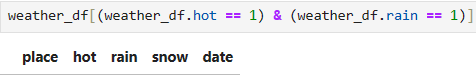
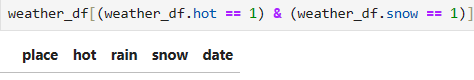


Таблица с погодой и их ценами также не содержит пропусков и аномальных значений.

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

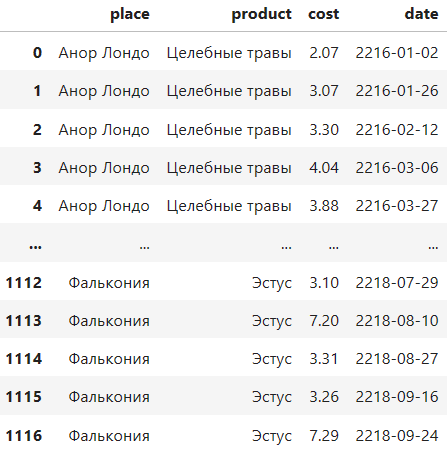




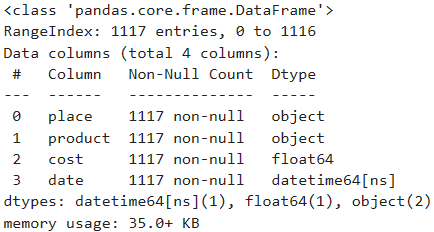


* 1. **Данные о себестоимостях**

Исходный датафрейм:



Информация о датафрейме:



Описательные статистики:

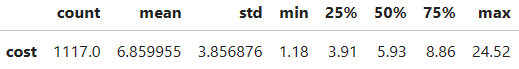


Таблица с себестоимостями не содержит пропусков и аномальных значений, в чем можно убедиться опять же, проанализировав датафрейм и статистики количественных признаков.Отклонение в пределах нормального.

1. **Объединение и группировка таблиц**

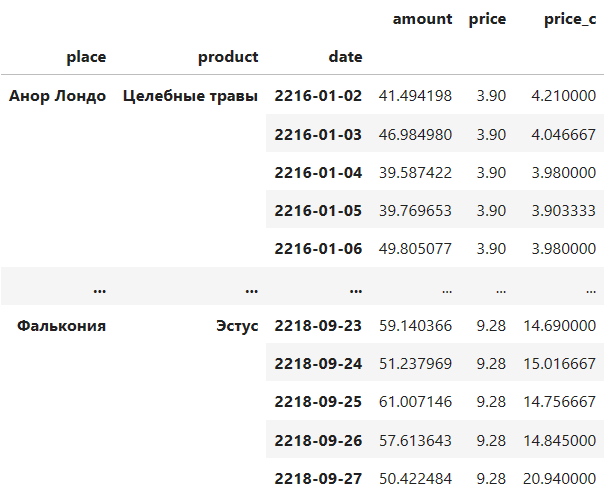
Перед тем, как брать временные ряды, анализировать их компоненты и прогнозировать их, необходимо объединить данные для сопоставления и сгруппировать по городу, продукту и дням.

**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.

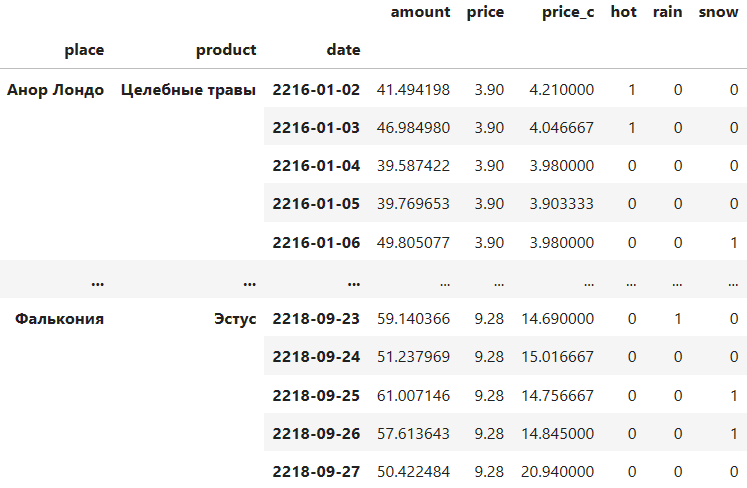
Итоговый объединённый и сгруппированный датафрейм на шаге 1 выглядит следующим образом:



**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:



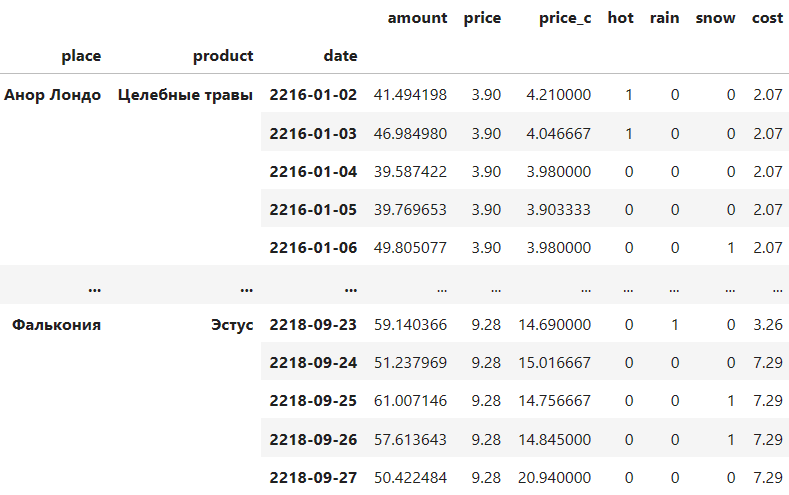
**2.3 Добавление себестоимостей**

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 в городе был произведён в день по цене , то он был доставлен целой партией в этот день 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:



Таким образом, итоговый датафрейм содержит данные о:

1. продажах;

2. цене, по которой они были проданы;

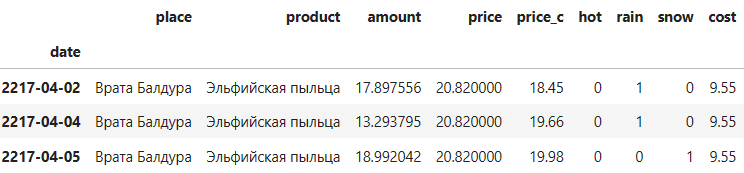
3. цене, предлагаемой конкурентами;

4. погодных условиях;

5. себестоимостях.

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

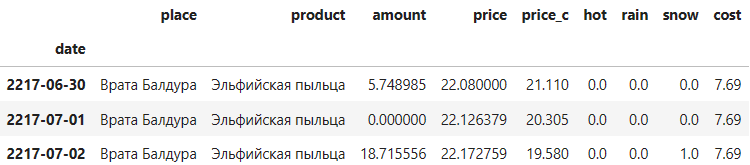
Замечание: было обнаружено, что в данном датафрейме транзакций нет данных по продажам в несколько дней в одном из городов (Врата Балдура) по одному из продуктов(Эльфийская пыльца) (пропущены даты 2217-04-03 и 2217-07-01), что тоже следует обработать, т.к. в дальнейшем анализе временных рядов это может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

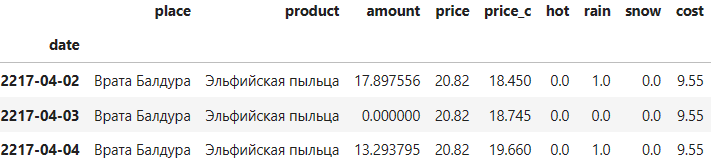




Возникшие пропуски можно заменить следующим образом: изначально в целом добавить строку с пропущенной датой в датафрейм, продажи поставить нулевые, цену и себестоимость интерполировать ценами и себестоимостями в соседние дни, цену конкурента можно заполнить исходя из датафрема *competitors\_df*, взяв 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.

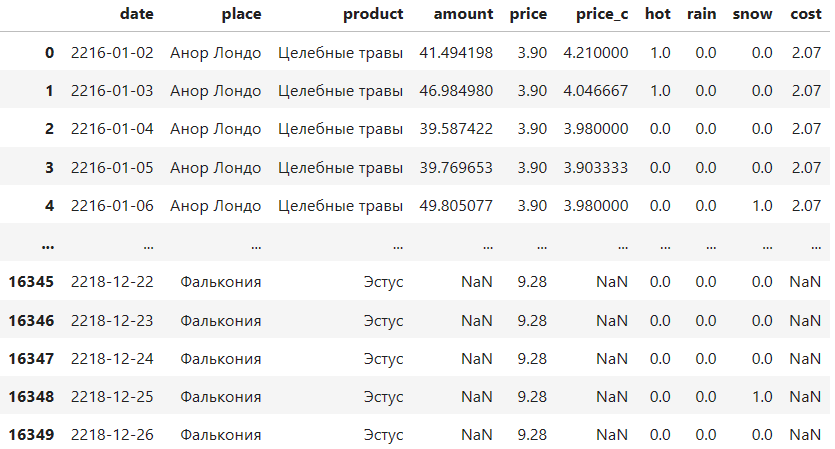
В итоге получим ряд без пропусков:





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:



1. **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.

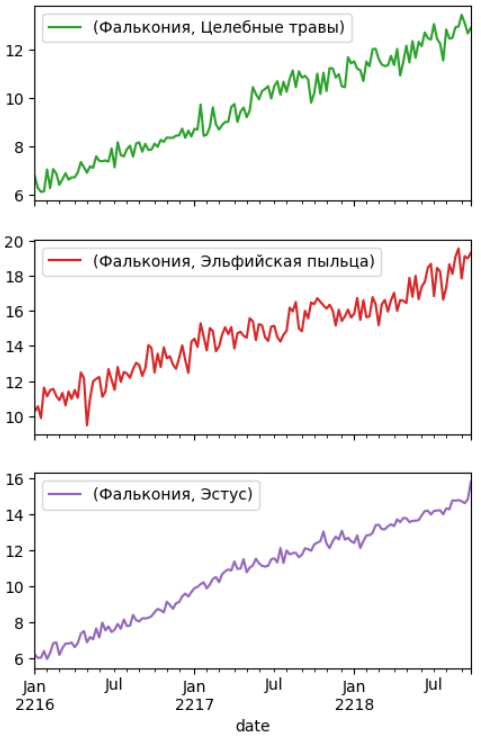
Для того, чтобы спрогнозировать спрос, нам необходимы факторы, от которых он зависит. В данном случае это цена конкурента и погода. Погоду мы уже «спрогнозировали», осталась цена конкурента.

**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**

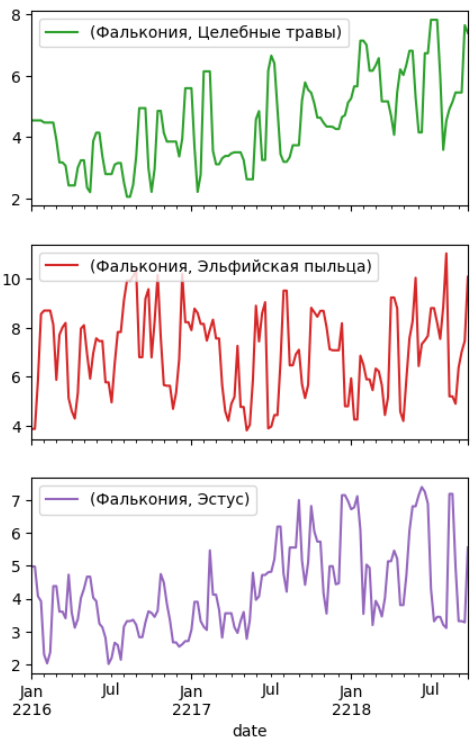
Исходные графики цен конкурентов имеют очень сильные колебания и похожи на шум, т.к. данные каждодневные. В них вообще трудно что-то выявить кроме тренда, поэтому можно агрегировать данные цены конкурентов по неделям (взять данные за всю неделю и усреднить), построить графики по усредненным данным и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

1. **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.

1. **Построение прогнозов**

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

где компонента, описывающая тренд;

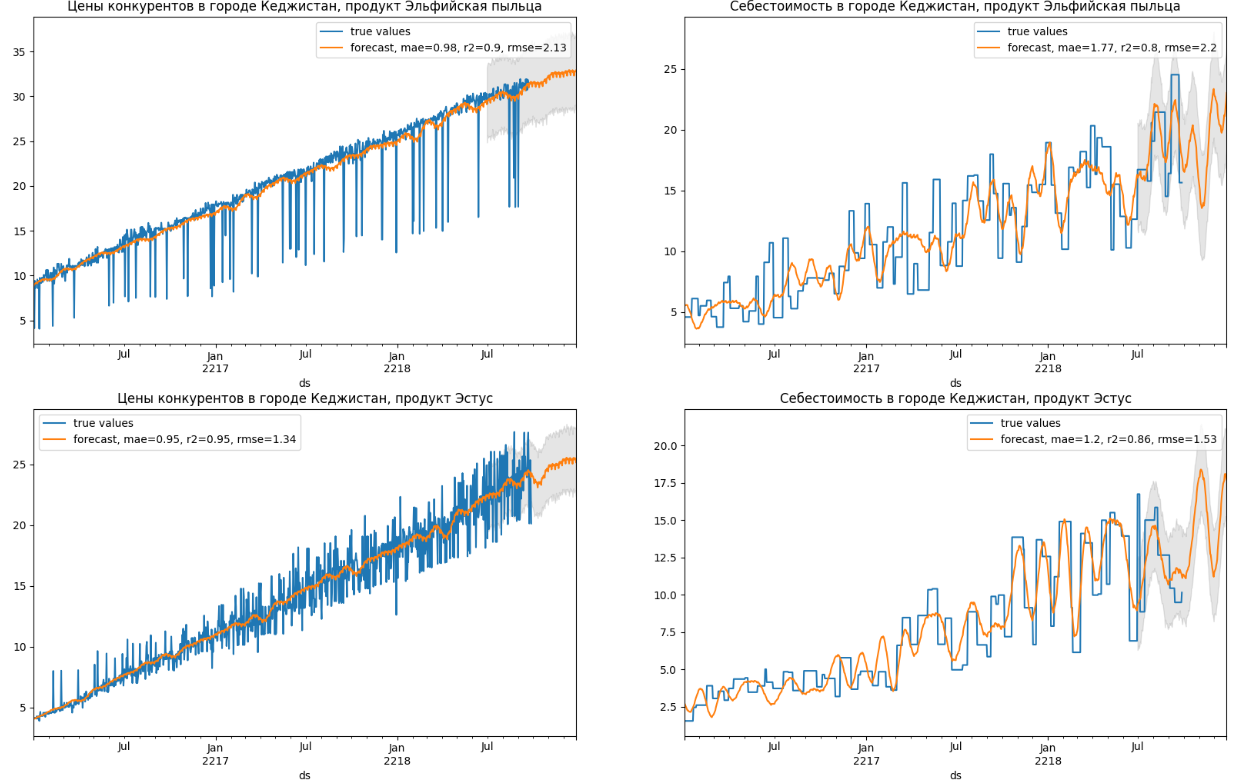
компонента, описывающая сезонные колебания;

компонента, отвечающая за праздники;

компонента, учитывающая экзогенные факторы;

ошибка прогноза.

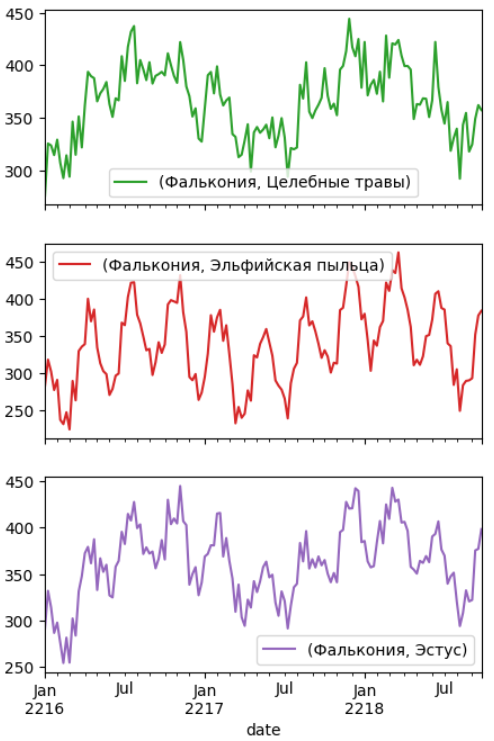
Результаты получились следующими:



Прогноз можно считать хорошим, т.к. полученные метрики () дают довольно хорошие результаты.

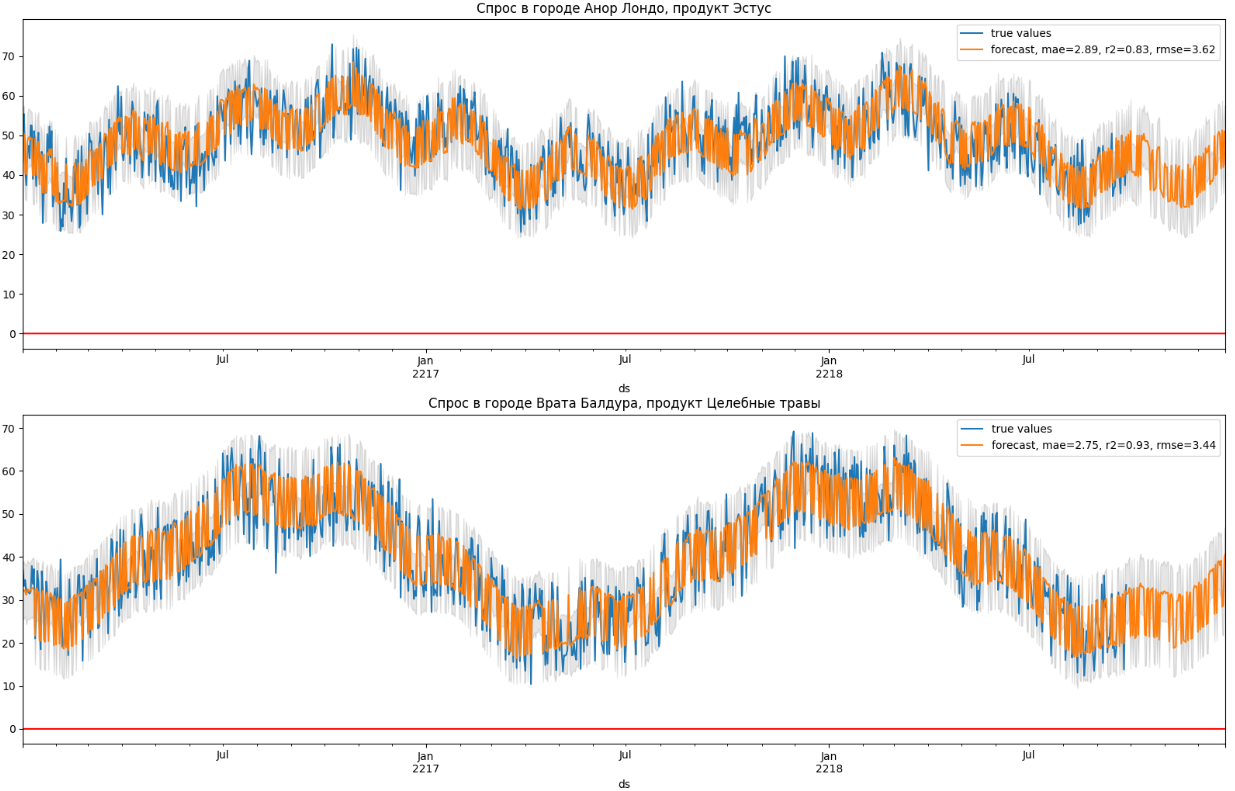
**4.2 Прогноз продаж**

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



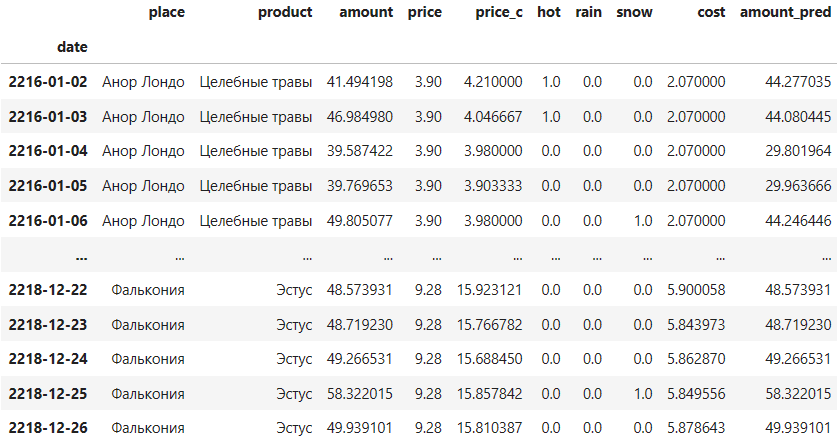
Анализ графиков продаж даёт гораздо больше информации: видно, что тренда, вообще говоря, никакого нет, отчётливо видна сезонность около 16-ти месяцев, а также ежеквартальная. Опять же, исходя из логических соображений, в дальнейшем добавим в модель недельную сезонность.

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



В среднем . На графиках видно, что модель очень хорошо улавливает сезонности и траекторию продаж.

Добавив полученные прогнозы в раннее созданный датафейм, получим:



1. **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**

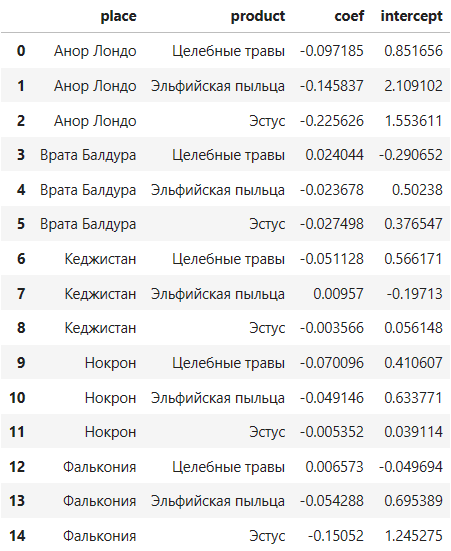
Спрос зависит от цены, а цена (будущая) зависит от спроса. Но мы спрогнозировали спрос так, что от цены он не зависит. Это неправильно, потому что если мы установим цену в какой-то из дней очень большой, например, 100, а спронозированный спрос на этот день был равен 50, то и спрос должен изменится - 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

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

Таким образом, в дальнейшем мы сможем формировать спрос, исходя из спрогнозированного спроса и поправки за счёт цены : как раз в таком случае, если цена будет очень большой, спрос будет уменьшаться, а если маленькой, увеличиваться.

Коэффициентыи представлены в таблице ниже (*coef* и *intercept*):



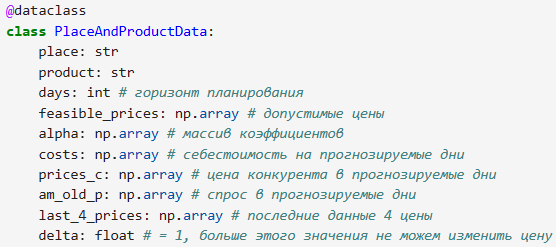
Видно, что в некоторых местах получилось , но это компенсируется за счёт отрицательного коэффициента .

**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:

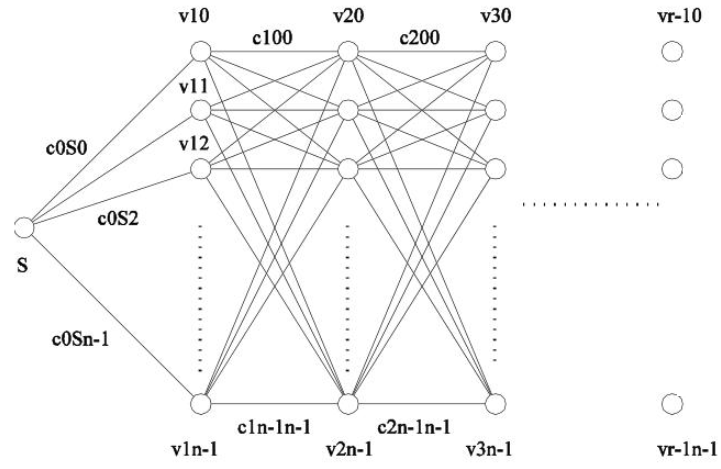
1. горизонт планирования (90 дней);
2. цены, которые мы можем установить в данном городе (берётся 200 вещественных цен от минимальной спрогнозированной стоимости до максимальной наблюдаемой цены конкурента в заданном городе на заданный продукт), умноженной на 1.2 (по сути - верхняя граница нашей цены по постановке);
3. коэффициенты и;
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).



**4.3 Алгоритм динамического ценообразования**

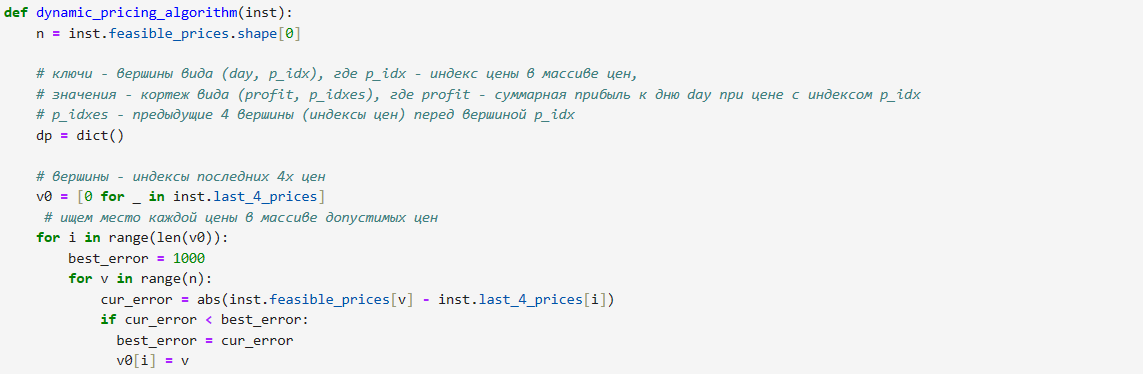
Идея выбранного алгоритма динамического ценообразования: мы представляем все в виде послойного графа, где каждая вершина представлена в виде , где - это конкретный день, а - какая-то цена из заранее равномерно нарезанных цен. На рисунке ниже представлен пример послойного графа:

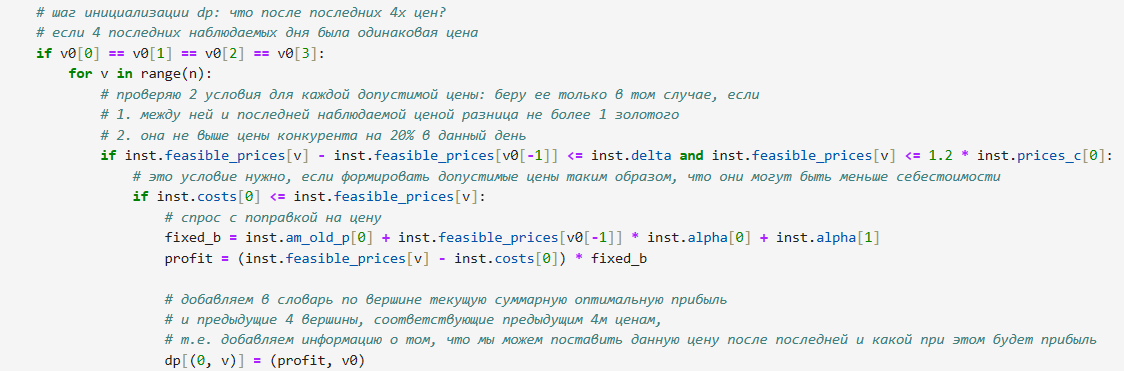


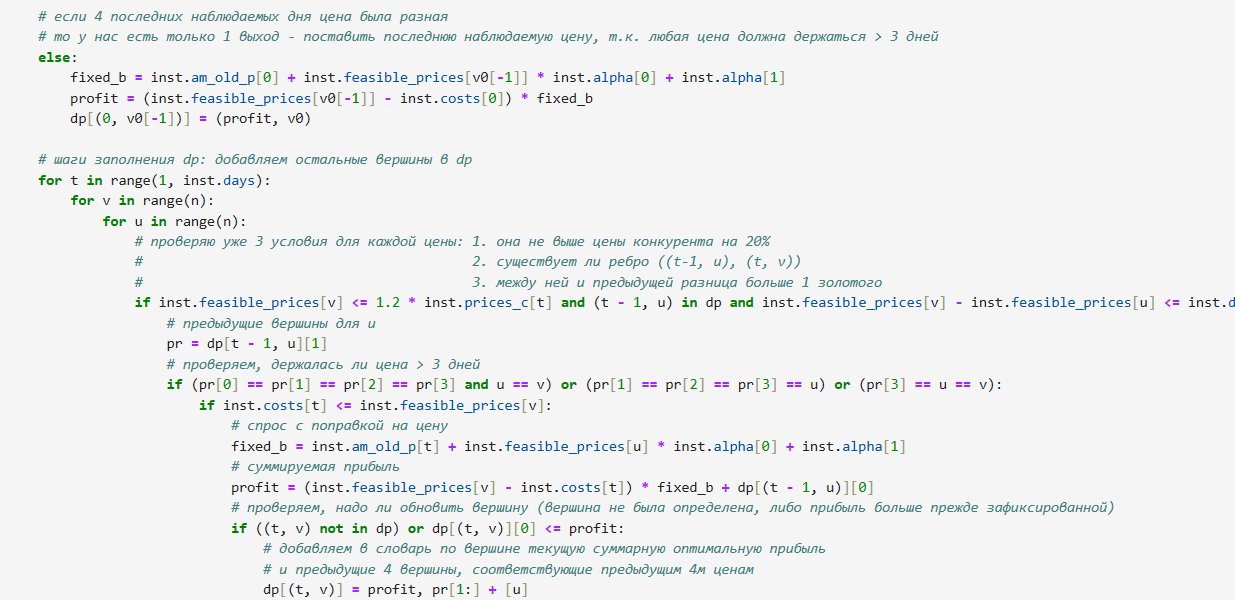
При поиске максимальной прибыли, на этом графе по сути мы будем решать задачу . Она является NP-трудной, но так как граф ациклический (и ещё и сразу топологически отсортированный), то это задача становится разрешимой за полиномиальное время. Это осуществляется с помощью динамического программирования.

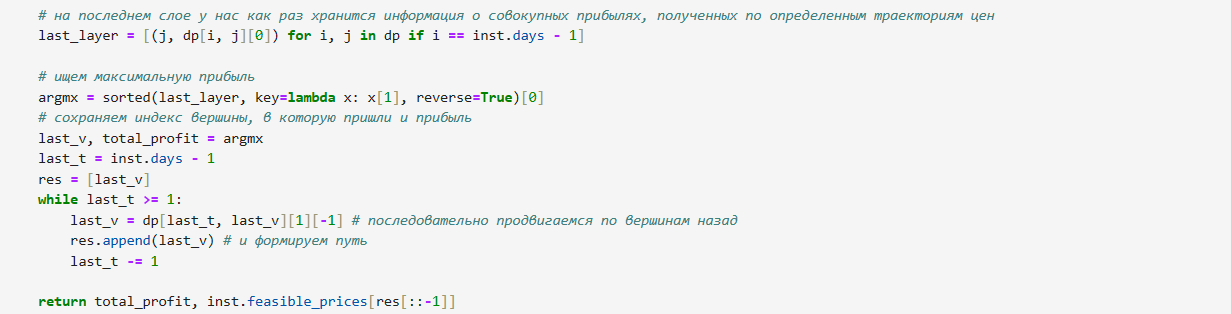
Для каждой вершины мы просматриваем все предыдущие вершины и среди них выбираем аргумент максимума величины . При этом вес формируется как прибыль, зависящая от спроса с поправкой на цену, . Также отсекаются ребра, по которым нельзя ходить (по условиям исходной задачи (не можем менять цену за раз более, 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:

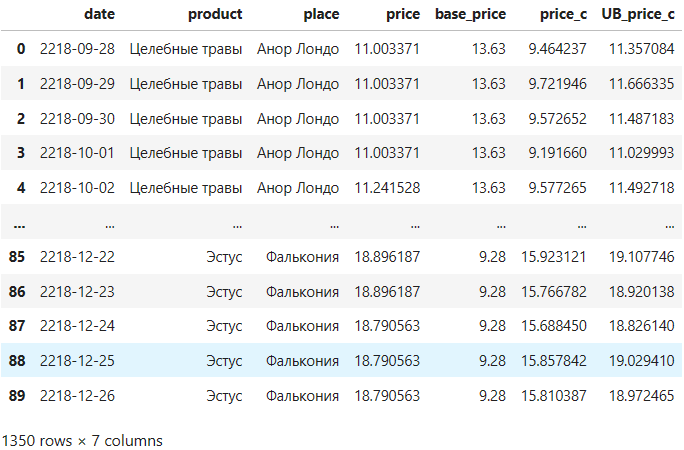




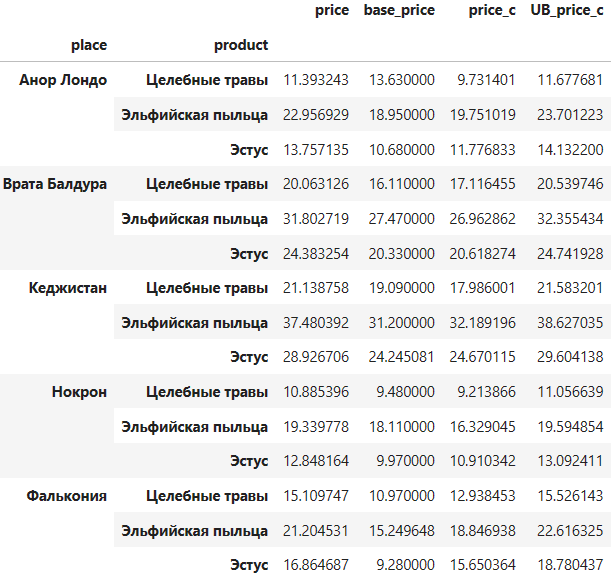




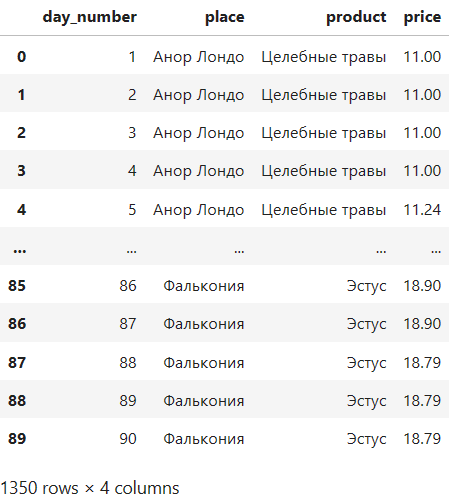
В итоге можно сформировать датафрейм с 1350 строками, где хранится информация об оптимальных с точки зрения прибыли ценах, удовлетворяющих всем поставленным ограничениям, которые необходимо установить на последующие 90 дней в каждом городе на каждый продукт. Также, просто для сравнения туда добавлены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



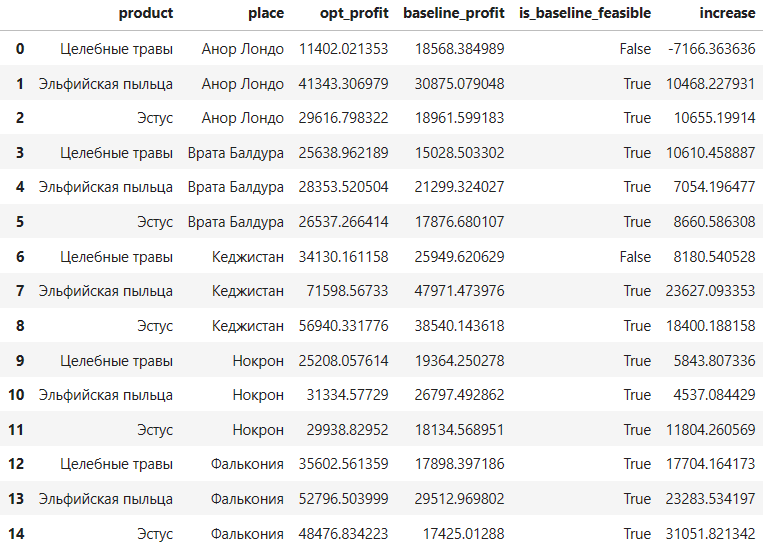
По нему можно посмотреть допустима ли цена с точки зрения верхней границы и т.д. По среднему, к слову, можно легко проверить допустимость наших цен, т.к. если каждая наша цена была меньше верхней границы в каждый день, то и среднее будет меньше среднего верхней границы.



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:



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.



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.