Статистические модели прогнозирования

Скользящее среднее

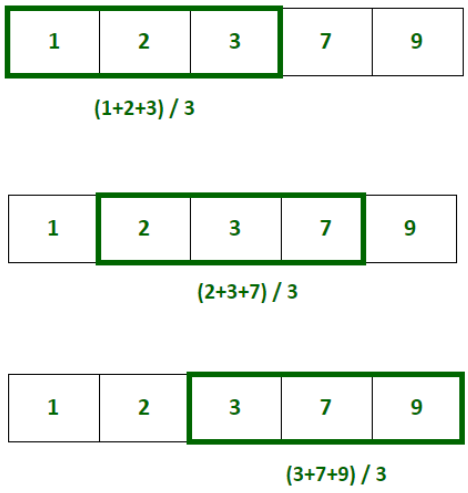
Мы получим новый временной ряд, каждый член которого — среднее арифметическое двух соседних значений исходного ряда:

$$MA_t = \frac{X_{t-1}+X_t}{2}$$

Чуть более продвинутый способ — усреднить сразу несколько наблюдений. Это так называемое **простое скользящее среднее (Simple Moving Average, SMA)**:

$$SMA_t = \frac{X_{t-q}+\ldots+X_t}{q}$$

Таким образом, в скользящем среднем мы суммируем несколько последовательных точек временного ряда и делим эту сумму на количество самих точек, то есть считаем математическое усреднение за определённый период.



Количество точек для суммирования определяется размером **окна (q)**. Чем выше значение этого окна, тем больше данные сглаживаются.

Для сглаживания мы будем использовать встроенный метод pandas.Series.rolling() — он принимает на вход параметр *window* и ожидает после себя агрегирующую функцию для сглаживания (обычно используется среднее). Из преимуществ этого метода можно отметить простоту реализации и интерпретации, из недостатков — чувствительность.

ARMA и ARIMA

> **ARMA** — это авторегрессионное скользящее среднее, или модель авторегрессии-скользящего среднего.

В ней *p* **авторегрессионных слагаемых** и *q* **слагаемых скользящего среднего шумовой компоненты:**

$$X_t = \alpha + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

Таким образом *ARMA* объединяет преимущества двух ранее изученных методов и имеет два параметра:

- *p* — параметр авторегрессионной модели *(AR(p))*;
- *q* — параметр скользящего среднего *(MA(q))*.

Параметр *p* мы определяли по графику частичной автокорреляции. Параметр *q* для скользящего среднего определяют так же, но по коррелограмме (графику автокорреляции).

ARIMA расшифровывается как **Autoregressive Integrated Moving Average** и включает в себя ещё один параметр *(d)*, который означает, что дифференцирование временного ряда порядка *d* приводит ряд к стационарности и будет подчиняться модели *ARMA*.

И *ARMA*, и *ARIMA* реализованы на *Python* в классе ARIMA из statsmodels. Данному классу необходимо передать в качестве параметров временной ряд и порядок order (ARIMA(dta, order=(2, 0, 0))). Для параметра order нужно указать

p, *d* и *q* (именно в таком порядке), причём для получения *ARMA* необходимо указать d=0.

Резюмируем:

- Если ряд стационарный, используем *ARMA*.
- Если ряд нестационарный, с помощью дифференцирования определяем порядок *d* и используем *ARIMA*.

SARIMA (Seasonal ARIMA)

Эта модель очень похожа на *ARIMA*, за исключением того, что в ней есть дополнительный набор компонентов авторегрессии и скользящего среднего.

SARIMA позволяет различать данные по сезонной частоте, а также по их несезонным отличиям. Нахождение лучших для модели параметров можно упростить с помощью средств автоматического поиска, таких как auto_arima из pmdarima.

SARIMAX и ARIMAX

Отличие **SARIMAX** от предыдущей версии заключается в том, что, помимо данных временного ряда, она учитывает **экзогенные** переменные *(Те переменные, которые могут влиять на значения временного ряда)*. Таким образом мы сможем учитывать не только зависимости внутри данных, но и внешние факторы.

Для запуска моделей *SARIMA* и *SARIMAX* на *Python* нужно воспользоваться классом SARIMAX. Если вы хотите использовать *SARIMA*, необходимо задать два обязательных параметра — order и seasonal_order. Order — это порядок для модели *(ARIMA(p, d, q))*. В seasonal_order необходимо передать ещё четыре параметра:

- *P* — сезонный авторегрессионный порядок;
- *D* — порядок дифференцирования сезонного ряда;

- Q — порядок сезонной скользящей средней;
- *m* — размер сезонного периода.

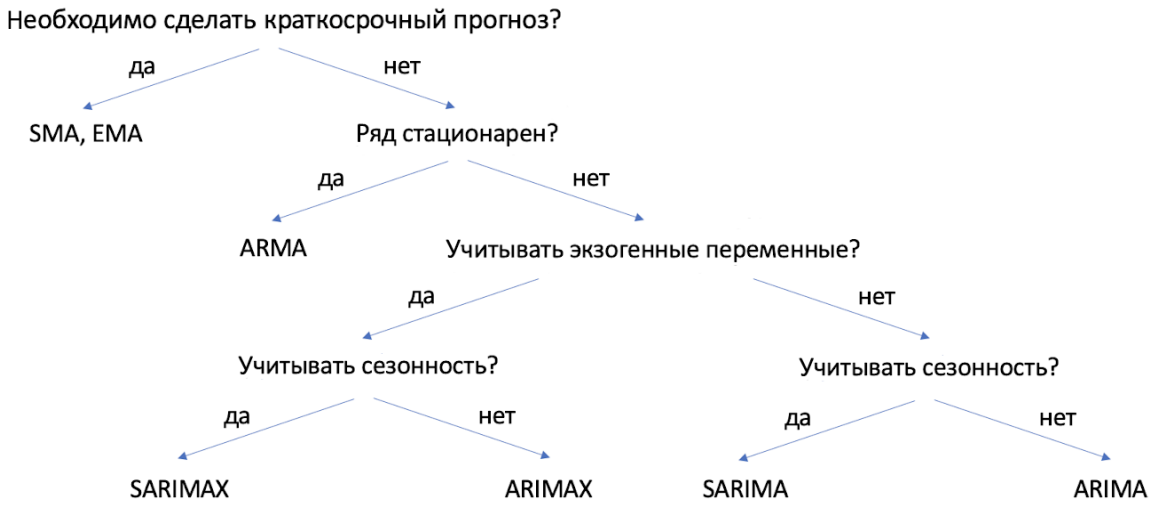
Для учёта экзогенных переменных необходимо передать в класс *SARIMAX* параметр exog=x.

В отличие от *SARIMAX*, **ARIMAX** не учитывает сезонную составляющую, но имеет все преимущества *ARIMA* и учитывает экзогенные переменные.

Как сравнивать эти модели?

Одним из распространённых способов является сравнение качества моделей по **критерию Акаике (AIC)**. Этот информационный критерий вознаграждает модель за качество приближения обученного временного ряда к фактическому, а также «штрафует» её за использование излишнего количества параметров. Принято считать, что модель с наименьшим значением критерия *AIC* является наилучшей.

Как выбрать подходящую модель?



Интерполяция и сэмплирование

Upsampling — это увеличение частоты выборки (повышение частоты дискретизации), например с минут до секунд. Также upsampling применяют для заполнения пропусков неизвестных значений. Для этой цели мы будем использовать **интерполяцию.**

Downsampling — это уменьшение частоты выборки, например с дней до месяцев.

Downsampling

Downsampling — это перегруппировка. Мы можем сгруппировать значения, полученные по дням, в значения, полученные за месяц, путём использования метода groupby(). Однако существует ещё один встроенный в DataFrame метод с чуть более широким функционалом — resample(). Этот метод позволяет делать нестандартные группировки, такие как «за три дня» или «за каждые шесть секунд».

Полный перечень правил группировки можно найти в документации (таблица *DateOffsets*).

Upsampling

Интерполяция — это нахождение некоторых промежуточных значений по функции, описывающей поведение данных. То есть если мы найдём такую функцию, значения которой будут совпадать с уже известными нам значениями, то можно предположить, что она поможет верно или приблизительно восстановить для нас неизвестные значения.

Мы будем использовать уже известный нам метод resample, чтобы декомпозировать данные, например от дня к часам, а затем воспользуемся

встроенным методом interpolate(), который принимает в качестве аргумента указания метода интерполяции: 'linear', 'nearest', 'spline', 'barycentric', 'polynomial' и другие.

Модели прогнозирования гетероскедастичности

> Неоднородность наблюдений, выражающаяся в неодинаковой дисперсии, называется **гетероскедастичностью**.

> Отсутствие гетероскедастичности называется **гомоскедастичностью**.

Волатильность представляет собой меру риска использования финансового инструмента за заданный промежуток времени. Иными словами, волатильность показывает меру изменчивости и часто измеряется в процентах или долях.
Модель *ARCH* используется зависимая от времени условная дисперсия, которая выражается через квадрат предыдущих значений:

$$\sigma^2(t) = a + \sum_{i=1}^{q} b_i r_{t-1}^2$$

Здесь *q* — количество слагаемых, которые влияют на текущее значение, а *b* — весовые коэффициенты, которые влияют на степень значимости предыдущих изменений дисперсии *(r)*. То есть волатильность моделируется в виде суммы константы (*a* — базовая волатильность, константа) и линейной функции абсолютных значений изменения нескольких последних цен.

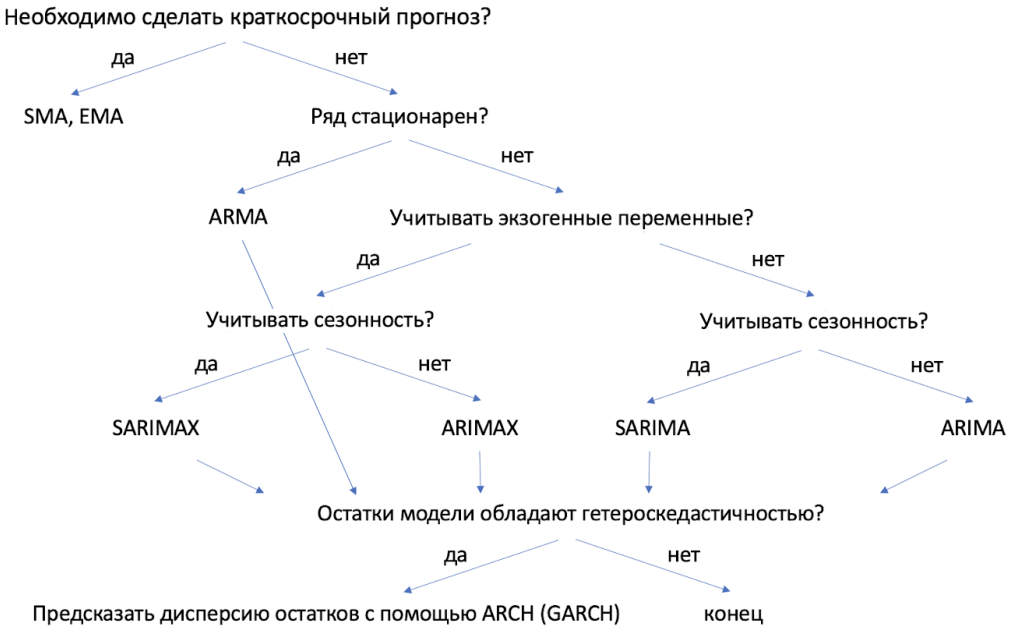
Модель **GARCH (Generalized Autoregressive Conditional Heteroscedastic Model)** предполагает, что на изменчивость дисперсии влияют не только предыдущие изменения показателей, но и предыдущие оценки дисперсии (значение дисперсии).

$$\sigma^2(t) = a + \sum_{i=1}^{q} b_i r_{t-1}^2 + \sum_{i=1}^{p} c_i \sigma_{t-1}^2$$

- первая часть формулы — *ARCH*-модель;
- *p* — количество оценок, предшествующих текущей, которые влияют на текущее значение;
- *c* — это весовые коэффициенты, которые влияют на степень значимости предыдущих дисперсий (σ^2).

Когда применять ARCH и GARCH?

- Когда ряд похож на белый шум, но при этом в нём присутствует гетероскедастичность.
  Чтобы определить, является ли ряд гетероскедастичным (с меняющейся дисперсией), можно отобразить его квадраты на графике и понаблюдать за поведением дисперсии.
- Когда после применения *AR*-модели остатки (ошибки модели) тоже являются гетероскедастичными. В этом случае вы также можете прогнозировать дисперсию ошибок и использовать её в итоговом предсказании. Для этого необходимо суммировать результаты *AR*-модели с результатами *ARCH*.



Валидация временных рядов

Для данных временного ряда разбиение на тренировочный и тестовый сеты нужно выполнять последовательно, иначе в алгоритм просочится информация из будущего, на которой он и обучается.

Аналог кросс-валидации для временных рядов

Вместо кросс-валидации используются:

- *walk forward validation*;
- множественное разбиение.

Другие методы предсказания временных рядов

Prophet

Prophet — это метод прогнозирования данных временных рядов на основе *AR*-модели, в которой учтены годовая, еженедельная и ежедневная сезонности, а также эффекты праздничных дней.

Prophet лучше всего работает с временными рядами, которые имеют сильные сезонные эффекты, а данные накоплены за несколько сезонов. Алгоритм устойчив к отсутствующим данным и сдвигам в тренде и обычно хорошо справляется с выбросами.

Prophet — библиотека с открытым исходным кодом, выпущенная командой *Facebook Core Data Science*. Для загрузки метод также доступен в PyPI (через `pip install`).

NeuralProphet

NeuralProphet — основанная на PyTorch усовершенствованная и более сложная модель, которая комбинирует в себе преимущества традиционных моделей для анализа временных рядов и методов глубокого обучения. *NeuralProphet* можно установить с помощью `pip` (`pip install neuralprophet`).

Модели классического ML

Можно использовать другие модели машинного обучения:

- другие линейные модели регрессии (логистическая, ридж- и лассо-регрессия);
- ансамбли лесов (случайный лес, градиентный бустинг над деревьями);
- *SVM* (метод опорных векторов);
- *KNN* (метод ближайших соседей);
- и др.

Для добавления признаков от даты удобнее всего пользоваться встроенными методами series.dt.

Итоги

Полезные ссылки модуля:

Официальная документация	Статьи
Статистические модели прогнозирования	
• ARMA; • ARIMA; • SARIMAX; • "SARIMAX and ARIMA: FAQ".	• "How to Build ARIMA Model in Python for time series forecasting?"; • «Прогнозирование временных рядов с помощью ARIMA в Python 3».
Интерполяция и сэмплирование	
• Правила группировки для метода resample().	
Модели прогнозирования гетероскедастичности	
• Introduction to ARCH Models.	• "Time Series Model(s) — ARCH and GARCH"; • "Time Series Talk : ARCH Model" **(видеолекция).**

Другие методы предсказания временных рядов	
• Prophet; • NeuralProphet; • pandas.Series.dt.	• «Предсказываем будущее с помощью библиотеки Facebook Prophet»; • "Time Series Forecasting With Prophet in Python"; • "Building load forecasting: Hospital in SF"; • "Facebook Prophet + Deep Learning = NeuralProphet"; • "Prophet vs. NeuralProphet".