УДК 004.04
ББК 32.372
Г90

Груздев А. В.

Г90 Прогнозирование временных рядов с помощью Facebook Prophet, ETNA, Sktime и Linkedin Greykite: Строим, настраиваем, улучшаем модели прогнозирования временных рядов с помощью специальных библиотек. – М.: ДМК Пресс, 2023. – 750 с.: ил.

ISBN 978-5-93700-212-9

Книга посвящена популярным библиотекам прогнозирования временных рядов Prophet, sktime, ETNA и Greykite. Разбирается математический аппарат и API каждой библиотеки. Показаны примеры решения задач прогнозирования, классификации и кластеризации временных рядов, проиллюстрированы темы конструирования и отбора признаков для временных рядов. Книга будет интересна специалистам по data science, регулярно решающим задачи с временными рядами. Для изучения материала желателен опыт работы на Python и базовые знания в области машинного обучения.

УДК 004.04
ББК 32.372

ISBN 978-5-93700-212-9

Оглавление

Предисловие

Временной ряд – это набор данных, собранных последовательно во времени. Например, представьте любую диаграмму, где ось абсцисс представляет собой некоторое измерение времени – что угодно, например количество звезд во Вселенной с момента Большого взрыва до сегодняшнего дня или количество энергии, высвобождаемой каждую наносекунду ядерной реакции. В обоих случаях данные представляют собой временные ряды. А как насчет графика в приложении «Погода» на вашем телефоне, показывающего ожидаемую температуру на следующие 7 дней? Это тоже график временного ряда.

В этой книге нас в основном будут интересовать события в человеческом масштабе – годы, месяцы, дни и часы, и все это будут данные временных рядов. Предсказывание будущих значений составляет суть прогнозирования.

Прогнозирование погоды, очевидно, было важно для людей на протяжении тысячелетий, особенно с момента появления сельского хозяйства. Фактически более 2300 лет назад греческий философ Аристотель написал трактат под названием «Метеорология», в котором обсуждались ранние прогнозы погоды. Само слово «прогноз» было придумано английским метеорологом в 1850-х годах Робертом Фитцроем, который прославился как капитан корабля «Бигль» во время новаторского путешествия Чарльза Дарвина.

Но данные временных рядов не ограничиваются погодой. Медицинская отрасль заимствовала методы анализа временных рядов, когда в 1901 году голландским врачом Виллемом Эйнтховеном была изобретена первая пригодная для практического применения электрокардиограмма. ЭКГ, как известно, воспроизводит хорошо известную всем нам картину сердечных сокращений, мы часто можем увидеть ее на аппарате рядом с кроватью пациента, наблюдая очередную медицинскую драму.

Сегодня одной из самых обсуждаемых областей прогнозирования является экономика. Есть целые телеканалы, посвященные анализу тенденций фондового рынка. Правительства используют экономическое прогнозирование для согласования политики центрального банка, политики используют экономическое прогнозирование для разработки своих платформ, а лидеры бизнеса используют экономическое прогнозирование для принятия решений.

В этой книге мы будем прогнозировать самые разные темы, такие как продажи продуктов в магазинах, уровень углекислого газа в атмосфере, количество велосипедистов, участвующих в программе общественного велосипедного проката в Чикаго, рост популяции волков в Йеллоустоне, циклы солнечных пятен, местные осадки и даже количество лайков в популярных аккаунтах Instagram.

Проблема с зависимыми данными

Итак, почему прогнозирование временных рядов требует собственного уникального подхода? Со статистической точки зрения вы можете построить то-

чечную диаграмму временных рядов с относительно четким трендом и попытаться подогнать линию, используя стандартную регрессию – метод построения прямой линии по данным. Проблема в том, что это нарушает предположение о независимости, которой требует линейная регрессия.

Проиллюстрируем зависимость временных рядов на конкретном примере. Предположим, что игрок бросает честную игральную кость. Я говорю вам, что он только что выбросил 2, и спрашиваю, каким будет следующее значение. Эти данные независимы: предыдущие броски не влияют на будущие броски, поэтому знание того, что предыдущим броском было 2, не дает никакой информации о следующем броске.

Однако в другой ситуации, скажем, я звоню вам из неизвестного места где-то на Земле и прошу угадать температуру в месте моего пребывания. Лучше всего было бы угадать среднюю глобальную температуру в этот день. А теперь представьте, что я говорю вам, что вчерашняя температура в месте моего пребывания была 90 °F. Это дает вам много информации, потому что вы интуитивно знаете, что вчерашняя температура и сегодняшняя температура каким-то образом связаны, эти данные уже не являются независимыми.

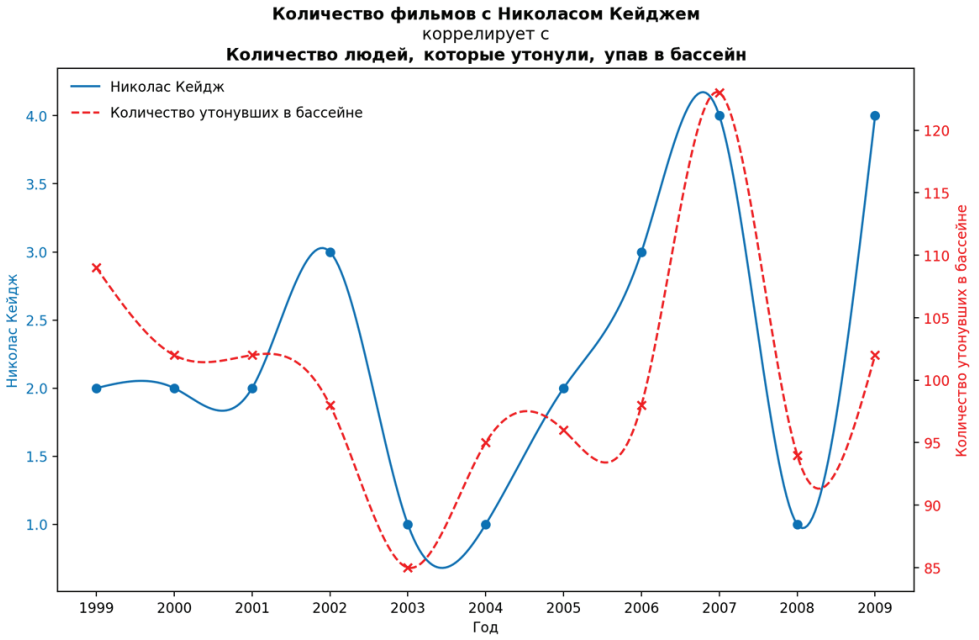
Имея данные временных рядов, вы не можете случайным образом перемешать данные, не нарушая тренда и оставаясь в пределах разумной погрешности. Порядок данных имеет значение, они не являются независимыми. Когда данные зависимы, как в нашем случае, регрессионная модель может показать статистическую значимость чисто случайно (даже если нет истинной корреляции) гораздо чаще, чем предполагает выбранный вами уровень достоверности.

Поскольку высокие значения, как правило, следуют за высокими значениями, а низкие значения, как правило, следуют за низкими значениями, набор данных временных рядов с большей вероятностью покажет больше кластеров высоких или низких значений, чем было бы в противном случае, а это, в свою очередь, может привести к появлению большего количества корреляций, чем было бы в противном случае.

Веб-сайт «Ложные корреляции» Тайлера Вигена специализируется на случаях, которые являются примерами, казалось бы, значимых, но совершенно нелепых корреляций во временных рядах. Вот один из примеров.

```
# импортируем необходимые библиотеки
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import random
from scipy.interpolate import make_interp_spline
%config InlineBackend.figure_format = 'retina'

# загружаем данные
df = pd.read_csv('Data/spurious_correlations.csv', sep=';')

# подготавливаем данные
x_new = np.linspace(df['Год'].min(),
                    df['Год'].max(), 300)
Nic_Cage_smooth = make_interp_spline(
    df['Год'],
    df['Количество фильмов с Николасом Кейджем'],
```

```
    k=3,
    bc_type='natural')(x_new)
pool_drownings_smooth = make_interp_spline(
    df['Год'],
    df['Количество людей, которые утонули, упав в бассейн'],
    k=3,
    bc_type='natural')(x_new)

df2 = pd.DataFrame(
    {'Год': x_new,
    'Количество фильмов с Николасом Кейджем': Nic_Cage_smooth,
    'Количество людей, которые утонули, упав в бассейн': pool_drownings_smooth})

# визуализируем данные
fig, ax1 = plt.subplots(figsize=(10, 6))
color = '#0072B2'
ax1.set_xlabel('Год')
ax1.set_ylabel('Николас Кейдж',
               color=color)
ax1.plot(df2['Год'],
         df2['Количество фильмов с Николасом Кейджем'],
         color=color,
         ls='-',
         label='Николас Кейдж')
ax1.scatter(df['Год'],
            df['Количество фильмов с Николасом Кейджем'],
            color=color,
            marker='o')
ax1.tick_params(axis='y', labelcolor=color)
ax1.legend(loc=(0.01, .94), frameon=False)
ax2 = ax1.twinx()

color = 'r'
ax2.set_ylabel('Количество утонувших в бассейне',
               color=color)
ax2.plot(df2['Год'],
         df2['Количество людей, которые утонули, упав в бассейн'],
         color=color,
         ls='--',
         label='Количество утонувших в бассейне')
ax2.scatter(df['Год'],
            df['Количество людей, которые утонули, упав в бассейн'],
            color=color,
            marker='x')
ax2.tick_params(axis='y', labelcolor=color)
ax2.legend(loc=(0.01, .89), frameon=False)

fig.tight_layout()
plt.title(r'$\bfКоличество\ фильмов\ с\ Николасом\ Кейджем$' +
          '\nкоррелирует с\n' +
          r'$\bfКоличество\ людей,\ которые\ утонули,\ упав\ в\ бассейн$')
plt.xticks(ticks=df['Год'].tolist(), labels=df['Год'].tolist())
plt.show()
```



Очевидно, что количество людей, которые тонут в бассейнах каждый год, совершенно не зависит от количества фильмов, в которых появляется Николас Кейдж. Они просто никак не влияют друг на друга. Однако Виген показал, что если совершить ошибку и рассматривать наблюдения временного ряда как независимые, по чистой случайности можно обнаружить, что два ряда данных действительно сильно коррелируют друг с другом. Такие случайные находки гораздо более вероятны при игнорировании зависимости данных временных рядов.

Теперь, когда вы понимаете, что такое данные временных рядов и что отличает их от других наборов данных, давайте рассмотрим несколько этапов в разработке прогнозных моделей, от самых ранних моделей до Prophet.

Скользящее среднее и экспоненциальное сглаживание

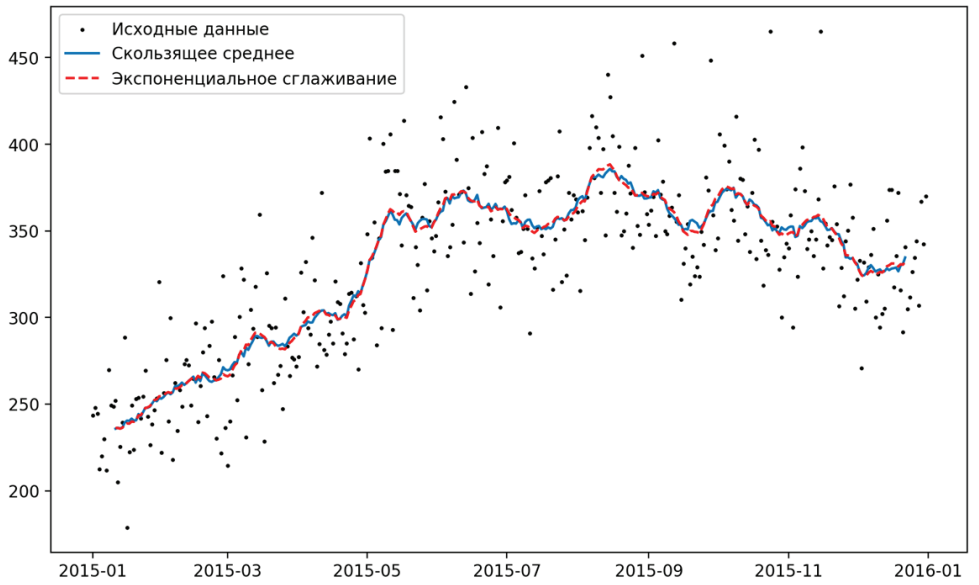
Возможно, самой простой формой прогнозирования является **скользящее среднее (moving average)**. Часто скользящее среднее используется в качестве **метода сглаживания (smoothing technique)**, чтобы найти более гладкую линию для данных с большим количеством вариаций. Каждую точку данных можно описать средним значением *n* окружающих точек данных, где *n* обозначает размер окна. Например, при размере окна 10 мы опишем точку данных так, чтобы ее значение представляло собой среднее 5 значений, предшествующих точке, и 5 значений, следующих после точки. При прогнозировании будущие значения рассчитываются как среднее *n* предыдущих значений, поэтому при размере окна 10 речь будет идти о среднем значении 10 предыдущих значений.

Суть сглаживания скользящим средним заключается в том, что вам нужен большой размер окна, чтобы сгладить шум и уловить фактический тренд, но

при большом размере окна ваши прогнозы будут значительно отставать от тренда, поскольку вы будете использовать все более ранние наблюдения для вычисления среднего. Идея экспоненциального сглаживания заключается в том, что мы применяем экспоненциально уменьшающиеся веса к усредняемым по времени значениям, придавая недавним значениям больший вес, а более ранним значениям – меньший. Это позволяет прогнозу быть более чувствительным к изменениям, игнорируя при этом значительное количество шума.

Как можно увидеть на следующем графике симулированных данных, линия скользящего среднего демонстрирует гораздо более грубое поведение, чем линия экспоненциального сглаживания, но обе линии по-прежнему адаптируются к изменениям тренда одновременно.

```
# загружаем и подготавливаем данные
data = pd.read_csv('Data/instagram_natgeo.csv')
data['created_time'] = pd.to_datetime(data['Date'])
df = data[(data['created_time'] >= '2015-01-01') &
          (data['created_time'] < '2016-01-01')][
    ['created_time', 'Average Likes Per Photo']]
df = df.groupby([df['created_time'].dt.date]).mean().reset_index()
df['likes'] = df['Average Likes Per Photo'] / 1000

# визуализируем данные
plt.figure(figsize=(10, 6))
plt.scatter(df['created_time'],
            df['likes'],
            label='Исходные данные',
            c='k',
            s=2)
plt.plot(df['created_time'],
         df['likes'].rolling(window=20, center=True).mean(),
         label='Скользящее среднее',
         ls='-',
         c='#0072B2')
plt.plot(df['created_time'],
         df['likes'].rolling(window=20,
                             center=True,
                             win_type='exponential').mean(tau=10),
         label='Экспоненциальное сглаживание',
         ls='--',
         c='r')
plt.legend()
plt.show()
```



Экспоненциальное сглаживание возникло в 1950-х годах на базе **простого экспоненциального сглаживания (simple exponential smoothing)**, которое не учитывает ни тренд, ни сезонность. Чарльз Холт усовершенствовал эту технику в 1957 году, чтобы она позволяла учитывать тренд, и назвал ее **двойным экспоненциальным сглаживанием (double exponential smoothing)**, а в сотрудничестве с Питером Винтерсом Холт добавил поддержку сезонности в 1960 году, техника получила название **«экспоненциальное сглаживание Холта–Винтерса» (Holt-Winters exponential smoothing)**, или **«тройное экспоненциальное сглаживание» (triple exponential smoothing)**.

Недостатком этих методов прогнозирования является то, что они могут медленно приспосабливаться к новым тенденциям, и поэтому прогнозируемые значения отстают от реальности – они плохо работают, когда требуются более длительные горизонты прогнозирования, и существует множество гиперпараметров для настройки, что может быть трудным и очень времязатратным процессом.

Линейная регрессия

Линейная регрессия, несмотря на свою простоту (сразу оговоримся: только кажущуюся), до сих пор успешно используется для прогнозирования временных рядов. Недостатком метода является неспособность линейной регрессии фиксировать нелинейные отношения и самостоятельно находить сложные взаимодействия высоких порядков.

ARIMA

В 1970 году математики Джордж Бокс и Гвилим Дженкинс опубликовали книгу «Time Series: Forecasting and Control», в которой описывался метод, сейчас из-

вестный как **модель Бокса–Дженкинса (Box-Jenkins model)**. Метод использовал идею скользящего среднего и получил свое развитие в ARIMA-модели. ARIMA и модель Бокса–Дженкинса часто используются как взаимозаменяемые термины, хотя технически модель Бокса–Дженкинса относится к методу оптимизации параметров для ARIMA-модели.

ARIMA – это аббревиатура трех понятий: **авторегрессия (Autoregressive – AR)**, **интегрированное (Integrated – I)** и **скользящее среднее (Moving Average – MA)**. Мы уже познакомились со скользящим средним. **Авторегрессия (Autoregressive)** означает, что модель использует зависимость между точкой данных и некоторым количеством запаздывающих точек данных (лагов). То есть модель делает прогнозы на основе предыдущих значений. Это похоже на предсказание того, что завтра будет тепло, потому что до сих пор было тепло всю неделю.

Интегрированное (Integrated) означает, что вместо применения любой исходной точки данных используется разница между этой точкой данных и некоторой предыдущей точкой данных. По сути, это означает, что мы преобразуем ряд значений в ряд изменений значений. Интуитивно это предполагает, что завтра будет более или менее такая же температура, как сегодня, потому что температура всю неделю сильно не менялась.

Каждый из компонентов AR, I и MA ARIMA-модели явно указывается как параметр в модели. Традиционно p используется как количество лагов, также известное как порядок лагов. Количество раз, с которым исходный ряд будет продифференцирован, или порядок дифференцирования, обозначается как d, а q представляет собой размер окна скользящего среднего (порядок скользящего среднего). Таким образом, возникает стандартное обозначение для ARIMA-модели $ARIMA(p, d, q)$, где p, d и q – все неотрицательные целые числа.

Проблема с ARIMA-моделями заключается в том, что они не поддерживают сезонность или данные с повторяющимися циклами, такими как повышение температуры днем и падение ночью или повышение летом и падение зимой. **SARIMA, или Seasonal ARIMA (сезонная ARIMA)**, была разработана для преодоления данного недостатка. Подобно нотации ARIMA, нотация для модели SARIMA представляет собой $SARIMA(p, d, q)(P, D, Q)m$, где P – порядок сезонной авторегрессии, D – порядок сезонного дифференцирования, Q – порядок сезонного скользящего среднего, а m – периодичность (количество периодов в полном сезонном цикле).

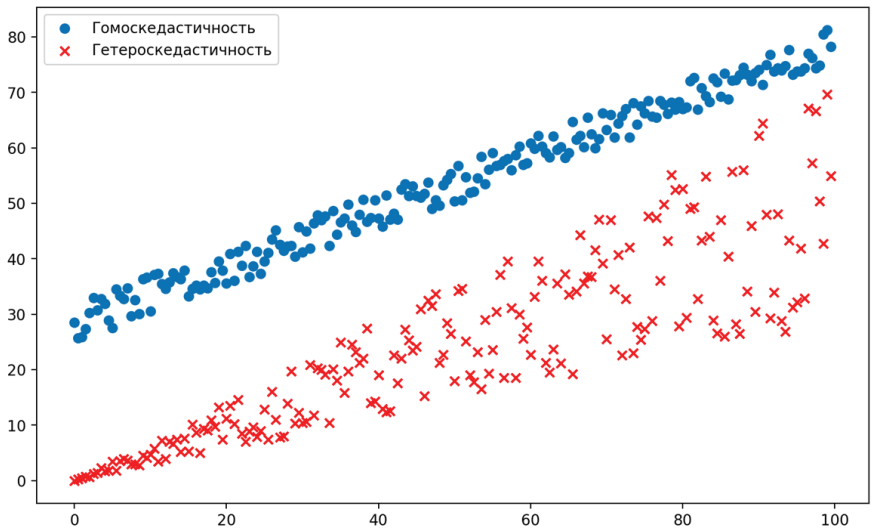
Вы также можете встретить другие варианты моделей ARIMA: **VARIMA (Vector ARIMA – векторная ARIMA**, для случаев с несколькими временными рядами в качестве векторов); **FARIMA (Fractional ARIMA – дробная ARIMA)** или **ARFIMA (Fractional Integrated ARIMA – дробная проинтегрированная ARMA)**, последние две включают дробную степень дифференцирования, обеспечивающую длительную память в том смысле, что наблюдения, удаленные друг от друга с точки зрения времени, могут иметь несущественные зависимости; **SARIMAX, сезонная ARIMA**, где X обозначает экзогенные или дополнительные переменные, добавленные в модель, например добавление прогноза осадков в модель прогнозирования температуры.

ARIMA обычно показывает очень хорошие результаты, но недостатком является сложность подбора параметров и необходимость тщательного разведочного анализа. Настройка и оптимизация моделей ARIMA часто требуют значительных вычислительных ресурсов, а успешные результаты могут зависеть от навыков и опыта прогнозиста. Метод лучше подходит для квалифицированных практиков.

ARCH/GARCH

Когда дисперсия данных не является постоянной во времени, ARIMA-модели сталкиваются с проблемами. В экономике и финансах непостоянство дисперсии может быть обычным явлением. Для ее анализа используют такой показатель, как волатильность. Волатильность характеризует изменчивость цены на что-либо. Например, криптовалюта демонстрирует высокую волатильность. Виртуальная валюта за короткий промежуток времени может резко вырасти в цене и так же резко упасть. Для решения этой проблемы были разработаны модели **авторегрессионной условной гетероскедастичности (Autoregressive Conditional Heteroscedasticity – ARCH)**. **Гетероскедастичность (heteroscedasticity)** – это способ сказать, что дисперсия или разброс данных не являются постоянными повсюду, а противоположным термином является **гомоскедастичность (homoscedasticity)**. Разница между гетероскедастичностью и гомоскедастичностью показана ниже.

```
# создаем данные
x = np.arange(0, 100, .5)
y1 = [25 + .5 * val + 7 * random.random() for val in x]
y2 = [.5 * val + 3 * (random.random() - .5) * val / 7 for val in x]

# визуализируем данные
plt.figure(figsize=(10, 6))
plt.scatter(x,
            y1,
            label='Гомоскедастичность',
            marker='o',
            c='#0072B2')
plt.scatter(x,
            y2,
            label='Гетероскедастичность',
            marker='x',
            c='r')
plt.legend()
plt.show()
```



Роберт Энгл представил первую ARCH-модель в 1982 году, описав условную дисперсию как функцию предыдущих значений. Например, существует гораздо больше неопределенности касательно использования электроэнергии в дневное время, чем в ночное. Таким образом, в модели потребления электроэнергии мы могли бы предположить, что электропотребление в ночные часы имеет меньшую дисперсию, чем в дневные часы.

Тим Боллерслев и Стивен Тейлор в 1986 году дополнили модель компонентой скользящего среднего, предложив модель Generalized ARCH, или GARCH. В примере с электричеством разница в потреблении была функцией от времени суток. Но, возможно, колебания волатильности не обязательно происходят в определенное время суток, колебания сами по себе случайны. Это как раз ситуация, когда GARCH-модель будет весьма полезна.

Обе модели ARCH и GARCH не могут обрабатывать ни тренд, ни сезонность, хотя на практике часто для работы с сезонными колебаниями и трендом временного ряда применяется ARIMA-модель, а затем для моделирования ожидаемой дисперсии используют ARCH-модель.

Градиентный бустинг

Градиентный бустинг сейчас стал популярен для прогнозирования временных рядов и при отсутствии какой-либо сложной предварительной подготовки показывает довольно неплохие результаты. Следует помнить, что деревья не умеют экстраполировать. Если наш временной ряд содержит тренд, можно попробовать детрендинг. Мы удаляем из ряда тренд, предварительно спрогнозированный линейной моделью. На полученных остатках обучаем градиентный бустинг и получаем прогнозы, к ним добавляем тренд. Если процедура детрендинга позволила повысить качество модели, можно попробовать в качестве признака добавить тренд. Преимуществом градиентного бустинга является способность фиксировать нелинейные зависимости и взаимодействия высоких порядков. Впрочем, полезно помнить старую шутку: «если данные содержат взаимодействия признаков, дерево решений обязательно найдет их; если данные не содержат никаких взаимодействий признаков, дерево решений все равно их обнаружит».

Нейронные сети

Относительно недавней разработкой в прогнозировании временных рядов является использование **рекуррентных нейронных сетей (Recurrent Neural Networks – RNN)**. Это стало возможным благодаря разработке Зеппом Хохрайтером и Юргеном Шмидхубером в 1997 году **долгой кратковременной памяти (long short-term memory – LSTM)**. По сути, LSTM позволяет нейронной сети обрабатывать последовательность данных типа речи или видеозаписи вместо одной точки данных, например изображения.

Стандартная рекуррентная нейронная сеть называется рекуррентной, потому что в нее встроены циклы (петли), которые и дают ей память, то есть дают доступ к предыдущей информации.

Базовую нейронную сеть можно обучить распознавать изображение пешехода на улице, изучая, как он выглядит на предыдущих изображениях, но ее нельзя обучить определять, что пешеход на видео скоро перейдет улицу, основываясь на приближении пешехода, наблюдаемом на предыдущих кадрах видео. Она не знает о последовательности изображений, которая приводит к тому, что пешеход выходит на дорогу. Кратковременная память – это то, что временно требуется сети для контекста, но эта память быстро истощается.

У ранних рекуррентных нейронных сетей была проблема с памятью: она просто была не очень длинной. В предложении *airplanes fly in the …* простая рекуррентная нейронная сеть может угадать, что следующим словом будет *sky*. Но в случае с предложением *I went to France for vacation last summer. That's why I spent my spring learning to speak …* рекуррентной нейронной сети не так-то просто догадаться, что следующим словом будет *French*. Она понимает, что дальше должно идти слово, связанное с языком, но забывает, что фраза начинается с упоминания Франции. Однако LSTM дает ему этот необходимый контекст. Это увеличивает долговечность кратковременной памяти сети. В случае временных рядов, где паттерны могут повторяться в течение длительного времени, LSTM потенциально могут работать очень хорошо.

Прогнозирование временных рядов с помощью LSTM все еще находится в зачаточном состоянии по сравнению с другими методами прогнозирования, обсуждаемыми здесь, тем не менее он подает надежды. Как и в случае с градиентным бустингом, одним из сильных преимуществ перед другими методами прогнозирования является способность нейронных сетей фиксировать нелинейные отношения и сложные взаимодействия высоких порядков. Впрочем, шутку про дерево можно отнести и к нейронным сетям. Кроме того, как и в случае с любой моделью глубокого обучения, LSTM требует большого объема данных, значительной вычислительной мощности и долгого времени обработки.

Наконец, необходимо принять множество решений относительно архитектуры модели и используемых гиперпараметров, что требует наличия очень опытного специалиста по прогнозированию. В большинстве практических задач, где необходимо учитывать бюджет и сроки, ARIMA-модель (при наличии должной квалификации), линейная модель и градиентный бустинг часто являются лучшим выбором.

В книге мы рассмотрим библиотеки ETNA и Sktime, являющиеся обертками над уже имеющимися классами библиотек statsmodels scikit-learn, catboost, lightgbm, xgboost, pytorch forecasting, в которых реализованы вышеупомянутые методы, и предлагающие некоторые собственные инструменты (работа с иерархическими временными рядами, получение прогнозов с помощью стратегий многошагового прогнозирования, классификация и кластеризация временных рядов), и библиотеки Facebook Prophet и Linkedin Greykite, предлагающие собственную математическую модель для прогнозирования рядов.

Часть 1

Библиотека Facebook Prophet

1. Математический аппарат Facebook Prophet (на основе адаптированного перевода статьи «Forecasting at scale» от Шона Дж. Тейлора и Бенджамина Летама, входящих в команду разработчиков Facebook Prophet)

1.1. Введение

Прогнозирование – это задача науки о данных, которая является центральной для многих видов деятельности внутри коммерческой организации.

Например, организации всех отраслей промышленности должны участвовать в планировании производственных мощностей для эффективного распределения ограниченных ресурсов и постановки целей, чтобы измерить собственную эффективность относительно некоторого базового уровня. Генерация качественных прогнозов – непростая задача как для машин, так и для большинства аналитиков. Мы отметили две основные проблемы, касающиеся получения бизнес-прогнозов. Во-первых, полностью автоматизированные методы прогнозирования могут быть трудно настраиваемыми и часто являются слишком негибкими, чтобы включать полезные предположения или эвристики. Во-вторых, аналитики, решающие задачи науки о данных в организации, обычно имеют глубокие знания предметной области касательно конкретных продуктов или услуг, которые они продают или оказывают, но часто не проходят обучение прогнозированию временных рядов. Аналитики, умеющие получать качественные прогнозы, таким образом, довольно редки, потому что прогнозирование – это специализированный навык, требующий значительного опыта.

В результате спрос на качественные прогнозы часто намного превышает время, в течение которого они могут быть получены. Этот вывод послужил стимулом для исследования, которое мы представили здесь. Мы намерены дать некоторые полезные рекомендации по составлению *масштабируемых* прогнозов, используя несколько аспектов масштабируемости.

Первые два аспекта масштабируемости, к которым мы обратились, заключаются в том, что методы бизнес-прогнозирования должны подходить для:

1) большого количества людей, которые делают прогнозы и при этом не прошли обучение прогнозированию временных рядов;
2) большого количества разнообразных задач прогнозирования с особенностями, характерными только для конкретной рассматриваемой задачи.

В разделе 2.3 мы предложим модель временных рядов Prophet, которая является достаточно гибкой для широкого диапазона временных рядов в бизнесе. При этом ее могут настроить неспециалисты, у которых есть знание предметной области, позволяющее судить о процессе генерации данных, но при этом у них мало знаний о моделях и методах прогнозирования временных рядов.

Третий аспект масштабируемости, к которому мы обратились, заключается в том, что в наиболее реалистичных условиях будет сгенерировано большое количество прогнозов, что потребует эффективных, автоматизированных средств их оценки и сравнения, а также выявления ситуаций, когда они, вероятно, будут работать плохо. Когда получены сотни или даже тысячи прогнозов, становится важным позволить машинам выполнять тяжелую работу по оценке и сравнению моделей, эффективно используя обратную связь человека для устранения проблем с качеством модели. В разделе 2.4 мы опишем систему оценки прогнозов, которая использует смоделированные исторические прогнозы для оценки показателей вне обучающей выборки и выявления проблемных прогнозов, чтобы понять, что пошло не так, и сделать необходимые корректировки модели.

Стоит отметить, что мы не акцентировали внимание на типичных аспектах масштабируемости: вычислении и хранении. Мы обнаружили, что вычислительные и инфраструктурные проблемы прогнозирования большого количества временных рядов являются относительно простыми – обычно эти процедуры обучения довольно легко распараллеливаются и прогнозы несложно хранить в реляционных базах данных. Актуальные проблемы масштабирования, которые мы наблюдали на практике, связаны со сложностью, вызванной разнообразием задач прогнозирования и необходимостью повысить доверие к большому количеству прогнозов после их получения.

Мы резюмировали наш подход analyst-in-the-loop («аналитик в курсе дел») к масштабируемому бизнес-прогнозированию на рис 1. Мы начинаем с того, что моделируем временные ряды с использованием гибкой спецификации, в которой каждый параметр имеет простую человеческую интерпретацию. Затем мы получаем прогнозы, а также значения метрик для диапазона прогнозируемых дат и оцениваем качество прогнозов. Когда наблюдается низкое

качество или какие-то аспекты прогнозов требуют вмешательства человека, мы предъявляем эти потенциальные проблемы аналитику в приоритетном порядке. Затем аналитик может проверить прогноз и потенциально скорректировать модель на основе этой обратной связи.

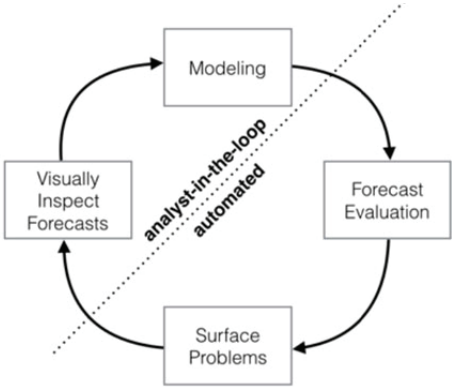


Рис. 1 Схематическое изображение подхода analyst-in-the-loop к масштабируемому прогнозированию, который лучше всего соответствует задачам, решаемым вручную человеком и автоматическим способом

1.2. Особенности временных рядов в бизнес-задачах

Существует множество задач бизнес-прогнозирования, но есть некоторые особенности, характерные для многих из них. На рис. 2 ниже показаны характерные временные ряды Facebook Events. Пользователи Facebook могут применять платформу Events для создания страниц мероприятий, приглашений других людей и взаимодействия с событиями различными способами. На рис. 2 представлены ежедневные данные по количеству событий, созданных в Facebook. Здесь отчетливо видно, что временной ряд содержит несколько сезонных эффектов: недельный и годовой циклы, а также ярко выраженный спад к Рождеству и Новому году. Эти типы сезонных эффектов возникают естественным образом, и их можно ожидать во временных рядах, полученных в результате человеческих действий. Временной ряд также показывает четкое изменение тренда в последние шесть месяцев, которое могло возникнуть во временном ряду под влиянием новых продуктов или изменений рынка. Наконец, настоящие наборы данных часто имеют выбросы, и этот временной ряд не является исключением.

Рис. 2 Количество событий, созданных на Facebook. Точка соответствует дню, и точки имеют цветовую кодировку по дням недели, чтобы показать недельный цикл. Особенности этого временного ряда являются характерными для многих временных рядов: несколько сильных сезонностей, изменения тренда, выбросы и праздничные эффекты

Этот временной ряд служит полезной иллюстрацией трудностей в получении разумных прогнозов с помощью полностью автоматизированных методов. На рис. 3 следующего слайда показаны прогнозы с использованием нескольких автоматизированных процедур из пакета R forecast, описанного в Hyndman et al. (2007). Прогнозы были получены по трем отрезкам времени, для обучения модели использовалась только часть временного ряда до первой точки соответствующего отрезка. На рис. 3 показаны следующие методы:

- *auto.arima*, который обучает набор моделей ARIMA и автоматически выбирает лучшую;
- *ets*, который обучает набор моделей экспоненциального сглаживания и выбирает лучшие (Hyndman и др. 2002);
- *snaive*, модель случайного блуждания, которая делает константные прогнозы, используя недельную сезонность (модель сезонного наивного прогноза);
- *tbats*, модель TBATS с недельной и годовой сезонностью (De Livera и др. 2011).

Методы, показанные на рис. 3 ниже, обычно не позволяют получить прогнозы, соответствующие характеристикам этих временных рядов. Прогнозы автоматизированной ARIMA склонны к большим ошибкам тренда, когда есть изменение тренда вблизи периода, по которому происходит отсечка, и они не могут уловить какую-либо сезонность. Экспоненциальное сглаживание и сезонные наивные прогнозы отражают недельную сезонность, но не учитывают более долгосрочную сезонность. Все методы слишком остро реагируют на падение в конце года, потому что они неадекватно моделируют годовую сезонность.

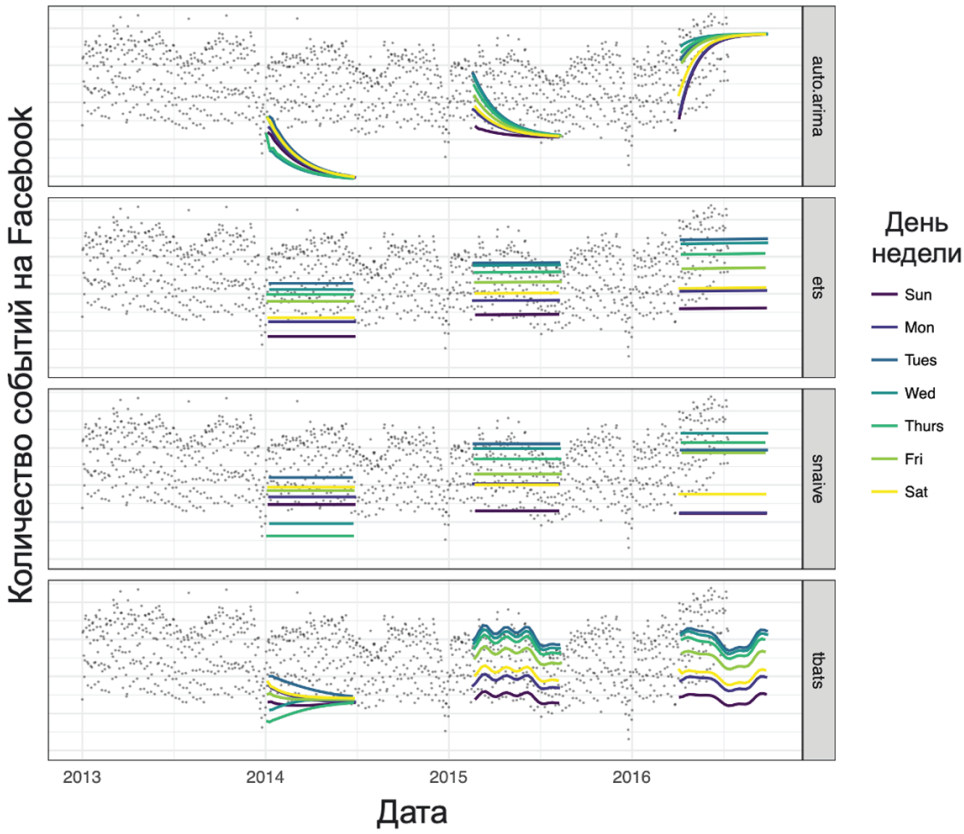


Рис. 3 Прогнозы по временным рядам из предыдущего рисунка с использованием набора автоматизированных процедур прогнозирования. Прогнозы генерировались по трем отрезкам времени, модели использовали только часть временного ряда до первой точки соответствующего отрезка. Прогнозы для каждого дня недели сгруппированы и раскрашены по дням недели для визуализации еженедельной сезонности. Мы удалили выбросы при отрисовке графиков, чтобы оставить больше места на рисунке по вертикали

В случае получения плохих прогнозов мы хотим получить возможность настроить параметры метода, который призван улучшить прогнозы. Настройка этих методов требует глубокого понимания того, как работают модели, лежащие в основе временных рядов. Например, в автоматизированной ARIMA речь пойдет о порядке дифференцирования, порядке авторегрессии, порядке скользящего среднего. Типичный аналитик не знает, как настроить эти параметры, чтобы избежать ситуации, показанной на этом рисунке, – это тот тип экспертизы, который трудно масштабировать.

1.3. Прогнозная модель Prophet

Теперь мы опишем модель прогнозирования временных рядов, предназначенную для обработки общих характеристик временных рядов, показанных

на рисунках ранее. Важно отметить, что у нее есть интуитивно понятные параметры, которые можно настроить, не зная деталей базовой модели. Они необходимы аналитику для эффективной спецификации модели. Наша реализация доступна в виде программного обеспечения с открытым исходным кодом на Python и R под названием Prophet (*https://facebook.github.io/prophet/*).

Мы используем модель разложимых временных рядов (Harvey & Peters 1990) с тремя основными компонентами: тренд, сезонность и праздники. Они объединены в следующее уравнение:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t. \quad (1)$$

Здесь $g(t)$ – это функция тренда, моделирующая непериодические изменения значений временного ряда, $s(t)$ представляет собой периодические изменения (например, еженедельную и ежегодную сезонность), а $h(t)$ представляет собой эффекты праздников, которые возникают в течение одного или нескольких дней. Член ошибки ε_t представляет собой любые случайные возмущения, которые не учитываются моделью, позже мы выдвинем предположение о том, что ε_t имеет нормальное распределение.

Эта спецификация аналогична обобщенной аддитивной модели (GAM) (Hastie & Tibshirani 1987), классу регрессионных моделей с потенциально нелинейными сглаживаниями, применяемыми к регрессорам. Здесь мы используем только время в качестве регрессора, но возможно использование нескольких линейных и нелинейных функций времени в качестве компонент.

Моделирование сезонности как аддитивной компоненты – это тот же подход, что применяется в экспоненциальном сглаживании (Gardner 1985). Мультипликативная сезонность, где сезонный эффект – это множитель, умноженный на $g(t)$, может быть получена с помощью логарифмического преобразования.

Выбор в пользу GAM имеет то преимущество, что она легко декомпозируется и допускает включение новых компонент по мере необходимости, например при выявлении нового источника сезонности. GAM также обучается очень быстро, либо с помощью бэкфиттинга, либо с помощью L-BFGS (Byrd et al. 1995) (мы предпочитаем последнее), чтобы пользователь мог интерактивно изменять параметры модели.

По сути, мы формулируем проблему прогнозирования как задачу подгонки кривой, которая внутренне отличается от моделей временных рядов, поскольку те явно учитывают структуру временной зависимости в данных. Хотя мы отказываемся от некоторых важных преимуществ использования генеративной модели типа ARIMA, выбор в пользу GAM обеспечивает ряд практических преимуществ:

- гибкость, заключающуюся в том, что мы можем легко учесть сезонность с несколькими периодами и позволить аналитикам выдвинуть разные предположения о трендах;
- в отличие от моделей ARIMA, измерения не должны находиться на одинаковом расстоянии друг от друга и нам не нужно интерполировать пропущенные значения, возникшие, например, по причине удаления выбросов;
- обучение выполняется очень быстро, что позволяет аналитику интерактивно исследовать большое количество спецификаций модели;

- прогнозная модель имеет легко интерпретируемые параметры, которые аналитик может изменить согласно выдвинутым предположениям относительно прогнозов. Более того, аналитики обычно имеют опыт работы с регрессией и легко могут расширить модель, включив в нее новые компоненты.

Автоматическое прогнозирование имеет долгую историю, и многие методы адаптированы к конкретным типам временных рядов (Tashman & Leach 1991, De Gooijer & Hyndman 2006). Наш подход обусловлен как характером временных рядов, которые мы прогнозируем в Facebook (кусочные тренды, множественная сезонность, плавающие праздники), так и проблемами, связанными с масштабируемым прогнозированием.

1.3.1. Модель тренда

Мы реализовали две модели тренда, которые применяются во многих задачах Facebook: кусочно-логистическую модель роста с насыщением и кусочно-линейную модель.

1.3.1.1. Нелинейный рост с насыщением

Для прогнозирования роста основной компонентой процесса генерации данных является модель, предсказывающая, как выросла численность населения и как она будет расти дальше. Моделирование роста в Facebook обычно похоже на рост населения в естественных экосистемах (например, Hutchinson 1978), где наблюдается нелинейный рост, который насыщается при предельной пропускной способности. Например, предельной пропускной способностью для количества пользователей Facebook в определенном регионе может быть количество людей, имеющих доступ к сети Интернет. Такой рост обычно моделируется с помощью логистической модели роста, которая имеет вид

$$g(t) = \frac{C}{1 + \exp\left(-k\left(t - m\right)\right)}, \tag{2}$$

где C – верхний порог (пропускная способность), k – скорость роста, а m – параметр смещения, позволяющий «сдвигать» функцию вдоль оси времени.

Есть два важных аспекта модели роста в Facebook, которые не отражены в уравнении (2).

Во-первых, предельная пропускная способность непостоянна, поскольку количество людей в мире, которые имеют доступ к сети Интернет, увеличивается. Таким образом, мы заменяем фиксированную пропускную способность C на изменяющуюся во времени пропускную способность $C(t)$. Во-вторых, скорость роста непостоянна. Выход новых продуктов может существенно изменить темпы роста в регионе, поэтому модель должна включать в себя изменяющуюся скорость для соответствия историческим данным.

Мы включаем изменения тренда в модель роста, явно определяя точки изменения (change points), в которых скорость роста может измениться. Предположим, что существует S точек изменений в моменты времени $s_j, j = 1, \ldots, S$. Определим вектор корректировки скорости роста $\boldsymbol{\delta} \in \mathbb{R}^S$, где δ_j – изменение скоро-

сти, происходящее в момент времени s_j. Тогда скорость в любой момент t равна базовой скорости k плюс все корректировки до этой точки: $k + \sum_{j:t>s_j} \boldsymbol{\delta}_j$. Более наглядно это можно представить с помощью такого вектора $\mathbf{a}(t) \in \{0,1\}^S$, что

$$a_j(t) = \begin{cases} 1, \textit{ если } t \geq s_j, \\ 0, \textit{ в остальных случаях}. \end{cases}$$

Тогда скорость в момент времени t равна $k + \boldsymbol{a}(t)^{\mathrm{T}} \boldsymbol{\delta}$. При изменении скорости k параметр смещения m тоже нужно изменить, чтобы обеспечить гладкий стык сегментов кривой тренда для соответствующей временной отметки. Правильная корректировка в точке изменения j легко вычисляется как

$$\gamma_j = \left(s_j - m - \sum_{l<j} \gamma_l \right) \left(1 - \frac{k + \sum_{l<j} \delta_l}{k + \sum_{l \leq j} \delta_l} \right).$$

Тогда кусочно-логистическая модель роста принимает вид:

$$g(t) = \frac{C(t)}{1 + \exp\left(\left(-\left(k + \mathbf{a}(t)^{\mathrm{T}} \boldsymbol{\delta} \right) \left(t - (m + \mathbf{a}(t)^{\mathrm{T}} \boldsymbol{\gamma} \right) \right) \right)}. \tag{3}$$

Важный параметр в нашей модели – $C(t)$, или ожидаемая пропускная способность системы в любой момент времени. По сути, $C(t)$ – это функция верхней границы тренда. У аналитиков, как правило, есть представление о размерах рынка, и они могут определять пропускную способность согласно этим размерам. Кроме того, могут быть внешние источники данных, которые позволяют оценить пропускную способность, например прогнозы численности населения от Всемирного банка.

Представленная здесь логистическая модель роста является особым случаем кривых обобщенной логистической модели роста, который представляет собой лишь отдельный тип сигмоиды. Распространение этой модели тренда на другие семейства кривых является довольно простым.

1.3.1.2. Линейный тренд с точками изменения

При прогнозировании задач, в которых нет роста с насыщением, кусочно-постоянная скорость роста дает экономную и часто полезную модель. В таком случае модель тренда выглядит следующим образом:

$$g(t) = \left(k + \mathbf{a}(t)^{\mathrm{T}} \boldsymbol{\delta} \right) t + \left(m + \mathbf{a}(t)^{\mathrm{T}} \gamma \right), \tag{4}$$

где, как и прежде, k – скорость роста, $\boldsymbol{\delta}$ содержит корректировки скорости, m – параметр смещения, а γ_j устанавливается равной $-s_j \delta_j$, чтобы функция была непрерывной.

1.3.1.3. Автоматический отбор точек изменения

Точки изменения s_j могут быть указаны аналитиком с использованием известных дат запуска продукта и других событий, влияющих на рост, или могут быть

автоматически выбраны с помощью набора значений. Автоматический отбор можно выполнить вполне естественным путем с помощью формул (3) и (4), выбрав априорное распределение Лапласа[1] для $\boldsymbol{\delta}$.

Мы указываем большое количество точек изменения (например, по одной в месяц для нескольких лет исторических данных) и используем $\delta_j \sim \text{Laplace}(0, \tau)$. Параметр τ напрямую контролирует гибкость модели при изменении ее скорости. Что немаловажно, применение распределения Лапласа для корректировок $\boldsymbol{\delta}$ не влияет на первоначальную скорость роста τ, поэтому когда τ стремится к 0, обучение сводится к стандартному (а не кусочному) логистическому или линейному росту.

1.3.1.4. Неопределенность прогноза тренда

Когда происходит экстраполяция модели за пределы исторических данных для получения прогноза, тренд получает постоянную скорость. Мы оцениваем неопределенность прогноза тренда, распространяя генеративную модель на будущее. Генеративная модель тренда состоит в том, что существует S точек изменения на основе истории, состоящей из T точек. Каждая из этих точек имеет изменение скорости $\delta_j \sim \text{Laplace}(0, \tau)$. Параметр τ – коэффициент масштаба для автоматического выбора точек смены тренда. Мы симулируем значения будущих изменений скорости, которые подражают прошлым значениям путем замены τ на дисперсию, оцениваемую по имеющимся данным. Это можно сделать, применяя теорему Байеса, установив случайную величину с априорным распределением в качестве τ, чтобы получить апостериорную вероятность, в противном случае мы можем использовать оценку максимального правдоподобия для изменения скорости: $\lambda = \frac{1}{S}\sum_{j=1}^{S}\left|\delta_j\right|$.

Будущие точки изменения выбираются случайным образом так, чтобы средняя частота точек изменения соответствовала средней частоте точек изменения в исторических данных:

$$\forall_j > T, \begin{cases} \delta_j = 0 \ \text{ с вероятностью } \ \frac{T-S}{T}, \\ \delta_j \sim \text{Laplace}\left(0,\ \lambda\right) \text{ с вероятностью } \ \frac{S}{T}. \end{cases}$$

Таким образом, мы измеряем неопределенность прогнозируемого тренда, предполагая, что в будущем у нас будет та же самая средняя частота и величина изменений скорости, которые мы наблюдали в исторических данных. После того как λ получен на основе данных, мы используем эту генеративную модель для моделирования возможных будущих трендов и смоделированные тренды для вычисления доверительных интервалов.

Предположение, что тренд продолжит меняться с той же частотой и скоростью изменений, что и в исторических данных, является довольно сильным, поэтому мы не ожидаем высокой точности от доверительных интервалов. Однако они являются полезным показателем уровня неопределенности и в осо-

[1] Вспомним, что распределение Лапласа называют еще двойным экспоненциальным распределением.

бенности показателем переобучения. По мере увеличения τ модель становится все более гибкой, обучаясь на исторических данных, поэтому ошибки обучения уменьшаются. Однако при прогнозировании будущего эта гибкость приведет к широким доверительным интервалам.

1.3.2. Сезонность

Временные ряды в бизнес-задачах часто имеют многопериодную сезонность как результат человеческого поведения, которое они отражают. Например, 5-дневная рабочая неделя может оказывать влияние на временной ряд, повторяющееся каждую неделю, в то время как графики отпусков и школьных каникул могут вызывать эффекты, повторяющиеся каждый год. Чтобы моделировать и прогнозировать эти эффекты, мы должны задать модели сезонности, которые являются периодическими функциями t.

Мы предложили использовать ряды Фурье, чтобы получить гибкую модель периодических изменений (Harvey & Shephard 1993). Пусть P – постоянное значение периода для рассматриваемого временного ряда (например, P = 365,25 для годовых данных или P = 7 для недельных данных, если мы выражаем нашу переменную времени с помощью дней). Мы можем аппроксимировать произвольные сезонные эффекты с помощью стандартного ряда Фурье:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P} \right) + b_n \sin\left(\frac{2\pi nt}{P} \right) \right).$$

Подгонка сезонности требует оценки $2N$ параметров $\beta = [a_1, b_1, \ldots, a_N, b_N]^T$. Это делается путем построения матрицы векторов сезонности для каждого значения t в наших исторических и прогнозных данных, например ниже приведен пример для годовой сезонности и N= 10.

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{365.25} \right), \ldots, \sin\left(\frac{2\pi(10)t}{365.25} \right) \right]. \quad (5)$$

Тогда сезонная компонента будет иметь вид

$$s(t) = X(t)\beta. \quad (6)$$

В нашей генеративной модели мы берем $\beta \sim$ Normal(0, σ^2), чтобы сгладить сезонность.

Для годовой и недельной сезонностей были найдены значения N = 10 и N = 3 соответственно, которые работают достаточно хорошо для большинства задач. Выбор этих параметров можно автоматизировать с помощью критерия отбора модели типа AIC.

1.3.3. Праздники и события

Праздники и события создают большие, в некоторой степени предсказуемые потрясения («шоки») для многих временных рядов и часто не имеют

периодичности, поэтому их влияние нельзя хорошо смоделировать с помощью плавного цикла. Например, День благодарения в США отмечается в четвертый четверг ноября. Суперкубок, одно из крупнейших событий, транслируемых по телевидению в США, происходит по воскресеньям в январе или феврале, которое сложно программно объявить. Во многих странах мира главные праздники следуют лунному календарю. Влияние того или иного праздника на временной ряд часто бывает одинаковым из года в год, поэтому важно включить его в прогноз.

Аналитик может задать настраиваемый список прошлых и будущих событий, идентифицируемых по уникальному названию события или праздника, как показано в таблице справа. У нас есть столбец для страны, чтобы сохранить список праздников для конкретной страны, помимо глобальных праздников. Для приведенной задачи прогнозирования мы используем объединенный набор глобальных праздников и праздников, специфических для конкретной страны.

Таблица 1 Примерный список праздников. В отдельном столбце указана страна, поскольку праздники могут встречаться в разные дни в разных странах

Holiday	Country	Year	Date
Thanksgiving	US	2015	26 Nov 2015
Thanksgiving	US	2016	24 Nov 2016
Thanksgiving	US	2017	23 Nov 2017
Thanksgiving	US	2018	22 Nov 2018
Christmas	*	2015	25 Dec 2015
Christmas	*	2016	25 Dec 2016
Christmas	*	2017	25 Dec 2017
Christmas	*	2018	25 Dec 2018

Включение этого списка праздников в модель упрощается, если предположить, что эффекты праздников независимы. Для каждого праздника i вводится D_i – набор прошлых и будущих дат этого праздника. Мы добавляем функцию-индикатор, проверяющую, является ли момент времени t событием i, и назначаем каждому событию i параметр k_i, который является соответствующим изменением прогноза. Это достигается тем же путем, что и в части с сезонной составляющей, путем создания матрицы регрессоров

$$Z(t) = \left[\mathbf{1}(t \in D_1), \ldots, \mathbf{1}(t \in D_L)\right]$$

и ввода ее в модель следующим образом:

$$h(t) = Z(t)\boldsymbol{\kappa}. \tag{7}$$

Как и в случае с сезонностью, мы используем $\boldsymbol{\kappa} \sim \text{Normal}(0, \nu^2)$. Здесь ν – это коэффициент масштаба праздничных дней. Чем он больше, тем сильнее праздничные дни в будущем будут влиять на прогноз.

Часто бывает важно включить эффект, создаваемый днями, хронологически расположенными рядом с определенным праздником, например выходные в День благодарения в США, первые две январские недели после Нового года в России. Чтобы учесть это, мы задаем дополнительные параметры для дней, хронологически расположенных рядом с праздником, по сути рассматривая каждый из дней в окне вокруг праздника как такой же праздник.

1.3.4. Складываем все вместе

Подытожим вышесказанное. В основе методологии прогнозирования Prophet лежит процедура обучения обобщенной аддитивной модели следующего вида:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t,$$

где $g(t)$ и $s(t)$ – функции, аппроксимирующие тренд ряда и сезонные колебания (например, годовые, недельные и т. п.) соответственно, $h(t)$ – функция, отражающая эффекты праздников и других влиятельных событий, а ε_t – нормально распределенные случайные возмущения.

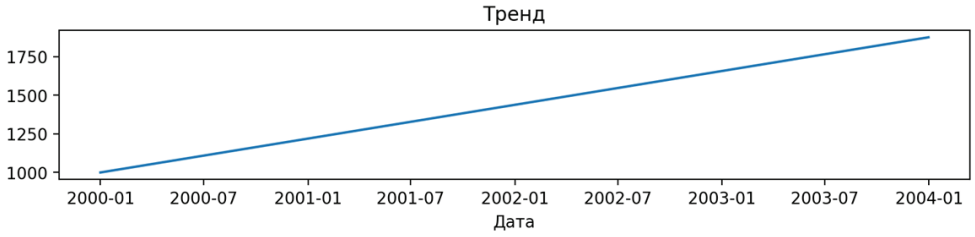
Для аппроксимации перечисленных функций используются следующие методы:

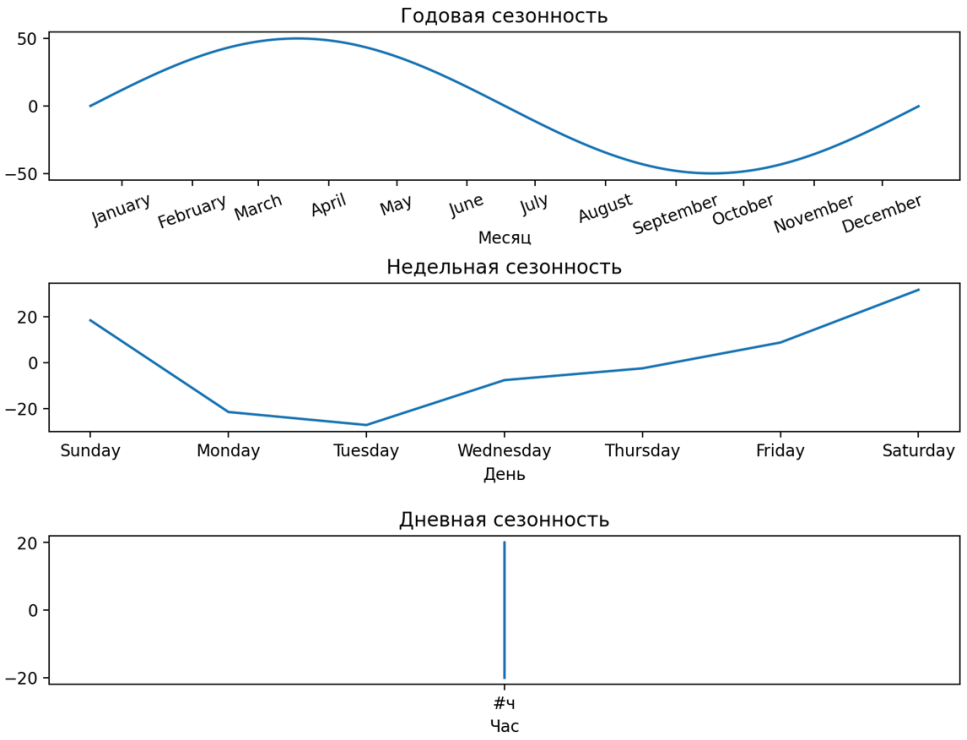
- тренд: кусочно-логистическая модель роста и кусочно-линейная модель;
- годовая и недельная сезонность: ряды Фурье;
- праздники и влиятельные события: индикаторные функции.

В качестве конкретного примера, иллюстрирующего суть обобщенной аддитивной модели, предположим, что у нас есть данные о продажах небольшого розничного интернет-магазина за четыре года, с 1 января 2000 года до конца 2003 года. Мы видим, что общий тренд постоянно увеличивается с течением времени от **1000** продаж в день примерно до **1800** в конце периода наблюдения. Мы также видим, что продажи весной примерно на **50** единиц выше среднего, а продажи осенью примерно на **50** единиц ниже среднего. Каждую неделю наблюдается следующая картина: продажи достигают максимального спада во **вторник** и растут в течение недели, достигая пика в субботу. Наконец, в течение дня продажи достигают пика в полдень и плавно падают до минимума в полночь. Вот как будут выглядеть эти отдельные кривые (обратите внимание на разные масштабы оси x на каждой диаграмме).

```
# создаем диапазон дат
x = pd.date_range('2000-01-01', '2003-12-31', freq='H')

# создаем линию тренда
y0 = [1000 + .025 * val for val in range(len(x))]

# добавляем годовые синусоидальные колебания
y1 = [50 * np.sin(idx * (360 / (365.25 * 24)) * (np.pi / 180))
      for idx in range(len(y0))]

# добавляем недельные колебания
y2 = [(25 * np.sin(((idx / 7) - 17) * (360 / 24) * (np.pi / 180))) +
      (10 * np.sin(((idx / 7) - 5) * (360 / 12) * (np.pi / 180))) +
```

```
      (25 * np.sin(((idx / 7) - 12) * (360 / 12) * (np.pi / 180))) +
      (15 * np.sin(((idx / 7) - 20) * (360 / 12) * (np.pi / 180)))
      for idx in range(len(x))]

# добавляем дневные синусоидальные колебания
y3 = [20 * np.sin((idx - 6) * (360 / 24) * (np.pi / 180))
      for idx in range(len(y2))]

# создаем аддитивные кривые
z0 = y0
z1 = [y0[idx] + y1[idx] for idx in range(len(x))]
z2 = [y0[idx] + y1[idx] + y2[idx] for idx in range(len(x))]
z3 = [y0[idx] + y1[idx] + y2[idx] + y3[idx] for idx in range(len(x))]

# визуализируем компоненты по отдельности
fig, ax = plt.subplots(4, 1, figsize=(10, 10))
plt.subplots_adjust(hspace=.7)

ax[0].set_title('Тренд')
ax[0].plot(x, y0)
ax[0].set_xlabel('Дата')

ax[1].set_title('Годовая сезонность')
ax[1].plot(x[:24 * 365], y1[:24 * 365])
ax[1].set_xticks([ts for ts in x[:24 * 365]
                  if ts.hour == 0 and ts.day == 15])
ax[1].set_xticklabels([ts.month_name()
                       for ts in x[:24 * 365]
                       if ts.hour == 0 and ts.day == 15],
                      rotation=20)
ax[1].set_xlabel('Месяц', labelpad=-5)

ax[2].set_title('Недельная сезонность')
ax[2].plot([ts.day_name() for ts in x[24: 24 * 8]
            if ts.hour == 12],
           [y2[idx] for idx in range(len(x[24: 24 * 8]))
            if x[idx].hour == 23])
ax[2].set_xlabel('День')

ax[3].set_title('Дневная сезонность')
ax[3].plot(x[:24].strftime('%#ч'), y3[:24])
ax[3].set_xlabel('Час')

plt.show()
```

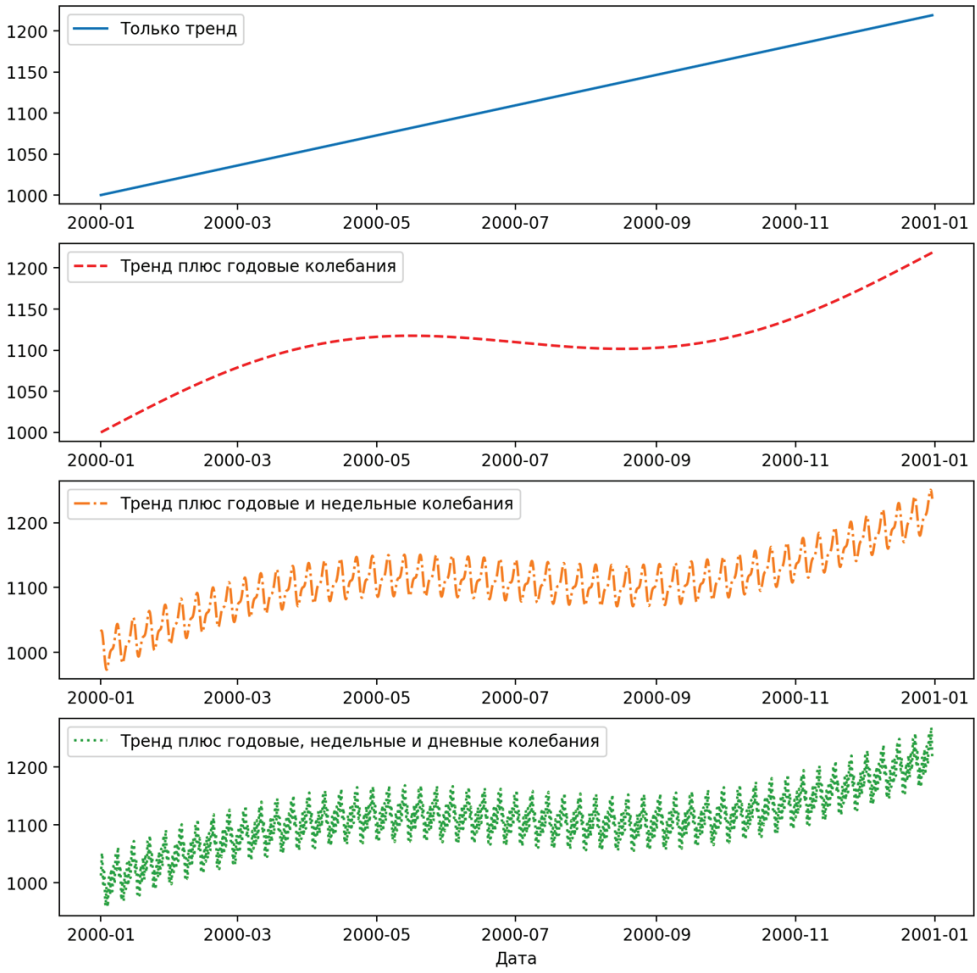




Аддитивная модель возьмет эти четыре кривые и просто сложит их друг с другом, чтобы получить окончательную модель продаж. Итоговая кривая становится все более и более сложной по мере добавления компонент.

```
# визуализируем процесс складывания компонент
fig, ax = plt.subplots(4, 1, figsize=(10, 10))

ax[0].plot(x[:8760],
           z0[:8760],
           label='Только тренд',
           c='#0072B2',
           ls='-')
ax[0].legend()

ax[1].plot(x[:8760],
           z1[:8760],
           label='Тренд плюс годовые колебания',
           c='r',
           ls='--')
ax[1].legend()

ax[2].plot(x[:8760],
           z2[:8760],
           label='Тренд плюс годовые и недельные колебания',
           c='tab:orange',
           ls='-.')
ax[2].legend()
```

```
ax[3].plot(x[:8760],
           z3[:8760],
           label='Тренд плюс годовые, недельные и дневные колебания',
           c='tab:green',
           ls=':')
ax[3].legend()
ax[3].set_xlabel('Дата')

plt.show()
```



На предыдущем графике показан только первый год, чтобы лучше увидеть недельные и дневные колебания, однако полная кривая охватывает 4 года.

1.3.5. Обучение модели

Для оценивания параметров обучаемой модели используются принципы байесовской статистики: либо поиск апостериорного максимума (MAP) с помощью L-BFGS, либо полный байесовский вывод. Для этого применяется платформа вероятностного программирования Stan.

Когда сезонность и индикаторы праздников для каждого наблюдения объединены в матрице X, а индикаторы точек изменения $a(t)$ объединены в матрице A, модель полностью можно выразить несколькими строками кода Stan:

```
model {
  // Priors
  k ~ normal(0, 5);
  m ~ normal(0, 5);
  epsilon ~ normal(0, 0.5);
  delta ~ double_exponential(0, tau);
  beta ~ normal(0, sigma);
  // Logistic likelihood
  y ~ normal(C ./ (1 + exp(-(k + A * delta) .* (t - (m + A * gamma)))) +
             X * beta, epsilon);
  // Linear likelihood
  y ~ normal((k + A * delta) .* t + (m + A * gamma) + X * beta, sigma);
}
```

Параметры `tau` и `sigma` контролируют силу регуляризации точек изменения модели и сезонности соответственно. Регуляризация важна для обеих компонент, чтобы избежать переобучения, однако не всегда бывает достаточно исторических данных, чтобы выбрать лучшие параметры регуляризации с помощью перекрестной проверки. Мы задаем значения по умолчанию, которые подходят для большинства задач прогнозирования, и когда эти параметры должны быть оптимизированы, это происходит в соответствии с концепцией «аналитик в курсе дел» (analyst-in-the-loop).

На рис. 4 показаны прогнозы модели Prophet для временного ряда, представляющего собой события Facebook. Эти прогнозы были получены для тех же самых отрезков времени, что были представлены на рис. 3 ранее, для обучения каждый раз использовались данные, предшествующие первой прогнозируемой дате. Прогнозная модель Prophet может предсказать как еженедельную, так и годовую сезонность, в отличие от базовых моделей на рис. 3, не слишком остро реагирует на праздники в первый год. Если рассматривать прогнозы для первого отрезка, модель Prophet слишком сложно восстанавливает годовую сезонность (признак переобучения), обучившись на данных только за один год. Если рассматривать прогнозы для третьего отрезка, то здесь модель пока не знает, что тренд изменился.

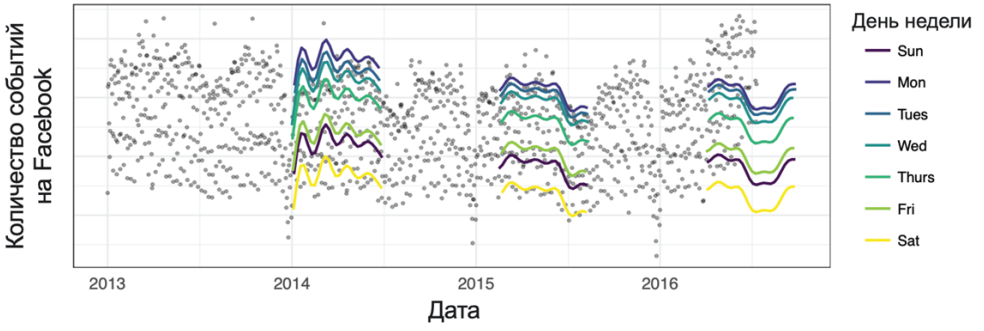


Рис. 4 Прогнозы Prophet, соответствующие прогнозам на рис. 3. Как и раньше, прогнозы сгруппированы по дням недели для визуализации еженедельной сезонности

А здесь на рис. 5 показано, что прогнозы уже учитывают изменение тренда (пунктирная линия).

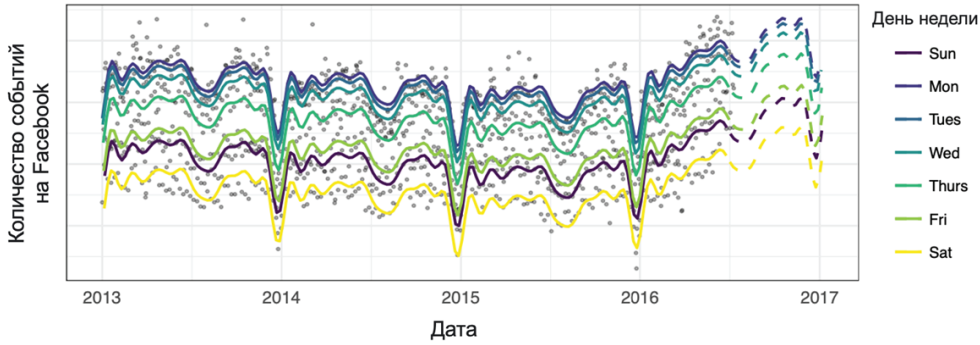


Рис. 5 Прогнозы Prophet с использованием всех доступных данных, включая интерполяцию исторических данных. Сплошные линии соответствуют прогнозам внутри выборки, пунктирные линии – прогнозам вне выборки

Важным преимуществом декомпозируемой модели является то, что она позволяет нам рассматривать каждую компоненту прогноза отдельно. На рис. 6 показаны тренд, недельная сезонность и годовая сезонность. Это дает полезный инструмент для аналитиков, позволяющий понять проблему прогнозирования, помимо простого получения прогноза.

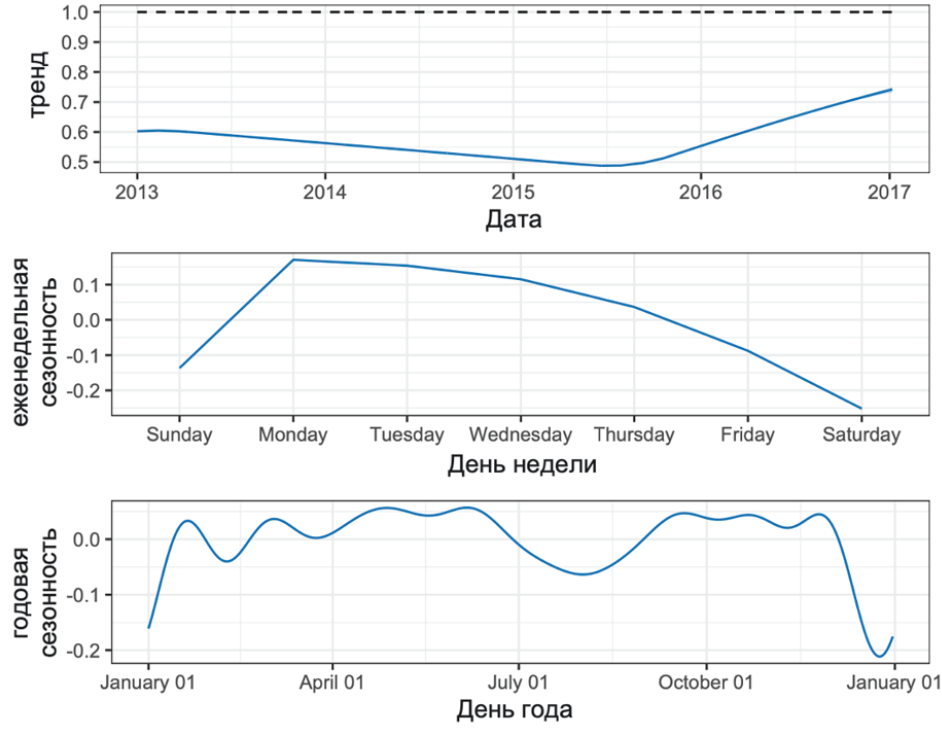


Рис. 6 Компоненты прогноза Prophet

1.3.6. Моделирование в соответствии с принципом «analyst-in-the-loop»

Аналитики, делающие прогнозы, часто обладают обширными знаниями о количественных характеристиках процесса, который они прогнозируют, но при этом могут обладать ограниченными познаниями в области статистики. В спецификации модели Prophet есть несколько параметров, с помощью которых аналитики могут изменить модель, чтобы применить свой опыт и внешние знания, при этом знания в области статистики не требуются.

Размеры рынка: аналитики могут располагать внешними данными об общем размере рынка и могут применять это знание напрямую, указав возможности.

Точки изменения: известные даты точек изменения, такие как даты изменения продукта, могут быть указаны явно.

Праздники и сезонность: аналитики, с которыми мы работаем, знают, какие праздники влияют на рост и в каких регионах, и они могут напрямую вводить соответствующие даты праздников и определять временные рамки сезонности.

Параметры сглаживания: регулируя τ, аналитик может выбирать более глобальные или локальные гладкие модели. Параметры сглаживания для сезонности и праздника (σ, ν) позволяют аналитику сообщить модели информацию о силе сезонных колебаний в будущем, основываясь на исторических данных.

С помощью хороших инструментов визуализации аналитики могут использовать эти параметры для улучшения качества обучения модели. Когда обучение модели строится на исторических данных, быстро становится очевидным, были ли точки изменения, которые были пропущены в процессе автоматического отбора точек изменения. Параметр τ представляет собой своего рода рычаг, который можно поворачивать для увеличения или уменьшения гибкости тренда, а σ – рычаг, позволяющий увеличивать или уменьшать силу сезонной составляющей. Визуализация предлагает множество дополнительных возможностей для ручной корректировки: линейный тренд или логистический рост, определение временных рамок сезонности и определение отдаленных периодов времени, которые следует исключить из обучения. Все эти корректировки можно выполнить без статистического опыта, они являются для аналитиков отличной возможностью применить свои идеи или предметные знания.

В литературе по прогнозированию часто проводится различие между статистическими прогнозами, которые основаны на моделях, соответствующих историческим данным, и экспертными прогнозами (также называемыми управленческими прогнозами), которые делают эксперты. У каждого из этих подходов есть свои преимущества. Статистические прогнозы требуют меньше предметных знаний и усилий со стороны аналитиков, и их очень легко масштабировать. Экспертные прогнозы могут включать больше информации и более оперативно реагировать на меняющиеся условия, но может потребоваться интенсивная работа аналитиков (Sanders 2005).

Наш подход к моделированию в соответствии с принципом «аналитик в курсе дел» – это альтернативный подход, который пытается сочетать преимущества как статистических, так и экспертных прогнозов, сосредоточив усилия аналитиков на улучшении модели, когда это необходимо, вместо того чтобы напрямую получать прогнозы через некоторые не до конца исследован-

ные процедуры. Мы обнаружили, что наш подход очень похож на цикл «преобразовывай–визуализируй–моделируй», предложенный Wickham & Grolemund (2016), в котором применение предметных знаний после ряда итераций позволяет получить улучшенную модель.

Обычное масштабируемое прогнозирование основывается на полностью автоматизированных процедурах, но экспертные прогнозы оказались очень точными во многих задачах (Lawrence et al. 2006). Предлагаемый нами подход позволяет аналитикам применять к прогнозам экспертные знания с помощью небольшого набора интуитивно понятных параметров и опций модели, сохраняя при этом возможность использовать при необходимости полностью автоматизированное статистическое прогнозирование.

Теперь мы опишем процесс автоматизации оценки качества прогнозов, который позволит аналитикам получить наиболее релевантные прогнозы.

1.4. Автоматизация оценки качества прогнозов

1.4.1. Использование прогнозов базовых моделей

При оценке любого метода прогнозирования важно сравнивать его с набором базовых методов. Мы предпочитаем использовать упрощенные прогнозы, которые делают сильные предположения о лежащем в основе процессе, но при этом могут дать разумный прогноз на практике. Мы нашли полезным сравнение с наивными моделями (прогнозирование последним известным значением и средним значением по выборке), а также автоматизированными методами прогнозирования, описанными в разделе 2.2.

1.4.2. Оценка качества прогноза

Прогнозы делаются на определенный *горизонт*, который мы обозначаем H. Горизонт – это количество дней в будущем, на которое мы прогнозируем: обычно это 30, 90, 180 или 365 дней в наших задачах. Таким образом, для любого прогноза с ежедневными наблюдениями мы генерируем H прогнозов, каждый из которых будет иметь некоторую ошибку. Нам нужно задать прогнозируемую целевую функцию для сравнения методов и отслеживания качества. Кроме того, понимание того, насколько подвержена ошибкам наша процедура прогнозирования, позволит бизнес-пользователям, работающим с прогнозами, определить, можно ли этим прогнозам вообще доверять.

Пусть $\hat{y}(t \mid T)$ представляет собой прогноз для момента времени t, полученный с помощью исторических данных до момента времени T, а $d(y, y')$ – метрика расстояния типа средней абсолютной ошибки $d(y, y') = |y - y'|$.

Выбор функции расстояния должен определяться конкретной задачей. De Gooijer & Hyndman (2006) сделали обзор нескольких таких метрик, на практике мы предпочитаем среднюю абсолютную процентную ошибку (MAPE) за ее интерпретируемость. Определим эмпирическое качество прогнозов для $h \in (0, H)$ периодов после момента T как

$$\phi(T,h) = d\left(\hat{y}(T+h \mid T),\, y(T+h)\right).$$

Чтобы оценить это качество и как оно изменяется с h, обычно задают параметрическую модель для члена ошибки и оценивают ее параметры на основе данных.

Например, если бы мы использовали $AR(1)$-модель $y(t) = \alpha + \beta y(t-1) + \nu(t)$, мы бы предположили, что $\nu(t) \sim \text{Normal}(0, \sigma_\nu^2)$, и сосредоточились на оценке дисперсии σ_ν^2, исходя из наших данных. Затем мы могли бы получить математическое ожидание, используя любую функцию расстояния посредством симуляций или используя аналитическое выражение для математического ожидания суммы ошибок. К сожалению, эти подходы дают правильные оценки ошибки только при условии, что задана модель, верно описывающая процесс генерации данных, – условие, которое вряд ли будет выполняться на практике.

Мы предпочитаем использовать непараметрический подход к оцениванию ожидаемых ошибок, который применим ко всем моделям. Подход аналогичен применению перекрестной проверки для оценки ошибки за пределами исторической выборки. Ее применяют к моделям, генерирующим прогнозы для данных, в которых каждое наблюдение происходит из одного и того же распределения и не зависит от других наблюдений. Для имеющегося набора прогнозов мы обучаем модель ожидаемой ошибки, которую получили бы для разных горизонтов прогноза h:

$$\xi(h) = \mathrm{E}\left[\phi(T, h)\right]. \tag{8}$$

Эта модель должна быть гибкой, но при этом может выдвигать некоторые простые предположения. Во-первых, функция должна быть локально гладкой по h, потому что мы ожидаем, что любые ошибки, которые мы делаем в последовательные дни, будут относительно схожими. Во-вторых, мы можем выдвинуть предположение, что функция должна слабо возрастать по h, хотя это вовсе не обязательно должно выполняться для всех прогнозных моделей. На практике мы используем локальную регрессию (Cleveland & Devlin 1988) или изотоническую регрессию (Dykstra 1981) в качестве гибких непараметрических моделей кривых ошибок.

Для получения ошибок прогнозов мы используем процедуру, которую называем *имитированием исторических прогнозов* (simulated historical forecasts, SHF).

1.4.3. Метод имитированных исторических прогнозов

Мы хотели бы обучить модель ожидаемой ошибки, как в (8), чтобы выполнить отбор и оценку модели. К сожалению, сложно использовать такой метод, как перекрестная проверка, потому что наблюдения не подлежат перестановке, мы не можем просто случайным образом разбить данные. Тогда был предложен метод *имитированных исторических прогнозов* (Simulated Historical Forecasts, SHF).

В пределах отрезка с исходными обучающими данными выбирают *K пороговых точек* (cut-off points по терминологии Prophet), на основе которых формируются блоки данных для выполнения перекрестной проверки: все исторические наблюдения, предшествующие k-й точке отсчета (а также сама эта точка), образуют обучающие данные для подгонки соответствующей модели, а H исторических наблюдений, следующих за точкой отсчета, образуют горизонт прог-

нозирования. Расстояние между точками отсчета называется *периодом* (period) и по умолчанию составляет $H/2$. Обучающие наблюдения в первом из K блоков образуют *начальный отрезок* (initial period). Длина этого отрезка по умолчанию составляет $3 \times H$, однако исследователь при желании может ее изменить.

Каждый раз после подгонки модели на обучающих данных из k-го блока рассчитываются предсказания для прогнозного горизонта того же блока, что позволяет оценить качество прогноза с помощью подходящей метрики (например, RMSE). Значения этой метрики, усредненные по каждой дате прогнозных горизонтов каждого блока, в итоге дают оценку качества предсказаний, которую можно ожидать от модели, построенной по всем исходным обучающим данным. Это, в свою очередь, позволяет сравнить несколько альтернативных моделей и выбрать оптимальную.

Рис. 7 Перекрестная проверка по методу имитированных исторических прогнозов в Facebook Prophet

Перекрестная проверка по методу имитированных исторических прогнозов выполняется в Prophet с помощью функции `cross_validation()`, которая имеет следующие параметры:

- `model` – модельный объект;
- `horizon` – длина прогнозного горизонта в каждом блоке данных, используемом для выполнения перекрестной проверки;
- `units` – название единицы измерения времени (например, `"days"`, `"hours"`, `"secs"`);
- `period` – расстояние между пороговыми точками;
- `initial` – длина начального отрезка с обучающими данными в первом блоке.

Как видно, процедура основана на классической оценке прогнозов скользящим способом (Tashman 2000), но использует только небольшую последовательность дат в качестве пороговых точек, вместо того чтобы делать один прогноз на одну дату. Главное преимущество применения меньшего количества пороговых точек (классическая скользящая оценка дает прогноз на один шаг вперед, т. е. на одну дату) заключается в том, что она требует меньшее время вычислений и при этом дает менее коррелированные измерения точно-

сти. Этот способ имеет то преимущество, что его легко объяснить аналитикам и лицам, принимающим решения, и является относительно бесспорным для понимания ошибок прогноза. Однако есть две основные проблемы, которые следует учитывать при использовании методологии SHF для оценки и сравнения методов прогнозирования.

Во-первых, чем больше пороговых точек мы делаем, тем больше коррелируют оценки ошибок прогнозов. В крайнем случае, когда мы делаем прогноз на один день, прогнозы вряд ли сильно поменяются с учетом информации одного дополнительного дня и ошибки от одного дня к другому будут почти идентичны друг другу. С другой стороны, если у нас будет очень мало пороговых точек, мы оценим в рамках прогнозирования меньшее количество наблюдений, чтобы составить итоговую оценку ошибки прогноза. Поэтому в качестве компромисса была выбрана эвристика $H/2$.

Во-вторых, методы прогнозирования могут работать лучше или хуже с увеличением объема данных. Более длинная историческая выборка может привести к худшим прогнозам, когда модель неверно указана и мы переобучаемся на прошлое, например используя выборочное среднее для прогнозирования временного ряда с трендом.

1.4.4. Выявление больших ошибок прогнозов

Когда мы получаем слишком большое количество прогнозов, которое невозможно проверить вручную, важно иметь возможность автоматически находить проблемные прогнозы. Автоматическое выявление плохих прогнозов позволяет аналитикам наиболее эффективно использовать свое ограниченное время и свой опыт для исправления любых проблем. Есть несколько способов применения SHF для определения вероятных проблем с прогнозами.

Если прогноз содержит большие ошибки относительно базовых моделей, скорее всего, задана неверная спецификация модели. При необходимости аналитики могут корректировать модель тренда или сезонность.

Большие ошибки прогнозов на определенную дату, характерные для большинства методов, указывают на выбросы. Аналитики могут выявить выбросы и удалить их.

Когда ошибка SHF для метода резко увеличивается при переходе от одной пороговой точки к другой, это может указывать на изменение процесса, генерирующего данные. Добавление точек изменения или моделирование разных фаз по отдельности может решить эту проблему.

Существуют аномалии, которые нелегко исправить, но большинство возникающих проблем можно исправить, указав точки изменения и удалив выбросы. Эти проблемы легко выявляются и исправляются после того, как прогноз был помечен для проверки и визуализирован.