КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультету радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

Прогнозування потоку відвідувачів торгово-розважального центру з використанням нейронних мереж

Дипломна робота бакалавра
студента 4 року навчання
Спеціальність: 123 «Комп'ютерна інженерія»
ОНП «Комп'ютерна системи та мережі»
Миколи НЕФЕДЧЕНКА

Науковий керівник:
канд. фіз.-мат. наук
Олександр БАРАБАНОВ,
доцент кафедри комп'ютерної інженерії

Рецензент:
канд. фіз.-мат. наук
Анатолій ШКАВРО,
доцент кафедри фізики та конденсованих
середовищ
інституту високих технологій

До захисту допускаю: Завідувач кафедрою

Юрій БОЙКО

Ухвалено на засіданні кафедри "____" _____2022 р., протокол № ____

Київ - 2022

Реферат

Випускна кваліфікаційна бакалаврська робота містить 55 сторінок, 19 рисунків, 1 додаток, 17 інформаційних джерел.

ЧАСОВІ РЯДИ, СЕЗОННІСТЬ, ТРЕНД, PROPHET, ПРОГНОЗУВАННЯ НЕЙРОННИМИ МЕРЕЖАМИ, NEURALPROPHET, AR-NET.

Тема: Прогнозування потоку відвідувачів торгово-розважального центру.

Об’єкт дослідження: Набір даних в xls форматі, що містить інформацію про кількість відвідувачів ТРЦ «Lavina» погодинно протягом року.

Мета роботи: Підбір методу прогнозування потоку відвідувачів для торгово-розважального центру з отриманням найменших похибок та аналіз вирішення подібних задач.

Розглянуто: Методи прогнозування, опис їхньої роботи, застосування, тощо.

Інструменти розроблення: Мова програмування Python, програмні бібліотеки математичного характеру, для роботи зі звітами ТРЦ у форматі Excel – pandas.

Реалізовано: Аналіз методів прогнозування, розробка плану, робота над програмою та висновок щодо подальшого використання даних методів для прогнозування кількості відвідувачів на основі річних даних.

Зміст

Вступ

Прогнозування є цінним для бізнесу, оскільки це дає можливість приймати обґрунтовані бізнес-рішення та розробляти стратегії на основі прогнозованих значень.

В умовах зростаючої конкуренції та концентрації комерційної нерухомості з'являється необхідність прогнозування відвідуваності торгівельних центрів. Математичні моделі, що здатні прогнозувати відвідуваність, можуть застосовуватися для планування заходів, закупівлі товару та енергоресурсів.

Для обслуговування торговельних центрів необхідна величезна кількість енергоресурсів, зокрема і електроенергії. Постачальники таких ресурсів надають можливість суттєво зекономити клієнтам у випадку, якщо замовник робить передзамовлення хоча б на один день (передзамовлення може бути на день, на тиждень чи на місяць наперед). Умовою для цього має бути впевненість, що замовник використає точно замовлений об’єм електроенергії. Це пов’язано з тим що електроенергія не може зберігатися у великій кількості, і уся електросистема має постійно бути в русі. В таких випадках актуальним є математичне прогнозування кількості відвідувачів, оскільки для торгівельних центрів, на рівні з погодними умовами, це є одним з головних факторів, від яких залежить кількість спожитої електроенергії.

Для вирішення цих задач доцільно застосовувати методи машинного навчання, як приклад, на основі нейронних мереж. Лідерами серед мов програмування для машинного навчання є Python та R.

Засоби програмної реалізації

Обгрунтування вибору мови програмування

Існує багато способів аналізу даних, які впорядковані в часі. Один із способів — просто помістити дані в електронну таблицю та використовувати вбудовані функції для створення прямої лінії тренду та дослідити нахил, щоб отримати прогнозовані зміни.

Недоліком простої прямої лінії тренду є тенденція групувати дані таким чином, щоб вони змішувалися разом, що дозволяє упускати багато важливих деталей, які існують у фактичних даних.

Створення моделі часового ряду з використанням Python дозволяє охопити більшу складність даних і включає всі елементи даних, які можуть бути важливими. Це також дає можливість коригувати різні параметри, налаштовуючи модель, щоб зробити її потенційно більш точною.

Python — це об’єктно-орієнтована мова програмування загального призначення та високорівнева мова, яка була вперше запущена в 1989 році. Вона виділяється читабельністю коду за рахунок значного використання вільного простору. Загалом, вона була побудована таким чином, щоб можна було порівняно інтуїтивно писати та розуміти код. Це і робить Python ідеальною мовою програмування для тих цілей, коли необхідна саме швидкість розробки.

Найбільші організації світу — від NASA до Netflix, Spotify, Google тощо — використовують Python у тій чи іншій формі для здійснення своїх послуг. Згідно з рейтингами, Python є третьою за популярністю мовою програмування в світі, поступаючись лише Java та C. Це тому що Python простий у використанні, має простий синтаксис, активну спільноту та є універсальним.

Python можна використовувати для різних проектів та задач, наприклад для аналізу даних і візуалізації до штучного інтелекту, розробки мови, дизайну та веб-розробки.

Python зручно використовувати для широкомасштабного розгортання машинного навчання, оскільки він має бібліотеки з такими інструментами, як TensorFlow, scikit-learn і Keras, які дозволяють створювати складні моделі даних, які можна підключати безпосередньо до виробничої системи.

Окрім Python для багатьох статистичних розрахунків використовується мова R, а саме для розробки статистичного програмного забезпечення та аналізу даних. З тих пір, як машинний інтелект та аналіз даних стали популярними, R також здобула свою популярність.

R також надає широкий спектр бібліотек для графічних методів. З допомогою R можна створювати статистичні графіки, які використовуються для графіків якості публікації. Також доступні динамічні та інтерактивні графіки. R має мережу архіву для всіх пакетів, які він підтримує. Він містить понад 10 000 пакетів. Взагалі мова R — це мова командного рядка, але існують кілька інтерфейсів, що забезпечують інтерактивний графічний інтерфейс для полегшення роботи.

Хоча R та Python популярні для подібних цілей, тобто для аналізу даних і машинного навчання, обидві мови мають різні особливості. Крім того, кожна мова має свої переваги та недоліки.

R був створений Россом Іхакою та Робертом Джентльменом у 1995 році. R зосереджена на мові кодування, створеній виключно для статистики та аналізу даних, тоді як Python має гнучкість із пакетами для адаптації даних. R чудово підходить, для роботи зі складними візуальними елементами із легким налаштуванням, тоді як Python не настільки хороший для візуалізації, готової до друку. Також, R важко інтегрувати з виробничим процесом. Здебільшого це інструмент статистичного аналізу та графіки,

тоді як Python легко інтегрується в робочий процес і може стати фактичною частиною продукту [2].

Хоча R все ще використовується науковцями та науковцями з даних, комерційні компанії, що працюють з даними та зацікавлені в їх аналітиці, звертаються до Python через його масштабованість і простоту використання.

Тому, враховуючи все вищесказане, для подальшої роботи з даними мною було обрано Python.

Бібліотеки для роботи з даними

Мова програмування Python при роботі з даними дозволяє обирати серед величезної кількості бібліотек ті, які найкраще підходять поставленому завданню. Для роботи з прогнозуванням даних потрібно застосовувати деякі з них, як от:

xlrd — це бібліотека для читання даних та форматування інформації з файлів Excel в історичному форматі .xls. Ця бібліотека обмежена роботою тільки з .xls файлами. Не підтримуються та ігноруватимуться діаграми, макроси, зображення, будь-який інший вбудований об’єкт, включаючи вбудовані аркуші, модулі VBA, формули (але витягуються результати розрахунків формул), коментарі, гіперпосилання, також не підтримуються файли, захищені паролем, і, відповідно, не можуть бути прочитані цією бібліотекою. Для іншої роботи з .xls файлами можуть бути використані інші бібліотеки, а для моєї роботи над побудовою прогнозу функціоналу цієї бібліотеки цілком достатньо.

Datetime – ця бібліотека дозволяє працювати над маніпулюванням датами та часом. Хоча арифметика дати й часу підтримується за замовчуванням, основна увага приділяється ефективному витягу атрибутів для форматування та маніпулювання результатами. Це може бути корисним у випадках, коли джерело надає дані «брудними», і потрібно провести операції відбору над даними.

Matplotlib — це бібліотека графіків для мови програмування Python та її числового математичного розширення NumPy. Він надає об'єктно-орієнтований API для вбудовування графіків у програми з використанням наборів інструментів GUI загального призначення.

Pandas — це програмна бібліотека, написана для мови програмування Python для маніпуляції та аналізу даних. Зокрема, вона пропонує структури даних та операції для маніпулювання числовими таблицями та часовими рядами. Pandas в основному використовується для

аналізу даних і пов’язаних з ними маніпуляцій з табличними даними в Dataframes, дозволяє імпортувати дані з різних форматів файлів, таких як значення, розділені комами, JSON, Parquet, таблиці чи запити бази даних SQL та Microsoft Excel. Pandas дозволяє виконувати різноманітні операції маніпулювання даними, такі як злиття, зміна форми, вибір, а також очищення даних та функції сперечання даних.

Методи прогнозування

Лінійна регресія

Підхід лінійної регресії полягає у тому, щоб сформувати прогноз на основі залежності двох змінних.

Лінійна регресія — це лінійний підхід для моделювання зв'язку між скалярною реакцією та однією або кількома пояснювальними змінними (також відомими як залежні та незалежні змінні). Випадок однієї пояснювальної змінної називається простою лінійною регресією; а для більш ніж однієї - множинною лінійною регресією. Цей термін відрізняється від багатовимірної лінійної регресії, де прогнозуються численні корельовані залежні змінні, а не одна скалярна змінна [7].

У найпростішому випадку лінійна регресія дозволяє описати залежність між модельованою змінною y та незалежною змінною x (приклад на рисунку 1):

$$y = \beta_0 + \beta_1 x_t + \varepsilon_t.$$

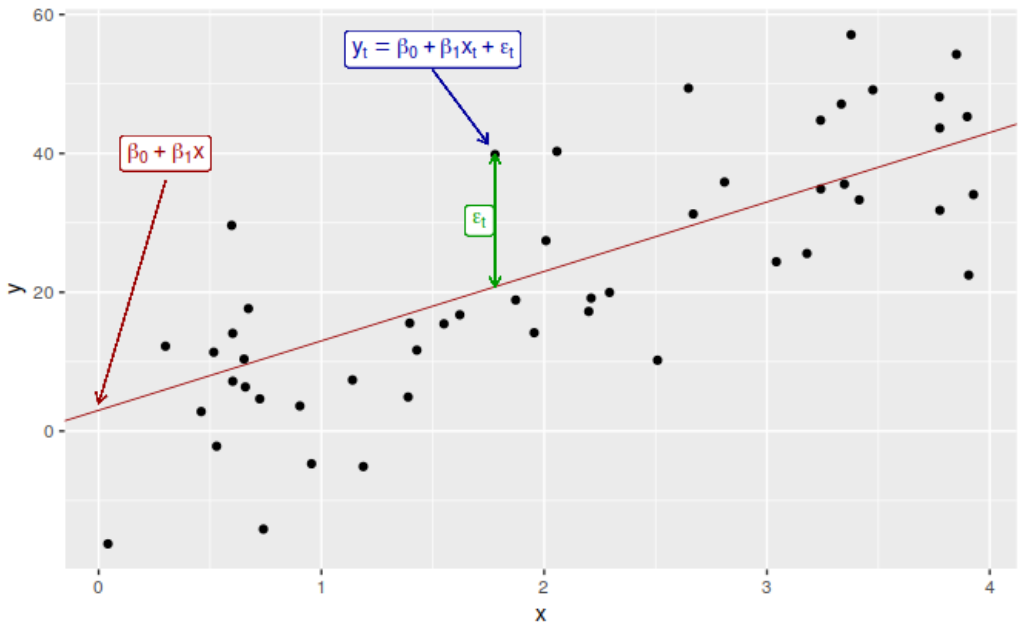


Рис. 1. Приклад даних змодельованих лінійною регресією.

Кожен аналіз y_t складається з систематичної частини моделі – $\beta_0 + \beta_1 x_t$ та випадковою «помилкою» ε_t. Це описує, що може впливати на значення y_t окрім x_t.

Множинна регресія – це узагальнена лінійна регресія, що враховує більше двох незалежних змінних та відображає зв'язок між ними. Базове рівняння для множинної лінійної регресії:

$$y = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t.$$

Коефіцієнти β_k вимірюють вплив кожної незалежної змінної на цільову змінну yt.

Модель множинної регресії базується на наступних припущеннях:

- між залежними змінними та незалежними змінними існує лінійна залежність;
- незалежні змінні майже не корелюють між собою;

спостереження відбираються рандомно серед набору.

Також, важливе припущення, що кожна незалежна змінна x_t не є випадковою.

Під час практичного застосування моделі, у наявні історичні дані, а коефіцієнти $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ потрібно оцінити, зазвичай це проводиться статичним програмним забезпеченням. Метод найменших квадратів забезпечує ефективний вибір коефіцієнтів шляхом мінімізації суми квадратів помилок [12]:

$$\sum_{t=1}^{T} \varepsilon_t^2 = \sum_{t=1}^{T} \left(y_t - \beta_0 - \beta_1 x_{1,t} - \beta_2 x_{2,t} - \cdots - \beta_k x_{k,t}\right)^2$$

Прогнозні значення y можуть бути отримані з обчислених оцінок коефіцієнтів $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ і прирівнюючи ε_t до нуля. Тобто,

$$\hat{y}_t = \hat{\beta}_0 + \hat{\beta}_1 x_{1,t} + \hat{\beta}_2 x_{2,t} + \cdots + \hat{\beta}_k x_{k,t}.$$

Підставляючи відповідні значення $x_{1,t}, \ldots, x_{k,t}$ для $t = 1, \ldots, T$, можна отримати значення y_t, що є прогнозом для оцінки моделі [14].

Експоненційне згладжування

Метод експоненційного згладжування був запропонований ще у 1950-их роках та досі успішно використовується у прогнозуванні часових рядів. Прогнози отримані за допомогою методу експоненційного згладжування – це середньозважені значення попередніх спостережень з вагами, що експоненційно зменшуються для старших спостережень. Тобто чим актуальніші спостереження, тим більшу вагу їм надається у порівнянні з застарілими даними [13].

Просте експоненційне згладжування – метод прогнозування, що найбільш широко використовується. Просте експоненційне згладжування вимагає мало обчислень. Цей метод використовується, коли шаблон даних приблизно горизонтальний (тобто немає ні циклічних змін, ні яскраво виражених трендів в історичних даних).

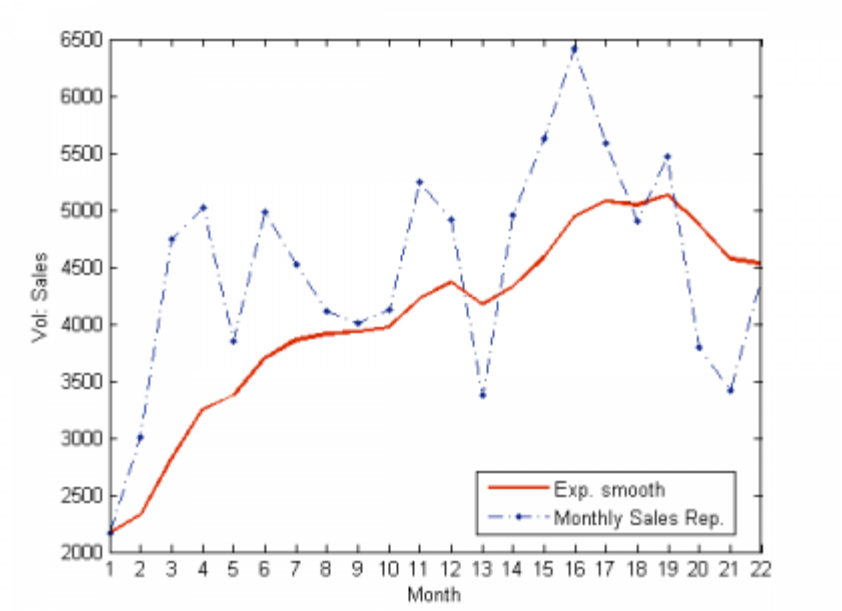


Рис. 2. Експоненційне згладжування.

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots,$$

де $0 \leq \alpha \leq 1$ – згладжувальний параметр. Прогнозування на один крок вперед для T+1 – середньозважене значення всіх спостережень $y_1, y_2, \ldots, y_T$. Швидкість зменшення позначається параметром α.

Рівняння набуває такий вигляд:

$$\hat{y}_{T+1|T} = \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T l_0,$$

де l_0 – початкове значення прогнозування, що відбувається у момент часу t =1.

Виходить, що метод простого експоненційного згладжування потребує такі параметри: α – параметр згладжування, l_0 – початкове значення.

У лінійних регресіях оцінити значення коефіцієнтів можна шляхом мінімізації суми квадратів помилок. А у цьому випадку – аналогічно, але оцінкою згладжувального параметра.

$$\sum_{t=1}^{T} e_t^2 = \sum_{t=1}^{T} \left(y_t - \hat{y}_{t|t-1}\right)^2$$

За припущення відсутності тенденції в даних, просте експоненціальне згладжування дає нормальні результати, але цей спосіб не підходить у випадку наявності тренду. Подвійне експоненційне згладжування використовується, коли в даних є лінійна тенденція [6].

Основна ідея подвійного експоненціального згладжування полягає в тому, щоб ввести термін у якому враховується можливість прогнозування, коли набір даних демонструють певний тренд. Цей компонент нахилу сам оновлюється за допомогою експоненційного згладжування.

Цей метод складається з рівняння для прогнозу та двох рівнянь згладжування:

$$\hat{y}_{t+h|t} = l_t + hb_t,$$
$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}),$$
$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1},$$

де l_t – оцінка рівня ряду в момент часу t, b_t – оцінка тренду ряду в момент часу t, α – згладжувальний параметр і $0 \le \alpha \le 1$, β^* – згладжувальний параметр для тренду ряда $0 \le \beta^* \le 1$.

$$\hat{y}_{t+h|t} = l_t + (\phi + \phi^2 + \cdots + \phi^h)b_t,$$
$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + \phi b_{t-1}),$$
$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)\phi b_{t-1}.$$

При ϕ =1 метод ідентичний методу лінійного тренду Хольта, значення $0 \leq \phi < 1$ гасять тренд так, що у майбутньому він перетворюється у константу.

За припущення про наявність лише лінійної тенденції в даних, подвійне експоненціальне згладжування дає результати, але воно не вдається якщо наявні і тренд і сезонність. Потрійне експоненціальне згладжування використовується, коли є тренд у даних разом із сезонними коливаннями.

Два методи Холта-Вінтерса розроблені для часових рядів, які демонструють лінійну тенденцію:

1. Адитивний метод Холта-Вінтерса: використовується для часових рядів з константою (адитивні) сезонних коливань:

$$\mu_p = (y_1 + y_2 + \cdots .. y_p) / p$$
$$b_p = ((y_{p+1} + y_{p+2} + \ldots y_{p+p}) - (y_1 + y_2 + \cdots y_p)) / p^2$$
$$S_i = y_i / \mu_p \quad i = 1,2,3 \ldots . p$$

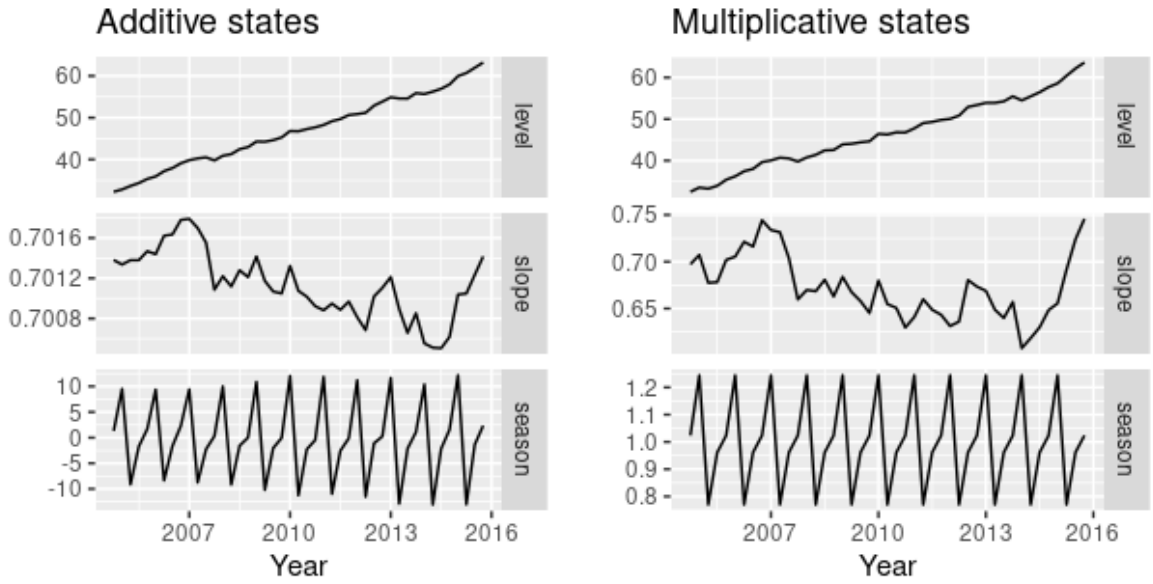


Рис. 3. Методи Холта-Вінтерса.

2. Мультиплікативний метод Холта-Вінтерса: використовується для часових рядів зі збільшенням (мультиплікаційні) сезонних коливань:

$$\mu_p = (y_1 + y_2 + \cdots .. y_p) / p$$

$$b_p = ((y_{p+1} + y_{p+2} + \ldots y_{p+p}) - (y_1 + y_2 + \cdots y_p)) / p^2$$

$$S_i = y_i / \mu_p \quad i = 1,2,3 \ldots . p$$

Модуль Prophet від Facebook

Алгоритм роботи Prophet

Prophet — це програмне забезпечення з відкритим вихідним кодом, випущене командою Facebook Core Data Science [18].

Prophet створений для побудови надійних прогнозів для планування та встановлення цілей, наприклад, для прогнозування кількості відвідувачів сайту. Але ми можемо його використати і для побудови інших прогнозувань.

Модуль Prophet був розроблений таким чином, щоб можна було легко підбирати параметри та робити прогнозування цільового значення.

Алгоритм Fbprophet розкладає часовий ряд на три частини, як показано в наступному рівнянні:

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

де $g(t)$ – функція, що описує тренд, $s(t)$ – функція, що описує сезонність та $h(t)$ – функція, що враховує ефект святкових днів/періодів. Похибка $\varepsilon(t)$ описує зміни, які не враховуються моделлю. Також, $\varepsilon(t) \sim N(0,1)$.

Для описання тренду в Prophet може застосовуватись модель насиченого зростання та кусково-лінійна функція [9].

Тренд

У моделі є можливість враховувати точки, де змінюється тренд. Нехай є множина S точок, де тренд змінюється, в момент часу $s_j, j = 1, \dots, S$. Вектор $\delta \in R^S$ – вектор змін темпів росту, тобто, δ_j – зміна темпу в момент часу s_j. Темп росту у довільний час t складається з таких компонентів:

$$k + \sum_{j:t>s_j} \delta_j.$$

Додатково визначимо вектор $a(t)$ компоненти якого:

$$a_j(t) = \{1, t \geq s_j \ \ 0,$$

інакше, темп росту в момент часу $t -\ \ k + a^T(t)\delta$. Тепер визначений темп росту у будь-який момент часу, отже, тепер потрібно, щоб параметр зміщення також був скорегований відповідно до кожного інтервалу – потрібно, щоб усі інтервали були з'єднані:

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

Остаточна модель для тренду:

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

У випадку якщо потрібно спрогнозувати значення, якому не характерне стрімке зростання, часто достатньо лінійної моделі тренду:

$$g(t) = (k + a^T(t)\delta)t + (m +)a^T(t)\gamma.$$

У Prophet впроваджено дві моделі трендів, які охоплюють багато додатків Facebook: модель насиченого зростання та кусково-лінійну модель [15].

Сезонність

У часових рядах, які пов’язані з людською поведінкою часто присутня мультиперіодична сезонність. Наприклад, будні дні – тижнева сезонність, відпустки та/або літні канікули – річна сезонність і т.д.

У моделі Prophet сезонність описується рядами Фур’є:

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

де P відповідає сезонному періоду, наприклад, у випадку тижневої сезонності $P = 7$, а у випадку річної $P = 365.25$.

Збільшення порядку N дозволяє ідентифікувати закономірності, які швидко змінюються, але водночас це може призвести до перенавчання моделі. Емпіричним шляхом було обрано найбільш оптимальні значення N: для річної сезонності $N = 10$, а для тижневої – $N = 3$. Інший спосіб обрати оптимальні значення N це порівнювати критерій Акайке для набору значень N та залишити те значення, яке відповідає найменшому критерію [15].

Святкові дні часто буває складно враховувати, оскільки їх ефект інколи неможливо змоделювати. Наприклад, окрім державних свят в Україні святкують і інші події, присутність яких може вплинути на кількість відвідувачів у магазинах, як от футбольні матчі, покупки перед шкільним сезоном, чорні п’ятниці, так само повторюються не циклічно – такі події складно запрограмувати та врахувати усі можливі зміни.

Модуль Prophet дозволяє створювати самостійно календар святкових днів та інших значимих подій або ж дає можливість користуватись уже створеним для кожної країни календарем. В залежності від цільової змінної, що прогнозується, можна обрати один з варіантів: якщо спеціаліст знає певні особливості, з якими можуть бути пов’язані різкі зміни поведінки часових рядів, то варто їх включити в модель, якщо ж поведінка часового

ряду повністю відповідає календарним святам – обиратиметься стандартний підхід Prophet.

Нехай наслідки від кожного святкового дня – незалежні. Визначимо для кожного свята i множину дат D_i – множина, на які дні раніше припадало свято та коли в майбутньому буде це свято. Визначимо індикаторну функцію:

$$Z(t) = [1(t \in D_1), \dots, 1(t \in D_L)]$$

яка є індикатором святкового дня. Нехай кожному святу i відповідає параметр k_i, тоді ефект від цього свята – функція:

$$h(t) = Z(t)k,$$
$$k \sim N(0,1).$$

Окрім ефекту святкових днів також може бути присутній ефект від днів до та після свята. У моделі визначається «вікно», яке відповідає святковим дням, та визначаються додаткові параметри для інтервалів, що потрапляють у відповідне «вікно».

Використання нейронних мереж для прогнозування часових рядів

Один з передових методів прогнозування часових рядів – це прогнозування за допомогою нейронних мереж. Для реалізації такого методу потрібна наявність великої кількості історичних даних для ефективного результату.

Штучні нейронні мережі – це методи прогнозування, які базуються на простих математичних моделях мозку. Вони допускають складні нелінійні зв'язки між змінною відповіді та її предикторами.

Нейронну мережу можна розглядати як мережу «нейронів», які організовані шарами. Прогнози (або вхідні дані) формують нижній шар, а прогнози (або результати) формують верхній шар. Також можуть бути проміжні шари, що містять «приховані нейрони».

Найпростіші нейронні мережі не містять прихованих шарів і еквівалентні лінійним регресіям. На рисунку показана нейромережева версія лінійної регресії з чотирма предикторами.

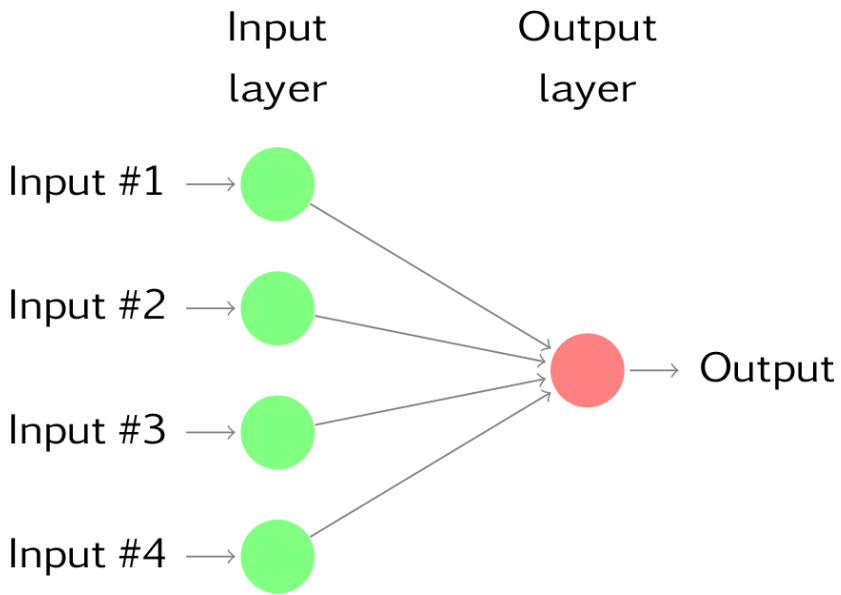


Рис. 4. Проста нейронна мережа.

Коефіцієнти, додані до цих предикторів, називаються «вагами». Прогнози отримують шляхом лінійної комбінації вхідних даних. Вагові коефіцієнти вибираються в структурі нейронної мережі за допомогою «алгоритму навчання», який мінімізує «функцію вартості», таку як MSE [1].

Якщо додати проміжний шар із прихованими нейронами, нейронна мережа стає нелінійною.

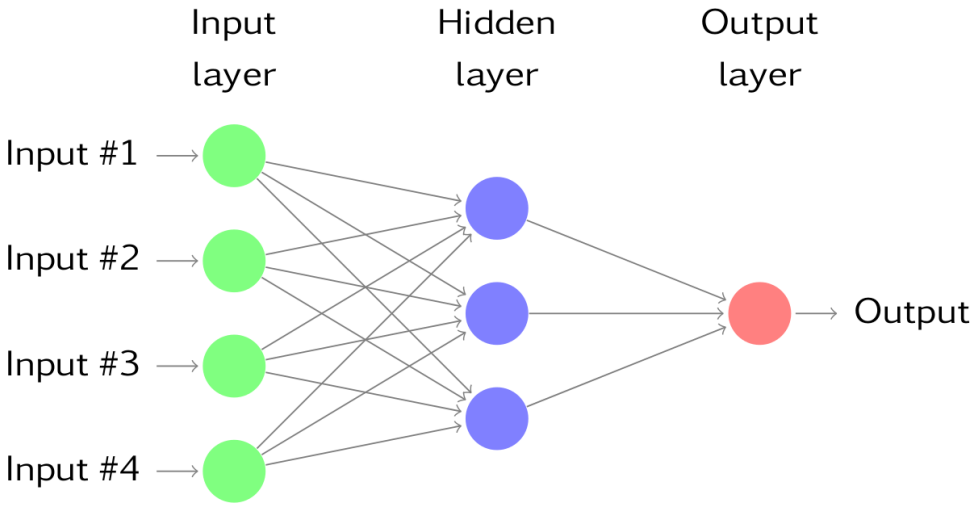


Рис. 5. Багаторівнева нейронна мережа.

Багаторівнева мережа з прямим зв'язком - це де кожен рівень вузлів отримує вхідні дані від попередніх шарів. Виходи вузлів одного шару є входами для наступного шару. Входи до кожного вузла об'єднуються за допомогою зваженої лінійної комбінації. Потім результат змінюється нелінійною функцією перед виведенням.

$$z_j = b_j + \sum_{i=1}^{4} = w_i \, , \, _j x_i$$

У прихованому шарі це потім змінюється за допомогою нелінійної функції, такої як сигмоїд, щоб дати вхідні дані для наступного шару. Це має тенденцію до зменшення впливу екстремальних вхідних значень, що робить мережу дещо стійкою до викидів [16].

NeuralProphet

NeuralProphet це еволюція алгоритму Prophet, що випущена командою Facebook Core Data Science. NeuralProphet є алгоритмом прогнозування часових рядів [19].

NeuralProphet — це новий набір інструментів прогнозування часових рядів з відкритим вихідним кодом, створений за допомогою PyTorch, заснований на нейронних мережах, яка дозволяє користувачам використовувати прості, але потужні моделі глибокого навчання, такі як AR-Net (авторегресивної нейронної мережі), для завдань прогнозування. Що робить NeuralProphet унікальним, так це його здатність включати додаткову інформацію, таку як тенденції, сезонність та повторювані події, під час генерування прогнозів та під час примірки [11].

Більшість проблем часових рядів вимагають легко зрозумілих прогнозів. У той же час бажаний ефективний прогноз. Ці два прагнення призводять до компромісу: інтерпретація проти продуктивності.

Значне підвищення ефективності часто відбувається завдяки складнішій моделі. Однак складність рідко є синонімом інтерпретації.

Існує два класи моделей:

- статистичні алгоритми (ARIMA, GARCH та інші), які можна математично пояснити, знаючи теорію;
- моделі нейронних мереж, які є значно більш заплутаними, але демонструють свою продуктивність.

NeuralProphet - це нейронні мережі, об’єднані з Prophet. Наміром авторів було об’єднати два класи моделей (статичні алгоритми та нейронні мережі), додавши нейронні мережі в Prophet, що є статистичний алгоритмом з метою підвищення ефективності при обмеженні втрати інтерпретації.

Нейронна мережа AR-net

У роботі NeuralProphet використовується нейронна мережа AR-net.

Нейронні мережі (NN) у своїй найпростішій формі складаються з шарів лінійних і нелінійних функцій, що чергуються і підігнані до цілі зі стохастичним градієнтним спуском за членом втрат.

Нейронну мережу можна розглядати як мережу «нейронів», які організовані шарами. Провісники утворюють нижній шар, а прогнози (або результати) формують верхній шар. Також можуть бути проміжні шари що містять «приховані нейрони».

Класичні моделі, такі як авторегресія (AR), використовують властиві характеристики часового ряду, що призводить до більш стислої моделі. Це можливо, оскільки модель робить сильні припущення щодо даних, наприклад, справжній порядок процесу AR. Ці моделі, однак, погано масштабуються для великого обсягу навчальних даних, особливо якщо є залежності на великі відстані або складні взаємодії.

Щоб подолати проблеми масштабованості, моделі від послідовності до послідовності стали популярними в обробці природної мови. Методи на основі RNN, зокрема, дозволяють створити більш виразну модель, не вимагаючи детальних функцій. Хоча ці моделі добре масштабуються для додатків із багатими даними, вони можуть бути занадто складними для типових даних часових рядів, що призводить до відсутності інтерпретації.

AR-Net має дві відмінні переваги:

- AR-Net добре масштабується до великих замовлень, роблячи можливим оцінку залежностей на великій відстані (важливо в програмах моніторингу з високою роздільною здатністю, наприклад, у домені центрів обробки даних).

- AR-Net автоматично вибирає та оцінює важливі коефіцієнти розрідженого процесу AR, усуваючи необхідність знати справжній порядок процесу AR.

Розглянемо часовий ряд y_1 , ... , y_t , виражений як процес AR. Щоб передбачити наступний часовий крок y_t , кожне з p минулих значень у множиться на вивчену вагу w_i (так званий коефіцієнт AR).

$$y_t = c + \sum_{i=1}^{p} w_i y_{t-i} + e_t$$

Для великого p (званого порядком) традиційний підхід може стати непрактично повільним у навчанні. Однак для моніторингу мілісекундних даних або даних другого рівня з високою роздільною здатністю потрібне велике замовлення. Щоб подолати проблему масштабованості, нейронну мережу навчають зі стохастичним градієнтним спуском для вивчення коефіцієнтів AR. Якщо відомий справжній порядок процесу, AR-Net ефективно вивчає майже ідентичні ваги, як і класичні реалізації AR, і однаково добре прогнозує наступне значення часового ряду.

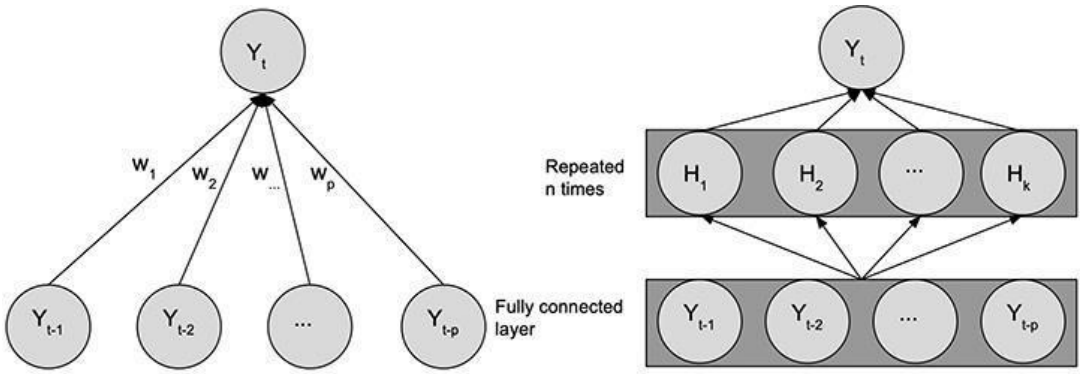


Рис. 6. Ліворуч: нейронна мережа, еквівалентна AR без прихованих шарів (найпростіша форма AR-Net). Справа: нейронна мережа AR, із n прихованими шарами (загальна AR-Net).

Якщо порядок невідомий, AR-Net автоматично дізнається відповідні ваги, навіть якщо основні дані генеруються шумним і надзвичайно розрідженим процесом AR. Це досягається, вводячи невеликий коефіцієнт регулювання вивчених ваг.

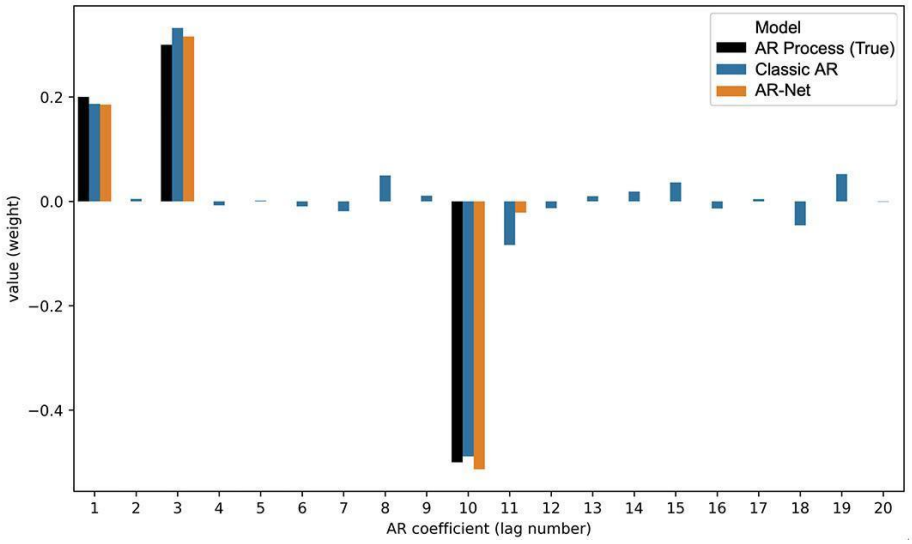


Рис. 7. AR-Net ефективно вивчає розріджені ваги, встановлюючи нерелевантні ваги на нуль. Класична AR переоцінює нерелевантні ваги. Підібрано на основі даних, створених шумним процесом AR-3 із розрідженістю (затримки 1, 3 і 10 відрізняються від нуля).

Це демонструє, що моделювання часових рядів за допомогою нейронних мереж можна так само інтерпретувати, як і використання класичних методів. Крім того, це дає можливість моделювати тимчасові дані без визначення справжнього порядку основного процесу AR, що дозволяє моделі автоматично вивчати точні далекі залежності без переобладнання.

Тренд

Класичним підходом до побудови тренду є моделювання його як комбінації зміщення m і швидкість росту k. Ефект тренду в момент часу t_1 визначається множенням темпу зростання на різницю в часі від початкової точки t_0 на вершині зміщення m.

$$T(t_1) = T(t_0) + k \cdot \Delta_t = m + k \cdot (t_1 - t_0)$$

NeuralProphet створює компонент тренду за допомогою цього класичного підходу, але дозволяє зростання швидкість зміни в кількох місцях. Тренд моделюється як безперервний поштучно лінійний ряд.

Можна узагальнити тренд, визначивши залежну від часу швидкість зростання $\delta(t)$ і зсув $\rho(t)$, що залежить від часу.

$$T(t) = \delta(t) \cdot t + \rho(t)$$

Частково-лінійний тренд змінює швидкість зростання лише на кінцевій кількості точок зміни. Між точками зміни, темп зростання тенденції залишається незмінним. Швидкість зростання і зміщення першого сегмента наведено як δ_0 і ρ_0 відповідно. Коригування швидкості в кожній точці зміни можна визначити як вектор $\delta \in R^{n_c}$, де δ_j – зміна швидкості на j^{th} й точка зміни. Швидкість зростання в момент часу t визначається додавання початкової швидкості зростання δ_0 із підсумовуванням коригування швидкості в усіх точках зміни до кроку часу t. Кожна зміна швидкості зростання δ_j є параметром, який потрібно підігнати до даних. Вектор коригування зміщення можна визначити як $\rho \in R^{n_c}$.

Аналогічно зміщення за часом t визначається початковим зміщенням ρ_0 і сумою коригувань зміщення в кожній точці зміни до час t. Однак зміщення для точки зміни c_j не є незалежним параметром, а визначається як $\rho_j = -c_j \delta_j$. Це конкретне визначення зміщення робить кусковий

лінійний ряд безперервним. Двійковий вектор $\Gamma(t) \in R^{n_c}$ показує, чи минув час t за кожну точку зміни.

$$T(t) = (\delta_0 + \Gamma(t)^{\mathrm{T}} \delta) \cdot t + (\rho_0 + \Gamma(t)^{\mathrm{T}} \rho)$$

NeuralProphet надає простий напівавтоматичний механізм для вибору відповідних точок зміни тренду. За бажанням, параметри швидкості зростання тренду можна упорядкувати під час навчання моделі. Це схоже на повністю автоматичний вибір точки зміни, оскільки тільки будуть вибрані найбільш релевантні точки змін, якщо такі є [11].

Сезонність

Сезонність у NeuralProphet обробляється за допомогою рядів Фур’є У цій техніці ряди Фур'є визначаються для кожної сезонності. Ряди Фур’є визначаються як пари синусів, косинусів і дозволяють моделювати множинні сезонності, а також сезонності з не цілою чисельною періодичністю, наприклад, річна сезонність із щоденними даними (p = 365,25) або з тижневими даними (p = 52,18).

$$S_p(t) = \sum_{j=1}^{k} (a_j \cdot cos\,(\frac{2\pi jt}{p}) + b_j \cdot sin\,(\frac{2\pi jt}{p}))$$

NeuralProphet автоматично активує щоденну, щотижневу або річну сезонність залежно від частоти і довжини даних. Кожен з цих трьох типів сезонної періодичності активується, якщо частота даних є більш високою роздільною здатністю, ніж відповідна періодичність, і якщо принаймні два повних періоди даних доступні [11].

Наприклад, модель буде вмикати щорічну сезонність, якщо дані охоплюють два роки або більше. Щотижнева сезонність також буде додана, якщо наявні дані двох або більше тижнів. Щоденна сезонність не буде активована, як щоденна частота не має вищої роздільної здатності, щоб врахувати сезонність протягом дня. Якщо не вказано інше, за замовчуванням.

Практична частина

Початок роботи

Для роботи над даною дипломною роботою було використано мову програмування Python, обчислення та робота над проектом проводилась у середовищі розробки Jupyter [17].

Jupyter Notebook - це некомерційний проект з відкритим кодом, створений у 2014 році, він розвивався для підтримки інтерактивної науки про дані та наукових обчислень на всіх мовах програмування. Він дозволяє працювати з кодом построково та построково відтворювати його, що полегшує процес відлагодження. Також зручно працювати з побудовою графіків та виведенням даних з таблиць.

До початку роботи з прогнозуванням була проведена робота з масивом даних, та переведення їх у відповідний формат. А саме:

- переведення таблиці у формат CSV;
- аналіз даних та відбір тих, що є для нас критично важливими та переведення їх у відповідний формат datetime та int (у нашому випадку це дві колонки, ds - що містить інформацію про дату та час, та y - що містить значення);
- відкидання пустих значень (це важливо, оскільки робота з лінійними рядами вимагає наявність усіх значень у наборі парних даних);
- побудова графіку з актуальних даних.

Для проведення цих робіт було використано бібліотеку Pandas.

Прогнозування за допомогою модулю Prophet

Перед роботою над прогнозуванням було проведено операції з даними, для того, аби формат аналізованих даних відповідав вимогам Prophet. Єдиними читабельними для Prophet параметрами є ds та y, де перше - має формат datetime (залежно від набору даних, це може бути рррр.мм.дд гг.хх або рррр.мм.дд чи інше), а друге - формат int. Результати прогнозування будуть сформовані у таких же форматах.

Наступним важливим кроком було проведення підбору параметрів, що використовуються для побудови прогнозу. Для цього було проведено тестування ймовірних параметрів та з них обрано ті, що дали найкращі результати. В подальшому ці параметри використовуватимуться для тренування даних та створення прогнозу і виведення графіків.

Для тестування у Prophet уже вкладений скрипт, запустивши який, можна порівняти метрики для усіх ймовірних параметрів (параметри ми обираємо і прописуємо вручну, деталі нижче), скрипт моделі створює прогноз на встановлену кількість кроків, але не відображає сам прогноз, а відображає похибку, отриману в результаті цих прогнозів. Далі, всі вказані параметри разом з метриками виводяться таблицею, або ж одразу можна відобразити кращі результати. Чим нижче значення отриманих метрик, тим більша точність результатів.

Тестування проводилося для наступних параметрів: changepoint prior scale, seasonality prior scale, holidays prior scale, seasonality mode та changepoint_range:

changepoint prior scale – параметр, що модулює гнучкість автоматичного вибору точки зміни;

seasonality prior scale – параметр, що модулює силу моделі сезонності;

holidays prior scale – параметр, що модулює силу моделі компонентів свята, якщо не змінено у введених вихідних;

seasonality mode – "адитивний" або "множинний";

changepoint_range – частка історії, в якій будуть оцінені точки зміни тенденції. За замовчуванням вказано 0,8 для перших 80 "точок зміни".

Для подальшої роботи над побудовою прогнозу з Prophet використовувались значення, що дали найкращий результат, тобто ті, у яких rmse нижче.

RMSE є мірою точності, щоб порівняти помилки прогнозування різних моделей для певного набору даних, а не між наборами даних, оскільки він залежить від масштабу.

RMSE - квадратний корінь з моменту другого зразка відмінностей між прогнозованими значеннями та спостережуваними значеннями або середнє квадратичне цих відмінностей.

Ці відхилення називаються залишками, коли обчислення виконуються за вибіркою даних, яка була використана для оцінки, або похибками (або помилками передбачення), коли обчислюються поза вибіркою.

RMSE служить для об'єднання величин помилок у прогнозах для різних точок даних в єдиний показник потужності прогнозування.

	changepoint_prior_scale	seasonality_prior_scale	rmse
0	0.001	0.01	94.672467
1	0.001	0.10	95.257041
2	0.001	1.00	94.677506
3	0.001	10.00	95.161043
4	0.010	0.01	94.462488
5	0.010	0.10	95.033794
6	0.010	1.00	95.000020
7	0.010	10.00	94.993031
8	0.100	0.01	95.306646
9	0.100	0.10	95.815840
10	0.100	1.00	95.798392
11	0.100	10.00	95.832211
12	0.500	0.01	96.365254
13	0.500	0.10	96.765743
14	0.500	1.00	96.739093
15	0.500	10.00	96.768465

Рис. 8. Результат тесту для підбору значень параметрів.

Changepoint prior scale визначає гнучкість тенденції, і, зокрема, наскільки змінюється тренд в точках зміни тренду. Якщо він занадто малий, тренд буде недостатнім, і дисперсія, яку слід було змоделювати зі змінами

тренду, натомість буде оброблятися за допомогою терміну шуму. Якщо він занадто великий, тенденція буде переповнена, і в самому крайньому випадку можна буде отримати тренд, яка враховує річну сезонність.

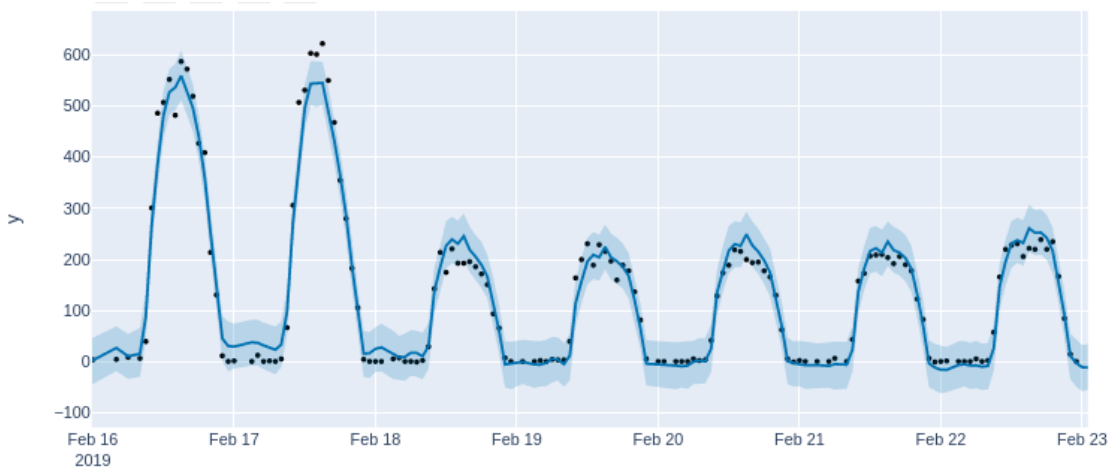
Seasonality prior scale контролює гнучкість сезонності. Так само, велике значення дозволяє сезонності відповідати великим коливанням, мале значення зменшує величину сезонності.

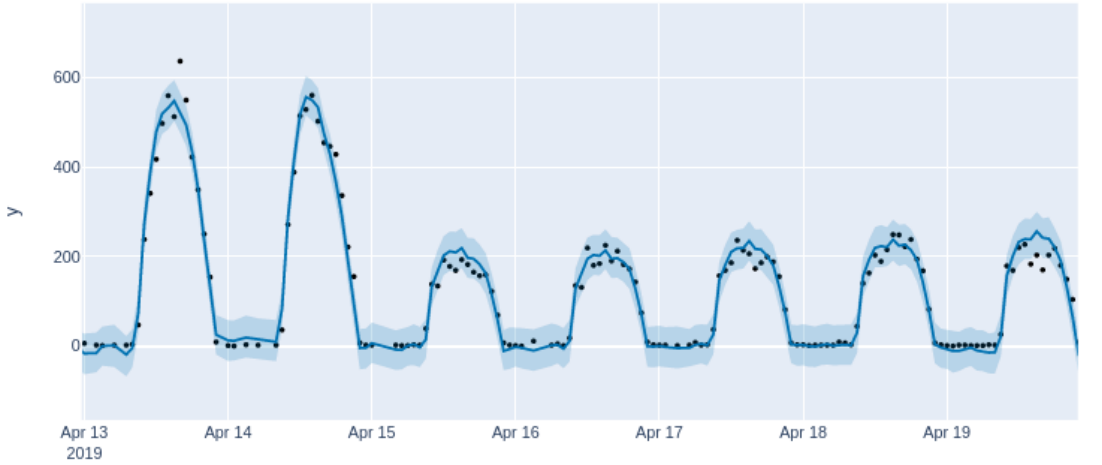
```
best_params = all_params[np.argmin(rmses)]
print(best_params)
```

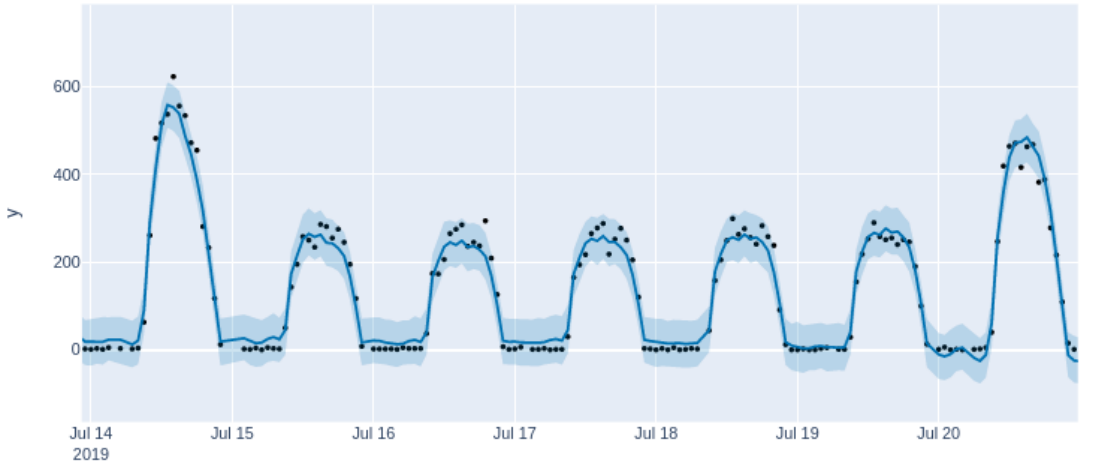
```
{'changepoint_prior_scale': 0.01, 'seasonality_prior_scale': 0.01}
```

Рис. 9. Найбільш точний результат можна отримати використавши ці значення параметрів.

На графіках, зображених на рисунку 10 відображені прогнози сформовані на 7 днів протягом року. Було обрано різні місяці для точнішої перевірки дієвості моделі Prophet.


y
600
500
400
300
200
100
0
−100
Feb 16
2019
Feb 17
Feb 18
Feb 19
Feb 20
Feb 21
Feb 22
Feb 23


y
600
400
200
0
Apr 13
2019
Apr 14
Apr 15
Apr 16
Apr 17
Apr 18
Apr 19


y
600
400
200
0
Jul 14
2019
Jul 15
Jul 16
Jul 17
Jul 18
Jul 19
Jul 20

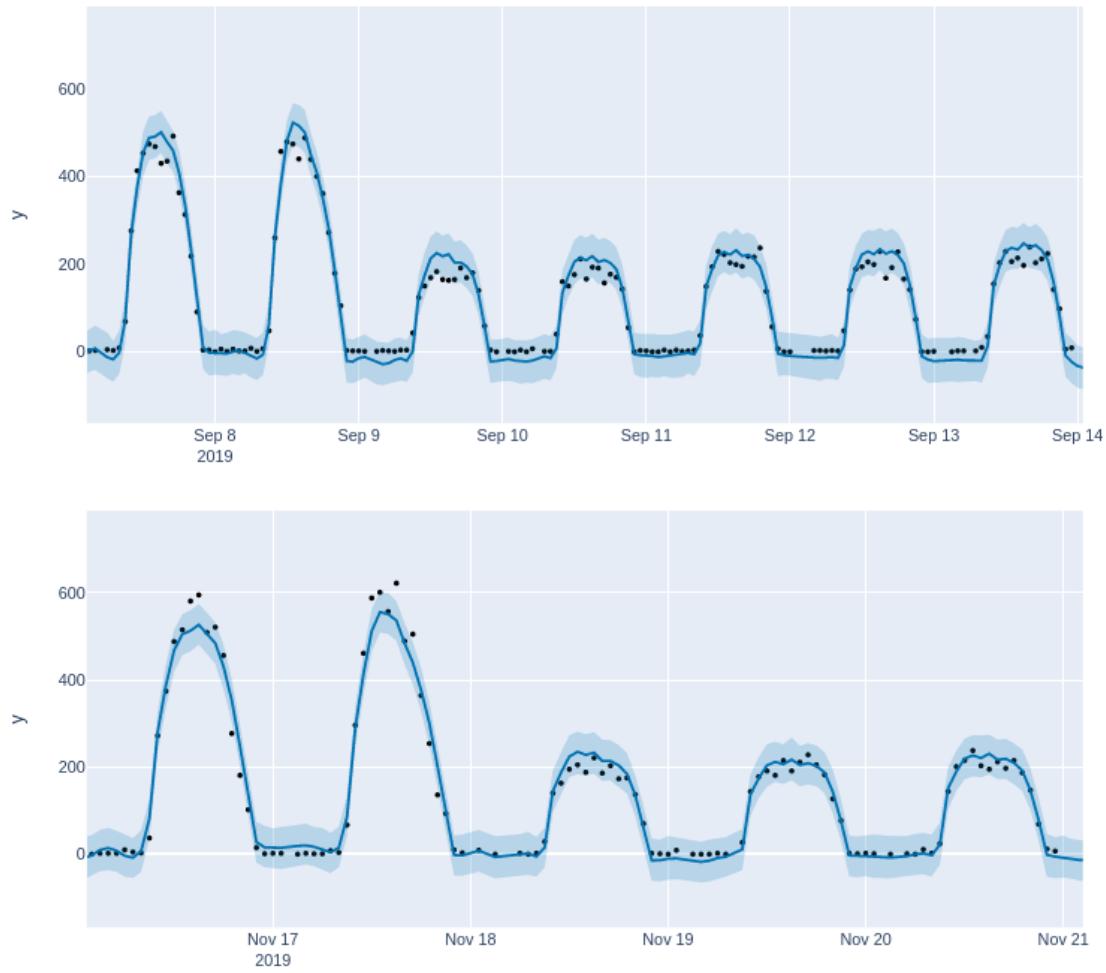
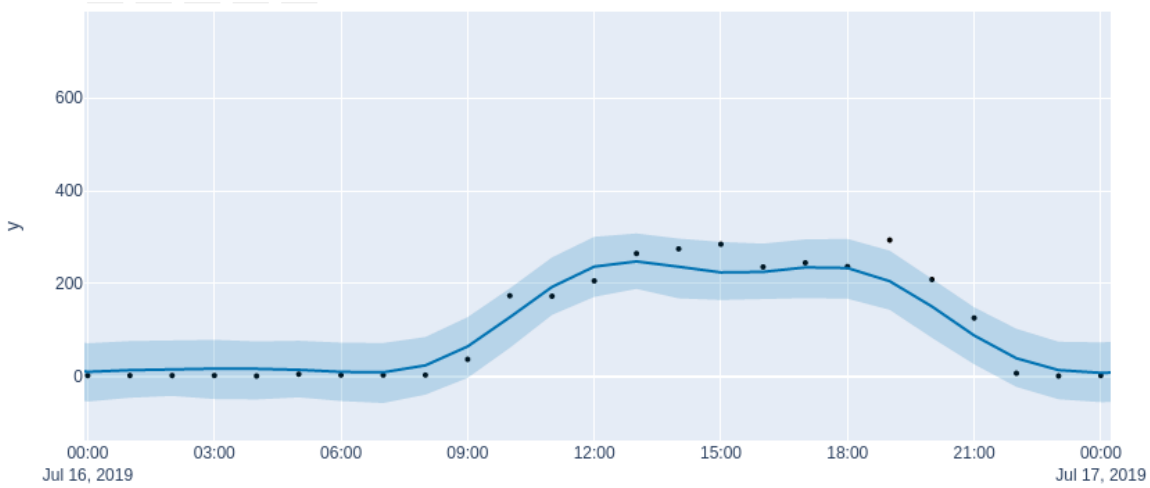
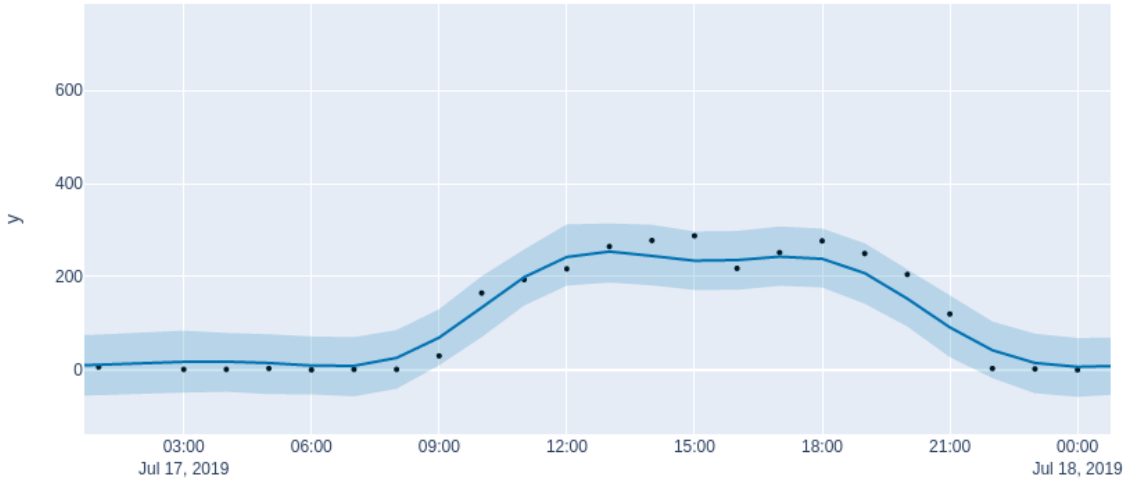
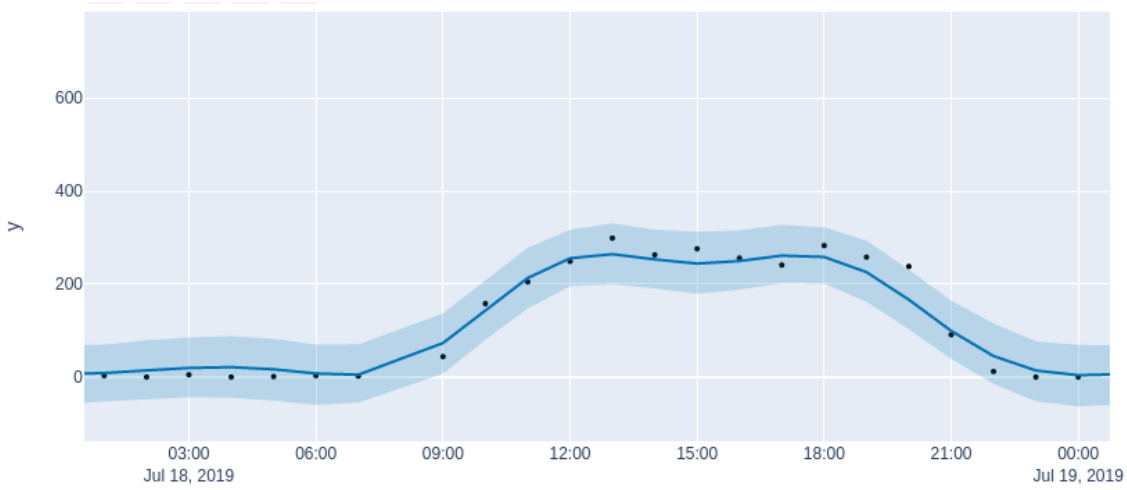


Рис. 10. Спрогнозовані дані з допомогою Prophet.

Як можна побачити, у звичні будні дні прогноз має більшу точність, ніж у вихідні.


600
400
200
0
y
00:00
Jul 16, 2019
03:00
06:00
09:00
12:00
15:00
18:00
21:00
00:00
Jul 17, 2019


600
400
200
0
y
03:00
Jul 17, 2019
06:00
09:00
12:00
15:00
18:00
21:00
00:00
Jul 18, 2019


600
400
200
0
y
03:00
Jul 18, 2019
06:00
09:00
12:00
15:00
18:00
21:00
00:00
Jul 19, 2019

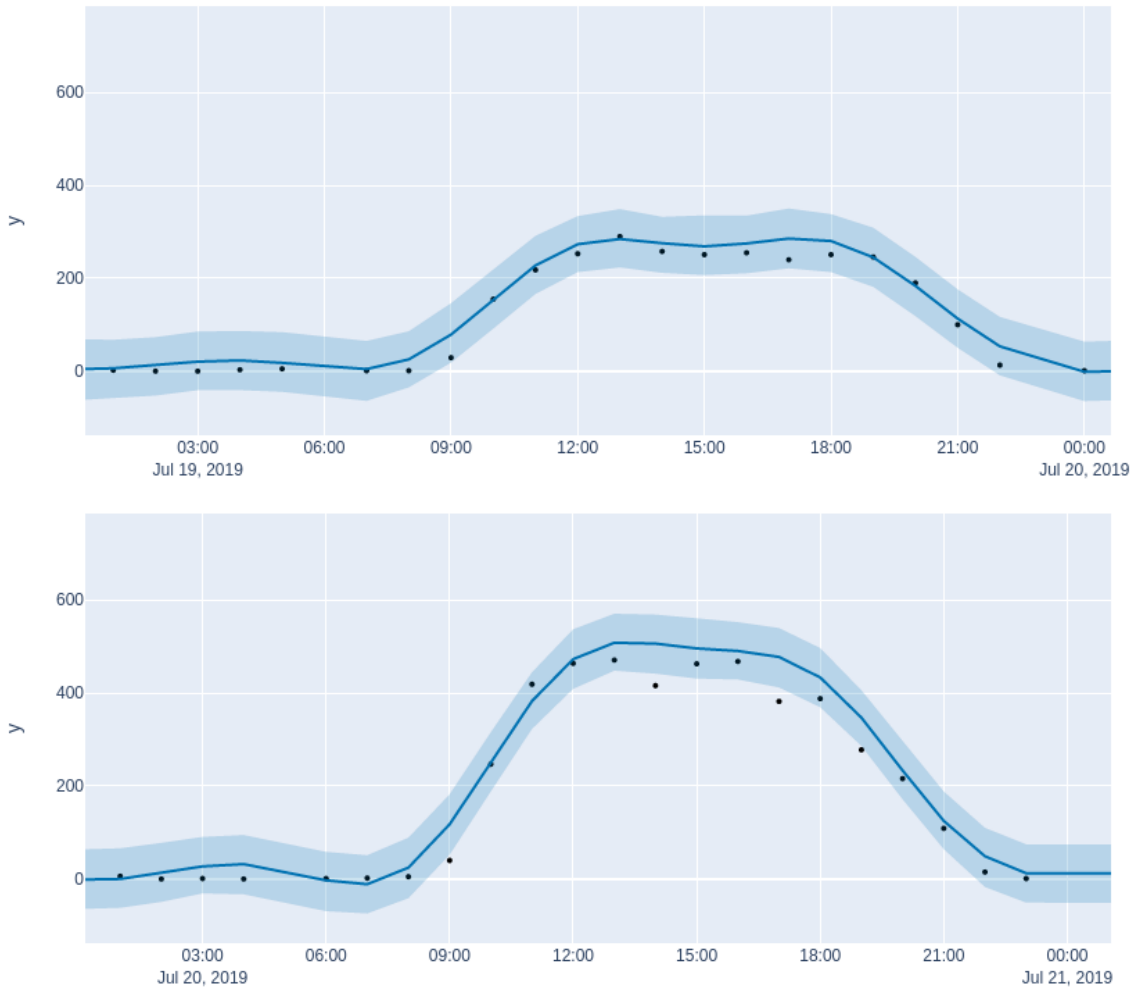


Рис. 11. Прогноз Prophet з детальнішим щоденним відображенням.

Сезонність у Prophet оцінюється за допомогою часткової суми Фур'є. Prophet за замовчуванням буде відповідати тижневим і річним сезонностям, якщо часовий ряд більше двох циклів. Він також відповідатиме добовій сезонності для тимчасових рядів. Сезонність відображаються на графіках компонентів на рисунку 12. За графіками очевидно, що під час вихідних, вечірніх годин спостерігається значне збільшення, яке повністю відсутнє в міжсезоння.

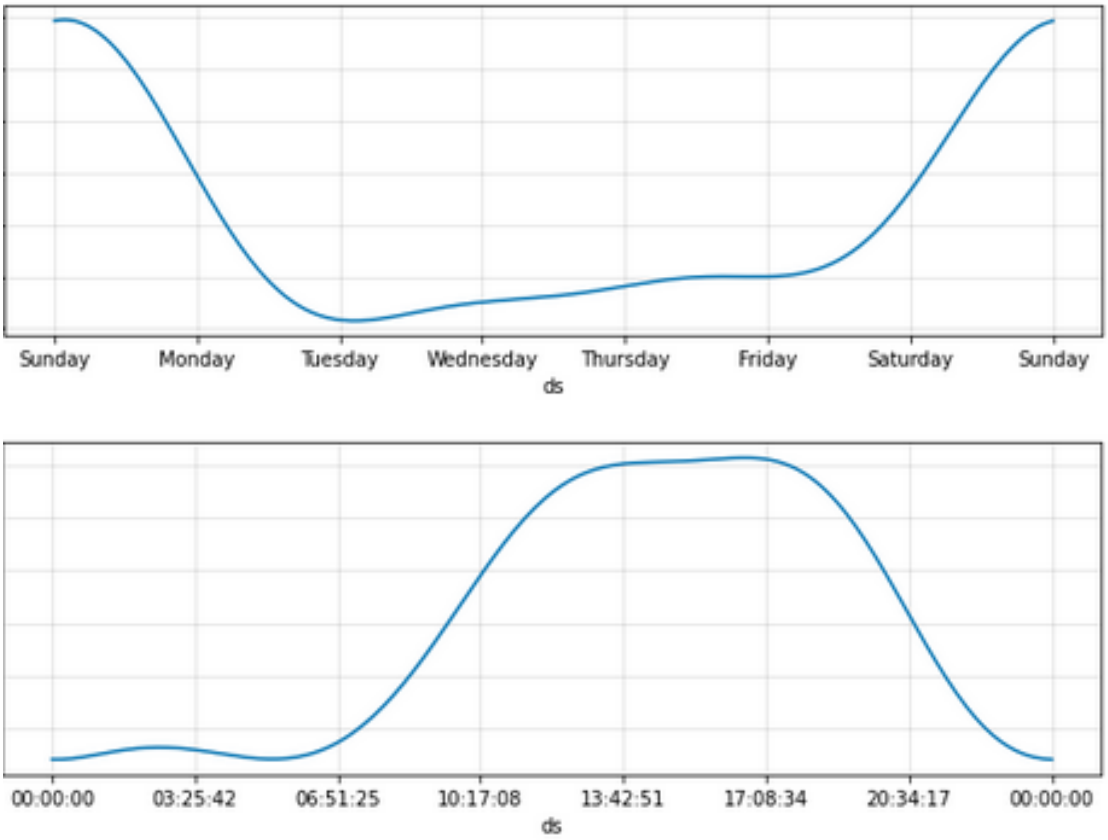


Рис. 12. Сезонність

Прогнозування за допомогою модулю NeuralProphet

Перевагою прогнозування за допомогою модулю NeuralProphet є швидкість налаштування моделі та простота у реалізації прогнозування.

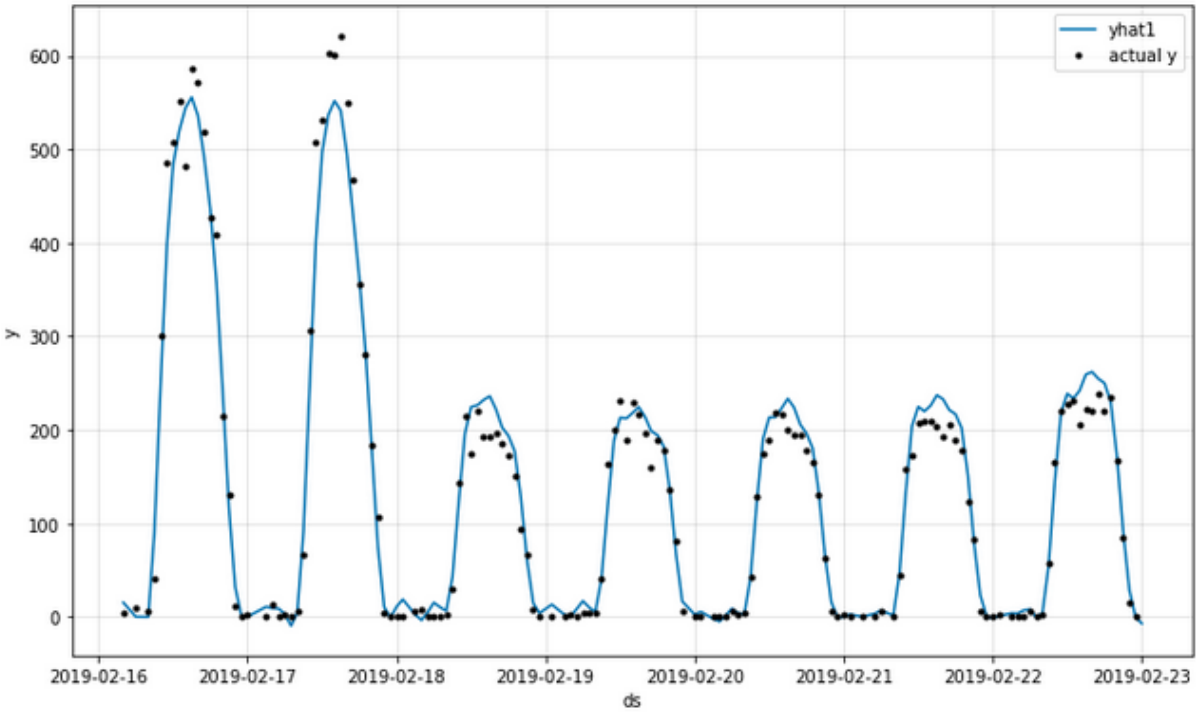
Перед роботою з прогнозування з NeuralProphet також проводився підбір параметрів за найкращими метриками, на основі яких було побудовано прогноз.

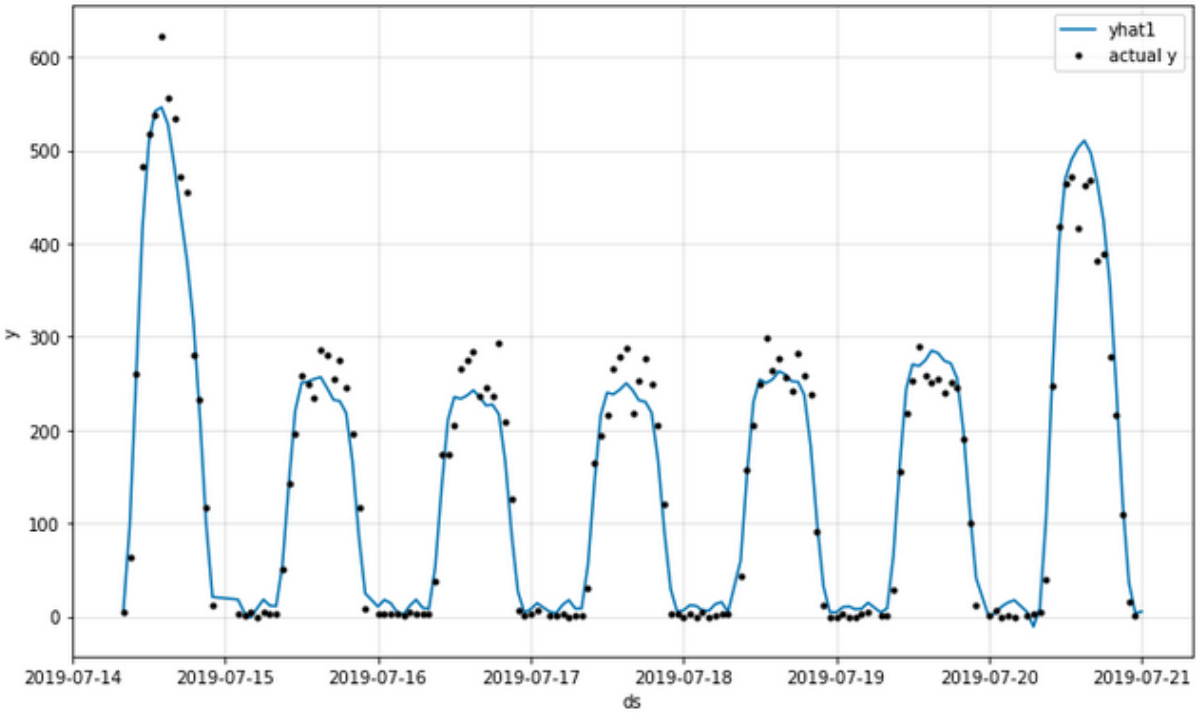
У наступному розділі буде детальніше про порівняння метрик з Prophet та NeuralPropher.

	horizon	mse	rmse	mae	mdape	smape	coverage
0	3 days 00:00:00	4640.128725	68.118490	51.597503	0.636583	0.955771	0.844583
1	3 days 01:00:00	4652.571850	68.209764	51.578133	0.637837	0.954728	0.843667
2	3 days 02:00:00	4665.176416	68.302097	51.568968	0.637837	0.955009	0.842126
3	3 days 03:00:00	4677.927622	68.395377	51.581002	0.636583	0.953913	0.841307
4	3 days 04:00:00	4688.138982	68.469986	51.556899	0.636583	0.952344	0.840488

Рис. 13. Метрики, отримані в результаті побудови прогнозу.

З використанням NeuralProphet було побудовано прогноз на 7 днів у різні місяці протягом року, для порівняння та аналізу роботи моделі у різних сезонах.


yhat1
actual y
600
500
400
300
200
100
0
y
2019-02-16
2019-02-17
2019-02-18
2019-02-19
2019-02-20
2019-02-21
2019-02-22
2019-02-23
ds


yhat1
actual y
600
500
400
300
200
100
0
y
2019-07-14
2019-07-15
2019-07-16
2019-07-17
2019-07-18
2019-07-19
2019-07-20
2019-07-21
ds

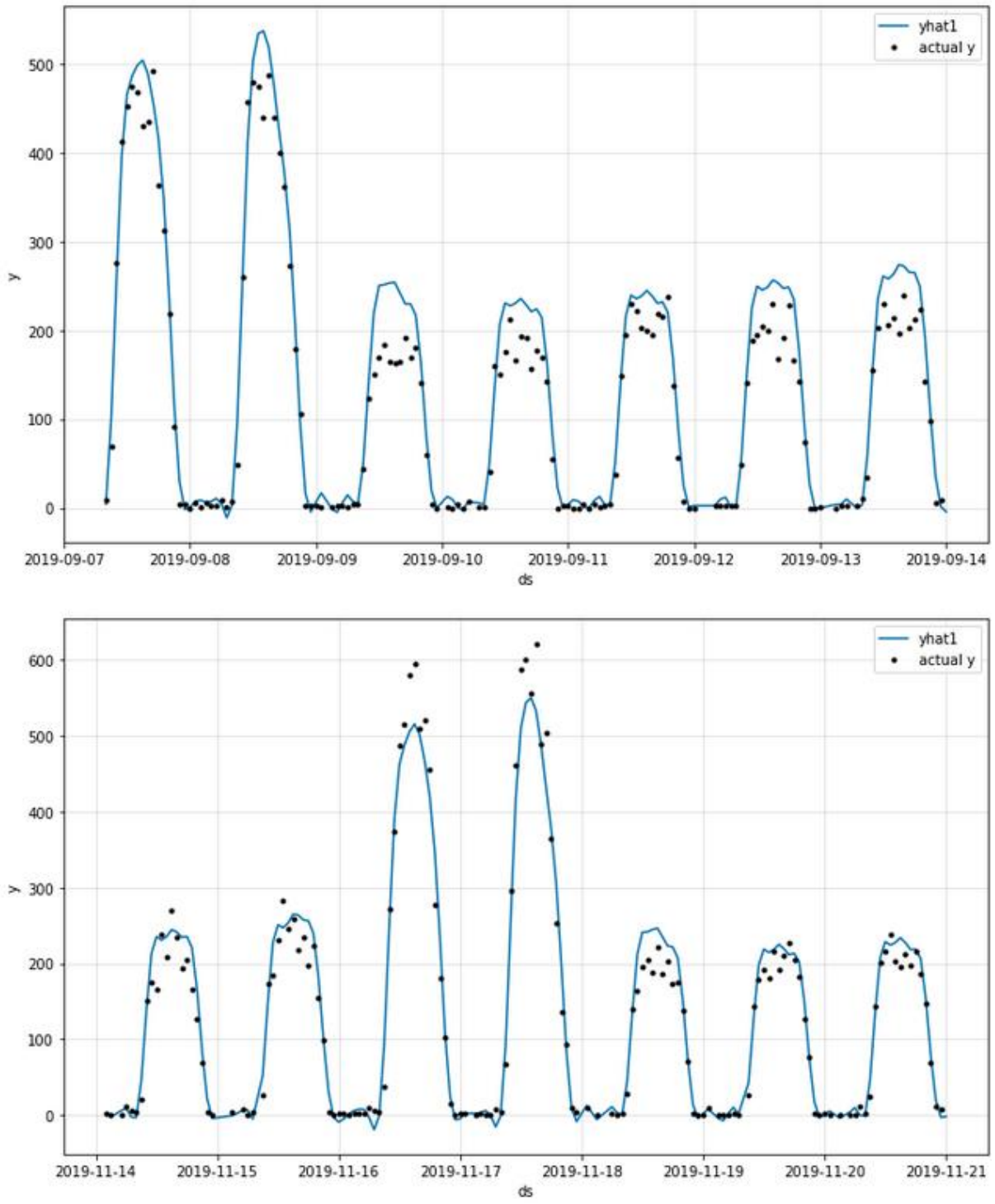


Рис. 14. Чорним кольором позначені актуальні дані, а синім - спрогнозовані.

Розгляд компонентів часових рядів на рисунку 15, вказує на наявність сезонності.

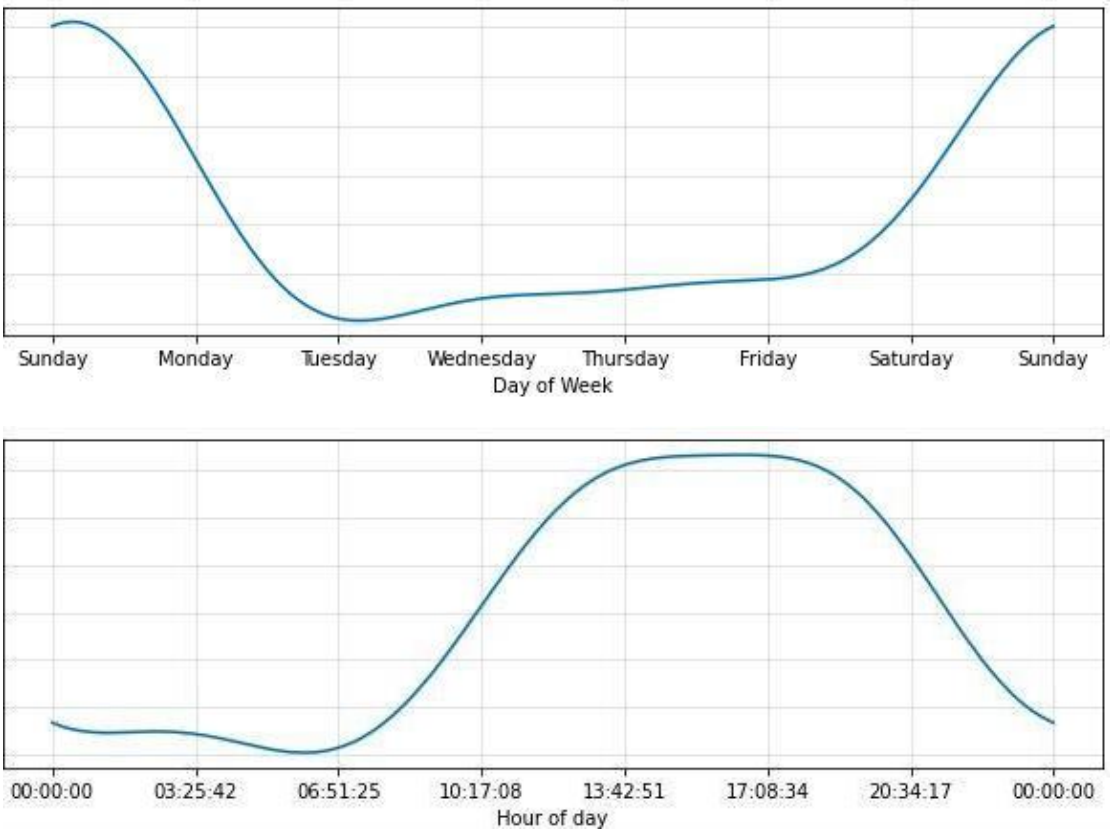
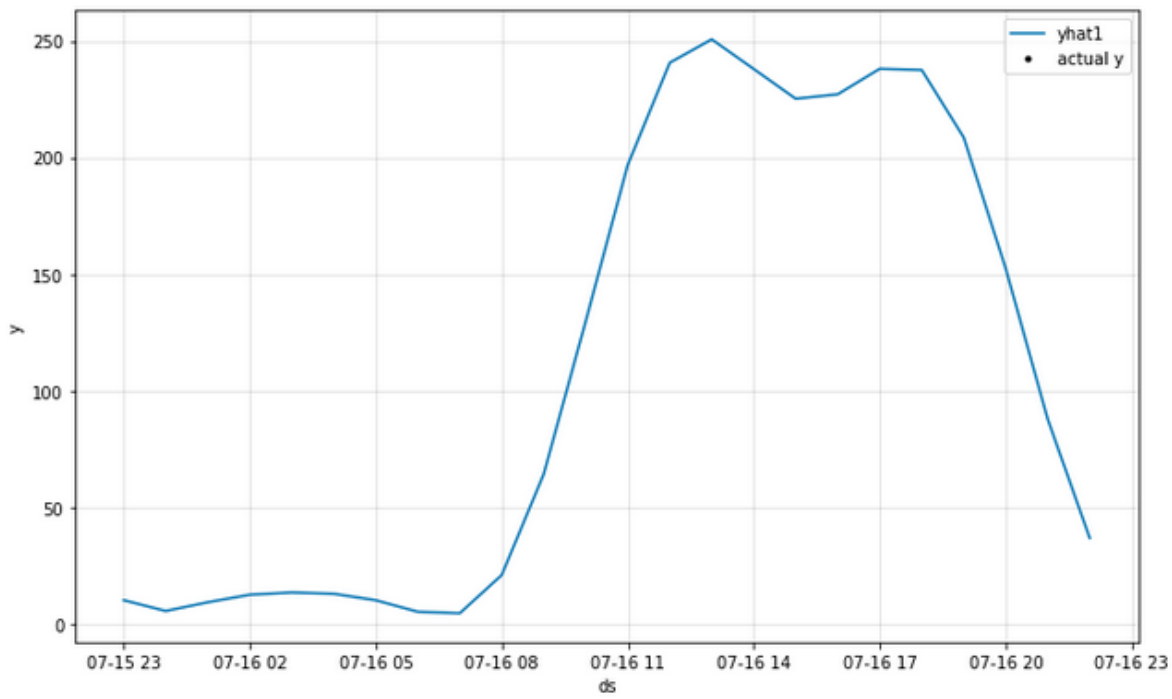
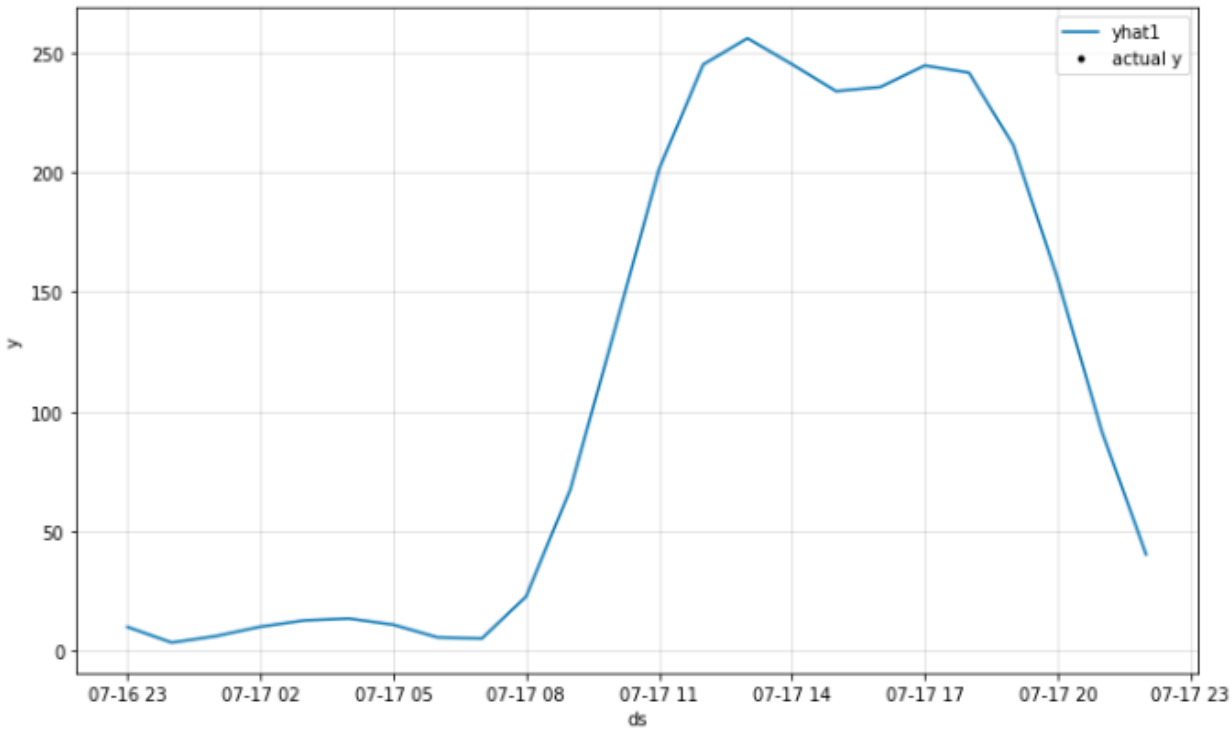
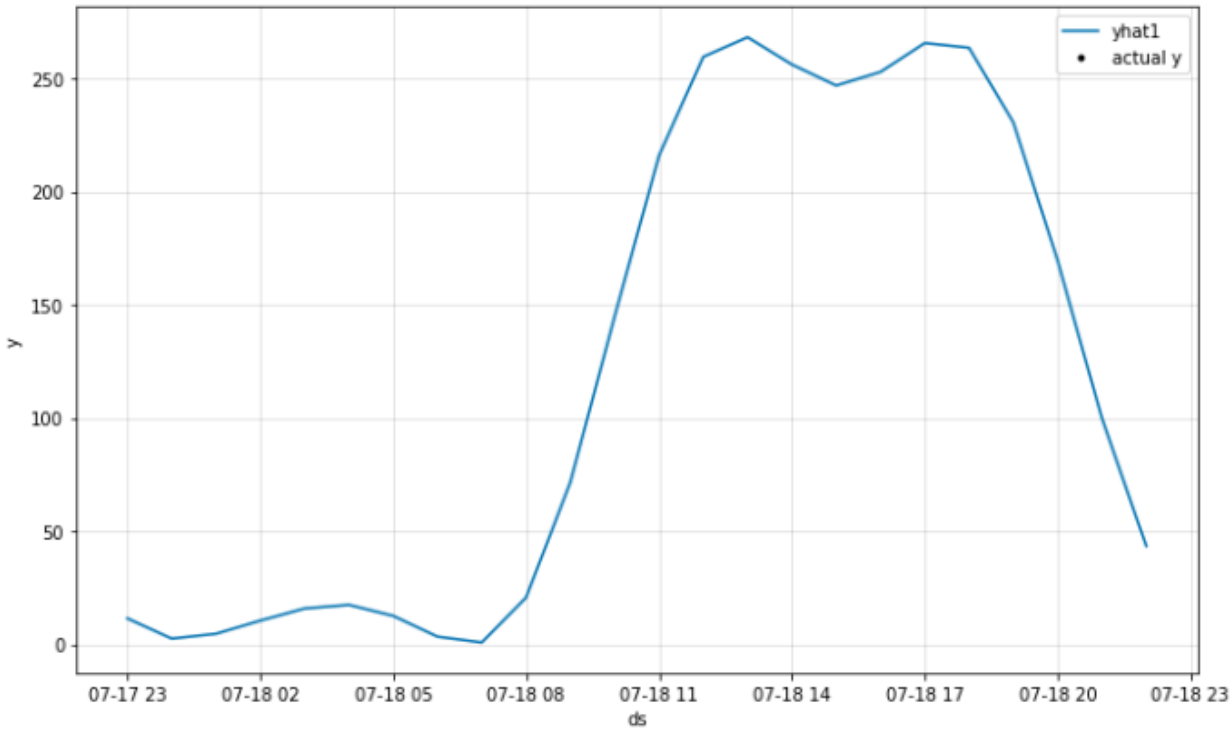


Рис. 15. Сезонність




yhat1
actual y
250
200
150
100
50
0
y
07-16 23
07-17 02
07-17 05
07-17 08
07-17 11
07-17 14
07-17 17
07-17 20
07-17 23
ds


yhat1
actual y
250
200
150
100
50
0
y
07-17 23
07-18 02
07-18 05
07-18 08
07-18 11
07-18 14
07-18 17
07-18 20
07-18 23
ds

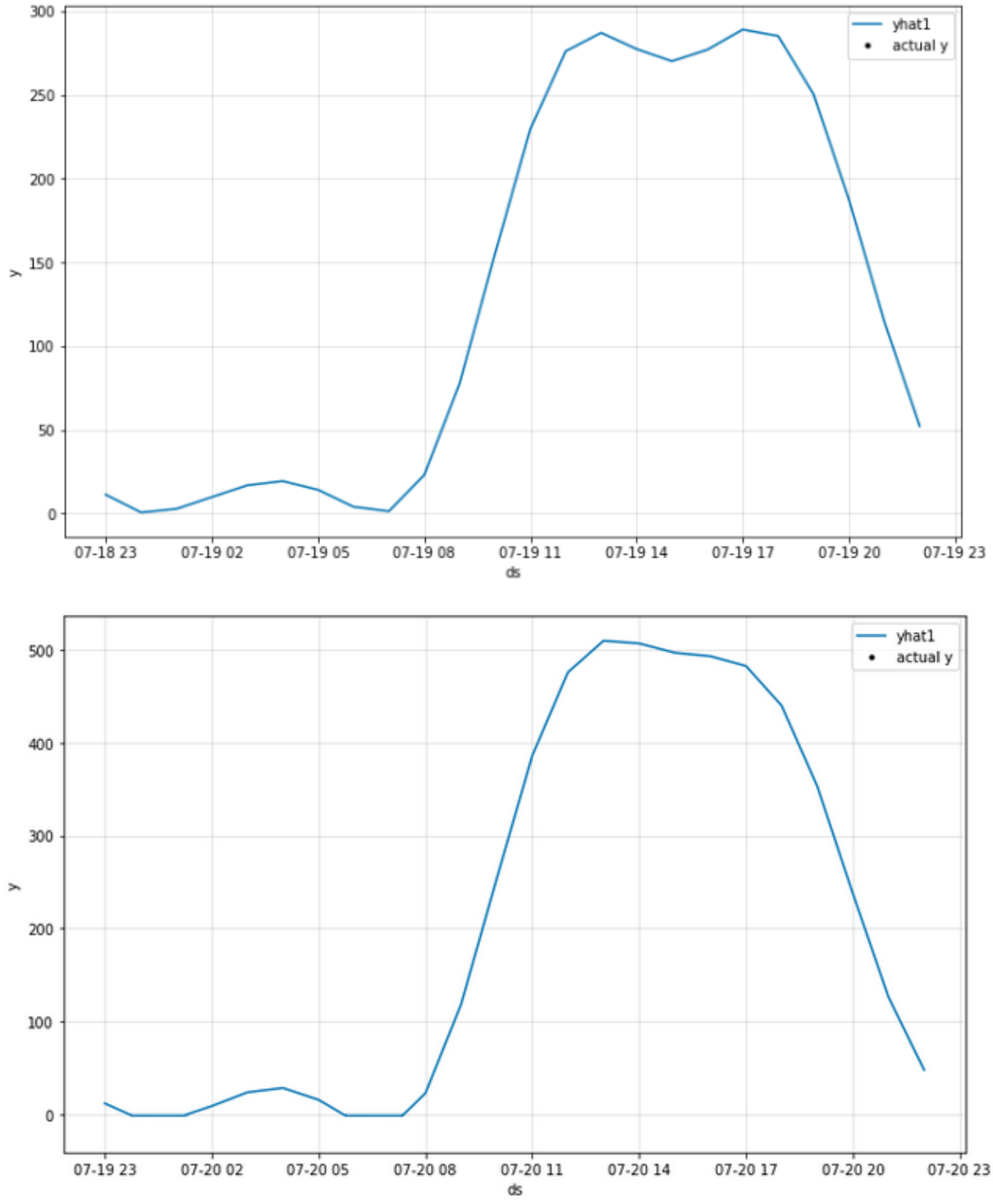


Рис. 16. Щоденний прогноз з використанням NeuralProphet

Аналіз та побудова даних з NeuralProphet відбувається значно швидше, порівняно з іншими модулями для прогнозування, що може бути великою перевагою у задачах, які потребують швидкого рішення та не потребують найкращої точності.

Порівняння модуля Prophet від Facebook з NeuralProphet

Для порівняння було визначено набір даних, моделі, які потрібно порівняти та встановлено їх гіперпараметри. Мною були використані ті гіперпараметри, які показали найвижчу точність у попередніх тестуваннях та при побудові прогнозів Prophet та NeuralProphet.

Виконавши скрипт для порівняння точності результатів роботи алгоритмів Prophet та NeuralProphet, було отримано значення обчислених статичних даних.

	data	model	params	MAE	MSE	MASE	RMSE
0	df	NeuralProphet	{'seasonality_mode': 'multiplicative', 'learni...	169.808716	50963.25	5.246953	225.750412
1	df	Prophet	{'yearly_seasonality': True, 'daily_seasonalit...	174.780899	52834.484375	5.400589	229.857529

Рис. 17. Обчислені статистичні дані.

Обчислені статистичні дані: середньоквадратична помилка (MSE), середньоквадратична помилка (RMSE), середня абсолютна помилка (MAE), середня абсолютна масштабована помилка (MASE).

	data	model	params	MASE	RMSE	MASE_std	RMSE_std	split
0	df	NeuralProphet	{'seasonality_mode': 'multiplicative', 'learni...	1.541820	78.985466	0.037883	8.834560	train
1	df	Prophet	{'yearly_seasonality': True, 'daily_seasonalit...	1.512693	73.988838	0.028856	7.936955	train
0	df	NeuralProphet	{'seasonality_mode': 'multiplicative', 'learni...	2.717139	143.655838	0.138568	10.694946	test
1	df	Prophet	{'yearly_seasonality': True, 'daily_seasonalit...	2.708968	140.987671	0.132671	20.908127	test

Рис. 18. Середнє значення показників і стандартне відхилення.

На рисунку 18 відображено метрику RMSE для Prophet та NeuralProphet. Це найкращі результати, що вдалося досягнути, проте метрика надалі є високою. В такому випадку, якщо RMSE розрахована і знайдена надто високою, є 4 варіанти вирішення проблеми:

- Знайти та видалити контрольні точки з великою RMSE, маючи на увазі, що це найменш точні точки (іноді це може призвести до виникнення ще більших помилок).
- Збільшити допуск RMSE.

- Збільшити складність функції трансформації, яка більш точно відповідатиме введеним точкам. RMSE точок при цьому зменшиться, однак використання складних криволінійних функцій може призвести до небажаних сильних спотворень прогнозу.
- Залишити лише точки, які гарантовано є правильними.

	data	model	params	MASE	RMSE
0	df	NeuralProphet	{'seasonality_mode': 'multiplicative', 'learni...	[2.9129484, 2.612483, 2.625985]	[135.3606, 158.75629, 136.85062]
1	df	Prophet	{'yearly_seasonality': True, 'daily_seasonalit...	[2.698858, 2.876274, 2.551771]	[124.88764, 170.51596, 127.5594]

Середня абсолютна масштабована помилка (MASE) - є мірою для визначення ефективності прогнозів, створених за допомогою алгоритму, шляхом порівняння прогнозів з результатами наївного підходу до прогнозування.

Середньоквадратична помилка (RMSE) - це квадратний корінь з дисперсії залишків. Нижчі значення RMSE вказують на кращу точність. RMSE є хорошим показником того, наскільки точно модель прогнозує відповідь, і це найважливіший критерій відповідності, якщо головною метою моделі є передбачення.

Розрахувавши значення помилок, та провівши порівняння можна дійти до висновку, результати прогнозування з використанням Prophet мають нижче значення помилок, а отже – більшу точність (Prophet - MASE 2.6, 2.8, 2.5, RMSE – 124.8, 170.5, 127.5; NeuralProphet – MASE 2.9, 2.6, 2.6, RMSE – 135.3, 158.7, 136.8).

Висновок

У дипломній роботі були розглянуті методи та моделі прогнозування кількості відвідувачів торгово-розважального центру. Це актуальна задача для бізнесу з декількох причин: по-перше, прогнозування кількості відвідувачів дозволяє краще розуміти тенденцію змін, і на цій основі будувати плани щодо закупівель, замовлень ресурсів чи тому подібне; по-друге, це дозволить ефективніше використовувати ресурси, підлаштовувати бізнес під попит, врешті решт - зекономити. Для бізнесу та його потреб прогнозування певних об'єктивних факторів може мати вирішальне значення для подальшого його існування, саме тому це є актуальним завданням і буде ним ще тривалий час.

У ході роботи над поставленим завданням було опрацьовано модуль PROPHET та NEURALPROPHET. Вибір впав саме на них, оскільки навчання даних на основі цих модулів займає до 10 хвилин, що дозволяє виконати завдання швидко та отримати наближені до реальних результати. Ці модулі можна використовувати не тільки для прогнозування кількості відвідувачів, а й для інших потреб побудови передбачень, "підбиваючи" роботу модулів під себе, обравши необхідні параметри.

Працюючи з раніше згаданими модулями вдалося досягнути відносно точних результатів, з низьким рівнем помилок. Дані побудовані моделі можна удосконалити, додавши майбутні ймовірні святкові дні (або дні, у які очікуються події, що приведуть до збільшення потоку відвідувачів, так як концерти, акції, розважальні заходи), що впливатимуть на тренд - це зробить прогнозовані дані ще більш точними.

При роботі з прогнозуванням кращі результати вдалось отримати з PROPHET (за результатами отриманих метрик Prophet - MASE 2.6, 2.8, 2.5, RMSE – 124.8, 170.5, 127.5; NeuralProphet – MASE 2.9, 2.6, 2.6, RMSE – 135.3, 158.7, 136.8). У випадку, якщо поставлена задача містить побудову прогнозу на основі аналізу величезної кількості даних (що становить,

приблизно, від 8000 одиниць даних і більше), то Prophet втрачатиме свою ефективність. Тоді доцільно використовувати NeuralProphet, оскільки він був створений саме для таких задач.

Список використаної літератури

1. Di W. Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling / W. Di, A. Bhardwaj, J. Wei., 2018. – 284 c.

2. Freidma J. The Elements of Statistical Learning / J. Freidma, T. Robert, H. Trevor., 2009.

3. Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists / S. Guido, A. Müller., 2016.

4. Han, Kamber & Pei. Data Mining: Concepts and Techniques, Third Edition / Han, Kamber & Pei., 2013.

5. Heydt M. Learning Pandas – Python Data Discovery and Analysis Made Easy / Michael Heydt.

6. Kumar A. Python: Advanced Predictive Analytics / A. Kumar, J. Babcock., 2017. – 660 c.

7. Miller T. Modeling Techniques in Predictive Analytics with Python and R: A Guide to Data Science / Thomas Miller. – 448 c.

8. Raschka S. Python Machine Learning - Second Edition / S. Raschka, V. Mirjalili., 2017. – 622 c. – (Packt).

9. Prophet documentation: https://facebook.github.io/prophet/docs/

10. Wes M. Python for Data Analysis.

11. NeuralProphet: https://neuralprophet.com/ Explainable Forecasting at Scale.

13. Hyndman R., Athanasopoulos G. Forecasting: Principles and Practice. Monash University, Australia, 2018.

14. Christopher F. Baum. An Introduction to Modern Econometric Using Stata, 2006.

15. Sean J. Taylor. Forecasting at Scale, 2017

16. Rob J Hyndman and George Athanasopoulos. Forecasting: Principles and Practice (3rd ed)

17. Jupiter documentation https://jupyter.org

18. Prophet documentation https://facebook.github.io/prophet/

19. NeuralProphet documentation https://neuralprophet.com/html/index.html

Додаток

```
import pandas as pd
from prophet import Prophet
from matplotlib import pyplot as plt
from datetime import datetime, timezone
import pickle
import xlrd
from prophet.plot import plot_plotly, plot_components_plotly, plot_yearly
from prophet.diagnostics import cross_validation
from prophet.plot import plot_cross_validation_metric
from fbprophet.diagnostics import performance_metrics
from fbprophet import Prophet
from neuralprophet import NeuralProphet
from neuralprophet.benchmark import SimpleExperiment,
CrossValidationExperiment
from neuralprophet.benchmark import ManualBenchmark,
ManualCVBenchmark
from neuralprophet import NeuralProphet, set_log_level
from neuralprophet.benchmark import Dataset, NeuralProphetModel,
ProphetModel
from neuralprophet.benchmark import SimpleBenchmark,
CrossValidationBenchmark
set_log_level("ERROR")
xls = pd.ExcelFile('temp.xls') #name of file
xls.sheet_names
df = pd.read_excel(xls,  'Ашан |Блок 2 (вх. №7)|')
df.to_csv('csvfile.csv', encoding='utf-8', index=False)
df = df[['Unnamed: 7', 'Unnamed: 8']]
df = df.drop([0, 1, 2, 3, 4, 5, 6])
```

```
df = df[['Unnamed: 7', 'Unnamed: 8']]
df.dropna(inplace=True)
df.columns = ['ds', 'y']
df['ds'] = pd.to_datetime(df['ds'])
df['y'] = df['y'].astype(float)
df.dtypes
plt.plot(df['ds'], df['y'])
plt.show()
#import numpy as np
#df['y'] = df['y'] + 1
#df['y'] = np.log(df['y'])
#get forecast
df = df.drop_duplicates(subset='ds', keep='first')
Dataset_list = [
    Dataset(df, name = "df", freq = "H"),
    # Dataset(df, name = "df", freq = "D"),
    # Dataset(df, name = "df", freq = "5min"),
 ]
model_classes_and_params = [
    (NeuralProphetModel, {"seasonality_mode": "multiplicative",
"learning_rate": 0.1, "changepoints_range": 0.1, "epochs": 1000}),
    (ProphetModel, {"yearly_seasonality": True, "daily_seasonality": True,
"weekly_seasonality": True, "changepoint_prior_scale": 0.01,
"seasonality_prior_scale": 0.01})
]
model_classes_and_params
benchmark = SimpleBenchmark(
    model_classes_and_params=model_classes_and_params, # iterate over this
list of tuples
```

```
    datasets=dataset_list, # iterate over this list
    metrics=["MAE", "MSE", "MASE", "RMSE"],
    test_percentage=25,
)
results_train, results_test = benchmark.run()
results_test
benchmark_cv = CrossValidationBenchmark(
    model_classes_and_params=model_classes_and_params, # iterate over this
list of tuples
    datasets=dataset_list, # iterate over this list
    metrics=["MASE", "RMSE"],
    test_percentage=10,
    num_folds=3,
    fold_overlap_pct=0,
)
results_summary, results_train, results_test = benchmark_cv.run()
results_summary
df = results_summary[results_summary['data'] == 'df']
df = df[df['split'] == 'test']
plt = df.plot(x='model', y='RMSE', kind='barh')
results_test
# Python
import itertools
import numpy as np
import pandas as pd
param_grid = {
    'changepoint_prior_scale': [0.001, 0.01, 0.1, 0.5],
    'seasonality_prior_scale': [0.01, 0.1, 1.0, 10.0],
    'yearly_seasonality': [True, False],
```

```
    'daily_seasonality': [True, False],
    'weekly_seasonality': [True, False],
 #   'growth': ["linear", "logistic"]
}
# Generate all combinations of parameters
all_params = [dict(zip(param_grid.keys(), v)) for v in
itertools.product(*param_grid.values())]
rmses = []  # Store the RMSEs for each params here
# Use cross validation to evaluate all parameters
for params in all_params:
    m = Prophet(**params).fit(df)  # Fit model with given params
    df_cv = cross_validation(m, initial = 100, period = 100, horizon="100 days")
    df_p = performance_metrics(df_cv, rolling_window=1)
    rmses.append(df_p['rmse'].values[0])
# Find the best parameters
tuning_results = pd.DataFrame(all_params)
tuning_results['rmse'] = rmses
print(tuning_results)
best_params = all_params[np.argmin(rmses)]
print(best_params)
m = Prophet( yearly_seasonality=120, daily_seasonality=24,
weekly_seasonality=28, changepoint_prior_scale=0.01,
n_changepoints=int(len(df) / 12),
          seasonality_prior_scale=0.01)
m.fit(df)
future = m.make_future_dataframe(periods=120, freq="H")
fcst = m.predict(future)
```

```
fig = m.plot(fcst)
lot_plotly(m, fcst)
m = NeuralProphet(
    growth="linear",
    changepoints= automatic,
    n_changepoints=int(len(df) / 12),
    changepoints_range=0.1,
    trend_reg=0,
    trend_reg_threshold=False,
    yearly_seasonality=False,
    weekly_seasonality=28,
    daily_seasonality=True,
    seasonality_mode="additive",
    seasonality_reg=0.1,
    n_forecasts=30,
    n_lags=0,
    learning_rate=0.1,
    epochs=2000,
    loss_func="MSE",
    normalize="minmax,
    impute_missing=True,
)
m.add_country_holidays(country_name='Ukraine')
m.fit(df, freq='auto')

future = m.make_future_dataframe(df, periods=120,
n_historic_predictions=True)
```

```
forecast = m.predict(future)
forecast.head()
plot1 = m.plot(forecast)
fig2 = m.plot(forecast[-4*38:])
```