Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет *інфокомунікацій*
(повна назва)

Кафедра *інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти *другий (магістерський)*

Модель прогнозу для даних про продажі на основі керованих веб-сервісів Amazon
(тема)

Виконав:
студент *2* курсу, групи *ІМІм-20-2*
Юр'єв Я.В.
(прізвище, ініціали)

Спеціальність *172 «Телекомунікації та радіотехніка»*
(код і повна назва спеціальності)

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *«Інформаційно-мережна інженерія»*
(повна назва освітньої програми)

Керівник *доц. Кривенко С.А.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2022 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____

Керівник_____

Харківський національний університет радіоелектроніки

Факультет *інфокомунікацій*
(повна назва)

Кафедра *інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

Спеціальність *172 «Телекомунікації та радіотехніка»*
(код і повна назва)

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *«Інформаційно-мережна інженерія»*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___»_____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Юр'єву Ярославу Валерійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Модель прогнозу для даних про продажі на основі керованих веб-сервісів Amazon*

затверджена наказом університету від *14 березня 2022* року № *379* Ст
2. Термін подання студентом роботи до екзаменаційної комісії *26.05.2022*
3. Вихідні дані до роботи: *Провести аналіз керованих веб-сервісів машинного навчання Amazon, які надають послуги навчання і розгортання моделі прогнозу та розробити таку модель для прогнозування з подальшим тестуванням на заготовлених експериментальних даних.*
4. Перелік питань, що потрібно опрацювати в роботі:
   1) Машинне навчання для прогнозування;
   2) Аналіз веб-сервісів машинного навчання Amazon для прогнозування;
   3) Модель прогнозування продажів;
   4) Тестування створення прогнозів через Amazon Forecast.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри _____

_____

_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доц. Кривенко С.А.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	*21.03*	*Виконано*
2	*Підбір літератури за темою роботи.*	*28.03*	*Виконано*
3	*Виконання розділу 1*	*04.04*	*Виконано*
4	*Виконання розділу 2*	*11.04*	*Виконано*
5	*Виконання розділу 3*	*18.04*	*Виконано*
6	*Виконання розділу 4*	*18.05*	*Виконано*
7	*Оформлення презентаційного матеріалу, підготовка до захисту у ЕК*	*19.05*	*Виконано*

Дата видачі завдання 14.03.2022

Студент _____ _____
(підпис) (прізвище, ініціали)

Керівник роботи _____ *доц. Кривенко С.А.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 114 стор., 29 рис., 28 табл., 89 лістингів, 6 джерел.

МАШИННЕ НАВЧАННЯ, ПРОГНОЗ, AWS, AMAZON SAGEMAKER, AMAZON FORECAST, ПРОДАЖІ

Об’єкт дослідження – Amazon Forecast.

Мета роботи – аналіз керованих веб-сервісів Amazon, що пропонують послуги машинного навчання для прогнозування, розробка моделі прогнозу продажів для роздрібної торгівлі і подальше її тестування на заготовлених експериментальних даних про попит з використанням сервісу Amazon Forecast.

Результати – в роботі зроблено огляд використання машинного навчання для створення прогнозів у різних галузях підприємництва, зокрема у роздрібній торгівлі, наведено різні алгоритми прогнозування, можливі виклики в галузі і їх рішення. Проведено аналіз керованих веб-сервісів Amazon, які використовуються для навчання і розгортання моделей прогнозування. Також розроблено експериментальну модель для прогнозу продажів, і потім виконано її тестування на заготовлених даних онлайн-магазину про попит на товари. У кінці ми порівняли отримані дані прогнозу порівняні з реальними даними за потрібний період і виявлена кореляція між ними.

ABSTRACT

Explanatory note: 114 pages, 29 figures, 28 tables, 89 listings, 6 sources.

MACHINE LEARNING, FORECAST, AWS, AMAZON SAGEMAKER, AMAZON FORECAST, SALES

Object of research – Amazon Forecast.

The purpose of the work is to analyze Amazon managed web services that offer machine learning services for forecasting, develop a sales forecast model for retail and further test it on the prepared experimental demand data using the Amazon Forecast service.

Results – the paper reviews the use of machine learning to create forecasts in various sectors of business, including retail, provides different forecasting algorithms, possible challenges in the industry and their solutions. An analysis of Amazon managed services used to train and deploy forecasting models has been conducted. An experimental model for sales forecasting was also developed, and then tested on the prepared demand data of the online store for goods. In the end, we compared the obtained forecast data with real data for the required period and found a correlation between them.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AI – Artificial intelligence – штучний інтелект;

ML – Machine Learning – машинне навчання;

RNN – Recurrent Neural Network – рекурентна нейронна мережа;

LSTM – Long Short-Term Memory – довга короткочасна пам'ять;

MLP – Multi-Layer Perceptron – багатошаровий персептрон;

ARIMA – AutoRegressive Integrated Moving Average – авторегресивне інтегроване ковзне середнє;

BNN – Bayesian Neural Network – Байєсівська нейронна мережа;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

NLP – Natural Language Processing – обробка природної мови;

Amazon EMR – Amazon Elastic MapReduce;

CI/CD – Continuous Integration/Continuous Delivery – безперервна інтеграція/безперервна доставка;

AWS – Amazon Web Services;

SDK – Software Development Kit – комплект для розробки програмного забезпечення;

Amazon C2 – Amazon Elastic Compute Cloud;

Amazon S3 – Amazon Simple Storage Service;

IDE – Integrated Development Environment – інтегроване середовище розробки

CLI – Command Line Interface – інтерфейс командного рядка;

UTC – Universal Coordinated Time – всесвітній координований час;

ETS – Exponential Smoothing – експоненціальне згладжування;

NPTS – Non-Parametric Time Series – непараметричний часовий ряд;

API – Application Programming Interface – інтерфейс прикладного програмування;

RMSE – Root Mean Square Error – середньоквадратична помилка;

JSON – JavaScript Object Notation – запис об'єктів JavaScript.

ВСТУП

Штучний інтелект вже багато років допомагає вирішувати проблеми бізнесу. Тим не менш, успіх ініціатив в галузі штучного інтелекту та машинного навчання залежить від розробки алгоритмів, здатних навчатися методом проб і помилок, щоб покращувати продуктивність з часом. У багатьох випадках існуючі бізнес-операції, які доповнюють процеси штучного інтелекту та машинного навчання, засновані на логічних інструкціях і правилах «якщо-то», або дотримуються матриці рішень.

У міру розвитку технологій застосування штучного інтелекту та машинного навчання буде продовжуватись революційна зміна світу бізнесу. Наприклад, використання AI та ML у прогнозуванні викликає величезний інтерес для більшості підприємств через його зручність використання в різних функціях. Традиційно підприємства покладалися на методи статистичного прогнозування, такі як експоненціальне згладжування та лінійні регресії, щоб керувати своїми рішеннями. Проте прогнозування на основі машинного навчання замінило традиційні методи в багатьох ініціативах щодо даних та аналітики в галузях і секторах [1].

Метою роботи є аналіз керованих веб-сервісів Amazon, що пропонують послуги машинного навчання для прогнозування, розробка моделі прогнозу продажів для роздрібної торгівлі і подальше її тестування на заготовлених експериментальних даних про попит з використанням сервісу Amazon Forecast.

Тема кваліфікаційної роботи є актуальною, так як тут розглядаються сучасні виклики і їх рішення для бізнесу.

1 МАШИННЕ НАВЧАННЯ І ПРОГНОЗУВАННЯ

Сучасний ринковий темп вимагає відповідної конкурентної переваги. Тобто, якщо компанія прагне зайняти міцну, прибуткову позицію серед бізнес-конкурентів у своїй ніші. Для цього можна (і потрібно) використовувати відповідні технології, які допоможуть залишатися принаймні на крок попереду своїх конкурентів [2].

Прогнозування даних пройшло довгий шлях після того, як були представлені неймовірні технології, що прискорюють обробку даних, як-от машинне навчання. Прогнозні моделі, засновані на ML, сьогодні можуть враховувати залежні від часу компоненти – сезонність, тенденції, цикли, нерегулярні компоненти тощо – щоб максимізувати точність прогнозів і прогнозів на основі даних.

Це називається прогнозуванням за допомогою машинного навчання, і воно може бути найбільш вигідним для всіх видів бізнесу, включаючи прогнозування продажів і попиту, прогнозування набору кадрів, прогноз погоди, прогнозування споживання контенту; прогнозне планування та обслуговування тощо.

1.1 Що таке прогнозування часових рядів у машинному навчанні

Часовий ряд – це певна послідовність даних спостережень, які система збирає протягом певних періодів часу – наприклад, щоденно, щомісяця чи щорічно. Спеціалізовані моделі використовуються для аналізу зібраних даних часових рядів – описують та інтерпретують їх, а також роблять певні припущення на основі зрушень і шансів у колекції. Ці зрушення та шанси можуть включати зміну тенденцій, сезонні стрибки попиту, певні повторювані зміни або несистематичні зміни звичайних моделей тощо.

Усі дані, зібрані раніше, нещодавно та в даний час, використовуються як вхідні дані для прогнозування часових рядів, де майбутні тенденції, сезонні зміни, порушення тощо розробляються на основі складних математичних алгоритмів. А завдяки машинному навчанню прогнозування часових рядів стає

швидшим, точнішим та ефективнішим у довгостроковій перспективі. Доведено, що ML допомагає краще обробляти як структуровані, так і неструктуровані потоки даних, швидко фіксуючи точні шаблони в масивах даних.

Можна з упевненістю сказати, що принципи машинного навчання часових рядів в основному перевершують класичний підхід прогнозування часових рядів. Таким чином, традиційні методи обмежуються обробкою лише попередньо зібраної, легкодоступної історії попиту. У свою чергу, ML виконує автономне визначення точки інтересу в необмеженому потоці даних, щоб потім узгодити їх із наявними даними клієнтів і провести аналіз «що, якщо». Це призводить до особливо ефективних заходів щодо стимулювання попиту, наприклад, у комерційному секторі.

Однак цей складний підхід прогнозування вигідно використовується в багатьох аспектах управління та оптимізації бізнесу в більшості різних ніш.

1.2 Застосування прогнозування часових рядів машинного навчання

Практично будь-яка компанія чи організація, які мають справу з постійно генерованими даними та потребою адаптуватися до операційних зрушень і змін, можуть використовувати прогнозування часових рядів. Машинне навчання слугує найкращим прискорювачом, дозволяючи краще обробляти дані з наведених нижче галузей (рис. 1.1).

Рисунок 1.1 – Галузі застосування машинного навчання для прогнозування

Прогнозування курсів акцій – дані про історію курсів акцій у поєднанні з даними як про регулярні, так і нерегулярні стрибки та падіння фондового ринку можуть бути використані для отримання глибоких прогнозів щодо найбільш ймовірних майбутніх змін курсу акцій.

Прогнозування попиту та продажів – дані моделей поведінки клієнтів разом із вхідними даними з історії покупок, часової шкали попиту, сезонного впливу тощо, дають змогу моделям ML визначити найбільш потенційно затребувані продукти та досягти позиції на динамічному ринку.

Прогнозування веб-трафіку – загальні дані про звичайні показники трафіку серед веб-сайтів-конкурентів об'єднуються з вхідними даними про моделі, пов'язані з трафіком, щоб передбачити швидкість веб-трафіку протягом певних періодів.

Прогноз клімату та погоди – дані, засновані на часі, регулярно збираються з численних взаємопов'язаних метеостанцій по всьому світу, а методи ML дозволяють ретельно аналізувати та інтерпретувати їх для майбутніх прогнозів на основі статистичної динаміки.

Демографічне та економічне прогнозування – у демографії та економіці існує безліч статистичних даних, які найбільш ефективно використовуються для прогнозування часових рядів на основі ML. В результаті можна вибрати найбільш підходящу цільову аудиторію та розробити найефективніші способи взаємодії з нею.

Прогнозування наукових досліджень – ML і принципи глибокого навчання різко прискорюють темпи відшліфування та впровадження наукових інновацій. Наприклад, наукові дані, які потребують необмеженої кількості аналітичних ітерацій, можна обробляти набагато швидше за допомогою шаблонів, автоматизованих за допомогою машинного навчання.

1.3 Методи машинного навчання для прогнозування часових рядів

Як саме працює прогнозування часових рядів машинним навчанням на практиці? За ці роки було впроваджено багато методів. Деякі з них можуть навіть вважатися застарілими. Так звані застарілі методи прогнозування

часових рядів або потребують більше часу та зусиль для впровадження, або дають порівняно недостатні результати (або обидва) на відміну від нещодавно введених альтернатив. Однак вони все одно можуть відповідати конкретним цілям краще, ніж інші підходи.

Крім того, існують класичні методи, які є добре випробуваними підходами, які залишаються за замовчуванням для більшості випадків прогнозування часових рядів і найбільш широко використовуються. Чудово те, що їх можна ефективно підкріпити можливостями ML для досягнення набагато кращих результатів. З іншого боку, актуальні методи – це методи, орієнтовані на конкретні ситуації та цілі, які відповідають конкретним сценаріям прогнозування.

Як і в будь-якому випадку роботи з машинним навчанням, ML-прогнозування може бути контрольованим (що вимагає введення конкретних даних для роботи) або неконтрольованим (постійні механізми обробки даних, що самонавчаються). Перераховані нижче методи можуть бути взаємозамінними.

1.3.1 Застарілі методи прогнозування часових рядів

Рекурентна нейронна мережа (RNN) – обробляє часовий ряд крок за кроком, підтримуючи внутрішній стан від кроку часу до кроку часу. Нейронні мережі чудово підходять для такого застосування, оскільки вони можуть вивчати часову залежність на основі даних. А розгляд вхідних послідовностей з точки зору часу відкриває горизонти для більш точних прогнозів. Однак метод вважається застарілим, оскільки «навчання» нейронних мереж може зайняти занадто багато часу.

Довга короткочасна пам’ять (LSTM) – це свого роду RNN, але, зберігаючи здатність RNN вивчати часову динаміку послідовних даних, LSTM, крім того, може впоратися з проблемою градієнтів, що зникають і вибухають. Таким чином, складні багатовимірні послідовності даних можуть бути точно змодельовані, а також необхідність встановлення заздалегідь визначених часових вікон (що вирішує багато завдань, які не можуть вирішити мережі з прямим зв’язком). Однак залишається недолік занадто трудомісткого

контрольованого навчання.

1.3.2 Класичні методи прогнозування часових рядів

Багатошаровий персептрон (MLP) – одновимірні моделі можна використовувати для моделювання задач прогнозування одновимірних часових рядів. Багатовимірні моделі MLP використовують багатовимірні дані, де є більше одного спостереження для кожного кроку часу. Крім того, існують багатокрокові моделі MLP – існує невелика різниця з моделлю MLP у прогнозуванні векторного виводу, який представляє різні вихідні змінні, або векторного виходу, який представляє кілька кроків часу однієї змінної. Це дуже широко використовуваний метод, який навіть перевершує LSTM в певних випадках авторегресії.

ARIMA – авторегресія використовує спостереження, зібрані на попередніх кроках часу, як вхідні дані для складання рівнянь регресії, які допомагають передбачити значення, яке буде створено на наступному кроці часу. ARIMA або авторегресивна інтегрована модель ковзного середнього поєднує принципи авторегресії та ковзного середнього, завдяки чому прогнози відповідають лінійним комбінаціям минулих значень змінних і помилок прогнозу, що є одним із найпопулярніших підходів завдяки цьому.

Байєсова нейронна мережа (BNN) – моделі BNN передбачають побудову попереднього розподілу та оновлення цього розподілу шляхом обумовлення фактичними даними. Це особливо корисно для фінансових даних через їх мінливу природу, оскільки ввімкнене нелінійне прогнозування часових рядів за допомогою машинного навчання. BNN розглядає ваги або параметри мережі як випадкові величини, будучи однією з найбільш широко використовуваних моделей.

1.3.3 Актуальні методи прогнозування часових рядів

Згорткова нейронна мережа (CNN) – хоча аналіз наборів даних зображень вважається їх основною сферою застосування, згорткові нейронні мережі можуть показати навіть кращі результати, ніж RNN, у випадках передбачення

часових рядів, що включають інші типи просторових даних. З одного боку, вони швидше навчаються, підвищуючи загальну продуктивність обробки даних. Однак CNN також можна об’єднати з RNN, щоб отримати найкраще з обох варіантів – тобто CNN легко розпізнає просторові дані та передає їх RNN для тимчасового зберігання даних.

Механізм уваги (Attention Mechanism) – це один з основних принципів глибокого навчання, який можна адаптувати з точки зору різних моделей прогнозування. Коротко, він імітує людський мозок з точки зору фокусування уваги на конкретних елементах, які виділяються з купи. Це дозволяє глибокій нейронній мережі зосередитися лише на відповідних точках даних серед безлічі різних вхідних даних, підвищуючи ефективність NLP та комп’ютерного зору.

Трансформаторні нейронні мережі – це передова архітектура, орієнтована на вирішення завдань від послідовності до послідовності. Його головна мета також полягає в тому, щоб легко обробляти залежності на великій відстані. Такі мережі досить популярні в моделях на основі ML, спрощуючи регресію, просто налаштовуючи функцію втрат. Це більш ніж зручно, коли справа доходить до регресії.

1.4 Поетапний процес прогнозування часових рядів за допомогою машинного навчання

Після всього вищенаведеного, давайте розглянемо деякі поширені практики прогнозування за допомогою машинного навчання. Покроковий розгляд тут має вирішальне значення для гладкого високоякісного прогнозного часового моделювання та результативного прогнозування.

Підготовчий етап 1. Визначення мети проекту – починаємо із всебічного окреслення та розуміння другорядних та основних етапів та цілей. Саме тут слід провести попередні дослідження, щоб найбільш правильно вирішувати особливості сфери бізнесу.

Збір і дослідження даних – продовжуючи ретельну підготовку, необхідно визначити конкретні типи даних, які підлягають аналізу та обробці. Для цього можна використовувати діаграми візуалізації даних та графіки.

Підготовка даних і розкладання часових рядів – спеціалізовані фахівці

повинні очистити всі залучені дані, отримати цінну інформацію та витягти відповідні змінні. Ці змінні потім можна використовувати для розкладання часових рядів.

Етап 2 моделювання. Оцінка моделей прогнозування – на основі всіх попередніх досліджень і підготовчих даних тестуються й оцінюються різні моделі прогнозування, щоб вибрати найбільш ефективні.

Навчання моделі прогнозування та оцінка продуктивності – вибрані алгоритми машинного навчання для часових рядів потім оптимізуються за допомогою перехресної перевірки та навчаються.

Етап 3 тестування. Моделі прогнозування працюють на основі даних тестування з відомими результатами – крок, необхідний для того, щоб переконатися, що обрані алгоритми виконують свою роботу належним чином.

Оптимізація точності та продуктивності – останній етап відшліфування алгоритмів для досягнення найкращої швидкості та точності прогнозування.

Етап 4 розгортання. Трансформація та візуалізація даних – щоб інтегрувати отриману модель прогнозування з поточним виробництвом, зібрані дані необхідно зручно трансформувати та візуалізувати для подальшої обробки.

Перегляд і вдосконалення моделей прогнозування – прогнозування часових рядів завжди ітеративне, що означає, що для постійного покращення ефективності прогнозування необхідно впроваджувати численні поточні перегляди та оптимізації.

1.5 Основні проблеми прогнозування часових рядів за допомогою моделей машинного навчання

Пункт вище має дати основне уявлення про структуру та виконання проекту прогнозування часових рядів. Однак потрібно мати на увазі, що можна зіткнутися з певними поширеними проблемами використання машинного навчання для часових рядів у процесі. До них можна віднести наступне:

– відсутність даних, пов’язаних із часом – чим більше навчальних даних система може витягти з наборів даних, тим вищої точності прогнозування можна досягти. Однак під час роботи з ML може виникнути відсутність сезонності/історичних даних для цільової змінної, що обмежує здатність

системи до навчання;

– прийнятна точність для оцінки моделі – можливо, доведеться багато експериментувати, намагаючись досягти найвищої ефективності прогнозування. Ось де дуже обізнаний підхід до оцінки найточніших прогнозних моделей є обов’язковим;

– нерозуміння бізнес-процесів галузі – тільки досвідчені спеціалісти з ніші можуть займатися розробкою функцій ML. Є вірогідність просто не впоратися з прогнозуванням за допомогою алгоритмів машинного навчання без належного досвіду в області.

1.6 Висновки до першого розділу

У першому розділі даної роботи виконано постановку задачі використання машинного навчання для прогнозування у різних галузях діяльності людини в загальному вигляді.

Ключові висновки з першого розділу.

Прогнозування за допомогою машинного навчання – це справді наступний рівень передбачення на основі даних. І немає причин, чому компанія або підприємець повинні упускати можливість посилити аналітику даних безпрецедентними можливостями ML. Однак ця сфера має ряд проблем і випадкових ускладнень, з якими може впоратися лише досвідчений фахівець.

Практично будь-яка компанія чи організація, які мають справу з постійно генерованими даними та потребою адаптуватися до операційних зрушень і змін, можуть використовувати прогнозування часових рядів. Машинне навчання слугує найкращим прискорювачом, дозволяючи краще обробляти дані з багатьох галузей.

Покроковий процес прогнозування часових рядів за допомогою машинного навчання включає в себе 4 етапи: підготовчий; моделювання; тестування та розгортання.

2 АНАЛІЗ ВЕБ-СЕРВІСІВ МАШИННОГО НАВЧАННЯ AMAZON ДЛЯ ПРОГНОЗУВАННЯ

2.1 Amazon SageMaker

Amazon SageMaker – це повністю керована служба машинного навчання. За допомогою SageMaker науковці та розробники даних можуть швидко й легко створювати й навчати моделі машинного навчання, а потім безпосередньо розгортати їх у готовому для виробництва середовищі розміщення. Сервіс надає інтегрований екземпляр блокнота для розробки Jupyter для легкого доступу до джерел даних для дослідження та аналізу, тому користувачу не потрібно керувати серверами. Він також надає загальні алгоритми машинного навчання, які оптимізовані для ефективної роботи з надзвичайно великими даними в розподіленому середовищі. Завдяки вбудованій підтримці алгоритмів і фреймворків користувача, SageMaker пропонує гнучкі розподілені варіанти навчання, які адаптуються до конкретних робочих процесів. Розгортання моделі у безпечному та масштабованому середовищі відбувається кількома клацаннями з SageMaker Studio або консолі SageMaker [3].

Amazon SageMaker включає наведені нижче функції.

SageMaker Studio – інтегроване середовище машинного навчання, де можна створювати, навчати, розгортати й аналізувати моделі в одній програмі.

SageMaker Canvas – служба автоматичного ML, яка дає людям, які не мають досвіду програмування, можливість будувати моделі та робити прогнози з ними.

SageMaker Ground Truth Plus – функція маркування даних «під ключ» для створення високоякісних навчальних наборів даних без необхідності створювати програми маркування та самостійно керувати персоналом маркування.

SageMaker Studio Lab – послуга, яка надає клієнтам доступ до обчислювальних ресурсів AWS у середовищі на основі відкритого коду JupyterLab.

SageMaker Training Compiler – швидше навчання моделі глибокого

навчання на масштабованих екземплярах GPU, якими керує SageMaker.

SageMaker Studio Universal Notebook – дозволяє легко знаходити, підключатися, створювати, припиняти та керувати кластерами Amazon EMR в конфігураціях одного облікового запису та кількох облікових записів безпосередньо з SageMaker Studio.

SageMaker Serverless Endpoints – опція кінцевої точки без сервера для розміщення моделі ML. Автоматично масштабує ємність для обслуговування трафіку кінцевої точки. Усуває необхідність вибору типів екземплярів або керування політиками масштабування на кінцевій точці.

SageMaker Inference Recommender – дозволяє отримати рекомендації щодо типів і конфігурацій екземплярів висновку (наприклад, кількість екземплярів, параметри контейнера та оптимізацію моделі), щоб використовувати власні моделі машинного навчання та робочі навантаження.

SageMaker Model Registry – історія версій, відстеження артефактів і походження, робочий процес затвердження та підтримка між обліковими записами для розгортання власних моделей машинного навчання.

SageMaker Projects – дозволяє створювати наскрізні рішення ML із CI/CD за допомогою проектів SageMaker.

SageMaker Model Building Pipelines – створення та керування конвеєрами машинного навчання, інтегрованими безпосередньо із завданнями SageMaker.

SageMaker ML Lineage Tracking – дозволяє відстежувати походження робочих процесів машинного навчання.

SageMaker Data Wrangler – імпорт, аналіз, підготовка та показ даних в SageMaker Studio. Можна інтегрувати Data Wrangler у власні робочі процеси машинного навчання, щоб спростити та впорядкувати попередню обробку даних та розробку функцій, майже не використовуючи кодування або взагалі не застосовуючи його. Також можна додати власні сценарії та перетворення Python, щоб налаштувати робочий процес підготовки даних.

SageMaker Feature Store – централізоване сховище функцій і пов’язаних метаданих, щоб функції можна було легко виявити та використовувати повторно. Можна створити два типи Store: Online або Offline. Online Store можна використовувати для низьких затримок у випадках використання результатів у реальному часі, а Offline Store можна використовувати для

навчання та пакетного висновку.

SageMaker JumpStart – дозволяє дізнатися про функції та можливості SageMaker за допомогою підібраних рішень одним клацанням, прикладів блокнотів і попередньо підготовлених моделей, які можна розгорнути. Також можна точно налаштувати моделі та розгорнути їх.

SageMaker Clarify – удосконалювання власних моделей машинного навчання, виявляючи потенційну упередженість і допомагаючи пояснити прогнози, які роблять моделі.

SageMaker Edge Manager – дозволяє оптимізувати власні моделі для периферійних пристроїв, створювати парки та керувати ними та запускати моделі з ефективним часом виконання.

SageMaker Ground Truth – дозволяє налаштовувати та керувати завданнями маркування для високоточних наборів навчальних даних за допомогою активного навчання та експертів.

Amazon Augmented AI – дозволяє створити робочі процеси, необхідні для перевірки прогнозів ML. Amazon A2I надає розробникам доступ до людського огляду, усуваючи недиференційовану важку роботу, пов’язану зі створенням систем огляду людиною або керуванням великою кількістю людей-рецензентів.

SageMaker Studio Notebooks – блокноти нового покоління SageMaker, які включають інтеграцію AWS Single Sign-On (AWS SSO), швидкий час запуску та спільний доступ одним клацанням.

SageMaker Experiments – керування та відстеження експериментів. Можна використовувати дані відстеження, щоб перебудувати експеримент, поступово розбудовувати експерименти і відстежувати походження моделі для відповідності та перевірок.

SageMaker Debugger – дозволяє перевіряти параметри навчання та дані протягом усього навчального процесу. Автоматично виявляти й сповіщати користувачів про поширені помилки, наприклад, занадто великі чи малі значення параметрів.

SageMaker Autopilot – користувачі без знань машинного навчання можуть швидко створювати моделі класифікації та регресії.

SageMaker Model Monitor –відстеження й аналіз моделі у виробництві (кінцеві точки), щоб виявити відхилення даних та відхилення в якості моделі.

SageMaker Neo – дозволяє навчати моделі машинного навчання один раз, а потім запускайте будь-де в хмарі та локальних рівнях.

SageMaker Elastic Inference – збільшує пропускну здатність і зменшує затримку отримання висновків у реальному часі.

Reinforcement Learning – дозволяє максимізувати довгострокову винагороду, яку отримує агент в результаті своїх дій.

Preprocessing – аналіз та попередня обробка даних, розробка функцій та оцінка моделі.

Batch Transform – дозволяє попередньо обробляти набори даних, виконувати висновки, коли не потрібна постійна кінцева точка, і пов’язувати вхідні записи з висновками, щоб допомогти інтерпретувати результати.

2.1.1 Машинне навчання з Amazon SageMaker

У цьому підпункті описується типовий робочий процес машинного навчання та підсумовується, як виконуються ці завдання за допомогою Amazon SageMaker.

У машинному навчанні ми «навчаємо» комп’ютер робити прогнози або висновки. Спочатку використовується алгоритм і приклади даних для навчання моделі. Потім інтегрується власна модель у програму, щоб генерувати результати в реальному часі та в масштабі. У виробничому середовищі модель зазвичай навчається на мільйонах прикладів даних і робить висновки за сотні або менше мілісекунд.

На рис. 2.1 проілюстровано типовий робочий процес для створення моделі машинного навчання.

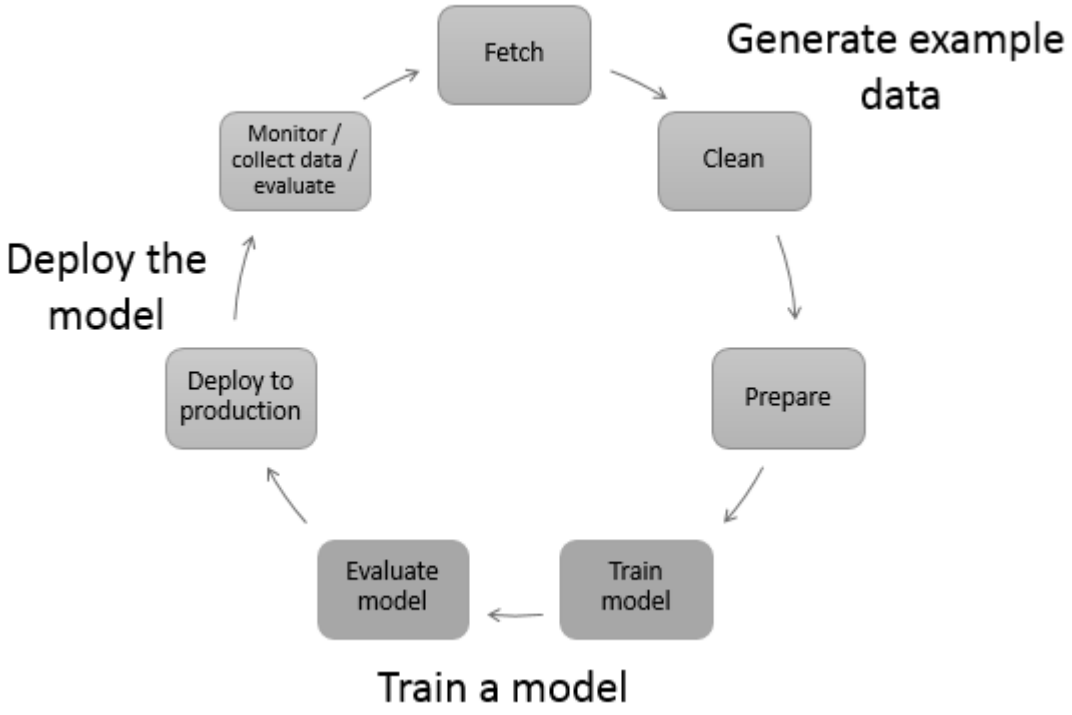

Рисунок 2.1 – Типовий робочий процес моделі машинного навчання

Зазвичай виконуються такі дії.

Крок 1. Створення прикладів даних – щоб навчити модель, потрібні приклади даних. Тип даних, які потрібні, залежить від бізнес-задачі, яку планується, щоб модель вирішила (висновки, які планується створити моделлю). Наприклад, припустимо, що планується створення моделі для передбачення числа, надавши вхідне зображення рукописної цифри. Щоб навчити таку модель, потрібні приклади зображень рукописних чисел.

Дослідники даних часто витрачають багато часу на вивчення та попередню обробку або «переборку» прикладних даних, перш ніж використовувати їх для навчання моделі. Щоб попередньо обробити дані, зазвичай робиться наступне:

- отримати дані – це можуть бути власні сховища прикладів даних, або можна використовувати загальнодоступні набори даних. Як правило, збираються набір даних або набори даних в одне сховище;

- очистити дані – щоб покращити навчання моделі, потрібно перевірити дані та очистити їх за потреби. Наприклад, якщо ваші дані мають атрибут назви країни зі значеннями Сполучені Штати та США, можна відредагувати дані, щоб

вони були узгоджені;

- підготувати або трансформувати дані – щоб підвищити продуктивність, можна виконати додаткові перетворення даних. Наприклад, можна об’єднати атрибути. Якщо модель передбачає умови, що вимагають проти обледеніння літака, замість того, щоб використовувати атрибути температури та вологості окремо, ви можете об’єднати ці атрибути в новий атрибут, щоб отримати кращу модель. У SageMaker попередньо обробляються приклади даних у блокноті Jupyter у примірнику блокнота користувача. Використовується власний блокнот, щоб отримати набір даних, досліджувати його та підготувати до навчання моделі.

Крок 2. Навчання моделі – навчання моделі включає як навчання, так і оцінку моделі.

Навчання моделі. Щоб навчити модель, потрібен алгоритм. Вибраний алгоритм залежить від ряду факторів. Щоб отримати швидке, готове рішення, можна використовувати один з алгоритмів, які надає SageMaker.

Також потрібні обчислювальні ресурси для навчання. Залежно від розміру навчального набору даних і того, наскільки швидко потрібні результати, можна використовувати ресурси від одного екземпляра загального призначення до розподіленого кластера екземплярів GPU.

Оцінка моделі. Після того, як навчання моделі, проводиться її оцінка, щоб визначити, чи прийнятна точність висновків. У SageMaker використовується або AWS SDK для Python (Boto), або бібліотека Python високого рівня, яку надає SageMaker, щоб надсилати запити до моделі для висновків.

Використовується блокнот Jupyter у власному екземплярі блокнота SageMaker для навчання та оцінки моделі.

Крок 3. Розгортання моделі — традиційно модель перепроєктовується перед інтеграцією в програму та розгортанням. За допомогою послуг хостингу SageMaker користувач може самостійно розгорнути свою модель, відокремивши її від коду програми.

Машинне навчання – це безперервний цикл. Після розгортання моделі відстежуються висновки, збирається «основна істину» та оцінюється модель, щоб визначити відхилення. Потім підвищується точність власних висновків, оновлюючи навчальні дані, щоб включати нещодавно зібрану основну істину.

Робиться це шляхом перенавчання моделі з новим набором даних. У міру того, як стає доступним все більше і більше прикладних даних, продовжується перенавчання власної моделі, щоб підвищити точність.

2.1.2 Дослідження, аналіз й обробка даних

Перш ніж використовувати набір даних для навчання моделі, спеціалісти з даних зазвичай досліджують, аналізують та попередньо обробляють його.

Amazon SageMaker Processing дає змогу виконувати завдання для попередньої та постобробки даних, розробки функцій та оцінки моделей на SageMaker легко та масштабовано. У поєднанні з іншими важливими завданнями машинного навчання, що надаються SageMaker, такими як навчання та розміщення, Processing надає переваги повністю керованого середовища машинного навчання, включаючи всю підтримку безпеки та відповідності, вбудовану в SageMaker. За допомогою Processing можна використовувати вбудовані контейнери обробки даних або створювати власні контейнери та надсилати спеціальні завдання для виконання в керованій інфраструктурі. Після того, як подається завдання, SageMaker запускає обчислювальні екземпляри, обробляє та аналізує вхідні дані, а після завершення звільняє ресурси.

2.1.3 Навчання моделі за допомогою Amazon SageMaker

На рис. 2.2 показано, як відбувається навчання та розгортання моделі за допомогою Amazon SageMaker.

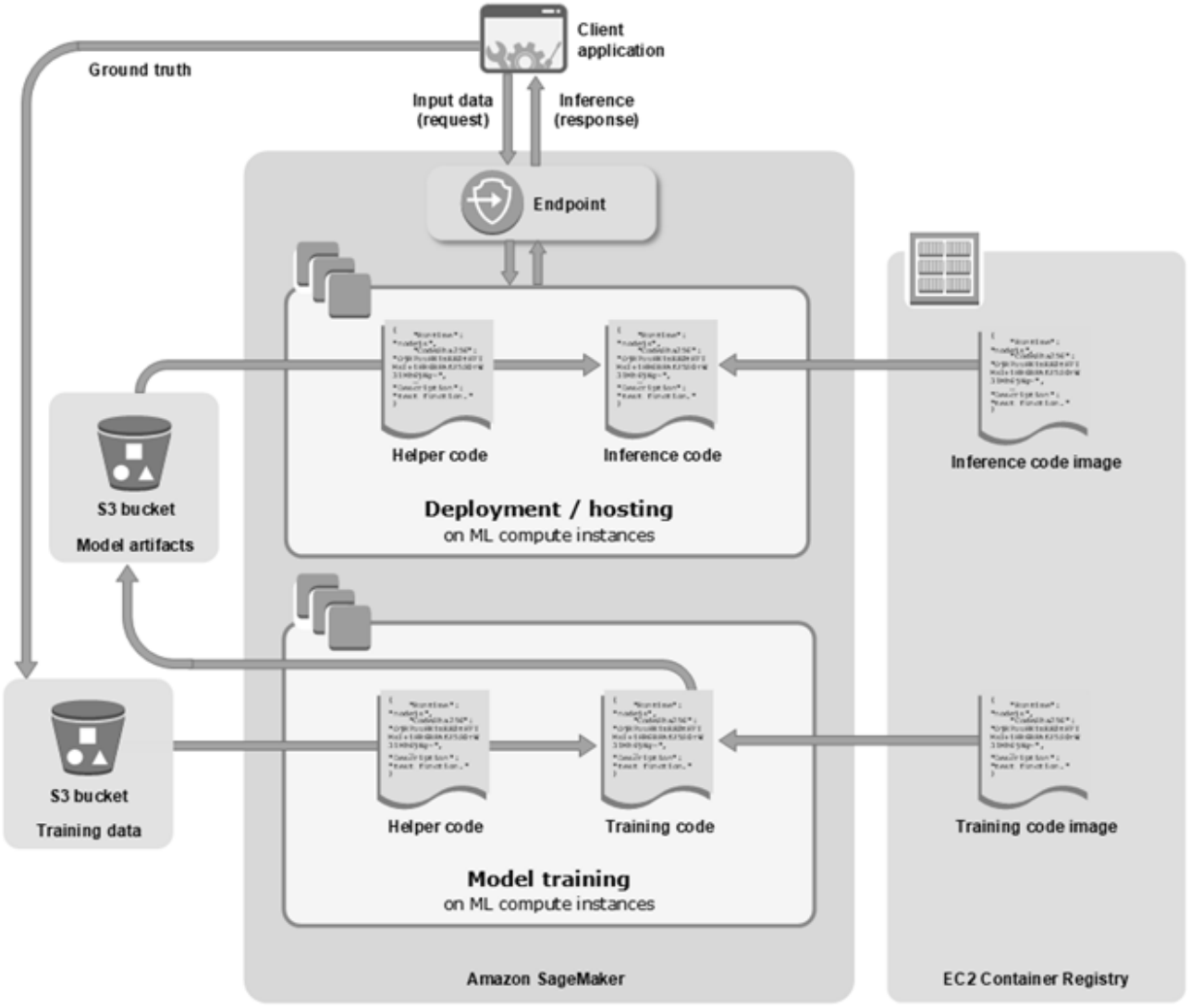

Рисунок 2.2 – Навчання і розгортання моделі в Amazon SageMaker

Область з позначкою SageMaker виділяє два компоненти SageMaker: навчання моделі та розгортання моделі.

Щоб навчити модель у SageMaker, створюється навчальне завдання. Навчальне завдання включає наступну інформацію:

– URL-адреса сегмента Amazon Simple Storage Service (Amazon S3), де зберігалися дані навчання;

– обчислювальні ресурси, які будуть використовуватися SageMaker для навчання моделі. Обчислювальні ресурси — це екземпляри обчислень ML, якими керує SageMaker;

– URL-адреса сегмента S3, де будуть збережені результати завдання;

– шлях до реєстру Amazon Elastic Container, де зберігається навчальний код.

Є наступні варіанти алгоритму навчання, це використання:

–алгоритму, наданого SageMaker – SageMaker надає алгоритми навчання. Якщо один з них відповідає потребам користувача, це чудове готове рішення для швидкого навчання моделі;

–SageMaker Debugger – для перевірки параметрів навчання та даних протягом усього навчального процесу під час роботи з платформами навчання TensorFlow, PyTorch і Apache MXNet або алгоритмом XGBoost. Debugger автоматично виявляє та попереджає користувачів про типові помилки, такі як значення параметрів, які стають занадто великими або малими;

– Apache Spark з SageMaker – SageMaker надає бібліотеку, яку можна використовувати в Apache Spark для навчання моделей за допомогою SageMaker. Використання бібліотеки, наданої SageMaker, подібне до використання Apache Spark MLLib;

– власного коду для навчання за допомогою фреймворків глибокого навчання. Можна надіслати власний код Python, який використовує TensorFlow, PyTorch або Apache MXNet для навчання моделі;

– власних алгоритмів користувача – об’єднати власний код в образ Docker і вказати шлях реєстру до образу у виклику API SageMaker CreateTrainingJob;

– алгоритму зі списку підписок на AWS Marketplace.

Після створення навчального завдання SageMaker запускає обчислювальні екземпляри ML і використовує навчальний код і навчальний набір даних для навчання моделі. Він зберігає отримані артефакти моделі та інші вихідні дані у хмарі Amazon S3, яка була вказана для цієї мети.

Користувач може створити навчальне завдання за допомогою консолі SageMaker або API.

Коли користувач створює навчальне завдання за допомогою API, SageMaker за замовчуванням реплікує весь набір даних на екземплярах обчислень ML. Щоб змусити SageMaker реплікувати підмножину даних на кожному обчислювальному екземплярі ML, потрібно встановити для поля S3DataDistributionType значення ShardedByS3Key. Можна встановити це поле за допомогою SDK низького рівня.

2.1.4 Розгортання моделі в Amazon SageMaker

Після того, як власна модель машинного навчання буде навчена, можна розгорнути її за допомогою Amazon SageMaker, щоб отримати прогнози будь-яким із наведених нижче способів, залежно від випадку використання:

– для постійних кінцевих точок у реальному часі, які роблять прогнозування по одному, використовуються послуги хостингу SageMaker в режимі реального часу;

– робочі навантаження, які мають періоди простою між стрибками трафіку та можуть переносити холодні запуски, використовують безсерверний висновок;

– запити з великими розмірами корисного навантаження до 1 Гб, тривалим часом обробки та вимогами до затримки майже в реальному часі використовують асинхронний висновок Amazon SageMaker;

– щоб отримати прогнози для всього набору даних, використовують пакетне перетворення SageMaker.

SageMaker також надає функції для керування ресурсами та оптимізації продуктивності висновку під час розгортання моделей машинного навчання щоб:

– керувати моделями на периферійних пристроях, щоб можна було оптимізувати, захищати, відстежувати й підтримувати моделі машинного навчання на таких периферійних пристроях, як розумні камери, роботи, персональні комп’ютери та мобільні пристрої, використовують SageMaker Edge Manager;

– оптимізувати моделі Gluon, Keras, MXNet, PyTorch, TensorFlow, TensorFlow-Lite та ONNX для висновків на машинах Android, Linux та Windows на основі процесорів Ambarella, ARM, Intel, Nvidia, NXP, Qualcomm, Texas Instruments та Xilinx, використовують функцію Neo.

2.1.5 Використання фреймворків машинного навчання, Python і R з Amazon SageMaker

Користувач може використовувати мови Python і R вбудовані в ядрах

блокнотів Amazon SageMaker. Існують також ядра, які підтримують певні фреймворки. Дуже популярним способом розпочати роботу з SageMaker є використання Amazon SageMaker Python SDK. Він надає API та контейнери Python з відкритим вихідним кодом, які полегшують навчання та розгортання моделей у SageMaker, а також приклади для використання з кількома різними фреймворками машинного навчання та глибокого навчання.

Amazon SageMaker Python SDK – це бібліотека з відкритим вихідним кодом для навчання та розгортання моделей машинного навчання на Amazon SageMaker.

За допомогою SDK користувач може навчати та розгортати моделі, використовуючи популярні фреймворки глибокого навчання, алгоритми, надані Amazon, або власні алгоритми, вбудовані в сумісні з SageMaker образи Docker.

Amazon SageMaker підтримує RStudio як повністю кероване інтегроване середовище розробки (IDE), інтегроване з доменом Amazon SageMaker. Завдяки інтеграції RStudio користувач може запустити середовище RStudio в домені, щоб запускати власні робочі процеси RStudio на ресурсах SageMaker.

Екземпляри блокнотів SageMaker підтримують мову R за допомогою попередньо встановленого ядра R. Крім того, ядро R має сітчасту бібліотеку, інтерфейс R до Python, тому можна використовувати функції SageMaker Python SDK зі сценарію R.

2.2 Amazon Forecast

Amazon Forecast – це повністю керований сервіс, який використовує статистичні алгоритми та алгоритми машинного навчання для надання високоточних прогнозів часових рядів (рис. 2.3). На основі тієї ж технології, що використовується для прогнозування часових рядів на Amazon.com, Forecast надає найсучасніші алгоритми для прогнозування майбутніх даних часових рядів на основі історичних даних і не вимагає досвіду машинного навчання [4].

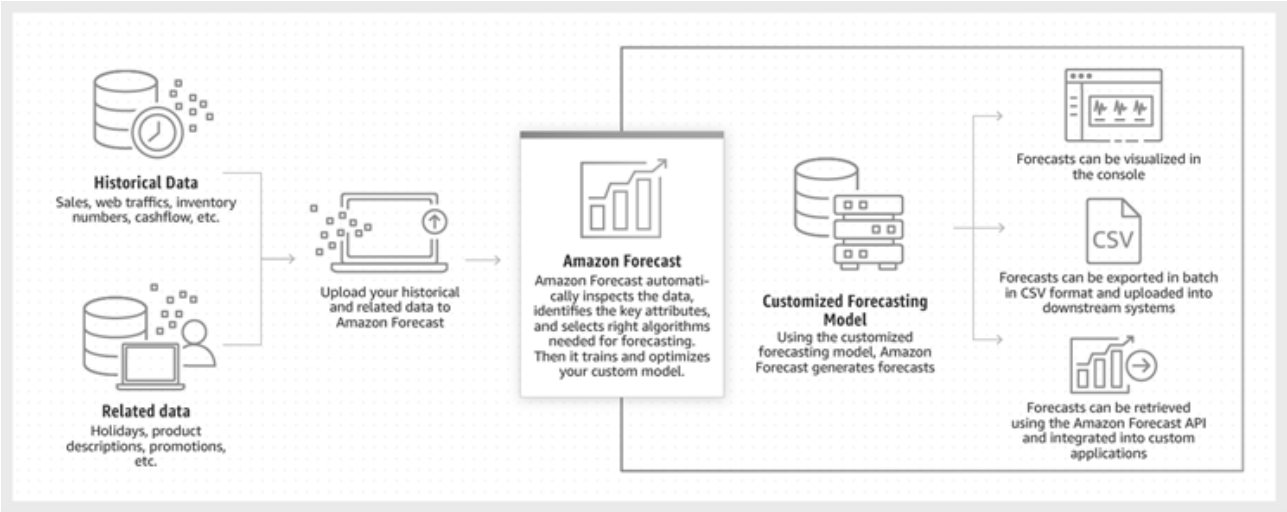

Рисунок 2.3 – Схема роботи Amazon Forecast

Прогнозування часових рядів корисно в багатьох сферах, включаючи роздрібну торгівлю, фінанси, логістику та охорону здоров'я. Також можна використовувати Forecast, щоб передбачити специфічні для галузі показники для інвентарю, робочої сили, веб-трафіку, потужності сервера та фінансів.

2.2.1 Як користуватися Amazon Forecast

Користувач може використовувати API, інтерфейс командного рядка AWS (AWS CLI), Python Software Development Kit (SDK) і консоль Amazon Forecast, щоб імпортувати набори даних часових рядів, навчати прогнозів і генерувати прогнози.

Ось кілька поширених випадків використання Amazon Forecast, це планування:

– роздрібного попиту – прогнозування попиту на продукцію дозволяє точніше варіювати запаси та ціни в різних магазинах;

–ланцюга поставок – прогнозування кількості необроблених товарів, послуг або інших ресурсів, необхідних для виробництва;

– ресурсів – передбачення потреб до персоналу, реклами, споживання енергії та потужності сервера;

– оперативне планування – прогнозування рівнів веб-трафіку, використання AWS та використання датчиків Інтернету речей.

2.2.2 Особливості Amazon Forecast

Amazon Forecast автоматизує більшу частину процесу прогнозування часових рядів, дозволяючи зосередитися на підготовці наборів даних та інтерпретації прогнозів.

Forecast надає такі функції:

– автоматичне машинне навчання – Forecast автоматизує складні завдання машинного навчання, знаходячи оптимальну комбінацію алгоритмів машинного навчання для наборів даних;

– найсучасніші алгоритми – користувач може застосовувати комбінацію алгоритмів машинного навчання, які базуються на тій же технології, що й на Amazon.com. Forecast пропонує широкий спектр алгоритмів навчання, від широко використовуваних статистичних методів до складних нейронних мереж;

– підтримка відсутніх значень – Forecast надає кілька методів заповнення для автоматичної обробки відсутніх значень у наборах даних;

– додаткові вбудовані набори даних – Forecast може автоматично включати вбудовані набори даних, щоб покращити модель. Ці набори даних уже розроблені і не потребують додаткової конфігурації.

2.2.3 Як працює Amazon Forecast

Створюючи проекти прогнозування в Amazon Forecast, користувач працює з такими ресурсами:

– імпорт наборів даних. Набори даних – це колекції ваших вхідних даних. Групи наборів даних – це колекції наборів даних, які містять додаткову інформацію. Алгоритми прогнозування використовують групи наборів даних для навчання користувацьких моделей прогнозування, які називаються предикторами;

– навчання предикторів. Предиктори – це спеціальні моделі, навчені на даних користувача. Можна навчити предиктор, вибравши попередньо створений алгоритм або вибравши параметр AutoML, щоб Amazon Forecast вибрав найкращий алгоритм для конкретного випадку;

– створення прогнозів. Користувач може створювати прогнози для даних часових рядів, запитувати їх за допомогою QueryForecast API або візуалізувати їх на консолі.

Набори даних містять дані, які використовуються для підготовки предиктора. Користувач створює один або кілька наборів даних Amazon Forecast та імпортує в них власні навчальні дані. Група наборів даних — це набір додаткових наборів даних, які детально описують набір параметрів, що змінюються протягом серії часу. Після створення групи наборів даних вона використовується для навчання предиктора.

Кожна група наборів даних може мати до трьох наборів даних, по одному для кожного типу набору даних: цільовий часовий ряд, пов’язаний часовий ряд і метадані елемента.

Для створення наборів даних і груп наборів прогнозів і керування ними можна використовувати консоль прогнозу, інтерфейс командного рядка AWS (AWS CLI) або AWS SDK.

Предиктор – це модель Amazon Forecast, яка навчається з використанням цільових часових рядів, пов’язаних часових рядів, метаданих елементів та будь-яких додаткових наборів даних, які обирає користувач. Можна використовувати предиктори для створення прогнозів на основі даних часових рядів.

За замовчуванням Amazon Forecast створює AutoPredictor, де Forecast застосовує оптимальну комбінацію алгоритмів до кожного часового ряду у наборах даних.

Після створення предиктора Amazon Forecast готовий створити прогноз. Прогноз включає передбачення для кожного елемента (item_id) у групі наборів даних, який використовувався для навчання предиктора. Щоб створити прогноз для підмножини елементів, треба імпортувати лише ті елементи до тих самих наборів даних, які використовуються для навчання предиктора, а потім створити власний прогноз.

Після створення прогнозу користувач може експортувати його до власного сегмента Amazon Simple Storage Service (Amazon S3).

2.3 Висновки до другого розділу

У другому розділі даної роботи виконано аналіз сервісів машинного навчання Amazon для прогнозування.

Ключові висновки з другого розділу.

Amazon SageMaker – це повністю керована служба машинного навчання. За допомогою SageMaker науковці та розробники даних можуть швидко й легко створювати й навчати моделі машинного навчання, а потім безпосередньо розгортати їх у готовому для виробництва середовищі розміщення. Сервіс надає інтегрований екземпляр блокнота для розробки Jupyter для легкого доступу до джерел даних для дослідження та аналізу, тому користувачу не потрібно керувати серверами. Він також надає загальні алгоритми машинного навчання, які оптимізовані для ефективної роботи з надзвичайно великими даними в розподіленому середовищі.

У машинному навчанні комп’ютер навчається робити прогнози або висновки. Спочатку використовується алгоритм і приклади даних для навчання моделі. Потім інтегрується власна модель у програму, щоб генерувати результати в реальному часі та в масштабі. У виробничому середовищі модель зазвичай навчається на мільйонах прикладів даних і робить висновки за сотні або менше мілісекунд.

Amazon Forecast – це повністю керований сервіс, який використовує статистичні алгоритми та алгоритми машинного навчання для надання високоточних прогнозів часових рядів. На основі тієї ж технології, що використовується для прогнозування часових рядів на Amazon.com, Forecast надає найсучасніші алгоритми для прогнозування майбутніх даних часових рядів на основі історичних даних і не вимагає досвіду машинного навчання.

Користувач може використовувати API, інтерфейс командного рядка AWS (AWS CLI), Python Software Development Kit (SDK) і консоль Amazon Forecast, щоб імпортувати набори даних часових рядів, навчати прогнозів і генерувати прогнози.

Amazon Forecast автоматизує більшу частину процесу прогнозування часових рядів, дозволяючи зосередитися на підготовці наборів даних та інтерпретації прогнозів.

3 МОДЕЛЬ ПРОГНОЗУВАННЯ ПРОДАЖІВ

Прогнозування є важливою областю машинного навчання. Це важливо, оскільки багато можливостей для прогнозування майбутніх результатів базуються на історичних даних. Багато з цих можливостей включають часовий компонент. Незважаючи на те, що компонент часу додає більше інформації, він також ускладнює вирішення проблем часових рядів, ніж інші типи передбачень.

Можна вважати дані часових рядів, які поділяються на дві великі категорії. Перший тип є одномірні, що означає, що він має лише одну змінну. Другий тип є багатомірним, що означає, що він має більше однієї змінної. На додаток до цих двох категорій більшість наборів даних часових рядів також слідують одному з наступних шаблонів:

– тенденційний – шаблон, який показує значення, коли вони збільшуються, зменшуються або залишаються незмінними з часом;

– сезонний – повторюваний шаблон, який базується на сезони в році;

– циклічний – деякі інші форми повторюваних шаблонів;

– нерегулярний – зміни даних з часом, які здаються випадковими або не мають помітного шаблону [5].

3.1 Обробка даних часових рядів

Дані часових рядів фіксуються в хронологічній послідовності протягом певного періоду часу. Введення часу в модель машинного навчання має позитивний вплив, оскільки модель може отримувати значення через зміну точок даних з часом. Дані часових рядів, як правило, корелюються, що означає, що існує залежність між точками даних.

Оскільки у нас є проблема регресії – і оскільки регресія передбачає незалежність точок даних – потрібно розробити метод для обробки залежності даних. Метою цього методу є підвищення достовірності прогнозів.

На додаток до даних часового ряду, можна додати пов’язані дані для доповнення моделі прогнозування. Наприклад, для прогнозування роздрібних продажів ми можемо включити інформацію про продукт, що продається

(наприклад, ідентифікацію товару або ціну продажу). Ця інформація є доповненням до кількості одиниць, які продаються за період часу.

Третій тип даних – це метадані про набір даних. Наприклад, для набору даних роздрібної торгівлі ми можемо включити метадані до групових результатів, наприклад назву бренду. Іншим прикладом метаданих може бути включення жанру для музики чи відео.

Чим більше у нас даних, тим краще. Проблема, яку можна очікувати з кількома джерелами даних, – це часова позначка даних. Можуть виявитися відмінності у форматі позначки часу, а також інші проблеми, наприклад, неповні дані. У деяких випадках можна визначити відсутні дані. Наприклад, уявімо, що у нас є якісь дані, які містять і місяць, і день, але не рік. Припустимо, що дані відображаються послідовно через номери місяців у базі даних і повторюються після 12. У такому випадку ми можемо додати рік, якщо відомо, коли почалися дані. Можна зробити висновок про майбутні роки на основі порядку даних.

Значна частина даних зберігається у форматі Universal Coordinated Time (UTC), але не всі дані. Потрібно перевіряти, чи є мітка часу місцевим або універсальним часом.

Іноді мітка часу не відображає час, який має. Наприклад, припустимо, що у нас є база даних товарів, які постачалися на склад. Чи вказує мітка часу час, коли товар прибув, був оброблений або забраний? Або це вказує, коли остаточний запис було введено в систему?

Можливо, у даних взагалі немає позначки часу. При цьому можуть бути інші способи екстраполяції часового ряду, залежно від даних та галузі.

3.1.1 Відсутні значення

Поширеним явищем у реальних задачах прогнозування є відсутність значень у вихідних даних. Відсутність значень ускладнює створення прогнозу моделі. Основним прикладом у роздрібній торгівлі є відсутність запасів у прогнозуванні попиту. Якщо товар зникає на складі, обсяг продажів за день буде нульовим. Якщо прогноз створено на основі цих нульових значень продажів, прогноз буде неправильним.

Відсутні значення можуть бути позначені як відсутні з різних причин. Відсутні значення можуть виникати через відсутність транзакції або, можливо, через помилки вимірювання. Можливо, служба, яка відстежувала певні дані, працювала неправильно, або вимірювання не могло відбутися належним чином.

Відсутні дані можна обчислити кількома способами (рис. 3.1):

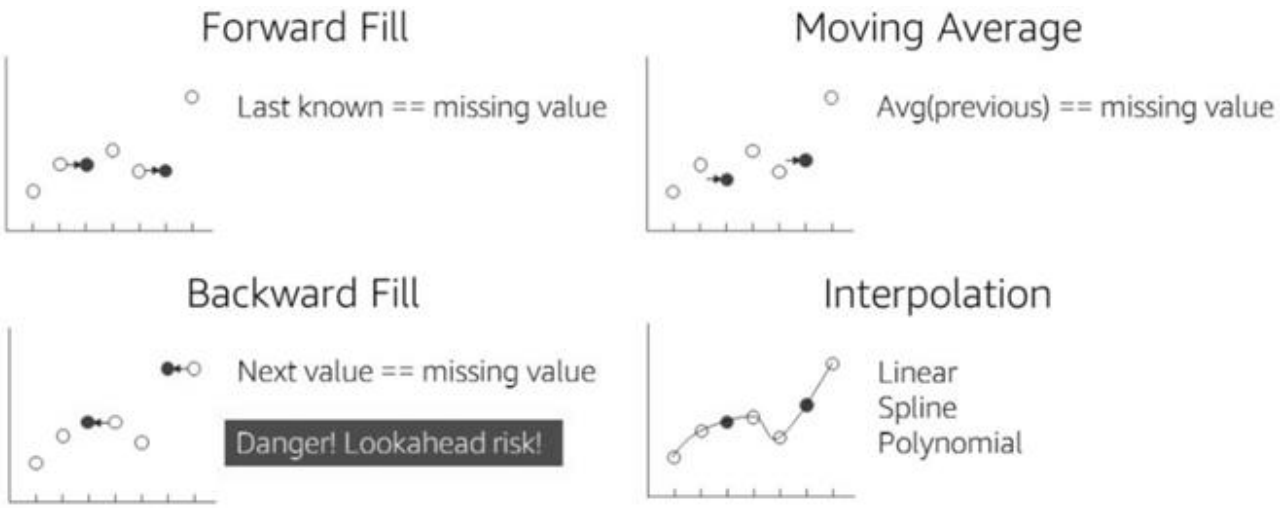

Рисунок 3.1 – Випадки обчислення відсутніх значень

– заповнення вперед – використовує останнє відоме значення для відсутнього значення;

– ковзне середнє – використовує середнє з останніх відомих значень для обчислення відсутнього значення;

– заповнення назад – використовує наступне відоме значення після пропущеного значення. Але треба мати на увазі, що використання майбутнього для обчислення минулого є потенційною небезпекою, що погано прогнозувати. Ця практика відома як перегляд уперед, і її слід уникати;

– інтерполяція – для обчислення відсутнього значення використовується рівняння.

Також можна використовувати нульове заповнення. Цей вибір часто використовується в роздрібній торгівлі, галузі, де не слід розраховувати відсутні дані про продажі. Відсутні дані не представляють жодних замовлень у цей день. Було б розумно дослідити, чому, але в цьому випадку ви не хочете заповнювати пропущене значення.

3.1.2 Зменшення і збільшення дискретизації

Ми можемо отримувати дані з різною частотою. Наприклад, можуть бути дані про продажі, які містять точну позначку часу, коли продаж був зафіксований. Однак дані інвентаризації можуть містити лише рік, місяць і день рівня інвентаризації. Якщо є дані, частота яких відрізняється від інших наборів даних, або вони несумісні з потрібним запитом, може знадобитися зменшити дискретизацію.

Зменшення дискретизації означає перехід від більш дрібнозернистого часу до менш дрібнозернистого часу. Приклад на рис. 3.2 перетворює погодинний набір даних у щоденний.

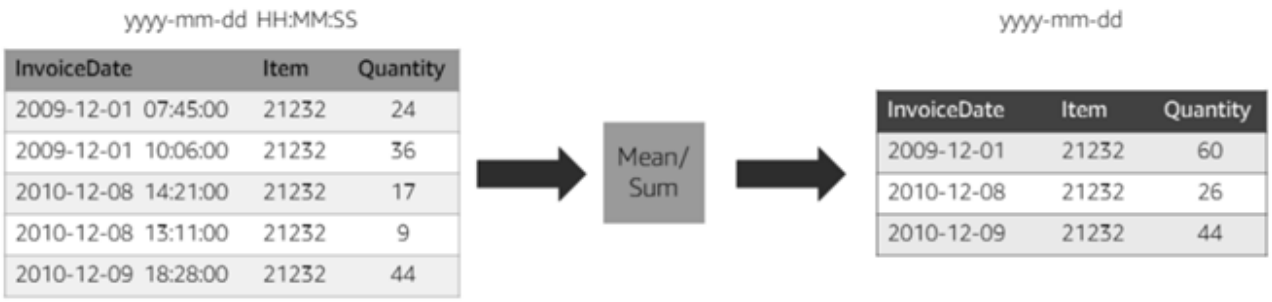

Рисунок 3.2 – Зменшення дискретизації часу

Коли ми зменшуємо дискретизацію, то повинні вирішити, як об’єднати значення. У разі даних про продажі найбільш доцільним є підсумовування кількості. Якщо дані – температура, то можна знайти середнє значення. Розуміння конкретних даних допоможе вирішити, що є найкращим способом дій.

Оберненим зниженню дискретизації є збільшення дискретизації. Проблема з збільшенням дискретизації полягає в тому, що в більшості випадків цього важко досягти. Припустимо, ми хочемо збільшити вибірку даних про продажі від щоденних продажів до погодинних. Якщо немає іншого джерела даних, на яке можна посилатися, ми не зможемо перейти від щоденних продажів до погодинних (рис. 3.3).

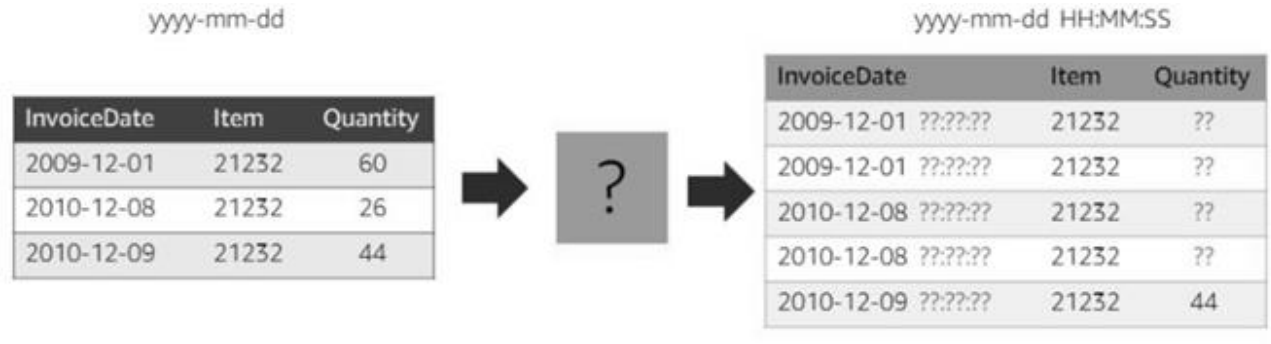

Рисунок 3.3 – Збільшення дискретизації часу

Причини для збільшення дискретизації:

– відповідність різним часовим рядам;

– неправильні числові ряди;

– знання у галузі.

У деяких випадках необхідно використовувати додаткові дані або знання. Наприклад, якщо потрібно відповідати частоті іншого часового ряду, у нас може бути неправильний часовий ряд або конкретні знання в області, які можуть допомогти. У таких випадках треба обережно конвертувати дані. Для прикладу роздрібної торгівлі найкраще, що можна зробити, це створити єдине замовлення на день в певну годину. Для температури можна скопіювати добову температуру в кожен із погодинних інтервалів або використати деяку формулу для обчислення кривої.

3.1.3 Згладжування даних

Викиди можуть бути проблемою в науці про дані. Те ж саме стосується даних часових рядів. Якщо ми перевіряємо дані про продажі і бачимо замовлення з надзвичайно великою кількістю товарів, можливо, не потрібно включати це замовлення в прогнозні розрахунки. Розмір замовлення може ніколи не повторюватися. Видалення цих викидів і аномалій відоме як згладжування (рис. 3.4).

Рисунок 3.4 – Згладжування викидів

Згладжування даних може допомогти впоратися з викидами та іншими аномаліями. Розглянемо можливість згладжування з наступних причин:

– підготовка даних – видалення значень помилок і викидів;

– візуалізація – зменшення шуму на графіку.

Проте треба розуміти, навіщо нам згладжувати дані та який вплив це може мати. Можливо, буде потрібно зменшити шум і створити кращу модель.

Як згладжування впливає на вихідну модель:

– очищення даних моделі;

– сумісність моделі;

– покращення виробництва.

Однак не менш важливо розглянути ці питання: чи може згладжування зашкодити моделі, чи очікується в моделі шум та чи можливо згладити дані у виробництві.

3.1.4 Сезонність

Сезонність у даних – це будь-який вид повторюваних спостережень, коли частота спостереження є стабільною. Наприклад, у продажах зазвичай можна бачити зростання продажів наприкінці кварталу та в четвертому кварталі (рис. 3.5). Споживча роздрібна торгівля демонструє цю закономірність ще більше в четвертому кварталі. Треба мати на увазі, що дані можуть мати кілька типів сезонності в одному наборі даних.

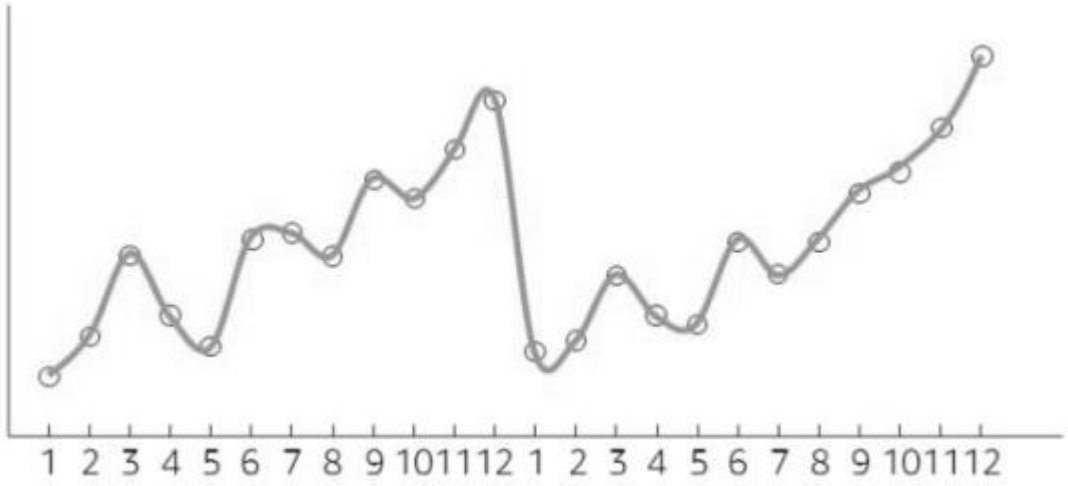

Рисунок 3.5 – Сезонність даних

Частота сезонності:

– погодинно, щоденно, квартально, щорічно;

– весною, влітку, восени, зимою;

– основні святкові розпродажі, зимовий святковий сезон.

Можна багаторазово включати інформацію про сезонність у власний прогноз. Місцеві свята є хорошим прикладом розпродажів.

3.1.5 Кореляції часових рядів

Кореляції не означають причинно-наслідкового зв’язку. Потрібно бути обережними щодо кореляцій, інтерпретуючи власні дані – ми не хочемо діяти на основі кореляцій, які не мають реального значення. В якості експерименту скажімо, що ми створюємо два випадкових набори даних часових рядів з числами від 0 до 1. Можна виявити, що вони мають низьку кореляцію. Однак, якщо ввести однаковий нахил для обох наборів даних, побачимо сильну кореляцію.

Важливо знати, наскільки стабільна система. Рівень стабільності, або стаціонарність, може сказати нам, наскільки потрібно очікувати, що минула поведінка системи буде визначати майбутню поведінку. Система з низькою стабільністю не підходить для прогнозування майбутнього.

Часто потрібно визначити тенденцію для часового ряду. Однак коригування ряду відповідно до тенденції може ускладнити його порівняння з

іншим рядом, яка також була відкоригована на тенденцію. Тенденції можуть домінувати над значеннями в ряді, що може призвести до переоцінки кореляції між двома рядами.

Автокореляція є однією з особливих проблем, з якими можна стикнутися в даних часових рядів (рис. 3.6). Як бачимо з інших проблем машинного навчання, метою побудови моделі ML є відокремлення сигналу від шуму. Автокореляція є формою шуму, оскільки окремі спостереження не є незалежними одне від одного.

Часовий ряд з автокореляцією може перебільшити точність створеної моделі. Деякі з алгоритмів, які будуть наведені нижче, можуть допомогти виправити автокореляцію.

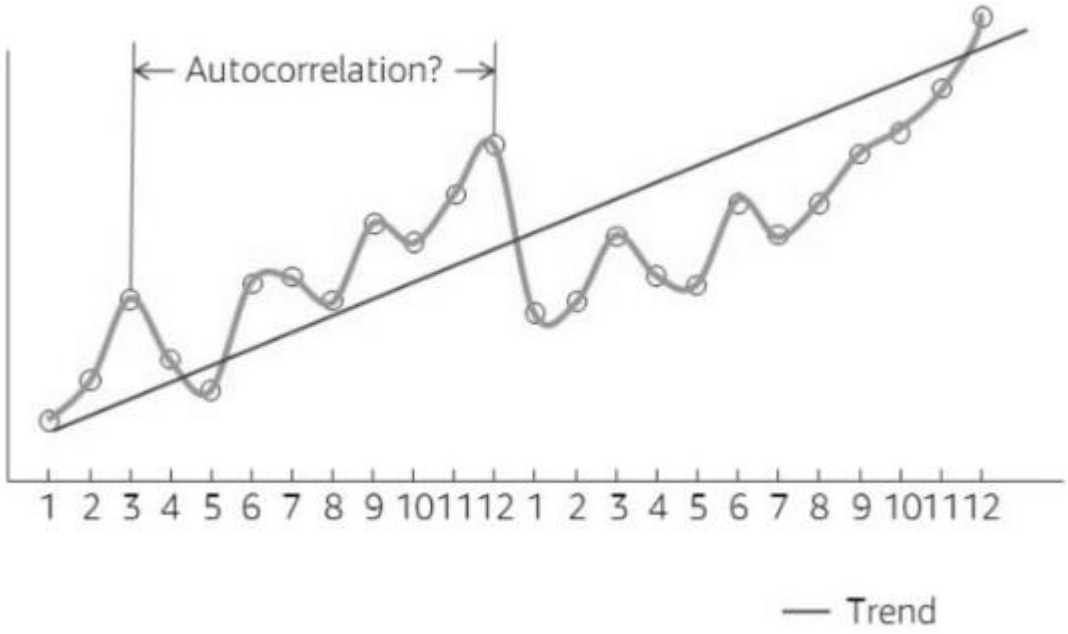

Рисунок 3.6 – Автокореляція

Ці фактори разом із сезонністю можуть вплинути на модель, яку ми виберемо для створення власного прогнозу. Деякі алгоритми обробляють сезонність і автокореляцію, а інші ні.

3.1.6 Бібліотека pandas для Python

Бібліотека pandas була розроблена з урахуванням аналізу фінансових даних. Таким чином, вона добре обробляє дані часових рядів.

Можна встановити індекс pandas DataFrame як datetime, що дає змогу використовувати дату та час для вибору даних. Можна використовувати діапазони, які містять часткові дати. Також можна витягти частини дати, такі як рік, місяць, день тижня тощо (лістинг 3.1).

```
Dataframe['2010-01-04']
dataframe['2010-02' : '2010-03']
dataframe['weekday_name'] = dataframe.index.weekday_name
```

Лістинг 3.1 – Індекс DataFrame

Для групування та повторної вибірки завдань у pandas є вбудовані функції для виконання обох (лістинг 3.2).

```
dataframe.groupby('StockCode')
dataframe.groupby('StockCode').resample('D').sum()
```

Лістинг 3.2 – Функція groupby

Нарешті, pandas може дати нам уявлення про автокореляцію (лістинг 3.3).

```
dataframe['Quantity'].autocorr()
```

Лістинг 3.3 – Функція autocorr

3.1.7 Алгоритми часових рядів

Одним із завдань при побудові програми для прогнозування є вибір відповідного алгоритму. Тип набору даних, який використовується, і особливості цього набору даних повинні визначати вибір алгоритму.

Amazon Forecast підтримує п’ять алгоритмів:

– авторегресивне інтегроване ковзне середнє (ARIMA) – цей алгоритм видаляє автокореляції, які можуть вплинути на модель спостережень;

– DeepAR+ – контрольований алгоритм навчання для прогнозування одновимірних часових рядів. Він використовує рекурентну нейронну мережу для навчання моделі для кількох часових рядів;

– експоненціальне згладжування (ETS) – цей алгоритм корисний для наборів даних із сезонністю. Він використовує середньозважене значення для всіх спостережень. Вагові коефіцієнти з часом зменшуються;

– непараметричний часовий ряд (NPTS) – передбачення ґрунтуються на вибірці з минулих спостережень. Доступні спеціалізовані версії для сезонних і кліматологічних наборів даних.

– Prophet – модель часового ряду Байєса. Вона корисна для наборів даних, які охоплюють тривалий період часу, містять відсутні дані або мають великі викиди.

3.2 Використання Amazon Forecast

Для генерації прогнозів ми можемо застосувати конвеєр розробки машинного навчання, який наведений на рис. 3.7.

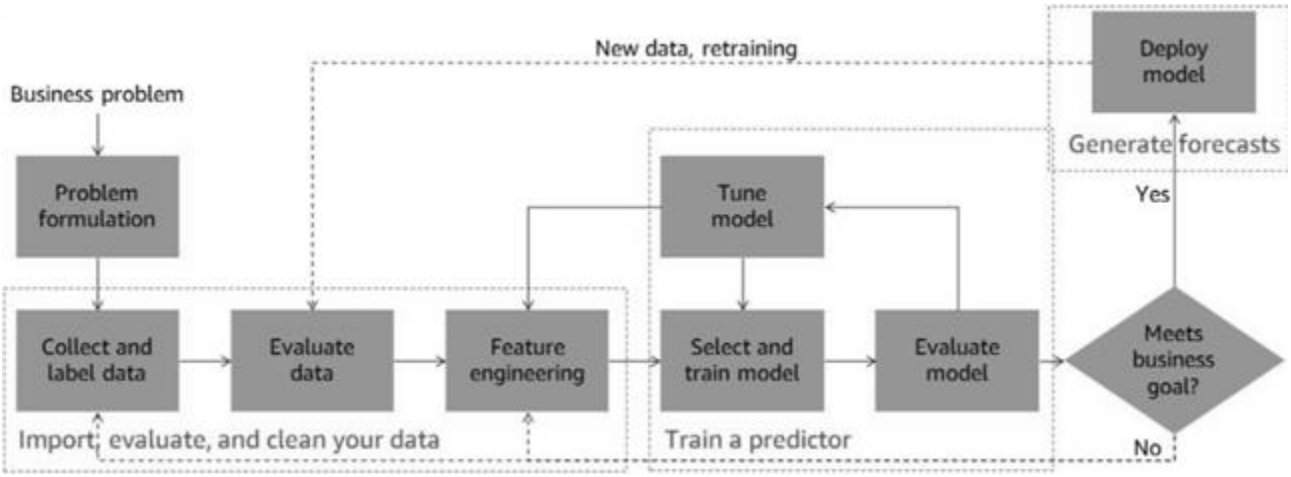

Рисунок 3.7 – Робочий процес Amazon Forecast

Імпорт даних – потрібно імпортувати стільки даних, скільки є – як історичних, так і пов’язаних даних. Перед використанням даних для навчання моделі слід виконати базову оцінку та розробку функцій.

Навчання предиктора – щоб навчити предиктор, треба вибрати алгоритм. Можна дозволити Amazon Forecast автоматично обрати алгоритм, вибравши функцію AutoML як потрібний алгоритм. Також необхідно вибрати галузь для даних, окрім типових опцій є варіант вибрати користувацькі налаштування. Галузі мають певні типи даних, які їм потрібні.

Створення прогнозів – як тільки є навчена модель, можна використовувати модель для створення прогнозу за допомогою групи вхідних наборів даних. Після створення прогнозу можна надіслати запит на прогноз або

експортувати його у хмару Amazon Simple Storage Service (Amazon S3). Також є можливість зашифрувати дані в прогнозі перед їх експортом.

Загальний процес роботи з Amazon Forecast полягає в імпорті історичних та пов'язаних даних. Amazon Forecast перевіряє дані, визначає ключові дані та вибирає відповідний алгоритм. Він використовує алгоритм для навчання та оптимізації спеціальної моделі та створення прогнозу. Ми створюємо прогнози, застосовуючи предиктор до набору даних. Потім можна отримати ці прогнози на консолі AWS або експортувати прогнози як файли, розділені комами. Також можна використовувати інтерфейс прикладного програмування (API) та команди інтерфейсу командного рядка (CLI) для створення та отримання прогнозів (рис. 3.8).

Рисунок 3.8 – Огляд Amazon Forecast

Коли ми працюємо з Amazon Forecast, слід вибрати відповідну галузь. Можна вибирати з наступного списку:

– роздрібна торгівля – попит на продукцію;

– планування запасів – вимоги до сировини;

– ємність EC2 – попит на ємність для Amazon Elastic Compute Cloud (Amazon EC2);

– робоча сила – прогнози робочого навантаження;

– веб-трафік – прогнозований трафік до одного або більше веб-сайтів;

– показники – прогнозування таких показників, як прибуток, продажі або грошові потоки;

– користувацькі – прогнози для галузі, яку не можеа зіставити з одним із

попередніх варіантів.

Вибираючи галузь, покращується ефективність предиктора. Кожна галузь має певні типи даних, які надаються під час створення предиктора. Наприклад, галузь роздрібної торгівлі очікує ідентифікатори товарів, мітку часу для спостереження, кількість продажів для цього товару та вказану мітку часу.

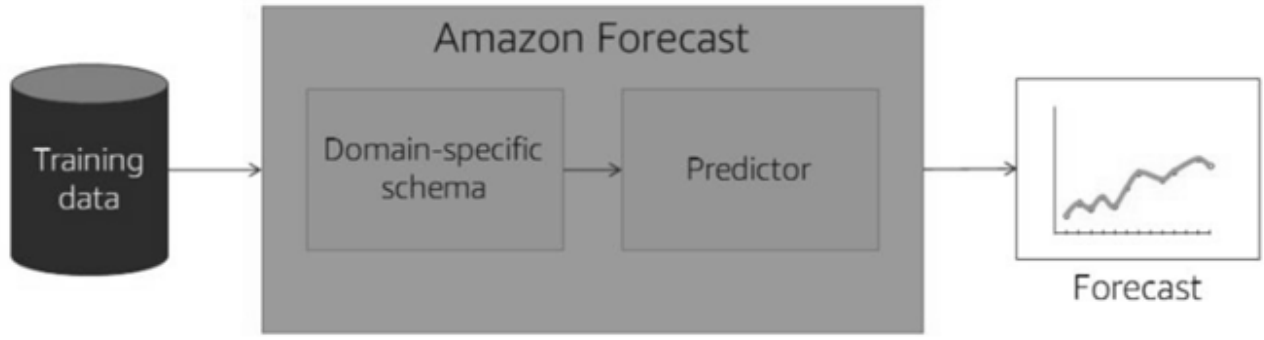

Рисунок 3.9 – Вибір галузі перед навчанням предиктора

3.2.1 Приклад прогнозу для роздрібної торгівлі

У наведеному нижче прикладі показано дані, необхідні для прогнозу роздрібного попиту (рис. 3.10).

Рисунок 3.10 – Приклад прогнозу роздрібного попиту

Для часового ряду потрібно вказати:

– відмітка часу – час, коли відбулася транзакція, в ідеалі у форматі UTC;

– предмет – ідентифікатор товару;

– кількість – скільки товарів було продано.

Метадані для елемента можуть включати категорію або колір предмета, наприклад. Посилання на дані часового ряду містить лише ідентифікатор товару, оскільки метадані продукту зазвичай не змінюються.

Ціна продажу або інші дані про рекламу є прикладами пов’язаних даних, які можуть створити більш корисний прогноз. Щоб зв’язати це із елементом, потрібно вказати позначку часу та ідентифікатор елемента.

3.2.2 Вибір алгоритму

Предиктори Amazon Forecast використовують алгоритм для навчання моделі. Потім вони використовують модель, щоб зробити прогноз, використовуючи групу вхідних наборів даних. Щоб допомогти розпочати роботу, Amazon Forecast надає такі попередньо визначені алгоритми:

– ARIMA – це широко використовуваний статистичний алгоритм для прогнозування часових рядів. Алгоритм особливо корисний для простих наборів даних із менш ніж 100 часовими рядами (лістинг 3.4);

```
arn:aws:forecast:::algorithm/ARIMA
```

Лістинг 3.4 – Виклик алгоритму ARIMA

– DeepAR+ – це запатентований алгоритм машинного навчання для прогнозування часових рядів із використанням рекурентних нейронних мереж (RNN). DeepAR+ найкраще працює з великими наборами даних, що містять сотні характерних часових рядів. Алгоритм приймає прогнозні пов’язані часові ряди та метадані елементів (лістинг 3.5);

```
arn:aws:forecast:::algorithm/Deep_AR_Plus
```

Лістинг 3.5 – Виклик алгоритму DeepAR+

– ETS – це часто використовуваний статистичний алгоритм для прогнозування часових рядів. Алгоритм особливо корисний для простих наборів даних із менш ніж 100 часовими рядами та наборів даних із сезонністю.

ETS обчислює середньозважене значення для всіх спостережень у наборі даних часового ряду як прогноз, з експоненціально зменшуючими ваговими показниками з часом (лістинг 3.6);

```
arn:aws:forecast:::algorithm/ETS
```

Лістинг 3.6 – Виклик алгоритму ETS

– NPTS – масштабований імовірнісний прогнозний алгоритм базової лінії. NPTS особливо корисний при роботі з розрідженими або переривчастими часовими рядами. Прогноз передбачає чотири варіанти алгоритму: стандартний NPTS, сезонний NPTS, кліматологічний синоптик і сезонний кліматологічний синоптик (лістинг 3.7);

```
arn:aws:forecast:::algorithm/NPTS
```

Лістинг 3.7 – Виклик алгоритму NPTS

– Prophet – це алгоритм прогнозування часових рядів, заснований на адитивній моделі, де нелінійні тенденції відповідають річній, тижневій та денній сезонності. Найкраще він працює з часовими рядами з сильними сезонними ефектами та історичними даними за кілька сезонів (лістинг 3.8).

```
arn:aws:forecast:::algorithm/Prophet
```

Лістинг 3.8 – Виклик алгоритму Prophet

Також можна використовувати функцію AutoML, яка пробує всі алгоритми, щоб побачити, який з них найкраще прогнозує дані.

3.2.3 Оцінка ефективності прогнозу моделі

Коли готуються дані для навчання у ML, зазвичай утримуються дані для використання під час перевірки та оцінки моделі. Утримувані дані зазвичай є випадковою вибіркою доступних даних. З даними часових рядів потрібно обробляти свої дані по-різному через кореляцію між часом.

Коли дані імпортуються, Amazon Forecast розбиває їх на тренувальні та тестові набори даних, що показано на рис. 3.11.

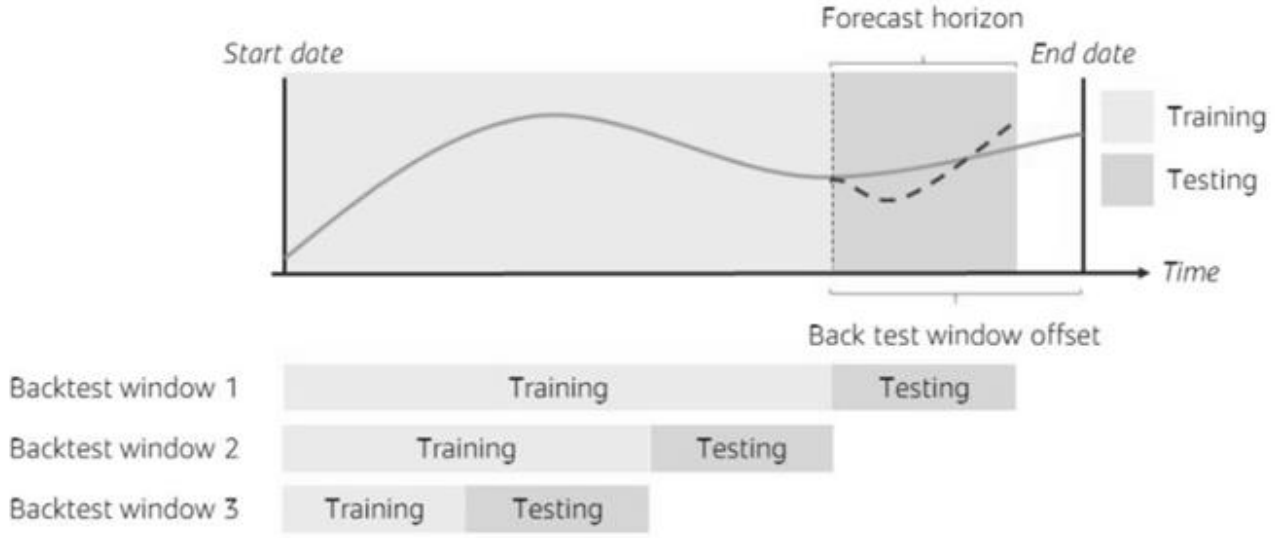

Рисунок 3.11 – Оцінка прогнозу зворотним тестуванням

Тренувальні дані використовуються для навчання моделі, яка потім перевіряється з даними, які були утримані. Ми можемо вказати кілька вікон зворотного тестування, які будуть розділяти дані кілька разів, навчати модель і використовувати метрики, щоб визначити, яка модель дає найкращі результати. Вікно зворотного тестування за замовчуванням – це 1 (рис. 3.11).

Можна змінити те, як Amazon Forecast розбиває дані, встановивши параметр BackTestWindowOffset під час створення предиктора. Якщо не встановити це значення, алгоритми використовують значення за замовчуванням.

Після того, як модель навчена, потрібно виміряти її точність, про що скажемо далі.

Першим показником оцінки Amazon Forecast є зважений квантиль втрати (wQuantileLoss). Коли Amazon Forecast створює прогноз, він надає імовірнісні прогнози для трьох різних квантилів — 10%, 50% і 90%. Ці квантилі прогнозу показують, скільки невизначеності пов’язано з кожним прогнозом (рис. 3.12).

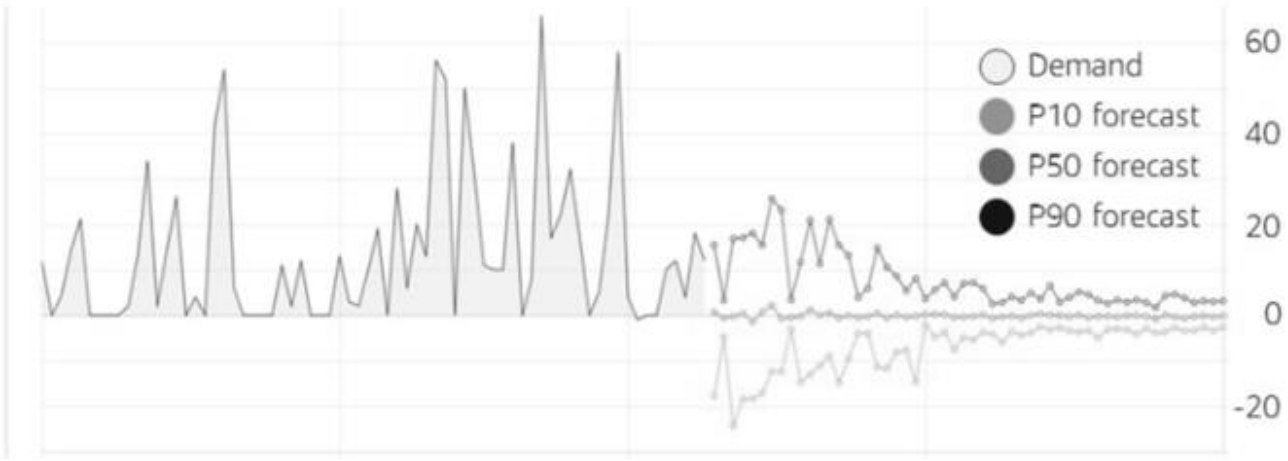

Рисунок 3.12 – Оцінка метрик через wQuantileLoss

Квантиль P10 передбачає, що в 10% випадків справжнє значення буде меншим за передбачене. Наприклад, припустимо, що ми – роздрібний продавець і хочемо спрогнозувати попит на зимові рукавички, які добре продаються лише восени та взимку. Скажімо, що у нас недостатньо місця для зберігання, а вартість інвестованого капіталу висока, або що нас турбує ціна надмірного запасу зимових рукавичок. Тоді можна використовувати квантиль P10, щоб замовити відносно невелику кількість зимових рукавичок. Ми знаємо, що прогноз P10 переоцінює попит на зимові рукавички лише в 10% випадків, тому ми розпродамо зимові рукавички в 90% випадків.

Квантиль P50 передбачає, що в 50% випадків справжнє значення буде меншим за передбачене. Продовжуючи приклад із зимовими рукавичками, скажімо, що ми знаємо, що попит на рукавички буде помірним, і нас не турбує надмірний запас. Тоді можна використовувати квантиль P50, щоб замовити рукавички.

Квантиль P90 передбачає, що в 90% випадків справжнє значення буде менше, ніж прогнозована вартість. Припустимо, ми визначили, що недостатній запас рукавичок призведе до великої втрати прибутку — наприклад, вартість непродавання рукавичок надзвичайно висока або вартість інвестованого капіталу низька. У цьому випадку можна використовувати квантиль P90 для замовлення рукавичок.

Amazon Forecast також обчислює пов’язані втрати (помилку) для кожного квантиля. Зважений квантиль втрат (wQuantileLoss) обчислює, наскільки

далекий від прогнозу певний квантиль від фактичного попиту в будь-якому напрямку. Нижчі показники wQuantileLoss означають, що прогнози моделі є більш надійними.

Середньоквадратична помилка (RMSE) є ще одним методом оцінки надійності прогнозів. Як і wQuantileLoss, RMSE обчислює, наскільки віддалені прогнозовані значення від фактичних даних тесту.

RMSE знаходить різницю між фактичним цільовим значенням у наборі даних і прогнозованим значенням для цього періоду часу, а потім квадратує відмінності (рис. 3.13, 3.14).

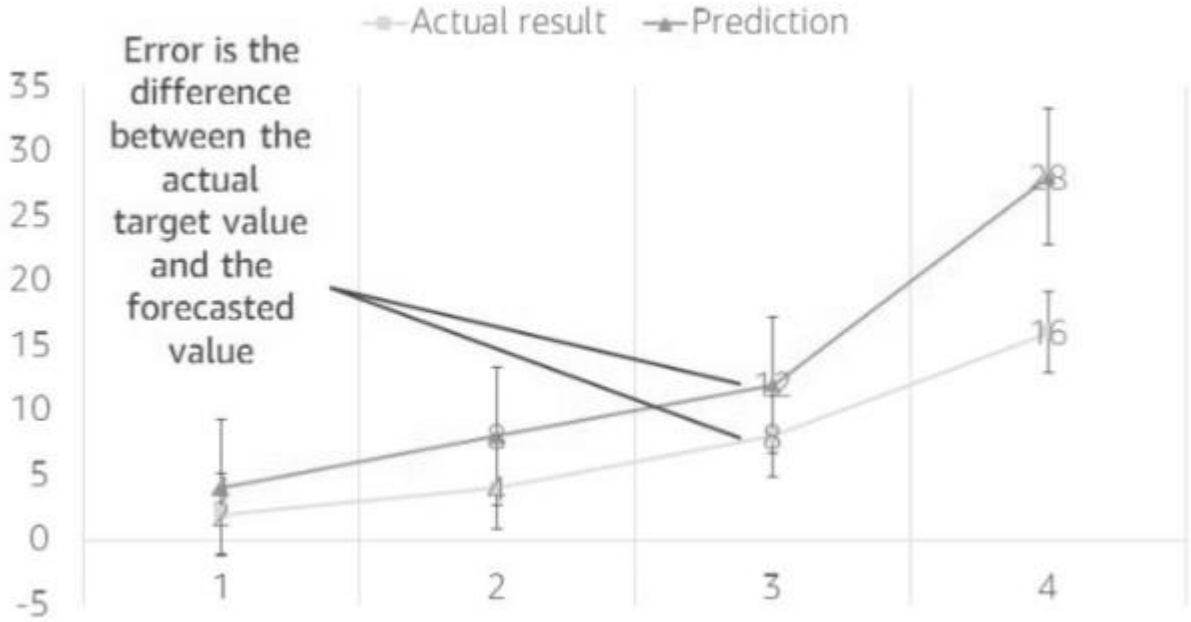

Рисунок 3.13 – Середньоквадратична помилка (графік)

Test	Actual result	Prediction	Deltas
1	2	4	2
2	4	8	4
3	8	12	4
4	16	28	12
		RMSE	6.708204

Рисунок 3.14 – Середньоквадратична помилка (обчислення)

У прикладі показано, як обчислити RMSE. Значення RMSE являє собою стандартне відхилення помилок передбачення. Цей тест хороший для валідності прогнозу, коли помилки переважно однакового розміру (тобто не так багато викидів). Нижчі показники RMSE показують, що прогнози моделі є більш надійними.

У прикладі на рис. 3.15 показано, як продавець може використовувати показники точності для оцінки прогнозу. Продавець хоче спрогнозувати попит на продажі певної марки взуття. Він вводить записи про продажі цього бренду в Amazon Forecast, щоб створити прогноз.

Рисунок 3.15 – Приклад точності моделі

Предиктор забезпечує прогнозований попит на 1000 пар із значеннями P10, P50 та P90. Значення wQuantileLoss вказують на те, що в 10% випадків (P10) буде продано менше 880 пар. Далі, у 50% випадків (P50) буде продано менше 1050 пар. Нарешті, у 90% випадків (P90) буде продано менше 1200 пар. Таким чином продавець може потім використовувати ці значення, щоб визначити, який рівень запасів утримувати. Визначення ґрунтується на оцінці ризику того, що продавець не може виконати замовлення або має надлишкові запаси.

3.3 Висновки до третього розділу

У третьому розділі даної роботи ми розглянули процес створення моделі для прогнозування продажів.

Наведемо ключові матеріали з третього розділу.

Дані часових рядів – це упорядковані дані, які містять елемент часу, що відрізняє їх від звичайних наборів даних.

Деякі задачі роботи з часом включають в себе обробку:

–різних форматів часу;

– відсутніх даних за допомогою зменшення дискретизації, підвищення дискретизації та згладжування даних;

– сезонності, як-от тижневі та річні цикли;

– уникнення поганих кореляцій.

Бібліотека pandas для Python пропонує підтримку даних часових рядів за допомогою функцій, які мають справу з часом.

У Amazon Forecast можна вибрати один із п’яти алгоритмів: ARIMA, DeepAR+, ETS, NPTS або Prophet.

Ми можемо використовувати Amazon Forecast для навчання та використання моделі для даних часових рядів.

Існують конкретні схеми, визначені для таких галузей, як роздрібна торгівля та планування ємності EC2, або можна використовувати користувацьку схему.

Для початку навчання моделі потрібно надати принаймні дані часового ряду, але також можна надати метадані та пов’язані дані, щоб розширити відомості моделі.

Як і в більшості задач контрольованого машинного навчання, дані поділяються на дані тренування та тестування, але цей поділ враховує елемент часу.

Метрики RMSE та wQuantileLoss використовуються для оцінки ефективності прогнозу моделі.

4 ТЕСТУВАННЯ СТВОРЕННЯ ПРОГНОЗІВ В AMAZON FORECAST

У цьому розділі описана підготовка набору даних для створення прогнозу за допомогою Amazon Forecast і тестування отриманої моделі.

4.1 Вхід у робоче середовище JupyterLab

Методика відкривання блокноту Jupyter з Amazon SageMaker, щоб підключитися до середовища JupyterLab.

Щоб відкрити JupyterLab (рис. 4.1) необхідно:

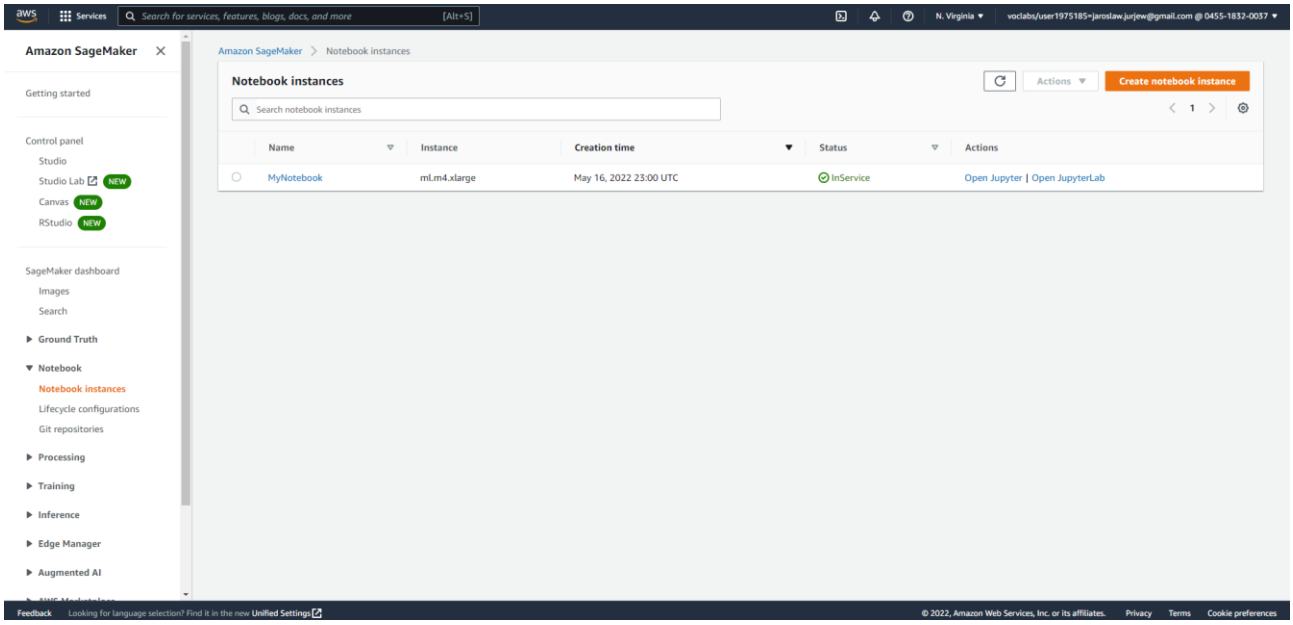

Рисунок 4.1 – Інтерфейс Amazon SageMaker

- на AWS Management Console у меню Services вибрати Amazon SageMaker;

- у навігаційному меню вибирати Notebook instances;

- знайти навчальний екземпляр блокнота і у кінці його рядка обираємо Open JupyterLab, щоб потрапити в середовище JupyterLab.

Далі необхідно відкрити екземпляр блокнота JupyterLab (рис. 4.2):

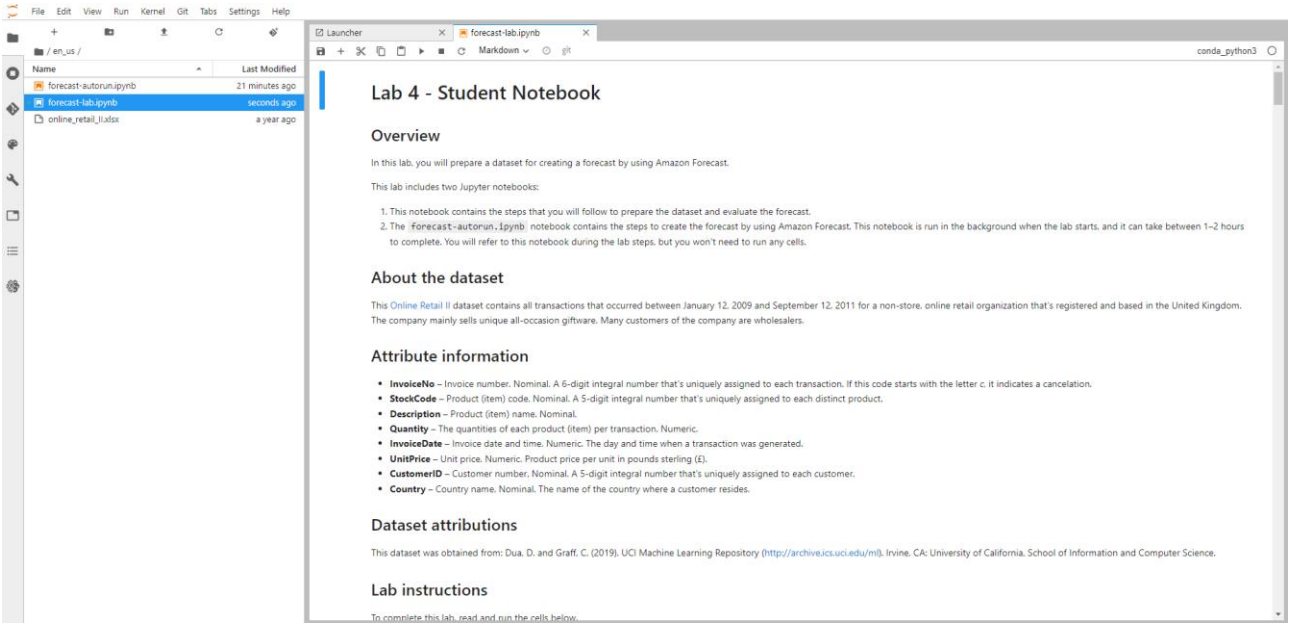

Рисунок 4.2 – Інтерфейс JupyterLab

- у середовищі JupyterLab перейти у браузер файлів на лівій панелі та знайти файл forecast-lab.ipynb;

- відкриємо файл en_us/forecast-lab.ipynb, вибравши його.

4.2 Про набір даних

Набір даних Online Retail II, який буде використовуватися як експериментальні дані для тестування, містить усі транзакції, які відбулися між 12 січня 2009 року та 12 вересня 2011 року для онлайн-магазину, який зареєстрований та розташований у Великобританії. В основному компанія продає унікальні сувеніри на всі випадки життя. Багато клієнтів компанії – оптові торговці [6].

Інформація про атрибути:

– InvoiceNo – номер рахунку-фактури. Номінальний. 6-значний цілісний номер, який унікально призначається кожній транзакції. Якщо цей код починається з літери «c», це вказує на скасування;

– StockCode – код продукту (товару). Номінальний. 5-значний цілісний номер, який унікально призначається кожному окремому продукту;

– Description – назва продукту (товару). Номінальна;

– Quantity – кількість кожного продукту (товару) за транзакцію. Числова;

– InvoiceDate – дата та час рахунка-фактури. Числова. День і час, коли була створена транзакція;

– UnitPrice – ціна за одиницю. Числова. Ціна продукту за одиницю в фунтах стерлінгів (£);

– CustomerID – номер клієнта. Номінальний. 5-значний цілісний номер, який унікально призначається кожному клієнту.

– Country – назва країни. Номінальна. Назва країни, де проживає клієнт.

4.3 Імпорт пакетів Python

Почнемо з імпортування пакетів Python, які нам потрібні. Дамо пояснення до послідуючого лістингу 4.1:

– boto3 представляє AWS SDK для Python (Boto3), який є бібліотекою Python для AWS;

– pandas надає DataFrames для маніпулювання даними часових рядів;

– matplotlib надає функції побудови графіків;

– sagemaker представляє API, необхідний для роботи з Amazon SageMaker;

– time, sys, os, io та json забезпечують допоміжні функції.

```
import warnings
warnings.filterwarnings('ignore')
bucket_name='c42001a55231l1480639t1w1491473470-forecastbucket-
1o8lgpvqlipvf'

import boto3
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import sagemaker
import time, sys, os, io, json
import xlrd
```

Лістинг 4.1 – Імпорт пакетів Python

4.4 Дослідження даних

Дані представлені у форматі Microsoft Excel, pandas може читати файли Excel. Завантаження цих даних може зайняти декілька хвилин (лістинг 4.2).

```
retail = pd.read_excel('online_retail_II.xlsx',engine='openpyxl')
```

Лістинг 4.2 – Завантаження даних

Згідно з описом набору даних, деякі значення відсутні. Щоб все було просто, видаляємо все, що має відсутнє значення (лістинг 4.3).

```
retail = retail.dropna()
```

Лістинг 4.3 – Видалення відсутніх даних

Почнемо з вивчення даних. Дізнаємося скільки рядків і стовпців міститься в наборі даних (лістинг 4.4).

```
retail.shape
(417534, 8)
```

Лістинг 4.4 – Визначення кількості рядків і стовпців

Дізнаємося які існують типи даних (лістинг 4.5, табл. 4.1).

```
retail.dtypes
```

Лістинг 4.5 – Команда визначення типів даних

Таблиця 4.1 – Типи даних у наборі

Invoice	object
StockCode	object
Description	object
Quantity	int64
InvoiceDate	datetime64[ns]
Price	float64
Customer ID	float64
Country	object
dtype:	object

Дізнаємося як виглядають дані (лістинг 4.6, табл. 4.2).

```
retail.head(20)
```

Лістинг 4.6 – Команда відображення даних з 20 рядків

Таблиця 4.2 – Дані набору

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom
5	489434	22064	PINK DOUGHNUT TRINKET POT	24	2009-12-01 07:45:00	1.65	13085.0	United Kingdom
6	489434	21871	SAVE THE PLANET MUG	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom
7	489434	21523	FANCY FONT HOME SWEET HOME DOORMAT	10	2009-12-01 07:45:00	5.95	13085.0	United Kingdom
8	489435	22350	CAT BOWL	12	2009-12-01 07:46:00	2.55	13085.0	United Kingdom
9	489435	22349	DOG BOWL , CHASING BALL DESIGN	12	2009-12-01 07:46:00	3.75	13085.0	United Kingdom
10	489435	22195	HEART MEASURING SPOONS LARGE	24	2009-12-01 07:46:00	1.65	13085.0	United Kingdom
11	489435	22353	LUNCHBOX WITH CUTLERY FAIRY CAKES	12	2009-12-01 07:46:00	2.55	13085.0	United Kingdom

Продовження таблиці 4.2

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
12	489436	48173C	DOOR MAT BLACK FLOCK	10	2009-12-01 09:06:00	5.95	13078.0	United Kingdom
13	489436	21755	LOVE BUILDING BLOCK WORD	18	2009-12-01 09:06:00	5.45	13078.0	United Kingdom
14	489436	21754	HOME BUILDING BLOCK WORD	3	2009-12-01 09:06:00	5.95	13078.0	United Kingdom
15	489436	84879	ASSORTED COLOUR BIRD ORNAMENT	16	2009-12-01 09:06:00	1.69	13078.0	United Kingdom
16	489436	22119	PEACE WOODEN BLOCK LETTERS	3	2009-12-01 09:06:00	6.95	13078.0	United Kingdom
17	489436	22142	CHRISTMAS CRAFT WHITE FAIRY	12	2009-12-01 09:06:00	1.45	13078.0	United Kingdom
18	489436	22296	HEART IVORY TRELLIS LARGE	12	2009-12-01 09:06:00	1.65	13078.0	United Kingdom
19	489436	22295	HEART FILIGREE DOVE LARGE	12	2009-12-01 09:06:00	1.65	13078.0	United Kingdom

Як ми вже знаємо Amazon Forecast має схеми для таких галузей, як роздрібна торгівля. Цільовий часовий ряд – це історичні дані часових рядів для кожного товару або продукту, які продаються роздрібною організацією. Необхідні для заповнення наступні поля:

– item_id (рядок) – унікальний ідентифікатор товару або продукту, на який ми хочемо передбачити попит;

– timestamp (мітка часу);

– demand (число з рухомою комою) – кількість продажів для цього товару на мітці часу. Це також цільове поле, для якого Amazon Forecast створює прогноз.

Перевіривши попередні дані, ми бачимо певні стовпці, які не потрібні для дослідження. Ми можемо скинути ці стовпці. Стовпці, які можна відкинути: Invoice, Description та Customer ID. Треба зауважити, що можливо елементи в тому самому порядку (як показано в стовпці Invoice) можуть мати кореляцію, яка впливає на модель, однак для нашого тестування ми проігноруємо цю можливість.

Відкинемо стовпці, які нам не потрібні (лістинг 4.7).

```
retail = retail[['StockCode','Quantity','Price','Country','InvoiceDate']]
```

Лістинг 4.7 – Визначення стовпців для подальшого розгляду

Стовпець InvoiceDate – це дані дати й часу. Ми можемо повідомити бібліотеку pandas про це за допомогою функції to_datetime. Можна досліджувати дані за часом, встановивши для індексу DataFrame стовпець InvoiceDate (лістинг 4.8).

```
retail['InvoiceDate'] = pd.to_datetime(retail.InvoiceDate)
retail = retail.set_index('InvoiceDate')
```

Лістинг 4.8 – Визначення даних дати і часу

Тепер перевіримо оновлений DataFrame. Кількість рядків і стовпців показана в лістингу 4.9.

```
retail.shape
(417534, 4)
```

Лістинг 4.9 – Оновлена кількість рядків і стовпців

Вигляд нових даних показано в лістингу 4.10 і табл. 4.3.

```
retail.head()
```

Лістинг 4.10 – Команда відображення даних

Таблиця 4.3 – Оновлені дані набору

InvoiceDate	StockCode	Quantity	Price	Country
2009-12-01 07:45:00	85048	12	6.95	United Kingdom
2009-12-01 07:45:00	79323P	12	6.75	United Kingdom
2009-12-01 07:45:00	79323W	12	6.75	United Kingdom
2009-12-01 07:45:00	22041	48	2.10	United Kingdom
2009-12-01 07:45:00	21232	24	1.25	United Kingdom

Зауважимо, що InvoiceDate є індексом, і він відображається в першому стовпці. Оскільки ми встановлюємо індекс для даних дати і часу, то можемо використовувати його для вибору даних. Щоб вибрати всі рядки з певної дати, використовуємо дату в індексі (лістинг 4.11, табл. 4.4).

```
retail['2010-01-04']
633 rows × 4 columns
```

Лістинг 4.11 – Вибір рядків за певну дату

Таблиця 4.4 – Результат вибору рядків за певну дату

InvoiceDate	StockCode	Quantity	Price	Country
2010-01-04 09:24:00	TEST001	5	4.50	United Kingdom
2010-01-04 09:43:00	21539	-1	4.25	United Kingdom
2010-01-04 09:53:00	TEST001	5	4.50	United Kingdom
2010-01-04 10:28:00	21844	36	2.55	United Kingdom

Продовження таблиці 4.4

InvoiceDate	StockCode	Quantity	Price	Country
2010-01-04 10:28:00	21533	12	4.25	United Kingdom
...	...	...	...	...
2010-01-04 17:39:00	90214G	1	1.25	United Kingdom
2010-01-04 17:39:00	90214N	1	1.25	United Kingdom
2010-01-04 17:39:00	90214N	1	1.25	United Kingdom
2010-01-04 17:39:00	90214C	1	1.25	United Kingdom
2010-01-04 17:39:00	21690	2	3.75	United Kingdom

Можна використовувати частини дати та діапазони дат. Переглянемо рядки за січень та лютий (лістинг 4.12, табл. 4.5).

```
retail['2010-01':'2010-02']
46345 rows × 4 columns
```

Лістинг 4.12 – Вибір рядків певного діапазону дат

Таблиця 4.5 – Результат вибору рядків певного діапазону дат

InvoiceDate	StockCode	Quantity	Price	Country
2010-01-04 09:24:00	TEST001	5	4.50	United Kingdom
2010-01-04 09:43:00	21539	-1	4.25	United Kingdom
2010-01-04 09:53:00	TEST001	5	4.50	United Kingdom

Продовження таблиці 4.5

InvoiceDate	StockCode	Quantity	Price	Country
2010-01-04 10:28:00	21844	36	2.55	United Kingdom
2010-01-04 10:28:00	21533	12	4.25	United Kingdom
...	...	...	...	...
2010-02-28 16:14:00	84279B	1	3.75	United Kingdom
2010-02-28 16:14:00	84882	1	3.75	United Kingdom
2010-02-28 16:14:00	84882	1	3.75	United Kingdom
2010-02-28 16:14:00	44242B	5	1.25	United Kingdom
2010-02-28 16:16:00	10133	40	0.85	United Kingdom

Лістинг 4.13 показує початок діапазону дат.

```
retail.index.min()
Timestamp('2009-12-01 07:45:00')
```

Лістинг 4.13 – Початок діапазону дат

Лістинг 4.14 показує кінець діапазону дат.

```
retail.index.max()
Timestamp('2010-12-09 20:01:00')
```

Лістинг 4.14 – Кінець діапазону дат

За допомогою pandas ми можемо легко витягувати інформацію про дату. Можна отримати інформацію про дату для подальшого вивчення даних і пошуку тенденцій, пов’язаних із часом. Витягнемо рік, місяць і день тижня (лістинг 4.15, 4.16).

```
retail['Year'] = retail.index.year
retail['Month'] = retail.index.month
retail['weekday_name'] = retail.index.day_name()
```

Лістинг 4.15 – Витягування інформації про рік, місяць і день

```
retail.head()
```

Лістинг 4.16 – Команда відображення даних

Таблиця 4.6 – Показ результату з інформацією про рік, місяць і день

	StockCode	Quantity	Price	Country	Year	Month	weekday_name
InvoiceDate							
2009-12-01 07:45:00	85048	12	6.95	United Kingdom	2009	12	Tuesday
2009-12-01 07:45:00	79323P	12	6.75	United Kingdom	2009	12	Tuesday
2009-12-01 07:45:00	79323W	12	6.75	United Kingdom	2009	12	Tuesday
2009-12-01 07:45:00	22041	48	2.10	United Kingdom	2009	12	Tuesday
2009-12-01 07:45:00	21232	24	1.25	United Kingdom	2009	12	Tuesday

Набір даних, який ми зараз маємо, включає покупки, зроблені між груднем 2009 і груднем 2010 року. Цілком розумно припустити, що ці дані мають певну сезонність. Тепер дізнаємося, чи є сезонність (лістинг 4.17, рис. 4.3).

```
retail.Month.value_counts(sort=False).plot(kind='bar')
```

Лістинг 4.17 – Команда виведення графіку за рік

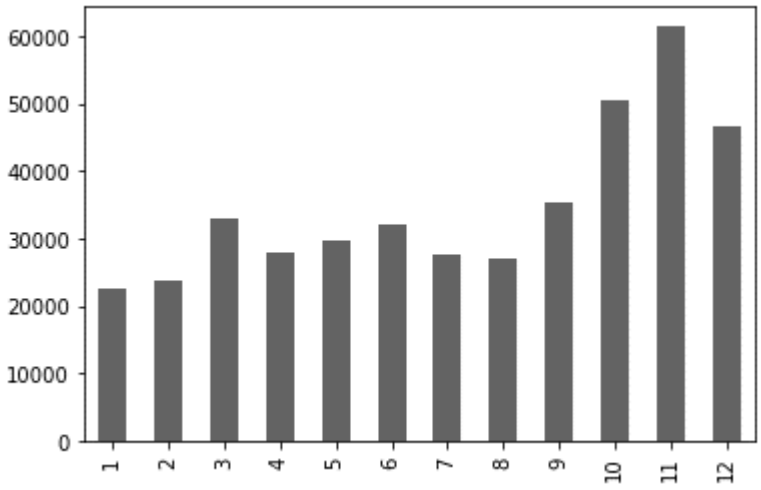

Рисунок 4.3 – Графік покупок, зроблених між груднем 2009 і груднем 2010

З діаграми можна вивести деяку сезонність: листопад і грудень здаються вищими за решту року. Здається, 4 квартал вищий, ніж інші квартали. Для першого, другого і третього кварталів: в останньому місяці кварталу (місяці 3, 6 та 9), здається, спостерігаються стрибки. Очевидні сезонні закономірності.

Тепер дослідимо, чи є сезонність протягом тижня (лістинг 4.18, рис. 4.4).

```
      day_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"]
      retail.weekday_name.value_counts(sort=False).loc[day_order].plot(kind='bar
')
```

Лістинг 4.18 – Команда виведення графіку за тиждень

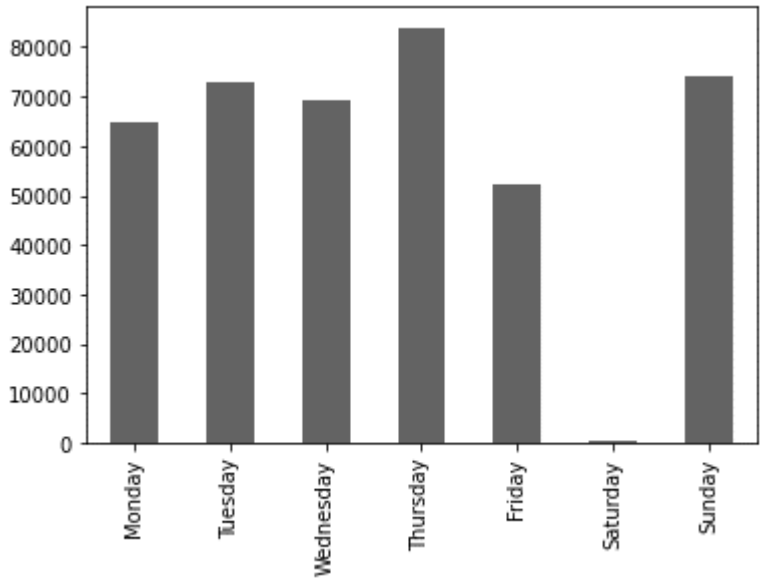

Рисунок 4.4 – Графік покупок у розрізі тижня

Субота показує дуже мало замовлень.

4.5 Очищення та зменшення розміру даних

У цьому пункті ми зменшимо розмір даних. Ми також видалимо будь-які аномалії, такі як від’ємні ціни, відхилення та дані про країни.

4.5.1 Скорочення числа країн

Переглянемо дані Country (лістинг 4.19, 4.20, табл. 4.7).

```
retail.Country.unique()
array(['United Kingdom', 'France', 'USA', 'Belgium', 'Australia', 'EIRE',
       'Germany', 'Portugal', 'Japan', 'Denmark', 'Netherlands', 'Poland',
       'Spain', 'Channel Islands', 'Italy', 'Cyprus', 'Greece', 'Norway',
       'Austria', 'Sweden', 'United Arab Emirates', 'Finland',
       'Switzerland', 'Unspecified', 'Nigeria', 'Malta', 'RSA',
       'Singapore', 'Bahrain', 'Thailand', 'Israel', 'Lithuania',
       'West Indies', 'Korea', 'Brazil', 'Canada', 'Iceland'],
      dtype=object)
```

Лістинг 4.19 – Масив даних зі списком країн

```
retail.Country.value_counts()
```

Лістинг 4.20 – Виведення кількості замовлень за країнами

Таблиця 4.7 – Кількість замовлень за країнами

Name: Country	dtype: int64
United Kingdom	379423
EIRE	8710
Germany	8129
France	5710
Netherlands	2769
Spain	1278
Switzerland	1187
Belgium	1054
Portugal	1024
Channel Islands	906
Sweden	883
Italy	731
Australia	654
Cyprus	554
Austria	537
Greece	517
Denmark	428
Norway	369
Finland	354
United Arab Emirates	318
Unspecified	280
USA	244
Japan	224
Poland	194
Malta	172
Lithuania	154
Singapore	117
Canada	77

Продовження таблиці 4.7

Name: Country	dtype: int64
Thailand	76
Israel	74
Iceland	71
RSA	65
Korea	63
Brazil	62
West Indies	54
Bahrain	42
Nigeria	30

Здається, більшість даних стосується Сполученого Королівства. Щоб полегшити роботу, відфільтруємо дані за Великобританією (лістинг 4.21).

```
country_filter = ['United Kingdom']
retail = retail[retail.Country.isin(country_filter)]
```

Лістинг 4.21 – Фільтрування даних за країною

Оскільки стовпець Country містить лише те саме значення, його можна видалити (лістинг 4.22, 4.23, табл. 4.8).

```
retail = retail[['StockCode','Quantity','Price']]
```

Лістинг 4.22 – Вибір потрібних стовпців

```
retail.head()
```

Лістинг 4.23 – Команда виведення даних

Таблиця 4.8 – Результат видалення стовпця Country

InvoiceDate	StockCode	Quantity	Price
2009-12-01 07:45:00	85048	12	6.95
2009-12-01 07:45:00	79323P	12	6.75
2009-12-01 07:45:00	79323W	12	6.75
2009-12-01 07:45:00	22041	48	2.10
2009-12-01 07:45:00	21232	24	1.25

4.5.2 Перевірка StockCode та видалення аномалій

Переглянемо розподіл стовпця StockCode (лістинг 4.24, табл. 4.9).

```
retail.StockCode.describe()
```

Лістинг 4.24 –Перегляд стовпця StockCode

Таблиця 4.9 – Результат роботи команди для перегляду стовпця StockCode

Name: StockCode	dtype: object
count	379423
unique	4015
top	85123A
freq	3140

Існує 4015 унікальних значень для StockCode. Швидкий графік підрахунку може дати деяке уявлення про те, як значення розподіляються (лістинг 4.25, рис. 4.5).

```
retail.StockCode.value_counts().plot()
```

Лістинг 4.25 – Виведення графіку розподілу значень StockCode

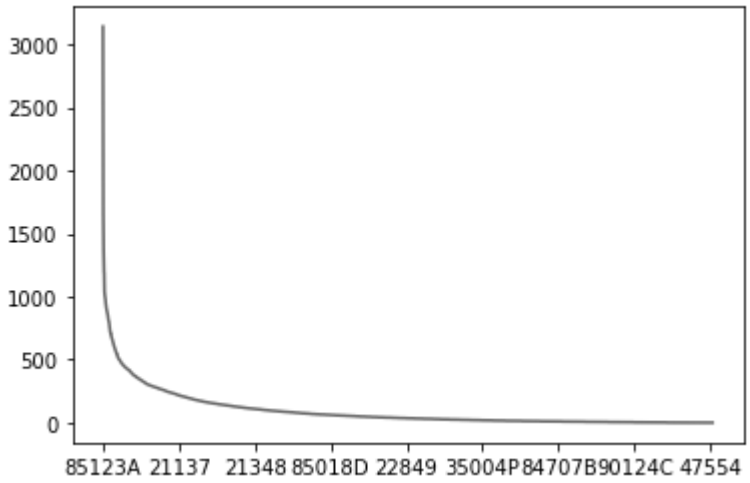

Рисунок 4.5 – Графік розподілу значень StockCode

Здається, що є кілька частопродаваних товарів, за спиною яких довгий хвіст. А зараз перевіримо стовпець Quantity (лістинг 4.26, 4.27, табл. 4.10, рис. 4.6).

```
retail.Quantity.describe()
```

Лістинг 4.26 – Перегляд стовпця Quantity

Таблиця 4.10 – Результат роботи команди для перегляду стовпця Quantity

Name: Quantity	dtype: float64
count	379423.000000
mean	11.451517
std	68.943709
min	-9360.000000
25%	2.000000
50%	4.000000
75%	12.000000
max	10000.000000

```
retail.Quantity.plot()
```

Лістинг 4.27 – Виведення графіку значень Quantity

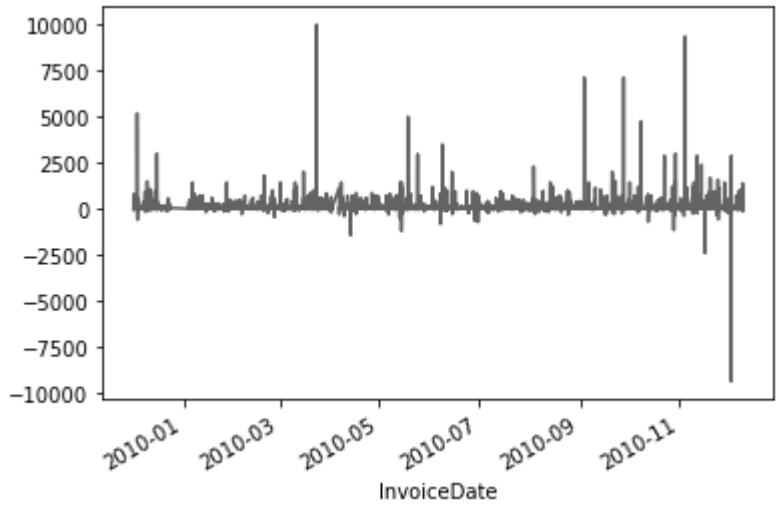

Рисунок 4.6 – Графік значень Quantity

З початкового графіку звернемо увагу на пару цікавих аспектів:

– Виявляється, є негативні величини;

– Протягом року спостерігаються дуже великі шипи.

Від'ємні та нульові величини можуть вплинути на прогноз, якщо ми не знаємо, чому існують ці значення. Щоб спростити ситуацію, видалимо від'ємні та нульові величини (лістинг 4.28).

```
retail = retail[retail.Quantity>0]
```

Лістинг 4.28 – Видалення від'ємних і нульових значень

Тепер перевіримо Price (лістинг 4.29, 4.30, табл. 4.11, рис. 4.7).

```
retail.Price.describe()
```

Лістинг 4.29 – Перегляд стовпця Price

Таблиця 4.11 – Результат роботи команди для перегляду стовпця Price

Name: Price	dtype: float64
count	370951.000000
mean	3.145220
std	30.551482
min	0.000000
25%	1.250000
50%	1.950000
75%	3.750000
max	10953.500000

```
retail.Price.plot()
```

Лістинг 4.30 – Виведення графіку значень Price

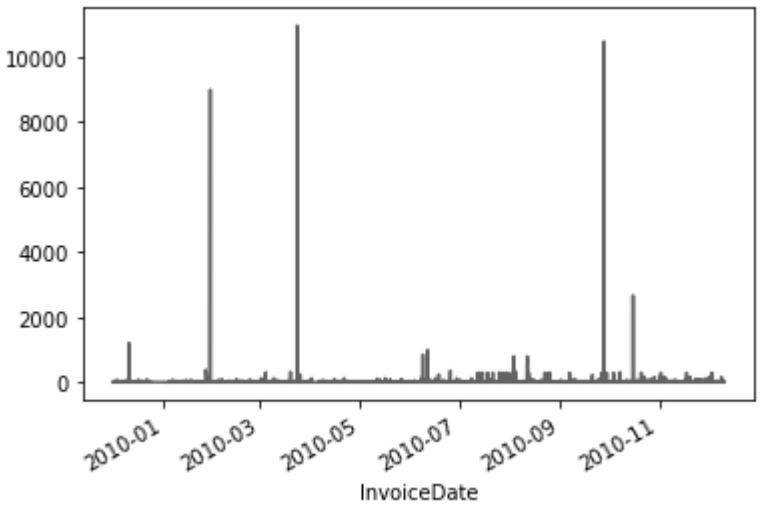

Рисунок 4.7 – Графік значень Price

Графік демонструє явні стрибки цін. Тепер ми спробуємо з’ясувати, чому існують ці шипи (лістинг 4.31, табл. 4.12).

```
retail[retail.Price>500].head()
```

Лістинг 4.31 – Виведення значень Price більше ніж 500

Таблиця 4.12 – Результат виведення значень Price більше ніж 500

InvoiceDate	StockCode	Quantity	Price
2009-12-10 11:50:00	M	1	1213.02
2010-01-29 11:04:00	M	1	8985.60
2010-03-23 15:22:00	M	1	10953.50
2010-06-08 16:39:00	M	1	849.45
2010-06-11 15:54:00	M	1	1000.63

Значення M у StockCode виглядає незвично. Якби у нас був доступ до експерта з галузі, ми могли б дізнатися про важливість M. Оскільки ми не можемо цього зробити, то для спрощення видалимо все, що має значення M у StockCode (лістинг 4.32, 4.33, табл. 4.13).

```
[retail = retail[retail.StockCode!='M']
```

Лістинг 4.32 – Видалення значень M у StockCode

```
retail.Price.describe()
```

Лістинг 4.33 – Перегляд стовпця Price

Таблиця 4.13 – Результат роботи команди для перегляду стовпця Price

Name: Price	dtype: float64
count	370576.000000
mean	3.009463
std	4.576951
min	0.000000
25%	1.250000
50%	1.950000
75%	3.750000
max	387.540000

Цей результат кращий, але максимальне значення все ще високе. Тепер детальніше дослідимо цю ситуацію (лістинг 4.34, табл. 4.14).

```
retail[retail.Price>300].head(20)
```

Лістинг 4.34 – Виведення значень Price більше ніж 300

Таблиця 4.14 – Результат виведення значень Price більше ніж 300

InvoiceDate	StockCode	Quantity	Price
2010-01-26 16:29:00	ADJUST	1	342.80
2010-01-26 17:28:00	ADJUST	1	387.54
2010-06-25 14:15:00	ADJUST2	1	300.13
2010-06-25 14:15:00	ADJUST2	1	358.47
2010-08-04 11:38:00	POST	1	334.88

Схоже, відбулися якісь коригування. Ми також видалимо всі дані, які

показують ці коригування (лістинг 4.35, 4.36, табл. 4.15).

```
stockcodes = ['ADJUST', 'ADJUST2', 'POST']
retail = retail[~retail.StockCode.isin(stockcodes)]
```

Лістинг 4.35 – Видалення значень коригування

```
retail.Price.describe()
```

Лістинг 4.36 – Перегляд стовпця Price

Таблиця 4.15 – Результат роботи команди для перегляду стовпця Price

Name: Price	dtype: float64
count	370554.000000
mean	3.002500
std	4.363688
min	0.000000
25%	1.250000
50%	1.950000
75%	3.750000
max	295.000000

Тепер переглянемо товари з нульовою ціною (лістинг 4.37, табл. 4.16).

```
retail[retail.Price==0].count
```

Лістинг 4.37 – Виведення товарів з нульовою ціною

Таблиця 4.16 – Результати виведення товарів з нульовою ціною

InvoiceDate	StockCode	Quantity	Price
2009-12-02 13:34:00	22076	12	0.0
2009-12-03 11:19:00	48185	2	0.0
2009-12-08 15:25:00	22065	1	0.0
2009-12-08 15:25:00	22142	12	0.0
2009-12-15 13:49:00	85042	8	0.0
2009-12-18 14:22:00	21143	12	0.0
2010-01-06 14:54:00	79320	24	0.0
2010-01-15 12:43:00	21533	12	0.0
2010-02-12 14:58:00	TEST001	5	0.0
2010-02-12 15:47:00	TEST001	5	0.0
2010-03-04 11:44:00	21662	1	0.0
2010-04-01 17:13:00	22459	8	0.0
2010-04-01 17:13:00	22458	8	0.0
2010-06-11 11:12:00	21765	1	0.0

Продовження таблиці 4.16

InvoiceDate	StockCode	Quantity	Price
2010-06-17 10:12:00	20914	2	0.0
2010-06-24 12:34:00	22423	5	0.0
2010-07-19 13:13:00	22690	6	0.0
2010-09-27 16:59:00	46000M	648	0.0
2010-09-30 12:19:00	22218	2	0.0
2010-10-18 15:13:00	22121	1	0.0
2010-11-07 14:26:00	21843	2	0.0

У цих результатах не так багато значень, тому ми можемо відмовитися від товарів із нульовою ціною (лістинг 4.38).

```
retail = retail[retail.Price>0]
```

Лістинг 4.38 – Виключення товарів з нульовою ціною

4.5.3 Розбиття даних

Дані часових рядів, які нам потрібні для створення прогнозу, вимагають позначки часу, ідентифікатора елемента та запиту. Ці функції відображатимуться у стовпцях InvoiceDate, StockCode та Quantity.

Пов’язані дані часових рядів мають мітку часу, ідентифікатор елемента та ціну. Ці функції відображатимуться у стовпцях InvoiceDate, StockCode та Price.

Створимо два DataFrame (лістинг 4.39).

```
df_time_series = retail[['StockCode','Quantity']]
df_related_time_series = retail[['StockCode','Price']]
```

Лістинг 4.39 – Створення часового ряду і пов’язаних з ним даних

4.5.4 Зменшення дискретизації

Тепер розглянемо один предмет (лістинг 4.40, табл. 4.17).

```
df_time_series[df_time_series.StockCode==21232]['2009-12-01']
```

Лістинг 4.40 – Вибір одиниці товару

Таблиця 4.17 – Результати вибору одиниці товару

InvoiceDate	StockCode	Quantity
2009-12-01 07:45:00	21232	24
2009-12-01 10:49:00	21232	48
2009-12-01 12:13:00	21232	3
2009-12-01 12:14:00	21232	20
2009-12-01 13:31:00	21232	4
2009-12-01 13:37:00	21232	12
2009-12-01 13:43:00	21232	24
2009-12-01 14:19:00	21232	12
2009-12-01 15:26:00	21232	12
2009-12-01 16:18:00	21232	12

Можна побачити кілька замовлень за кожен день. Ми хочемо створити прогноз, який передбачає щоденний попит. Ми повинні зменшити вибірку даних з окремих замовлень до щоденної загальної суми. Замовлення на кожен день можна підсумувати, тому що загальний попит за день – це значення, яке прогнозується.

pandas надає для цієї мети функцію resample (лістинг 4.41). Функція sum буде підсумовувати стовпець Quantity. Ми також скинемо індекс на основі значення InvoiceDate. Однак цього разу це буде дата без годинної частини (лістинг 4.42, 4.43, табл. 4.18). Цей процес може зайняти близько хвилини.

```
df_time_series=df_time_series.groupby('StockCode').resample('D').sum().res
et_index()
```

Лістинг 4.41 – Робота функції resample

```
df_time_series['InvoiceDate'] = pd.to_datetime(df_time_series.InvoiceDate)
df_time_series = df_time_series.set_index('InvoiceDate')
df_time_series.head()
```

Лістинг 4.42 – Прибирання годинної частини

Таблиця 4.18 – Результати прибирання годинної частини

InvoiceDate	StockCode	Quantity
2009-12-01	10002	12
2009-12-02	10002	0
2009-12-03	10002	7
2009-12-04	10002	25
2009-12-05	10002	0

```
df_time_series=df_time_series.groupby('StockCode').resample('D').sum().res
et_index().set_index(['InvoiceDate'])
```

Лістинг 4.43 – Скидання індексу InvoiceDate

Перевіримо новий DataFrame (лістинг 4.44, табл. 4.19).

```
df_time_series[df_time_series.StockCode==21232]
373 rows × 2 columns
```

Лістинг 4.44 – Перегляд нового DataFrame зі значенням StockCode 21232

Таблиця 4.19 – Відображення товару з кодом запасу 21232

InvoiceDate	StockCode	Quantity
2009-12-01	21232	171
2009-12-02	21232	164
2009-12-03	21232	192
2009-12-04	21232	264
2009-12-05	21232	36
...	...	...
2010-12-04	21232	0
2010-12-05	21232	4
2010-12-06	21232	12
2010-12-07	21232	28
2010-12-08	21232	61

Тепер замовлення має один запис для кожного дня.

Повторімо цей процес із пов’язаними даними часових рядів (лістинг 4.45, 4.46, 4.47, табл. 4.20, 4.21).

```
df_related_time_series.head()
```

Лістинг 4.45 – Виведення пов’язаних даних

Таблиця 4.20 – Результат виведення пов’язаних даних

InvoiceDate	StockCode	Price
2009-12-01 07:45:00	85048	6.95
2009-12-01 07:45:00	79323P	6.75
2009-12-01 07:45:00	79323W	6.75
2009-12-01 07:45:00	22041	2.10
2009-12-01 07:45:00	21232	1.25

```
df_related_time_series2=df_related_time_series.groupby('StockCode').resamp
le('D').mean().reset_index().set_index(['InvoiceDate','StockCode'])
```

Лістинг 4.46 – Скидання індексу InvoiceDate

```
df_related_time_series2.head(20)
```

Лістинг 4.47 – Команда виведення даних

Таблиця 4.21 – Результат виконання команди виведення даних

InvoiceDate	StockCode	Price
2009-12-01	10002	0.85
2009-12-02	10002	NaN
2009-12-03	10002	0.85
2009-12-04	10002	0.85
2009-12-05	10002	NaN
2009-12-06	10002	0.85
2009-12-07	10002	0.85
2009-12-08	10002	NaN
2009-12-09	10002	NaN
2009-12-10	10002	NaN
2009-12-11	10002	0.85
2009-12-12	10002	NaN
2009-12-13	10002	NaN
2009-12-14	10002	0.85
2009-12-15	10002	NaN
2009-12-16	10002	NaN
2009-12-17	10002	NaN
2009-12-18	10002	NaN
2009-12-19	10002	NaN
2009-12-20	10002	NaN

Деякі з попередніх значень відображаються як NaN. Вони означають, що на цей продукт не було замовлень на ті дні, і тому він не має ціни. Слід заповнити ці значення NaN числовим значенням (лістинг 4.48, табл. 4.22).

```
retail[retail.StockCode == 10002]['2009-12']
```

Лістинг 4.48 – Вибір одиниці товару

Таблиця 4.22 – Результат вибору одиниці товару

InvoiceDate	StockCode	Quantity	Price
2009-12-01 09:08:00	10002	12	0.85
2009-12-03 13:49:00	10002	1	0.85
2009-12-03 13:49:00	10002	1	0.85
2009-12-03 19:13:00	10002	1	0.85
2009-12-03 20:03:00	10002	4	0.85
2009-12-04 08:46:00	10002	12	0.85
2009-12-04 12:20:00	10002	12	0.85
2009-12-04 17:31:00	10002	1	0.85
2009-12-06 15:24:00	10002	1	0.85
2009-12-07 16:40:00	10002	2	0.85
2009-12-11 12:21:00	10002	9	0.85
2009-12-14 12:02:00	10002	12	0.85
2009-12-14 14:12:00	10002	24	0.85
2009-12-21 13:29:00	10002	12	0.85
2009-12-23 12:07:00	10002	1	0.85

Ми можемо використовувати функцію pad (лістинг 4.49) для заповнення наперед ціни. Попереднє значення буде використано для заповнення пробілу для кожного відсутнього значення (лістинг 4.50, табл. 4.23).

```
df_related_time_series3 =
df_related_time_series2.groupby('StockCode').pad()
```

Лістинг 4.49 – Робота функції pad

```
df_related_time_series3.head(20)
```

Лістинг 4.50 – Команда виведення даних

Таблиця 4.23 – Результат роботи команди виведення даних

InvoiceDate	StockCode	Price
2009-12-01	10002	0.85
2009-12-02	10002	0.85
2009-12-03	10002	0.85
2009-12-04	10002	0.85
2009-12-05	10002	0.85
2009-12-06	10002	0.85
2009-12-07	10002	0.85
2009-12-08	10002	0.85
2009-12-09	10002	0.85
2009-12-10	10002	0.85
2009-12-11	10002	0.85
2009-12-12	10002	0.85
2009-12-13	10002	0.85
2009-12-14	10002	0.85
2009-12-15	10002	0.85
2009-12-16	10002	0.85
2009-12-17	10002	0.85
2009-12-18	10002	0.85
2009-12-19	10002	0.85
2009-12-20	10002	0.85

4.6 Огляд створення прогнозу

Наступні лістинги демонструють виклики API, необхідні для створення прогнозу на основі даних, з якими ми працювали. Створення прогнозу за допомогою Amazon Forecast складається з трьох етапів, це створення :

- наборів даних та імпорт даних. Зазвичай цей процес займає 5-10 хвилин;
- предиктора. Цей процес навчає модель за допомогою наданих нами даних. На виконання потрібно 30–60 хвилин;
- прогнозу. Цей процес створює прогноз для певного товару за допомогою предиктора. На виконання також потрібно 30–60 хвилин.

4.6.1 Створення наборів даних та імпорт даних

Першим кроком є створення групи набору прогнозних даних (лістинг 4.51).

```
      session = boto3.Session()
      forecast = session.client(service_name='forecast')
      create_dataset_group_response =
forecast.create_dataset_group(DatasetGroupName=dataset_group_name,
Domain="RETAIL")
      dataset_group_arn = create_dataset_group_response['DatasetGroupArn']
```

Лістинг 4.51 – Створення групи набору даних

Функція create_dataset вимагає кількох параметрів:

– DOMAIN – цей параметр визначає галузь, наприклад роздрібну торгівлю, яка має використовуватися прогнозом;

– DatasetType – для даних часового ряду для цього параметра буде встановлено значення TARGET_TIME_SERIES;

– DatasetName – цей параметр визначає назву набору даних;

– DataFrequency – цей параметр визначає частоту. Для щоденного набору даних це буде D;

– Schema – цей параметр визначає схему набору даних.

Схема набору даних для даних часового ряду наведена у лістингу 4.52.

```
schema ={
   "Attributes":[
      {
         "AttributeName":"timestamp",
         "AttributeType":"timestamp"
      },
      {
         "AttributeName":"item_id",
         "AttributeType":"string"
      },
      {
         "AttributeName":"demand",
         "AttributeType":"float"
      }
   ]
}
```

Лістинг 4.52 – Схема набору даних

Код для створення набору даних наведений у лістингу 4.53.

```
time_series_response=forecast.create_dataset(
                    Domain="RETAIL",
                    DatasetType='TARGET_TIME_SERIES',
                    DatasetName='retail_time_series_data',
                    DataFrequency='D',
                    Schema = schema
)
dataset_arn = time_series_response['DatasetArn']
```

Лістинг 4.53 – Створення набору даних

Тепер, коли набір даних визначено, потрібна робота для імпорту даних (лістинг 4.54).

```
      ds_import_job_response=forecast.create_dataset_import_job
(DatasetImportJobName='retail_import_job',
      DatasetArn=dataset_arn,
      DataSource= data_source,
      TimestampFormat=timestamp_format
      )
```

Лістинг 4.54 – Робота імпорту даних

Звернімо увагу, що data_source – це шлях до даних, які зберігаються в Amazon Simple Storage Service (Amazon S3).

Останнім кроком є додавання набору даних до групи наборів даних (лістинг 4.55).

```
forecast.update_dataset_group(DatasetGroupArn=dataset_group_arn,
DatasetArns=[dataset_arn])
```

Лістинг 4.55 –Додавання набору даних до групи наборів даних

Процес додавання пов’язаних даних або метаданих виконується таким же чином: змінюючи назви, схему та тип набору даних. Хоча ми й підготували ці дані, ми не використовуватимемо їх у предикторі, оскільки на модель не вплинули додаткові дані.

4.6.2 Створення предиктора

Наступним кроком є створення предиктора. Команда create_predictor потребує кількох параметрів:

– PredictorName – цей параметр визначає ім’я, яке ми надаємо предиктору (лістинг 4.56);

```
predictor_name= prefix+'_deeparp_algo'
```

Лістинг 4.56 – Параметр імені предиктора

– AlgorithmArn – цей параметр є шляхом до алгоритму, який ми будемо використовувати. У нашому випадку ми будемо використовувати DeepAR+ (лістинг 4.57);

```
algorithm_arn = 'arn:aws:forecast:::algorithm/Deep_AR_Plus
```

Лістинг 4.57 – Параметр вибору алгоритму

– EvaluationParameters – цей параметр дозволяє вказати кількість і розмір вікон зворотного тестування. Цей параметр контролює розмір і кількість вікон тестування, які створюються з даних (лістинг 4.58);

```
evaluation_parameters= {"NumberOfBacktestWindows": 1,
"BackTestWindowOffset": 30}
```

Лістинг 4.58 – Параметр зворотного тестування

– ForecastHorizon – скільки одиниць прогнозувати (у нашому випадку одиницями є дні, лістинг 4.59);

```
forecast_horizon = 30
```

Лістинг 4.59 – Параметр кількості одиниць для прогнозування

– InputDataConfig – цей параметр визначає дані разом з додатковими днями відпустки (лістинг 4.60);

```
input_data_config = {"DatasetGroupArn": dataset_group_arn,
"SupplementaryFeatures": [ {"Name": "holiday","Value": "UK"} ]}
```

Лістинг 4.60 – Параметр визначення даних

– FeaturizationConfig – цей параметр встановлює частоту, але його також можна використовувати для визначення методів заповнення даних (лістинг 4.61);

```
featurization_config= {"ForecastFrequency": dataset_frequency }
```

Лістинг 4.61 – Параметр частоти

Код для створення предиктора наведений у лістингу 4.62.

```
create_predictor_response=forecast.create_predictor(PredictorName =
predictor_name,
            AlgorithmArn = algorithm_arn,
            ForecastHorizon = forecast_horizon,
            PerformAutoML = False,
            PerformHPO = False,
            EvaluationParameters= evaluation_parameters,
            InputDataConfig = input_data_config,
            FeaturizationConfig = featurization_config
           )
```

Лістинг 4.62 – Створення предиктора

Після створення предиктора можна створити прогноз.

4.6.3 Створення прогнозу

Щоб створити прогноз, скористаємося методом create_forecast (лістинг 4.63).

```
predictor_arn = create_predictor_response['PredictorArn']
create_forecast_response=forecast.create_forecast
(ForecastName=forecast_Name,
PredictorArn=predictor_arn)
```

Лістинг 4.63 – Метод create_forecast

Після створення прогнозу результати можна запитати за допомогою методу query_forecast (лістинг 4.64):

```
forecast_response = forecast_query.query_forecast(
ForecastArn=forecast_arn,
Filters={"item_id":"22423"}
)
```

Лістинг 4.64 – Метод query_forecast

4.7 Очікування завершення створення прогнозу

Тепер слід створити прогноз. Ми можемо перевірити, чи завершено створення прогнозу. Спочатку створимо допоміжний метод для відображення статусу (лістинг 4.65).

```
import sys

class StatusIndicator:

    def __init__(self):
        self.previous_status = None
        self.need_newline = False

    def update( self, status ):
        if self.previous_status != status:
            if self.need_newline:
                sys.stdout.write("\n")
            sys.stdout.write( status + " ")
            self.need_newline = True
            self.previous_status = status
        else:
            # sys.stdout.write(".")
            print('.',end='')
            self.need_newline = True
        sys.stdout.flush()

    def end(self):
        if self.need_newline:
            sys.stdout.write("\n")
```

Лістинг 4.65 – Допоміжний метод для відображення статусу створення прогнозу

Далі створимо екземпляри прогнозу та об’єктів запиту прогнозу (лістинг 4.66).

```
bucket='mlf-lab4-forecastbucket-12sb9sjex9iv'

session = boto3.Session()
forecast = session.client(service_name='forecast')
forecast_query = session.client(service_name='forecastquery')
```

Лістинг 4.66 – Екземпляри та об’єкти запиту прогнозу

Спочатку прочитаємо змінні зі сховища та перевіримо, чи визначено прогноз. Після визначення прогнозу чекатимемо, поки його статус стане активним (лістинг 4.67).

```
print('Waiting for the predictor arn to be available')
while True:
    %store -r
    is_local = "forecast_arn" in locals()
    if is_local: break
    print('.',end='')
    time.sleep(10)

print('Waiting for the predictor to be available')
status_indicator_predictor = StatusIndicator()
while True:
    status =
forecast.describe_predictor(PredictorArn=predictor_arn)['Status']
    status_indicator_predictor.update(status)
    if status in ('ACTIVE', 'CREATE_FAILED'): break
    time.sleep(10)

status_indicator_predictor.end()

print('Waiting for forecast to be available')
status_indicator = StatusIndicator()
while True:
    status =
forecast.describe_forecast(ForecastArn=forecast_arn)['Status']
    status_indicator.update(status)
    if status in ('ACTIVE', 'CREATE_FAILED'): break
    time.sleep(10)

status_indicator.end()

Waiting for the predictor arn to be available
Waiting for the predictor to be available
ACTIVE
Waiting for forecast to be available
ACTIVE
```

Лістинг 4.67 – Перевірка статусу визначення прогнозу

4.8 Використання прогнозу

На цьому етапі прогноз для запиту готовий. Переконаємося, що ми отримуємо дані для наступного тестового коду запасу: 21232 (лістинг 4.68).

```
print()
forecast_response = forecast_query.query_forecast(
ForecastArn=forecast_arn,
Filters={"item_id":"21232"}
)
print(forecast_response)
```

Лістинг 4.68 – Перевірка даних прогнозу

Виведені дані для тестового коду запасу перенесено в Додаток В.

4.8.1 Побудова реальних результатів

Раніше ми розділяли дані та зберігали значення за листопад і грудень. Побудуємо графік цих значень проти прогнозованих значень за той самий період часу.

Почнімо з читання тестових значень назад у DataFrame (лістинг 4.69, табл. 4.24).

```
      actual_df = pd.read_csv(test,
names=['InvoiceDate','StockCode','Quantity'])
      actual_df['InvoiceDate'] = pd.to_datetime(actual_df.InvoiceDate)
      actual_df = actual_df.set_index('InvoiceDate')
      actual_df.head()
```

Лістинг 4.69 – Читання тестових значень

Таблиця 4.24 – Результат читання тестових даних

InvoiceDate	StockCode	Quantity
2010-11-01	21232	0
2010-11-02	21232	60
2010-11-03	21232	130
2010-11-04	21232	255
2010-11-05	21232	24

Переконаємося, що у нас є дані лише для коду запасу 21232 (лістинг 4.70, 4.71, табл. 4.25).

```
stockcode_filter = ['21232']
actual_df = actual_df[actual_df['StockCode'].isin(stockcode_filter)]
```

Лістинг 4.70 – Перевірка даних прогнозу

```
actual_df.head()
```

Лістинг 4.71 – Команда виведення даних

Таблиця 4.25 – Результат роботи команди виведення даних

InvoiceDate	StockCode	Quantity
2010-11-01	21232	0
2010-11-02	21232	60
2010-11-03	21232	130
2010-11-04	21232	255
2010-11-05	21232	24

Можна зробити швидкий графік даних. Пам’ятаємо, що ці дані є тестовими, тому фактичні значення відображаються на графіку. На наступному кроці ми побудуємо графік прогнозованих значень (лістинг 4.72, рис. 4.8).

```
actual_df.Quantity.plot()
```

Лістинг 4.72 – Побудова графіку для тестових даних

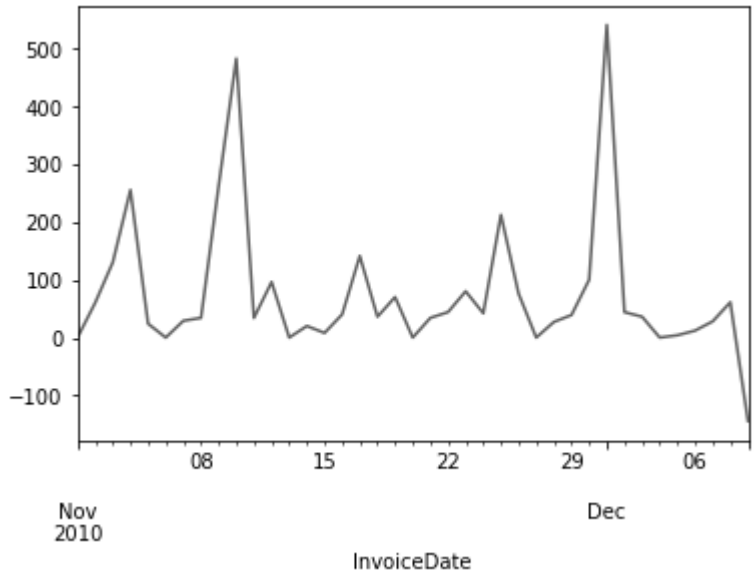

Рисунок 4.8 – Графік тестових даних

4.8.2 Побудова прогнозу

Далі ми повинні перетворити відповідь JSON з предиктора у DataFrame, який можна побудувати.

Почнемо з отримання прогнозів P10 (лістинг 4.73, табл. 4.26).

```
      # Generate DF
      prediction_df_p10 =
pd.DataFrame.from_dict(forecast_response['Forecast']['Predictions']['p10'])
      prediction_df_p10.head()
```

Лістинг 4.73 – Отримання прогнозу для P10

Таблиця 4.26 – Результати прогнозу для P10

	Timestamp	Value
0	2010-11-01T00:00:00	-9.757121
1	2010-11-02T00:00:00	29.454025
2	2010-11-03T00:00:00	36.668056
3	2010-11-04T00:00:00	40.201225
4	2010-11-05T00:00:00	-20.331486

Далі побудуємо прогнози P10 (лістинг 4.74, рис. 4.9).

```
# Plot
prediction_df_p10.plot()
```

Лістинг 4.74 – Побудова графіку прогнозу для P10

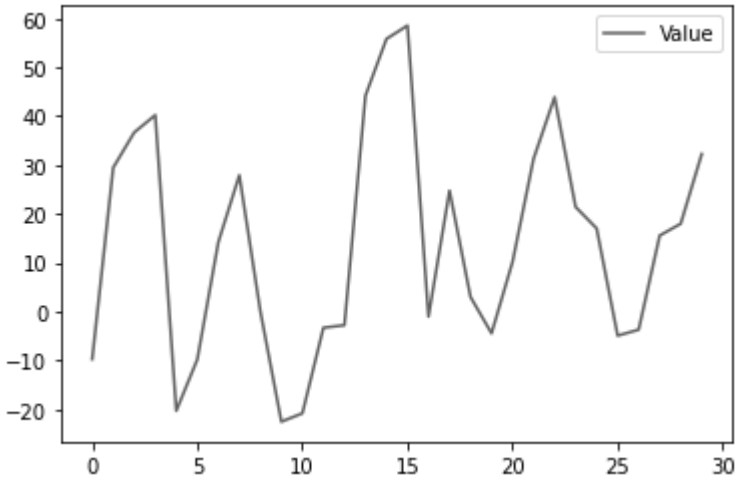

Рисунок 4.9 – Графік прогнозу для P10

Попередній код лише отримував значення P10 і поміщав їх у DataFrame. Тепер виконаємо той же процес для значень P50 і P90 (лістинг 4.75).

```
      prediction_df_p50 =
pd.DataFrame.from_dict(forecast_response['Forecast']['Predictions']['p50'])
      prediction_df_p90 =
pd.DataFrame.from_dict(forecast_response['Forecast']['Predictions']['p90'])
```

Лістинг 4.75 – Отримання прогнозу для P50 і P90

4.8.3 Порівняння прогнозу з реальними результатами

Після того, як ми отримали DataFrames, наступне завдання – побудувати їх разом, щоб визначити найкращу відповідність (лістинг 4.76).

```
      # Start by creating a DataFrame to house the content. Here, Source will be
which DataFrame it came from.
      results_df = pd.DataFrame(columns=['timestamp','value','Source'])

      results_df.head()
```

Лістинг 4.76 – Створення нового DataFrame

Імпортуємо спостережувані значення в DataFrame (лістинг 4.77).

```
      import dateutil.parser
      for index, row in actual_df.iterrows():
      #clean_timestamp = dateutil.parser.parse(index)
      results_df = results_df.append({'timestamp' : index , 'value' :
row['Quantity'], 'Source': 'Actual'} , ignore_index=True)
```

Лістинг 4.77 – Імпорт спостережуваних значень

Тепер подивимося на створений DataFrame (лістинг 4.78, табл. 4.27)

```
      # To show the new DataFrame
      results_df.head()
```

Лістинг 4.78 – Виведення нового DataFrame

Таблиця 4.27 – Результат виведення нового DataFrame

	timestamp	value	Source
0	2010-11-01	0	Actual
1	2010-11-02	60	Actual
2	2010-11-03	130	Actual
3	2010-11-04	255	Actual
4	2010-11-05	24	Actual

Додаємо значення P10, P50 і P90 (лістинг 4.79).

```
      # Now add the P10, P50, and P90 Values
      for index, row in prediction_df_p10.iterrows():
      clean_timestamp = dateutil.parser.parse(row['Timestamp'])
      results_df = results_df.append({'timestamp' : clean_timestamp , 'value' :
row['Value'], 'Source': 'p10'} , ignore_index=True)
      for index, row in prediction_df_p50.iterrows():
      clean_timestamp = dateutil.parser.parse(row['Timestamp'])
      results_df = results_df.append({'timestamp' : clean_timestamp , 'value' :
row['Value'], 'Source': 'p50'} , ignore_index=True)
      for index, row in prediction_df_p90.iterrows():
      clean_timestamp = dateutil.parser.parse(row['Timestamp'])
      results_df = results_df.append({'timestamp' : clean_timestamp , 'value' :
row['Value'], 'Source': 'p90'} , ignore_index=True)
```

Лістинг 4.79 – Прогнози для варіантів P10, P50 і P90

Створюючи опорну точку на даних, ми можемо порівняти фактичні значення P10, P50 і P90 (лістинг 4.80, табл. 4.28).

```
      pivot_df = results_df.pivot(columns='Source', values='value',
index="timestamp")
      pivot_df
```

Лістинг 4.80 – Створення опорної точки на даних

Таблиця 4.28 – Результати реальних і прогнозованих даних

Source	Actual	p10	p50	p90
timestamp				
2010-11-01	0	-9.75712	-8.60563	-7.50653
2010-11-02	60	29.454	32.2054	34.7498
2010-11-03	130	36.6681	39.0319	41.509
2010-11-04	255	40.2012	43.9338	47.7142
2010-11-05	24	-20.3315	-16.2843	-12.5096
2010-11-06	0	-9.8204	-4.21475	3.06106
2010-11-07	29	14.1882	23.8522	32.2038
2010-11-08	34	27.8944	46.3265	69.0688
2010-11-09	262	0.0503235	68.4266	132.382
2010-11-10	482	-22.5631	69.2805	164.465
2010-11-11	34	-20.8479	62.0143	161.216
2010-11-12	96	-3.32487	37.7758	90.8258
2010-11-13	0	-2.81147	11.8609	38.9117
2010-11-14	20	44.1767	57.7247	73.7879
2010-11-15	8	55.8097	72.565	93.5307
2010-11-16	40	58.5392	92.6501	141.247
2010-11-17	141	-1.03651	65.4985	131.274
2010-11-18	36	24.7278	47.3432	115.908
2010-11-19	70	2.96779	31.6962	66.4337
2010-11-20	0	-4.46552	5.71138	29.4722
2010-11-21	34	10.1678	44.7913	81.2667

Продовження таблиці 4.28

Source	Actual	p10	p50	p90
timestamp				
2010-11-22	44	31.2767	51.913	83.0257
2010-11-23	80	43.8765	77.8986	122.102
2010-11-24	42	21.3958	60.5322	104.615
2010-11-25	212	17.011	37.9557	66.2912
2010-11-26	76	-4.94802	12.8958	37.0283
2010-11-27	0	-3.7253	8.00491	27.4109
2010-11-28	27	15.5468	24.598	50.3028
2010-11-29	39	17.9712	32.1774	62.0303
2010-11-30	100	32.2371	43.3186	56.574
2010-12-01	540	NaN	NaN	NaN
2010-12-02	44	NaN	NaN	NaN
2010-12-03	36	NaN	NaN	NaN
2010-12-04	0	NaN	NaN	NaN
2010-12-05	4	NaN	NaN	NaN
2010-12-06	12	NaN	NaN	NaN
2010-12-07	28	NaN	NaN	NaN
2010-12-08	61	NaN	NaN	NaN
2010-12-09	-144	NaN	NaN	NaN

Діаграми легше аналізувати, ніж вихідні значення (лістинг 4.81, рис. 4.10).

```
pivot_df.plot(figsize=(20,10))
```

Лістинг 4.81 – Побудова кінцевого графіку

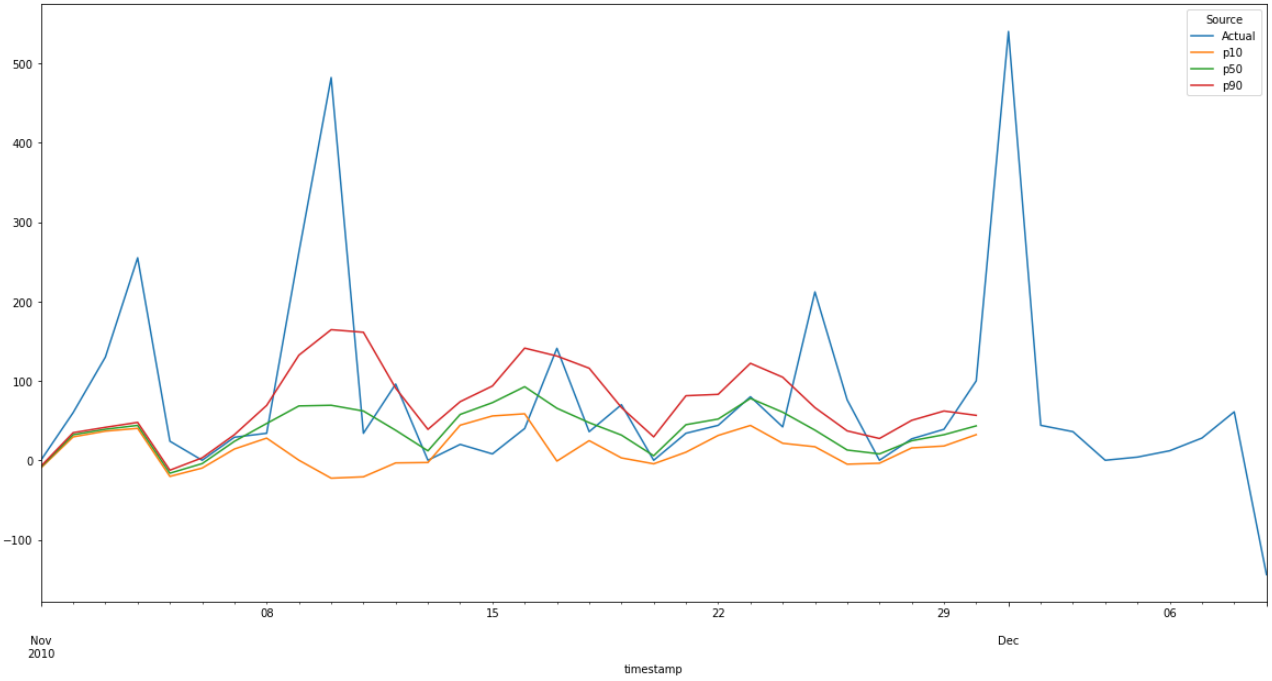

Рисунок 4.10 – Графік порівняння реальних і прогнозованих значень

4.9 Висновки для четвертого розділу

На попередньому графіку можна побачити деяку кореляцію між прогнозованими і реальними значеннями. Кореляція може бути поганою, і для цього може бути кілька причин:

– продажі в основному оптові, але вони включають деякі менші замовлення;

– дані під час дослідження були приховані, а це означає, що цілий сезон не був включений до даних про тренування;

– можливо, не достатньо корисних даних про категорію або стимулювання збуту.

Як і у всіх моделях машинного навчання, результати такі ж хороші, як і дані, які використовуються для навчання моделі. Модель можна покращити за допомогою більшої кількості даних.

У четвертому розділі даної роботи наведені результати тестування створення прогнозу для заготовлених експериментальних даних онлайн-магазину і виконано порівняльний аналіз реальних даних з прогнозованими.

ВИСНОВКИ

Завдання магістерської кваліфікаційної роботи виконане в повному обсязі.

В роботі зроблено огляд використання машинного навчання для створення прогнозів у різних галузях підприємництва, зокрема у роздрібній торгівлі, наведено різні алгоритми прогнозування, можливі виклики в галузі і їх рішення. Проведено аналіз керованих веб-сервісів Amazon, які використовуються для навчання і розгортання моделей прогнозування. Також розроблено експериментальну модель для прогнозу продажів, і потім виконано її тестування на заготовлених даних онлайн-магазину про попит на товари. Виконано порівняльний аналіз отриманих даних прогнозу і реальних даних за потрібний період і виявлено кореляцію між ними. Додатково підкреслено важливість використання прогнозування машинного навчання в умовах конкурентної боротьби бізнесу.

У першому розділі даної роботи виконано постановку задачі використання машинного навчання для прогнозування у різних галузях діяльності людини.

Прогнозування за допомогою машинного навчання – це справді наступний рівень передбачення на основі даних. І немає причин, чому компанія або підприємець повинні упускати можливість посилити аналітику даних безпрецедентними можливостями ML. Однак ця сфера має ряд проблем і випадкових ускладнень, з якими може впоратися лише досвідчений фахівець.

Практично будь-яка компанія чи організація, які мають справу з постійно генерованими даними та потребою адаптуватися до операційних зрушень і змін, можуть використовувати прогнозування часових рядів. Машинне навчання слугує найкращим методом прискорення, дозволяючи краще обробляти дані з багатьох галузей.

Покроковий процес прогнозування часових рядів за допомогою машинного навчання включає в себе етапи: підготовчий; моделювання; тестування; розгортання.

У другому розділі даної роботи проведено аналіз сервісів машинного

навчання Amazon для прогнозування.

Amazon SageMaker – це повністю керована служба машинного навчання. За допомогою SageMaker науковці та розробники даних можуть швидко й легко створювати й навчати моделі машинного навчання, а потім безпосередньо розгортати їх у готовому для виробництва середовищі розміщення. Сервіс надає інтегрований екземпляр блокнота для розробки Jupyter для легкого доступу до джерел даних для дослідження та аналізу, тому користувачу не потрібно керувати серверами. Він також надає загальні алгоритми машинного навчання, які оптимізовані для ефективної роботи з надзвичайно великими даними в розподіленому середовищі.

У машинному навчанні комп'ютер навчається робити прогнози або висновки. Спочатку використовується алгоритм і приклади даних для навчання моделі. Потім інтегрується власна модель у програму, щоб генерувати результати в реальному часі та в масштабі. У виробничому середовищі модель зазвичай навчається на мільйонах прикладів даних і робить висновки за сотні або менше мілісекунд.

Amazon Forecast – це повністю керований сервіс, який використовує статистичні алгоритми та алгоритми машинного навчання для надання високоточних прогнозів часових рядів. На основі тієї ж технології, що використовується для прогнозування часових рядів на Amazon.com, Forecast надає найсучасніші алгоритми для прогнозування майбутніх даних часових рядів на основі історичних даних і не вимагає досвіду машинного навчання.

Користувач може використовувати API, інтерфейс командного рядка AWS, Python SDK і консоль Amazon Forecast, щоб імпортувати набори даних часових рядів, навчати прогнозів і генерувати прогнози.

Amazon Forecast автоматизує більшу частину процесу прогнозування часових рядів, дозволяючи зосередитися на підготовці наборів даних та інтерпретації прогнозів.

У третьому розділі даної роботи створена модель для прогнозування продажів.

Дані часових рядів – це упорядковані дані, які містять елемент часу, що відрізняє їх від звичайних наборів даних.

Деякі задачі роботи з часом включають в себе обробку:

– різних форматів часу;

– відсутніх даних за допомогою зменшення дискретизації, підвищення дискретизації та згладжування даних;

– сезонності, як-от тижневі та річні цикли;

– уникнення поганих кореляцій.

Бібліотека pandas для Python пропонує підтримку даних часових рядів за допомогою функцій, які мають справу з часом.

У Amazon Forecast можна вибрати один із п’яти алгоритмів: ARIMA, DeepAR+, ETS, NPTS або Prophet.

Можна використовувати Amazon Forecast для навчання та використання моделі для даних часових рядів.

Існують конкретні схеми, визначені для таких галузей, як роздрібна торгівля та планування ємності EC2, або можна використовувати користувацьку схему.

Для початку навчання моделі потрібно надати принаймні дані часового ряду, але також можна надати метадані та пов’язані дані, щоб розширити відомості моделі.

Як і в більшості задач контрольованого машинного навчання, дані поділяються на дані тренування та тестування, але цей поділ враховує елемент часу.

Метрики RMSE та wQuantileLoss використовуються для оцінки ефективності прогнозу моделі.

У четвертому розділі даної роботи наведені результати тестування створення прогнозу для заготовлених експериментальних даних онлайн-магазину і порівняння реальних даних з прогнозованими.

На остаточному графіку прогнозу можна побачити деяку кореляцію між прогнозованими і реальними значеннями. Кореляція може бути поганою, і для цього може бути кілька причин:

– продажі в основному оптові, але вони включають деякі менші замовлення;

– дані під час дослідження були приховані, а це означає, що цілий сезон не був включений до даних про тренування;

– можливо, не достатньо корисних даних про категорію або

стимулювання збуту.

Як і у всіх моделях машинного навчання, результати такі ж хороші, як і дані, які використовуються для навчання моделі. Модель можна покращити за допомогою більшої кількості даних.

Результати роботи було апробовано на двадцять шостому міжнародному молодіжному форумі "Радіоелектроніка та молодь у XXI столітті" та подані на публікацію тези доповіді за тематикою кваліфікаційної роботи (Додаток Б), але через воєнний стан в країні, вони будуть опубліковані пізніше.

СПИСОК ЛІТЕРАТУРИ

1. The evolution of forecasting techniques. Genpact. URL: https://www.genpact.com/insight/technical-paper/the-evolution-of-forecasting-techniques-traditional-versus-machine-learning-methods (дата звернення: 17.05.2022).

2. How to Use Machine Learning for Time Series Forecasting. NIX United. URL: https://nix-united.com/blog/find-out-how-to-use-machine-learning-for-time-series-forecasting/ (дата звернення: 04.05.2022).

3. What Is Amazon SageMaker? Amazon Web Services. URL: https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html (дата звернення: 12.05.2022).

4. What Is Amazon Forecast? Amazon Web Services. URL: https://docs.aws.amazon.com/forecast/latest/dg/what-is-forecast.html (дата звернення: 12.05.2022).

5. AWS Academy – Training and Certification. Amazon Web Services. URL: https://aws.amazon.com/training/awsacademy/ (дата звернення: 12.05.2022).

6. Online Retail II Data Set. UCI Machine Learning Repository. URL: https://archive.ics.uci.edu/ml/datasets/Online+Retail+II (дата звернення: 15.05.2022).