КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

Дипломна робота магістра

«Прогнозування ієрархічних часових рядів за допомогою рекурентних нейронних мереж»

«Forecasting hierarchical time series using recurrent neural networks»

Виконав студент 2 курсу магістратури
спеціальності 123 «комп'ютерна інженерія»
_____ Олег ФЕДОРЕНКО

Науковий керівник:
асистент кафедри комп'ютерної інженерії
к.ф.-м.н.
_____ Андрій КОНОВАЛОВ

До захисту допускаю

Завідувач кафедри: кандидат
фіз.-мат. наук, доцент **Юрій БОЙКО**
Протокол засідання кафедри від
"___" _____2023р. №_____

КИЇВ 2023

Зміст

РЕФЕРАТ

Робота присвячена вдосконалення моделі прогнозування ієрархічних часових рядів в рамках підходу «знизу-вгору» (bottom up approach). Проведено аналіз літератури з досліджень прогнозування ієрархічних часових рядів за допомогою нейроних мереж. Побудовані моделі прогнозування та здійснено оптимізацію їх архітектур. Підвищено точність прогнозування використовуючи подання даних у вигляді різниць та використовуючи рекурентні нейроні мережі LSTM та GRU. Здійснено аналіз впливу глибини ієрархії на основі якої здійснюється прогноз. Отримані точності результатів прогнозування на наборі «Hierarchical sales data of an Italian grocery store» які перевищують існуючі в літературі результати в рамках цього ж підходу.

Робота містить 39 сторінок, 11 рисунків, 2 таблиці та 20 посилань.

Ключові слова: ієрархічні часові ряди, модель, прогнозування, рекурентні нейроні мережі, «знизу-вгору», глибина ієрархії, архітектура моделі, LSTM, GRU, MASE.

ВСТУП

Прогнозування є одним із стовпів систем підтримки прийняття рішень у різних областях такі як погода, веб-трафік, продажі та енергетика. Це можна визначити просто як раціональне передбачення майбутніх подій на основі минулих і поточних подій, де всі події представлені у вигляді часових рядів спостережень. Прогнозування часових рядів з обґрунтованою точністю є необхідною, але досить складною проблемою, оскільки вона співвідносить багато факторів з величезної кількості спостережень.

Ієрархічний часовий ряд — це колекція зв'язаних часових рядів, які можна розбити на більш дрібні компоненти. Ці компоненти можуть бути розглянуті як окремі часові ряди зі своїми власними характеристиками, але також пов'язані між собою за допомогою ієрархічних відносин. Проблема прогнозування ієрархічних часових рядів часто виникає в бізнесі та економіці, де величини, що змінюються в часі, потрібно прогнозувати на різних рівнях деталізації.

Існують різні підходи до прогнозування ієрархічних часових рядів, один з них підхід "знизу-вгору" (bottom-up approach), де спочатку прогнозуються окремі часові ряди на нижчих рівнях ієрархії, а потім прогнози узгоджуються для забезпечення дотримання обмежень ієрархії. Ця робота присвячена вдосконаленню моделі прогнозування ієрархічних часових рядів в рамках підходу "знизу-вгору".

Для моделювання довготривалих залежностей в часових рядах доцільно використовувати рекурентні нейроні мережі, що зберігають в пам’яті попередній стан ряду вхідних даних.

Метою випускної кваліфікаційної роботи магістра є розробка моделі для багатокрокового прогнозування ієрархічних часових рядів, використовуючи підхід "знизу-вгору" (bottom-up approach) для агрегації даних, за допомогою рекурентних нейронних мереж.

РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ

1.1 Ієрархічні часові ряди

Ієрархічний часовий ряд — це сукупність часових рядів, організованих в ієрархічну структуру, які можна агрегувати на різних рівнях [1]. Як приклад, продажі складського підрозділу агрегують до продажів підкатегорії продукту, які далі агрегують до категорій продукту [2]. Ієрархічне прогнозування є дуже важливим застосуванням експертних систем для прийняття рішень. Для підтримки прийняття рішень на різних рівнях ієрархії складним завданням є генерація узгоджених прогнозів. Прогнози окремих рядів є узгодженими, коли вони належним чином підсумовуються за рівнями, зберігаючи ієрархічну структуру.

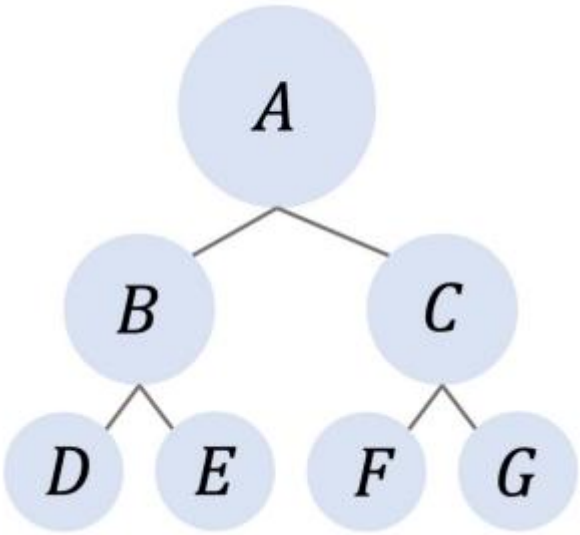

Рис. 1.1 Ієрархічна часова структура

Узгодженість може знадобитися або на між секційному рівні, або на часовому рівні. Наприклад, на між секційному рівні прогнози регіональних продажів повинні підсумовуватися для прогнозів державних продажів, які, у свою чергу, повинні підсумовуватися для прогнозів загальнодержавних продажів. Натомість для часової узгодженості прогнози на денному рівні мають узгоджено підсумовуватися на рівні тижня, потім на рівні місяця тощо. На прикладі Рис 1.1:

$$A_t = B_t + C_t$$

$$B_t = D_t + E_t$$

$$C_t = F_t + G_t$$

Загалом в ієрархічних часових рядах з K > 0 рівнів, рівень 0 це повністю агрегований рівень. Кожен рівень від 1 до K-2 позначає подальшу дезагрегацію вниз до рівня K-1, який є найбільш дезагрегований рівень ієрархічного часового ряду. В ієрархічний часових рядах значення в вищих рівнях є сумою рядів нижче. Нехай $y_t^k = R^{m_k}$ буде вектор спостереження на рівні k = 1…,K-1 і t = 1,…,T, де m_k – це номер ряду на рівні k і $M = \sum_{k=0}^{K-1} m_k$ це загальна кількість рядів в ієрархії. Вектор всіх спостережень у ієрархії:

$$\boldsymbol{y_t} = \begin{pmatrix} y_t^0 \\ \boldsymbol{y_t^1} \\ \vdots \\ \boldsymbol{y_t^{K-1}} \end{pmatrix},$$

де y_t^0 – це спостереження найбільш агрегованого рівня і вектор y_t^{K-1} містить спостереження рівня внизу ієрархії.

Структура ієрархії визначається матрицею підсумовування S, яка визначає обмеження агрегації: $y_t = Sy_t^{K-1}$. Матриця підсумовування S — це матриця, що має елементи, що належать до {0, 1} розміру $M \times m_{K-1}$.

Враховуючи спостереження в момент часу t = 1, … , T і горизонт багатокрокового прогнозування h, метою прогнозування є прогнозування кожної серії на кожному рівні в момент часу t = T+1, … , T+h.

1.2 Методи прогнозування ієрархічних часових рядів

В існуючих дослідженнях прогнози для ієрархічних часових рядів зазвичай складаються у два незалежних етапи. Спочатку складаються прогнози для всіх або деяких часових рядів. Потім прогнози узгоджуються для забезпечення дотримання обмежень ієрархії. Основні методи узгодження прогнозів часових рядів відомі як «знизу вгору» (bottom-up approach), «зверху вниз» (top-down approach), «оптимальне узгодження» (optimal reconciliation) та «мінімізація слідів» (trace minimization) [3].

Підхід «знизу вгору» (bottom-up approach) [4] полягає у створенні прогнозів лише для нижнього рівня ієрархії з подальшим агрегуванням їх до вищих рівнів. Це найпростіший спосіб звірки. Підхід зосереджується на створенні базових прогнозів на крок вперед для кожного ряду на найнижчому рівні $\hat{y}_h^{K-1}$ та агрегуванні їх на верхніх рівнях ієрархії відповідно до матриці підсумовування. Його можна представити так:

$$\tilde{y}_h = S\hat{y}_h^{K-1}$$

де $\tilde{y}_h$ – вектор когерентних прогнозів на h кроків вперед для всіх рядів ієрархії. Перевагою цього підходу є те, що ми безпосередньо прогнозуємо ряд на нижньому рівні, і жодна інформація не втрачається через агрегацію. З іншого боку, ряди нижнього рівня можуть бути шумними та складнішими для моделювання та прогнозування. Недоліком цього підходу є потреба в наявності багатьох часових рядів для прогнозування на найнижчому рівні ієрархії.

Підхід «зверху вниз» (top-down approach) полягає у створенні прогнозів на верхньому рівні ієрархії та розподілі їх на нижчих рівнях. Різним методам «згори вниз» відповідають різні стратегії розподілу.

Дезагрегація прогнозів верхнього рівня досягаються за допомогою пропорції $p = (p_1, \dots, p_{m_{K-1}})^T$, які представляють відносний внесок ряду нижнього рівня в

агрегат верхнього рівня. Два найбільш часто використовуваних низхідних підходи – це середні історичні пропорції (Average Historical Proportions AHP) і пропорції історичних середніх (Proportions of the Historical Averages PHA) [5]. У випадку AHP пропорції розраховуються наступним чином:

$$p_i = \frac{1}{T}\sum_{t=1}^{T}\frac{y_{t,i}^{K-1}}{y_t^0}, i = 1, \cdots, m_{K-1}.$$

У підході PHA пропорції отримують таким чином:

$$p_i = \frac{\sum_{t=1}^{T}\frac{y_{t,i}^{K-1}}{T}}{\sum_{t=1}^{T}\frac{y_t^0}{T}}, i = 1, \ldots, m_{K-1}$$

Для цих двох методів після створення прогнозів нижнього рівня на крок вперед вони агрегуються для створення узгоджених прогнозів для решти рядів ієрархії за допомогою матриці підсумовування. Враховуючи вектор пропорцій p, підходи зверху вниз можна представити у вигляді:

$$\tilde{y}_h = Sp\tilde{y}_h^0$$

де $Sp\tilde{y}_h^0$ – підсумовуюча матриця для пропорцій прогнозів верхнього рівня ієрархії на h кроків вперед

Підхід «зверху вниз», заснований на історичних пропорціях, зазвичай дають менш точні прогнози на нижчих рівнях ієрархії, ніж підхід «знизу вгору», оскільки він не враховує, що ці пропорції можуть змінюватися з часом. Щоб вирішити цю проблему, замість використання статичних пропорцій, як у AHP і PHA, науковці пропонують метод прогнозованої частки (FP) [6], у якому пропорції базуються на прогнозах, а не на історичних даних. Спочатку створюється незалежний базовий прогноз для всіх серій в ієрархії, а потім для кожного рівня, від верхнього до нижнього, обчислюється частка кожного базового прогнозу до сукупності всіх базових прогнозів на цьому рівні. Для ієрархії з K рівнями маємо:

$$p_i = \prod_{k=0}^{K-2} \frac{\hat{y}_{t,i}^{k}}{\hat{\sigma}_{t,i}^{k+1}}, i = 1, \dots, m_{K-1}$$

Де $\hat{y}_{t,i}^{k}$ є базовим прогнозом ряду, який відповідає вузлу, який знаходиться на k рівнях вище вузла i, і $\hat{\sigma}_{t,i}^{k+1}$ є сумою базових прогнозів нижче ряду, який знаходиться на k рівнях вище вузла та безпосередньо контактує з цим рядом.

Підходи TD і BU можуть змішуватися в підході Middle-Out (MO). За допомогою MO ми спочатку вибираємо проміжний рівень ієрархії; потім прогнози обчислюються через BU для верхніх рівнів і TD для нижніх рівнів.

Метод оптимальної комбінації (OC). У дослідженні [3] пропонують новий підхід, який забезпечує оптимальні прогнози, які є кращими, ніж прогнози, створені підходом «зверху вниз» або «знизу вгору». Їхня пропозиція передбачає незалежне прогнозування всіх рядів на всіх рівнях ієрархії, а потім використання моделі лінійної регресії для оптимального поєднання та узгодження цих прогнозів. Їхній підхід використовує узагальнену оцінку методом найменших квадратів, яка вимагає оцінки коваріаційної матриці помилок, які виникають через некогерентність. У недавній статті [7] показують, що цю матрицю неможливо оцінити на практиці, і вони пропонують найсучасніший підхід узгодження прогнозів, який називається мінімальним слідом (MinT), який включає інформацію з повної коваріаційної матриці прогнозних помилок для отримання набір узгоджених прогнозів. MinT мінімізує середню квадратичну помилку узгоджених прогнозів по всій ієрархії з обмеженням неупередженості. Переглянуті прогнози є узгодженими, неупередженими та мають мінімальну дисперсію серед усіх комбінованих прогнозів. Перевага підходу до оптимального узгодження полягає в тому, що він дозволяє встановити кореляції між рядами на кожному рівні, використовуючи всю доступну інформацію в межах ієрархії. Однак це обчислювально дорого у порівнянні з іншими методами, запровадженими досі, оскільки вимагає індивідуального прогнозування часових рядів на всіх рівнях.

1.3 Аналіз існуючих досліджень

1.3.1 Прогнозування обсягів продажів товарів магазину

У статті [9] пропонується підхід машинного навчання для прогнозування ієрархічних часових рядів шляхом створення узгоджених прогнозів.

Ідея даного дослідження полягає в тому, щоб використовувати глибоку нейронну мережу для безпосереднього створення точних і узгоджених прогнозів. Використовується здатність глибокої нейронної мережі отримувати інформацію, яка фіксує структуру ієрархії. Накладається узгодження під час навчання, мінімізуючи налаштовану функцію втрат. У багатьох практичних застосуваннях, окрім даних часових рядів, ієрархічні часові ряди включають пояснювальні змінні, які є корисними для підвищення точності прогнозування. Використовуючи цю додаткову інформацію, підхід пов’язує зв’язок між характеристиками часових рядів, отриманими на будь-якому рівні ієрархії, і пояснювальними змінними в наскрізну нейронну мережу, що забезпечує точні та узгоджені точкові прогнози. Ефективність підходу перевірено на трьох реальних наборах даних, де метод перевершує результати широко поширених методів в ієрархічному прогнозуванні.

Запропонований в дослідженні підхід складає прогнози за допомогою двох сусідніх часових рядів та складається з двох етапів. У першому вибирається найкраща модель прогнозування для агрегованого часового ряду, і нейронна мережа навчається з реальними значеннями навчального набору дворівневого часового ряду. На другому кроці прогнози для агрегованих часових рядів передаються в нейронну мережу для отримання прогнозів для всіх часових рядів нижчого рівня.

Для підвищення точності результату прогнозування у статті застосовувалась перехресна перевірка. Для кожної серії найефективніша модель після фази перехресної перевірки перевіряється за допомогою даних у вибірці, а прогнози отримують рекурсивно протягом періоду поза вибіркою. Наведена вище

процедура вимагає вимірювання помилки прогнозу. Для цього використовувалась середня абсолютна масштабована помилка (MASE).

У статті використовували на наборах даних:

1. Італійський набір даних. В наборі розглядаються дані про продажі, зібрані в італьському продуктовому магазині [10]. Набір складається з 118 щоденних часових рядів, що представляють попит на макарони з 01.01.2014 по 31.12.2018. Окрім одновимірних даних часових рядів, кількість проданих товарів інтегрується за допомогою інформації про наявність або відсутність рекламної акції. Ієрархічна структура набору складається з 3 рівнів: на вершині ієрархії є загальна серія на рівні продажів магазину, яка отримана за допомогою агрегування серії на рівні бренду, що в свою чергу отримана шляхом агрегування індивідуального попиту на рівні товару. Повністю агрегований ряд розбивається на 4 (від B1 до B4), які розбиваються на 42, 45, 10 і 21 відповідно.

Результат прогнозування порівнювався з найвідомішими методами прогнозування в літературі такі як: «згори-вниз» за допомогою мережі випадкового лісу (Random forrest), «знизу-вгору», «метод оптимальної комбінації» і на більшості рівнях ієрархії запропонований метод показав результат з меншим значенням абсолютної масштабованої помилки ніж інші методи.

1.3.2 Прогнозування інтенсивності туризму по регіонах Австралії

У дослідженні [12] застосували підхід використання мереж глибокої довготривалої короткочасної пам’яті (DLSTM) у поєднанні з підходом кодування даниз на взоді у модель та розкодуванням при виході для прогнозування ієрархічних часових рядів. Традиційні методи машинного прогнозування, включаючи глибокі нейроні мережі, вимагають вибірки даних навчання та тестування з одного домену та вимагають великих обчислювальних витрат, через що складність обчислення зростає експоненціально. Тому в дослідженні

використовується трансферне навчання, яке може вирішити цю проблему і знизити витрати на обчислення.

DLSTM може ефективно представляти нелінійний зв’язок між входами та виходами динамічної системи. Ці переваги можуть відповідати вимогам створення стабільних і динамічних «базових» прогнозів, де отримані знання або функції передаються на верхні рівні для отримання узгоджених прогнозів.

Також для прогнозування застосовується підхід Авто-кодувальника (AE). AE — це модель нейронної мережі, яка часто використовується для генерації даних. Він складається з трьох рівнів, а саме вхідного рівня (кодер), прихованого рівня та вихідного рівня (декодера). Процедура навчання AE складається з двох фаз; кодування та декодування. На етапі кодування модель вивчає стисне представлення (або приховані змінні) вхідних даних. Навпаки, на фазі декодування модель реконструює ціль зі стисненого представлення під час фази кодування.

У дослідженні поєднали підхід авто-кодувальника (AE) та мереж глибокої довготривалої короткочасної пам’яті (DLSTM) створили підхід DLSTM-AE. По суті це авто-кодувальник в якому шар кодеру та декодеру замінені на DLSTM. Рівень кодера перетворює дану вхідну послідовність у вектор фіксованої довжини, який діє як зведення вхідної послідовності. Цей контекстний вектор подається як вхідний сигнал до рівня декодера, а кінцевий стан кодера – як початковий стан декодера для прогнозування вихідної послідовності.

Щоб отримати справедливу оцінку, автори статті порівняли ефективність запропонованого підходу з кількома існуючими підходами, використовуючи два приклади, що належать до різних областей. Оцінка базувалася на двох критеріях: точність прогнозування та здатність створювати узгоджені прогнози. Ефективність усіх претендентів оцінювалася за трьома різними показниками

продуктивності в режимі багатокрокового прогнозування. В обох прикладах запропонований підхід дав найвищу точність порівняно з іншими аналогами.

Розроблений підхід застосований на наборі даних внутрішнього туризму Австралії «Australian Domestic Tourism». Цей набір даних щорічно збирався за допомогою комп'ютерних телефонних інтерв'ю з більш ніж 120 000 австралійцями у віці 15 років. Дані класифікуються за кількома цілями подорожі: відпустка, відвідування друзів і родичів, бізнес та інші. Включені спостереження охоплюють період з 1998 по 2006 рік, загалом 36 квартальних спостережень. Для навчання моделі взята вибірка, яка складалась з 12 спостережень, тобто з 1 кварталу 1998 по четвертий квартал 2000 року).

Набір розподілений на 4 рівні, найбільш дезагрегований це дані про візити міст або інший місць у 28 регіонах Австралії, тобто всього 56 значень у ряду. Наступний по ієрархії рівень це візити регіонів і цей ряд складався з 28 значень, далі йде ряд в яких дані агреговані по типу візиту (відпочинок, робоча поїздка, візит родичів або інше). Найбільш агрегований ряд це загальна кількість подорожуючих у Австралії за квартал.

1.4 Рекурентні нейроні мережи

Рекурентна нейронна мережа LSTM (мережі довгої короткотривалої пам'яті) є особливим типом штучної нейронної мережі, який використовується для обробки послідовних даних, таких як часові ряди. У відміну від нейронних мереж прямого розповсюдження, які використовуються для незалежних точок даних, LSTM має концепцію пам'яті, що дозволяє зберігати стани або інформацію про попередні вхідні дані для створення наступних послідовностей виведення.

LSTM (мережі довгої короткотривалої пам'яті) [14] — це вид рекурентних нейронних мереж. LSTM включає в себе спеціальний механізм пам'яті, який дозволяє зберігати довгострокову залежність між елементами послідовності та уникнути проблеми зниклої градієнтної проблеми. Цей механізм складається зі

спеціальних блоків, що називаються "клітинами" (cell), які містять деякі розрахунки та контрольний механізм, що вирішує, які значення повинні бути збережені в пам'яті, а які - викинуті. Це можна уявити як конвеєр, через який інформація просто "тече", залишаючись незмінною.

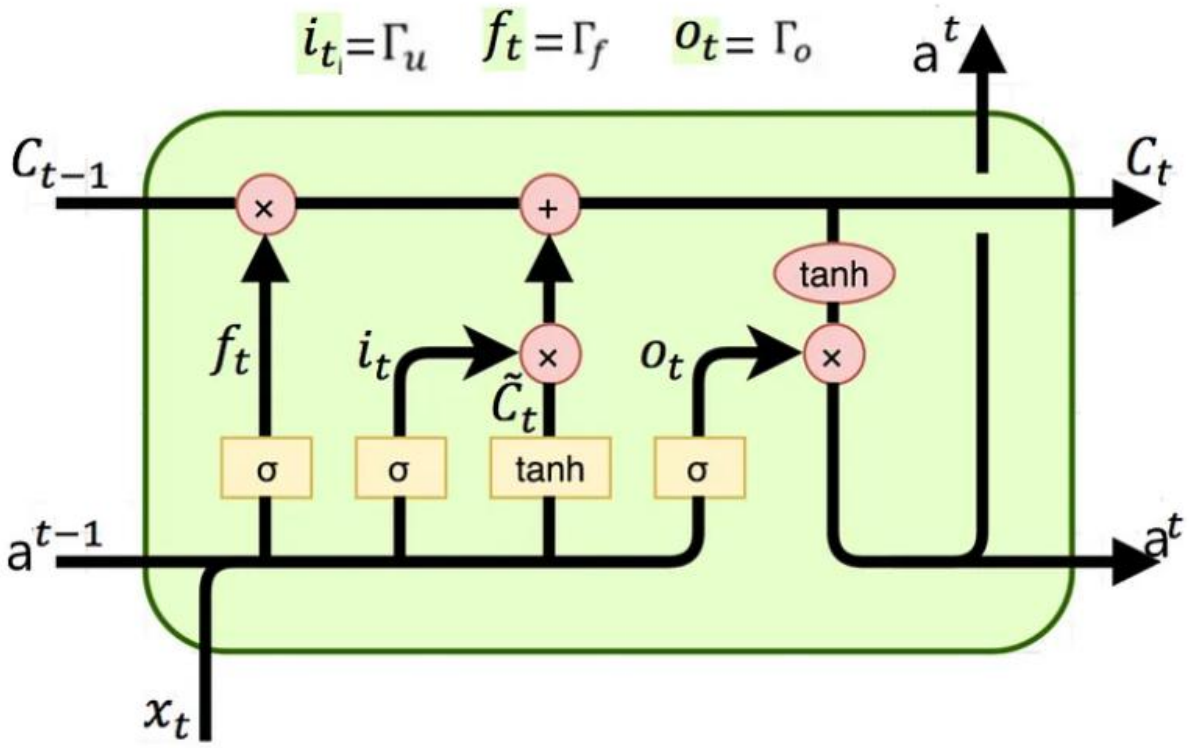

Рис. 1.4.1 Комірка LSTM

Як показано на рисунку 1.4.1, комірка пам'яті LSTM містить три вентилі, а саме:

1. Вентиль "забування" (forget gate): контролює, які значення мають бути видалені з пам'яті комірки. Цей вентиль використовує сигмоїдну функцію для вирахування ймовірності того, що значення будуть видалені з пам'яті.

2. Вентиль "вхід" (input gate): контролює, які нові значення мають бути додані до пам'яті комірки. Цей вентиль також використовує сигмоїдну функцію для вирахування ймовірності того, що нове значення буде додане до пам'яті, а

також використовує тангенціальну функцію для вирахування значення, яке потрібно додати до пам'яті.

3. Вентиль "вихід" (output gate): контролює, які значення мають бути виведені з комірки. Цей вентиль також використовує сигмоїдну функцію для вирахування ймовірності того, що значення будуть виведені з комірки, а також використовує тангенціальну функцію для обчислення значення, яке має бути виведене з комірки.

Комбінація цих трьох вентилів дозволяє комірці LSTM контролювати, які значення зберігати та які - викинути, тим самим забезпечуючи роботу з послідовностями даних та збереження довгострокових залежностей.

Модель GRU (Gated Recurrent Unit) - це інший тип рекурентної нейронної мережі, яка використовується для обробки послідовних даних. Вона є спрощеною версією LSTM і має менше вентилів, а саме два: вхідний вентиль і вихідний вентиль. GRU відрізняється від LSTM тим, що має один загальний вентиль замість окремих вентилів для контролю запису, стирання та модуляції даних. Архітектуру комірки GRU наведено на рисунку 1.4.2

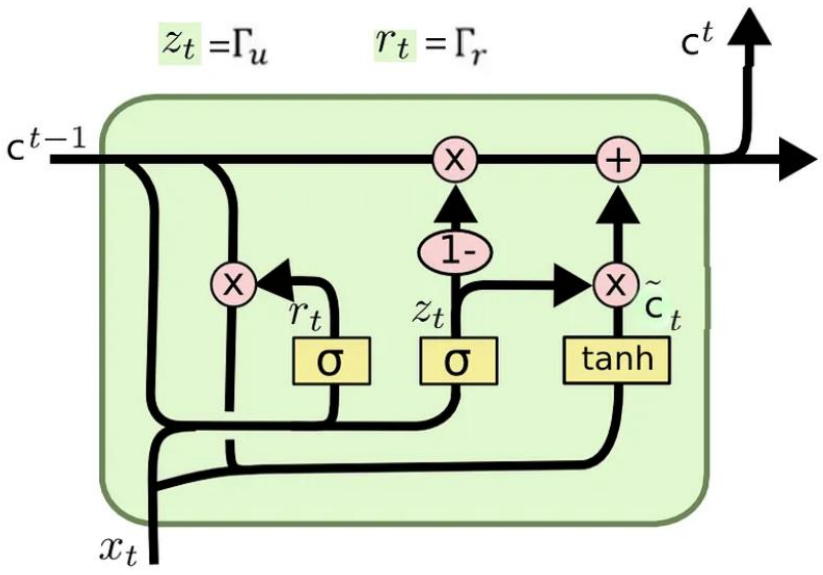

Рисунок 1.4.2 Комірка GRU

У GRU також є стан, який може зберігати інформацію про попередні вхідні дані, але він є менш складним в порівнянні зі станом комірки LSTM. GRU дозволяє контролювати, яка інформація передається через модель, і які дані зберігаються у стані, що дозволяє краще управляти залежностями між даними.

GRU також може бути використана для вирішення різних завдань, таких як прогнозування часових рядів, машинний переклад, розпізнавання мови, і багато інших. Вона є популярним вибором серед різноманітних архітектур рекурентних нейронних мереж і може бути використана в різних сферах дослідження та застосування машинного навчання.

1.5 Постановка задач дослідження

Виходячи з усього вищесказаного прогнозування часових рядів є важливою задачею в багатьох галазюх та використання рекурентних мереж є актуальним напрямом досліджень в галузі машинного навчання, було сформовано відповідну мету:

Метою дипломної роботи магістра є побудова моделі багатокрокового прогнозування ієрархічних часових рядів в рамках підходу до агрегації даних «знизу-вгору» (bottom-up approach) за допомогою рекурентних нейронних мереж на прикладі набору даних «Hierarchical sales data of an Italian grocery store».

Для досягнення сформульованої мети поставленні такі задачі:

1. Побудова моделей багатокрокового прогнозування ієрархічних часових рядів за допомогою рекурентних нейронних мереж LSTM і GRU.
2. Підбір оптимальних архітектур мереж LSTM і GRU та гіперпараметрів їх навчання з метою максимізації точності прогнозування.
3. Оцінювання точності прогнозування оптимальних моделей на тестовій вибірці даних.
4. Порівняння отриманих результатів з літературними даними.

РОЗДІЛ 2. МЕТОДИКА ДОСЛІДЖЕНЬ

2.1 Програмний інструментарій

2.1.1 Бібліотеки Python

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією [15].

Під час побудови моделі для створення прогнозів на основі ієрархічних часових рядів, були застосовані наступні бібліотеки:

1. Numpy – це бібліотека мови програмування Python, що надає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами. Використовувалась для роботи з базою даних у вигляді таблиці.

2. Pandas – це бібліотека з ліцензією BSD з відкритим кодом, побудована на основі Numpy, що забезпечує високопродуктивні, прості у використанні структури даних та засоби аналізу даних для мови програмування Python. [16] Використовувалась для завантаження набору даних та його перетворення.

3. Keras – відкрита нейромережева бібліотека, написана мовою Python. Спроектована для уможливлення швидких експерементів з мережами глибинного навчання, її зосереджено на тому, щоб вона була зручною в користуванні модульною та розширюваною.

4. Scikit-learn (sklearn) – це відкрита бібліотека машинного навчання для мови програмування Python. Вона містить багато класичних алгоритмів машинного навчання, таких як класифікація, регресія, кластеризація, а також інші інструменти для підготовки даних та оцінки результатів. Використовувалась для нормалізації даних перед навчанням.

2.1.2 Google Colaboratory

Colaboratory, або скорочено Colab - це продукт компанії Google Research. Colab дозволяє будь-кому писати та виконувати довільний код мови python через браузер, і особливо добре підходить для машинного навчання та аналізу даних. Більш технічно, Colab - це розміщена послуга ноутбуків Jupyter, яка не потребує налаштування, одночасно надаючи безкоштовний доступ до обчислювальних ресурсів, включаючи графічні процесори. У Colaboratory встановлені Tensorflow більшість необхідні для роботи Python-бібліотеки [17].

Середовище Google Colab працює на базі Ubuntu 17.10 64 біт та використовує процесор Intel Xeon з одним ядром 2,3 ГГц (з hyperthreading). Найбільш важливою особливістю, яка відрізняє Colab від інших безкоштовних хмарних сервісів, є те, що GC надає доступ до швидких процесорів GPU Tesla K80 з 12 ГБ оперативної пам’яті, 2496 ядер CUDA при 560 МГц [18].

2.2 Підготовка даних

Після підготовки середовища та завантаження набору даних продажів італійського магазину «Hierarchical sales data of an Italian grocery store» [10], описаний в пункті 1.3.1, розраховуються ризниці значень часових рядів. Тобто створюється новий набір де кожне значення набору представляє собою різницю між значеннями часового ряду та попереднє значення цього ж часового ряду. Це може допомогти моделі виявити складні закономірності та залежності між спостереженнями, що можуть бути корисні для точного прогнозування майбутніх значень.

Часовий індекс значення	Вхідні дані	Дані у вигляді різниць
2014-01-02	7	–
2014-01-03	5	-2
2014-01-04	9	4
2014-01-05	5	-4
2014-01-06	1	-4
2014-01-07	12	11
2014-01-08	2	-10

Таблиця 2.1 Перетворення для часового ряду товару QTY_B1_1 з набору [10] у часовий ряд у вигляді різниць.

Після задається на яку кількість кроків num_future вперед буде відбуватись прогнозування ряду (горизонт прогнозування) та num_past кількість кроків на основі яких буде відбуватись прогнозування на кожному кроці (глибина ієрархії). В залежності від встановлених значень створюються два списки X та Y. Список X містить підмножини довжиною num_past, які будуть використовуватись для прогнозування на кожному кроці, тоді як список Y містить відповідні майбутні значення ряду довжиною num_future. Ці підмножини створюються шляхом переміщення вікна довжиною num_past та num_future через ряд. Якщо кінець вікна num_future виходить за межі ряду, то створення списків X та Y припиняється.

Після прогнозування виконується зворотня обробка отриманних результатів. Спрогнозований список Y_pred, в якому кожне значення є масивом довжиною num_past у вигляді різниць додається до відповідного істиного значення продажу товару в цьому часовому ряду

Для подальшої роботи з набором даних було використано поділ на три вибірки: навчальну, валідаційну та тестову. Навчальний набір містив дані з 2014 по 2017 роки, що складалося з 1080 спостережень. Валідаційна вибірка включала

дані за 2017 рік і складалася з 362 спостережень, тоді як тестова вибірка містила дані за 2018 рік і складалася з 319 спостережень.

Після розбиття набору на підвибірки було проведене масштабування даних за допомогою методу StandardScaler, бібліотеки Sklearn. Цей метод використовується для центрування та нормування ознак в наборі даних. Ознаки центруються за допомогою віднімання середнього значення кожної ознаки зі значень відповідних ознак. Далі, для нормування значень, кожне значення ознаки ділиться на стандартне відхилення всіх значень ознаки. Як результат, всі ознаки мають середнє значення 0 та стандартне відхилення 1.

Масштабування проведено для пришвидшення навчання моделей прогнозування. Функція масштабування вчиться на навчальній вибірці, після чого застосовється на валідаційній на тестовій виборках.

Після прогнозування на отриманих результатах проводилась зворотнє масштабування за допомогою тієї самої наваної функції, тобто дані перетворювались до початкового вигляду.

Для агрегування часових рядів на наступних рівнях ієрархії (рівня прдажу брендів та рівня продажів магазину) спрогнозовані спостереження були об'єднані відповідно до їх принадлежності до брендів. Після цього для кожного часового кроку була порахована сума товарів, що належать до відповідного бренду.

2.3 Оптимізація моделей на основі рекурентних нейроних мереж

Моделі створювались за допомогою класу Sequential моделей бібліотеки Keras, який дозволяє створювати послідовні нейронні мережі. Він дозволяє додавати шари послідовно , в порядку, в якому вони повинні бути виконані.

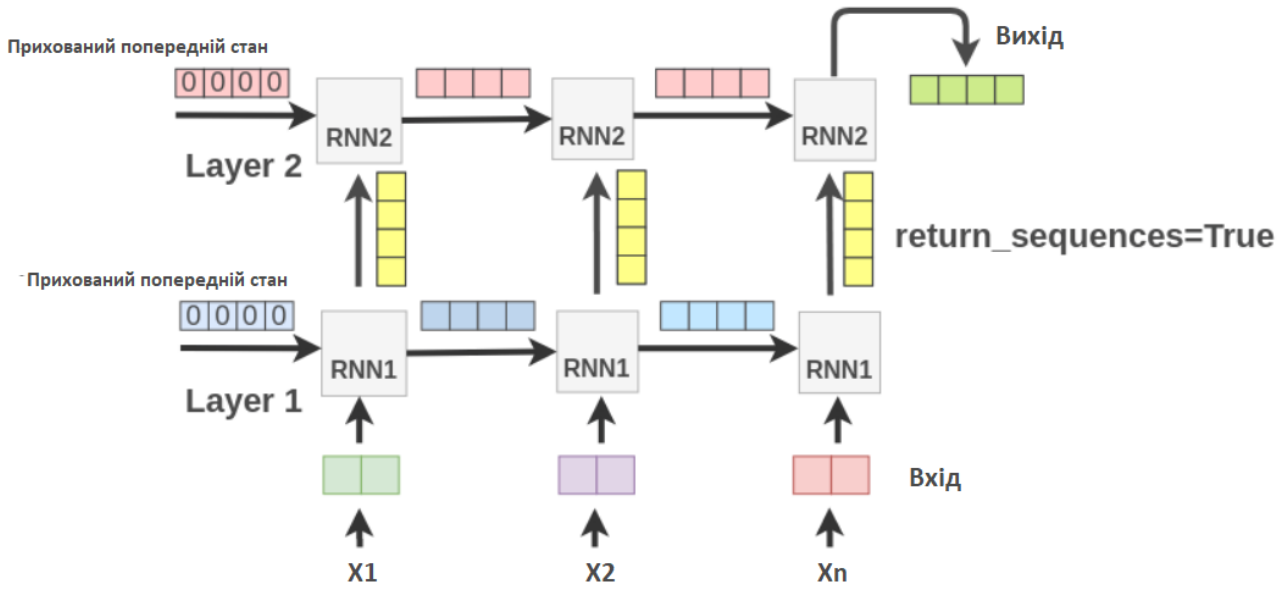

Рисунок 2.3.1 Приклад архітектури моделі з декількома зв’язаними шарами на основі рекурентних нейроних мереж

В моделі використовувались шари класу Layers бібліотеки Keras:

1. Dense шари з відповідними рекурентних нейроних мереж LSTM чи GRU.
2. Dropout – шари з ймовірністю випадкового відключення для регуляризації моделі та запобігання перенавчання.
3. RepeatVector – шар для повторення вектора на виході останнього шару нейронної мережі n_future разів, де n_future - кількість кроків у майбутньому, які необхідно передбачити.
4. TimeDistributed – шар для застосування Dense шару до кожної часової кроку на виході останнього шару нейронної мережі.

Для навчання моделі використовувалась функція втрат «mse» (Mean Squared Error) – середньоквадратична помилка та оптимізатор Adam – адаптивний оптимізатор, який використовується в процесі навчання глибоких нейронних мереж.

Для запобіганню перенавчання моделі використовувалась функція ранньої зупинки, яка спостерігала за валідаційною вибіркою при навчанні і зупиняла навчання, якщо протягом останніх 100 ітерацій функція втрат не зменшувалась, та відновлювала ваги моделі відповідно до ітерації, де функція втрат була найменшою.

Для перевірки та порівняння отриманих результатів, розраховувалась метрика MASE (Mean Absolute Squared Error) – це метрика якості прогнозів часових рядів [19]. Вона використовується для оцінки точності прогнозів, порівнюючи їх зі значеннями ряду вихідних даних.

MASE обчислюється як відношення середнього абсолютного значення помилки прогнозу до середнього абсолютного значення помилки наївного прогнозу Формула MASE виглядає наступним чином:

$$MASE = \frac{\frac{1}{n}\sum(|y_t - y'_t|)}{\frac{1}{n-1}\sum(|y_t - y_{t-1}|)},$$

де n – кількість спостережень, y_t– спостереження в момент часу t, y'_t– відповідний прогноз в момент часу t, y_{t-1}– спостереження в попередній момент часу t-1.

Отримане значення MASE вказує на те, наскільки краще прогнозується часовий ряд в порівнянні з наївним прогнозом. Значення менше 1 вказують на кращі прогнози, ніж наївний прогноз, а значення більше 1 вказують на гірші прогнози, ніж наївний прогноз. MASE дорівнює 1 для ідеального прогнозу.

Оптимізація моделей виконувалась через:

1. Підбір кількості шарів Dense та Dropout.
2. Підбір кількості нейронів у шарах які містять рекурентні нейроні мережі.

3. Встановленням коефіцієнта Dropout у шарах (відсотку випадково відключених шарів).

Оптимізація проводилась за метрикою MASE на валідаційній виборці, після цього були обрані найкращі моделі і отриманий результат на тестовій виборці.

Код описаної методики наведено в **Додатку А.**

РОЗДІЛ 3. АНАЛІЗ РЕЗУЛЬТАТУ

3.1 Результати побудови моделей на основі рекурентних нейроних мереж

В результаті оптимізації гіперпараметрів архітектури моделей побудована архітектура для моделі з використанням рекурентних нейроних мереж LSTM, зображена на рисунку 3.1.1. Перший шар LSTM складався з 15 нейронів, другий – 10, третій – 7 та четвертий з 5. Шари Dropout використовувались з коефіцієнтом 0.3.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 35, 15)            8040

 dropout (Dropout)           (None, 35, 15)            0

 lstm_1 (LSTM)               (None, 35, 10)            1040

 dropout_1 (Dropout)         (None, 35, 10)            0

 lstm_2 (LSTM)               (None, 7)                 504

 dropout_2 (Dropout)         (None, 7)                 0

 repeat_vector (RepeatVector  (None, 7, 7)             0
 )

 lstm_3 (LSTM)               (None, 7, 5)              260

 dropout_3 (Dropout)         (None, 7, 5)              0

 time_distributed (TimeDistr  (None, 7, 118)           708
 ibuted)

=================================================================
Total params: 10,552
Trainable params: 10,552
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.1.1 Архітектура моделі з використанням 4 шарів мереж LSTM, 4 шарів Dropout, шару RepeatVector та шару TimeDistributed

Для моделей з використанням мереж GRU побудована архітектура моделі з найкращим результатом на валідаційній виборці зображена на рисунку 3.1.2. Перший шар GRU складався з 30 нейронів, другий – 20, третій – 15 та четвертий з 20. Шари Dropout використовувались з коефіцієнтом 0.3.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 gru_4 (GRU)                  (None, 35, 30)            13500

 dropout_8 (Dropout)          (None, 35, 30)            0

 gru_5 (GRU)                  (None, 35, 20)            3120

 dropout_9 (Dropout)          (None, 35, 20)            0

 gru_6 (GRU)                  (None, 15)                1665

 dropout_10 (Dropout)         (None, 15)                0

 repeat_vector_2 (RepeatVect  (None, 7, 15)             0
 or)

 gru_7 (GRU)                  (None, 7, 20)             2220

 dropout_11 (Dropout)         (None, 7, 20)             0

 time_distributed_2 (TimeDis  (None, 7, 118)            2478
 tributed)

=================================================================
Total params: 22,983
Trainable params: 22,983
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.1.2 Архітектура моделі з використанням 4 шарів мереж GRU, 4 шарів Dropout, шару RepeatVector та шару TimeDistributed

3.2 Представлення результатів прогнозування та їх аналіз

В дослідженні продоводився аналіз впливу глибини ієрархії на отриманий результат, які порівнювались за допомогою метрики MASE. Прогноз проводився на тиждень вперед. Результати метрики MASE прогнозування на рівні продажів брендів на основі моделі LSTM наведені на рисунку 3.2.1.

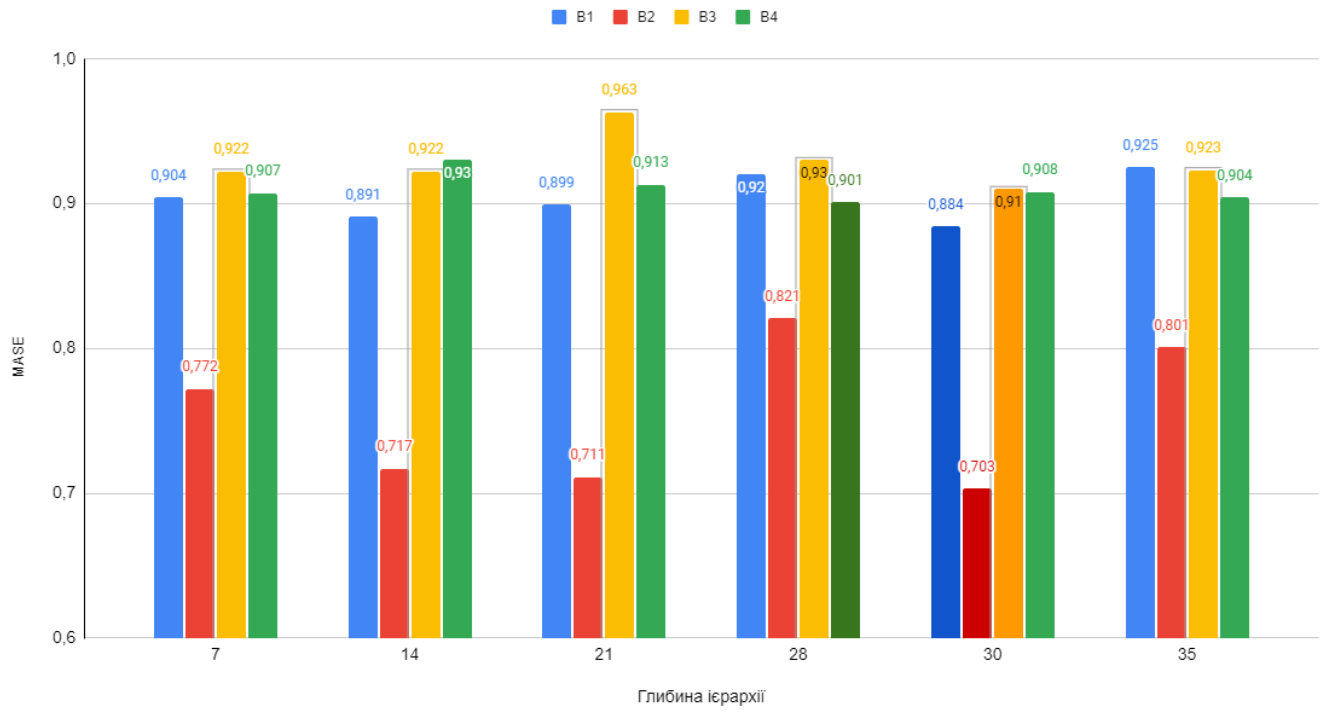

Рисунок 3.2.1 Результати MASE на рівні продажів брендів з використанням моделі LSTM у відповідності до глибини ієрархії на основі якої робиться прогноз. B1, B2, B3, B4 – це позначки для брендів до яких відносяться товари.

Відповідно до результатів представлених на рисунку 3.2.1 для бренду B1 найкращий результат метрики MASE 0,884 досягається при кількості попередніх спостережень – 30. Для бренду B2 – MASE 0,703 при 30 спостережень. Для B3 – MASE 0,910 при 30 минулих спостережень. Для B4 – MASE 0,901 при 28 минулих спостережень.

На рисунку 3.2.2 представлені результати залежності MASE від кількості попередніх спостережень для моделей на основі GRU.

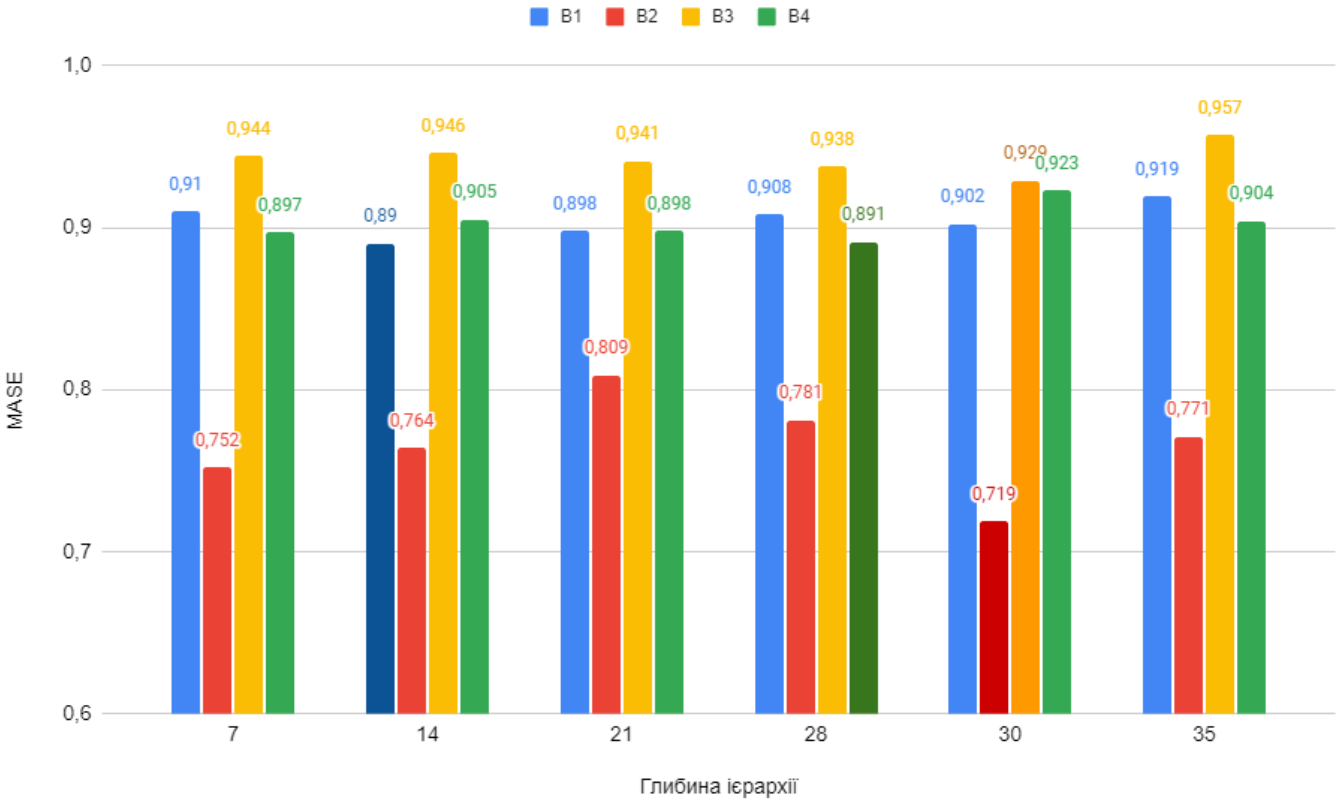

Рисунок 3.2.2 Результати прогнозування на рівні продажів брендів з використанням моделі GRU у відповідності до глибини ієрархії на основі яких робиться прогноз. B1,B2,B3,B4 – позначки брендів, до яких відносяться відповідні товари.

Відповідно до результатів представлених на рисунку 3.2.2 для бренду B1 найкращий результат метрики MASE 0,89 досягається при кількості попередніх спостережень – 14. Для бренду B2 – MASE 0,719 при 30 спостережень. Для B3 – MASE 0,929 при 30 минулих спостережень. Для B4 – MASE 0,891 при 28 минулих спостережень.

Порівнюючі результати для рівня продажу брендів моделей з LSTM та GRU найкращий результат для B1 – у моделі з використанням LSTM при 30 кроках – 0.884, для B2 – у моделі з використанням LSTM при 30 кроках – 0.703, B3 – у

моделі з використанням LSTM при 30 кроках – 0.910, B4 – у моделі з використанням GRU при 28 кроках – 0.891.

Результати метрики MASE прогнозів для рівня продажів магазину відповідно до глибини ієрархії для моделей LSTM та GRU представлеі на рисунку 3.2.3.

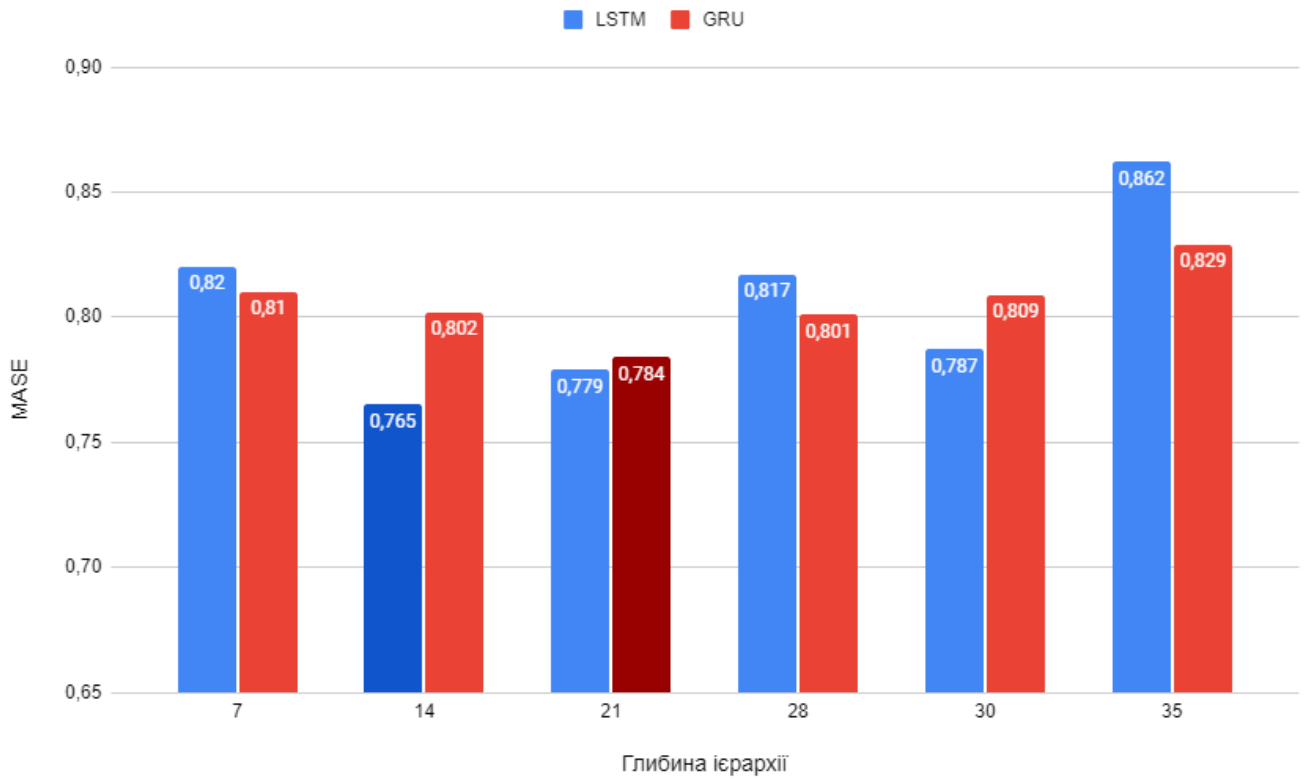

Рисунок 3.2.3 Результати прогнозування на рівні продажів магазину з використанням моделей LSTM та GRU у відповідності до кількості спостережень на основі якої робиться прогноз, де горизонтальна вісь відповідає за глибину ієрархії, а вертикальна метрика MASE.

Відповідно до результатів представлених на рисунку 3.2.3 моделі на основі LSTM найкращий результат метрики MASE при 14 минулих спостережень – 0.765, для моделі на базі GRU при 21 спостереженні – 0.784. Найменша метрика MASE

розрахована з використанням моделі LSTM при 14 попередніх спопстережень – 0.765.

Порівняння середніх метрик MASE розрахованих по всім товарам для моделей LSTM та GRU наведено на рисунку 3.2.4.

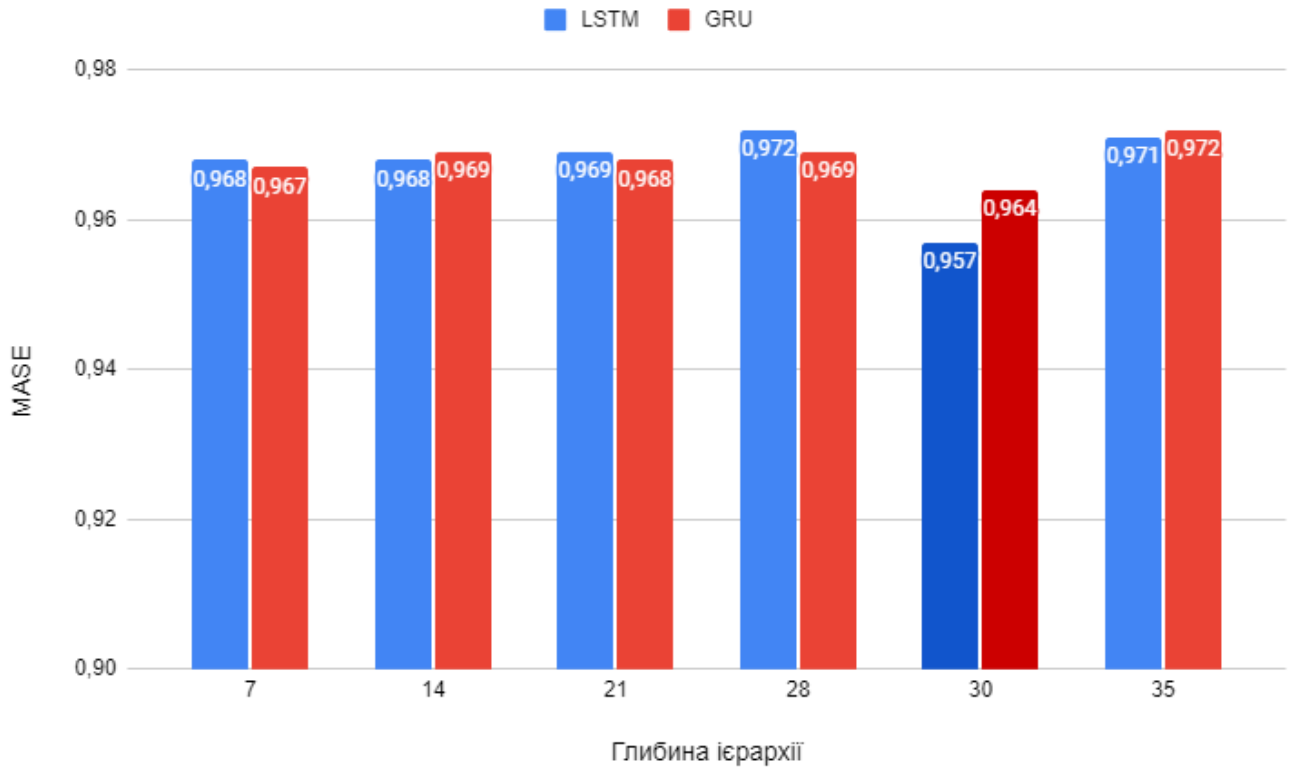

Рисунок 3.2.4 Порівняння середньої метрики MASE розрахованої для всіх товарів у відповідності до глибини ієрархії на основі якої проводився прогноз для моделей на основі LSTM та GRU.

Відповідно до рисунку 3.2.4 найкраща середня метрика MASE спостерігається при кількості попередніх спостережень 30 для обох моделей і найкраща з них у моделі з використанням LSTM – 0.957.

Аналізуючі наведені вище результати моделі на основі рекурентної нейроної мережі LSTM дала кращі результати ніж модель на основі GRU. Серед моделей

LSTM у відповідності до кількості минулих спостережень на основі яких утворюється проноз для рівня продажу брендів та середнього значення по товарам метрики MASE, кращий результат отриманий при 30 спостереженнях, для рівня продажу магазину для 14 минулих спостережень.

Для подальшого порівння з існуючими в літературі результатами на цьому ж наборі даних розглядається модель на основі LSTM з глибиною ієрархії – 30 та горизонтом прогнозування – 7, оскільки в більшості критеріїв вона дала найменшу метрику MASE.

В таблиці 3.1 наведено порівняння реальних значень продажу товару QTY_B1_4 з результатом прогнозування отриманої моделі.

Часовий індекс спостереження	Значення з набору	Спрогнозоване значення
2018-02-05	2	1.94
2018-02-06	2	2.41
2018-02-07	1	2.26
2018-02-08	0	0.03
2018-02-09	2	2.7
2018-02-10	3	0.8
2018-02-11	0	1.09

Таблиця 3.1 Порівнянн результатів прогнозування з реальними значеннями набору

3.3 Порівняння результатів з існуючим в літературі

Для порівняння результатів наведених в підрозідлі 3.2 на наборі «Hierarchical sales data of an Italian grocery store» з представленими у дослідженні [10] в рамках підходу до агрегації даних «знизу-вгору» (bottom-up approach), який проводився на основі 30 попередніх спостережень на 7 кроків вперед, який

розрахований за допомогою підходу випадкового лісу (random forest) в поєднані з алгоритмом градієнтного бустингу ансамблю дерев рішень (XGBoost). На рисунку 3.3.1 наведене порівння метрики MASE для дослідження [10] в порівнянні з результатими цієї роботи.

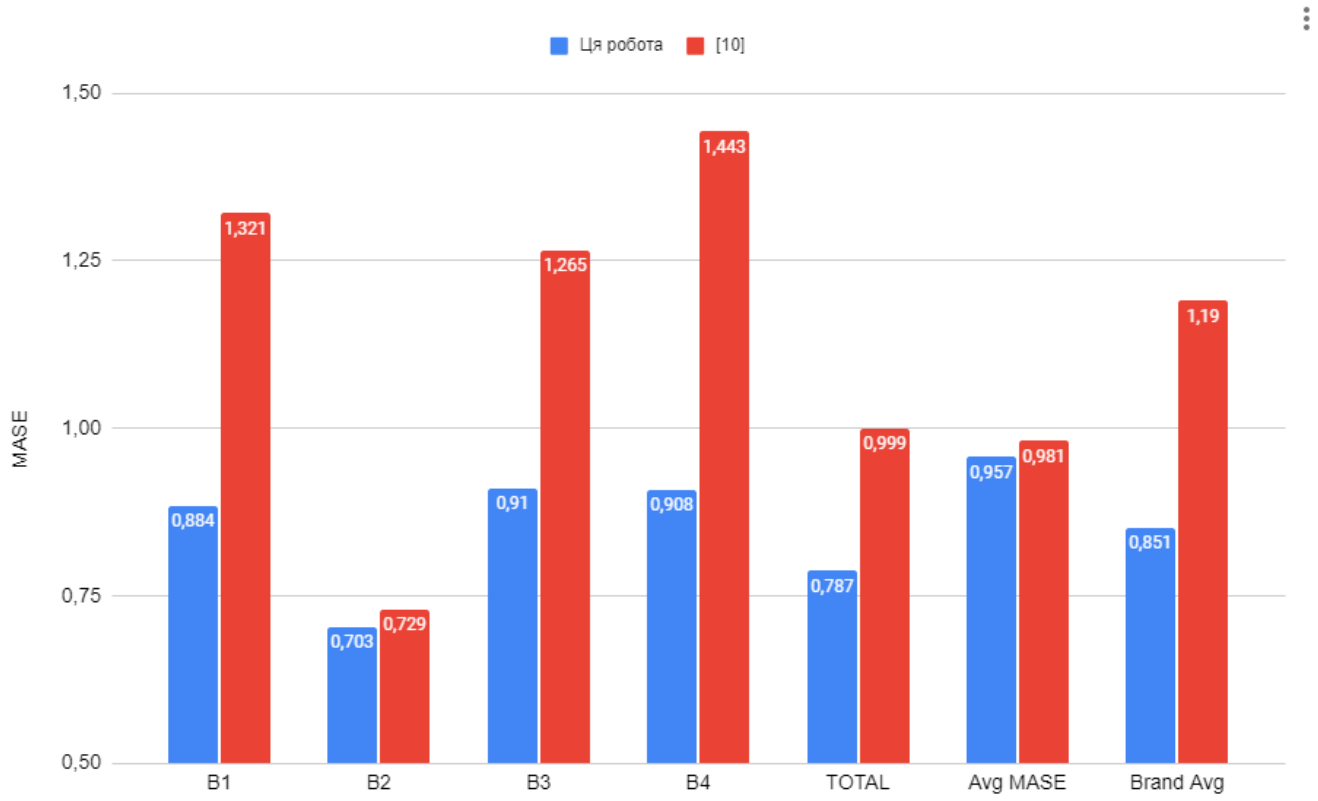

Рисунок 3.3.1 Результат порівння метрик MASE отриманої в цій роботі з використанням моделі на основі LSTM та кількості попередніх досліджень 30 з дослідженням [10], де B1, B2, B3, B4 – результат для відповідних об’ємів продажів брендів; TOTAL – метрика MASE для рівня об’ємів продажів магазину; Avg MASE – середнє значення MASE для всіх товарів; Brand Avg – середнє MASE для рівнів брендів.

Як бачимо з рисунку 3.3.1 розроблена в цьому дослідженні модель прогнозування ієрархічних часових рядів в рамках підходу «знизу-вгору», з використанням рекурентних нейроних мереж LSTM та з поданням даних у вигляді

різниць значно краще по всім рівням ніж наведені в існуючому дослідженні [10]. Для рівня бренду B1 метрика MASE менша на 0.437, для B2 на 0.26, для B3 на 0.355, для B4 на 0.535. Для рівня продажу всього магазину метрика MASE менша 0.212 від існуючого дослідження. Середнє значення MASE для всіх товарів менша на 0.024, а середнє значення для рівня брендів менше на 0.339.

Виходячи з наведеного вище мона стверджувати, що використання рекурентних нейроніï мереж LSTM та подання даних у вигляді різниць, в рамках підходу «знизу-вгору», значно покращує результат багатокрокового прогнозування ієрархічних часових рядів

ВИСНОВОК

В результаті виконання дипломної роботи магістра було зроблено:

1. Побудовані моделі для прогнозування ієрархічних часових рядів з використанням рекурентних нейроних мереж LSTM та GRU.

2. Проведена оптимізація параметрів за допомогою підбору гіперпараметрів архітектури рекурентних нейроних мереж.

3. Проведено порівняльний аналіз отриманих результатів на тестовій виборці та встановлено модель, яка дає найкращий результат – модель на основі LSTM з глибиною ієрархії на основі яких здійснюється прогнозування часових рядів на 7 днів вперед рівною 30.

4. Проведено порівняння отриманих результатів з результатми отриманими в дослідженні [10], яке показало, що по всім критеріям модель з поданням даних у вигляді різниць та використанням рекурентних нейроних мереж дала кращий результат прогнозування ієрархічних часових рядів в рамках підходу «знизу-вгору» на наборі даних продажів італійського продуктового магазину.

СПИСОК ЛІТЕРАТУРИ

1. Hyndman, R., Lee, A., Wang, E., Wickramasuriya, S., & Wang, M. E. Package ‘hts’, 2022.

2. FRANSES, Philip Hans; LEGERSTEE, Rianne. Combining SKU-level sales forecasts from models and experts. Expert Systems with Applications, 38.3: 2011, 2365-2370.

3. Rob J Hyndman, Roman A Ahmed, George Athanasopoulos, and Han Lin Shang. Optimal combination forecasts for hierarchical time series. Computational statistics & data analysis, 55(9): 2011, 2579–2589.

4. Charles W Gross and Jeffrey E Sohl. Disaggregation methods to expedite product line forecasting. Journal of forecasting, 1990, 9(3):233–254.

5. George Athanasopoulos, Roman A Ahmed, and Rob J Hyndman. Hierarchical forecasts for australian domestic tourism. International Journal of Forecasting, 25(1): 2009, 146–166.

6. Athanasopoulos, G., Ahmed, R. A., & Hyndman, R. J. Hierarchical forecasts for australian domestic tourism. International Journal of Forecasting, 2009, 25 , 146–166.

7. Wickramasuriya, S. L., Athanasopoulos, G., & Hyndman, R. J. Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. Journal of the American Statistical Association, 2019, 114 , 804–819.

8. Shanika L Wickramasuriya, George Athanasopoulos, and Rob J Hyndman. Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. Journal of the American Statistical Association, 114(526):804–819, 2019.

9. MANCUSO, Paolo; PICCIALLI, Veronica; SUDOSO, Antonio M. A machine learning approach for forecasting hierarchical time series. Expert Systems with Applications, 2021.

10. Mancuso, P., Piccialli, V., & Sudoso, A. M. Hierarchical sales data of an italian grocery store. Mendeley Data, V1. 2021.

11. M5-Competitiion [Електронний ресурс] – Режим доступу до ресурсу: https://mofc.unic.ac.cy/m5-competition/ (дата звернення 09.11.2022)

12. SAGHEER, Alaa; HAMDOUN, Hala; YOUNESS, Hassan. Deep LSTM-based transfer learning approach for coherent forecasts in hierarchical time series. Sensors, 2021, 21.13: 4379.

13. Sagheer, A.; Kotb, M. Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing, 2019, 323, 203–213

14. YU, Yong, et al. A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 2019, 31.7: 1235-1270.

15. Guido van Rossum, Python Reference Manual, release 2.4.4, 18 October 2006.

16. Pandas [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/docs/ (дата звернення 17.11.2022)

17. Google Colaboratory [Електронний ресурс] – Режим доступу до ресурсу: https://colab.research.google.com/notebooks/intro.ipynb. (дата звернення 2.12.2022)

18. Carneiro, T., Da Nóbrega, R. V. M., Nepomuceno, T., Bian, G. B., De Albuquerque, V. H. C., & Reboucas Filho, P. P. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. IEEE Access, 2018, 6, 61677-61685.

19. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. J. Big Data 2016, 3, 9.

20. Hyndman, R. J. et al. Another look at forecast-accuracy metrics for intermittent demand. Foresight: The International Journal of Applied Forecasting, 2006, 4 , 43–46.

Додаток А

Код програми

```
!cp /content/drive/MyDrive/hierarchical_sales_data.csv /content
import pandas as pd
import numpy as np
import tensorflow as tf
from numpy.random import seed
from pandas import read_csv, DataFrame, concat
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, StandardScaler
from pandas import Series, concat, datetime
from keras.layers import TimeDistributed, BatchNormalization, RepeatVector, Input, GRU, LSTM, Dense, Dropout
from keras.models import Model, load_model, Sequential
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
from math import sqrt
import matplotlib.pyplot as plt

def MASE(x_true,x_pred):
  result1=0
  result2=0
  for i in range(len(x_true)):
    result1 += abs(x_true[i]-x_pred[i])
  for i in range(len(x_true)-1):
    result2 += abs(x_true[i+1]-x_pred[i])
  result1 /= len(x_true)
  result2 /= len(x_true) - 1
  return result1/result2

def Mase0(x_true, x_pred, column = 0):
  result=0
  arr_len=len(x_true)
  return MASE(x_true.loc[:, x_true.columns[column]], x_pred.loc[:, x_pred.columns[column]])

b1_len = 42
b1 = 42
b2_len = 45
b2 = b1_len + b2_len
b3_len = 21
b3 = b2 + b3_len
b4_len = 10
b4 = b3 + b4_len

def Mase_All(x_true, x_pred):
  columns_pred = x_pred.columns
  result=0
  result_b1=0
```

```
  result_b2=0
  result_b3=0
  result_b4=0
  len = x_pred.shape[1]
  for i in range(len):
    mase_i = Mase0(x_true, x_pred, i)
    result += mase_i
    if i < b1:
      result_b1 += mase_i
    elif i >= b1 and i < b2:
      result_b2 += mase_i
    elif i >= b2 and i < b3:
      result_b3 += mase_i
    elif i >= b3 and i < b4:
      result_b4 += mase_i
    elif i >= b4:
      print('MASE',columns_pred[i],' : ', mase_i)
  print('MASE mean B1: ', result_b1/b1_len)
  print('MASE mean B2: ', result_b2/b2_len)
  print('MASE mean B3: ', result_b3/b3_len)
  print('MASE mean B4: ', result_b4/b4_len)
  print('MASE mean ALL: ', result/len)

  return result/len

def split_series(series, num_past, num_future):
  X, y = list(), list()
  for window_start in range(len(series)):
    past_end = window_start + num_past
    future_end = past_end + num_future
    if future_end > len(series):
      break
    past, future = series[window_start:past_end, :], series[past_end:future_end, :]
    X.append(past)
    y.append(future)
  return np.array(X), np.array(y)


n_past = 35
n_future = 7
n_features = 118
df_all = pd.read_csv('hierarchical_sales_data.csv', index_col=0, parse_dates=True)
df = df_all.iloc[:,:n_features]
df_diff = pd.DataFrame((df.iloc[1:].values - df.iloc[:-1].values), columns=df.columns, index=df.index[1:])
X, y = split_series(df_diff.values, n_past, n_future)
index = df_diff.iloc[:-(n_future-1+n_past)].index #past 7 -13 14 -20 30 -36
index_train_mask = index <= '2016-12-31'
index_val_mask = (index > '2017') & (index <= '2017-12-31')
```

```
index_test_mask = index > '2018'

index_train = index[index_train_mask]
index_val = index[index_val_mask]
index_test = index[index_test_mask]

X_train = X[index_train_mask]
y_train = y[index_train_mask]

X_val = X[index_val_mask]
y_val = y[index_val_mask]

X_test = X[index_test_mask]
y_test = y[index_test_mask]
scaler = StandardScaler()
scaler.fit(df_diff.loc[index_train].values)
X_train_scaled = scaler.transform(X_train.reshape(-1, X_train.shape[-1])).reshape(X_train.shape)
y_train_scaled = scaler.transform(y_train.reshape(-1, y_train.shape[-1])).reshape(y_train.shape)

X_val_scaled = scaler.transform(X_val.reshape(-1, X_val.shape[-1])).reshape(X_val.shape)
y_val_scaled = scaler.transform(y_val.reshape(-1, y_val.shape[-1])).reshape(y_val.shape)

X_test_scaled = scaler.transform(X_test.reshape(-1, X_test.shape[-1])).reshape(X_test.shape)
y_test_scaled = scaler.transform(y_test.reshape(-1, y_test.shape[-1])).reshape(y_test.shape)
model = Sequential()
model.add(LSTM(15, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(10, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(7))
model.add(Dropout(0.2))
model.add(RepeatVector(n_future))
model.add(LSTM(5, return_sequences=True))
model.add(Dropout(0.2))
model.add(TimeDistributed(Dense(n_features)))
model.compile(optimizer='adam', loss='mse')
model.summary()
es = EarlyStopping(patience=100, monitor='val_loss', restore_best_weights=True, verbose=1)

hist=model.fit(X_train_scaled,y_train_scaled,epochs=900,validation_data=(X_val_scaled, y_val_scaled),verbose=2,shuffle
=False, callbacks=es)
y_test_pred=model.predict(X_test_scaled)
y_test_pred_resc = scaler.inverse_transform(y_test_pred.reshape(-1, y_test_pred.shape[-1])).reshape(y_test_pred.shape)
df_test = df.loc['2018':].copy()
df_diff_train = df_diff.loc[:'2017-12-31'].copy()
df_diff_test = df_diff.loc['2018':].copy()
df_real_test = df_all.loc['2018'].iloc[n_past:,:123].copy() #7 - 2018-01-09 14 2018-01-16
```

```
df_pred = df_test.iloc[n_past:,:].copy()
def inverse_dif(real, pred_diff):
  for col in range(len(df_test.columns)):
    y_test_pred_value = pred_diff[0, :, col].cumsum() + df_test.iloc[n_past-1,col]
    y_test_pred_value[y_test_pred_value < 0] = 0
    real.iloc[:n_future,col] = y_test_pred_value
    for i in range(real.shape[0]-n_future):
      pred_val_for_7_days = pred_diff[i, :, col].cumsum() + df_test.iloc[n_past+i, col]
      pred_val_for_7_days[pred_val_for_7_days < 0] = 0
      real.iloc[n_future+i,col] = pred_val_for_7_days[-1]

inverse_dif(df_pred, y_test_pred_resc)
Mase_All(df_real_test, df_pred)
df_pred['B1'] = df_pred.iloc[:,:42].sum(axis=1)
df_pred['B2'] = df_pred.iloc[:,42:87].sum(axis=1)
df_pred['B3'] = df_pred.iloc[:,87:108].sum(axis=1)
df_pred['B4'] = df_pred.iloc[:,108:118].sum(axis=1)
df_pred['TOTAL'] = df_pred.iloc[:,:118].sum(axis=1)
print('B1')
print(MASE(df_real_test['B1'].values, df_pred['B1'].values))
print('B2')
print(MASE(df_real_test['B2'].values, df_pred['B2'].values))
print('B3')
print(MASE(df_real_test['B3'].values, df_pred['B3'].values))
print('B4')
print(MASE(df_real_test['B4'].values, df_pred['B4'].values))
print('TOTAL')
print(MASE(df_real_test['TOTAL'].values, df_pred['TOTAL'].values))
```