https://www.proconference.org/index.php/usc/article/view/usc30-00-002
DOI: 10.30888/2709-2267.2025-30-00-002

УДК 004.9

MODELING ARIMA AND LSTM TIME SERIES MODELS IN PYTHON

МОДЕЛЮВАННЯ ARIMA ТА LSTM МОДЕЛЕЙ ЧАСОВИХ РЯДІВ В PYTHON

Doroshenko I.V. / Дорошенко І.В.
c. p.-m.s., as.prof. / к. ф.-м.н., доц.
ORCID: 0000-0001-8729-1768
Tarasov M.S. / Тарасов М.С.
magistr / магістр
Chernivtsi National University, Chernivtsi, Kotsyubynskoho 2, 58012
Чернівецький національний університет, Чернівці, вул.Коцюбинського 2, 58012

Анотація. *У цій статті розглянуто теоретичні основи та практичні аспекти застосування моделей ARIMA, архітектури RNN, принципи роботи LSTM, а також їх використання для прогнозування часових рядів у середовищі Python за допомогою бібліотек Statsmodels та Keras. Порівняння цих підходів дозволяє оцінити їхні переваги та обмеження, що допоможе вибрати оптимальний метод для конкретних завдань прогнозування.*

Ключові слова: *аналіз часових рядів, машинне навчання, нейронні мережі.*

Abstract. *This article examines the theoretical foundations and practical aspects of applying ARIMA models, RNN architecture, and the working principles of LSTM, as well as their use for time series forecasting in Python using the Statsmodels and Keras libraries. Comparing these approaches allows for an evaluation of their advantages and limitations, helping to select the optimal method for specific forecasting tasks.*

Keywords: *time series analysis, machine learning, neural networks.*

Вступ.

Часові ряди є важливим інструментом аналізу в багатьох сферах науки та бізнесу, дозволяючи прогнозувати майбутні тенденції на основі історичних даних. У сучасному аналізі часових рядів застосовуються різні методи, серед яких особливе місце займають статистичні підходи, такі як моделі ARIMA, та методи машинного навчання, включаючи рекурентні нейронні мережі (RNN) і їхні покращені варіанти, зокрема LSTM.

1. ARIMA моделі

ARIMA моделі (авторегресивні інтегровані моделі ковзного середнього) пропонують ефективний підхід для аналізу та прогнозування часових рядів. ARIMA моделі, запропоновані Боксом і Дженкінсом у 1970 році, розширюють можливості ARMA моделей, додаючи компоненти для обробки нестабільних даних. Вони об'єднують авторегресивні та ковзні середні моделі з диференціюванням для врахування трендів і сезонності.

Побудови моделей ARIMA

Побудова моделей ARIMA передбачає декілька основних кроків.

1. Аналіз даних: Починаємо з побудови графіка часового ряду та візуального огляду даних для виявлення будь-яких аномалій, трендів або сезонності.

2. Трансформація даних: Якщо у даних спостерігається нестабільність дисперсії або інші аномалії, може бути застосована трансформація даних, така як логарифмування або Box-Cox трансформація, для стабілізації дисперсії та вирішення інших проблем.

3. Визначення порядку моделі: Далі потрібно визначити порядок моделі ARIMA, тобто параметри p, d і q , де p - порядок авторегресії, d - ступінь різницювання та q - порядок ковзного середнього.

4. Параметризація моделі: Після визначення порядку моделі, застосовуємо методи оцінки параметрів для ARIMA моделі, такі як метод найменших квадратів або метод максимальної правдоподібності.

5. Діагностика моделі: Після оцінки параметрів проводиться діагностика моделі, щоб переконатися, що модель відповідає даним. Це може включати аналіз залишкових після побудови моделі, тестування на незалежність та нормальність залишків.

6. Вибір оптимальної моделі: На основі результатів діагностики обирається найбільш оптимальна модель ARIMA для прогнозування майбутніх значень часового ряду.

2. Архітектура рекурентних нейронних мереж (РРН)

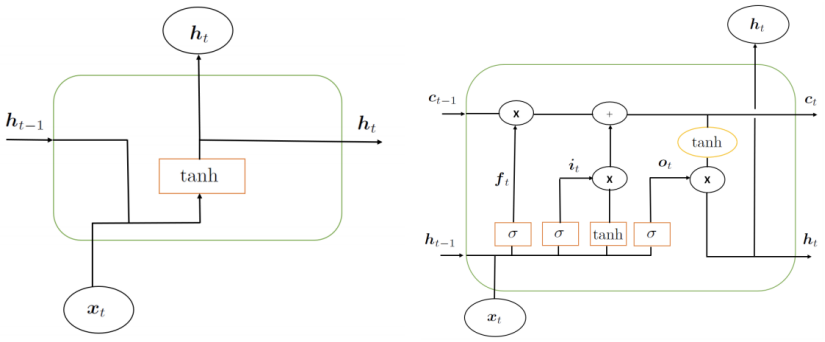
Прогнозування часових рядів є важливою галуззю машинного навчання з метою передбачення майбутнього. Прогноз майбутніх точок грає вирішальну роль у прийнятті рішень в таких самих областях, як прогнозування попиту.

Традиційні нейронні мережі не підходять для прогнозування часових рядів, оскільки вони розглядають кожний вхід та вихід незалежно один від одного. Замість цього, можна скористатися рекурентними нейронними мережами (РРН), які враховують залежності від попередніх точок даних і є більш ефективними для прогнозування часових рядів.

Однак РРН мають проблему з вивченням довгострокових залежностей. Проблему можна вирішити за допомогою більш продуктивних ітерацій РРН, таких як нейронні мережі з довгостроковою та короткостроковою пам'яттю (LTSM).

3. Опис архітектури LSTM

Модель довготривалої та короткотривалої пам'яті (LSTM) була розроблена для подолання недоліків рекурентних нейронних мереж (RNN), таких як проблеми зникливого градієнта. У RNN складається з послідовності повторюваних модулів нейронної мережі, де кожен модуль має структуру, що включає один шар *tanh*. *Tanh* використовується для обмеження значень між -1 та 1. Вхідні дані зберігаються в прихованому стані h_t, який також містить дані з попереднього прихованого стану h_{t-1}. Це досить просто. У LSTM також є така ж послідовність повторюваних модулів, як у RNN, за винятком того, що структура модуля LSTM складається з чотирьох шарів замість одного шару, як у RNN, що допомагає вирішити проблему зникливого градієнта у RNN. На рисунку 1 показана різниця.



(a) (b)

Рисунок 1 - «RNN (a) та LSTM (b)»

Основна перевага LSTM полягає в тому, що вона може ефективно управляти інформацією, яка перебуває в пам'яті моделі на тривалий час, що дозволяє їй враховувати довгострокові залежності в часі.

Основні компоненти LSTM включають:

- Клітинний стан (Cell State): Це основна "пам'ять" моделі LSTM, яка зберігає інформацію на тривалий термін.

- Ворота входу (Input Gate): Визначає, яка частина нової інформації буде додана до клітинного стану.
- Ворота забування (Forget Gate): Визначає, яка частина інформації в клітинному стані буде забута або проігнорована.
- Ворота виходу (Output Gate): Визначає, яка частина клітинного стану буде використана для генерації виходу.

4. Моделювання в Python

І Statsmodels, і Keras надають потужні інструменти для моделювання часових рядів за допомогою ARIMA та LSTM відповідно. Вибір між ними залежить від характеру даних та вимог до моделі. ARIMA, реалізована в Statsmodels, підходить для аналізу стаціонарних часових рядів з автокореляцією, тоді як LSTM у Keras є кращим вибором для складних даних з довгостроковими залежностями.

Statsmodels для моделювання ARIMA

Statsmodels є потужним інструментом для статистичного аналізу та економетричного моделювання в Python. Він забезпечує широкий спектр методів для аналізу часових рядів, включаючи ARIMA. ARIMA використовується для моделювання часових рядів, що демонструють автокореляцію, тобто залежність поточних значень від попередніх.

Основна ідея ARIMA полягає в поєднанні трьох компонентів: автогресії (AR), інтеграції (I) та ковзного середнього (MA). AR-компонент моделює залежність між поточним і попередніми значеннями, I-компонент застосовується для перетворення нестаціонарних рядів у стаціонарні шляхом диференціювання, а MA-компонент враховує залежність поточних значень від попередніх помилок.

За допомогою Statsmodels можна легко створити, оцінити та передбачити моделі ARIMA. Пакет надає функцію ARIMA, яка дозволяє визначити порядок моделі (p, d, q) та інші параметри. Після оцінки моделі можна використовувати її для прогнозування майбутніх значень та оцінки точності моделі. Statsmodels також забезпечує зручні інструменти для діагностики моделей та візуалізації результатів.

Функція *arima_model(series, data_split, params, future_periods, log)* створює поточний прогноз моделі ARIMA для заданого вхідного часового ряду. Це дозволяє користувачеві вибирати значення *p*, *q* і *d*, а також вказувати, чи здійснювати log перетворення. Етапи роботи функції включають:

- log трансформація даних;
- створення тренувальних/тестових вибірок;
- створення моделі ARIMA для тренувального набору;
- прогнозування першого значення в тестовому наборі з наступним додаванням цього значення до навчального набору та ремоделювання, прогнозування наступного значення в тестовому наборі, додавання цього другого значення до навчальної вибірки і так далі;
- зворотне перетворення даних;
- створення графіків і генерація показників помилок.

Ця функція використовуватиметься нижче з налаштуванням параметрів *p*, *d*, *q* та логарифмічного перетворення, щоб мінімізувати значення RMSE для даних поза вибіркою, пов’язаних із різними часовими рядами.

Keras для моделювання LSTM

Keras, високорівневий API для нейронних мереж, є популярним вибором для побудови та навчання рекурентних нейронних мереж (RNN), включаючи LSTM (довготривалу короткочасну пам'ять). LSTM-моделі особливо корисні для аналізу часових рядів, оскільки вони можуть вловлювати довгострокові залежності в даних.

Використання Keras для побудови LSTM-моделей є досить простим. Спочатку визначається модель, додаючи шари LSTM та інші необхідні шари, такі як Dropout або Dense. Після цього модель компілюється з вибраною функцією втрат та оптимізатором. Процес навчання здійснюється шляхом подачі навчальних даних та налаштування кількості епох та розміру пакету. Завдяки своїй простоті та ефективності, Keras дозволяє швидко створювати та експериментувати з моделями LSTM, що робить його відмінним інструментом для задач прогнозування часових рядів.

Обчислення LSTM використовують три різні функції для моделювання, навчання та тестування нейронної мережі LSTM. Далі здійснюється інвертування будь-яких перетворень даних, щоб побудувати графіки та проаналізувати помилки в початковому масштабі.

Функція *create_dataset (data_series, look_back, split_frac, transforms)* використовується для створення наборів даних, необхідних для навчання та тестування нейронних мереж LSTM. Вона приймає часовий ряд, кількість попередніх періодів, які користувач хотів би змоделювати, дроби поділу тренувань/тестів, а також те, чи потрібно виконувати диференційні або логарифмічні перетворення даних, щоб зробити їх стаціонарними. Вона також нормалізує всі дані від 0 до 1 для введення в LSTM.

Об'єкти, що підлягають навчанню, – це, по суті, періоди часу $t - n$. Отже, якщо моделювати дані за поточний місяць $(t - 0)$, вхідною функцією будуть дані за попередній місяць $(t - 1)$, тоді як цільове значення буде фактичним значенням для цього поточного місяця (за умови, що воно відоме). Збільшення look_back додасть додаткові попередні місяці для включення, напр. два місяці тому $(t - 2)$ і три місяці тому $(t - 3)$.

Функція зворотного перетворення *inverse_transforms(train_predict, y_train, test_predict, y_test, data_series, train_dates, test_dates, scaler)* просто скасовує будь-які перетворення, виконані під час створення набору даних. Інверсія перетворень дозволяє базувати прогнози моделі на тому ж масштабі, що й вихідний набір даних, для більш інтуїтивної інтерпретації результатів. І функція створення моделі, і функція зворотного перетворення автоматично викликаються у функції LSTM.

Для виклику наведеної нижче моделі LSTM *lstm_model(data_series, look_back, split, transforms, lstm_params)* потрібен лише набір даних часового ряду data_series, кількість бажаних періодів ретроспективного аналізу look_back, поділ тренування/тесту split, чи потрібно здійснювати log перетворення чи різницю даних, а також параметри для навчання *lstm_params*, такі як кількість вузлів і епох. У межах функції він створює набори даних навчання та тестування

– як функції, так і залежної змінної, – а потім навчає модель LSTM, після чого виконується прогнозування даних поза вибіркою. Прогнози з моделі, а також фактичні значення потім обернено перетворюються, генеруються графіки та показники помилок.

Функція *gauss_compare(original_series, predictions, data_split)* дозволяє порівнювати ряди прогнозованих даних, відфільтрованих за Гауссом, із вихідними рядами даних за допомогою візуалізації та звітування про RMSE.

5. Модельний приклад

Було здійснено порівняння моделі авторегресії інтегрованого ковзного середнього (ARIMA) та моделі довгострокової короткочасної пам'яті (LSTM) для різних часових рядів. Було створено узагальнені функції, які можуть швидко тестувати, повторювати та оптимізувати моделі ARIMA та LSTM для заданих вхідних даних часового ряду. Моделі ARIMA та LSTM використано для прогнозування семи наборів даних, які було вибрано через їх унікальні форми, щоб перевірити різні сезонні зміни, зростання значень з часом та різкі розбіжності. Для кожного набору даних побудовано графік вихідних даних та результати тестування поза вибіркою. Крім того, у дослідження включено результати, які використовують фільтрацію Гауса для згладжування вихідного набору даних, з подальшим моделюванням згладженого набору даних. Згладжені результати моделювання порівнюються з вихідними даними часового ряду для того, щоб визначити, чи покращила фільтрація продуктивність моделей.

Висновки.

Модельний приклад показав, що ARIMA дала нижчі середньоквадратичні похибки (RMSE) у 5/7 досліджуваних часових рядів порівняно з моделлю LSTM. У багатьох випадках моделі давали подібні похибки, але в цілому ARIMA забезпечувала результати вищої якості. Результати дослідження показують, що як ARIMA, так і LSTM є ефективними алгоритмами для прогнозування часових рядів. Модель ARIMA демонструє дещо менші похибки, але може стикатися з проблемами збіжності на рядах з різкими градієнтами. Модель LSTM здатна навчатися та прогнозувати будь-які часові ряди, хоча точність прогнозів

потребує окремої оцінки.

Гауссова фільтрація набору даних перед створенням моделі давала менші похибки в кожному випадку, навіть якщо порівнювати результати моделі з вихідними, невідфільтрованими даними. У середньому прогнози, відфільтровані Гауссом, зменшили RMSE майже на 100%. Додатково, фільтрація набору даних за допомогою Гауссового фільтру покращувала прогнози в усіх випадках, навіть коли порівнювати відфільтровані дані з оригінальними без фільтрації. Середні RMSE для нефільтрованих даних, ARIMA: 21,69 та LSTM: 23,54. Середні RMSE для відфільтрованих даних за Гауссом, ARIMA: 10,98 і LSTM: 12,22.

Література:

[1] Тарасов М. С. Порівняння ARIMA та LSTM моделей часових рядів / Кваліфікаційна робота.- ЧНУ. – 2024. – 66 с.

[2] Brownlee J. Introduction to Time Series Forecasting with Python. — 1st ed. — 2020. — 365 p.

Стаття відправлена: 10.02.2025 р.