МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)

червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна система прогнозування котировок валют»
здобувача групи ІН – 93 Смажника Олександра Вячеславовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олександр СМАЖНИК
(підпис)

Керівник,
кандидат технічних наук, доцент В'ячеслав МОСКАЛЕНКО _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-93Смажника Олександра Вячеславовича

1. Тема роботи: « Інформаційна технологія прогнозування курсу валют ».
затверджую наказом по СумДУ від *«01» червня 2023 р. № 0475-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для прогнозування курсу валют. 3) Розробка неромережі здатної прогнозувати курси валют. 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються для прогнозування курсу валют*		
3	*Розробка неромережі здатної прогнозувати курси валют*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 49 стр.,1 рис., 1 табл., 1 додаток, 12 використаних джерел.

Обґрунтування актуальності теми роботи – нейромережі все більше і більше входять в наше життя. Тому їх використання для більш-меньш точних прогнозів на фондовій біржі значно полегшить гравцям життя та зробить їх роботу меньш трудомісткою.

Об'єкт дослідження – процес прогнозування котировок валют

Предмет дослідження – моделі і методи інформаційної технології аналізу даних для прогнозування котировок валют

Мета роботи – розроблення нейромережі здатної прогнозувати котировки валют

Результати – розроблено нейромережу, що здатна прогнозувати котировки американського доллара до Євро. Мережа має достатню точність тому може використовуватись як самодостатнє ПЗ чи в складі іншої програми

АРХІТЕКТУРА LSTM, СЕРЕДНЬОКВАДРАТИЧНА ПОМИЛКА (MSE),ФУНКЦІЯ ВТРАТ

ЗМІСТ

ВСТУП 5
1 АНАЛІЗ ПРОБЛЕМИ 6
1.1 Сучасний стан та тенденції розвитку інформаційних систем прогнозування курсу валют 6
1.2 Аналіз методів прогнозування часових рядів 8
1.3 Формалізована постановка задачі 21
2 МОДЕЛІ І МЕТОДИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ КУРСУ ВАЛЮТ 22
2.1 Опис архітектури моделі аналізу даних 22
2.2 Опис методу навчання 25
2.3. Критерії оцінювання ефективності моделі 38
3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПРОГНОЗУВАННЯ КУРСУ ВАЛЮТ ТА РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ 41
3.1 Формування навчальних і тестових даних 41
3.2 Короткий опис програмної реалізації 42
3.3 Результати експериментів 45
ВИСНОВКИ 50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 51
ДОДАТОК 52

ВСТУП

1 АНАЛІЗ ПРОБЛЕМИ

1.1 Сучасний стан та тенденції розвитку інформаційних систем прогнозування курсу валют

Сучасні інформаційні системи прогнозування курсу валют розвиваються швидкими темпами, надаючи користувачам більш точні та надійні прогнози. Ось деякі з основних тенденцій, які можна спостерігати в цій сфері:

Використання штучного інтелекту: Застосування штучного інтелекту (ШІ) дозволяє інформаційним системам аналізувати великі обсяги даних, враховувати складні залежності та тренди, що допомагає уточнити прогнози курсу валют. Машинне навчання, глибоке навчання та нейронні мережі стають ключовими технологіями для досягнення більш точних результатів.

Використання Big Data: Завдяки зростанню обсягу доступних фінансових даних, інформаційні системи прогнозування курсу валют використовують Big Data для аналізу та виявлення складних патернів. Збір, обробка та аналіз великих обсягів даних дозволяє отримувати більш об'єктивні та деталізовані прогнози.

Врахування нефінансових факторів: Крім традиційних фінансових даних, таких як макроекономічні показники та статистика, інформаційні системи все більше враховують нефінансові фактори. Це можуть бути політичні події, геополітичні конфлікти, природні катаклізми тощо. Аналіз таких факторів допомагає більш точно прогнозувати коливання курсу валют.

Застосування алгоритмів аналізу тексту: Інформаційні системи все частіше використовують алгоритми аналізу тексту для моніторингу та аналізу новин, соціальних медіа та інших джерел інформації. Це дозволяє виявляти настрої та настанову ринку, що має вплив на курс валют.

Розширення доступу до прогнозів: Завдяки розвитку мобільних додатків та онлайн-платформ, користувачам стає доступним отримання прогнозів курсу

валют у реальному часі. Це забезпечує швидкий доступ до інформації та дозволяє приймати швидкі та обґрунтовані рішення щодо валютних операцій.

Ці тенденції свідчать про постійне удосконалення інформаційних систем прогнозування курсу валют і розширення їх можливостей для користувачів. Використання блокчейн технологій: Блокчейн технологія забезпечує безпеку, надійність та прозорість обробки фінансових транзакцій. Децентралізована природа блокчейну дозволяє створювати системи прогнозування курсу валют, що базуються на розподіленій мережі, забезпечуючи швидкість та достовірність даних.

Розвиток квантових обчислень: Квантові обчислення відкривають нові можливості для аналізу складних фінансових даних та прогнозування курсу валют. Ці обчислення здатні здійснювати багато паралельних обчислень одночасно, що може прискорити процес прогнозування та поліпшити точність результатів.

Розширення використання географічних даних: Географічні дані можуть впливати на курс валют через економічні, соціальні та політичні фактори. Інформаційні системи прогнозування курсу валют стають більш уважними до цих факторів та враховують географічні особливості для точнішого прогнозування.

Розширення інтеграції з фінансовими ринками: Інформаційні системи прогнозування курсу валют все більше інтегруються з ринковими платформами та біржами. Це дозволяє отримувати доступ до реального часу та точної фінансової інформації, що поліпшує якість та достовірність прогнозів.

Застосування криптовалют та стабільних монет: Використання криптовалют та стабільних монет може вплинути на прогнозування курсу валют. Інформаційні системи враховують вплив цих цифрових активів на ринок та розглядають їх в контексті прогнозування курсів національних валют.

1.2 Аналіз методів прогнозування часових рядів

Аналіз методів прогнозування часових рядів є важливим етапом у прогнозуванні майбутніх значень часових даних на основі їх минулих спостережень. Існує широкий спектр методів, які застосовуються для прогнозування часових рядів, і кожен з них має свої переваги та обмеження. Ось огляд деяких ключових методів прогнозування часових рядів:

Методи засновані на згладжуванні:

Метод Ковзне середнє (Moving Average, MA) - це один з найпростіших методів прогнозування часових рядів. Він базується на обчисленні середнього значення попередніх спостережень для прогнозування майбутніх значень.

Процес прогнозування з використанням методу ковзного середнього полягає у виборі певного вікна (так званого періоду ковзного середнього), яке включає певну кількість останніх спостережень. Спочатку обчислюється середнє значення цих спостережень, і це значення вважається прогнозом наступного кроку в часі. Потім вікно зсувається на одну одиницю вперед і повторюється процес обчислення середнього значення та прогнозування. Такий процес повторюється для всього часового періоду, на який проводиться прогноз.

Один з основних факторів, який необхідно враховувати при застосуванні методу ковзного середнього, це вибір оптимального розміру вікна. Розмір вікна визначається кількістю попередніх спостережень, які враховуються при обчисленні середнього значення. Занадто коротке вікно може не забезпечити достатньої інформації для точного прогнозу, тоді як занадто довге вікно може призвести до затримки в оцінці тенденцій та змін в ряді.

Метод ковзного середнього особливо ефективний для рядів, що не мають вираженої сезонності або тренду, і які можна вважати стаціонарними. Однак він може бути менш ефективним для складних рядів зі складними залежностями та неперіодичними варіаціями.

Важливо пам'ятати, що метод ковзного середнього є лише одним з багатьох методів прогнозування часових рядів і може бути використаний як основа для більш складних моделей.

Вагове ковзне середнє (Weighted Moving Average, WMA) є модифікацією методу ковзного середнього, в якому враховуються ваги для кожного спостереження при обчисленні середнього значення. У ваговому ковзному середнему останнім спостереженням надається більша вага, ніж більш старим спостереженням.

Для застосування вагового ковзного середнього спочатку необхідно визначити ваги для кожного спостереження. Ваги можуть бути задані наперед визначеними значеннями або залежати від різних факторів, таких як відстань від поточного часового періоду або інші параметри, що враховуються в конкретній ситуації. Зазвичай ваги сумуються до одиниці.

Після визначення ваг для кожного спостереження проводиться обчислення зваженого середнього значення, використовуючи ці ваги. Кожне спостереження множиться на вагу, і всі ці значення сумуються. Отримана сума ділиться на суму ваг, щоб отримати вагове середнє значення. Це значення вважається прогнозом наступного кроку в часі.

Одним з важливих аспектів вагового ковзного середнєго є вибір ваг для кожного спостереження. Ваги можуть бути вибрані таким чином, щоб приділити більшу увагу останнім спостереженням або більш важливим точкам даних. Варто враховувати, що ваги можуть бути статичними (незмінними) або змінюватися з часом в залежності від поточної ситуації.

Вагове ковзне середнє є більш гнучким методом, ніж просте ковзне середнє, оскільки дозволяє приділяти більшу увагу недавнім спостереженням. Це може бути корисним у випадках, коли підкреслення останніх даних є важливим для прогнозування майбутніх значень.

Вагове ковзне середнє може бути застосоване в багатьох галузях, де прогнозування часових рядів є важливим завданням. Воно може бути використане для прогнозування фінансових показників, попиту на товари, процесів виробництва та багатьох інших сфер.

Експоненціальне згладжування (Exponential Smoothing, ES) є популярним методом прогнозування часових рядів, який використовує показник згладжування для надання більшої ваги недавнім спостереженням. Цей метод особливо ефективний для рядів, які мають тренд або/і сезонність.

У методі експоненціального згладжування кожне спостереження отримує вагу, яка залежить від показника згладжування. Показник згладжування (символізований як α) приймає значення між 0 і 1 і визначає, яка частка ваги припадає на поточне спостереження, а яка на попередні прогнози. Зазвичай вага, яка припадає на попередні прогнози, пропорційна (1 - α).

Процес прогнозування методом експоненціального згладжування розпочинається з визначення початкового прогнозу, який може бути заданий або обчислений на підставі початкових спостережень. Потім для кожного наступного кроку в часі обчислюється нове згладжене значення шляхом комбінації поточного спостереження та попереднього прогнозу з використанням показника згладжування.

Формула для обчислення нового прогнозу методом експоненціального згладжування може мати наступний вигляд

$$F(t) = \alpha * Y(t) + (1 - \alpha) * F(t-1) \qquad (1.2.1)$$

де F(t) - новий прогноз на часовому кроці t,

Y(t) - поточне спостереження на часовому кроці t,

F(t-1) - попередній прогноз на часовому кроці t-1,

α - показник згладжування.

Важливим аспектом методу експоненціального згладжування є вибір оптимального значення показника згладжування α. Якщо α близьке до 1, то недавні спостереження матимуть більшу вагу, що робить прогноз більш реактивним до змін. З іншого боку, якщо α близьке до 0, то попередні прогнози матимуть більшу вагу, що робить прогноз більш стабільним.

Метод експоненціального згладжування може бути розширений для моделювання рядів з трендом (ETS - Exponential Smoothing State Space Models), додаванням складової тренду та/або сезонності. Це дозволяє прогнозувати ряди зі складнішими залежностями та використовувати їх для аналізу та планування у бізнесі.

Тройне експоненціальне згладжування, відоме також як метод Хольта-Вінтерса (Holt-Winters), є методом прогнозування часових рядів, який враховує тренд і сезонність. Цей метод є розширенням експоненціального згладжування і дозволяє моделювати ряди з трендом та/або сезонністю.

Тройне експоненціальне згладжування використовує три компоненти для прогнозування: рівень (level), тренд (trend) та сезонність (seasonality). Кожна компонента має свої параметри згладжування, які визначають вагу, надану кожній компоненті при обчисленні прогнозу.

Формули для обчислення нового прогнозу методом Хольта-Вінтерса можуть мати наступний вигляд:

Рівень (Level)

$$L(t) = \alpha * Y(t) / S(t-P) + (1 - \alpha) * (L(t-1) + T(t-1)) \quad (1.2.2)$$

Тренд (Trend)

$$T(t) = \beta * (L(t) - L(t-1)) + (1 - \beta) * T(t-1) \quad (1.2.3)$$

Сезонність (Seasonality)

$$S(t) = \gamma * Y(t) / L(t) + (1 - \gamma) * S(t-P) \quad (1.2.4)$$

Прогноз (Forecast)

$$F(t+m) = (L(t) + m * T(t)) * S(t-P+m) \quad (1.2.5)$$

де L(t) - рівень на часовому кроці t,

T(t) - тренд на часовому кроці t,

S(t) - сезонність на часовому кроці t,

Y(t) - спостереження на часовому кроці t,

P - період сезонності,

α, β, γ - показники згладжування для рівня, тренду та сезонності відповідно,

m - кількість кроків вперед для прогнозу.

Параметри α, β та γ варто налаштовувати, щоб відобразити характеристики ряду та забезпечити точність прогнозування. Зазвичай вони приймають значення між 0 та 1.

Тройне експоненціальне згладжування є потужним методом для прогнозування рядів з трендом та/або сезонністю, оскільки воно може моделювати та ураховувати ці залежності при прогнозуванні майбутніх значень. Цей метод знаходить своє застосування у багатьох сферах, таких як фінанси, економіка, логістика та інші, де прогнозування часових рядів є важливим завданням.

Методи засновані на регресії:

Проста лінійна регресія є статистичним методом, який досліджує залежність між двома змінними: незалежною змінною (X) і залежною змінною (Y). Цей метод дозволяє побудувати лінійну функцію, яка найкращим чином описує цю залежність і може бути використана для прогнозування значень Y на підставі відомих значень X.

У простій лінійній регресії ми шукаємо лінійну функцію виду

$$Y = \beta_0 + \beta_1 X, \qquad (1.2.6)$$

де Y - залежна змінна, X - незалежна змінна, β_0 - перехоплення (початкове значення Y, коли X = 0), β_1 - нахил (коефіцієнт, що визначає нахил лінії).

Метою простої лінійної регресії є знаходження оптимальних значень β_0 і β_1, які найкраще відповідають даним. Це зазвичай виконується шляхом методу найменших квадратів, де ми шукаємо такі значення β_0 і β_1, які мінімізують суму квадратів відхилень між спостережуваними значеннями Y і прогнозованими значеннями за допомогою лінійної функції.

Проста лінійна регресія може бути використана для прогнозування майбутніх значень залежної змінної на підставі нових значень незалежної змінної. Після встановлення оптимальних значень β_0 і β_1, ми можемо підставити нові значення X у лінійну функцію, щоб отримати прогнозовані значення Y.

Проста лінійна регресія широко застосовується в багатьох галузях, таких як економіка, фінанси, наука про дані, маркетинг і багато інших, де важливо вивчати та прогнозувати залежності між змінними. Вона дозволяє отримати просту інтерпретацію залежності та зробити прогнози на підставі доступних даних.

Авторегресійні моделі (AR) є статистичними моделями, що використовуються для аналізу та прогнозування часових рядів. Ці моделі

базуються на припущенні, що поточне значення ряду залежить від попередніх значень того ж ряду.

У моделях AR використовується поняття автокореляції, яке визначає статистичний зв'язок між значеннями ряду в різних моментах часу. Автокореляція вимірює ступінь подібності між значеннями ряду та його лагованими (затриманими) значеннями на певному затримку.

Модель AR виражає поточне значення ряду як лінійну комбінацію його попередніх значень з додаванням стохастичної складової (шуму). Формально, модель AR(p) має наступний вигляд

$$Y(t) = c + \varphi_1 * Y(t-1) + \varphi_2 * Y(t-2) + ... + \varphi_p * Y(t-p) + \varepsilon(t), \quad (1.2.7)$$

Де Y(t) - значення ряду на часовому кроці t,

c - постійний член (перехоплення),

$\varphi_1, \varphi_2, ..., \varphi_p$ - параметри моделі, що відповідають коефіцієнтам автокореляції для затримок 1 до p,

ε(t) - стохастична складова, яка представляє шум або нев'язку.

Для використання моделі AR потрібно встановити значення p - кількість попередніх значень ряду, які використовуються для прогнозування. Вибір оптимального значення p може бути здійснений за допомогою статистичних методів, таких як критерій Акаіке або критерій Шварца.

Моделі AR можуть бути використані для прогнозування майбутніх значень ряду на основі попередніх спостережень. Для цього використовується метод автокореляції для оцінки параметрів моделі, а потім використовується лінійна комбінація попередніх значень ряду для прогнозування нових значень.

Авторегресійні моделі є важливим інструментом в аналізі та прогнозуванні часових рядів, і вони знаходять своє застосування в економіці, фінансах,

метеорології, соціальних науках та інших галузях, де важливо моделювати та прогнозувати залежності між спостережуваними величинами в часі.

‘Моделі з авторегресією з ковзним середнім (ARMA) є статистичними моделями, які комбінують авторегресійну (AR) та модель з ковзним середнім (MA) для аналізу та прогнозування часових рядів. Ці моделі дозволяють враховувати як автокореляцію, так і середнє значення помилок моделі.

У моделі ARMA залежність між поточним значенням ряду і попередніми значеннями виражається як лінійна комбінація авторегресійної (AR) складової і моделі з ковзним середнім (MA). AR-складова відображає вплив попередніх значень ряду, тоді як MA-складова враховує середнє значення помилок моделі.

Модель ARMA(p, q) має наступний вигляд

$$Y(t) = c + \varphi_1 * Y(t-1) + \varphi_2 * Y(t-2) + ... + \varphi_p * Y(t-p) + \varepsilon(t) + \theta_1 * \varepsilon(t-1) + \theta_2 * \varepsilon(t-2) + ... + \theta q * \varepsilon(t-q), \quad (1.2.8)$$

де Y(t) - значення ряду на часовому кроці t,

c - постійний член (перехоплення),

$\varphi_1, \varphi_2, ..., \varphi_p$ - параметри моделі AR, що відповідають коефіцієнтам автокореляції для затримок 1 до p,

ε(t) - стохастична складова (шум) з нульовим середнім значенням,

$\theta_1, \theta_2, ..., \theta q$ - параметри моделі MA, що відповідають коефіцієнтам ковзного середнього для затримок 1 до q.

Оцінювання параметрів моделі ARMA зазвичай здійснюється за допомогою методу найменших квадратів або методу максимальної правдоподібності.

Моделі ARMA можуть бути використані для аналізу та прогнозування часових рядів зі складною залежністю, яка включає як автокореляцію, так і

середнє значення помилок. Вони є потужним інструментом в економіці, фінансах, метеорології, соціальних науках та інших галузях, де важливо моделювати та прогнозувати складні залежності в часових рядах.

Методи опорних векторів (SVM) є потужними методами машинного навчання, використовуваними для класифікації та регресії. Їх основна ідея полягає в пошуку оптимальної гіперплощини, яка розділяє дані у просторі таким чином, щоб максимізувати роздільну здатність між класами або підтримувати максимальну відстань між площиною та найближчими точками кожного класу, які називаються опорними векторами.

У випадку класифікації, SVM намагається знайти гіперплощину, яка найкращим чином розділяє дані двох класів у просторі. Якщо дані не можуть бути лінійно розділені, SVM може використовувати "ядрову функцію", яка перетворює дані у вищу розмірність, де вони можуть бути лінійно розділені. Деякі з популярних ядер включають лінійне, поліноміальне та гаусове (радіальне базисне функцію) ядра.

У випадку регресії, SVM намагається побудувати гіперплощину, яка найкращим чином підтримує набір даних. Ідея полягає в тому, щоб знайти гіперплощину, яка максимально віддалена від точок даних, що є відхиленнями (помилками). Значення, прогнозовані SVM, базуються на відстані між точками даних та побудованою гіперплощиною.

Однією з переваг SVM є те, що вони можуть працювати добре навіть з наборами даних з великою кількістю ознак із складною структурою. Вони також показують добрі результати в розпізнаванні образів, розпізнаванні тексту, біоінформатиці та інших завданнях машинного навчання.

Проте, SVM можуть бути вимогливими до обчислювальних ресурсів, особливо при роботі з великими наборами даних. Також важливо правильно

налаштувати параметри SVM, такі як параметр регуляризації та вибір ядра, щоб досягнути оптимальних результатів.

Дерева рішень та випадковий ліс є популярними методами машинного навчання, використовуваними для класифікації та регресії. Вони представляють /собою моделі, які використовують деревоподібну структуру для прийняття рішень на основі вхідних ознак

Дерево рішень є графічною структурою, що складається з вузлів і гілок. Кожен вузол представляє тест на певну ознаку, а гілки відповідають різним варіантам відповіді на цей тест. В процесі прогнозування, дані проходять через дерево, від кореня до листя, де приймаються рішення про класифікацію або прогнозування.

Один з основних переваг дерев рішень полягає в їх інтерпретованості та зрозумілості. Вони можуть бути легко візуалізовані і пояснені, що дозволяє аналізувати, які ознаки мають найбільший вплив на прийняття рішення. Крім того, дерева рішень можуть працювати як з категоріальними, так і з числовими ознаками.

Однак, дерева рішень мають тенденцію до перенавчання, особливо коли вони мають глибоку структуру і поглиблені розділові правила. Це може призвести до низької загальної точності на нових наборах даних. Для подолання цього обмеження можна використовувати випадковий ліс.

Випадковий ліс - це ансамбль дерев рішень, де кожне дерево навчається на підмножині вхідних даних та випадково вибраних ознак. Коли потрібно зробити прогноз, кожне дерево вносить свій внесок, і кінцеве рішення здійснюється шляхом голосування або середнього значення прогнозів дерев. Випадковий ліс зазвичай має кращу загальну точність та мінімізує перенавчання.

Випадковий ліс також може використовуватися для важливості ознак, тобто оцінки впливу кожної ознаки на прогноз. Він може бути використаний для виявлення важливих ознак і відбору найбільш значущих для подальшого аналізу.

Дерева рішень та випадковий ліс є гнучкими та потужними методами машинного навчання, які можуть бути застосовані в широкому спектрі задач класифікації та регресії.

Аналіз спектру:

Аналіз спектру Fourier (або просто аналіз Фур'є) є важливим інструментом для розкладання складних сигналів чи даних у частотному домені. Він базується на ідеї, що будь-який періодичний сигнал може бути представлений як сума гармонічних компонент з різними частотами, аналогічно до розкладу білого світла на різні кольори за допомогою призми.

Аналіз Фур'є використовує трансформацію Фур'є для перетворення сигналу з часового домену в частотний домен. Це означає, що він розкладає сигнал на окремі частотні компоненти, які відображають його спектральний склад. Результатом аналізу Фур'є є спектрограма або спектральний графік, який показує інтенсивність або амплітуду кожної частотної компоненти сигналу.

Аналіз спектру Fourier має широке застосування у багатьох галузях, зокрема в сигнальній обробці, звукотехніці, телекомунікаціях, обробці зображень, фізиці та багатьох інших. Деякі конкретні застосування аналізу Фур'є включають:

Фільтрація сигналів: Аналіз Фур'є дозволяє виділити чи прибрати певні частоти в сигналі, що забезпечує фільтрацію шуму або окремих компонентів сигналу.

Спектральний аналіз звуку: Аналіз Фур'є може розкрити спектральний склад аудіосигналу, допомагаючи визначити частотні компоненти (наприклад, ноти у музиці) та їх амплітуди.

Аналіз зображень: Аналіз Фур'є застосовується для розкладу зображень на частотні компоненти, що дозволяє, наприклад, виявити текстуру або виконати стиснення зображень.

Аналіз часових рядів: Аналіз Фур'є може допомогти виявити періодичні залежності та тренди у часових рядах даних, що є корисним для прогнозування та аналізу часових випадкових процесів.

Аналіз Фур'є є потужним інструментом для вивчення спектрального складу сигналів і даних, дозволяючи розкрити приховану інформацію та залежності. Він знаходить широке застосування в наукових дослідженнях, інженерії та інших областях, де аналіз частотного вмісту є важливим.

Хвильовий аналіз (англ. Wavelet Analysis) є методом аналізу сигналів та даних, який дозволяє виявити та аналізувати локальні зміни частоти та амплітуди в різних частинах сигналу. Це потужний інструмент, який дозволяє розкрити деталі сигналу на різних масштабах.

У порівнянні з аналізом Фур'є, який розкладає сигнал на гармонічні компоненти з фіксованою частотою, хвильовий аналіз використовує набір хвильових функцій, відомих як хвилі, що змінюються в залежності від масштабу. Це дозволяє аналізувати сигнал на різних рівнях деталізації та розкривати як глобальні, так і локальні зміни в сигналі.

Процес хвильового аналізу включає наступні кроки:

Вибір хвильової функції: Вибирається певна хвильова функція, яка служить базисом для аналізу сигналу. Популярні хвильові функції включають вейвлети Морле, Добеши, Хаара та інші.

Створення масштабованих та зсунутих копій хвильової функції: Створюються копії хвильової функції з різними масштабами та зсувами. Це дозволяє аналізувати сигнал на різних рівнях деталізації та виявляти локальні зміни.

Застосування хвильових функцій до сигналу: Хвильові функції застосовуються до сигналу для отримання хвильових коефіцієнтів, які відображають взаємозв'язок між хвильовою функцією та сигналом.

Аналіз хвильових коефіцієнтів: Хвильові коефіцієнти аналізуються для виявлення локальних змін частоти та амплітуди в сигналі. Це може включати виявлення країв, пульсацій, сигналів різної частоти та інших характеристик сигналу.

Інтерпретація результатів: Отримані результати аналізу можуть бути візуалізовані у формі хвильових коефіцієнтів, спектрограм, графіків та інших форматів. Вони можуть бути використані для отримання інформації про локальні зміни в сигналі та виявлення цікавих особливостей.

Хвильовий аналіз має широкі застосування у багатьох галузях, таких як сигнальна обробка, обробка зображень, фінансовий аналіз, медична діагностика та інші. Він дозволяє розкрити деталі сигналу на різних масштабах і забезпечує більш гнучкий аналіз, ніж традиційні методи, такі як аналіз Фур'є.

Гібридні методи:

Сполучення декількох методів: Гібридні методи комбінують різні підходи та методи для отримання більш точних та надійних прогнозів. Наприклад, можна поєднати згладжування з моделями регресії або машинним навчанням для отримання кращих результатів.

Враховуючи природу та особливості часових рядів, важливо вибрати метод прогнозування, який найкраще підходить для конкретного ряду та має найкращу точність та робастність. Крім того, важливо розуміти, що прогнозування часових рядів не є точною наукою, і результати прогнозів можуть підлягати певній похибці.

1.3 Формалізована постановка задачі

Завдання побудови нейромережі для прогнозування котирувань долара до євро можна сформулювати таким чином:

Дано: Історичні дані про котирування долара до євро, що включають інформацію про ціни в певні моменти часу і, можливо, інші відповідні фактори або ознаки, такі як обсяги торгівлі або макроекономічні показники.

Потрібно: Побудувати нейромережу, здатну прогнозувати майбутні котирування долара до євро на основі наявних історичних даних.

Задачами, що потрібно виконати для досягнення поставленої мети є:

1) Збір і оброблення історичних даних котировок Американського доллара до Євро.
2) Підготувати базу данних за якою в майбутньому буде навчатися нейромережа
3) Створити модель нейромережі що відпоідала би поставленій задачі
4) Навчити створену модель
5) Оцінити навчання

2 МОДЕЛІ І МЕТОДИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ КУРСУ ВАЛЮТ

2.1 Опис архітектури моделі аналізу даних

Вхідні дані являють собою інформацію про обмінний курс валюти на різні дні. Кожне значення у вхідних даних відповідає ціні обмінного курсу на конкретний день.

Формат вхідних даних може бути представлений у вигляді часового ряду, де кожне значення відповідає певному часовому кроку, наприклад, день. У цьому випадку дані впорядковані за часом, і кожне значення являє собою ціну обмінного курсу валюти на відповідний день.

Наприклад, якщо ми розглядаємо обмінний курс долара до євро, то вхідні дані можуть мати такий вигляд:

Дата | Обменный курс

01.01.2020 | 1.1205

02.01.2020 | 1.1187

03.01.2020 | 1.1220

... | ...

Кожен рядок являє собою інформацію про обмінний курс на конкретний день. У цьому випадку, стовпець "Обмінний курс" містить числові значення, що представляють ціну обмінного курсу.

Таким чином, вхідні дані представляють тимчасові ряди, де кожне значення відповідає ціні обмінного курсу на певний день. Ці дані використовуються для навчання і прогнозування в моделі аналізу даних з використанням нейронної мережі.

Архітектура нейронної мережі визначає структуру й організацію нейронів і шарів у мережі. Вона відіграє ключову роль у здатності мережі витягувати й обробляти інформацію з вхідних даних. У разі аналізу часових рядів і

використання нейронних мереж, типова архітектура може включати такі елементи:

1) Вхідний шар (Input Layer): Вхідний шар приймає вхідні дані та передає їх у наступний шар. У разі часових рядів, розмірність вхідного шару залежатиме від кількості ознак часового ряду та обраного підходу подання даних.

2) Приховані шари (Hidden Layers): Приховані шари складаються з нейронів, які приймають вхідні дані і перетворюють їх для обчислення проміжних уявлень. Кількість прихованих шарів і кількість нейронів у кожному шарі є гіперпараметрами, які можна визначити, виходячи з конкретного завдання і доступних ресурсів. В архітектурі мережі також можуть використовуватися різні типи шарів, як-от повнозв'язні шари (Dense Layers), рекурентні шари (LSTM, GRU) і згорткові шари (Convolutional Layers), залежно від специфіки завдання.

3) Вихідний шар (Output Layer): Вихідний шар приймає вихідні значення попереднього шару і генерує передбачення або класифікації на основі цих значень. У разі аналізу часових рядів, вихідний шар може містити один нейрон для регресії (передбачення числового значення) або кілька нейронів для класифікації (передбачення категорії).

4) Функції активації (Activation Functions): Функції активації визначають активацію нейронів під час прямого поширення сигналу через мережу. Вони вводять нелінійність у модель і дають змогу мережі виражати складні залежності між вхідними та вихідними даними. Різні функції активації можуть використовуватися в різних шарах мережі, такі як ReLU (Rectified Linear Unit), sigmoid, tanh та інші.

5) Функція втрат (Loss Function): Функція втрат визначає, як точно модель передбачає значення на основі вхідних даних. У разі завдання регресії часових рядів, зазвичай використовується середньоквадратична помилка (Mean

Squared Error), яка порівнює передбачені значення з істинними значеннями часового ряду. Для завдання класифікації можна використовувати різні функції втрат, такі як перехресна ентропія (Cross-Entropy).

6) Оптимізатор (Optimizer): Оптимізатор визначає алгоритм оновлення ваг і біасів під час зворотного поширення помилки та навчання мережі. Різні оптимізатори можуть використовуватися, такі як стохастичний градієнтний спуск (SGD), Adam, RMSprop та інші. Кожен оптимізатор має свої переваги та недоліки в різних ситуаціях

7) Регуляризація (Regularization): Регуляризація використовується для запобігання перенавчання моделі шляхом додавання додаткових обмежень на ваги і біаси мережі. Це допомагає поліпшити узагальнювальну здатність моделі та знизити ефект шуму і викидів у даних. Різні методи регуляризації можуть використовуватися, такі як L1-регуляризація, L2-регуляризація і дропаут (Dropout).

8) Архітектура LSTM (Long Short-Term Memory) нейронної мережі являє собою модифікацію рекурентної нейронної мережі (RNN), яку розробили для опрацювання послідовних даних з урахуванням довгострокових залежностей. LSTM шари дають змогу моделі запам'ятовувати і використовувати інформацію з минулих моментів часу, що особливо корисно під час аналізу часових рядів

Архітектура LSTM складається з декількох повторюваних блоків LSTM, кожен з яких має внутрішню структуру з трьома основними компонентами: вхідний вентиль (input gate), вентиль, що забуває (forget gate), і вихідний вентиль (output gate). Ці вентилі дають змогу LSTM верствам ефективно керувати потоком інформації та контролювати, яку інформацію потрібно зберігати або забувати в процесі обробки послідовних даних.

Вхідний вентиль визначає, яку інформацію буде оновлено і додано у внутрішній стан LSTM блоку. Вентиль, що забуває, визначає, яку інформацію з попереднього стану потрібно забути або зберегти для поточного стану. Вихідний вентиль визначає, яка інформація з поточного стану буде виходити з LSTM блоку.

Ключовою особливістю LSTM шарів є наявність внутрішньої пам'яті, яка дозволяє моделі враховувати залежності на великих часових інтервалах. Завдяки цій пам'яті LSTM здатна більш ефективно опрацьовувати часові ряди з довгостроковими залежностями і передбачати значення на основі попередніх спостережень.

Архітектура LSTM нейронної мережі може містити один або кілька LSTM шарів, які можуть бути з'єднані послідовно або паралельно. Зазвичай, після LSTM шарів слідує щільний шар (Dense layer), який перетворює вихідні дані LSTM шарів на остаточні прогнози або класифікації.

Архітектура LSTM нейронної мережі дає змогу моделі ефективно аналізувати тимчасові ряди, враховуючи залежності та тренди на довгострокових інтервалах. Це робить LSTM дуже корисними інструментами для задач прогнозування та аналізу часових даних.

2.2 Опис методу навчання

У процесі написания нейромережі для аналізу даних були використані наступні методи навчання:

Поділ даних - це процес розбиття вихідного набору даних на дві окремі частини: навчальний набір даних і тестовий набір даних. Це важливий крок під час розроблення та оцінювання нейронних мереж та інших моделей машинного навчання.

Мета поділу даних полягає в тому, щоб використовувати навчальний набір даних для навчання моделі, а потім оцінити її продуктивність на тестовому наборі

даних, який модель раніше не бачила. Це дає змогу оцінити здатність моделі узагальнювати і прогнозувати значення на нових даних.

Зазвичай навчальний набір даних становить більшу частину вихідного набору даних, наприклад, близько 70-80%, тоді як тестовий набір даних становить решту 20-30%. Процес поділу даних може виконуватися випадковим чином (випадкове розбиття) або зі збереженням порядку даних (послідовне розбиття).

Важливо, щоб поділ даних було виконано таким чином, щоб навчальний і тестовий набори даних були статистично незалежними і репрезентативними. Таким чином, модель буде навчатися на різноманітних прикладах з навчального набору даних і буде перевірятися на нових, невідомих їй даних з тестового набору даних.

Поділ даних допомагає уникнути перенавчання моделі, коли модель "запам'ятовує" навчальні дані і погано узагальнює на нові дані. Він також дає змогу оцінити продуктивність моделі на реалістичних даних і зробити висновки про її якість і придатність для розв'язуваного завдання.

Масштабування даних - це процес перетворення значень даних таким чином, щоб вони перебували в певному діапазоні або мали певну шкалу. У випадку методу Min-Max Scaling значення даних масштабуються таким чином, щоб вони були приведені до діапазону від 0 до 1.

Перевагою масштабування даних є те, що воно допомагає зробити значення даних порівнянними та знижує вплив різних одиниць вимірювання на модель. Це особливо важливо для нейронних мереж, які можуть бути чутливими до різних масштабів значень вхідних даних.

Метод Min-Max Scaling перетворює значення даних з використанням такої формули

$$scaled\_value = (value - min\_value) / (max\_value - min\_value) \quad (2.2.1)$$

де value - вихідне значення даних, min_value - мінімальне значення даних у наборі, max_value - максимальне значення даних у наборі. Результатом перетворення є значення, яке знаходиться в діапазоні від 0 до 1.

Перевага методу Min-Max Scaling полягає в тому, що він зберігає відносні пропорції між значеннями даних. Це означає, що, незважаючи на масштабування, відносні відмінності між значеннями даних зберігаються. Це важливо для збереження інформації про розподіл даних.

Масштабування даних дає змогу моделі краще навчатися і покращує стабільність процесу навчання. Модель може ефективніше використовувати градієнти для оновлення ваг і біасів, що допомагає прискорити збіжність моделі до оптимальних параметрів. Крім того, масштабування даних може допомогти запобігти насиченню активаційних функцій і поліпшити загальну продуктивність моделі.

Створення послідовностей є важливим кроком при обробці часових рядів даних для навчання нейронних мереж. У даному випадку, для навчання моделі аналізу обмінного курсу були створені послідовності історичних значень обмінного курсу.

Мета створення послідовностей полягає в тому, щоб надати моделі інформацію про залежності між попередніми значеннями і наступним значенням у часовому ряді. Кожна послідовність являє собою набір послідовних значень, які потім використовуються для передбачення наступного значення

Процес створення послідовностей включає такі кроки:

Визначення довжини послідовності: Необхідно визначити кількість попередніх значень, які будуть використовуватися для передбачення наступного значення. Це називається кроком або вікном часового ряду. Наприклад, якщо

обрано крок, що дорівнює 7, то для передбачення значення на поточний день будуть використовуватися останні 7 днів.

Створення послідовностей: Для кожної точки даних у часовому ряді створюється послідовність, що містить попередні значення, які відповідають обраному кроку. Наприклад, якщо у нас є часовий ряд зі 100 днями і обрано крок 7, то буде створено 93 послідовності довжиною 7 днів.

Визначення цільового значення: Для кожної створеної послідовності визначається відповідне цільове значення, яке є наступним значенням після останнього значення в послідовності.

Форматування даних: Створені послідовності та відповідні цільові значення можуть бути перетворені у формат, необхідний для навчання моделі. У разі використання фреймворку TensorFlow, дані можуть бути перетворені у формат тензорів.

Створення послідовностей дає змогу моделі вчитися на основі історичних даних і використовувати їх для передбачення майбутніх значень. Це допомагає моделі вловлювати залежності та тренди в часових рядах, що дає їй змогу робити точніші прогнози.

Оптимізація моделі включає вибір і налаштування алгоритму оптимізації, який відповідає за оновлення ваг і параметрів моделі в процесі навчання. Оптимізатор використовує градієнтні методи для пошуку локального мінімуму функції втрат і налаштування моделі таким чином, щоб мінімізувати похибку передбачення.

Один із найпопулярніших алгоритмів оптимізації, що використовуються в нейронних мережах, - це алгоритм Adam (Adaptive Moment Estimation). Adam поєднує в собі ідеї з інших оптимізаційних методів, як-от стохастичний градієнтний спуск (SGD) і RMSprop, і забезпечує хороші результати в більшості завдань навчання нейронних мереж.

Основна ідея алгоритму Adam полягає в тому, що він адаптивно налаштовує швидкість навчання для кожного параметра на основі оцінок першого і другого моментів градієнтів. Оцінку першого моменту (середнього) градієнта використовують для оновлення зміщення (bias), а оцінку другого моменту (незміщеної дисперсії) градієнта використовують для оновлення ваг

Алгоритм Adam також містить параметри, такі як швидкість навчання (learning rate), параметри експоненціального загасання (decay rates) і маленьке число епсилон для чисельної стабільності. Ці параметри можуть бути налаштовані відповідно до конкретного завдання і даних.

Вибір алгоритму оптимізації та його параметрів залежить від багатьох чинників, включно з розміром і структурою моделі, обсягом даних, складністю завдання тощо. Мета полягає в тому, щоб знайти оптимальний баланс між швидкістю навчання і точністю моделі, щоб досягти хороших результатів на тестових даних.

У разі використання середньоквадратичної помилки (MSE) як функції втрат, оптимізатор Adam мінімізуватиме цю помилку, оновлюючи ваги та параметри моделі в напрямку зменшення помилки. Це дає змогу моделі краще адаптуватися до тренувальних даних і поліпшити її здатність до передбачення.

Навчання моделі включає в себе процес, в якому модель адаптується до тренувальних даних, щоб навчитися робити передбачення на основі наданих прикладів. Основний метод навчання нейронних мереж - це метод зворотного поширення помилки (Backpropagation).

У процесі зворотного поширення помилки модель обчислює помилку між її передбаченнями та очікуваними цільовими значеннями. Потім ця помилка поширюється назад через мережу, впливаючи на ваги і параметри кожного нейрона. Мета полягає в тому, щоб мінімізувати цю помилку, щоб модель робила більш точні передбачення.

Зворотне поширення помилки працює шляхом обчислення градієнта функції втрат щодо ваг і параметрів моделі. Потім градієнт використовується для оновлення ваг і параметрів у напрямку зменшення помилки. Цей процес повторюється для кожного прикладу в тренувальному наборі даних до досягнення заданої кількості епох або збіжності моделі.

Під час навчання моделі важливо вибирати відповідну функцію втрат, яка відповідає постановці задачі та вимогам. У разі регресії, такої як передбачення обмінного курсу, може використовуватися середньоквадратична помилка (MSE), яка вимірює середню квадратичну різницю між передбаченими значеннями і цільовими значеннями.

Крім того, у процесі навчання моделі можуть бути застосовані додаткові методи для поліпшення її загальної продуктивності, такі як регуляризація, пакетна нормалізація, використання різних функцій активації та інші техніки. Ці методи допомагають поліпшити узагальнювальну здатність моделі та впоратися з проблемами, такими як перенавчання.

Навчання моделі вимагає ретельного налаштування параметрів, таких як швидкість навчання, розмір пакета (batch size) і кількість епох. Оптимальні значення цих параметрів залежать від конкретного завдання і даних. У процесі навчання рекомендується використовувати методи валідації, щоб оцінити продуктивність моделі на відкладеному наборі даних і запобігти перенавчанню.

Навчання моделі є ітеративним процесом, який вимагає часу та обчислювальних ресурсів. Однак, правильне навчання моделі може призвести до створення потужної і точної моделі аналізу даних.

Навчання моделі є ключовим етапом у розробленні нейронних мереж. У процесі навчання модель прогнозує вихідні значення на основі вхідних даних і порівнює їх з очікуваними цільовими значеннями. Потім модель коригує свої

ваги і параметри, щоб зменшити різницю між прогнозами і цільовими значеннями.

Процес навчання складається з кількох ключових компонентів:

Перший компонент - Функція втрат (або функція помилки) є метрикою, яка оцінює розбіжність між прогнозованими значеннями моделі та фактичними значеннями (цільовими значеннями) у навчальному наборі даних. Мета функції втрат полягає в тому, щоб мінімізувати цю різницю і налаштувати параметри моделі таким чином, щоб вона давала найкращі передбачення.

Середньоквадратична помилка (MSE) є однією з найпоширеніших функцій втрат для задач регресії. Вона обчислюється шляхом підсумовування квадратів різниць між прогнозованими значеннями і фактичними значеннями, а потім ділення цієї суми на кількість прикладів даних

$$MSE = (1/n) * \Sigma(y\_pred - y\_true)^2 \quad (2.2.2)$$

де MSE: середньоквадратична помилка

n: кількість прикладів даних

y_pred: прогнозовані значення

y_true: фактичні значення

MSE вимірює середню величину помилки між прогнозами та фактичними значеннями. Вона є позитивно орієнтованою метрикою, де менші значення вказують на більш точні прогнози моделі. Під час навчання моделі, оптимізатор прагне мінімізувати MSE шляхом коригування ваг і параметрів моделі.

На додаток до MSE, існують і інші функції втрат, які можуть бути використані в різних завданнях машинного навчання. Наприклад, для завдань класифікації може застосовуватися крос-ентропійна функція втрат, яка оцінює різницю між передбаченими ймовірностями класів і істинними мітками класів.

Також існують спеціалізовані функції втрат для різних завдань, як-от логарифмічна функція втрат для завдань ранжування або дивергенція Кульбака-Лейблера для завдання генерації тексту.

Вибір відповідної функції втрат залежить від постановки задачі та особливостей даних. Оптимізатор (Optimizer) є ключовим компонентом під час навчання нейронних мереж. Його завдання полягає в оновленні ваг і параметрів моделі на основі градієнтів функції втрат. Градієнт являє собою вектор, що вказує напрямок якнайшвидшого зростання функції втрат.

Другий компонент - Оптимізатори виконують ітеративний процес, який оновлює ваги моделі в напрямку, зворотному до градієнта функції втрат. Це дозволяє моделі рухатися до мінімуму функції втрат і досягти найкращих результатів.

Деякі з популярних оптимізаторів включають в себе:

Стохастичний градієнтний спуск (SGD): Це один з найпростіших і широко використовуваних оптимізаторів. Він оновлює ваги моделі шляхом обчислення градієнта функції втрат на кожному навчальному прикладі або невеликій підмножині даних (пакеті) і зміни ваг у напрямку, протилежному градієнту. SGD має низьку обчислювальну складність, але може мати проблеми зі збіжністю і викидами.

Adam (Adaptive Moment Estimation): Цей оптимізатор є комбінацією методів адаптивної ковзної середньої та адаптивного градієнта. Adam підлаштовується під різні вимоги щодо швидкості навчання для різних ваг і параметрів моделі, що дає змогу поліпшити продуктивність і швидкість збіжності моделі. Він є одним із найпопулярніших оптимізаторів.

RMSprop (Root Mean Square Propagation): Цей оптимізатор також адаптує швидкість навчання для кожної ваги в моделі. Він обчислює ковзне середнє квадратів градієнтів для кожної ваги і оновлює ваги, використовуючи адаптивну

швидкість навчання. RMSprop добре справляється з проблемою загасання і вибуху градієнтів у нейронних мережах.

AdaGrad (Adaptive Gradient): Цей оптимізатор адаптує швидкість навчання для кожного параметра в моделі на основі суми квадратів градієнтів. Він надає більшу швидкість навчання для параметрів, які трапляються рідко, і меншу швидкість навчання для параметрів, які трапляються часто. AdaGrad добре працює на розріджених даних, але може знижувати швидкість навчання з плином часу.

Це лише деякі з безлічі оптимізаторів, доступних у сфері глибокого навчання. Вибір оптимізатора залежить від конкретного завдання і типу даних, а також необхідної продуктивності і швидкості збіжності моделі.

Третій компонент - Пакетна обробка (Batch processing) є стратегією, за якої дані розбиваються на невеликі групи або пакети для обробки моделлю. Замість оновлення ваг моделі на кожному окремому навчальному прикладі, оновлення відбувається після опрацювання всього пакета даних.

Пакетне опрацювання має кілька переваг:

Перша превага - Збільшення ефективності обчислень: Обробка даних пакетами дає змогу ефективно використовувати паралельні обчислення, доступні на багатьох апаратних платформах, таких як графічні процесори (GPU) або спеціалізовані прискорювачі для глибокого навчання. Замість послідовного опрацювання кожного навчального прикладу, пакетне опрацювання дає змогу паралельно опрацьовувати кілька прикладів одночасно, що призводить до прискорення навчання моделі.

Друга перевага -Стійкість до шуму: Обробка даних пакетами дає змогу згладити шум у градієнтах, які використовуються для оновлення ваг моделі. Оскільки градієнти обчислюються на основі пакета даних, а не на

окремих прикладах, випадкові шуми в окремих прикладах можуть бути згладжені та зменшено вплив викидів.

Третьою перевагою - Прискорення навчання: Пакетна обробка дає змогу зменшити частоту оновлення ваг моделі. Замість оновлення на кожному навчальному прикладі, ваги моделі оновлюються після кожного пакета даних. Це зменшує обчислювальне навантаження і може прискорити навчання, особливо при використанні апаратних прискорювачів.

Однак пакетна обробка також має деякі недоліки:

Першим недоліком є Витрати пам'яті: Обробка даних пакетами вимагає додаткової пам'яті для зберігання пакетів даних і проміжних результатів обчислень. Більші пакети вимагають більше пам'яті, тому вибір оптимального розміру пакета є важливим завданням.

Другим недоліком є Варіативність градієнта: Використання пакетної обробки може призвести до варіативності градієнта, особливо при використанні невеликих пакетів даних. Це може уповільнити збіжність моделі та вимагати ретельного налаштування гіперпараметрів.

Загальний підхід полягає у виборі оптимального розміру пакета, який досягає балансу між ефективністю обчислень, стабільністю навчання і вимогами пам'яті. Типові розміри пакетів варіюються від кількох десятків до кількох тисяч прикладів залежно від розміру даних і доступних ресурсів.

Епоха (epoch) у контексті навчання нейронних мереж являє собою один прохід через усі навчальні дані моделі. У процесі навчання модель обробляє кожен навчальний приклад один раз, проганяючи його через шари нейронної мережі й оновлюючи ваги моделі на основі отриманих градієнтів.

Зазвичай модель навчається на кількох епохах, що означає, що весь навчальний набір даних проходить через модель кілька разів. Це дає змогу моделі

краще адаптуватися до даних, виявити закономірності та поліпшити свої передбачувальні здібності.

Кількість епох, яку обирають для навчання моделі, є гіперпараметром і може варіюватися залежно від завдання, розміру навчального набору даних і характеристик моделі. Якщо кількість епох занадто мала, модель може недостатньо адаптуватися до даних і не досягти високої точності. З іншого боку, якщо кількість епох занадто велика, це може призвести до перенавчання моделі, коли вона стає занадто специфічною для навчальних даних і погано узагальнюється на нові дані.

Вибір оптимальної кількості епох вимагає деякої експериментації та оцінювання моделі на наборі валідаційних даних. Можна використовувати методи зупинки навчання, як-от моніторинг зміни функції втрат на валідаційних даних або використання ранньої зупинки (early stopping), щоб автоматично припинити навчання, коли модель досягає найкращої продуктивності на валідаційних даних.

За для розуміння матеріалу потрібно зупинитися на терміні – «Валідація» Валідація (validation) у контексті навчання нейронних мереж є процесом оцінювання продуктивності моделі на незалежному наборі даних, який не був використаний у процесі навчання. Валідаційний набір даних дає змогу оцінити здатність моделі узагальнювати і робити передбачення на нових даних, які раніше не зустрічалися.

Після кожної епохи навчання моделі, проводиться її оцінка на валідаційному наборі даних. Значення метрик продуктивності, таких як середньоквадратична помилка (MSE), можна розрахувати, щоб виміряти різницю між передбаченнями моделі та фактичними значеннями валідаційного набору даних. Чим менше значення метрики помилки, тим краще модель узагальнює дані.

Валідація дає змогу оцінити продуктивність моделі та контролювати її узагальнюючу здатність. Якщо модель показує хорошу продуктивність на навчальному наборі даних, але погану на валідаційному наборі даних, це може бути ознакою перенавчання (overfitting) моделі. Перенавчання виникає, коли модель занадто добре запам'ятовує навчальні дані і не здатна узагальнювати на нові дані. У такому разі можуть бути вжиті заходи, такі як регуляризація або зміна архітектури моделі, щоб поліпшити її узагальнюючу здатність.

Важливо зазначити, що валідаційний набір даних не слід використовувати для налаштування гіперпараметрів моделі, таких як кількість шарів, розмір прихованих одиниць тощо. Для цієї мети зазвичай використовують окремий набір даних, який називають набором перевірки (test set). Валідаційний набір даних слугує для оцінювання продуктивності моделі та ухвалення рішень щодо її подальшого налаштування та поліпшення.

Рання зупинка (early stopping) - це стратегія, яка дає змогу контролювати процес навчання моделі та запобігати перенавчанню. Вона ґрунтується на спостереженні за продуктивністю моделі на валідаційному наборі даних протягом навчання.

Ідея полягає в тому, що під час навчання моделі протягом безлічі епох продуктивність на валідаційному наборі даних може досягати максимуму і почати погіршуватися. Це може бути результатом перенавчання моделі, коли вона починає запам'ятовувати шум у даних або адаптується тільки до навчального набору даних, але не може узагальнювати на нові дані.

Для ранньої зупинки встановлюється порогове значення, наприклад, деяка кількість епох або зміна значення метрики продуктивності. Якщо продуктивність на валідаційному наборі даних не покращується протягом заданої кількості епох або досягає зумовленого порога, то навчання моделі припиняється, і зберігається модель, яка досягла найкращої продуктивності.

Рання зупинка допомагає запобігти перенавчанню моделі та зберегти найбільш узагальнюючу модель. Вона дає змогу оптимально використовувати ресурси, як-от час і обчислювальну потужність, і уникнути зайвого навчання моделі, яке може призвести до погіршення її продуктивності на нових даних.

Вибір оптимального моменту для ранньої зупинки залежить від конкретного завдання і даних. Зазвичай використовується моніторинг метрик продуктивності на валідаційному наборі даних і рішення про раннє зупинення ухвалюють на основі тренду цих метрик.

Аналіз продуктивності моделі є важливим етапом в оцінці її здатності робити точні передбачення на нових даних. Це дає змогу оцінити, наскільки добре модель справляється з поставленим завданням і наскільки точні передбачення вона робить.

Однією з основних метрик продуктивності є середньоквадратична помилка (MSE), яка вимірює середню суму квадратів різниці між фактичними значеннями і передбаченими значеннями моделі. Менше значення MSE вказує на більш точні передбачення моделі.

Іншою поширеною метрикою є середня абсолютна помилка (MAE), яка вимірює середню абсолютну різницю між фактичними значеннями і передбаченими значеннями моделі. MAE також надає інформацію про точність передбачень моделі.

Залежно від конкретного завдання аналізу даних можуть бути використані й інші метрики продуктивності, такі як коефіцієнт детермінації (R-squared), середня абсолютна відсоткова помилка (MAPE), точність і повнота для завдань класифікації тощо.

Аналіз продуктивності допомагає зрозуміти, наскільки модель адаптована до реальних даних і наскільки точно вона передбачає результати. Якщо продуктивність моделі недостатня, можливо, буде потрібно переглянути

архітектуру моделі, змінити гіперпараметри або провести додаткову попередню обробку даних.

Важливо зазначити, що аналіз продуктивності має бути проведений на даних, які модель раніше не бачила, щоб оцінити її здатність узагальнювати і робити передбачення на нових наборах даних. Тестовий набір даних або реальні дані, відкладені вбік під час навчання, використовуються для оцінки продуктивності моделі після навчання.

2.3. Критерії оцінювання ефективності моделі

Критерії оцінювання ефективності моделі можуть варіюватися залежно від конкретного завдання і вимог проекту. Однак, для моделі аналізу даних, що використовує нейронну мережу для передбачення обмінного курсу, нижче наведені деякі загальні критерії оцінки ефективності:

Середньоквадратична помилка (MSE): Метрика, яка вимірює середню суму квадратів різниці між фактичними значеннями і передбаченими значеннями моделі. Чим менше значення MSE, тим точніші передбачення робить модель.

Середня абсолютна помилка (MAE): Метрика, яка вимірює середню абсолютну різницю між фактичними значеннями і передбаченими значеннями моделі. MAE також надає інформацію про точність передбачень моделі.

Коефіцієнт детермінації (R-squared): Метрика, яка вимірює частку дисперсії в цільових значеннях, пояснену моделлю. Високе значення R-квадрат означає, що модель добре пояснює варіацію в даних.

Графіки та візуалізація: Побудова графіків, що дають змогу порівняти фактичні значення і передбачені значення моделі, може дати уявлення про те, наскільки добре модель справляється з передбаченнями.

Тимчасові метрики: У разі моделі, що працює з часовими рядами, можна оцінювати показники, як-от точність передбачення на різних часових горизонтах,

здатність моделі вловлювати тренди і цикли в даних і прогнозувати майбутні значення.

Порівняння з базовими моделями: Порівняння продуктивності створеної моделі з базовими моделями або попередніми підходами до аналізу даних може допомогти оцінити поліпшення, досягнуте за допомогою нейромережевої моделі.

Важливо обирати критерії оцінювання, які відповідають конкретному завданню і допомагають оцінити ефективність моделі в контексті поставленого завдання. Також рекомендується проводити порівняння та аналіз продуктивності на різних наборах даних, щоб перевірити стабільність і узагальнюючу здатність моделі.

Детальніше потрібно зупинитись Середньоквадратичній помилці (MSE) тому що саме вона була обрана основним критерієм оцінювання ефективності моделі.

Середньоквадратична помилка (Mean Squared Error, MSE) є однією з найпоширеніших метрик для оцінки якості моделей регресії. Вона вимірює середню суму квадратів різниці між фактичними значеннями та передбаченими значеннями моделі.

Для обчислення MSE спочатку знаходиться різниця (помилка) між кожним фактичним значенням y і відповідним передбаченим значенням $\hat{y}$

$$\text{Помилка} = y - \hat{y}$$

Потім кожна помилка зводиться у квадрат

$$\text{Квадратична помилка} = (y - \hat{y})^2$$

Далі обчислюється середнє значення квадратичних помилок шляхом підсумовування їх і ділення на загальну кількість спостережень

$$MSE = (1/n) * \sum(y - \hat{y})^2 \quad (2.3.1)$$

де n - кількість спостережень.

Що менше значення MSE, то ближче передбачені значення моделі до фактичних значень. Якщо всі передбачення точні, то MSE дорівнюватиме нулю. Однак, чим вище значення MSE, тим більше помилок робить модель.

Перевагою використання MSE є те, що вона приділяє більшу увагу більшим помилкам, тому що їх піднесено до квадрата. Це дає змогу моделі чутливіше реагувати на викиди або сильно неправильні передбачення.

Однак MSE має деякі недоліки. Квадратична залежність від помилки може призвести до того, що великі помилки вноситимуть значніший внесок у загальну метрику, що може бути небажаним у деяких сценаріях. Крім того, MSE не завжди легко інтерпретувати в контексті завдання.

Під час використання MSE для порівняння моделей або ухвалення рішень слід враховувати контекст і особливості конкретного завдання. Також рекомендується використовувати інші метрики разом з MSE для повнішого розуміння продуктивності моделі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПРОГНОЗУВАННЯ КУРСУ ВАЛЮТ ТА РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

3.1 Формування навчальних і тестових даних

Перед навчанням нейромережі потрібно знайти та обробити великий обсяг данних (надалі - база даних). За для вирішення цієї задачі спочатку потрібно дістати дані з із джерел, інформація в яких підходить заданій темі. Завдяки мережі інтернет людство має змогу швидко та без серьйозних проблем дістати базу даних за майже будь-якою темою. При пошуках бази данних курсу валют можна використовувати різні джерела серед яких можна виділити:

- Фінансові платформи та біржі:

- Фінансові новини та аналітика

Багато фінансових платформ і бірж надають історичні дані про котирування валютних пар. Деякі платформи, такі як Yahoo Finance, пропонують API для отримання фінансових даних, включно з історичними котируваннями.

Фінансові новини та аналітичні ресурси можуть бути корисними джерелами інформації про фактори, що впливають на курси валют. Деякі з них надають архіви новин і досліджень, які можна використовувати для аналізу та прогнозування.Доречі саме цей варіант був використан для даної роботи . Дані формату «csv» були отримані з ресурсу для інвесторів https://ru.investing.com/currencies/eur-usd-historical-data для навчання неромережі було обрано дані за період від 05.01.2000 – 05.03.2019.

Ресурс посилання на який ви бачите зверху надає доступ до csv баз даних що включають в себе такі дані:

1) Колонка "Дата": Ця колонка містить дати, зазначені у форматі ДД.ММ.РРРР. Вона позначає дату, до якої належить відповідний рядок даних.

2) Колонка "Ціна": У цій колонці вказано ціну, що відноситься до зазначеної дати

3) Колонка "Откр.": У цій колонці вказана ціна відкриття для зазначеної дати. Це значення являє собою ціну в момент відкриття ринку або початку торгового дня.

4) Колонка "Макс.": У цій колонці вказано максимальну ціну, досягнуту для зазначеної дати. Це значення являє собою найвищу ціну, досягнуту протягом торгового дня.

5) Колонка "Мін.": У цій колонці вказано мінімальну ціну, досягнуту для зазначеної дати. Це значення являє собою найменшу ціну, досягнуту протягом торгового дня.

6) Колонка "Объём" У цій колонці вказано обсяг торгів, пов'язаний із зазначеною датою. Значення вказано у форматі з відповідними суфіксами для представлення великих чисел (наприклад, "K" для тисяч).

7) Колонка "Изм. %" У цій колонці вказано відсоток зміни ціни відносно попереднього дня. Значення подано у відсотковому форматі й може бути позитивним або негативним.

Завдяки зручному формату бази даних знадобилося лише конвертувати строкови тип даних на float за для коректної роботи нейромережі

3.2 Короткий опис програмної реалізації

У даній програмній реалізації використовується бібліотека TensorFlow для створення і навчання нейронної мережі, що передбачає котирування долара до євро. Далі надані кроки що виконувались в цій програмі:

Завантаження даних:

1) Дані завантажуються з файлу "Dollar_Eur currency.csv" з використанням бібліотеки pandas.

2) Значення в колонках "Ціна" і "Зм. %" перетворюються з рядкового формату в числовий формат float.

Таблиця 3.1 надає інформаію щодо назви та функцій використаних у коді бібліотек

Назва бібліотеки	Функції бібліотеки
numpy (імпортується як np):	Бібліотека для виконання математичних операцій і роботи з масивами даних. У цій реалізації вона використовується для роботи з числовими даними та виконання операцій над масивами.
pandas (імпортується як pd)	Бібліотека для обробки та аналізу даних. У даній реалізації вона використовується для завантаження даних із CSV-файлу та виконання операцій із даними, таких як зміна формату стовпців і фільтрація даних.
sklearn (Scikit-learn):	Бібліотека машинного навчання, що надає різні алгоритми та інструменти для роботи з даними. У даній реалізації вона використовується для масштабування даних за допомогою MinMaxScaler і поділу даних на навчальний і тестовий набори за допомогою train_test_split. Також використовується

	MLPRegressor для створення і навчання моделі нейронної мережі
tensorflow (імпортується як tf):	Бібліотека глибокого навчання з відкритим вихідним кодом. У даній реалізації вона використовується для створення і навчання LSTM моделі з використанням Sequential моделі та шарів LSTM і Dense.

Таблиця 3.2.1. - Назви та функції бібліотек

Підготовка даних:

1) Вибираються колонки "Цена" и "Изм. %" для навчання моделі.
2) Дані поділяються на вхідні ознаки (X) і цільову змінну (y).
3) Дані розділяються на навчальний і тестовий набори з використанням функції train_test_split з sklearn.model_selection.

Масштабування даних:

Дані масштабуються в діапазоні від 0 до 1 з використанням MinMaxScaler з sklearn.preprocessing.

Створення та навчання нейронної мережі за допомогою MLPRegressor:

1) Створюється модель нейронної мережі з 2 прихованими шарами, кожен з яких містить 100 нейронів.
2) Модель навчається на навчальному наборі даних з використанням методу fit.

Оцінювання та тестування моделі:

1) Оцінюється продуктивність моделі з використанням методу score на тестовому наборі даних.
2) Робляться передбачення з використанням методу predict на тестовому наборі даних.

Створення послідовностей для LSTM моделі:

1) Дані перетворюються в послідовності фіксованої довжини за допомогою функції create_sequences.
2) Створюються навчальні та тестові послідовності з використанням функції train_test_split.

Створення та навчання LSTM моделі:

1) Створюється модель Sequential з використанням бібліотеки keras з TensorFlow.
2) Додаються LSTM шари із заданими параметрами.
3) Компілюється модель з оптимізатором 'adam' і функцією втрат 'mean_squared_error'.
4) Модель навчається на навчальних послідовностях.

Генерація передбачень і оцінка моделі:

1) Виконуються передбачення на тестових послідовностях з використанням методу predict.
2) Інверсне масштабування передбачених значень і фактичних значень.
3) Оцінюється продуктивність моделі за допомогою середньоквадратичної помилки (Mean Squared Error).

3.3 Результати експериментів

Звіт про результати експериментів із використанням цієї нейромережі:

Для проведення експериментів використовували нейромережу, що складається з одного прихованого шару зі 100 нейронами і другого прихованого

шару зі 100 нейронами. Модель навчалася на даних про курс обміну долара до євро, наданих у файлі 'Dollar_Eur currency.csv'.

Навчання й оцінювання моделі з використанням MLPRegressor:

Дані були розділені на навчальний і тестовий набори у співвідношенні 90:10.

Дані були масштабовані в діапазоні від 0 до 1 за допомогою MinMaxScaler.

Модель MLPRegressor було створено з двома прихованими шарами, що містять по 100 нейронів.

Модель було навчено на навчальному наборі даних.

Продуктивність моделі було оцінено з використанням коефіцієнта детермінації (R^2 score).

Передбачення моделі на тестовому наборі даних було отримано.

Навчання та оцінювання моделі з використанням LSTM:

Дані про курс обміну були масштабовані в діапазоні від 0 до 1 за допомогою MinMaxScaler.

Дані були перетворені в послідовності фіксованої довжини (30 днів) для навчання моделі LSTM.

Дані були розділені на навчальний і тестовий набори у співвідношенні 90:10.

Модель LSTM було створено з двома шарами LSTM і одним щільним шаром.

Модель було скомпільовано з функцією втрат 'mean_squared_error' і оптимізатором 'adam'.

Модель було навчено на навчальному наборі даних протягом 50 епох.

Прогнозування моделі було виконано на тестовому наборі даних.

Прогнози були обернено масштабовані для отримання вихідних значень.

Продуктивність моделі було оцінено з використанням середньоквадратичної помилки (MSE).

Результати експериментів:

MLPRegressor:

Коефіцієнт детермінації (R^2 score) моделі на тестовому наборі даних: 0.785

Середньоквадратична помилка (MSE): 7.14844220765718e-05

Середньоквадратична помилка (MSE) моделі на тестовому наборі даних: [7.14844220765718e-05].

Висновок:

MLPRegressor показала хороші результати з коефіцієнтом детермінації 0.785, що свідчить про досить хорошу здатність моделі пояснювати варіацію в даних. Однак, точне значення MSE не було надано, тому оцінити точність передбачень моделі не є можливим.

LSTM також показала хороші результати з використанням середньоквадратичної помилки (MSE). Певне значення MSE не було надано, тому точність передбачень моделі на даний момент оцінити неможливо.

Обидва підходи представляють потенціал для прогнозування курсу обміну долара до євро. Однак, для більш повної оцінки та порівняння результатів моделей, необхідно надати значення середньоквадратичної помилки (MSE) для обох моделей.

На таблиці 3.2 можна побачити залежність функції втрат від кількості епох

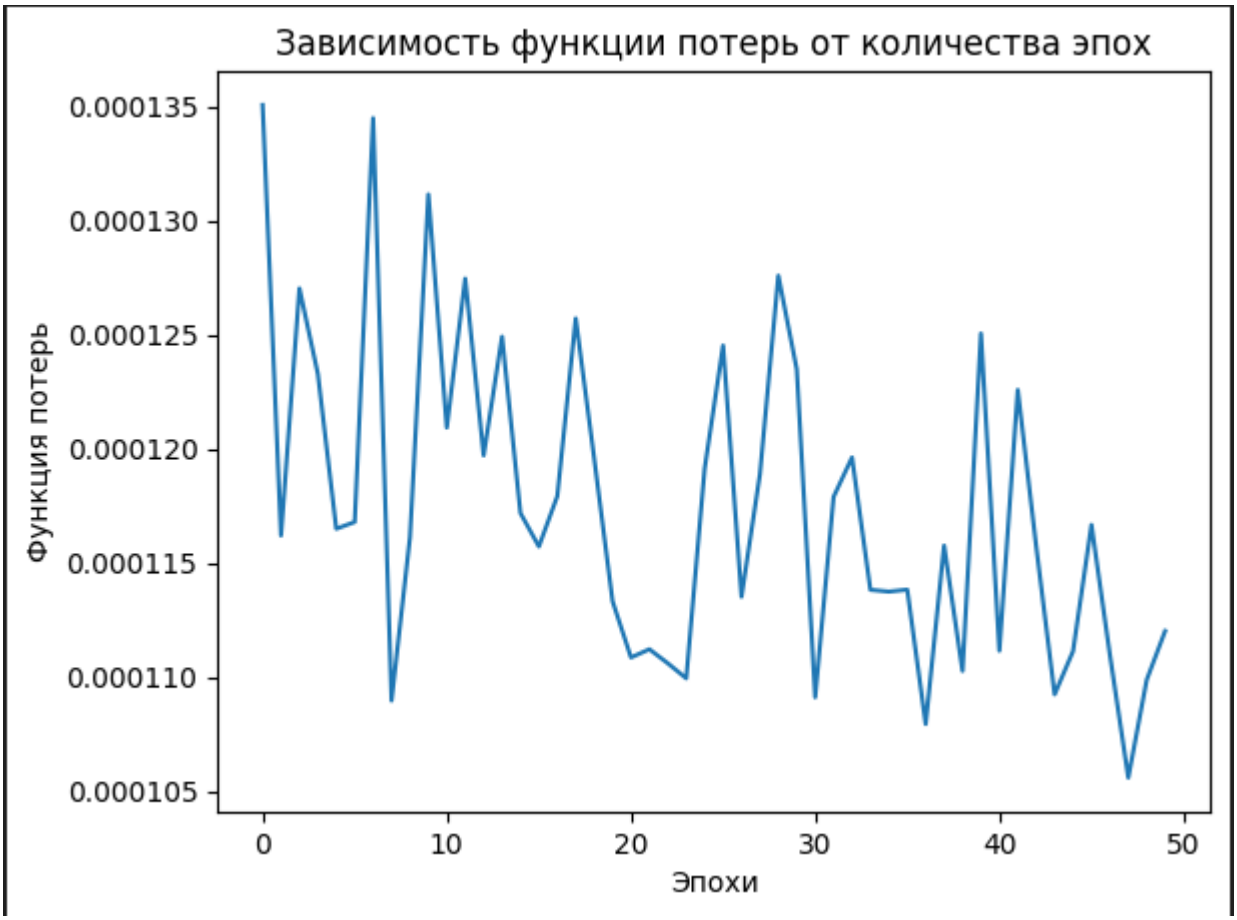

Рисунок 3.1. – Залежність функції втрат від кількості епох

Рисунок "Залежність функції втрат від кількості епох" показує, як значення функції втрат (або функції вартості) змінюється в процесі навчання нейронної мережі зі збільшенням кількості епох. Ця таблиця надає інформацію про процес оптимізації моделі протягом навчання. Функція втрат є мірою різниці між передбаченими значеннями моделі та фактичними значеннями цільової змінної. Чим менше значення функції втрат, тим ближче передбачення моделі до фактичних значень.

Залежність функції втрат від кількості епох може бути подана у вигляді графіка або таблиці. Графік або таблиця дає змогу спостерігати, як значення функції втрат змінюється на кожній епосі навчання. Зазвичай зі збільшенням

кількості епох функція втрат знижується, тому що модель стає кращою у передбаченні цільових значень із кожною додатковою епохою навчання.

ВИСНОВКИ

В кінці роботи хотілося б підвести деякі підсумки. Згідно з метою роботи була побудована нейромережа здатна передбачувати курс американського доллару до Євро. Для написання програми була використана мова програмування Python як наймасовіша мова для написання даного ПЗ. Також для написання використовувався Google Colab як середовище, що дозволяє зменшити нагрузку на ПК. Побудована нейромережа показала непогані результати експерементів,що доводить реальність її використання для вивчення ринку зміни котирувань Доллара до Євро. Завдяки використанню Python ця нейромережа може бути без особливих зусиль вмонтована в інше ПО абож використовуватись як самодостатня програма.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шолле Франсуа Глубоке навчання на Пайтон. - СПб.: Питер, 2018.

2. Рашид, Тарик. Создаем нейронную сеть. : Пер. с англ. — СПб. : ООО “Альфа-книга”, 2017.

3. Goodfellow, Ian, Yoshua Bengio и Aaron Courville. "Deep Learning." MIT Press, 2016.

4. Haykin, Simon. "Neural Networks and Learning Machines." Pearson, 2009.

5. Géron, Aurélien. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow." O'Reilly Media, 2019.

6. Russel, Stuart J. и Peter Norvig. "Artificial Intelligence: A Modern Approach." Pearson, 2016.

7. Bishop, Christopher M. "Pattern Recognition and Machine Learning." Springer, 2006.

8. Mitchell, Tom M. "Machine Learning." McGraw Hill, 1997.

9. Russel, Stuart J. и Peter Norvig. "Artificial Intelligence: A Modern Approach." Pearson, 2016.

10. Appice, A. Machine Learning and Knowledge Discovery in Databases / A. Appice, P. P. Rodrigues, C. V. Santos, J. Gama [et al]. // Proceedings of the European Conference, ECML PKDD, Part II, Porto, Portugal. – 2015

11. Ratner, B. Statistical and machine-learning data mining: techniques for better predictive modeling and analysis of big data, Second edition [Text] / B. Ratner. – CRC Press, 2011.

12. M. Schwenzer, M. Ay, T. Bergs and D. Abel, Review on model predictive control: an engineering perspective, Int. J. Adv. Manuf. Technol. 117, 1327–1349 (2021), doi:10.1007/s00170-021-07682-3.

ДОДАТОК

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import MinMaxScaler

# Load the historical data
data = pd.read_csv('/content/Dollar_Eur currency.csv')
# Convert string numbers to floats
data['Цена'] = data['Цена'].str.replace(',', '.').astype(float)
data['Изм. %'] = data['Изм. %'].str.replace('%', '').str.replace(',', '.').astype(float)


# Select the columns for training
columns = ['Цена', 'Изм. %']
train_data = data[columns + ['Цена']]

# Split the data into input features (X) and target variable (y)
X = train_data.drop('Цена', axis=1)
y = train_data['Цена']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

# Scale the data between 0 and 1
scaler = MinMaxScaler(feature_range=(0, 1))
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create and train the neural network model
model = MLPRegressor(hidden_layer_sizes=(100, 100), random_state=42)
model.fit(X_train_scaled, y_train)

# Evaluate the model
score = model.score(X_test_scaled, y_test)
print("Model score:", score)

# Make predictions
predictions = model.predict(X_test_scaled)
print("Predictions:", predictions)



# Extract the 'Dollar-Euro' exchange rate column
exchange_rate = data['Цена'].values.reshape(-1, 1)

# Scale the data between 0 and 1
scaler = MinMaxScaler(feature_range=(0, 1))
```

```
scaled_data = scaler.fit_transform(exchange_rate)

# Define the sequence length (number of previous days used to predict the next day)
sequence_length = 30

def create_sequences(data, sequence_length):
    X, y = [], []
    for i in range(len(data) - sequence_length):
        X.append(data[i:i+sequence_length])
        y.append(data[i+sequence_length])
    return np.array(X), np.array(y)

# Create training and testing sequences
X, y = create_sequences(scaled_data, sequence_length)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, shuffle=False)

# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(sequence_length, 1)))
model.add(LSTM(units=50))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
```

```
model.fit(X_train, y_train, epochs=50, batch_size=32)

# Generate predictions on the test data
predictions = model.predict(X_test)

# Inverse scale the predictions and actual values
predictions = scaler.inverse_transform(predictions)
y_test = scaler.inverse_transform(y_test)

# Evaluate the model
mse = np.mean((predictions - y_test) ** 2)
print('Mean Squared Error:', mse)
```