НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.896

До захисту допущено
В. о. завідувача кафедри ММСА
О.Л.Тимощук
«___» _____ 2020 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 124 Системний аналіз
на тему: «Нейромережевий вибір найкращої моделі прогнозування»

Виконав:
студент ІІ курсу, групи КА-92мп
Романчук Юрій Юрійович _____

Керівник:
доцент кафедри ММСА,
к.т.н., доц. Жиров О. Л. _____

Рецензент:
доцент кафедри системного проектування
КПІ ім. Ігоря Сікорського,
к.т.н., доц. Кисельов Г. Д.

_____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань
Студент _____

Київ
2020

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 124 «Системний аналіз»

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри ММСА
О. Л. Тимощук
«___» _____ 2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту Романчуку Юрію Юрійовичу

1. Тема дисертації: «Нейромережевий вибір найкращої моделі прогнозування», науковий керівник дисертації Жиров Олександр Леонідович, к.т.н., доцент, затверджені наказом по університету від «02» листопада 2020р. № 3182-с

2. Термін подання студентом дисертації: 13 грудня 2020 р.

3. Об'єкт дослідження: нейронні мережі їх можливості та перспективи у сфері фінансового прогнозування

4. Предмет дослідження: моделі та методи застосування нейронних мереж для задач прогнозування, шляхи покращення існуючих методів та систем прогнозування.

5. Перелік завдань, які потрібно розробити:

1. Огляд предметної області та аналіз існуючих систем підтримки прийняття рішень, архітектур нейромереж;
2. Розробка нових підходів до прогнозування на основі використання методів штучного інтелекту;
3. Розробка програмного комплексу, що забезпечуватиме використання існуючих та розроблених методів для вирішення задачі прогнозування часових рядів фінансового ринку.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:

1). Схема побудованої нейронної мережі
2). Приклади функціонування створеного програмного продукту
3). Таблиці у розділі стартап-проекту
7. Дата видачі завдання: 05 вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації
1.	Концептуальний вступ дисертації. Формулювання об’єкта, предмета, цілі, завдань, новизни, практичної значущості результатів	18.09.2020—20.09.2020
2.	Перший розділ. Огляд літературно-інформаційних джерел.	21.09.2020—30.09.2020
3.	Другий розділ. Розробка DNN,LSTM,RNN моделей	01.10.2019—09.11.2019
4.	Третій розділ. Огляд програмного продукту та аналіз отриманих результатів	10.11.2020—16.11.2020
6.	Четвертий розділ. Стартап-проект	17.11.2020—20.11.2020
7.	Концептуальні висновки. Перспективи розвитку отриманих рішень	21.11.2020—26.11.2020

Студент Романчук Ю.Ю.

Науковий керівник дисертації Жиров О.Л.

РЕФЕРАТ

Магістерська дисертація:91 с., 15 рис., 18 табл., 1 додаток, 17 джерел.

Тема магістерської дисертації «Нейромережевий вибір найкращої моделі прогнозування».

Мета роботи — розробка програми для вибору найкращої моделі серед запропонованих нейромереж

Об’єкт дослідження - нейронні мережі, їх можливості та перспективи у сфері прогнозування.

Предмет дослідження - моделі та методи застосування нейронних мереж для задач прогнозування, шляхи покращення існуючих методів та систем прогнозування.

Для досягнення мети були поставлення наступні задачі:

1. Огляд предметної області та аналіз існуючих рішень, архітектур нейромереж;
2. Розробка нових підходів до прогнозування на основі використання елементів штучного інтелекту;
3. Розробка програмного комплексу, що забезпечуватиме просте використання існуючих та розроблених методів для вирішення задачі прогнозування часових рядів.

Програмний продукт реалізовано з використанням мови програмування Python який надає широкий спектр бібліотек для обробки та аналізу даних.

НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ДАНИХ, ТЕНЗОР, ПРОГНОЗУВАННЯ, МАШИННЕ НАВЧАННЯ.

ABSTRACT

Master's thesis:91 pp., 15 fig., 18 tabl., 1 application, 17 sources.

The theme of my master's thesis is «Neural network selection of the best forecasting model».

The purpose of the work is to develop a program for choosing the best model among the proposed neural networks

The object of the work is neural networks, their capabilities and prospects in the field of forecasting.

The subject of the work - models and methods of application of neural networks for forecasting tasks, ways to improve existing methods and forecasting systems.

To achieve this goal, the following tasks were set:

1. Review of the subject area and analysis of existing solutions, architectures neural networks;

2. Development of new approaches to forecasting based on use elements of artificial intelligence;

3. Development of a software package that will provide ease of use existing and developed methods for solving the problem of forecasting time series.

The software product is implemented using the Python programming language which provides a wide range of libraries for data processing and analysis.

NEURAL NETWORKS, DATA PROCESSING, TENSOR, FORECASTING, MACHINE LEARNING.

ЗМІСТ

ВСТУП

Будь-який вид прогнозування - це складне завдання, особливо там, де майбутнє дуже мінливе. Беручи до уваги великий обсяг даних, що зараз доступні в багатьох галузях, таких як інженерія, біомедицина, фінанси тощо та та все більшу обчислювальну потужність прогнозування стає невідємною частиною кожної галузі. Все більше і більше користувачів машиного навчання не є професіоналами, тобто не мають належних знань в цій області. Тому все більше і більше актуальною задачою стає ефективний нейромережевий вибір архітектури мережі та точне налаштування її гіперпараметрів. Для конкретного набору даних це трудомістке завдання, враховуючи приголомшливу кількість можливих альтернатив.

Саме тому метою роботи є розробка системи на основі нейромереж для визначення яка модель нейромережі краще підходить з різними гіперпараметрами. Для досягнення поставленої мети в роботі вирішуються наступні завдання:

- ознайомлення з підходами та методами прогнозування;
- тестування розробленого алгоритму;
- створення алгоритму на основі якого буде визначатись найкраща модель нейромережі;
- створення інтерфейсу для кращого користування алгоритмом.

РОЗДІЛ 1 АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ ТА ОПИС МЕТОДІВ ПРОГНОЗУВАННЯ

1.1 Актуальність дослідження

Прогнозування – це один з найважливіших моментів в прийнятті рішень в управлінні. Прогнозування є загальним статистичним завданням у бізнесі, де воно допомагає інформувати про рішення щодо планування виробництва, транспорту та персоналу, а також надає керівництву інформації щодо довгострокового стратегічного планування. Однак прогнозування бізнесу часто робиться погано, і його часто плутають з плануванням та цілями.

Машинне навчання вивчає алгоритми, які вдосконалюються завдяки досвіду. Беручи до уваги велику кількість даних, що зараз доступні у багатьох галузях, таких як інженерія, фінанси тощо, і все більше методів машинного навчання зараз практикують люди з дуже різноманітним досвідом. Більше користувачів інструментів машинного навчання неспеціалісти, яким потрібні готові рішення. Тим не менше, користувачеві все одно потрібно зробити певний вибір, який може бути не очевидним чи інтуїтивним при виборі алгоритму навчання, гіперпараметрів, тощо, що призводить до вибору неоптимальних моделей.

Останнім часом такі моделі глибокого навчання, як згорткові та рекурентні нейронні мережі привернули багато уваги завдяки підвищенню ефективності вирішення складних проблем навчання та гнучкості та загальності для вирішення великої кількості проблем, включаючи прогнозування, класифікацію, обробку природних мов, рекомендації системи тощо. Крім того, існує багато бібліотек програмного забезпечення, які полегшують їх реалізацію. Незважаючи на наявність таких бібліотек та інструментів, завдання вибору правильної моделі нейронної мережі та її гіперпараметри, як правило, є складними та повторюваними за своєю суттю, особливо серед неспеціалістів. Зазвичай процес вибору відповідної моделі

машинного навчання для певної проблеми здійснюється ітеративно. По-перше, вхідний набір даних повинен бути перетворений із специфічного для домену формату в функції, які передбачають поле інтересу. Після того як функції вибрані, користувачі повинні вибрати налаштування навчання, яке відповідає їх проблемі.

Потім користувачі повинні вибрати відповідну модель, таку як машини векторної підтримки (SVM), логістична регресія або будь-який вид нейронних мереж. Кожне сімейство моделей має ряд гіперпараметрів, таких як ступінь регуляризації, швидкість навчання, кількість нейронів тощо, і кожен з них повинен бути налаштований для досягнення оптимальних результатів. Нарешті, користувачі повинні вибрати пакет програмного забезпечення, який може навчити свою модель, налаштувати одну або кілька машин для проведення навчання та оцінки якості моделі. Зробити правильний вибір, зіткнувшись із такою кількістю ступенів свободи, може бути складно, змушуючи багатьох користувачів вибирати модель на основі інтуїції чи випадковості та / або залишати гіперпараметри встановленими за замовчуванням. Цей підхід, як правило, дасть неоптимальні результати. Це передбачає природну проблему для машинного навчання: задавши набір даних, автоматично та одночасно вибирати алгоритм навчання та встановлювати його гіперпараметри для оптимізації продуктивності. Комбінований простір алгоритму навчання та гіперпараметри дуже складними для пошуку: функція може бути зашумлена, а простір є багатовимірним, включаючи як категоріальні, так і безперервні варіанти та містить ієрархічні залежності, наприклад гіперпараметри алгоритму мають значення лише в тому випадку, якщо обраний цей алгоритм.

Таким чином, ідентифікація високоякісної моделі, як правило, дорого коштує в тому сенсі, що тягне за собою багато обчислювальних зусиль і забирає багато часу. Для вирішення цієї проблеми буде запропоновано автоматичний вибір моделі, гнучкий і масштабований метод для автоматизації процесу вибору моделей штучних нейронних мереж. Ключовими внесками методу є:

- Просте кодування нейронних мереж на основі списку як генотипів для еволюційних алгоритмів обчислень.
- Нові оператори кросоверу та мутації для створення дійсних моделей нейронних мереж з еволюційного алгоритму.
- Ведення функція фітнесу, яка враховує як точність моделі, так і її складність.
- Метод вимірювання подібності між двома нейронними мережами.

Всі ці компоненти можна використовувати для пошуку оптимальної архітектури нейронної мережі для даного набору даних.

1.2 Формалізація постановки задачі

Розглянемо набір даних D, який складається з навчальних точок $d_i = (x_i, y_i) \in X \times Y$ де X – набір точок даних і Y – набір міток. Крім того, розділяємо набір даних на навчальну вибірку D_t, перевірочну вибірку D_v та тестову D_p. Враховуючи архітектуру нейронної мережі шукаємо простір H, швидкість архітектури нейронної мережі $\varphi \in H$ навчена на D_t і перевірена за допомогою D_v визначається як

$$p = \text{Швид}(\varphi(D_t), D_v) \tag{1.1}$$

де $\text{Швид}(.)$ - вимірювання помилки узагальнення, досягнутої алгоритмом навчання $\varphi(.)$ на наборі перевірки D_v.

Поширеними показниками помилок прогнозування є середня квадратична помилка.

Як правило, ефективність φ не вимірюється. Дійсно, може бути кілька кандидатів, якими можна досягти подібні показники з підвищеною ефективністю. Під ефективністю розуміємо як швидко навчиться φ порівняно з іншими можливими рішеннями. Завданням цієї роботи є знаходження моделей які не лише демонструють хороші показники на *D*, виміряне за *p*, але також досягають таких показників, використовуючи просту структуру

Для вимірювання складності архітектури, використовується кількість навчальних параметрів нейронної мережі $w(\varphi)$. Також $w(\varphi)$ іншими словами можна назвати як "розмір" нейронної мережі.

Проблема пошуку нейронної мережі φ, яка досягає хороших показників на наборі даних D під час використання простої моделі можна математично сформулювати як багатофакторну задачу оптимізації:

$$\min_{\varphi \in H}(p(\varphi), w(\varphi)) \tag{1.2}$$

1.3 Огляд моделей прогнозування

1.3.1 Регресійні моделі

Регресійний аналіз - це метод прогнозного моделювання, який аналізує зв'язок між цільовою або залежною змінною та незалежною змінною в наборі даних. Різні методи регресійного аналізу використовуються, коли цільова та незалежна змінні показують лінійну або нелінійну залежність між собою, а цільова змінна містить безперервні значення. Регресійні моделі застосовується головним чином для визначення сили предиктора, тенденції прогнозу, часових рядів, а також у випадку причинно-наслідкових зв’язків [4].

Регресійний аналіз є основним методом вирішення проблем регресії в машинному навчанні за допомогою моделювання даних. Метою регресійного аналізу є визначення залежності між вихідною змінної і безліччю зовнішніх факторів (регресорів).

Існує багато методів регресійного аналізу, і використання кожного методу залежить від багатьох факторів. Ці фактори включають тип цільової змінної, форму кривої регресії та кількість незалежних змінних. Найпростішим варінтом регресійної моделі є лінійна регресія.

За основу моделі положено твердження, що існує дискретний зовнішній фактор $X(t)$, що виявляє вплив на шуканий процес $Z(t)$, при цьому зв'язок між процесом і зовнішнім фактором лінійний. Модель прогнозування на основі лінійної регресії описується рівнняням:

$$Z(t) = a_0 + a_1 X(t) + \varepsilon_t \tag{1.3}$$

де a_0, a_1 – коєфіцієнти регресії,

ε_t – помилка моделі.

Для отримання прогнозних значень в момент часу t необхідно мати значення $X(t)$ в тей же самий момент часу t, що на практиці вкрай малоймовірно.

Багатозначна модель прогнозування має вигляд

$$Z(t) = a_0 + a_1 X_1(t) + a_2 X_2(t) + \cdots + a_s X_s(t) + \varepsilon_t \tag{1.4}$$

В основі нелінійної регресійної моделі лежить твердження, що існує відома функція, яка описує залежність між вихідним процесом $Z(t)$ і зовнішнім фактором $X(t)$

$$Z(t) = F(X(t), A) \tag{1.5}$$

В рамках побудови моделі потрібно визначити параметри функції $A = [a_1, a_0]$. Але на практиці рідко зустрічаються процеси для яких вид функціональної залежності між процесом $Z(t)$ і зовнішнім фактором $X(t)$ є відомим. У зв'язку з цим нелінійні регресійні моделі застосовуються рідко [4].

1.3.2 Авторегресійні моделі

В основу авторегресійних моделей закладено припущення про те, що значення процесу $Z(t)$ лінійно залежить від деякої кількості попередніх значень того ж процесу $Z(t-1), \dots, Z(t-p)$.

В області аналізу часових рядів модель авторегресії (autoregressive, AR) і модель ковзного середнього (moving average, MA) є однією з найбільш використовуваних. Модель авторегресії є виключно корисною для опису часових рядів. У цій моделі поточне значення процесу виражається як кінцева лінійна сукупність попередніх значень процесу і імпульсу, який називається білим шумом

$$Z(t) = C + k_1 Z(t-1) + k_2 Z(t-2) + \cdots + k_p Z(t-p) + \varepsilon_t \tag{1.6}$$

де C - константа,

$k_1, \dots, k_p$- коефіцієнти,

ε_t- помилка моделі.

Формула (1.6) описує процес авторегресії порядку p, який в літературі часто позначається AR(p). Для визначення k_p і C використовують метод найменших квадратів або метод максимальної правдоподібності.

Інший тип моделі має велике значення в описі часових рядів і часто використовується спільно з авторегресією, який називається моделлю ковзного середнього порядку q і описується рівнянням

$$Z(t) = \frac{1}{q}(Z(t-1) + Z(t-2) + \cdots + Z(t-q)) + \varepsilon_t \quad (1.7)$$

де q – порядок ковзного середнього,

ε_t- помилка прогнозування.

У літературі процес (1.7) часто позначається *MA (q)*; Модель ковзного середнього є по суті фільтром низьких частот. Потрібно відзначити, що існують прості, зважені, кумулятивні, експоненціальні моделі ковзного середнього.

Для досягнення більшої гнучкості в підгонці моделі часто доцільно об'єднати в одній моделі авторегресію і ковзне середнє. Загальна модель позначається ARMA(p, q) яка поєднує в собі фільтр у вигляді ковзного середнього порядку q і авторегресію фільтрованих значень процесу порядку p.

Якщо в якості вхідних даних використовуються не самі значення часового ряду, а їх різниця d-того порядку (на практиці d необхідно визначати, проте в більшості випадків d ≤ 2), то модель носить назву авторегресії та інтегрованого ковзного середнього ARIMA (p, d, q)

Розвитком моделі ARIMA (p, d, q) є модель ARIMAX (p, d, q), яка описується рівнянням

$$Z(t) = AR(p) + a_1X_1(t) + a_2X_2(t) + \cdots + a_sX_s(t) \quad (1.8)$$

де $a_0, .., a_1$ – коєфіцієнти коефіцієнти зовнішніх факторів $X_1(t), .., X_s(t)$.

У даній моделі найчастіше процес $Z(t)$ є результатом моделі MA (q), тобто відфільтрованими значеннями вихідного процесу. Далі для прогнозування $Z(t)$ використовується модель авторегресії, в якій введені додаткові регресори зовнішніх факторів X1 (t), ..., XS (t).

Авторегресійна модель з умовною гетероскедастичності (autoregressive conditional heteroskedasticity, GARCH) була розроблена в 1986 році Тімом Петером Борреслевом і є моделлю залишків для моделі AR(p). На першому етапі для вихідного часового ряду визначається модель AR(p). Далі передбачається, що помилка моделі ε_t має дві складові

$$\varepsilon_t = \delta_t \alpha_t \tag{1.9}$$

де δ_t - залежне від часу стандартне відхилення;

α_t - випадкова величина, що має нормальний розподіл, середнє значення, рівне 0, і стандартне відхилення, яке дорівнює 1.

При цьому залежне від часу стандартне відхилення описується рівнянням

$$\delta_t^2 = \beta_0 + \beta_0 \varepsilon_{t-1}^2 + \cdots + \beta_\alpha \varepsilon_{t-\alpha}^2 + \gamma_1 \delta_{t-1} + \cdots + \gamma_p \delta_{t-p} \tag{1.10}$$

де $\beta_0, \ldots, \beta_q$ і $\gamma_1, \ldots, \gamma_p$ - коефіцієнти.

Рівняння (1.10) називається моделлю GARCH (p, q) і має два параметри: p характеризує порядок авторегресії квадратів залишків; q - кількість попередніх оцінок залишків.

Найбільш часте застосування дана модель отримала в фінансовому секторі, де за допомогою неї моделюється волатильність. На сьогоднішній день існує ряд модифікацій моделі під назвами NGARCH, IGARCH, EGARCH, GARCH-M і інші.

1.3.3 Моделі експоненціального згладжування

Експоненціальні методи згладжування відносно прості, але надійні підходи для прогнозування. Вони широко використовується в бізнесі для прогнозування прибутку для інвесторів.

Три основні варіації експоненціального згладжування зазвичай використовуються:

- просте експоненціальне згладжування;
- модель Хольта;
- метод Хольта – Вінтерса.

Відмінною рисою цих підходів є те, що часові ряди, як передбачається, будуються з неспостережених компонентів, як рівень, зростання та сезонні ефекти; ці компоненти потребують адаптації з часом, коли серії попиту відображають наслідки структурних змін на товарних ринках.

Звичайне експоненціальне згладжування застосовується в разі відсутності в даних тренда або сезонності. У цьому випадку прогноз є зваженою середньою всіх доступних попередніх значень ряду; ваги при цьому з часом геометрично зменшуються в міру просування в минуле (назад). Тому (на відміну від методу ковзного середнього) тут немає точки, на якій ваги обриваються, тобто зануляються.

Модель Хольта застосовується для моделювання процесів, що мають тренд. В цьому випадку в моделі необхідно розглядати дві складові: рівень і тренд.

$$Z(t) = S(t) + \varepsilon_t; \quad (1.11)$$

$$S(t) = a_0 Z(t-1) + (1-\alpha)(S(t-1) - B(t-1)); \quad (1.12)$$

$$B(t) = \gamma\big(S(t-1) - S(t-2)\big) + (1+\gamma)B(t-1)_. \quad (1.13)$$

де α – коефіцієнт згладжування рівня,

γ - коефіцієнт згладжування тренда.

Модель Хольта-Вінтерса застосовується для процесів, які мають тренд і сезонну складову.

1.3.4 Моделі на основі класифікаційних та регресійних дерев

Дерева класифікації та регресії або CART - це термін, введений Лео Брейманом для позначення алгоритмів дерева рішень, які можуть бути використані для класифікаційних або регресійних проблем прогнозуючого моделювання. Алгоритм CART забезпечує основу для таких важливих алгоритмів, випадкові ліси та підсилені дерева рішень. Представлення для моделі CART - це бінарне дерево. Кожен кореневий вузол представляє одну вхідну змінну (x) і точку розбиття на цю змінну (припускаючи, що змінна числова).

Листові вузли дерева містять вихідну змінну (y), яка використовується для прогнозування. Дерево можна зберігати як графік або набір правил.

Вибір, яку вхідну змінну використовувати, а також конкретну точку розбиття чи вирізання вибирається за допомогою жадібного алгоритму для мінімізації функції витрат. Побудова дерева закінчується з використанням заздалегідь визначеного критерію зупинки, такого як мінімальна кількість навчальних примірників, призначених кожному листовому вузлу дерева.

Створення бінарного дерева рішень насправді є процесом розподілу вхідного простору. Жадібний підхід використовується для поділу простору, званого рекурсивним бінарним розщепленням. Це числова процедура, коли всі значення вибудовуються, а різні точки поділу випробовуються та перевіряються за допомогою функції витрат. Вибирається спліт з найкращою вартістю (найнижча вартість, оскільки ми мінімізуємо витрати).

Процедура рекурсивного двійкового розщеплення повинна знати, коли припинити розщеплення, коли вона рухається вниз по дереву з даними навчальних даних. Найпоширеніша процедура зупинки полягає у використанні мінімального підрахунку кількості навчальних екземплярів, призначених кожному листовому вузлу. Якщо підрахунок менше деякого мінімуму, то розбиття не приймається, і вузол приймається як кінцевий вузол.

Кількість учасників навчання налаштовується на набір даних, наприклад 5 або 10. Він визначає, наскільки специфічним для навчальних даних буде дерево. Занадто конкретний (наприклад, підрахунок 1), і дерево перевершить дані навчання та, ймовірно, матиме низьку ефективність на наборі тестів.

Критерій зупинки є важливим, оскільки він сильно впливає на ефективність вашого дерева.

Складність дерева рішень визначається як кількість розділень у дереві. Віддають перевагу простим деревам. Їх легко зрозуміти (ви можете роздрукувати їх і показати експертам), і вони з меншою ймовірністю перебільшать ваші дані.

Найшвидший і найпростіший метод обрізки - це обробка кожного листового вузла на дереві та оцінка ефекту від його видалення за допомогою набору тестів на витримку. Листові вузли видаляються, лише якщо це призводить до падіння загальної функції витрат на всьому наборі тестів. Ви припиняєте видаляти вузли, коли подальших удосконалень зробити не вдається.

Можуть бути використані більш складні методи обрізки, такі як обрізка складності витрат (також звана обрізка найслабшого посилання), де навчальний

параметр (альфа) використовується для зважування того, чи можна видалити вузли на основі розміру піддерева.

1.3.5 Метод опорних векторів

Завдання класифікації відноситься до навчання з учителем. SVM - алгоритм навчання з учителем. Потрібно додати, що SVM може застосовуватися і для задач регресії.

Головна мета SVM як класифікатора - знайти рівняння розділення гіперплощини

$$w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = 0 \tag{1.14}$$

яка б розділила два класи якимось оптимальним чином. Загальний вигляд перетворення F об'єкта в мітку класу Y.

$$F(x) = sign(w^T x - b) \tag{1.15}$$

Після настройки ваг алгоритму w і b (навчання), всі об'єкти, які потрапляють по одну сторону від побудованої гіперплощини, будуть прогнозуватись як перший клас, а об'єкти, що потрапляють по іншу сторону - другий клас.

Усередині функції sign стоїть лінійна комбінація ознак об'єкта з вагами алгоритму, саме тому SVM відноситься до лінійних алгоритмів. Розділяючу гіперплощину можна побудувати різними способами, але в SVM ваги і налаштовуються таким чином, щоб об'єкти класів лежали якнайдалі від розділяючої гиперплощини. Іншими словами, алгоритм максимізує зазор між гіперплощиною і

об'єктами класів, які розташовані найближче до неї. Такі об'єкти і називають опорними векторами. Звідси і назва алгоритму.

Щоб розділяюча гіперплощина якнайдалі відстояла від точок вибірки, ширина смуги повинна бути максимальною. Вектор w - вектор нормалі до розділяючої гіперплощини. Проекцію вектора, кінцями якого будуть опорні вектора різних класів на вектор Вектор w. Ця проекція і буде показувати ширину розділяючої смуги (рис 1.1):

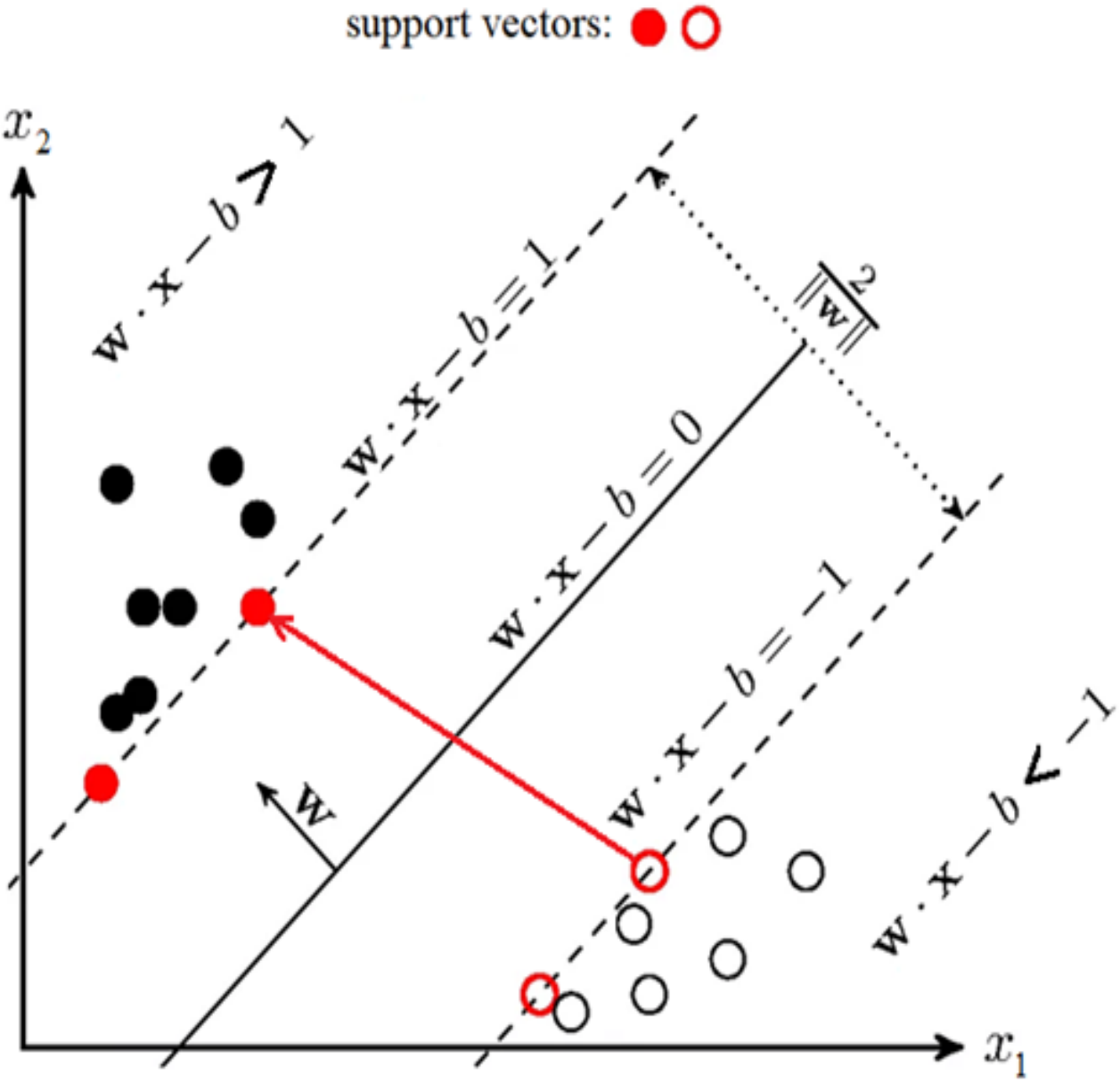

Рисунок 1.1 – Висновок правил налаштування ваг

1.4 Оцінка точності прогнозованої моделі

Вимірювання точності прогнозу є критичним для бізнесу та постійного вдосконалення процесу прогнозування. Прогнозування слід розглядати як процес постійного вдосконалення. Для цього потрібно знати, що працює, а що ні.

Наприклад, багато організацій формують базові прогнози, використовуючи статистичні підходи, а потім вносять в них оціночні корективи, щоб охопити свої знання про майбутні події. Організації, які відстежують точність як статистичних, так і скоригованих прогнозів, дізнаються, де коригування покращують прогнози, а де вони погіршують їх. Ці знання дозволяють зосередити свій час та увагу на елементах, де коригування додають цінності.

Прогноз - це більше, ніж число. Щоб ефективно використовувати прогноз, потрібно розуміння очікуваної точності. Статистика в межах вибірки та обмеження довіри дають деяке уявлення про очікувану точність; однак вони майже завжди недооцінюють фактичну (поза вибіркою) помилку прогнозування. Це пов'язано з тим, що параметри статистичної моделі вибираються, щоб мінімізувати встановлену помилку щодо історичних даних. Таким чином, параметри адаптовані до історичних даних та відображають будь-які його особливості. Іншими словами, модель оптимізована для минулого, а не для майбутнього.

Взагалі кажучи, статистика поза вибіркою (тобто історичні помилки прогнозу) дає кращий показник очікуваної точності прогнозу, ніж статистика у вибірці. Якщо пощастить бути в галузі з опублікованою статистикою щодо точності прогнозу, порівняння вашої точності з цими тестами дає змогу зрозуміти ефективність вашого прогнозування. Якщо галузеві контрольні показники недоступні періодичне порівняння поточної точності прогнозу з попередньою точністю прогнозу дозволяє виміряти ваше покращення. Різка несподівана зміна точності прогнозу часто є наслідком якоїсь основної події. Наприклад, якщо

невідомо для вас ключовий замовник вирішить перевезти конкуруючий товар, першим показником може бути незвично велика помилка прогнозу. Постійний моніторинг помилок прогнозу дозволяє виявляти, досліджувати та реагувати на ці зміни на ранніх стадіях - до того, як вони стануть більшими проблемами.

Середня абсолютна відсоткова помилка(MAPE) вимірює розмір помилки у відсотках. Вона обчислюється як середнє значення відсоткової помилки.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \tag{1.16}$$

Багато організацій зосереджуються насамперед на MAPE при оцінці точності прогнозу. Якщо більшості людей зручно мислити у відсотках, MAPE легко інтерпретувати. Він також може передавати інформацію, коли ви не знаєте обсяг попиту на товар. MAPE чутливий до масштабу і не повинен використовуватися при роботі з мало об'ємними даними. MAPE є невизначеним, коли Фактичний попит дорівнює нулю. Крім того, коли фактичне значення не дорівнює нулю, а досить мало, MAPE часто набуває екстремальних значень. Ця чутливість шкали робить MAPE неефективним як показник похибки для даних низького обсягу.

Середня абсолютна помилка (MAE) - це просто середнє значення абсолютної різниці між цільовим значенням та значенням, передбаченим моделлю.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{1.17}$$

Вимірювання похибки прогнозу для одного елемента є досить простим. Якщо працювати з малооб'ємним елементом, тоді MAE є хорошим вибором, тоді як слід уникати MAPE та інших статистичних даних на основі відсотків. Точність відстеження прогнозу є важливою частиною процесу прогнозування. Якщо ви не можете оцінити точність поточного процесу, його дуже важко вдосконалити. Крім

того, відстеження точності прогнозу забезпечує розуміння очікуваної ефективності, дозволяє вам порівняти прогнози та дозволяє виявляти, досліджувати та реагувати на проблеми раніше. Щоб відстежувати точність, ми повинні зберігати прогнози з часом, щоб пізніше ми могли порівняти ці прогнози з тим, що насправді сталося. Що стосується метрик точності прогнозу, то MAPE та MAE є найбільш часто використовуваною статистикою вимірювання помилок; однак і те, і інше за певних обставин може ввести в оману. MAPE чутливий до масштабу, і потрібно бути обережним при використанні MAPE з мало об’ємними предметами.

1.5 Висновки до розділу

В цьому розділі було розглянуто актуальність дослідження, визначено що задача прогнозування часових рядів має високу актуальність і є невіємною частиною роботи багатьох компаній. На даний момент існує багато моделей для вирішення різних задач прогнозування, але в даній дисертації робота більше спрямована на вибір найкрощої моделі серед нейромереж. Виявлені переваги та недоліки розглянутих моделей. Встановлено, що суттєвим недостатком авторегресійних моделей є велике число вільних параметрів, що вимагають ідентифікації; недостатками нейросетевих моделей є її непрозорість моделювання та складність навчання мереж. Також були розглянуті основні критерії для оцінки прогнозу та виявлення найкращої моделі.

РОЗДІЛ 2 МЕТОДИКА ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ ПРОГНОЗУВАННЯ

2.1 Огляд моделей побудови нейромережі

2.1.1 Багатошаровий персептрон

Багатошаровий персептрон (БНМ) є базовою формою штучних нейромереж. Назва походить від того, що входи подаються вперед через мережі на виходи в одному напрямку. Найпростіший екземпляр БНМ - це модель, яка складається з єдиного шару. Шар - це набір вузлів, які застосовують афінне перетворення, слідом за нелінійною активацією. Однак ваги кожного вузла є різними, що дозволяє їм представляти, можливо, іншу функцію. Афінне перетворення кожного вузла є навченим, у тому сенсі, що ваги перетворення дізнаються під час навчання [2,5]. Для більшої наочності проілюстровано просту структуру одного вузла (рис 2.1).

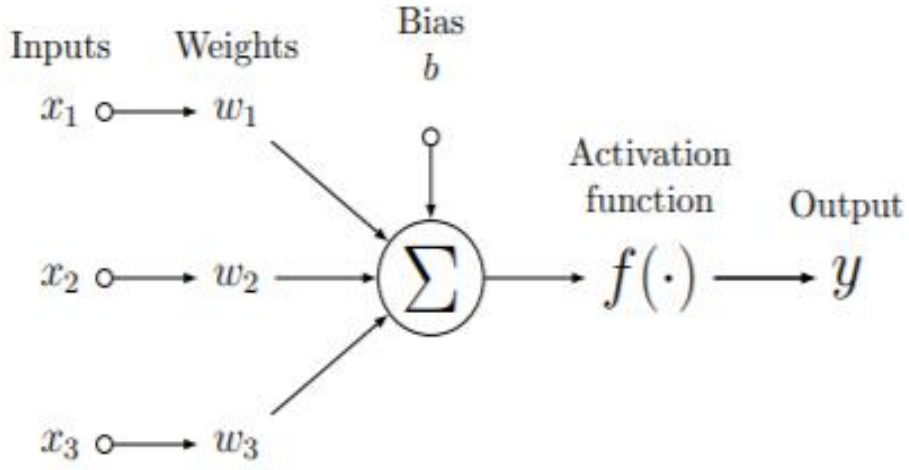

Рисунок 2.1 – Структура одного шару

У випадку одношарового персептрона єдиний рівень називається вихідним шаром, і кожен з його вузлів повністю зв’язаний зі всіма вхідними функціями. Нелінійна функція активації на вихідних вузлах зазвичай використовується для обмеження виводу у відповідному цільовому домені, тоді як кількість вузлів у вихідному рівні залежить від конкретного завдання навчання. Наприклад, у випадку регресії та двійкової класифікації вихідний рівень складається з одного вузла, який виводить дійсне значення та ймовірність відповідно. Модель логістичної регресії можна розглядати як окремий випадок єдиного шару персептрона з одним вихідним вузлом, який застосовує сигмоподібну функцію як нелінійну активацію для отримання ймовірнісних результатів [3].

Вивчення параметрів гауссового розподілу вимагає двох виходів: один для середнього значення, і один для дисперсії за допомогою функції активації, що обмежує параметр, щоб бути строго додатним. Для більш загальних проблем навчання, таких як багато класова класифікація та переклад мови, кількість вихідних вузлів може бути набагато більшою. У більш загальному випадку багатошарових персептронів шари складаються один на одного, щоб навчити більш складні нелінійні набори даних. Проміжні шари (між вхідним і вихідним шарами) називаються прихованими шарами. Вузли в кожному рівні мережі повністю з'єднані з усіма вузлами попереднього рівня. Вихід останнього прихованого шару можна розглядати як деяке нелінійне представлення ознак, отримане з входів мережі. Вихідний рівень, який, по суті, є одношаровим персептроном, вивчає відображення цих нелінійних ознак у фактичну ціль. Навчання за допомогою БНМ та, загальніше, за допомогою нейронних мереж, можна розглядати як процес вивчення нелінійної карти об’єктів входів, а також вивчення взаємозв’язку між цією картою об’єктів та фактичною ціллю [8].

Одним з основних обмежень БНМ є те, що вони не використовують структуру, яка часто присутня в даних у таких додатках, як комп'ютерний зір, обробка природними мовами та прогнозування часових рядів. Більше того,

кількість входів і виходів фіксована, що робить їх непридатними до проблем з різними вхідними та вихідними розмірами, як у прогнозуванні часових рядів. У наступних розділах ми обговорюємо більш складні архітектури, які долають ці обмеження, для яких БНМ часто використовуються як основні будівельні блоки [6].

2.1.2 Згорткові нейронні мережі

Згорткові нейронні мережі (ЗНМ) - це спеціальний клас нейронних мереж, призначений для додатків, де входи мають відому порядкову структуру, такі як зображення та часові ряди. ЗНМ - це локально з'єднані нейронні мережі, які використовують згорткові шари для використання структури, присутньої у вхідних даних. Згортковий шар застосовує функцію згортки до менших околиць вхідних даних. Тут згортка стосується процесу обчислення рухомої зваженої суми шляхом ковзання так званого фільтра або ядра по різних частинах вхідних даних. Розмір сусідства, а також те, як фільтр ковзає по вході, є частиною параметрів моделі. Потім нелінійна активація застосовується до результату операції згортки після додавання зміщення [1].

Фільтр містить ваги, яким слід навчитися. Мета полягає в тому, що ці ваги регулюються таким чином, що фільтр витягує відповідні характеристики з вихідних даних. Наприклад, у випадку з програмами обробки зображень, фільтр може навчитися виявляти краї на даних зображеннях. Оскільки точне розташування об'єкта, який витягується у вхідних даних, не є актуальним, один і той же фільтр використовується для скручування різних частин вводу для вилучення однієї ознаки. Цей розподіл ваг у фільтрі різко зменшує кількість вільних параметрів порівняно з щільними шарами БНМ і призводить до кращої продуктивності

узагальнення, особливо у випадку даних зображень, де кількість входів дуже велика. Більше того, оскільки дані вхідні дані можуть мати різні корисні функції, що мають відношення до даної задачі, в згортковому рівні зазвичай вивчається більше одного фільтра [11].

ЗНМ також використовують шар об'єднання, щоб зменшити розмір подання об'єкта, а також зробити об'єкти, вилучені із згорткового шару, більш надійними. Подібно до операції згортки, операція об'єднання застосовується до менших околиць, пересуваючи відповідний фільтр над входом. А шар об'єднання, однак, не має ваг, що піддаються вивченню, і, отже, і згортка, і шар об'єднання вважаються одним шаром в ЗНМ.

Можна далі розширити це поняття вилучення ознак, склавши кілька згорткових шарів один на одного, що відповідає поєднанню низькорівневих ознак, вилучених у попередніх шарах, для отримання ознак вищого порядку. Таке ієрархічне вилучення функцій зробило ЗНМ надзвичайно успішним у багатьох програмах обробки зображень та комп'ютерного зору.

2.1.3 Рекурентні нейронні мережі

Рекурентні нейронні мережі (РНН) - це нейронні мережі, спеціально розроблені для обробки послідовних даних, що виникають у таких сферах, як часові ряди, обробка природної мови та розпізнавання мови.

Класичні БНМ можна адаптувати для вирішення послідовного характеру даних, розглядаючи час як явну частину вхідних даних. Однак такий підхід має невід'ємні труднощі, а саме неможливість обробити послідовності різної довжини та виявити незмінні у часі закономірності в даних. Більш прямий підхід полягає у

використанні періодичних з'єднань, які з'єднують приховані блоки нейронних мереж назад із собою із затримкою в часі.

Оскільки приховані одиниці вивчають певні подання функцій необробленого введення, подача прихованих одиниць назад до себе на кожному часовому кроці може бути інтерпретована як надання мережі динамічної пам'яті. Простий РНН був запропонований на основі цієї ідеї, де на кожному кроці часу *t* мережа отримує зовнішній вхід для часу *t* і вихід прихованих одиниць з попереднього кроку часу *t - 1*. Однією з найважливіших деталей є те що та сама мережа використовується для всіх часових кроків; тобто ваги мережі розподіляються за кроками часу. Рисунок 2.2 ілюструє загальну структуру РНН.

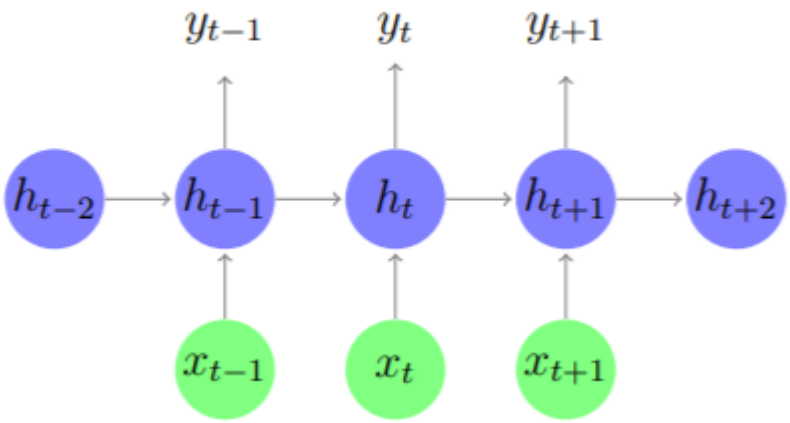

Рисунок 2.2 – Структура РНН

Ця ідея розподілу ваги подібна до ідеї ЗНН, де один і той же фільтр використовується в різних частинах вхідних даних. Це дозволяє РНН обробляти послідовності різної довжини під час навчання і, що більш важливо, узагальнювати на довжини послідовностей, яких не спостерігалося під час навчання.

Хоча РНН широко використовуються на практиці, виявляється, що навчити їх досить складно, враховуючи те, що вони, як правило, застосовуються до дуже

довгих послідовностей даних. Поширена проблема при навчанні дуже глибоких нейронних мереж методами, що базуються на градієнті, використовуючи зворотне поширення це зникнення чи вибух градієнтів [4], що робить навчання неможливим.

Для вирішення цієї проблеми було запропоновано мережі довгострокової пам’яті. Замість того, щоб використовувати просту мережу на кожному часовому кроці, довгострокові нейромережі використовують більш складну архітектуру, що складається з комірки та воріт, які контролюють потік вхідних даних до комірки, а також вирішують, яку інформацію слід зберігати всередині комірки та що слід розповсюджувати до наступного кроку часу [13].

Осередок має стан пам’яті, який розповсюджується за кроками часу разом із вихідними даними блоку, який сам по собі є функцією стану комірки. На відміну від вихідних даних модуля, стан клітини зазнає мінімальних змін протягом часу, отже похідна щодо стану клітини не занепадає і не експоненційно зростає. Отже, існує принаймні один шлях, де градієнт не зникає і не вибухає, роблячи довгострокові нейромережі придатними для обробки довгих послідовностей. Довгострокові нейромережі мають величезний успіх у широкому діапазоні застосувань, таких як прогнозування, машинний переклад, обробка мови та розпізнавання рукописного вводу. Існує кілька варіантів довгострокових нейромереж, які широко використовуються на практиці.

2.2 Навчання нейронної мережі

Стохастичний градієнтний спуск - це поточний метод за замовчуванням для навчання нейронної мережі, використовуючи метод зворотного поширення для обчислення градієнтів. Метод зворотного поширення помилки - це фактично рекурсивне застосування ланцюгового правила вздовж обчислювального графіка

мережі для обчислення градієнтів щодо параметрів. Використовуючи алгоритм оптимізації на основі градієнта, ваги мережі оновлюються для зменшення втрат, поки мережа не сходиться до стану, який не може покращитися далі.

Вибір функції активації для нейронної мережі має величезний вплив на спосіб навчання мережі. Історично склалося, що найпоширеніша нелінійна функція активації була сигмоїдна функція, яка стискала числа в діапазоні. Хоча популярний вибір протягом багатьох десятиліть, сигмоїдна функція страждає від трьох основних проблем:

- По-перше, вона посилює проблему зникаючого градієнта, оскільки насичені вузли (тобто вони мають значення, далекі від нуля) мають майже нульовий градієнт.
- По-друге, сигмоподібні виходи не мають нульового центру, що впливає на конвергенцію навчання.
- Нарешті, експоненціальна функція обчислюється дорожче порівняно з іншими альтернативами.

Кращим вибором, ніж сигмоїдна функція, є функція tanh, оскільки вона страждає від тих самих проблем, але вирішує проблему нульово орієнтованих виходів.

Однією з важливих причин, що перешкоджала ефективному навчанню складних архітектур нейронних мереж у перші дні, був наївний спосіб ініціалізації ваг (параметрів) мереж. Ініціалізація, як правило, проводилася з випадковими малими числами, що сприяло зникненню проблеми градієнта, що унеможливлювало підготовку мережі з багатьма шарами. Цю проблему розглядали Глорот та Бенджо з правилом нормалізації ваги, що залежить від розміру входів.

Це правило ініціалізації дає постійний розподіл (стандартний нормальний) на вихідних значеннях кожного шару. Зараз він відомий як ініціалізація Xavier. За допомогою цього методу для нормалізації проміжних значень між існуючими рівнями мережі вставляється додатковий рівень нормалізації партії. Добре

задокументовано, що пакетна нормалізація покращує стабільність і швидкість навчання нейронних мереж, хоча теоретичні причини цього залишаються предметом обговорення [15].

Нейронні мережі зазвичай навчають із використанням стохастичного градієнтного спуску, методу, за допомогою якого градієнт обчислюється для невеликої, випадково вибраної підмножини даних перед оновленням ваг моделі.

За допомогою цього методу навчання нейронних мереж є мало збіжним. Деякі вдосконалення простого стохастичного градієнтного спуску з використанням імпульсу або прискореного імпульсу Нестерова використовуються протягом десятиліть, але їх вплив на швидкість зближення був незначним. Протягом 2010-х багато досліджень було присвячено пошуку кращих правил оновлення градієнтів для збільшення рівня збіжності НС. Це дало помітні вдосконалення, такі як Adagrad, RMSProp , та Adam. Можливо, останній став оптимізатором за замовчуванням. Паскану запропонував метод відсікання градієнта для запобігання вибуху градієнтів та регулятор градієнта для запобігання зникнення градієнтів. Ці методи зробили навчання складних нейронних архітектур значно швидшим та стабільнішим.

Методи регуляризації широко використовуються в літературі про машинне навчання та статистику. У регресійних або класифікаційних моделях загальним підходом є включення в ціль регуляризаційного терміну, який обмежує простір параметрів та покращує можливості узагальнення моделі, уникаючи переобладнання. Функції регуляризації, такі як l_1 та l_2 були популярні в класичних часових рядах та прогнозуванні; *l-норма* регуляризації широко застосовується в нейронних мережах з тією ж метою, що і в регресійних моделях.

Метод регуляризації виключення був спеціально розроблений для нейронних мереж. Ідея виключення дуже проста: щоб запобігти перенавчанню, деякі вузли випадково маскуються з мережі під час кожного проходження під час навчання. Мабуть, найбільш інтуїтивний спосіб зрозуміти чому виключення допомагає - це

розглядати його як дуже ефективний засіб реалізації потужного підходу до ансамблевого навчання. Навчання багатьох НН для створення ансамблю можливо, але дорого в обчисленні. Виключення досягає по суті тієї ж мети і значно покращує продуктивність узагальнення без збільшення обчислювальних витрат [11].

Збільшення обчислювальної потужності в сучасних комп'ютерах, безсумнівно, зіграло ключову роль у недавньому успіху нейронних мереж. Грубе порівняння між найпотужнішими комп’ютерними процесорами, наявними на момент зародження нейронних мереж наприкінці 50-х років, з тими, що існують сьогодні, виявляє збільшення обчислювальної потужності на дванадцять порядків. Використання багатоядерних центральних процесорних блоків (ЦП) і особливо використання графіки значно скоротили час, необхідний для підготовки великомасштабних моделей. Це відкрило шлях для дослідження все більш складних архітектур, які було непрактично навчати через обмеження в часі [7].

Одночасно зі збільшенням обчислювальної потужності відбулося збільшення кількості даних, доступних для навчання моделей. Це призвело до створення великих наборів даних з мільйонами прикладів, загальнодоступних для дослідників. Публічна доступність цих масивних наборів даних дозволила дослідженням скористатися методологічними досягненнями, обговореними вище, для підготовки моделей із дедалі складнішими архітектурами та кращими властивостями узагальнення.

2.3 Види функції активації нейромережі

Східчаста функція активація визначається як

$$f(x) = \begin{cases} 0, \text{якщо } 0 > x \\ 1, \text{якщо } x \geq 0 \end{cases} \quad (2.1)$$

де вихідний результат дорівнює 1, якщо значення x більше, ніж рівне нулю, і 0, якщо значення x менше нуля.

Східчаста функція не диференціюється в нулі. Нейронна мережа використовує метод зворотного поширення разом із градієнтним спуском для обчислення ваги різних шарів. Оскільки ступінчаста функція не є диференційованою в нулі, вона не може досягти прогресу при підході до градієнтного спуску і не виконує завдання оновлення ваг.

Для подолання цієї проблеми було введено сигмоїдну функцію замість східчастої. Сигмоїдна функція або логістична функція має вигляд

$$\delta(z) = \frac{1}{1+e^{-z}} \quad (2.2)$$

Значення функції прагне до нуля, коли z, або незалежна змінна прямує до негативної нескінченності та до 1, коли z прагне до нескінченності. Слід мати на увазі, що ця функція являє собою наближення поведінки залежної змінної та є припущенням. Причини використання ми використовуємо сигмоїдну функцію як одну з апроксимаційних функцій:

1. Вона фіксує нелінійність даних. Хоча в наближеній формі, але поняття нелінійності є важливим для точного моделювання.

2. Сигмоїдна функція повністю диференційована, і, отже, може бути використана при градієнтному підході та зворотному поширенні помилки для розрахунку ваги різних шарів

3. Припущення про залежну змінну, яка дотримується сигмовидної функції, за своєю суттю передбачає розподіл Гауса для незалежної змінної, що є загальним розподілом.

Однак сигмоїдна функція також страждає від проблеми зникнення градієнтів. Сигмоїдна функція стискає вхід в дуже малий діапазон виходу [0,1] і має дуже круті градієнти. Таким чином, залишаються великі області вхідного простору, де навіть великі зміни спричиняють дуже незначні зміни у виході. Це називається проблемою зникнення градієнта. Ця проблема збільшується із збільшенням кількості шарів і, таким чином, застоює навчання нейронної мережі на певному рівні [10].

Функція *tanh(z)* - це масштабована версія сигмоїди (рис. 2.3), і її діапазон виводу становить [- 1,1] замість [0,1]. Загальна причина використання функції *tanh(z)* в деяких місцях замість сигмовидної функції полягає в тому, що оскільки дані центрируються навколо 0, похідні вище. Більш високий градієнт допомагає покращити швидкість навчання.

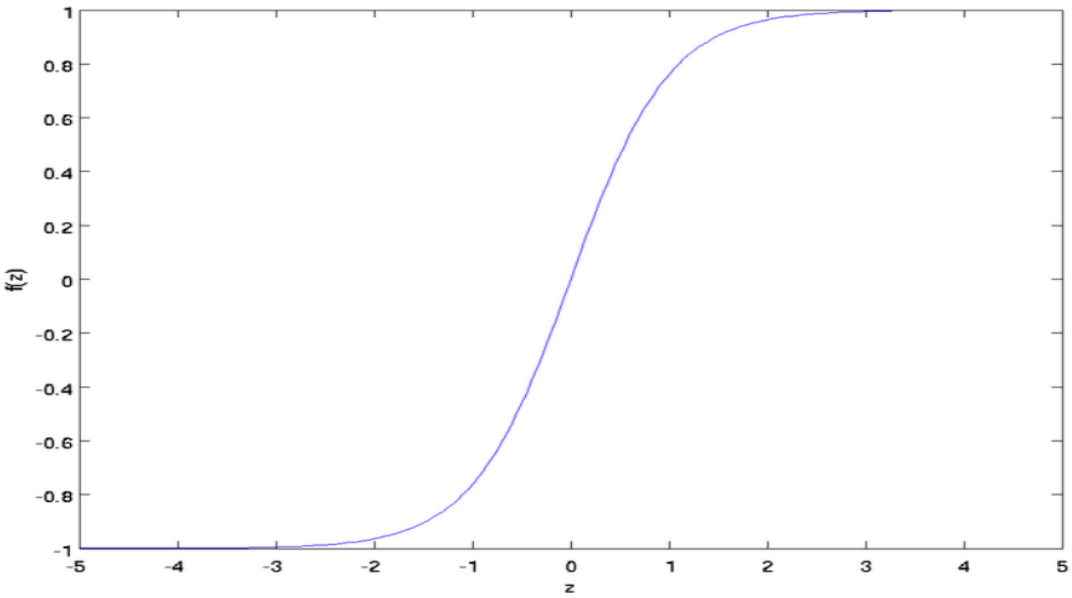

Рисунок 2.3 – Графік функції tanh

Однак проблема зникнення градієнтів все ще зберігається у функції *tanh*.

Функція ReLu - це найбільш часто використовувана функція активації в моделях глибокого навчання. Функція повертає 0, якщо отримує будь-який негативний вхід, але для будь-якого додатного значення x вона повертає це значення назад. Отже, це можна записати як

$$f(x) = \max(0, x) \tag{2.3}$$

Це означає що:

- Обчислення є достатньо дешевим, оскільки немає складної математики. Тому модель може зайняти менше часу на навчання.
- Вона швидше сходиться. У нього немає проблеми зникаючого градієнта, якої страждають інші функції активації, такі як сигмоїдна або tanh.
- Він рідко активований. Оскільки ReLU дорівнює нулю для всіх негативних входів, можливо, для будь-якого даного блоку взагалі не активується. Це часто бажано.

2.4 Гіперпараметри моделі

Гіперпараметри моделі - властивості, які регулюють весь навчальний процес. Вони включають змінні, що визначають структуру мережі (наприклад, кількість прихованих шарів) та змінні, які визначають, як мережа навчається (наприклад, швидкість навчання). Модельні гіперпараметри встановлюються перед начанням (перед оптимізацією ваг та ухилу).

Наприклад, ось кілька вбудованих змінних конфігурації моделі:

- Швидкість навчання.
- Кількість епох.
- Приховані шари.
- Приховані одиниці.
- Функції активації.

Гіперпараметри важливі, оскільки вони безпосередньо контролюють поведінку навчального набору, що має значний вплив на результативність моделі, що навчається.

Вибір відповідних гіперпараметрів відіграє ключову роль у успіху архітектур нейронних мереж, враховуючи вплив на вивчену модель. Наприклад, якщо рівень навчання занадто низький, модель буде пропускати важливі закономірності в даних; навпаки, якщо він високий, він може мати зіткнення.

Вибір хороших гіперпараметрів забезпечує дві основні переваги:

- Ефективний пошук у просторі можливих гіперпараметрів.
- Простіше управління великим набором експериментів з налаштування гіперпараметрів.

Гіперпараметри можна приблизно розділити на 2 категорії:

- Гіперпараметри оптимізатора.
- Специфічні для моделі гіперпараметри.

Гіперпараметри оптимізації пов’язані більше з процесом оптимізації та навчання. Якщо швидкість навчання моделі набагато менша за оптимальні значення, для досягнення ідеального стану знадобиться набагато більше часу (сотні чи тисячі) епох. З іншого боку, якщо швидкість навчання набагато більша за оптимальне, тоді це перевищує ідеальний стан, і алгоритм може не сходитися [16]. Розумний стартовий рівень навчання = 0,001. Важливо врахувати, що:

- модель матиме сотні і тисячі параметрів, кожен зі своєю кривою похибки. І рівень навчання повинен враховувати всіх них;

- криві помилок - це не чисті u-форми; натомість вони, як правило, мають більш складні форми з місцевими мінімумами.

Розмір серії нетривіальним чином впливає на потреби в ресурсах навчального процесу, швидкість та кількість ітерацій.

Історично склалося суперечка щодо проведення стохастичного навчання, коли ви підбираєте один приклад набору даних до моделі і, використовуючи лише один приклад, виконуєте прямий прохід, обчислюєте помилку або зворотне поширення помилки та встановлюєте скориговані значення для всіх гіперпараметрів. А потім повторюєте це для кожного прикладу в наборі даних. Або, можливо, краще подати всі дані на навчальний крок і розрахувати градієнт, використовуючи помилку, породжену переглядом усіх прикладів у наборі даних. Це називається груповим навчанням.

Щоб вибрати правильну кількість епох для етапу навчання, метрикою, на яку слід звернути увагу, є помилка валідації. Інтуїтивно зрозумілий ручний спосіб полягає в тому, щоб модель навчала стільки ітерацій, скільки похибка перевірки постійно зменшується. Існує техніка, яка може бути використана під назвою „Рання зупинка”, щоб визначити, коли припинити тренування моделі; мова йде про зупинку навчального процесу на випадок, якщо помилка перевірки не покращилася за останні 10 або 20 епох. Кількість прихованих одиниць є одним з найбільш загадкових гіперпараметрів.

Нейронні мережі - це універсальні апроксиматори функцій, і для того, щоб вони навчились апроксимувати функцію (або завдання прогнозування), вони повинні мати достатню ‘здатність’ для вивчення функції. Кількість прихованих одиниць є основним показником навчальної здатності моделі. Для простої функції може знадобитися менша кількість прихованих одиниць. Чим складніша функція, тим більший навчальний потенціал знадобиться моделі. Трохи більша кількість одиниць, ніж оптимальна кількість, не є проблемою, але набагато більша кількість призведе до переобладнання (тобто, якщо ви надаєте модель із занадто великою

ємністю, вона може мати тенденцію до перенавчання, намагаючись "запам'ятати" набір даних, що впливає на здатність до групування).

Процес пошуку найбільш оптимальних гіперпараметрів у машинному навчанні називається оптимізацією гіперпараметрів.

Загальні алгоритми це:

- Жадібний пошук.
- Випадковий пошук.
- Байєсова оптимізація.

Пошук сітки - це традиційний прийом реалізації гіперпараметрів. Пошук сітки вимагає створення двох наборів гіперпараметрів:

- Швидкість навчання.
- Кількість шарів.

Пошук у сітці навчає алгоритм для всіх комбінацій, використовуючи два набори гіперпараметрів (швидкість навчання та кількість шарів) та вимірює ефективність, використовуючи техніку перехресної перевірки. Ця техніка перевірки забезпечує, що навчена модель отримує більшість шаблонів із набору даних. Метод пошуку сітки - це найпростіший алгоритм для використання, але він не дуже підходить, якщо дані мають великі розмірні простори.

Випадковий пошук виконує вибірку простору пошуку та обчислює набори із заданого розподілу ймовірностей. Замість того, щоб намагатися перевірити всі 100000 зразків, наприклад, ми можемо перевірити 1000 випадкових параметрів. Однак недоліком використання алгоритму випадкового пошуку є те, що він не використовує інформацію попередніх експериментів для вибору наступного набору. Більше того, важко передбачити наступний експеримент.

Налаштування гіперпараметра максимізує продуктивність моделі на наборі для перевірки. Для алгоритмів машинного навчання часто потрібна точна настройка гіперпараметрів моделі [9,12].

Більш привабливий спосіб оптимізації та налаштування гіперпараметрів передбачає увімкнення підходу автоматизованої настройки моделі - наприклад, за допомогою байєсівської оптимізації. Модель, яка використовується для наближення цільової функції, називається сурогатною моделлю. Популярною сурогатною моделлю для байєсівської оптимізації є Гаусів процес. Байєсова оптимізація, як правило, працює, припускаючи, що невідома функція була відібрана з Гауссового процесу і підтримує розподіл для цієї функції під час здійснення спостережень.

При виконанні байєсівської оптимізації слід зробити два основних рішення:

- Вибрати функції, які виражатимуть припущення щодо оптимізованої функції;
- Далі вибрати функцію отримання, яка використовується для побудови функції корисності апостеріорної моделі, що дозволяє нам визначити наступну точку для оцінки.

2.5 Висновки до розділу

В цьому розділі був даний огляд використання нейромереж для прогнозування. Розкрито основні поняття про нейромережі, вибраними за ступенем їх важливості для прогнозування. Розглянуто останні досягнення моделей нейронного прогнозування та представлено широкий спектр областей застосування при прогнозуванні, де ці методи довели свою ефективність.

Нейронні методи прогнозування допомагають при вирішенні проблем прогнозування з часовими рядами та при витягуванні слабких сигналів та складних закономірностей з великих обсягів даних. Наявність ефективних систем програмування допомагає полегшити багато проблемних моментів, з якими досить

часто стикаються розробники за допомогою інших методів, таких як ручне проектування функцій або необхідність для виведення градієнтів.

Нейронні мережі - це не срібна куля. Для багатьох важливих класів проблем прогнозування, таких як довгострокові макроекономічні прогнози чи інші проблеми, що вимагають знань із зовнішнього домену, яких не можна дізнатись із даних, нейронні методи прогнозування не є найбільш підходящим вибором і, ймовірно, ніколи не будуть ними. Тим не менше, нейронні мережі належать до набору інструментів кожного прогнозиста, як у промисловості, так і в наукових колах.

3 ОГЛЯД ПРОГРАМНОГО ПРОДУКТУ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Обґрунтування вибору платформи та мови програмування

Машинне навчання набирає велику популярність за останні роки. Основною метою цієї галузі є перетворення значущих даних у маркетингові та бізнес-стратегії, що допомагає компанії розвиватися. Дані зберігаються та досліджуються, щоб отримати логічне рішення. Раніше в цій галузі брали участь лише провідні IT-компанії, але сьогодні підприємства з різних галузей та таких галузей, як електронна комерція, охорона здоров’я, фінанси та інші, використовують аналіз даних. Для аналізу даних доступні різні інструменти, такі як Hadoop, програмування R, SAS, SQL та багато іншого. Однак найпопулярнішим та простим у використанні інструментом аналізу даних є Python. Він відомий як швейцарський армійський ніж у світі кодування, оскільки підтримує структуроване програмування, об'єктно-орієнтоване програмування, а також функціональну мову програмування та інші.

Python є найпопулярнішою мовою програмування у світі, а також відома як найбільш підходяща мова для інструментів та додатків науки про дані. Python має унікальний атрибут і простий у використанні, коли йдеться про кількісні та аналітичні обчислення. Він є лідером галузі вже досить давно і широко використовується в різних галузях, таких як нафта і газ, обробка сигналів, фінанси та інші. Крім того, Python був використаний для зміцнення внутрішньої інфраструктури Google та для побудови таких додатків, як YouTube. Python широко використовується і є улюбленим інструментом, будучи гнучкою мовою з відкритим кодом. Його масивні бібліотеки використовуються для обробки даних, і їх дуже легко вивчити навіть для початківця аналітика даних.

Окрім незалежної платформи, вона також легко інтегрується з будь-якою існуючою інфраструктурою, яка може бути використана для вирішення найскладніших проблем. Чому Python віддають перевагу іншим інструментам науки про дані?

Потужний і простий у використанні - Python вважається мовою для початківців, і будь-який студент або дослідник, що володіє лише базовими знаннями, може почати працювати над ним. Час, витрачений на налагодження кодів та на різні обмеження програмної інженерії, також мінімізований. У порівнянні з іншими мовами програмування, такими як C, Java та C #, час на впровадження коду менший, що допомагає розробникам та інженерам програмного забезпечення витрачати більше часу на роботу в своїх алгоритмах. Вибір бібліотек - Python забезпечує величезну базу даних бібліотек та штучного інтелекту та машинного навчання. Деякі з найпопулярніших бібліотек включають Scikit Learn, TensorFlow, Seaborn, Pytorch, Matplotlib та багато інших. В Інтернеті доступні багато навчальних посібників та ресурсів з інформатики та машинного навчання, до яких можна легко отримати доступ. Порівняно з іншими мовами програмування, такими як Java та R, Python зарекомендував себе як дуже масштабована та швидша мова.

Це забезпечує гнучкість для вирішення проблем, які неможливо вирішити за допомогою інших мов програмування. Багато підприємств використовують його для розробки швидких додатків та інструментів усіх видів.

На Python доступні різні варіанти візуалізації. Його бібліотека Matplotlib забезпечує міцну основу, навколо якої будуються інші бібліотеки, такі як ggplot, pandas plotting, pytorch та інші. Ці пакети допомагають створювати діаграми, готові до Інтернету графіки, графічні макети тощо.

Як Python використовується на кожному етапі науки та аналізу даних? Ми повинні знати і розуміти, який тип форми приймають дані. Вам потрібно отримати статистику, виконуючи деякі функції та шукаючи певний тип даних у кожному рядку, а також стовпці. Для виконання цього обчислювального завдання може

знадобитися багато часу та напруженої роботи. Отже, ви можете використовувати бібліотеки Python, такі як Pandas та Numpy, які можуть швидко виконувати роботу за допомогою паралельної обробки. Наступна перешкода - отримання необхідних даних. Оскільки дані нам не завжди легко доступні, нам потрібно відповідно витягти дані з Інтернету.

Саме тому для вирішення задачі нейромережевого вибору найкращої методу використовується мова python та бібліотека Keras.

3.2 Аналіз алгоритму роботи системи

В даній дисертації запропонований наступний алгоритм порівняння та вибору найкращої моделі серед різних нейромереж. Система складається з 4 нейромереж:

- DNN.
- RNN.
- LSTM.

Для кожної з цих трьох моделей нейромереж проводимо наступні дії:

- Попередню обробку даних.
- Визначаємо форму нейронної мережі та компіляцію моделей.
- Для кожної моделі визначаємо найкращі гіперпараметри, функцію оптимізації та функцію втрат.
- Оцінка моделі використовуючи валідаційний набір даних.
- Візуалізація прогнозу.

Підготовка даних передбачає використання таких методів, як нормалізація та стандартизація для масштабування вхідних та вихідних змінних до підготовки

моделі нейронної мережі. Моделі нейронних мереж глибокого навчання вивчають відображення від вхідних змінних до вихідних змінних. Таким чином, масштаб і розподіл даних, отриманих з домену, можуть бути різними для кожної змінної.

Вхідні змінні можуть мати різні одиниці виміру (наприклад, фути, кілометри та години), що, в свою чергу, може означати, що змінні мають різні масштаби. Різниця у шкалах між вхідними змінними може збільшити складність моделі, що моделюється. Прикладом цього є те, що великі вхідні значення (наприклад, розкид на сотні або тисячі одиниць) можуть призвести до моделі, яка засвоює великі значення ваги. Модель з великими значеннями ваги часто нестабільна, це означає, що вона може страждати від поганих показників під час навчання та чутливості до введених значень, що призводить до вищої помилки узагальнення. Одна з найпоширеніших форм попередньої обробки складається з простого лінійного масштабування вхідних змінних. Цільова змінна з великим розкидом значень, у свою чергу, може призвести до великих значень градієнта помилок, що спричиняють різкі зміни вагових значень, роблячи процес навчання нестабільним.

Масштабування вхідних та вихідних змінних є критичним кроком у використанні моделей нейронних мереж. На практиці майже завжди вигідно застосовувати переробки перед обробкою до вхідних даних, перш ніж вони будуть представлені в мережі. Подібним чином, виходи мережі часто обробляються після отримання необхідних вихідних значень.

Показники для оцінки моделей нейромереж є дуже важливі. Вибір метрик впливає на те, як вимірюється та порівнюється ефективність. Вони впливають на те, яку кінцеву модель вибрати. В алгоритмі використовуються MAE та MSE.

Для визначення найкращої моделі серед кожного типу нейромереж застосовується жадіюний пошук за допомогою бібліотеки sklearn. Жадібний пошук - це техніка налаштування, яка намагається обчислити оптимальні значення гіперпараметрів. Це вичерпний пошук, який виконується за певними значеннями

параметрів моделі. Модель також відома як оцінювач. Для жадібного пошуку було задано такі функції активації:

- ReLu.
- Tanh.
- Сигмоїдальна.
- Лінійна.
- SoftMax.

Також для жадібного пошуку було задано кількість епох та розмір партії.

3.3 Інтерфейс користувача

Для того щоб краще візуалізувати та без особливих зусиль відстежувати, керувати, порівнювати був розроблений простий інтерфейс на основі системи з відкритим кодом Allegro Trains. Запускаючи експермент можна подивитись на результати функції втрат та оцінку точності відразу з програми(рис 3.1)

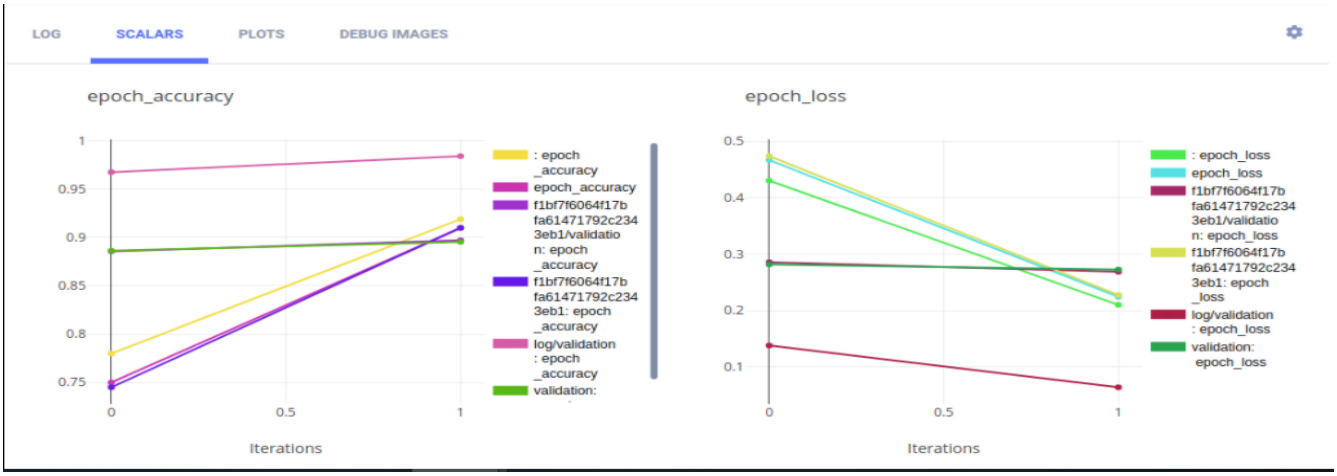

Рисунок 3.1 – Результати навчання нейромережі в інтерфейсі

Також можливо подивитись на гіперпараметри моделі(рис 3.2) та порівняти результати(рис 3.3). Легко знаходити відмінності та вплив параметрів конфігурації експерименту, метрик, скалярів. Також можна швидко подивитись та визначити найкращу моделі для конкретного експерименту.

Рисунок 3.2 – Гіперпараметри моделі

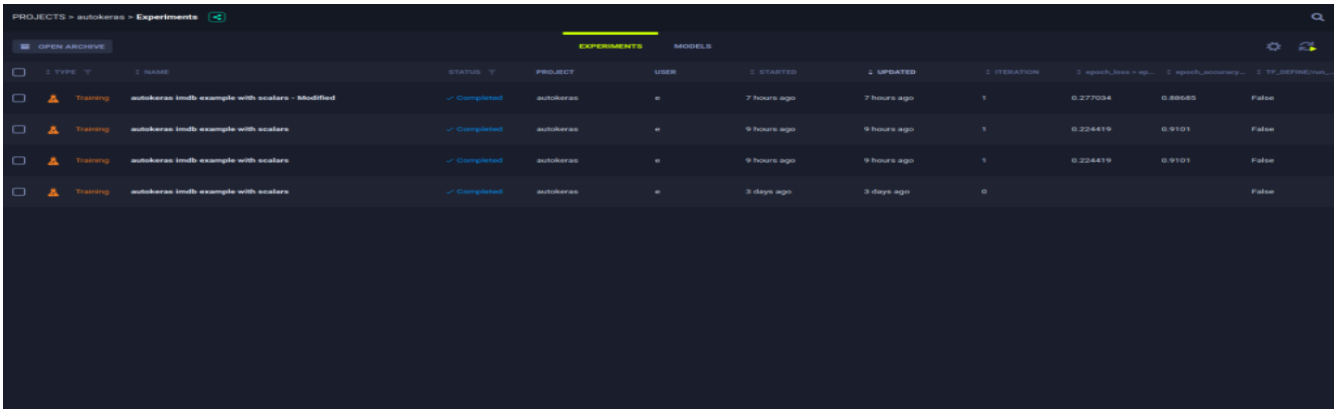

Рисунок 3.3 – Порівняння результатів

3.4 Аналіз якості роботи системи

Для перевірки якості роботи алгоритму був взятий набір даних оснований на цінах на будинки. Попросіть покупця будинку описати будинок своєї мрії, і, ймовірно, це не почнеться з висоти стелі підвалу чи близькості до залізниці схід. Але цей набір даних змагань доводить, що на переговори про ціну впливає набагато більше, ніж кількість спалень чи огорожа. Завдяки 79 пояснювальним змінним, що описують (майже) кожен аспект житлових будинків в Еймісі, штат Айова, потрібно передбачити кінцеву ціну кожного будинку. На рисунку 3.4 показано графік розподілення цін, а також ковзне середнє та стандартне відхилення.

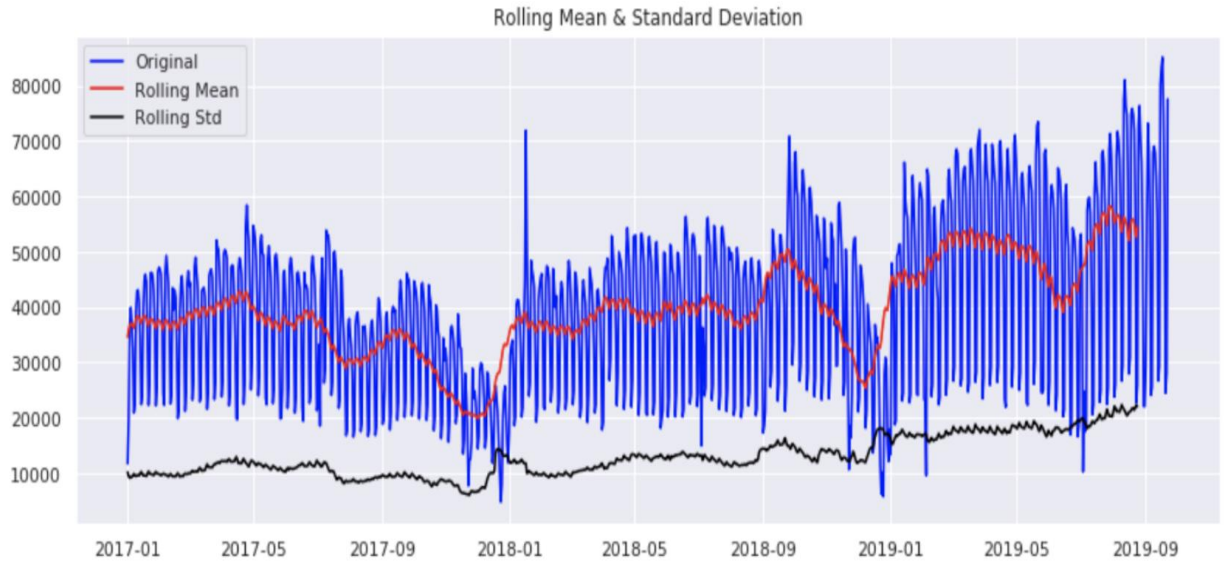

Рисунок 3.4 – графік розподілення набору даних

Після нормалізації даних та розділення вибірки на тестову та навчальну у співвідношенні 80 на 20 було запущено жадібний пошук для кожного типу нейромережі, щоб вибрати найкращу модель для кожного типу нейромережі на цьому наборі даних. Результати наведено в таблиці 3.1.

Таблиця 3.1 Результати роботи жадібного пошуку

Тип нейромережі	Кількість епох	Функція активація	Розмір партії	Найкращий результат
DNN	100	ReLu	30	0.82
RNN	100	ReLu	50	0.84
LSTM	100	SoftMax	50	0.86

Тепер визначамо показники помилок на кожній моделі з гіперпараметрами з таблиці 3.1. Почнемо експеримент з глибокої нейромережі. Глибока нейронна мереж (DNN) - це штучна нейронна мережа (ANN) з декількома шарами між вхідним та вихідним шарами(рис3.5).

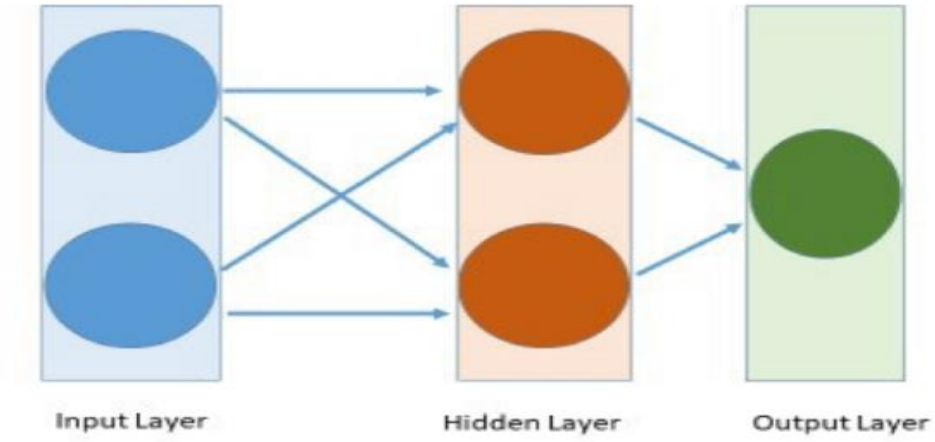

Рисунок 3.5 – Глибока нейронна мережа з одним прихованим шаром

За замовчуванням експеримент відбувається з глибокою нейроною мережею з одним прихованим шаром, кількість епох – 100, розмір партії – 30. Після оцінки

моделі отримали наступні результати – RMSE на тестовій вибірці склав 6579.23, а MAE – 4034.69.(рис 3.6)

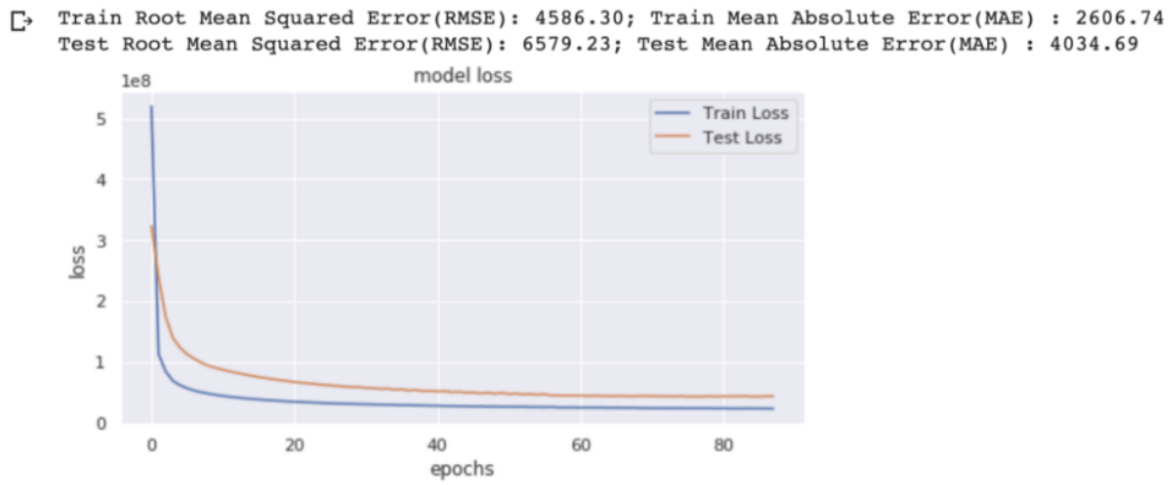

Рисунок 3.6 - метрики помилок та графік втрат глибокої нейронної мережі

Наступною буде проводитися прогноз за допомогою рекурентної нейронної мережі, яка буде складатися з двох прихованих шарів (рис.3.7)

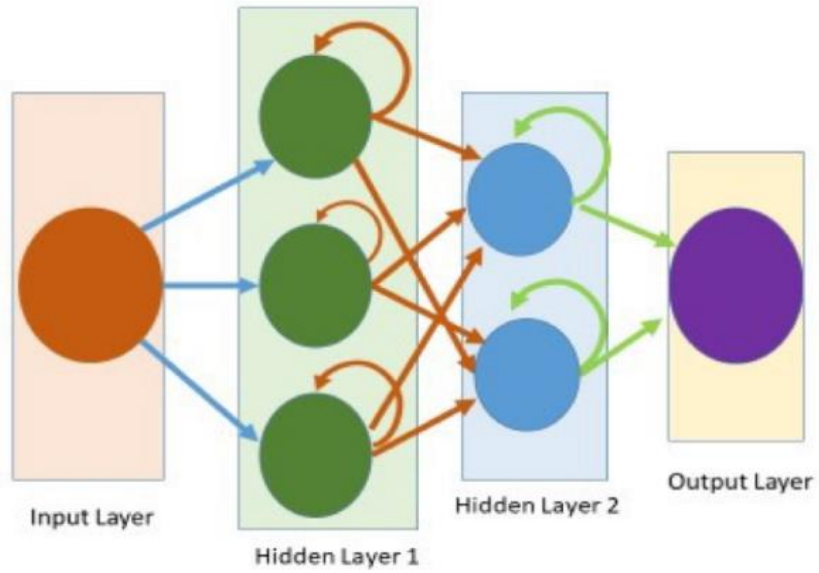

Рисунок 3.7 – Рекурентна нейронна мережа

За замовчуванням для цього експерименту використовується функція активації ReLu, кількість епох – 100, розмір партії – 50. Після оцінки моделі отримали наступні результати – RMSE на тестовій вибірці склав 8168.90, а MAE – 5088.06.(рис 3.8).

Рисунок 3.8 - метрики помилок та графік втрат рекурентної нейронної мережі

Остання побудована нейромережа є LSTM яка складається з 4 основних компонентів: Вхідні ворота, вихідні ворота, комірка пам'яті та забуті ворота. LSTM чутливі до масштабу вхідних даних. Під час етапу попередньої обробки було застосувано клас попередньої обробки MinMaxScaler із модуля scikit-learn для нормалізації, масштабування набору даних. За замовуванням було взято дво шарову LSTM без прихованих шарів, функція активації SoftMax, кількість епох – 100, розмір партії – 50. Після оцінки моделі отримали наступні результати – RMSE на тестовій вибірці склав 6732.69, а MAE – 3744.30.(рис 3.9).

```
Train Root Mean Squared Error(RMSE): 4836.35; Train Mean Absolute Error(MAE) : 2675.77
Test Root Mean Squared Error(RMSE): 6732.69; Test Mean Absolute Error(MAE) : 3744.30
```

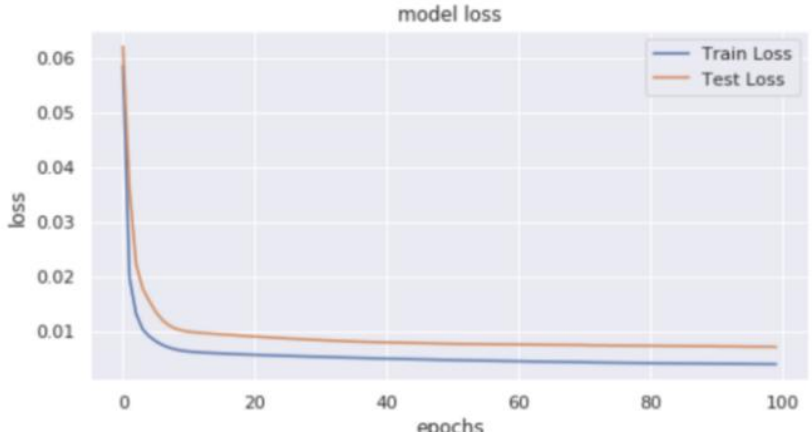

Рисунок 3.9 - метрики помилок та графік втрат LSTM

Порівнявши показники помилок між моделями, найкращі результати було отримано моделю LSTM. На її основі побудуємо прогноз(рис 3.10).

Прогноз зроблено на 65 днів при MAE дорівнює 3744. Іншими словами, 2 місяці прогнозних рекламних витрат становитимуть близько 3744 доларів проти фактичних витрат. Це дуже хороший прогноз.

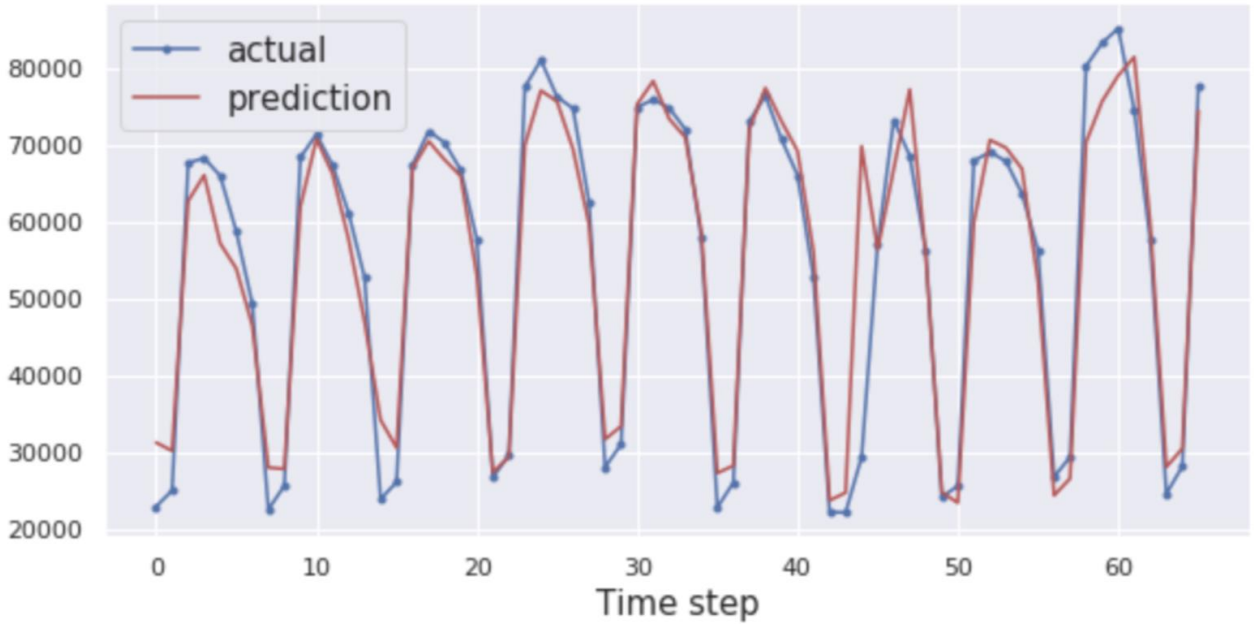

Рисунок 3.10 – прогноз на основі LSTM

Незважаючи на те, що LSTM не може ідеально зафіксувати всі вершини та западини, але він справді зробив кращу роботу з прогнозування порівняно з моделлю DNN або RNN у цьому випадку.

3.5 Висновки до розділу

В даному розділі розглянуто та обгрунтовано вибір програмної мови та архітектури, на яких базується реалізація алгоритму. Створено користувацький інтерфейс за допомогою якого можна з легкістю подивитись результати роботи алгоритму, а подивитись гіперпараметри моделі та порівняти результати. Було проведено якісний аналіз трьох моделей на датасеті та обрано найкращий, а також побудовано прогноз. Найкращою за результатими виявилася LSTM модель.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

На сьогоднішній день все більше стартапів перетворюють ідеї пов’язані із нейромережевими технологіями у працюючі бізнес моделі. Інтерес інвесторів до стартапів, що працюють із нейронними мережами зростає з кожним роком. Все більше і більше з’являється прикладів успішного застосування штучного інтелекту у найрізноманітніших галузях діяльності людини, значно зростає кількість корпорацій що впроваджують системи прогнозування із застосуванням нейромереж у свою операційну діяльність.

Окрім цього, нейронні мережі знайшли широке застосування в прогнозуванні. Нова і перспективна технологія швидко привернула увагу венчурних інвесторів. У цьому розділі розглянута розробка стартап проекту системи визначення найкращої моделі для нейромережі.

4.1. Опис ідеї проекту

Блок прогнозування є важливим компонентом інтелектуальної системи управління. Результати його роботи впливають на результат аналізу проблемної ситуації і, як наслідок, на вибір керуючих впливів для її усунення, здійснюваний системою управління. Ефективність роботи блоку прогнозування залежить від застосовуваних методів прогнозування. Кожен метод прогнозування має певну область застосування, в якій він ефективний. Від вибору методу залежить точність результату прогнозування. Отже, невизначеність при виборі методу прогнозування є фактором, що робить негативний вплив на достовірність результату роботи блоку прогнозування і на надійність системи управління в цілому. Даний факт дозволяє

говорити про високу актуальність проблеми вибору оптимальних методів прогнозування в залежності від області застосування. Сучасні системи прогнозування не враховують комплексно кількісні та якісні фактори, що впливають на зміну курсів акцій або враховують у векторному вигляді, який обмежує можливості представлення вхідних даних до нейромережі. Такі системи прогнозування зазвичай мають точність 53-58% вірного прогнозу. Цієї точності замало для того, щоб використовувати такі системи як повноцінний інструмент для економічного аналізу та прогнозування. Отже, ідеєю стартап проекту є вибір найкращої моделі прогнозування(табл. 4.1).

Таблиця 4.1 Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	1.Фінансова аналітика	Дешевша, порівняно з аналогами, краща точність прогнозування тренду
	2. Біо-інженерія	Можливість отримувати точний прогноз без особливих навичок в машинному навчання

Застосування зрозумілої математичної моделі спрощує реалізацію, а доведення кращих показників прогнозування може сприяти розповсюдженню системи. Безумовно, на ринку є аналоги. Для порівняння розробленої системи проведемо аналіз потенційних техніко-економічних переваг ідеї.

Метою аналізу техніко-економіних переваг є чітке виокремлення технічних і маркетингових особливостей розробленого продукту:

1) дослідження характеристик і властивостей розробленої системи;

2) дослідження конкурентів, товарів-аналогів, товарів-замінників та загальної ситуації на ринку де буде комерціалізувати стартам;
3) проведення порівняльного аналізу слабких, нейтральних та сильних характеристик розробленої системи.

Визначимо сильні, слабкі та нейтральних характеристик ідеї проекту (табл. 4.2).

Таблиця 4.2. Визначення сильних, слабких та нейтральних характеристик проекту.

№п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильнастор она)
		Мій проект	Neural Builder 2015	Neural Shell	Neurall Trader			
1.	Вартість ПЗ	Низь	Вис	Вис	Вис			+
2.	Доступність	Низь	Вис	Вис	Сер		+	
3.	Кроссплатфо	Так	Так	Ні	Ні			+
4.	Підтримка	+	-	+	+		+	

З таблиці можна зробити висновок, що розроблена система є конкурентоспроможною.

4.2. Технологічний аудит проекту.

Метою технічного аудиту є визначення переліку технологій, за допомогою який реалізована система і їх аналіз. Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 4.3):

1) за допомогою якої технологією розроблена система згідно ідеї проекту;
2) чи існують у відкритому доступі ці технології, чи їх потрібно додатково розробляти або купувати;
3) чи має доступ розробник до описаних технологій.

Таблиця 4.3. Технологічна здійсненність ідеї проекту

№п/п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Створення системи для найкращого вибору метода прогнозування за допомогою нейромережі	Рекурентні або згорткові нейронні мережі	Наявна	Доступна
Обраною мовою програмування є python, використовуються нейронні мережі із довгою короткостроковою пам’яттю				

За результатами аналізу можна зробити висновок щодо можливості технологічної реалізації проекту. Технологічним шляхом реалізації проекту було обрано мову програмування python через її доступність та безкоштовність.

4.3. Аналіз ринкових можливостей запуску стартап-проекту.

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 4.4).

Таблиця 4.4 - Попередня характеристика потенційного ринку стартап проекту

№ п/п	Показники стану ринку(найменування)	Характеристика
1	Кількість систем-конкурентів на ринку, од	4
2	Загальний обсяг продаж, грн/ум.од	234000
3	Динаміка ринку	Стагнує
4	Наявність обмежень для входу (вказати характер обмежень)	Нема
5	Специфічні вимоги до стандартизації та сертифікації	Нема

Проаналізувавши результати, можна зробити висновок що проект є придатним до інвестицій, оскільки його рентабельність перевищує відсоток депозиту. За результатами порівняння що було наведене у таблиці 4.4 було

зроблено висновок, що ринок є придатним для розповсюдження системи як стартап-проекту.

Після цього були досліджені потенційні категорії клієнтів, їх особливості та затверджено перелік вимог до системи для кожної категорії клієнтів (табл. 4.5)

Таблиця 4.5 — Характеристика потенційних клієнтів стартап-проект

№ п/п	Потреба, що формує ринок систем прогнозування	Цільова аудиторія та потенційні клієнти	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Програмне забезпечення для прогнозування	Компанії що займаються фінансовою аналітикою,	Цільові групи клієнтів мають різні доходи і потенційний прибуток від застосування системи, різні очікування та мотивація	Зручний інтерфейс, наявність інструкції з користування, кроссплатформеність.

Після дослідження потенційних категорій клієнтів було проведено дослідження ринкового середовища: складено таблиці факторів, що сприяють

реалізації розробленої системи як стартап-проекту, та факторів, що йому заважають (табл. 4.6, 4.7).

Таблиця 4.6 - Фактори загроз

№п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Вихід на ринок систем з кращими Характеристиками та моделлю надання послуг	Вийти на ринок зосередивши увагу на власної переваги системи. Покращення якісних характеристик системи прогнозування. Обрати цільову аудиторію
2	Зміна потреб користувачів	Клієнту потрібна буде система з більшою точністю прогнозування і новими функціями	Передбачити можливість розширення системи та підвищення точності прогнозування

Таблиця 4.7 - Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання можливостей потенційних покупців.	Ріст зацікавленості до продукту серед інших груп користувачів.	Додати підказки, інструкції та демонстрації роботи системи.
2	Зниження довіри до конкурента 3.	У ПЗ конкурента №3 нещодавно була знайдена помилка.	При виході на ринок звертати увагу покупців на безпеку нашого ПЗ.

Надалі було проведено дослідження пропозиції: визначили загальні харантеристик конкуренції на ринку (таблиця 4.8).

Таблиця 4.8 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому виражена дана характеристика	Вплив на діяльність компанії (можливі дії для підвищення конкурентоспроможності)
1.Вказати тип конкуренції - монополія	На ринку присутні декілька постачальників-конкурентів, але їх товар дещо відрізняється від нашої сситеми.	Підтримка якості системи, безперервний розвиток, покращення, вдосконалення, оновлення та підтримка.

Продовження таблиці 4.8

2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти з інших країн	Створити основу системи прогнозування таким чином, щоб можна було легко локалізувати її для використання у інших країнах.
3.Загалузевою ознакою - внутрішньогалузева	Система може застосовуватися в одній галузі, але її різних сферах.	Постійне вдосконалення системи прогнозування, що не має прив'язки до сфери, але має до галузі
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між видами систем прогнозування, їх особливостями.	Створити систему прогнозування, враховуючи недоліки конкурентів
5. За характером конкурентних переваг - цінова	Покращення процесу створення програмного продукту, мінімізація витрат на оновлення, застосування безперервної інтеграції	Використання відкритих та дешевших технологій для побудови системи в порівнянні з системами-конкурентами, але тільки якщо ці технології відповідають необхідним критеріям якості.
6. За інтенсивністю - не марочна	Бренд присутній, але його роль незначна	Реклама, участь у конференціях, семінарах, виставках.

Було проведено аналіз конкуренції у галузі за моделлю М. Портера (табл. 4.9).

Таблиця 4.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товаризамінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження в ринок	Визначити фактори сили постачальників	Визначити фактори сили споживачів в	Фактори загроз з боку замінників
Висновки:	Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 3 — його рішення представлене у вигляді ПЗ	Так, можливості для входу на ринок є, бо наше рішення покращує та пришвидшує роботу спеціаліста.	Постачальники відсутні	Важливим для користувача є кросплатформеність ПЗ та якість його роботи	Товари-замінники можуть використати більш дешеву технологію створення ПЗ та зменшити собівартість товару

За результатами порівняльного дослідження що наведене у табл. 4.9 було зроблено висновок про доцільнсть виходу на український та міжнародні ринки.

На основі аналізу ситуації на ринках, проведеного в табл. 4.9, а також із урахуванням характеристик розробленої системи (табл. 4.2), вимог потенційних клієнтів до товару (табл. 4.5) та особливостей ринкового середовища (таблиці 4.6, 4.7) досліджується та визначається перелік факторів конкурентоспроможності розробленої системи прогнозування фінансових ринків. Аналіз факторів конкурентоспроможності розробленої системи наведено у табл. 4.10

Таблиця 4.10 Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Виконання програмного забезпечення у кросплатформеному вигляді	Можливість використання програмного забезпечення на будь-якій платформі.
2	Ціна	Дане програмне рішення має невисоку ціну

За визначеними факторами конкурентоспроможності (табл. 4.10) проведено аналіз сильних та слабких сторін стартап-проекту (табл. 4.11).

Таблиця 4.11 — Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентноспроможності	Бали 1-20	Рейтинг систем-конкурентів у порівнянні з розробленою системою прогнозування						
			-3	-2	-1	0	+1	+2	+3
1	Ціна	15					*		
2	Кросплатформність	20			*				
3	Орієнтованість на кінцевого споживача	7					*		

Кінцевим кроком маркетингового дослідження можливостей реалізації системи прогнозування фінансових ринків як стартап-проекту є побудова SWOT-аналізу (матриці сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (таблиця 4.12) на основі описаних конкурентних та маркетингових загроз та можливостей, та сильних і слабких сторін (таблиця 4.11). Список маркетингових загроз та можливостей було складено на основі дослідження факторів загроз та факторів можливостей ринкової ситуації. Маркетингові загрози та можливості є наслідками (прогнозованими результатами) впливу ринкових факторів.

Таблиця 4.12 — SWOT-аналіз стартап-проекту

Сильні сторони: ціна, кросплатформеність	Слабкі сторони: у деяких випадках можуть стомлюватися руки
Можливості: більш широке розповсюдження технологій з підтримкою віртуальної реальності, поява нових технологій моніторингу навколишнього середовища.	Загрози: видавлювання з ринку конкурентами, зміна потреб користувачів.

За результатами SWOT-аналізу було сформовано альтернативи ринкової стратегії (перелік заходів) для виведення розробленої системи як стартап-проекту на український та міжнародні ринки та розрахований оптимальний час їх ринковоїреалізації з урахуванням потенційних розробок конкурентів, що можуть бути виведені на ринок (див. таблицю 4.9, аналіз потенційних конкурентів). Запропоновані альтернативи були проаналізовані з точки зору часу реалізації та ймовірності отримання ресурсів (таблиця 4.13)

Таблиця 4.13 — Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розробка програмного продукту, PR, просування бренду	90%	9 місяців
2	Розробка програмного продукту, безкоштовне розповсюдження	60%	6 місяців

4.4. Розроблення ринкової стратегії проекту.

Розроблення ринкової стратегії першим етапом передбачає визначення стратегії охоплення ринку: дослідження цільових груп потенційних клієнтів, які приведенні в таблиці 4.14.

Таблиця 4.14. Вибір цільових груп потенційних споживачів

№п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент

Продовження таблиці 4.14

1	Компанії які займаються прогнозуванням	Готові	Необхідно	Висока	Середня

Визначена цільова група клієнтів: Компанії які займаються прогнозуванням

Дослідивши потенційні групи клієнтів було визначено цільові категорію, для яких буде пропонуватися розроблена система, та визначено стратегію охоплення ринку - стратегію диференційованого маркетингу.

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Розробка програмного продукту, PR, просування бренду	Масовий маркетинг	Екстрановий спосіб рухової взаємодії, який не потребує громіздких додаткових девайсів	Стратегія диференціації

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопроходцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Забирати існуючих	Ні	Стратегія наслідування

Таблиця 4.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Невисока ціна	Позиціювання за показниками ціни	Ціна	Ціна, зручність, швидкість роботи

За результатами аналізу вимог клієнтів визначених категорій до розробника стартап-проекту та до продукту (див. таблицю 4.5), а також в залежності від обраної базової стратегії розвитку (таблиця 4.15) та стратегії конкурентної поведінки (таблиця 4.6) розроблено стратегію позиціонування (таблиця 4.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект. За результатами дослідження була

сформована система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап- компанії українському та міжнародному ринках.

4.5. Розроблення маркетингової програми стартап-проекту

Визначено маркетингову концепцію системи вибору найкращого методу прогнозування, яку буде купувати споживач. Основна концепція продукту — письмовий опис фізичних та програмних характеристик системи, які сприймаються споживачем, і набору вигод, які він обіцяє певній групі споживачів.

За результатами дослідження було сформовано трирівневу маркетингову модель системи: ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання:

- 1-й рівень вирішує питання щодо того, засобом вирішення якої потреби і / або проблеми буде система, яка її основна вигода;
- 2-й рівень являє рішення того, як буде реалізована система в реальному ринковому середовищі. Рівень включає в себе якість, властивості, дизайн, упаковку, ціну;
- 3-й рівень визначає додаткові послуги та переваги для клієнта системи, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін).

Після цього визначимо цінові межи, якими необхідно керуватись при встановленні ціни на систему, яке включає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової категорії клієнтів Аналіз проведено експертним метод.Фінальною складовою маркетингової програми є визначення концепції маркетингових комунікацій.

Результатом дослідження стала робоча ринкова програма, що включає в себе концепції системи, збуту, просування та попереднє дослідження ціноутворення на продукт, базується на цінностям та потребах категорій покупців, конкурентні переваги системи, стан та динаміку маркетингового середовища, в межах якого впроваджено системи прогнозування фінансових ринків яз стартап-проект, та відповідну обрану альтернативу ринкової поведінки.

4.6. Висновки до розділу

В цьому розділі було проведено аналіз розробленої системи прогнозування фінансових ринків у якості стартап-проекту. Варто зауважити, що проект має можливість ринкової комерціалізації, через те, що ринок систем прогнозування потребує якісного та інноваційного продукту для вибору найкращого методу прогнозування в різних областях.

Результатом роботи є розроблений стартап-проект, план виходу на ринки програмного забезпечення та маркетингова стратегія. Розроблений стартап-проект доцільно застосувати при комерціалізації розробки.

Вискока конкуренція зумолює необхідність виваженого підходу до просування продукту. Проте кількість учасників фінансових ринків зростає з кожним днем, що зумовлює хороші перспективи для комерціалізації системи. Для впровадження ринкової реалізації проекту була обрана стратегія, яка включає розробку програмної системи із класичною моделлю ліцензування за певну плату.

Можна сказати, що подальший розвиток проекту є доцільним, оскільки кількість потенційних користувачів системи зростає кожного року.

ВИСНОВКИ

У процесі виконання магістерської дисертації було, по-перше, вивчено і проаналізовано предметну область, що стосується теми дипломної роботи, а саме зроблено огляд моделей прогнозування, досліджено його напрями та задачі.

По-друге, було наведено детальну формалізацію нейромережевих моделей, способи їх реалізації, методи їх навчання, метрики виміру якості.

По-третє, була реалізована програма, яка має зручний та зрозумілий інтерфейс. Користувач має має змогу подивитись на параметри моделі, порівняти результати і вибрати найкращу модель.

В кінці було запропоновано стартап-проект, що має на меті впровадження програми яка базується на виборі найкращої моделі нейромережі.

Результатом виконання дипломної роботи є реалізація моделей прогнозування, що вирішують складні, але дуже важливі проблеми бізнесу, дають змогу ефективно вести політику компанії та впроваджувати штучний інтелект в будь-якій сфері.

ПЕРЕЛІК ПОСИЛАНЬ

1. Савчук О.В., Кривенко К.С. Інтелектуальний аналіз діагностичної інформації електро- радіокомпонентів в умовах невизначеності: Інтелектуальний аналіз інформації: зб. праць. Київ: Просвіта, 2013. 217 с.
2. Терехов В.А., Ефимов Д.В., Тюкин И.Ю. Нейросетевые системы управления: учеб. пособие для вузов. Москва: Высшая школа, 2002. 183 с.
3. Горбань А.Н. Нейронные сети на персональном компьютере. Новосибирск: Наука, 2006. 230 с.
4. Тихонов Э.Е. Прогнозирование в условиях рынка. Москва: Мир, 2006. 221 с.
5. Царегородцев В.Г. Перспективы распараллеливания программ снейросетевого анализа и обработки данных: материалы с III Всерос. конф. «*Математика, информатика, управление* – 2008». Иркутск, 2014. 114 с.
6. Рутковская Д.А., Пилиньский М.О., Рутковский Л.Р. Нейронные сети, генетические алгоритмы и нечёткие системы: пер. с польск. И.Д. Рудинского. Москва: Горячая линия Телеком, 2006. 452 с.
7. Саймон Хайкин Нейронные сети: полный курс, 2-е издание: пер. с англ. Москва: Издательский дом "Вильямс", 2006. 1045 с.
8. Осовский С. Нейроные сети для обработки информации: пер. с польского. Москва: Финансы и статистика, 2002. 344 с.
9. Круглов В.В. Нечёткая логика и искусственные нейронные сети. Москва: Физматлит, 2001. 200-224 с.
10. Дюк В.В, Самойленко А.В. Data Mining: учебный курс, СПб: Питер. 2001. 368 с.
11. Нейромережеве відображення дійсності.– Режим доступу: http://studies.in.ua/mpd_seminar/1313-neyrmerezh.html Дата доступу: 29.09.2019

12. Моделі нейронних мереж – Режим доступу:
https://studme.com.ua/1246122010028/neural/models.htm
Дата доступу: 27.08.2019
13. Моделі нейронних мереж. – Режим доступу:
http://techn.sstu.ru/kafedri/подразделения/1/MetMat/Terin/neiro/neiro.htm
Дата доступу: 28.09.2019
14. Simonyan K. Very Deep Convolutional Networks for Large-Scal Image Recognition – Режим доступу до ресурсу:
http://arxiv.org/abs/1409.1556
Дата доступу: 08.05.2019
15. Improving neural networks by preventing co-adaptation of feature detectors / G. Hinton, S. Nitish, A. Krizhevsky et. al. 2012. – Режим доступу до ресурсу:
https://arxiv.org/abs/1207.0580
Дата доступу: 08.05.2019
16. Learning Deep Features for Discriminative Localization / B. Zhou, A. Khosla, A. Lapedriza et. al. 2015. – Режим доступу до ресурсу:
https://arxiv.org/abs/1512.04150
Дата доступу: 08.05.2019
17. Visual Explanations from Deep Networks via Gradient-based Localization / R. Selvaraju, M. Cogswell, A. Das et. al. 2017. – Режим доступу до ресурсу: https://arxiv.org/abs/1610.02391
Дата доступу: 08.05.2018

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>TimeSeries@TensorFlow.js</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@0.13.3/dist/tf.min.js"></script>
<script src="https://cdn.plot.ly/plotly-1.2.0.min.js"></script>
<script src="./src/generators.js"></script>
<script src="./src/model.js"></script>

<script type="text/javascript">

var input_dataset = [], result = [];
var data_raw = []; var sma_vec = [];

function Init() {
initTabs('Dataset'); initDataset();
document.getElementById("n-items").value = "50";
document.getElementById("window-size").value = "12";
document.getElementById('input-data').addEventListener('change', readInputFile, false);
```

```
}

function initTabs(tab) {
var navbar = document.getElementsByClassName("nav navbar-nav");
navbar[0].getElementsByTagName("li")[0].className += "active";
document.getElementById("dataset").style.display = "none";
document.getElementById("graph-plot").style.display = "none";

setContentView(tab);
}

function setTabActive(event, tab) {
var navbar = document.getElementsByClassName("nav navbar-nav");
var tabs = navbar[0].getElementsByTagName("li");
for (var index = 0; index < tabs.length; index++)
if (tabs[index].className == "active")
tabs[index].className = "";

if (event.currentTarget != null) {
event.currentTarget.className += "active";
}

var callback = null;
if (tab == "Neural Network") {
callback = function () {
document.getElementById("train_set").innerHTML = getSMATable(1);
}
}
```

```
setContentView(tab, callback);
}

function setContentView(tab, callback) {
var tabs_content = document.getElementsByClassName("container");
for (var index = 0; index < tabs_content.length; index++)
tabs_content[index].style.display = "none";

if (document.getElementById(tab).style.display == "none")
document.getElementById(tab).style.display = "block";

if (callback != null) {
callback();
}
}

function readInputFile(e) {
var file = e.target.files[0];
var reader = new FileReader();
reader.onload = function(e) {
var contents = e.target.result;
document.getElementById("input-data").value = "";
parseCSVData(contents);
};
reader.readAsText(file);
}

function parseCSVData(contents) {
data_raw = []; sma_vec = [];
```

```
var rows = contents.split("\n");

var params = rows[0].split(",");
var size = parseInt(params[0].split("=")[1]);
var window_size = parseInt(params[1].split("=")[1]);

document.getElementById("n-items").value = size.toString();
document.getElementById("window-size").value = window_size.toString();

for (var index = 1; index < size + 1; index++) {
var cols = rows[index].split(",");
data_raw.push({ id: cols[0], timestamp: cols[1], price: cols[2] });
}

sma_vec = ComputeSMA(data_raw, window_size);
onInputDataClick();
}

function initDataset() {

var n_items = parseInt(document.getElementById("n-items").value);
var window_size = parseInt(document.getElementById("window-size").value);

data_raw = GenerateDataset(n_items);
sma_vec = ComputeSMA(data_raw, window_size);

onInputDataClick();
}
```

```
async function onTrainClick() {

var inputs = sma_vec.map(function(inp_f) {
return inp_f['set'].map(function(val) { return val['price']; })});
var outputs = sma_vec.map(function(outp_f) { return outp_f['avg']; });

var n_epochs     = parseInt(document.getElementById("n-epochs").value);
var window_size  = parseInt(document.getElementById("window-size").value);
var lr_rate      = parseFloat(document.getElementById("learning-rate").value);
var n_hl         = parseInt(document.getElementById("hidden-layers").value);
var n_items      = parseInt(document.getElementById("n-items-percent").value);

var callback = function(epoch, log) {
var log_nn = document.getElementById("nn_log").innerHTML;
log_nn += "<div>Epoch: " + (epoch + 1) + " Loss: " + log.loss + "</div>";
document.getElementById("nn_log").innerHTML = log_nn;
document.getElementById("training_pg").style.width = ((epoch + 1) * (100 /
n_epochs)).toString() + "%";
document.getElementById("training_pg").innerHTML = ((epoch + 1) * (100 /
n_epochs)).toString() + "%";
}

result = await trainModel(inputs, outputs,
n_items, window_size, n_epochs, lr_rate, n_hl, callback);

alert('Your model has been successfully trained...');
}

function onPredictClick(view) {
var inputs = sma_vec.map(function(inp_f) {
```

```
return inp_f['set'].map(function (val) { return val['price']; }); });
var outputs = sma_vec.map(function(outp_f) { return outp_f['avg']; });

var n_items = parseInt(document.getElementById("n-items-percent").value);

var outps = outputs.slice(Math.floor(n_items / 100 * outputs.length), outputs.length);

var pred_vals = Predict(inputs, n_items, result['model']);

var data_output = "";
for (var index = 0; index < pred_vals.length; index++) {
data_output += "<tr><td>" + (index + 1) + "</td><td>" +
outps[index] + "</td><td>" + pred_vals[index] + "</td></tr>";
}

document.getElementById("pred-res").innerHTML = "<table class=\"table\"><thead><tr><th
scope=\"col\">#</th><th scope=\"col\">Real Value</th> \
<th scope=\"col\">Predicted Value</th></thead><tbody>" + data_output + "</tbody></table>";

var window_size  = parseInt(document.getElementById("window-size").value);

var timestamps_a = data_raw.map(function (val) { return val['timestamp']; });
var timestamps_b = data_raw.map(function (val) {
return val['timestamp']; }).splice(window_size, data_raw.length);

var timestamps_c = data_raw.map(function (val) {
return val['timestamp']; }).splice(window_size + Math.floor(n_items / 100 * outputs.length),
data_raw.length);

var sma = sma_vec.map(function (val) { return val['avg']; });
```

```
var prices = data_raw.map(function (val) { return val['price']; });

var graph_plot = document.getElementById('graph-pred');
Plotly.newPlot( graph_plot, [{ x: timestamps_a, y: prices, name: "Series" }], { margin: { t: 0 } }
);
Plotly.plot( graph_plot, [{ x: timestamps_b, y: sma, name: "SMA" }], { margin: { t: 0 } } );
Plotly.plot( graph_plot, [{ x: timestamps_c, y: pred_vals, name: "Predicted" }], { margin: { t: 0
} } );
}

function getInputDataTable() {
var data_output = "";
for (var index = 0; index < data_raw.length; index++)
{
data_output += "<tr><td>" + data_raw[index]['id'] + "</td><td>" +
data_raw[index]['timestamp'] + "</td><td>" + data_raw[index]['price'] + "</td></tr>";
}

return "<table class=\"table\"><thead><tr><th scope=\"col\">#</th><th
scope=\"col\">Timestamp</th> \
<th scope=\"col\">Feature</th></thead><tbody>" + data_output + "</tbody></table>";
}

function getSMATable(view) {
var data_output = "";
if (view == 0) {
for (var index = 0; index < sma_vec.length; index++)
{
var set_output = "";
var set = sma_vec[index]['set'];
```

```
for (var t = 0; t < set.length; t++) {
set_output += "<tr><td width=\"30px\">" + set[t]['price'] +
"</td><td>" + set[t]['timestamp'] + "</td></tr>";
}

data_output += "<tr><td width=\"20px\">" + (index + 1) +
"</td><td>" + "<table width=\"100px\" class=\"table\">" + set_output +
"<tr><td><b>" + "SMA(t) = " + sma_vec[index]['avg'] + "</b></tr></td></table></td></tr>";
}

return "<table class=\"table\"><thead><tr><th scope=\"col\">#</th><th scope=\"col\">Time Series</th>\
</thead><tbody>" + data_output + "</tbody></table>";
}

else if (view == 1) {
var set = sma_vec.map(function (val) { return val['set']; });

for (var index = 0; index < sma_vec.length; index++)
{
data_output += "<tr><td width=\"20px\">" + (index + 1) +
"</td><td>[ " + set[index].map(function (val) {
return (Math.round(val['price'] * 10000) / 10000).toString(); }).toString() +
" ]</td><td>" + sma_vec[index]['avg'] + "</td></tr>";
}

return "<table class=\"table\"><thead><tr><th scope=\"col\">#</th><th scope=\"col\">\
Input</th><th scope=\"col\">Output</th></thead><tbody>" + data_output +
"</tbody></table>";
}
```

```
        }

        function onInputDataClick() {
        document.getElementById("dataset").style.display = "block";
        document.getElementById("graph-plot").style.display = "block";
        document.getElementById("data").innerHTML = getInputDataTable();

        var timestamps = data_raw.map(function (val) { return val['timestamp']; });
        var prices = data_raw.map(function (val) { return val['price']; });

        var graph_plot = document.getElementById('graph');
        Plotly.newPlot( graph_plot, [{ x: timestamps, y: prices, name: "Stocks Prices" }], { margin: { t:
0 } } );
        }

        function onSMAClick() {
        document.getElementById("data").innerHTML = getSMATable(0);

        var sma = sma_vec.map(function (val) { return val['avg']; });
        var prices = data_raw.map(function (val) { return val['price']; });

        var window_size = parseInt(document.getElementById("window-size").value);

        var timestamps_a = data_raw.map(function (val) { return val['timestamp']; });
        var timestamps_b = data_raw.map(function (val) {
        return val['timestamp']; }).splice(window_size, data_raw.length);

        var graph_plot = document.getElementById('graph');
        Plotly.newPlot( graph_plot, [{ x: timestamps_a, y: prices, name: "Series" }], { margin: { t: 0 } }
);
```

```
Plotly.plot( graph_plot, [{ x: timestamps_b, y: sma, name: "SMA" }], { margin: { t: 0 } } );
}


</script>
</head>
<body onload="Init()">
<table>
<tbody>
<tr>
<td>
<nav class="navbar navbar-default">
<div class="container-fluid">
<div class="navbar-header">
<a class="navbar-brand" href="#">TimeSeries@TensorFlow.js</a>
</div>
<ul class="nav navbar-nav">
<li onclick="setTabActive(event, 'Dataset')"><a href="#">Dataset</a></li>
<li onclick="setTabActive(event, 'Neural Network')"><a href="#">Neural Network</a></li>
<li onclick="setTabActive(event, 'Prediction')"><a href="#">Prediction</a></li>
</ul>
</div>
</nav>
</td>
</tr>
<tr>
<td>
<div id="Dataset" class="container">
<table width="100%">
<tr>
```

```
<td>
<table width="100%">
<tr>
<td width="60%" align="left">
<table width="100%">
<tr>
<td width="10px"><b> N-Items: </b></td>
<td width="120px"><input class="form-control input-sm" id="n-items" type="text" size="1" value="500"></td>
<td width="120px"><b>   Window Size: </b></td>
<td width="100px"><input class="form-control input-sm" id="window-size" type="text" size="1" value="12"></td>
<td width="180px" align="center"><button type="button" class="btn btn-primary" onclick="initDataset()">Generate Data...</button></td>
</tr>
</table>
</td>
<td width="40%" align="right">
<form class="md-form">
<div class="file-field">
<div class="btn btn-primary btn-sm float-left">
<span>select *.csv data file</span>
<input type="file" id="input-data">
</div>
</div>
</form>
</td>
</tr>
</table>
</td>
```

```
</tr>

<tr>

<td width="100%" id="dataset"><hr/>

<table width="50%">

<tr>

<td align="left"><button type="button" class="btn btn-primary"
onclick="onInputDataClick()">Input Data</button></td>

<td align="right"><button type="button" class="btn btn-primary"
onclick="onSMAClick()">Simple Moving Average</button></td>

</tr>

</table>

<hr/>

<div id="data" style="overflow-y: scroll; max-height: 300px;"></div>

</td>

</tr>

<tr><td width="100%" id="graph-plot"><hr/><div id="graph" style="width:100%;
height:350px;"></div></td></tr>

</table>

</div>

<div id="Neural Network" class="container">

<table width="100%">

<tr>

<td>

<button type="button" class="btn btn-primary" onclick="onTrainClick()">Train
Model...</button><hr/>

<div class="progress">

<div id="training_pg" class="progress-bar" role="progressbar" aria-valuenow="70" aria-
valuemin="0" aria-valuemax="100" style="width:0%"></div>

</div>

<hr/>

</td>
```

```
</tr>
<tr>
<td>
<table width="100%" height="100%">
<tr>
<td width="80%"><div id="train_set" style="overflow-x: scroll; overflow-y: scroll; max-width: 900px; max-height: 300px;"></div></td>
<td>
<table width="100%" class="table">
<tr>
<td>
<label>Size (%):</label>
<input class="form-control input-sm" id="n-items-percent" type="text" size="1" value="50">
</td>
</tr>
<tr>
<td>
<label>Epochs:</label>
<input class="form-control input-sm" id="n-epochs" type="text" size="1" value="200">
</td>
</tr>
<tr>
<td>
<label>Learning Rate:</label>
<input class="form-control input-sm" id="learning-rate" type="text" size="1" value="0.01">
</td>
</tr>
<tr>
<td>
<label>Hidden Layers:</label>
```

```
<input class="form-control input-sm" id="hidden-layers" type="text" size="1" value="4">
</td>
</tr>
</table>
</td>
</tr>
<tr>
<td><hr/><div id="nn_log" style="overflow-x: scroll; overflow-y: scroll; max-width: 900px; max-height: 250px;"></div></td>
</tr>
</table>
</td>
</tr>
</table>
</div>
<div id="Prediction" class="container">
<table width="100%">
<tr><td><button type="button" class="btn btn-primary" onclick="onPredictClick()">Predict</button><hr/></td></tr>
<tr><td><div id="pred-res" style="overflow-x: scroll; overflow-y: scroll; max-height: 300px;"></div></td></tr>
<tr><td id="graph-pred"><hr/><div id="graph" style="height:300px;"></div></td></tr>
</table>
</div>
</td>
</tr>
</tbody>
</table>
</body>
</html>
```

run.py:

```
def convert2matrix(data_arr, look_back):
 X, Y =[], []
 for i in range(len(data_arr)-look_back):
  d=i+look_back
  X.append(data_arr[i:d,0])
  Y.append(data_arr[d,0])
 return np.array(X), np.array(Y)

train_size = 900
train, test =df1.values[0:train_size,:],df1.values[train_size:len(df1.values),:]
# setup look_back window
look_back = 30
#convert dataset into right shape in order to input into the DNN
trainX, trainY = convert2matrix(train, look_back)
testX, testY = convert2matrix(test, look_back)model.js:

def create_model():
    # default values
    activation='relu' # or linear
    dropout_rate=0.0 # or 0.2
    init_mode='uniform'
    weight_constraint=0 # or  4
    optimizer='adam' # or SGD
    lr = 0.01
    momemntum=0
    # create model
```

```
    model = Sequential()
    model.add(Dense(8,
                input_dim=input_dim, kernel_initializer=init_mode,
                activation=activation,
                kernel_constraint=maxnorm(weight_constraint)))
    model.add(Dropout(dropout_rate))
    model.add(Dense(1, kernel_initializer=init_mode, activation='sigmoid'))
    # Compile model
    model.compile(loss='binary_crossentropy',
                optimizer=optimizer,
                metrics=['accuracy'])
    return model


# create model
model = KerasClassifier(build_fn=create_model, batch_size=1000, epochs=10)
# use verbose=0 if you do not want to see progress

############################################################
# Use scikit-learn to grid search
activation =  ['relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear'] # softmax, softplus, softsign
momentum = [0.0, 0.2, 0.4, 0.6, 0.8, 0.9]
learn_rate = [0.001, 0.01, 0.1, 0.2, 0.3]
dropout_rate = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
weight_constraint=[1, 2, 3, 4, 5]
neurons = [1, 5, 10, 15, 20, 25, 30]
init = ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal',
'he_uniform']
optimizer = [ 'SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
############################################################
# grid search epochs, batch size
```

```
epochs = [1, 10] # add 50, 100, 150 etc
batch_size = [1000, 5000] # add 5, 10, 20, 40, 60, 80, 100 etc
param_grid = dict(epochs=epochs, batch_size=batch_size)
#################################################################
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X, Y)
#################################################################
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
def model_dnn(look_back):
    model=Sequential()
    model.add(Dense(units=32, input_dim=look_back, activation='relu'))
    model.add(Dense(8, activation='relu'))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error',  optimizer='adam',metrics = ['mse', 'mae'])
    return model
model.add(tf.layers.dense({units: input_layer_neurons, inputShape: [input_layer_shape]}));
model.add(tf.layers.reshape({targetShape: rnn_input_shape}));

def model_loss(history):
    plt.figure(figsize=(8,4))
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Test Loss')
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epochs')
    plt.legend(loc='upper right')
    plt.show();
```

```
train_score = model.evaluate(trainX, trainY, verbose=0)
print('Train Root Mean Squared Error(RMSE): %.2f; Train Mean Absolute Error(MAE) : %.2f '
% (np.sqrt(train_score[1]), train_score[2]))
test_score = model.evaluate(testX, testY, verbose=0)
print('Test Root Mean Squared Error(RMSE): %.2f; Test Mean Absolute Error(MAE) : %.2f '
% (np.sqrt(test_score[1]), test_score[2]))
model_loss(history)
```