НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
(повненайменування інституту, факультету)

Автоматизованих систем обробки інформації і управління
(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Програмний комплекс для розв’язування біомеханіки опорно-рухового апарату за допомогою машинного навчання*

Виконав: студент IV курсу, групи

ІП-61 Смірнов Денис Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. викл. Недашківський Є.А.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Консультант з графічної документації *доц. к.т.н. Ліщук К. І.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Рецензент:

доцент кафедри електронної інженерії, к.т.н., Попов А.О. _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
12.06.2020 15:46:20 EEST

Дата звіту:
12.06.2020 15:51:01 EEST

ID перевірки:
1004000090

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: Smirnov_ip61

ID файлу: 1004013038 **Кількість сторінок:** 44 **Кількість слів:** 7595 **Кількість символів:** 57274 **Розмір файлу:** 87.43 KB

7.5% Схожість

Найбільша схожість: 4.88% з джерело бібліотеки. **ID файлу:** 1000020588

5.48% Схожість з Інтернет джерелами	15	Page 46
7.37% Текстові збіги по Бібліотеці акаунту	150	Page 46

0.28% Цитат

Цитати	3	Page 47

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

Підміна символів

Не знайдено замінених символів

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)

“___”_____2020р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Смірнова Дениса Сергійовича
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Програмний комплекс для розв’язування біомеханікиопорно-рухового апарату за допомогою машинного навчання»*

керівник проєкту *Недашківський Євген Анатолійович, ст. викл.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “07” травня 2020р. №1081-с

2. Термін подання студентом проєкту«08» *червня 2020 року*

3.Вихідні дані до проєкту

Технічне завдання

4.Зміст пояснювальної записки

1)Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог

2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмногозабезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення

3) Розгортання та впровадження програмного забезпечення

4) Керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1) Діаграма розгортання програмного забезпечення

2) Схема структурна класів програмного забезпечення

3) Структура нейронної мережі

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7.Дата видачі завдання«10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*		
2.	*Аналіз існуючих методів розв'язання задачі*		
3.	*Постановка та формалізація задачі*		
4.	*Аналіз вимог до програмного забезпечення*		
5.	*Алгоритмізація задачі*		
6.	*Моделювання програмного забезпечення*		
7.	*Обґрунтування використовуваних технічних засобів*		
8.	*Розробка архітектури програмного забезпечення*		
9.	*Розробка програмного забезпечення*		
10.	*Налагодження програми*		
11.	*Виконання графічних документів*		
12.	*Оформлення пояснювальної записки*		
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*		
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____Смірнов Д.С.
(підпис)

Керівник проєкту _____Недашківський Є.А.
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	A4		Завдання на дипломний проект	2	
2	A4	КПІ.ІП-6101.045490.01.81	Пояснювальна записка	51	
3	A4	КПІ.ІП-6101.045490.02.13	Опис програми	41	
4	A4	КПІ.ІП-6101.045490.03.81	Технічне завдання	10	
5	A4	КПІ.ІП-6101.045490.04.51	Програма та методика тестування	4	
6	A4	КПІ.ІП-6101.045490.05.34	Керівництво користувача	4	
7	A3	КПІ.ІП-6101.045490.06.99 СС	Схема структурна класів програмного забезпечення	1	
8	A3	КПІ.ІП-6101.045490.06.99 СС	Діаграма розгортання програмного забезпечення	1	
9	A3	КПІ.ІП-6101.045490.06.99 Р	Структура нейронної мережі	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6101.045490.00.90		
Розробн.		Смірнов Д.С.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Недашківський Є.А.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-61	
Н/контр.		Ліщук К.І.					
В.о.зав.каф		Павлов О.А.					

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6101.045490.02.81	«Програмний комплекс для розв'язування біомеханіки опорно-рухового апарату за допомогою машинного навчання».		
			Пояснювальна записка	51	
3	А4	КПІ.ІП-6101.045490.03.81	«Програмний комплекс для розв'язування біомеханіки опорно-рухового апарату за допомогою машинного навчання».		
			Технічне завдання	10	
4	А4	КПІ.ІП-6101.045490.04.51	«Програмний комплекс для розв'язування біомеханіки опорно-рухового апарату за допомогою машинного навчання».		
			Програма та методика тестування	6	

АНОТАЦІЯ

До бакалаврської роботи дипломної роботи Смірнов Дениса Сергійовича на тему: «Програмний комплекс для розв’язування біомеханіки опорно-рухового апарату за допомогою машинного навчання».

Дипломна робота присвячена розробці програмного забезпечення для розв’язування частини задачі біомеханіки опорно-рухового апарату за допомогою машинного навчання. У роботі проведено аналіз існуючих підходів та розроблено програмний комплекс для тренування та аналізу моделей.

У першому розділі було детально проаналізовано предметну область, наведено загальну схему перетворення сигналів електроміографії у механіку рухів, проаналізовані існуюч підходи для розв’язування задачі перетворення постави опорно-рухового апарату в м’язові характеристики та на їх основі розроблено вимоги до програмного забезпечення.

У другому розділі наведено опис розробки програмного комплексу для даної задачі, проведено аналіз використаних технологій та розроблена архітектура програмного забезпечення.

У третьому розділі наведено підходи до тестування програмного забезпечення та описано контрольний приклад.

У четвертому розділі наведена інструкція по розгортанню розроблюваного програмного забезпечення та інструкція користувача.

Загальний обсяг роботи: 51 сторінки, 14 рисунків, 7 таблиць, 21 джерел.

Ключові слова: МАШИННЕ НАВЧАННЯ, БІОМЕХАНІКА ОПОРНО-РУХОВОГО АПАРАТУ.

ABSTRACT

Abstract of the bachelor thesis of Smirnov Denys: “Software for solving musculoskeletal biomechanics with machine learning”.

This thesis is devoted to the development of software for solving a part of the musculoskeletal biomechanics using machine learning. In this work a comparative analysis of existing approaches for the decoding EMG signals to the movement was done and the software for the model training and analyzing was developed.

In the first section, the subject area was analyzed, general schema of decoding EMG signals into the movement was presented, existing approaches for the solving of transforming posture into muscle properties were analyzed and the software requirements on their basis were developed.

In the second section the process of software development was described, a comparative analysis of used technologies was conducted and the software architecture was developed.

In the third section approaches for software testing and the main task case were described.

In the fourth section the instruction for the deploying of the software and user instruction were provided.

Work includes 51 pages, 14 figures, 7 tables, 21 citations.

Keyword: MACHINE LEARNING, MUSCULOSKELETAL BIOMECHANICS.

Пояснювальна записка
до дипломного проеєкту

на тему: “Програмний комплекс для розв’зування біомеханіки опорно-рухового апарату за допомогою машинного навчання”

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЕМГ – електроміограма.

ANN – штучна нейронна мережа.

HTTP – протокол передачі інформації між інформаційними вузлами мережі.

HTTPS – захищена версія протоколу передачі даних HTTP.

LGB – легкий метод градієнтного підсилювання.

R&D – дослідження та розробка.

REST – шаблон мережевих протоколів для забезпечення обміну інформацією між інформаційними вузлами в мережі.

ВСТУП

Головна мета розробників протезів – створити зручний в керуванні пристрій, який легко придбати, одягнути та користуватися без складного процесу навчання. Людство з часів середньовіччя і по сьогоднішній день прагне створити протези – найбільш схожі на втрачену кінцівку як зовні, так і функціонально. На разі поширеними є біонічні протези, які механічно найбільш наближені до функціоналу тіла здорової людини, однак проблема якісного управління такими пристроями на сьогоднішній день не має готового рішення.

На даний момент всі R&D проєкти сфокусовані на двох напрямках: здешевлення самого протезу та поліпшення системи управління. Якщо для першої проблеми існують відносно готові рішення, то в області розробки систем управління все знаходиться на етапі експерименту. Найбільшої уваги вчені та розробники приділяють вивченню такого типу взаємодії людини з протезом як електроміографія. Електроміографія – це метод аналізу м'язової активності, заснований на вимірюванні різниці потенціалів в двох точках, між якими під шкірою по мембранами м'язових волокон поширюється потенціал дії. В основному, розробники використовують вже готові електроміографічні підсилювачі, і, отримавши сигнал, його обробляють. Проблема цього підходу полягає в кількості ступенів свободи кінцівки та кількості каналів запису ЕМГ, що нелінійно ускладнює створення такого типу протезу.

Розробка швидкого та інтуїтивного інтерфейсу з кінцівкою, яка має велику кількість ступенів свободи – складне завдання, яке все частіше намагаються вирішити за допомогою алгоритмів розпізнавання шаблонів на основі алгоритмів машинного навчання. Протягом останніх десятиліть порівняно невелика кількість досліджень була присвячена застосуванню методів машинного навчання для опорно-рухової динаміки. Було розроблено кілька типів нейронних мереж, наприклад, рекурентні нейронні мережі[1], вейвлет-нейронна мережа [2]. Було також досліджено те, що поєднання згорткових та рекурентних штучних нейронних мереж точно і надійно

відображає перетворення сигналів від представлення багатоканального ЕМГ-сигналу до руху кінцівок [3]. Конференція NeurIPS безперервно підтримує різні застосування засобів машинного навчання з підкріпленням для вирішення проблем на перетині нейронауки, біомеханіки та оптимального контролю. Тут проводяться глобальні відкриті змагання з навчання з підсиленням, наприклад, розробка контролера, який дозволить моделі, заснованій на фізіології людини, з протезом замість нижньої кінцівки рухатися з заданим напрямком і швидкістю [4] та побудувати контролер для моделі опорно-рухового апарату людини, щоб змусити модель “подорожувати” настільки далеко наскільки можливо протягом 10 секунд, імітуючи опорно-руховий апарат людини [5].

Вчені в сфері машинного навчання частіше використовували обчислювально важкі методи глибокого навчання та навчання з підкріпленням. Це може бути пояснено тим, що біологічна система демонструє складну поведінку в просторі часу, проблема має високо-розмірний простір рішень, навколишнє середовище є нелінійним. До того ж отримання експериментальних даних, необхідних для навчання моделей, стає більш легким, що дозволяє тренувати моделі з більш складною архутектурою.

У даній роботі метою було обрано вивчення динаміки опорно-рухового апарату за допомогою декількох простих технологій машинного навчання. Для цього була поставлена ціль розробити програмний комплекс для тренування та аналізу моделей машинного навчання для розв`язування частини біомеханіки опорно-рухового апарату та зручний веб-додаток для взаємодії та аналізу цих моделей.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

1.1.1 Загальне визначення моделі опорно-рухового апарату

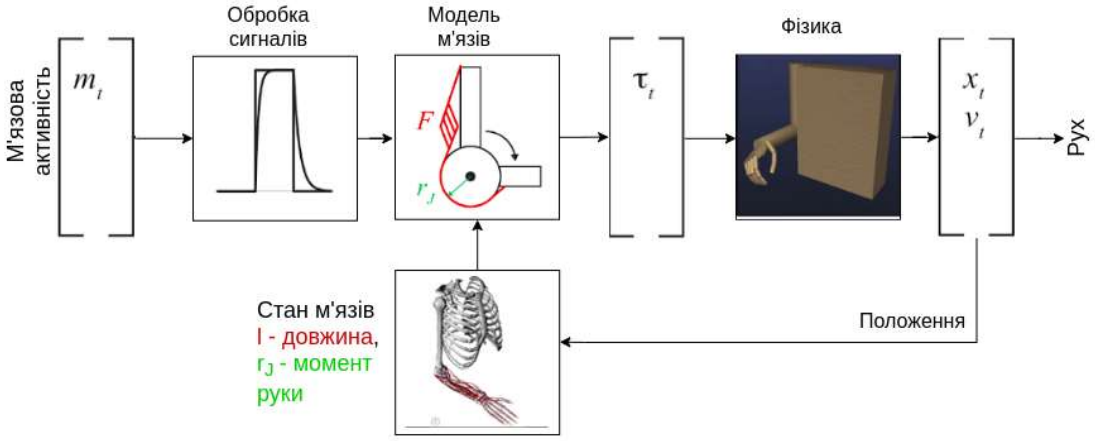


Рисунок 1.1 – Загальна концепція перетворення сигналів електроміографії в рух

На рисунку 1.1 схематично зображено один із варіантів перетворення входів електроміограми (ЕМГ) в рух кінцівки. Сигнали ЕМГ за допомогою блоку обробки сигналів перетворюються в сигнали рекрутингу м’язів (англ. the recruitment of muscle) і йдуть як вхід до 3-го блоку “м’язова модель”. Інший вхід до цього блоку є набір м’язових характеристик: довжина, швидкість та моменти плечей на кожну ступінь свободи. Модель м’язів аналізує їх та дає на вхід блоку “фізики” крутні моменти на ступені свободи. Останній блок, вирішуючи рівняння руху, оновлює стан кінцівки (позицію та швидкість) та призводить її у рух.

В дипломному проєкті охоплювалася тільки частина м’язової моделі, яка з постави верхної кінцівки аналізує довжини її м’язів та моменти плечей. Кожен з можливих рухів в системі регулюється за допомогою 33-х м’язів, кожен з яких, в свою чергу, охоплює від 3 до 6 ступеней свободи.

1.2 Змістовний опис і аналіз предметної області

1.2.1 Визначення вхідних та вихідних параметрів м’язової моделі

Загалом модель людської кінцівки являє собою модель з 18-а ступенями свободи, яка приводиться в дію за допомогою 33 м’язів. М’язи характеризуються своєю довжиною та моментами плечей на ступені свободи, що покривають. Кожен м’яз охоплює в середньому 3 (від 1 до 6) ступені свободи. Отже, щоб проаналізувати будь-яку поставу кінцівки, яка описується 18 ступенями свободи, потрібно визначити довжину 33-х м’язів та моменти плечей цих м’язів на кожну ступінь (тобто 33x18 значень), які відповідають за дане конкретне положення опорно-рухового апарату.

1.2.2 Методи машинного навчання

1.2.2.1 Загальне визначення

На дуже високому рівні машинне навчання – це процес навчання комп'ютерної системи робити точні прогнози або рішення при подачі набору даних. Машинне навчання здатне вирішувати задачі класифікації, розпізнавання об’єктів, генерації тексту та зображень. Ключова відмінність від традиційного алгоритмічного програмування полягає в тому, що розробник не пише код та логіку для обробки даних системою. Натомість модель машинного навчання програмують самостійно вивчати корисні шаблони в даних.

Модель, отримана в результаті машинного навчання – це алгоритм, який використовує історичні, іноді розмічені дані для вироблення висновків щодо нових даних, які не були доступні під час навчання.

Глибоке навчання – галузь машинного навчання, яка імітує роботу мозку людини при обробці даних та пошуку шаблонів для прийняття рішень. Штучні нейронні мережі побудовані як людський мозок, нейронні вузли з'єднані між собою, як павутина.

Існують велика кількість методів машинного навчання, які так чи інакше, мають властивість вирішувати тільки окремі види задач. Так, наприклад, згорткові нейроні мережі застосовують здебільшого в задачах комп'ютерного зору, де йде аналіз зображень, які в свою чергу являють собою тривимірні матриці. Рекурентні нейроні мережі використовуються в сфері обробки природньої (людської) мови, де на вхід моделі машинного навчання приходить текст у вигляді послідовності векторів, що характеризують слова. Лінійні регресії та дерева рішень чудово підходять для вирішення багатьох задач в сфері фінансів, в задачах розпізнання аномальної поведінки інтернет користувачів, тощо.

В цій роботі була поставлена задача регресії, тобто перетворення одних числових даних в інші. В контексті м'язової моделі відбувається навчання перетворення вектору з 18 значень ступенів свободи суглобів у значення вектору довжин 33-х м'язів та матриці 33x18 моментів плечей цих м'язів. Для задачі регресії можна використовувати поліноміальні регресії, різновиди випадкових лісів та методів градієнтного підсилювання (англ. gradient boosting), повнозв'язні штучні нейроні мережі (англ. fully-connected feed-forward neural network).

1.2.2.2 Легкий метод градієнтного підсилювання (LGB)

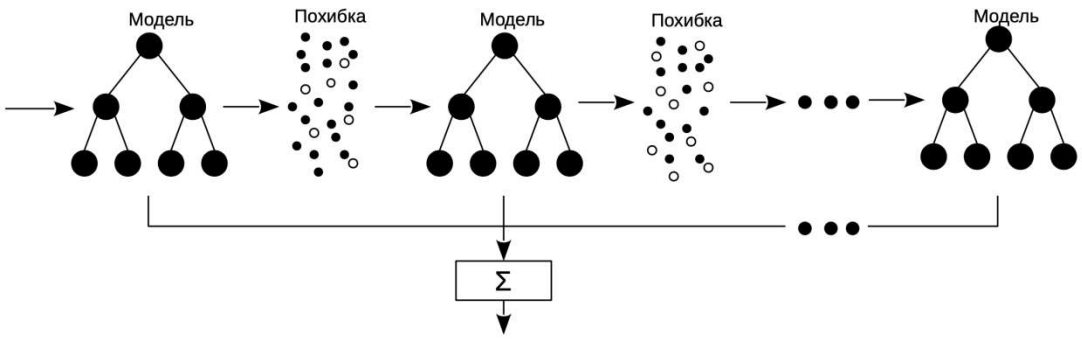


Рисунок 1.2 – Загальна структура методу градієнтного підсилювання на деревах рішень

Легкий метод градієнтного підсилювання (англ. Light Gradient Boosting Machine) належить до групи методів градієнтного підсилювання, заснованих на ітераційній побудові простих функцій-учнів, які намагаються знайти глобальний мінімум у функції витрат. Градієнтне підсилювання – це метод ансамблювання (англ. ensemble method) слабких моделей прогнозування, наприклад дерев рішень, в результаті якого створюється єдина більш потужна модель.

Дерево регресії – це модель машинного навчання, яка використовує бінарні рекурсивні рішення для побудови графу. Алгоритм створює таку систему розгалужень, щоб подібні дані в кінцевих гілках (так званих листях) були згруповані разом. Таким чином, тренувальний процес для такого дерева - це пошук оптимального маршруту входів, а процес умовиводу (англ. inference, evaluation) – пошук висновку подібних, вже вивчених даних. Лекий метод градієнтного підсилення використовує градієнтний метод, який будує послідовності декількох дерев регресії, які обробляють помилки поетапно для підвищення точності виведення. Метод створює ланцюг регресійний дерев, кожен з яких навчається на помилках попереднього. На перше дерево подаються оригінальні дані, а кожне наступне намагається виправити вже існуюче прогнозування (Рисунок 1.2). Під час тестування такої моделі за результат береться сума усіх прогнозів кожного регресійного дерева.

Лекий метод градієнтного підсилювання – це метод градієнтного підсилювання, який впроваджує додаткові покращення продуктивності [6]. Модель використовує так звану однобічну вибірку на основі градієнта – це процес, який вибирає набір входів для кожного (не першого) регресійного дерева з таким розподілом: вибірка з k елементів, на якому попередній учень має найбільшу похибку в функції наближення, і k інших випадково вибраних помилок. Кількість елементів k залежить від сумарної кількості тренувальних даних, але

$$2 * k < n \quad (1.1),$$

де n - кількість елементів тренувального набору даних.

Крім того, в такому методі підсилювання структура регресійних дерев будується не по рівням (або в ширину), як це зазвичай робиться, а у глибину, тобто нарощуючи спочатку листя з максимальною втратою дельти росту (Рис. 1.3).

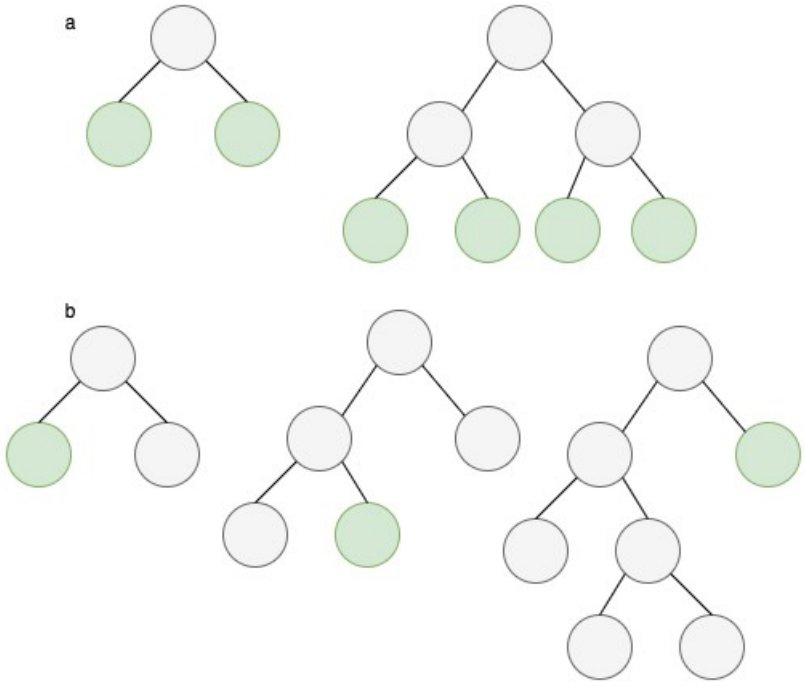


Рисунок 1.3 – Побудова дерев: a – горизонтально (по рівнях), b – вертикально (у глибину)

1.2.2.3 Повнозв’язні нейроні мережі

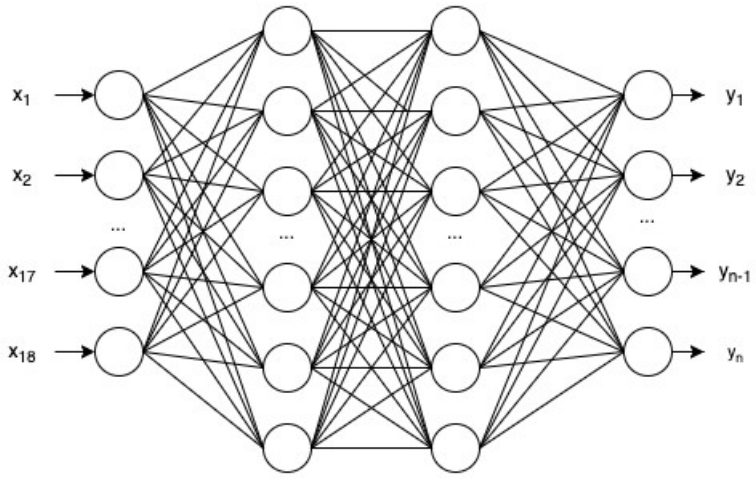


Рисунок 1.4 – Структура повнозв’язної нейроної мережі

Штучна нейронна мережа (ANN) – це послідовність нейронів, з'єднаних між собою синапсами. Структура нейронної мережі являє собою аналог роботу людського мозку. Завдяки такій побудові модель отримує можливість аналізувати та запам'ятовувати різну інформацію. Нейрон – це обчислювальна одиниця, яка отримує інформацію, виробляє над нею прості обчислення і передає далі. Велика кількість нейронів створює шар. Вони діляться на три основних типи: вхідний, прихований та вихідний. Вхідний шар отримує інформацію, приховані шари її обробляють, а вихідний шар виводить результат. Синапс – це зв'язок між двома нейронами. У синапсів є один параметр – вага. Завдяки йому вхідна інформація змінюється, коли передається від одного нейрона до іншого.

Для побудови нелійнійності нейронної мережі, кожен нейрон має в собі функцію активації. Найпоширеніші функції активації: сігмоїдна (логічна) (1.2), виправлена лінійна (1.3) та гіперболічний тангенс (1.3).

$$\sigma(x) = \frac{1}{1+e^x} \text{ (1.2)}$$

$$f(x) = \max(0, x) \text{ (1.3)}$$

$$f(x) = \tanh(x) \text{ (1.4)}$$

Повнозв'зна штучна нейрона мережа – це нейрона мережа, яка складається з послідовності шарів, кожен нейрон в якому зв'язаний з кожним нейроном наступного шару (Рис. 1.4). Під час тренування таких моделей вводяться додатково також термін епоха. Під епохою розуміється один прогін тренувальних даних нейронною мережею. Цей параметр зазвичай встановлюють вручну, чим він більший – тим краще буде точність вихідної моделі після етапу тренування. З іншої сторони тим більше часу займає саме тренування.

1.3 Аналіз успішних ІТ-проєктів

За останні 5 років з'явилося багато компаній, що займаються розробкою різноманітних протезів та ортезів. В основному фокус робиться на дешеві

біонічні пристрої з пластикових деталей, виконаних в тому числі за допомогою технологій 3D друку. Є вже готові продукти, наприклад, від OpenBionics, OttoBock, iLimb, тощо. Але їх розвиток спрямований не на здешевлення протезів, а скоріше на механіку рухів (плавність, природність, точність). При подібному підході функціональна частина протеза розвивається, але керованість залишається колишньою.

Існує відносно невелика кількість рішень для заміни пошкодженої частини кінцівки людини. Коли мова йде про реабілітацію після ампутації, найбільш простим рішенням є косметичний протез. Крім естетичного призначення, такі протези не виконують ніяких функцій і не мають переваг в порівнянні із середньовічними протезами-крюками.

Інше рішення – це тягові протези. Їх кисті вже можуть стискатися і розтискати за рахунок, наприклад, рухів променевого або ліктьового суглоба решти руки. Ці рухи приводяться за допомогою механічних натягів ниток. Така кисть «вміє» тільки стискати кулак і розтискати його. Вона відрізняється швидкодією і непоганою надійністю.

Третій клас – механічні протези, керовані м'язової активністю. Такі пристрої, як правило, виконані з металу, мають велику міцність, але володіють тільки двома ступенями свободи – стиснення і розтискання. Керувати механічним протезом не дуже зручно: для того, щоб розтиснути кулак, потрібно напружити зовнішню сторону передпліччя, а для того, щоб стиснути - навпаки, напружити внутрішню сторону передпліччя. Це так званий критичний спосіб управління: або м'язова активність є – тоді рух активується, або м'язової активності немає. На жаль, така система управління може призводити до помилкових спрацьовувань. Механічні протези мають «зовнішність» косметичних і функціональність тягових, живлячись від акумулятора, який розміщується на протезі. Металевий каркас і мотор, що приводить в рух кінцівку, дозволяють називати конструкцію надійної: наприклад, якщо потрібно тримати якийсь предмет, механічна рука зможе стиснути його сильно

і надовго, і це практично не потребують зусиль з боку людини. Незручне управління і обмежена функціональність – основні недоліки механічних протезів.

Останній, четвертий клас – біонічні протези, в яких кожен палець керується окремим мотором. Це дає більшу перевагу в плані маніпуляцій з предметами. Система управління біонічної кистю руки така ж, як і у механічної, на основі стиснення і розтискання - тому цими протезами складно користуватися. Для полегшення використання додають будь-які зовнішні перемикачі - важелі на протезі або додатки на смартфоні.

«Біонічність» має на увазі крім поповнення механічних функцій втраченої руки, природність її використання. Розробники сфокусовані на оптимізації побудови протезів - потрібні максимально міцні, ергономічні, функціональні з точки зору механіки рішення. Проте, завдання забезпечення максимальною функціональністю управління, не має готового рішення на ринку. А незручні і обмежено функціональні протези коштують від $ 30 000 до $ 70. 000.

1.3.1 Аналіз відомих технічних рішень

Всі сьогоднішні R&D проєкти в основному сфокусовані в напряму поліпшення системи управління. В ідеалі людина, що користується протезом, не повина помічати системи управління. Тобто інтерфейс між людиною і протезом повинен використовувати природні механізми управління, яким людина навчилася ще в дитинстві. Один з найперспективніших типів такого інтерфейсу може бути електроміографія. Використання системи управління, індивідуально налаштованої на шаблони рухів кисті конкретної людини, наближає нас до створення природного інтерфейсу між людиною і протезом. З одного боку, він не інвазивний і має велику функціональність, з іншого - швидко налаштовується і стійкий до зовнішнього впливу. Такою системою управління може бути деяка система машинного навчання, яка змогла б перетворювати карту сигналів ЕМГ у визначення руху опорно-рухової системи.

Протягом останніх десятиліть порівняно невелика кількість досліджень була присвячена застосуванню методів машинного навчання для опорно-рухової динаміки. Francisco Sepulveda у своїй роботі [7], використовуючи прості штучні нейроні мережі, склав карту єдиної картини ЕМГ для кутів суглобів та моментів суглобів для стегна, коліна та голеностопа. Незважаючи на те, що в цьому дослідженні не були змодельовані м’язові шляхи та їх динаміка генерування сил, декодування рухового наміру була продемонстрована з низькими помилками. Однак, те саме відображення, правда, може бути виражене стандартним аналізом основних компонентів з високою точністю та низькою обчислювальною вартістю [8]. Незважаючи на те, що аналіз основних компонентів та інші альтернативи статистичної класифікації є надійними, ці методи чутливі до часової динаміки, наприклад, зміни структури внаслідок зміни швидкості руху зв'язків.

Було також досліджено, що поєднання згорткових та рекурентних штучних нейронних мереж точно і надійно відображає перетворення від частотно-часових кадрів багатоканального ЕМГ до руху кінцівок [9].

Був також представлений комбінований підхід до навчання в режимі реального часу, щоб керувати роботизованими верхніми кінцівками, спираючись на положення кінцевого ефектора [10]. Модель використовує метод goal babbling, щоб зменшити простір пошуку, нейронну мережу для представлення стану роботизованої руки та онлайн процеси Гаусса для регресії. Rane у своїй роботі [11] порівнює три типи штучних нейроних мереж (повнозв’язні, згорткові та рекурентні) при прогнозуванні сил, демонструючи переваги від використання контрольованих методів навчання в наближенні відображення від кінематичного простору до м'язового простору.

Останнім часом також навчання з підкріпленням привернуло багато уваги з боку спільноти, особливо працюючи над завданнями відображення в режимі реального часу. Навчання з підкріпленням забезпечує структуру, яка дозволяє моделі отримувати дані з навколишнього середовища в режимі реального часу,

виконувати необхідні дії, отримувати зворотній зв'язок та коригувати свою поведінку. У своїй роботі Zhou [12] вирішував проблему намірів опорно-рухового апарату вдосконаленою оптимізацією за допомогою навчання з підкріпленням, а також запропоновав навчальне середовище для опорно-рухового апарату.

1.3.2 Аналіз відомих програмних продуктів

Готових протезів на основі машинного навчання на ринку на сьогоднінній день немає, всі вони знаходяться або на етапі розробки, або на етапі експерименту.

Інженери з університету Ньюкасла у 2018 році розробили «руку, що бачить», що являє собою біонічну кінцівку з камерою [13]. Камера миттєво фотографує об’єкт перед собою, оцінює його форму та розміри, та запускає послідовність рухів в протезі. Використовуючи згорточні нейроні мережі, розробники навчили модель розпізнавати «хватку», необхідну для різних об’єктів. Проєкт виграв премію Netexplo UNESCO Award у 2018 році та IET William James Award. Такий метод на основі машинного навчання є перспективним, але все ще експерементальним. Програмне забезпечення таких протезів не є оптимізованим.

1.4 Аналіз вимог до програмного забезпечення

Незважаючи на зростаючу кількість досліджень опорно-рухової динаміки, історична увага спільноти машинного навчання була схильна до вирішення задачи опорно-рухового апарату єдиною моделлю, використовуючи обчислювально важкі методи глибокого навчання та навчання з підкріпленням, залишаючи окремі більш прості. Це може бути пояснено тим, що біологічна система демонструє складну поведінку в просторі часу, проблема має багатовимірний простір рішень, навколишнє середовище є невидимим і впливає на систему нелінійно. У поточному проєкті метою булообрано вивчення

динаміки опорно-рухового апарату за допомогою декількох простіших технологій машинного навчання, шукаючи більш обчислювально ефективне рішення. Для цього була поставлена ціль розробити програмний комплекс для тренування та аналізу моделей машинного навчання для розв`язування біомеханіки опорно-рухового апарату та зручний веб-додаток для взаємодії та аналізу цих моделей.

1.4.1 Розроблення функціональних вимог

В системі передбачено наступні функціональні варіанти використання:

Таблиця 1.1 – Варіант використання UC001

Назва	Перетворення постави опорно-рухової системи в м'язові характеристики.
Опис	Обробка вектору із 18 ступенів свободи опорно-рухової системи в значення довжин 33-х м'язів та 33x18 моментів плечейцих м’язів.
Учасники	Користувач системи
Передумови	
Післяумови	Користувач отримав числові характеристики точності вибраної їм моделі
Основний сценарій	1) Користувач відкриває веб-застосунок; 2) Користувач натискає кнопку «Interact» та переходить у вікно роботи з моделями; 3) Користувач задає значення для кожного степеню свободи в заданих проміжках, або вибирає вже готові значення з файлів; 4) Натискає кнопку «predict»;

Продовження таблиці 1.1

	5) Система виконує обробку зазначених значень, рахує помилки моделі та відображає результати.

Виходячи з вище описаних вимог та аналізу аналогів можна сформувати наступні функціональні вимоги:

Таблиця 1.2 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Точність моделей
Опис	Моделі машинного навчання в даній роботі повинні мати похибку нормалізованої середньо-квадратичної похибки нижче за 2%.

Таблиця 1.3 – Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Швидкість моделей
Опис	Моделі машинного навчання в даній роботі мають виконувати внутрішні обчислення менш ніж за 1 мс.

Таблиця 1.4 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Обробка вхідних значень
Опис	Веб-застосунок має виконувати обробку заданих вхідних значень

1.4.2 Розроблення нефункціональних вимог

Програмне забезпечення повинне відповідати наступним нефункціональним вимогам:

– локалізація інтерфейсу – англійська;

– підтримувана версія браузера: Google Chrome 49 або вище.

1.4.3 Постановка комплексу завдань модулю

Розроблюване програмне забезпечення призначене для вирішення та аналізу задачі перетворення 18-и ступеней свободи опорно-рухового апарату кінцівки у м’язові характеристики.

Мета роботи – створення програмного комплексу для тренування та аналізу моделей машинного навчання для розв’язуванняпроблеми опорно-рухової системи, також створення зручного веб-додатку для демонстрування та аналізу цих моделей.

Для досягнення даної мети необхідно вивчити динаміку опорно-рухового апарату за допомогою декількох простих технологій машинного навчання, проаналізувати їх сильні та слабкі сторони.

Розроблений веб-застосунок повинен працювати на пристроях зі встановленим браузером Chrome.

1.5 Висновки по розділу

В першому розділі введені основні поняття для визначення опорно-рухової системи моделі руки та розглянуті кілька регресійних моделей машинного навчання, які можуть бути використанні для вирішення проблеми аналізу динаміки руху людини використовуючи сигнали електроміографії.

Було також проаналізовано основні типи протезів та ортезів, які на разі розробляються та використовуються у світі, розглянуто ранні роботи вчених на тему використання машинного навчання для вивчення динаміки опорно-рухового апарату людини. На їх основі були розроблені основні функціональні

та нефункціональні вимоги до розроблювального програмного забезпечення. Базуючись на цих вимогах було виконано постановку комплексного завдання.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

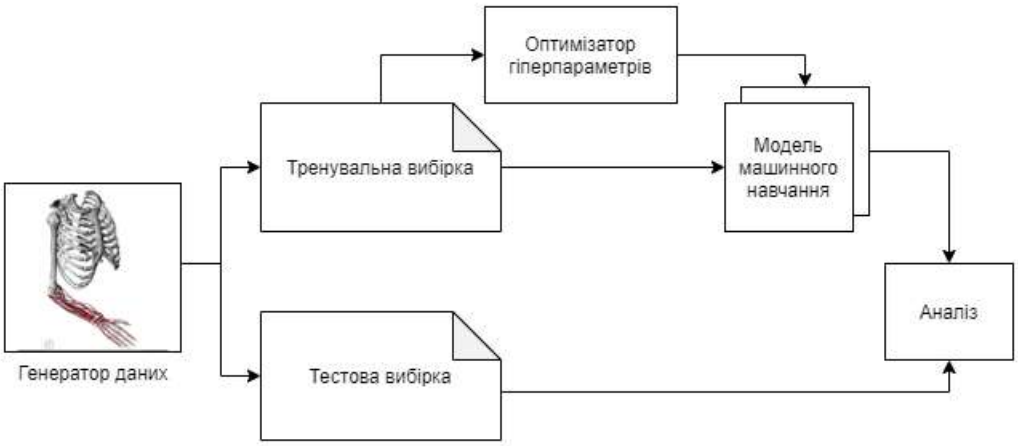


Рисунок 2.1 – Загальна схема програмного забезпечення для тренування та аналізу моделей машинного навчання

На Рис. 2.1 зображена загальна схема програмного забезпечення для тренування та аналізу моделей машинного навчання. Вона складається з 4-х основних блоків. Генератор даних відповідальний за генерацію тренувальних та тестових даних. Блок оптимізатору гіперпараметрів приймає на вхід модель машинного навчання та тренувальні дані, і за допомогою баєсовської оптимізації підбирає ті гіперпараметри для кожної із моделей машинного навчання, що дозволяють якнайкраще розв’язувати конкретну задачу. Гіперпараметри – це ті параметри моделі, які задаються вручну перед тренуванням (отже, під час самого єтапу тренування вони не оптимізуються) задля покращення точності, швидкості моделі, або інших характеристик. В блоці “Моделі машинного навчання” прописується архітектура методів машинного навчання, функції для їх тренування та створення висновків, а також збереження та завантаження навчених моделей. В останьому блоці відбувається аналіз моделей: підрахунок точності моделей, їх швидкість, створення графіків, тощо.

2.1.1 Генератор даних

Раніше був розроблений метод автогенерованих поліномільних моделей [14]. У цій роботі, довжина м`язів опорно-рухового апарату і моменти плечей суглобів для кожного м’яза у моделі верхньої кінцівки [15] були точно апробовані за допомогою підбору поліному до 5-го терміна, де довжина та моменти плечей руки були з'єднані через часткову похідну довжини м'яза в локальних координатах, що відповідають поставі кінцівки. Метою цієї розробки було обійти дорогі обчислення перетворень з якісними наближеннями. Модель демонструвала високу точність із кінематичними помилками нижче 1%. Ці поліноміальні моделі стану м'язової залежності від постави використовувались в основі генератора даних.

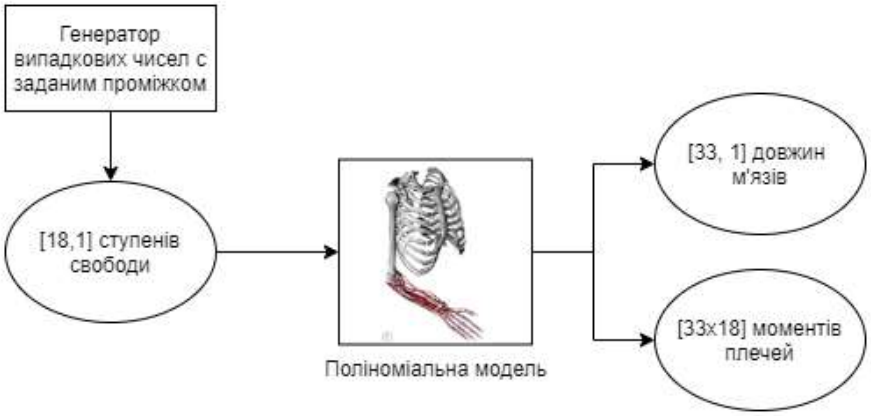


Рисунок 2.2 – Загальна схема блоку генерації даних

Набори даних для тренування, валідації та тестування моделей були сформовані за допомогою двох систем поліноміальних моделей. Перша вирішує задачу перетворення постави опорно-рухового апарату в довжини м’язів, друга – перетворення постави в моменти плечей м’язів в кожну ступінь свободи. На вхід цим моделям подається вектор зі значеннями 18 ступенів свободи, які формувалися випадковим чином за рівномірним розподілом. На виході з цієї системи виходять дві матриці – матриця довжин м’язів із розмірністю [33, 1] та матриця моментів плечей кожного м’яза на 18 ступенів свободи з розмірністю [33x18] (Рис 2.2). Таким чином поліноміальна модель створює тренувальний, валідаційний та тестувальний набори даних.

Навчальний набір даних використовувався для двох завдань, налаштування гіперпараметрів моделей та навчання моделі, щоб максимізувати точність моделі при тестуванні потрібних результатів із заданими входами. Для цього набору генерувався один мільйон зразків, що були розподілені на тренувальний набір даних (80%, 800 000 зразків) та набір даних для валідації (20%, 200 000 зразків). Набір даних для валідації використовувався для запобігання перенавчання, тобто більш високої точності моделей на даних тренування порівняно з результатами даних тестування. Для тестування був сформований набір даних, який містив 50 000 зразків.

2.1.2 Методи машинного навчання

В цьому дослідженні були навчені та проаналізовані декілька моделей машинного навчання, але успішними були лише дві: легкий метод градієнтного підсилювання та повнозв’язна штучна нейрона мережа.

2.1.2.1 Легкий метод градієнтного підсилювання

Однією з моделей машинного навчання, яка використовувалась в даному дослідженні був легкий метод градієнтного підсилювання (LGB), алгоритм якого був опублікований вченими компанії Microsoft у 2017 році.

Кожен зв’язок довжини і моменту плечей верхньої кінцівки з поставою був оснащений однією моделлю LGB. Модель повної руки була змодельована 132-а моделями бустингу: 33 моделі для перетворення довжин та 99 – для моментів сил.

Оскільки кожен м’яз перетинає в середньому 3 (від 1 до 6) ступенів свободи, то кількість моментів плечей була оптимізована і зменшена від 33x18 до 99. Моменти плечей кожного м’яза на ті ступені свободи, яка вона не перетинає, поверталися з нульовими значеннями.

Навчені моделі зберігалися у пам'яті за допомогою алгоритму серіалізації - процес перетворення об’єкта в потік байтів, а завантажувалися за допомогою зворотньої операції десеріалізації.

2.1.2.2 Повнозв'язна штучна нейрона мережа

Було розроблено дві моделі повнозв’язних штучних нейроних мереж для оцінки довжини м’язів та моментів руки від заданої постави. В цьому проєкті використовувалися повнозв’язні нейроні мережі з одним вхідним, одним вихідним і двома прихованими шарами (Рис. 1) зі стандартним перетвореннями.

Було створено мережі, що складаються з наступної кількості вузлів вхідного, двох прихованих та вихідних шарів: [18, 1024, 512, 33] для наближення 33 довжин м’язів та [18, 2048, 1024, 99] для наближення 99 моментів сил. Для цієї моделі кількість моментів плечей була також оптимізована таким же чином, як і в методі градієнтного підсилення.

Початкові ваги для кожного шару були вибрані з нормального розподілу з нульовим середнім значенням, а його відхилення обчислювались із кількості входів і виходів цього шару, що називається методом ініціалізації Xavier:

$$std = \sqrt{\frac{2}{fan_{in} + fan_{out}}} \quad (2.1)$$

де fan_{in} – кількість вхідних параметрів до шару, а fan_{out} – кількість вихідних.

Мережі навчалися з партіями вибіркових даних по 256 зразків, використовуючи метод стохастичної оптимізації на основі градієнта, мінімізуючи власну функцію витрат. В дипломному проєкті бала розроблена спеціальна функція витрат, яка намагалася зосередитися на оптимізації точності, використовуючи найгірші (аномальні) показники ефективності. Функція вартості оцінювалася за допомогою середньо-квадратичної похибки з найгірших 5% пар входу-виходу з кожного м'яза. Скалярна вартість оцінювалася як середнє значення всіх помилок у верхньому діапазоні 30%.

Динаміка навчання контролювалася за допомогою коефіцієнту швидкості навчання. Початковий коефіцієнт швидкості був встановлений як 0,001, і зменшувався на 20% в тому випадку, коли вимірюваний показник точності

припиняв покращуватися після 2 повних епох, або 2 повних проходів через весь набір даних. Моніторинг перенавчання здійснювався за допомогою даних валідації під час тренування.

Збереження моделі являє собой збереження ваг синапсів нейроної мережі у файлову систему. Під час завантаження моделі спочатку ініціалізується архітектура штучної нейроної мережі, а потім випадково ініціалізовані ваги замінюються на ваги, збережені у файлі.

2.1.3 Оптимізатор гіперпараметрів

Перед навчанням моделей машинного навчання в блоці оптимізатору за допомогою баєсовської оптимізації [16] підбиралися гіперпараметри для кожного алгоритму. На вхід цьому блоку подаються 800 тисяч зразків тренувальної вибірки, яка ділиться на дві частини з розподілом 70% на 30%. Перша частина використовується для тренування моделей, друга - для їх валідації та вибору оптимальних значень гіперпараметрів. Баєсовський метод оптимізації ітеративно проводить задану кількість тренувань, під час якої використовує інформацію з попередніх випробувань для створення набору гіперпараметрів для наступного випробування, таким чином будуючу баєсовську модель значень гіперпараметрів для оптимізації функції наближення.

2.1.3.1 Оптимізація LGB

Для тренування моделей легкого методу градієнтного підсилення були оптимізовані три основні типи гіперпараметрів:

– кількість листків у одному дереві рішень в дапазоні від 20 до 100;

– мінімальна кількість елементів в одному листку в діапазоні між 10 та 100;

– максимальна глибина дерева або максимальна кількість рівнів в діпазоні від 1 до 100.

Інші гіперпараметри в моделях LGB, наприклад кількість простих моделей, було обрано за замовчуванням згідно офійціної реалізації цього алгоритму.

2.1.3.2 Оптимізація архітектури нейроної мережі

Для штучної нейроної мережі гіперпараметрами були:

– різні функції активації для кожного шару: випрямлена лінійна (англ. rectified linear unit), нещільна випрямлена лінійна (англ. Leaky rectified linear unit);

– додаткові шари виключення (англ. dropout layer) на виході прихованих шарів з вірогідністю деактивації нейронів в діапазоні від 0.2 до 0.6;

– додатковий шар нормалізації (англ. batch normalization layer) на виході прихованих шарів.

Шар виключення потрібен для того, щоб запобігти перенавчання та забезпечити узагальнене рішення, розподілене по декількох вузлах в штучної нейроної мережі. При кожній ітерації навчання модель будується тільки з частини своїх первинних вузлів. За це відповідає коефіцієнт деактивації нейронів, який набуває значення від 0 до 1. У випадку 0.5, шар в моделі будується лише з половини зініціалізованих нейронів. Шар нормалізації використовується для нормалізації значень на виході шару.

2.1.4 Блок аналізу моделей

В цьому блоці рахується точність навчених моделей машинного навчання, вимірюється їх швидкість та будуються графіки для їх порівняння. Точність натренованих моделей оцінювалася на тестувальному наборі, який не використовувся під час етапу тренування.

Для оцінки точності прогнозування моделі машинного навчання в контексті м’язової моделі опорно-рухового апарату використовувалася нормалізована середньо-квадратична похибка (RMSE). Значення RMSE для довжин м’язів розраховувалася як абсолютна різниця між очікуваними та

прогнозованими значеннями. Для нормалізації результатів кожна очікувана та прогнозована довжина м'яза нормалізувалася відповідно до діапазону значень довжини конкретного м'яза:

$$RMSE_L = \frac{1}{m}\sum_{l=1}^{m}\frac{1}{n}\sum_{i=1}^{n}\sqrt{\left(\frac{x_{r,l,i}-x_{p,l,i}}{L_{l,max}-L_{l,min}}\right)^2} \quad (2.2),$$

де m – кількість довжин (m=33); n – кількість тестових зразків, $x_{r,l,i}$ та $x_{p,l,i}$ – вибірки очікуваних та прогнозованих довжин відповідно; $L_{l,\ max}$ та $L_{l,\ min}$ – максимальні та мінімальні можливі значення для l-ої довжини м'яза.

Аналогічно, RMSE для моментів плеча обчислювалася як абсолютна різниця між очікуваними та прогнозованими величинами. Для нормалізації результатів кожне значення моменту ділилося на максимальне значення моменту для кожного м'яза відповідно:

$$RMSE_{MA} = \frac{1}{m}\sum_{l=1}^{m}\frac{1}{n}\sum_{i=1}^{n}\sqrt{\left(\frac{x_{r,l,i}-x_{p,l,i}}{M_{l,max}}\right)^2} \quad (2.3),$$

де m – кількість моментів значень; n - кількість тестових зразків, $x_{r,l,i}$ та $x_{p,l,i}$ – вибірки очікуваних та прогнозованих довжин відповідно; $M_{l,\ max}$ – максимальне можливе значення для l-го м'язового моменту плеча.

2.1.5 Експериментальні результати

2.1.5.1 Точності моделей відносно величини тренувального набору даних

На першому етапі аналізу моделей було досліджено, скільки тренувальних даних достатньо для тренування алгоритмів машинного навчання задля отримання точних алгоритмів. Для оцінки точності використовувалося значення нормалізованої середньо-квадратичної похибки (RMSE).

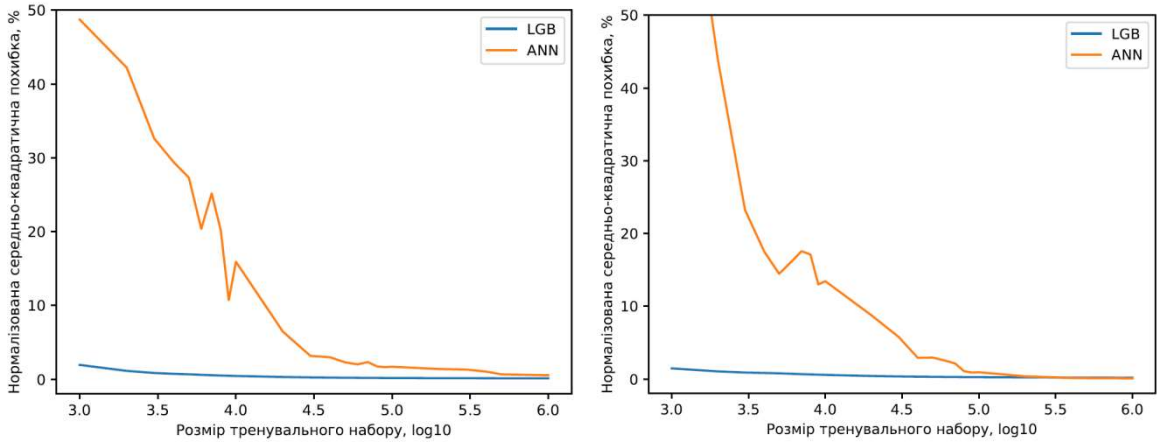


Рисунок 2.3 – Середня похибка для алгоритмів як функцій розміру набору даних. Ліворуч – для моментів плеча, праворуч – для довжин м'язів

На рисунку 2.3 представлені залежності нормалізованих середньо-квадратичних похибок довжин та моментів плечей від розміру набору даних N для всіх м'язів. Розміри наборів даних були розподілені між N=1000 та N=1000000.

Штучна нейрона мережа показує стабільне зменшення помилок з кожною новою частиною даних, що додаються до навчального набору. Починаючи з похибки 87% для довжини і 48% для моментів плечей з 1000 навчальних зразків, алгоритм досягає найкращого результату під час подачі 1000000 зразків: 0,08%±0.05% для довжин і 0,53%±0.29% моментів плеча.

Легкий метод градієнтного підсилювання показує похибки в 1,45% для довжин та 1,94% для моментів сил з найменшою кількістю даних (1000 зразків), але не в змозі досягти такого ж результату, як штучна нейрона мережа з 1 мільйоном зразків. Маючи максимальний набір даних для навчання, алгоритм досягає 0,18%±0.06% нормалізованої середньо-квадратичної похибки для довжини та 0,12%±0.07% для моментів плеча. Метод градієнтного підсилювання також вимагає набагато менших навчальних даних для отримання відносно невеликих помилок, оскільки після 100000 зразків показники перестали вдосконалюватися і не показували ніяких покращень аж до 1000000 зразків.

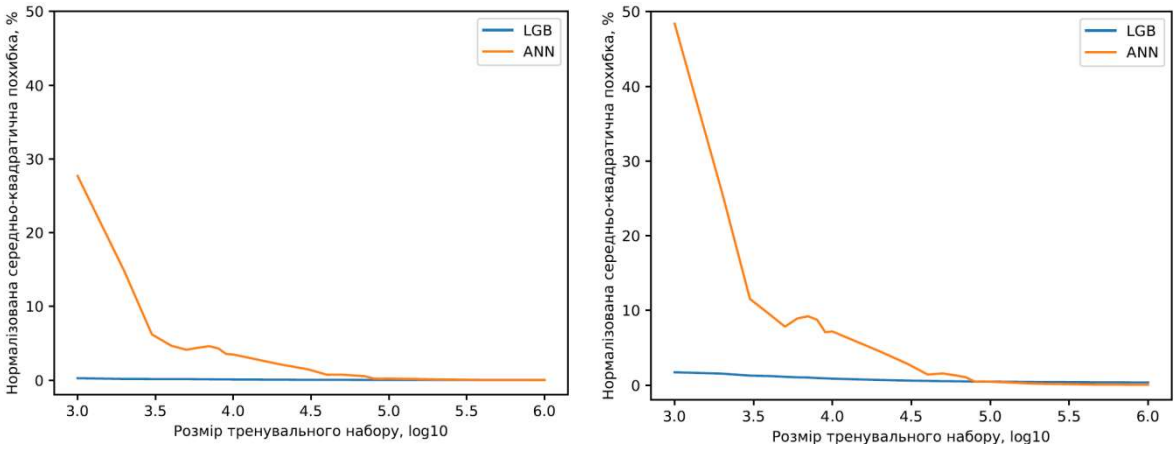


Рисунок 2.4 – Середня похибка для алгоритмів як функцій розміру набору даних для Biceps Brachii Long Head (зліва) та Extensor Pollicis Longus (справа)

Немає суттєвої різниці в поведінці залежності для м’язів різної складності. Для прикладу можна взяти Biceps Brachii Long Head та Extensor Pollicis Longus (Рис. 2.4), оскільки вони мають один із найпростіших та один із найскладніших математичних описів відповідно до поліноміальної моделі. Для опису довжини Biceps Brachii Long Head потрібне лише положення самої Biceps Brachii Long Head, тоді як для Pollicis Longus потрібна інформація про шість додаткових м'язів: Biceps Brachii Long Head, Biceps Brachii Short Head, Pronator Teres, Supinator, Pronator quadratus and Extensor Carpi Radialis Longus. Моделі обох цих м'язів демонструють однакову залежність від розміру набору тренувальних даних.

2.1.5.2 Оцінка точності моделей

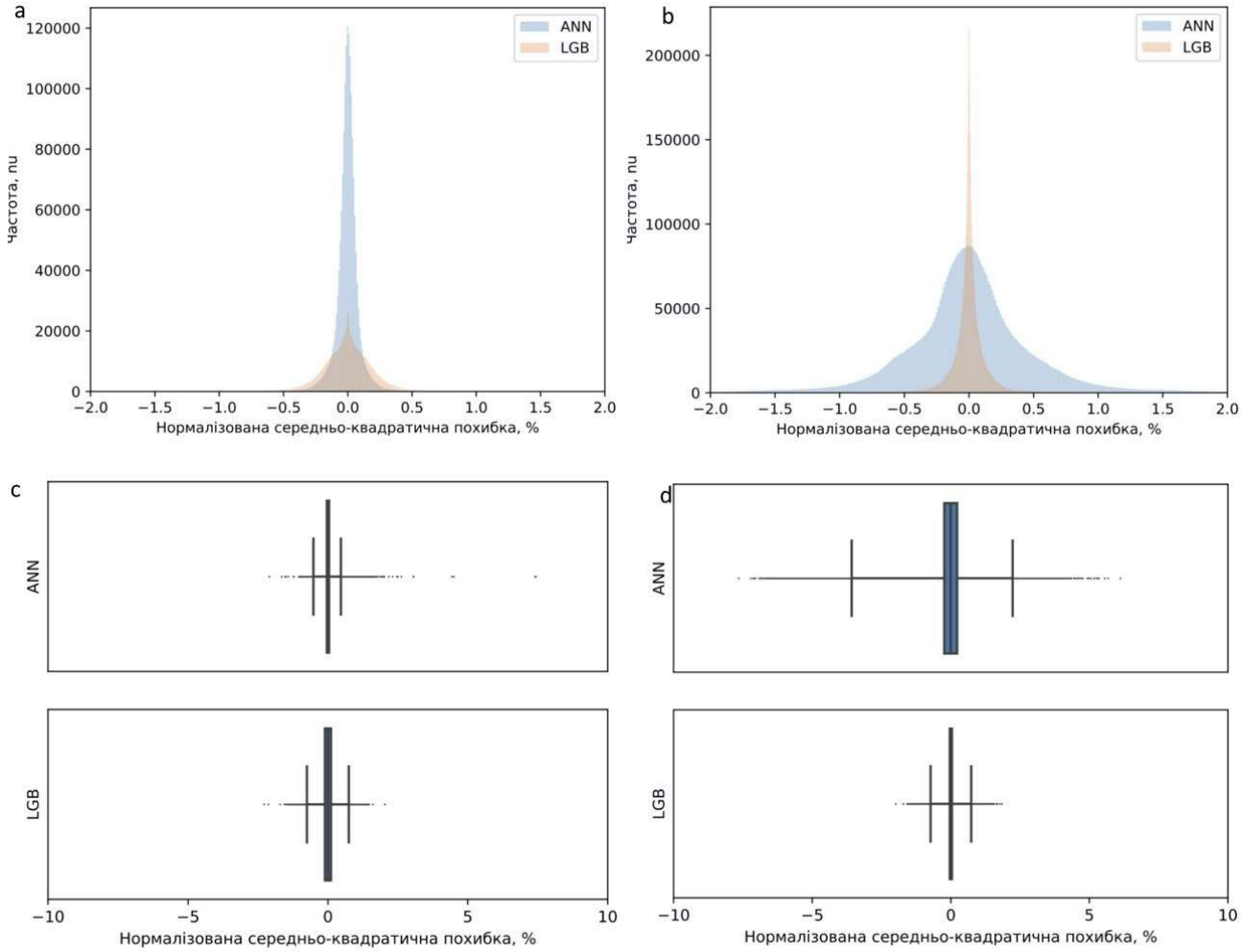


Рисунок 2.5 - Нормалізований розподіл похибок всіх м’язів для довжин (a, c) та моментів плечей (b, d)

На другому етапі порівнювалася точність моделей на одному тестовому наборі даних. На рисунках 2.5a та 2.5b представлені гістограми нормалізованих розподілів похибок для довжин та моментів плеча м’язів, що характеризують точність моделей штучної нейронної мережі та легкого методу градієнтного підсилювання, отримані з використанням тестової вибірки з 50000 елементів.

Точність прогнозування довжин штучної нейронної мережі лежить в діапазоні між -0,038% і +0,037%, а методу градієнтного підсилювання – між -

0.108% та +0.108% (Рис. 2.5c). Для вирішення задачі довжин м’язів штучна нейронна мережа має менше відхилення похибок приблизно в три рази.

Для моментів плеча розподіл помилок для двох моделей має іншу поведінку порівняно з довжинами. Нейронна мережа (рис. 2.5d) має розподіл похибок від -0.238% до +0.226%, тоді як помилки для моделі градієнтного підсилювання охоплюють діапазон [-0,039, +0,039%]. Варто відмітити також, що аномальні значення в штучній нейронній мережі досягають нижньої межі значень до -7.34%, а верхньої - до +6.12%.

Штучна нейрона мережа та метод градієнтного підсилювання демонструють порівняно схожі показники зі значеннями нормалізованих середньо-квадратичних похибок: 0,08% та 0,18% для довжини м’язів та 0,53% та 0,12% для м'язових моментів плеча відповідно.

2.1.5.3 Швидкість опрацьовування даних моделями

Обладнання, що використовувалося для розробки та тестування моделей, – процесор 1.4 GHz Quad-Core Intel Core i5 8-го покоління. Повна обробка одного набору даних в штучній нейронній мережі займає 1.1±0.6 мс, тоді як в методі градієнтного підсилювання - 43.1±8.3 мс. Нейронна мережа показує результат в 39 разів кращий, ніж легкий метод градієнтного підсилювання, що робить його потенційно кращим вибором для майбутньої розробки програмного забезпечення у режимі реального часу.

2.1.6 Веб-додаток

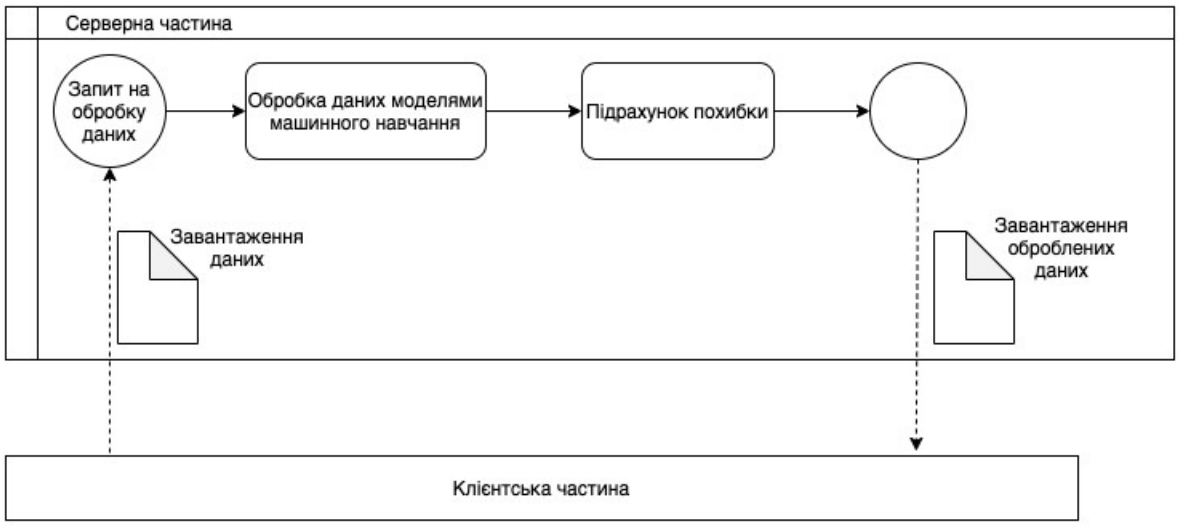


Рисунок 2.6 - Схема процесу обробки завантажених даних

Для демонстрації та взаємодії з навченими моделями машинного навчання був розроблений веб-додаток, який складається з серверної та клієнтської частин. Серверна частина відповідає за завантаження моделей та обробку вхідних даних цими моделями. На клієнтську частину вона повертає передбачені значення конкретної моделі для заданого набору даних і нормалізовану похибку для цих значень. Клієнтська частина відповідальна за обробку неправильно введених даних та відображення результатів обробки даних.

Наведена схема містить узагальнені процеси, які проходить користувач під час використання розробленого застосунку. Ці процеси включають в себе завантаження введених користувачем 18 значень ступенів свободи для верхньої кінцівки, процес обробки цих значень серверною частиною, процес відображення прогнозованих значень та їх нормалізованої середньо-квадратичної похибки (Рис. 2.6).

Нижче наведена діаграма діяльності для відображення взаємодії частин користувача та сервера.

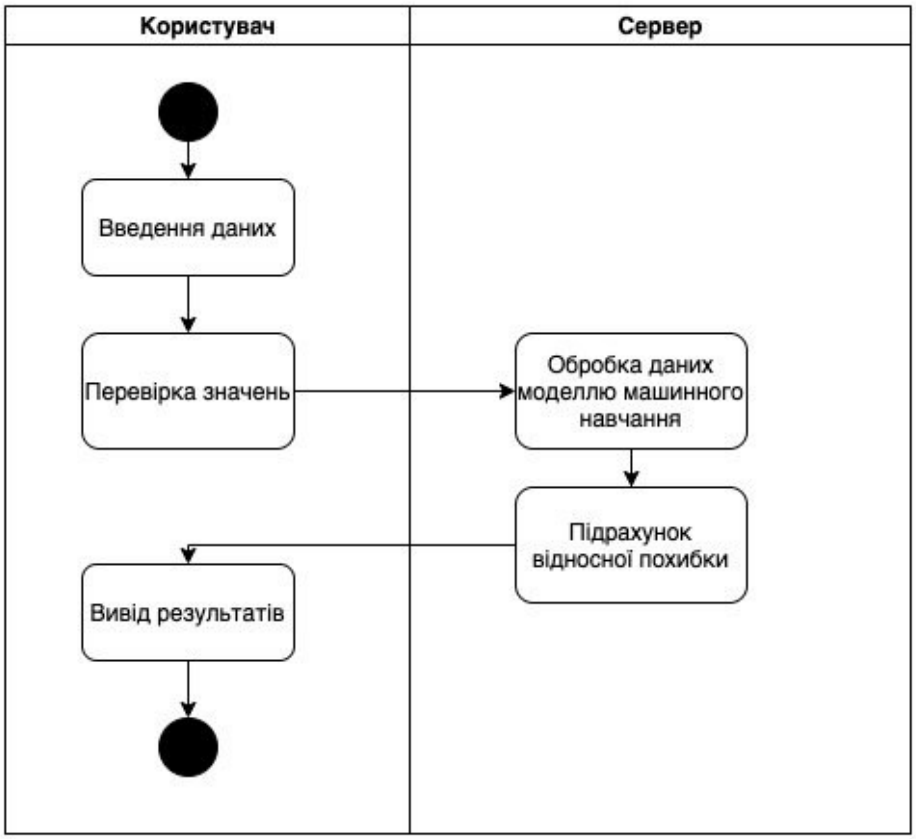


Рисунок 2.7 – Діаграма діяльності для відображення взаємодії частин користувача та сервера

2.2 Архітектура програмного забезпечення

Для вирішення поставленого завдання було обрано клієнт-серверну архітектуру системи. Діаграму розгортання системи наведено у документі: «Діаграма розгортання програмного забезпечення». Структурна схема класів програмного забезпечення наведена у докуменіх: «Структурна схема класів програмного забезпечення».

Для простої взаємодії з моделями машинного навчання був створений інтерфейс IModel, за допомгою якого тренування різних моделей машинного навчання узагальнюється, а у веб-застосунку – спрощується взаємодія серверної частини з цими моделями. Методи інтерфейсу описані в таблиці 2.1.

Таблиця 2.1 - Методи інтерфейсу IModel

Назва методу	Аргументи	Значення, що повертається	Коди помилок	Призначення мметоду
Fit	x_train, y_train_len, y_train_momarm	-	-	Тренування системи моделей
predict	x_test	np.ndarray	-	Обробка вхідних даних методами машинного навчання та повернення значень для довжин м’язів та їх моментів плеча
predict_len	x_test	np.ndarray	-	Обробка вхідних даних методами машинного навчання та повернення значень для довжин м’язів
predict_momarm	x_test	np.ndarray	-	Обробка вхідних даних методами машинного навчання та повернення значень для

Продовження таблиці 2.1

				моментів плеча м’язів
			-	повернення значень для моментів плеча м’язів
load	model_path	-	0x1, 0x2	Збереження моделі на диск
save	model_path	-	0x3, 0x4	Завантаження моделі з диску в пам’ять

Таблиця 2.2 - Помилки

Код помилки	Системна назва	Значення помилки
0x1	ERR_FILE_NOT_FOUND	Вказаний файл не існує в файловій системі
0x2	ERR_INVALID_FORMAT	Файл зі збережною моделю має невірний формат і не може бути завантажений
0x3	ERR_CAN_NPT_CREATE_FILE	Неможливо створити вихідний файл. Вказаний файл не існує в файловій системі або користувач не має прав на його створення
0x4	ERR_NOT_ENOUGH_SPACE	Неможливо створити файл зі збереженою моделю, оскільки в системі не достатньо місця для неї

2.3 Конструювання програмного забезпечення

Програмне забезпечення було розроблене з використанням наступних технологій та фрейморків:

– Python;

– Numpy;

– Tensorflow;

– Flask;

– ReactJS.

2.3.1 Python

Python був розроблений у 1991 році, як легка для розуміння і зручна у використанні мова. Інтерпретатор Python та його стандартні бібліотеки доступні як у вихідній так і у скомпільованій формі на всіх основних платформах. Python є мультипарадигменою мовою та підтримує окрім об'єктно орієнтованого підходу також декілька інших парадигм, зокрема: процедурну, функціональну та аспектно-орієнтовану [17]. Перевагами мови можна назвати простий та зрозумілий синтаксис та швидку розробку будь-яких додатків. Сьогодні її використовують для розробки веб-застосунків (зокрема бек-енд частини), програмування математики та написання системних скриптів. На сьогоднішній момент більша частина R&D проєктів в сфері машинного навчання використовують саме мову Python для своїх експериментів. Недоліком мови є її повільність виконання в порівняні з компільованими аналогами.

2.3.2 Numpy

Невід’ємною частиною будь-якого R&D проєкту в сфері машинного навчання є бібліотека numpy. Numpy – це бібліотека для наукових обчислень у Python, яка забезпечує роботу з багатовимірними масивами та високопродуктивними інструментами для роботи з цими масивами. Варто

зазначити також, що будь-який алгоритм, який може бути виражений як послідовність операцій над масивами і матрицями з використанням бібліотеки Numpy, працює настільки ж швидко як еквівалентний код написаний на мові C. Різниця двох підходів полягає в значно зрозумілішому синтаксисі мові Python та меншому об'ємі написаного коду [18]. Перевагою цієї бібліотеки серед інших аналогів є швидкість опрацьовування даних та простота маніпуляцій з ними.

2.3.3 Tensorflow

TensorFlow – загально-доступний фреймворк для роботи з алгоритмами глибинного машинного навчання. TensorFlow був розроблений компанією Google як внутрішня платформа для навчання та тестування алгоритмів машинного навчання в реальному часі з надвеликими масивами інформації. У листопаді 2015 року фреймворк був відкритий іншим розробникам під ліцензією Apache2.0.9.

TensorFlow надає для користувачів мови Python зручне та зрозуміле API. Фактичною фреймворк є обгорткою над високопродуктивною мовою C++. Всі бібліотеки та функції написані саме на цій мові, а Python лише координує їх роботу.

Код, написаний на TensorFlow, компілюєтсья на будь-яких процесорах та майже на всіх платформах: пристрої iOS і Android, локальна машина, кластер у хмарі, центральні або графічні процесори. Хмара від Google надає також можливість запустити Tensorflow на спеціальному процесорі TPU для отримання більшого прискорення в обчисленнях [19].

Перевагами Tensorflow серед аналогів є швидкість робити, простота в виконанні маніпуляцій над тензорами, наявність пачкової обробки тензорів, підтримка розширень на мові C++. Але фреймворк має також і один суттєвий недолік - складність налаштування програмних додатків при використанні складної архітектури моделей глибинного навчання.

2.3.4 Flask

Flask – це фреймворк Python, який був створений для спрощення розробки веб-додатків. Найбільш популярними проєктами, які базуться на фрейморку Flask являються веб сайти Pinterest, LinkedIn, тощо.

Flask вважається міні-фреймворком, оскільки він не потребує спеціальних сторонніх бібліотек чи засобів для розробки програмного забезпечення. У ньому присутні всі необхідні рівні абстракцій для роботи з файлами, роботи з базами даних та іншими додатками, які надають необхідні функції для розробників. Flask має також підтримку великої кількості розширень, які забезпечують додаткові можливості розробнику. Серед них можна виділити розширення для перевірки форм, контролю процесу завантаження, розгирення для маніпуляцій об'єктно-реляційних зв'язків, розширення для підтримки відкритих технологій аутентифікації [20].

Ключовими властивостями Flask є його простота в розробці програмного забезпечення, система керування запитами REST, підтримка шаблонів Jinja2, підтримка cookie на стороні веб-браузера, документація. Також варто відмітити розширення від сторонніх розробників, функціональну підтримку інших мов та підтримку Google App Engine.

Перевагами цього фреймворку можна назвати швидкість розробки на ньому, RESTful API, докладну документацію та підтримку Unicode. Flask має також декілька недоліків, наприклад, складність в створенні складних багаторівневих веб-додатків та організація їх архітектури.

2.3.5 ReactJS

Бібліотека React була вперше випущена компанією Facebook у 2013 році. React це бібліотека для створення користувацьких інтерфейсів. Однією з її характерних особливостей є можливість використовувати JSX, мова програмування з близьким до HTML синтаксисом, який компілюється в JavaScript. Розробники можуть вимагати більшої продуктивності додатків за

допомогою Virtual DOM. Створені компоненти можуть бути з легкістю змінені та використані заново в нових проєктах. Високий відсоток перевикористання коду збільшує покриття тестами, що, у свою чергу, призводить до більш високого рівня контролю якості [21].

Перевагами ReactJS є його простота в використанні, можливість рендеринга на сервері, зв'язування JavaScript і HTML в JSX робить компоненти простими для розуміння. Недоліками можна назвати погану та незрозумілу документацію, відсутність системи подій, шару даних, promises, тощо.

2.4 Аналіз безпеки даних

У даному проєкті відсутні будь-які персональні дані користувачів.

2.5 Висновки по розділу

У даному розділі було продемонстровано розробку програмного забезпечення для тренування та аналізу алгоритмів машинного навчання, яке складається з чотирьох блоків: генератор даних, оптимізатор гіперпараметрів, блок моделей машинного навчання та блок аналізу результатів тренування та тестування навчених моделей. Був розроблений веб-додаток для взаємодії з моделями машинного навчання та розроблені схеми розгортання цього веб-застосунку та схеми структурних класів програмного забезпечення. Також було описано переваги та недоліки обраного набору засобів розробки.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування розробленого програмного забезпечення є невід'ємною частиною циклу його розробки в усіх методологіях програмування. Воно гарантує чітку відповідність програми до технічного завдання, коректність реалізації всіх її основних блоків, наявність всіх заявлених можливостей та відсутність будь-яких критичних помилок.

При розробці алгоритмів для розв'язування задач біомеханіки, тестування є дуже важливим, оскільки розроблений програмний продукт є комплексним та вимагає перевірки усіх часткових випадків. Проте, оскільки в його основі лежить ймовірнісний алгоритм, то не можливо покрити 100% випадків, а тільки основні. Для цього створюються елементарні одиничні тести, які перевіряють лише основні компоненти програмного забезпечення.

Ключовими типами тестування даного програмного комплексу є навантажувальне тестування, що дозволяє перевірити швидкодію компонентів, що можуть бути чутливими до навантаження на систему. Це обумовлено високою обчислювальною складністю розробленого продукту.

Також будуть протестовані наступні компоненти програмного комплексу:

- Тестування часу відповіді сервера на запити;
- Тестування API сервера;
- Тестування моделей машинного навчання на коректність їх прогнозування;
- Тестування завантаження моделей машинного навчання в пам'ять;
- Тестування роботи інтерфейсу клієнтської частини;
- Навантажувальне тестування сервера;

Тестування цих функцій дозволить повністю перевірити отриманий програмний продукт на відповідність функціональним вимогам.

3.2 Опис процесів тестування

Для тестування даного програмного забезпечення було обрано режим чорної коробки, при якому ігнорується внутрішня логічна структура. Будуть проводитися наступні типи тестів:

- Димне тестування роботи API системи моделей машинного навчання;
- Димне тестування роботи API веб-додатку;
- Інтеграційне тестування компонентів програмного додатку;
- Навантажувальне тестування серверу.

Навантажувальне тестування є одним з основних компонентів для даного продукту, оскільки існують чіткі вимоги до швидкодії розроблювального програмного продукту. Для його проведення було проведено генерацію 1000 зразків вхідних даних та було замірено середню швидкість відповіді сервера. Тестування роботи API системи моделей машинного навчання було здійснене за допомогою юніт-тестів. Тестування роботи API веб-додатку виконується засобами програми Postman.

Працездатність веб-додатку перевіряється шляхом:

- Динамічного ручного тестування – введенням граничних та недопустимих значень в поля введення значень ступенів свободи;
- Динамічного ручного тестування на відповідність функціональним вимогам;
- Тестування веб-додатку в різних веб-браузерах та на різних діагоналей моніторів.
- Тестування зручності використання;
- Тестування коректності відображення графіків;

3.3 Опис контрольного прикладу

В даному розділі розглянуто проведення тестування на прикладі сценарію порівняння моделей штучної нейронної мережі, навчених на різному розмірі вхідних даних. Даний приклад наведений у таблиці 3.1.

Таблиця 3.1 – Опис контрольного прикладу

Мета тесту	Порівняння моделей штучної нейроної мережі, навчених на різному розмір вхідних даних
Початковий стан	Відкрита сторінка «Interact»
Вхідні дані	Вхідні дані беруться з файлу «Example1»
Схема проведення тесту	1. Вибрати тип моделі ANN та кількість вхідних даних – 1000; 2. Натиснути «Predict»; 3. Вибрати тип моделі ANN та кількість вхідних даних – 1000000; 4. Натиснути «Predict»;
Очікуваний результат	Для моделі ANN при кількості вхідних даних 1000 значення нормалізованої похибки для довжини повинне бути в діапазоні 0.1% до 170%, де найбільша похибка належить APB, а для моментів плеча – від 0.3% до 108%, де найбільша похибка належить EIND_9. Для моделі ANN при кількості вхідних даних 1000000 значення нормалізованої похибки для довжини повинне бути в діапазоні 0.01% до 2.5%, а для моментів плеча – від 0.02% до 11%.
Кінцевий результат	Після проведення даного тесту стан системи не повинен змінитися.

Розглянутий тест відноситься до тестів базової функціональності та виконується за допомогою графічного інтерфейсу користувача. Для його виконання повинні бути запущені сервер обробки та сервер клієнтської частини. Під час проведення тесту слід перевіряти чи співпадає отриманий результат х очікуваним результатом. Результат виконання тесту наведеного в таблиці 3.1 зображено на рисунках 3.1, 3.2 та 3.3.

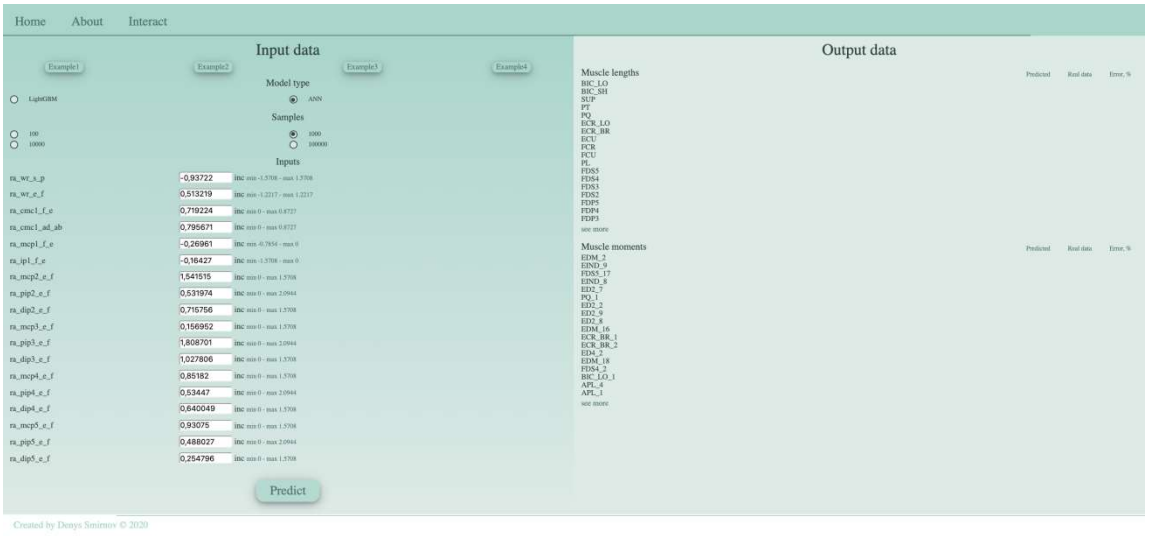


Рисунок 3.1 – Вхідні дані

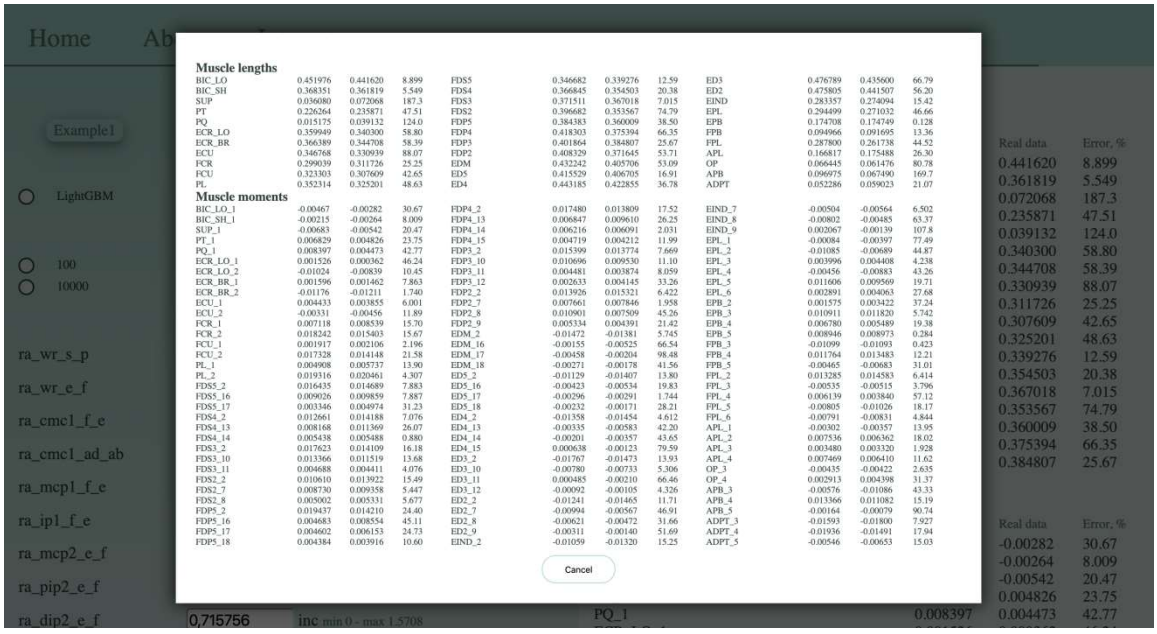


Рисунок 3.2 – Результат обробки даних моделі ANN, що навчалася з набором даних у 1000 зразків

Рисунок 3.3 – Результат обробки даних моделі ANN, що навчалася з набором даних у 1000000 зразків

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання розробленого програмного забезпечення необхідно виконати наступні кроки.

Крок 1. Інсталювати Python 3.7.

Крок 2. Інсталювати залежності виконавши команду «pip install –r requirements.txt».

Крок 3. Запустити сервер по обробці даних виконавши команду «python3 app.py».

Крок 4. Інсталювати node.js та npm.

Крок 5. Інсталювати залежності виконавши команду “nmpinstall”.

Крок 6. Виконати команду “npmrun-dev ” для того щоб стартувати веб-сервер.

4.2 Робота з програмним забезпеченням

Розроблене програмне забезпечення передбачає наступний сценарій використання.

Крок 1. Відкриття застосунку в браузері.

Крок 2. Введення 18-и значень ступеней свободи постави або вибір цих значень із гтових файлів.

Крок 3. Натискання кнопки «Predict».

Крок 4. Вивід результатів та аналізу обробки.

ВИСНОВКИ

У ході дипломного проєкту був проведений аналіз області біомеханіки опорно-рухового апарату, описано сильні та слабкі сторони сучасних рішень протезування верхніх кінцівок людини, вивчені R&D проєкти на основі машинного навчання, що вирішують проблему перетворення сигналів електроміограми активації м’язів у рух. На основі цього був розроблений програмний комплекс для вирішення та аналізу частини задачі біомеханіки опорно-рухового апарату людини, а саме визначення м’язових характеристик системи з її постави відносно 18 ступенів свободи.

В цій роботі було розроблене програмне забезпечення для генерування даних, тренування моделей машинного навчання, підбору гіперпараметрів тааналізу, а також результуюче програмне забезпечення для вирішення задачі динаміки опорно-рухового апарату. Для демонстрації та аналізу навчених моделей був створений веб-застосунок. Найкращі показники в швидкості та точності показали алгоритми градієнтного підсилювання та повнозв’язної штучної нейронної мережі. Нормалізовані похибки для довжин та моментів плеча нейронної мережі були 0.08% та 0.18% відповідно, легкого методу градієнтного підсилювання - 0.53% та 0.12%.

Раніше було застосовано досить багато рішень у застосуванні технологій машинного навчання для проблеми м’язово-скелетної динаміки, в основному зосереджених на прогнозуванні м'язових сил і кутів суглобів із сигналів електроміограми. Але жодна не використовувала поставу суглобів як вхід для підрахунку довжини та моментів плечей м'язів верхньої кінцівки, необхідні для здійснення руху. Раніше це завдання було вирішене моделями, які вимагають моделювання нелінійних взаємозв'язків між основними біологічними процесами. У цій роботі було продемонстровано, як методи машинного навчання, які не мають прямого визначення та опису взаємозв’язків між різними біомеханічними деталями, можуть вирішити задачу перетворення постави суглобів в м’язові характеристики для здійснення руху.

ПЕРЕЛІК ПОСИЛАНЬ

1) Song R, Tong KY. 2005. Using recurrent artificial neural network model to estimate voluntary elbow torque in dynamic situations. Med Biol Eng Comput43:473–480. doi:10.1007/BF02344728

2) Zhang B, Horváth I, Molenbroek JFM, Snijders C. 2010. Using artificial neural networks for human body posture prediction. Int J Ind Ergon40:414–424. doi:10.1016/j.ergon.2010.02.003

3) Xia P, Hu J, Peng Y. 2018. EMG-Based Estimation of Limb Movement Using Deep Learning With Recurrent Convolutional Neural Networks. Artif Organs42:E67–E77. doi:10.1111/aor.13004

4) Kidziński Ł, Ong C, Mohanty SP, Hicks J, Carroll S, Zhou B, Zeng H, Wang F, Lian R, Tian H, Jaśkowski W, Andersen G, Lykkebø OR, Toklu NE, Shyam P, Srivastava RK, Kolesnikov S, Hrinchuk O, Pechenko A, Ljungström M, Wang Zhen, Hu X, Hu Z, Qiu M, Huang J, Shpilman A, Sosin I, Svidchenko O, Malysheva A, Kudenko D, Rane L, Bhatt A, Wang Zhengfei, Qi P, Yu Z, Peng P, Yuan Q, Li W, Tian Y, Yang R, Ma P, Khadka S, Majumdar S, Dwiel Z, Liu Y, Tumer E, Watson J, Salathé M, Levine S, Delp S. 2020. Artificial Intelligence for Prosthetics: Challenge Solutions In: Escalera S, Herbrich R, editors. The NeurIPS ’18 Competition. Cham: Springer International Publishing. pp. 69–128. doi:10.1007/978-3-030-29135-8_4

5) Kidziński Ł, Mohanty SP, Ong C, Huang Z, Zhou S, Pechenko A, Stelmaszczyk A, Jarosik P, Pavlov M, Kolesnikov S, Plis S, Chen Z, Zhang Z, Chen J, Shi J, Zheng Z, Yuan C, Lin Z, Michalewski H, Miłoś P, Osiński B, Melnik A, Schilling M, Ritter H, Carroll S, Hicks J, Levine S, Salathé M, Delp S. 2018. Learning to Run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. ArXiv180400361 Cs Stat.

6) Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors.

Advances in Neural Information Processing Systems 30. Curran Associates, Inc. pp. 3146–3154.

7) Sepulveda F, Wells DM, Vaughan CL. 1993. A neural network representation of electromyography and joint dynamics in human gait. J Biomech26:101–109. doi:10.1016/0021-9290(93)90041-C

8) Patla AE. 1985. Some characteristics of EMG patterns during locomotion: implications for the locomotor control process. J Mot Behav17:443–461.

9) Xia P, Hu J, Peng Y. 2018. EMG-Based Estimation of Limb Movement Using Deep Learning With Recurrent Convolutional Neural Networks. Artif Organs42:E67–E77. doi:10.1111/aor.13004

10) Hartmann C, Boedecker J, Obst O, Ikemoto S, Asada M. 2013. Real-time inverse dynamics learning for musculoskeletal robots based on echo state gaussian process regression.Robotics: Science and Systems VIII. Presented at the Robotics: Science and Systems Conference. University of Sydney, Sydney, NSW, Australia: MIT Press. pp. 113–120.

11) Rane L, Ding Z, McGregor AH, Bull AMJ. 2019. Deep Learning for Musculoskeletal Force Prediction. Ann Biomed Eng47:778–789. doi:10.1007/s10439-018-02190-0

12) Zhou J, Chen J, Deng H, Qiao H. 2019. From rough to precise: human-inspired phased target learning framework for redundant musculoskeletal systems. Front Neurorobotics13:61. doi:10.3389/fnbot.2019.00061

13) Newcastle University engineers are developing new prosthetic limbs to empower people's lives. [Електроний ресурс]. – 2020 – Режим доступу: https://www.ncl.ac.uk/research/impact/casestudies/prosthetics/

14) Sobinov A, Boots MT, Gritsenko V, Fisher LE, Gaunt RA, Yakovenko S. 2019. Approximating complex musculoskeletal biomechanics using multidimensional autogenerating polynomials. bioRxiv 759878. doi:10.1101/759878

15) Gritsenko V, Hardesty RL, Boots MT, Yakovenko S. 2016. Biomechanical constraints underlying motor primitives derived from the musculoskeletal anatomy of the human arm. PLoS ONE11:1–18. doi:10.1371/journal.pone.0164050

16) Snoek J., Larochelle H., Adams R. P. Practical bayesian optimization of machine learning algorithms //Advances in neural information processing systems. – 2012. – С. 2951-2959.

17) Python. [Електроний ресурс]. - 2020 - : https://www.python.org/about/

18) Numpy. [Електроний ресурс]. -2020 -: https://numpy.org/

19) Tensorflow. [Електроний ресурс]. - 2020- : https://www.tensorflow.org/

20) Flask. [Електроний ресурс]. 2020 - : https://flask.palletsprojects.com/en/1.1.x/

21) ReactJS. [Електроний ресурс]. - 2020 - : https://reactjs.org/

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
“___” _____ 2020 р.

«Програмний комплекс для розв’язування біомеханіки опорно-рухового апарату»

Опис програми

КПІ.ІП-6101.045490.02.13

“ПОГОДЖЕНО”
Керівник проекту:
_____Є.А. Недашківський

Нормоконтроль:
_____ К.І. Ліщук

Виконавець:
_____Д.С. Смірнов

Київ – 2020 року

Тексти програмного коду

Програмний комплекс для розв'язування біомеханіки опорно-рухового апарату

(Найменування програми (документа))

DVD-R

(Вид носія даних)

41арк, 1.2 Гб

(Обсяг програми (документа) , арк.,)

Київ – 2020

```
import os
import sys
import logging

from flask import Flask, jsonify
from flask_cors import CORS

from logger import create_logger
from flask_app.extractor import Extractor
from configs import APP_DIR, GHIDORAH_APP, GHIDORAH_DATA


app = Flask(__name__)
cors = CORS(app, resources={r"/*": {"origins": "*"}})
logger = create_logger('app', level=logging.DEBUG)

sys.path.append(APP_DIR)
sys.path.append(GHIDORAH_APP)

error_message = lambda msg: jsonify({'status': 'fail', 'message': msg})

extractor = Extractor(
    features_path=os.path.join(GHIDORAH_DATA, 'test/features.csv'),
    len_path=os.path.join(GHIDORAH_DATA, 'test/len.csv'),
    momarm_path=os.path.join(GHIDORAH_DATA, 'test/momarm.csv'),
    examples_path=os.path.join(APP_DIR, 'examples/'),
)

import json

from flask import request, Response

from functools import wraps
from typing import Callable, Tuple, Dict

from flask_app import app, error_message, logger
from flask_app.evaluation import evaluation as model_evaluation, load_model


def unexpected_error(func: Callable):
    """ Catch all unexpected errors in the application """
    @wraps(func)
    def wrapper(*args, **kwargs):
        try:
```

```
            func(*args, **kwargs)
        except Exception as e:
            logger.exception(e)
        return error_message('Internal server error.')
    return wrapper


def get_data_from_request(request_) -> Tuple[str, int, Dict[str, float]]:
    data = json.loads(tuple(request_.form.to_dict().keys())[0])['data']
    logger.info(f'Get params: {data}')
    transformer = {'ANN': 'pct', 'LightGBM': 'lgb'}
    return (
        transformer[data['model']],
        int(data['samples']),
        {key: float(value) if value else .0 for key, value in data['indata'].items()},
    )


@unexpected_error
@app.route('/', methods=['GET'])
def test():
    return Response(json.dumps({'status': 'success', 'message': 'Here we are!'}),
                    headers={'Access-Control-Allow-Origin': '*'},
                    )


@unexpected_error
@app.route('/evaluation', methods=['POST'])
def evaluation():
    model_type, train_size, features = get_data_from_request(request_=request)
    if not model_type or not train_size or not features:
        return error_message('Invalid params')
    model = load_model(model_type=model_type, train_size=train_size)
    data = model_evaluation(model=model, features=features)
    return Response(json.dumps(dict(
        status          = 'success',
        predicted_len    = data['predicted_len'],
        predicted_momarm = data['predicted_momarm'],
        real_len         = data['real_len'],
        real_momarm      = data['real_momarm'],
        error_len        = data['error_len'],
        error_momarm     = data['error_momarm'],
    )), headers={'Access-Control-Allow-Origin': '*'},
    )
```

```
import os

import numpy as np

from functools import wraps
from typing import Callable, Union, Dict

from models import MuscleModel
from metrics import normalizer
from configs import GHIDORAH_APP
from flask_app import logger, extractor


def model_cache(func: Callable) -> Union[Callable, MuscleModel]:
    _cache = {}

    @wraps(func)
    def wrapper(model_type: str, train_size: str):
        if model_type in _cache:
            if train_size in _cache[model_type]:
                logger.info('Get model from the created ones.')
                model = _cache[model_type][train_size]
            else:
                model = func(model_type=model_type, train_size=train_size)
                _cache[model_type][train_size] = model
        else:
            model = func(model_type=model_type, train_size=train_size)
            _cache[model_type] = {}
            _cache[model_type][train_size] = model
        return model

    return wrapper


def load_model(model_type: str, train_size: int) -> MuscleModel:
    if model_type not in {'lgb', 'pct'}:
        raise TypeError(f'There is no such a model: {model_type}')
    model_path = os.path.join(GHIDORAH_APP,
f'saved_models/{model_type}/{model_type}_{train_size}')
    if not os.path.exists(model_path):
        raise OSError(f'There is no such a model: {model_path}')
    model = MuscleModel(model_type=model_type)
    model.load(model_path)
    return model
```

```
def evaluation(model: MuscleModel, features) -> Dict[str, Dict[str, float]]:
    data = extractor.get_data(features=features)

    features = np.array((list(data['features'].values()), ))
    y_len    = model.predict_len(x_test=features).iloc[0].to_dict()
    y_momarm = model.predict_momarm(x_test=features).iloc[0].to_dict()

    return {
        'predicted_len'    : y_len,
        'predicted_momarm' : y_momarm,
        'real_len'         : data['len'],
        'real_momarm'      : data['momarm'],
        'error_len'        : {
            muscle: normalizer.normalize(value  = abs(y_len[muscle]-
data['len'][muscle]),
                                         task   = 'len',
                                         muscle = muscle,
                                         ) * 100  # to get a percent value
            for muscle in y_len.keys()
        },
        'error_momarm'     : {
            muscle: normalizer.normalize(value  = abs(y_momarm[muscle]-
data['momarm'][muscle]),
                                         task   = 'momarm',
                                         muscle = muscle,
                                         ) * 100  # to get a percent value
            for muscle in y_momarm.keys()
        },
    }
import os
import json

from typing import Union, Dict

from preprocessing import data_reader


class Extractor(object):

    def __init__(self, examples_path: str, features_path: str, len_path: str,
momarm_path: str, cache_size: int = 10):
        self._features = data_reader.read(features_path)
```

```
            self._len      = data_reader.read(len_path)
            self._momarm   = data_reader.read(momarm_path)

            self._examples = []
            for file in os.listdir(examples_path):
                with open(os.path.join(examples_path, file), 'r') as f:
                    self._examples.append(json.load(f))
            self.original_size = len(self._examples)
            self.limit = cache_size

        def get_example_features(self, features: Dict[str, float]) -> Union[None,
Dict[str, Dict[str, float]]]:
            for ix, example in enumerate(self._examples):
                if all([abs(features[key] - example['features'][key]) < 1e-3 for key in
example['features'].keys()]):
                    return example
            return None

        def get_data(self, features: Dict[str, float]) -> Dict[str, Dict[str, float]]:
            example = self.get_example_features(features=features)
            if example is not None:
                return example

            index = self.get_feature_index(features)
            new_example = {
                'features': dict(self._features.iloc[index].to_dict()),
                'len'     : dict(self._len.iloc[index].to_dict()),
                'momarm'  : dict(self._momarm.iloc[index].to_dict()),
            }
            if len(self._examples) >= self.limit:
                self._examples.pop(self.original_size)
            self._examples.append(new_example)
            return self._examples[-1]

        def reset(self):
            self._examples = self._examples[self.original_size-1:]

    import os
    import logging

    from os.path import join as path_join
    from configs import LOGS_FOLDER
```

```
    def create_logger(file_name: str, level: int = logging.DEBUG):
        """
        Create basic logger
        :param file_name: name of log file
        :param level: level of logger to output
        :return: logger
        """
        logger = logging.getLogger(name=file_name)

        logger.handlers.clear()
        logger.setLevel(level)

        # create a file and Stream handlers
        if not os.path.exists(LOGS_FOLDER):
            os.mkdir(LOGS_FOLDER)
        fh = logging.FileHandler(filename=path_join(LOGS_FOLDER,
f'{file_name}.log'), mode='a', encoding='utf-8')
        ch = logging.StreamHandler()
        fh.setLevel(level)
        ch.setLevel(level)

        # create a logging format
        formatter = logging.Formatter('[%(asctime)s - %(levelname)s %(module)s
%(funcName)s] - %(message)s')
        fh.setFormatter(formatter)
        ch.setFormatter(formatter)

        # add the handlers to the logger
        logger.addHandler(fh)
        logger.addHandler(ch)

        return logger

    import os

    # Path to the application
    APP_DIR: str = str(os.path.dirname(os.path.abspath(__file__)))
    # Path to project
    PROJECT_DIR: str = APP_DIR[:APP_DIR.rfind('/')]

    # Path to the ghidorah project
    GHIDORAH_APP = '../../ghidorah/experiments/'
    GHIDORAH_DATA = '../../ghidorah/data/'
```

```
LOGS_FOLDER = os.path.join(APP_DIR, 'logs/')

# Flask app configs
HOST: str = '127.0.0.1'
PORT: str = '8000'

import os
from waitress import serve

from flask_app import app, endpoints
from configs import HOST, PORT


if __name__ == '__main__':
    serve(
        app,
        host=os.environ.get('HOST', HOST),
        port=os.environ.get('PORT', PORT),
        expose_tracebacks=True,
    )

from models.cls.lightgbm import LightGBM, PARAMS as LGB_PARAMS,
SPACE as LGB_SPACE
from models.cls.linear_regression import LinearRegression, PARAMS as
LR_PARAMS, SPACE as LR_SPACE
from models.cls.polynomial_regression import PolynomialRegression,
PARAMS as PLR_PARAMS, SPACE as PLR_SPACE
from models.cls.perceptron import Perceptron, PARAMS as PCT_PARAMS,
SPACE as PCT_SPACE
from models.cls.interface import ModelInterface
from models.cls.saver import Saver

from typing import Dict, Any, Tuple, List


class ModelChooser(object):
    def __init__(self):
        self._models = {
            'lgb': {
                'model': LightGBM, 'params': LGB_PARAMS, 'space': LGB_SPACE,
'type': 'one_task_learner',
            },
            'lr': {
```

```
                'model': LinearRegression, 'params': LR_PARAMS, 'space':
LR_SPACE, 'type': 'one_task_learner',
            },
            'plr': {
                'model': PolynomialRegression, 'params': PLR_PARAMS, 'space':
PLR_SPACE, 'type': 'one_task_learner',
            },
            'pct': {
                'model': Perceptron, 'params': PCT_PARAMS, 'space': PCT_SPACE,
'type': 'multi_task_learner',
            },
        }

    def __getitem__(self, model: str) -> Tuple[ModelInterface, Dict[str, Any],
Dict[str, Any]]:
        if model not in self.models.keys():
            raise TypeError(f'There is no such a model: {model}')
        return self.models[model]['model'], self.models[model]['params'],
self.models[model]['space']

    def __iter__(self):
        return self

    def __next__(self) -> Tuple[str, ModelInterface, Dict[str, Any], Dict[str,
Any]]:
        for key, value in self.models:
            yield key, value['model'], value['params'], value['space']
        raise StopIteration

    def type_of(self, model: str) -> str:
        if model not in self.models.keys():
            raise TypeError(f'There is no such a model: {model}')
        return self.models[model]['type']

    def __len__(self) -> int:
        return len(self.keys())

    def keys(self) -> List[str]:
        return list(self.models.keys())

    @property
    def models(self):
        return self._models
```

```
model_chooser = ModelChooser()


__all__ = [
    'LightGBM', 'LGB_PARAMS', 'LGB_SPACE',
    'PolynomialRegression', 'LR_PARAMS', 'LR_SPACE',
    'LinearRegression', 'PLR_PARAMS', 'PLR_SPACE',
    'Perceptron', 'PCT_PARAMS', 'PCT_SPACE',
    'model_chooser', 'ModelInterface',
]
import os
import joblib

import pandas as pd

from utils import not_fitted_error

from models.cls.interface import ModelInterface


def load_model(model_path: str, model_cls):
    """ Load the model """
    model = model_cls()
    model = model.load(model_path=model_path)
    return model


class BaseModel(ModelInterface):
    def __init__(self):
        self._model = None

    def fit(self, x_train: pd.DataFrame, y_train: pd.DataFrame):
        """ Fit the model """
        self.model.fit(x_train, y_train)

    @not_fitted_error
    def predict(self, x_test: pd.DataFrame):
        """ Returns the decision """
        return self.model.predict(x_test)

    @property
    def model(self):
        """ Returns internal model """
```

```
        return self._model

    def set_params(self, **params):
        """
        Set params to the model
        :param params: hyper parameters
        """
        self.model.set_params(**params)

    def save(self, model_path: str):
        """ Save model """
        path, file = os.path.split(model_path)
        if not os.path.exists(path):
            os.mkdir(path)
        joblib.dump(self.model, filename=model_path, protocol=4)

    def load(self, model_path: str):
        """ Load the model """
        if not os.path.exists(model_path):
            raise OSError(f'There is no such a model: {model_path}')
        model = joblib.load(model_path)
        if isinstance(model, dict):
            for key, value in model.items():
                setattr(self, key, value)
        else:
            self._model = model
        return self
class ModelInterface(object):

    def fit(self, x_train, y_train, *args, **kwargs):
        raise NotImplementedError

    def predict(self, x_test):
        raise NotImplementedError

    def save(self, model_path: str):
        raise NotImplementedError

    def load(self, model_path: str):
        raise NotImplementedError
class LightGBM(BaseModel):

    def __init__(self, **params):
        super(BaseModel, self).__init__()
```

```
            self._model = lightgbm.LGBMRegressor(**params)

        def fit(self, x_train: pd.DataFrame, y_train):
            """ Fit the model """
            train_inds, val_inds = train_test_split(
                [i for i in range(len(x_train))], shuffle=True, test_size=0.2,
random_state=6223,
            )

            self.model.fit(
                X          = x_train.iloc[train_inds],
                y          = y_train.iloc[train_inds],
                eval_set   = [(x_train.iloc[val_inds], y_train.iloc[val_inds])],
                eval_metric = 'mse',
                verbose    = -1,
            )

    import numpy as np
    import pandas as pd

    from math import sqrt
    from typing import Sequence, List, Union
    from sklearn.metrics import mean_squared_error

    from config import LNORM, RNORM, lens, momarms
    import os
    import warnings

    import numpy as np
    import pandas as pd
    import tensorflow as tf

    from hyperopt import hp
    from hyperopt.pyll import scope
    from sklearn.model_selection import train_test_split

    from keras.models import Sequential, model_from_json
    from keras.layers import Dense, BatchNormalization, Dropout
    from keras.callbacks import EarlyStopping, ReduceLROnPlateau

    from models.cls.interface import ModelInterface
    from metrics import mean_rmse_quantile, mean_normalized_error_quantile
```

```
warnings.filterwarnings('ignore')
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

# Set random seeds
seed_value = 12321
tf.random.set_random_seed(seed_value)
np.random.seed(seed_value)
os.environ['PYTHONHASHSEED'] = str(seed_value)

# static model hyper parameters
PARAMS = {
   'batch_size' : 256,
   'epochs'     : 24,
}
# distributions of model parameters
SPACE = {
   'nodes_1':
      scope.int(hp.qloguniform('model_nodes_1', np.log(16), np.log(2056), 2)),
   'nodes_2':
      hp.choice('model_nodes_2', [None,
scope.int(hp.qloguniform('model_inner_nodes_2', np.log(16), np.log(2056), 2))]),
   'norm':
      hp.choice('model_norm', [True, False]),
   'dropout':
      hp.choice('model_dropout', [True, False]),
}


def load_model(model_path: str):
   """ Load the model """
   model = Perceptron(num_features=1, outputs=1)
   model = model.load(model_path=model_path)
   return model


class Perceptron(ModelInterface):

   def __init__(self,
            num_features    : int  = 18,
            outputs         : int  = 33,
            nodes_1         : int  = None,
            nodes_2         : int  = None,
            norm            : bool = False,
```

```
                dropout         : bool  = False,
                batch_size      : int   = 32,
                epochs          : int   = 1,
                ):
        """
        :param num_features: number of inputs
        :param outputs: number of outputs
        :param batch_size: number of samples to put into one batch
        :param epochs: number of iterations
        """
        if (nodes_1 is not None and nodes_1 < 0) or (nodes_2 is not None and
nodes_2 < 0) or (nodes_2 and nodes_1 is None):
            raise ValueError('Check perceptron input.')

        self.batch_size = batch_size
        self.epochs     = epochs

        if nodes_1 is not None:  # It is shit, don't look at me like that
            if nodes_2 is not None:
                self.nodes = [(num_features, nodes_1), (nodes_1, nodes_2), (nodes_2,
outputs)]
            else:
                self.nodes = [(num_features, nodes_1), (nodes_1, outputs)]
        else:
            self.nodes = [(num_features, outputs)]

        self.norm           = norm
        self.dropout        = dropout

        self._model = Sequential()
        for n_inputs, n_outputs in self.nodes[:-1]:
            self.model.add(Dense(n_outputs, input_dim=n_inputs,
init='glorot_normal', activation=tf.nn.leaky_relu))
            if self.norm:
                self.model.add(BatchNormalization())
            if self.dropout:
                self.model.add(Dropout(rate=0.5, seed=413))
        self.model.add(Dense(self.nodes[-1][1], input_dim=self.nodes[-1][0],
init='glorot_normal', activation='linear'))

    @property
    def num_features(self) -> int:
        return self.nodes[0][0]
```

```
        @property
        def num_outputs(self) -> int:
            return self.nodes[-1][1]

        @property
        def model(self) -> Sequential:
            return self._model

        @staticmethod
        def _loss(model, output):
            errors = tf.math.sqrt(tf.square(tf.subtract(model, output)))
            quantile = tf.contrib.distributions.percentile(errors, q=95., axis=[0])
            mask = tf.greater_equal(errors, quantile)
            loss = [tf.reduce_mean(tf.boolean_mask(errors[:, i], mask[:, i])) for i in
range(errors.shape[1])]

            output_quantile = tf.contrib.distributions.percentile(loss, q=70., axis=0)
            output_mask = tf.greater_equal(loss, output_quantile)
            output_means = tf.boolean_mask(loss, output_mask)

            return tf.reduce_mean(output_means)

        def fit(self,
                x_train            : pd.DataFrame,
                y_train            : pd.DataFrame,
                verbose             : bool = False,
                use_multiprocessing : bool = False,
                ) -> ModelInterface:
            """
            :param x_train: x sample for training
            :param y_train: target sample for training
            :param verbose:
            :param use_multiprocessing:
            """
            self.model.compile(loss      = self._loss,
                               optimizer = 'adam',
                               metrics   = [self._loss],
                               )
            self.model.fit(
                x                = x_train,
                y                = y_train,
                epochs            = self.epochs,
                batch_size         = self.batch_size,
                verbose           = verbose,
```

```
                validation_split    = 0.2,
                shuffle             = True,
                use_multiprocessing = use_multiprocessing,
                callbacks = [
                    EarlyStopping(monitor='val_loss', min_delta=0, patience=4,
verbose=verbose, mode='auto'),
                    ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2,
min_lr=1e-8, verbose=verbose),
                ]
            )
            return self

        def predict(self, x_test: pd.DataFrame, use_multiprocessing: bool = False):
            """ Makes the predictions """
            return self.model.predict(x                  = x_test,
                                      batch_size         = self.batch_size,
                                      use_multiprocessing = use_multiprocessing,
                                      )

        def save(self, model_path: str):
            """ Saves model """
            path, file = os.path.split(model_path)
            if not os.path.exists(path):
                os.mkdir(path)
            model_json = self.model.to_json()  # serialize model to JSON
            with open(os.path.join(model_path, 'model.json'), "w") as json_file:
                json_file.write(model_json)
            self.model.save_weights(os.path.join(model_path, "model.h5"))  # serialize
weights to HDF5

        def load(self, model_path: str) -> ModelInterface:
            if not os.path.exists(model_path):
                raise OSError(f'There is no such a model: {model_path}')
            with open(os.path.join(model_path, 'model.json'), 'r') as json_file:  # load
json and create model
                loaded_model_json = json_file.read()
            self._model = model_from_json(loaded_model_json,
custom_objects={'leaky_relu': tf.nn.leaky_relu})
            self.model.load_weights(os.path.join(model_path, "model.h5"))  # load
weights into new model
            return self


    if __name__ == '__main__':
```

```
        import shutil
        from utils import timeit_context
        from preprocessing import data_reader
        from config import DATA_DIR
        # ============================== Test the perceptron training
==============================

        with timeit_context('Read data'):
            x_train = data_reader.read(os.path.join(DATA_DIR, 'test/features.csv'))
            y_train = data_reader.read(os.path.join(DATA_DIR, 'test/len.csv'))

        with timeit_context('Split data'):
            x_train, x_test, y_train, y_test = train_test_split(x_train, y_train,
test_size=0.2, random_state=324)

        with timeit_context('Fitting time'):
            model = Perceptron(num_features = x_train.shape[1],
                               outputs      = y_train.shape[1],
                               nodes_1      = 256,
                               nodes_2      = 512,
                               norm         = True,
                               batch_size   = 128,
                               epochs       = 1000,
                               )
            model.fit(x_train=x_train, y_train=y_train, verbose=True,
use_multiprocessing=True)

        with timeit_context('Saving time'):
            model.save('temp_pct/')
            del model

        with timeit_context('Loading time'):
            model = Perceptron(num_features=x_train.shape[1],
outputs=y_train.shape[1])
            model = model.load('temp_pct/')

        with timeit_context('Predicting time'):
            predictions = model.predict(x_test)
        print(f'MSE: {mean_rmse_quantile(y_true=y_test,
y_pred=pd.DataFrame(predictions)):.5f}m')
        print(f'MNE: {mean_normalized_error_quantile(y_true=y_test,
y_pred=pd.DataFrame(predictions))*100:.2f}%')

        shutil.rmtree(path='temp_pct/')
```

```
from models.prod_models.interface import IModel
from models.prod_models.one_task_model import OneTaskModel
from models.prod_models.multy_task_model import MultiTaskModel
from models.prod_models.muscle_model import MuscleModel


__all__ = [
    'IModel', 'OneTaskModel', 'MultiTaskModel', 'MuscleModel',
]
import pandas as pd


class IModel(object):
    def fit(self, x_train: pd.DataFrame, y_train_len: pd.DataFrame,
y_train_momarm: pd.DataFrame) -> object:
        raise NotImplementedError

    def predict(self, x_test: pd.DataFrame) -> pd.DataFrame:
        raise NotImplementedError

    def predict_len(self, x_test: pd.DataFrame):
        raise NotImplementedError

    def predict_momarm(self, x_test: pd.DataFrame):
        raise NotImplementedError

    def load(self, model_path: str) -> object:
        raise NotImplementedError

    def save(self, model_path: str):
        raise NotImplementedError
import os
import json

import pandas as pd

from typing import List

from models.cls import ModelInterface, model_chooser
from models.prod_models.interface import IModel
from config import MODELS_HP_DIR


class MultiTaskModel(IModel):
```

```
        def __init__(self, model_type: str, pretrained: str = None):
            self._len_model: ModelInterface = self._load_pretrained(
                model      = model_chooser[model_type][0],
                prefix     = 'length',
                pretrained = pretrained if pretrained is not None else
os.path.join(MODELS_HP_DIR, model_type),
            )
            self._momarm_model: ModelInterface = self._load_pretrained(
                model      = model_chooser[model_type][0],
                prefix     = 'momarm',
                pretrained = pretrained if pretrained is not None else
os.path.join(MODELS_HP_DIR, model_type),
            )
            self._len_columns = []
            self._momarm_columns = []

        def fit(self,
                x_train       : pd.DataFrame,
                y_train_len   : pd.DataFrame,
                y_train_momarm : pd.DataFrame,
                verbosity     : bool = True,
                ) -> object:
            if verbosity:
                print('Len model training...')
            self.len_model.fit(x_train=x_train, y_train=y_train_len)
            self._len_columns = list(y_train_len.columns)
            if verbosity:
                print('Momarm model training...')
            self.momarm_model.fit(x_train=x_train, y_train=y_train_momarm)
            self._momarm_columns = list(y_train_momarm.columns)
            return self

        def predict(self, x_test: pd.DataFrame) -> pd.DataFrame:
            return pd.concat([self.predict_len(x_test=x_test),
self.predict_momarm(x_test=x_test)], axis=1)

        def predict_len(self, x_test: pd.DataFrame) -> pd.DataFrame:
            return pd.DataFrame(self.len_model.predict(x_test=x_test),
columns=self.len_columns)
            # return self.len_model.predict(x_test=x_test)

        def predict_momarm(self, x_test: pd.DataFrame) -> pd.DataFrame:
```

```
            return pd.DataFrame(self.momarm_model.predict(x_test=x_test),
columns=self.momarm_columns)
            # return self.momarm_model.predict(x_test=x_test)

        def load(self, model_path: str) -> object:
            """ Load the model """
            if not os.path.exists(model_path):
                raise OSError(f'There is no such a path: {model_path}')
            with open(os.path.join(model_path, 'meta.json'), 'r') as f:
                meta = json.load(f)
                self._len_columns    = meta['len_columns']
                self._momarm_columns = meta['momarm_columns']
            self._len_model    = self.len_model.load(os.path.join(model_path, 'len'))
            self._momarm_model =
self.momarm_model.load(os.path.join(model_path, 'momarm'))
            return self

        def save(self, model_path: str):
            if not os.path.exists(model_path):
                os.mkdir(model_path)
            if not os.path.exists(os.path.join(model_path, 'len/')):
                os.mkdir(os.path.join(model_path, 'len/'))
            if not os.path.exists(os.path.join(model_path, 'momarm//')):
                os.mkdir(os.path.join(model_path, 'momarm/'))
            self.len_model.save(model_path=os.path.join(model_path, 'len'))
            self.momarm_model.save(model_path=os.path.join(model_path,
'momarm'))
            with open(os.path.join(model_path, 'meta.json'), 'w') as f:
                json.dump(
                    {
                        'len_columns'    : self.len_columns,
                        'momarm_columns' : self.momarm_columns,
                    },
                    f,
                )

        @property
        def len_model(self) -> ModelInterface:
            return self._len_model

        @property
        def momarm_model(self) -> ModelInterface:
            return self._momarm_model
```

```
        @property
        def len_columns(self) -> List[str]:
            return self._len_columns

        @property
        def momarm_columns(self) -> List[str]:
            return self._momarm_columns

        @staticmethod
        def _load_pretrained(model: type, prefix: str, pretrained: str = None) ->
ModelInterface:
            if not os.path.exists(pretrained):
                raise OSError(f'The path does not exist: {pretrained}')
            with open(os.path.join(pretrained, f'{prefix}.json'), 'r') as f:
                return model(**json.load(f))

        def set_n_jobs(self, n_jobs: int = -1):
            pass
    from models.cls import model_chooser
    from models.prod_models.interface import IModel
    from models.prod_models.multy_task_model import MultiTaskModel
    from models.prod_models.one_task_model import OneTaskModel


    class MuscleModel(IModel):

        def __init__(self, model_type: str, pretrained: str = None):
            if model_type not in model_chooser.keys():
                raise TypeError(f'There is no such a model: {model_type}. Available
models: {list(model_chooser.keys())}')
            if model_chooser.type_of(model_type) == 'one_task_learner':
                self._model = OneTaskModel(model_type=model_type,
pretrained=pretrained)
            elif model_chooser.type_of(model_type) == 'multi_task_learner':
                self._model = MultiTaskModel(model_type=model_type,
pretrained=pretrained)
            else:
                raise NotImplemented(f'Implement the type
{model_chooser.type_of(model_type)}')

        def __getattr__(self, item: str):
            return getattr(self.model, item)

        def __getattribute__(self, item: str):
```

```
            if item == 'model' or item == '_model':
                return object.__getattribute__(self, item)
            return getattr(self.model, item)

        @property
        def model(self) -> IModel:
            return self._model
    import os
    import json

    import pandas as pd

    from tqdm import tqdm
    from typing import Tuple, Dict

    from models.cls import ModelInterface, model_chooser
    from models.prod_models.interface import IModel
    from config import MODELS_HP_DIR


    class OneTaskModel(IModel):

        def __init__(self, model_type: str, pretrained: str = None):
            self._len_models: Dict[str, ModelInterface] = self._load_pretrained(
                model      = model_chooser[model_type][0],
                prefix     = 'length_',
                pretrained = pretrained if pretrained is not None else
os.path.join(MODELS_HP_DIR, model_type),
            )
            self._momarm_models: Dict[str, ModelInterface] = self._load_pretrained(
                model      = model_chooser[model_type][0],
                prefix     = 'momarm_',
                pretrained = pretrained if pretrained is not None else
os.path.join(MODELS_HP_DIR, model_type),
            )

        def fit(self, x_train: pd.DataFrame, y_train_len: pd.DataFrame,
y_train_momarm: pd.DataFrame, verbosity: bool = True) -> object:
            # Train length models
            for muscle in tqdm(y_train_len.columns, total=y_train_len.shape[1],
desc='Length training') if verbosity else y_train_len.columns:
                if muscle not in self.len_models:
                    raise KeyError(f'There is no such a model for the muscle {muscle}.')
```

```
            self.len_models[muscle].fit(x_train=x_train,
y_train=y_train_len[muscle])
        self._len_models = {muscle: self.len_models[muscle] for muscle in
y_train_len.columns}

        # Train momarm models
        for muscle in tqdm(y_train_momarm.columns,
total=y_train_momarm.shape[1], desc='Momarm training') if verbosity else
y_train_momarm.columns:
            if muscle not in self.momarm_models:
                raise KeyError(f'There is no such a model for the muscle {muscle}.')
            self.momarm_models[muscle].fit(x_train=x_train,
y_train=y_train_momarm[muscle])
        self._momarm_models = {muscle: self.momarm_models[muscle] for
muscle in y_train_momarm.columns}
        return self

    def predict(self, x_test: pd.DataFrame) -> pd.DataFrame:
        return pd.concat([self.predict_len(x_test=x_test),
self.predict_momarm(x_test=x_test)], axis=1)

    def predict_len(self, x_test: pd.DataFrame) -> pd.DataFrame:
        return pd.DataFrame({key: model.predict(x_test=x_test) for key, model in
self.len_models.items()})

    def predict_momarm(self, x_test: pd.DataFrame) -> pd.DataFrame:
        return pd.DataFrame({key: model.predict(x_test=x_test) for key, model in
self.momarm_models.items()})

    def load(self, model_path: str) -> object:
        """ Load the model """
        if not os.path.exists(model_path):
            raise OSError(f'There is no such a path: {model_path}')
        with open(os.path.join(model_path, 'meta.json'), 'r') as f:
            meta = json.load(f)
        self._len_models = {
            muscle: model.load(os.path.join(model_path, 'len', muscle))
            for muscle, model in self.len_models.items()
        }
        self._momarm_models = {
            muscle: model.load(os.path.join(model_path, 'momarm', muscle))
            for muscle, model in self.momarm_models.items()
        }
```

```
            self._len_models    = {muscle: self.len_models[muscle] for muscle in
meta['len_columns']}
            self._momarm_models = {muscle: self.momarm_models[muscle] for
muscle in meta['momarm_columns']}
            return self

        def save(self, model_path: str):
            if not os.path.exists(model_path):
                os.mkdir(model_path)
            if not os.path.exists(os.path.join(model_path, 'len/')):
                os.mkdir(os.path.join(model_path, 'len/'))
            if not os.path.exists(os.path.join(model_path, 'momarm//')):
                os.mkdir(os.path.join(model_path, 'momarm/'))
            for key, model in self.len_models.items():
                model.save(model_path=os.path.join(model_path, 'len', key))
            for key, model in self.momarm_models.items():
                model.save(model_path=os.path.join(model_path, 'momarm', key))
            with open(os.path.join(model_path, 'meta.json'), 'w') as f:
                json.dump(
                    {
                        'len_columns'   : list(self.len_models.keys()),
                        'momarm_columns': list(self.momarm_models.keys()),
                    },
                    f,
                )

        @property
        def len_models(self) -> Dict[str, ModelInterface]:
            return self._len_models

        @property
        def momarm_models(self) -> Dict[str, ModelInterface]:
            return self._momarm_models

        @staticmethod
        def _load_pretrained(model: type, prefix: str, pretrained: str = None) ->
Dict[str, ModelInterface]:
            if not os.path.exists(pretrained):
                raise OSError(f'The path does not exist: {pretrained}')
            models = {}
            for file in os.listdir(pretrained):
                if not file.startswith(prefix):
                    continue
                with open(os.path.join(pretrained, file), 'r') as f:
```

```
                models[file.replace('.json', '').replace(prefix, '')] =
model(**json.load(f))
        return models

    @staticmethod
    def _get_meta_muscle(muscle: str) -> Tuple[str, str]:
        separator = muscle.find('_')
        return muscle[:separator], muscle[separator+1:]

    def set_n_jobs(self, n_jobs: int = -1):
        for model in self.len_models.keys():
            self.len_models[model].model.n_job = n_jobs
        for model in self.momarm_models.keys():
            self.momarm_models[model].model.n_job = n_jobs
from models.cls import LinearRegression, LightGBM, Perceptron,
PolynomialRegression
from models.prod_models import OneTaskModel, MultiTaskModel,
MuscleModel

__all__ = [
    'LightGBM', 'LinearRegression', 'PolynomialRegression', 'Perceptron',
    'OneTaskModel', 'MuscleModel', 'MultiTaskModel',
]
import os
import sys
import time

from functools import wraps
from typing import Callable


def format_time(total_time: float):
    """
    Format input time to format: '{hours}h {minutes}m {seconds}s'
    :param total_time: time in seconds
    :return: str
    """
    h    = total_time // 3600
    min_ = total_time % 3600 // 60
    sec  = total_time % 3600 % 60
    if h:
        return f'{h:.0f}h {min_:.0f}min {sec:.0f}sec'
    return f'{min_:.0f}min {sec:.0f}sec'
```

```
    def timeit(msg: str = None, output=sys.stdout):
        """
        Decorator. Print time of function`s execution with the message: msg
        :param msg: message to print with the time
        :param output: a file-like object (stream); defaults to the current sys.stdout.
        """
        def _timeit(func):
            @wraps(func)
            def wrapper(*args, **kwargs):
                start = time.time()
                res = func(*args, **kwargs)
                print('{msg}:\t{time}'.format(msg=msg, time=format_time(time.time() -
start)), file=output)
                return res

            nonlocal msg
            if msg is None:
                msg = 'Time to execute
{func_name}'.format(func_name=func.__name__)
            return wrapper

        return _timeit


    class timeit_context:
        """
        Context manager. Print time of block's execution with message: msg
        :param msg: message to print with the time
        :param output: a file-like object (stream); defaults to the current sys.stdout.
        :return:
        """

        def __init__(self, msg: str = None, output=sys.stdout):
            if msg:
                self.message = msg
            else:
                self.message = 'Time to execute block'
            self.output = output
            self.start_time = time.time()

        def __enter__(self):
            return self
```

```
        def __exit__(self, exc_type, exc_val, exc_tb):
            if self.output:
                print('{msg}:\t{time}'.format(
                    msg=self.message, time=format_time(time.time() - self.start_time)),
file=self.output,
                )


    def not_fitted_error(func):
        """ Decorator that handles AttributeException for the model """
        @wraps(func)
        def wrapper(*args, **kwargs):
            try:
                return func(*args, **kwargs)
            except AttributeError:
                raise NotFittedError(
                    'The model is not fitted yet. Call `fit` with appropriate arguments
before using this method.'
                )
        return wrapper


    class NotFittedError(Exception):
        pass


    def print_function(output_file: str, print_func: Callable = None) -> Callable:
        def wrapper(txt, *args, **kwargs):
            with open(output_file, 'a') as file:
                print(txt, *args, file=file, **kwargs)
            if print_func is not None:
                print_func(txt)

        if os.path.exists(output_file):
            with open(output_file, 'w') as file:
                file.write('')
        return wrapper

    const inputs = [
      {'name': 'ra_wr_s_p',
            'min': '-1.5708',
            'max': '1.5708',
            'desc': 'wrist rotation motion'},
      {'name': 'ra_wr_e_f',
```

```
        'min': '-1.2217',
        'max': '1.2217',
        'desc': 'wrist flexion/extension motion'},
    {'name': 'ra_cmc1_f_e',
        'min': '0',
        'max': '0.8727',
        'desc': 'thumb proximal flexion/extension motion'},
    {'name': 'ra_cmc1_ad_ab',
        'min': '0',
        'max': '0.8727',
        'desc': 'thumb proximal abduction/adduction motion'},
    {'name': 'ra_mcp1_f_e',
        'min': '-0.7854',
        'max': '0',
        'desc': 'thumb central flexion/extension motion'},
    {'name': 'ra_ip1_f_e',
        'min': '-1.5708',
        'max': '0',
        'desc': 'thumb distal flexion/extension motion'},
    {'name': 'ra_mcp2_e_f',
        'min': '0',
        'max': '1.5708',
        'desc': 'index proximal flexion/extension motion'},
    {'name': 'ra_pip2_e_f',
        'min': '0',
        'max': '2.0944',
        'desc': 'index central flexion/extension motion'},
    {'name': 'ra_dip2_e_f',
        'min': '0',
        'max': '1.5708',
        'desc': 'index distal flexion/extension motion'},
    {'name': 'ra_mcp3_e_f',
        'min': '0',
        'max': '1.5708',
        'desc': 'middle proximal flexion/extension motion'},
    {'name': 'ra_pip3_e_f',
        'min': '0',
        'max': '2.0944',
        'desc': 'middle central flexion/extension motion'},
    {'name': 'ra_dip3_e_f',
        'min': '0',
        'max': '1.5708',
        'desc': 'middle distal flexion/extension motion'},
    {'name': 'ra_mcp4_e_f',
```

```
        'min': '0',
        'max': '1.5708',
        'desc': 'ring proximal flexion/extension motion'},
    {'name': 'ra_pip4_e_f',
        'min': '0',
        'max': '2.0944',
        'desc': 'ring central flexion/extension motion'},
    {'name': 'ra_dip4_e_f',
        'min': '0',
        'max': '1.5708',
        'desc': 'ring distal flexion/extension motion'},
    {'name': 'ra_mcp5_e_f',
        'min': '0',
        'max': '1.5708',
        'desc': 'pinky proximal flexion/extension motion'},
    {'name': 'ra_pip5_e_f',
        'min': '0',
        'max': '2.0944',
        'desc': 'pinky central flexion/extension motion'},
    {'name': 'ra_dip5_e_f',
        'min': '0',
        'max': '1.5708',
        'desc': 'pinky distal flexion/extension motion'}];

export default inputs

export const obj = {
  'ra_wr_s_p': '',
  'ra_wr_e_f': '',
  'ra_cmc1_f_e': '',
  'ra_cmc1_ad_ab': '',
  'ra_mcp1_f_e': '',
  'ra_ip1_f_e': '',
  'ra_mcp2_e_f': '',
  'ra_pip2_e_f': '',
  'ra_dip2_e_f': '',
  'ra_mcp3_e_f': '',
  'ra_pip3_e_f': '',
  'ra_dip3_e_f': '',
  'ra_mcp4_e_f': '',
  'ra_pip4_e_f': '',
  'ra_dip4_e_f': '',
  'ra_mcp5_e_f': '',
  'ra_pip5_e_f': '',
```

```
  'ra_dip5_e_f': '',
}
export const obj1 = {
  'ra_wr_s_p': 1,
  'ra_wr_e_f': 2,
  'ra_cmc1_f_e': 0.003,
  'ra_cmc1_ad_ab': 0.2,
  'ra_mcp1_f_e': 0,
  'ra_ip1_f_e': 0,
  'ra_mcp2_e_f': 1,
  'ra_pip2_e_f': 0.008,
  'ra_dip2_e_f': 0,
  'ra_mcp3_e_f': 0,
  'ra_pip3_e_f': 0,
  'ra_dip3_e_f': 0,
  'ra_mcp4_e_f': 0,
  'ra_pip4_e_f': 0.5,
  'ra_dip4_e_f': 0.006,
  'ra_mcp5_e_f': 0,
  'ra_pip5_e_f': 0,
  'ra_dip5_e_f': 1,
}
const momarm = {
  EDM_2: '',
  EIND_9: '',
  FDS5_17: '',
  EIND_8: '',
  ED2_7: '',
  PQ_1: '',
  ED2_2: '',
  ED2_9: '',
  ED2_8: '',
  EDM_16: '',
  ECR_BR_1: '',
  ECR_BR_2: '',
  ED4_2: '',
  EDM_18: '',
  FDS4_2: '',
  BIC_LO_1: '',
  APL_4: '',
  APL_1: '',
  FDP5_2: '',
  APL_3: '',
  APL_2: '',
```

FDP2_9: ",
FDP2_8: ",
FDP2_7: ",
FDP2_2: ",
OP_3: ",
FDS2_2: ",
FCU_2: ",
FDS2_7: ",
FDS2_8: ",
FDS3_11: ",
FDS3_10: ",
EIND_2: ",
EIND_7: ",
FCR_1: ",
FCR_2: ",
FDP3_12: ",
FPB_4: ",
FPB_5: ",
FPB_3: ",
FDS5_2: ",
PT_1: ",
FPL_6: ",
FPL_4: ",
FPL_5: ",
FPL_2: ",
FPL_3: ",
FDP5_17: ",
APB_5: ",
APB_4: ",
APB_3: ",
FDP4_2: ",
ED4_15: ",
ED4_14: ",
FDP3_2: ",
ED4_13: ",
ADPT_4: ",
ADPT_5: ",
ADPT_3: ",
FDP5_18: ",
FDP5_16: ",
FDS3_2: ",
OP_4: ",
FDS5_16: ",
PL_1: ",

```
  PL_2: '',
  FDS4_13: '',
  FDS4_14: '',
  FDP4_13: '',
  FDP4_15: '',
  FDP4_14: '',
  EPB_3: '',
  EPB_2: '',
  FCU_1: '',
  FDP3_10: '',
  FDP3_11: '',
  EPB_5: '',
  EPB_4: '',
  BIC_SH_1: '',
  ECU_1: '',
  ECU_2: '',
  ECR_LO_2: '',
  ECR_LO_1: '',
  EPL_1: '',
  EPL_3: '',
  EPL_2: '',
  EPL_5: '',
  EPL_4: '',
  ED5_18: '',
  EPL_6: '',
  ED5_16: '',
  ED5_17: '',
  ED3_2: '',
  SUP_1: '',
  ED5_2: '',
  EDM_17: '',
  ED3_10: '',
  ED3_11: '',
  ED3_12: '',
}
const len = {
  BIC_LO: '',
  BIC_SH: '',
  SUP: '',
  PT: '',
  PQ: '',
  ECR_LO: '',
  ECR_BR: '',
  ECU: '',
```

```
  FCR: '',
  FCU: '',
  PL: '',
  FDS5: '',
  FDS4: '',
  FDS3: '',
  FDS2: '',
  FDP5: '',
  FDP4: '',
  FDP3: '',
  FDP2: '',
  EDM: '',
  ED5: '',
  ED4: '',
  ED3: '',
  ED2: '',
  EIND: '',
  EPL: '',
  EPB: '',
  FPB: '',
  FPL: '',
  APL: '',
  OP: '',
  APB: '',
  ADPT: '',
};
export const obj2 = {
  predicted_len: len,
  predicted_momarm: momarm,
  error_len: len,
  error_momarm: momarm,
  real_len: len,
  real_momarm: momarm
};
export const example1 = {"ra_wr_s_p": -0.9372269511, "ra_wr_e_f":
0.5132192373, "ra_cmc1_f_e": 0.7192248106, "ra_cmc1_ad_ab":
0.7956714034000001, "ra_mcp1_f_e": -0.269611299, "ra_ip1_f_e": -0.1642750502,
"ra_mcp2_e_f": 1.5415153503, "ra_pip2_e_f": 0.531974256, "ra_dip2_e_f":
0.7157563567, "ra_mcp3_e_f": 0.156952545, "ra_pip3_e_f": 1.8087017536,
"ra_dip3_e_f": 1.0278061628, "ra_mcp4_e_f": 0.8518201709000001, "ra_pip4_e_f":
0.5344702601, "ra_dip4_e_f": 0.6400498152, "ra_mcp5_e_f": 0.930750072,
"ra_pip5_e_f": 0.4880279899, "ra_dip5_e_f": 0.2547964752}
export const example2 = {"ra_wr_s_p": 0.5645097494, "ra_wr_e_f":
0.0695229769, "ra_cmc1_f_e": 0.3191470802, "ra_cmc1_ad_ab": 0.504709363,
```

"ra_mcp1_f_e": -0.7150073647, "ra_ip1_f_e": -0.5428017378, "ra_mcp2_e_f": 0.8898081183, "ra_pip2_e_f": 1.0090668201000002, "ra_dip2_e_f": 0.0854025185, "ra_mcp3_e_f": 0.7386418581, "ra_pip3_e_f": 1.7423783541, "ra_dip3_e_f": 0.5762346983, "ra_mcp4_e_f": 0.5544494986999999, "ra_pip4_e_f": 1.3775169849000002, "ra_dip4_e_f": 1.2807859182, "ra_mcp5_e_f": 1.2117609978, "ra_pip5_e_f": 1.8010421991, "ra_dip5_e_f": 0.3536667228}

```
export const example3 = {"ra_wr_s_p": 1.4600178003, "ra_wr_e_f": 1.0450892448, "ra_cmc1_f_e": 0.5845512152000001, "ra_cmc1_ad_ab": 0.3118136823, "ra_mcp1_f_e": -0.4342938364, "ra_ip1_f_e": -1.1401605606, "ra_mcp2_e_f": 1.1387401819, "ra_pip2_e_f": 1.3704600334, "ra_dip2_e_f": 1.0820456743, "ra_mcp3_e_f": 1.1913590431, "ra_pip3_e_f": 1.7404507399, "ra_dip3_e_f": 1.5157710314, "ra_mcp4_e_f": 0.9871973395, "ra_pip4_e_f": 1.2374255657, "ra_dip4_e_f": 0.5290784239999999, "ra_mcp5_e_f": 0.9532599449, "ra_pip5_e_f": 1.3495132923, "ra_dip5_e_f": 0.4378182888}
export const example4 = {"ra_wr_s_p": 1.3410359621, "ra_wr_e_f": 0.0482894182, "ra_cmc1_f_e": 0.8661386967, "ra_cmc1_ad_ab": 0.1254002899, "ra_mcp1_f_e": -0.1056627035, "ra_ip1_f_e": -0.6467231512, "ra_mcp2_e_f": 0.2794427276, "ra_pip2_e_f": 0.2429862469, "ra_dip2_e_f": 0.9790592194, "ra_mcp3_e_f": 1.0086711645, "ra_pip3_e_f": 1.9122941494, "ra_dip3_e_f": 0.4065728188, "ra_mcp4_e_f": 0.2257127762, "ra_pip4_e_f": 2.081248045, "ra_dip4_e_f": 0.4184978604, "ra_mcp5_e_f": 1.1738936901, "ra_pip5_e_f": 1.5529900789, "ra_dip5_e_f": 0.3362826407}
//colours
$light-light-green: #DDE9E6;
$light-green: #A9D4CA;
$green: #50766D;
$dark-green: #32433F;
$white: #FFF;
.ddiv {
  background-color: yellow;
}
import React, { Component } from "react";
import styles from './app.module.scss';
import Userpage from "../../pages/Userpage";
import { Provider } from 'react-redux';
class App extends Component {
  render() {
        return <Userpage />;
  }
}
export default App;
@import "../../assets/variables";

.footer {
```

```
    background-color: $white;
    height: 7%;
    position: absolute;
    width: 100%;
    &Text {
      font-size: 1.5em;
      margin: 0.5em 1.2em 0;
      color: $light-green;
    }
    &::after {
      content: "";
      display: block;
      width: 90%;
      height: 2px;
      position: absolute;
      bottom: 96%;
      right: 0;
      background-color: $light-green;
    }
  }
  import React from "react";
  import styles from "./footer.module.scss"

  const Footer = () => (
    <footer className={styles.footer}>
          <p className={styles.footerText}>Created by Denys Smirnov &copy;
2020</p>
    </footer>
  );

  export default Footer;
  @import "../../../assets/variables";
  .headerButton {
    display: inline-block;
    font-size: 2em;
    margin: 0.8em 1em 0 1em;
    & > a {
      text-decoration: none;
      color: $green;
      &::after {
        content: "";
        display: block;
        width: 1px;
        margin: auto;
```

```
      height: 2px;
      top: 97%;
      background-color: $green;
      opacity: 0;
      transition: all 1s;
     }
     &:hover{
      &::after {
       opacity: 1;
       width: 100%;
      }
     }

  }
}
import React from "react";
import PropTypes from 'prop-types';
import styles from './headerButton.module.scss'
import {Link} from "react-router-dom";

const HeaderButton = ({
  text,
  link,
}) => {
  return (
        <div className={styles.headerButton}>
              {}
              <Link to={link}>
                    <div>
                          {text}
                    </div>
              </Link>
        </div>

  )
};

HeaderButton.propTypes = {
  text: PropTypes.string,
  link: PropTypes.string,
};
HeaderButton.defaultProps = {
  text: '',
  link: '',
```

```
};
export default HeaderButton;
@import "../../assets/variables";

.header {
  position: absolute;
  height: 5em;
  background-color: $light-green;
  width: 100%;
  top: 0;

  &::after {
    content: "";
    display: block;
    width: 90%;
    height: 2px;
    position: absolute;
    top: 97%;
    background-color: $green;
  }
}
import React from "react";
import HeaderButton from "./HeaderButton";
import styles from './header.module.scss'

const Header = () => (
  <header className={styles.header}>
          <HeaderButton link='' text='Home'/>
          <HeaderButton link='/about' text='About'/>
          <HeaderButton link='/interact' text='Interact'/>
  </header>
);

export default Header;
import React, { useMemo } from "react";
import PropTypes from 'prop-types';
import MainText from "../shared/MainText";
import figure3a from "../../../public/images/figure3a.png"
import figure3b from "../../../public/images/figure3b.png"
import figure4a from "../../../public/images/figure4a.png"
import figure4b from "../../../public/images/figure4b.png"
import styles from "./infoBlock.module.scss"
import Page from '../../../public/pages/figure3a_ua.html';
var htmlDoc = {__html: Page};
```

```
import {Redirect} from "react-router";
const figures = {
  figure3: [figure3a, figure3b],
  figure4: [figure4a, figure4b],
};

const InfoBlock = ({
  text,
  type,
  name,
}) => {
  return (
        <div className={styles.infoBlock}>
              <MainText type={type} text={text} classes={styles.aboutText}/>
              <div className={styles.imageBlock}>
                    {figures[name].map((item, i) => <img onClick={() =>
window.open(name + (i === 0 ? 'a' : 'b'), "_blank")} className={styles.image}
src={item} alt="" />)}
              </div>
        </div>
  )
};

InfoBlock.propTypes = {
  text: PropTypes.string.isRequired,
  type: PropTypes.string,
  name: PropTypes.string,
};

InfoBlock.defaultProps = {
  type: 'center',
  name: '',
};

export default InfoBlock;
@import "src/assets/variables";
.image {
  width: 40%;
  height: auto;
}
.infoBlock {
  align-items: center;
  justify-content: space-between;
  width: 100%;
```

```
  margin: auto;

 }
 .infoBlock:before {
  content: "";
  display: block;
  width: 95%;
  height: 2px;
  background-color: $dark-green;
  margin-bottom: 1em;
 }
 .aboutText {
  font-size: 1.2em;
  margin-bottom: 2em;
 }
 .imageBlock {
  display: flex;
  justify-content: space-around;
  margin-bottom: 2em;
 }@import "../../../assets/variables";

 .button {
  cursor: pointer;
  padding: 10px;
  border-radius: 20px;
  text-decoration: none;
  width: 10em;
  display: inline-block;
  box-shadow: 0 6px 28px rgba(0,0,0,0.25), 0 5px 10px rgba(0,0,0,0.22);

  & > div, a {
   text-align: center;
   text-decoration: none;
   font-size: 2em;
  }
  &:hover {
   box-shadow: 0 6px 28px rgba(0,0,0,0.3), 0 5px 20px rgba(0,0,0,0.22);
   transition: .2s;
  }
 }
 .light {
  background-color: rgba(169, 212, 202, 0.78);
  & > div, a {
   color: $green;
```

```
  }
 }
 .dark {
  background-color: rgba(80, 118, 109, 0.8);
  & > div, a {
   color: $light-green;
  }
 }
 import React from "react";
 import PropTypes from 'prop-types';
 import styles from './button.module.scss'
 import classnames from "classnames";
 import {Link} from "react-router-dom";

 const Button = ({
  text,
  onClick,
  light,
  classes,
  link,
 }) => {
  return (
        <div onClick={onClick} className={classnames(styles.button, classes,
light ? styles.light : styles.dark)}>
            {link ?
            <Link to={link}>
                <div>
                    {text}
                </div>
            </Link> :
            <div>{text}</div>}
        </div>
  )
 };
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

«Програмний комплекс для розв’язування біомеханіки опорно-рухового апарату за допомогою машинного навчання»

Технічне завдання

КПІ.ІП-6102.045490.03.81

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Є.А. Недашківський

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ Д.С. Смірнов

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: «Програмний комплекс для розв’язування біомеханіки опорно-рухового апарату за допомогою машинного навчання».

Галузь застосування: системи обробки сигналів протезів та ортезів.

Наведене технічне завдання поширюється на розробку «Програмний комплекс для розв’язування біомеханіки опорно-рухового апарату» КПІ.ІП-6101.045490, котра використовується для систем обробки інформації в протезах та ортезах, та призначена для використання в їх розробці.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки «Програмний комплекс для розв’язування біомеханіки опорно-рухового апарату» є завдання на дипломне проєктування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в системах обробки сигналів в протезах та ортезах.

Метою розробки є створення програмного комплексу для розробки, тренування та аналізу моделей машинного навчання.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання перетворення постави верхньої кінцівки із 18 ступенів свободи в м’язові характеристики. Розробку виконати на платформі Linux.

4.2 Вимоги до надійності

Програмне забезпечення повинно передбачити контроль введення інформації та захист від некоректних дій користувача.

4.3 Умови експлуатації

Програмне забезпечення повинно відповідати умовам експлуатації згідно СанПін 2.2.2.542 – 96. Розроблюване програмне забезпечення функціонує в автоматичному режимі та не вимагає обслуговування.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах. Мінімальна конфігурація технічних засобів:

- центральний процесор: 1.4 GHz Quad-Core 8-th generation Intel Core i5;
- оперативна пам’ять: 8 GB;
- постійна пам’ять: 256 GB.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Unix. Вхідні дані повинні бути представлені в наступному форматі: масив чисел з плаваючою точкою розмірністю [18, 1], де кожне значення знаходиться в діапазоні згідно з діапазону ступеней свободи. Результати повинні бути представлені в наступному форматі: Шість масивів: три масиви розмірністю [33,1], що відповідають за довжини м’язів, та три

масиви розмірінстю [99, 1], що відповідать за значення моментів плеча. Перший масив в кожному типі вказує на зпрогнозоване значення, другий - на реальне (тобто те, що повинне бути), третій – нормалізовану середньо-квадратичну похибку між попередніми масивами. Програмне забезпечення повинно бути розроблено на мові програмування Python з використанням фреймворків Tensorflow, Scikit-Learn та Flask в IDE PyCharm.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи:

- Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6);
- Технічне завдання;
- Керівництво користувача;
- Програма та методика тестування;
- Графічна частина повинна бути виконана на 3 листах формату А3, котрі включаються у якості додатків до пояснювальної записки: «Структурна схема класів програмного забезпечення», «Діаграма розгортання програмного забезпечення» та «Структура нейронної мережі»

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

	Назва етапу	Строк	Звітність
1	Вивчення літератури за тематикою проєкту	06.04.2020	
2	Розробка технічного завдання	10.04.2020	Технічне завдання
3	Аналіз вимог та уточнення специфікацій	13.04.2020	Специфікації програмного забезпечення
4	Проєктування структури програмного забезпечення, проєктування компонентів	16.04.2020	Схема структурна програмного забезпечення та специфікація компонентів
5	Програмна реалізація програмного забезпечення	22.04.2020	Вихідні тексти програмного забезпечення
6	Тестування програмного забезпечення	29.04.2020	Тести, результати тестування
7	Розробка матеріалів текстової частини проєкту	08.05.2020	Пояснювальна записка.
8	Розробка матеріалів графічної частини проєкту	21.05.2020	Графічний матеріал проєкту
9	Оформлення технічної документації проєкту	25.05.2020	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
“___” _____ 2020 р.

«Програмний комплекс для розв’язування біомеханіки опорно-рухового апарату за допомогою машинного навчання»

Програма та методика тестування

КПІ.ІП-6101.045490.04.51

“ПОГОДЖЕНО”
Керівник проекту:
_____ Є.А. Недашківський

Нормоконтроль:
_____ К.І. Ліщук

Виконавець:
_____Д.С. Смірнов

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Web-додаток для аналізу та взаємодії з моделями машинного навчання, що навчені розв’язувати біомеханіку опорно-рухового апарату, створений за допомогою фрейморку Flask на мові програмування Python та бібліотеки ReactJS мови JavaScript.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

– функціональна працездатність елементів сторінок web-ресурсу;

– функціональна працездатність моделей машинного навчання;

– функціональна працездатність алгоритмів пре- та постобробки;

– відповідність форматів та протоколів пересилання даних з веб-сервером;

– забезпечення належного рівня безпеки даних;

– зручність роботи з web-сайтом;

– відповідність дизайну вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування відбуватиметься у режимі чорної скриньки. Для надійності функціонал буде протестований як при позитивних, так і при негативних умовах. Перед початком тестування нової версії програмного продукту необхідно обов’язково провести димне тестування API сервера по обробці значень типу float та димне тестування інтерфейсу користувача.

Тестування відбувається на рівні «ітеграціного тестування». Використовуються наступні методи:

– функціональне тестування, зокрема на рівні Criticalpathtest (базове тестування);

– тестування продуктивності програмного забезпечення, зокрема Stabilitytesting (тестування стабільності) та Loadtesting (навантажувальне тестування);

– тестування інтерфейсу;

– тестування API;

– функціональне тестування коректності моделей біомеханіки опорно-рухового апарату.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію Postman.

Працездатність web-ресурсу перевіряється шляхом:

– динамічного ручного тестування – введенням значень з можливого діапазону та некорректних значень в поля, які можна редагувати;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування web-ресурсу в різних web-браузерах;

– тестування при максимальному навантаженні;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
“___” _____ 2020 р.

«Програмний комплекс для розв'язування біомеханіки опорно-рухового апарату за допомогою машинного навчання»

Керівництво користувача

КПІ.ІП-6101.045490.05.34

“ПОГОДЖЕНО”
Керівник проекту:
_____Є.А. Недашківський

Нормоконтроль:
_____ К.І. Ліщук

Виконавець:
_____Д.С. Смірнов

Київ – 2020 року

ЗМІСТ

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Вхід на сайт

Перед початком роботи з програмним продуктом користувач повинен завантажити стартову сторінку веб-додаток (Рисунок 1.1).

Рисунок 1.1 – Стартова сторінка

1.2 Введення значень

Після завантаження стартової сторінку користувачеві необхідно перейти на сторінку «Interact». На ній, для того, щоб провести дослідження, треба вибрати тип моделі, кількість тренувальних даних та ввести значення 18 ступенів свободи або ж завантажити їх з готових прикладів. (Рисунок 1.2).

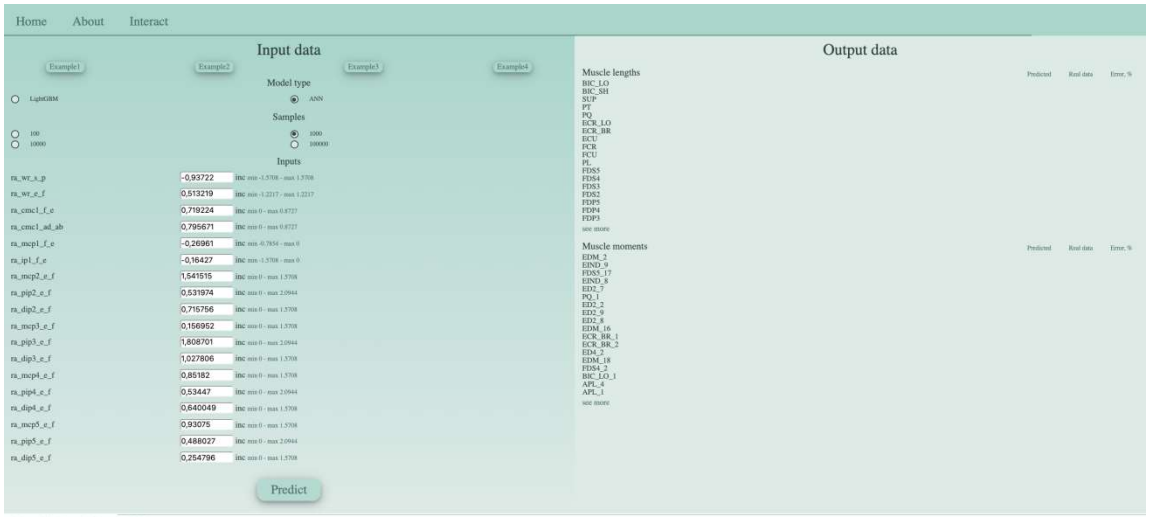


Рисунок 1.2 – Введення значень

1.3 Обробка зображення

Після коректного введення усіх значень, сервер відправить їх на бек-енд частину для обробки. Через декілька секунд користувачеві буде відображено результат роботи програмного забезпечення (рисунок 1.3).

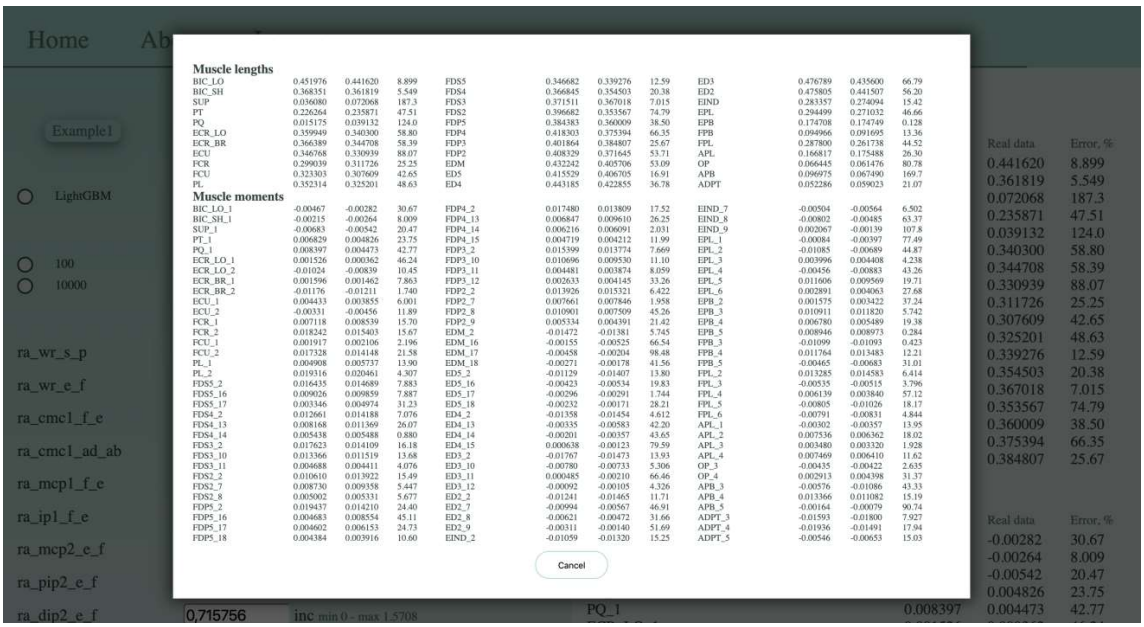
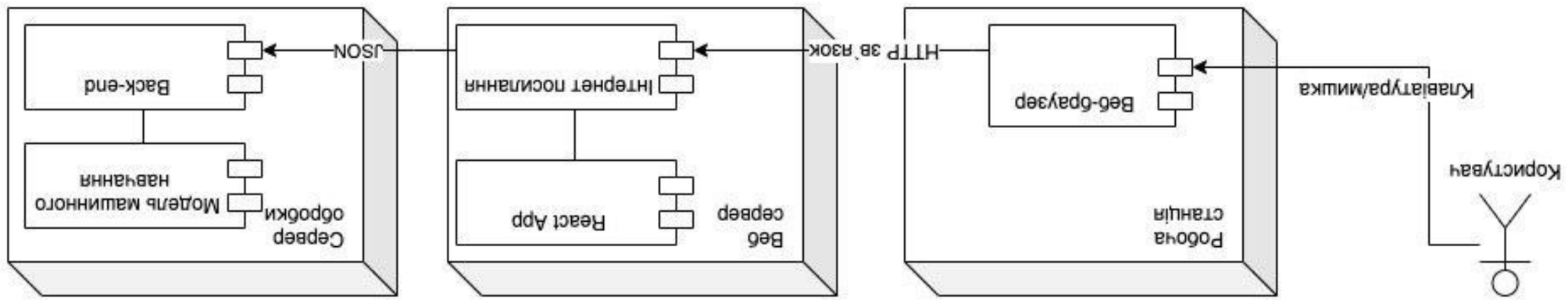
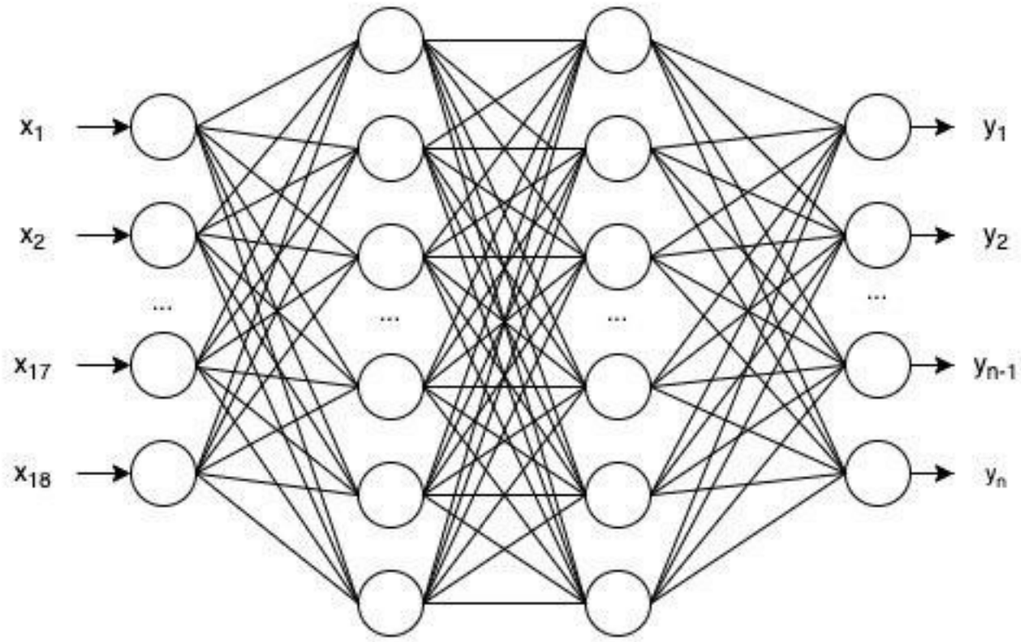


Рисунок 1.3 – Результат роботи програмного додатку



								КПІ.ІП-6101.045490.06.99 СС	
Зм.	Арк.	№ докум.	Підпис	Дата					
Розробив		Сидоров Д.С.				Діаграма розгортання програмного забезпечення	Літ.	Маса	Масш.
Перевірив		Недашківський Є.А.							
Т. Контр.							Аркуш 1		Аркушів 1
Н. Контр.		Лищук К.І.				Програмний комплекс для розв'язання біомеханіки опорно-рухового апарату	КПІ ім. Ігоря Сікорського Кафедра АСОІУ Група ІП-61		
Затверд.									

Структура нейронної мережі

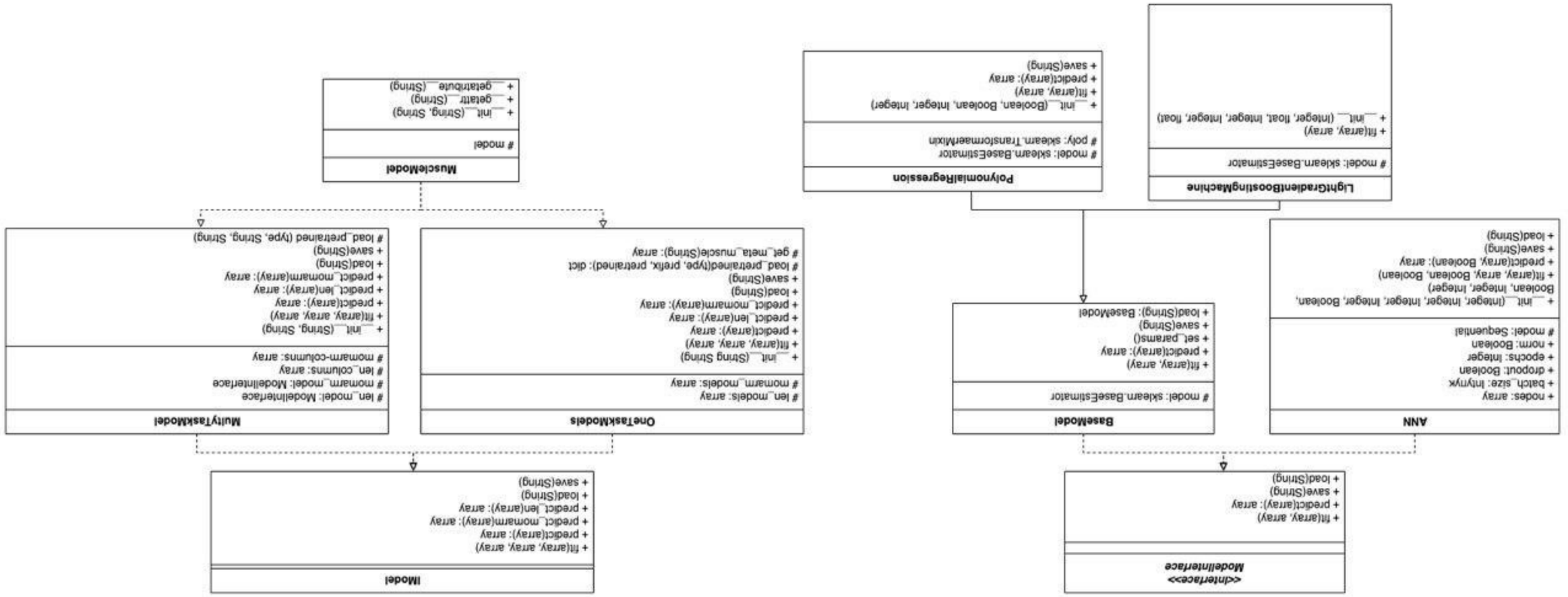


Демонстраційний плакат до дипломного проєкту

Програмний комплекс для розв'язування біомеханіки опорно-рухового апарату

Виконав студент гр. ІП-61 Смірнов Д.С.

Керівник Недашківський Є.А.



КП.ІП-6101.045490.06.99 СС

Зм.	Арк.	№ докум.	Підпис	Дата	КПІ.ІП-6101.045490.06.99 СС		
Розробив		Сидоров Д.С.			Схема структура класів програмного забезпечення	Літ.	Маса	Масш.
Перевірив		Недашків-ський Є.А.						
Т. Контр.						Аркуш 1	Аркушів 1	
Н. Контр.		Лищук К.І.			Програмний комплекс для розв'язання біомеханіки опорно-рухового апарату	КПІ ім. Ігоря Сікорського Кафедра АСОІУ Група ІП-61		
Затверд.								

ВІДГУК
керівника дипломного проєкту
на здобуття ступеня бакалавра

виконаний на тему: «Програмний комплекс для розв'язування біомеханіки опорно-рухового апарту за допомогою машинного навчання» _____
студентом Смірновим Денисом Сергійовичем_____
(прізвище, ім'я, по батькові)

Бакалаврська дипломна робота присвячена розробці програмного забезпечення для тренування та аналізу моделей машинного навчання для розв'язування частини задачі біомеханіки опорно-рухового апарату.

Програмне забезпечення створене для того, щоб спростити подальший розвиток розробки протезу верхної кінцівки. За допомогою єдиного інтерфейсу для моделей та зручної послідовності блоків, програмний комплекс здатний навчати велику кількість різноманітних регресійних моделей машинного навчання, підбирати гіперпараметри, та аналізувати результати їх роботи: вимірювати швидкість обробки інформації алгоритмів, порівнювати їх якість та аналізувати кількість даних, потрібних для тренування. В програмний комплекс також включений веб-додаток, за допомогою якого можна наочно продивитися результати експериментів, вивчити онлайн графіки та безпосередньо взаємодіяти з натренованими моделями.

Студент Смірнов Денис у своїй роботі розбив програмне забезпечення на чотири блоки. Блок генерації даних відповідальний за створення тренувальних, валідаційних та тестових даних. Блоки оптимізатору гіперпараметрів та моделей машинного навчання забезпечують створення, навчання та тестування алгоритмів. Останній блок дозволяє розробнику або користувачеві продивитися результати навчених моделей. Для демонстрації роботи програмного комплексу студент використав два абсолютно різні методи машинного навчання: легкий метод градієнтного підсилювання та повнозв'язну нейронну мережу; та створив зручний веб-застосунок для їх демонтрації та взаємодії з ними. Для оцінки точності моделей машинного навчання використовувалась нормалізована середньо-квадратична похибка.

Під час виконання дипломної роботи студент Смірнов Денис Сергійович показав себе як відмінний спеціаліст-інженер, який здатний розв’язувати складні та нетривіальні задачі, на основі теорії реалізовувати та імплементувати складні алгоритми, а також продемонстрував глибокі знання в професійному спрямуванні.

В цілому дипломний проєкт відповідає вимогам університету до дипломних проєктів. Вважаю, що студент Смірнов Денис Сергійович заслуговує присудження ступеня бакалавра зі спеціальності «121 Інженерія програмного забезпечення» та кваліфікації бакалавр з інженерії програмного забезпечення.

Керівник дипломного проєкту

ст. викл		Недашківський Є.А.
(посада, науковий ступінь, вчене звання)	(підпис)	(ініціали, прізвище)

РЕЦЕНЗІЯ
на дипломний проєкт
на здобуття ступеня бакалавра

виконаний на тему: «Програмний комплекс для розв'язування біомеханіки опорно-рухового апарту за допомогою машинного навчання» _____
студентом Смірновим Денисом Сергійовичем_____
(прізвище, ім'я, по батькові)

Робота Смірнова Д.С. присвячена актуальній на сьогодні тематиці розробки алгоритмів машинного навчання для управління протезами та ортезіами. Зміст та сутність роботи, наданої на рецензію, відповідають темі. Всі пункти завдання розкриті у дипломному проєкті в достатньому обсязі. Проєкт виконаний якісно, відповідно до вимог КПІ ім. Ігоря Сікорського.

У вступі студент доводить актуальність обраної проблеми, окреслює мету роботи та формулює задачі проєкту. Розглянуті необхідні відомостіз методів машинного навчання. Робиться досить детальний огляд відомих технічних рішень у сфері протезування, наводяться наукові роботи відомих дослідників та вчених, які займалися проблемами біомеханіки опорно-рухового апарату за допомогою машинного навчання,та описується отримані ними результати.

Мета роботи полягає в розробці програмного комплексу для розв'язування частини задачі біомеханіки опорно-рухового апарату за допомогою машинного навчання. Це необхідний етап розробки програмного забезпечення для протезування верхньої кінцівки людини. Після отримання даних електроміограми реальних людей цей програмний комплекс буде використований в програмній частині протезу чи ортезу. У своїй роботі студент розробляє дві моделі машинного навчання, які мають різну архітектуру та логіку, та демонструє, як вони можуть вирішити проблему перетворення постави верхньої кінцівки в м'язові характеристики: довжину та моменти плеча м'язів.

В роботі був розроблений програмний комлекс, який складається з чотирьох частин: блоку генерації даних, блоку оптимізатору гіперпараметрів, блоку моделей машинного навчання та блоку аналізу. Також був створений

зручний веб-додаток, за допомогою якого можно наочно проаналізувати онлайн графіки результатів створених алгоритмів та взаємодіяти з навченими моделями в реальному часі.

У своїй роботі студент зосереджує увагу на аналізі якості та швидкості обробки інформації навчених моделей машинного навчання, а також на залежності точності алгоритмів від кількості тренувальних даних. Порівняння точності моделей було здійснено за допомогою нормалізованої середно-квадратичної похибки. Прийняті під час виконання проєкту рішення та висунуті положення повністю підтверджені експериментально, отримані практичні результати є оригінальними і обґрунтованими.

Наданий на рецензію дипломний проєкт Смірнова Дениса Сергійовича свідчить про його обізнаність у галузі розробки програмного забезпечення з використання алгоритмів машинного навчання, а саме алгоритмів глибинного навчання та алгоритмів, заснованих на ансамблюванні дерев рішень, глибоке розуміння предметної області, знання проблем у цій галузі та способів їх вирішення.

Викладені в дипломному проєкті положення містять наукову новизну в галузі розвитку принципів побудови математичних моделей розрахунку параметрів м’язів методами машинного навчання. Результати роботи (програмний комплекс та веб-додаток) становлять практичну цінність та можуть бути впроваджені в практику розробки систем машинного навчання для вирішення задач керування протезами та ортезами.

В цілому дипломний проєкт відповідає вимогам університету до дипломних проєктів. Вважаю, що студент Смірнов Денис Сергійович заслуговує присудження ступеня бакалавра зі спеціальності «121 Інженерія програмного забезпечення» та кваліфікації бакалавр з інженерії програмного забезпечення.

Рецензент

доцент каф. електронної інженерії,

к.т.н., доцент _____ _____ А.О. Попов

(посада, науковий ступінь, вчене звання) (підпис) (ініціали, прізвище)