НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва інституту/факультету)
кафедра БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва кафедри)

«На правах рукопису»
УДК 615-84

«До захисту допущено»
В. о. завідувача кафедри БМІ
_____ О. В. Лебедєв
(підпис) (ініціали, прізвище)
“ ____ ” _____ 2018 р.

Магістерська дисертація
на здобуття ступеня магістра

зі спеціальності 163 «Біомедична інженерія»
(код і назва)

на тему: Алгоритм покращення результатів аналізу епілептичних сигналів ЕЕГ методами машинного навчання

Виконала: студентка 6-го курсу, групи БМ-71мп
Реп’ях Оксана Володимирівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент каф. БМІ, к.т.н. Дубко Андрій Григорович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант ст. вик. каф. БМІ Білошицька Оксана Костянтинівна _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доц. каф. БМК, к.т.н. Носовець Олена Костянтинівна _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Нормоконтролер інженер 1 категорії Андреєв Петро Іванович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ – 2018

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва інституту/факультету)
кафедра БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва кафедри)

Рівень вищої освіти – другий (магістерський)
Спеціальність 163 «Біомедична інженерія»
(код і назва)

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри БМІ
_____ О. В. Лебедєв
(підпис) (ініціали, прізвище)
“ ____ ” _____ 2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Реп’ях Оксані Володимирівні
(прізвище, ім’я, по батькові)

1. Тема дисертації: Алгоритм покращення результатів аналізу епілептичних сигналів ЕЕГ методами машинного навчання
науковий керівник дисертації к.т.н., доцент каф. БМІ Дубко Андрій Григорович
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від

2. Термін подання студентом дисертації: 12 грудня 2018 року.

3. Об’єктом дослідження є сигнали головного мозку (електроенцефалограма).

4. Предмет дослідження: методи машинного навчання.

5. Перелік завдань, які потрібно розробити:
- Визначити основні напрямки досліджень штучного інтелекту з використанням електроенцефалограми.
- Дослідити переваги та недоліки методів аналізу.
- Провести попередню обробку сирих даних та проаналізувати їх.
- Виділити найефективніший набір характеристик сигналів.
- На основі платформи програмування Python 2.7.15 побудувати модель класифікації.
- Провести дослідження з прогнозування епілепсії за допомогою методів машинного навчання.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: 40 рисунків, 6 таблиць, 2 додатки.

7. Орієнтовний перелік публікацій: 2 публікації враховуючи тези та статті.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
3	ст. вик. каф. БМІ Білошицька О. К.		

9. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Визначення теми наукового пошуку	Вересень 2017	
2.	Аналіз літератури з даного напрямку	Жовтень 2017	
3.	Огляд розвитку досліджень штучного інтелекту	Листопад 2017	
4.	Формування задач згідно теми наукового пошуку	Грудень 2017	
5.	Розробка теоретичної частини дисертації	Грудень 2017 – Січень 2018	
6.	Проведення дослідження з аналізу даних	Лютий – Квітень 2018	
7.	Розробка практичної частини та отримання результатів	Травень 2018	
8.	Аналіз отриманих результатів	Червень 2018	
9.	Програмування моделі автоматичного виявлення патологічних частотних ритмів в сигналі	Липень – Жовтень 2018	
10.	Подання дисертації для рецензування	Грудень 2018	
11.	Подання дисертації на захист	Грудень 2018	
12.	Підготовка до захисту	Грудень 2018	
13.	Захист магістерської дисертації	Грудень 2018	

Студент _____ О. В. Реп’ях
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ А. Г. Дубко
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Обсяг магістерської дисертації становить 85 сторінок, містить 41 ілюстрацію, 6 таблиць та 2 додатки. Загалом було опрацьовано 44 джерела.

Робота присвячена розробці програмного алгоритму для автоматичного розпізнавання та прогнозування епілетриформних частотних ритмів в сигналах ЕЕГ, за допомогою методів машинного навчання.

Метою є створення програмної моделі для автоматичного розпізнавання та прогнозування епілетриформних частотних ритмів в сигналах ЕЕГ, за допомогою методів машинного навчання.

Об’єктом дослідження є сигнали електроенцефалограми.

Предметом дослідження виступають методи машинного навчання.

У магістерській дисертації визначені основні напрямки досліджень штучного інтелекту, з використанням сигналів електроенцефалограми; досліджені переваги та недоліки методів аналізу; проведена попередня обробка сирих даних та сформовані набори вхідних даних; відфільтровано найефективніший та найінформативніший набір ознак; на основі платформи програмування Python 2.7.15 побудовано модель класифікації сигналів ЕЕГ; проведені дослідження з прогнозування епілепсії за допомогою методів машинного навчання.

За результатами роботи опубліковано: стаття «Classification of epileptiform activity in EEG using machine learning techniques» у науковому журналі «Science, Research, Development» (червень 2018 року); тези «Розпізнавання епілептичної активності в сигналах ЕЕГ за допомогою методів машинного навчання» у науково-практичному журналі «Інформаційні системи та технології в медицині» ISM-2018 (листопад 2018 року).

Ключові слова: ЕЕГ, машинне навчання, штучний інтелект, наука про дані, метод опорних векторів, наївний Баєсів класифікатор, Градієнтний бустинг над вирішальними деревами.

ABSTRACT

The subject of the undergraduate practice is «Classification of epileptiform EEG using machine learning techniques».

The volume of the report is 85 pages, 42 figures, 6 tables, 7 formulas, two applications are included. In total 47 references were analyzed.

Epilepsy is the fourth most common neurological problem in the world. When diagnosing epilepsy, the most informative is the registration of EEG, which helps distinguish epileptic seizures from non-epileptic seizures and classify them.

Aim: EEG signal classification model based on machine learning methods.

In the master's dissertation were determined the basic directions of research of artificial intelligence, using signals of an electroencephalogram. Were investigated the advantages and disadvantages of the analysis methods. Preprocessing of raw data have been done and formed input datasets. The most effective and informative set of features filtered out. A model of the classification of EEG signals was constructed using the programming platform Python 2.7.15. Researches have been conducted on the prediction of epilepsy with by the machine learning methods.

The article «Classification of epileptiform activity in EEG using machine learning techniques» was published in the journal «Science, Research, Development» (June 2018) and thesis «Recognition of epileptic activity in EEG signals using machine learning methods» was published in the journal «Information systems and technologies in medicine ISM–2018» (November 2018) based on research results.

Key words: EEG, machine learning, artificial intelligence, data science, Support Vector Machine, Gaussian Naïve Bayes, Gradient Boosting Classifier

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

ЕЕГ – Електроенцефалографія

BCI – Нейрокомп’ютерний Інтерфейс (Brain-Computer Interface)

MI – Рухома/Моторна Уява (Motor Imagery)

MSPCA – Багатомасштабний Аналіз Головних Компонент (Multiscale Principle Component Analysis)

WPD – Вейвлет Перетворення (Wavelet Packet Decomposition)

SVM – Метод Опорних Векторів (Support Vector Machine)

LDA – Лінійний Дискримінантний Аналіз (Linear Discriminant Analysis)

ELM – Екстремальне Навчання (Extreme Learning Machine)

MLP – Багатошаровий Персептрон (Multi-Layer Perceptron)

BPN – Метод Зворотного Поширення Помилки (Back Propagation Network)

RF – Метод Випадкового Лісу (Random Forest)

PAC – Аналіз Головних Компонент (Principle Component Analysis)

GNB – Наївний Баєсів Класифікатор (Gaussian Naïve Bayes)

STFT – Короткочасне (Віконне) Перетворення Фур’є (Short Time Fourier Transform)

CNN – Згорткова Нейронна Мережа (Convolution Neural Network)

DT – Дерево прийняття рішень (Decision Trees)

ВСТУП

Епілепсія – хронічне захворювання головного мозку, яке вражає людей різного віку. Приблизно 50 мільйонів людей в усьому світі страждають на епілепсію, що робить його другим найбільш поширеним неврологічним захворюванням після мігрені. [1] Хоча прогноз лікування на сьогодні досить хороший, адже до 30-ти відсотків хворих не мають ремісії, терапія антиепілептичними препаратами має шкідливий вплив на індивідуальне здоров'я, якість життя та є важким навантаження на суспільство [2]. Визначальною характеристикою епілепсії є рецидивні напади, які вражають безконтрольно. Симптоми можуть варіюватися від тимчасової втрати обізнаності або свідомості, порушенням руху, відчуття (у тому числі зір, слух та смак), настрій чи інші когнітивні функції до фізичних проблем (таких як переломи та синяки від травм, пов'язаних з нападами), а також більш високі показники психологічних захворювань, включаючи тривогу та депресію. Значна частина причин смерті, пов'язаних з епілепсією у країнах з низьким і середнім рівнем доходу, такі як падіння, потоплення, опіки та тривалі судоми, потенційно може бути запобіжна. [1, 3]

Виявлення епілептичних порушень активності головного мозку відіграє ключову роль у підвищенні якості життя хворих на епілепсію. Електроенцефалограма (далі – ЕЕГ) – основний сигнал, широко використовуваний для діагностики епілепсії. Візуальний аналіз ЕЕГ є трудомістким та дороговартісним. Крім того, близько 75% людей хворих на епілепсію живуть у країнах з низьким і середнім рівнем доходу та не мають можливості дозволити собі консультації з невропатологами або практикуючими спеціалістами [4]. Ці обмеження спонукають вчених до розробки автоматичних систем виявлення епілептичної патології на основі ЕЕГ.

Темою даного дослідження є «Алгоритм покращення результатів аналізу епілептичних сигналів ЕЕГ методами машинного навчання».

Метою роботи є програмна модель для автоматичного виявлення епілепсії на основі ЕЕГ-сигналів за допомогою методів машинного навчання.

Об’єктом дослідження є сигнали електроенцефалограми.

Предметом дослідження є методи машинного навчання.

Задачі магістерської дисертації:

1) попередня обробка даних (ЕЕГ-сигналів);

2) формування навчальної та тестової вибірки;

3) підбір методів машинного навчання для аналізу біомедичних сигналів;

4) створення програмної моделі на основі обраного методу для автоматичного розпізнавання епілептиформної активності;

5) проведення експериментів з прогнозування епілепсії.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ СИГНАЛІВ ЕЕГ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

1.1 Системи нейрокомп’ютерного інтерфейсу (BCI)

Системи нейрокомп’ютерного інтерфейсу (далі – BCI) можуть поліпшити якість життя людей з обмеженими фізичними можливостями. Вона дозволяє їм виконувати такі завдання, як захоплення об’єктів, включення світла, зміна телевізійних каналів тощо. Фактично, BCI є механізмом виявлення команд мозку і перетворення їх у дію, через процесор.

Робота [1] присвячена розробці ефективної системи BCI (рис. 1.1), в якій сигнали ЕЕГ використовуються як команди мозку. Різні види діяльності можуть спричинити різницю між сигналами ЕЕГ, які в свою чергу можна класифікувати. У дослідженні сигнал посилюється за допомогою методу багатомасштабного аналізу головних компонент (далі – MSPCA). Ознаки з посиленого сигналу витягуються за допомогою пакету вейвлет аналізу (далі – WPD). Розраховані ознаки використовуються для вивчення ефективності різних класифікаторів у класифікації сигналів ЕЕГ, зареєстровані від п’яти різних осіб, в момент уявлення рухів правою ногою та рукою.

Рисунок 1.1 – Етапи розробки системи класифікації для BCI [1]

Для порівняння отриманих результатів використовувалась загальна точність класифікації, значення якої, після групування методів MSPCA, WPD та RF, досягло

98,45%. Даний результат демонструє, що запропонований підхід є потенційним кандидатом для моделювання та розробки майбутніх систем BCI.

Найбільш зручною основою для проектування нейрокомп'ютерного інтерфейсу є сигнали моторних образів (далі – MI), записані за допомогою ЕГГ. Оскільки сигнали MI, як основа BCI, забезпечують високий ступінь свободи, вони допомагають людям з обмеженими руховими можливостями спілкуватися з пристроєм. Але варіабельність показників, розрахунок специфічних для користувача ознак та підвищення точності класифікатора, все ще залишається складним завданням для MI.

У роботі [2] автори пропонують підхід для подолання вищезгаданих проблем. Запропонований підхід включає в себе наступні етапи: метод «трубопроводу» для відбору каналів реєстрації, смуговий фільтр, розрахунок ознак, відбір ефективних ознак за допомогою двох методів та моделювання з використанням наївного Байєсового класифікатора. Оскільки оптимальні ознаки вибираються за допомогою методів відбору, це допомагає подолати їх варіабельність та підвищує продуктивність класифікатора. Автори стверджують що, запропонована методологія (рис. 1.2) ще не була використана для розробки BCI на базі MI.

Рисунок 1.2 – Блок-схема побудови експериментальної моделі [2]

Запропонований підхід був протестований на наборі даних із змагань з нейрокомп'ютерного інтерфейсу (BCI Competition III), організованих групою Berlin

BCI. Результат розробленого методу порівнюється з двома традиційними класифікаторами, такими як лінійний дискримінантний аналіз (далі – LDA) та метод опорних векторів (далі – SVM). Результати підтверджують, що запропонований метод забезпечує поліпшену точність в 95,47%, ніж класифікатори LDA (91,10%) та SVM (92,26%). Автори стверджують, що цей метод може стати основою для подальшого розвитку для розробки надійної та реальної програми BCI на базі MI.

Продуктивність моторних образів, як основи системи нейрокомп'ютерного інтерфейсу, значно залежить від методу розрахунку ознак. Для таких систем вже запропоновано багато алгоритмів фільтрації даних. Наприклад, блок фільтрів (filter bank) дозволяє розрахувати більше число ознак порівнюючи з оригінальними загальними просторовими характеристиками та може бути використаний, як інструмент вирішення проблеми перенавчання.

У дослідженні [3], в якості методу відбору ознак автори використали центральність власного вектора (eigenvector centrality) та вейвлет перетворення, а метод екстремального навчання (далі – ELM) – для покращення продуктивності моторних образів в системах нейрокомп'ютерного інтерфейсу та уникнення перенавчання. Найкраща точність класифікації методами SVM та ELM досягла значення 88,33%. Крім того, обчислювальна швидкість була покращена саме за допомогою методу екстремального навчання, яка склала 0,0016 секунди, тоді як найшвидший результат SVM – 0,0155 секунди.

Все більшої популярності набуває спосіб біометричної автентифікації за допомогою ЕЕГ сигналів. У дослідженні [4] пропонується схема (рис. 1.3) ідентифікації людини з використанням сигналів ЕЕГ, отриманих з недорогих бездротових пристроїв реєстрації ЕЕГ (рис. 1.4).

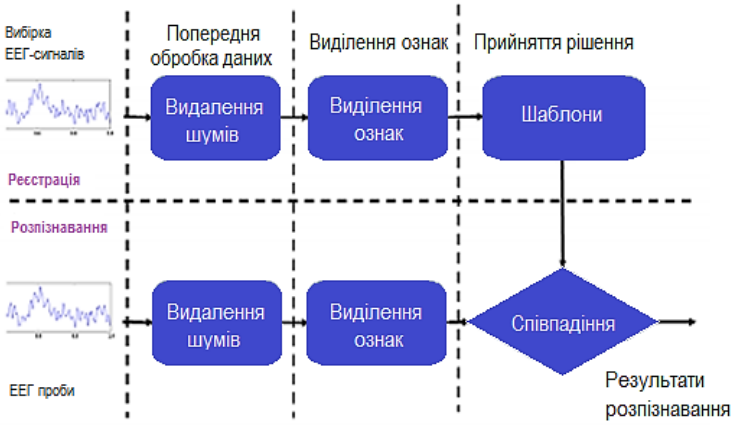

Рисунок 1.3 – Блок-схема системи розпізнавання ЕЕГ-сигналів [4]

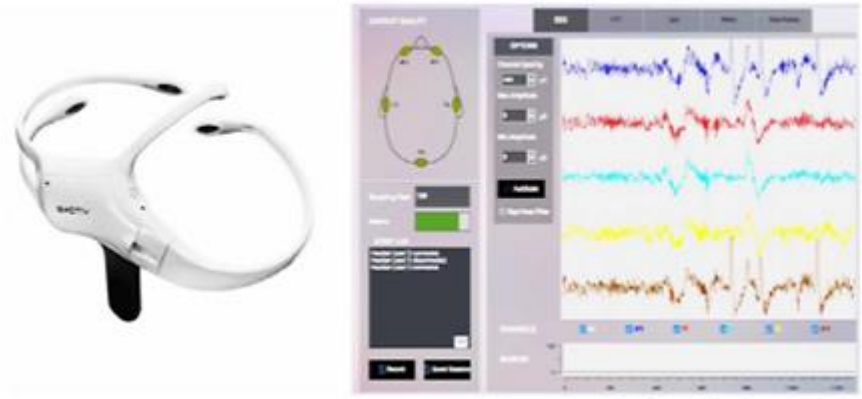

Рисунок 1.4 – Бездротовий пристрій реєстрації ЕЕГ та інтерфейс з зображенням запису сигналу [4]

ЕЕГ-сигнал спочатку проходить попередню обробку для видалення шуму та артефактів за допомогою смугового фільтра зі скінченною імпульсною характеристикою. Потім ці сигнали діляться на непересічні сегменти. На етапі розрахунку ознак застосовуються три способи, а саме: багатомасштабний опис форми (multiscale shape description, MSD), багатомасштабний статистичний вейвлет аналіз (multiscale wavelet packet statistics, WPS) та багатомасштабний статистичний вейвлет аналіз енергетики (multiscale wavelet packet energy statistics, WPES). Далі, ці функції, використовуються для підготовки керованої мультикласової моделі кодування вихідного коду, що коректує помилки (error-correcting output code), з використанням

класифікатора SVM, який в остаточному підсумку може розпізнавати людей з тестових ЕЕГ-сигналів. Попередній експеримент з 9 записами ЕЕГ від 9 суб'єктів показав справжній позитивний рівень запропонованого методу. Найвища точність в 94,44% була отримана на наборі ознак – WPS.

Ідея мозково-машинного інтерфейсу (Brain Machine Interface, BMI) полягає в тому, щоб забезпечити джерело взаємодії між людиною та машиною через думку. У роботі [5] визначено та розглянуто три основні частини ефективного BMI: класифікація думки, здійснення необхідної дії та забезпечення ефективним інтерфейсом користувача. Авторами даної роботи пропонується ефективний спосіб класифікації думок та підхід виконання відповідних дій із заданою послідовністю сигналів. Вони демонструють ефективність ELM для класифікації різних думок при порівняно невеликій кількості навчальних зразків з 5-канальної гарнітури ЕЕГ. Дослідники перетворюють дані електроенцефалографа на набір функцій для моделі ELM, оцінюючи логарифмічну потужність коефіцієнтів дискретного вейвлет-перетворення, які відповідають п'ятьом частотним смугам. ELM забезпечує від 90% до 100% точності класифікації в залежності від навчальних зразків та кількості прихованих вузлів, у порівнянні з 52% - 60% для багатошарового персептрону (далі – MLP).

Основним джерелом для роботи технології BCI є сигнали ЕЕГ. BCI використовується як прямий зв'язок між мозком і зовнішнім пристроєм. Він в основному використовується для підтримки, збільшення або відновлення людської когнітивної або сенсорної рухової функції.

Авторами роботи [6] запропонована класифікація ЕЕГ-сигналів, зареєстрованих в 2-х пробах: з відкритими та закритими очима (моторна уява). Набір даних є еталонними даними, отриманими від Каліфорнійського університету, Ірвін. Були розглянуті такі класифікатори навчання: ELM, екстремальне навчання, що кодується за фазою (Phase Encoded Extreme Learning Machine, PE-CELM) та Fully Complex Valued Fast Learning Classifier (FC-FLC). ELM – це швидкий класифікатор, у якому

ваги є випадково призначеними й ніколи не уточнюваними. PE-CELM і FC-FLC є нещодавно розробленими складними ціннісними нейронними класифікаторами, які можуть бути використані для вирішення задачі класифікації сигналів ЕЕГ. Загалом мережі (Real Valued Networks) мають меншу обчислювальну здатність, порівняно з складними нейронними мережами комплексної оцінки. PE-CELM і FC-FLC мають кращі показники точності (табл. 1.1), ніж ELM-класифікатор, завдяки ортогональній границі рішень.

Таблиця 1.1 – Результати класифікації сигналів в стані з відкритими очима [6]

Класифікатор	Загальна ефективність випробувань, %	Середня ефективність випробувань
PE-CELM	72	50
FC-FLC	72,3	50
ELM	60	50

Для досліджень нейрокомп’ютерного інтерфейсу також важливий аналіз швидкості руху з використанням сигналів ЕЕГ. Однак мало які дослідження навчають, як декодувати швидкість складного руху. У статті [7] автори застосували ELM для вивчення способу декодування швидкості складного руху сигналів ЕЕГ. Дослідники спроектували нову експериментальну парадигму та проаналізували ефекти впливу кількості прихованих нейронних вузлів та діапазону частот на продуктивність декодування. В даній роботі вирішується задача побудови чітких дешифраторів руху з сигналів ЕЕГ для розробки протезів та систем реабілітації на основі BCI. Як показано на рис. 1.5, найвища точність декодування асоціюється з смугою частот 0,16 - 1 Гц, а точність декодування інших діапазонів частот відносно низька.

Ідентифікація емоцій необхідна, наприклад, у застосуванні некомп’ютерного інтерфейсу, при емоційній терапії та медичній реабілітації. Деякі емоційні стани можна характеризувати за допомогою частот сигналу ЕЕГ, в тому числі, збудження, розслаблення і сум. Сигнал, зареєстрований у певній частоті, корисний для розпізнавання трьох емоційних станів.

Класифікація сигналу ЕЕГ в реальному часі залежить від методів забезпечення необхідної відмінності класів та методів розпізнавання з високою швидкістю розрахунків.

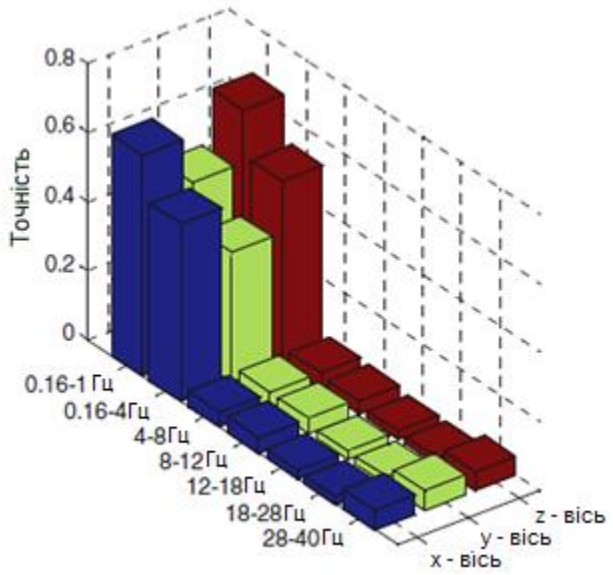

Рисунок 1.5 – Точність моделі дешифрування ELM, по семи аналізованих вхідних частотних діапазонах у 3-х суб'єктів [7]

У роботі [8] запропоновано моніторинг емоцій людей в реальному часі, використовуючи вейвлет аналіз та лінійну квантизацію векторів (Learning Vector Quantization, LVQ). До початку автоматичної класифікації, було здійснено навчання, використовуючи дані навчальної вибірки від 10 суб'єктів, після 10 випробувань, 3-х класів та 16-ти сегментів (480 об'єктів в наборі даних). Кожен сегмент проходить вейвлет-аналіз. Отримані параметри далі використовуються для системи ідентифікації трьох емоційних станів з використанням LVQ. Результати показали, що точність зростає від 72% до 87% з використанням вейвлет пакету і збільшенні навчальної вибірки. Експериментальна система була інтегрована в бездротову систему моніторингу ЕЕГ, для розпізнавання емоцій у реальному часі кожні 10 секунд (рис. 1.6).

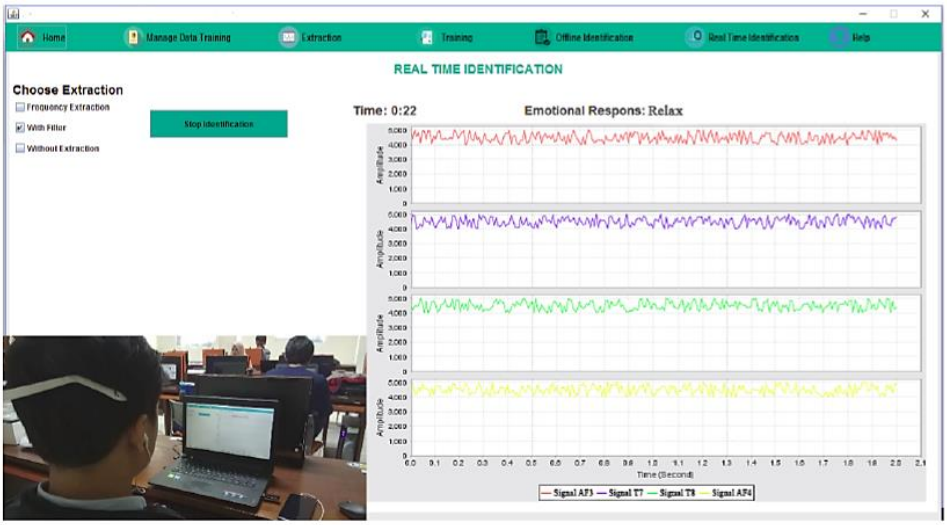

Рисунок 1.6 – Система моніторингу емоційного стану в реальному часі [8]

Швидкість прийняття рішення складає 0,44 секунди, що не є значущим у відношенні до 10 секунд.

1.2 Дослідження епілепсії

Електроенцефалографія є найважливішим інструментом діагностики та лікування епілепсії. Вона дозволяє спостерігати за проявами, які тісно пов’язані з епілепсією або епілептичними нападами, і розташуванням зон головного мозку, які викликають симптоми епілепсії.

У роботі [9] представлена автоматизована класифікація сигналів ЕЕГ для виявлення епілептичних судом при використанні вейвлет-перетворення та ектремального навчання для кожного каналу окремо. Мета полягає в тому, щоб створити систему з зменшеним часом і ресурсами обчислення, та мінімальною кількістю необхідних електродів. Процес прийняття рішень складається з трьох етапів: (а) попередньої обробки, (б) вилучення ознак на основі вейвлет-перетворення та (в) класифікації за допомогою методу ектремального навчання. Запропонований

алгоритм був протестований на трьох різних наборах даних з бази даних CHB-MIT EEG з використанням тільки каналу FT10-T8. Запропонований метод досягає точності класифікації 94,85% (табл. 1.2).

Таблиця 1.2 – Результати класифікації запропонованої системи [9]

Пацієнт	Точність класифікації, %	Тривалість навчання, сек
chb03	96,88	0,0121
chb08	94,97	0,0121
chb13	95,05	0,0121
Середнє значення	96,85	0,0121

Оскільки довгострокова реєстрація (12-24 годин і більше) електроенцефалограми і ЕЕГ-відео-моніторинг стають дедалі частішими в клінічній практиці, обсяг даних ЕЕГ, які потребують анотації клінічними експертами, зростає експоненціально, підкреслюючи необхідність автоматизованих систем виявлення епілептичних приступів. Як рішення були запропоновані методології, що базуються на методах машинного навчання, але вони потребують анотованих даних ЕЕГ від декількох пацієнтів для навчання, тоді як методи без навчання, які не мають таких обмежень, рідкість.

Таким чином, авторами роботи [10] була розроблена безконтрольна методологія виявлення приступів, яка забезпечує високу продуктивність виявлення судом при значному зменшенні часу та зусиль, необхідних для перевірки великих обсягів даних із сигналів ЕЕГ. Для того щоб знайти інтенсивне накопичення енергії сигналу над основними дельта, тета та альфа-частотними діапазонами, ритмічна активність виявлась шляхом аналізу спектральної інформації кожного каналу ЕЕГ незалежно. Ритмічна активність, яка виражається під час епілептичного нападу, виявляється за допомогою набору із чотирьох простих умов виявлення судом. Запропонована методологія протестована на загальнодоступної базі даних CHB-MIT EEG, і результати показують, що середня чутливість 95,1% може бути отримана з коефіцієнтом помилкового виявлення 10,13 FP/h.

Епілепсія – це одне з серйозних і хронічних неврологічних розладів, що вражає мільйони людей у світі. Епілептичні напади, проявляються у різних формах симптомів, починаючи від короткої втрати свідомості до важких м'язових судом. Сучасна діагностика таких розладів виконується вручну неврологами, доступність яких обмежена. З появою комп'ютерного діагнозу, який залежить від ЕЕГ, прийняття рішення можна набагато пришвидшити. Для комп'ютерної діагностики важливу роль відіграють алгоритми обробки сигналів та методи машинного навчання. ЕЕГ-сигнал є неустаненим і різко відрізняється за своїм характером, тому його можна проаналізувати за допомогою нелінійних методів. У роботі [11] реалізована концепція перетворювачів коду. Так як, первинні результати класифікації були не задовільними, то наступним етапом обробки була додаткова оптимізація за допомогою класифікатора Adaboost.

Результати показали, що після використання Adaboost як пост-класифікатора, середнє значення досконалої класифікації становить близько 94,58%, середня точність класифікації – 97,29%, середній показник ефективності – 94,51% та середнє значення якості – 21,82.

Поява раптового, надмірно електричного імпульсу в мозку, є поширеним явищем у хворих на епілепсію: неврологічним розладом, який вражає близько 70 мільйонів людей у світі. Епілепсія в основному ділиться на два типи – локалізаційно-обумовлені (фокальні) напади та генералізовані напади (без осередкових проявів). Електроенцефалограма представляє собою запис електричних імпульсів головного мозку, але діагностика епілепсії та визначення правильного класу захворювання, з її допомогою, вимагає багато часу, а також може бути дороговартісним процесом, через необхідність підготовки фахівців для виконання рутинної інтерпретації характеру сигналу, який зазвичай «забруднений» шумами та артефактами, що впливають на візуальний аналіз і погіршують результати діагностики.

Автори роботи [12] пропонують технологію знешумлення сигналу ЕЕГ, використовуючи ансамбль методів розкладання сигналу на функції, який отримав

назву «емпіричний мод» (Ensemble Empirical Mode Decomposition, EEMD) та класифікацію на основі методів машинного навчання, використовуючи платформу MATLAB. Окрім EEMD, в експериментах використовувались: метод k-найближчих сусідів разом з PCA та метод зворотного поширення помилки (далі – BPN). Саме BPN показав найкращу точність класифікації (табл. 1.3)

Таблиця 1.3 – Точність алгоритму BPN по всім каналам одночасно [12]

	Навчальна вибірка	Перевірка достовірності	Тестова вибірка	Загальна
Точність, %	100	100	83,3	97,4

Електроенцефалограма є інструментом для моніторингу діяльності мозку, важливим для виявлення епілепсії. Автоматична ідентифікація епілептичних приступів є складним завданням і корисною допомогою нейрофізіологам. У дослідженні [13] порівнюються деякі алгоритми машинного навчання для вилучення ознак та класифікації для розпізнавання приступів епілепсії на основі даних ЕЕГ. В цьому дослідженні порівнюються наступні алгоритми класифікації: квантування навчального вектора (Generalized Relevance Learning Vector Quantization, GRLVQ), метод зворотного поширення помилки, метод опорних векторів та випадковий ліс (далі – RF), у поєднанні з аналізом головних компонент та вейвлет перетворенням. Сигнали ЕЕГ використанні в цьому дослідженні, були отримані з набору даних, який був сформований Університетом Бонна. Набір даних має п’ять класів. Класи A і B – це п’ять здорових людей у пробах з відкритими та закритими очима. Класи C, D і E – п’ять суб’єктів з епілепсією, де C та D – сигнали, які не місять приступів, а E – лише епілептична активність. Завдання полягає в тому, щоб розпізнати та класифікувати 5 класів із єдиного набору даних. Використовувались такі вимірювання для оцінки методів: точність, повнота (recall), точність для кожного класу окремо на навчальній вибірці та час класифікації тестової вибірки. Найкращим методом вилучення ознак став PCA, найкращим показником при визначенні п’яти класів в наборі даних – GRLVQ, з загальною точністю, точністю в межах класу та повнотою – 0,9866, а час тестування становив – менше 0,1 секунди.

У роботі [14] проводиться дослідження з виявлення епілептичного нападу у сигналах ЕЕГ в три етапи. На першому етапі використовується дискретне вейвлет перетворення для розкладання сигналу ЕЕГ на відповідні частотні діапазони. На другому етапі були отримані статистичні ознаки для кожної частотної смуги, далі класифікація сигналів ЕЕГ з епілептичною активністю та нормальною здійснювалась використовуючи метод SVM. Результати експериментального аналізу показують точність запропонованого методу в 95%, з параметром поділу вибірки на навчальну та тестову «70-30».

Автори роботи [15] презентують новий метод виявлення епілептичних нападів з використанням ЕЕГ-сигналу за допомогою короткочасного перетворення Фур'є (STFT) та методу CNN. Порівняння результатів представленого підходу та вже класичного методу опорних векторів, показало, що підхід, представлений у цій роботі (рис. 1.7), є кращими.

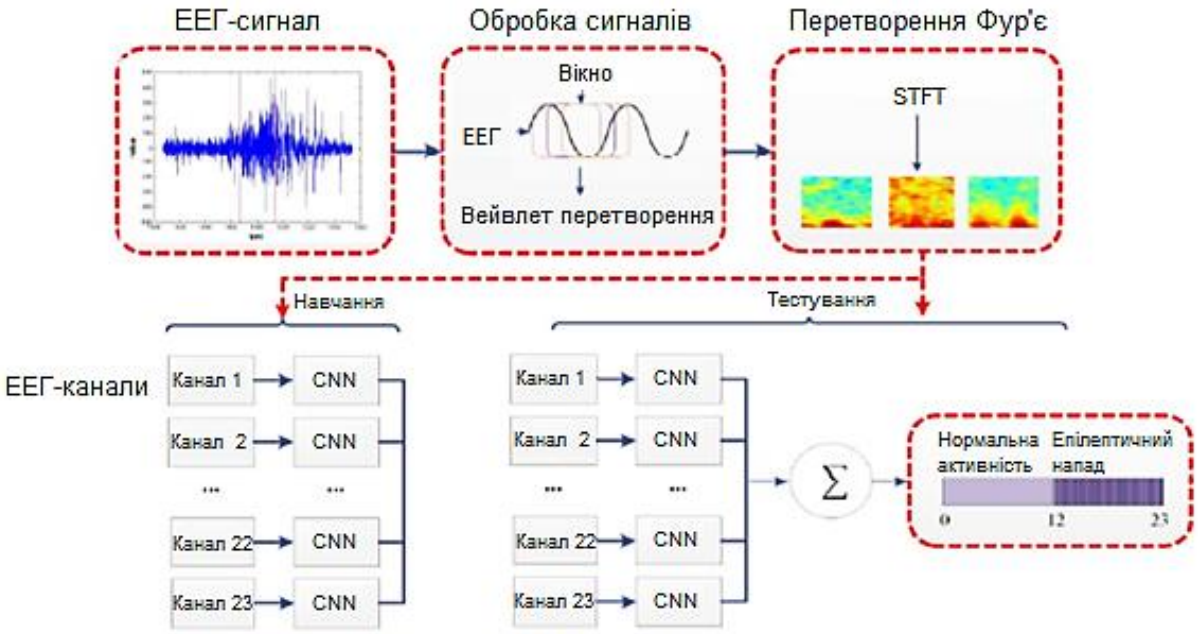

Рисунок 1.7 – Загальна структура алгоритму [15]

Експериментальний результат на одному каналі досягає середньої точності в 86%. Крім того, метод багатоканального розпізнавання може збільшити середню

точність до 90%, а чутливість (true positive rate) до 96,5% при зменшенні специфічності алгоритму (false positive rate) до 7%. Всі ці показники показують високу продуктивність та стійкість підходу для виявлення епілептичного нападу.

1.3 Дослідження з покращення алгоритмів автоматизованого аналізу електроенцефалограми

Автори роботи [16] зосереджені на використанні машинного навчання для інтерпретації та розуміння мозкових хвиль, під час виконання завдання (рис. 1.8).

Рисунок 1.8 – Блок-схема моделі управління [16]

Тож, сигнали активності головного мозку використовувались, для моделі управління роботизованих рук і руху пальців (рис. 1.9). Це було зроблено шляхом декодування нейронної активності, пов’язаної з рухом пальців, в режимі реального часу. Результати були використані для тренувань роботів з гнучкими руками, які можуть надати можливість людям з пошкодженнями спинного мозку, інсульту мозку

і аміотрофічним бічним склерозом (amyotrophic lateral sclerosis) керувати роботопротезом, думаючи про рухи.

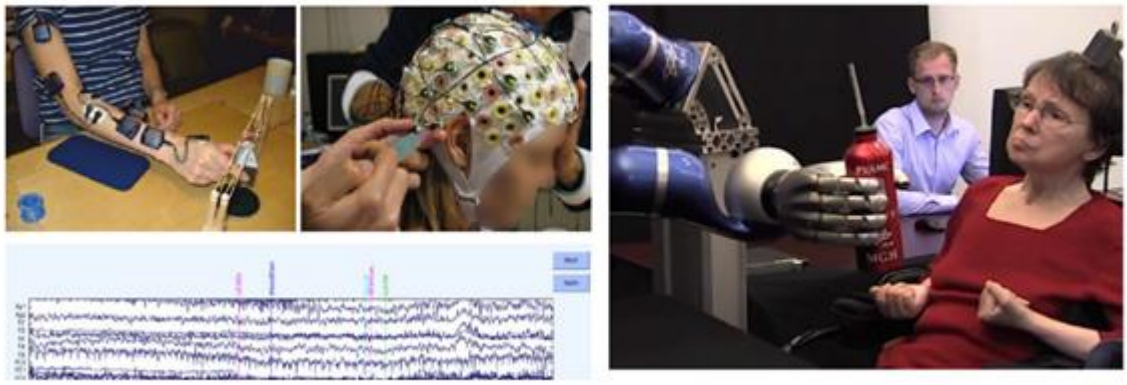

Рисунок 1.9 – Реєстрація сигналів ЕЕГ під час виконання рухів пальцями рук [16]

Проект є новим у певному сенсі, він спирається на виявленні ознак, використовуючи метод головних компонент на першому етапі дослідження мозкових хвиль. Наступним кроком є використання нечіткої нейронної мережі (Neuro Fuzzy system) для вивчення ознак (рис. 1.10).

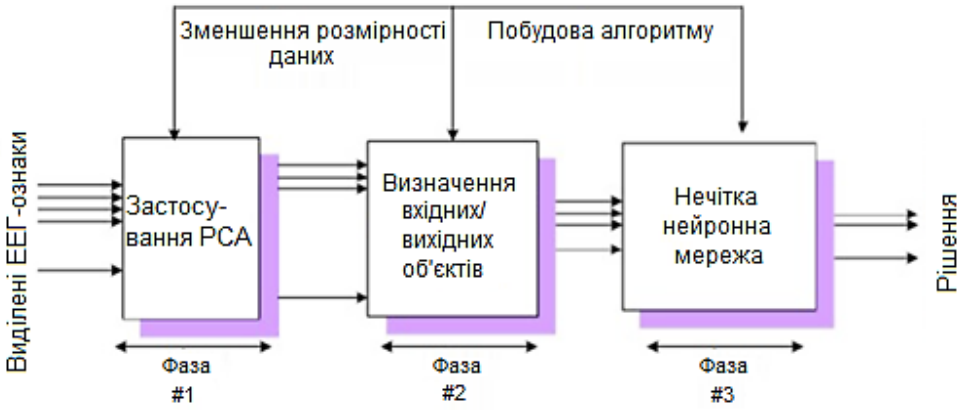

Рисунок 1.10 – Схема навчання нечіткої нейронної мережі після застосування PCA [16]

У статті представлені деталі спроектованого інтелектуального роботизованого ручного протеза, який вивчає передбачувану поведінку людини, використовуючи хвилі нейронної активності мозку.

Електроенцефалограма містить величезну кількість даних про діяльність мозку людини, але все ще вони розглядаються та аналізуються, перш за все, лікарями. У більшості випадків ці дані заражені частотами немозкових імпульсів, які називаються артефактами та можуть бути дуже складними для візуального виявлення, що призводить до помилкового діагнозу. Метою роботи [17] є виявлення артефактів шляхом визначення найбільш відповідних ознак як у часовій, так і частотній областях, а також тренування різних алгоритмів контролю: дерево прийняття рішень (далі – DT), SVM та метод k-найближчих сусідів (далі – KNN), щоб відрізняти чисті та зашумлені сигнали. На рис. 1.11 зображено рейтинговий індекс інформативності кожної ознаки, які використовувались для навчання.

Рисунок 1.11 – Оцінка інформативності ознак [17]

Продуктивність запропонованого методу перевищила результати аналогічних досліджень, а саме: відсоток об’єктів, для яких класифікатор прийняв правильне рішення склав 98,78%, а точність (precision) та повнота (recall) – 98,30% та 98,40% відповідно, для DT класифікатора.

Робота [18] присвячена розробці нового підходу для автоматичної класифікації стадії сну на основі даних електроенцефалограми. Запропонована методологія використовує сучасні математичні інструменти, такі як дослідження синхронізації та метрики теорії графів, що застосовуються до ЕЕГ-даних для дослідження сну. Отримані функції використовувалися для навчання трьох методів машинного навчання, а саме k-найближчих сусідів, SVM та нейронної мережі. Оцінка їх порівняльної продуктивності досліджується відповідно до їх точності (відсоток об’єктів, для яких класифікатор прийняв правильне рішення). Найкращий показник в 89,07% продемонстрував метод SVM.

У роботі [19] автори пропонують 3 методи машинного навчання, такі як RF, беггінг (Bagging) та SVM, разом з часовими характеристиками для класифікації етапів сну на основі одноканальної ЕЕГ. Нічні полісомнограми були зареєстровані від 25 суб’єктів з використанням стандарту R&K. Для виявлення стадії сну використовувались сигнали ЕЕГ. Результати автоматичного та ручного аналізу сигналів були об’єднані, епоха за епохою. База даних для дослідження складалась з 96000 30-ти секундних ЕЕГ-епох в стані сну. ЕЕГ-епохи класифікувались на шість стадій (W/S1/S2/S3/S4/REM) (рис. 1.12).

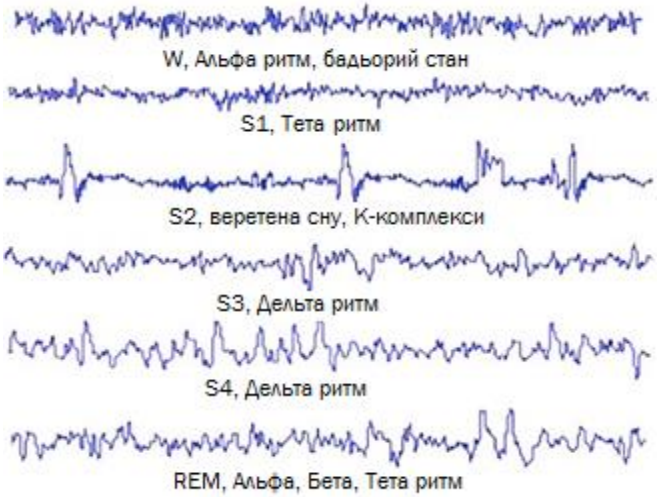

Рисунок 1.12 – Стадії сну на електроенцефалограмі [19]

Найкращі результати класифікації (рис. 1.13) були отримані за допомогою методу випадкового лісу досягли загальної точності, показників специфічності та чутливості в 97,73%, 96,3% та 99,51% відповідно.

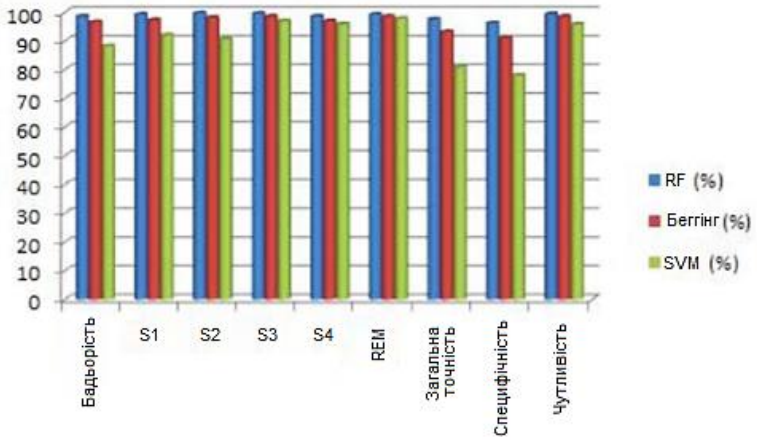

Рисунок 1.13 – Порівняння результатів методів RF, беггінгу та SVM [19]

Традиційний моніторинг сну, проведений у професійних лабораторіях для сну та проаналізований спеціалістом із сну, є дорогим та трудомістким процесом. Нещодавня розробка для реєстрації ЕЕГ (light-weight EEG headband) забезпечує можливістю для домашнього моніторингу сну. В дослідженні [20] запропоновано підхід для автоматичного моніторингу сну за допомогою методів машинного навчання (рис. 1.14).

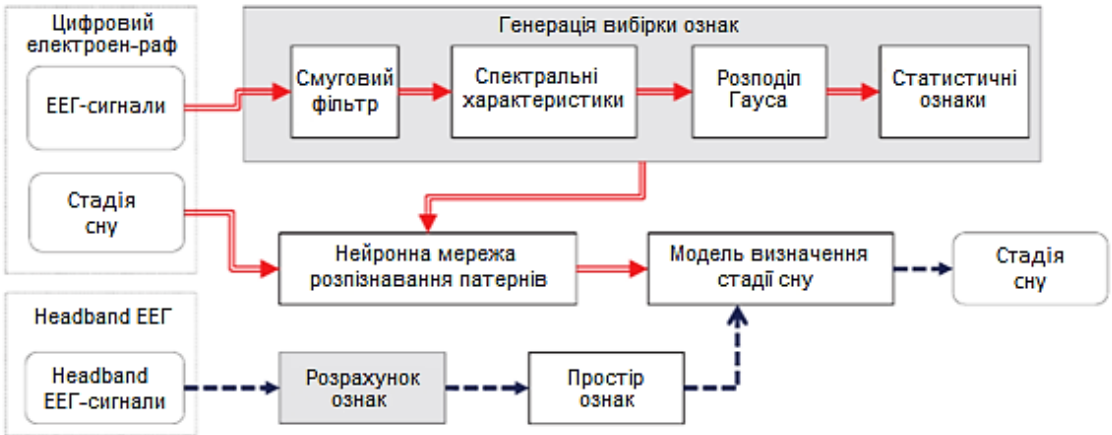

Рисунок 1.14 – Схема моделі розпізнавання стадії сну [20]

Набір ефективних та дієвих функцій витягується з даних ЕЕГ-сигналів. Використання колекції добре анотованих даних щодо сну забезпечила високу якість навчальної моделі. Авторами було запропоновано використання алгоритму картографування ознак згенерованих з даних ЕЕГ, для відображення їх просторів. ЕЕГ-сигнали, тривалістю 1-ї години, були зареєстровані в лабораторії під час дрімоти (сну). Попередній результат (рис. 1.15) показує, що стадії сну, виявлені запропонованим методом, мають достатньо високі оцінки.

Дослідження сну важливі для діагностики порушень сну, таких як безсоння, нарколепсія або апное сну. Вони покладаються на ручний аналіз станів сну з сирих сигналів полісомнографії, що є виснажливою візуальною задачею та вимагає високої кваліфікації професіоналу.

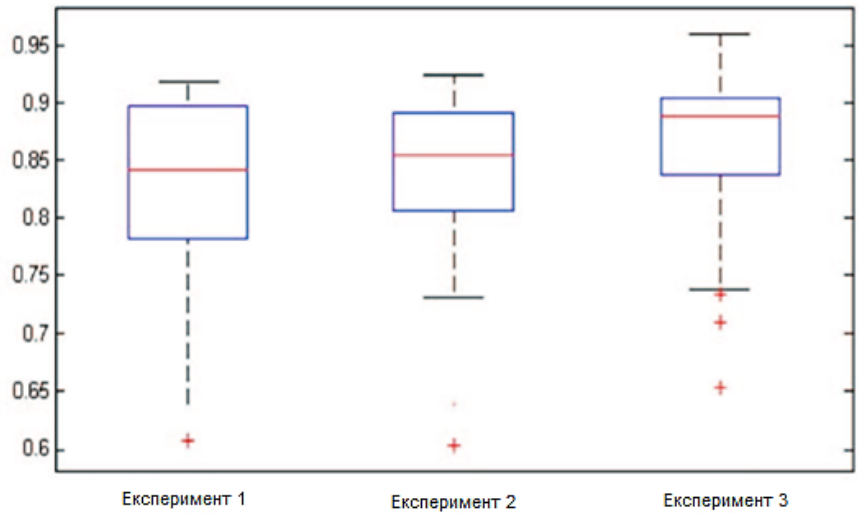

Рисунок 1.15 – Порівняння точності розпізнавання стадії сну [20]

Тож, за останні роки було здійснено багато зусиль для автоматичного аналізу на основі методів машинного навчання. У роботі [21] автори застосовують багатоетапний спектральний аналіз для створення візуально інтерпретованих зображень моделей сну з сигналів ЕЕГ, які надходять на вхід до глибинної згорткової нейронної мережі (deep convolutional network), яка навчається для вирішення задачі візуального розпізнавання. Представлена система, здатна точно класифікувати етапи

сну у нових пацієнтів. Оцінки класифікації, здійсненої на широко використовуваному загальнодоступному наборі даних, вигідно відрізняються від найсучасніших результатів, а також створюють основу для візуальної інтерпретації результатів.

Висновок до Розділу 1

У даному розділі представлений огляд останніх досліджень з аналізу сигналів електроенцефалограми з використанням методів машинного навчання. Були виділені основні напрямки досліджень:

- системи нейрокомп’ютерного інтерфейсу, які здатні керувати протезами за допомогою думок, через сигнали ЕЕГ; такі системи можуть значно «спростити» та покращити життя людям з фізичними обмеженнями;
- дослідження з виявлення та попередження патології, а саме епілепсії;
- покращення якості сну, за допомогою аналізу ЕЕГ сигналів.

РОЗДІЛ 2
МЕТОДИ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АНАЛІЗУ ЕЛЕКТРОЕНЦЕФОЛОГРАМИ

Аналіз ЕЕГ донині залишається актуальним інструментом для пошуку і ефективного застосування витончених математичних методів, що дозволяють вирішувати нові завдання.

2.1 Наївний Баєсів класифікатор

Байєсів класифікатор – широкий клас алгоритмів класифікації, заснований на принципі максимуму апостеріорної ймовірності. Для класифікуємого об’єкта обчислюються функції правдоподібності кожного з класів, по ним обчислюються апостеріорні ймовірності класів. Об’єкт відноситься до того класу, для якого апостеріорна ймовірність максимальна. [22]

Байєсівський підхід лежить в основі багатьох досить вдалих алгоритмів класифікації: наївний Байєсів класифікатор, лінійний дискримінант Фішера, квадратичний дискримінант, метод Парзенівского вікна, метод радіальних базисних функцій, логістична регресія.

Наївний Байєсів класифікатор (далі – НБК) – це алгоритм класифікації, заснований на теоремі Байеса з припущенням про незалежність ознак. Іншими словами, НБА передбачає, що наявність якої-небудь ознаки в класі не пов’язано з наявністю будь-якої іншої ознаки. Наприклад, фрукт може вважатися яблуком, якщо він червоний, круглий і його діаметр становить близько 8 сантиметрів. Навіть якщо ці ознаки залежать один від одного або від інших ознак, в будь-якому випадку вони вносять незалежний внесок у ймовірність того, що цей фрукт є яблуком. У зв’язку з таким припущенням алгоритм називається «наївним». [23]

Теорема Байєса дозволяє розрахувати апостеріорну ймовірність P (c|x) на основі P (c), P (x) і P (x|c) [24]:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \quad (2.1)$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c) \quad (2.2)$$

Якщо компоненти формули (2.1) розглядати на прикладі 35 річного чоловіка, з зарплатою 40 тис. і хорошою кредитною історією, то:

P (c|x) – умовна ймовірність, того, що клієнт x буде купувати комп'ютер за умови, що ми знаємо вік і дохід клієнта (постеріорна ймовірність c);

P (c) – ймовірність того, що клієнт буде купувати комп'ютер, незалежно від віку та доходів (апріорна ймовірність c);

P (x|c) – ймовірність того, що особа 35 років, заробляє $ 40 000 точно купила комп'ютер;

P (x) – ймовірність того, що людина з нашої вибірки клієнтів 35 років, заробляє $ 40 000. [25]

Теорема Байєса стверджує [26], якщо щільності розподілу кожного з класів відомі, то цей алгоритм оптимальний, тобто володіє мінімальною ймовірністю помилок.

На практиці щільності розподілу класів, як правило, не відомі. Їх доводиться оцінювати (відновлювати) за навчальною вибіркою. В результаті Баєсів алгоритм перестає бути оптимальним, так як відновити щільність по вибірці можна тільки з деякою погрішністю. Чим коротше вибірка, тим вище шанси підігнати розподіл під конкретні дані і зіткнутися з ефектом перенавчання.

Також суттєво негативною стороною НБА є те, що якщо в тестовому наборі даних присутнє значення категорійної ознаки, яке не зустрічалося в навчальному наборі даних, тоді модель присвоїть нульову ймовірність цього значення і не зможе зробити прогноз. Це явище відоме під назвою «нульова частота» (zero frequency). Дану проблему можна вирішити за допомогою згладжування. Одним з найпростіших методів є згладжування Лапласа (Laplace smoothing). [23]

З іншого боку, моделі на основі НБА досить прості і вкрай корисні при роботі з дуже великими наборами даних. При своїй простоті НБА здатний перевершити навіть деякі складні алгоритми класифікації. Класифікація, в тому числі багатокласова, виконується легко і швидко. У тих рідкісних випадках, коли ознаки дійсно незалежні (або майже незалежні), наївний Байесів класифікатор (майже) оптимальний і при цьому вимагає менший обсяг навчальних даних. [27]

Області застосування: банківський бізнес (кредити), маркетинг, медицина.

2.2 Градієнтний бустинг над вирішальними деревами

Бустинг – це процедура послідовної побудови композиції слабких алгоритмів машинного навчання, коли кожен наступний алгоритм прагне компенсувати недоліки всіх попередніх алгоритмів [27]. Слабким класифікатором називається класифікатор, який дає лише злегка кращий результат, ніж випадкове вгадування (його передбачення слабо корельовані з істинним розподілом класів). Прогнози ж сильного класифікатора сильно корельовані з істинним розподілом. Фінальний класифікатор шукається у вигляді лінійної комбінації класифікаторів. [28]

Кількісні оцінки узагальнюючої здатності бустингу формулюються в термінах відступу. Ефективність бустингу пояснюється тим, що в міру додавання базових алгоритмів збільшуються відступи між навчальними об’єктами. Причому бустинг

продовжує збільшувати відстань між класами навіть після досягнення безпомилкової класифікації навчальної вибірки. [29]

Для вирішення задачі розпізнавання об’єктів з багатовимірного простору X з простором міток Y, необхідна навчальна вибірка $\{x_i\}_{i=1}^{N}$, де $x_i \in X$. Істинні значення міток відомі для кожного об’єкта $\{y_i\}_{i=1}^{N}$, $y_i \in Y$. Необхідно побудувати оператор, який якомога точніше зможе передбачати мітки для кожного нового об’єкта $x \in X$. Якщо в якості базового сімейства алгоритмів H розглядати регресивні дерева, кожен елемент $h(x; a) \in H: X \rightarrow R$ якого визначається деяким вектором параметрів $a \in A$. Тоді кожне вирішальне дерево має J листових вершин, які відповідні J непересічним областям $\{R_j\}_{j=1}^{J}$, на які розбивається простір об’єктів X. Кожній листової вершині відповідає деяке значення регресії b_j, яке буде відповіддю класифікатора в разі потрапляння аналізованого об’єкта у відповідну область. Можна записати цей факт наступною формулою:

$$h\left(x, \{a_j, R_j\}_{j=1}^{J}\right) = \sum_{j=1}^{J} a_j I[x \in R_j], \tag{2.3}$$

де I[A] – індикатор події A.

Видно, що в цій сумі рівно один доданок буде ненульовим. Тоді додавання доданка в градієнтному бустнигу буде відбуватися так:

$$F_m(x) = F_{m-1}(x) + b_m \sum_{j=1}^{J} a_{jm} I[x \in R_j] =$$

$$= F_{m-1}(x) + \sum_{j=1}^{J} c_{im} I[x \in R_j], c_{jm} = a_{jm} b_{jm} \tag{2.4}$$

Таким чином ми просто додаємо до алгоритму деяке інше вирішальне дерево. Замість того щоб виконувати лінійний пошук коефіцієнта перед новою складовою, як в класичному градієнтному бустингу, параметри дерева з фіксованими $\{R_j\}$ повністю переналаштовуються. Це дозволяє будувати більш якісну композицію. Такий різновид бустингу називається Tree-boost.

На сьогоднішній день бустинг є одним з найпотужніших алгоритмів розпізнавання, що досягається адатпивною технікою побудови композиції. До того ж, бустинг надає безліч можливостей для варіацій. По-перше, можна розглядати різні функції втрат. Це дозволяє вирішувати як завдання класифікації, так і завдання регресії. По-друге, можливо розглядати будь-яке сімейство базових алгоритмів.

Бустинг над вирішальними деревами вважається одним з найбільш ефективних варіантів бустингу. А враховуючи, що вирішальні дерева в свою чергу теж використовують базові алгоритми (наприклад, порогові, лінійні і т.д.), в результаті виходить величезна кількість варіантів для налаштувань.

Зрозуміло, бустинг не позбавлений недоліків. Бустинг – трудомісткий метод, і працює він досить повільно. Без додаткових модифікацій він має властивість повністю підлаштовуватися під дані, в тому числі під помилки і викиди в них. Ідея бустингу зазвичай погано застосовна до побудови композиції з досить складних і потужних алгоритмів. Побудова такої композиції займає дуже багато часу, а якість істотно не збільшується. А результати роботи бустингу складно інтерпретовані, особливо якщо в композицію входять десятки алгоритмів. [30]

2.3 Метод опорних векторів

Метод опорних векторів заснований на ідеї поділу простору об’єктів, на підпростори, відповідні класам. У разі бінарної класифікації навчання методу зводиться до пошуку гіперплощини з деякою товщиною, яка є математичною

сутністю методу, що розділяє об’єкти різних класів навчальної вибірки [31]. Наприклад, так як це показано на рис. 2.1, де у тривимірному просторі площина відділяє кульки синього кольору від червоних кульок.

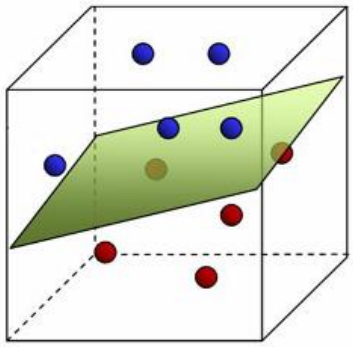

Рисунок 2.1 – Приклад відділяючої гіперплощини [31]

Нехай маємо два класи $\{-1, +1\}$, в яких об’єкти простору X описуються n-вимірними дійсними векторами: $X = R^n$, $Y = \{-1, +1\}$. Тоді лінійний пороговий класифікатор матиме вигляд [32]:

$$a(x) = \text{sign}\left(\sum_{j=1}^{n} w_j x^j - w_0\right) = \text{sign}\left(\langle w, x\rangle - w_0\right), \quad (2.5)$$

де $x = (x^1, \ldots, x^n)$ – вектор ознак x; вектор $w = (w^1, \ldots, w^n) \in R^n$ і $w_0 \in R$ – параметри алгоритму, які необхідно знайти по навчальній вибірці; рівняння $(w, x) = w_0$ – описує гіперплощину, яка розділяє класи в просторі R^n.

Зокрема, метод опорних векторів шукає відділяючу гіперплощину, максимально віддалену від будь-яких точок даних. Відстань між цією гіперплощиною і найближчою точкою даних називається зазором класифікатора. У методі опорних векторів обов’язково мається на увазі, що вирішальна функція цілком визначається (звичайно малою) підмножиною даних, що впливають на положення відділяючої гіперплощини. Ці точки називаються опорними векторами (рис. 2.2). [33]

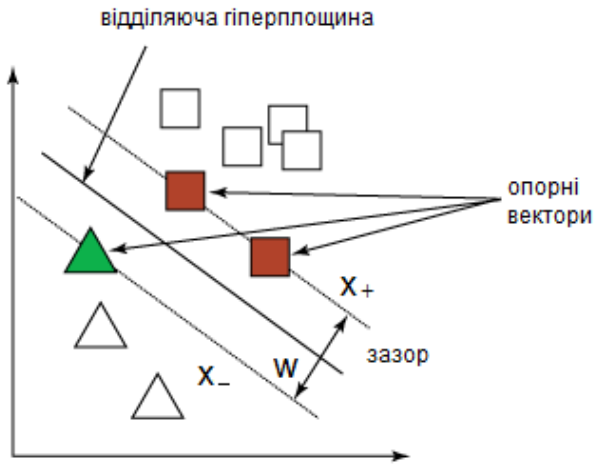

Рисунок 2.2 – Двовимірне представлення оптимальної гіперплощини і опорні вектори [33]

Для того, щоб розділяюча гіперплощина якнайдалі стояла від точок вибірки, ширина смуги повинна бути максимальною і дорівнює [32]:

$$\frac{\langle x_+ - x_-, w\rangle}{\|w\|} = \frac{2}{\|w\|}, \tag{2.6}$$

де, x_- та x_+ – класи об’єктів.

Спосіб, яким можна провести відділяючу гіперплощину за методом опорних векторів, не є унікальним (рис. 2.3). Завжди існує багато різних можливостей розташування гіперплощини.

Об’єкти, що класифікуються, не завжди можуть бути розділені гіперплощиною. У реальних системах будуть наявними похибки в даних (рис. 2.4), внаслідок яких гіперплощина не виконає розподіл абсолютно точно. [31]

Звісно, що метод опорних векторів не повинен враховувати забагато похибок класифікації об’єктів, тому потрібно вводити додатковий параметр, котрий встановлює скільки невірно класифікованих об’єктів можуть перетинати зазори

гіперплощини і як далеко вони можуть розташовуватись відносно них. Таким чином, вводиться так звана м’яка межа похибки навколо гіперплощини.

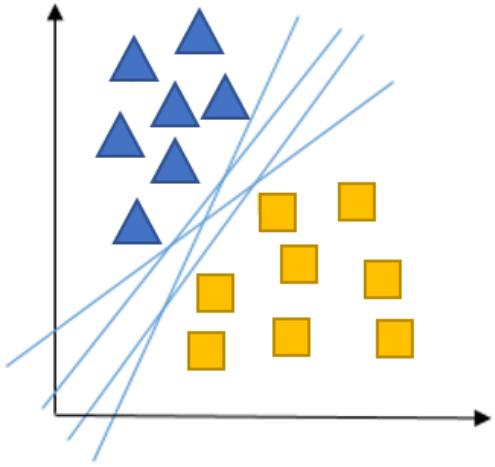

Рисунок 2.3 – Можливі варіанти розташування гіперплощини у двовимірному просторі

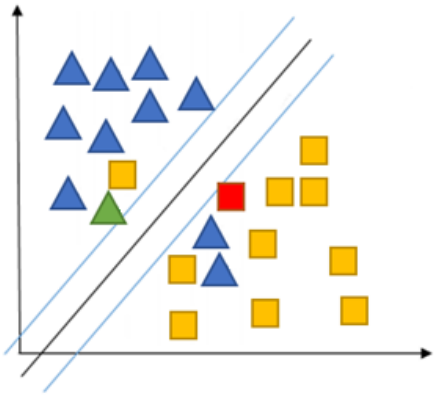

Рисунок 2.4 – Можливі похибки під час класифікації об’єктів

Об’єкти, що класифікуються, можуть бути поділені лінійно лише в окремих випадках. Здебільшого вони не є такими. Для вирішення проблеми лінійного розподілення використовують функції ядра, що проектують дані з низьковимірного простору у багатовимірний.

При вірному виборі функції ядра об’єкти можуть бути розділені лінійно гіперплощиною у багатовимірному просторі. [31]

2.4 Метод незалежних компонент

Метод незалежних компонент (далі – МНК) активно використовується багатьма дослідниками для вирішення різних проблем аналізу ЕЕГ [34]. Сигнали ЕЕГ з різних датчиків не є незалежними. В свою чергу, МНК полягає в лінійному комбінуванні сигналів в компоненти так, щоб отримані компоненти були незалежні [35], дозволяє очистити ЕЕГ-сигнали від артефактів, відокремити різного роду сигнали, локалізувати джерела імпульсів і т. д. [34]

Важливою особливістю МНК є той факт, що поділ на незалежні компоненти (далі – НК) відбувається не за допомогою прямої просторової фільтрації, а з використанням інформації, що міститься безпосередньо в реєстрованому наборі даних. В основі лежить оманливо просте, але статистично і фізіологічно правдоподібне припущення про те, що джерела активності перебувають на деякій відстані один від одного і функціонують незалежно.

Таким чином, локально когерентні сигнали, що виходять від одного джерела ЕЕГ, теоретично, будуть згруповані разом в одну НК, в той час як всі сигнали, що не мають відношення до цього джерела, будуть усунені і згруповані в інші НК. [36]

Для того щоб дати визначення МНК, необхідно виконати обчислення: є n випадкових векторів $x_1, \dots, x_n$, які представляють собою лінійні комбінації n випадкових векторів $s_1, \dots, s_n$, кожен елемент яких, відповідно, дорівнює:

$$x_{i,j} = a_{i,1}s_{1,j} + a_{i,2}s_{2,j} + \cdots + a_{i,n}s_{n,j} \qquad (2.7)$$

де $a_{i,j}$ – дійсні коефіцієнти (i, j = 1, … , n).

За визначенням s_i статистично незалежні один від одного.

Іншими словами, є набір векторів (матриця X), які є лінійними комбінаціями незалежних компонент (матриця S). Таким чином, модель МНК може бути записана у вигляді:

$$X = WS, \tag{2.8}$$

де X – матриця даних вихідного простору;
W – матриця вагів для переходу з простору S в простір X;
S – матриця незалежних компонент.

Мета методу незалежних компонент полягає у визначенні матриці W^{-1}, за допомогою якої можна буде визначити матрицю НК S за формулою:

$$S = W^{-1}X \tag{2.9}$$

Максимальна кількість незалежних компонент, для зручності обчислень, узгоджується з розмірністю вхідних даних. Тобто для n різних зареєстрованих векторів ми отримаємо m різних незалежних компонент, де $m \leq n$. (Хоча існують алгоритми МНК, що дозволяють отримати $m > n$ незалежних компонент). [37]

Висновки до Розділу 2

В другому розділі були представлені методи машинного навчання для вирішення задачі розпізнавання та прогнозування епілептиформної активності головного мозку, а в якості даних, використовуються сигнали ЕЕГ. А саме, наївний Баєсів класифікатор, метод градієнтного бустингу та незалежних компонент, для покращення якості вхідних даних.

Головними перевагами їх використання є висока швидкість навчання, здатність до аналізу великих об’ємів інформації, стійкість до перенавчання, саме тому вони будуть використані в практичній частині.

РОЗДІЛ 3
МОДЕЛЬ ДЛЯ РОЗПІЗНАВАННЯ ТА ПРОГНОЗУВАННЯ ЕПІЛЕПТИЧНОЇ АКТИВНОСТІ

3.1 База даних

Для дослідження були використані ЕЕГ-сигнали з електронної бази даних вільного доступу Physionet [38]. Сигнали були зареєстровані в дитячій клініці Бостона та містять записи як здорових суб'єктів так і з патологією. Суб'єкти знаходились під цілодобовим моніторингом декілька днів після припинення приймання протисудомних препаратів, щоб охарактеризувати напади та оцінити їх кандидатуру для можливого хірургічного втручання. Групи сигналів були сформовані для 22 пацієнтів (5 хлопців та 17 дівчат віком від 2 до 22 років (табл. 3.1)), кожна група в середньому містить 24 – 26 сигналів (включаючи додаткові сигнали електрокардіограми і блукаючого нерва для деяких суб'єктів).

Для кожного суб'єкта було зареєстровано від 9 до 42 неперервних сигналів. Тривалість реєстрації в більшості випадків складає 1 годину, але також присутні файли тривалістю від 2-х до 4-х годин; сигнали з нападами можуть буди дещо коротшими.

Таблиця 3.1 – Інформація про пацієнтів

Пацієнт	Стать	Вік	Кількість зареєстрованих нападів
01	ж	11	7
02	ч	11	3
03	ж	14	7
04	ч	22	3
05	ж	7	5
06	ж	2	7
07	ж	15	3
08	ч	4	5
09	ж	10	3
10	ч	3	7
11	ж	12	3

Продовження таблиці 3.1 – Інформація про пацієнтів

12	ж	2	39
13	ж	3	12
14	ж	9	8
15	ч	16	20
16	ж	7	10
17	ж	12	3
18	ж	18	6
19	ж	19	3
20	ж	6	8
21	ж	13	4
22	ж	9	3

Всі сигнали були зареєстровані із міжнародною системою накладання електродів 10 – 20 і оцифровані з частотою 256 значень за секунду. В цілому, набір даних містить 664 сигнали, 198 з яких з патологічною активністю. Для кожної групи було додано інформаційний файл з детальним описом даних (про тривалість, початок та кінець нападу в секундах). [39]

3.2 Методи оцінки якості роботи класифікатора

В якості метрик оцінювання методів машинного навчання було застосовано декілька математичних функцій. Перша і найпростіша метрика – це точність (accuracy), яка є відсотком правильних відповідей класифікатора на тестовій вибірці:

$$\text{accuracy} = \frac{\text{P}}{\text{N}} \tag{3.1}$$

де P – кількість об’єктів, для яких класифікатор прийняв правильне рішення; N – розмір навчальної вибірки.

Проте, у цієї метрики є одна особливість яку необхідно враховувати. Вона привласнює всім об’єктам однаковий ваговий коефіцієнт, що може бути не коректно в разі, якщо розподіл об’єктів в навчальній вибірці сильно зміщений в бік одного або декількох класів. В цьому випадку у класифікатора є більше інформації по цих класах і відповідно в рамках цих класів він буде приймати більш адекватні рішення. На практиці це призводить до того, що маємо accuracy, скажімо, 80%, але при цьому в рамках якогось конкретного класу класифікатор працює на порядок гірше не розпізнаючи правильно навіть третину об’єктів тестової вибірки. [40]

Наступним чисельним методом оцінки якості є точність в межах класу (precision) та повнота (recall) у вигляді матриці неточностей (Confusion Matrix). У випадку, коли кількість класів відносно невелика (в даному досліджені реалізована бінарна модель), цей підхід дозволяє досить наочно представити результати роботи класифікатора.

Матриця неточностей – це матриця розміру N на N, де N – це кількість класів. Стовпці цієї матриці резервуються за експертними рішеннями, тобто реальним значеннями класів, до яких належать об’єкти вибірки, а ряди за рішеннями класифікатора. Коли алгоритм класифікує об’єкт з тестової вибірки, інкрементується число, яке стоїть на перетині рядка класу, який обрав класифікатор, і стовпця класу, до якого дійсно відноситься об’єкт. [41] Маючи таку матрицю точність і повнота для кожного класу розраховується дуже просто. Точність дорівнює відношенню відповідного діагонального елемента матриці і суми всього рядка класу (ф. 3.2). Повнота – це відношення діагонального елемента матриці і суми всього стовпчика класу (ф. 3.3). Формально [40]:

$$\text{precision}_c = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{c,i}} \tag{3.2}$$

$$\text{recall}_c = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{i,c}} \quad (3.3)$$

Результуюча точність класифікатора розраховується як арифметичне середнє його точності по всіх класах. Те ж саме з повнотою. Технічно цей підхід називається macro-averaging.

Зрозуміло що чим вище точність і повнота, тим краще. Але в реальному житті максимальна точність і повнота недосяжні одночасно і доводиться шукати якийсь баланс. Тому, хотілося б мати якусь метрику яка об'єднувала б у собі інформацію про точність та повноту нашого алгоритму. Саме такою метрикою є F-міра. F-міра є гармонійним середнім між точністю і повнотою. Вона прямує до нуля, якщо точність або повнота прямує до нуля. [40]

Дана формула (3.4) надає однаковий ваговий коефіцієнт точності і повноті, тому F-міра буде падати однаково при зменшенні і точності, і повноти.

$$F = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.4)$$

Можливо розрахувати F-міру надавши різний ваговий коефіцієнт (ф. 3.5), якщо завчасно відомо пріоритет однієї з цих метрик при розробці алгоритму.

$$F = (\beta^2 + 1)\frac{\text{Precision} \times \text{Recall}}{\beta^2\text{Precision} + \text{Recall}} \quad (3.5)$$

де β приймає значення в діапазоні $0 < \beta < 1$ якщо ви хочете віддати пріоритет точності (рис. 3.1а), а при $\beta > 1$ пріоритет віддається повноті (рис. 3.1б).

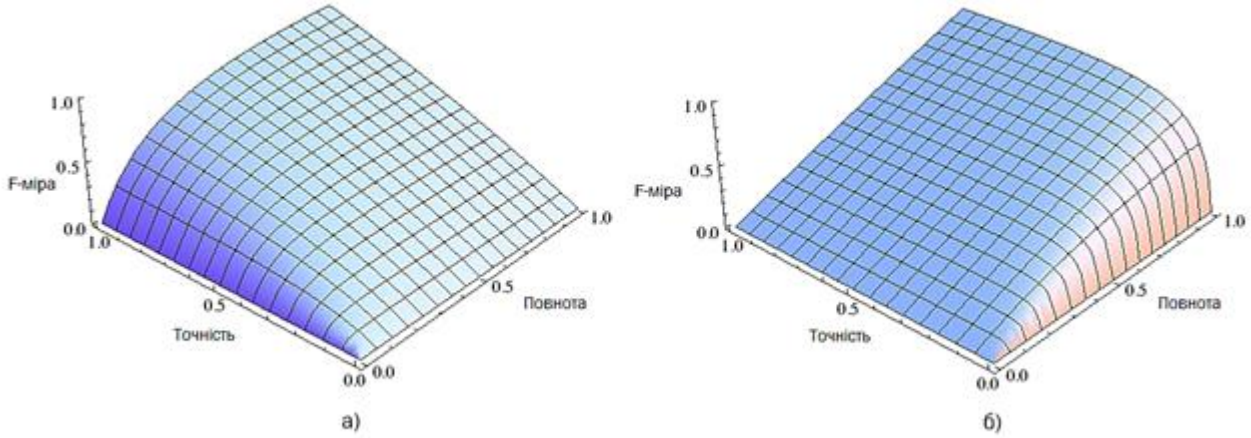

Рисунок 3.1 – а) F-міра з пріоритетом точності ($\beta^2 = ¼$), б) F-міра з пріоритетом повноти ($\beta^2 = 2$) [37]

При $\beta = 1$ формула зводиться до попередньої і тоді отримуємо збалансовану F-міру (рис. 3.2).

F-міра зводить до одного числа дві інших основоположних метрики: точність і повноту. Маючи в своєму розпорядженні подібний механізм оцінки буде набагато простіше прийняти рішення про позитивні чи негативні зміни в алгоритмі. [41]

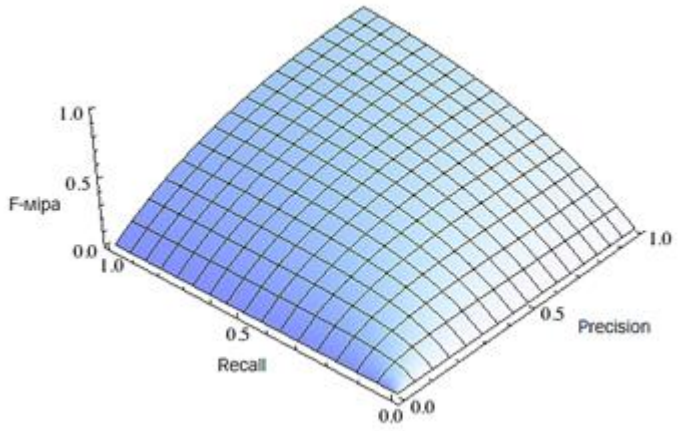

Рисунок 3.2 – Збалансована F-міра [37]

При прогнозуванні вірогідності, чим більшу ми можемо отримати повноту при меншій специфічності (частка об’єктів від загальної кількості, що не несуть ознаки, помилково класифікованих як несучих ознаку), тим краща якість класифікатора. Тому можемо ввести наступну метрику, що оцінює якість роботи класифікатора, що обчислює ймовірність приналежності об’єкта до позитивного класу (ф. 3.6) [41]:

$$\mathrm{AUC} = \int_0^1 \mathrm{TPRdFPR} \tag{3.6}$$

де TPR – повнота; FPR – специфічність.

Це значення є площею під графіком ROC-кривої (area under curve, AUC) та має такі характеристики:

– $\mathrm{AUC} \in [0,1]$;

– якщо AUC = 1, це означає, що класифікатор ідеально розділяє класи;

– класифікатор з AUC = 1/2 еквівалентний класифікатору, який оцінює ймовірність приналежності об’єкта до позитивного класу як випадкове число від 0 до 1 (рівномірно розподілене, і ніяк не залежить від x). Звідси якщо AUC < 1/2 то класифікатор $b(x) = 1 - a(x)$ прогнозує клас об’єкта x краще ніж $a(x)$ (тут мається на увазі, що $a(x)$ і $b(x)$ повертають ймовірність приналежності до позитивного класу).

– AUC дорівнює ймовірності того, що випадково обраний об’єкт позитивного класу виявиться в відсортованому списку правіше випадково обраного об’єкта негативного класу. [42]

Нехай класифікатор видає на вибірці $x_1, x_2, \dots, x_N$ ймовірності $p_1, p_2, \dots, p_N$ відповідно. Відсортуємо ймовірності в порядку зростання: $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(N)}$, і цим можливостям відповідають об’єкти $x_{(1)}, x_{(2)}, \dots, x_{(N)}$ і мітки класів

$y_{(1)}, y_{(2)}, \ldots, y_{(N)}$ відповідно. Тоді можна вивести таку формулу для обчислення AUC-ROC метрики на практиці [42]:

$$AUC = \frac{1}{N^{+}N^{-}} \sum_{i<j} \mathbb{I}\left[y_{(i)} < y_{(j)}\right] \tag{3.7}$$

де N^{+}, N^{-} – кількість об'єктів в позитивному і негативному класах відповідно.

3.3 Модель розпізнавання епілептичної активності

Головною задачею даного дослідження є побудова моделі для автоматичного розпізнавання та прогнозування появи патологічних частотних ритмів в сигналах ЕЕГ за допомогою алгоритмів штучного інтелекту, до моменту початку нападу (рис. 3.3).

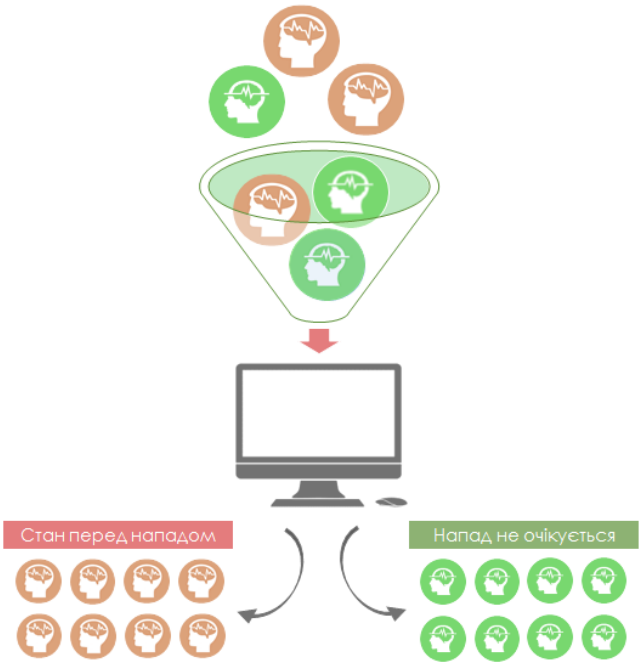

Рисунок 3.3 – Схематичне зображення задачі розпізнавання

Етапи реалізації такої моделі детально описані в наступних пунктах цього розділу.

3.3.1 Попередня обробка даних

Для дослідження була отримана база даних з 194 сигналів з епілептиформною активністю та 194 сигналів нормальної активності мозку, з рівною кількістю каналів для кожного сигналу.

На першому етапі було виділено три класи даних:

– клас 1 – відрізки сигналу ЕЕГ, в яких була зареєстрована епілептиформна активність та додаткові 30 секунд стану перед нападом (іктальний та преіктальний стан);

– клас 2 – відрізки сигналу ЕЕГ, 30 секунд стану перед нападом (преіктальний стан);

– клас 3 – сигнали з нормальною активністю головного мозку.

Тож для порівняльної характеристики роботи алгоритмів було сформовано дві вибірки. Вибірка №1 складається з сигналів ЕЕГ нормального стану пацієнта (клас 3) та сегментів сигналів ЕЕГ, в яких була зареєстрована патологічна активність разом з преіктальним станом (30 секунд до нападу) (клас 1). Вибірка №2 складається з сигналів ЕЕГ нормального стану пацієнта (клас 3) та сегментів преіктального стану сигналів ЕЕГ, в яких була зареєстрована епілептиформна активність (клас 2). Саме вибірка №2 є задачею прогнозування.

На рис. 3.4 схематично зображені набори даних, які були використані у дослідженні.

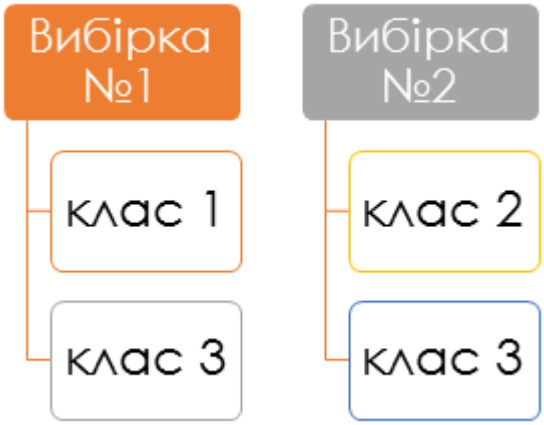

Рисунок 3.4 – Набори даних, які використовувались у дослідженні

Спершу сигнали без патологічної активності були поділені на відрізки тривалістю 30 секунд. Для зведення даних до єдиної розмірності, так як тривалість нападів варіюється від десятків секунд до декількох хвилин, всі сигнали були поділені на епохи. Тривалістю епохи було обрано 10 секунд.

3.3.2 Виділення ознак

Так як сигнал електроенцефалограми є числовим рядом, то для його класифікації необхідно підібрати ознаки, для його характеристики. Тож, для перетворення неперервних відрізків ЕЕГ-сигналів у комплексний інформативний набір даних, розраховувались частотні та просторово-часові характеристики для кожної епохи, що схематично зображено на рис. 3.5. Набір ознак складався з таких характеристик:

1) частота спектрального краю;
2) матриця кореляції між частотними смугами та її власні значення;
3) параметри Hjorth (активність, мобільність, і складність).

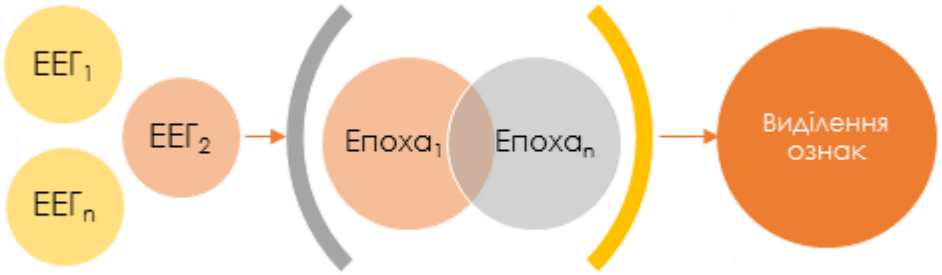

Рисунок 3.5 – Блок-схема аналізу ЕЕГ-сигналів

Матриця кореляції та параметри Hjorth розраховувались для кожного каналу окремо, їх було 23. В результаті чого, загальна розмірність вибірки № 1 склала 1566 об’єктів × 93 ознак, та вибірка № 2 – 1044 об’єктів × 93 ознак (табл. 3.2).

Таблиця 3.2 – Розмірність вхідних даних

Набір даних	Об’єктів	Ознак
Вибірка №1	1566	93
Вибірка №2	1044	93

Набір ознак був сформований на основі результатів бакалаврської роботи на тему «Класифікація патологічних сигналів ЕЕГ за допомогою методів машинного навчання». Були обрані такі характеристики, які мали найбільший ваговий коефіцієнт при класифікації нормальних сигналів та епілептичних приступів. Який був використаний для зменшення кількості ознак та виділення найінформативніших, за допомогою функції SelectFromModel, яка трансформує вибірку на основі важливості ознак, оцінених методом Extra trees. Дана функціє є однією з переваг алгоритмів класифікації на основі дерев.

3.3.3 Класифікація ЕЕГ-сигналів

Задача класифікації полягає в тому що є безліч об’єктів, в даному випадку сигналів, розділених деяким чином на класи (передепілептична та нормальна активність) – двокласова класифікація. Визначена скінченна множина об’єктів, для яких відомо, до яких класів вони належать. Така множина називається навчальною вибіркою. Класова приналежність інших об’єктів не відома – тестова вибірка. Потрібно побудувати алгоритм, здатний класифікувати довільний об’єкт з тестової вибірки. У машинному навчанні завдання класифікації відноситься до розділу навчання з учителем.

На рис. 3.6 зображена блок-схема реалізації програмної моделі класифікації. На етапі навчання дані з попередньою обробкою, якщо необхідно, подаються на вхід моделі. Далі розрахунок ознак (часові, частотні, та частотно-часові характеристики сигналу). Отриманий набір ознак використовується для навчання алгоритму, який на даному етапі знає правильні відповіді класової приналежності об’єктів навчальної вибірки. На етапі прогнозування, алгоритм отримує лише вхідні дані, тобто тестову вибірку, та вирішує до якого класу належить об’єкт.

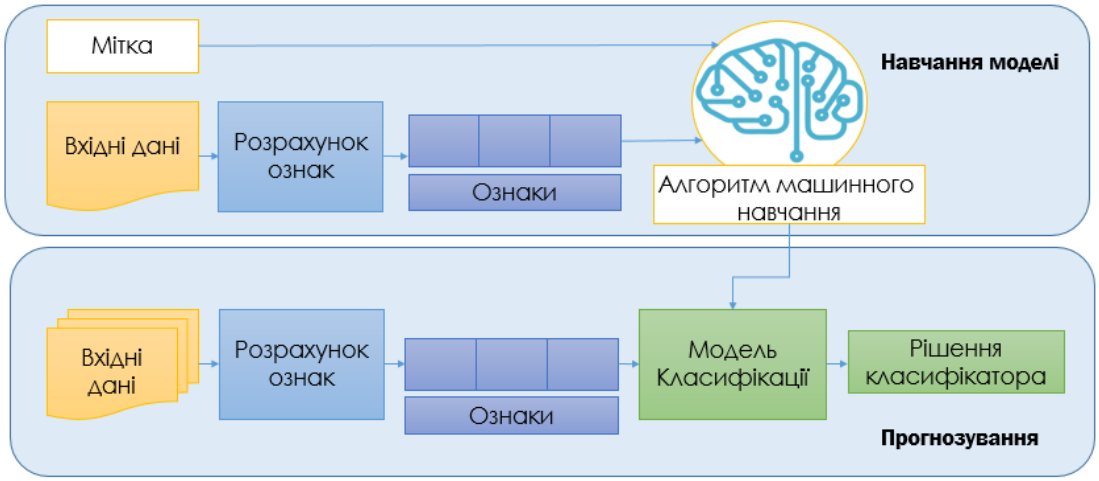

Рисунок 3.6 – Блок-схема етапів реалізації програмної моделі

На основі літературного аналізу та бакалаврської роботи були обрані наступні методи машинного навчання, які безпосередньо вирішують задачу класифікації. Метод опорних векторів – був використаний в попередніх дослідженнях, тому для порівняння цікавим є його результат розпізнавання на новому наборі даних. Наївний Баєсів класифікатор є одним із найстарших алгоритмів штучного інтелекту, який до сьогодні залишаться ефективним інструментом для багатьох дослідників науковців по всьому світі. В противагу йому був використаний більш новий та удосконалений ансамблевий метод градієнтного бустингу над вирішальними деревами. Результати розпізнавання епілептичної активності саме цими трьома алгоритмами будуть оцінені на двох наборах даних.

Всі розрахунки та аналіз даних в даній роботі виконувались за допомогою потужного середовища програмування Python 2.7.15, використовуючи бібліотеки numpy, pandas, scikit-learn, matplotlib [19] та веб-додаток jupyter notebook [20].

Перед тим як подавати набір даних на вхід алгоритму для класифікації, необхідно розділити загальну вибірку на навчальну, на якій безпосередньо здійснюватиметься навчання, та тестову, об’єкти якої, алгоритм самостійно розподіляє між класами. Такий поділ вибірки виконується функціями крос-валідації. Для цього була використана функція крос-валідації train_test_split, з параметрами: X – загальний набір даних, Y – цільова змінна, розмір тестового набору (test_size) – 20% від загального. Перші результати розпізнавання для трьох алгоритмів представлені в таблиці 3.3.

Метод опорних векторів був реалізований за допомогою функції svm.SVC з ядром типу «poly». Точність розпізнавання класу 1 з патологічними частотами склала 82% правильних відповідей, та класу 2, нормальної активності – 95%. На рисунках 3.7 та 3.8 зображені матриці неточностей для класифікатора на основі методу опорних векторів.

Таблиця 3.3 – Результати розпізнавання

Алгоритм	Вибірка № 1			Вибірка № 2		
	Загальна точність (accuracy)	Точність в межах класу 1	Точність в межах класу 3	Загальна точність (accuracy)	Точність в межах класу 2	Точність в межах класу 3
Метод опорних векторів	90 %	82%	95%	72%	68%	79%
Наївний Баєсів класифікатор	67%	44%	91%	42%	82%	9%
Градієнтний бустинг	**100%**	100%	100%	**87%**	82%	90%

Головна діагональ матриці неточностей показує кількість об’єктів для яких рішення класифікатора та реальне значення класу співпали.

Тож, класифікатор на основі методу опорних векторів із загальної кількості тестової вибірки – 314 об’єктів (20% від загальної кількості вибірки №1), правильно класифікував 284 об’єкта.

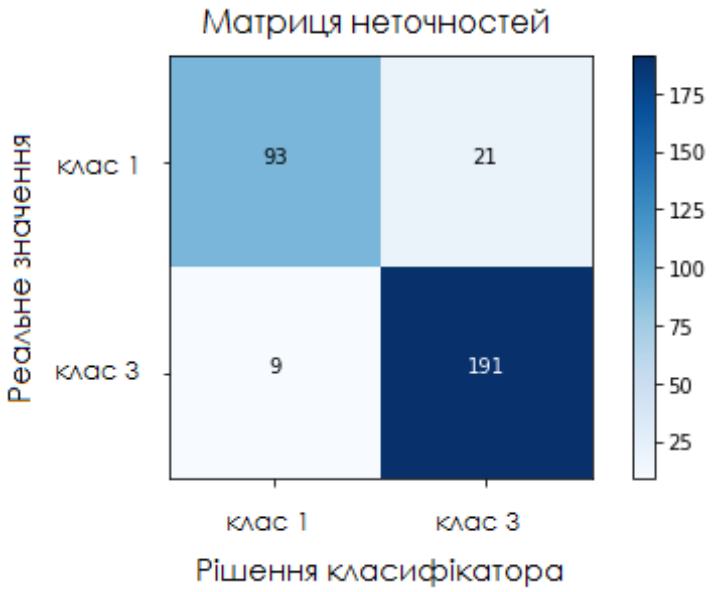

Рисунок 3.7 – Матриця неточностей для методу опорних векторів

Помилково 9 об’єктів класу 3 були віднесені до класу 1, та 21 об’єкт класу 1 до класу 3.

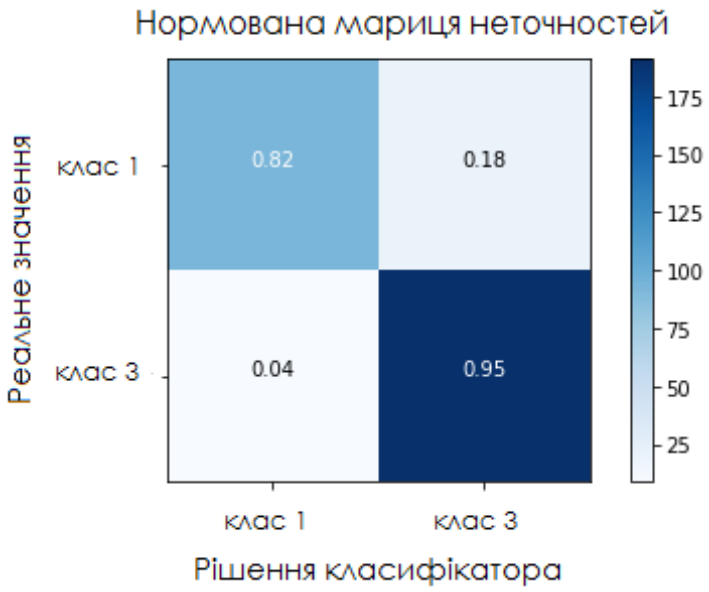

Рисунок 3.8 – Нормована матриця неточностей для методу опорних векторів

Головна діагональ нормованої матриці неточностей демонструє точність класифікації в межах класу.

На рис. 3.9 та 3.10 зображені матриці неточностей для Наївного Баєсівого класифікатора.

Рисунок 3.9 – Матриця неточностей для Наївного Баєсівого класифікатора

Загальна точність класифікації за допомогою НБК склала 67% правильних рішень класифікатора. Проте, аналізуючи результати на рис. 3.9 можна зробити висновки, що Наївний Баєсів класифікатор з високою точністю розпізнає клас з нормальною активністю – 91%, але в той же час помилково 64 об’єкта із 114 класу 3 відносить до класу 1. В результаті чого, точність класифікації в межах класу 1 склала 44%.

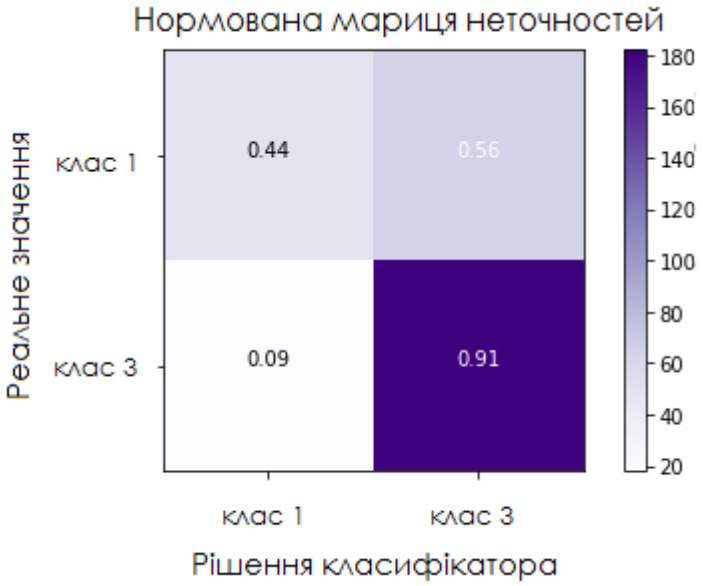

Рисунок 3.10 – Нормована матриця неточностей для Наївного Баєсівого класифікатора

На рис. 3.11 та 3.12 зображені матриці неточностей для методу градієнтного бустингу над вирішальними деревами.

Загальна точність розпізнавання та в межах класу склала 100%, тобто всі об’єкти були класифіковані правильно. Такий результат може бути або чудовим показником роботи алгоритму, або ж проблемою перенавчання. Коли модель добре пояснює тільки приклади з навчальної вибірки, адаптуючись до навчальних прикладів, замість того щоб вчитися класифікувати об’єкти, які не брали участі в навчанні (втрачаючи здатність до узагальнення).

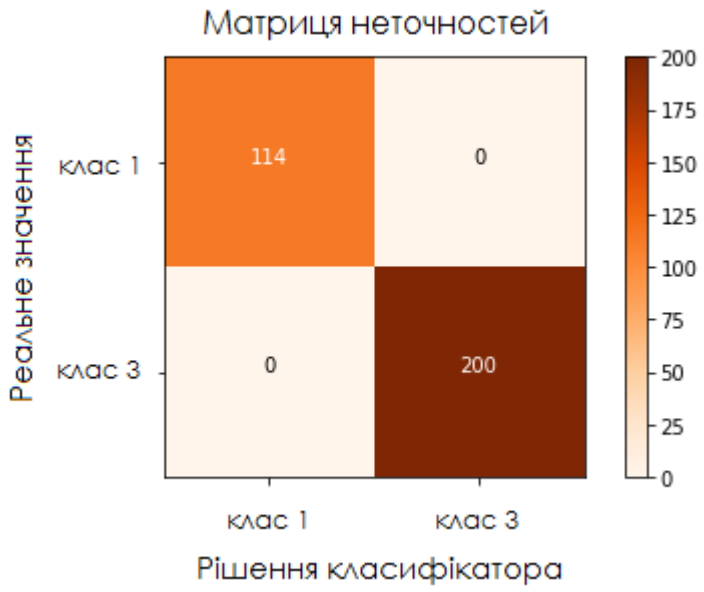

Рисунок 3.11 – Матриця неточностей для методу градієнтного бустингу над вирішальними деревами

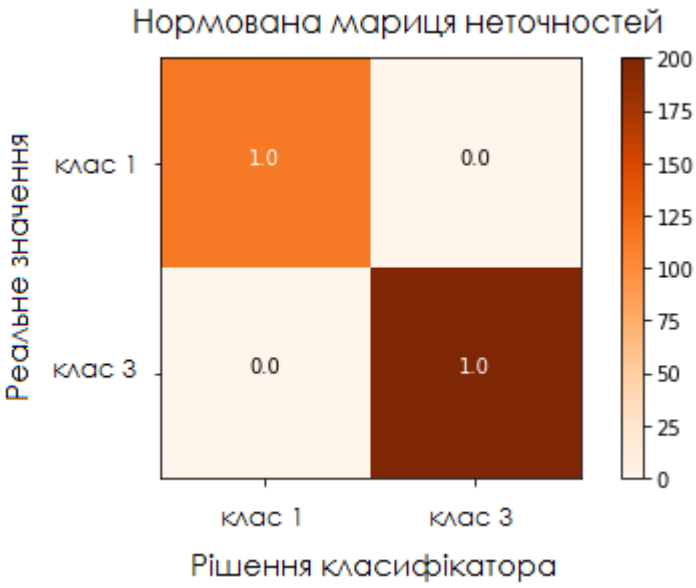

Рисунок 3.12 – Нормована матриця неточностей для методу градієнтного бустингу над вирішальними деревами

Впевнитись в природі такого результату можуть допомогти показники класифікації вибірки №2, яка містить в собі менш диференційовані дані, адже не

містить в собі явно виражені та різкі частотні зміни. Така класифікація є складнішою для алгоритмів, адже необхідно розпізнати зачатки патологічної активності, які не помітні для людського зору. Але вже не є нормальною активністю головного мозку.

На рис. 3.14 та 3.15 зображено кількість правильно класифікованих об’єктів, та міжкласова точність відповідно, методом опорних векторів для вибірки №2.

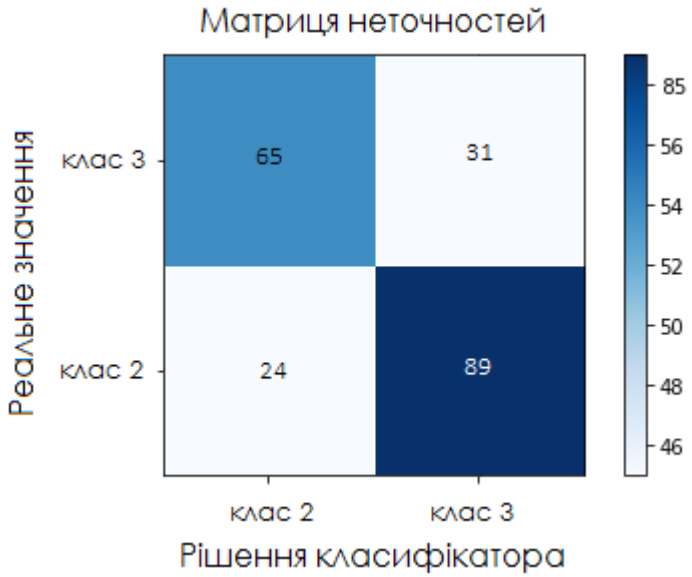

Рисунок 3.14 – Матриця неточностей для методу опорних векторів, вибірка №2

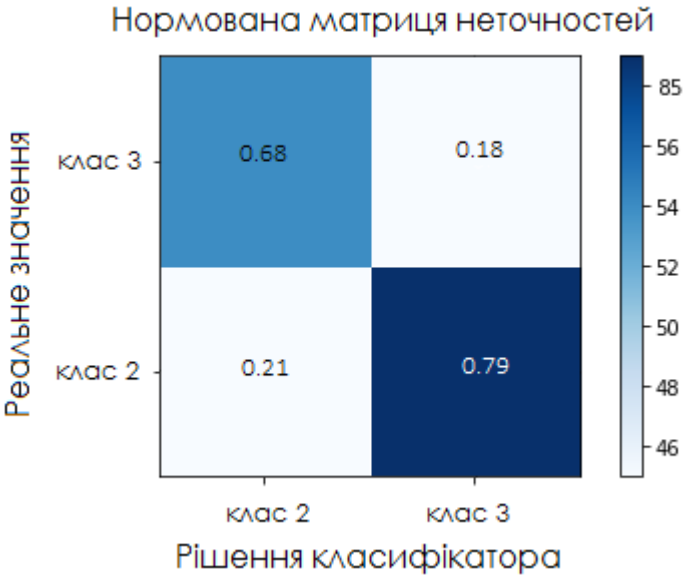

Рисунок 3.15 – Нормована матриця неточностей для методу опорних векторів, вибірка №2

Загальна точність для склала 72%.

Наївний Баєсів класифікатор не зміг розпізнати відмінності між об’єктами класів 2 і 3. Помилково 91% об’єктів з нормальною активністю були віднесені до класу з передепілептичним станом (рис. 3.16, 3.17).

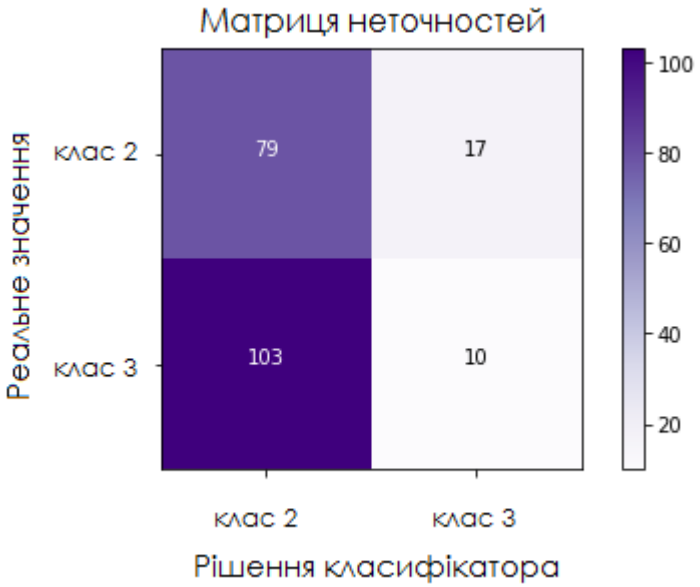

Рисунок 3.16 – Матриця неточностей для НБК, вибірка №2

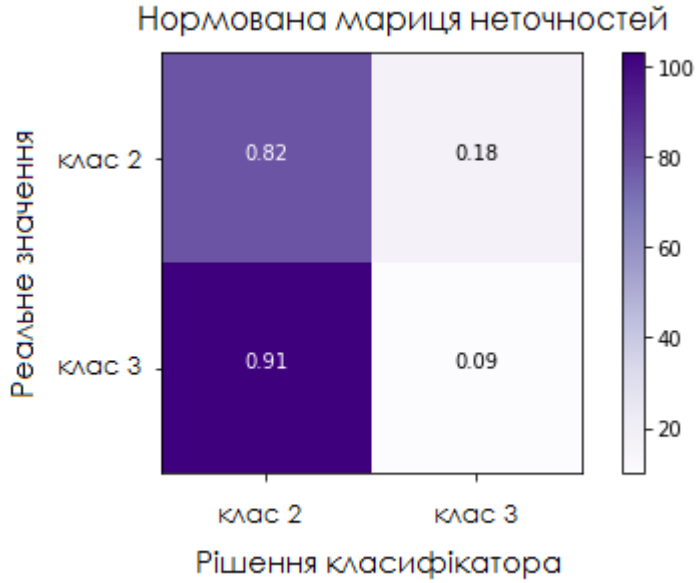

Рисунок 3.17 – Нормована матриця неточностей для НБК, вибірка №2

Натомість градієнтний бустинг з загальною точністю класифікації в 87%, правильно класифікував 79 об’єктів класу 2 із 96, та 102 об’єкта із класу 3 із 113 (рис. 3.18), міжкласова точність досягла значень 82% та 90% відповідно (рис. 3.19).

Рисунок 3.18 – Матриця неточностей для методу градієнтного бустингу, вибірка №2

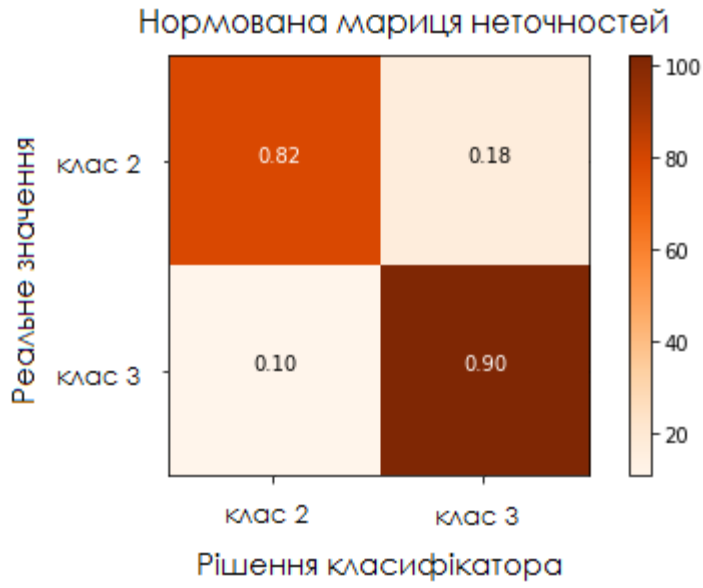

Рисунок 3.19 – Нормована матриця неточностей для методу градієнтного бустингу, вибірка №2

Для вибірки №2 навчальний набір даних склав 835 об’єктів, а тестовий – 209 об’єктів.

На рис. 3.20 зображені три типи розподілу вхідних даних та площини, за допомогою яких алгоритми розподіляють об’єкти на класи. За допомогою цього рисунку можна наочно оцінити розподіл об’єктів між класами.

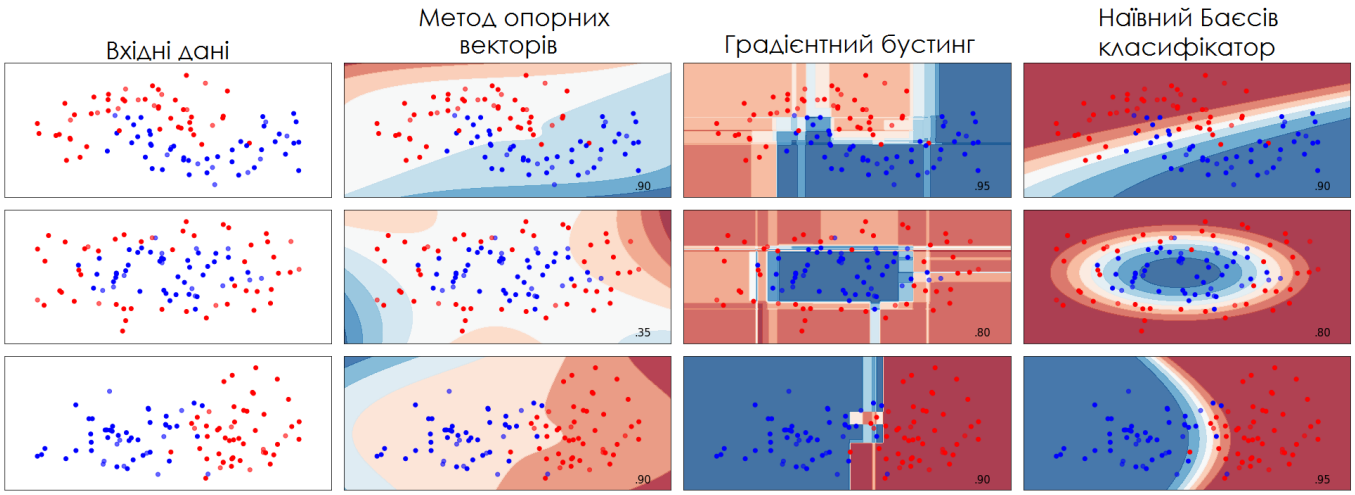

Рисунок 3.20 – Розділяючі площини алгоритмів

Як бачимо Наївний Баєсів класифікатор, містить найбільшу кількість об’єктів, поза межами «своїх» класів. Дещо нечітко проходить розділяюча площина між класами у методу опорних векторів, на її межі також бачимо деякі похибки. Найчіткіше розділяюча площина нарисована у методу градієнтного бустингу. Найдостовірніше має вигляд метод опорних векторів, адже площина між класами проходить достатньо чітко, але повністю не повторює криву між класами, що є ознакою відсутності ефекту перенавчання.

За допомогою функції VotingClassifier був побудований ансамбль двох класифікаторів, методу опорних векторів та градієнтного бустингу (рис. 3.21), які продемонстрували однаково високий результат окремо.

Рисунок 3.21 – Блок-схема класифікації за допомогою ансамблю методів

Точність розпізнавання за допомогою ансамблю склала 80%.

Також було проведено порівняння результатів методу опорних векторів. На діаграмі (рис. 3.22) показано точність правильних відповідей алгоритму на трьох наборах даних: вибірка №1, вибірка №2 та вибірка №3 (дослідження бакалаврської роботи), яка складається з сигналів ЕЕГ без патологічної активності та сегментів лише з епілептичною (іктальною) активністю.

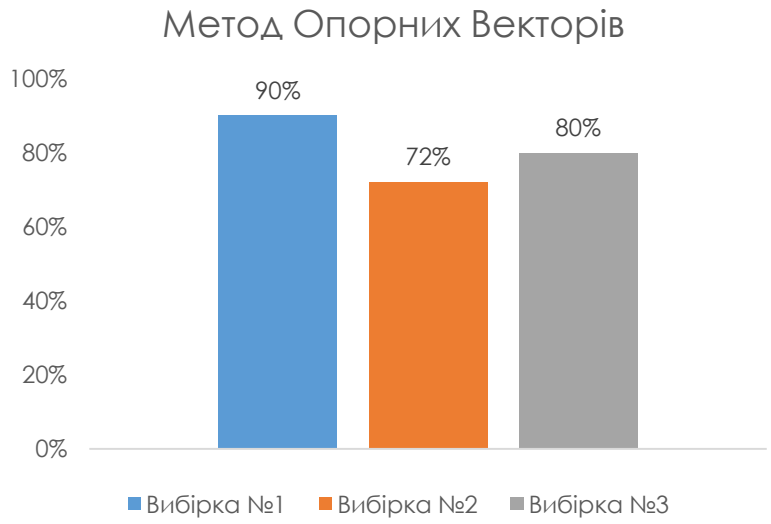

Рисунок 3.22 – Результативність методу опорних векторів в ході 2-х досліджень

Аналізуючи отримані показники точності, можна зробити висновки, що найкращий результат класифікації був отриманий з використанням вибірки №1, яка в свою чергу є найбільшим набором даних, а вибірка №2 навпаки, є найменшою за чисельною кількістю. Тож, зі збільшенням набору даних, результативність роботи алгоритму також зростає.

Висновки до розділу 3

У даному розділі був проведених аналіз ЕЕГ-сигналів та їх попередня обробка, в результаті якої було виділено три класи даних та сформовано дві вибірки для порівняльного аналізу.

На основі літературного огляду та результатів бакаврської роботи, був відібраний найефективніший набір ознак для розпізнавання та прогнозування епілептиформної активності, а саме: частота спектрального краю, матриця кореляції між частотними смугами та параметри Hjorth (активність, мобільність, і складність). Також були обрані такі методи машинного навчання як: метод опорних векторів, наївний Баєсів класифікатор та метод градієнтного бустингу над вирішальними деревами.

Наївний Баєсів класифікатор не зміг розділити мало диференційовані данні двох класів, та класифікував більшість об’єктів класу з нормальною активністю до патологічної. Методи опорних векторів та градієнтного бустингу продемонстрували хорошу точність на обох наборах вхідних даних в 72% та 87% відповідно.

Зниження співпадінь у відповідях алгоритмів та реальних значеннях класів об’єктів може бути пояснене тим, що вибірка №2 містить мало диференційовані дані та є найскладнішою для класифікації, адже відмінність між класами в яких ледь помітна, та зовсім не видима для людського зору.

ВИСНОВКИ

В даній роботі проаналізований сучасний стан вивчення та застосування сигналів ЕЕГ у наукових дослідженнях, використовуючи методи машинного навчання. В результаті чого були виділені основні напрямки розвитку досліджень: вдосконалення функціонування систем нейрокомп'ютерного інтерфейсу, діагностика та прогнозування епілепсії, дослідження сну та вдосконалення існуючих методів аналізу ЕЕГ сигналів штучним інтелектом.

Для вирішення задачі класифікації в даному дослідженні були обрані такі методи: метод опорних векторів, наївний Баєсів класифікатор та метод градієнтного бустингу над вирішальними деревами.

Із бази, яка складається з 194 ЕЕГ-сигналів з епілептиформною активністю та 194 сигналів нормальної активності мозку, в результаті попередньої обробки були виділені три класи даних: ділянки з преіктальною та іктальною активністю, окремо преіктальні сегменти та сигнали без патологічних ритмів, які в свою чергу були поділені на відрізки тривалістю 30 секунд. Для зведення даних до єдиної розмірності всі відрізки були поділені на епохи, тривалістю 10 секунд.

Для кожного каналу та кожної епохи розраховувався набір ознак, який був відібраний в результаті літературного огляду та на основі досліджень бакаврської роботи. Були обрані ознаки з найбільшим ваговим коефіцієнтом, із загального набору із попередніх досліджень:

- частота спектрального краю;
- матриця кореляції між частотними смугами;
- параметрів Hjorth (активність, мобільність, і складність);

Ознаки, які були відфільтровані: спектральна та ентропія Шеннона на шести частотних смугах (дельта (0,1-4 Гц), тета (4-8 Гц), альфа (8-12 Гц), бета (12-30 Гц) та гама-ритм (в діапазоні 30-70 Гц та 70-180 Гц)); ентропія Шеннона для діадних

частотних смуг; матриця кореляції між каналами ЕЕГ-сигналу; фрактальна розмірність; статистичні характеристики (асиметрія, коефіцієнт ексцесу, середнє значення для кожного каналу); перетин нуля (Zero Crossing).

Були отримані такі результати класифікації:

– вибірка №1: НБК – 67%, метод опорних векторів – 90%, метод градієнтного бустингу – 100%;

– вибірка №2: НБК – 42%, метод опорних векторів – 72%, метод градієнтного бустингу – 87%.

Зі збільшенням кількості об’єктів у вибірці, точність методу опорних векторів також зростає.

Фінальний результат точності був отриманий за допомогою ансамблю методів опорних векторів та градієнтного бустингу, не враховуючи наївний Баєсів класифікатор, адже цей алгоритм не зміг прийняти адекватні рішення при класифікації об’єктів двох класів ЕЕГ сигналів. Результуюча точність програмної моделі розпізнавання епілептичних частот з використанням методів штучного інтелекту склала 80%.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Behri M. Comparison of machine learning methods for two class motor imagery tasks using EEG in brain-computer interface / M. Behri, A. Subasi, S. Qaisar. // 2018 Advances in Science and Engineering Technology International Conferences (ASET). – 6 Feb. - 5 April 2018, Abu Dhabi, United Arab Emirates/IEEE

2. Motor Imagery EEG Signal Processing and Classification Using Machine Learning Approach / [Sreeja. S. R, J. Rabha, K. Y. Nagarjuna and other]. // 2017 International Conference on New Trends in Computing Sciences (ICTCS). – 11-13 Oct. 2017, Amman, Jordan/IEEE

3. Motor imagery EEG classification with optimal subset of wavelet based common spatial pattern and kernel extreme learning machine / H.Park, J. Kim, B. Min, B. Lee. // 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). – 11-15 July 2017, Seogwipo, South Korea/ IEEE

4. Md. Khayrul Bashar Human identification from brain EEG signals using advanced machine learning method EEG-based biometrics / Md. B. Khayrul, I. Chiaki, H. Yoshida. // 2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES) – 4-8 Dec. 2016, Kuala Lumpur, Malaysia/ IEEE – P. 475 - 479.

5. Brain machine interface for useful human interaction via extreme learning machine and state machine design / [G. Sargent, H. Zhang, A. Morgan та ін.]. // 2017 IEEE Symposium Series on Computational Intelligence (SSCI) – 27 Nov. -1 Dec. 2017, Honolulu, HI, USA/IEEE

6. Keerthika P. Eye state EEG signal classification using complex valued neural classifiers / P. Keerthika, M. Sivachitra, M. Ponni Bala// 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS) – 17-18 March 2017, Coimbatore, India/ IEEE

7. Motion velocity estimation from electroencephalography signals with extreme learning machine / L.Su, L. Bi, W. Fei, J. Lian. // 2017 Chinese Automation Congress (CAC) – 20-22 Oct. 2017, Jinan, China/ IEEE – Page(s): 4901 - 4905

8. C. Djamal E. EEG based emotion monitoring using wavelet and learning vector quantization / E. C. Djamal, P. Lodaya// 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) – 19-21 Sept. 2017, Yogyakarta, Indonesia/ IEEE – Page(s): 1 - 6

9. Ammar S. Seizure detection with single-channel EEG using Extreme Learning Machine / S. Ammar, M. Senouci. // 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA). – 19-21 Dec. 2016, Sousse, Tunisia/IEEE

10. Unsupervised detection of epileptic seizures from EEG signals: A channel-specific analysis of long-term recordings / [K. M. Tsiouris, S. Konitsiotis, S. Markoula and other]. // 2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI). – 4-7 March 2018, Las Vegas, NV, USA/IEEE

11. Rajaguru H. Analysis of adaboost classifier from compressed EEG features for epilepsy detection / H. Rajaguru, S. Prabhakar. // 2017 International Conference on Computing Methodologies and Communication (ICCMC). – 18-19 July 2017, Erode, India/IEEE

12. Dutta K.K. Removal of muscle artifacts from EEG based on ensemble empirical mode decomposition and classification of seizure using machine learning techniques / K.K. Dutta, K. Venugopal, S.A. Swamy. //2017 International Conference on Inventive Computing and Informatics (ICICI) – 23-24 Nov. 2017, Coimbatore, India/IEEE – Page(s): 861 - 866

13. Imah E.M. A comparative study of machine learning algorithms for epileptic seizure classification on EEG signals / E.M. Imah, A. Widodo. // 2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 28-29 Oct. 2017, Bali, Indonesia/ IEEE – Page(s): 401 - 408

14. Epileptic seizure detection using discrete wavelet transform based support vector machine / P.Deshmukh, R. Ingle, V. Kehri, R. Awale. // 2017 International Conference on Communication and Signal Processing (ICCSP), 6-8 April 2017, Chennai, India/ IEEE. – Page(s): 1933 - 1937

15. Epileptic seizure auto-detection using deep learning method / Yuzhen Cao, Yixiang Guo, Hui Yu, Xuyao Yu. // 2017 4th International Conference on Systems and Informatics (ICSAI). – 11-13 Nov. 2017, Hangzhou, China/IEEE

16. Ebrahim A. Mattar. Machine learning for electroencephalography decoding and robotics dextrous hands movement / Ebrahim A. Mattar, Hessa Jassim Al-Junaid. // 2018 International Conference on Power, Signals, Control and Computation (EPSCICON). – 6-10 Jan. 2018, Thrissur, India/IEEE

17. Artifact detection in EEG using machine learning / [E. Nedelcu, R. Portase, R. Tolas and others]. // 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 7-9 Sept. 2017, Cluj-Napoca, Romania/ IEEE. – Page(s): 77 - 83.

18. Automatic Sleep Stage Classification Applying Machine Learning Algorithms on EEG Recordings / [P. Chriskos, D. Kaitalidou, G. Karakasis and others]. //2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS), 22-24 June 2017, Thessaloniki, Greece/ IEEE. – Page(s): 435 - 439.

19. Qureshi S. Evaluate different machine learning techniques for classifying sleep stages on single-channel EEG / S. Qureshi, S. Vanichayobon. // 2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE), 12-14 July 2017, Nakhon Si Thammarat, Thailand/ IEEE. – Page(s): 1 - 6.

20. Zhang Z. An accurate sleep staging system with novel feature generation and auto-mapping / Z. Zhang, C. Guan// 2017 International Conference on Orange Technologies (ICOT), 8-10 Dec. 2017, Singapore, Singapore/ IEEE. – Page(s): 214 - 217.

21. Vilamala A. Deep convolutional neural networks for interpretable analysis of EEG sleep stage scoring / A. Vilamala, L. Hansen, K. Madsen. //2017 IEEE 27th International

Workshop on Machine Learning for Signal Processing (MLSP), 25-28 Sept. 2017, Tokyo, Japan/ IEEE. – Page(s): 1 - 6.

22. Месюра В. І. Розробка теоретико-методичних засад оцінювання правдивості новинної інформації. визначення неправдивих новин за допомогою наївного баєсівого класифікатора / В. І. Месюра, М. О. Гранік. //Матеріали XLVII науково-технічної конференції підрозділів ВНТУ, Вінниця, 14-23 березня 2018 р.

23. 6 простых шагов для освоения наивного байесовского алгоритма (с примером кода на Python) [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/ (дата звернення 18.10.2018).

24. How the naive bayes classifier works in machine learning [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: http://dataaspirant.com/2017/02/06/ naive-bayes-classifier-machine-learning/ (дата звернення 18.10.2018).

25. Онищук Р. Наївний баєсівський класифікатор [Електронний ресурс] / Р. Онищук // Фізико-технічний інститут, Кафедра інформаційної безпеки – Режим доступу до ресурсу: http://is.ipt.kpi.ua/wp-content/uploads/sites/4/2015/06/ Onyshchuk_Presentation.pdf (дата звернення 18.10.2018).

26. Наивный байесовский классификатор [Електронний ресурс] – Режим доступу до ресурсу: http://www.machinelearning.ru/wiki/index.php?title=Наивный_ байесовский_классификатор (дата звернення 18.10.2018).

27. Байесовский классификатор [Електронний ресурс] – Режим доступу до ресурсу: http://www.machinelearning.ru/wiki/index.php?title=Байесовский_классификатор (дата звернення 18.10.2018).

28. Вялый Е. Ю. Суррогат файла: статический анализ файловой системы / Е. Ю. Вялый. // Московский физико-технический институт. – 2014.

29. Бустинг [Електронний ресурс] – Режим доступу до ресурсу: http://www.machinelearning.ru/wiki/index.php?title=Бустинг (дата звернення 18.10.2018).

30. Фонарев А. Ю. Обзор алгоритмов бустинга [Електронний ресурс] / А. Ю. Фонарев // Московский Государственный Университет имени М.В. Ломоносова. – 2012. – Режим доступу до ресурсу: http://www.machinelearning.ru/wiki/images/9/9a/Fonarev.Overview_of_Boosting_Methods.pdf (дата звернення 18.10.2018).

31. Ермаков П. Д. Исследование методов машинного обучения в задаче автоматического определения тональности текстов на естественном языке / П. Д. Ермаков, Р. В. Федянин. // Новые информационные технологии в автоматизированных системах. – 2015. – №18. – С. 600 – 616.

32. Воронцов К. В. Лекции по методу опорных векторов / К. В. Воронцов. – 2007.

33. Guazzelli A. Predictive modeling techniques [Електронний ресурс] / Alex Guazzelli. – 2012. – Режим доступу до ресурсу: https://www.ibm.com/developerworks/library/ba-predictive-analytics2/index.html.

34. Марченко О. П. Конференция, посвященная применению метода независимых компонент к ЭЭГ и ССП, – Первый Симпозиум INTERBRAIN / О. П. Марченко. // Экспериментальная психология. – 2011. – №4. – С. 130–134.

35. Смирнов Н. Кластеризация компонент ЭЭГ [Електронний ресурс] / Н. Смирнов // Лаборатория НГУ-Интел – 2013. – Режим доступу до ресурсу: http://conf.ict.nsc.ru/files/conferences/ictm-2013/presentation/168013/168014/2013-06-07_SmirnovN.pdf (дата звернення 18.10.2018).

36. Захаров И. С. Компонентный анализ в задаче слепого разделения спонтанной электроэнцефалограммы / И. С. Захаров. // Молодежный научно-технический вестник. – 2013.

37. Жуков А. А. Классификация движений в однонаправленном нейрокомпьтерном интерфейсе / А. А. Жуков. // Московский физико-технический институт. – 2015.

38. CHB-MIT Scalp EEG Database [Електронний ресурс] // PhysioNet – Режим доступу до ресурсу: https://physionet.org/pn6/chbmit/ (дата звернення 19.10.2018).

39. Shoeb A. Application of Machine Learning To Epileptic Seizure Detection / A. Shoeb, J. Guttag// 27th International Conference on Machine Learning (ICML), June 21-24, 2010, Haifa, Israel.

40. Оценка классификатора (точность, полнота, F-мера) [Електронний ресурс]. – 2012. – Режим доступу до ресурса: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html (дата звернення 28.11.2018).

41. Brownlee J. What is a Confusion Matrix in Machine Learning [Електронний ресурс] / J. Brownlee. – 2016. – Режим доступу до ресурса: https://machinelearningmastery.com/confusion-matrix-machine-learning/ (дата звернення 28.11.2018).

42. AUC-ROC [Електронний ресурс] – Режим доступу до ресурсу: http://ru.learnmachinelearning.wikia.com/wiki/AUC-ROC (дата звернення 28.11.2018).

43. Pedregosa F. Scikit-learn: Machine Learning in Python / [F. Pedregosa, G. Varoquaux, A. Gramfort and other]. // Journal of Machine Learning Research. – 2011. – Page(s): 2825-2830.

44. The Jupyter Notebook [Електронний ресурс] – Режим доступу до ресурсу: http://jupyter.org/. (дата звернення 28.11.2018).

Додаток А

Виділення ознак

```
import sys
import os
import numpy as np
import pandas as pd
from math import *
from scipy.io import loadmat
from scipy.stats import skew, kurtosis

def corr(data,type_corr):
    C = np.array(data.corr(type_corr))
    C[np.isnan(C)] = 0
    C[np.isinf(C)] = 0
    w,v = np.linalg.eig(C)
    #print(w)
    x = np.sort(w)
    x = np.real(x)
    return x

def calculate_features(file_name): #, length_epoch
    eegData = pd.read_csv (file_name, header=None)
    #eegData = file_name
    #eegData.drop('Err', axis=1, inplace=True)
    #eegData.drop(eegData.index[0])
    fs = 256;
    [nt, nc] = eegData.shape
    print((nt, nc))
    subsampLen = floor(fs*10)
    numSamps = int(floor(nt/subsampLen));      # Num of 1-min
samples
    sampIdx =
np.arange(0,(numSamps+1)*(subsampLen),(subsampLen))
    #print(sampIdx)
    feat = []
    feat_ep=[]
    for i in range(1, numSamps+1):
```

```
        print('processing file {} epoch
{}'.format(file_name,i))
        epoch = eegData.ix[sampIdx[i-1]:sampIdx[i], :]

        # compute Shannon's entropy, spectral edge and
correlation matrix
        # segments corresponding to frequency bands
        lvl = np.array([0.1, 4, 8, 12, 30, 70, 180])
#Frequency levels in Hz
        lseg = np.round(nt/fs*lvl).astype('int')
        D = np.absolute(np.fft.fft(epoch, n=lseg[-1],
axis=0))
        D[0,:]=0                              # set the DC
component to zero
        D /= D.sum()                          # Normalize each
channel

        dspect = np.zeros((len(lvl)-1,nc))
        for j in range(len(dspect)):
            dspect[j,:] = 2*np.sum(D[lseg[j]:lseg[j+1],:],
axis=0)

        # Find the shannon's entropy
        spentropy = -
1*np.sum(np.multiply(dspect,np.log(dspect)), axis=0)

        # Find the spectral edge frequency
        sfreq = fs
        tfreq = 40
        ppow = 0.5

        topfreq = int(round(nt/sfreq*tfreq))+1
        A = np.cumsum(D[:topfreq,:])
        B = A - (A.max()*ppow)
        spedge = np.min(np.abs(B))
        spedge = (spedge - 1)/(topfreq-1)*tfreq

        # Calculate correlation matrix and its eigenvalues
(b/w channels)
        data = pd.DataFrame(data=epoch)
        type_corr = 'pearson'
        lxchannels = corr(data, type_corr)
```

```
        # Calculate correlation matrix and its eigenvalues
(b/w freq)
        data = pd.DataFrame(data=dspect)
        lxfreqbands = corr(data, type_corr)

        # Spectral entropy for dyadic bands
        # Find number of dyadic levels
        ldat = int(floor(nt/2.0))
        no_levels = int(floor(log(ldat,2.0)))
        seg = floor(ldat/pow(2.0, no_levels-1))

        # Find the power spectrum at each dyadic level
        dspect = np.zeros((no_levels,nc))
        for j in range(no_levels-1,-1,-1):
            dspect[j,:] =
2*np.sum(D[int(floor(ldat/2.0))+1:ldat,:], axis=0)
            ldat = int(floor(ldat/2.0))

        # Find the Shannon's entropy
        spentropyDyd = -
1*np.sum(np.multiply(dspect,np.log(dspect)), axis=0)

        # Find correlation between channels
        data = pd.DataFrame(data=dspect)
        lxchannelsDyd = corr(data, type_corr)

        # Fractal dimensions
        no_channels = nc
        #fd = np.zeros((2,no_channels))
        #for j in range(no_channels):
        #    fd[0,j] = pyeeg.pfd(epoch[:,j])
        #    fd[1,j] = pyeeg.hfd(epoch[:,j],3)
        #    fd[2,j] = pyeeg.hurst(epoch[:,j])

        # [mobility[j], complexity[j]] =
pyeeg.hjorth(epoch[:,j)
        # Hjorth parameters
        # Activity
        activity = np.var(epoch, axis=0)
        # print('Activity shape: {}'.format(activity.shape))
        # Mobility
```

```
        mobility = np.divide(
                             np.std(np.diff(epoch, axis=0)),
                             np.std(epoch, axis=0))
        # print('Mobility shape: {}'.format(mobility.shape))
        # Complexity
        complexity = np.divide(np.divide(
                                          # std of second
derivative for each channel

np.std(np.diff(np.diff(epoch, axis=0), axis=0), axis=0),
                                          # std of second
derivative for each channel
                                          np.std(np.diff(epoch,
axis=0), axis=0))
                                  , mobility)
        # print('Complexity shape:
{}'.format(complexity.shape))
        # Statistical properties
        # Skewness
        sk = skew(epoch)

        # Kurtosis
        kurt = kurtosis(epoch)

        # mean
        mean = epoch.mean()

        # zero crossing
        epoch = np.array(epoch)
        zerocrossing = ((epoch[:-1] * epoch[1:]) < 0).sum()

        # compile all the features
        feat = np.concatenate((
                                 #spentropy.ravel(),
                                 spedge.ravel(),
                                 #lxchannels.ravel(),
                                 lxfreqbands.ravel(),
                                 #spentropyDyd.ravel(),
                                 lxchannelsDyd.ravel(),
                                 #fd.ravel(),
                                 activity.ravel(),
                                 mobility.ravel(),
```

```
                                complexity.ravel(),
                                sk.ravel(),
                                kurt.ravel(),
                                mean.ravel(),
                                zerocrossing.ravel(),
                                 ))
        if len(feat_ep) == 0:
            feat_ep=feat
        else:
            feat_ep=np.vstack((feat_ep,feat))

    return feat_ep
```

Додаток Б

Програмна модель розпізнавання епілептичної активності

```
%matplotlib inline
import sys
import os
import numpy as np
import pandas as pd
from math import *
from scipy.io import loadmat
from scipy.stats import skew, kurtosis

# Data preprocessing
Data = []
path = 'E:\database_preseizure_60sec'
os.chdir(path)
Data_flag = 0
k_files = len([name for name in os.listdir('.') if
os.path.isfile(name)])
print k_files

for i in range (k_files-1):  #(k_files-1): (194-1)
    i+=1
    data = pd.read_csv ('preseizure_60sec_'+str(i)+'.dat',
header=None) #'Seizure_'+str(i)+'.csv'
    data = data.dropna(axis=0, how='any')#.T #Delete all rows
with NAN

    if not os.path.exists('preseizure_60sec_'+str(i)+'.dat'):
        continue
    if data.shape[0] == 24:
        data = data.drop(data.index[len(data)-1]) #Delete the
last row in dataframe=data
        Data_flag = True
    elif data.shape[0] == 18:
        continue
        Data_flag = False
    elif data.shape[0] == 28:
        data = data.drop(data.index[[4,9,12,17,22]])
        Data_flag = True
```

```
    elif data.shape[0] == 25:
        continue
        Data_flag = False
    elif data.shape[0] == 32:
        data =
data.drop(data.index[[10,24,25,26,27,28,29,30,31]])
#data=data[[0,4,6,8:22],:]
        Data_flag = True

    if data.shape[0] != 23:
        print 'File: preseizure_60sec_'+str(i)+'.dat' + '
Number of channels: '+ str(data.shape[0])

    data = data.T
    data.to_csv('preseizure_60sec_'+str(i)+'.csv',
header=None, index=False, index_label=False)

# database_seizures
fs = 256;
path = 'E:\database_seizures_csv'
os.chdir(path)
X_seizure=[]
for i in range (126-1): #(147): --> for seizure_60
    i+=1
    global str
    if not os.path.exists('Seizure_'+str(i)+'.csv'):
        continue
    if len(open('Seizure_'+str(i)+'.csv').readlines())/fs >=
10 and len(open('Seizure_'+str(i)+'.csv').readlines())/fs <=
150:
        if len(X_seizure) == 0:
            X_seizure =
calculate_features('Seizure_'+str(i)+'.csv')
        else:
            X_seizure =
np.vstack((X_seizure,calculate_features('Seizure_'+str(i)+'.c
sv')))

# database_ preseizure_30sec
fs = 256;
path = 'E:\database_preseizure_30sec'
os.chdir(path)
```

```
X_healthy_30 = []
for i in range(185-1):
    i+=1
    global str

    if not os.path.exists('preseizure_30sec_'+str(i)+'.csv'):
        continue
    if
len(open('preseizure_30sec_'+str(i)+'.csv').readlines())/fs
>= 10 and
len(open('preseizure_30sec_'+str(i)+'.csv').readlines())/fs
<= 150:
        if len(X_healthy_30) == 0:
            X_healthy_30 =
calculate_features('preseizure_30sec_'+str(i)+'.csv')
        else:
            X_healthy_30 =
np.vstack((X_healthy_30,calculate_features('preseizure_30sec_
'+str(i)+'.csv')))

# database_ norma
fs = 256;
path = 'E:\database_preseizure_60sec'
os.chdir(path)
X_healthy = []
for i in range(185-1):
    i+=1
    global str
    if not os.path.exists('preseizure_60-
30sec_'+str(i)+'.csv'):
        continue

    if len(open('preseizure_60-
30sec_'+str(i)+'.csv').readlines())/fs >= 10 and
len(open('preseizure_60-
30sec_'+str(i)+'.csv').readlines())/fs <= 150:
        if len(X_healthy) == 0:
            X_healthy = calculate_features('preseizure_60-
30sec_'+str(i)+'.csv')
        else:
```

```
            X_healthy =
np.vstack((X_healthy,calculate_features('preseizure_60-
30sec_'+str(i)+'.csv')))

y1 = np.ones((X_patology.shape[0]))
y0 = np.zeros((X_healthy.shape[0]))
Y = np.concatenate((y0,y1),axis=0)
print Y
print Y.shape

X = np.concatenate((X_patology,X_healthy),axis=0)
#print X
print X.shape

#метод независимых компонент
from sklearn.decomposition import FastICA
transformer = FastICA(n_components = 2, random_state = 0)
X_transformed = transformer.fit_transform(X)
X_transformed.shape

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2, random_state=0) #X_new

#naive_bayes
print(__doc__)
import itertools
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import BernoulliNB

# import some data to play with
target_names = np.array(['seizure', 'healthy'], dtype='|S10')
class_names = target_names

# Split the data into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2, random_state=0)
```

```
# Run classifier, using a model that is too regularized (C
too low) to see
# the impact on the results

classifier = BernoulliNB()
y_pred = classifier.fit(X_train, y_train).predict(X_test)

def accuracy(confusion_matrix):
    diagonal_sum = confusion_matrix.trace()
    sum_of_all_elements = confusion_matrix.sum()
    return diagonal_sum / sum_of_all_elements

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Purples): #Blues
Purples Greens Oranges
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:,
np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)
    print accuracy(cm)
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
        #plt.text(j, i, format(cm[i, j], '.2f'),
```

```
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else
"black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, y_pred)
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(1)
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without
normalization')

# Plot normalized confusion matrix
plt.figure(2)
plot_confusion_matrix(cnf_matrix, classes=class_names,
normalize=True,
                      title='Normalized confusion matrix')

#GradientBoosting
plt.show()
print(__doc__)
import itertools
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import GradientBoostingClassifier

# import some data to play with
target_names = np.array(['seizure', 'healthy'], dtype='|S10')
class_names = target_names

# Split the data into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2, random_state=0)
```

```
# Run classifier, using a model that is too regularized (C
too low) to see
# the impact on the results

classifier = GradientBoostingClassifier(n_estimators=25,
learning_rate=1.0, max_depth=5, random_state=0)
y_pred = classifier.fit(X_train, y_train).predict(X_test)


def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Oranges): #Blues
Purples Greens Oranges
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:,
np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else
"black")
```

```
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, y_pred)
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(1)
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without
normalization')

# Plot normalized confusion matrix
plt.figure(2)
plot_confusion_matrix(cnf_matrix, classes=class_names,
normalize=True,
                      title='Normalized confusion matrix')

plt.show()

#SVM
print(__doc__)
import itertools
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

# import some data to play with
target_names = np.array(['seizure', 'healthy'], dtype='|S10')
class_names = target_names

# Split the data into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2, random_state=0)

# Run classifier, using a model that is too regularized (C
too low) to see
# the impact on the results
```

```
classifier = svm.SVC(kernel='poly', C=0.01)
y_pred = classifier.fit(X_train, y_train).predict(X_test)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues): #Blues Purples
Greens Oranges
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:,
np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
        #plt.text(j, i, cm[i, j],
        plt.text(j, i, format(cm[i, j], '.2f'),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else
"black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, y_pred)
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(1)
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without
normalization')

# Plot normalized confusion matrix
plt.figure(2)
plot_confusion_matrix(cnf_matrix, classes=class_names,
normalize=True,
                      title='Normalized confusion matrix')

plt.show()


from sklearn.svm import SVC
from itertools import product
from sklearn.ensemble import VotingClassifier
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import cross_val_score


# Training classifiers
clf1      =      GradientBoostingClassifier(n_estimators=100,
learning_rate=1.0, max_depth=5, random_state=0)
clf2 = svm.SVC(kernel='poly', C=1)
eclf  =  VotingClassifier(estimators=[('gb',  clf1),  ('svc',
clf2)], voting='soft', weights=[2,1])

for clf, label in zip([clf1, clf2, eclf], ['GradientBoosting',
'SVM', 'Ensemble']):
    scores    =    cross_val_score(clf,    X,    Y,    cv=5,
scoring='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(),
scores.std(), label))
```