МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістр

на тему:

“КЛАСТЕРИЗАЦІЯ ТА КЛАСИФІКАЦІЯ ЧАСОВИХ РЯДІВ”

Виконав: студент 2 курсу, групи МФ-61
спеціальність 122 «Комп’ютерні науки»
освітньо-професійна програма
«Інформатика»

Качанов С. А.
Керівник ст. викладач Власенко Д. І.
Рецензент

Харків – 2023 року

ЗМІСТ

ВСТУП

В сучасному світі величезна кількість даних генерується з різних джерел, таких як соціальні мережі, фінансові ринки, медичні технології, промислові процеси, наукові дослідження, комп’ютерні технології, космічна сфера тощо. Багато з цих даних можна описати як часові ряди, що містять інформацію про зміни певної величини в часі.

Тобто, це послідовність значень, які вимірюються відповідно до певного часового інтервалу. Часові ряди використовуються для відображення зміни певної величини з плином часу, такої як ціна акцій, кількість продажів, температура тощо. Часові ряди зазвичай мають складну структуру і можуть містити тренди, сезонність, циклічність та інші складові, які роблять їх аналіз складним завданням.

Вони використовуються в багатьох сферах і галузях нашого життя, а саме в економіці, медицині, фінансах та навіть юриспруденції.

У фінансовій сфері, наприклад, аналітики вивчають часові ряди цін на акції, курси валют, ціни на сировину та інші фінансові показники, щоб зробити передбачення щодо майбутніх трендів та прийняти рішення про інвестування коштів.

У медицині, лікарі використовують часові ряди для відстеження змін у показниках здоров'я пацієнтів, таких як температура, пульс, артеріальний тиск, рівень глюкози в крові, серцебиття та інших.

У промисловості, компанії аналізують часові ряди для відстеження рівня виробництва, ефективності роботи обладнання, кількості браку та інших показників, щоб покращити процес виробництва та зменшити витрати. Саме тому дуже важливо вміти якісно їх групувати за різними критеріями, залежностям і характеристикам.

В юридичній сфері можуть використовувати часові ряди, щоб аналізувати динаміку кількості злочинів в певній місцевості за різні періоди часу. Це може допомогти встановити закономірності та тенденції в динаміці злочинності та впливати на розробку політики щодо протидії злочинності.

Часові ряди містять в собі багато задач, які можуть бути вирішені з використанням аналізу даних [1], таких як:

- Прогнозування, тобто визначення майбутніх значень ряду, на основі вже відомих попередніх даних. Наприклад, прогнозування курсу доллару чи біткоіна, ріст чи падіння акцій певної компанії або цін на певний продукт чи річ.
- Виявлення аномалій [2]: виявлення незвичайних значень в часовому ряді, які можуть свідчити про несподівані події або помилки в даних. Наприклад, виявлення аномальної кількості відвідувань веб-сайту протягом певного часового періоду або зависоких чи занизьких значень температури.
- Класифікація: розподіл значень часового ряду на різні категорії в залежності від визначених параметрів. Наприклад, розподілити пісні по їх жанрам. Свого роду відомий додаток Shazam [3], також, як би це дивно не звучало, є задачею класифікації часового ряду. Як відомо на вхід застосунку подається фрагмент звукозапису, який і є часовим рядом, і його треба класифікувати до певної пісні, які є в базі цього застосунку.
- Кластеризація: розбиття часового ряду на групи, які мають схожі характеристики. Наприклад, кластеризація даних про продажі товарів в різні періоди року з метою визначення сезонних тенденцій.

В своїй кваліфікаційній роботі я якраз дослідив дві останні задачі: класифікацю і кластеризацію часових рядів, які є важливими завданнями в аналізі даних.

Однак, класифікація та кластеризація часових рядів є складними завданнями, оскільки ряди можуть мати різну довжину, форму та амплітуду коливань, що робить непростим встановлення схожості між рядами. У зв'язку з цим, багато різних методів були розроблені для вирішення цих завдань, включаючи методи на основі статистичних моделей, нейронних мереж та інші.

Існує багато алгоритмів, призначених для класифікації часових рядів. В своїй науковій роботі я дослідив основні з них, та обгрунтував, чому саме ці алгоритми, моделі та методи є найкращими для вирішення цих задач.

Залежно від даних один тип може забезпечити вищу точність класифікації, ніж інші типи. Ось чому важливо розглянути низку алгоритмів, занурюючись у проблему класифікації та кластеризації часових рядів.

Дана кваліфікаційна робота складається зі вступу, 3 розділів, 16 підрозділів, загальних висновків, списку використаної літератури (20), додатків (1). Уся робота висвітлена на 44 сторінках основного тексту містить 1 діаграму і 17 рисунків. Всі результати дослідження і код можна переглянути за даним посиланням:
https://colab.research.google.com/drive/18bUs2pw_yVfYAV6pBB-CNU_JIgpaiy9M?usp=sharing

2 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

2.1 Основні типи часових рядів

Усі часові ряди поділяються на стаціонарні і нестаціонарні, які в свою чергу поділяються на більше типів.

Стаціонарний часовий ряд - це послідовність даних, у якої статистичні характеристики, такі як середнє значення та дисперсія, залишаються сталими з часом. Для стаціонарного часового ряду характерна постійна та стабільна поведінка з часом, що дозволяє зробити певні припущення щодо майбутніх значень. Це дає можливість застосовувати математичні моделі для прогнозування майбутніх значень часового ряду.

Нестаціонарний часовий ряд - це послідовність даних, у якої статистичні характеристики змінюються з часом. Для нестаціонарного часового ряду характерні зміни в тренді [4], сезонності та/або циклічності з часом. Нестаціонарність ускладнює прогнозування майбутніх значень, оскільки певні припущення щодо поведінки часового ряду з часом можуть стати недійсними. Для розуміння поведінки часового ряду необхідно визначити його тип та зрозуміти, які ефекти впливають на його змінність з часом. Наприклад, якщо часовий ряд має явний тренд на зростання, то він не буде стаціонарним, тому що середнє значення змінюється з часом. У такому випадку можна використовувати методи для стабілізації тренду, щоб отримати стаціонарний часовий ряд та зробити прогноз майбутніх значень.

У свою чергу стаціонарні і нестаціонарні поділяються на більш спеціалізовані. Розглянемо найпоширеніші з них, а саме:

- трендовий часовий ряд - це послідовність даних, яка має чіткий тенденційний рух у певному напрямку з часом. Такий ряд характеризується зміною середнього значення величини з часом. На

жаль, через російські обстріли енергетичної інфраструктури набагато зріс попит на електрогенератори, тобто часовий ряд на кількість продажів електрогенераторів в Україні буде трендовим, причому завдяки ЗСУ та новітнім засобам ППО цей тренд вже зменшується, бо стабілізується ситуація з електропостачанням. На Рис. 1 продемонстровано явний зріст попиту на електрогенератори, тобто трендовий часовий ряд [5].

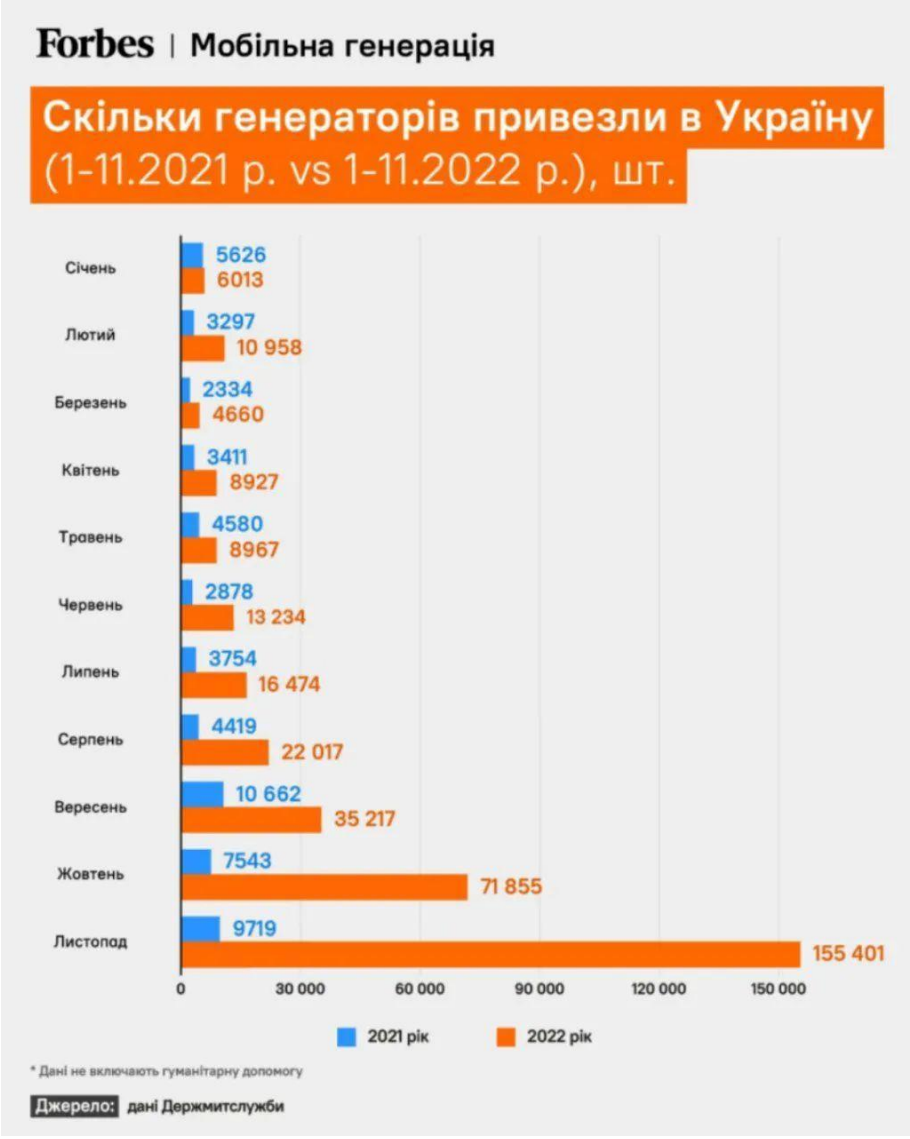


Рис. 1 – Графік кількості електрогенераторів завезених в Україну в 2021-2022 роках

- сезонний часовий ряд - це послідовність даних, яка має повторювані зміни з часом в межах певного періоду. Сезонність зазвичай пов'язана з природою, звичками та побутом людей, або з іншими факторами, що змінюються з часом. Наприклад, часовий ряд споживання електроенергії має річну сезонність, бо очевидно, що в різну пору року по-різному споживається електроенергія. На Рис. 2 продемонстровано сезонність у часовому ряді споживання електроенергії в 2019-2021 роках в Україні [6].

Рис. 2 – Споживання електроенергії в Україні

- циклічний часовий ряд - це послідовність даних, яка має змінну величину з часом, яка повторюється з певною періодичністю, але не є повністю сезонним. На Рис. 3 видно приклад циклічного часового

ряду – в даному випадку звичайна кардіограма. Так як серцебиття відбувається регулярно і з циклічною періодичністю, то це і є часовий ряд даного типу.

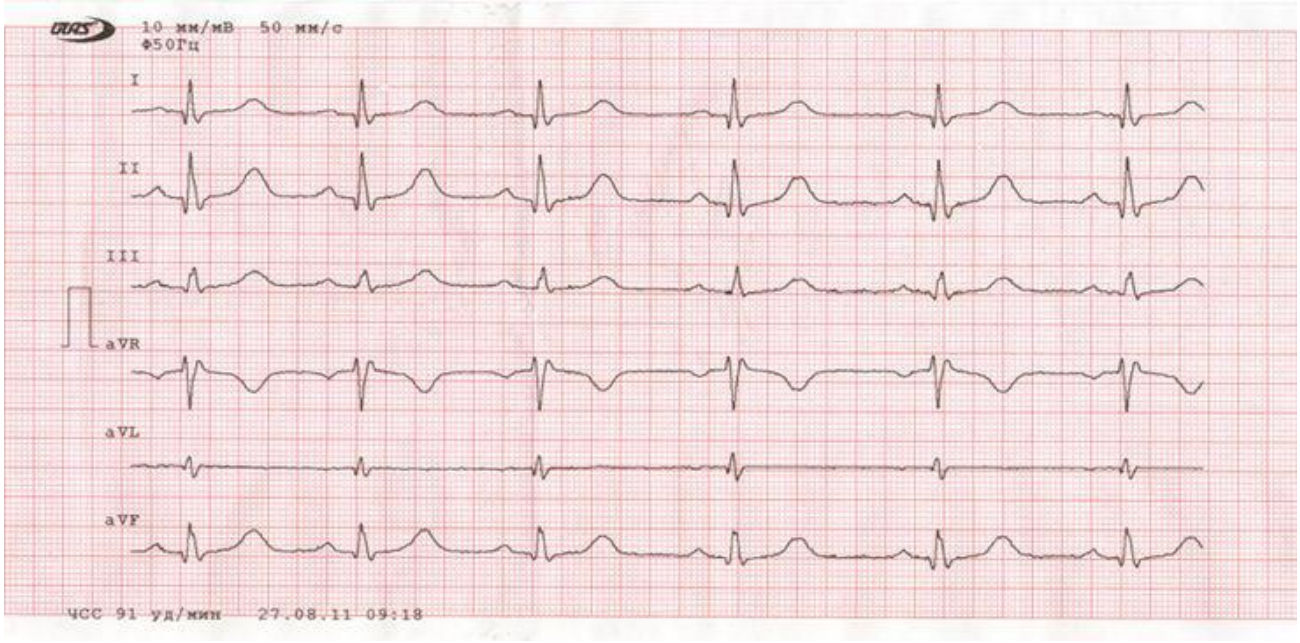


Рис.3 – Електронна кардіограма

Також часові ряди можуть бути згруповані за різними ознаками. Передусім, виділяють два типи динамічних рядів залежно від способу вказання часу. Якщо вказується момент часу, то значення показника характеризує явище в окремий чітко визначений момент часу. Якщо вказується часовий інтервал, то значення показника відображає всю кількість змін цього явища за цей період часу. У першому випадку йдеться про моментні динамічні ряди, а в другому - про інтервальні.

Як момент часу, зазвичай виступає конкретне число або певна дата. Можемо розглянути кількість хворих коронавірусом або чисельність населення у світі на якусь конкретну дату кожного року. Але може бути і більш детальна динаміка, наприклад на початок зміни, тобто вказується не тільки дата, але й

час. Слід зазначити, що в моментних динамічних рядах між рівнями ряду залишаються часові проміжки, в яких значення досліджуваного показника залишаються невідомими.

Тобто, чим частіше облік моменту, тим детальніше відображається розвиток досліджуваного явища. Кожен рівень моментного динамічного ряду відображає стан чогось (населення, запасів на складі, чисельності студентів і т.п.)

Якщо в інтервальному ряді представлені дані за неперервний проміжок часу, наприклад за кожні з аналізованих 5 років, то між ними проміжків немає, і можна представити всю картину зміни явища за цей 5-річний часовий проміжок. Зазвичай, величина рівня інтервального ряду залежить від величини заданого часу (річні, місячні, тижневі, денні, часові хвилині і секундні дані). Сума рівнів інтервального часового ряду має реальний змістовний зміст.

Показники, які характеризують певне явище, не завжди можуть бути представлені за рівні проміжки часу. За відстанню між рівнями ряду часові ряди поділяються на ряди з рівновіддаленими та нерівновіддаленими рівнями за часом. Частіше для послідовного вивчення часових змін використовують динамічні ряди з рівновіддаленими рівнями (щорічні, щомісячні дані). Ряди з нерівно віддаленими рівнями можуть використовуватися при нестачі початкової інформації, а також у випадках, коли необхідно дати порівняння зі значущими для цього явища рівнями (наприклад, в дореформений період).

На основі типу показників рівнів ряду виділяють динамічні ряди абсолютних, відносних та середніх величин. При аналізі інтервальних динамічних рядів абсолютних величин, в разі неперервних даних, сумування рівнів ряду дає уявлення про значення досліджуваного показника за більший період часу. Наприклад, при аналізі щорічних даних про видобуток нафти, сумування рівнів за, скажімо, п'ять років, дає характеристику обсягу видобутої

нафти за це п'ятиліття. Серед динамічних рядів абсолютних величин іноді виділяють ряди, що характеризують зміну кількості одиниць якої-небудь сукупності (кількості населення регіону, кількості працівників фірми, поголів'я худоби фермерського господарства). Аналіз динаміки відносних та середніх показників дозволяє порівнювати динаміку цих показників за різними об'єктами або динаміку різних показників. Наприклад, можна порівнювати динаміку частки експортованої нафти в загальному обсязі її видобутку в різних країнах, або зміну середнього грошового доходу населення в різних регіонах країни за останнє десятиліття, або порівнювати зміни середньої заробітної плати та продуктивності праці на підприємстві.

Останнім часом було розроблено багато алгоритмів класифікації часових рядів, і вибір найбільш ефективного залежить від конкретної задачі та характеристик часових рядів. Однак можна виділити кілька алгоритмів, які часто використовуються в розв'язанні задач класифікації часових рядів і показують хороші результати.

2.2 Огляд відомих рішень і підходів

У відомій науковій статті Time series clustering based on the characterisation of segment typologies [7] пропонується новий метод кластеризації часових рядів, який базується на типології сегментів. Метод пропонує розподіляти часовий ряд на декілька сегментів з різною динамікою і використовувати характеристики кожного сегменту для класифікації ряду в певний кластер.

Автори статті також провели додаткові експерименти для дослідження впливу різних параметрів на результати кластеризації. Вони виявили, що

правильний вибір кількості сегментів і характеристик, використаних для кластеризації, є критичним для досягнення кращих результатів.

Узагальнюючи, стаття пропонує новий метод кластеризації часових рядів на основі типології сегментів, який показав свою ефективність для різних типів даних. Цей метод може бути застосований і в інших областях і галузях, де важливо класифікувати часові ряди на групи з подібною динамікою.

У своєму виступі та тезах на XIV Міжнародній науково-практичній конференції [8] я робив огляд основних підходів та алгоритмів класифікації часових рядів та пояснював, чому саме обрані алгоритми найкраще підходять для цих задач.

Там згадується, з-поміж інших, два методи класифікації часових рядів: метод головних компонент (PCA) та LSTM (Long Short-Term Memory).

Я зазначав, що метод головних компонент (PCA) - це метод зменшення розмірності даних, який дозволяє перетворити вихідний набір даних на новий набір змінних, який називається головними компонентами. Головні компоненти є лінійними комбінаціями вихідних змінних, які пояснюють найбільшу частину варіації в даних. Використовуючи головні компоненти зменшеної розмірності, можна зробити класифікацію часових рядів, наприклад, використовуючи метод головних компонент для створення прогнозів за допомогою лінійної регресії або методу опорних векторів (SVM).

Щодо LSTM, то докладніше це буде розглянуто в наступному розділі, але якщо взяти основне, то це нейронна мережа, що має здатність зберігати та використовувати інформацію про попередній стан. Ця здатність особливо корисна для моделювання послідовних даних, таких як часові ряди. LSTM може бути використана для класифікації часових рядів, наприклад, для прогнозування часових рядів з множинними змінними або для виявлення аномалій у часових

рядах. LSTM зазвичай показує високі результати на часових рядах зі складною структурою або залежностями.

Отже, обидва методи можуть бути використані для класифікації часових рядів, але вони мають різні підходи до обробки та аналізу даних. PCA спрямований на зменшення розмірності даних, щоб зробити їх більш розумними для моделювання, тоді як LSTM використовується для аналізу послідовностей даних та виявлення складних залежностей між ними, що дозволяє прогнозувати майбутні значення часових рядів з високою точністю. Обидва методи можуть бути використані як окремо, так і в поєднанні з іншими методами для досягнення більш точних результатів у класифікації часових рядів.

Враховуючи усю цю отриману інформацію, та дослідивши переваги та недоліки цих методів для свого дослідження я обрав якраз таки рекурентну нейронну мережу довгої короткочасної пам’яті (LSTM), з-поміж інших методів і алгоритмів, які оглядалися у цих тезах.

3 ОПИС ОБРАНИХ АЛГОРИТМІВ ТА ПРОГРАМНИХ РІШЕНЬ

3.1 Рекурентна нейронна мережа LSTM

У своїй кваліфікаційній роботі я обрав (у вигляді основного), за різними оцінками, один з найкращих підходів для поставленої задачі – це використання глибокого навчання і зокрема рекурентної мережі з довгою короткочасної пам’яттю – LSTM. В своїй роботі на науковій конференції [8] я вже зазначав чому саме LSTM чудово підходить для задач класифікації часових рядів, тому більш докладно ще раз опишу роботу LSTM і чому вона гарно підходить для часових рядів.

LSTM є одним з найбільш відомих типів рекурентної нейронної мережі [9] (RNN) і використовується для обробки послідовностей даних, таких як мовлення, текст та часові ряди.

Основна проблема з традиційними RNN полягає в тому, що вони мають проблему з довгостроковим запам'ятовуванням інформації, яка входить у послідовність. LSTM була розроблена, щоб вирішити цю проблему, вона має спеціальну структуру, яка дозволяє їй зберігати та використовувати інформацію на протязі довгих часових періодів.

LSTM складається з невеликої кількості повторюваних блоків, які звуть LSTM-клітинками [10]. Кожна клітина має троє "ворот": ворота забування, вхідні ворота та вихідні ворота.

- Ворота забування (forget gate) відповідають за відбір того, що треба забути з попереднього стану клітинки. Воно приймає попередній стан клітинки та поточний вхід та вирішує, що треба зберегти, а що забути.
- Вхідні ворота (input gate) відповідають за відбір нових значень, які повинні бути збережені в пам'яті клітинки. Воно приймає

попередній стан клітинки та поточний вхід та вирішує, які значення потрібно додати до пам'яті.

- Вихідні ворота (output gate) відповідають за вибір значень, які будуть виведені з клітинки. Воно приймає поточний вхід та попередній стан клітинки та вирішує, які значення потрібно вивести з клітинки.

Крім того, LSTM має додатковий параметр, який називається "внутрішній стан". Це значення зберігає поточний стан клітинки та використовується для згадування довгострокової інформації. Давайте поглянемо на кожен етап в деталях:

- Ворота забування (Forget gate):
  - Приймає попередній стан LSTM-клітинки та поточний вхід
  - Генерує число від 0 до 1 для кожного елемента попереднього стану LSTM-клітинки
  - Чим ближче до 1, тим більшу вагу має цей елемент і тим більшу кількість інформації буде збережено.
- Вхідні ворота (Input gate):
  - Приймає попередній стан LSTM-клітинки та поточний вхід
  - Генерує число від 0 до 1 для кожного елемента поточного входу та для кожного елемента попереднього стану LSTM-клітинки
  - Чим ближче до 1, тим більшу вагу має цей елемент і тим більшу кількість інформації буде збережено.
- Оновлення внутрішнього стану:
  - Попередній стан LSTM-клітинки помножується на вагу воріт забування (Forget gate) та додається до поточного входу

- Це дає нову інформацію, яка потенційно може бути збережена в пам'яті LSTM-клітинки.

- Вихідні ворота (Output gate):
  - Приймає попередній стан LSTM-клітинки та поточний вхід
  - Генерує число від 0 до 1 для кожного елемента поточного стану LSTM-клітинки
  - Чим ближче до 1, тим більшу вагу має цей елемент і тим більшу кількість інформації буде виведено з LSTM-клітинки.

Таким чином, LSTM може зберігати та використовувати інформацію з попередніх кроків часу, що робить його особливо корисним для обробки послідовностей даних.

3.2 Застосування LSTM для класифікації часових рядів

Рекурентна нейронна мережа LSTM дуже ефективна для задач класифікації часових рядів. Очевидно, що для цього треба зробити препроцесінг даних і підготувати дані у відповідному форматі, тобто ввести послідовний вхід та очікуваний вихід для кожного кроку часу.

LSTM дуже якісно вирішує ці задачі, оскільки вона може зберігати та використовувати довготривалу залежність між даними у часовому ряді.

У звичайних нейронних мережах, які використовуються для класифікації, зазвичай використовується пряме поширення (feedforward) сигналу, при цьому кожен вхідний сигнал обробляється окремо. Такі моделі не можуть враховувати динаміку зміни в часі, оскільки вони не мають жодної пам'яті про попередні вхідні дані. LSTM, натомість, має здатність зберігати і використовувати попередні стани внутрішніх блоків, що називається пам'яттю LSTM. Це

дозволяє моделі зберігати інформацію про попередній контекст, а також розуміти залежності між даними у часі.

Таким чином, LSTM може здійснювати передбачення на основі повної історії часового ряду з урахуванням всіх попередніх значень, що дозволяє їй бути ефективним в задачах класифікації часових рядів. Крім того, LSTM має здатність автоматично визначати, яку інформацію слід забути та яку треба зберегти в пам'яті. Це дозволяє моделі прибрати зайву інформацію, яка може заважати правильному класифікуванню.

Для класифікації часового ряду за допомогою LSTM потрібно спочатку побудувати LSTM-модель з відповідними входами та вихідними шарами. Вхідний шар повинен бути розміру (кількість кроків часу, кількість ознак), де кількість кроків часу відповідає довжині часового ряду, а кількість ознак - кількості ознак у кожному кроці часу.

У своєму дослідженні я використовував датасет в якому 5 класів, тобто вихідний шар це буде ймовірності належності до певного класу.

3.3 Алгоритм KNN та його використання в цій задачі

Алгоритм k-найближчих сусідів (k-Nearest Neighbors, KNN) - це метод машинного навчання без учителя, який використовується для класифікації та регресії.

У задачах класифікації, KNN [11] використовує набір навчальних даних для знаходження k найбільш близьких за відстанню точок до тестового зразка. На наступній ітерації метод використовує більшість голосів цих точок, щоб визначити клас тестового зразка.

Для виконання класифікації часових рядів, KNN використовує подібність між часовими рядами, яка вимірюється за допомогою функцій відстані. Кожен

часовий ряд розглядається як вектор, де кожна точка відповідає значенню часового ряду в певний момент часу.

Для класифікації нового часового ряду, KNN знаходить k найближчих часових рядів з навчального набору за допомогою відстані між векторами, яку можна обчислити за допомогою різних функцій відстані, таких як Евклідова відстань, манхеттенська відстань, і т.д. Потім класифікація нового часового ряду здійснюється шляхом голосування кращих k сусідів.

Одна з головних переваг KNN в класифікації часових рядів - це його простота та ефективність. Він не потребує великої кількості навчальних даних та може добре працювати з шумними даними. Крім того, KNN може добре працювати з великими наборами даних, оскільки він не потребує часу для тренування моделі.

Проте, у KNN також є деякі недоліки [12]. Він може бути чутливим до викидів у даних, оскільки він не враховує структуру даних та може класифікувати тестові зразки з помилкою, якщо навчальний набір містить зміщені дані або викиди.

Щодо застосування KNN для класифікації часових рядів, то цей метод машинного навчання може бути особливо ефективним, оскільки він може допомогти знайти схожість між часовими рядами та використовувати цю інформацію для класифікації нових даних.

Крім того, KNN може допомогти виявляти різні патерни та залежності між даними, що може бути корисним при класифікації часових рядів зі складними залежностями. Незважаючи на все це, KNN не є єдиним методом для досягнення якісного рівня класифікації часового ряду. Інші методи машинного навчання, такі як нейронні мережі, можуть бути більш ефективними в деяких випадках, особливо якщо часові ряди мають складні структури та залежності.

Отже, можна дійти висновку, що вибір методу для класифікації часових рядів повинен бути зроблений на основі конкретної задачі та характеристик даних, які необхідно аналізувати.

3.4 Метод K-means для кластеризації часових рядів

Метод к-середніх (K-means) є одним з найпоширеніших методів кластеризації [13], який дозволяє розділити множину даних на кластери на основі схожості між їх елементами.

Основна ідея полягає в тому, щоб розділити дані на k кластерів шляхом мінімізації суми квадратів відстаней між елементами та центрами кластерів.

Алгоритм K-means складається з наступних кроків:

- Обрання кількості кластерів (k).
- Випадкове обрання k точок-центроїдів.
- Присвоєння кожному елементу найближчого до нього центроїда.
- Обчислення нових центроїдів кожного кластеру як середнє значення всіх елементів, які входять до кластеру.
- Повторення кроків 3 та 4 до збіжності або до досягнення максимальної кількості ітерацій.

Відповідно отримуємо, що після відпрацювання алгоритму кожен елемент буде належати до певного кластеру.

З цих самих міркувань метод к-середніх підходить і для роботи з часовими рядами. Використання методу к-середніх для кластеризації часових рядів полягає в тому, щоб розділити множину часових рядів на кластери на основі їх схожості. Для цього, зазвичай, використовують такі метрики, як Евклідова відстань або манхеттенська відстань, або косинусна відстань між векторами.

Крім того, можна використовувати різні методи для побудови векторів-ознак для часових рядів, такі як метод головних компонент.

Він є досить ефективним для задачі кластеризації часових рядів і має багато переваг:

- Простота та ефективність алгоритму.
- Відсутність необхідності в попередній підготовці даних.
- Здатність обробляти великі обсяги даних.
- Можливість використання різних метрик та методів побудови ознак для досягнення більш точної кластеризації.

Недоліками методу к-середніх є необхідність попереднього визначення кількості кластерів та залежність результатів від початкового вибору центроїдів. Крім того, він не завжди ефективний для кластеризації даних зі складними структурами, де границі між кластерами не є чітко визначеними (це і буде з’ясовано при реалізації алгоритму на наборі даних, який я обрав для дослідження).

У задачах класифікації часових рядів метод к-середніх можна використовувати для попередньої кластеризації даних та подальшого застосування методів класифікації для кожного кластеру окремо. Це може допомогти поліпшити якість класифікації та знизити час виконання алгоритму.

3.5 Алгоритм кластеризації DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - це алгоритм кластеризації, який побудований на основі густоти точок в просторі.

Головна ідея алгоритму полягає в тому, щоб групувати точки, які знаходяться в зоні високої густоти, і відрізняти їх від точок, які знаходяться в зоні низької густоти [14]. Алгоритм був розроблений для роботи з даними, які

можуть мати складну структуру, саме тому він є дуже ефективним для кластеризації часових рядів.

Процес роботи DBSCAN складається з двох основних параметрів: радіус ε та мінімальна кількість точок, що необхідні для формування кластера (minPts). Перший параметр ε визначає радіус деякої зони навколо кожної точки, а другий параметр minPts визначає мінімальну кількість точок, які повинні знаходитися в цій зоні, щоб вважати точку як центр кластера.

Алгоритм починається з вибору довільної точки датасету, яка не була відвідана [15]. Після цього алгоритм перевіряє, чи достатня кількість сусідніх точок знаходиться в радіусі ε від цієї точки. Якщо так, то ці точки вважаються складовими кластера. Якщо ні, то точка вважається шумом. Якщо точка належить до кластера, то її сусідні точки також перевіряються для визначення чи вони належать до того ж кластера. Якщо так, то ці точки додаються до кластера, а процес продовжується далі. Якщо ні, то ці точки також вважаються шумом. Процес продовжується до тих пір, поки всі точки датасету не будуть відвідані.

В результаті алгоритму формуються різні кластери, які можуть мати різну форму та розмір. Якщо в зоні, яка задана радіусом ε, знаходиться менша кількість точок, ніж мінімальна кількість точок minPts, то точка вважається шумом і не призначається жодному кластеру.

Одна з головних переваг DBSCAN полягає в тому, що він може працювати з даними різної густини та форми кластерів, і він добре підходить для кластеризації часових рядів, де зазвичай зустрічаються складні форми та різні рівні густини даних. Крім того, DBSCAN може ідентифікувати шумові точки, які не належать до жодного кластера.

Однак, існує деяка кількість недоліків, пов'язаних з використанням DBSCAN. Наприклад, якщо розмір датасету дуже великий, то алгоритм може

стати надто повільним (мій обраний датасет не є дуже об’ємним, тому алгоритм досить швидко).

Крім того, вибір правильних значень параметрів ε та minPts може бути досить складним завданням, і це може вплинути на результати кластеризації. Також, при використанні DBSCAN для кластеризації часових рядів, може виникнути проблема з вибором відповідної міри відстані, оскільки традиційні міри відстані, такі як Евклідова, можуть бути недостатніми для опису відносин між точками в часовому ряді.

У порівнянні з методом k-середніх, DBSCAN має деякі переваги. Наприклад, DBSCAN може працювати з даними з різною густотою та формою, тоді як k-середніх передбачає, що кластери мають сферичну форму та рівномірну густину.

Іншою перевагою DBSCAN є його здатність виявляти кластери будь-якої форми, у той час як метод k-середніх може видавати тільки круглі кластери, оскільки він розрахований на глобальну мінімізацію суми квадратів відстаней між кластерними центрами та прикладами. DBSCAN замість цього використовує локальну густину точок для виявлення кластерів, тому може знаходити кластери будь-якої форми.

Недоліком DBSCAN є потреба в попередньому визначенні двох гіперпараметрів: eps (радіус околу для пошуку сусідніх точок) та min_samples [16] (мінімальна кількість сусідів, необхідна для того, щоб точка була визнана як "ядерна"). Ці параметри впливають на кількість та форму кластерів, які будуть знайдені, тому їх вибір може бути нетривіальним та вимагати попередньої експертизи дослідника. Крім того, DBSCAN відносно громіздкий у плані обчислювальних витрат, особливо для великих наборів даних.

Отже, можна заявити з упевненістю, що DBSCAN - це потужний метод кластеризації часових рядів, який дозволяє виявляти кластери будь-якої форми

та не вимагає заздалегідь визначених кількості кластерів. Його недоліки полягають у потребі в налаштуванні гіперпараметрів та високих обчислювальних витратах порівняно з методом k-середніх, але варто зазначити, що і в методі k-середніх теж підбір гіперпараметрів відіграє важливу роль.

3.6 Опис прийнятих програмних рішень

В своєму дослідженні і аналізі я застосував найзручнішу, на думку багатьох експертів, мову програмування для таких задач – Python. Як відомо, ця мова програмування набула популярності серед дослідників та інженерів, які займаються аналізом даних, оскільки має багатий набір інструментів для роботи з даними. Також вона має дуже широку систему для роботи з машинним навчанням, а зокрема різними алгоритмами класифікації і кластеризації.

Python має бібліотеки, такі як Pandas і NumPy, які забезпечують високу ефективність та зручний інтерфейс для роботи з даними, які я і використовував у своїй роботі. Крім того, Python має бібліотеку Scikit-Learn [17], яка містить набір класифікаційних та кластерних алгоритмів, призначених для роботи з даними в часових рядах. Саме ці бібліотеки з-поміж інших я використовував у своєму дослідженні (див. Рис. 4).

Також важливу роль для побудови мережі LSTM, відіграла бібліотека Keras, що є високорівневим інтерфейсом до TensorFlow, який дозволяє легко створювати, тренувати та оцінювати глибокі нейронні мережі. TensorFlow, з іншого боку, є бібліотекою машинного навчання з відкритим кодом, яка дозволяє створювати, тренувати та застосовувати різноманітні моделі машинного навчання.

Саме тому TensorFlow та Keras є потужними інструментами для роботи з класифікацією та кластеризацією часових рядів, адже містять спеціалізовані бібліотеки, які дозволяють створювати ефективні та точні моделі

```
import os
import glob
import fnmatch
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
#import matplotlib as plt
import IPython.display as ipd
import math
import tensorflow as tf


from keras.utils import np_utils
#from keras.layers.merge import concatenate
from tensorflow.keras.layers import concatenate
from tensorflow.keras.models import Sequential, Model, load_model

from tensorflow.keras.layers import Conv1D, Conv2D, SeparableConv1D, MaxPooling1D, MaxPooling2D
from tensorflow.keras.layers import Input, add, Flatten, Dense, BatchNormalization, Dropout, LSTM, GRU
from tensorflow.keras.layers import GlobalMaxPooling1D, GlobalMaxPooling2D, Activation, LeakyReLU, ReLU

from tensorflow.keras import regularizers
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import Adamax
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import DBSCAN
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,matthews_corrcoef
from sklearn.metrics import cohen_kappa_score,roc_auc_score,confusion_matrix,classification_report
```

Рис. 4 – Усі інструменти і бібліотеки, використані в роботі

Як видно на Рис. 4 у вигляді інструменту для досліджень я обрав Jupyter Notebook, який є інтерактивною оболонкою для програмування, яка дозволяє виконувати код Python в окремих клітинах та відображати результати безпосередньо в ноутбуці.

Використання Jupyter Notebook має одразу декілька переваг:

- Інтерактивний інтерфейс: Jupyter Notebook дозволяє виконувати код в режимі реального часу та отримувати результати на місці. Це дозволяє дослідникам та аналітикам швидко відкривати та

виправляти помилки, а також візуалізувати результати аналізу даних безпосередньо в ноутбуці.

- Інтерактивна візуалізація даних: Jupyter Notebook дозволяє створювати візуальні графіки, діаграми та інші візуальні елементи безпосередньо в ноутбуці, що дозволяє швидко аналізувати та візуалізувати дані.
- Легкість використання та налагодження: Jupyter Notebook має простий інтерфейс та дозволяє легко виконувати код Python, робити експерименти, аналізувати дані та візуалізувати результати. Крім того, Jupyter Notebook підтримує багато популярних бібліотек Python для машинного навчання та аналізу даних, таких як Pandas, NumPy [18], Scikit-Learn, TensorFlow, які я і використовував у своїй роботі (див. Рис. 4).

Отже, використання всіх цих бібліотек, мови програмування, Jupyter Notebook та підходів є обгрунтованим для задачі класифікації і кластеризації часових рядів.

4 ДОСЛІДЖЕННЯ ТА ЙОГО РЕЗУЛЬТАТИ

4.1 Актуальність обрання даних та вирішення цієї задачі, опис даних

Для обрання максимально релевантних даних, я вирішив обрати відомий архів, де зібрані дані за останні 20 років (синтетичні та натуральні), які найкраще підходять для наукових досліджень. Архів часових рядів UCR [19], запроваджений у 2002 році, став важливим ресурсом для спільноти з видобутку даних часових рядів, з щонайменше тисячею опублікованих статей, які використовують принаймні один набір даних з архіву. Оригінальна версія архіву містила шістнадцять наборів даних, але з того часу він регулярно розширювався, і зараз містить більше сотні, тому саме з цих даних я і обрав датасет для дослідження алгоритмів.

Дані, які я обрав, є звуковим записом серцебиття реальних людей і основна задача полягає в класифікації цих даних (оскільки в своїй роботі я досліджую також і кластеризацію, то для розв’язання задачі кластеризації я використовував нерозмічені дані з цього датасету).

Але ж давайте з’ясуємо актуальність саме цього набору даних. В описі даних [20] зазначається, що за даними Всесвітньої організації охорони здоров'я, серцево-судинні захворювання є головною причиною смерті у світі: щорічно від ССЗ помирає більше людей, ніж від будь-якої іншої причини.

За оцінками, у 2004 році від ССЗ померло близько 17,1 мільйонів людей, що становило 29% всіх смертей у світі. З них приблизно 7,2 мільйонів смертей були пов'язані з ішемічною хворобою серця. Будь-який метод, який може допомогти виявити ознаки серцевої хвороби, може мати значний вплив на здоров'я людей у всьому світі. Ця задача полягає у створенні методів для досягнення саме цього. Конкретно, нас цікавить створення першого рівня

скринінгу серцевих патологій як у лікарні за допомогою цифрового стетоскопа, так і вдома пацієнтом за допомогою мобільного пристрою.

Саме тому ця проблема (і цей датасет) особливо цікава для дослідників машинного навчання, оскільки вона включає в себе класифікацію аудіоданих, де відрізнення класів цікавості є нетривіальним. Дані збираються в реальних умовах і часто містять шум будь-якого можливого типу. Різниця між звуками серця, що відповідають різним симптомам серцевих захворювань, також може бути дуже тонкою і складною для розрізнення. Успіх у класифікації цього типу даних вимагає дуже міцних класифікаторів. Незважаючи на його медичне значення, на сьогоднішній день це є досить малодослідженою областю в задачах класифікації часових рядів.

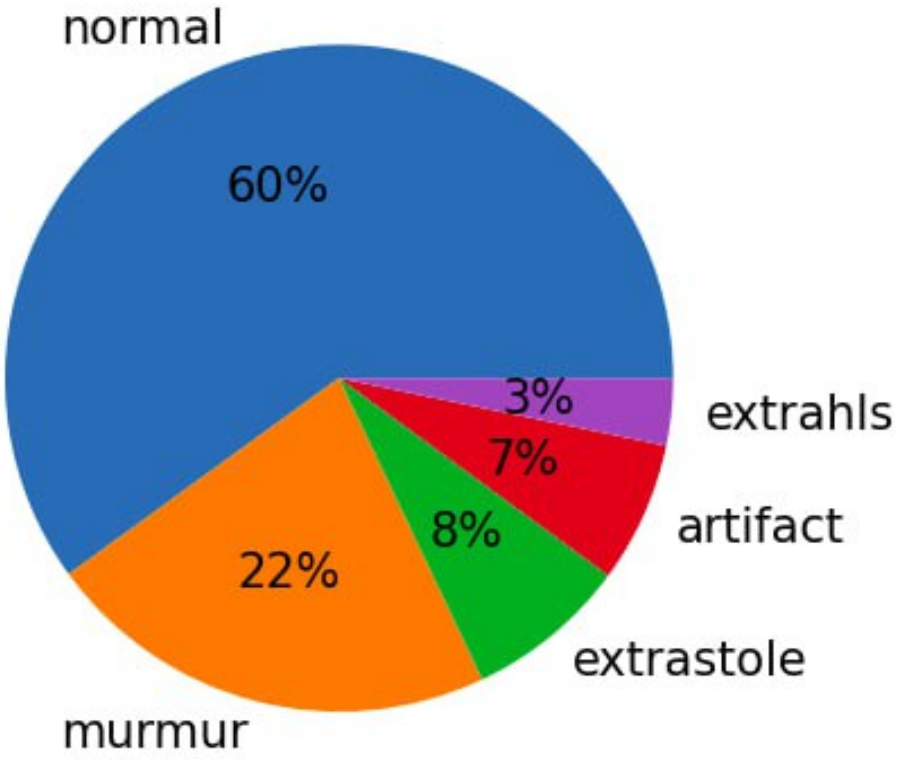
4.2 Категорії даних

Аудіофайли мають різну довжину від 1 до 30 секунд (деякі були обрізані, щоб зменшити надмірний шум та надати виокремлений фрагмент звуку). Більшість інформації у звуках серця міститься у низькочастотних компонентах, а шум - у вищих частотах. Давайте тепер розберемось із категоріями даних:

- Нормальний звук (normal). У категорії "Нормальний звук" є нормальні, здорові звуки серця. Вони можуть містити шум у останній секунді запису, коли прилад відіймається від тіла. Вони можуть містити різноманітні фонові шуми (від транспорту до радіо). Вони також можуть містити випадковий шум, що відповідає диханню або торканню мікрофона одягом або шкірою. Нормальний звук серця має чіткий шаблон «луб даб, луб даб», з часом від «луб» до «даб» коротшим, ніж час від «даб» до наступного «луб» (коли частота серцевих скорочень менше 140 ударів за хвилину).

- Категорія бурчання (murmur). Серцеві шуми звучать так, ніби є «свист, рев, гуркіт або бурхлива рідина» в одному з двох тимчасових місць: (1) між «луб» і «даб» або (2) між «даб» і «луб». Вони можуть бути симптомом багатьох захворювань серця, деякі серйозні. Одна з речей, яка бентежить людей, які не мають медичної освіти, полягає в тому, що шуми виникають між між «луб» і «даб» або між «даб» і «луб».
- Категорія додаткового серцевого звуку (extrahls). Додаткові серцеві звуки можна ідентифікувати, оскільки існує додатковий звук, наприклад «луб-луб даб» або «луб-даб-даб». Додатковий серцевий звук може не бути ознакою хвороби. Однак у деяких ситуаціях це важливий ознака хвороби, яку, якщо виявити рано, може допомогти людині. Додатковий серцевий звук важливо виявляти, оскільки його не можна добре виявити ультразвуком.
- Категорія «Артефакт» (artifact). В цій категорії є широкий спектр різних звуків, включаючи звук зворотнього зв'язку і ехо, мову, музику та шум. Зазвичай немає відчутних звуків серця і має мало або жодної тимчасової періодичності на частотах нижче 195 Гц. Ця категорія найбільш відрізняється від інших. Відрізняти цю категорію від чотирьох інших, дуже важливо щоб той, хто збирає дані, здійснив повторну спробу.
- Категорія «Надзвичайний серцевий ритм» (extrastole). Звуки цього ритму можуть з'являтися час від часу і можуть бути визначені тим, що серцевий ритм порушений через додаткові або пропущені серцеві скорочення, наприклад, «луб-луб-даб» або «луб-даб-даб» (це не те саме, що додатковий серцевий звук, оскільки ця подія не відбувається регулярно). Надзвичайний серцевий ритм може не бути

ознакою хвороби. Це може бути у здорових дорослих та може бути дуже досить поширеним серед дітей. Однак у деяких ситуаціях надзвичайні ритми можуть бути спричинені серцевими захворюваннями. Якщо ці захворювання виявляються раніше, то лікування ймовірно буде більш ефективним.



Діаграма 1 – Відсотковий розподіл даних за категоріями

На круговій діаграмі вище (Діаграма 1) видно розподіл даних за категоріями: найбільше(що логічно) нормальних звичайних записів серцебиття (60%), далі йде категорія «бурчання» (22%), за нею записи з надзвичайним серцевим ритмом (8%), далі некоректні записи-артефакти (7%) і найменше з категорії додаткового серцевого звуку.

4.3 Аналіз та процесинг даних

Для аналізу звукових даних я використовував спеціальну бібліотеку librosa. Librosa - це відкрите програмне забезпечення для аналізу і обробки аудіосигналів на мовному Python. Вона надає інструменти для виконання задач, таких як отримання мел-спектрограм, розбиття аудіо на фрагменти, виявлення таких елементів як ритм, темп і тональність, екстракція характеристик звуку, аналіз звукового спектра та багато іншого.

```
def show_audio_waveform(audio_sample):
    fig = plt.figure(figsize=(20,5))
    fig = plt.axes()
    librosa.display.waveshow(audio_sample, sr = 22050)
    #plt.title("Sound")
    fig.set_xlabel("Time")
    fig.set_ylabel("Amplitude")
    plt.show()
```

Рис. 5 – Функція для отримання звукової хвилі для звукозапису

На Рис.5 зображена функція, яка звуковий файл конвертує в звукову хвилю. Давайте подивимось, як виглядають звукові хвилі для різних категорій аудіозаписів серцебиття.

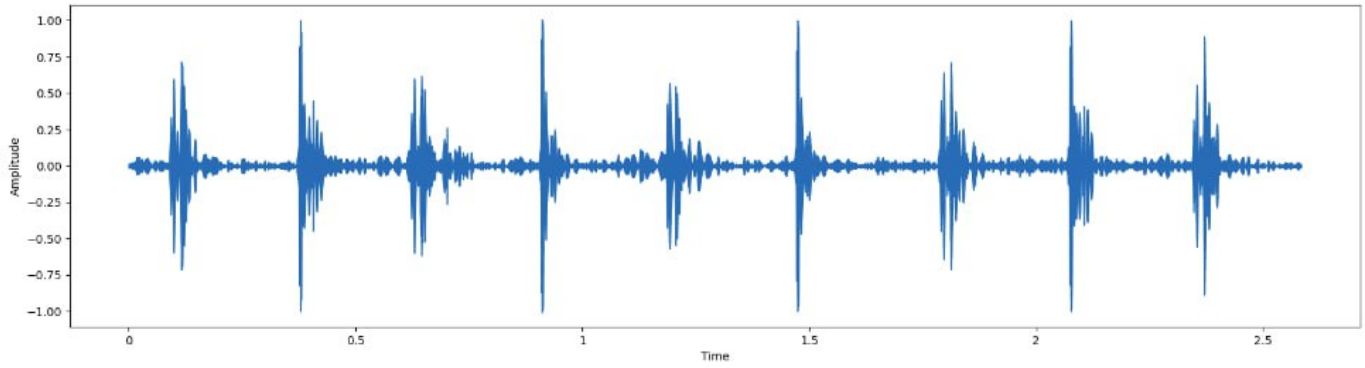


Рис. 6 – Звукова хвиля для нормального серцебиття(normal)

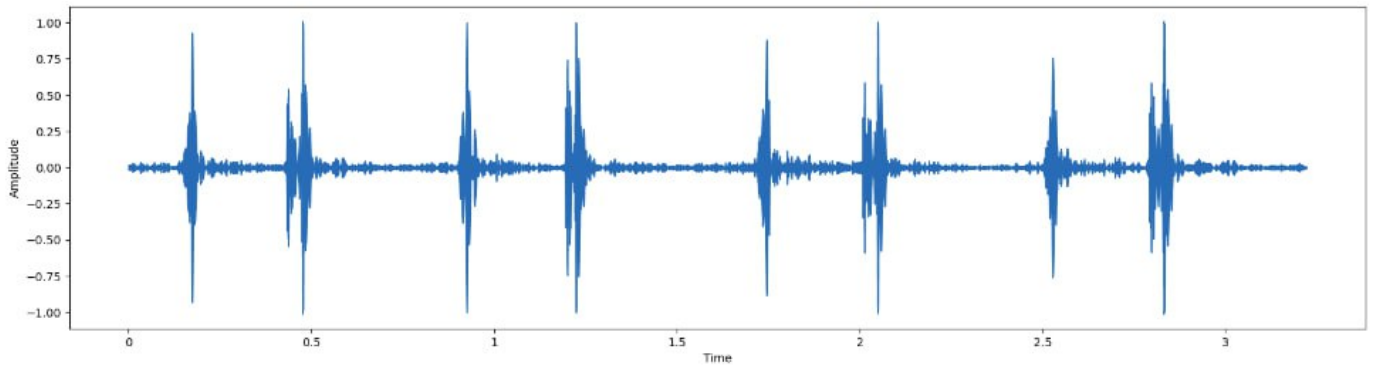


Рис. 7 – Звукова хвиля для серцебиття з бурчанням (murmur)

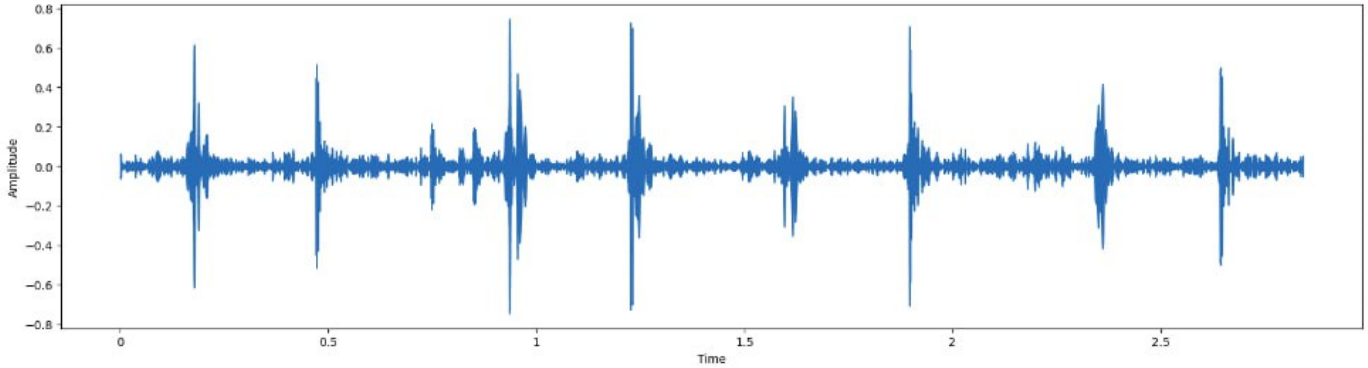


Рис. 8 – Звукова хвиля для серцебиття з надзвичайним серцевим ритмом (extrastole)

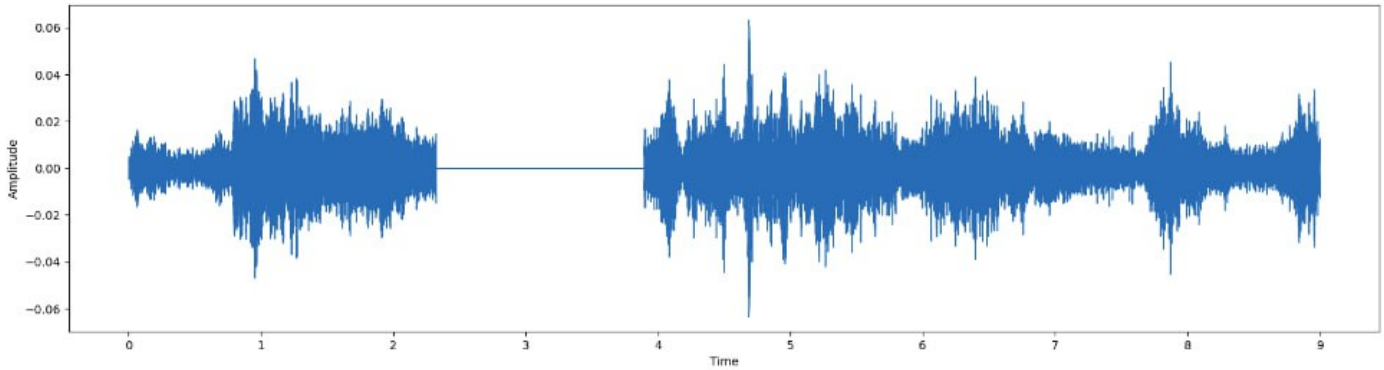


Рис. 9 – Звукова хвиля для неправильних записів (artifacts)

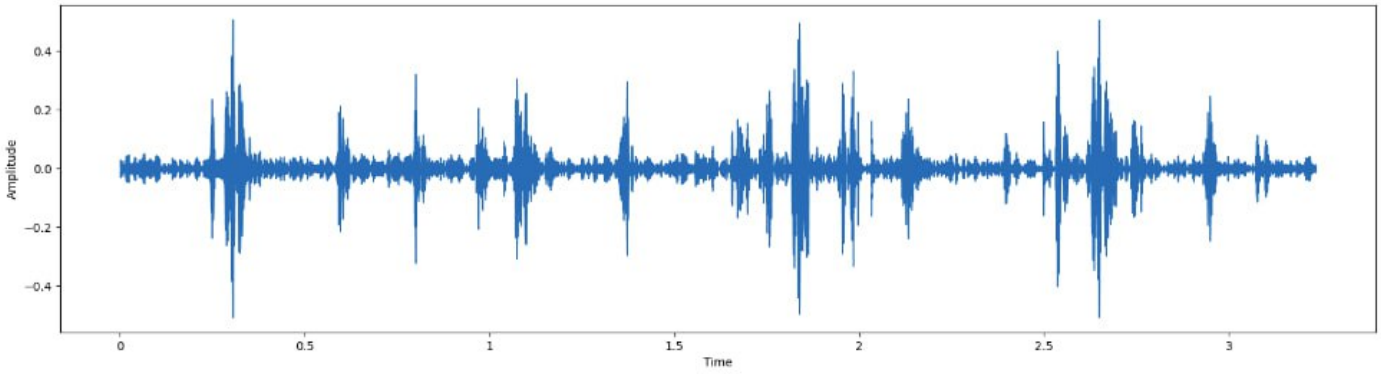


Рис. 10 – Звукова хвиля для серцебиття з додатковим серцевим звуком (extrahls)

Проаналізувавши Рис. 6 неозброєнним оком видно яскравий цикл при нормальному серцебитті. Навіть візуально видно той самий нормальний звук серця, який має чіткий шаблон «луб даб, луб даб», це і є цикл нормального серцебиття.

На Рис. 7 теж видно те саме «бурчання», паузи між «луб» і «даб» нерівномірні, пікові значення та їх довжина різна.

На Рис. 8 теж приблизно проглядаються додаткові або пропущені серцеві скорочення, ті самі, «луб-луб-даб» або «луб-даб-даб». А на Рис. 9 явна демонстрація некоректних даних, які не відображають хоч якесь серцебиття.

На Рис. 10 більше пікових значень, тобто якраз і видно цей додатковий серцевий ритм.

Також у Додатку А винесені усі 5 спектограм на кожну окрему категорію серцевих скорочувань, і видно, загальну циклічність при нормальному серцебитті, повну невідповідність при викидах і деяку асиметричність при інших категоріях.

Далі додаємо невелекий шум, в тих фрагментах даних, де це потрібно (див. Рис. 11).

```
def add_noise(data,x):
    noise = np.random.randn(len(data))
    data_noise = data + x * noise
    return data_noise
```

Рис. 11 – Функція додавання шуму в аудіоряд

Також робимо інші операції серед яких, зокрема, one-hot encoding, для подальшої кращої класифікації (див. Рис. 12).

```
# One-Hot encoding for classes
y_train = np.array(tf.keras.utils.to_categorical(y_train, len(CLASSES)))
y_test = np.array(tf.keras.utils.to_categorical(y_test, len(CLASSES)))
y_val = np.array(tf.keras.utils.to_categorical(y_val, len(CLASSES)))
test_y=np.array(tf.keras.utils.to_categorical(test_y, len(CLASSES)))
```

Рис. 12 – Для значень таргету робимо 5 різних колонок

4.4 Побудова моделей для класифікації і кластеризації часових рядів

В цьому підрозділі ми опишемо саме побудову моделей, а вже оцінка якості та аналіз результатів буде наведено в наступному підрозділі 4.5.

Як згадувалось у попередньому розділі я обрав по 2, найкращих для цієї задачі, на мою думку, основану на багатьох аналізах та статтях, алгоритма кластеризації і класифікації. А саме методи K-means і DBSCAN для кластеризації часових рядів, а KNN і LSTM для класифікації.

Оскільки в нас дані розмічені, а задачі кластеризації відносяться до так званого unsupervised learning (навчання без учителя), то я відповідно видалив усі таргети для використовування кластеризації.

Для методу K-means основним параметром є те саме число K, тобто кількість кластерів. Я задав це число зі значенням 5, і результати були найкращі.

Для алгоритму DBSCAN аналогічно видалив таргети і запустив навчання. На щастя, через не дуже великі об’єми даних DBSCAN навчився досить швидко, бо повільність навчання - є його відомим недоліком.

Для алгоритмів кластеризації я навпаки використовував значень категорій, бо класифікація – це навчання з учителем і тому нам потрібен таргет.

```
# split data into Train, Validation and Test
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, train_size=0.8, random_state=42, shuffle=True)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, train_size=0.9, random_state=42, shuffle=True)
```

Рис. 13 – Розділення даних на тренувальні, тестові та валідаційні

Алгоритм KNN ефективно відпрацював з дефолтними гіперпараметрами, і як показав подальший аналіз для цих даних він є ефективним у співвідношенні якість/швидкості.

І в кінці я аналізував найскладніший, але при цьому найефективніший підхід – рекурентну нейронну мережу LSTM.

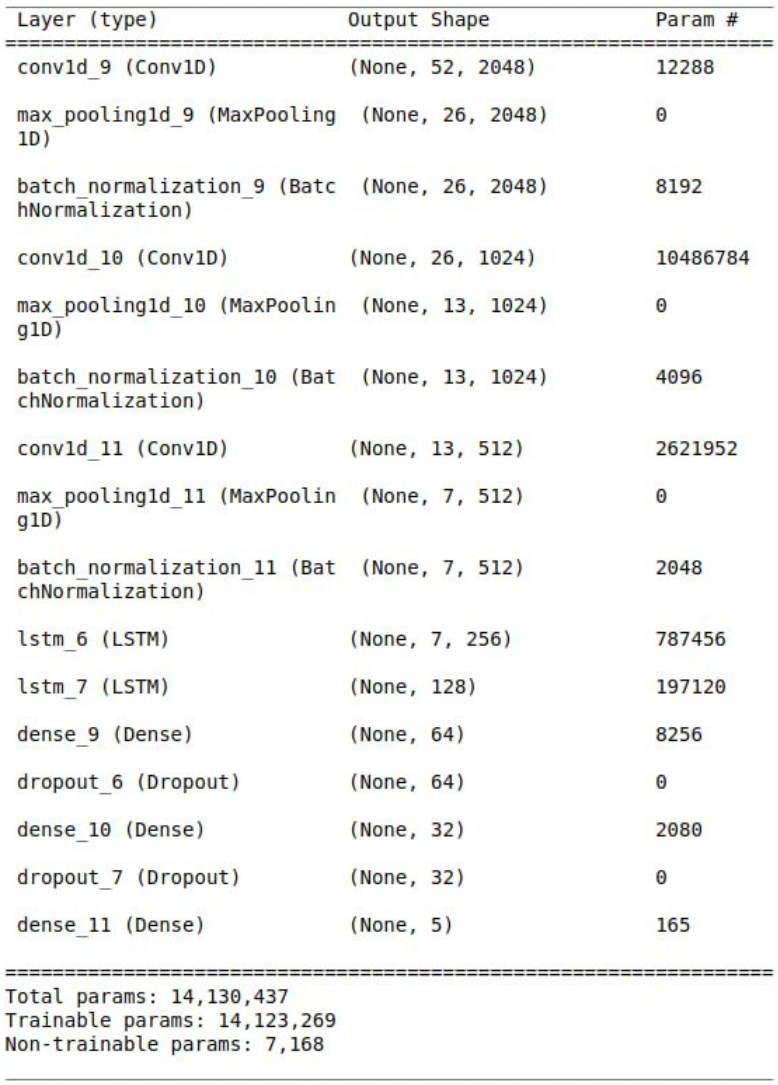
Build Model

```
lstm_model = Sequential()

lstm_model.add(Conv1D(2048, kernel_size=5, strides=1, padding='same', activation='relu', input_shape=(52, 1)))
lstm_model.add(MaxPooling1D(pool_size=2, strides = 2, padding = 'same'))
lstm_model.add(BatchNormalization())

lstm_model.add(Conv1D(1024, kernel_size=5, strides=1, padding='same', activation='relu', input_shape=(52, 1)))
lstm_model.add(MaxPooling1D(pool_size=2, strides = 2, padding = 'same'))
lstm_model.add(BatchNormalization())

lstm_model.add(Conv1D(512, kernel_size=5, strides=1, padding='same', activation='relu'))
lstm_model.add(MaxPooling1D(pool_size=2, strides = 2, padding = 'same'))
lstm_model.add(BatchNormalization())

lstm_model.add(LSTM(256, return_sequences=True))
lstm_model.add(LSTM(128))


lstm_model.add(Dense(64, activation='relu'))
lstm_model.add(Dropout(0.5))

lstm_model.add(Dense(32, activation='relu'))
lstm_model.add(Dropout(0.5))

lstm_model.add(Dense(5, activation='softmax'))

lstm_model.summary()
```

Рис. 14 – Побудова моделі LSTM

Як видно на Рис. 13 це багатошарова рекурентна мережа із функціями активації ReLU, а вже на вихідному шарі активація Softmax.

При аналізі використання меншої кількості шарів давала меншу ефективність моделі, але при цьому через велику загальну кількість параметрів (див. Рис. 14), LSTM навчалась дуже довго і, на жаль, всієї обчислювальної продуктивності мого комп’ютеру не вистачало для більшої кількості епох моделі, бо була пряма залежність від кількості епох і результуючих метрик.



Layer (type)	Output Shape	Param #
conv1d_9 (Conv1D)	(None, 52, 2048)	12288
max_pooling1d_9 (MaxPooling 1D)	(None, 26, 2048)	0
batch_normalization_9 (Batc hNormalization)	(None, 26, 2048)	8192
conv1d_10 (Conv1D)	(None, 26, 1024)	10486784
max_pooling1d_10 (MaxPoolin g1D)	(None, 13, 1024)	0
batch_normalization_10 (Bat chNormalization)	(None, 13, 1024)	4096
conv1d_11 (Conv1D)	(None, 13, 512)	2621952
max_pooling1d_11 (MaxPoolin g1D)	(None, 7, 512)	0
batch_normalization_11 (Bat chNormalization)	(None, 7, 512)	2048
lstm_6 (LSTM)	(None, 7, 256)	787456
lstm_7 (LSTM)	(None, 128)	197120
dense_9 (Dense)	(None, 64)	8256
dropout_6 (Dropout)	(None, 64)	0
dense_10 (Dense)	(None, 32)	2080
dropout_7 (Dropout)	(None, 32)	0
dense_11 (Dense)	(None, 5)	165

Total params: 14,130,437
Trainable params: 14,123,269
Non-trainable params: 7,168

Рис. 15 – Інформація про модель LSTM

4.5 Аналіз отриманих результатів кластеризації часових рядів

Для усіх результатів ми виокремимо дві категорії: результати кластеризації і результати класифікації. Очевидно, що через різні підходи відповідно будуть різні метрики оцінювання якості алгоритмів і моделей.

Для алгоритмів DBSCAN і K-means я обрав чотири метрики, які однаково можуть використовуватись як для К-середніх, так і для DBSCAN: silhouette_score, adjusted_rand_score, davies_bouldin_score та adjusted_mutual_info_score. Вони є широко використовуваними метриками для оцінки якості кластеризації в машинному навчанні.

- Silhouette Score - оцінює наскільки схожі між собою об'єкти в середині кластера та наскільки вони відрізняються від об'єктів в інших кластерах.
- Adjusted Rand Score - оцінює наскільки схожі кластеризації на істинні мітки. Значення 1 означає, що кластери повністю збігаються з істинними мітками.
- Davies-Bouldin Score - оцінює суміш внутрішньокластерної схожості та зовнішньокластерної відмінності для кожного кластера та робить узагальнення для всієї кластеризації. Чим нижче значення, тим краща кластеризація.
- Adjusted Mutual Information Score - оцінює наскільки взаємозв'язок між кластеризацією та істинними мітками відмінний від того, який очікується випадковим чином. Чим більше значення, тим краща кластеризація.

Одна з причин, чому метрики силуета, адаптована взаємна інформація, відстань Девіса-Болдуїна та адаптований рандомізований індекс чистоти підходять для кластеризації часових рядів, полягає в тому, що вони оцінюють якість кластерів, а не якість класифікації.

Ці метрики оцінюють, наскільки добре об'єкти в кожному кластері схожі між собою, і наскільки відмінні вони від об'єктів інших кластерів. Це дуже важливо для часових рядів, оскільки вони мають складну структуру та можуть

мати різні форми. Метрики також дозволяють оцінити, чи є розділення на кластери зрозумілим та придатним для подальшого аналізу.

Таким чином, ці метрики допомагають зробити висновки про якість кластеризації та знайти найкращі параметри для алгоритмів кластеризації.

Тепер давайте проаналізуємо результати:

Для K-середніх:

- Silhouette Score = 0.345
- Adjusted Rand Score = 0.006
- Davies-Bouldin Score = 0.876
- Adjusted Mutual Information Score = 0.049

Для DBSCAN:

- Silhouette Score = -0.6336115
- Adjusted Rand Score = 0.013803153550193674
- Davies-Bouldin Score = 1.339207713289357
- Adjusted Mutual Information Score = 0.037943281139379566

За результатами метрик можна сказати, що алгоритм K-means працює краще, ніж DBSCAN, для цього конкретного набору даних.

Він досягнув значень Silhouette Score близько до 0.34, що свідчить про те, що кластери добре відокремлені один від одного. Оцінка Adjusted Rand Score для K-means низька, а це може бути пов'язано з тим, що дані можуть містити шум, або кластеризація може бути досить складною.

Davies-Bouldin Score для K-means більше 0.87, що вказує на те, що кластери забезпечують добру відмінність один від одного, але можуть бути не такими оптимальними, як би ми хотіли. Adjusted Mutual Information Score для

K-means також низький, що свідчить про низький рівень взаємозалежності між вхідними даними та отриманими кластерами.

З іншого боку, результати метрик для DBSCAN не такі високі. Silhouette Score близький до -0.63, що показує, що кластери майже не відокремлені один від одного.

Оцінка Adjusted Rand Score для DBSCAN дещо вища, ніж для K-means, але все ще низька. Це означає, що кластери можуть мати різну кількість елементів і не відповідати оригінальним міткам. Davies-Bouldin Score для DBSCAN більше 1.33, що свідчить про те, що кластери не дуже відокремлені один від одного. Adjusted Mutual Information Score для DBSCAN також низький, що свідчить про низький рівень взаємозалежності між вхідними даними та отриманими кластерами.

Отже, можна зробити висновок, що алгоритм K-means більш ефективний для даного набору даних, оскільки досягнув більш високих значень метрик, але також варто зазначити, що результати метрик все рівно є високими, що свідчить про правильно обрані алгоритми та методи для кластеризації записів людського серцебиття.

4.6 Аналіз отриманих результатів класифікації часових рядів

Для оцінки алгоритму класифікації я обрав найпопулярніші метрики для класифікації: accuracy, f1, precision, recall та матриця помилок (confusion matrix).

Метрики accuracy, f1, precision, recall та матриця помилок (confusion matrix) дуже добре підходять для задач класифікації часових рядів через те, що

вони дають змогу оцінити якість класифікації на різних рівнях: загальну точність (accuracy), точність визначення позитивних класів (precision), точність визначення негативних класів (recall) та збіги та розбіжності в класифікації кожного з класів (confusion matrix). Враховуючи, що класифікація часових рядів є завданням з високою вимогою до точності та чутливості, використання цих метрик є дуже важливим для оцінки результатів роботи алгоритмів.

Для KNN:

Accuracy: 0.658, F1 score: 0.651, Recall: 0.658, Precision: 0.667

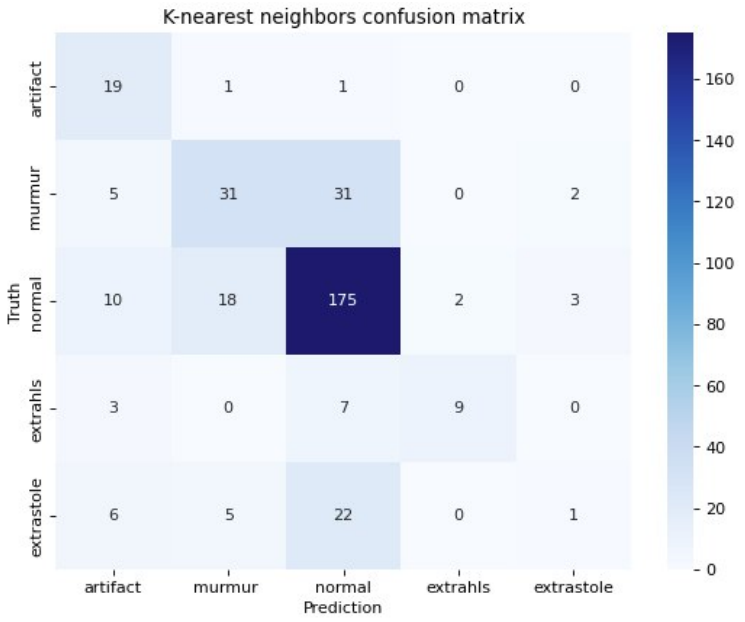


Рис. 16 – Матриця помилок для алгоримту KNN

Для LSTM:

Accuracy: 0.681, F1 score: 0.693, Recall: 0.681, Precision: 0.713

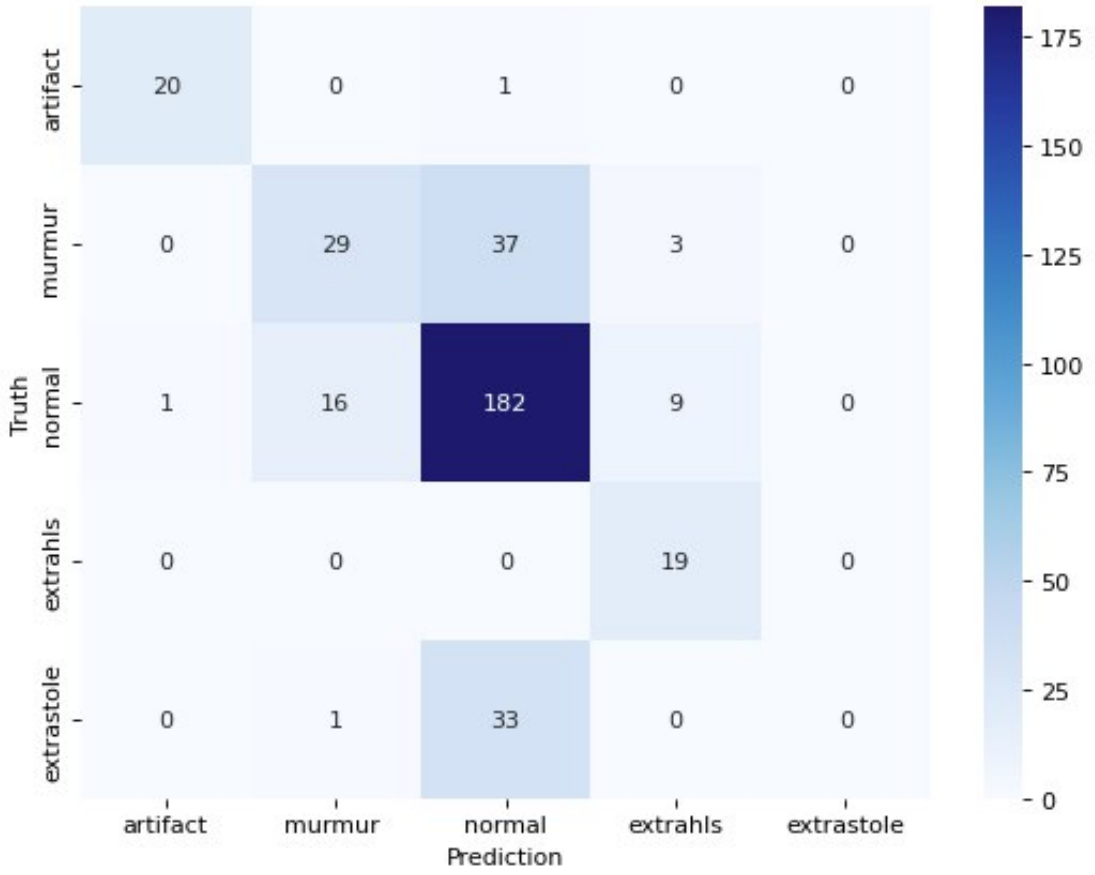


Рис. 17 – Матриця помилок для моделі LSTM

Дані результати метрик Accuracy, F1 score, Recall та Precision є високими для алгоритму K-Nearest Neighbors (KNN) у багатокласовій класифікації часових рядів.

Значення Accuracy вказує на те, що наш алгоритм правильно класифікував 65,8% зразків датасету, що є досить високим результатом. F1 score є гармонічним середнім між точністю (Precision) та повнотою (Recall), і також демонструє високу точність та повноту класифікації нашого алгоритму. Значення Precision та Recall становлять відповідно 0,667 та 0,658, що також є досить високими показниками.

Отже, високі результати метрик свідчать про ефективність використання алгоритму KNN для багатокласової класифікації часових рядів.

Як ми бачимо, результати метрик для LSTM не набагато краще, і може здатись, що алгоритм KNN, який простіший і швидший, краще підходить для даного набору даних, але це не так. Основна проблема полягає в тому, що мій комп’ютер не може обробити велику кількість епох і обривається лише на декількох десятках епох, бо немає достатніх обчислювальних можливостей. Таким чином можна з упевненістю заявити, що отримані результати для LSTM можна значно покращити.

Отже, обрані алгоритми дуже ефективно класифікували дані в обраному датасеті і, дійсно, гарно класифікують часові ряди.

ВИСНОВКИ

У цій науковій роботі роботі було досліджено актуальність обраної теми, її важливість в сьогоденні, зроблено докладний огляд використаних алгоритмів і підходів, а також застосування цих методів кластеризації та класифікації часових рядів на прикладі звукових записів серцебиття людей.

Для кластеризації були використані алгоритми k-means та DBSCAN, що дозволило розділити записи на кілька категорій залежно від характеристик звуків. Для класифікації були використані алгоритми KNN та LSTM, що дозволило відрізнити звукові записи різних категорій та визначити, до якої категорії відноситься конкретний запис.

Отримані результати свідчать про ефективність використаних методів для аналізу звукових записів серцебиття людей та можуть бути використані для діагностики різних захворювань серця. Дослідження можуть бути продовжені з використанням інших алгоритмів та датасетів з метою поліпшення точності класифікації та кластеризації.

Таким чином, моя робота показала, що кластеризація та класифікація часових рядів з використанням алгоритмів k-means, DBSCAN, KNN і LSTM є ефективним методом для аналізу даних серцевих звуків.

Проведене дослідження показало, що з використанням цих алгоритмів можна ефективно визначити категорії серцевих звуків, що в свою чергу може виявитися корисним для розробки нових методів діагностики та лікування серцевих захворювань.

Зокрема, алгоритм k-means дозволяє кластеризувати дані серцевих звуків за їх характеристиками та дозволяє виявляти спільні риси між різними звуками. DBSCAN може бути корисним у виявленні аномальних звуків та відокремленні їх від нормальних. Алгоритм KNN забезпечує ефективну класифікацію звуків за їх характеристиками, тоді як LSTM може використовуватися для класифікації звуків на основі їх часових характеристик.

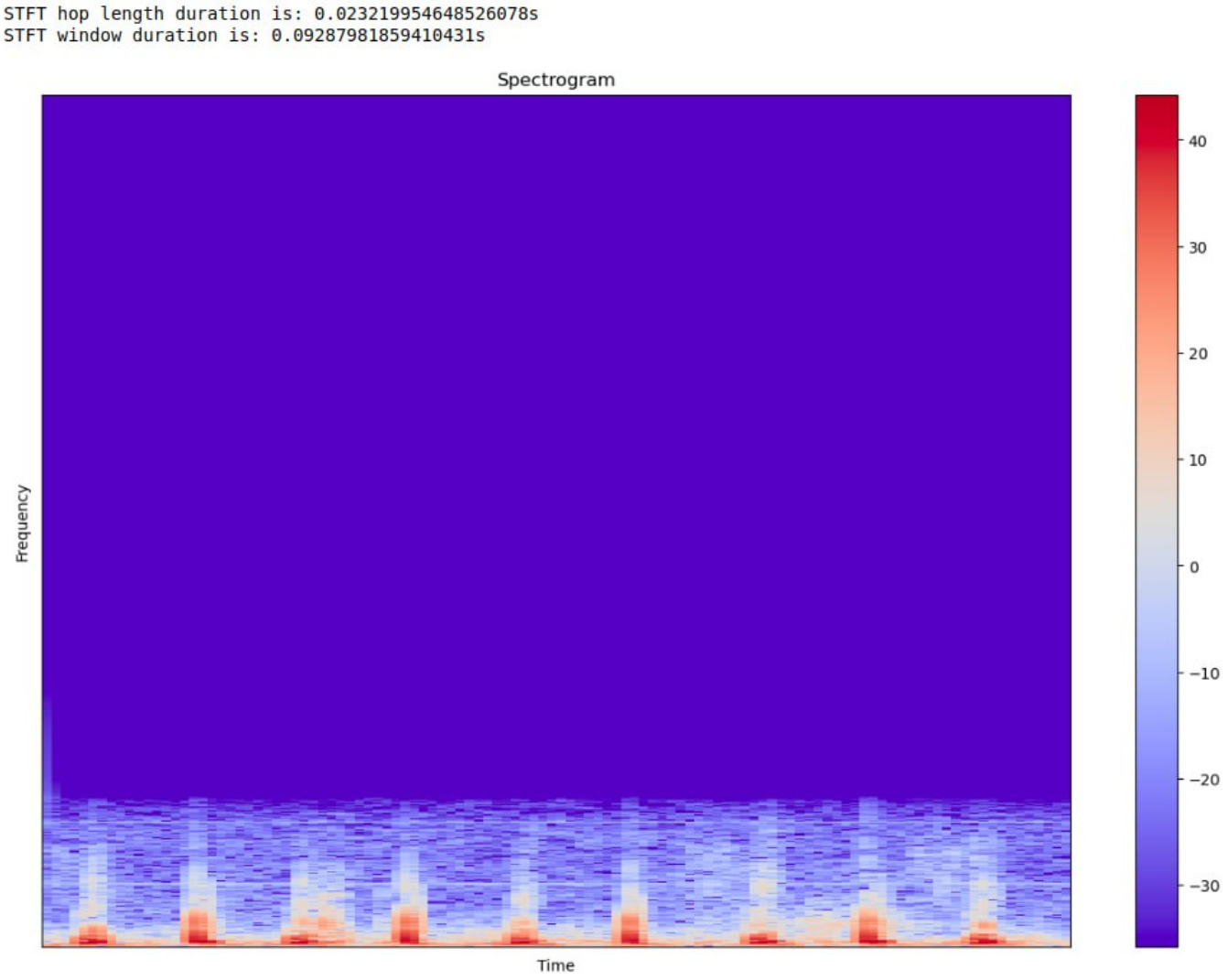
Отже, в даній роботі було успішно використано кластеризацію та класифікацію часових рядів для аналізу даних зі звуковими записами серцебиття людей, ділячи їх на категорії. Результати цієї роботи можуть бути корисними для медичних досліджень та можуть допомогти у розробці нових методів діагностики та лікування серцевих захворювань, а якщо їх продовжувати і розвивати, то і стати справді революційним методом у діагностуванні та виявленні захворювань серця і судинної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Box, George; Jenkins, Gwilym (1976). *Time Series Analysis: forecasting and control, rev. ed*. Oakland, California: Holden-Day.
2. Hamilton, James (1994). *Time Series Analysis*. Princeton University Press. ISBN 0-691-04289-6.
3. URL: medium.com/@brainrus/как-работает-shazam-интервью-с-разработчиками-dec08618c1ed
4. A. B. Montazeri and H. Abdollahzadeh. Non-stationary Time Series Analysis and Applications, Springer, 2014.
5. URL: forbes.ua/money/v-ukrainu-zavezli-generatoriv-yak-odin-energoblok-aes-a-benzinu-potribno-na-sotni-milyoniv-yak-pobudovana-generatorna-ekonomika-krai ni-09012023-10939
6. URL: vse.energy/news/pek-news/electro/1940-ee-consumption-12
7. URL: arxiv.org/pdf/1810.11624.pdf
8. URL: csnt.nau.edu.ua/files/2023/sbirnyk2023.pdf
9. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
10. Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232.
11. Dasarathy, B. V. (1991). Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press.

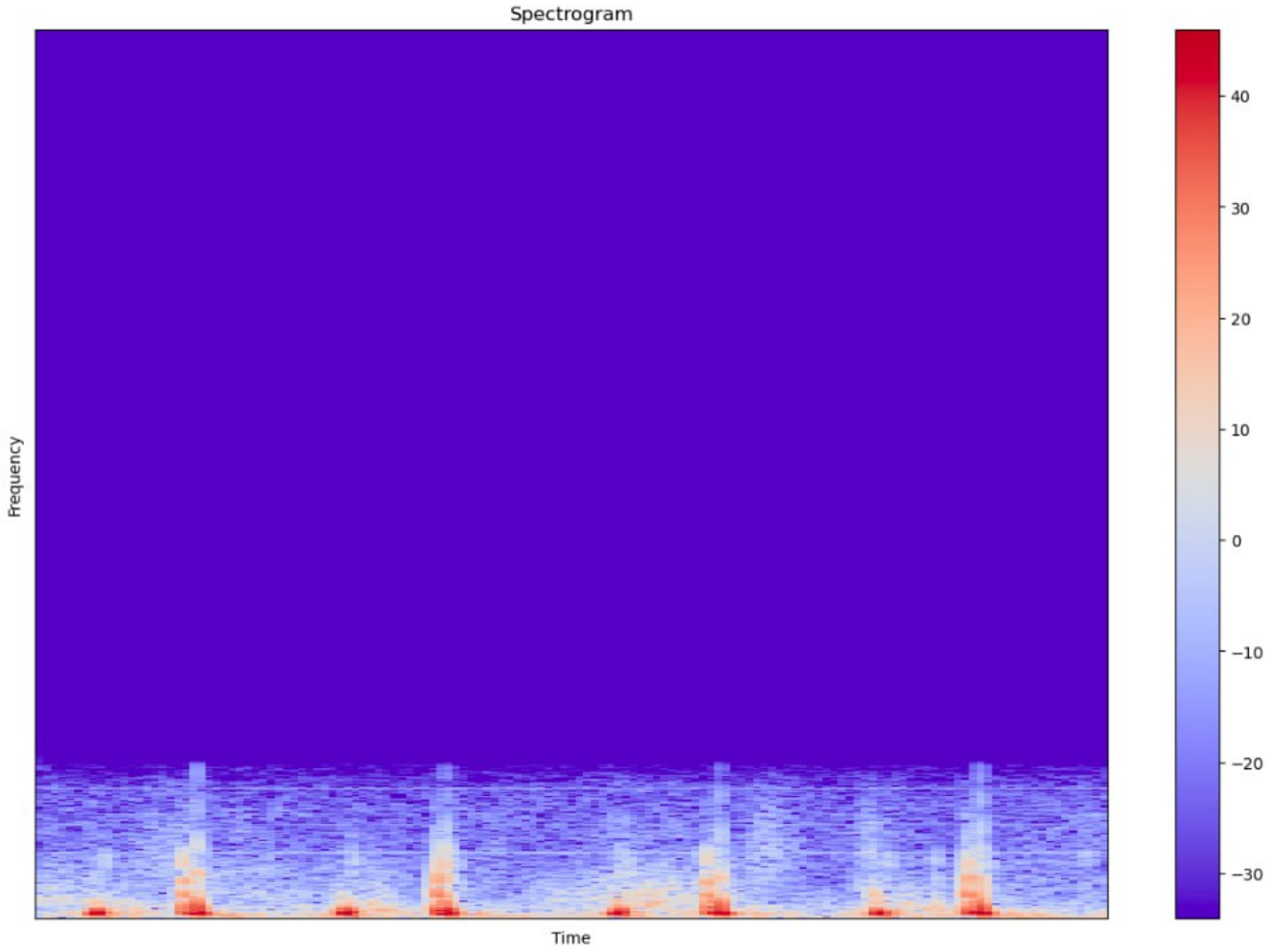
12. Zhang, Z. (2004). Nearest neighbor search algorithms and applications. Springer.
13. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.
14. Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. IEEE Transactions on Neural Networks, 16(3), 645–678.
15. Martin Ester, Jörg Sander (1996). "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications". Data Mining and Knowledge Discovery. 2 (2): 169–194.
16. Schubert, E., Sander, J., Ester, M., Kriegel, H.-P., & Xu, X. (2017). "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN". ACM Transactions on Database Systems (TODS), 42(3), 19.Schubert, E., Sander, J., Ester, M., Kriegel, H.-P., & Xu, X. (2017). "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN". ACM Transactions on Database Systems (TODS), 42(3), 19.
17. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.
18. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
19. URL: arxiv.org/abs/1810.07758
20. URL: www.peterjbentley.com/heartchallenge/

ДОДАТОК А



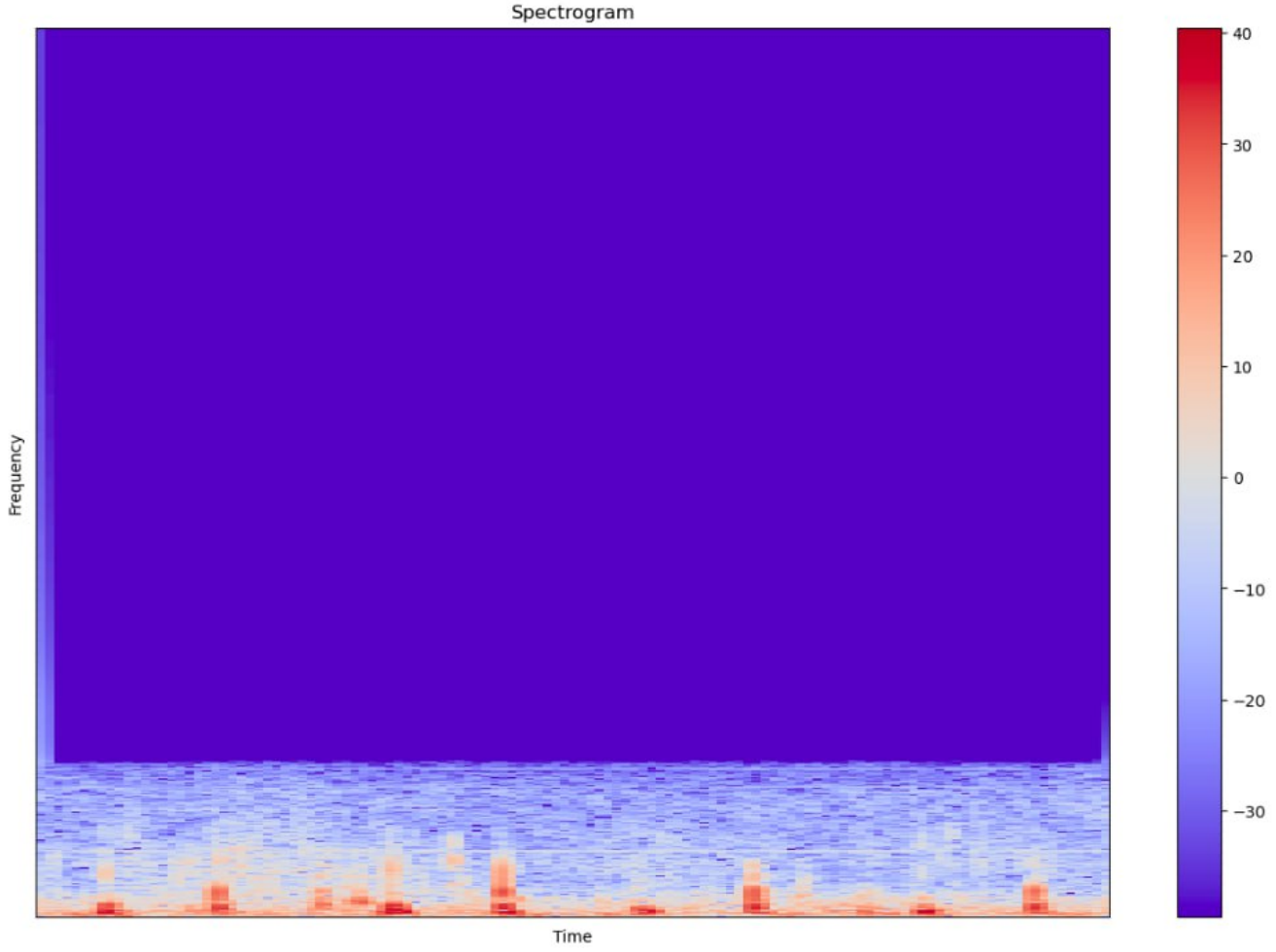
Спектограма для нормального серцебиття

```
STFT hop length duration is: 0.023219954648526078s
STFT window duration is: 0.09287981859410431s
```



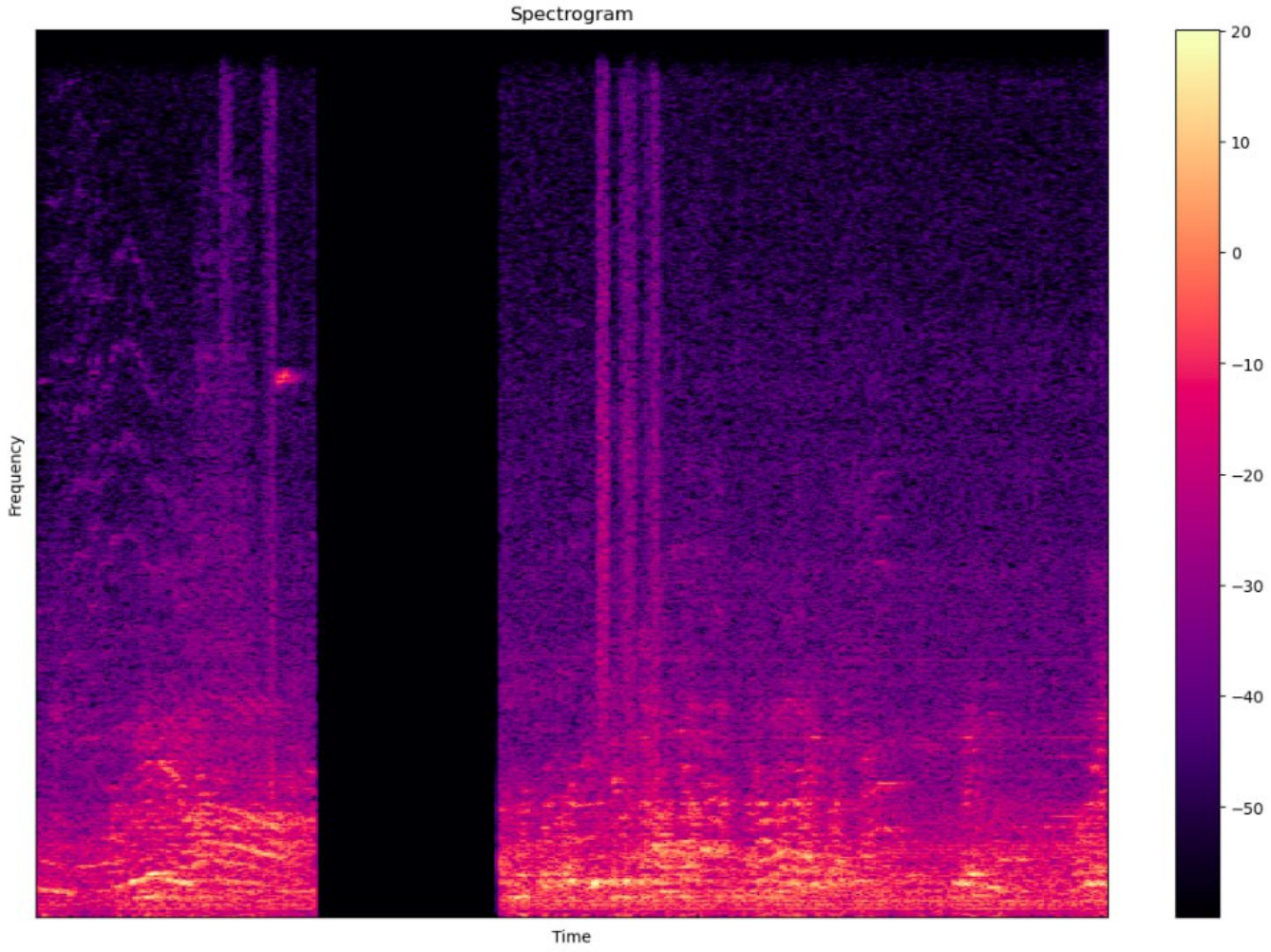
Спектограма для серцебиття з бурчанням

```
STFT hop length duration is: 0.023219954648526078s
STFT window duration is: 0.09287981859410431s
```



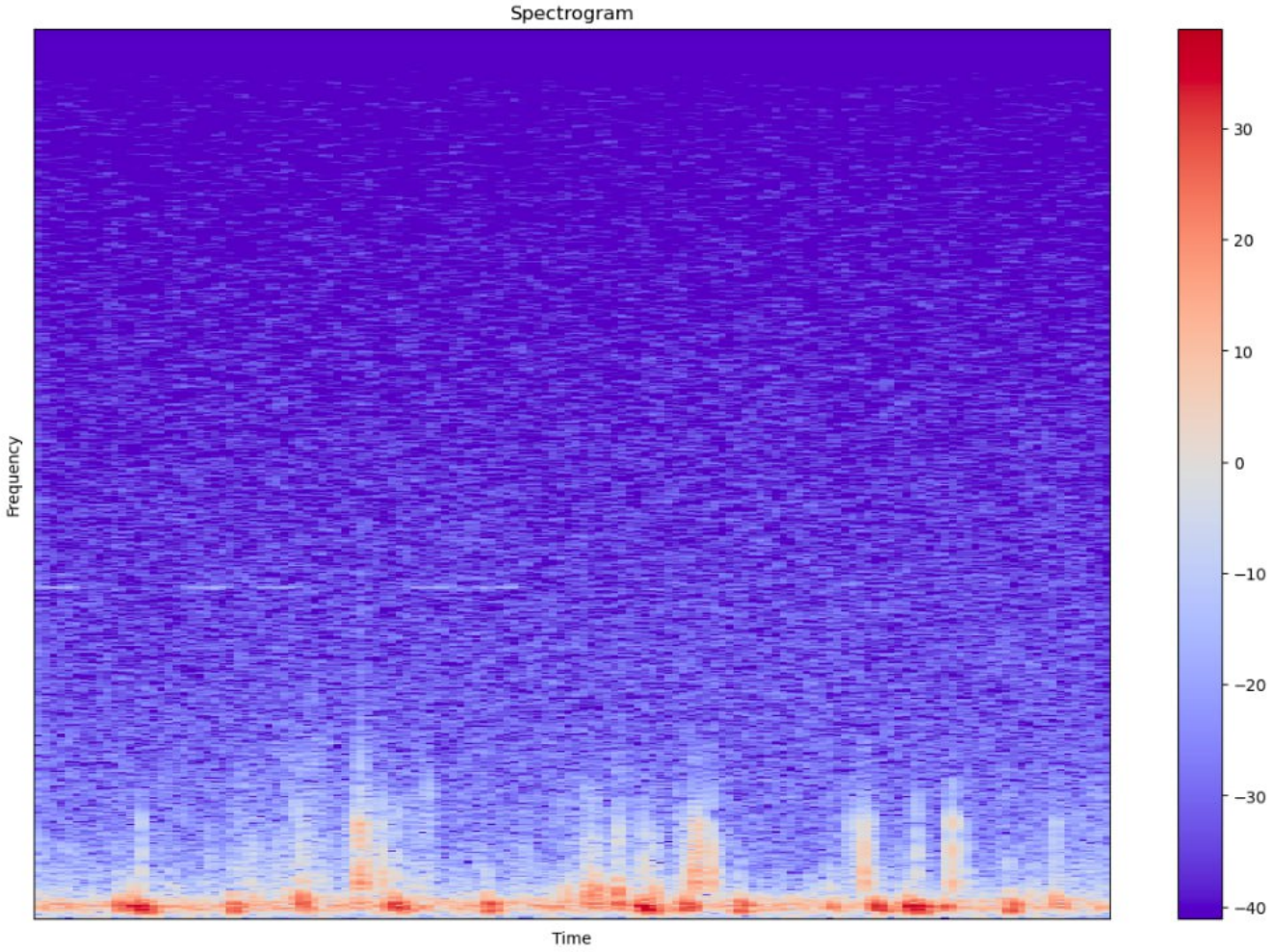
Спектограма надзвичайного серцевого ритму

```
STFT hop length duration is: 0.023219954648526078s
STFT window duration is: 0.09287981859410431s
```



Спектограма викидів даних

```
STFT hop length duration is: 0.023219954648526078s
STFT window duration is: 0.09287981859410431s
```



Спектограма додаткового серцевого звуку