УДК 519.6:004.94
DOI https://doi.org/10.24144/2616-7700.2025.46(1).262-272

**В. М. Самусь[1], Є. І. Самусь[2]**

[1] ДВНЗ «Ужгородський національний університет»,
аспірант кафедри системного аналізу та теорії оптимізації
vasyl.samus@uzhnu.edu.ua
ORCID: https://orcid.org/0000-0002-1682-689X

[2] ДВНЗ «Ужгородський національний університет»,
ст. викладач кафедри комп'ютерних систем та мереж
yevgenija.samus@uzhnu.edu.ua
ORCID: https://orcid.org/0009-0003-9601-289X

# ФОРМАЛІЗАЦІЯ ОЗНАК ЕКГ-СИГНАЛІВ ДЛЯ ПОБУДОВИ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

ЕКГ, або електрокардіограма, — це метод неінвазивного дослідження, який дозволяє зафіксувати електричну активність серця за допомогою спеціальних електродів, розташованих на шкірі. Цей метод використовується для оцінки ритму серця, виявлення порушень електричної провідності, а також для діагностики різноманітних серцевих патологій, таких як ішемія, інфаркт міокарда та аритмії. Завдяки своїй простоті та доступності, ЕКГ є одним із основних інструментів у кардіології для моніторингу стану пацієнта як у стаціонарних, так і в амбулаторних умовах.

Підготовка сигналу ЕКГ до машинного навчання — це комплекс процесів, що включає кілька ключових етапів для забезпечення високої якості даних і їх оптимальної придатності для аналізу. Спочатку проводиться попередня обробка сигналу, яка полягає у фільтрації для усунення шумів та артефактів, наприклад, базового дрейфу чи високочастотних перешкод. Далі виконується нормалізація, яка забезпечує однорідність даних, приводячи амплітуди сигналу до спільного масштабу та знижуючи вплив зовнішніх факторів.

Наступним етапом є сегментація, коли безперервний сигнал розбивається на окремі фрагменти навколо піків R, що дозволяє виділити окремі комплекси серцевих скорочень. Після цього проводиться виділення ознак: з кожного сегмента вилучаються ключові характеристики, такі як тривалість інтервалів, амплітуди окремих хвиль, а також часово-частотні параметри. Ці ознаки допомагають моделі розрізняти нормальні та патологічні стани серця. Завершальним кроком є перетворення обробленого сигналу у формат, зручний для подачі в алгоритми машинного навчання. Такий комплексний підхід дозволяє ефективно навчати моделі розпізнавати закономірності в роботі серця, що є основою для діагностики та прогнозування різних кардіологічних станів.

Дана стаття присвячена побудові алгоритмів для підготовки даних цифрової кардіограми до машинного навчання. Розроблений алгоритм дозволяє отримати статистичні характеристики цифрової кардіограми, що дає можливість використати їх в подальшому аналізі з використанням методів машинного навчання.

**Ключові слова:** електрична цифрова кардіограма, ектопічність, статистичні характеристики, машинне навчання, сегментація.

## 1. Вступ.

**Постановка проблеми.** Сигнал ЕКГ характеризується високим рівнем шумів, артефактів та природною варіабельністю між пацієнтами, що ускладнює його безпосереднє застосування в алгоритмах машинного навчання. Необхідно розробити комплексний підхід, який включає попередню обробку сигналу з метою видалення небажаних компонентів, нормалізацію для приведення даних до

єдиного масштабу, а також сегментацію сигналу на окремі комплекси для точного визначення ключових елементів, таких як піки R, комплекси QRS, QS, P та T хвилі.

**Аналіз останніх досліджень і публікацій.** Пошук матеріалів для статті було здійснено за термінами «ECG», «ventricular», «supraventricular», «fibrilation» та «machine learning» у період з січня 2010 року по лютий 2025 року. Обрані дослідження зосереджені на визначенні та побудові базових принципів формування математичної моделі, що стосуються як окремих аспектів електрокардіографічних сигналів, так і можливостей застосування методів машинного навчання для їх аналізу [2, 5, 7]. Більшість публікацій мають вузьку спеціалізацію, розглядаючи частинні випадки, що дозволяє виокремити окремі характеристики, які можна інтегрувати у загальну базову модель. Проте практичні аспекти реалізації математичної моделі з використанням сучасних програмних засобів, зокрема при застосуванні машинного навчання для автоматизованого аналізу сигналів, залишаються відкритими і потребують подальшого дослідження.

**Мета статті.** Метою даної статті є дослідження практичних аспектів підготовки даних електрокардіографії (ЕКГ) для застосування алгоритмів машинного навчання. Особлива увага приділяється використанню відкритих наборів даних ЕКГ, які забезпечують доступ до великої кількості даних для тренування, тестування та валідації моделей.

**2. Виклад основного матеріалу.** З метою структурованого підходу до аналізу даних ЕКГ та побудови ефективних алгоритмів необхідно створити математичну модель процесу. Для її реалізації ми використовуватимемо мову програмування Python, що вирізняється широким спектром бібліотек для обробки даних, машинного навчання та аналізу сигналів. Крім того, інтеграція відкритих датасетів ЕКГ дозволить не лише розширити обсяг даних для тренування і валідації моделей, але й забезпечити більш глибокий аналіз та перевірку отриманих результатів.

В якості джерела відкритих датасетів було обрано набори MIT-BIH Database. Цей набір даних був розроблений у спільній роботі Массачусетського технологічного інституту (MIT) та Beth Israel Deaconess Medical Center [9] і широко використовується в медичних дослідженнях завдяки високій якості записів і різноманіттю патологічних випадків.

Для початку визначимо основні моделі для даних.

```
class PhysioNetDataset(str, Enum):
    MIT_BIH_ARRHYTHMIA = "mitdb"  # Аритмії
    MIT_BIH_AFIB = "afdb"  # Фібриляція передсердь
    MIT_BIH_SUPRAVENTRICULAR = "svdb"  # Надшлуночкові аритмії
    MIT_BIH_NORMAL = "nsrdb"  # Нормальний ритм
    MIT_BIH_MALIGNANT_VENTRICULAR = "vfdb"  # Шлуночкові аритмії
```

Цей клас описує набір датасетів з PhysioNet, які використовуються для аналізу екстрасистол. Наприклад, значення "mitdb" відповідає аритміям, "afdb" — фібриляції передсердь, "svdb" — надшлуночковим аритміям, "nsrdb" — нормальному ритму, а "vfdb" представляє набір даних, пов'язаний із шлуночковою фібриляцією. Використання такого підходу спрощує роботу з різними джерелами даних, забезпечує однозначну ідентифікацію кожного набору.

Датасет організовано за допомогою формату WFDB і складається з декіль-

кох типів файлів, що забезпечують як сам сигнал, так і супутню анотаційну інформацію. Основні компоненти структури даних:

Файл заголовків (.hea): містить метадані ЕКГ, зокрема кількість сигналів, частоту дискретизації, кількість семплів, параметри перетворення аналогового сигналу у цифровий (наприклад, коефіцієнти масштабування, базову лінію тощо).

Файл анотацій (.atr або .ann): містить інформацію про кардіологічні події, зокрема, позиції (номери семплів), де відбуваються окремі удари, а також їх класифікацію (наприклад, нормальний ритм, різні види аритмій тощо). Анотації дозволяють пов'язати конкретні фрагменти сигналу з подіями, що є ключовим для розробки та валідації алгоритмів автоматичного аналізу.

Файл даних (.dat): зберігає необроблені цифрові значення сигналів. Зазвичай дані представлені у вигляді 12- або 16-бітових чисел, що відображають амплітуду електрокардіографічного сигналу для кожного каналу. Як приклад, наведемо частину ЕКГ з датасету svdb #821, що зображено на рисунку 1:

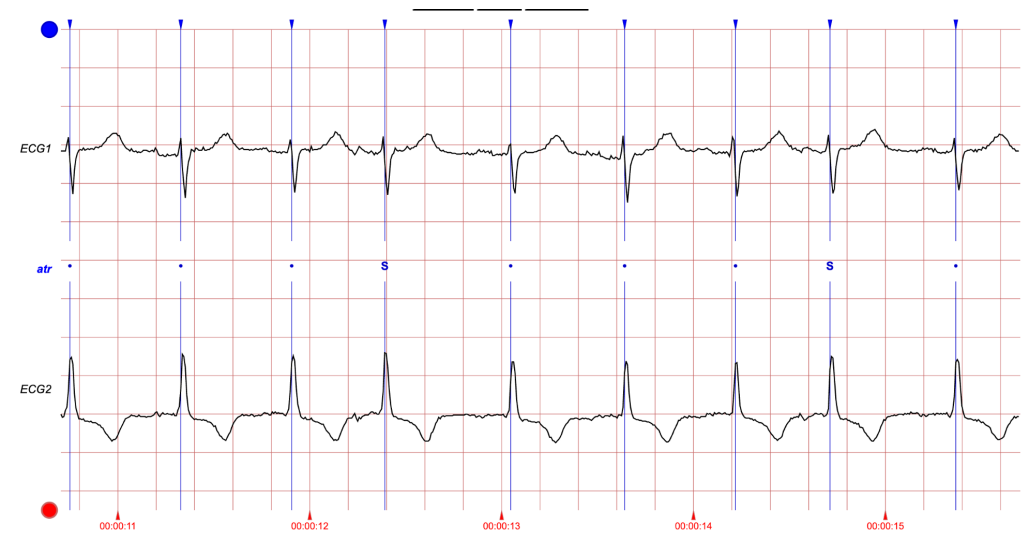


Рис. 1. Фрагмент ЕКГ.

На рисунку бачимо наявність сигналу у двох відведеннях - позначених відповідно ECG1 та ECG2[10]. Додатково нанесено анотацію для кожного удару. В даному випадку нормальний сегмент позначається крапкою, а вироджений відповідною літерою(s - supraventricular). Для роботи з типами серцевих скорочень створимо клас:

```
class BeatAnnotation(str, Enum):
    # Normal & Bundle branch
    NORMAL = "N"   # Normal beat
    LEFT_BBB = "L"   # Left bundle branch block beat
    RIGHT_BBB = "R"   # Right bundle branch block beat
    # Atrial
    ATRIAL_PREMATURE = "A"   # Atrial premature beat
    ABERRATED_ATRIAL = "a"   # Aberrated atrial premature beat
    ATRIAL_ESCAPE = "e"   # Atrial escape beat
    # Supraventricular
    SUPRAVENTRICULAR_PREMATURE = "S"   # Supraventricular beat
```

```
    # Ventricular
    PVC = "V"  # Premature ventricular contraction
    V_ESCAPE = "E"  # Ventricular escape beat
   ...

   @classmethod
    def normal(cls):
        return {cls.NORMAL, cls.LEFT_BBB, cls.RIGHT_BBB}

    @classmethod
    def atrial(cls):
 return {cls.ATRIAL_PREMATURE, cls.ABERRATED_ATRIAL, cls.
   ATRIAL_ESCAPE}

    @classmethod
    def supraventricular(cls):
        return {cls.SUPRAVENTRICULAR_PREMATURE}

    @classmethod
    def ventricular(cls):
        return {cls.PVC, cls.V_ESCAPE}
...
```

Кожен елемент переліку представляє конкретний тип удару, наприклад, "N"для нормального ритму, "L"та "R"для блокад лівої або правої гілки, "V"для шлуночкової екстрасистоли тощо. Крім того, клас включає методи, що групують окремі типи ударів за категоріями (normal, atrial, supraventricular, ventricular).

На наступному етапі створимо клас для завантаження сигналів із вказаних датасетів та їх локального кешування.

```
class PhysioNetLoader:
    def __init__(self, dataset: PhysioNetDataset):
        self._dataset = dataset.value
        os.makedirs(f"{DATA_DIR}/{self._dataset}", exist_ok=True)

    @property
    def _data_path(self):
        return os.path.join(DATA_DIR, self._dataset)

    def get_local_record_list(self, only_dat=False):
        dat_files = get_file_names_in_dir(self._data_path, ".dat")
        if only_dat:
            return dat_files
        else:
           atr_files = get_file_names_in_dir(self._data_path, ".atr
   ")
           hea_files = get_file_names_in_dir(self._data_path, ".hea
   ")
            return dat_files \& atr_files \& hea_files

    def download_record(self, record_id):

        try:
            os.chdir(self._data_path)
            wfdb.dl_files(
```

```
                self._dataset,
                dl_dir=".",
                files=[
                    f"{record_id}.dat",
                    f"{record_id}.hea",
                    f"{record_id}.atr",
                ],  # noqa E501
            )
            print(f"Запис {record_id} завантажено!}
            ")
        except FileNotFoundError:
            print(f"Запис {record_id} не знайдено")
```

Основними його можливостями є отримання запису або з локального кешу, або завантаження безпосередньо з PhysioNet.

Отримання списку записів: метод get_local_record_list дозволяє отримати список записів, що вже завантажені локально, за умови наявності файлів з розширеннями .dat, .atr і .hea. Метод get_record_list використовує функцію з бібліотеки wfdb для отримання списку всіх записів із вибраного датасету безпосередньо з PhysioNet.

Завантаження записів: метод download_record завантажує окремий запис за вказаним ідентифікатором. Метод download_all_records автоматизує завантаження всіх записів із датасету, перебираючи отриманий список записів. Метод load_out_record завантажує запис з PhysioNet, якщо його немає локально, а також зберігає його локально за потребою. Метод load_record перевіряє наявність запису локально та, залежно від цього, викликає відповідний метод завантаження.

Завантаження та збереження локальних записів: load_local_record завантажує запис, його анотації та заголовок із локальної директорії. check_record_exists перевіряє, чи існують у локальному сховищі всі необхідні файли для запису. Методи save_record_local, save_annotations_local та save_headers_local відповідають за збереження відповідних компонентів запису у локальну директорію.

Створивши інструмент для отримання записів, можна перейти до підготовки даних до навчання. Для навчання потрібно сформувати два масиви - матрицю ознак та цільову змінну. Вони відповідно позначаються X та y. X представляється у вигляді матриці, де кожен рядок відповідає окремому зразку (спостереженню), а кожен стовпець — конкретній ознаці. Відповідно y є вектором, де кожне значення відповідає одному зразку в X. Ці позначення є стандартними, оскільки вони застосовуються як у випадку простих моделей, так і у більш складних глибоких нейронних мережах. В процесі дослідження розглядалися 3 способи побудови матриці ознак. Першим способом є передача на вхід цілого фрагменту ЕКГ, який відповідає конкретному церцевому удару. В такому випадку розмірність масиву X: X.shape = (n_samples, n_features), де n_features залежить від частоти сигналу та тривалості серцевого скорочення. Графічно це можна зобразити наступним чином (див. рисунок 2):

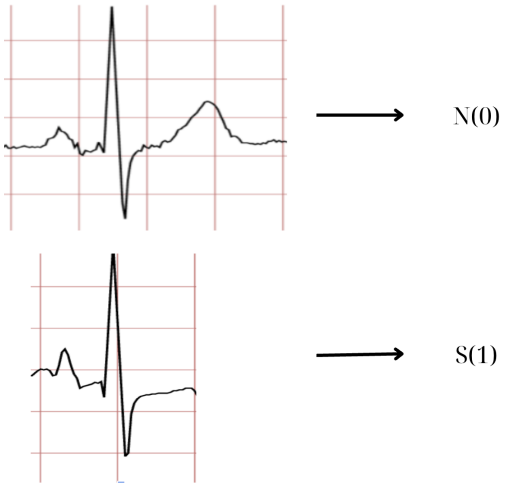


Рис. 2. Нормальне та ектопічне скорочення.

Визначимо клас, який буде описувати цю модель:

```
class PrepareECGToLearn:
    def __init__(
        self,
        dataset: PhysioNetDataset,
        beat_label_map: BeatAnnotationLabelMap,
        samples=1,
    ):
        self._dataset = dataset
        self._beat_label_map = beat_label_map
        self._loader = PhysioNetLoader(self._dataset)
        self._samples = samples
        self._records = {}
        self._annotations = {}
        self._headers = {}

    def _load_signals(self, rec_name):
        record = self._records.get(rec_name)
        annotation = self._annotations.get(rec_name)
        header = self._headers.get(rec_name)
        if not (record and annotation and header):
            record, annotation, header = self._loader.load_record(
                rec_name, save_locally=True
            )  # noqa 501
            self._records[rec_name] = record
            self._annotations[rec_name] = annotation
            self._headers[rec_name] = header
        return record, annotation, header

    @staticmethod
    def get_segment_label(segment, ann_samples, ann_symbols):
        start = segment["Index"].min()
        end = segment["Index"].max()
        indices = np.where((ann_samples >= start) \& (ann_samples
    <= end))[0]
        if len(indices) > 0:
            center = (start + end) // 2
            closest_idx = indices[
```

```
                np.argmin(np.abs(ann_samples[indices] - center))
            ]  # noqa 501
            return ann_symbols[closest_idx]
        else:
            return None

    def prepare_signal_to_learn(self, rec_name):
        _segments = []
        _labels = []
        record, annotation, header = self._load_signals(rec_name)
        signal = record.p_signal[:, 0]  # одне відведення
        sampling_rate = record.fs
ecg_signals, info = nk.ecg_process(signal, sampling_rate=
  sampling_rate)
        r_peaks = info["ECG_R_Peaks"]
        epochs = nk.epochs_create(
            signal,
            events=r_peaks,
            sampling_rate=sampling_rate,
            epochs_start=-0.3,
            epochs_end=0.3,
        )
        annotation_samples = annotation.sample
        annotation_symbols = annotation.symbol
        for _, epoch in epochs.items():
            segment_symbol = self.get_segment_label(
                epoch, annotation_samples, annotation_symbols
            )  # noqa 501
            if not segment_symbol:
                continue
  segment_label = self._beat_label_map.beat_to_label(
  segment_symbol)
            if segment_label < 0:
                continue
            _segments.append(epoch["Signal"])
            _labels.append(segment_label)
        return _segments, _labels

    def prepare_epoch_data(self):
        all_segments = []
        all_labels = []
        # Отримуємо список записів
        records = self._loader.get_local_record_list()
        for record in random.sample(list(records), self._samples):
            segments_, labels_ = self.prepare_signal_to_learn(
  record)
            all_segments.extend(segments_)
            all_labels.extend(labels_)
        _X = np.array(all_segments)
        # Додавання виміру для каналу: (n_samples, segment_length, 1)
        _X = _X[..., np.newaxis]
        _y = np.array(all_labels)
        return _X, _y

    def prepare_stats_data(self, rec_name):
        record, _annotation, _header = self._load_signals(rec_name)
```

```
        signal = record.p_signal[:, 0]  # одне відведення
        ecg = ECG(signal)
        ecg.analyze()
        return ecg
```

Розглянемо основні компоненти.

1) Ініціалізація та кешування даних;
   При створенні об'єкта класу передаються вибраний датасет (типу PhysioNetDataset), мапа міток ударів (BeatAnnotationLabelMap) та кількість записів, які необхідно обробити. У конструкторі ініціалізується завантажувач даних та створюються словники для зберігання завантажених записів, анотацій та заголовків.
2) Завантаження сигналів;
   Метод _load_signals перевіряє наявність запису, анотацій та заголовка в кеші. Якщо якихось компонентів немає, викликається завантаження запису за допомогою PhysioNetLoader, після чого отримані дані зберігаються в словниках для подальшого використання.
3) Виділення сегментів та визначення мітки;
   Метод prepare_signal_to_learn отримує сигнал з одного відведення запису та обчислює інформацію про обробку ЕКГ за допомогою бібліотеки NeuroKit2. Далі використовуючи nk.epochs_create сигнал розбивається на сегменти (епохи) навколо кожного R-піку. Для кожного сегмента викликається статичний метод get_segment_label, який знаходить відповідну анотацію всередині сегмента і повертає відповідний символ. Цей символ перетворюється на числову мітку через мапу міток (beat_label_map), а сегмент та його мітка додаються до списків.
4) Підготовка кінцевих даних для навчання;
   Метод prepare_epoch_data вибирає випадковим чином задану кількість записів із сховища, обробляє кожен з них, збираючи сегменти та відповідні мітки, після чого формує два масиви:
   _X – масив сегментів сигналу;
   _y – масив міток.
5) Обчислення статистичних характеристик;

Метод prepare_stats_data завантажує сигнал запису та передає його об'єкту класу ECG, після чого викликає аналіз сигналу через метод analyze(). Це дозволяє отримати статистичну інформацію, що буде використана для додаткової оцінки сигналу.

Згідно досліджень [1], [4], [6] нейромережа Long Short-Term Memory (LSTM) є хорошим вибором для навчання на даних у вигляді часових рядів. На наступному прикладі коду покажемо, як можна навчити модель розрізняти надшлуночкові екстрасистоли:

```
supraventricular_bynary_map = BeatAnnotationLabelMap(
    {0: BeatAnnotation.normal(), 1: BeatAnnotation.supraventricular
    ()}
)

mit_db_preparation = PrepareECGToLearn(
    dataset=PhysioNetDataset.MIT_BIH_SUPRAVENTRICULAR,
    beat_label_map=supraventricular_bynary_map,
```

```
    samples=2,
)
X, y = mit_db_preparation.prepare_epoch_data()
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
# Побудова моделі LSTM
model = Sequential()
model.add(LSTM(64, input_shape=(X.shape[1], 1)))
model.add(Dropout(0.2))
model.add(Dense(1, activation="sigmoid"))
model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss="binary_crossentropy",
    metrics=["accuracy"],
)
model.summary(print_fn=logging.info)
history = model.fit(
    X_train,
    y_train,
    epochs=10,
    batch_size=32,
    validation_split=0.2,
    callbacks=[LoggingCallback()],
)  # noqa 501
loss, accuracy = model.evaluate(X_test, y_test)
```

Результати навчання та модель зберігаються у файл в директорію ./data/saved_models. Наведемо приклад виводу: INFO - Форма даних X: (4830, 77, 1)

```
INFO - Форма міток y: (4830,)
INFO - Model: "sequential"
```

| Layer (type) | Output Sh | Param |
|---|---|---|
| lstm (LSTM) | (None, 64) | 16896 |
| dropout (Dropout | (None, 64) | 0 |
| dense (Dense) | (None, 1) | 65 |

```
Total params: 16,961 (66.25 KB)
Trainable params: 16,961 (66.25 KB)
Non-trainabl
e params: 0 (0.00 B)
Епоха 10: {'accuracy': 0.932, 'loss': 0.251, 'val\_accuracy': 0.932,
    'val\_loss': 0.246}
```

**3. Висновки.** Розроблена процедура автоматизує отримання, попередню обробку та зберігання ключових параметрів ЕКГ-сигналу. Це дає можливість швидко формувати готові вибірки для навчання моделей машинного навчання. За рахунок автоматизації рутинних процесів, можна здійснювати серії експериментів з різними наборами ознак, проводити їх порівняння та вивчати вплив на результат. Програмна реалізація відповідає основним принципам ООП, що забезпечує модульність та гнучкість у розширенні функціональності. Це дозволяє легко масштабувати рішення для роботи з різними наборами даних та адаптувати його під нові дані.

### Список використаної літератури

1. Zhang Y., Wei S., Wang H. Arrhythmia classification of LSTM autoencoder based on time series clustering. *Computers in Biology and Medicine.* 2021. Vol. 139. P. 104944.
2. Liu F., Liu C. CLINet: A novel deep learning network for ECG signal classification. *Journal of Electrocardiology.* 2024. Vol. 82. pp. 37–44.
3. Lourenço A., Silva H., Leite P., Lourenço R., and Fred A. L. (2012). Real-time electrocardiogram segmentation for finger-based ECG biometrics. *In Biosignals.* pp. 49–54.
4. Wang, J., & Zhang, Y. An Arrhythmia Classification Model Based on a CNN-LSTM-SE Network. *Sensors.* 2024. Vol. 24, No. 19. P. 6306.
5. Chen T., Li X. A hybrid deep learning network for automatic diagnosis of cardiac arrhythmias. *Scientific Reports.* 2023. Vol. 13. P. 17261.
6. Alamatsaz N., et al. A lightweight hybrid CNN-LSTM model for ECG-based arrhythmia detection. *arXiv preprint.* 2022. arXiv:2209.00988.
7. Kumar P., Kumar R. Cardiac arrhythmia detection using deep learning approach and Morse wavelet transform. *Frontiers in Physiology.* 2023. Vol. 14. P. 10588016.
8. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter (3rd ed.). O'Reilly Media, 2022.
9. PhysioNet. Open databases. URL: https://physionet.org/about/database (date of access: 01.04.2025).
10. Goldberger A., Amaral L., Glass L., Hausdorff J., Ivanov P. C., Mark R., ..., and Stanley H. E. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation [Online].* 2000. Vol. 101, No. 23. pp. e215–e220.


**Samus V. M., Samus Ye. I.** Formalization of ECG-Signal Features for Building Machine-Learning Models.


An ECG, or electrocardiogram, is a non-invasive method that records the heart's electrical activity via electrodes placed on the skin. It is used to assess heart rhythm, detect conduction disorders, and diagnose a variety of cardiac pathologies such as ischemia, myocardial infarction, and arrhythmias. Because of its simplicity and availability, the ECG is one of the primary tools in cardiology for monitoring a patient's condition in both inpatient and outpatient settings.

Preparing an ECG signal for machine-learning analysis involves several key stages to ensure high-quality data and optimal suitability for modelling. The process starts with pre-processing, where filtering is applied to remove noise and artefacts, such as baseline wander or high-frequency interference. Normalization then brings signal amplitudes to a common scale, reducing the influence of external factors.

The next stage is segmentation, in which the continuous signal is divided into individual segments around the R peaks, allowing isolation of separate cardiac-cycle complexes. Feature extraction follows: each segment is analyzed to obtain key characteristics such as interval durations, wave amplitudes, and time-frequency parameters. These features enable the model to distinguish between normal and pathological heart conditions. The final step is to convert the processed signal into a format suitable for input to machine-learning algorithms.

This comprehensive approach facilitates effective model training to recognize cardiac patterns, forming the basis for diagnosis and prognosis of various cardiological conditions. The present article focuses on designing algorithms that prepare digital ECG data for machine learning. The proposed algorithm yields statistical characteristics of the digital ECG, which can subsequently be used in further analysis with machine-learning methods.


**Keywords:** digital electrocardiogram, ectopy, statistical features, machine learning, segmentation.


### References

1. Zhang, Y., Wei, S., & Wang, H. (2021). Arrhythmia classification of LSTM autoencoder based on time series clustering. *Computers in Biology and Medicine,* 139, 104944.

2. Liu, F., & Liu, C. (2024). CLINet: A novel deep learning network for ECG signal classification. *Journal of Electrocardiology,* 82, 37–44.
3. Lourenço, A., Silva, H., Leite, P., Lourenço, R., & Fred, A. L. (2012). Real-time electrocardiogram segmentation for finger-based ECG biometrics. *In Biosignals,* 49–54.
4. Wang, J., & Zhang, Y. (2024). An Arrhythmia Classification Model Based on a CNN-LSTM-SE Network. *Sensors,* 24(19), 6306.
5. Chen, T., & Li, X. (2023). A hybrid deep learning network for automatic diagnosis of cardiac arrhythmias. *Scientific Reports,* 13, 17261.
6. Alamatsaz, N., & et al. (2022). A lightweight hybrid CNN-LSTM model for ECG-based arrhythmia detection. *arXiv preprint,* arXiv:2209.00988.
7. Kumar, P., & Kumar, R. (2023). Cardiac arrhythmia detection using deep learning approach and Morse wavelet transform. *Frontiers in Physiology,* 14, 10588016.
8. McKinney, W. (2022). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter (3rd ed.).* O'Reilly Media.
9. PhysioNet. Open databases. Retrieved from https://physionet.org/about/database
10. Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R., ... & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation [Online],* 101(23), e215–e220.

Одержано 12.04.2025