Применение методов машинного обучения для определения состояния сердечно-сосудистой системы на основе анализа показателей квантового фазового пространства ритма сердечно-сосудистой системы

Магистр МиКМ Цветков А.И.

Кандидат ф-м наук Стрельцова О.И.

Доктор ф-м наук Цветков В.П.

Тверь 2023

Актуальность

- По данным Всемирной организации здравоохранения, каждый год сердечно-сосудистые заболевания убивают более 17 миллионов человек во всем мире. Это значительно больше, чем любое другое заболевание на планете, и относительно этого факта исследования в области сердечно-сосудистых заболеваний имеют важное значение.
- Исследования в области сердечно-сосудистых заболеваний могут помочь развивать более эффективные методы профилактики и уменьшения риска заболевания сердечно-сосудистой системы.

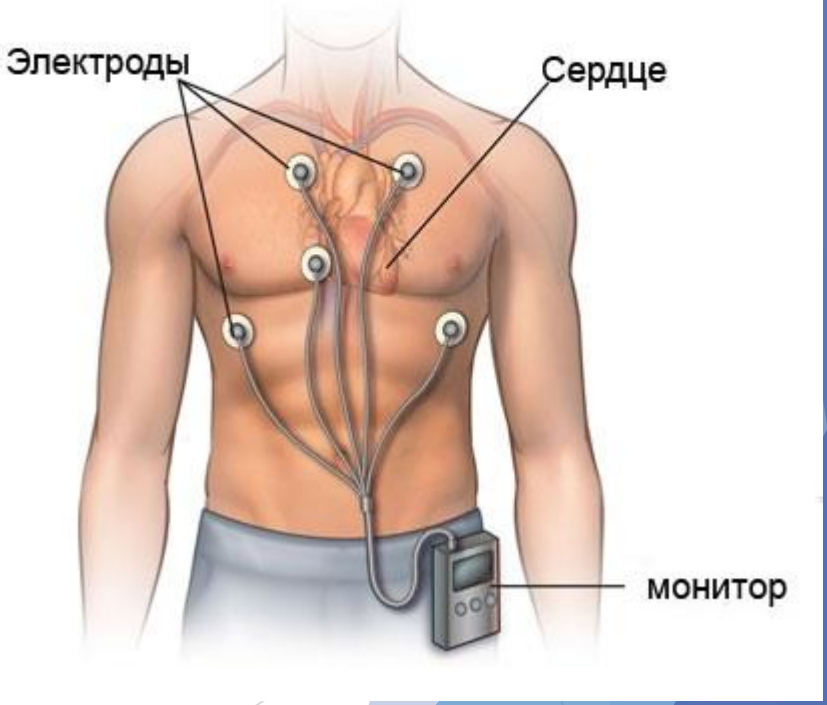
Цели и задачи

- Разработка алгоритмов и создание комплекса программ для решения задачи классификации сердечно-сосудистых заболеваний.
- Основная задача классификации - определение болен человек или здоров.
- Задача разработки и применения математических моделей и программного обеспечения для выявления свойств динамики мгновенного сердечного ритма с использованием больших массивов данных, полученных в результате холтеровского мониторирования.

Холтеровское мониторирование

Суточное (холтеровское) мониторирование - метод исследования, который позволяет производить непрерывную регистрацию динамики сердца на ЭКГ с помощью портативного устройства (холтера), отслеживать изменения в работе сердца и контролировать артериальное давление пациента в течение всего дня в условиях его активности.

Холтеровское мониторирование производится посуточно



Структура исследуемых данных

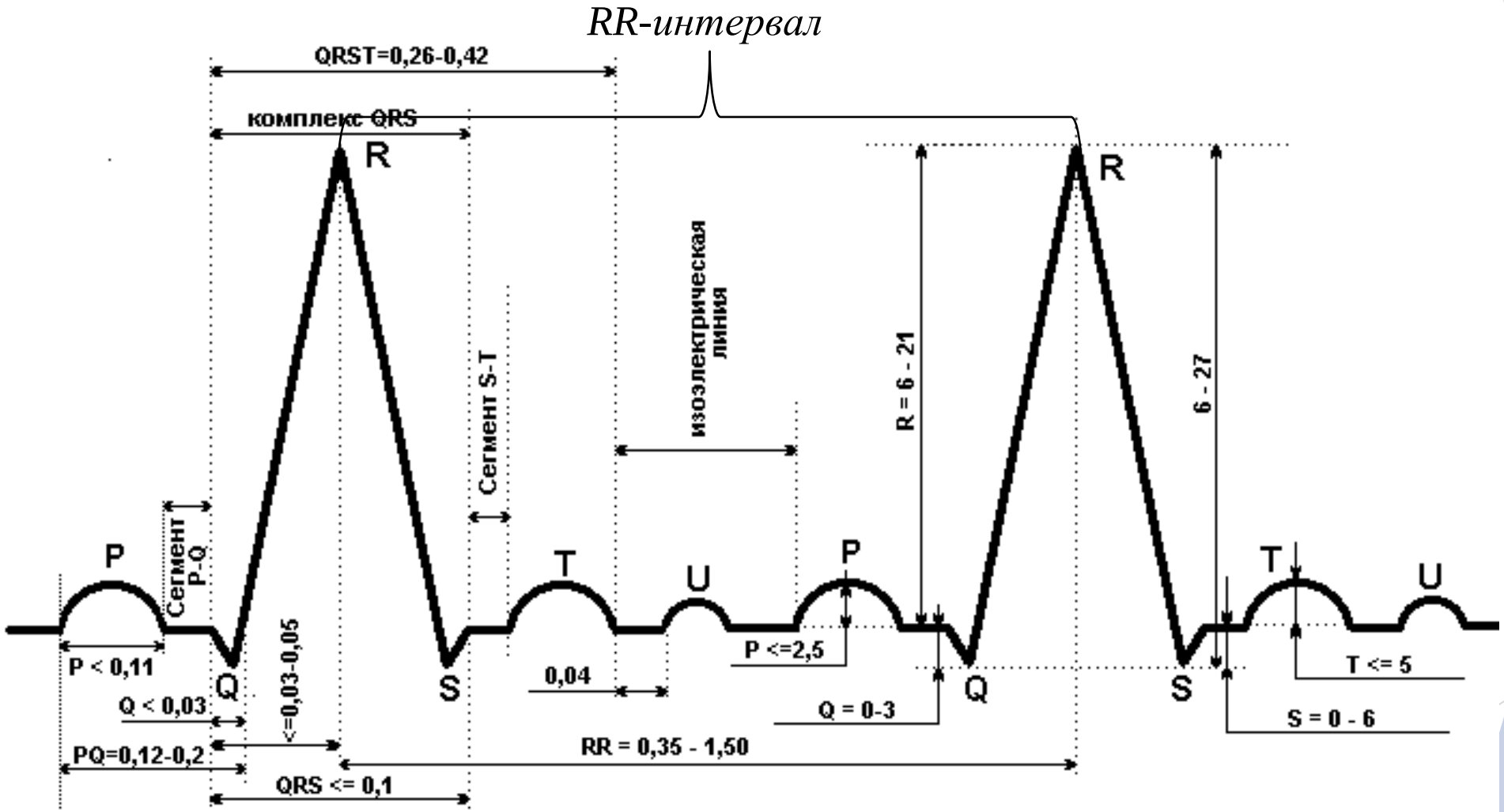


Рис.1. Зубцы и интервалы нормальной ЭКГ

Структура исследуемых данных

Пику R-зубца соответствует момент времени t_i. Тогда $T_{RRi} = t_{i+1} - t_i$. Значения времени t_i измеряются в секундах, а мгновенный ритм y_i в минутах$^{-1}$. На промежутке времени $t_i \le t \le t_{i+1}$ кривая сердечного ритма $y(t)$ дается формулой:

$$y(t) = y_i + (y_{i+1} - y_i)\frac{t - t_i}{t_{i+1} - t_i} \qquad (1)$$

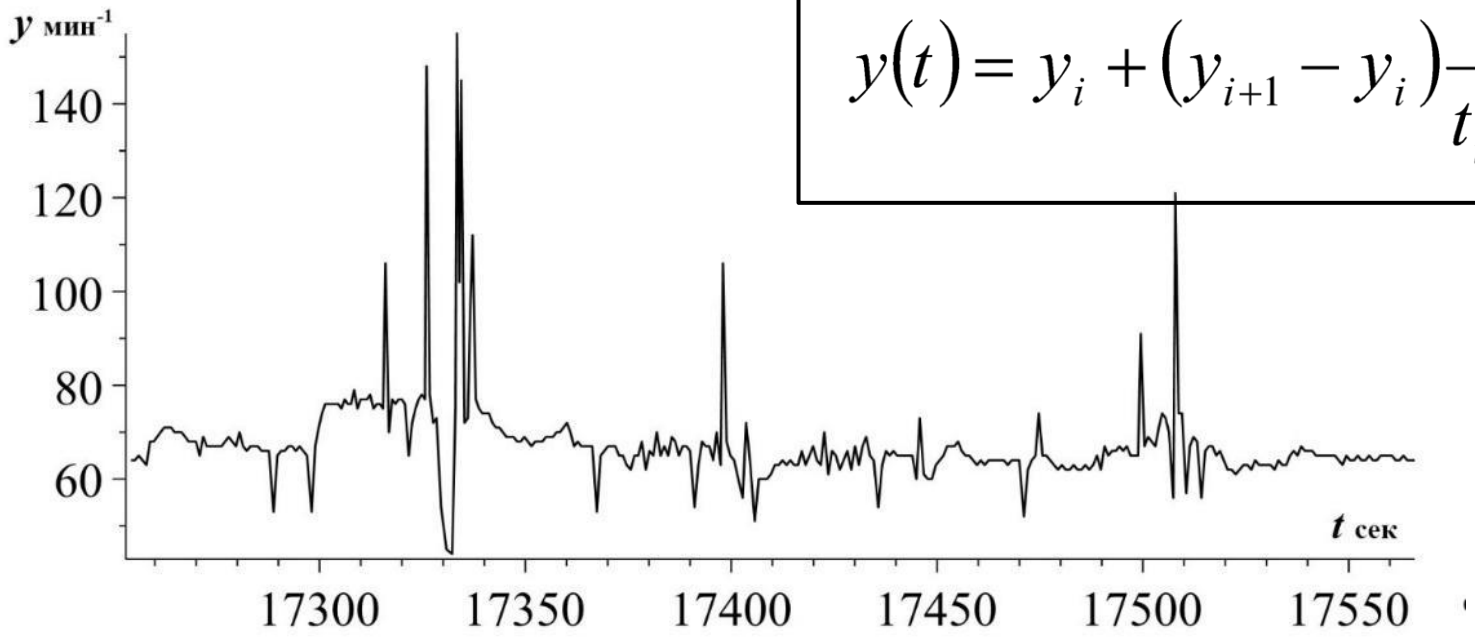


Рис.2. Кривая МСР

Структура исследуемых данных

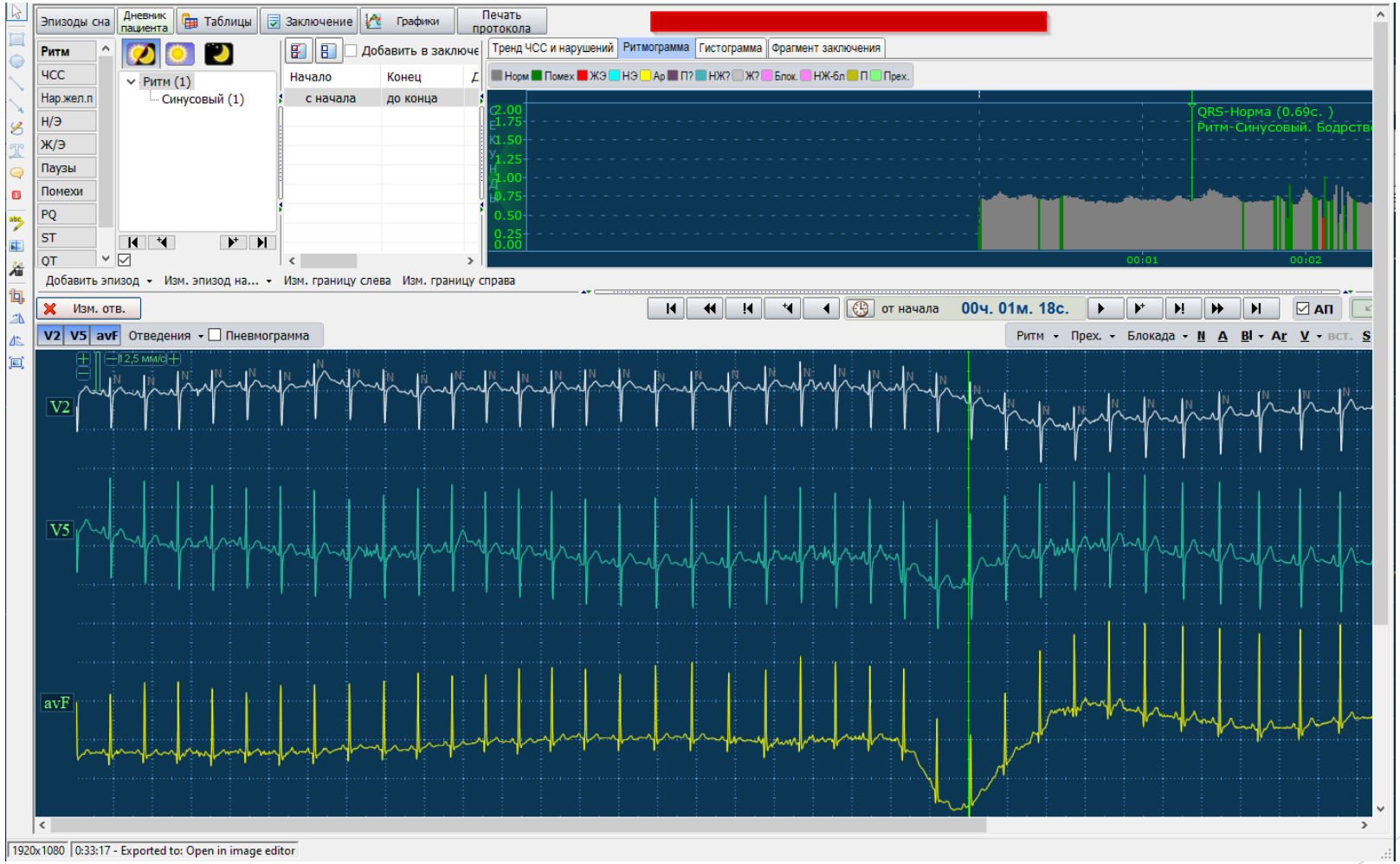


Рис.3. Пример обработки QRS в программе Miokard

Набор данных (dataset)

- Всего 64 случая суточного холтеровского мониторирования.
- 50 случаев для обучения (25 – случаев ХМ норма и 25 случаев ХМ с отклонениями).
- 14 случаев для тестирования (9 – случаев ХМ норма и 5 случаев ХМ с отклонениями).

Данные предоставлены Тверской областной клинической больницей

Подход квантового фазового пространства

- Квантовое фазовое пространство ритма сердечно-сосудистой системы является отображением колебательной динамики системы, выраженной в виде временного ряда интервалов между последовательными сердечными сокращениями (RR-интервалами). Это пространство представляет собой многомерный фазовый портрет, отражающий динамические свойства системы.

1. А. П. Иванов, А. Н. Кудинов, Д. Ю. Лебедев, В. П. Цветков, И. В. Цветков. Анализ мгновенного сердечного ритма в модели мультифрактальной динамики на основе холтеровского мониторирования. // Математическое моделирование, 2015, т.27, №4, с.16–30.
2. А.Н. Кудинов, Д.Ю. Лебедев, В.П. Цветков, И.В. Цветков. Математическая модель мультифрактальной динамики и анализ сердечных ритмов // Математическое моделирование, 2014, т.26, №10, с.127-136.

Подход квантового фазового пространства

Основной особенностью **квантового фазового пространства** является его квантование - процесс деления фазового пространства на элементарные ячейки конечной величиной h. Параметр h называют постоянной или шагом квантования. Квантование ФП МСР производится согласно алгоритму:

$$y_i = h[y(t)h^{-1}], v_i = h[v(t)h^{-1}], i = 1, 2,... N(h, t), \quad (2)$$

где [ ] - оператор округления до ближайшего целого числа. Значения y_i , v_i - кратны постоянной квантования h. Кратности значений y_i , v_i определяют значения чисел заполнения n_i элементарных ячеек КФП.

Таким образом множество значений $\{n_i, y_i, v_i\}$ образует **квантовое фазовое пространство.**

Подход квантового фазового пространства

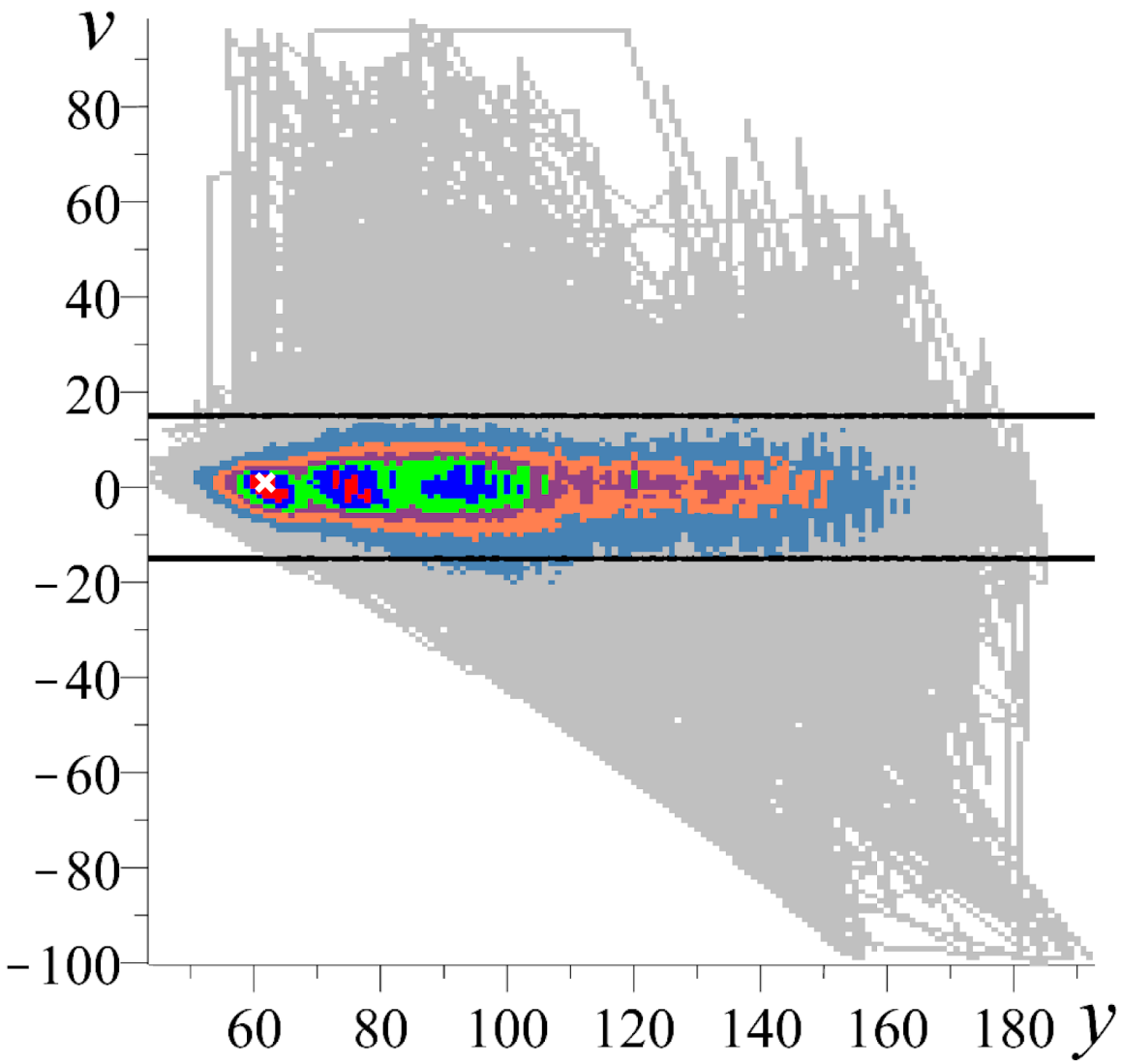


Рис.4. Проекция гистограммы (Норма)

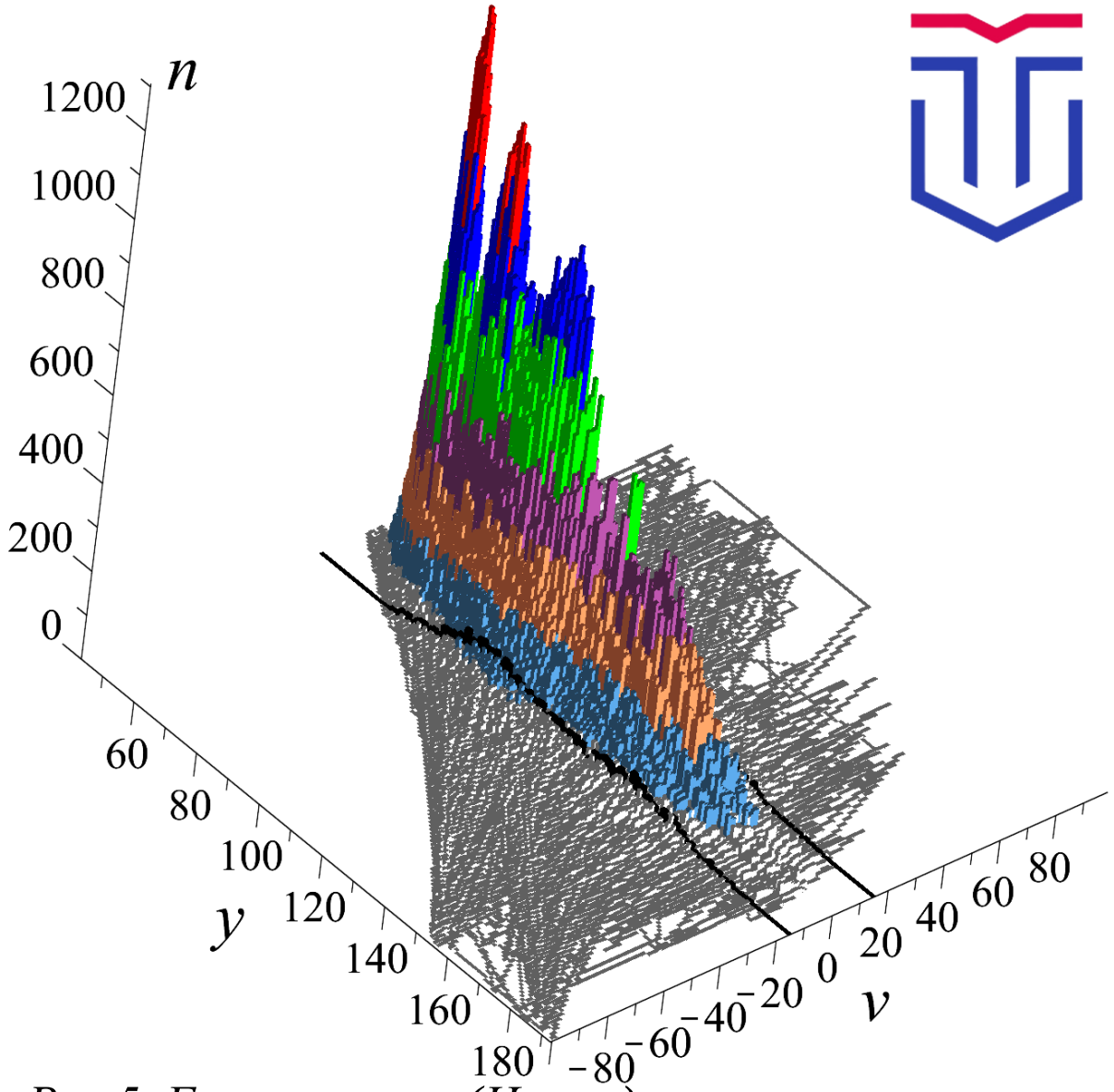


Рис.5. Гистограмма (Норма)

Подход квантового фазового пространства

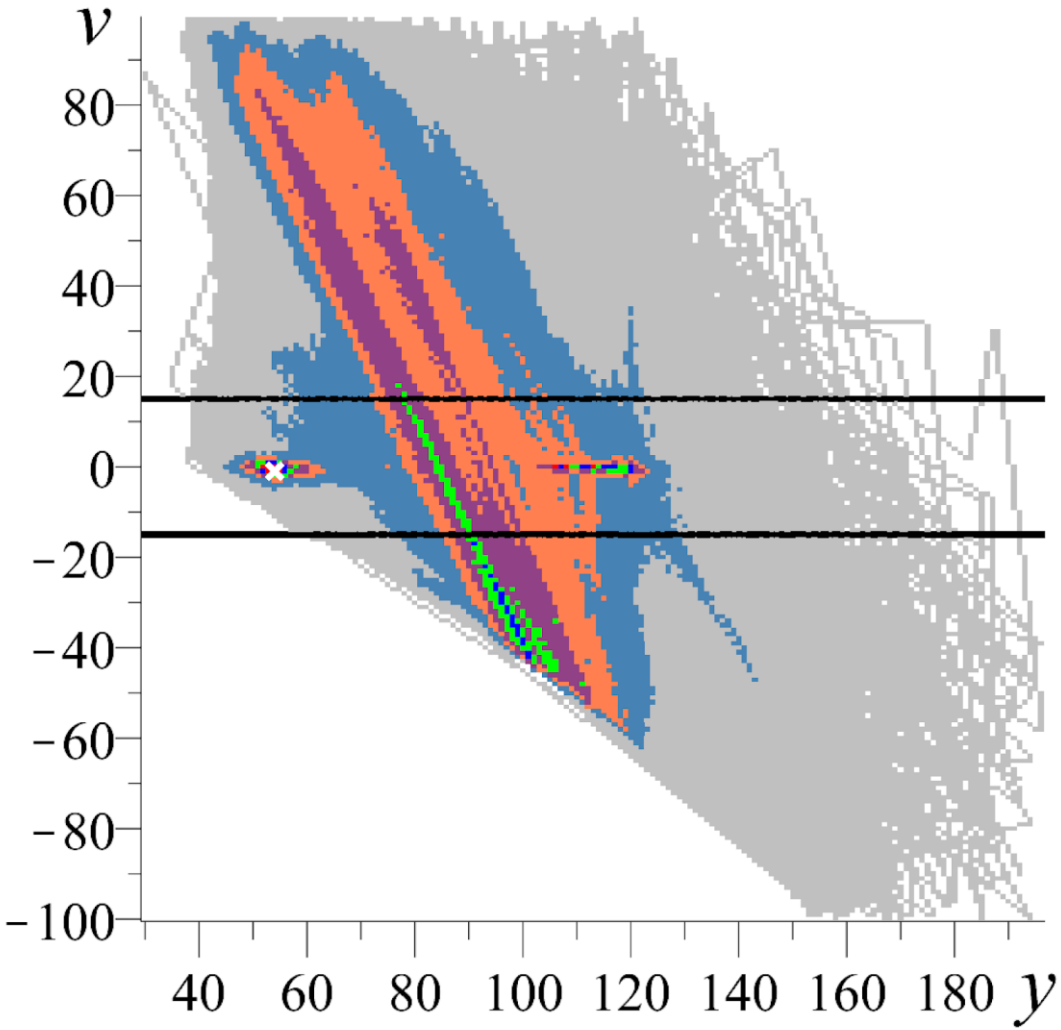


Рис.6. Проекция гистограммы (Желудочковая экстрасистолия)

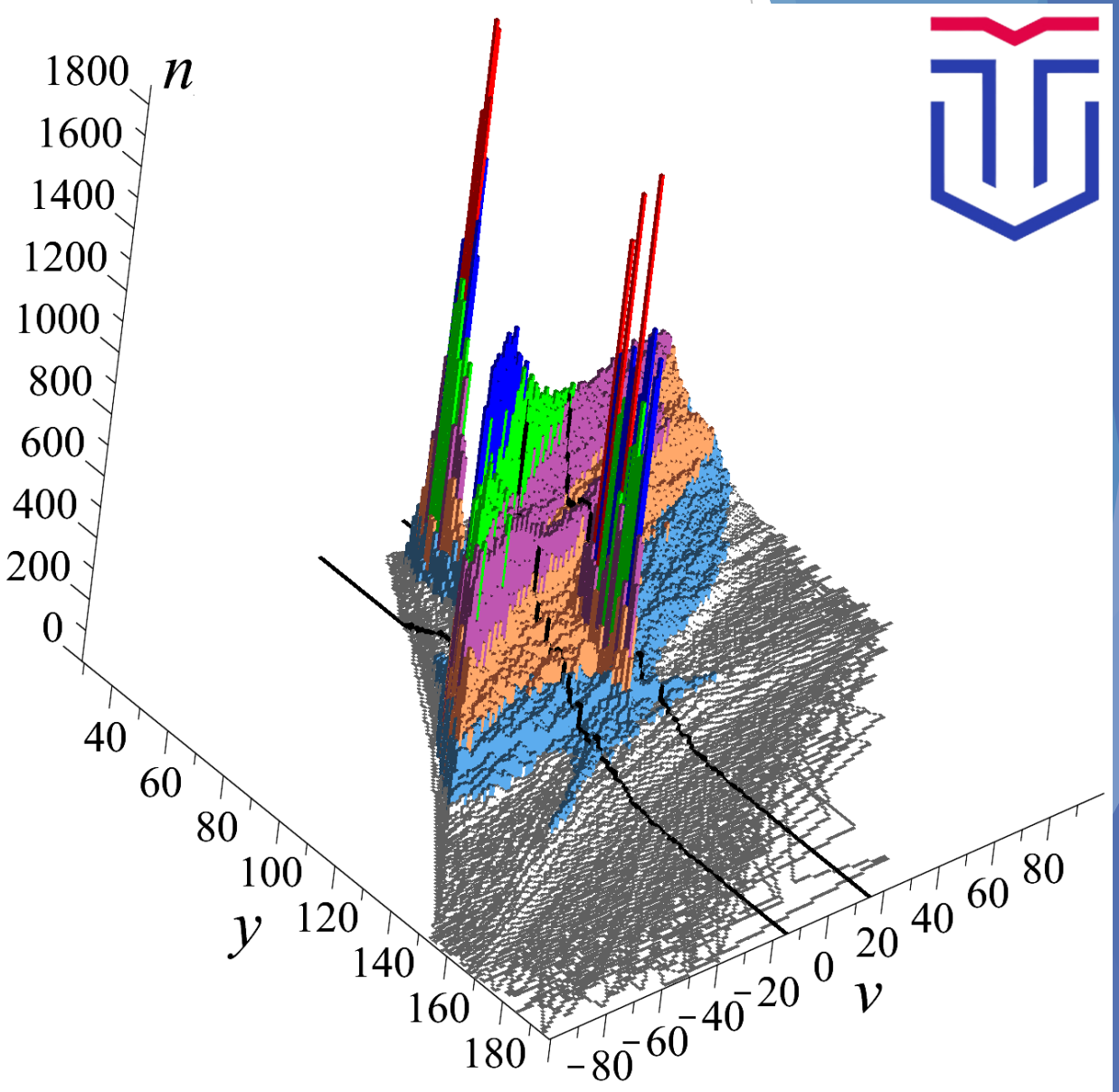


Рис.7. Гистограмма (Желудочковая экстрасистолия)

Подход квантового фазового пространства

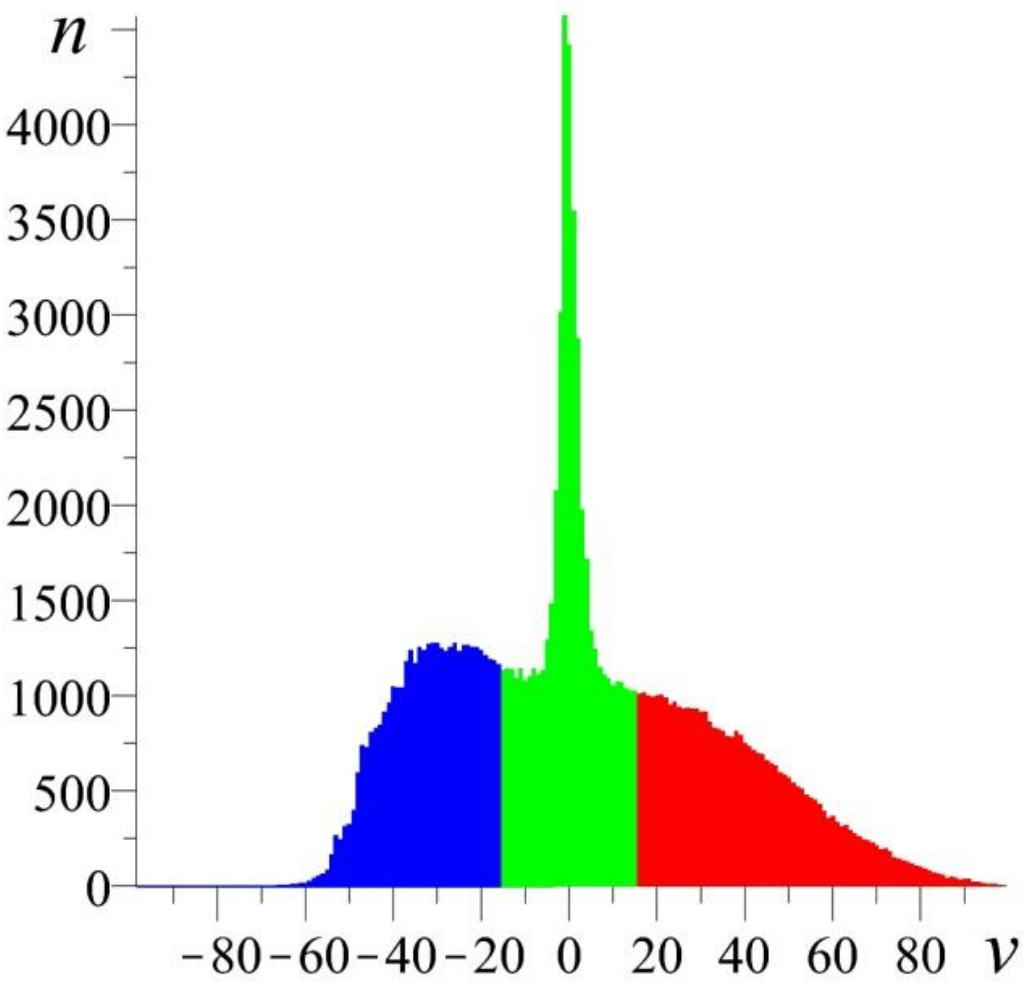


Рис.8. Проекция гистограммы для срезов (Желудочковая экстрасистолия)

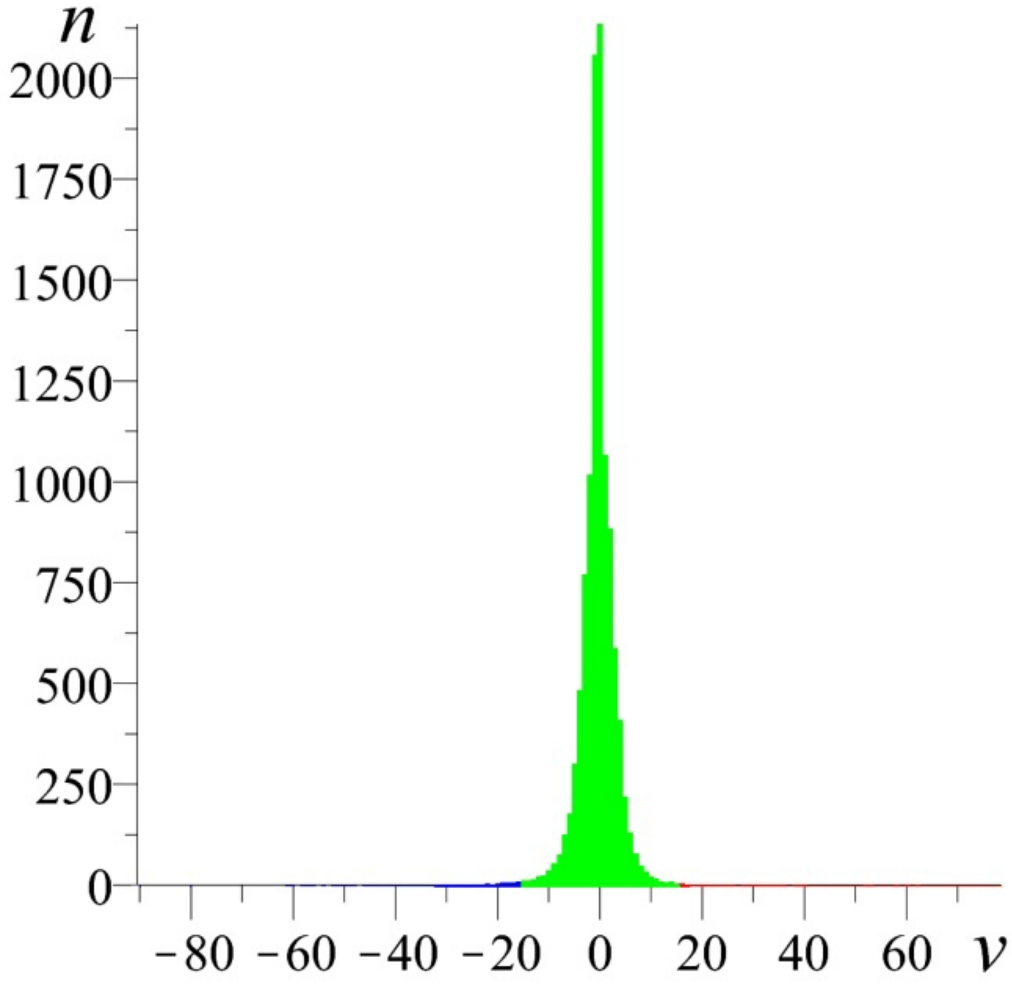


Рис.9. Гистограмма для срезов (Норма)

Инструментарий

- NumPy - это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами.
- Pandas - библиотека, которая применяется для обработки и анализа табличных данных.
- Scikit-learn (sklearn) - один из наиболее широко используемых пакетов Python для Data Science и Machine Learning.
- Matplotlib - популярная Python-библиотека для визуализации данных.

Формирование интервальных признаков

- Исходные данные представляют собой набор пар точек вида [v,n], где v это значение MCP, а n - количество подобных точек на срезе.
- Далее мы производим сортировку этих данных и разбиваем точки на группы значений от -80 до +80 с шагом 0.5 и получаем 320 групп векторов, расположенных в порядке возрастания их значений.

Формирование интервальных признаков

```python
def importData(input_file_path):
    file_list = glob.glob(input_file_path + "/*.xlsx")
    excl_list = []
    data = []
    for file in file_list:
        excl_list.append(pd.read_excel(file))

    for i in range(0, len(file_list)-1):
        wb = openpyxl.load_workbook(file_list[i])
        sh = wb.active
        for j in range(1,300):
            if sh.cell(row=j,column=1).value is not None:
                rr = sh.cell(row=j,column=1).value
                n = sh.cell(row=j,column=2).value
                for k in range(int(n)):
                    data.append(rr)
            else:
                j = 301
    return np.array(data)
```

Рис.12. Выгрузка данных из excel файлов

```python
def sortDataByVectors(data):
    vectorData = []
    data = sorted(data)
    for i in range(-100,100):
        j = 0
        vector = [0.0] * 1000
        for elem in data:
            if (elem > i and elem < (i+0.5)):
                vector[j] = elem
                j += 1
        vector[:]=(value for value in vector if value != 0.0)
        if(len(vector) != 0):
            vectorData.append(vector)
    return np.array(vectorData)
```

Рис.13. Сортировка исходных данных по векторам в порядке возрастания их значений

Формирование интервальных признаков

```python
def setArraySignsAmount(data):
    dataSet = np.zeros((len(data),160))
    for i in range(0, len(data)):
        goodData = np.zeros(160)
        j = 0
        for elem in data[i]:
            if j != 160:
                goodData[j] = len(elem)
                j += 1
            else:
                break
        dataSet[i] = goodData
    return dataSet
```

Рис.14. Представление полученных данных в виде массива размерностей

```
In [74]: x_train[0]

Out[74]: array([  2.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,
                  4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,
                  4.,   4.,   4.,   4.,   6.,   6.,   4.,   4.,   5.,   6.,   6.,
                  4.,   6.,   6.,   8.,   8.,  10.,  12.,  12.,  12.,  12.,  12.,
                 12.,  14.,  16.,  16.,  16.,  16.,  16.,  18.,  18.,  20.,  20.,
                 20.,  20.,  22.,  22.,  22.,  23.,  26.,  26.,  30.,  29.,  31.,
                 34.,  45., 103., 170., 443., 699., 498., 235.,  74.,  46.,  44.,
                 42.,  32.,  33.,  28.,  26.,  24.,  24.,  22.,  22.,  22.,  20.,
                 16.,  16.,  16.,  16.,  16.,  16.,  12.,  10.,  10.,  10.,  10.,
                 10.,  10.,  10.,  10.,  10.,   8.,   8.,   8.,   8.,   8.,   6.,
                  4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,
                  4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,   4.,
                  4.,   4.,   4.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
                  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
                  0.,   0.,   0.,   0.,   0.,   0.])
```

Рис.15. Пример итогового массива данных одного из случаев

Модель SVM (Support Vector Machine)

Метод опорных векторов (SVM, support vector machine) — набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа. Принадлежит семейству линейных классификаторов. Особым свойством метода опорных векторов является непрерывное уменьшение эмпирической ошибки классификации и увеличение зазора, поэтому метод также известен как метод классификатора с максимальным зазором.

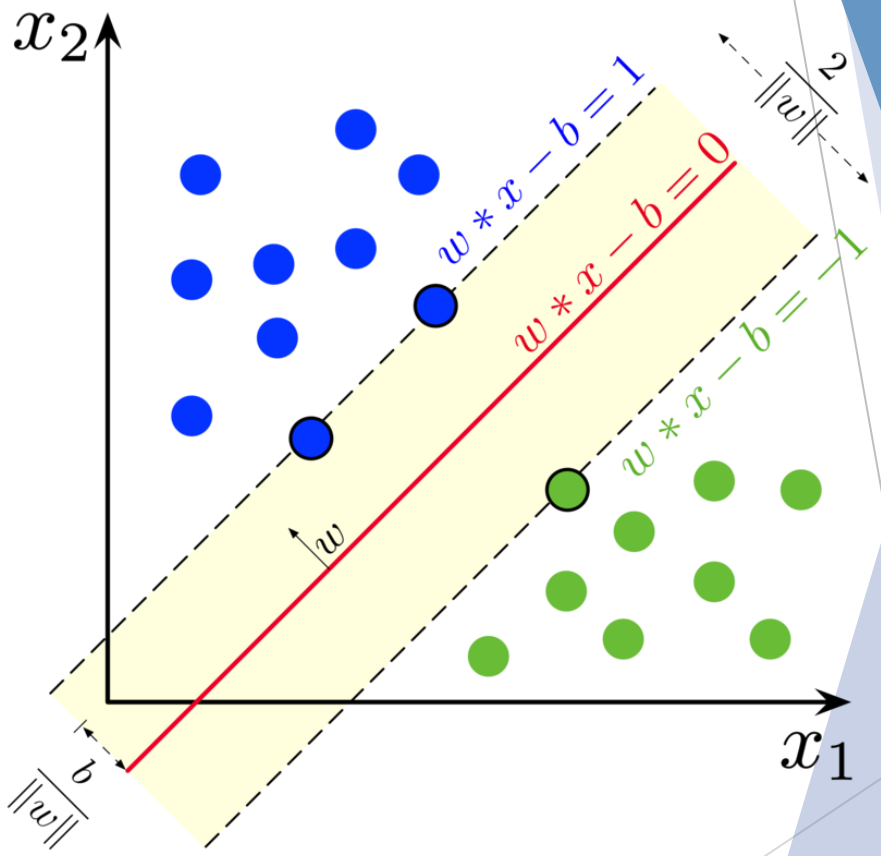


Рис.16. Гиперплоскость с максимальным запасом и поля для SVM, обученного с образцами из двух классов. Выборки на границе называются опорными векторами.

Функции ядра

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$, где d определяется степенью параметра r по coef0.
- rbf: $exp(-\gamma||x - x'||^2)$, где γ определяется степенью параметра gamma, должен быть больше 0.
- sigmoid $tanh(\gamma \langle x, x' \rangle + r)$, где r определяется coef0.

Метрики для оценки результатов обучения

- accuracy рассчитывается как отношение числа правильно классифицированных образцов к числу всех образцов.
- precision рассчитывается также, но относительно лишь образцов отобранных классификатором.
- recall представляет собой отношение tp / (tp + fn), где tp — количество истинных положительных результатов, а fn — количество ложноотрицательных результатов.
- Под recall понимается интуитивная способность классификатора находить все положительные образцы.

Прогноз	Реальность: +	Реальность: –
+	***True Positive (истинно-положительное решение)***: прогноз совпал с реальностью, результат положительный произошел, как и было предсказано ML-моделью	***False Positive (ложноположительное решение)***: ошибка 1-го рода, ML-модель предсказала положительный результат, а на самом деле он отрицательный
–	***False Negative (ложноотрицательное решение)***: ошибка 2-го рода – ML-модель предсказала отрицательный результат, но на самом деле он положительный	***True Negative (истинно-отрицательное решение)***: результат отрицательный, ML-прогноз совпал с реальностью

Рис.17. Пример матрицы ошибок.

Результаты обучения и предсказание

```
In [71]: print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

         Accuracy: 0.9285714285714286

In [72]: # Model Precision: what percentage of positive tuples are labeled as such?
         print("Precision:",metrics.precision_score(y_test, y_pred))

         # Model Recall: what percentage of positive tuples are labelled as such?
         print("Recall:",metrics.recall_score(y_test, y_pred))

         Precision: 0.8333333333333334
         Recall: 1.0
```

Рис.18. Оценка результатов предсказания SVM модели

Результаты обучения и предсказание

```
# Создаем экземпляр SVM и обучаем модель с использованием линейного ядра
C = 1.0  # параметр регуляризации SVM
linear_svc = svm.SVC(kernel='linear', C=C).fit(x_train, y)
# Создаем экземпляр SVM и обучаем модель с использованием RBF-ядро
rbf_svc = svm.SVC(kernel='rbf', C=C).fit(x_train, y)
# Создаем экземпляр SVM и обучаем модель с использованием полиномиального ядра
poly_svc = svm.SVC(kernel='poly', C=C).fit(x_train, y)
# Создаем экземпляр SVM и обучаем модель с использованием сигмоидного ядра
sig_svc = svm.SVC(kernel='sigmoid', C=C).fit(x_train, y)
# оцениваем качество моделей
print('Accuracy of linear kernel:', accuracy_score(y_test, linear_svc.predict(x_test)))
print('Accuracy of polynomial kernel:', accuracy_score(y_test, poly_svc.predict(x_test)))
print('Accuracy of RBF kernel:', accuracy_score(y_test, rbf_svc.predict(x_test)))
print('Accuracy of sigmoid kernel:', accuracy_score(y_test, sig_svc.predict(x_test)))
```

```
Accuracy of linear kernel: 0.8571428571428571
Accuracy of polynomial kernel: 0.8571428571428571
Accuracy of RBF kernel: 0.9285714285714286
Accuracy of sigmoid kernel: 0.6428571428571429
```

Рис.19. Оценка точности предсказания моделей с разными функциями ядра.

Матрица ошибок

Таблица с 4 различными комбинациями прогнозируемых и фактических значений. Прогнозируемые значения описываются как положительные и отрицательные, а фактические – как истинные и ложные.

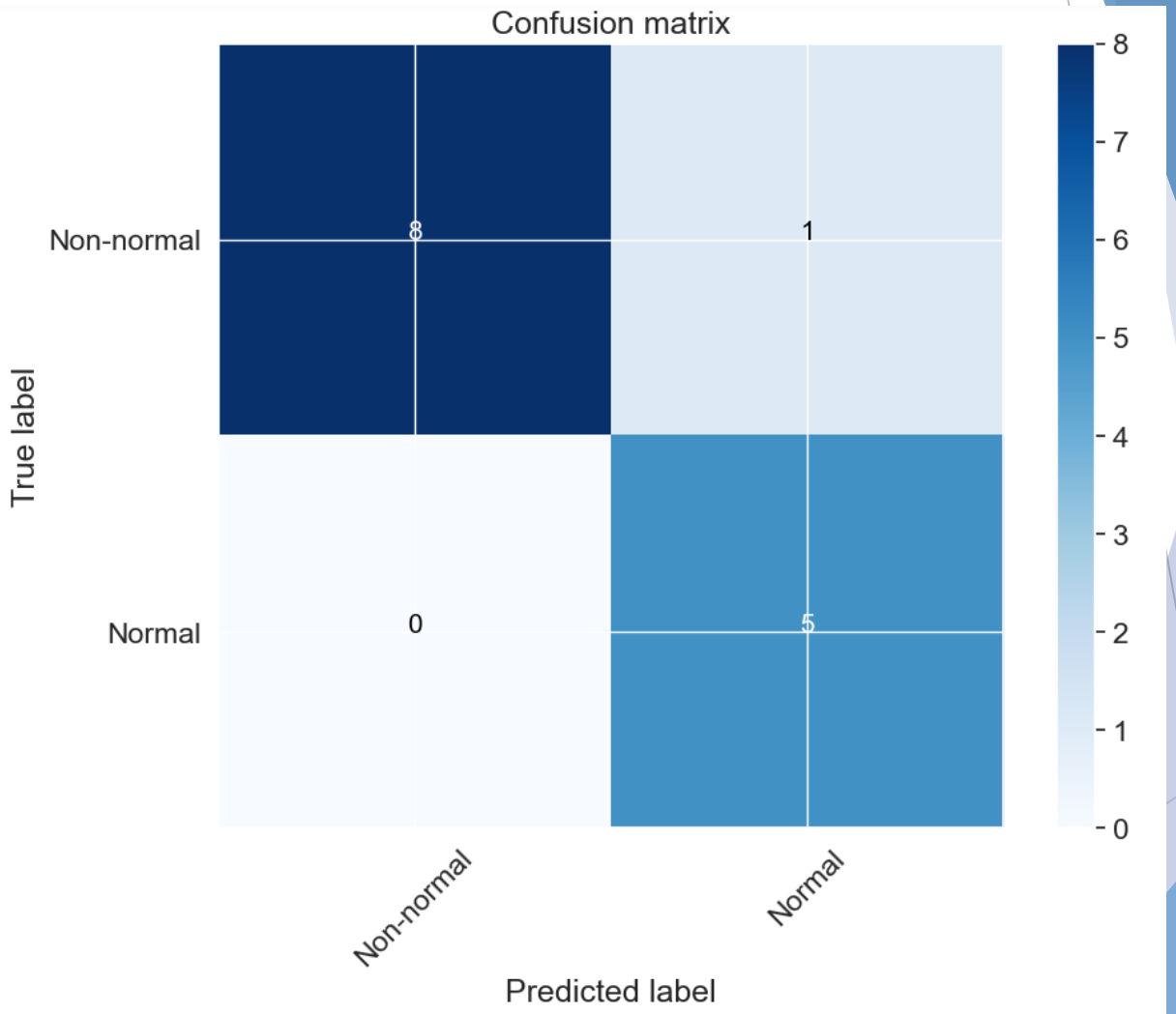


Рис.20. Матрица ошибок.

Перекрестная проверка

При перекрестной проверке данные разделяются несколько раз
и обучаются несколько моделей.

```
In [35]: scores = cross_val_score(clf, x_train, y)
         print("Cross-validation scores: {}".format(scores))

         Cross-validation scores: [0.8 0.7 1.  0.8 0.9]
```

Рис.21. Результаты перекрестной проверки.

Заключение

Для решения задачи классификации заболеваний сердечно-сосудистой системы на основе результатов холтеровского мониторирования разработан следующий алгоритм:

1. Предобработка данных на основе подхода квантового фазового пространства;
2. Бинаризация числовых признаков;
3. Применение методов машинного обучения для решения задачи классификации;

Реализован метод машинного обучения SVM для анализа срезов 3D гистограмм с целью классификации исследуемых данных на категории (норма, отклонения от нормы) с точностью 93%.

Планируется рассмотреть решение задачи на большем наборе данных и довести исследование до программного комплекса, который может служить помощником для принятия диагностических решений врачам-кардиологам.

Спасибо за внимание