МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Системное программирование и компьютерные технологии
(направленность (профиль)/специализация

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Применение лепестковых диаграмм для снижения размерности признакового пространства»

Студент С.М. Салимов
(И.О. Фамилия) (личная подпись)

Руководитель к.ф.-м.н., О.В. Лелонд
(ученая степень, звание, И.О. Фамилия)

Консультант К.А. Селиверстова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

# АННОТАЦИЯ

Тема бакалаврской работы: «Применение лепестковых диаграмм для снижения размерности признакового пространства».

В данной бакалаврской работе исследуется способы снижения размерности признакового пространства временных рядов.

В работе предложен алгоритм снижения размерности признакового пространства временных рядов основанный на построение по набору данных лепестковой диаграммы и определение ее 5 характеристик: периметр диаграммы, площадь диаграммы, максимальное значение на диаграмме, координаты X и Y центра диаграммы. Таким образом, количество признаков временного ряда снижается до 5. Разработано программное обеспечение, реализующее предложенный алгоритм. Проведен вычислительный эксперимент, доказывающий состоятельность предложенного подхода.

Структура бакалаврской работы представлена введением, тремя главами, заключением, списком литературы.

Во введении описывается актуальность проводимого исследования, дается краткая характеристика проделанной работы.

В первой главе проводится анализ развития алгоритмов снижения размерности признакового пространства.

Во второй главе описывается математический аппарат предложенного алгоритма снижения размерности признакового пространства, основанного на анализе параметров лепестковой диаграммы.

В третьей главе приведены примеры использования предложенных подходов, в также описано разработанное программное обеспечение.

В заключении представлены выводы по проделанной работе.

В работе использовано 2 таблицы, 26 рисунков, список литературы содержит 20 литературных источников. Общий объем бакалаврской работы составляет 46 страниц.

## ABSTRACT

The theme of the bachelor's work: "Application of paddle diagrams to reduce the dimension of the characteristic space".

In the given bachelor's work ways of decrease in dimensional space of time series are investigated.

In the work the algorithm of decreasing the characteristic space of time series is offered. It is based on the construction on the data set of the radar chart and definition of its 5 characteristics: perimeter of the chart, the chart area, the maximum value on the chart, X and Y coordinates of the chart center. Thus, the number of time series features is reduced to 5. The software is developed to implement the proposed algorithm. A computational experiment proving the validity of the proposed approach has been conducted.

The structure of the bachelor's work is represented by an introduction, three chapters, conclusion and a list of literature.

The introduction describes the relevance of the research being conducted and gives a brief description of the work done.

The first chapter analyses the development of algorithms for reducing sign space.

The second chapter describes the mathematical apparatus of the proposed algorithm for reducing sign space, based on an analysis of the petal diagram parameters.

The third chapter gives examples of using the proposed approaches, and the developed software is also described.

In conclusion, the conclusions on the work done are presented.

In work 2 tables, 26 figures are used, the list of literature contains 20 literature sources. The total volume of the bachelor's work is 46 pages.

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

При статистическом анализе данных каждый объект описывается вектором его характеристик. Во многих случаях перед аналитиками возникает задача снижения размера этого вектора. Например, когда требуется улучшить результаты распознавания образов, большое количество простых признаков заменяют на их комбинированные аналоги меньшего количества. Кроме того, снижение размера вектора признаков приводит к снижению вычислительных затрат при анализе данных.

В машинном обучении снижение размерности признакового пространства приводит к значительному уменьшению размера обучающей выборки, используемой для установления определения зависимостей между входными и выходными параметрами.

Таким образом, актуальным вопросом остается разработка новых алгоритмов снижения размерности признакового пространства. В рамках данной работы рассматривается вопрос снижения размерности признакового пространства только для временных рядов.

Целью работы является разработка алгоритма снижения размерности признакового пространства временных рядов на основе параметров лепестковой диаграммы.

В работе предложен алгоритм снижения размерности признакового пространства временных рядов основанный на построении по набору данных лепестковой диаграммы и определении ее 5 характеристик: периметр диаграммы, площадь диаграммы, максимальное значение на диаграмме, координаты X и Y центра диаграммы. Таким образом, количество признаков временного ряда снижается до 5. Разработано программное обеспечение, реализующее предложенный алгоритм. Проведен вычислительный эксперимент на данных сервиса «UEA & UCR Time Series Classification Repository», доказывающий состоятельность предложенного подхода.

# ГЛАВА 1 АНАЛИЗ ПЕРСПЕКТИВ РАЗВИТИЯ АЛГОРИТМОВ СНИЖЕНИЯ РАЗМЕРНОСТИ ПРИЗНАКОВОГО ПРОСТРАНСТВА

Математически задача снижения размерности признакового пространства можно описать следующим образом. Предположим, что у нас имеется выборка данных с описанием объектов $X = \{x_n\}_{n=1}^{N}, x_n \in R^D$. Необходимо сгенерировать представление этих данных в пространстве меньшей размерности $T = \{t_n\}_{n=1}^{N}, t_n \in R^d$, где $d < D$ [1-4].

Как показывает анализ литературных источников, снижение размерности признакового пространства может преследовать разные цели, среди которых:

1. Получение новых признаков. Так получение новых признаков в результате преобразования $X \to T$ может оказывать решающее значение при решении задач распознавания [5-8].

2. Представление/визуализации данных большой размерности. Проецирование данных в двухмерное или трехмерное пространство позволяет графически представлять выборку [9-12].

3. Сжатие объема данных для повышения эффективности при организации их хранения. Здесь стоит отметить, что в этом случае необходимо применять такие алгоритмы снижения размерности признакового пространства, которые смогут обеспечить выполнение и обратной операции преобразования ($T \to X$) [13-18].

4. Сокращение вычислительных затрат, сопровождающих анализ данных [18-21].

5. Снижение требований к размеру обучающей выборки при использовании алгоритмов искусственного интеллекта. Чем больше независимых параметров у объектов обучающей выборке, тем больший ее объем требуется для установления зависимостей между параметрами.

Особенно сильно это эффект проявляется в задачах анализа изображений, где обучающие выборки по объему достигают сотни гигабайт [21-25].

Таким образом, актуальным вопросом остается разработка новых алгоритмов снижения размерности признакового пространства.

В рамках данной работы рассматривается вопрос снижения размерности признакового пространства только для временных рядов.

Таким образом, целью исследования является – разработка алгоритма снижения размерности признакового пространства временных рядов на основе параметров лепестковой диаграммы.

Для достижения поставленной цели предполагается решить следующие задачи:

1. Провести анализ перспектив развития алгоритмов снижения размерности признакового пространства

2. Разработать алгоритм снижения размерности признакового пространства временных рядов на основе параметров лепестковой диаграммы.

3. Проверить работу алгоритма на данных, представленных на сервисе «UEA & UCR Time Series Classification Repository»

4. Разработать программное обеспечение, реализующее предложенный алгоритм снижения размерности признакового пространства.

**Выводы по главе**

Дано описание задачи снижения размерности признакового пространства. На основе литературных источников определены сферы исследований, где решение данной задачи носит практическую пользу. Сформулирована цель бакалаврской работы, определены задачи которые необходимо решить для ее достижения.

# ГЛАВА 2 РАЗРАБОТКА АЛГОРИТМА СНИЖЕНИЯ РАЗМЕРНОСТИ ПРИЗНАКОВОГО ПРОСТРАНСТВА ДЛЯ ВРЕМЕННЫХ РЯДОВ

## 2.1 Математический аппарат алгоритма

Сначала следует отметить, что в рамках данной работы рассматривается вопрос снижения размерности признакового пространства только для временных рядов. Изначально лепестковая диаграмма не предназначена для отображения на ней временных рядов. Поэтому чтобы отобразить временной ряд на такой диаграмме надо выбрать направление отсчета значений $Z_1$, $Z_2 \ldots Z_n$ и положение оси с которой будет откладываться первое значение (index 1) временного ряда. Здесь на рисунке 2.1 и далее первое значение откладывается на горизонтальной оси Index 1, направление отсчета следующих значений – против часовой стрелки.

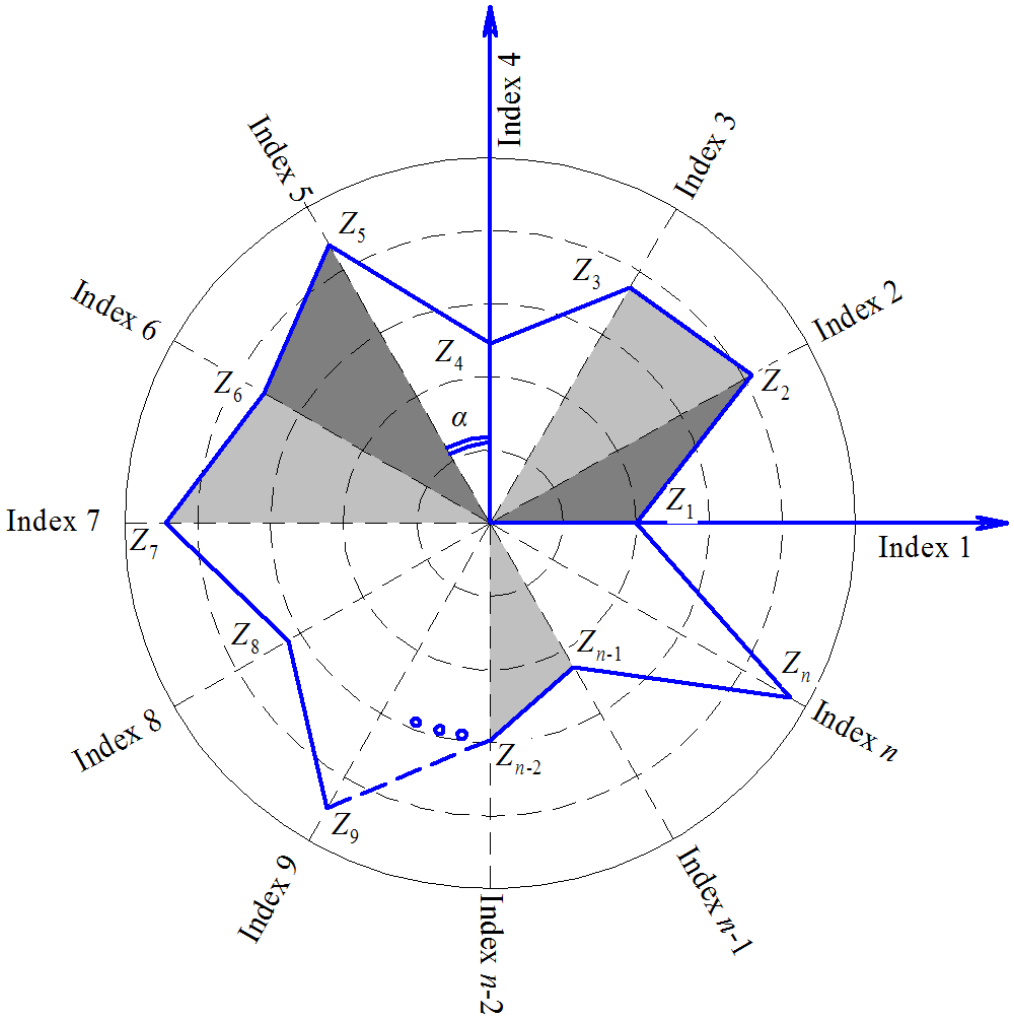



Рисунок 2.1 – Построение лепестковой диаграммы на основе данных временного ряда

Если откладывать значения временного ряда на лепестковой диаграмме, то можно получать различные контуры.

Далее представлены пример лепестковой диаграммы для временного ряда с ярко выраженным максимумом (рисунок 2.2) и пример лепестковой диаграммы для временного ряда с плавным уменьшением значений (рисунок 2.3).

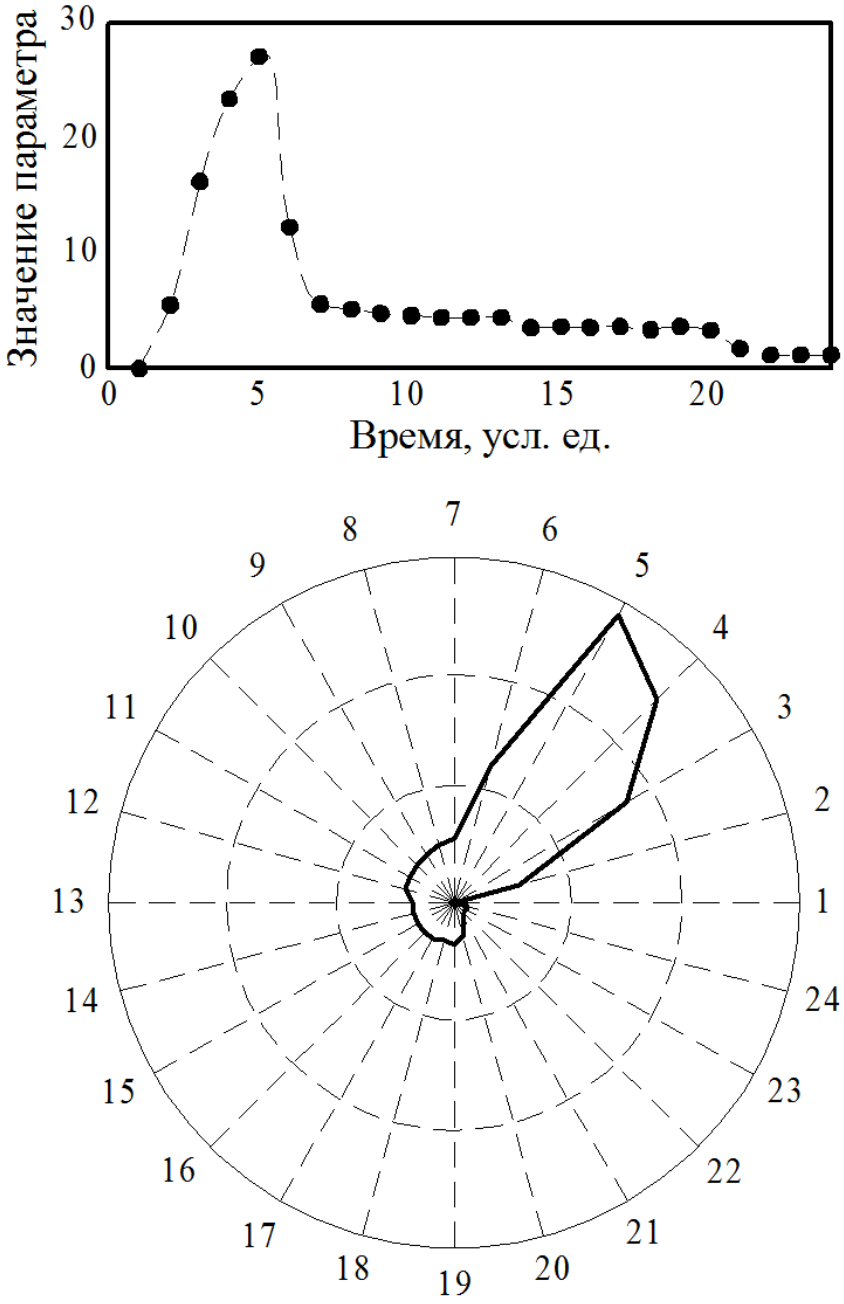



Рисунок 2.2 – Пример лепестковой диаграммы для временного ряда с ярко выраженным максимумом

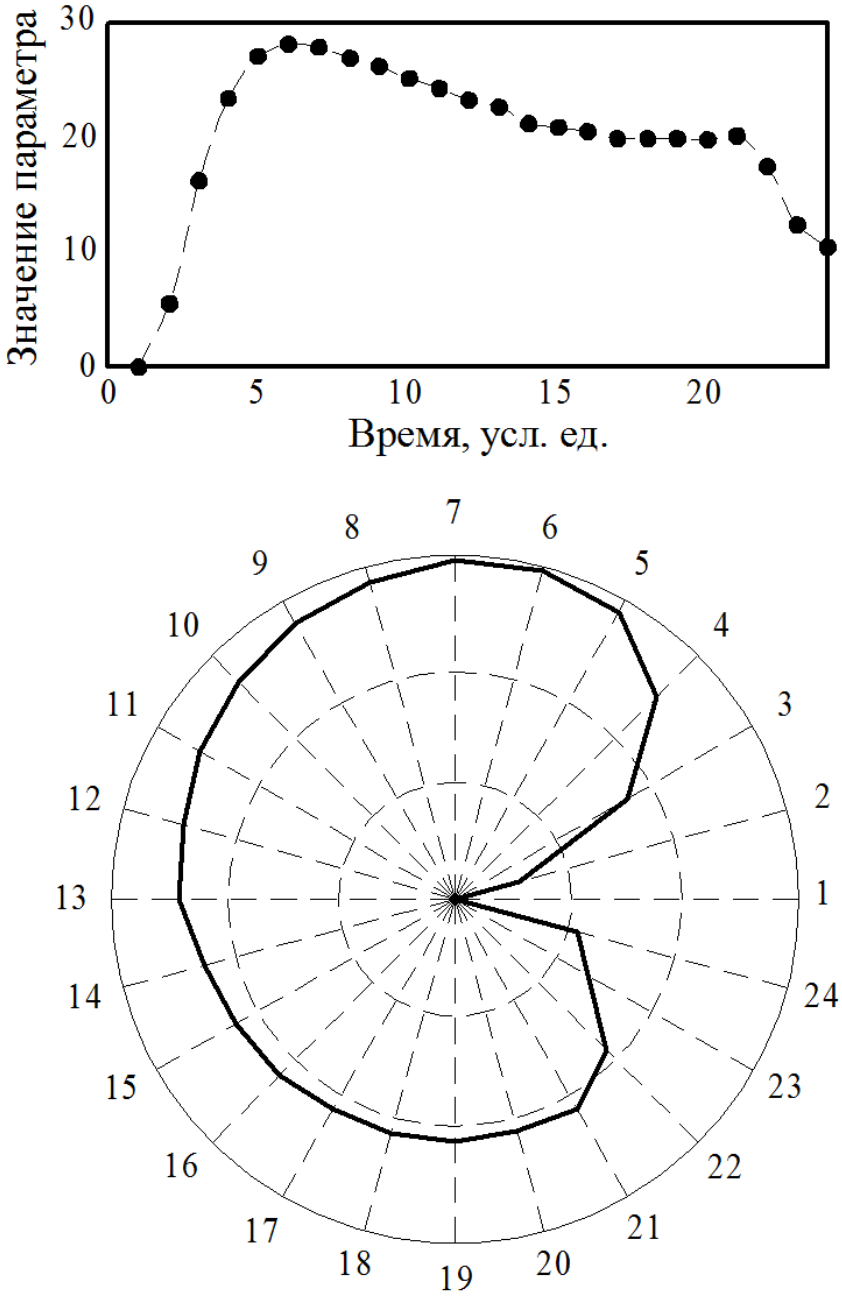



Рисунок 2.3 – Пример лепестковой диаграммы для временного ряда с плавным уменьшением значений

Как видно из рисунков 2.2 и 2.3 характер изменения значений временного ряда влияет на геометрические характеристики контура лепестковой диаграммы. Поэтому предложено, использовать геометрические признаки контура как обещающие характеристики временного ряда. Т.е. при

анализе и построения моделей данных предложено заменять временной ряд $Z_1, Z_2, ..., Z_n$ на характеристики контура лепестковой диаграммы.

В качестве используемых характеристик лепестковой диаграммы при снижении размерности признакового пространства предложено использовать (рисунок 2.4):

- площадь *Area* контура лепестковой диаграммы
- периметра *Grith* контура лепестковой диаграммы
- максимального значения ряда *D_max*,
- координаты *X* и *Y* центра *Gravity* лепестковой диаграмм.

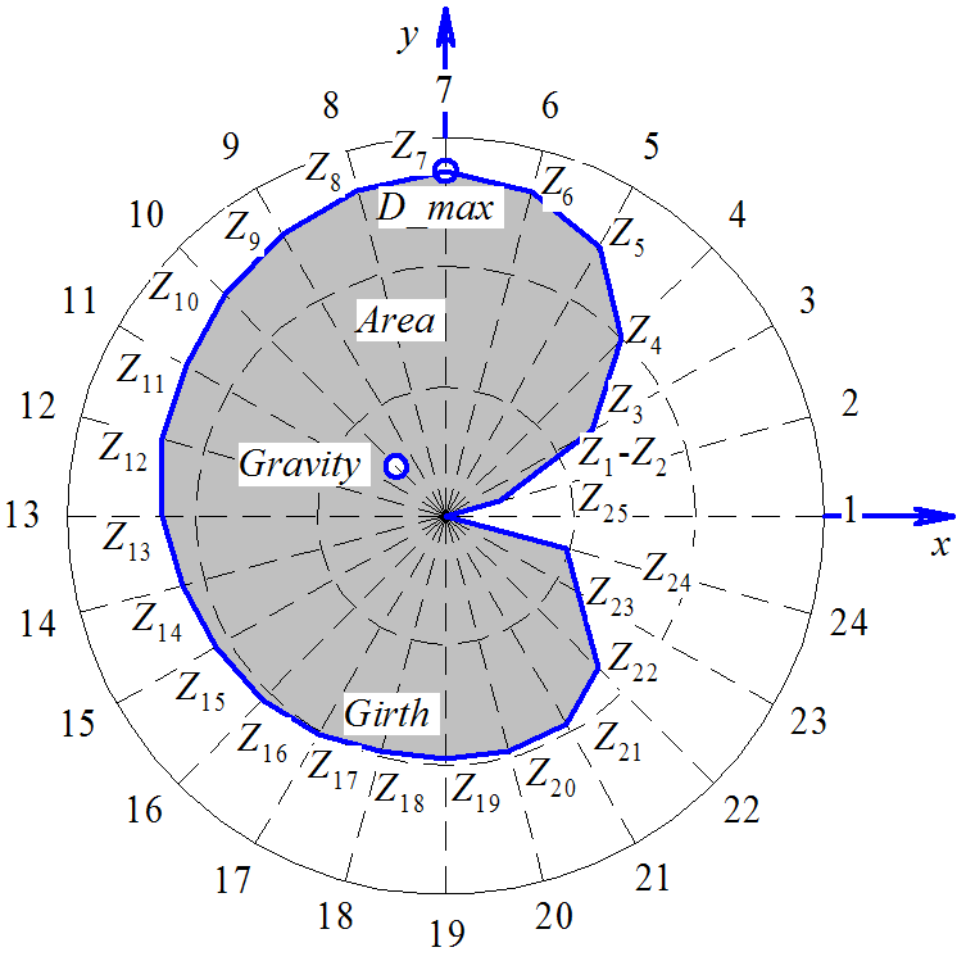



Рисунок 2.4 – Признаки лепестковой диаграммы: *D_max*, *Area*, *Gravity_x*, *Gravity_y*, *Girth*)

Приведем формулы расчета этих характеристик на основе значений временного ряда $Z_1, Z_2, ..., Z_n$.

В общем случае площадь *Area* и периметр *Girth* рассчитывается так, как это показано на формулах (2.1) и (2.2).

$$Area = \sum_{j=1}^{n} S_j = \frac{1}{2}\sin(\alpha)\left(\sum_{j=1}^{n-1} Z_j Z_{j+1} + Z_n Z_1\right) \tag{2.1}$$

$$Girth = \sum_{j=1}^{n-1}\sqrt{Z_j^2 + Z_{j+1}^2 - 2Z_j Z_{j+1}\cos\left(\frac{2\pi}{n}\right)} + \sqrt{Z_n^2 + Z_1^2 - 2Z_1 Z_n \cos\left(\frac{2\pi}{n}\right)} \tag{2.2}$$

Если во временной ряд состоит из 24 значение, то эти формулы можно представить как (2.3) и (2.4):

$$Area = \sum_{j=1}^{24} S_j = \frac{1}{2}\sin\left(\frac{\pi}{12}\right)\left(\sum_{j=1}^{24} Z_j Z_{j+1}\right), \quad Z_{25} = Z_1 \tag{2.3}$$

$$Girth = \sum_{j=1}^{24}\sqrt{Z_j^2 + Z_{j+1}^2 - 2Z_j Z_{j+1}\cos\left(\frac{\pi}{12}\right)}, \quad Z_{25} = Z_1 \tag{2.4}$$

Координаты *X* и *Y* центра *Gravity* лепестковой диаграмм рассчитывается по формулам (2.5) и (2.6).

$$Gravity\_x = \frac{\sum_{i=1}^{n-1} x_i^2 y_{i+1} + x_n^2 y_1 - \sum_{i=1}^{n-1} x_{i+1}^2 y_i - x_1^2 y_n + \sum_{i=1}^{n-1} x_i x_{i+1} y_{i+1} + x_n x_1 y_1}{3\left(\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right)} - \frac{\sum_{i=1}^{n-1} x_i x_{i+1} y_i + x_n x_1 y_n}{3\left(\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right)} \tag{2.5}$$

$$Gravity\_y = \frac{\sum_{i=1}^{n-1} x_i y_{i+1}^2 + x_n y_1^2 - \sum_{i=1}^{n-1} x_{i+1} y_i^2 - x_1 y_n^2 + \sum_{i=1}^{n-1} x_i y_i y_{i+1} + x_n y_n y_1}{3\left(\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right)} - \frac{\sum_{i=1}^{n-1} x_i x_{i+1} y_i + x_n x_1 y_n}{3\left(\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right)} \tag{2.6}$$

Таким образом, по формулам (2.1-2.5) осуществляется преобразование признакового пространства вида $z_1, z_2, ..., z_n \to D\_max, Area, Grith, X, Y$

## 2.2 Описание выборки данных для тестирования алгоритма

Протестируем предложенный алгоритм снижения размерности признакового пространства на выборке данных временных рядов по названием «Dataset: Chinatown», взятой с сервиса «UEA & UCR Time Series Classification Repository».

Данная выборка данных содержит почасовую информацию о количестве людей двигающихся в городе Мельбурн (Австралия) по улице «Chinatown-Swanston St.» за 2017 год. Всего в данной выборке данных информация о трафике за 363 дней. Сервис «UEA & UCR Time Series Classification Repository» предлагает использовать эти данные для построения модели классификации, способной по изменению трафика людей за день определять выходной этот день (СБ, ВС) или будний (ПН, ВТ, СР, ЧТ, ПТ).


Рисунок 2.5 – Данные о трафике города Мельбурн

Данные о трафике разделены на 2 файла: «chinatown-train.csv» и «chinatown-test.csv». В первом файле содержится информация для построения модели классификации, а во втором – данные с для тестирования точности полученной модели классификации.

Фрагмент данных содержащихся в файле chinatown-train.csv представлен на рисунке 2.6.

Столбец z1 показывает количество людей, двигающихся по улице «Chinatown-Swanston St.» в период времени с 0:00 до 1:00; столбец z2 – с 1:00 до 2:00; столбец z3 – с 2:00 до 3:00 и т.д. Последний столбец «C» указывает какой это был день: значение «1» означает, что это был выходной день (СБ или ВС), а значение «2» – что рабочий день (ПН, ВТ, СР, ЧТ, ПТ). В файле chinatown-train.csv содержатся данные за 343 дня.

В файле chinatown-test.csv сдержаться аналогичные данные, только за другие 20 дней.

| | z1 | z2 | z3 | z4 | z5 | ... | z20 | z21 | z22 | z23 | z24 | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 501.0 | 328.0 | 195.0 | 218.0 | 67.0 | ... | 1144.0 | 905.0 | 690.0 | 386.0 | 192.0 | 1 |
| **1** | 880.0 | 752.0 | 913.0 | 863.0 | 402.0 | ... | 979.0 | 706.0 | 585.0 | 356.0 | 187.0 | 1 |
| **2** | 493.0 | 389.0 | 174.0 | 121.0 | 82.0 | ... | 1697.0 | 1456.0 | 1319.0 | 1179.0 | 848.0 | 1 |
| **3** | 616.0 | 323.0 | 162.0 | 166.0 | 68.0 | ... | 1282.0 | 1078.0 | 857.0 | 498.0 | 248.0 | 1 |
| **4** | 389.0 | 276.0 | 161.0 | 124.0 | 35.0 | ... | 1057.0 | 1014.0 | 987.0 | 836.0 | 680.0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **338** | 140.0 | 57.0 | 45.0 | 32.0 | 19.0 | ... | 1301.0 | 1056.0 | 846.0 | 635.0 | 378.0 | 2 |
| **339** | 120.0 | 57.0 | 37.0 | 28.0 | 13.0 | ... | 1055.0 | 753.0 | 630.0 | 393.0 | 232.0 | 2 |
| **340** | 207.0 | 147.0 | 71.0 | 57.0 | 39.0 | ... | 1478.0 | 1506.0 | 1290.0 | 1106.0 | 769.0 | 2 |
| **341** | 293.0 | 180.0 | 73.0 | 96.0 | 85.0 | ... | 1505.0 | 1553.0 | 1399.0 | 993.0 | 557.0 | 2 |
| **342** | 149.0 | 88.0 | 39.0 | 30.0 | 22.0 | ... | 1355.0 | 1189.0 | 961.0 | 690.0 | 328.0 | 2 |

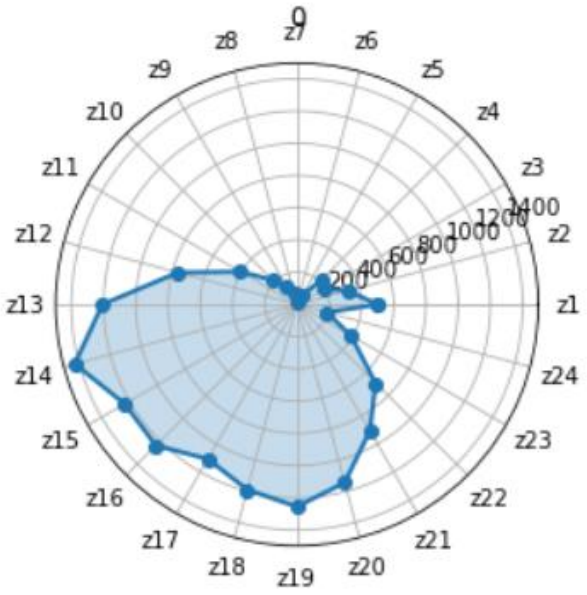
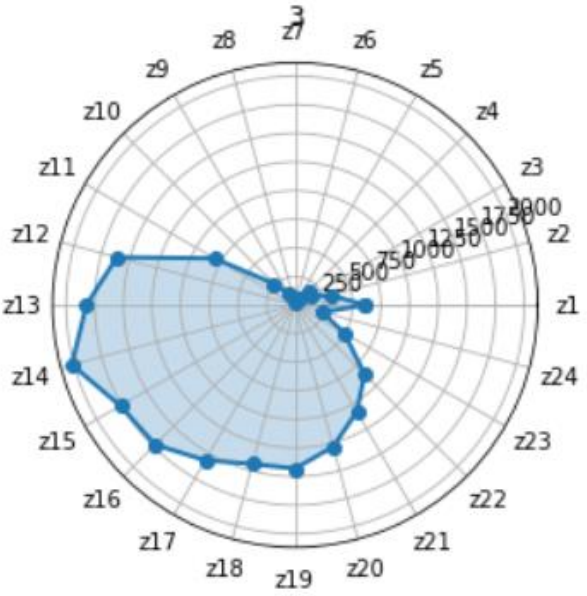
Рисунок 2.6 – Фрагмент исходных данных

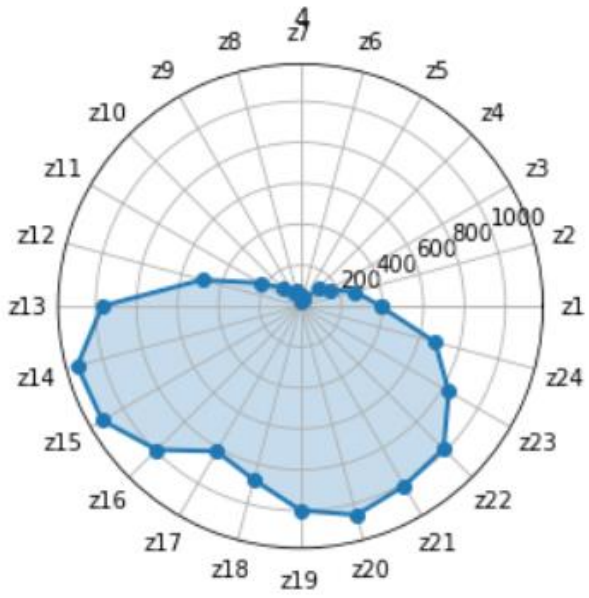
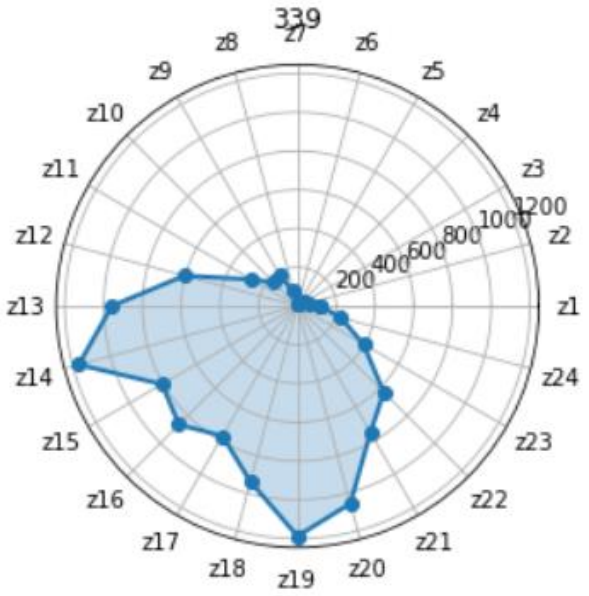
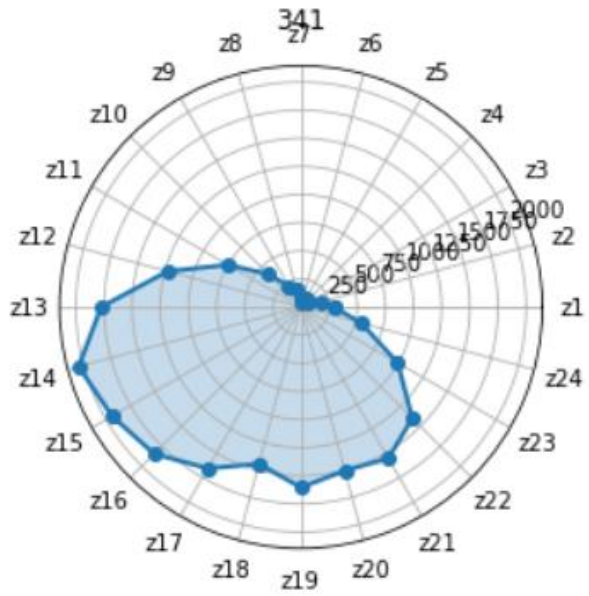
### 2.3 Результаты тестирования алгоритма

Протестируем предложенный алгоритм снижения размерности признакового пространства, на данных, описанных в разделе 2.2.

Фрагмент данных после преобразования признакового пространства временных рядов показан на рисунке 2.7. А в таблице 1 приведены примеры лепестковых диаграмм временных рядов из разных классов.

Таблица 1 – Лепестковые диаграммы и их параметры для некоторых временных рядов из описанной выше выборки данных

| **Лепестковая диаграмма** | **Показатели** | **Класс** |
|---|---|---|
|  | *D_max*= 1427.0<br>*Area*= 1859942.63<br>*Grith*= 5874.55<br>*X*= -3783.1628<br>*Y*= -6668.371 | Метка «C1» – Выходной день (СБ, ВС) |
|  | *D_max*= 2019.0<br>*Area*= 3488158.49<br>*Grith*= 7768.86<br>*X*= -6831.2882<br>*Y*= -8313.2885 | Метка «C1» – Выходной день (СБ, ВС) |

| | | |
|---|---|---|
|  | *D_max*= 1128.0<br>*Area*= 1484795.55<br>*Grith*= 4826.70<br>*X*= -1415.7864<br>*Y*= -6490.031 | Метка «C1» – Выходной день (СБ, ВС) |
|  | *D_max*= 1187.0<br>*Area*= 1166173.91<br>*Grith*= 4603.11<br>*X*= -3190.7563<br>*Y*= -5702.0652 | Метка «C2» – Рабочий день (ПН, ВТ, СР, ЧТ, ПТ) |
|  | *D_max*= 2045.0<br>*Area*= 4088495.99<br>*Grith*= 7809.67<br>*X*= -6007.2622<br>*Y*= -9982.6549 | Метка «C2» – Рабочий день (ПН, ВТ, СР, ЧТ, ПТ) |

| | D_max | Area | Girth | X | Y | C |
|---|---|---|---|---|---|---|
| **0** | 1427.0 | 1.859943e+06 | 5874.559152 | -3783.1628 | -6668.3710 | C1 |
| **1** | 1193.0 | 1.658074e+06 | 6416.925548 | -930.0078 | -4273.2215 | C1 |
| **2** | 1697.0 | 3.013639e+06 | 6704.434678 | -1973.6967 | -9675.0181 | C1 |
| **3** | 2019.0 | 3.488158e+06 | 7768.866533 | -6831.2882 | -8313.2885 | C1 |
| **4** | 1128.0 | 1.484796e+06 | 4826.703199 | -1415.7864 | -6490.0310 | C1 |
| **...** | ... | ... | ... | ... | ... | ... |
| **338** | 1301.0 | 1.476727e+06 | 4980.163460 | -2624.7002 | -6664.1868 | C2 |
| **339** | 1187.0 | 1.166174e+06 | 4603.114037 | -3190.7563 | -5702.0652 | C2 |
| **340** | 1506.0 | 2.507736e+06 | 6446.992784 | -2460.4749 | -8715.4805 | C2 |
| **341** | 2045.0 | 4.088496e+06 | 7809.674970 | -6007.2622 | -9982.6549 | C2 |
| **342** | 1355.0 | 1.797546e+06 | 5510.707813 | -3230.1488 | -7168.6934 | C2 |

Рисунок 2.7 – Фрагмент данных после снижения размерности признакового пространства

На основе исходных данных временных рядов (рисунок 2.6) с использованием алгоритма CART построено дерево классификации. Графическое представление дерева показано на рисунке 2.8. Прямоугольниками на рисунке обозначены узлы дерева, первое строчка в каждом узле – условие разбиения; значение «Samples» показывает количество элементов таблицы, попавшей в данный узел; листах «Class» обозначает возвращаемую метку класса (Метка «С1» – Выходной день (СБ, ВС), Метка «С2» – Рабочий день (ПН, ВТ, СР, ЧТ, ПТ)).

Аналогичное дерево классификации построено для данных временных рядов после преобразования признакового пространства (рисунок 2.7). Графическое представление второго дерева показано на рисунке 2.9.


Рисунок 2.8 – Дерево принятие решений, полученное без использования алгоритма снижения размерности признакового пространства

На основе анализа полученных деревьев (рисунок 2.8 и рисунок 2.9) можно провести анализ целесообразности использования предложенного алгоритма снижения размерности признакового пространства к исходным данным (временным рядам изменения потока людей в различные дни года).

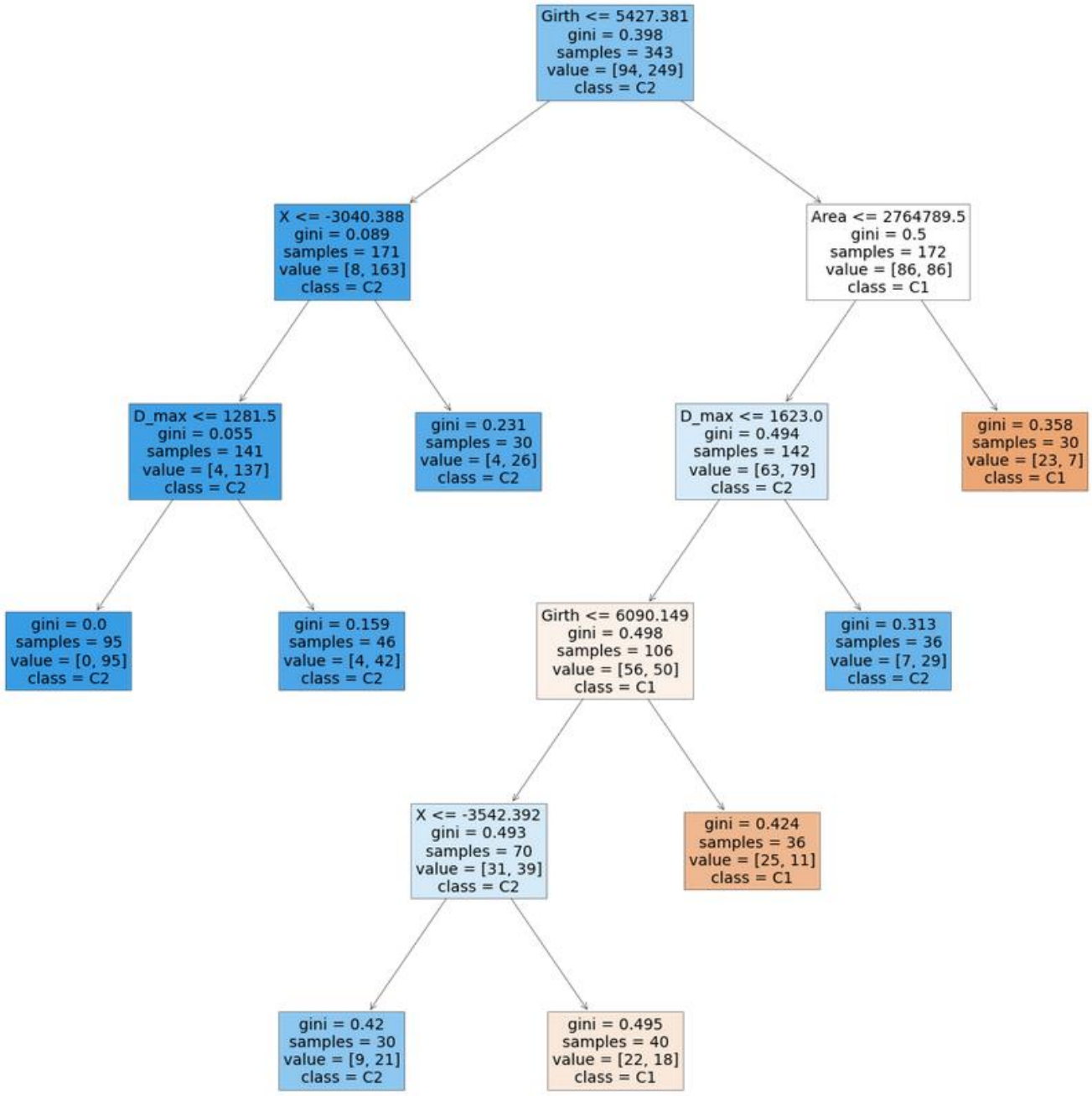



Рисунок 2.9 – Дерево принятие решений, полученное по тем же исходным данным, но с использованием алгоритма снижения размерности признакового пространства

Для этого сведем характеристики этих деревьев классификации в таблицу 2. При сравнении деревьев будем использовать следующие характеристики: количество листов (чем меньше, тем лучше), максимальная длина правила (чем меньше, тем лучше), минимальное количество объектов в листах (чем больше, тем лучше), максимальное количество объектов в листах

(чем меньше, тем лучше), точность классификации на тестовой выборке данных.

Таблица 2 – Результаты сравнения полученных деревьев классификации для оценки эффективности предложенного алгоритма (подчеркивание выделены лучшие параметры)

| № | Характеристика дерева | Первое дерево классификации без преобразования признакового пространства (рисунок 2.8) | Второе дерево классификации с применением преобразования признакового пространства (рисунок 2.9) |
|---|---|---|---|
| 1 | Количество листов (чем меньше, тем лучше) | 10 листов | 8 листов |
| 2 | Максимальная длина правила (чем меньше, тем лучше) | 6 узлов до достижения листа | 5 узлов до достижения листа |
| 3 | Минимальное количество объектов в листах (чем больше, тем лучше) | 1 объект | 30 объектов |
| 4 | Максимальное количество объектов в листах (чем меньше, тем лучше) | 244 объекта | 95 объектов |
| 5 | Точность классификации на тестовой выборке данных | 90% | 80% |

Результаты, представленные в таблице 2, показывают, что благодаря использованию предложенного алгоритма преобразования признакового

пространства дерево классификации получилось более компактным, количество правил уменьшилось на 20%, из структуры дерева пропали частные правила (которые распространяются на малое количество случаев). При этом точность классификации уменьшилось незначительно.

Следовательно, предложенный алгоритм преобразования признакового пространства может быть использован для предобработки данных временных рядов, и может оказывать положительное влияние на качество получаемых моделей классификации.

Дополнительно стоит отметить что эффективность предложенного алгоритма в большой мере зависит от природы данных. Поэтому целесообразность применения алгоритма нужно проверять отдельно для каждого набора данных, также как это было описано выше.

**Выводы по главе**

Предложен алгоритм снижения размерности признакового пространства временных рядов основанный на построении по набору данных лепестковой диаграммы и определении ее 5 характеристик: периметр диаграммы, площадь диаграммы, максимальное значение на диаграмме, координаты X и Y центра диаграммы. Таким образом, количество признаков временного ряда снижается до 5. Проведенный вычислительный эксперимент на данных сервиса «UEA & UCR Time Series Classification Repository» показал, что использование предложенного алгоритма снижения размерности признакового пространства практически не влияет на точность получаемого дерева классификации (90% - точность без снижения размерности, 80% - точность при использовании предложенного алгоритма). Однако классификационная модель получается более компактной, а правила в модели более общие.

# ГЛАВА 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛОЖЕННЫХ РЕШЕНИЙ

## 3.1 Особенности разработанного программного обеспечения

В рамках выполнения бакалаврской работы на высокоуровневом языке программирования общего назначения Python было разработано приложение для тестирования, предложенного во второй главе алгоритма. В качестве основных функциональных особенностей приложения можно выделить следующее:

1. Реализована возможность импортирования исходных данных (обучающая выборка с временными рядами) из файлов формата csv. При этом загруженные данные визуализируются в приложении в виде таблицы.

2. В приложении реализован алгоритм снижения размерности признакового пространства временных рядов. Поддерживаемое количество независимых переменных ограничивается только памятью ПК.

3. Реализованы следующие возможности по визуализации данных: построение лепестковой диаграммы для любого временного ряда из исходных данных (обращение к временному ряду осуществляется по номеру строки); построение двух деревьев принятия решений (при использовании алгоритма снижения размерности признакового пространства и без его использования).

4. Реализован расчет основных показателей лепестковых диаграмм (площади *Area,* периметра *Grith,* максимального значения ряда *D_max,* координаты центра лепестковой диаграммы *X* и *Y*).

5. Реализована возможность преобразования таблицы с исходными данными со столбцами вида $z_1$, $z_2$, …, $z_n$ в таблицу со столбцами *D_max, Area, Grith, X, Y*:

$$z_1, z_2, ..., z_n \rightarrow D\_max, Area, Grith, X, Y, \quad (3.1)$$

где $z_1, z_2, \ldots, z_n$ – значения временного ряда, $n$ – размерность ряда.

6. Реализована возможность расчета точности моделей классификации построенных с использованием предложенного алгоритма и без его использования.

Алгоритм работы с программным обеспечением следующий:

1. Пользователь задает в программе путь до файла формата csv, который содержит в себе данные нескольких временных рядов и метки класса. Количество столбцов в csv файле равно размерности *n* временного ряда плюс столбец *C*, отвечающий за метку класса. В таблице следующие столбцы – $z_1, z_2, \ldots, z_n$, *C*.

2. Затем пользователь, задавая номер строки в исходной таблице, может просматривать внешний вид лепестковых диаграмм, построенных по этим данным.

3. Затем пользователь запускает расчета по исходным данным новой таблицы, содержащей в себе параметры лепестковой диаграммы (площади *Area,* периметра *Grith,* максимального значения ряда *D_max,* координаты центра лепестковой диаграммы *X* и *Y*) и столбец *C*, содержащий за метку класса. В данной таблице следующие столбцы – *D_max*, *Area*, *Grith*, *X*, *Y, C*.

4. Таким образом, в программе на данном этапе будет содержаться две таблицы: исходные временные ряды и данные временных рядов преобразованные с использованием предложенного алгоритма снижения размерности признакового пространства.

5. Для оценки эффективности применения алгоритма снижения размерности признакового пространства к исходным данным пользователь может инициировать построение двух классификационных моделей: дерева принятия решения по исходным данным и дерева принятия решений по данным лепестковых диаграмм. Оба дерева будут отображены в окне программы. По тестовой выборке данных будет оценена точность классификации обоих деревьев.

6. Пользователь оценивает полученные с помощью программы данные. Если точность классификации от использования алгоритма снижения размерности признакового пространства изменилась не сильно, а размер дерева классификации уменьшился, то делается вывод, что для этих данных алгоритм подходит. В противном случае делается вывод, что алгоритм исходных данных не эффективен.

### 3.2 Описание программного кода

Опишем программный код, описание результатов работы каждого блока кода будет приведено на исходных данных, описанных в разделе 2.2.

Программный код разделен на 2 файла: spidercharts-draw.ipynb и spidercharts-trees.ipynb. В файле spidercharts-draw.ipynb храниться код, отвечающий за графическое отображение лепестковых диаграмм, а в файле spidercharts-trees.ipynb – код отвечающий, за реализацию алгоритма снижения размерности признакового пространства и оценку алгоритма с точки зрения построения классификационных моделей (деревьев принятия решений).

Рассмотрим содержание файла spidercharts-draw.ipynb, в нем используются следующие компоненты (рисунок 3.1):

- библиотека pandas, которая используется для работы с фрагментами данных (data frames);
- библиотека numpy, которая используется для вызова математических функций и при работе с массивами данных;
- библиотека matplotlib, которая используется для построения графиков;
- библиотека seaborn, которая является дополнением для matplotlib, и предназначена для визуализации данных.

```
# Import libs
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

Рисунок 3.1 – Подключение библиотек к проекту

После подключения описанных выше компонентов в программном коде осуществляется импорт данных временных рядов с использование функции read_csv() из библиотеки pandas. Затем для контроля результата процедуры импорта на экран выводится таблица с первыми пятью и последними пятью строками с загруженными из файла данными, а также размер сформированной таблицы (количество строк и столбцов).

В дальнейшем доступ к данным из таблицы осуществляется через объект df типа data frame (рисунок 3.2).

```
# Импорт

df=pd.read_csv("chinatown_train.csv")
df
```

| | z1 | z2 | z3 | z4 | z5 | z6 | z7 | z8 | z9 | z10 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 501.0 | 328.0 | 195.0 | 218.0 | 67.0 | 17.0 | 28.0 | 72.0 | 132.0 | 215.0 | ... |
| **1** | 880.0 | 752.0 | 913.0 | 863.0 | 402.0 | 112.0 | 60.0 | 112.0 | 119.0 | 186.0 | ... |
| **2** | 493.0 | 389.0 | 174.0 | 121.0 | 82.0 | 36.0 | 27.0 | 64.0 | 127.0 | 203.0 | ... |
| **3** | 616.0 | 323.0 | 162.0 | 166.0 | 68.0 | 26.0 | 34.0 | 68.0 | 123.0 | 263.0 | ... |
| **4** | 389.0 | 276.0 | 161.0 | 124.0 | 35.0 | 26.0 | 51.0 | 75.0 | 71.0 | 126.0 | ... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **338** | 140.0 | 57.0 | 45.0 | 32.0 | 19.0 | 23.0 | 32.0 | 89.0 | 159.0 | 207.0 | ... |
| **339** | 120.0 | 57.0 | 37.0 | 28.0 | 13.0 | 14.0 | 18.0 | 80.0 | 187.0 | 174.0 | ... |
| **340** | 207.0 | 147.0 | 71.0 | 57.0 | 39.0 | 29.0 | 36.0 | 85.0 | 240.0 | 210.0 | ... |
| **341** | 293.0 | 180.0 | 73.0 | 96.0 | 85.0 | 44.0 | 86.0 | 159.0 | 218.0 | 414.0 | ... |
| **342** | 149.0 | 88.0 | 39.0 | 30.0 | 22.0 | 9.0 | 30.0 | 64.0 | 215.0 | 267.0 | ... |

343 rows × 25 columns

Рисунок 3.2 – Импорт данных из csv файла и их отображение

Теперь, для подписей осей на лепестковых диаграммах требуется получить массив с названиями столбцов. Это осуществляется путем операции среза. Результат среза сохраняется переменную list_labels. Для контроля правильности выполнения операции содержимое переменной выводится на экран (рисунок 3.3).

```
list_labels=np.array(df.columns[0:-1])
list_labels
```

```
array(['z1', 'z2', 'z3', 'z4', 'z5', 'z6', 'z7', 'z8', 'z9', 'z10', 'z11',
       'z12', 'z13', 'z14', 'z15', 'z16', 'z17', 'z18', 'z19', 'z20',
       'z21', 'z22', 'z23', 'z24'], dtype=object)
```

```
len(df.index) #количество строк
```

```
343
```

Рисунок 3.3 – Получение массива с названиями столбцов

На следующем шаге пользователь может задать номер временного ряда (номер строки в таблице) для которого требуется построить лепестковую диаграмму. Заданный номер храниться в переменной id_stats. Из объекта df в соответствии с значение переменной id_stats извлекается массив значений выбранного временного ряда. Для контроля результата, извлечённые значения временного ряды выводятся на экран (рисунок 3.4).

```
# Получение данных по id_stats - номеру строки
id_stats=340
labels=list_labels
stats=df.loc[id_stats,labels].values
stats
```

```
array([ 207.,  147.,   71.,   57.,   39.,   29.,   36.,   85.,  240.,
        210.,  391.,  624., 1343., 1473., 1178., 1150., 1090., 1200.,
       1440., 1478., 1506., 1290., 1106.,  769.])
```

Рисунок 3.4 – Получение массива со значениями определенного временного ряда

Следующим шагом проводиться расчет вспомогательных значений, необходимых для построения лепестковой диаграммы: углов angles расположение осей и координат stats точек на этих осях (рисунок 3.5).

```
# Вспомогательные расчеты
angles=np.linspace(0, 2*np.pi, len(labels), endpoint=False)
stats=np.concatenate((stats,[stats[0]]))
angles=np.concatenate((angles,[angles[0]]))
```

Рисунок 3.5 – Вспомогательные расчеты для построения лепестковой диаграммы

Затем по полученным данным строиться лепестковая диаграмма, которая выводиться на экран пользователя (рисунок 3.6).

```
# Построение графика
fig = plt.figure()
ax = fig.add_subplot(111, polar=True)
ax.plot(angles, stats, 'o-', linewidth=2)
ax.fill(angles, stats, alpha=0.25)
ax.set_thetagrids(angles * 180/np.pi, labels)
ax.set_title(str(id_stats))
ax.grid(True)

plt.show()
```

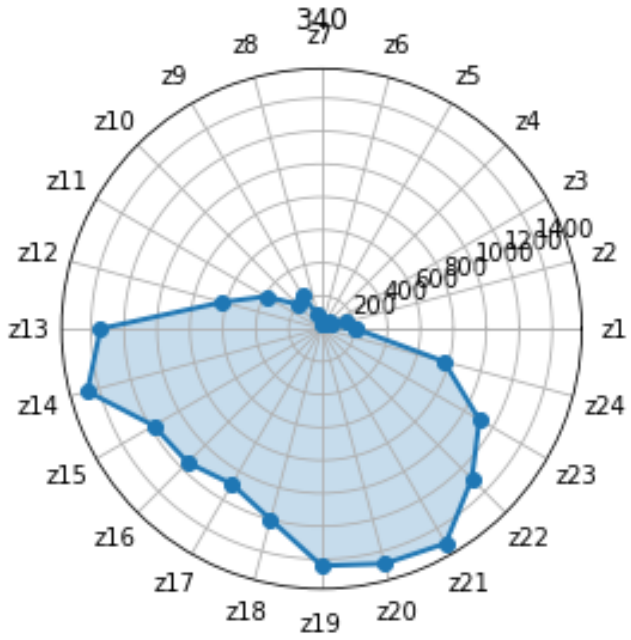



Рисунок 3.6 – Построение и вывод лепестковой диаграммы

Отдельно стоит отметить, что в библиотеке matplotlib отсутствует стандартная реализация построение лепестковых диаграмм. По этой причине данная диаграмма строить как полярная система координат. Поэтому и требуются дополнительные расчёты, которые были представлены выше.

Теперь рассмотрим код, содержащийся в файле spidercharts-trees.ipynb.

```
#Подключение билиотек
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn import tree
```

```
# Импорт
df=pd.read_csv("chinatown_train.csv")
df
```

| | z1 | z2 | z3 | z4 | ... | z21 | z22 | z23 | z24 | C |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 501.0 | 328.0 | 195.0 | 218.0 | ... | 905.0 | 690.0 | 386.0 | 192.0 | 1 |
| **1** | 880.0 | 752.0 | 913.0 | 863.0 | ... | 706.0 | 585.0 | 356.0 | 187.0 | 1 |
| **2** | 493.0 | 389.0 | 174.0 | 121.0 | ... | 1456.0 | 1319.0 | 1179.0 | 848.0 | 1 |
| **3** | 616.0 | 323.0 | 162.0 | 166.0 | ... | 1078.0 | 857.0 | 498.0 | 248.0 | 1 |
| **4** | 389.0 | 276.0 | 161.0 | 124.0 | ... | 1014.0 | 987.0 | 836.0 | 680.0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **338** | 140.0 | 57.0 | 45.0 | 32.0 | ... | 1056.0 | 846.0 | 635.0 | 378.0 | 2 |
| **339** | 120.0 | 57.0 | 37.0 | 28.0 | ... | 753.0 | 630.0 | 393.0 | 232.0 | 2 |
| **340** | 207.0 | 147.0 | 71.0 | 57.0 | ... | 1506.0 | 1290.0 | 1106.0 | 769.0 | 2 |
| **341** | 293.0 | 180.0 | 73.0 | 96.0 | ... | 1553.0 | 1399.0 | 993.0 | 557.0 | 2 |
| **342** | 149.0 | 88.0 | 39.0 | 30.0 | ... | 1189.0 | 961.0 | 690.0 | 328.0 | 2 |

343 rows × 25 columns

Рисунок 3.7 – Подключение библиотек к проекту и импорт исходных данных

Сначала в коде происходит подключение компонентов pandas, seaborn, numpy, matplotlib (их описание приводилось выше). А также библиотеки sklearn, отвечающей за построение деревьев классификации. Затем производиться импорт данных (временных рядов) из файла формата csv.

Результат импорта данных выводиться на экран пользователю и для контроля результатов отображается количество импортированных столбцов и строк (рисунок 3.7). Полученная таблица храниться в объекте df типа data frame.

Следующим шагом осуществляется премаркировка значений последнего столбца выборки данных (столбец «C»): значение «1» заменяется на «C1», «2» на «C2» и т.д. Это делается для того, чтобы методы библиотеки sklearn при построении деревьев воспринимали данные значения как метки класса. В коде перемаркировка реализована через словарь my_dict и функцию map() (рисунок 3.8).

Для построения дерева классификации также требуется получить массив, содержащий название всех столбцов таблицы. Данная операция реализована посредством выполнения среза данных. Результат среза сохраняется в переменную list_labels.

```
my_dict = {1 : 'C1', 2 : 'C2'} # замена 1 на C1, замена 2 на C2 чисто для нашей выборки
df[df.columns[-1]]=df[df.columns[-1]].map(my_dict)
#df.head(5), df.tail()
```

```
list_labels=np.array(df.columns[0:-1])
list_labels
```

```
array(['z1', 'z2', 'z3', 'z4', 'z5', 'z6', 'z7', 'z8', 'z9', 'z10', 'z11',
       'z12', 'z13', 'z14', 'z15', 'z16', 'z17', 'z18', 'z19', 'z20',
       'z21', 'z22', 'z23', 'z24'], dtype=object)
```

```
df_len=len(df.index) #количество строк
```

Рисунок 3.8 – Перемаркировка меток класса и получение массива с названиями независимых переменных

Также, для выполнения преобразования признакового пространства необходимо знать количество временных рядов в исходной таблице. Для этого используется функция len(), а результат сохраняется в переменной df_len (рисунок 3.8).

Следующим шагом является преобразования исходных данных временных рядов в признаковое пространство на основе характеристик лепестковых диаграмм. Для этого создается новый объект df1 типа data frame со столбцами D_max, Area, Girth, X, Y, C. Наполнение объекта df1 данным осуществляется в цикле, потем построчного считывания данных из df и расчета по ним показателей лепестковой диаграммы. Так в коде, представленном на рисунке 3.9, производится расчет максимального значения временного ряда *D_max* и площади *Area*.

```
# Получение данных по id_stats - номеру строки
id_stats=0

column_names = ["D_max", "Area", "Girth", "X", "Y", "C"]
df1 = pd.DataFrame(columns = column_names)

for id_stats in range(df_len):
    labels=list_labels
    stats=df.loc[id_stats,labels].values

    D_max=max(stats)
    stats_len=len(stats)
    #print(stats_len)

    area_sum=0
    for j in range(0, stats_len-1):
        area_sum+=stats[j]*stats[j+1]
    area_sum+=stats[stats_len-1]*stats[0]
    area=np.sin(2*np.pi/stats_len)*area_sum*0.5
```

Рисунок 3.9 – Начало цикла преобразования таблицы с исходными данными (расчет максимального значения ряда *D_max* и площади *Area)*

На рисунке 3.10 представлена заключительная часть цикла с расчетом периметра Grith и координат центра лепестковой диаграммы X и Y

```
girth_sum=0
for j in range(0, stats_len-1):
    girth_sum+=np.sqrt(np.square(stats[j])+np.square(stats[j+1]) \
    -(2*stats[j]*stats[j+1]*np.cos(2*np.pi/stats_len)))
girth_sum+=np.sqrt(np.square(stats[stats_len-1])+np.square(stats[0]) \
-(2*stats[stats_len-1]*stats[0]*np.cos(2*np.pi/stats_len)))
girth=girth_sum

x=0.0
y=0.0
for j in range(0, stats_len):
    x+=stats[j]*np.cos(2*np.pi/stats_len*j)
    #print("x=",x)
    y+=stats[j]*np.sin(2*np.pi/stats_len*j)
    #print("y=",y)
x=np.round(x, 4)
y=np.round(y, 4)
C=df.iloc[id_stats][-1]
#print(C)
df1.loc[id_stats] = [D_max, area, girth, x, y, C]
```

Рисунок 3.10 – Конец цикла преобразования таблицы с исходными данными

```
df1
```

| | D_max | Area | Girth | X | Y | C |
|---|---|---|---|---|---|---|
| **0** | 1427.0 | 1.859943e+06 | 5874.559152 | -3783.1628 | -6668.3710 | C1 |
| **1** | 1193.0 | 1.658074e+06 | 6416.925548 | -930.0078 | -4273.2215 | C1 |
| **2** | 1697.0 | 3.013639e+06 | 6704.434678 | -1973.6967 | -9675.0181 | C1 |
| **3** | 2019.0 | 3.488158e+06 | 7768.866533 | -6831.2882 | -8313.2885 | C1 |
| **4** | 1128.0 | 1.484796e+06 | 4826.703199 | -1415.7864 | -6490.0310 | C1 |
| **...** | ... | ... | ... | ... | ... | ... |
| **338** | 1301.0 | 1.476727e+06 | 4980.163460 | -2624.7002 | -6664.1868 | C2 |
| **339** | 1187.0 | 1.166174e+06 | 4603.114037 | -3190.7563 | -5702.0652 | C2 |
| **340** | 1506.0 | 2.507736e+06 | 6446.992784 | -2460.4749 | -8715.4805 | C2 |
| **341** | 2045.0 | 4.088496e+06 | 7809.674970 | -6007.2622 | -9982.6549 | C2 |
| **342** | 1355.0 | 1.797546e+06 | 5510.707813 | -3230.1488 | -7168.6934 | C2 |

343 rows × 6 columns

Рисунок 3.11 – Вывод данных после применения алгоритма снижения размерности признакового пространства

Результат преобразования временных рядов в новое признаковое пространство отображается пользователю в виде таблице, как это показано на рисунке 3.11.

На данном этапе у нас в df  хранятся временные ряды в неизменном виде, а в df1 – временные ряды в признаковом пространстве на основе лепестковых диаграмм. Теперь на основе этих двух таблиц будут построены два классификатора. Сравнение свойств классификаторов позволит сделать выводы о целесообразности применения алгоритма снижения размерности признакового пространства к исходному набору временных рядов.

Для того разметим данные, хранящиеся в двух таблицах, задав массивы независимых переменных first_x, second_x, и соответствующие им  массивы целевых значений first_y, second_y. Для задания массивов в нашем случае используется операции среза, как это показано на рисунке 3.12

```
first_names = df.columns[:-1]
first_x = df[first_names]
first_y = df[df.columns[-1]]
first_c_names=df[df.columns[-1]].unique()
first_c_names=np.sort(first_c_names)
first_c_names
```

```
array(['C1', 'C2'], dtype=object)
```

```
second_names = df1.columns[:-1]
second_x = df1[second_names]
second_y = df1[df1.columns[-1]]
second_c_names=df1[df1.columns[-1]].unique()
second_c_names=np.sort(second_c_names)
second_c_names
```

```
array(['C1', 'C2'], dtype=object)
```

Рисунок 3.12 – Формирование данных для построения двух деревьев классификации

Теперь, с использованием получившихся массивов данных построим два дерева классификации. Для этого воспользуемся алгоритмом CART, который реализован в одной из подключенных библиотек.

Дерево классификации first_tree будет построено на исходных данных временных рядов, а Дерево классификации second_tree на данных, полученных после преобразования признакового пространства. Код представлен на рисунке 3.13

```
first_tree = tree.DecisionTreeClassifier(criterion='gini', random_state=0, min_samples_leaf=1).fit(f
second_tree = tree.DecisionTreeClassifier(criterion='gini', random_state=0, min_samples_leaf=30).fit
```

Рисунок 3.13 – Задание параметров построения двух деревьев

Задание параметров отображения первого дерева (first_tree) показано на рисунке 3.14, а второго дерева (second_tree) показано на рисунке 3.15.

```
plt.style.use('seaborn-poster')
plt.figure(figsize=(20,20))
tree.plot_tree(first_tree.fit(first_x, first_y), \
               filled=True, feature_names=first_names, \
               class_names=first_c_names, fontsize=14)
```

```
[Text(418.5, 1009.5428571428572, 'z2 <= 209.0\ngini = 0.398\nsamples = 343\nvalue = [94, 249]\nclass = C2'),
 Text(186.0, 854.2285714285715, 'z4 <= 131.5\ngini = 0.008\nsamples = 245\nvalue = [1, 244]\nclass = C2'),
 Text(93.0, 698.9142857142858, 'gini = 0.0\nsamples = 244\nvalue = [0, 244]\nclass = C2'),
 Text(279.0, 698.9142857142858, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]\nclass = C1'),
 Text(651.0, 854.2285714285715, 'z1 <= 298.5\ngini = 0.097\nsamples = 98\nvalue = [93, 5]\nclass = C1'),
 Text(465.0, 698.9142857142858, 'z5 <= 75.5\ngini = 0.444\nsamples = 3\nvalue = [1, 2]\nclass = C2'),
 Text(372.0, 543.6, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]\nclass = C2'),
 Text(558.0, 543.6, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]\nclass = C1'),
 Text(837.0, 698.9142857142858, 'z21 <= 1630.0\ngini = 0.061\nsamples = 95\nvalue = [92, 3]\nclass = C1'),
 Text(744.0, 543.6, 'z3 <= 114.0\ngini = 0.042\nsamples = 94\nvalue = [92, 2]\nclass = C1'),
 Text(558.0, 388.28571428571433, 'z12 <= 751.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]\nclass = C1'),
 Text(465.0, 232.97142857142865, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]\nclass = C1'),
 Text(651.0, 232.97142857142865, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]\nclass = C2'),
 Text(930.0, 388.28571428571433, 'z22 <= 656.0\ngini = 0.022\nsamples = 92\nvalue = [91, 1]\nclass = C1'),
 Text(837.0, 232.97142857142865, 'z13 <= 1297.5\ngini = 0.198\nsamples = 9\nvalue = [8, 1]\nclass = C1'),
 Text(744.0, 77.65714285714284, 'gini = 0.0\nsamples = 8\nvalue = [8, 0]\nclass = C1'),
 Text(930.0, 77.65714285714284, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]\nclass = C2'),
 Text(1023.0, 232.97142857142865, 'gini = 0.0\nsamples = 83\nvalue = [83, 0]\nclass = C1'),
 Text(930.0, 543.6, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]\nclass = C2')]
```

Рисунок 3.14 – Задание параметров отображения первого дерева

```
plt.style.use('seaborn-poster')
plt.figure(figsize=(20,20))
tree.plot_tree(second_tree.fit(second_x, second_y), filled=True, feature_names=second_names, class_names=second_
```

```
[Text(620.0, 996.6, 'Girth <= 5427.381\ngini = 0.398\nsamples = 343\nvalue = [94, 249]\nclass = C2'),
 Text(372.0, 815.4000000000001, 'X <= -3040.388\ngini = 0.089\nsamples = 171\nvalue = [8, 163]\nclass = C2'),
 Text(248.0, 634.2, 'D_max <= 1281.5\ngini = 0.055\nsamples = 141\nvalue = [4, 137]\nclass = C2'),
 Text(124.0, 453.0, 'gini = 0.0\nsamples = 95\nvalue = [0, 95]\nclass = C2'),
 Text(372.0, 453.0, 'gini = 0.159\nsamples = 46\nvalue = [4, 42]\nclass = C2'),
 Text(496.0, 634.2, 'gini = 0.231\nsamples = 30\nvalue = [4, 26]\nclass = C2'),
 Text(868.0, 815.4000000000001, 'Area <= 2764789.5\ngini = 0.5\nsamples = 172\nvalue = [86, 86]\nclass = C1'),
 Text(744.0, 634.2, 'D_max <= 1623.0\ngini = 0.494\nsamples = 142\nvalue = [63, 79]\nclass = C2'),
 Text(620.0, 453.0, 'Girth <= 6090.149\ngini = 0.498\nsamples = 106\nvalue = [56, 50]\nclass = C1'),
 Text(496.0, 271.79999999999995, 'X <= -3542.392\ngini = 0.493\nsamples = 70\nvalue = [31, 39]\nclass = C2'),
 Text(372.0, 90.59999999999991, 'gini = 0.42\nsamples = 30\nvalue = [9, 21]\nclass = C2'),
 Text(620.0, 90.59999999999991, 'gini = 0.495\nsamples = 40\nvalue = [22, 18]\nclass = C1'),
 Text(744.0, 271.79999999999995, 'gini = 0.424\nsamples = 36\nvalue = [25, 11]\nclass = C1'),
 Text(868.0, 453.0, 'gini = 0.313\nsamples = 36\nvalue = [7, 29]\nclass = C2'),
 Text(992.0, 634.2, 'gini = 0.358\nsamples = 30\nvalue = [23, 7]\nclass = C1')]
```

Рисунок 3.15 – Задание параметров отображения первого дерева

Отображение деревьев в виде графов было показано во второй главе и, во избежание повторов, рисунки здесь продублированы не будут. Отметим только, что дерево first_tree выглядит так, как это показано на рисунке 2.8, а дерево second_tree – на рисунке 2.9.

Теперь, когда оба классификатора (деревья классификации) построены, можно перейти к проверке их точности на основе тестовой выборки данных. Для этого импортируем файл csv, содержащий тестовую выборку данных и с помощью функции score() проверим точность классификации первого дерева. Результаты тестирования выводятся пользователю на экран (рисунок 3.16).

```
df_test=pd.read_csv("chinatown_test.csv") # Расчет точности на тестовой выборке
my_dict = {1 : 'C1', 2 : 'C2'} # замена 1 на C1, замена 2 на C2 чисто для нашей выборки
df_test[df_test.columns[-1]]=df_test[df_test.columns[-1]].map(my_dict)
first_names_test = df_test.columns[:-1]
first_x_test = df_test[first_names_test]
first_y_test = df_test[df_test.columns[-1]]
first_tree.score(first_x_test,first_y_test)
```

```
0.9
```

Рисунок 3.16 – Загрузка тестовой выборки данных из файла csv и определение точности классификации (без использования алгоритма, точность 90%)

Аналогично проверим точность второго дерева (рисунок 3.17).

```
second_names_test = df1_test.columns[:-1]
second_x_test = df1_test[second_names_test]
second_y_test = df1_test[df1_test.columns[-1]]
second_tree.score(second_x_test,second_y_test)
```

```
0.8
```

Рисунок 3.17 – Определение точности классификации (с использованием алгоритма снижения размерности признакового пространства, точность 80%)

Теперь полученные данные пользователь может использовать для оценки целесообразности использования алгоритма снижения размерности признакового пространства для исходных данных (временных рядов).

**Выводы по главе**

На языке программирования Python было разработано программное обеспечение, реализующее предложенный алгоритм снижения размерности признакового пространства временных рядов. В программном обеспечении реализован функционал для сравнения результатов построение деревьев классификации, как при использовании предложенного алгоритма и без его применения. Такой функционал позволяет проводить оценку целесообразности применения алгоритма (снижения размерности признакового пространства) для конкретных анализируемых данных.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения бакалаврской работы были получены следующие результаты:

- Обзор литературных источников [1-25] показал, что перспективным направлением в области анализа данных является развитие алгоритмов снижения размерности признакового пространства. Их использование совместно с алгоритмами машинного обучения позволяется частично решить необходимости наличия большой обучающей выборки при обучении моделей классификации данных.
- В работе предложен алгоритм снижения размерности признакового пространства временных рядов основанный на построении по набору данных лепестковой диаграммы и определении ее 5 характеристик: периметр диаграммы, площадь диаграммы, максимальное значение на диаграмме, координаты X и Y центра диаграммы. Таким образом, количество признаков временного ряда снижается до 5..
- Проведенный вычислительный эксперимент на данных сервиса «UEA & UCR Time Series Classification Repository» показал, что использование предложенного алгоритма снижения размерности признакового пространства практически не влияет на точность получаемого дерева классификации (90% - точность без снижения размерности, 80% - точность при использовании предложенного алгоритма). Однако классификационная модель получается более компактной, а правила в модели более общие.
- Снижение количества независимых переменных позволяет производить обучение моделей классификации на меньшем наборе данных в обучающей выборке.
- На языке программирования Python было разработано программное обеспечение, реализующее предложенный алгоритм снижения

размерности признакового пространства временных рядов. В программном обеспечении реализован функционал для сравнения результатов построение деревьев классификации, как при использовании предложенного алгоритма и без его применения. Такой функционал позволяет проводить оценку целесообразности применения алгоритма (снижения размерности признакового пространства) для конкретных анализируемых данных.

В разработанном программном обеспечении присутствуют средства визуализации: построение лепестковых диаграмм для выбранного набора данных, графическое отображение деревьев классификации.

- Дальнейшее развитие предложенного подхода возможно за счет использования дополнительных параметров лепестковой диаграммы при описании временного ряда. Кроме того открытым для обсуждения является разработка рекомендаций для подбора оптимального сочетания параметров при описании конкретного временного ряда.

Таким образом, все задачи бакалаврской работы выполнены, а поставленная цель достигнута.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Афонин, А. Реализация алгоритма grassman&stiefel eigenmaps на языке python / А. Афонин, Ю. Янович // Междисциплинарная школа-конференция иппи ран "информационные технологии и системы 2018" (ИТИС 2018). Казань, 25-30 сентября 2018 г. – Казань : Издатель Институт проблем передачи информации им. А.А. Харкевича РАН, Москва, 2017. – Р. 52 – 59. – Текст : непосредственный.

2. Волков, К.В. Классификация радиолокационных объектов на основе выявления кластерной структуры данных поляризационной матрицы рассеяния / К.В. Волков, // Минцевские чтения. Москва, 30-31 января 2015 г. – Москва : Издатель Московский государственный технический университет имени н.Э. Баумана (национальный исследовательский университет), Москва, 2015. – Р. 101 – 114. – Текст : непосредственный.

3. Киров, Е.А. Использование методов снижения размерности в задаче классификации ковариационных матриц ЭЭГ / Е.А. Киров, М.Г. Беляев, Р.Х. Люкманов // Информационные технологии и системы 2015. Сочи, 07-11 сентября, 2015 г. – Сочи : Издатель Институт проблем передачи информации им. А.А. Харкевича РАН, Москва, 2015. – Р. 1113 – 1127. – Текст : непосредственный.

4. Дорофеюк, А.А. Методология структурно-классификационного исследования сложно организованной информации в задачах интеллектуального анализа данных / А.А. Дорофеюк, А.Ю. Дорофеюк, // XII всероссийское совещание по проблемам управления ВСПУ-2014. Москва, 16-19 июля 2014 г. – Москва : Издатель Институт проблем управления им. В.А. Трапезникова РАН Москва, 2014. – Р. 8369 – 8381. – Текст : непосредственный.

5. Ильина, М.А. Универсальный алгоритм визуализации решений задачи классификации / М.А. Ильина, А.А. Тузовский // Современные проблемы естественных и технических наук. Новосибирск, 24-25 мая 2016 г.

– Новосибирск : Издатель Новосибирский государственный архитектурно-строительный университет Сибстрин, Новосибирск, 2016. – P. 46 – 50. – Текст : непосредственный.

6. Рыцарев, И.А. Применение метода главных компонент для выявления семантических различий и анализа изменения положения в пространстве при анализе информационного контента сетевых сообществ / И.А. Рыцарев, Р.А. Парингер, А.В. Куприянов // V международная конференция и молодежная школа "информационные технологии и нанотехнологии". Самара, 21-24 мая 2019 г. – Самара : Издатель Новая техника , Самара, 2019. – P. 780 – 787. – Текст : непосредственный.

7. Исаченко, Р.В. Снижение размерности с помощью проекции на скрытое пространство в задаче декодирования сигналов / Р.В. Исаченко, В.В. Стрижков // Интеллектуализация обработки информации. Москва, Россия - Гаэта, Италия, 08-12 октября 2018 г. – Москва : Издатель Общество с ограниченной ответственностью "ТОРУС ПРЕСС", Москва, 2018. – P. 86 – 87. – Текст : непосредственный.

8. Вельдяйкин, Н. Алгоритм laplacian eigenmaps для точек вне обучающей выборки / Н. Вельдяйкин, Ю. Янович // Информационные технологии и системы 2017 (ИТИС 2017)Уфа, 14-17 сентября 2017 г. – Уфа : Издатель Институт проблем передачи информации им. А.А. Харкевича РАН, Москва, 2017. – P. 74 – 80. – Текст : непосредственный.

9. Буланов, О. Тестирование гипотезы о многообразии / О. Буланов, Ю. Янович // Информационные технологии и системы 2017 (ИТИС 2017). Уфа, 14-17 сентября 2017 г. – Уфа : Издатель Институт проблем передачи информации им. А.А. Харкевича РАН, Москва, 2017. – P. 41 – 48. – Текст : непосредственный.

10. Сидорова, В.А. Выбор размерности и детальности данных дистанционного зондирования земли при кластеризации гистограммным иерархическим алгоритмом / В.А. Сидорова // Актуальные проблемы вычислительной и прикладной математики, Новосибирск, 19-23 октября 2015

г.– Новосибирск : Издатель Институт вычислительной математики и математической геофизики СО РАН, Новосибирск, 2015. – P. 664 – 669. – Текст : непосредственный.

11. Palumbo, F. Clustering and Dimensionality Reduction to Discover Interesting Patterns in Binary Data / F. Palumbo, A.I. D'Enza // Advances in Data Analysis, Data Handling and Business Intelligence: Proceedings of the 32nd Annual Conference of the Gesellschaft für Klassifikation e.V., Joint Conference with the British Classification Society (BCS) and the Dutch/Flemish Classification Society (VOC), Helmut-Schmidt-University, Hamburg, July 16-18, 2008. – Hamburg : Springer-Verlag Berlin Heidelberg, 2009. – P. 45- 55. – Text : direct.

12. Lathauwer, D.R. Dimensionality Reduction in ICA and Rank-(*R*1,*R*2,...,*RN*) Reduction in Multilinear Algebra / D.R. Lathauwer, J.Vandewalle // International Conference on Latent Variable Analysis and Signal Separation, Independent Component Analysis and Blind Signal Separation: Fifth International Conference, ICA 2004, Granada, Spain, September 22-24, 2004. Proceedings. – Granada : Springer-Verlag Berlin Heidelberg, 2004. – P. 295- 302. – Text : direct.

13. Wenbin, Q. A Consistency-Based Dimensionality Reduction Algorithm in Incomplete Data / Q. Wenbin, S. Wenhao, W. Yinglong // Asia-Pacific Web Conference, Web Technologies and Applications: 16th Asia-Pacific Web Conference, APWeb 2014, Changsha, China, September 5-7, 2014. Proceedings. – Changsha : Springer International Publishing Switzerland, 2014. – P. 576 - 583. – Text : direct.

14. Charu, V.V. Dimensionality Reduction Using PCA Algorithm for Improving Accuracy in Prediction of Cardiac Ailments in Diabetic Patients / V.V. Charu, S. M. Ghosh // Proceedings of International Conference on Wireless Communication 17 November 2019, Mumbai, India. – Mumbai : Springer Nature Singapore Pte Ltd. 2020. – P. 443 - 452. – Text : direct.

15. Watanabe, K. Simultaneous Clustering and Dimensionality Reduction Using Variational Bayesian Mixture Model / K. Watanabe, S. Akaho, S. Omachi, M. Okada // Classification as a Tool for Research: Proceedings of the 11th IFCS

Biennial Conference and 33rd Annual Conference of the Gesellschaft für Klassifikation e.V., Dresden, March 13-18, 2009. – Dresden : Springer-Verlag Berlin Heidelberg 2010. – P. 81 - 89. – Text : direct.

16. Vidya, A. Dimensionality Reduction for Efficient Classification of DNA Repair Genes / A. Vidya, V. Manohar, V.P. Shwetha, K.R. Venugopal, L.M Patnaik // International Conference on Information Processing, Wireless Networks and Computational Intelligence: 6th International Conference on Information Processing, ICIP 2012, Bangalore, India, August 10-12, 2012. Proceedings. – Bangalore : Springer-Verlag Berlin Heidelberg 2012 . – P. 536 - 645. – Text : direct.

17. Reiter, S. Using an Autoencoder for Dimensionality Reduction in Quantum Dynamics / S. Reiter, T. Schnappinger, R. Vivie-Riedle // International Conference on Artificial Neural Networks, Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings. – Munich : Springer Nature Switzerland AG, 2019. – P. 783 - 787. – Text : direct.

18. Guo, Y. Twin Kernel Embedding with Relaxed Constraints on Dimensionality Reduction for Structured Data / Y. Cuo, J. Gao, P.W. Kwan // Australasian Joint Conference on Artificial Intelligence, AI 2007: Advances in Artificial Intelligence: 20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia, December 2-6, 2007. Proceedings. – Gold Coast : Springer-Verlag Berlin Heidelberg, 2007 . – P. 659 - 663. – Text : direct.

19. Xiang, Z. Unsupervised and Semi-supervised Dimensionality Reduction with Self-Organizing Incremental Neural Network and Graph Similarity Constraints / Z. Xiang, X. Zhu, H. Yourong, D. Wang, B. Fu, W. Chen // Pacific-Asia Conference on Knowledge Discovery and Data Mining, Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part I. – Auckland

: Springer International Publishing Switzerland, 2016 . – P. 191 - 202. – Text : direct.

20. Masip, D. An Experimental Comparison of Dimensionality Reduction for Face Verification Methods / D. Masip, J. Vitrià // Iberian Conference on Pattern Recognition and Image Analysis, Pattern Recognition and Image Analysis: First Iberian Conference, IbPRIA 2003, Puerto de Andratx, Mallorca, Spain, JUne 4-6, 2003. Proceedings. – Mallorca : pringer-Verlag Berlin Heidelberg, 2003. – P. 530 - 537. – Text : direct.

21. Zhu, L. Improvement of Decision Tree ID3 Algorithm / L. Zhu, Y. Yang // International Conference on Collaborative Computing: Networking, Applications and Worksharing, Collaborate Computing: Networking, Applications and Worksharing: 12th International Conference, CollaborateCom 2016, Beijing, China, November 10–11, 2016, Proceedings. – Beijing : ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2017. – P. 595 - 600. – Text : direct.

22. Bartczuk, L. A New Version of the Fuzzy-ID3 Algorithm / L. Bartczuk, D. Rutkowska // International Conference on Artificial Intelligence and Soft Computing, Artificial Intelligence and Soft Computing – ICAISC 2006: 8th International Conference, Zakopane, Poland, June 25-29, 2006. Proceedings. – Zakopane : Springer-Verlag Berlin Heidelberg, 2006 . – P. 1060 - 1070. – Text : direct.

23. Li, F. Fuzzy ID3 Algorithm Based on Generating Hartley Measure / F. Li, D. Jiang // International Conference on Web Information Systems and Mining, Web Information Systems and Mining: International Conference, WISM 2011, Taiyuan, China, September 24-25, 2011, Proceedings, Part II. – Taiyuan : Springer-Verlag Berlin Heidelberg, 2011 . – P. 188 – 195. – Text : direct.

24. Симонова, Д.А. Применение алгоритма ID3 для обучения робота / Д.А. Симонова, М.И. Зернов // Международная научно-техническая конференция "энергетика, информатика, инновации - 2016" .Смоленск, 24-25

ноября 2016 г. – Смоленск : Издатель Универсум Смоленск, 2016 . – Р. 343 – 346. – Текст : непосредственный.

25. Ляшов, Д.И. Программное средство визуализации сравнительного анализа алгоритмов построения деревьев решений / Д.И. Ляшов, Т.А. Медведева // VI международный семинар "Системный анализ, управление и обработка информации". Ростов-на-Дону, 19-24 октября 2015 г. – Ростов-на-Дону : Издатель Донской государственный технический университет, Ростов-на-Дону, 2015. – Р. 232 – 238. – Текст : непосредственный.