

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем и управления (КСУП)

К ЗАЩИТЕ ДОПУСТИТЬ

Заведующий кафедрой КСУП

доктор техн. наук, профессор

_____ Ю. А. Шурыгин

«___» _____ 2023 г.

Бакалаврская работа по направлению 09.03.01

«Информатика и вычислительная техника»

**РАЗРАБОТКА ПРОГРАММЫ НА ОСНОВЕ МАШИННОГО
ОБУЧЕНИЯ ДЛЯ ДИАГНОСТИКИ БОЛЕЗНИ ПАРКИНСОНА НА
НЕСБАЛАНСИРОВАННОМ НАБОРЕ ДАННЫХ HANDPD**

Студент гр. 589-3

_____ М. В. Пахомов

«___» _____ 2023 г.

Руководитель

доцент каф. КСУП, к.т.н.

_____ М. Б. Бардамова

«___» _____ 2023 г.

Томск 2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра компьютерных систем в управлении и проектировании (КСУП)

УТВЕРЖДАЮ

Заведующий кафедрой КСУП

д-р техн. наук, проф.

_____ Ю. А. Шурыгин

«___» _____ 2023г.

Индивидуальное задание

на выполнение выпускной квалификационной работы

студенту _____ Пахомову М. В. _____ гр. 589-3 факультета вычислительных систем

1. Тема работы: Разработка программы на основе машинного обучения для диагностики болезни Паркинсона на несбалансированном наборе данных HandPD. (утверждена приказом от _____ № _____).

2. Цель работы: Разработать программу для выявления болезни Паркинсона по оцифрованным изображениям рукописных рисунков и выполнить проверку её эффективности на наборе данных HandPD.

3. Содержание работы (перечень вопросов, подлежащих разработке):

3.1. Изучение предметной области и исследование набора данных HandPD.

3.2. Разработка программы для выявления БП по оцифрованным изображениям.

3.3. Апробация программы на наборе данных HandPD.

4. Дата выдачи задания: «___»___ 2023 г.

5. Дата сдачи работы на кафедру: «___»___ 2023 г.

Руководитель

Доцент каф. КСУП ТУСУР, к.т.н

Бардамова М.Б _____ «__» _____ 2023 г.
(Ф.И.О., должность) (подпись)

Задание согласовано:

Консультант по нормам и требованиям ЕСКД

доцент каф КСУП ТУСУР, к.т.н. Черкашин М.В.. _____
(Ф.И.О., должность) (подпись)

Задание принял к исполнению
студент группы 589-3

Пахомов М.В _____ «__» _____ 2023 г.
(Ф.И.О. студента) (подпись)

Реферат

Бакалаврская работа, 65 страниц, 22 рисунка, 16 таблиц, 32 источника и 1 приложение.

МАШИННОЕ ОБУЧЕНИЕ С УЧИТЕЛЕМ, КЛАССИФИКАЦИЯ НЕСБАЛАНСИРОВАННЫХ ДАННЫХ, ДИАГНОСТИКА ПАЦИЕНТОВ С ЗАБАЛИВАНИЕМ ПАРКИНСОНА, ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ, ДЕРЕВЬЯ РЕШЕНИЙ, НЕЙРОННЫЕ СЕТИ МНОГОСЛОЙНЫХ ПЕРЦЕПТРОНОВ, НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР, МЕТОД ОПОРНЫХ ВЕКТОРОВ, ДОБАВЛЕНИЕ ОБРАЗЦОВ, УДАЛЕНИЕ ОБРАЗЦОВ, КОМБИНИРОВАННЫЕ АЛГОРИТМЫ БАЛАНСИРОВКИ.

Объект исследования: алгоритмы машинного обучения с учителем, методы работы с несбалансированными данными, набор данных с результатами клинических исследований болезни Паркинсона.

Предмет исследования: программа для диагностики БП по оцифрованным рисункам на основе машинного обучения.

Цель данной работы: разработать программу для выявления болезни Паркинсона по оцифрованным изображениям рукописных рисунков и выполнить проверку её эффективности на наборе данных HandPD.

В результате были изучены алгоритмы машинного обучения, методы работы с несбалансированными данными, созданы модели классификации, проведены эксперименты по использованию моделей для диагностики болезни Паркинсона. Разработана программа на основе машинного обучения для диагностики болезни Паркинсона на несбалансированном наборе данных HandPD.

Выпускная квалификационная работа выполнена в текстовом редакторе Microsoft Word в соответствии с ОС ТУСУР 01–2021.

Abstract

Bachelor's thesis, 65 pages, 22 illustrations, 16 tables, 32 sources and 1 appendix.

SUPERVISED LEARNING, CLASSIFICATION OF IMBALANCED DATA, DIAGNOSIS OF PATIENTS WITH PARKINSON'S DISEASE, LOGISTIC REGRESSION, DECISION TREE, MULTILAYER PERCEPTRON NEURAL NETWORKS, NAÏVE BAYES CLASSIFIER, SUPPORT VECTOR MACHINE, OVER-SAMPLING, UNDER-SAMPLING, COMBINED BALANCING ALGORITHMS.

Object of study: machine learning algorithms with a teacher, methods for dealing with unbalanced data, dataset with the results of clinical studies of Parkinson's disease.

Subject of the study: a program for diagnosing PD from digitized drawings based on machine learning.

Purpose of this work: to develop a program for detecting Parkinson's disease from digitized images of handwritten drawings and to test its effectiveness on the HandPD dataset.

As a result, we studied machine learning algorithms, methods of working with unbalanced data, created classification models, and conducted experiments on using the models to diagnose Parkinson's disease. A machine learning-based program was developed to diagnose Parkinson's disease on an unbalanced HandPD dataset.

The graduate qualification work was done in the text editor Microsoft Word in accordance with "OS TUSUR 01-2021".

Оглавление

Введение	7
1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ.....	9
1.1 Машинное обучение	9
1.2 Применение машинного обучения в медицине	11
1.3 Диагностика болезни Паркинсона	13
2. ОСНОВНАЯ ЧАСТЬ.....	16
2.1 Исследование набора данных	16
2.2 Методы и алгоритмы	23
2.2.1 Методы нормализации данных	23
2.2.2 Методы отбора признаков	26
2.2.3 Методы кросс-валидации.....	30
2.2.4 Алгоритмы классификации	31
2.2.5 Алгоритмы балансировки данных	41
2.3 Описание разработанной программы, решающей задачу диагностики болезни Паркинсона на несбалансированном наборе данных.....	49
3. ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ.....	51
3.1 Описание экспериментов	51
3.2 Результаты экспериментов на несбалансированных данных	54
3.3 Результаты экспериментов на сбалансированных данных	55
3.3.1 Добавление экземпляров (Over-sampling)	55
3.3.2 Удаление экземпляров (Under-sampling)	56
3.3.3 Комбинированная выборка	57
3.4 Результаты экспериментов из аналогичных исследований	57
Заключение.....	60
Список использованных источников.....	62
Приложение А.....	66
Компакт диск:	(на обороте обложки)
— ПЗ;	
— презентация;	
— исходные коды.	

Введение

Болезнь Паркинсона (БП) — это хроническое дегенеративное заболевание, которое проявляется в виде тремора, медлительности движений, мышечной скованности, изменениях речи и навыков письма вследствие неврологических расстройств. Болезнь была впервые описана Джеймсом Паркинсоном, и ее симптомы хорошо известны научному сообществу. Однако, до сих пор не существует точной методики диагностики БП на ранних стадиях. Болезнь Паркинсона возникает в результате постепенного разрушения нервных клеток, производящих дофамин. В свою очередь, отсутствие этого вещества приводит к нарушениям передачи нервных импульсов и возникновению многих других симптомов, таких как депрессия, нарушения сна, ухудшение памяти и расстройства вегетативной нервной системы. Болезнь Паркинсона может быть вызвана как внешними, так и наследственными факторами [1].

Диагностика болезни Паркинсона является важной задачей, особенно на ранних стадиях, когда симптомы еще не слишком явны. В настоящее время методы диагностики болезни Паркинсона включают в себя наблюдение за симптомами, анамнез пациента и неврологические тесты. Однако, эти методы неэффективны на ранних стадиях, когда пациенты еще не проявляют сильных симптомов, что может привести к задержке в диагностике и лечении. Кроме того, эти методы являются дорогостоящими и могут быть недоступны для большинства людей.

Развитие технологий машинного обучения предоставляет новые возможности в области диагностики болезни Паркинсона. Анализ рукописных данных, таких как написание текста или рисование, является одним из подходов, который может быть использован для разработки модели машинного обучения, которая поможет в ранней диагностике болезни Паркинсона. Данная модель может использоваться для анализа несбалансированных рукописных данных, которые содержат в себе

информацию о движении руки, таких как скорость, амплитуда и другие параметры, которые могут быть связаны с болезнью Паркинсона.

Использование модели машинного обучения для диагностики болезни Паркинсона на основе анализа рукописных данных имеет потенциал ускорить и удешевить процесс диагностики, снизить количество ложноположительных и ложноотрицательных диагнозов, и улучшить качество жизни пациентов.

Цель данной работы состоит в разработке программы на основе машинного обучения для диагностики болезни Паркинсона по оцифрованным рисункам. Для достижения данной цели будет произведен анализ набора данных HandPD, включающего в себя данные о выполнении заданий на рисование простых фигур пациентами с болезнью Паркинсона и здоровых людей. На основе этого анализа будет проведен выбор оптимальных признаков и алгоритмов машинного обучения для разработки модели, которая сможет диагностировать болезнь Паркинсона на ранних стадиях на основе рукописных данных. Разработанная программа может помочь в повышении эффективности диагностики болезни Паркинсона на ранних этапах и снижении затрат на диагностику, что является важным вопросом в медицинской практике [2].

1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Машинное обучение

Машинное обучение (МО) — это подход в области искусственного интеллекта, который позволяет компьютерным системам обучаться и улучшать свою производительность в определенных задачах [3].

Вместо того, чтобы явно задавать правила и инструкции для выполнения определенных задач, системы машинного обучения обучаются на основе большого количества данных, используя различные алгоритмы и методы. Эти алгоритмы и методы позволяют системе автоматически обнаруживать закономерности и выявлять скрытые зависимости в данных, что позволяет системе делать более точные предсказания или принимать решения.

Существует три основных типа машинного обучения: обучение с учителем, обучение без учителя и обучение с подкреплением [4].

1 обучение с учителем является наиболее распространенным типом машинного обучения. В этом случае, компьютеру предоставляются наборы данных, где каждый элемент содержит входные данные и соответствующие им выходные данные. Модель обучается на основе этих данных, чтобы предсказывать выходные данные для новых входных данных;

2 обучение без учителя — это тип машинного обучения, в котором компьютеру предоставляется набор данных без указания выходных данных. В этом случае, модель обучается на основе структуры данных, чтобы выявить скрытые закономерности и структуры;

3 обучение с подкреплением — это тип машинного обучения, где модель обучается на основе обратной связи на протяжении серии действий. В этом случае, модель получает информацию о том, какие действия приводят к желаемому результату, а какие – нет.

Машинное обучение находит свое применение во многих областях, включая медицину, финансы, транспорт и т.д. Некоторые типовые задачи,

которые можно решать с помощью машинного обучения, включают в себя [5]:

- классификация — это задача, в которой необходимо разделить данные на несколько классов. Например, на основе симптомов пациента необходимо определить, болен он определенной болезнью или нет. Модели, используемые для классификации, обучаются на основе наборов данных, которые содержат входные данные и соответствующие им метки классов. Примеры алгоритмов классификации: логистическая регрессия, метод опорных векторов, случайный лес;

- регрессия/аппроксимация — это задача, в которой необходимо предсказать численный выход на основе входных данных. Например, на основе площади, количества комнат и других факторов необходимо предсказать цену на жилье. Модели, используемые для регрессии, обучаются на основе наборов данных, которые содержат входные данные и соответствующие им численные значения. Примеры алгоритмов регрессии: линейная регрессия, полиномиальная регрессия, метод опорных векторов;

- рекомендательные системы — это задача, в которой необходимо предложить пользователям персонализированные рекомендации на основе их предыдущих действий и интересов. Например, на основе истории просмотров пользователей можно предложить им фильмы или музыку, которые им могут понравиться. Примеры алгоритмов рекомендательных систем: коллаборативная фильтрация, содержательная фильтрация, гибридные системы;

- обработка естественного языка — это задача, в которой необходимо обрабатывать и анализировать естественный язык, к примеру, тексты на естественных языках. Так же на основе отзывов пользователей можно определить их настроение и оценку продукта. Примеры алгоритмов обработки естественного языка: классификация текстов, машинный перевод, генерация текстов.

Таким образом, использование методов машинного обучения может помочь в решении многих задач, где нет явного алгоритма решения или он слишком сложный. Однако, необходимо быть внимательным и аккуратным при выборе методов и алгоритмов, чтобы получить корректные и точные результаты.

1.2 Применение машинного обучения в медицине

Машинное обучение может использоваться в медицине для решения широкого спектра задач, таких как диагностика, прогнозирование, мониторинг и терапия. Оно имеет большой потенциал для улучшения качества здравоохранения, оптимизации лечения и улучшения результатов.

Машинное обучение может помочь улучшить точность диагностики, выявить скрытые паттерны и прогнозировать потенциальные проблемы здоровья до их появления. В медицине существует огромное количество данных, которые могут быть использованы для поддержки принятия решений. Эти данные могут быть собраны из различных источников, включая медицинские карты, изображения, лабораторные исследования и результаты клинических испытаний. Машинное обучение может использоваться для обработки и анализа этих данных, чтобы извлечь ценные знания и сделать предсказания о здоровье пациентов, что поможет в определении наиболее эффективных методов лечения и индивидуального подхода к каждому пациенту.

Машинное обучение может использоваться для:

- распознавания образований на изображениях. МО может использоваться для распознавания образований на рентгеновских снимках, ультразвуковых изображениях и МРТ. Например, система диагностики рака груди MammoScreen использует МО для автоматического распознавания и классификации изображений молочных желез. Так в исследовании [6] было проведено обширное тестирование MammoScreen, сравнение его работы с работой опытных рентгенологов. Инструмент ИИ продемонстрировал

высокую точность в обнаружении аномалий рака молочной железы, превзойдя отдельных радиологов в нескольких случаях. Более того, когда рентгенологи использовали инструмент ИИ в качестве вспомогательного средства, общая точность их диагнозов значительно повысилась. Этот совместный подход между ИИ и человеческим опытом продемонстрировал потенциал машинного обучения в преобразовании скрининга рака молочной железы;

- диагностика заболеваний. МО может помочь в диагностике заболеваний, особенно тех, которые трудно диагностировать на ранних стадиях. Например, система диагностики рака легких Lunit использует МО для анализа МРТ и рентгеновских снимков легких для обнаружения признаков рака легких. Например, в работе [7] интерпретация рентгеновских снимков грудной клетки требует экспертных знаний и опыта, поскольку даже незначительные отклонения от нормы могут указывать на основные заболевания. Однако субъективность и различный опыт рентгенологов могут привести к несоответствию диагноза и потенциальным ошибкам в интерпретации. Эти расхождения могут повлиять на результаты лечения пациентов, что подчеркивает необходимость объективного и надежного решения для повышения точности показаний рентгеновских снимков грудной клетки. Результаты показали, что алгоритм достиг значительного уровня совпадения с экспертами-рентгенологами. Более того, в некоторых сценариях производительность алгоритма даже превосходила производительность отдельных радиологов. Благодаря включению решения на основе глубокого обучения в диагностический процесс, исследование показало значительное улучшение точности и согласованности диагностики;

- прогнозирование прогноза заболевания. МО может использоваться для прогнозирования прогноза заболевания и определения лучшего пути лечения. Например, модель прогнозирования риска болезни сердца использует МО для анализа медицинских данных пациента, чтобы

определить риск развития сердечно-сосудистых заболеваний и разработать индивидуальный план лечения;

- оптимизация лечения. МО может использоваться для оптимизации лечения, например, для предсказания эффективности лекарственных препаратов и лучшего времени для их применения. Например, модель оптимизации лечения гепатита С использует МО для анализа медицинских данных пациента, чтобы определить наиболее эффективное время и способ лечения.

В целом, применение МО в медицине может снизить вероятность ошибок в диагностике, а также сократить время, затрачиваемое на диагностику и лечение пациентов. Это может быть особенно важно в случаях, когда настоящее время имеет решающее значение, например, в случае сердечного приступа или инсульта.

Кроме того, МО может помочь в разработке новых лекарств и лечебных методик. Анализ данных с помощью МО может выявлять новые связи между генетическими, метаболическими и другими факторами и заболеваниями. Это может привести к созданию более точных и эффективных методов лечения.

Таким образом, МО имеет огромный потенциал для улучшения качества медицинского ухода, снижения затрат на здравоохранение и сокращения времени на диагностику и лечение заболеваний.

1.3 Диагностика болезни Паркинсона

Болезнь Паркинсона — это неврологическое заболевание, характеризующееся постепенной дегенерацией дофаминергических нейронов в головном мозге. Эта дегенерация приводит к двигательным симптомам, таким как тремор, ригидность и брадикинезия, а также может влиять на когнитивные и эмоциональные функции. К сожалению, диагностика болезни Паркинсона может быть сложной задачей из-за отсутствия точных

диагностических тестов и совпадения симптомов с другими двигательными расстройствами.

Одним из подходов к решению этой проблемы является использование машинного обучения, которое позволяет повысить точность диагностики путем анализа закономерностей в больших массивах данных о пациентах. В этом контексте диагностика болезни Паркинсона может быть сведена к задаче классификации или регрессии, где целью является предсказание бинарной или непрерывной переменной на основе входных данных.

Алгоритмы классификации обычно используются в диагностике болезни Паркинсона для определения наличия или отсутствия заболевания у пациентов. Эти алгоритмы берут исходные данные, такие как клинические симптомы, биомаркеры и результаты визуализации, и используют статистические методы для выявления закономерностей, которые отличают пациентов с болезнью Паркинсона от пациентов без болезни. Например, некоторые алгоритмы могут полагаться на присутствие определенных биомаркеров в крови или спинномозговой жидкости пациента, чтобы указать на наличие заболевания.

Регрессионные алгоритмы, с другой стороны, используются для прогнозирования тяжести болезни Паркинсона у пациентов. Эти алгоритмы используют входные данные для создания непрерывных числовых результатов, которые отражают прогрессирование болезни. Например, некоторые регрессионные алгоритмы могут предсказывать будущие двигательные симптомы пациента на основе его текущих симптомов и демографических факторов, таких как возраст и пол.

В обоих случаях алгоритмы машинного обучения используют большие массивы данных о пациентах для обучения моделей, которые могут делать точные прогнозы относительно диагностики болезни Паркинсона. Затем эти модели могут быть применены к новым данным пациентов для выявления пациентов с болезнью Паркинсона, оценки тяжести заболевания и отслеживания прогрессирования болезни во времени.

К примеру, в работе [8] пять пациентов и семь здоровых людей были использованы для распознавания болезни Паркинсона с помощью анализа голоса. Для выполнения этой задачи, голоса пациентов были записаны с помощью Isomax HeadSet E60P5L, каждый сеанс записи длился около 25 минут, в общей сложности авторы использовали 50 предварительно записанных подсказок, состоящих из эмоциональных предложений, произнесенных профессиональной актрисой.

В работе [9] оценили различные алгоритмы, основанные на показателях дисфонии и направленные на распознавание БП. Первоначально, для дальнейшего отбора признаков, использовалось 132 акустических признака, и авторы пришли к выводу, что информация о дисфонии и существующие признаки в конечном итоге помогают в распознавание БП.

В работе [10] утверждают, что симптомы БП можно обнаружить за пять лет до постановки клинического диагноза, а симптомы, проявляющиеся в речи, включают снижение громкости, усиление вокального тремора и отдышки. В своей работе авторы использовали набор данных, который включает в себя 263 фонации от 43 испытуемых (17 женщин и 26 мужчин, из которых 10 были здоровыми людьми, а 33 - с диагнозом БП). В результатах работы авторы утверждают, что возможно диагностировать БП до появления каких-либо заметных клинических симптомов.

В целом, машинное обучение открывает большие перспективы для повышения точности и эффективности диагностики болезни Паркинсона, что может привести к улучшению результатов лечения и, в конечном итоге, к повышению качества жизни пациентов с этим изнурительным расстройством.

2. ОСНОВНАЯ ЧАСТЬ

2.1 Исследование набора данных

Набор данных HandPD был собран на медицинском факультете Университета штата Сан-Паулу в Бразилии. Он содержит изображения, полученные из письменных испытаний 92 человек. Эти люди были разделены на две группы: контрольную группу, состоящую из 18 экзаменов здоровых людей (6 мужчин и 12 женщин), и группу пациентов, в которую входят 74 экзамена людей, страдающих болезнью Паркинсона (59 мужчин и 15 женщин). Таким образом, 80,44% набора данных составляют пациенты, а 19,56% - контрольные лица. Хотя набор данных несбалансирован, можно достичь схожих пропорций, добавляя больше контрольных лиц, чем пациентов.

Контрольная группа состоит из 16 правшей и 2 левшей, средний возраст составляет $44,22 \pm 16,53$ лет. В группе пациентов есть 69 правшей и 5 левшей, средний возраст составляет $58,75 \pm 7,51$ лет. Таким образом, можно отметить, что набор данных не содержит возрастных искажений. Фактически, большинство пациентов значительно старше 60 лет, поскольку болезнь Паркинсона обычно прогрессирует в этой возрастной группе. Однако набор данных достаточно разнообразен и включает в себя даже 38-летнего пациента мужского пола.

Для создания набора данных каждому участнику предлагалось заполнить форму, выполняя определенные задания, такие как рисование кругов, спиралей и меандров. Однако анализ изображений был сосредоточен только на заданиях, где необходимо было обвести спирали и меандры.

После заполнения формы рисунки отцифровывались для дальнейшего извлечения характеристик спиралей и меандров. Этот шаг выполнялся вручную, при этом каждый рисунок обрезался до своего минимального ограничивающего прямоугольника (или близкого к нему). Затем обрезанные

изображения спиралей и меандров нумеровались следующим образом: от 1 до 4 для спиралей слева направо и от 5 до 8 для меандров слева направо.

Примеры пустого бланка и бланка, заполненного пациентом с болезнью Паркинсона представлены на рисунках 2.1 – 2.2.

No: ____ -- ____ -- ____ -- ____ -- ____ -- ____ -- ____ -- RG: _____		Field study: Unesp 2010 University of Applied Sciences Regensburg Biometric Smart Pen Project Universidade Estadual Paulista Faculdade de Medicina (FMB), Botucatu
Idade: _____ Mão dominante: () direita () esquerda		
a, b	Desenhando círculo 12 vezes no mesmo lugar sem parar.	Desenhando círculo no ar 12 vezes no mesmo lugar sem parar.
c	Desenhando espiral após sinal sonoro, de dentro para fora.	
d	Desenhando meander após sinal sonoro, de dentro para fora.	
e	Diadococinesia: Mão direita 20 segundos.	
f	Diadococinesia: Mão esquerda 20 segundos.	

Рисунок 2.1 – Пустой бланк с шаблонами

22/11/2010

No: _____

RG: _____

Distonia e Hiperkinesia Medicamentosa
 (semu medicamento às 14h, após 15:30 Botucatu)

Idade: 56 Mão dominante: ☒ direita () esquerda

Field study: Unesp 2010
 University of Applied Sciences
 Regensburg
 Biometric Smart Pen Project
 Universidade Estadual Paulista
 Faculdade de Medicina (FMB),
 Jorja e 1530 Botucatu






<p>Desenhar círculo 12 vezes no mesmo lugar sem parar.</p> <p>a, b</p> 	<p>Desenhar círculo no ar 12 vezes no mesmo lugar sem parar.</p>
<p>Desenhar espiral após sinal sonoro, de dentro para fora.</p> <p>c</p> 	
<p>Desenhar meander após sinal sonoro, de dentro para fora.</p> <p>d</p> 	
<p>Diadococinesia: Mão direita 20 segundos.</p> <p>e</p> 	
<p>Diadococinesia: Mão esquerda 20 segundos.</p> <p>f</p> 	

Рисунок 2.2 - Бланк, заполненный пациентом с болезнью Паркинсона

Процесс сравнения рукописных следов и экзаменационных шаблонов включает извлечение особенностей из обоих изображений и оценку степени различия между ними. Процесс извлечения включал в себя различные классические методы обработки изображений, к примеру, фильтры размытия и математическая морфология, с процессом выделения контуров. На рисунках 2.3 – 2.4 приведены результаты работы извлечения рукописных следов и экзаменационных шаблонов для спиралей и меандров [11].

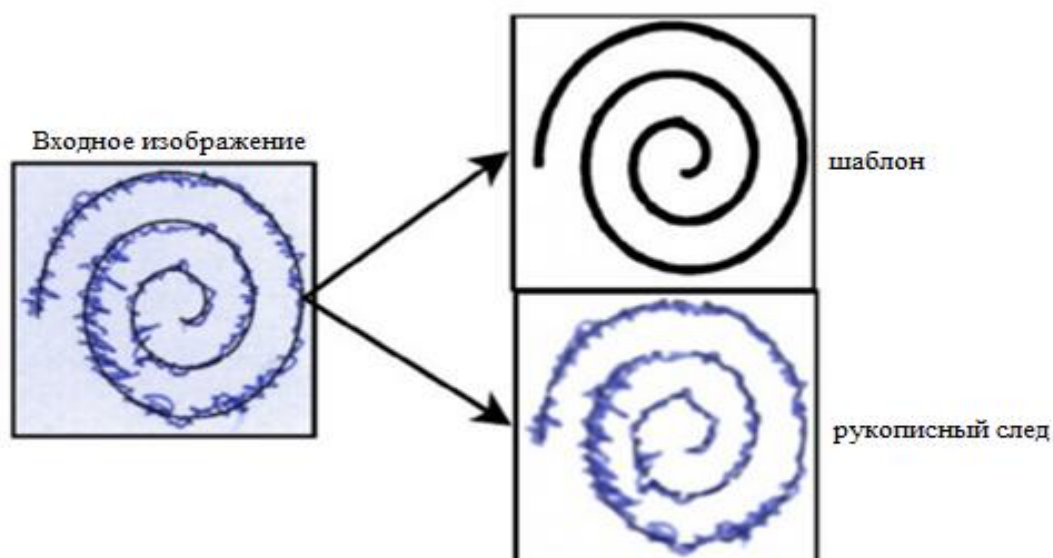


Рисунок 2.3 - Результат работы извлечения рукописных следов и экзаменационных шаблонов для спиралей

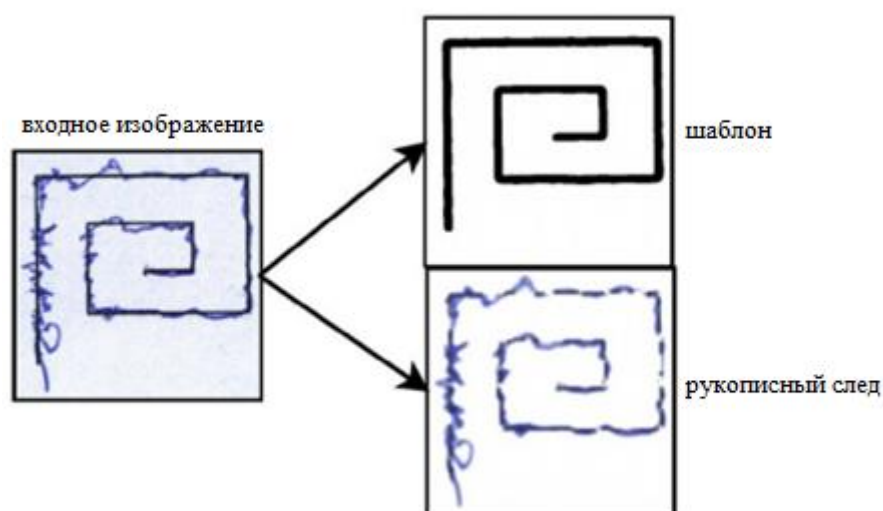


Рисунок 2.4 - Результат работы извлечения рукописных следов и экзаменационных шаблонов для спиралей

Для достижения краткого и компактного представления изображений используется алгоритм прореживания Чжан-Суэна для извлечения “скелета” пороговых изображений, который состоит из двух параллельных процедур [11]:

1. удаляются юго-восточные граничные точки и северо-западные угловые точки;

2. удаляются северо-западные граничные точки и юго-восточные угловые точки.

Результат прореживания шаблонов спирали и меандра, а также рукописного следа, представлены на рисунках 2.5 – 2.6.

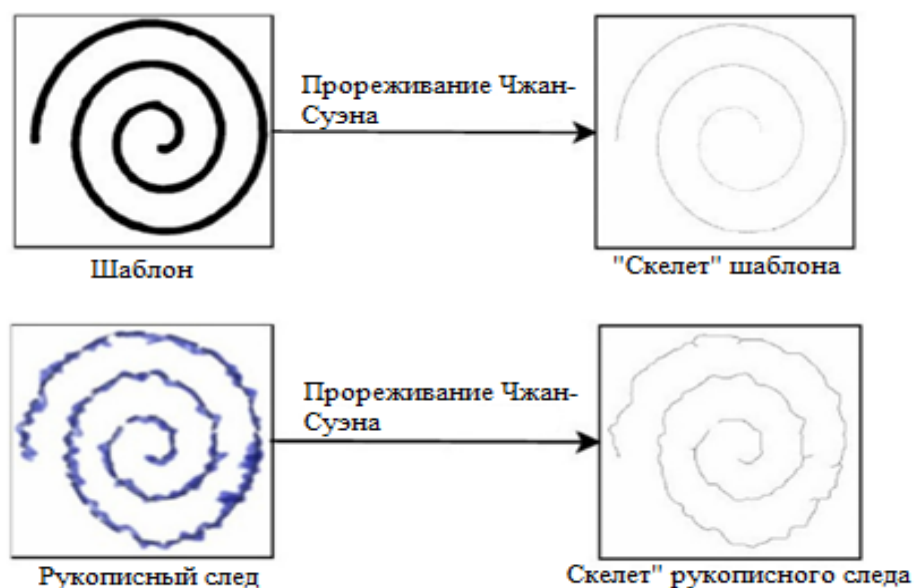


Рисунок 2.5 - Пример работы алгоритма прореживания Чжан-Суэна на спиралях

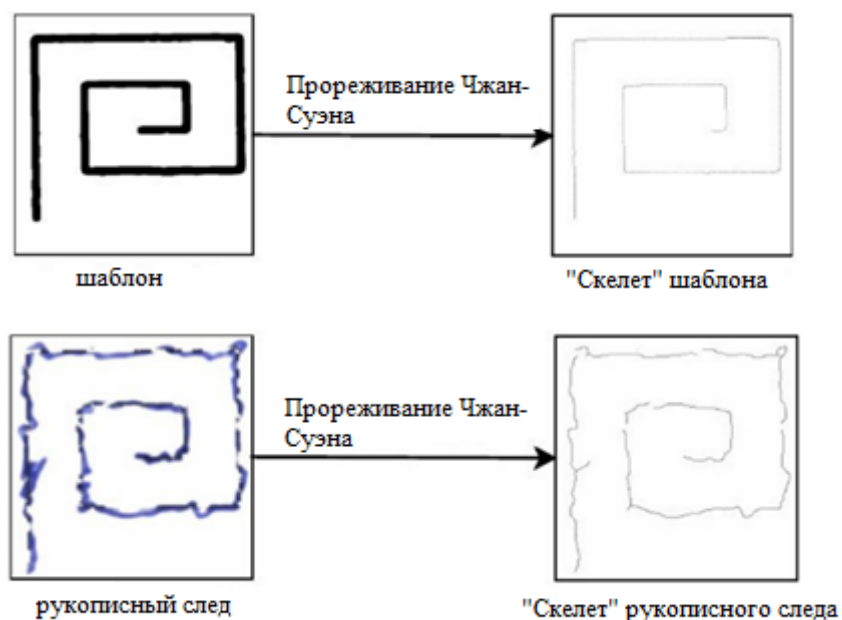


Рисунок 2.6 - Пример работы алгоритма прореживания Чжан-Суэна на меандрах

Однако изображения все еще могут содержать небольшие разрывы, поэтому необходим тщательный выбор точек выборки. Для этого

необходимо провести лучи от центра спирали/меандра к границам изображения и зафиксировать их пересечения с шаблоном и рукописным следом. Точки в областях с разрывами отбрасываются для обеспечения справедливой выборки, а полученные точки используются для извлечения девяти числовых характеристик из каждого скелета. Перед описанием этих характеристик вводится понятие "радиус" точки спирали или меандра, который представляет собой длину прямой линии, соединяющей точку с центром спирали/меандра. Центр представлен красной точкой, а скелет соединен со случайными белыми точками стрелками с прямыми линиями. Пример представлен на рисунке 2.7.

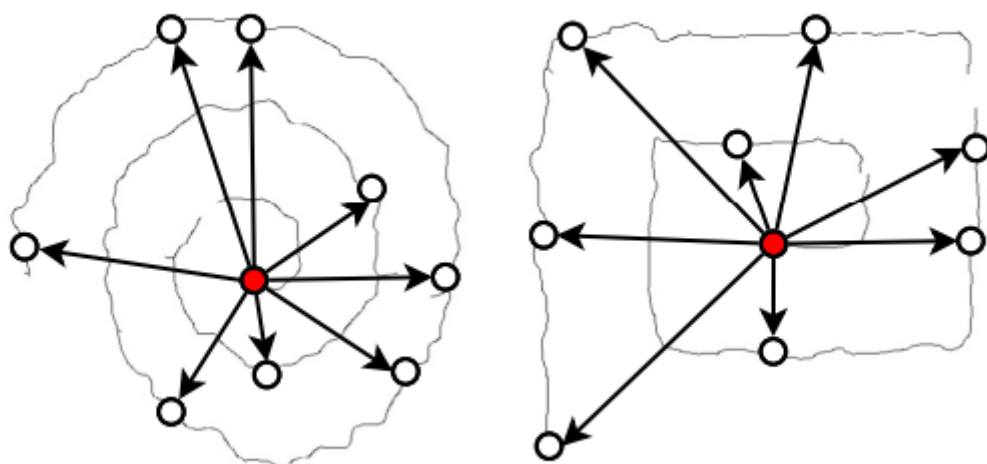


Рисунок 2.7 - Некоторые случайные точки и прямые линии

Ниже приведено описание 9 числовых признаков извлеченных из пар спиралей и меандров [11].

Признак a_0 – Среднеквадратичное отклонение (СКО) разницы между рукописный следом (НТ) и радиусом экзаменационного шаблона (ЕТ):

$$a_0 = \sqrt{\frac{1}{n} \sum_{i=0}^n (r_{HT}^i - r_{ST}^i)^2}$$

где n – количество случайно выбранных точек на линии рисунка,

i – точка на рукописной и шаблонной линиях,

r_{HT}^i – расстояние (радиус) от центра рисунка и до точки i на рукописном рисунке,

r_{ST}^i – расстояние от точки i на шаблонной линии до центра рисунка.

Признаки a_1 и a_2 – максимальное и минимальное различие между радиусами r_{HT}^i и r_{ST}^i .

$$a_1 = \max(\{|r_{HT}^1 - r_{ST}^1|, \dots, |r_{HT}^n - r_{ST}^n|\}), a_2 = \min(\{|r_{HT}^1 - r_{ST}^1|, \dots, |r_{HT}^n - r_{ST}^n|\})$$

Признак a_3 – стандартное отклонение различий между радиусами r_{HT}^i и r_{ST}^i .

Признак a_4 – Средний относительный тремор (MRT), определяемый как средняя разница между радиусом данного образца и его d ближайших соседей слева.

$$a_4 = \frac{1}{n-d} \sum_{i=d}^n |r_{HT}^i - r_{HT}^{i-d+1}|$$

Признаки a_5 и a_6 – максимальное и минимальное значение радиуса НТ, a_7 – стандартное отклонение значений НТ.

Признак a_8 – количество раз, когда разница между НТ и ЕТ радиусом меняется с отрицательного на положительный, или наоборот.

Полученный в результате извлечения признаков набор данных включает в себя 368 признаков спиралей и меандров, которые представлены вещественными числами, где 296 относятся к группе пациентов, а 72 к контрольной группе. Таким образом, коэффициент дисбаланса определяется как отношение большего класса к меньшему ($296 / 72$) и равняется 4.11. Так же в наборе содержится целевая переменная, которая указывает к какой группе относится испытуемый, а именно к группе пациентов или к контрольной группе.

На рисунках 2.8 – 2.9 приведены примеры признаков спиралей и меандров хранящихся в наборе данных.

	a0	a1	a2	a3	a4	a5	a6	a7	a8	Class
70	4868.249023	7934.421875	36170.38672	0.129718	22.837444	186.092102	0.019654	1676.030762	0.182390	0
71	4698.312500	7376.146973	39721.41797	0.026583	24.054586	177.467529	0.030080	1496.090088	0.211726	0
72	3946.162598	6825.418457	33951.33594	0.008171	20.519480	176.385178	0.011262	1398.989990	0.233227	1
73	4364.036133	6585.653809	32985.66797	0.002812	23.647997	167.954895	0.002320	1543.166626	0.255591	1
74	5156.546875	7618.953613	33945.66406	0.011237	25.074373	169.448547	0.006654	1710.564697	0.218954	1

Рисунок 2.8 – Признаки, относящиеся к спиральм из набора данных HandPD.

	a9	a10	a11	a12	a13	a14	a15	a16	a17	Class
70	3611.760010	8655.983398	42686.10156	0.000000	20.015533	208.805435	0.000000	551.243042	0.125000	0
71	2881.450195	5306.169434	30050.64648	0.000000	18.925398	182.359207	0.000000	656.486755	0.122581	0
72	4648.249512	6156.082031	32504.22266	0.026440	19.206491	191.923462	0.000000	678.250366	0.207143	1
73	3870.988525	5147.582520	35888.95703	0.011366	18.129059	180.435669	0.070075	576.030029	0.193309	1
74	5944.031738	8428.365234	69755.87500	0.972488	21.550133	192.816681	0.120103	944.668396	0.172662	1

Рисунок 2.9 – Признаки, относящиеся к меандрам из набора данных HandPD.

2.2 Методы и алгоритмы

2.2.1 Методы нормализации данных

Нормализация данных – важнейший этап предварительной обработки в машинном обучении и анализе данных. Она включает в себя преобразование данных в стандартный масштаб, диапазон и распределение, чтобы устранить любые смещения, которые могут существовать в данных. Основная причина нормализации заключается в приведении характеристик данных к одному масштабу, чтобы ни одна характеристика не доминировала над другими в процессе обучения модели. Нормализация также повышает эффективность и точность алгоритмов машинного обучения и гарантирует, что алгоритмы сходятся быстрее и надежнее [12].

Существует несколько методов нормализации данных, которые обычно используются, включая нормализацию по методу Min-Max, нормализацию по Z-score [13].

2.2.1.1 Min-Max нормализация

Нормализация Min-Max — это простой и эффективный метод, который преобразует значения признака в диапазон от 0 до 1. Этот метод особенно полезен для алгоритмов, чувствительных к масштабу входных данных, таких как нейронные сети и машины опорных векторов [14].

Процесс нормализации включает два этапа: нахождение минимального и максимального значений признака, а затем применение линейного преобразования для переноса исходных значений в нужный диапазон. Формула для нормализации по минимуму и максимуму выглядит следующим образом:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

где x — исходное значение признака,

x_{min} — минимальное значение признака,

x_{max} — максимальное значение признака,

x_{norm} — нормализованное значение признака.

Одно из преимуществ Min-Max нормализации заключается в том, что она сохраняет форму исходного распределения данных. Это означает, что связи между значениями признака сохраняются, что может быть важно для некоторых алгоритмов машинного обучения, которые полагаются на эти связи. Однако она чувствительна к выбросам, поскольку они могут существенно повлиять на диапазон признака.

2.2.1.2 Z-score нормализация

Нормализация Z-score, также известная как стандартизация, является распространенной техникой, используемой в машинном обучении для масштабирования числовых данных. Цель этой техники - преобразовать данные таким образом, чтобы их среднее значение было равно нулю, а стандартное отклонение - единице. Это достигается путем вычитания среднего значения признака из каждой точки данных, а затем деления

полученного результата на стандартное отклонение. Полученные значения представляют собой число стандартных отклонений, на которое точка данных удалена от среднего значения [15].

Процесс нормализации Z-score включает в себя несколько этапов. Сначала рассчитывается среднее значение признака путем суммирования всех точек данных и деления на общее количество точек данных. Затем рассчитывается стандартное отклонение путем взятия квадратного корня из суммы квадратов разницы между каждой точкой данных и средним значением, деленной на общее количество точек данных минус один. После расчета среднего и стандартного отклонения каждая точка данных преобразуется путем вычитания среднего и деления на стандартное отклонение. Формула для нормализации Z-score выглядит следующим образом:

$$x_{norm} = \frac{x - \mu}{\sigma},$$

где x – исходное значение,

μ – среднее значение признака,

σ – стандартное отклонение признака,

x_{norm} – стандартизированное значение.

Нормализация Z-score имеет ряд преимуществ. Одно из главных преимуществ заключается в том, что она позволяет сравнивать признаки по единой шкале. Это особенно полезно при работе с признаками, имеющими различные единицы измерения или шкалы, поскольку это гарантирует, что все признаки будут рассматриваться одинаково. Кроме того, нормализация Z-score повышает точность моделей за счет уменьшения влияния выбросов и экстремальных значений, которые могут исказить результаты. Это происходит потому, что процесс нормализации гарантирует, что данные имеют нормальное распределение, причем 68% данных находятся в пределах одного стандартного отклонения от среднего значения, а 95% - в пределах

двух стандартных отклонений. Это облегчает алгоритмам машинного обучения выявление закономерностей и составление точных прогнозов.

Еще одно преимущество нормализации Z-score заключается в том, что она улучшает интерпретируемость моделей. Это происходит потому, что нормализованные данные легче понять и интерпретировать, чем необработанные. Например, если функция имеет среднее значение 100 и стандартное отклонение 10, то точка данных со значением 110 будет иметь Z-score 1, что означает, что она на одно стандартное отклонение выше среднего. Это облегчает понимание значимости каждой точки данных и ее вклада в общий результат.

Нормализация данных — это необходимый этап подготовки данных для машинного обучения и анализа данных. Она гарантирует, что признаки данных имеют одинаковый масштаб и распределение, что позволяет избежать смещения при обучении модели и повысить эффективность и точность алгоритмов машинного обучения.

2.2.2 Методы отбора признаков

Отбор признаков — это важный этап машинного обучения, который заключается в определении наиболее релевантного и информативного подмножества признаков для использования при обучении модели. Этот процесс необходим, поскольку не все признаки могут быть полезны для прогнозирования конечной переменной, а использование нерелевантных или избыточных признаков может привести к переобучению, увеличению вычислительных затрат и снижению эффективности модели [16].

Существует несколько методов отбора признаков, включая методы фильтрации, обертывания и встраивания. Методы фильтрации оценивают релевантность признаков на основе их статистических свойств, таких как корреляция, взаимная информация или статистическая значимость. Методы "обертки" используют сам алгоритм обучения для оценки релевантности признаков путем рекурсивного выбора и оценки подмножеств признаков.

Встроенные методы включают отбор признаков в процесс обучения алгоритма обучения, где наиболее релевантные признаки отбираются во время обучения модели [16].

2.2.2.1 Метод отбора признаков chi2

Хи-квадрат (Chi2) — это статистический тест, обычно используемый в машинном обучении для определения взаимосвязи между двумя категориальными переменными. В частности, он измеряет разницу между наблюдаемой частотой определенного события и ожидаемой частотой этого события. Тест основан на принципе, что если между двумя переменными нет связи, то наблюдаемая частота должна быть близка к ожидаемой [17].

Тест хи-квадрат работает путем расчета ожидаемой частоты каждого события на основе нулевой гипотезы (т.е. предположения об отсутствии связи между двумя переменными). Это делается путем умножения предельных итоговых значений каждой переменной и деления на общий объем выборки. Затем наблюдаемая частота каждого события сравнивается с ожидаемой частотой с помощью формулы хи-квадрат:

$$X^2 = \sum \frac{(O-E)^2}{E},$$

где O - наблюдаемая частота,

E - ожидаемая частота.

Полученное значение X^2 сравнивается с критическим значением, основанным на степенях свободы. Степени свободы рассчитываются как:

$$df = (r - 1) * (c - 1),$$

где r – количество строк,

c – количество столбцов в таблице случайностей.

Если рассчитанное значение X^2 больше критического значения, то нулевая гипотеза отвергается и делается вывод о наличии значимой связи между двумя переменными.

2.2.2.2 Метод отбора признаков `mutual_info_classif`

Взаимная информация (Mutual information) — это мера зависимости между двумя переменными, которая используется в машинном обучении для отбора признаков и задач классификации. На выходе получается балл, который показывает релевантность каждого признака целевой переменной, где более высокие баллы указывают на большую релевантность [18].

Взаимная информация между двумя переменными X и Y определяется как уменьшение неопределенности в отношении X с учетом знания Y , или наоборот. Другими словами, она измеряет, сколько информации дает одна переменная о другой. Взаимная информация основана на концепции энтропии, которая является мерой количества неопределенности или случайности в переменной.

Функция `mutual_info_classif` работает путем вычисления взаимной информации между каждым признаком и целевой переменной, используя непараметрический метод, основанный на оценке энтропии. Сначала она дискретизирует непрерывные признаки на категории, а затем оценивает совместные и маргинальные распределения признаков и целевых переменных с помощью метода гистограмм. Затем рассчитывается взаимная информация с использованием совместного и маргинального распределений.

Функция `mutual_info_classif` полезна для отбора признаков, поскольку она позволяет выявить релевантные признаки, которые не обязательно линейно связаны с целевой переменной. Она также может обрабатывать нелинейные зависимости и взаимодействия между признаками, которые могут быть пропущены другими методами отбора признаков. Однако она может не подходить для высокоразмерных наборов данных или наборов данных с сильно коррелированными признаками, поскольку в этих случаях она требует больших вычислительных затрат и может давать нестабильные результаты.

2.2.2.3 Метод отбора признаков *f_classif*

f_classif — это статистический тест, используемый в машинном обучении для определения значимости связи между набором признаков и целевой переменной. Тест вычисляет отношение дисперсии между классами к дисперсии внутри классов. Это отношение известно как F-статистика, и оно измеряет, насколько хорошо средства различных классов отделены друг от друга. Чем выше F-статистика, тем больше разделение между классами и тем важнее признак для прогнозирования целевой переменной [19].

Для проведения теста *f_classif* данные сначала разбиваются на классы на основе целевой переменной. Затем для каждого признака рассчитывается дисперсия между классами и дисперсия внутри классов. Эти значения используются для расчета F-статистики для каждого признака. F-статистика сравнивается с таблицей распределения для определения вероятности случайного получения данного значения. Если вероятность ниже заданного уровня значимости, нулевая гипотеза отвергается, и делается вывод о значимости признака для прогнозирования целевой переменной.

В процессе разработки программы были проведены тесты с представленными методами отбора признаков. Ниже в таблицах 2.1 – 2.3 указана лучшая точность для каждого из классификаторов с использованием определенного метода отбора признаков.

Обозначения строк:

- *Classifier* – название классификатора;
- *Accuracy* – средняя точность, которая была получена для классификатора за 10 запусков.

Таблица 2.1 – Метод *mutual_info_classif* с 9 параметрами

<i>Classifier</i>	<i>Accuracy, %</i>
LogisticRegression	71,93
DecisionTreeClassifier	63,7
MLPClassifier	72,55

Продолжение таблицы 2.1

MultinomialNB	67,65
SVC	73,74

Таблица 2.2 – Метод chi2 с 14 параметрами

<i>Classifier</i>	<i>Accuracy, %</i>
LogisticRegression	72,55
DecisionTreeClassifier	64,63
MLPClassifier	67,3
MultinomialNB	72,16
SVC	73,16

Таблица 2.3 – Метод f_classif с 15 параметрами

<i>Classifier</i>	<i>Accuracy, %</i>
LogisticRegression	72,35
DecisionTreeClassifier	60,06
MLPClassifier	71,83
MultinomialNB	68,05
SVC	71,21

По результатам тестов в качестве основного метода фильтрации был выбран хи-квадрат, так как именно с его помощью удалось добиться максимальной точности для большего количества классификаторов.

В целом, выбор признаков — это важный этап машинного обучения, целью которого является определение и отбор наиболее значимых признаков для алгоритма обучения. Методы фильтрации, такие как chi2, mutual_info_classif и f_classif, являются популярными подходами для отбора признаков, которые оценивают релевантность признаков на основе их статистических свойств.

2.2.3 Методы кросс-валидации

Кросс-валидация — это важный метод машинного обучения, который позволяет оценить производительность и обобщающую способность прогностической модели. По сути, она предполагает разбиение набора

данных на несколько уникальных подмножеств и использование каждого из них поочередно в качестве обучающего и тестового множества, что позволяет оценить эффективность модели на невидимых данных [20].

Кросс-валидация необходима в машинном обучении для предотвращения переобучения, которое происходит, когда модель становится слишком специализированной для обучающих данных и не может хорошо обобщать новые данные. Оценивая модель на различных подмножествах данных, кросс-валидация дает более точную оценку эффективности модели на невидимых данных, помогая выявить и устранить перебор.

Одним из распространенных примеров кросс-валидации является k -кратная кросс-валидация, при которой набор данных разбивается на k непересекающихся подмножеств, или фолдов, примерно одинакового размера. Модель обучается на $k-1$ фолдов и тестируется на оставшихся фолдах, и этот процесс повторяется k раз, причем каждый фолд используется в качестве тестового набора один раз. Затем производительность модели усредняется за k итераций, чтобы получить более точную оценку ее производительности [20].

В целом, кросс-валидация — это необходимый метод машинного обучения, позволяющий оценить производительность и обобщающую способность прогностической модели. Она помогает предотвратить перебор и дает более точную оценку производительности модели на невидимых данных.

2.2.4 Алгоритмы классификации

Алгоритмы классификации представляют собой компьютерные методы и процедуры, разработанные для решения задач классификации, которые включают определение принадлежности объектов к определенным категориям или классам. Классификация является фундаментальной задачей в области машинного обучения и анализа данных, и ее цель заключается в

разработке моделей, способных автоматически классифицировать новые, ранее неизвестные объекты на основе их признаков.

Алгоритмы классификации обычно основаны на математических и статистических методах, которые обрабатывают входные данные и строят модель, способную выявить закономерности и обобщить их для классификации новых объектов. Эти алгоритмы могут использовать различные подходы, такие как методы байесовской классификации, линейные модели, деревья принятия решений, метод опорных векторов, искусственные нейронные сети и многие другие.

В процессе работы алгоритмы классификации обучаются на основе размеченных обучающих данных, где каждый объект имеет известную метку класса. Они применяют различные методы для обнаружения и выявления важных признаков, которые могут быть характерными для каждого класса. Затем модель может использоваться для классификации новых, неизвестных объектов, присваивая им соответствующие метки классов на основе изученных закономерностей.

2.2.4.1 Логистическая регрессия (LogisticRegression)

Логистическая регрессия — это алгоритм контролируемого машинного обучения, используемый для задач двоичной классификации, где целью является предсказание вероятности принадлежности экземпляра к определенному классу (обычно 0 или 1). Алгоритм представляет собой тип обобщенной линейной модели, которая использует логистическую функцию для моделирования связи между входными признаками и выходной переменной [21].

Алгоритм логистической регрессии направлен на поиск наилучшего набора параметров (весов и смещений), которые минимизируют ошибку между прогнозируемым выходом и истинным выходом. Результатом логистической регрессии является значение вероятности между 0 и 1,

которое представляет собой вероятность принадлежности экземпляра к определенному классу.

Логистическая функция, используемая в логистической регрессии, — это сигмоидная функция, которая определяется как:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

где x - входные данные функции.

Сигмоидная функция отображает любое входное значение на значение от 0 до 1.

В логистической регрессии входные признаки умножаются на соответствующие веса, суммируются с членом смещения и пропускаются через сигмоидную функцию. Математически результат логистической регрессии можно представить в виде:

$$y = \sigma(w \cdot x + b),$$

где x - вектор входных признаков,

w - вектор весов,

b - член смещения,

σ - сигмоидная функция.

Цель логистической регрессии - найти оптимальные значения весового вектора w и члена смещения b , которые минимизируют ошибку между прогнозируемым выходом и истинным выходом. Это достигается путем определения функции потерь, которая измеряет разницу между прогнозируемым и истинным выходом. Наиболее часто используемой функцией потерь для логистической регрессии является бинарная потеря перекрестной энтропии, которая определяется как:

$$L(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

где y - истинная метка (0 или 1),

\hat{y} - предсказанное значение вероятности.

Двоичная потеря перекрестной энтропии сильнее наказывает модель за неправильные предсказания, которые находятся дальше от истинной метки.

Чтобы найти оптимальные значения w и b , логистическая регрессия использует алгоритм оптимизации, такой как градиентный спуск. Алгоритм итеративно обновляет значения w и b в направлении наиболее крутого спуска функции потерь.

Для оценки эффективности модели логистической регрессии можно использовать различные метрики, такие как точность, полнота и F1 score. Кривая рабочей характеристики приемника (ROC) — это графическое представление производительности модели логистической регрессии при различных пороговых значениях. Площадь под ROC-кривой (AUC) является общепринятой метрикой для оценки общей эффективности модели. Пример ROC кривой приведен на рисунке 2.10.

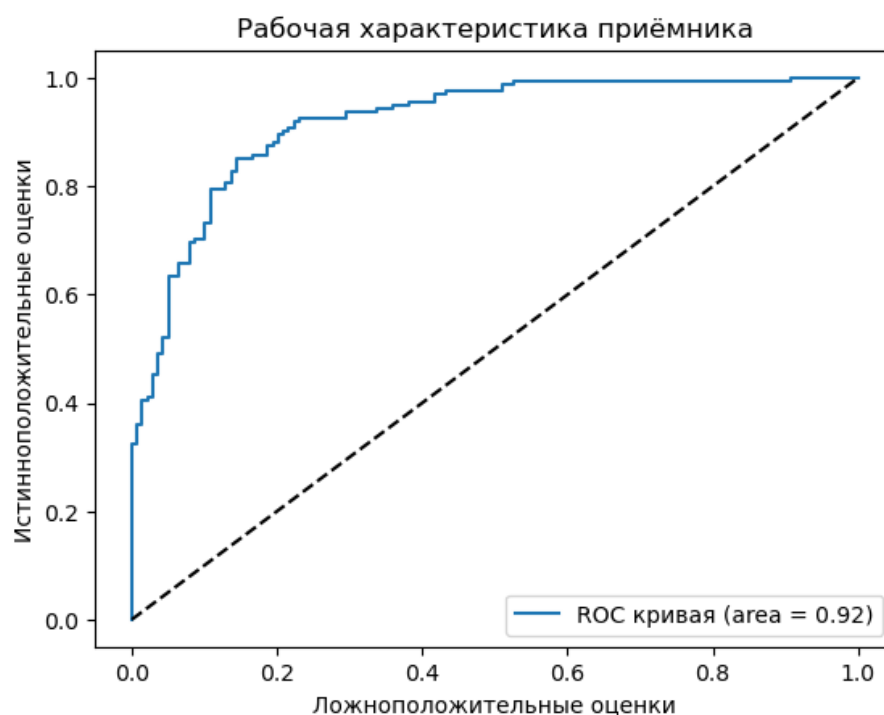


Рисунок 2.10 – Пример ROC кривой

В целом, логистическая регрессия — это широко используемый алгоритм для задач двоичной классификации в машинном обучении. Он использует сигмоидальную функцию для моделирования взаимосвязи между входными признаками и показателем.

2.2.4.2 Дерево решений (DecisionTreeClassifier)

Дерево решений — это алгоритм контролируемого обучения, используемый для решения задач классификации в машинном обучении. Алгоритм создает дерево решений путем рекурсивного разбиения набора данных на основе наиболее информативного признака, в результате чего получается набор узлов и ребер, которые могут быть использованы для классификации новых экземпляров. Пример дерева решений, построенного на наборе данных HandPD представлен на рисунке 2.11 [22].

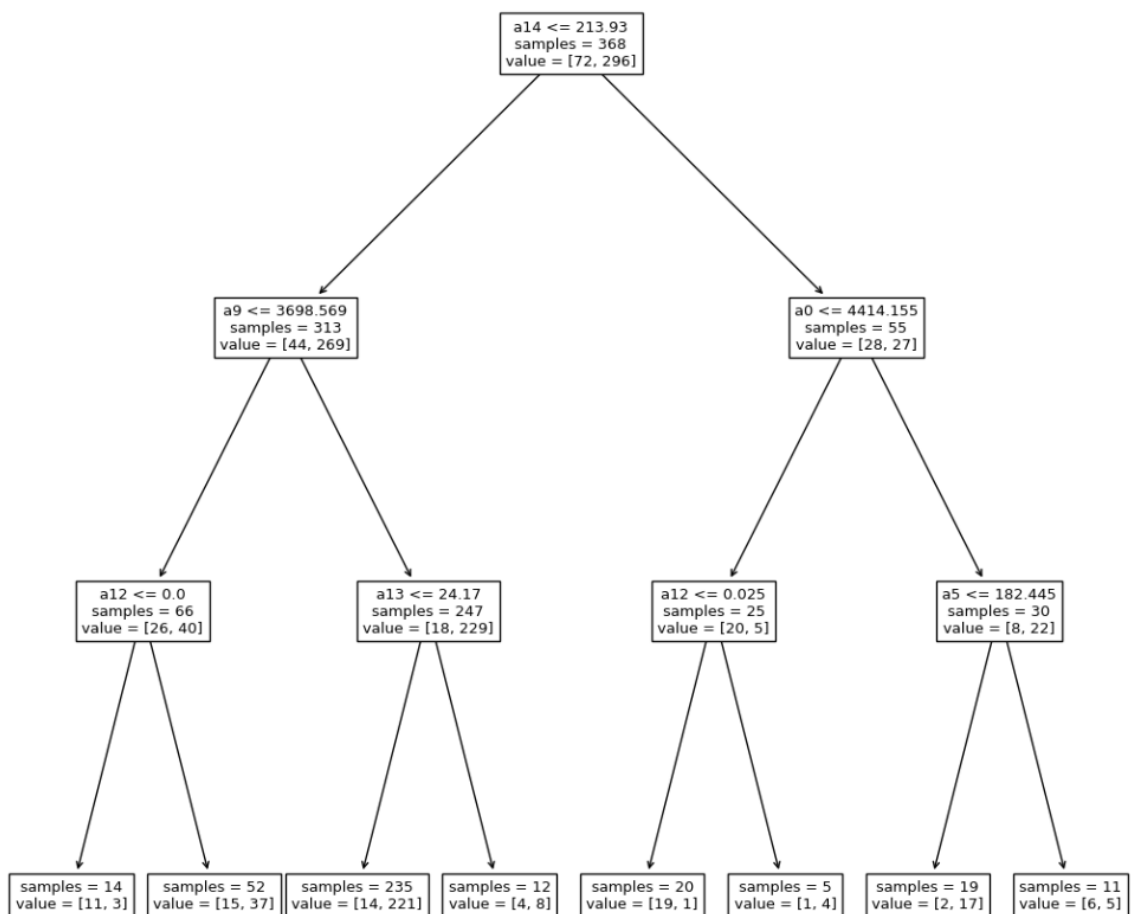


Рисунок 2.11 - Пример дерева решений

Дерево решений состоит из внутренних узлов, представляющих признак, и ребер, соединяющих их с дочерними узлами, представляющими возможные результаты использования признака. В каждом внутреннем узле алгоритм выбирает признак, обеспечивающий максимальный прирост

информации, который рассчитывается путем измерения уменьшения энтропии или коэффициента Джини в наборе данных.

Энтропия — это мера примеси или случайности в наборе данных. Цель алгоритма — минимизировать энтропию путем выбора наиболее информативного признака в каждом узле. Примесь Джини — это еще одна мера примеси, которая измеряет вероятность неправильной классификации случайного экземпляра из набора данных.

Дерево решений строится путем рекурсивного разбиения набора данных на подмножества на основе выбранного признака до тех пор, пока все экземпляры не будут принадлежать к одному классу или пока не останется больше признаков для разбиения. Этот процесс создает древовидную структуру, где каждый листовой узел представляет собой метку класса.

Чтобы избежать перегрузки, дерево решений может быть обрезано путем удаления узлов, которые не улучшают точность дерева. Обрезка может быть выполнена с помощью различных методов, таких как минимальное разбиение выборки, максимальная глубина и минимальные выборки листьев.

В целом, дерево решений — это мощный алгоритм классификации, который создает дерево решений путем рекурсивного разбиения набора данных на основе наиболее информативного признака. Алгоритм минимизирует энтропию или примесь Джини в каждом узле, чтобы выбрать лучший признак для разбиения. Дерево решений можно подрезать, чтобы избежать перебора, а полученное дерево можно визуализировать в виде графа, чтобы получить представление о процессе принятия решений алгоритмом.

2.2.4.3 Нейронные сети многослойных перцептронов (MLPClassifier)

MLPClassifier — это алгоритм классификации, широко используемый в области искусственного интеллекта и машинного обучения, основанный на архитектуре многослойного перцептрона (MLP). Это алгоритм контролируемого обучения, который можно использовать для

классификации входных данных по различным классам на основе маркированных обучающих данных [23].

MLPClassifier состоит из нескольких слоев нейронов, причем каждый нейрон в данном слое соединен со всеми нейронами в предыдущем и следующем слоях. Первый слой — это входной слой, который получает входные данные. Последний слой — выходной слой, который производит прогнозируемый результат. Слои между ними называются скрытыми слоями, они выполняют вычисления для преобразования входных данных в формат, который может быть использован выходным слоем для классификации. Пример скрытого слоя MLP приведен на рисунке 2.12.

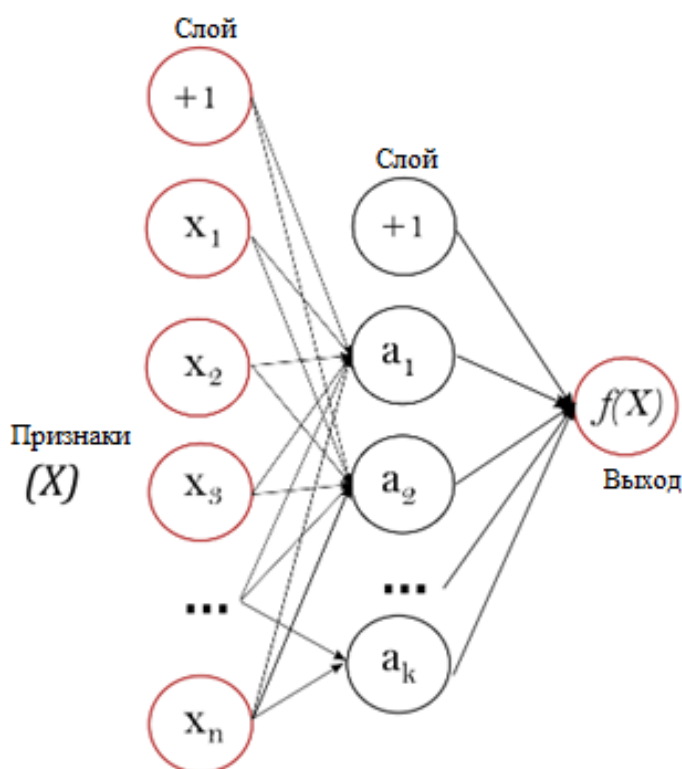


Рисунок 2.12 - Пример скрытого слоя MLP

Во время обучения MLPClassifier регулирует веса и смещения нейронов в каждом слое, чтобы минимизировать ошибку между прогнозируемым и фактическим выходом. Для этого используется алгоритм оптимизации, такой как стохастический градиентный спуск (SGD), который обновляет веса и смещения на основе градиента функции ошибки по отношению к весам и смещениям.

Одним из важных понятий в MLPClassifier является функция активации, которая используется для определения выхода нейрона на основе его входа. Наиболее часто используемой функцией активации является сигмоидная функция, которая отображает любое вещественное число в значение от 0 до 1. Другие популярные функции активации включают функцию ReLU (rectified linear unit), которая возвращает входное значение, если оно положительное, и 0 в противном случае, и функцию softmax, которая используется для много классовой классификации и возвращает распределение вероятности по классам на выходе.

В целом, MLPClassifier — это мощный алгоритм классификации, который можно использовать для широкого спектра приложений в искусственном интеллекте и машинном обучении.

2.2.4.4 Мультиномиальный Наивный Байес (MultinomialNB)

Алгоритм классификации мультиномиальный наивный Байес (MultinomialNB) — алгоритм в области искусственного интеллекта и машинного обучения, который часто используется для задач классификации текста, таких как фильтрация спама, анализ настроений и классификация тем. Алгоритм основан на теореме Байеса и предполагает, что признаки независимы друг от друга, учитывая класс. MultinomialNB расширяет это предположение для работы с несколькими классами и небинарными признаками [24].

Алгоритм MultinomialNB работает путем построения модели распределения вероятностей различных классов в обучающих данных. Это делается путем оценки предварительной вероятности каждого класса, а также условной вероятности каждого признака с учетом каждого класса. Эти вероятности рассчитываются на основе обучающих данных, а затем используются для прогнозирования новых данных.

Для классификации новых входных данных алгоритм MultinomialNB рассчитывает вероятность каждого класса с учетом признаков входных

данных, используя теорему Байеса. Затем класс с наибольшей вероятностью выбирается в качестве прогнозируемого класса для входных данных.

Одним из важных понятий в алгоритме MultinomialNB является сглаживание, которое используется для обработки случая, когда признак не встречается в обучающих данных для определенного класса. Это может привести к нулевым оценкам вероятности, что может вызвать проблемы при классификации. Для решения этой проблемы алгоритм использует параметр сглаживания, например, сглаживание по Лапласу, чтобы добавить небольшое количество вероятности к каждому признаку для каждого класса.

В целом, алгоритм MultinomialNB — это простой и эффективный алгоритм классификации, который можно использовать для широкого спектра задач классификации текстов в искусственном интеллекте и машинном обучении.

2.2.4.5 Метод опорных векторов (SVM)

Метод опорных векторов Support Vector Machine (SVM) и его разновидность, Support Vector Classifier (SVC), является популярными алгоритмами в области искусственного интеллекта и машинного обучения, который может использоваться как для линейных, так и для нелинейных задач классификации. Алгоритм работает путем нахождения гиперплоскости в пространстве признаков, которая разделяет различные классы с максимальным запасом. График, иллюстрирующий работу алгоритма SVC приведен на рисунке 2.13 [25].

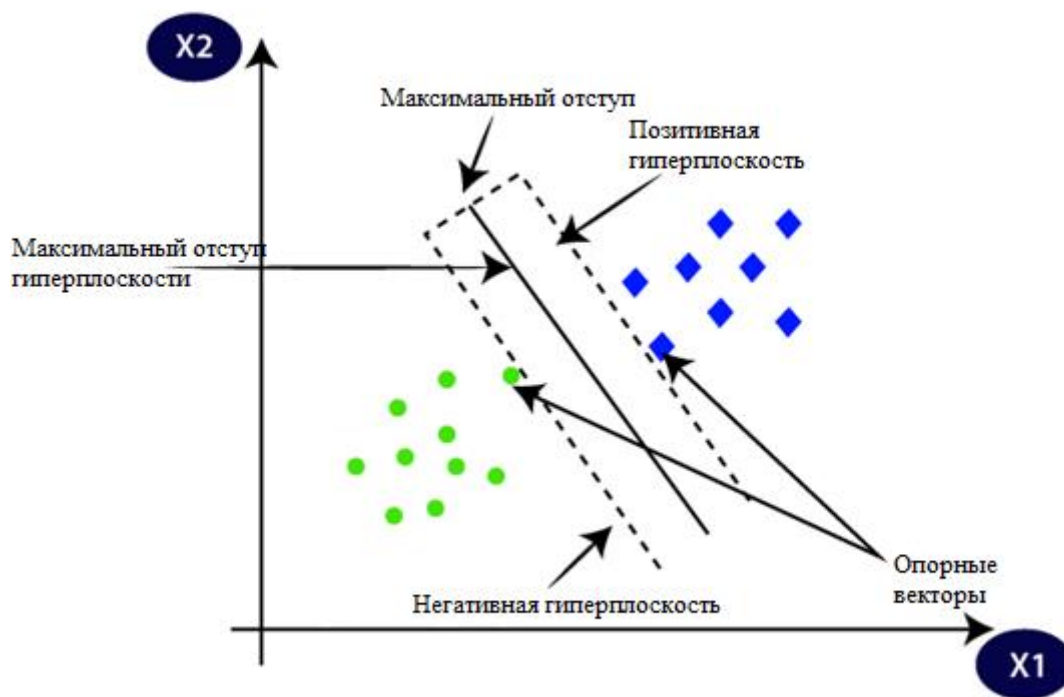


Рисунок 2.13 - График, иллюстрирующий работу алгоритма классификации SVC

Алгоритм SVC работает, сначала отображая входные данные в более высокоразмерное пространство признаков с помощью функции ядра, например, ядра радиальной базисной функции (RBF). В этом более высокоразмерном пространстве алгоритм затем находит гиперплоскость, которая максимально разделяет различные классы с максимальным запасом. Эта гиперплоскость определяется опорными векторами - точками данных, которые лежат ближе всего к гиперплоскости.

Во время обучения алгоритм SVC оптимизирует параметры гиперплоскости и функции ядра, чтобы минимизировать ошибку классификации на обучающих данных. Для этого используется алгоритм оптимизации, такой как градиентный спуск или последовательная минимальная оптимизация (SMO), который итеративно обновляет гиперплоскость и функцию ядра на основе градиента объективной функции.

Чтобы классифицировать новый входной сигнал, алгоритм SVC отображает входной сигнал в более высокоразмерное пространство признаков, используя ту же функцию ядра, а затем использует положение

входного сигнала относительно гиперплоскости для определения предсказанного класса.

В целом, алгоритм SVC — это мощный и универсальный алгоритм классификации, который можно использовать для широкого спектра задач классификации в искусственном интеллекте и машинном обучении.

2.2.5 Алгоритмы балансировки данных

Балансировка данных - важный шаг в машинном обучении, направленный на решение проблемы дисбаланса классов, когда количество образцов в каждом классе значительно отличается. В задаче классификации дисбаланс классов может привести к смещенной производительности модели, когда алгоритм склонен предсказывать большинство классов чаще, чем меньшинство. Такое поведение объясняется объективной функцией, оптимизируемой в процессе обучения модели, которая стремится минимизировать общий коэффициент ошибок.

Балансировка данных с помощью методов удаления или добавления экземпляров может помочь смягчить проблемы дисбаланса классов, создавая более репрезентативный обучающий набор. Under-sampling предполагает уменьшение количества образцов в классе большинства, а over-sampling предполагает дублирование или создание новых образцов в классе меньшинства. Оба метода направлены на создание более сбалансированного набора данных, что позволяет модели обучаться на разнообразном наборе примеров и достигать лучших показателей обобщения. В целом, балансировка данных является важным этапом машинного обучения, особенно для несбалансированных наборов данных, чтобы улучшить производительность модели и избежать необъективных прогнозов.

2.2.5.1 Добавление экземпляров (Over-sampling)

Добавление экземпляров (Over-sampling) — это популярная техника балансировки данных в машинном обучении, которая направлена на

устранение дисбаланса классов путем увеличения количества выборок в меньшинстве классов. Методы избыточной выборки генерируют новые синтетические выборки в классе меньшинства с помощью различных алгоритмов, чтобы сделать распределение классов более сбалансированным [26].

Одним из часто используемых алгоритмов добавления экземпляров является RandomOverSampler, который случайным образом дублирует существующие выборки в классе меньшинства, чтобы количество выборок в классе большинства совпадало. Этот подход может внести избыточность в данные, что приведет к чрезмерной подгонке, особенно когда класс меньшинства мал. На рисунке 2.14 приведен пример работы алгоритма RandomOverSampler [26].

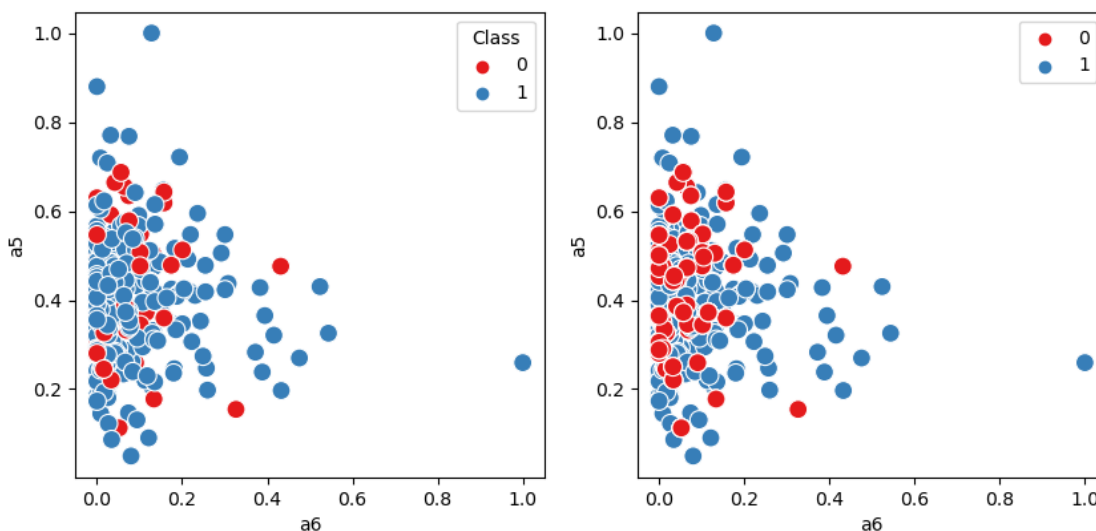


Рисунок 2.14 - Пример работы алгоритма RandomOverSampler

Для решения этой проблемы адаптивная синтетическая выборка (ADASYN) генерирует новые синтетические выборки в классе меньшинств на основе плотности распределения данных. ADASYN создает больше синтетических выборок в регионах, где плотность класса меньшинств низкая, что позволяет получить более разнообразный и репрезентативный набор данных. На рисунке 2.15 приведен пример работы алгоритма ADASYN [26].

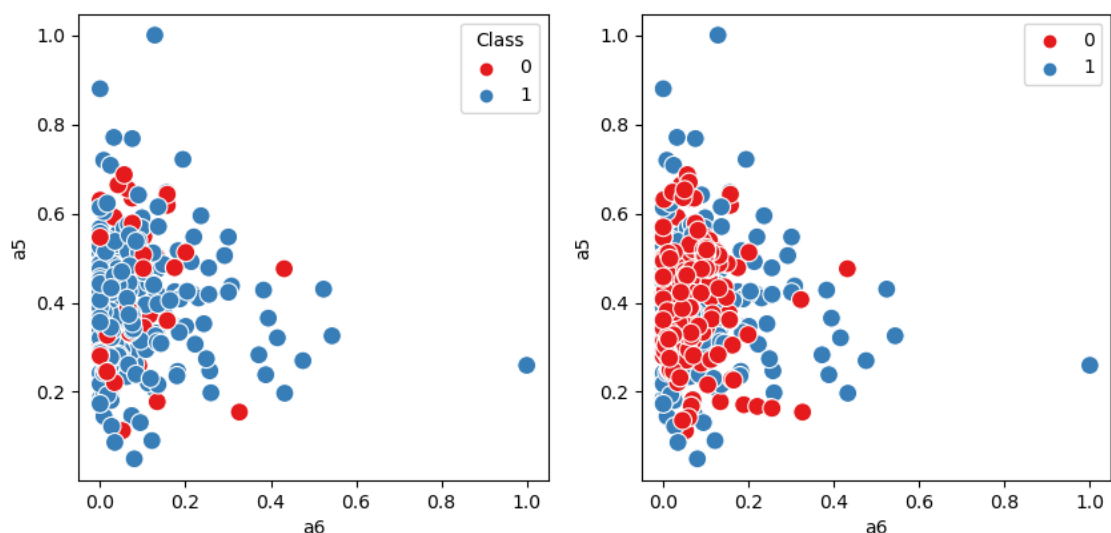


Рисунок 2.15 - Пример работы алгоритма ADASYN

Другим алгоритмом добавления экземпляров является Synthetic Minority Over-sampling Technique (SMOTE), который создает новые синтетические выборки путем интерполяции между существующими выборками в классе меньшинств. SMOTE выбирает пары похожих образцов в классе меньшинств и создает новые синтетические образцы вдоль соединяющего их отрезка линии. Этот подход создает более информативные синтетические выборки, избегая избыточности в данных. На рисунке 2.16 приведен пример работы алгоритма SMOTE [26].

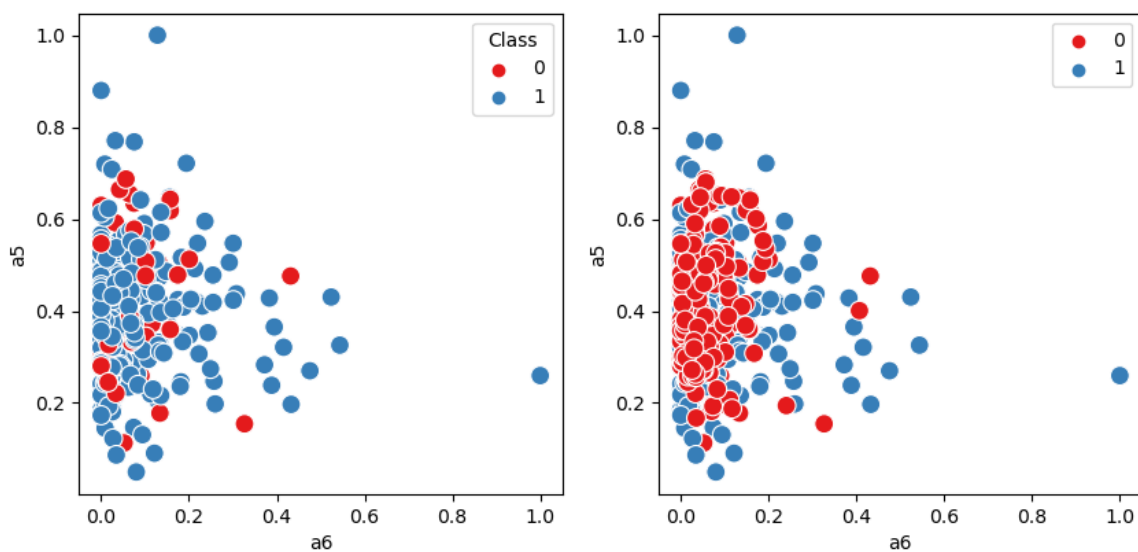


Рисунок 2.16 – Пример работы алгоритма SMOTE

В целом, методы добавления экземпляров, такие как RandomOverSampler, ADASYN и SMOTE, могут помочь решить проблемы дисбаланса классов в машинном обучении путем создания более представительных и сбалансированных наборов данных. Эти методы могут привести к улучшению производительности модели и более точным прогнозам для класса меньшинств.

2.2.5.2 Удаление экземпляров (Under-sampling)

Удаление экземпляров (Under-sampling) — это метод балансировки данных в машинном обучении, который уменьшает количество образцов в большинстве классов для решения проблем дисбаланса классов. Этот подход направлен на создание более сбалансированного набора данных путем удаления избыточных или нерелевантных образцов в большинстве классов, что позволяет модели сосредоточиться на меньшинстве классов [27].

RandomUnderSampler — это простой алгоритм удаления экземпляров, который случайным образом выбирает подмножество образцов из класса большинства, чтобы соответствовать количеству образцов в классе меньшинства. Этот подход может отбрасывать информативные выборки в классе большинства, что приводит к потере информации и потенциальной недоподгонке. На рисунке 2.17 приведен пример работы алгоритма RandomUnderSampler [27].

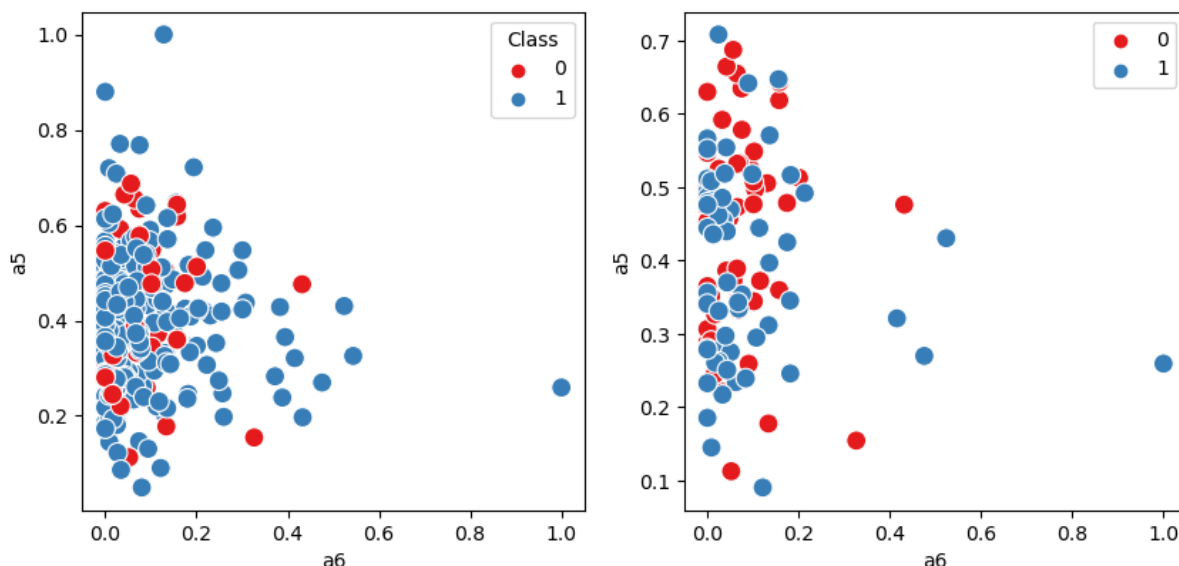


Рисунок 2.17 - Пример работы алгоритма RandomUnderSampler

CondensedNearestNeighbour (CNN) — это алгоритм удаления экземпляров, который выбирает подмножество информативных образцов из класса большинства, используя классификатор ближайших соседей. CNN начинает с небольшого подмножества большинства класса и итеративно добавляет образцы, которые правильно классифицированы классификатором ближайших соседей, пока не останется больше образцов. На рисунке 2.18 приведен пример работы алгоритма CondensedNearestNeighbour [27].

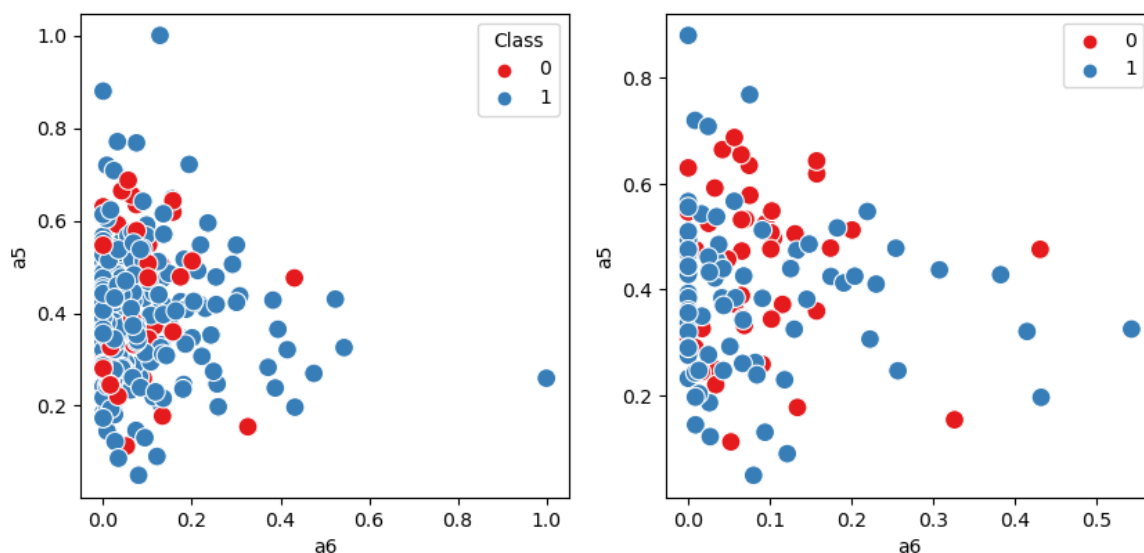


Рисунок 2.18 - Пример работы алгоритма CondensedNearestNeighbour

TomekLinks — это алгоритм удаления экземпляров, который удаляет выборки из класса большинства, которые близки к классу меньшинства. Этот

подход сохраняет информативные выборки в классе большинства, одновременно удаляя перекрывающиеся выборки, которые могут вызвать путаницу в модели. На рисунке 2.19 приведен пример работы алгоритма TomekLinks [27].

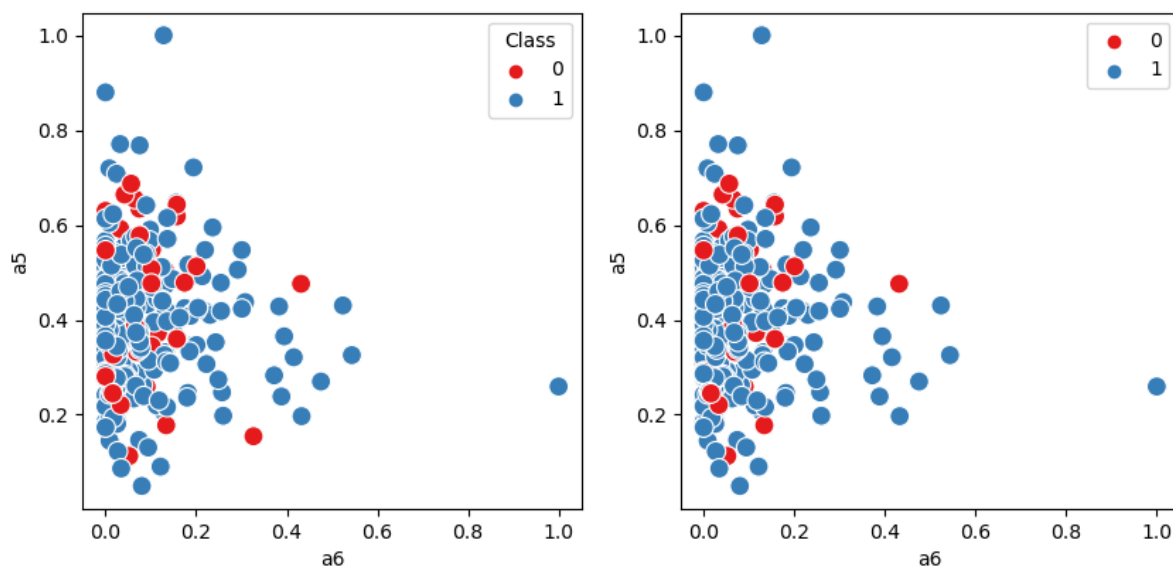


Рисунок 2.19 - Пример работы алгоритма TomekLinks

ClusterCentroids — алгоритм удаления экземпляров, который создает новые центры для класса большинства путем кластеризации существующих образцов. Этот подход направлен на уменьшение количества образцов в мажоритарном классе при сохранении информативной структуры данных. На рисунке 2.20 приведен пример работы алгоритма ClusterCentroids [27].

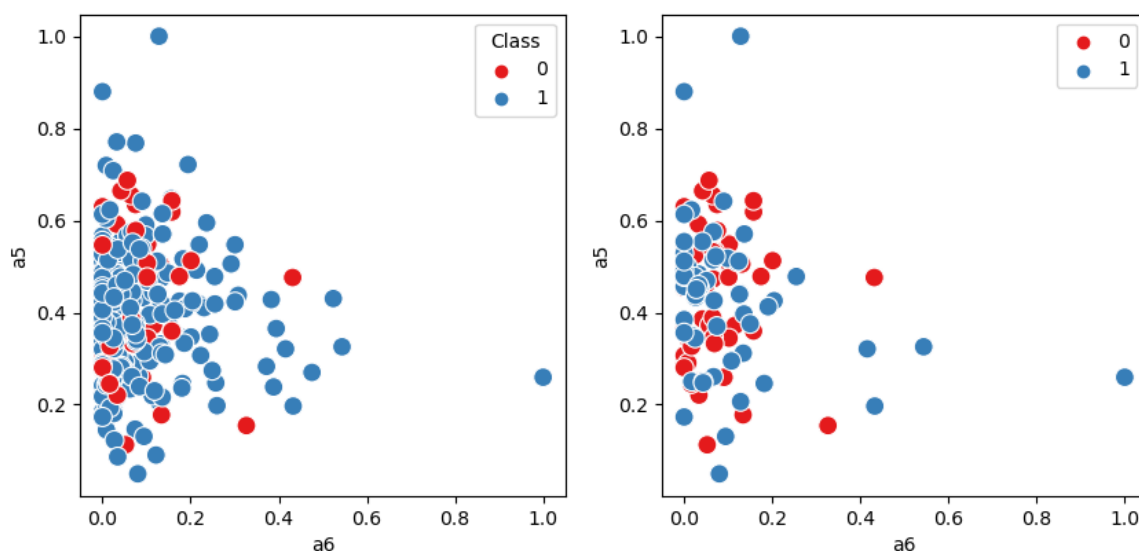


Рисунок 2.20 - Пример работы алгоритма ClusterCentroids

В целом, методы удаления экземпляров, такие как RandomUnderSampler, CondensedNearestNeighbor, TomekLinks и ClusterCentroids, могут помочь решить проблемы дисбаланса классов в машинном обучении, создавая более представительный и сбалансированный набор данных. Эти методы могут улучшить способность модели обнаруживать закономерности и делать точные прогнозы как для меньшинства, так и для большинства классов.

2.2.5.3 Комбинированная выборка

Комбинированная балансировка данных в машинном обучении — это подход, который предполагает одновременное использование нескольких методов балансировки для решения проблем дисбаланса классов. Этот метод направлен на создание более сбалансированного и репрезентативного набора данных путем генерации новых синтетических образцов в классе меньшинства при удалении избыточных или перекрывающихся образцов в классе большинства [28].

Одним из популярных методов комбинированной балансировки данных является алгоритм SMOTEENN, который сочетает метод добавления экземпляров SMOTE с методом удаления экземпляров EditedNearestNeighbors (ENN).

SMOTEENN сначала применяет SMOTE для создания новых синтетических образцов в классе меньшинства, а затем применяет ENN для удаления шумных или избыточных образцов как из класса меньшинства, так и из класса большинства. Этот подход может повысить точность класса меньшинства, одновременно уменьшая шум и избыточность в классе большинства. На рисунке 2.21 приведен пример работы алгоритма SMOTEENN [28].

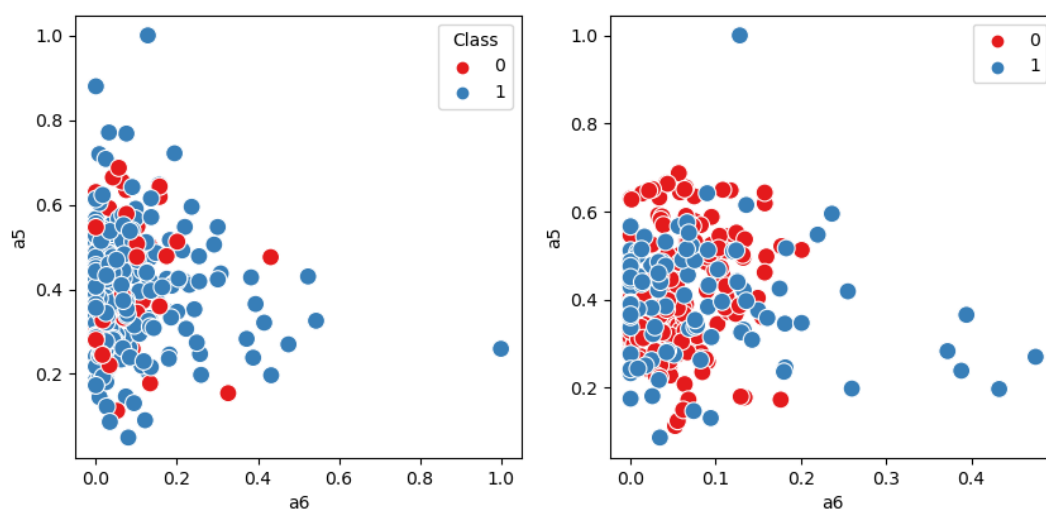


Рисунок 2.21 - Пример работы алгоритма SMOTEENN

Другим популярным методом комбинированной балансировки данных является алгоритм SMOTETomek, который сочетает метод избыточной выборки SMOTE с методом недостаточной выборки TomekLinks.

SMOTETomek сначала применяет SMOTE для создания новых синтетических образцов в классе меньшинства, а затем применяет TomekLinks для удаления образцов из классов меньшинства и большинства, которые близки к другому классу. Этот подход позволяет сохранить информативные выборки в классе большинства и одновременно удалить перекрывающиеся выборки, которые могут запутать модель. На рисунке 2.22 приведен пример работы алгоритма SMOTETomek [28].

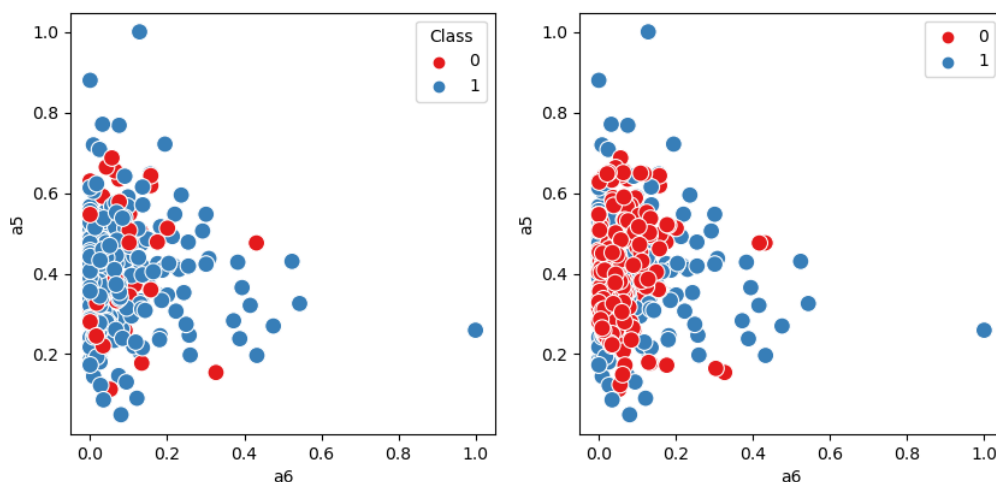


Рисунок 2.22 - Пример работы алгоритма SMOTETomek

В целом, комбинированные методы балансировки данных, такие как SMOTEENN и SMOTETomek, могут помочь решить проблемы дисбаланса классов в машинном обучении путем создания более сбалансированного и репрезентативного набора данных. Эти методы могут улучшить способность модели обнаруживать закономерности и делать точные прогнозы для классов меньшинства и большинства, что приведет к улучшению общей производительности модели.

2.3 Описание разработанной программы, решающей задачу диагностики болезни Паркинсона на несбалансированном наборе данных

В данном исследовании представлена программа, разработанная для выявления болезни Паркинсона по данным оцифрованных рисунков. Программа выполняет анализ различных методов и инструментов, используемых в машинном обучении для обработки несбалансированных наборов данных. Она включает предварительную обработку данных, выбор признаков, настройку гиперпараметров и оценку модели. Программа принимает на вход набор данных с несбалансированными классами и применяет различные методы балансировки, с целью повышения точности предсказаний. Затем она обучает и оценивает несколько классификаторов с помощью кросс-валидации и рассчитывает различные показатели эффективности, такие как сбалансированная точность, точность, f1-score,

частота ложных положительных и частота ложных отрицательных результатов. Код также визуализирует распределение классов в исходном и сбалансированном наборах данных с помощью диаграмм рассеяния.

Основная задача представленной программы – это повышение точности прогнозирования болезни Паркинсона путем решения проблемы дисбаланса классов. Также она предоставляет практический пример того, как обрабатывать несбалансированные наборы данных в машинном обучении, и сравнить производительность различных методов выборки и классификаторов. Проведенные эксперименты показали, что методы повторной выборки могут значительно улучшить производительность классификаторов на несбалансированных наборах данных, и что различные методы балансировки могут работать лучше для различных классификаторов. Результаты представленные в разделе 3, демонстрируют важность выбора признаков и настройки гиперпараметров для улучшения производительности моделей. Полный код программы представлен в приложении А.

3. ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ

3.1 Описание экспериментов

Целью проведенных экспериментов было выявление наилучшего сочетания метода устранения дисбаланса в данных и метода классификации.

Для сравнения результатов использовались некоторые метрики, а именно: общая точность (accuracy), сбалансированная точность (balanced_accuracy), f1, вероятность ложноположительной классификации (FP), вероятность ложноотрицательной классификации (FN). Сбалансированная точность является ключевым фактором оценки классификатора, т.к. в используемом наборе данных существует дисбаланс.

Общая точность вычисляется путем деления суммы, верно, предсказанных примеров на общее количество примеров [29]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

где TP (*True Positive*) – количество верно предсказанных примеров положительного класса,

TN (*True Negative*) – количество верно предсказанных примеров отрицательного класса,

FP (*False Positive*) – количество ошибочно предсказанных примеров положительного класса,

FN (*False Negative*) – количество ошибочно предсказанных примеров отрицательного класса.

Общая точность принимает значения от 0 до 1, где 1 означает идеальное предсказание, а 0 - полное неправильное предсказание. Высокое значение общей точности указывает на хорошую производительность модели, когда она способна правильно классифицировать большую долю примеров.

Однако общая точность может быть обманчивой в случае несбалансированных наборов данных, где один класс преобладает над

другим. В таких случаях, даже если модель предсказывает только самый распространенный класс, общая точность может быть высокой, но модель будет неэффективной в классификации редкого класса. В таких случаях рекомендуется использовать другие метрики, такие как F1-мера, сбалансированная точность и другие, которые учитывают несбалансированность классов.

Сбалансированная точность представляет собой среднее арифметическое между чувствительностью и специфичностью модели [30]:

$$BA = \frac{TP_{rate} + TN_{rate}}{2}$$

TP_{rate} (*True Positive Rate*) – доля верно предсказанных примеров положительного класса (чувствительность),

TN_{rate} (*True Negative Rate*) – доля верно предсказанных примеров отрицательного класса (специфичность).

Сбалансированная точность принимает значения от 0 до 1, где 1 означает идеальную производительность модели, а 0 – случайную классификацию. В случае, когда классы сбалансированы, сбалансированная точность будет совпадать с общей точностью (ассигасу). Однако, в случае дисбаланса классов, сбалансированная точность может быть более информативной метрикой для оценки модели, чем общая точность.

F1 мера вычисляется путем гармонического среднего точности и полноты [31]:

$$F1 = 2 * \frac{(Precision * Recall)}{(Precision + Recall)}$$

Precision (*точность*) – это отношение числа верно положительных примеров к общему числу примеров, которые были предсказаны как положительные,

Recall (полнота) – это отношение числа верно положительных примеров к общему числу примеров положительного класса в наборе данных.

F1-мера принимает значения от 0 до 1, где 1 означает идеальное предсказание, а 0 означает плохую производительность модели. F1-мера достигает высокого значения, если модель способна одновременно достичь высокой точности и полноты. F1-мера является особенно полезной метрикой, когда классы несбалансированы или когда один класс более важен, чем другой, поскольку она учитывает оба аспекта производительности - как правильно классифицированы положительные примеры, так и как мало примеров было упущено.

Так же для получения наилучших результатов используемыми методами, предварительно производилось:

- 1) нормализация данных;
- 2) подбор гиперпараметров;
- 3) отбор признаков.

Ниже приведены таблицы с результатами экспериментов с различными сочетаниями классификаторов и методов баланса.

Обозначения строк:

- LogisticRegression – классификатор логистической регрессии;
- DecisionTreeClassifier – классификатор дерева решений;
- MLPClassifier – классификатор нейронных перцептронов (нейронная сеть глубоко обучения);
- MultinomialNB – мультиномиальный наивный Байесовский классификатор;
- SVC - классификатор метода опорных векторов;
- Classifier – название классификационное алгоритма;
- BA (*Balanced accuracy*) – сбалансированная точность, доля правильных предсказаний;

- *Accuracy* – общая точность, отношение верно предсказанных классов к общему количеству;
- *F1* – гармоническое среднее точности и полноты;
- *FP* (false positive rate) – доля ложнопозитивных предсказаний;
- *FN* (false negative rate) – доля ложноотрицательных предсказаний.

3.2 Результаты экспериментов на несбалансированных данных

В таблице 3.1 представлены результаты экспериментов, которые были произведены без использования алгоритмов балансировки и без подбора гиперпараметров для классификаторов. В таблице 3.2 представлены результаты, которые так же были произведены без алгоритмов балансировки, но уже с подобранными гиперпараметрами.

Таблица 3.1 - Результаты экспериментов без подбора гиперпараметров и без использования алгоритмов балансировки

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	53,83	81,92	89,9	92,33	0
DecisionTreeClassifier	61,18	76,44	85,41	63,67	13,97
MLPClassifier	57,65	82,59	90,1	83,33	1,36
MultinomialNB	50	80,45	89,16	100	0
SVC	56,27	82,63	90,23	87	0,45

Таблица 3.2 - Результаты экспериментов с подобранными гиперпараметрами, но без использования алгоритмов балансировки

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	67,85	84,38	90,71	59,33	4,96
DecisionTreeClassifier	68,53	77,58	85,63	46,33	16,6
MLPClassifier	70,17	84,39	90,53	53,33	6,32
MultinomialNB	68,37	73,9	82,43	40,67	22,59
SVC	72,77	78,94	86,18	37,33	17,13

3.3 Результаты экспериментов на сбалансированных данных

3.3.1 Добавление экземпляров (Over-sampling)

В таблицах 3.3 – 3.5 представлены результаты экспериментов с использованием методов добавления признаков: RandomOverSampler, ADASYN, SMOTE.

Таблица 3.3 – Результаты экспериментов с методом RandomOverSampler

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	78,83	78,83	79,47	24,27	18,06
DecisionTreeClassifier	84,28	84,23	82,26	5,79	25,65
MLPClassifier	79,07	79,05	79,57	23,81	18,04
MultinomialNB	72,76	72,75	73,5	31,07	23,42
SVC	79,3	79,27	80,03	25,16	16,25

Таблица 3.4 – Результаты экспериментов с методом SMOTE

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	78,4	78,38	78,4	22,43	20,77
DecisionTreeClassifier	76,81	76,82	75,3	17,45	28,93
MLPClassifier	78,87	78,83	78,45	20,18	22,08
MultinomialNB	72,78	72,73	72,81	28,87	25,67
SVC	82,04	81,98	81,66	17	18,93

Таблица 3.5 – Результаты экспериментов с методом ADASYN

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	76,17	76,1	75,76	23,02	24,64
DecisionTreeClassifier	77,99	78,05	76,9	16,56	27,45
MLPClassifier	71,96	72,08	69,42	22,69	33,4
MultinomialNB	71,6	71,63	69,5	23,12	33,68
SVC	77,21	77,19	76,79	22,27	23,3

3.3.2 Удаление экземпляров (Under-sampling)

В таблицах 3.6 – 3.9 представлены результаты экспериментов с использованием методов удаления признаков: RandomUnderSampler, CondensedNearestNeighbor, TomekLinks, ClusterCentroids.

Таблица 3.6 – Результаты экспериментов с методом RandomUnderSampler

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	80,17	79,82	79,53	20,67	19
DecisionTreeClassifier	61,83	62,09	58,11	33,67	42,67
MLPClassifier	78,17	77,91	78,1	24,33	19,33
MultinomialNB	74,5	75,18	71,33	17	34
SVC	77,5	78	79	27	18

Таблица 3.7 – Результаты экспериментов с методом CondensedNearestNeighbor

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	63,88	65,67	71,29	51	21,25
DecisionTreeClassifier	46	45,26	46,1	53	55
MLPClassifier	66,71	68,21	72,68	45,33	21,25
MultinomialNB	55,87	56,54	60,13	49,33	38,93
SVC	60,21	60,9	64,89	46	33,57

Таблица 3.8 – Результаты экспериментов с методом TomekLinks

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	69,7	87,81	92,94	56,5	4,09
DecisionTreeClassifier	63,87	80,94	88,62	61	11,26
MLPClassifier	69,68	88,18	93,22	57	3,64
MultinomialNB	68,73	77,96	86,16	44,5	18,04
SVC	75,28	82,05	88,69	34,5	14,94

Таблица 3.9 – Результаты экспериментов с методом ClusterCentroids

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	74,67	74	73	25,33	25,33

Продолжение таблицы 3.9

DecisionTreeClassifier	59,33	59	56,36	38,33	43
MLPClassifier	75,33	74,91	74,64	25,67	23,67
MultinomialNB	69,5	69,27	61,95	14,67	46,33
SVC	73,5	73,45	73,76	27,67	25,33

3.3.3 Комбинированная выборка

В таблицах 3.10 – 3.11 представлены результаты экспериментов с использованием комбинированных методов балансировки: SMOTEENN, SMOTETomek.

Таблица 3.10 – Результаты экспериментов с методом SMOTEENN

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	90,2	92,35	87,05	4,05	15,56
DecisionTreeClassifier	84,08	88,45	78,72	4,05	27,78
MLPClassifier	89,14	91,31	85,42	5,05	16,67
MultinomialNB	84,09	86,02	77,58	10,71	21,11
SVC	91,75	91,98	87,72	7,69	8,89

Таблица 3.11 – Результаты экспериментов с методом SMOTETomek

<i>Classifier</i>	<i>BA, %</i>	<i>Accuracy, %</i>	<i>F1, %</i>	<i>FP, %</i>	<i>FN, %</i>
LogisticRegression	79,86	79,86	80,15	21,68	18,6
DecisionTreeClassifier	78,24	78,27	77,11	16,8	26,72
MLPClassifier	81,21	81,22	81,7	21,68	15,91
MultinomialNB	74,9	74,9	74,08	23	27,19
SVC	81,9	81,9	81,42	16,23	19,98

3.4 Результаты экспериментов из аналогичных исследований

В данной главе представлен обзор результатов экспериментов из статей, где проводились аналогичные исследования в области диагностики болезни Паркинсона на наборе данных HandPD. Эти статьи предлагают различные подходы к решению проблемы классификации на несбалансированном наборе данных HandPD и описывают результаты, достигнутые с использованием различных алгоритмов и моделей машинного обучения.

В работе [6] исследователи оценивали три метода классификации: наивный Байесовский классификатор (NB), лес оптимальных путей (OPF) и метода опорных векторов (SVM). Точность для контрольной группы и группы пациентов рассчитывается с помощью стандартного подхода (соотношение между правильными классификациями и общим количеством образцов для каждого конкретного класса).

Обозначения строк:

- Control group – контрольная группа (группа здоровых испытуемых);
- Patient group – группа пациентов (испытуемых страдающих болезнью Паркинсона);
- Global – сбалансированная точность.

Таблица 3.12 – Результаты экспериментов из работы [6]

	<i>OPF, %</i>	<i>NB, %</i>	<i>SVM, %</i>
Control group	64,96 ± 16,27	27,30 ± 37,36	12,50 ± 25,00
Patient group	60,23 ± 4,73	70,36 ± 39,08	96,49 ± 2,50
Global	55,86 ± 3,63	45,79 ± 4,15	58,61 ± 2,84

В исследовании [32] был проведен анализ неинвазивных биометрических методов для обнаружения и прогнозирования нейродегенеративных заболеваний. Были рассмотрены различные модальности, используемые для диагностики и мониторинга таких заболеваний. В частности, были рассмотрены рукописные данные, электроэнцефалограмма, речь, походка, движение глаз, а также применение комбинации данных от разных модальностей. В таблице 3.13 приведены результаты аккумулирующие исследования из различных источников, где использовался набор данных HandPD.

Таблица 3.13 – Результаты исследований из работы [32]

<i>Classifier</i>	<i>Accuracy, %</i>
ГП	72,36
AdaBoost + chisquare	76,44

Продолжение таблицы 3.13

CNN	79,62
LR, SVM	Меандры: 72,16 Спирали: 77,45

По результатам проведенных экспериментов с помощью разработанной программы удалось добиться максимальной точности в размере 92,35% на классификаторе LogisticRegression в связке с алгоритмом балансировки SMOTEENN. По сравнению с аналогами. Полученная точность выше на 26,49%, чем в работе [6] и на 12,39%, чем в работе [32] соответственно.

Заключение

Болезнь Паркинсона — это хроническое дегенеративное заболевание, от которого страдают миллионы людей во всем мире. Несмотря на хорошо известные симптомы, диагностика болезни Паркинсона на ранних стадиях остается сложной задачей. Существующие методы диагностики основаны на наблюдении симптомов, истории болезни и неврологических тестах, которые могут быть неэффективными на ранних стадиях заболевания и недоступны для большинства людей. Достижения в технологии машинного обучения открывают новые возможности для разработки более точных и экономически эффективных методов диагностики болезни Паркинсона.

В данной работе была представлена программа на основе машинного обучения для диагностики болезни Паркинсона на несбалансированном наборе данных HandPD. Программа использует оптимальные признаки и алгоритмы машинного обучения для анализа несбалансированных рукописных данных от пациентов с болезнью Паркинсона и здоровых людей. Программа имеет потенциал для улучшения ранней диагностики болезни Паркинсона, снижения стоимости диагностики и улучшения качества жизни пациентов.

В результате работы были изучены следующие методы и алгоритмы:

- алгоритмы классификации (LogisticRegression, DecisionTreeClassifier, MLP, MultinomialNB, SVC);
- методы балансировки данных: over-sampling (SMOTE, ADASYN, RandomOverSampler), under-sampling (RandomUnderSampler, CondensedNearestNeighbor, TomekLinks, ClusterCentroids), комбинированные методы (SMOTEENN, SMOTETomek);
- методы отбора данных (f_classif, mutual_info_classif, chi2).

Также была проведена серия экспериментов на разработанной модели, направленных на выявления болезни Паркинсона среди здоровых лиц и пациентов. В результате этих экспериментов удалось выяснить, что

благодаря извлечению и нормализации признаков, настройке гиперпараметров и балансировке данных с помощью добавления/удаления экземпляров и комбинации обоих методов, программа смогла выдавать более точные прогнозы, т.е. с более высокой точностью обнаружить болезнь Паркинсона среди здоровых людей и пациентов с болезнью Паркинсона.

Также были сделаны выводы, что лучшим показателем точности обладает классификатор `LogisticRegression` в связке с алгоритмом балансировки данных `SMOTEENN`. А наименьший показатель точности оказался у `MultinomialNB` в связке с алгоритмом балансировки данных `CondensedNearestNeighbor`.

Таким образом, разработанная программа представляет собой важный шаг вперед в диагностике болезни Паркинсона, и будущие исследования могут продолжить совершенствование и развитие этой технологии для повышения точности и доступности диагностики. Используя возможности машинного обучения и анализа почерка, мы можем потенциально изменить способ диагностики и лечения болезни Паркинсона, что приведет к улучшению жизни как для пациентов, так и для их семей.

Список использованных источников

1. Болезнь Паркинсона [Электронный ресурс]. URL: <https://www.who.int/ru/news-room/fact-sheets/detail/parkinson-disease> (дата обращения: 25.03.2023).
2. HandPD dataset [Электронный ресурс]. URL: <https://www.fc.unesp.br/~papa/pub/datasets/Handpd/> (дата обращения: 06.04.2023).
3. Что такое машинное обучение? [Электронный ресурс]. URL: <https://aws.amazon.com/ru/what-is/machine-learning/> (дата обращения 06.04.2023).
4. THE THREE CATEGORIES OF ALGORITHMS WITHIN MACHINE LEARNING EXPLAINED. [Электронный ресурс]. URL: <https://tmc-employeneurship.com/articles/the-three-categories-of-algorithms-within-machine-learning-explained> (дата обращения 07.04.2023).
5. Most Common Machine Learning Tasks. [Электронный ресурс]. Режим доступа: <https://vitalflux.com/7-common-machine-learning-tasks-related-methods/> (дата обращения 07.04.2023).
6. Improving breast cancer detection accuracy of mammography with the concurrent use of an artificial intelligence tool / S. Pacilè [et al.] // Radiology: Artificial Intelligence. 2020. V. 2. №. 6. P. e190208. (дата обращения 08.04.2023).
7. Concordance rate of radiologists and a commercialized deep-learning solution for chest X-ray: Real-world experience with a multicenter health screening cohort / Eun Young Kim [et al.] // Plos one. 2022. V. 17. №. 2. P. e0264383. (дата обращения 08.04.2023).
8. Automatic detection of expressed emotion in Parkinson's Disease / S. Zhao [et al.] // IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2014. P. 4813-4817.

9. Novel speech signal processing algorithms for high-accuracy classification of Parkinson's disease / A. Tsanas [et al.] // IEEE transactions on biomedical engineering. 2012. V. 59. №. 5. P. 1264-1271.
10. Variability in fundamental frequency during speech in prodromal and incipient Parkinson's disease: A longitudinal case study / Harel B, Cannizzaro M, Snyder PJ // Brain and cognition. 2004. V. 56. №. 1. P. 24-29.
11. A new computer vision-based approach to aid the diagnosis of Parkinson's disease / C. R. Pereira [et al.] // Computer Methods and Programs in Biomedicine. 2016. V. 136. P. 79-88.
12. Understand Data Normalization in Machine Learning. [Электронный ресурс]. URL: <https://towardsdatascience.com/understand-data-normalization-in-machine-learning-8ff3062101f0> (дата обращения 9.04.2023).
13. Normalization. [Электронный ресурс]. URL: <https://developers.google.com/machine-learning/data-prep/transform/normalization> (дата обращения 9.04.2023).
14. Feature scaling. [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Feature_scaling (дата обращения 11.04.2023).
15. Z-оценка. [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/Z%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B0> (дата обращения 10.04.2023).
16. How to Choose a Feature Selection Method For Machine Learning. [Электронный ресурс]. URL: <https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/> (дата обращения 12.04.2023).
17. Chi-Square Test for Feature Selection in Machine learning. [Электронный ресурс]. URL: <https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223> (дата обращения 12.04.2023).
18. Mutual Information. [Электронный ресурс]. URL: <https://www.kaggle.com/code/ryanholbrook/mutual-information> (дата обращения 12.04.2023).

19. F Statistic / F Value: Simple Definition and Interpretation. [Электронный ресурс]. URL: [h https://www.statisticshowto.com/probability-and-statistics/f-statistic-value-test/#WhatisF](https://www.statisticshowto.com/probability-and-statistics/f-statistic-value-test/#WhatisF) (дата обращения 12.04.2023).
20. Кросс-валидация. [Электронный ресурс]. URL: <https://academy.yandex.ru/handbook/ml/article/kross-validaciya> (дата обращения 16.04.2023).
21. Логистическая регрессия. [Электронный ресурс]. URL: <https://scikit-learn.ru/1-1-linear-models/#logistic-regression> (дата обращения 18.04.2023).
22. Деревья решений. [Электронный ресурс]. URL: <https://scikit-learn.ru/1-10-decision-trees/> (дата обращения 20.04.2023).
23. Многослойный перцептрон. [Электронный ресурс]. URL: <https://scikit-learn.ru/1-17-neural-network-models-supervised/#multi-layer-perceptron> (дата обращения 22.04.2023).
24. Мультиномиальный Наивный Байес. [Электронный ресурс]. URL: <https://scikit-learn.ru/1-9-naive-bayes/#gaussian-naive-bayes> (дата обращения 24.04.2023).
25. Метод опорных векторов SVM. [Электронный ресурс]. URL: <https://scikit-learn.ru/1-4-support-vector-machines/#classification> (дата обращения 26.04.2023).
26. Over-sampling. [Электронный ресурс]. URL: https://imbalanced-learn.org/stable/over_sampling.html (дата обращения 3.05.2023).
27. Under-sampling. [Электронный ресурс]. URL: https://imbalanced-learn.org/stable/under_sampling.html#under-sampling (дата обращения 7.05.2023).
28. Combination of over- and under-sampling. [Электронный ресурс]. URL: <https://imbalanced-learn.org/stable/combine.html> (дата обращения 12.05.2023).

29. Classification: Accuracy. [Электронный ресурс]. URL: <https://developers.google.com/machine-learning/crash-course/classification/accuracy> (дата обращения 20.05.2023).

30. Что такое сбалансированная точность? [Электронный ресурс]. URL: <https://www.codecamp.ru/blog/balanced-accuracy/> (дата обращения 20.05.2023).

31. F score. [Электронный ресурс]. URL: <https://en.wikipedia.org/wiki/F-score> (дата обращения 21.05.2023).

32. Биометрические данные и методы машинного обучения в диагностике и мониторинге нейродегенеративных заболеваний: обзор / И. А. Ходашинский, К. С. Сарин, М. Б. Бардамова [и др.] // Компьютерная оптика. 2022. Т. 46, № 6. С. 988-1020.

Приложение А

(Обязательное)

Исходный код программы

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from typing import Any
import sklearn
from sklearn.preprocessing import MinMaxScaler, normalize
from sklearn.model_selection import train_test_split
from sklearn.metrics import make_scorer, accuracy_score, f1_score,
balanced_accuracy_score, ConfusionMatrixDisplay, confusion_matrix
from sklearn.model_selection import cross_validate, StratifiedKFold,
GridSearchCV, RandomizedSearchCV
from sklearn.feature_selection import SelectKBest,
mutual_info_classif, chi2, f_classif
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
import imblearn
from imblearn.over_sampling import RandomOverSampler, SMOTE, ADASYN
from imblearn.under_sampling import RandomUnderSampler,
CondensedNearestNeighbour, TomekLinks, ClusterCentroids
from imblearn.combine import SMOTEENN, SMOTETomek
def resemple_data(
    balancing_alg: Any,
```

```

    title: str,
    x_before: pd.DataFrame,
    y_before: pd.Series,
) -> tuple[pd.DataFrame, pd.DataFrame]:
    print(title)
    x_after, y_after = balancing_alg.fit_resample(x_before,
y_before.ravel())
    print(f'Initial data : {Counter(y_before)}')
    print(f'Edited data : {Counter(y_after)}')
    return x_after, y_after
def scattering_matrix(
    x_after_sampling: pd.DataFrame,
    x_before_sampling: pd.DataFrame,
    y_after_sampling: pd.Series,
    y_before_sampling: pd.Series,
    alg_name: str
) -> None:
    print(f'Training data before and after used {alg_name}')
    figure = plt.figure(figsize=(10, 10))
    ax_1 = figure.add_subplot(2, 2, 1)
    sns.scatterplot(data=x_before_sampling, x='a6', y='a5',
hue=y_before_sampling, palette='bright', s=100)
    ax_3 = figure.add_subplot(2, 2, 2)
    sns.scatterplot(data=x_after_sampling, x='a6', y='a5',
hue=y_after_sampling, palette='bright', s=100)
def fp(y_true: pd.Series, y_pred: pd.Series): return
confusion_matrix(y_true, y_pred, normalize='true')[0, 1]
def fn(y_true: pd.Series, y_pred: pd.Series): return
confusion_matrix(y_true, y_pred, normalize='true')[1, 0]
def cross_validating_score(classifier: Any, x: pd.DataFrame, y:
pd.Series) -> float:
    scoring = {
        'balanced_accuracy': make_scorer(balanced_accuracy_score),

```

```

        'accuracy': make_scorer(accuracy_score),
        'f1': make_scorer(f1_score),
        'fp': make_scorer(fp),
        'fn': make_scorer(fn),
    }

    result = cross_validate(
        estimator=classifier,
        X=x,
        y=y,
        scoring=scoring,
        cv=StratifiedKFold(10),
        n_jobs=-1,
        return_train_score=False,
        return_estimator=True,
    )

    balanced_accuracy = np.round(100 *
np.mean(result["test_balanced_accuracy"]), 2);
    print(f'Classifier: {classifier}')
    print(f'balanced accuracy: {balanced_accuracy}%')
    print(f'accuracy: {np.round(100 *
np.mean(result["test_accuracy"]), 2)}%')
    print(f'f1: {np.round(100 * np.mean(result["test_f1"]), 2)}%')
    print(f'fp: {np.round(100 * np.mean(result["test_fp"]), 2)}%')
    print(f'fn: {np.round(100 * np.mean(result["test_fn"]), 2)}%')
    return balanced_accuracy, result['estimator']

def get_classifier_with_best_params(
    classifier: Any,
    parameters: dict[str, Any],
    method: Any,
    x: pd.DataFrame,
    y: pd.Series

```

```

) -> Any:
    estimator = method(classifier(), parameters, cv=10,
scoring='balanced_accuracy')
    estimator.fit(x, y)
    return estimator.best_estimator_
def show_best_k_value(
    filtering_alg: Any,
    x: pd.DataFrame,
    y: pd.Series,
    classifiers: list[Any]
) -> pd.DataFrame:

    y_copy = y.copy(deep=True)
    x_copy = x.copy(deep=True)

    max_balanced_accuracy = {}
    best_parameters_count = {}
    for clf in classifiers:
        max_balanced_accuracy[clf] = 0
        best_parameters_count[clf] = 0
    features_after_selection = None
    for k in np.arange(1, x.shape[1]):
        print(f"Parameters count {k}")
        features_after_selection =
get_features_after_selection(filtering_alg, k, x_copy, y_copy)
        for clf in classifiers:
            balanced_accuracy = cross_validating_score(clf,
features_after_selection, y_copy)[0]
            if max_balanced_accuracy[clf] < balanced_accuracy:
                max_balanced_accuracy[clf] = balanced_accuracy
                best_parameters_count[clf] = k
    print('\n')

```

```

    for key in max_balanced_accuracy.keys():
        print(f'Parameters count {best_parameters_count[key]} \n
Classifier: {key} \n value = {max_balanced_accuracy[key]} \n')
def get_features_after_selection(
    filtering_alg: Any,
    k_best_value: int,
    x: pd.DataFrame,
    y: pd.Series
) -> pd.DataFrame:
    filter = SelectKBest(filtering_alg, k=k_best_value).fit(x, y)
    x_after_selection = filter.fit_transform(x, y)

    x_names_short = filter.get_feature_names_out(x.columns) # Save
names emitted features

    x_after_selection = pd.DataFrame(x_after_selection,
columns=x_names_short)

    return x_after_selection
def cross_validating_score_all(classifiers: Any, x_train:
pd.DataFrame, y_train: pd.Series):
    for clf in classifiers:
        cross_validating_score(clf, x_train, y_train)[1][0]
        print('\n')
data = pd.read_excel('path_name')
class_value_counts = data.Class.value_counts() # (1 - 296, 0 - 72)
IR = class_value_counts[1] / class_value_counts[0]
print(f'IR: {IR}')
data.describe()
data = data.sample(frac=1)
target = data['Class'] # y
features = data.drop('Class', axis=1) # x
scaler = MinMaxScaler()
normalized_features = scaler.fit_transform(features)
features = pd.DataFrame(normalized_features, columns=features.columns)

```

```

# show_best_k_value(chi2, features, target, classifiers)
features = get_features_after_selection(chi2, 14, features, target)
random_state = 42
features_train, features_test, target_train, target_test =
train_test_split(
    features,
    target,
    train_size=0.75,
    random_state=random_state,
    shuffle=True,
    stratify=target
)
logreg = LogisticRegression(random_state=random_state)
tree = DecisionTreeClassifier(random_state=random_state)
mlp = MLPClassifier(random_state=random_state)
nb = MultinomialNB()
svc = SVC(random_state=random_state)
classifiers = [logreg, tree, mlp, nb, svc]
cross_validating_score_all(classifiers, features_train, target_train)
logreg_params = {
    'penalty': ['l1', 'l2'],
    'C': [0.1, 1, 10],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'max_iter': [100, 200, 300],
    'random_state': [random_state],
}
logreg = get_classifier_with_best_params(LogisticRegression,
logreg_params, GridSearchCV, features_train, target_train)
tree_params = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [3, 5, 7, 9],
    'min_samples_split': [2, 5, 10],

```

```

    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2'],
    'random_state': [random_state],
}

tree = get_classifier_with_best_params(DecisionTreeClassifier,
tree_params, GridSearchCV, features_train, target_train)

mlp_params = {
    'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)],
    'activation': ['relu', 'tanh', 'logistic'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.001, 0.01, 0.1],
    'learning_rate': ['constant','adaptive'],
    'random_state': [random_state],
}

mlp = get_classifier_with_best_params(MLPClassifier, mlp_params,
RandomizedSearchCV, features_train, target_train)

nb_params = {
    'alpha': [0.1, 0.5, 1, 5, 10],
    'fit_prior': [True, False],
    'class_prior': [None, [0.25, 0.75], [0.4, 0.6], [0.6, 0.4], [0.5,
0.5]],
}

nb = get_classifier_with_best_params(MultinomialNB, nb_params,
GridSearchCV, features_train, target_train)

svc_params = {
    'C': range(1, 4),
    'kernel': ('linear', 'poly', 'rbf', 'sigmoid'),
    'class_weight' : ('balanced', None),
    'random_state': [random_state]
}

svc = get_classifier_with_best_params(SVC, svc_params, GridSearchCV,
features_train, target_train)

classifiers = [logreg, tree, mlp, nb, svc]

```



```

cross_validating_score_all(classifiers, features_train, target_train)
features_before_sampling = features_train.copy(deep=True)
target_before_sampling = target_train.copy(deep=True)
features_after_sampling, target_after_sampling = resemple_data(
    RandomOverSampler(random_state=random_state),
    "RandomOverSampler",
    features_before_sampling,
    target_before_sampling,
)
cross_validating_score_all(classifiers, features_after_sampling,
target_after_sampling)
scattering_matrix(
    features_after_sampling,
    features_before_sampling,
    target_after_sampling,
    target_before_sampling,
    "RandomOverSampling"
)
features_before_sampling = features_train.copy(deep=True)
target_before_sampling = target_train.copy(deep=True)
features_after_sampling, target_after_sampling = resemple_data(
    SMOTE(k_neighbors=3, n_jobs=-1, random_state=random_state),
    "SMOTE",
    features_before_sampling,
    target_before_sampling,
)
cross_validating_score_all(classifiers, features_after_sampling,
target_after_sampling)
scattering_matrix(
    features_after_sampling,
    features_before_sampling,
    target_after_sampling,

```

```

        target_before_sampling,
        "SMOTE"
    )
    features_before_sampling = features_train.copy(deep=True)
    target_before_sampling = target_train.copy(deep=True)
    features_after_sampling, target_after_sampling = reample_data(
        ADASYN(n_neighbors=2, n_jobs=-1, random_state=random_state),
        "ADASYN",
        features_before_sampling,
        target_before_sampling,
    )
    cross_validating_score_all(classifiers, features_after_sampling,
                               target_after_sampling)
    scattering_matrix(
        features_after_sampling,
        features_before_sampling,
        target_after_sampling,
        target_before_sampling,
        "ADASYN"
    )
    features_before_sampling = features_train.copy(deep=True)
    target_before_sampling = target_train.copy(deep=True)
    features_after_sampling, target_after_sampling = reample_data(
        RandomUnderSampler(random_state=random_state),
        "RandomUnderSampler",
        features_before_sampling,
        target_before_sampling,
    )
    cross_validating_score_all(classifiers, features_after_sampling,
                               target_after_sampling)
    scattering_matrix(
        features_after_sampling,

```

```

        features_before_sampling,
        target_after_sampling,
        target_before_sampling,
        "RandomUnderSampler"
    )
    features_before_sampling = features_train.copy(deep=True)
    target_before_sampling = target_train.copy(deep=True)
    features_after_sampling, target_after_sampling = reample_data(
        CondensedNearestNeighbour(sampling_strategy='not minority',
                                   n_neighbors=5, n_seeds_S=5, n_jobs=-1, random_state=random_state),
        "CondensedNearestNeighbour",
        features_before_sampling,
        target_before_sampling,
    )
    cross_validating_score_all(classifiers, features_after_sampling,
                               target_after_sampling)
    scattering_matrix(
        features_after_sampling,
        features_before_sampling,
        target_after_sampling,
        target_before_sampling,
        "CondensedNearestNeighbour"
    )
    features_before_sampling = features_train.copy(deep=True)
    target_before_sampling = target_train.copy(deep=True)
    features_after_sampling, target_after_sampling = reample_data(
        TomekLinks(sampling_strategy='not majority', n_jobs=-1),
        "TomekLinks",
        features_before_sampling,
        target_before_sampling,
    )

```

```

cross_validating_score_all(classifiers, features_after_sampling,
target_after_sampling)
scattering_matrix(
    features_after_sampling,
    features_before_sampling,
    target_after_sampling,
    target_before_sampling,
    "TomekLinks"
)
features_before_sampling = features_train.copy(deep=True)
target_before_sampling = target_train.copy(deep=True)
features_after_sampling, target_after_sampling = resemple_data(
    ClusterCentroids(sampling_strategy='not minority', voting='hard',
random_state=random_state),
    "ClusterCentroids",
    features_before_sampling,
    target_before_sampling,
)
cross_validating_score_all(classifiers, features_after_sampling,
target_after_sampling)
scattering_matrix(
    features_after_sampling,
    features_before_sampling,
    target_after_sampling,
    target_before_sampling,
    "ClusterCentroids"
)
features_before_sampling = features_train.copy(deep=True)
target_before_sampling = target_train.copy(deep=True)
features_after_sampling, target_after_sampling = resemple_data(
    SMOTEENN(sampling_strategy='auto', random_state=random_state),
    "SMOTEENN",

```

```

        features_before_sampling,
        target_before_sampling,
    )
cross_validating_score_all(classifiers, features_after_sampling,
target_after_sampling)
scattering_matrix(
    features_after_sampling,
    features_before_sampling,
    target_after_sampling,
    target_before_sampling,
    "SMOTEENN"
)
features_before_sampling = features_train.copy(deep=True)
target_before_sampling = target_train.copy(deep=True)
features_after_sampling, target_after_sampling = reample_data(
    SMOTETomek(sampling_strategy='auto', random_state=random_state),
    "SMOTETomek",
    features_before_sampling,
    target_before_sampling,
)
cross_validating_score_all(classifiers, features_after_sampling,
target_after_sampling)
scattering_matrix(
    features_after_sampling,
    features_before_sampling,
    target_after_sampling,
    target_before_sampling,
    "SMOTETomek"
)

```