МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

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

Кафедра «Прикладная математика и информатика»
(наименование)

01.04.02 Прикладная математика и информатика
(код и наименование направления подготовки)

Математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Анализ онкологических патологий с использованием нейронных сетей»

Студент К.А. Сержантов
(И.О. Фамилия) (личная подпись)

Научный руководитель доцент, Т.Г. Султанов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Оглавление

Введение

Темой магистерской диссертации является анализ онкологических патологий с использованием нейронных сетей.

Заболеваемость онкологическими патологиями в России составляет порядка 330 случаев на сто тысяч населения. Рак кожи стоит на первом месте по диагностируемости. По статистике за 2018 год, представленной на рисунке 1.1, было выявлено 78.5 тысяч заболевших из которых 5.46 тысяч случаев приходится на меланому [26]. Заболеваемость меланомными кожи в Российской Федерации и в мире увеличивается, по данным Всемирной организации здравоохранения(ВОЗ) к 2025 году ожидается прирост заболеваний меланомы кожи на 25% [24].

Высокая летальность связанна с трудностями диагностики меланомы врачами общей практики. Существуют сложности в интерпретации признаков заболевания из-за которых невозможно на ранних стадиях отличить меланому от доброкачественных пигментных образований. А риск вызвать резкое прогрессирования опухоли лишает врачей возможности использовать инвазивные методы исследования (биопсия с гистологическим или цитологическим исследованием).

В выпускной квалификационной работе (ВКР) рассматривается неинвазивная методика диагностики патологий – спектроскопия комбинационного рассеяния (КР, Рамановская спектроскопия).

Рамановская спектроскопия основана на анализе неупругого рассеяния фотонов, предоставляя информацию о внутримолекулярных и межмолекулярных колебаниях помогая получить более полную картину состава тканей кожи. Поскольку опухоль имеет иную биохимическую структуру нежели здоровая ткань, спектроскопия комбинационного рассеяния дает характерную картину молекулярных колебаний («молекулярный отпечаток») и тем самым решает проблему дифференциации патологически измененной ткани кожи [24].

Разработка и тестирование программных модулей осуществлялась на языке программирования высокого уровня Python 3.8. Алгоритмы машинного обучения брались из библиотеки с открытым исходным кодом Scikit-learn v0.21.2, разработка программного кода осуществлялась в редакторе кода Visual Studio Code. По мимо этого в работе использовались и другие библиотеки программирование. Все они имею открытый исходный код и репозитории на github.

Проект разрабатывается при поддержке Самарского национального исследовательского университета имени академика С.П. Королева, материалы для исследования предоставлены ГБУЗ Самарским областным клиническим онкологическим диспансером. Протокол исследования был одобрен этическим комитетом Самарского государственного медицинского университета и проведен в Самарском клиническом онкологическом диспансере. Всем пациентам было не менее 18 лет. От каждого пациента было получено добровольное согласия на проведение in vivo исследования.

Данная магистерская диссертация является продолжением выпускной квалификационной работы, защищенной в 2018 году по теме: «Разработка системы анализа результатов диагностики кожных патологий для выявления злокачественных новообразований на основе нейронных сетей», в ходе которой проводилось исследование использования глубоких нейронных сетей в задачи дифференциации патологий.

Результатом предыдущей работы стал разработанный алгоритм принятия взвешенного решения, основанном на логическом выводе предобученных и тонко настроенных глубоких нейронных сетей. Точность работы данного алгоритма составила 70%, специфичность – 53%, чувствительность – 93,64 [31].

Низкие показатели точности и специфичности в совокупности с использованием сравнительно небольшого набора данных стали причиной отказа от использования нейронных сетей на данном этапе исследования. В

качестве альтернативы было решено исследовать использование алгоритмов машинного обучения.

Цель выпускной квалификационной работы: повышение качества дифференциации злокачественных новообразований от здоровой кожи на спектрах комбинационного рассеяния с помощью алгоритмов машинного обучения.

Объект исследования – процесс дифференциации образцов биоткани по результатам спектроскопии комбинационного рассеивания.

Предмет исследования – алгоритмы машинного обучения.

Выпускная квалификационная работа состоит из аннотации, введения, четырех глав и заключения.

Во введении описывается актуальность рассматриваемой темы, определяются объект и предмет выпускной квалификационной работы, ставится цель и выявляются задачи.

В первой главе выпускной квалификационной работы была обоснована актуальность рассматриваемой темы. Рассмотрены патологии, присутствующие в наборе данных. Выявлены перспективы использования машинного обучения в медицине. Проведен анализ актуальных способов диагностики патологий различными алгоритмами машинного обучения, на основе различных научных работ в данной области.

Во второй главе выпускной квалификационной работы был рассмотрен обучающий набор данных представленный ГБУЗ Самарским областным клиническим онкологическим диспансером, содержащий спектрограммы патологий и образцов кожи. Опираясь на исследования спектров комбинационного рассеивания были сформированы рекомендации по предобработке обучающего набора. Была составлена математическая модель обучения системы дифференциации патологий, а также написан псевдокод для иллюстрации ее работы. Были рассмотрены основные алгоритмы машинного обучения применяемые в данного рода задачах: k-NN, наивный

байесовский алгоритм, деревья принятия решений, логическая регрессия, опорные вектора.

В третьей главе был осуществлен выбор средств разработки подсистема обучения классификаторов. На основе математической модели и псевдокода сформулированных во второй главе, был реализован класс Classifications, включающий в семя метод уменьшения размерности входных данных, функции нормализации, обучения и тестирования классификаторов.

В четвертой главе проводилось тестирование разработанного программного кода.

В заключении подводятся итоги исследования, формируются окончательные выводы по рассматриваемой теме.

Практическая значимость диссертационного исследования заключается в применении системы в качестве вспомогательного инструмента врача в диагностике заболеваний, что позволит повысить эффективность и уменьшить коэффициент ошибки.

Методы исследования, которые использовались в процессе формирования диссертационной работы: анализ и синтез модели, математическое моделирование, экспериментальные измерения и анализ.

Диссертационное исследование производилось с 2018 по 2020 гг. в три этапа:

1. Констатирующий этап исследования (2018 г.) состоял в формализации темы, цели, задач, гипотезы исследования, подтверждении актуальности решения проблемы, произведения обзора современного состояния темы и определении методики решения задач.

2. Моделирующий этап (2018-2019 гг.) состоял в обзоре и выборе методов классификаций, моделировании математической модели, апробации результатов исследования на научных конференциях и формализации статей.

3. Экспериментальный этап (2020 г.) состоял в программной реализации разработанной математической модели системы

дифференциации патологий. И разработке пограничных методов необходимых для работы системы в целом

1. Участие в онлайн конференции SPIE.PHOTONICS EUROPE Digiral Forum 6-10 April 2020, Статья и видео презентация.

2. Публикация во II всероссийской научной конференции с международным участием «информационные технологии в моделировании и управлении: подходы, методы, решения».

3. Публикация в III всероссийской научной конференции с международным участием «информационные технологии в моделировании и управлении: подходы, методы, решения».

Научная новизна заключается в получении высокой точности дифференциации патологий по спектрам комбинационного рассеивания в задаче бинарной классификации на основе набора данных в котором большую долю патологий занимают меланомы и ее пограничные состояния с помощью различных алгоритмов машинного обучения и их ансамблей.

На защиту предоставляются:

1. Задача дифференциации патологий по спектрам комбинационного рассеивания.

2. Математическая модель системы дифференциации патологий.

3. Программный код системы дифференциации патологий.

4. Обученный классификатор.

Диссертационная работа состоит из введения, четырех глав, заключения, списка литературы и пяти приложений. Работа изложена на 102 страницах, включает 41 иллюстрацию, 5 таблиц, 29 формул, 38 использованных источников.

Глава 1 Анализ проблемы онкологических заболеваний и методов диагностики

1.1 Злокачественные опухоли

Злокачественная опухоль (рак) – это группа злокачественных клеток, способных расти и разрушать близлежащие ткани, а также имеющие способность метастазировать в другие части тела.

Заболеваемость онкологическими патологиями в России составляет порядка 330 случаев на сто тысяч населения. Рак кожи стоит на первом месте по диагностируемости. По статистике за 2018 год, представленной на рисунке 1, было выявлено 78.5 тысяч заболевших из которых 5.46 тысяч случаев приходится на меланому [1]. Заболеваемость меланомными кожи в Российской Федерации и в мире увеличивается, по данным Всемирной организации здравоохранения(ВОЗ) к 2025 году ожидается прирост заболеваний меланомы кожи на 25% [24].

Рисунок 1 – Статистика диагностики злокачественных опухолей за 2018 год

За последние десятилетия заболеваемость меланомными и немеланомными типами рака кожи увеличилась.

В настоящее время ежегодно во всем мире регистрируют от 2 до 3 миллионов случаев немеланомного рака кожи и 132 000 случаев меланомного рака кожи. Каждый третий диагноз рака – это рак кожи, и, согласно статистике Фонда по раку кожи, каждый пятый американец заболевает раком кожи в течение жизни [36].

С каждым годом уровень озона в атмосфере уменьшается. Озоновый слой все хуже выполняет роль защитного фильтра земли позволяя солнечному ультрафиолетовому излучению достигает поверхности Земли. По оценкам экспертов, снижение уровня озона на 10 процентов приведет к дополнительным 300 000 немеланомных и 4500 случаев меланомного рака кожи [22].

Основные факторы, предрасполагающие к развитию меланомы, связаны с местным воздействием солнца и наличием тяжелых солнечных ожогов в анамнезе. Но данные факторы находится в зоне ответственности каждого отдельного человека [22].

Многочисленные исследования в этой области доказали, что воздействие ультрафиолета помимо немеланомных типов рака кожи вызывает и меланому. А также повышает риск превращения доброкачественных кожных образований в меланому.

Из-за относительного отсутствия пигментации кожи европейские расы обычно имеют гораздо более высокий риск заболеть кожно-раковыми заболеваниями, в сравнении с темнокожими [22].

Люди имеющие естественный коричневые/черные цвет кожи, типы V, VI, они представлены на рисунке 2 и в таблице 1, в большинстве случаев (исключаются индивидуальные особенности организма) могут безопасно переносить относительно высокие уровни воздействия солнечного облучения, не получая значительных солнечных ожогов.

Люди с бледной или веснушчатой кожей, светлыми или рыжими волосами и голубыми глазами попадают в самую высокую группу риска (типы кожи I, II);

Таблица 1 – Классификация кожи по фототипу

Классификация типов кожи	Вероятность получить солнечной ожог	Вероятность приобрести загар после пребывания на солнце
I	Высокая	Очень низкая
II	Средняя	Низкая
III	Низкая	Средняя
IV	Очень низкая	Высокая
V	Естественный коричневый цвет	
VI	Естественный черный цвет	

Рисунок 2 – Классификация кожи по фототипу

Люди с темными волосами и глазами, которые обычно не загорают на солнце, подвергаются среднему риску развития рака кожи. Они относятся к III, IV типу кожи.

Ниже представлены некоторые индивидуальные факторы риска развития рака кожи, не зависящие от расовой принадлежности:

- светлая кожа;
- голубые, зеленые или карие глаза;
- светлые волосы;
- тяжелые солнечные ожоги в анамнезе
- обилие родинок;
- наличие веснушек;
- рак кожи в семейном анамнезе.

Согласно проведенному обзору научных статей за период 2018-2020 года, статистическим данным ВОЗ и Министерству здравоохранения Российской Федерации делается вывод, что исследования в области ранней диагностики раковых патологий на данный момент являются актуальными. Ежегодное увеличение случаев заболевания кожно-раковыми патологиями, а также увеличение солнечной активности в совокупности с неграмотностью населения в вопросах защиты от ультрафиолетового излучения.

Сохранении процента летальных исходов меланомных кожно-раковых опухолях при увеличении количества диагностируемых случаем, может говорить о нескольких проблемах:

1. низкая точность диагностики патологий на ранней стадии развития;
2. неосведомленность населения о первичных признаках развития злокачественной патологии, с которыми стоит обратится к специалисту.

Был проведен обзор различных кожно-раковых патологий.

1.2 Классификация кожно-раковых патологий

Злокачественные опухоли кожи обычно делят на две большие подгруппы меланомные и немеланомные.

Меланомный рак кожи начинается в клетках – меланоцитах. Меланоциты образуют пигмент – меланин. Данный пигмент придает коже, волосам и глазам их цвет. Меланоциты способны группироваться и образовывать родинки на коже. Они появляются в виде бугорков или пятен, имеющие обычно коричневые или розовые оттенки. У большинства людей с рождения имеются родинки, каждая из них по своей сути являются незлокачественными (доброкачественными) опухолями.

В некоторых случаях изменения в меланоцитах может вызвать меланомный рак кожи. Изменение цвета, размера или формы родинки обычно является первым признаком меланомного рака кожи.

Существует 4 основных клинические формы меланомы кожи. Наиболее распространенной формой является поверхностно-распространяющаяся меланома. реже встречается узловая меланома (нодулярная), лентигинозная меланома (Lentigo Maligna) и акролентигинозная меланома (Acral Lentigo Maligna).

Рассмотрим патологии (невус, меланома, диспластические меланоцитарные невусы) присутствующие в материалах исследования предоставленных ГБУЗ Самарским областным клиническим онкологическим диспансером. Протокол исследования был одобрен этическим комитетом Самарского государственного медицинского университета и проведен в Самарском клиническом онкологическом диспансере. Всем пациентам было не менее 18 лет. От каждого пациента было получено добровольное согласия на проведение исследования.

1.2.1 Меланома

Меланома, данная патология представлена на рисунке 3, является самой смертельной формой рака кожи. В структуре злокачественных

опухолей кожи она составляет 6 – 7%, однако ввиду очень агрессивного течения заболевания занимает первое место по смертности.

Рисунок 3 – Меланома

Меланома – это рак, который начинается в меланоцитах. Большинство клеток меланомы все еще производят меланин, поэтому опухоли меланомы обычно коричневых или черных оттенков. Но в некоторые случаях клетки меланомы не вырабатывают меланин потому они могут быть розового или белого цвета.

Наличие темного пигмента в кожа снижает риск развития меланомы. Но на теле человека присутствуют места, в которых меланин практически отсутствует: ладони, подошвы ног или места под ногтями. Локализация меланомы в данных местам более вероятна у афроамериканцев, чем у людей с более светлыми оттенками кожи.

Меланомы могут также образовываться в других частях вашего тела, таких как глаза, рот, гениталии и анальная область, но данные случаи встречаются гораздо реже, чем меланома кожи.

Высокая летальность данной патологии связанна с трудностями диагностики меланомы на ее ранних стадиях врачами общей практики. Из-за

сложности в интерпретации внешних признаков заболевания на ранних стадиях сложно отличить меланому от доброкачественных пигментных образований, без сдачи дополнительных анализов, но риск вызвать резкое прогрессирования опухоли затрудняет использование инвазивных методов исследования. Потому поиск, усовершенствование неинвазивных способов диагностики патологий является приоритетной задачей.

1.2.2 Невус

Невусы – является доброкачественным новообразованием (Рисунок 4). Они классифицируются на основе сочетания клинических и гистопатологических критериев. Все невусы можно поделить на врожденные и приобретенные. Все приобретенные невусы можно разделить на основании расположения гнезд меланоцитов и невоцитов в коже на внутридермальные невусы (скопление меланоцитов глубоко под кожей), эпидермальные (скопление меланоцитов в верхнем слое кожи) и пограничные или смешанные (скопление меланоцитов между эпидермисом и дермой. Было показано, что одним из ключей к распознаванию меланомы является знание многих сторон доброкачественных поражений.

Рисунок 4 – Меланоцитарный невус

Ниже приведены основные виды данной патологии:

Глобулярный (врожденный) невус. В понятие глобулярного врожденного невуса обычно включают доброкачественные патологии(родинки), присутствующие при рождении или появившиеся до полового созревания.

С клинической, дермоскопической или гистопатологической точки зрения, найти различие между врожденными родинками и рано приобретенными обычно невозможно. Потому, что мелкие врожденные родинки и рано приобретенные родинки часто имеют сходные черты. Они имеют чаще всего шаровидную структуру, тогда как приобретенные невусы взрослых чаще всего ретикулярны. У детей врожденные невусы коричневого цвета, плоские или слегка приподнятые, симметричные, обычно размером менее 15 мм.

Приобретенный (ретикулярный) невус.

Ретикулярный невус – это поздно приобретенные меланоцитарные патологии, чаще всего наблюдаемые у взрослых. Подразделяются на маленькие (меньше 6 мм) и большие (6 мм и более), но обычно их размеры не превышают 15 мм. Бывают плоскими или слегка приподнятыми, с коричневыми или черными повреждениями.

Небольшие поражения обычно симметричны и мономорфны. Дермоскопически они характеризуются регулярной пигментной сеткой с участками гипопигментации или без них и / или бесцветной коричневой или черной окраской. Большие поражения, как правило, имеют многоочаговый характер, состоящий из пятнистого распределения множества гипо- и гиперпигментированных областей, но иногда проявляют нетипичные признаки.

1.2.3 Диспластические меланоцитарные невусы

Данная патология является пограничным состоянием между невусов и меланомой. Не смотря на ее принадлежность к классу невусов она была выделенна отдельно т.к. имеет отдельный класс в используемой в работе базе данных.

Диспластические (атипичные, по Кларку) меланоцитарные невусы приобретают пигментированные меланоцитарные пролиферации кожи с отчетливыми клиническими и гистологическими признаками (Рисунок 5).

Рисунок 5 – Диспластические меланоцитарные невусы

Гистологическая дисплазия может быть легкой, средней или тяжелой. Выраженные патологические критерии диспластического невуса:

- диспластический невус может быть соединительным невусом (когда меланоциты обнаруживаются в эпидермодермальном соединении) или сложным невусом (когда меланоциты обнаруживаются в эпидермодермальном соединении и в дерме);
- данные образования часто имеют неправильные размеры и форму и могут «соединяться» или объединяться;
- клетки могут быть веретенообразными (удлиненными) или эпителиоидными (широкими, напоминающими эпидермальные кератиноциты);
- может быть цитологическая атипия (клетки, которые меньше или больше, чем обычно);
- может присутствовать фиброз или рубцы в дерме;
- воспалительные клетки могут проникать в очаг поражения;
- связанные кровеносные сосуды могут быть увеличены или увеличены.

В соответствующих клинических условиях диспластические (атипичные) меланоцитарные невусы являются кожными маркерами для развития семейных и несемейных меланом.

Диспластические меланоцитарные невусы, невусы Кларка, невусы с архитектурным расстройством и цитологическая атипия меланоцитов являются синонимичными терминами для описания приобретенной атипичной меланоцитарной пролиферации [29].

Люди с 5 и более атипичными невусами имеют высокие риски развития меланомы в сравнении с остальным населением. Считается, что наличие данной патологии в шесть раз повышает риск развития меланомы.

Меланоцитарные невусы безвредны (доброкачественные) и не нуждаются в удалении. Однако даже опытному дерматологу не всегда легко определить, является ли патология невусом или меланомой, особенно если имеются нетипичные признаки. В случае сомнений, подозрительный или измененный атипичный невус должен быть удален при помощи иссечения ткани, после чего должна быть проведена биопсия всей удаленной ткани. Частичной биопсии лучше избегать, так как при исследовании можно пропустить небольшой очаг меланомы [20].

1.3 Метод проведения исследования.

Инвазивные методы исследования имеют самые высокие показатели точности диагностики патологий. Но при работе с меланомными патологами рекомендуется избегать любых ее повреждений из-за риска спровоцировать метастазирование опухоли.

Потому инвазивные исследования кожных патологий проводится лишь для подтверждения ранее иссеченной ткани. Удаление же каждого пигментированного новообразования неприемлемо для пациента. Исследование [34] показало, что 40% биопсий, полученных с подозрением на злокачественные поражения кожи, были доброкачественными, и поэтому

неуместная хирургическое вмешательство является довольно частым случаем.

Для диагностирования меланомных раковых патологий применяются различные неинвазивные методы исследования, имеющие относительно невысокую точность, зависящую от квалификации врача. К числу таких методов относят нижеследующие.

Дерматоскопия – метод визуальной оценки кожных поражений для быстрого подтверждения диагноза (дает возможность распознавать морфологические структуры, невидимые невооруженным глазом) [8].

In vivo конфокальная микроскопия – лазерное оптическое сканирование, позволяет получить изображение слоев кожи с высокой степенью разрешения, в реальном времени в трех измерениях (высота, ширина и глубина). Оптическая когерентная томография для зондирования биоткани используется оптическое излучение ближнего инфракрасного диапазона; метод обладает высоким разрешением, контрастностью, глубина исследования 1,5 мм [8].

Ультразвуковое исследование кожи – неинвазивный метод исследования кожи, обладающий высокой точностью и воспроизводимостью, позволяющий осуществлять дифференцировку слоев кожи, оценивать состояние васкуляризации; оперирующий понятиями «эхогенность», «эхоструктура» [8].

Методы, представленные выше не являются надежным неинвазивным способом диагностики кожно-раковых заболеваний. Данная область является развивающейся. Рассматриваются различные альтернативные методы диагностике злокачественных опухолей и одним из таких методов являются оптическая спектроскопия. Они позволяют неинвазивно диагностировать раковые опухоли опираясь на состав кожи выявляя сигнатуры различных видов опухолей.

Данная технология применяется в различных областях жизнедеятельности: для определения компонентного состава газов,

жидкостей, порошков, твердых тел так же используется в фармакологии, материаловедении, контроле продукции и других сферах.

Отдельно рассмотрим подкласс методов оптической спектроскопии – спектроскопию комбинационного рассеивания.

1.4 Спектроскопия комбинационного рассеивания

Спектроскопия комбинационного рассеивания является эффективным методом анализа общего химического состава образцов из различных профессиональных областей, благодаря своей способности регистрировать спектральные характеристики молекул.

Рамановская спектроскопия основана на изменении длины волны возбуждения после взаимодействия с молекулами исследуемого образца. Зарегистрированный спектр позволяет определять химические связи в исследуемом образце.

При возбуждении новообразований кожи длиной волны лазера в ближней инфракрасной области рамановский и автофлуоресцентный сигналы регистрируются одновременно. Таким образом, анализ всего зарегистрированного спектра тканей, включая рамановские и аутофлуоресцентные сигналы, позволяет анализировать как биохимические, так и структурные особенности различных новообразований кожи. Тем не менее, нет конкретных рамановских и/или автофлуоресцентных полос, которые соответствуют определенным новообразованиям кожи. Более того, одни и те же химические компоненты вносят вклад в разные спектральные полосы, в результате чего анализируемые спектральные данные имеют множественные корреляции [32].

Исследование новообразований кожи in vivo проводилось с использованием портативной системы, регистрирующей спектры комбинационного рассеивания. Подробная схема спектроскопической установки показан на рисунке 6.

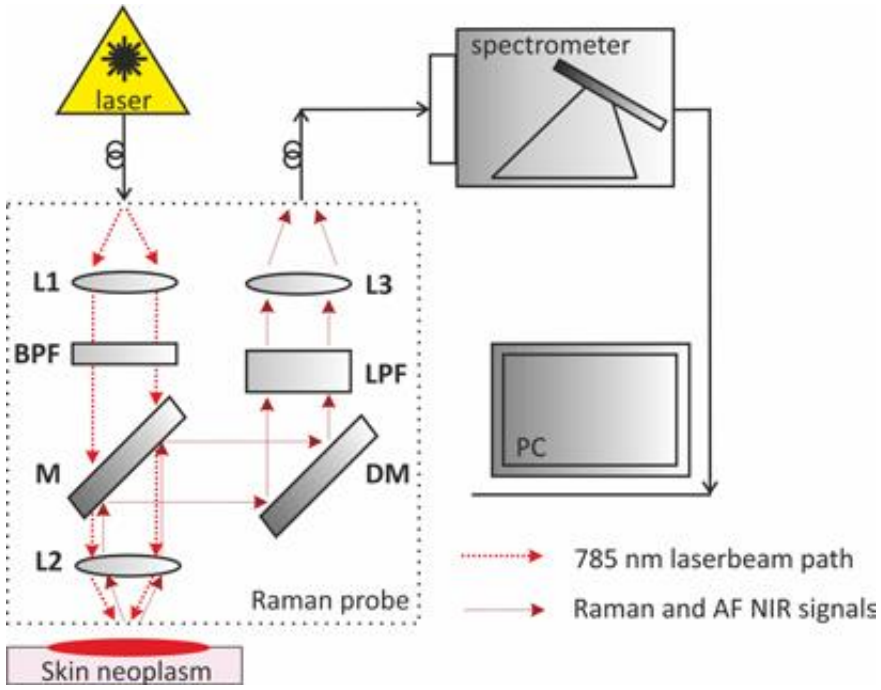

Рисунок 6 – Схема экспериментальной установки для регистрации спектров комбинационного рассеивания

Стимуляция собранных спектров осуществляется лазерным модулем LuxxMaster LML-785.0RB-04 (PD-LD, Нью-Джерси, США) с центральной длиной волны 785 нм. Рамановский зонд RPB785 (InPhotonics, Массачусетс, США) позволяет нам фокусировать возбуждающее излучение, собирать и фильтровать рассеянное излучение.

Фокусное расстояние используемого рамановского зонда составляет 7,5 мм; следовательно, расстояние между тестируемым образцом кожи и выходной линзой рамановского зонда составляло 7 мм. Выходная мощность лазера от рамановского зонда составляла 150 мВт. Собранный сигнал разлагается на спектр с использованием портативного спектрометра QE65Pro (Ocean optics, Флорида, США) [32].

1.5 Способы анализа спектров комбинационного рассеивания

Был проведен анализ научных работ проводившихся на базе материалов, представленных ГБУЗ Самарским областным клиническим онкологическим диспансером.

В данной работе исследование оптических свойств тканей кожи с помощью рамановской спектроскопии и автофлуоресцентного анализа быловыполнено. Анализ различных новообразований кожи проводился на основе спектров комбинационного рассеяния и автофлуоресценции. лазером 785 нм.

В работе использовались различные математические методы такие как Полная ширина на уровне половинной амплитуды (англ. FWHM — full width at half maximum), бинарная логическая регрессия (binary logistic regression). Предложенные методы диагностики позволили достичь следующих результатов: точность до 82,1%. чувствительность 83,3% и специфичность 80,9% [33].

Данные исследования показывают эффективность математических методов при работе со спектрами. Позволяя выставить гипотезу эффективности использования алгоритмов машинного обучения для задачи дифференциации патологий. Рассмотрим машинное обучение подробнее.

1.6 Машинное обучение и перспективы его использования в медицине

Искусственный интеллект (ИИ) считается инновационном прорывам в здравоохранении и медицине. Последние шесть десятилетий ИИ успешно и широко применяется во многих областях медицинских исследований и клинических практик [7].

За последнее десятилетие в научном сообществе стремительно растет число опубликованных работ, посвященных влиянию ИИ в различных сферах деятельности. Библиометрическое исследование показало, что за последние 3 года число исследований по применению ИИ в медицине

утроилось, причем наибольший интерес вызывают именно исследования рака [38].

Представим несколько определений для терминов, часто используемых в работе:

Искусственный интеллект – это отрасль компьютерной науки, которая использует машины и программы для моделирования интеллектуального поведения человека.

Машинное обучение (Machine learning, ML) – один из инструмент искусственного интеллекта, включающий в себя множество видов машинного обучения, которое позволяет достичь целей искусственного интеллекта (Рисунок 7). Машинное обучение охватывает множество алгоритмов и статистических методов обработки данных: логистическую регрессию, случайный лес, глубокое обучение.

Классификатор – это инструмент, применяемый в машинном обучении, который использует классифицированные данные и на их основании пытается предсказать, к какому классу стоит отнести новые данные.

Рисунок 7 – Искусственный интеллект и его подмножества

В последнее время машинное обучение обращает на себя внимание из-за своего повсеместного использования в повседневной жизни: от персонализированной рекламы и новостей в интернете до автомобилей с автоматическим управлением.

В медицине и в частности в дерматологии имеется большое количество данных: клинические записи, демографические данные пациентов, результатов обследований и информация, полученная из анкет. Это огромное количество информации, которая может перевернуть медицину [15].

Последние достижения в цифровой области такие как более быстрая обработка и более дешевое хранение данных, позволили разработать и использовать алгоритмы машинного обучения с искусственным интеллектом, которые имеют многочисленные применения в дерматологии [15].

Чтобы оценить эффективность этих новых технологий, крайне важно, чтобы дерматологи имели базовое представление об искусственном интеллекте и ML.

Существует множество алгоритмов ML, которые используются в дерматологии. Большинство алгоритмов ML являются примерами статистического обучения; например, некоторые из наиболее распространенных статистических методов обучения - это линейная регрессия, логистическая регрессия, **k**-ближайших соседей **(k**-NN), метод опорных векторов (SVM), случайный лес (RF).

Неинвазивная компьютерная диагностика поражений кожи на данный момент является развивающееся областью. Большая часть существующих исследований затрагивает изучение немеланомные случаи рака кожи, в то время как дифференциация меланомы, так и остается актуальной диагностической задачей.

Данная ситуация во многом связанна со сложностью визуального определения меланомы. Для каждого зарегистрированного случая необходима проведение инвазивного исследования, что критично при

данного рода заболевании. Повреждение опухоли может привести к резкому росту раковых клеток и летальному исходу пациента.

Использование искусственного интеллекта для диагностики в дерматологии имеет благоприятную тенденцию на данный момент. Эти системы обычно используют некоторую форму машинного обучения, которая представляет собой подмножество ИИ, включающее методы, позволяющие машинам делать прогнозы на основе их предыдущих данных и опыта. В отличие от обычных моделей, которые явно запрограммированы для обработки статического набора случаев, алгоритмы ML могут выводить свои собственные решения на основе обучающего набора и точно выполнять их в новых сценариях.

1.7 Примеры использования машинного обучения в медицине

Большая часть существующих методик компьютерной диагностики поражений кожи основывает свои исследования на анализе изображений. В статьях затрагивающих данную область исследования сообщается, что эффективность выше или равна диагностической точности среднестатистического дерматолога, но относительно немногие из исследователей представили высокий уровень доказательной базы.

Поиск в базе данных дал 8657 результатов, из которых 2285 были признаны уникальными. Названия и рефераты уникальных исследований были рассмотрены, и 2211 статей были признаны неактуальными и исключены.

Ручной обзор ссылок, цитируемых в оставшихся 74 исследованиях, выявил семь дополнительных исследований, которые могут иметь отношение к делу, в общей сложности было найдено 81 исследование, которое было прочитано целиком для оценки соответствия критериям

Из этих 81 исследований 42 были исключены из-за дисквалификации методологий или недостаточного представления результатов. Таким образом, в общей сложности осталось 39 [13].

Из 39 исследований было отобрано 6 работ согласно следующим критериям in vivo, точность больше 80%, проведено не раньше 2016г. Данные критерии были выбраны исходя из специфики проводимого исследования.

В исследование [31] сказано, что их CNN обошел в классификации гистопатологических изображений меланом 11 гистопатологов. Для этого исследования было классифицировано 695 поражений экспертом гистопатологом используя окрашенные стекла. Этот набор изображений был разделен на обучающею и тестовую выборку на 595 и 100 изображений соответственно для обучения и тестирования алгоритма. Далее были разосланы по 100 изображений дерматологом, которые показали результаты в среднем чувствительностью 74.1% и специфичности 60%. CNN имела результат при чувствительность 74.1% в средним специфичность 86.5%. Всего участвовало 157 дерматологов с разным опытом работы. Главные врачи из этих участников имели среднею специфичность 69.2% и среднею чувствительность 73.3%. При такой же специфичности CNN показала чувствительность 84.5% [31].

В исследование [30] указывают на несколько проблем в исследование [31]. Первое, дерматологом давалось по 12 секунд на оценку одного изображение, чего в реальных условиях не практикуют. Дерматологи указали, что сильное влияние имеет качества и ясности клинической информации.

Второе, это то, что набор данных для алгоритма классифицировал один гистопатолог. Как сказано в [34] расхождение в решение между врачами может составлять 25-26%. То есть, в определении изображений для алгоритма должно участвовать не менее трех специалистов.

Третье, это то, что изображения были обрезаны на случайные части поражений, а на практике врачи выносят диагноз на полном изображении поражения. В [30] указано еще на то, что 15% изображений не имеют узнаваемого меланоцитарного поражения.

Большая часть исследований основываются на использовании цифровых фотографий, часто дополненных данными дерматоскопа. Учитывая почти повсеместное использование цифровых камер и дерматоскопов в дерматологической практике, алгоритмы ML на основе изображений обладают большими шансами быть реализованными в медицинских учреждениях.

Но, методы оптической спектроскопии имеют больший потенциал получения высокой точности дифференциации патологий. Так как анализ опухоли проводится с учетом химического состава, то поиск характерных маркеров, характеризующих патологию позволяет с большей точностью определять диагноз пациента. Использование машинного обучения дает шанс создать классификатор способный регистрировать скрытые зависимости и извлекать шаблоны, отвечающие за классификацию кожно-раковых заболеваний, позволяя увеличить точность распознавании патологий, сравнительно с других неинвазивных методов.

Использование данной методики в промышленном масштабе является актуальной проблемой. Стоимость оборудования для проведения спектрального анализа начинается от 500 тысяч рублей, что является значительной суммой для государственных больниц.

Потенциальными пользователями программного продукта могут стать исследовательских центры и онкологические диспансеры, оснащенные данным оборудованием. Так же реализации программного обеспечения может осуществлять совместно с поставщиками оборудования для спектрального анализа.

В исследовании рассматривается проблема неинвазивной диагностики раковых заболеваний тканей кожи человека.

Проводится анализ результатов спектроскопии комбинационного рассеивания, для выявления признаков дифференциации кожных патологий с использованием различных алгоритмов машинного обучения.

Исходя из объекта и предмета исследования для достижения поставленной цели были определены следующие задачи:

- Анализ предметной области, изучение литературы сравнение существующих решений дифференциации патологий.
- Анализ набора данных представленного ГБУЗ Самарским областным клиническим онкологическим диспансером.
- Анализ способов предобработки обучающих данных.
- Анализ алгоритмов машинного обучения, и методов их оптимизации.
- Составление математической модели системы обучения дифференциации патологий.
- Предобработка обучающих данных и создание нескольких наборов данных.
- Программная реализация разработанной математической модели системы обучения дифференциации патологий.
- Тестирование работы программного кода.
- Анализ работы реализованной математической модели.

Выводы к главе 1

В первой разделе выпускной квалификационной работы была обоснована актуальность рассматриваемой темы. Рак кожи является самым распространенным раковым заболеванием. Темпы заболеваемости, которым от года к году лишь увеличиваются. Так же по данным ВОЗ к 2025 году ожидается прирост заболеваний меланомы кожи на 25% [24]. Потому исследования в данной области на сегодняшний день являются как никогда актуальными.

Базируясь на методах спектрального анализа, мы неминуемо проигрываем в удобстве в сравнении с исследованиями, основывающими свои работы на изучении фотографий, подкрепленных дермоскопией. Но методы оптической спектроскопии имеют больший потенциал получения высокой точности дифференциации патологий.

Научные исследование основанные на материалах ГБУЗ Самарского областного клинического онкологического диспансера показывают эффективность математических методов при работе со спектрами. Позволяя выставить гипотезу эффективности использования алгоритмов машинного обучения для задачи дифференциации патологий.

Обучая классификаторы на спектральных характеристиках патологии существует вероятность безошибочного определения исследуемых образца с помощь алгоритмов машинного обучения. Созданные на их основе классификаторы обладают потенциалом регистрировать скрытые зависимости, извлекать шаблоны, отвечающие за классификацию кожно-раковых заболеваний, позволяя увеличить точность в распознавании патологий.

Глава 2 Поиск решения задачи дифференциации кожных патологий с использованием алгоритмов машинного обучения

2.1 Поиск оптимального подхода к дифференциации патологий по спектрам комбинационного рассеивания

Машинного обучение можно разделить на три большие категории:

- контролируемое обучение (обучение с учителем);
- неконтролируемое обучение (обучение без учителя);
- обучение с подкреплением.

Контролируемое обучение требует, чтобы информация для обучения была представлена в виде набора неких входных данных и соответствующих им меток – выходных данных.

Разберем процесс обучения алгоритма, решающего задачу классификации пигментированных патологией на два класса «Меланома», «Доброкачественное новообразование». В данном случае входными данными будет являться информация характеризующая пигментированное поражение (фотографии, спектрограммы, данные полученные дерматоскопом).

Метками, буду соответствующие каждому образцу категориальные данные содержащие информацию о принадлежности образца к классам «Меланома» «Доброкачественное новообразование».

Первоначально алгоритм тренируется с данными с заранее известными метками. Затем компьютер обобщает эту информацию, создавая таким образом классификатор пригодный для работы с ранее невиданным им набором данных.

Обучение с учителем является наиболее распространенным типом обучения, используемым в дерматологии. Составленные врачами Анамнезы пациентов группируются им присваиваются метки – диагнозы. Составляются базы данных содержащие в себе совокупности различных признаков начиная от места работы заканчивая наследственностью. Грамотная работа по

составлению и организации баз данных пациентов является основным подспорьем развития машинного обучения в медицинской сфере.

Для обучения без учителя требуются лишь входные данные. Данные в которых отсутствуют или намеренно изъяты метки классов. В случае данного подхода алгоритм может идентифицировать неизвестные кластеры или аномалии в наборах данных. Помимо прямого назначения, слепого разделения данных на подгруппы, данный алгоритм применяется для прореживания данных – поиска в наборах плохо классифицируемых образцов. Не корректное снятые показаний, уникальные единичные случаи, выпадающие из общей структуры образцы, все это являются аномалиями требуют дополнительного ручного изучения.

Обучение с подкреплением представляет собой гибрид обучения с учителем и без учителя, при котором система учится методом проб и ошибок получая отклик от среды. Примером обучения с подкреплением является алгоритм AlphaGo разработанная компанией Google DeepMind. Использование данного подхода в медицине, в частности в дермоскопии только предстоит изучить.

ГБУЗ Самарским областным клиническим онкологическим диспансером был предоставил набор данных содержащий спектрограммы патологий и образцов кожи (Таблица 2). Каждая спектрограмма (Рисунок 8) содержит метки класса характеризующие каждый образец. Всего 137 образцов.

Таблица 2 – Набор обучающих данных

Наименование патологии	Количество	Всего в наборе
Nevus	33	76
Melanoma	32	
Melanocytic dysplasia	11	
Normal skin	61	61

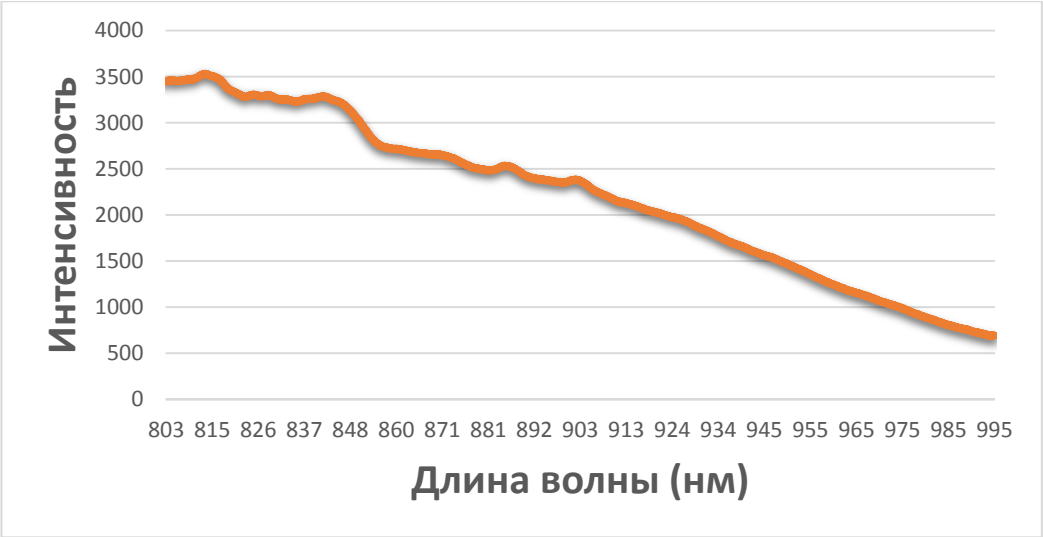

Рисунок 8 – Пример образца из обучающего набора

Был проведен анализ данных представленных в таблице 2, для выявления основных этапов предварительной обработки.

2.2 Составление математической модели

Для качественной работы с набором данных процесс разработки был поделен на несколько этапов:

- работа с набором данных (препроцессинг, предобработка) – подразумевается процесс создания альтернативного набора данных. Результат: наборы данных пригодные для обучения классификаторов;
- обучение классификаторов – использования набора данных для обучения базовых моделей. Результат: набор базовых классификаторов;
- усиление классификаторов – составление ансамблей классификаторов на основе базовых моделей. Результат: новый классификатор объединяющий базовые классификаторы.

Задача математической модели – получение ансамбля сильных классификаторов обученных на наборе спектров комбинационного рассеивания.

Формулирование математической модели:

В математической модели классификаторы и алгоритмы их построения, имеют вид функции принимающей в качестве аргумента обучающие данные.

Предобработка базового обучающего набора данных:

$$preprocessing\_data = preprocessing(data), \quad (1)$$

где $preprocessing\_data$ – предобработанные данные, $preprocessing$ – функция предобработки, $data$ – обучающий набор.

Классификаторы обучаются на основе предобработанных данных и на их основе создается ансамбль классификаторов:

$$classifers = \text{ensemble}(\bigcup_{n=1}^{m}(F1\_score(MLA_n (preprocessing\_data)) \geq limit)), \quad (2)$$

где $classifers$ – множество обученных классификаторов, MLA – алгоритм машинного обучения, m – количество алгоритмов машинного обучения, $F1\_score$ – метрика оценки классификатора, $limit$ – предел точность.

Для иллюстрации работы математической модели формируется псевдокод:

Листинг 1 – Псевдокод математической модели обучения системы дифференциации патологий

```
ИТЕРАТОР по алгоритмам машинного обучения:
    классификатор = обучения классификатора (обучающий набор)
    добавления классификатора в список(классификатор)
предел_точности = 90
ЦИКЛ_1 пока правда:
    ИТЕРАТОР по списку классификаторов:
        Вычисление точности классификатора
        ЕСЛИ точность классификатора < предел_точности:
            Добавляем классификатор в отдельный список
    ЕСЛИ список классификаторов больше> =2:
```

Составляем ансамбль из классификаторов в списке.

Выход из ЦИКЛ_1

ИНАЧЕ:

уменьшаем предел точность и проводим проверку классификаторов снова

сохраняем обученные классификаторы

Для разработки программного кода, реализующего функционал составленной математической модели, формула 2 и псевдокода, листинг 1, рассмотрены различные способы предобработки набора данных спектров комбинационного и алгоритмы машинного обучения для последующего составления ансамблей алгоритмов.

2.3 Методы предварительной обработки обучающих данных

Опираясь на исследования характерного для опухолей химического состава в спектре был выделен информативный диапазон 803.01-945 нм, исключающий часть спектра с индуцированным лазером пиком [37].

Для выполнения второго шага сглаживания выбран метод фильтрации Савицкого-Голея. Данный алгоритм был выбран из-за его повсеместного успешного применения в работе со спектрами. Считается, что он эффективно устраняет влияние шума, не нарушая диапазона чувствительности [7].

Метод фильтрации Савицкого-Голея, вместо линейного приближения вблизи каждой точки измерения строится аппроксимирующий полином n-го порядка с использованием метода наименьших квадратов. Значение чувствительности для данного волнового числа k заменяется полиномиальным значением в этой точке [7].

В расчетах используется интервал усреднения (окно сглаживания), который включает по $m = (n–1)/2$ точек слева и справа от текущего значения волнового числа. Коэффициенты такого полинома не зависят от начального

набора значений чувствительности МФП, а определяются только размером сглаживающего окна и порядком аппроксимационного полинома.

Задача фильтрации сводится к вычислению коэффициента вектора $a = \{a_0, a_1, \ldots, a_n\}$ используя критерий минимальной среднеквадратичной оценки согласно матричному расчету (формула 3).

$$\left(Z^T Z\right)^{-1} \cdot a = -Z^T \cdot sx, \tag{3}$$

где sx – вектор измерения, Z – матрица Вандермонда с размерностью $(n + 1)\cdot(2m + 1)$, m – ширина окна сглаживания.

Из этого матричного уравнения коэффициент a_i ($i = 0\ldots n$) вычисляется как результат умножения i-й строки матрицы – $(A^T A)^{-1} A^T$ на вектор измерения. Сглаженное значение функции спектральной характеристики или ее производных t-го порядка определяется в виде свертки (формула 4).

$$sg^{(t)} = t! \sum_{i=-m}^{m} a_i SX_i. \tag{4}$$

При малых значениях n коэффициенты могут быть рассчитаны аналитически, но если $n > 3$ (аппроксимирующие полиномы высоких степеней) коэффициенты фильтра определяются с помощью матричного вычисления на основе стандартных процедур вычисления линейной алгебры [34].

Второй шаг – стандартизация спектров.

Стандартизация подразумевает предобработку данных результатом которой является спектр, у которого каждый признак имеет среднее значение 0 и дисперсию 1.

В качестве алгоритма стандартизации использовался метод скалярной стандартизации. Где из каждой точки вычитается среднее значение спектра, и все делится на стандартное отклонение (формула 5):

$$Z = \frac{x - \frac{1}{N}\sum_{i=1}^{n}(x_i)}{\sqrt{\frac{1}{N}\sum_{i=1}^{n}\left(x_i - \frac{1}{N}\sum_{i=1}^{n}(x_i)\right)}}, \quad (5)$$

где N – количество элементов в спектре, x_i – i-й элемент спектра.

При работе с данными высокой размерности, велика вероятность, что признаки сильно зависят друг от друга и их одновременное наличие может быть избыточно. Подразумевается, что некое свойство может характеризоваться совокупностью признаков, и выражая совокупность через новый признак уменьшается размерность исходного набора формируя новый набор признаков.

2.4 Метод понижения размерности

Рассматривается метод главных компонент (principal component analysis, PCA).

Пусть имеется матрица переменных X размерностью $(I \times J)$, где I – число образцов (строк), а J – это число независимых переменных (столбцов), которых, как правило, много (J>>1). В методе главных компонент используются новые, формальные переменные t_a (a=1,…A), являющиеся линейной комбинацией исходных переменных x_j (j=1,…J) (формула 6).

$$\mathrm{ta} = \mathrm{pa1x1} + \cdots + \mathrm{paJxJ} \quad (6)$$

С помощью этих новых переменных матрица X разлагается в произведение двух матриц T и P (формула 7).

$$X = TP^t + E = \sum_{a=1}^{A} t_a p_a^t + E \quad (7)$$

Матрица T называется матрицей счетов (scores). Ее размерность (I×A). Матрица P называется матрицей нагрузок (loadings). Ее размерность (J×A). E – это матрица остатков, размерностью (I×J) (рисунок 9).

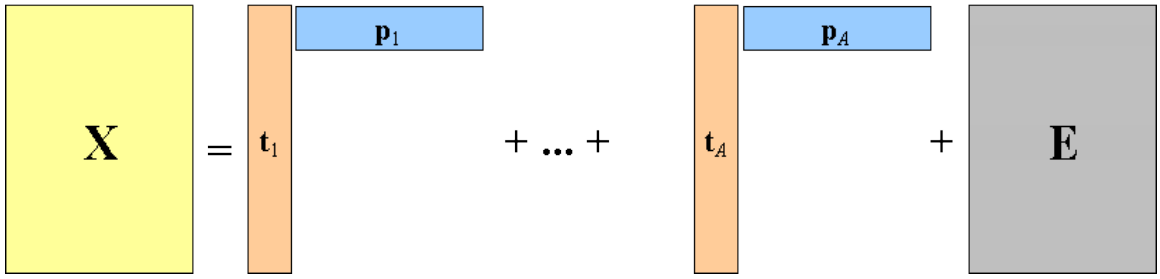

Рисунок 9 – Разложение по главным компонентам

Новые переменные t_a называются главными компонентами (Principal Components), поэтому и сам метод называется методом главных компонент (PCA). Число столбцов – t_a в матрице T, и p_a в матрице P, равно A, которое называется числом главных компонент (PC). Эта величина заведомо меньше числа переменных J и числа образцов I.

Важным свойством PCA является ортогональность (независимость) главных компонент. Поэтому матрица счетов T не перестраивается при увеличении числа компонент, а к ней просто прибавляется еще один столбец – соответствующий новому направлению. Тоже происходит и с матрицей нагрузок P.

Важным критерием использования PCA является поиск оптимального значения характеризующее количество признаков до которого алгоритм уменьшает исходный вектор признаком, иначе методов может существенно снизить точность классификации данных.

2.5 Методы машинного обучения

Исходя из специфики набора данных, а именно деление на классы, для решения задачи классификации подойдут алгоритмы работающие по принципу обучения с учителем. К ним относятся k-ближайший сосед (kNN), методы опорных векторов (SVM), деревья принятия решений, нейронные сети и другие.

Производится анализ алгоритмов машинного обучения и рассмотрение их математические модели. Так как решаемая задача является задачей бинарной классификации, работа алгоритмов машинного обучения с мультиклассовыми данными рассматриваться не будет.

Использование принципов глубокого обучения (deep learning) не оправданно в контексте малого (меньше 1 тыс. образцов на класс) обучающего набора. Проведенное в бакалаврской работе исследование показало не эффективность данного подхода при малом количестве данных [10].

2.5.1 Метод «Логическая регрессия»

Логистическая регрессия (Logistic Regression, LR) – это алгоритм классификации, используемая для предсказания вероятности принадлежности входных данных с помощью логистической функции. Логистическая регрессия преобразует выходные данные с использованием функции логистической сигмоиды, возвращая вероятностное значение принадлежности входных данных к одному или более классам.

Основная идея логистической регрессии состоит в том, что пространство начальных значений может быть разделено некой границе, линией в двумерном пространстве, плоскостью в трехмерном и так далее, на области, соответствующих классов (Рисунок 2.3).

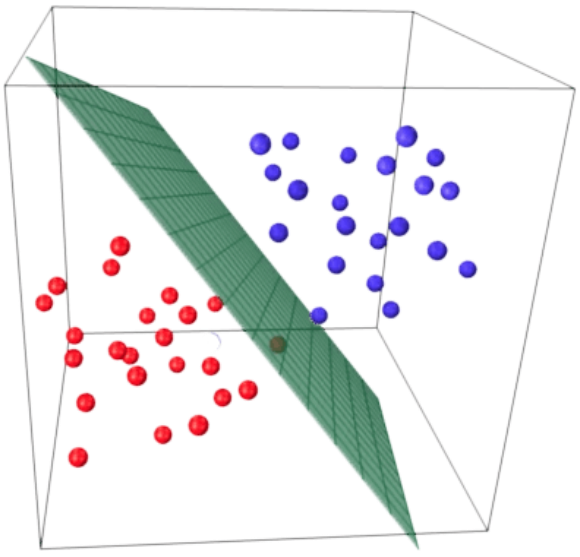

Рисунок 10 – Линейно разделимые объекты

Данная плоскость – линейным дискриминант, является линейной с точки зрения своей функции, позволяет классификатору производить разделение(дискриминацию) точек на разные классы. Для каждого нового объекта рассчитывается вероятностная оценка принадлежности ко всем областям для которых был обучен классификатор и в зависимости от результата присваивается соответствующая метка класса.

Дан набор данных (X, Y), где X – матрица с n признаками и m объектами, а Y – вектор с m объектами.

Задача – обучить модель предсказывать класс новых объектов.

В первую очередь, создается матрицу весов со случайной инициализацией. Затем умножается на количество признаков (формула 8).

$$a = w_0 + w_1 x_1 + w_2 x_2 + \ldots w_n x_n . \quad (8)$$

Далее, результат из формулы 8 передается в функцию связи (формула 9):

$$\hat{yi} = 1/\left(1 + e^{-a}\right). \quad (9)$$

Рассчитывается весовой коэффициент итерации по формуле 10:

$$\cos t(w) = (-1/m)\sum_{i=1}^{i=m} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i). \quad (10)$$

Берется производная от этого коэффициента и получен градиент весов (формула 11):

$$dw_j = \sum_{i=1}^{i=n} (\hat{y} - y) x_j^i . \quad (11)$$

После чего обновляются веса (формула 12):

$$w_i = w_j - \left(\alpha \cdot dw_j\right). \quad (12)$$

Логистическая регрессия применяется для решения задач, связанных с клиническими исследования в медицинской сфере, в банковском скоринге для оценки коэффициента доверия заемщиков, исследовании поведения покупателей в сферах связанных с маркетингом а так же для оценки кредитных рисков.

2.5.2 Метод опорных векторов

Метод опорных векторов (англ. Support Vector Machines, SVM) — это модель машинного обучения, способная выполнять линейную или нелинейную классификацию, регрессию и даже выявление выбросов.

Она является одной из самых популярных моделей в машинном обучении. Методы SVM особенно эффективны при классификации сложных, но относительно небольших наборов данных.

Свойства SVM

- возможность выбора различных функций близости (Ядер);
- разреженность решения при работе с большими объемами обучающих данных;
- переобучение может контролироваться использованием штрафа;
- выпуклая оптимизационная задача, гарантировано сходится к одному глобальному минимуму;
- возможен отбор значащих для распознавания переменных.

Модель линейной классификации SVM прогнозирует класс нового образца x, вычисляя функцию решения $w^T * x + b = w_1 x_1 + \cdots + w_n x_n + b$: если результат положительный, то спрогнозированный класс является положительным (1), а иначе – отрицательным (0), подробнее показано в уравнение 2.13 – Прогноз линейного классификатора SVM:

$$y = \begin{cases} 0, \text{если } w^T * x + b < 0 \\ 1, \text{если } w^T * x + b \geq 0 \end{cases}, \quad (13)$$

где b – член смещения, w – вектор весов признаков.

Обучение линейного классификатора SVM означает нахождение таких значений w и b, которые делают этот зазор как можно более широким, одновременно избегая нарушений зазора (жесткий зазор) или ограничивая их (мягкий зазор).

Рассматривается наклон функции: он тождественен норме вектора весов, $||\ w\ ||$. Если мы разделим наклон на 2, тогда точки, в которых функция решения равна ± 1, будут в два раза дальше от границы решений. Другими словами, деление наклона на 2 умножит зазор на 2. Это легче представить в двумерном виде на рисунке 11. Чем меньше вектор весов w, тем шире зазор.

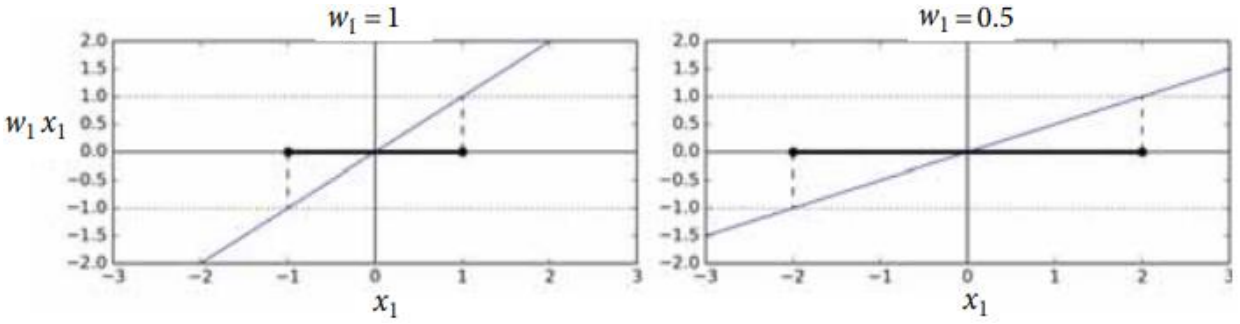

Рисунок 11 – Меньший вектор весов приводит к более широкому зазору

Необходимо довести до максимума $||w||$, чтобы получить широкий зазор. Однако также для избежания любых нарушений зазора (иметь жесткий зазор) нужно, чтобы функция решения была больше 1 для всех положительных обучающих образцов и меньше -1 для отрицательных обучающих образцов. Если мы определим $t(i) = -1$ для отрицательных образцов (когда $y(i) = 0$) и $t(i) = 1$ для положительных образцов (когда $y(i) = 1$), тогда можно выразить такое ограничение, как $t^{(i)}\ (w^T \cdot x^{(i)} + b) \geq 1$ для всех образцов. Следовательно, можно выразить цель линейного классификатора SVM с жестким зазором как задачу условной оптимизации (constrained optimization) в 7. Цель линейного классификатора SVM с

жестким зазором минимизировать $w, b,$ при условии, продемонстрированном в формуле 14.

$$\frac{1}{2} w^T * w$$
$$t^{(i)} * \left(w^T * x^{(i)} + b\right) \geq 1 \text{ для } i = 1,2, \dots m, \tag{14}$$

где b – член смещения, w – вектор весов признаков.

Чтобы достичь цели мягкого зазора, необходимо ввести фиктивную переменную $\zeta\ (i)\ \geq\ 0$ для каждого образца: $\zeta(i)$ измеряет, насколько i-тому образцу разрешено нарушать зазор.

Теперь имеются две противоречивые цели: делать фиктивные переменные, как можно меньшими, чтобы сократить нарушения зазора, и делать, как можно меньшим, чтобы расширить зазор. Именно здесь в игру вступает гиперпараметр C: он позволяет определить компромисс между указанными двумя целями [6].

2.5.3 Деревья принятия решений

Деревья решений (DT) – это непараметрический метод машинного обучения с учителем, используемый для задач классификации и регрессии. Данная модель представляет собой древовидную структуру, похожую на блок-схему (рисунок 12), где каждый внутренний узел представляет собой тест по проверке входного атрибута, каждая ветвь представляет результат данного теста.

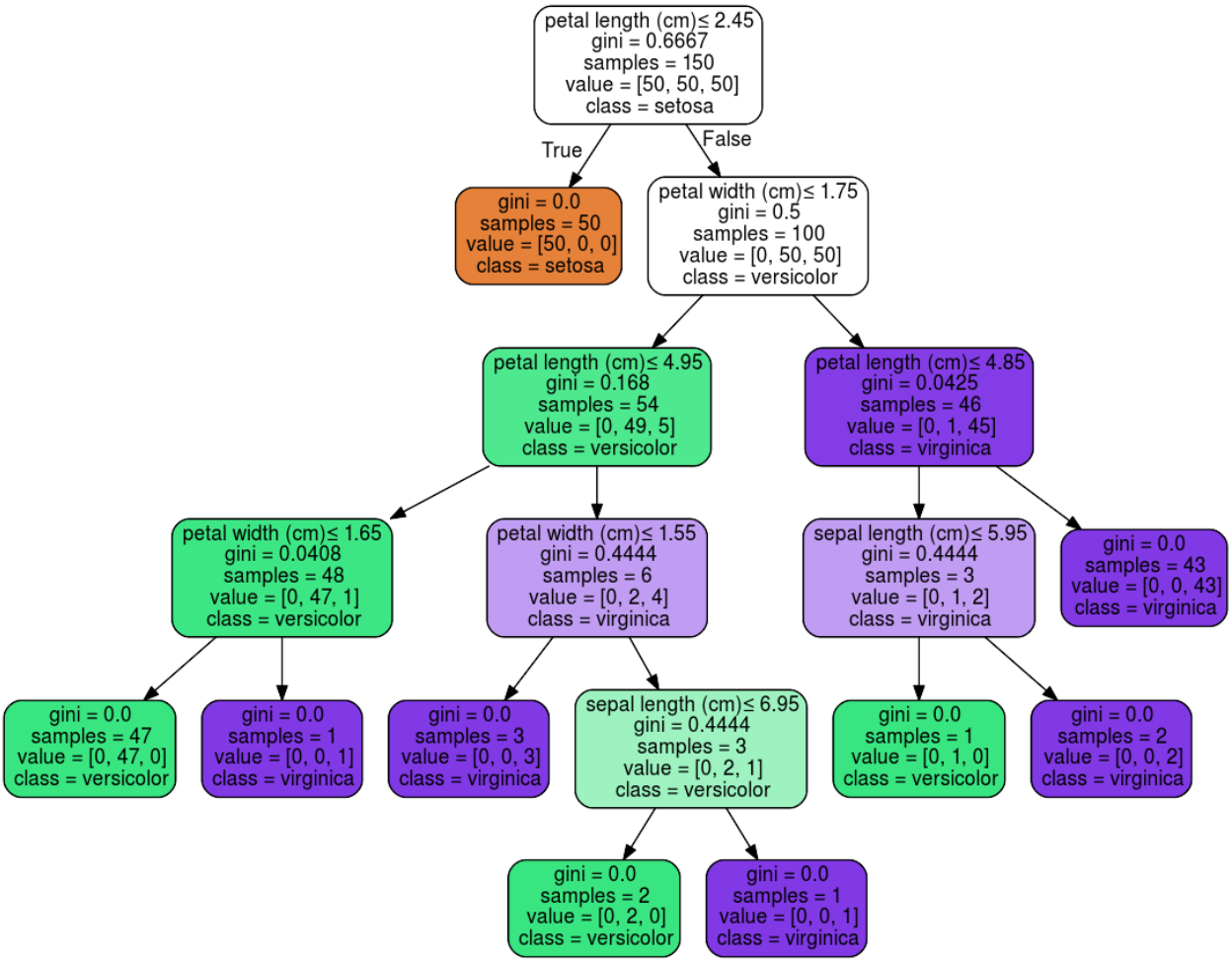

Рисунок 12 – Пример визуализация дерева принятия решений

Преимущества деревьев принятия решений:

- алгоритм легко понять и интерпретировать;
- использует модель белого ящика – каждое условие может быть интерпретировано и доказано по каким параметрам оно помещено в узел дерева;
- не требует больших объемы обучающих данных;
- устойчив к выбросам;
- способен обрабатывать как числовые, так и категориальные данные;
- обладает высокая скоростью работы.

В качестве недостатков деревьев решений стоит выделить следующие моменты:

- переоснащение – большая размерность входных данных может создать излишне сложные деревья;
- деревья решений могут быть нестабильными, поскольку небольшие изменения в обучающем наборе данных могут привести к созданию совершенно другого классификатора;
- плохо работают с несбалансированными наборами данных [16].

Рассматривается алгоритм построения дерева принятия решения – CART (формула 15).

$$Gini(T) = 1 - \sum_{i=1}^{n} p_i^2, \tag{15}$$

где $Gini(T)$ – индекс оценки разбиения,

p_i – вероятность (относительная частота) класса i в T,

T – набор данных.

Если набор T разбивается на две части T_1 иT_2 с числом примеров в каждом N_1 и N_2 соответственно, тогда показатель качества разбиения будет равен (формула 16):

$$Gini_{split}(T) = \frac{N_1}{N} \cdot Gini(T_1) + \frac{N_2}{N} \cdot Gini(T_2). \tag{16}$$

Оценка качества разбиения производится по формуле 17:

$$Gini_{split} = \frac{L}{N} \cdot \left(1 - \sum_{i=1}^{n} \left(\frac{l_i}{L}\right)^2\right) + \frac{R}{N} \cdot \left(1 - \sum_{i=1}^{n} \left(\frac{r_i}{R}\right)^2\right) \rightarrow min, \tag{17}$$

где N – число примеров в узле – предке,

L и R – число примеров соответственно в левом и правом потомке,

l_i и r_i – число экземпляров i-го класса в левом/правом потомке.

Пример программной реализации дерева принятия решения алгоритмом CART на языке C++ доступен в репозитории на GitHab[27].

2.5.4 Метод «k-ближайший сосед»

Метод ближайших соседей – это метод классификации объектов основанный на поиске сходств признаков из которых состоит объект. Схожие объекты группируются, а наиболее отличные объекты выбиваются из группировки, располагаясь наиболее удаленно от скопления схожих объектов (Рисунок 13). Таким образом, расстояние между двумя объектами является мерой их различия.

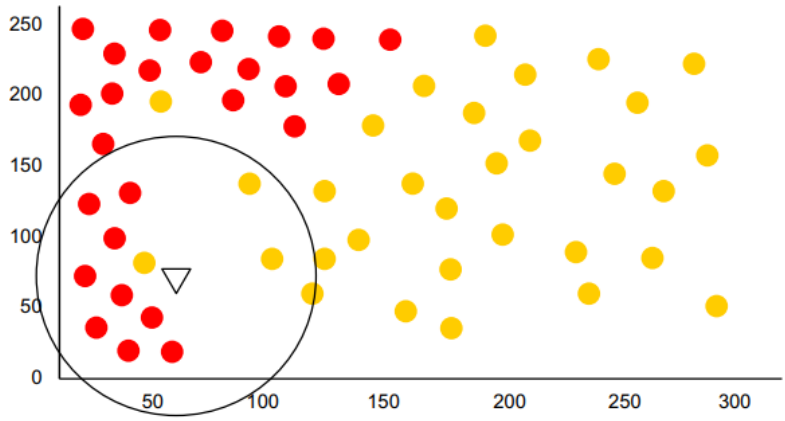

Рисунок 13 – Визуализация работы алгоритма «Ближайшего соседа»

Объекты, которые находятся рядом друг с другом, называются «соседями». Когда на вход подается новый объект, вычисляется его расстояние до каждого из объектов в модели. Классификации наиболее похожих случаев - ближайших соседей - подсчитываются, и новый случай помещается в категорию, которая содержит наибольшее количество ближайших соседей.

Рассматривается работа классификатора на задаче бинарной классификации кожа/патология по спектрограмме.

Имеется m образцов. Каждому образцу соответствует метка класса характеризующая его принадлежность к коже/патологии. Необходимо определить класс для нового образца. Входными данными является вектор интенсивности. Все значения являются числовые и равнозначными.

На первом шаге алгоритма следует задать количество ближайших соседей(k).

Коэффициент k показывает сколько «ближайших соседей» стоит брать во внимание при вынесении конечно решения о принадлежности нового образца к определенному классу.

Если задать $k = 1$, то алгоритм потеряет обобщающую способность так как новой записи будет присвоен класс самой близкой к ней. Если установить слишком большое значение, то многие локальные особенности не будут выявлены.

На втором шаге находятся k записей с минимальным расстоянием до вектора признаков нового образца патологии.

Для упорядоченных значений атрибутов находится Евклидово расстояние (формула 18):

$$D_E = \sqrt{\sum_i^n \left(x_i - y_i\right)^2}, \tag{18}$$

где n – количество атрибутов.

Функция для расчета расстояния должна отвечать следующим правилам:

- $d(x,y) \geq 0$, $d(x,y) = 0$ тогда и только тогда, когда $x = y$;
- $d(x,y) = d(y,x)$;
- $d(x,z) \leq d(x,y) + d(y,z)$, при условии, что точки **x**, **y**, **z** не лежат на одной прямой.

Где x, y, z – векторы признаков образца кожи.

Для последующей работы с векотором необходимо провести нормализацию датасета выборки по формуле 19.

$$X_n = \frac{X - min(X)}{max(X) - min(X)}, \tag{19}$$

где X – ненормализованный элемент, min(X)-минимальных элемент выборки, max(X) – максимальный элемент выборки, X_n нормализованное значение.

На следующем шаге, когда найдены записи, наиболее похожие на новую, необходимо решить, как они влияют на класс новой записи. Для этого используется функция сочетания (combination function). Одним из основных вариантов такой функции является простое невзвешенное голосование (simple unweighted voting).

В такой ситуации учитывается также и расстояние до новой записи. Чем меньше расстояние, тем более значимый вклад вносит голос. Голоса за класс находятся по следующей формуле 20:

$$votes(class) = \sum_{i=1}^{n} \frac{1}{d^2(X, Y_i)}, \quad (20)$$

где $d^2(X, Y_i)$ – квадрат расстояния от известной записи Y_i до новой X, n – количество измщвестных записей класса, для которого рассчитываются голоса, *class* – наименование класса.

Класс, набравший наибольшее количество голосов, присуждается новой записи [23].

2.5.5 Наивный байесовский алгоритм

Наивный байесовский алгоритм (НБА) – это алгоритм машинного обучения, основанный на теореме Байеса с допущением о независимости признаков. Имеется ввиду что классификатор предполагает, что наличие какого-либо признака в классе не связано с наличием какого-либо другого признака. Толстеть они независимы и существуют отдельно друг от друга.

К примеру, патология может считаться меланомой, если она находится на коже, имеет неровные очертания и в диаметре составляет около 5 миллиметров. Даже если эти признаки зависят друг от друга или от других признаков, в любом случае они считаются равнозначными при вынесении окончательного результата.

Приведена теорема Байеса:

Пусть $H_1, H_2 \ldots H_n$ — полная группа событий, и A — некоторое событие, вероятность которого положительна. Тогда условная вероятность того, что имело место событие H_k, если в результате эксперимента наблюдалось событие A, может быть вычислена по формуле 21:

$$\mathrm{P}(H_k \mid A) = \frac{\mathrm{P}(H_k)\mathrm{P}(A \mid H_k)}{\sum_{i=1}^{\infty} \mathrm{P}(H_i)\mathrm{P}(A \mid H_i)}, \tag{21}$$

где H_i – полная группа событий, H_k – произошедшее событие, A – некоторое событие вероятность которого положительна, P – условная вероятность [29].

Модели на ее основе просты, но крайне эффективны при работе с большими наборами данных. При своей простоте НБА способен превзойти даже некоторые сложные алгоритмы классификации.

Теорема Байеса позволяет рассчитать апостериорную вероятность $P(c|x)$ на основе $P(c)$, $P(x)$ и $P(x|c)$ (формула 22, 23).

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}, \tag{22}$$

$$P(c|X) = P(x_1|c) * P(x_2|c) * \ldots * P(x_n|c) * P(c), \tag{23}$$

где $P(c|x)$ – апостериорная вероятность данного класса c (т.е. данного значения целевой переменной) при данном значении признака x, P(c) – априорная вероятность данного класса, $P(x|c)$ – правдоподобие, т.е. вероятность данного значения признака при данном классе, $P(x)$ – априорная вероятность данного значения признака.

2.6 Ансамблирование классификаторов

После обучения всех классификаторов, можно сделать допущение, что каждый из алгоритмов ввиду своей уникальной архитектуры реагирует на разные скрытые зависимости, присутствующие в спектрах. В таком случае эффективным способом увеличения точно работы может стать построение композиций алгоритмов – ансамблирование.

Стекинг (Stacked Generalization или Stacking) – объединение нескольких алгоритмов в классификатор решение которого основывается на усреднении или голосованию большинства.

Рассматривается алгоритм мягкого голосования.

Идея мягкого голосования заключается в объединении различные классификаторов машинного обучения используя средние прогнозируемые вероятности классификации. Каждый входящий в ансамбль классификатор, возвращает вероятность принадлежности входных данных к классу. И итоговая метка высчитывается как усредненное произведение полученной вероятности и веса классификатора (формула 24) [9].

$$\hat{y} = argmax \sum_{j=1}^{m} w_j p_{ij}, \tag{24}$$

где $\hat{y}$ – метка класса, w_j – вес классификатора, p_{ij} – предсказание классификатора.

Основной проблемой использование других типов объединения алгоритмов является необходимость использования для каждого обученного классификатора свою подвыборку, что критично при работе с небольшими обучающими наборами. Данное условие позволяет избежать возможности переобучения ансамбля алгоритмов. При работе со стекингом достаточным условием является деление выборки на две части: обучающую и тестовую

где обучающая выборка используется при обучении всех классификаторов ансамбля.

2.7 Оценка качества работы классификаторов

Для оценки качества работы классификаторов используются метрики. Значения метрик показывают характеристики обученной модели. Для оценки качества работы классификатора используются следующие метрики: *Accuracy*, F-мера, истинно положительный показатель (TPR, Recall, чувствительность), ложноположительный показатель (FPR, специфичность, Precision).

Вводятся некоторые обозначения:

- если прогноз и фактическое значение класса совпадают, то результат работы классификатора является истинно положительным (True Positive, TP);
- если прогноз указал на непринадлежность образца к классу, то результат является истинно отрицательным (True Negative, TN);
- если прогнозное указал на принадлежность к неверному классу, то результат является ложно положительным (False Positive, FP);
- если прогнозное значение отрицательно, а фактическое значение положительное, то результат является ложно отрицательным (False Negative, FN) [36].

Приводятся формулы для вычисления всех используемых метрик.

Метрика Accuracy не эффективна при работе с неравномерно распределенными классами. Количество элементов в классе влияет на оценку точности классификатора. Формула 25 вычисления метрики Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (25)$$

Чувствительность (Recall, полнота) – показывает какую долю объектов нашел классификатор положительного класса из всех объектов этого класса (формула 26).

$$Recall = \frac{TP}{TP + FN}. \quad (26)$$

Формула вычисления специфичности (Precision, точности) – показывает правильную классификацию доли положительных объектов (формула 27)

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \quad (27)$$

F-мера — это среднее гармоническое между точностью и полнотой. Он стремится к нулю, если точность или полнота стремится к нулю [35]. Формула 28 вычисления F-меры:

$$\mathrm{F} = 2\frac{\mathrm{Precision} \times Recall}{\mathrm{Precision} + Recall}. \quad (28)$$

Для визуализации эффективности работы классификаторов рекомендуются использовать матрицы ошибок (confusion matrix) (рисунок 2.7).

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Рисунок 2.7 – Структурный вид матрицы ошибок y – это метка класса (показывает принадлежность к классу); $\hat{y}$ – ответ алгоритма (предсказание принадлежности к классу)

Оптимальным выводом матрицы ошибок является ее отображение в виде тепловой карты, где в зависимости от доли верных ответов классификатора цвет ячейки приобретает более насыщенный цвет, как показано на рисунке 14.

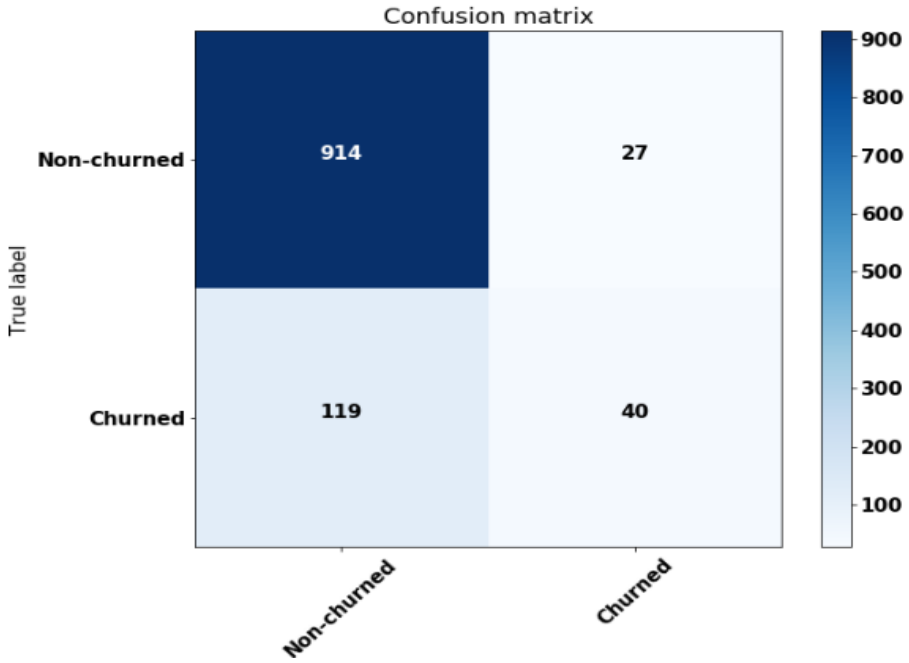

Рисунок 14 – Графический вид матрицы ошибок в виде тепловой карты

ROC-кривая (receiver operating characteristic) или кривая ошибок, AUC (area under the curve) – площадь под кривой ошибок. Пример представлен на рисунке 16. AUC ROC популярный метод метрики для задач бинарной классификации. Этот метод позволяет оценить результат работы алгоритма с помощь площади под кривой ошибок.

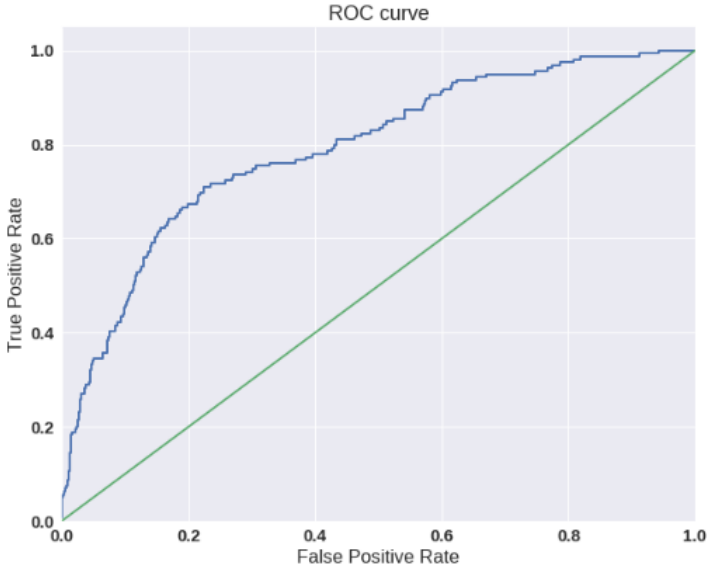

Рисунок 15 – График AUC-ROC

ROC-кривая отрисовывается следующим образом. На координатной плоскости берется единичный квадрат и делится по вертикале и горизонтали линиями, где n – число вертикальных линий, равная количеству нулей в тесте алгоритма, а m – число горизонтальных линий, равная количеству правильных меток в тесте равных 1. Кривая строится от точки (0,0) до точки (1,1). Если какое-то количество объектов имеют одинаковые оценки, то кривую строим вверх на столько блоков, сколько единиц в группе объектов с одинаковой оценкой и правее на количество нулей в этой группе. Если у всех объектов оценка одинаковая, то кривая строится прямо от точки (0,0) до точки (1,1).

Как сказано выше, рисуется сетка $m \times n$. Получается столько же пар, состоящих из объекта с меткой класса – 1 и объекта с меткой класса- 0. Под кривой каждый блок соответствует такой паре, для которой алгоритм правильно определил порядок (объект с меткой класса – 1 имеет оценку выше, чем у объекта с меткой класса 0) и обратное для блоков над кривой.

То есть, AUC ROC равен доле пар объектов с верной упорядоченностью (объект с меткой класса – 1 стоит в упорядоченном списке раньше) (формула 29).

$$\frac{\sum_{i=1}^{q}\sum_{j=1}^{q} I\left[y_i < y_j\right] I'\left[a_i < a_j\right]}{\sum_{i=1}^{q}\sum_{j=1}^{q} I\left[y_i < y_j\right]}, \tag{29}$$

$$\text{где } I'\left[a_i < a_j\right] = \begin{cases} 0,\ a_i > a_j, \\ 0.5\ a_i = a_j, \\ 1,\ a_i < a_j, \end{cases} I\left[y_i < y_j\right] = \begin{cases} 0,\ y_i \geq y_j, \\ 1,\ y_i < y_j, \end{cases}$$

где a_j – ответ алгоритма, y_i – класс, q – количество объектов.

Располагая сравнительно небольшим обучающим набором, использование общепринятой оценки классификаторов через деление выборки на тестовую и обучающую не является эффективной. Потому для каждого классификатора будет подсчитан доверительный интервал показывающий возможных распределение точности классификатора при дифференциации образцов обучающего набора. Пример доверительного представлен на рисунке 15.

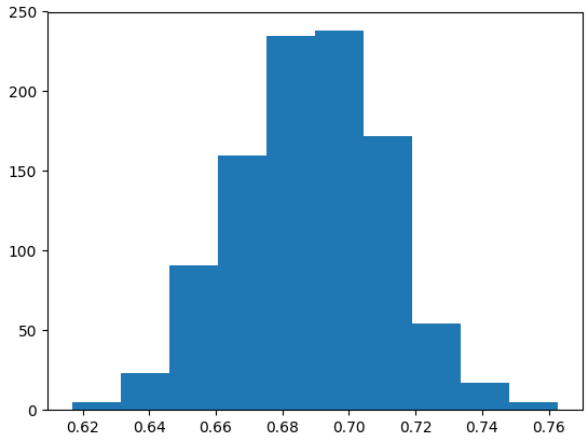

Рисунок 16 – Доверительный интервал

Для расчета доверительного интервала обучающая выборка делится N раз (в зависимости от количества циклов обучения), на обучающую и тестовую выборку в соотношении 1:1. Затем классификатор обучается N раз на обучающем наборе данных. При каждой итерации производится тестирование классификатора, значения характеристик записываются отдельно для последующего вывода в виде гистограммы.

Выводы к главе 2

Во втором разделе выпускной квалификационной работы был рассмотрен обучающий набор данных представленный ГБУЗ Самарским областным клиническим онкологическим диспансером, содержащий спектрограммы патологий и образцов кожи.

Исходя из количества исследуемых образцов датасет был поделен на два класса патология/здоровая кожа. В класс патология вошли образцы меланомы, невуса, мелоцетарной дисплозии.

Опираясь на специфику задачи составлена математическая модель обучения системы дифференциации патологий, включающая в себя работу над обучающей выборкой, обучения классификаторов, объединение классификаторов в ансамбли. Также написан псевдокод, иллюстрирующий работу математической модели.

Опираясь на исследования спектров комбинационного рассеивания были сформированы рекомендации по предобработке обучающего набора. Каждый спектр комбинационного рассеивания стоит обрезать, выделив на нем информативную часть, располагающуюся на длине волны 800-900 нм.

Так же необходимо удаление шумов. Для данной задачи обычно используется фильтр Савицкого-голея.

В качестве заключительного этапа предобработки исходные спектры должны быть обработаны алгоритмом уменьшения размерности – PCA. С помощью данного алгоритма мы сформируем отдельный набор данных и сравним эффективность его применения относительно целых спектров.

Опираясь на характеристики обучающего набора для решения задачи дифференциации было решено использовать алгоритмы машинного обучения, использующие принцип «обучение с учителем».

В главе были рассмотрены основные алгоритмы машинного обучения применяемые в данного рода задачах: k-NN, наивный байесовский алгоритм, деревья принятия решений, опорные вектора, логическая регрессия.

После обучения всех классификаторов, можно сделать допущение, что каждый из алгоритмов ввиду своей уникальной архитектуры реагирует на разные скрытые зависимости, присутствующие в спектрах. В таком случае эффективным способом увеличения точно работы может стать построение композиций алгоритмов – ансамблирование. Из-за специфики обучающего набора, а именно сравнительно малого количества образцов, объединение классификаторов осуществляется методом стекинга, а именно алгоритм взвешенное голосование. При работе с данным алгоритмом достаточным условием является деление выборки на две части: обучающую и тестовую где обучающая выборка используется при обучении всех классификаторов ансамбля.

Глава 3 Разработка системы дифференциации патологий

3.1 Общие сведения о системе дифференциации патологий

Произведено разделение системы дифференциации патологий на несколько функциональных подсистем, опираясь на логику работы с данными:

1) подсистема обучения классификаторов (ПОК);
2) подсистема работы с образцами (ПРСО).

В состав первой, входят:

- программные модули загрузки и обработки обучающего набора;
- скрипты для обучения;
- скрипты оценки классификаторов, как визуальные, так и автоматические;
- программные модули сохранения обученных классификаторов;

В состав второй системы входят:

- программный модуль обработки новых входных данных;
- программный модуль загрузки обученных классификаторов;
- база данных для хранения новых образцов патологий;
- пользовательский интерфейс для загрузки и диагностики спектров.

Произведено описание логики работы системы дифференциации патологий. ПОК и ПРНД могут функционировать независимо друг от друга. В зависимости от установленных условий (в качестве условий, администратор системы использует крон функции), происходит обучения классификаторов. Затем, в зависимости от полученных характеристик, производится сравнение нового и актуального классификаторов.

Информация об обучении доводится до администратора и им принимается решения о загрузки нового классификатора в ПРСО.

ПРНД представляет собой приложение для работы с пользователем. Загруженный в него классификатор готов работать с новыми данными. Перед загрузкой в классификатор данные пользователя проходят предобработку.

Выделены основные требование, которые необходимы для работоспособности проекта(MVP): ПОК – полная реализация; ПРСО – минимальный интерфейс, система хранения новых образцов.

Для выбора средств реализации программных модулей ПОК проведен сравнительный анализ языков программирования, наиболее используемых в машинном обучении.

Данная информация поможет понять какой из языков программирования является актуальным для решения задачи в области ML, позволит узнать уровень его поддержи сообществом и количество готовых решений (библиотек, фреймворков) которые могут быть использованы для реализации системы.

3.2 Выбор средств реализации

В январе 2019 года сервис GitHub опубликовал рейтинг самых популярных языков программирования, используемых для машинного обучения. GitHub — является авторитетным веб-сервисом для хостинга ИТ-проектов и их совместной разработки. По состоянию на август 2019 г. На сервис насчитывает порядка 100 миллионов репозетириев [13]. По информации на 2020 год сервис принадлежит компании Microsoft Corporation [17].

В тройку лидеров вошли такие языки программирования как Python – 1, C ++ – 2, JavaScript – 3. При составлении списка учитывались репозетории, авторы которых указывали что в их работах используются алгоритмы машинного обучения [18].

Не смотря на вторую строчку в списке популярности язык программирования, C++ во многих смыслах является низкоуровневым

языком программирования. Его используют для реализации алгоритмов машинного обучения, оптимизированных на низком уровне. Библиотеки на его основе имея удобный интерфейс используются различными языков программирования Matlab, Python. К примеру, на C++ написаны такие библиотеки машинного обучения как SciPy, Shogun Toolbox, Scikit-learn, TensorFlow, доступные для использования на Python [29 , 30, 31, 32].

Можно сделать вывод что, популярность Python в машинном обучении не случайна. Используя язык программирования Python в ML, мы получаем скорость языка C++, а также простоту и удобство являющиеся его главными достоинствами.

Для реализации алгоритмов машинного обучения на языке Python была выбрана библиотека Scikit-learn. Данная библиотека является самым популярным средством при работе с машинным обучением на Python. Она используется в порядок 40% всех проектов, связанных с ML [28].

Она построена на базе таких библиотеках как NumPy, SciPy и matplotlib, с поддержкой plotly, pandas и т.д. И как упоминалось выше некоторые модули написаны на языке C++ что является большим преимуществом при работе с машинным обучение ускоряя работу алгоритмов обучения и работу классификаторов.

Помимо этого, будут использоваться такие библиотеки как NumPy, Matplotlib, Pandas и т.д.

3.3 Разработка модуля предобработки данных

Обучающий набор представляет собой файл расширения xlsx, содержащий горизонтально расположенные спектры комбинационного рассеивания – значения интенсивности на длине волны 803.01 до 994,54 нм с шагом 0.21 нн. Каждый спектру присвоена метка класса (диагноз): меланоцитарная дисплозия, меланома, нормальная кожа, невус. Визуализация данных представлена на рисунке 17 и таблице 3.

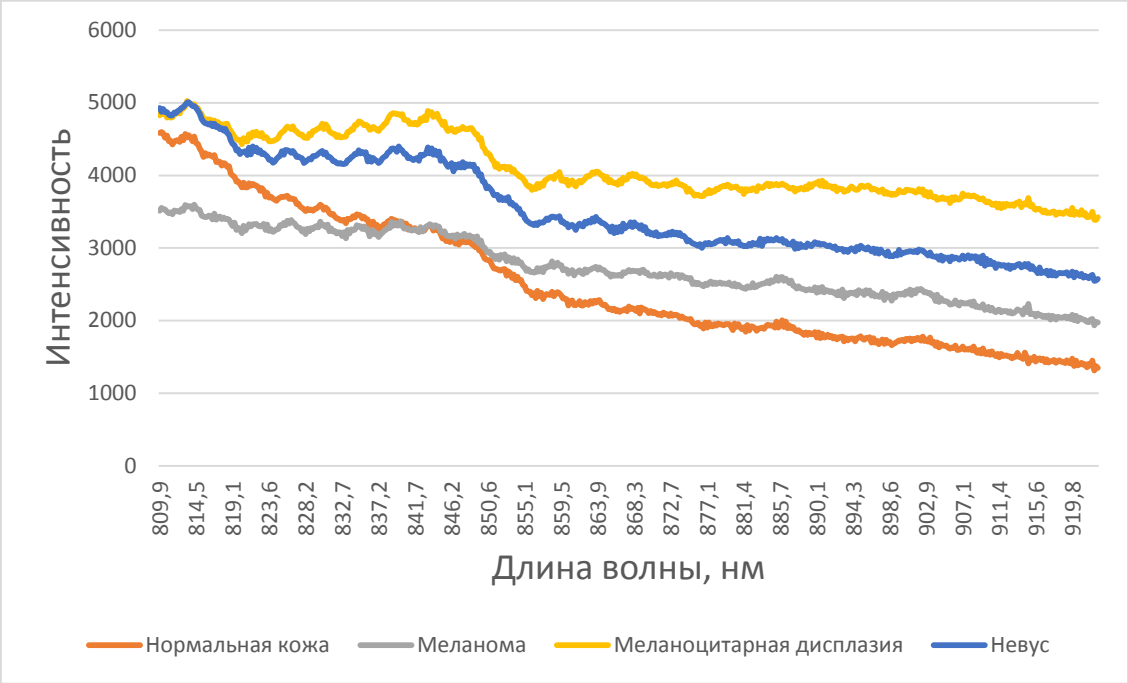

Рисунок 17 – Спектры комбинационного рассеивания образцов биоткани, с выделенной информативной частью

Таблица 3 – Пример структуры данных для обучения классификатора

Диагноз	Идентификатор пациента	809,92	803,24	806,01
нормальная кожа	125294	1454,778	1456,513	1448,116
невус	123556	11926,42	11941,08	11920,03
меланома	412312	1862,917	1865,385	1851,232
меланоцитарная диспдазия	234234	1741,999	1742,726	1761,614

Чтение и разбор файла осуществляется средствами библиотеки pandas.

Каждый спектр в наборе данных является необработанным. В разделе 2.3 было установлено что все спектры должен быть обрезаны для последующей работы только с информативной частью. Информативными являются значения интенсивности на длине волны от 809.92 до 922,72 нм.

В соответствии с задачей бинарной классификации спектры группируются по двум классам патология/здоровая кожа. Информация о группировки набора данных продемонстрирована в таблице 4.

Таблица 4 – Набор обучающих данных с указанием класса

Наименование патологии	Количество	Всего	Метка класса
Невус	33	76	1
Меланома	32		
Меланоцитарная дисплазия	11		
Нормальная кожа	61	61	0

Для выполнения второго шага сглаживания был выбран метод фильтрации Савицкого-Голея. Данный алгоритм был выбран из-за его повсеместного успешного применения в работе со спектрами, так как он эффективно устраняет влияние шума, не нарушая диапазона чувствительности [34].

Программная реализация фильтра Савицкого-Голея существует в библиотеке scipy. Из документации следует что в качестве входных параметров функции принимает длину окна, порядок полинома [21].

Созданные наборы данных сохраняем в виде файлов формата xlsx вместе с новыми метками классов удалив деление по диагнозам. Сохраненные наборы данных будут использоваться при обучении классификаторов. Листинг программного кода, отвечающего за предобработку данных представлен в приложении А.

3.4 Разработка диаграммы классов

Опираясь на математическую модель и псевдокод сформированные в главе 2.2, был выделен основной функционал разрабатываемого ПО.

Подсистема обучения классификаторов должна реализовывать функции предобработки набора данных, обучения классификаторов,

построение ансамблей классификаторов, тестирование и визуализация обучения классификаторов.

Была разработана UML диаграмма классов программы RAMAN CLASSIFER SISTEM. Она представлена на рисунке 18.

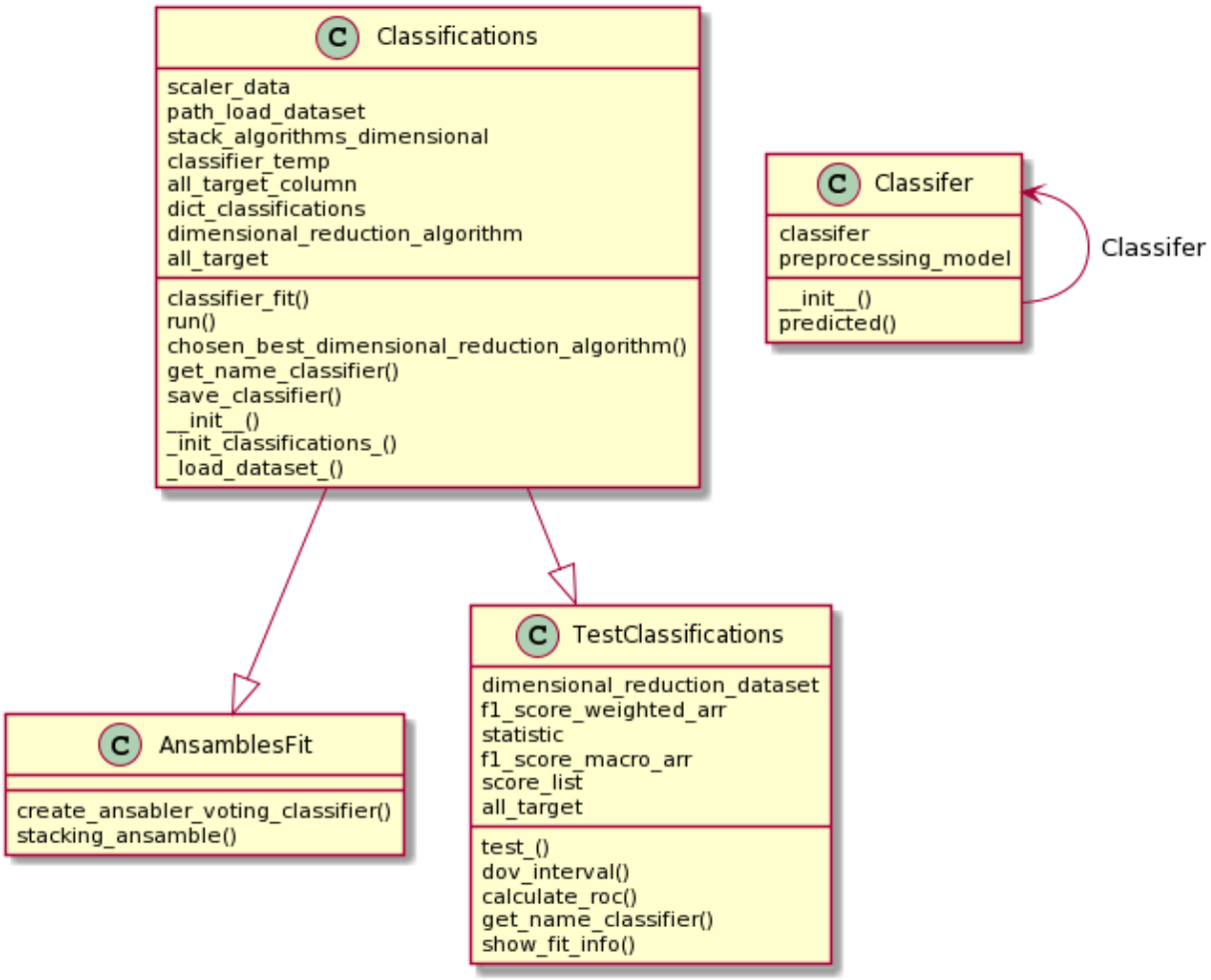

Рисунок 18 – Диаграмма класса

Вся системы была поделена на четыре класса: Classifer, AnsamblesFit, TistingClassifications, Classifications.

Classifer – класс классификатор. Используется для объединения моделей предобработки и классификации в одну структуру для дальнейшего сохранение использования.

Атрибуты:

- classifer – обученный классификатор;

- preprocessing_model – модель предобработки входных данных.

Методы:

- __init__() – метод инициализация атрибутов;
- predicted() – метод предсказания метки класса.

Classifications – используется для обучения классификаторов.

Методы и атрибуты:

- scaler_data – атрибут в котором хранятся нормированные данные;
- path_load_dataset – путь к набору данных;
- stack_algorithms_dimensional – список алгоритмов уменьшения размерности;
- classifier_temp – тестовый классификатор на котором проверяется алгоритмы уменьшения размерности;
- all_target_column – список меток набора данных представленных в виде одномерного массива;
- dict_classifications – список классификаторов;
- dimensional_reduction_algorithm – алгоритм уменьшения размерноти показавший лучшие характеристики;
- all_target - список меток набора данных представленных в виде двумерного массива;
- chosen_best_dimensional_reduction_algorithm – метод поиска оптимального алгоритма уменьшения размерности;
- save_classifier – метод сохранении сериализаванного объекта класса Classifer;
- __init__() – инициализация атрибутов;
- _load_dataset_() - загрузка обучающего набора;
- classifier_fit() – обучения классификатора;
- _init_classifications_() – инициализация классификаторов;
- get_name_classifier() – метод получения названия классификатора;
- run() – главный метод класса.

AnsamblesFit – расширяет класс «Classifications» методы создания ансамбля классификаторов: create_ansabler_voting_classifier – метод создания ансамбля на основе алгоритма взвешенного голосования.

TistingClassifications – расширяет класс «Classifications» функциями тестирования и визуализации обучения классификаторов.

Атрибуты:

- f1_score_weighted_arr – буфер значений средней f-меры;
- f1_score_macro_arr – буфер значений f-меры;
- score_list – буфер значений точности;
- all_target – список меток классов набора данных.

Методы:

- test_ – метод объединяющий различные способы тестирования классификаторов;
- dov_interval – метод подсчета доверительного интервала;
- calculate_roc – рассчитывает и выводит значения ROC кривой;
- get_name_classifier – возвращает имя классификатора;
- show_fit_info – метод вывода информации о обучении классификатора.

Полный листинг разработанных программные классов представлен в приложении. Classifer – приложение Д, AnsamblesFit – приложение Г, TistingClassifications – приложение В, Classifications – приложение Б.

3.5 Разработка модуля уменьшения размерности

Первым шагом обучения классификатора является применение алгоритма уменьшения размерности входных данных. Воспользуемся алгоритмом PCA описанном в параграфе 2.3.

Уменьшение размерности является не обязательным условием работы программного кода. При инициализации класса указав для флага is_pca значение False (по умолчанию True) обучение классификаторов будет

проводится с использованием полных спектров. Листинг программного кода представлен на рисунке 19. Для уменьшения размерности могут использоваться другие алгоритмы. Данный момент учтен при разработке метода chosen_best_dimensional_reduction_algorithm.

```
 def chosen_best_dimensional_reduction_algorithm(self, classifier,
stack_algorithms):
        """return best minimized demention algorithm"""
        result = {}
        for dimensional_reduction_algorithm in stack_algorithms():
            pipe = Pipeline(steps=[
                (self.get_name_classifier(dimensional_reduction_algorithm['model']),
dimensional_reduction_algorithm['model']),
                (self.get_name_classifier(classifier), classifier)
            ])
            param_grid = {}
            for params in dimensional_reduction_algorithm['params']:
                param_grid[
                    '__'.join([
self.get_name_classifier(dimensional_reduction_algorithm['model']),params[0]])
                    ] = params[1]
            param_grid['__'.join([ self.get_name_classifier(classifier),'C'])] =
np.logspace(-2, 2, num=3),
            param_grid['__'.join([ self.get_name_classifier(classifier),'gamma'])] =
np.logspace(-3, 2, num=3)
            search = GridSearchCV(pipe, param_grid, n_jobs=-1)
            search.fit(self.scaler_data, self.all_target)
            result[search.best_score_] = dimensional_reduction_algorithm['model']
        self.dimensional_reduction_algorithm = result[max(result.keys())]
        self.dimensional_reduction_algorithm.fit(self.scaler_data)
```

Рисунке 19 – Пример листинга обучения классификатора – логическая регрессия

В качестве входных данных используется список алгоритмов, установленный по усмотрению пользователя. Поиск оптимального количества признаков спектра осуществляется с помощью алгоритма GridSearchCV и тестировании обученного классификатора LogisticRegression, из которых будет выбран лучший исходя из кроссвалидационой проверке при обучении алгоритма Линейной регрессии(LogisticRegression).

Используемая в коде функция GridSearchCV создает матрицу различных сочетаний входных данных и проводит тестирование создавая новые наборы данных. Каждый раз на обучая классификатор Логической регрессии. В зависимости от значения кросс-валидации подбирались оптимальная размерность вектора признаков. Спектр удалось уменьшить до 5 компонентов.

3.6 Разработка модуля обучение классификаторов

В качестве алгоритмов классификации использовались LogisticRegression, LinearSVC, GaussianNB, KNeighborsClassifier, DecisionTreeClassifier. Для обучения классификатор использовалась их реализация в библиотеке Sklearn.

Все алгоритмы машинного обучения имеют схожую структуру обучения, отличную лишь входными параметрами каждой модели:

1) проводится поиск по сетке используя метод GridSearchCV;

2) находятся оптимальные параметры для работы алгоритма и осуществляется обучение классификатора с оптимальными параметрами;

3) обученный классификатор тестируется и сохраняется.

Листинг примера обучения классификатора представлен на рисунке 20.

```
from sklearn.linear_model import LogisticRegression
#GridSearchCV
tuned_parameters = {'C': list(range(1,150))}
classifier = GridSearchCV(LogisticRegression(solver='liblinear'),
                          param_grid=tuned_parameters,
                          cv=5,
                          verbose=True)
classifier.fit(normalized_train_dataset, train_target)
all_models['LogisticRegression'] = LogisticRegression(solver='liblinear',C=best_parameter)
all_models['LogisticRegression'].fit(normalized_train_dataset, train_target)
expected = test_target
predicted = all_models['LogisticRegression'].predict(normalized_test_dataset)
```

Рисунке 20 – Пример листинга обучения классификатора – логическая регрессия

Не является целесообразным использования схожего по своей структуре кода для обучения каждого классификатора, потому была реализована программный код для динамической работы с алгоритмами машинного обучения представленными в sklearn. В структуре класса данный программный модуль размещен в методе classifier_fit. Листинг метода представлен на рисунке 21.

```
def classifier_fit(
        self,
        classifier,
        params_grid,
        train_dataset,
        train_target
    ):
    """[summary]
    Arguments:
        classifier {[object]} -- обьект классификатор
        params_grid {[list]} -- список параметров для поиска оптимальных значений
        train_dataset {[list]} --  обучающие данные
        train_target {[list]} --  метки обучающих данных
    """
    print("---classifier_fit---")
    pipe = Pipeline(steps=[
        (
            self.get_name_classifier(classifier),
            classifier
        )
    ])
    if params_grid is not None:
        stack_param_grid = {'__'.join([self.get_name_classifier(classifier), item[0]]):
item[1] for item in params_grid}
    else:
        stack_param_grid = {}

    search = GridSearchCV(
        pipe,
        stack_param_grid,
        n_jobs=-1
    )
    search.fit(train_dataset, train_target)
    print("Best parameter (CV score=%0.3f):" % search.best_score_)
    print(search.best_params_)
    classifier.fit(train_dataset, train_target)
```

Рисунке 21 – Пример листинга обучения классификатора – логическая регрессия

Примечание по работе с классификаторами:

Существуют несколько алгоритмов построения деревьев принятия решений. Среди них выделяются алгоритмы CART, C4.5, C5.0. Алгоритм C5.0 является улучшенной версией C4.5. Он использует меньше памяти и создает меньшие наборы правил, при этом являясь более точным. Данный алгоритм распространяется по закрытой лицензии.

Из-за ограничений используемой нами библиотеки scikit-learn, в качестве алгоритма, реализующего построение дерева принятия решений, использовался алгоритм CART. Библиотека scikit-learn использует его оптимизированную версию [16].

Оба алгоритма, C4.5 и CART являются робастными, т.е. устойчивыми к шумам и выбросам данных. Разница в эффективности алгоритмов не существенна [25] потому использование посторонних средств реализации алгоритма C4.5 не является целесообразным так как использование посторонних библиотек помимо scikit-learn реализующих алгоритмы машинного обучения затруднит дальнейшее построение ансамблей увеличив сложность разработки программных модулей.

Стоит отметить, что грамотная разработка программного кода позволит в дальнейшем масштабировать приложение добавляя интерфейсы для взаимодействия с другими программными модулями, реализованными с использованием посторонних библиотек.

3.7 Разработка модуля ансамблирования классификаторов

Опираясь на математическую модель и псевдокод сформированные в главе 2.2, перед построением ансамбля классификаторов необходимо провести сравнение классификаторов опираясь на значение f1 меры.

Программная реализация создания классификатора, представленная на рисунке 22.

```
def create_ansabler(self, all_models, score_limite = 0.75):
    """[summary]

    Arguments:
        all_models list -- список обученных классификаторов

    Keyword Arguments:
        score_limite float -- ограничение точности (default: {0.75})

    Returns:
        VotingClassifier object -- ансамбль классификаторов
    """

    estimators = []
    for key, score in self.f1_score_weighted_arr.items():
        if score > score_limite and all_models.get(key):
            estimators.append((key, all_models[key]))
    return VotingClassifier(estimators=estimators, voting='soft',
flatten_transform=True)
```

Рисунке 22 – Листинг создания ансамбля классификаторов

Ансамбля классификаторов является основным. На его основе проводится дальнейшая работа со спектрами. Созданный данным программным модулем классификатор сериализуется и сохраняется отдельным файлом пригодным для загрузки в систему.

3.8 Разработка модуля визуализации обучения классификаторов

Метод show_fit_info выводит значения основных метрик классификаторов, рассмотренных в параграфе 2.7, а также отображает матрицу ошибок в виде темповой карты вместе с ROC кривой. Значения метрик добавляются в словари с метками классификаторов для последующего вывода на общих графиках. Листинг метода представлен на рисунке 23.

```
def show_fit_info(
        self,
        dataset,
        target,
        classifier
    ):
        predicted = classifier.predict(dataset)
        cm = metrics.confusion_matrix(target, predicted)
        metrics_stack = metrics.classification_report(target, predicted)
        ax = plt.subplot()
        sns.heatmap(cm, annot=True, ax = ax)
        ax.set_xlabel('Predicted labels')
        ax.set_ylabel('True labels')
        ax.set_title('Confusion Matrix')
        plt.show()
        f1_score_weighted=f1_score(target, predicted, average='weighted')
        f1_score_macro=f1_score(target, predicted, average='macro')
        buff_score = classifier.score(dataset, target)
        print("""
            f1_score_weighted: {0};
            f1_score_macro: {1};
            buff_score: {2};
        """.format(
            round(f1_score_weighted,3),
            round(f1_score_macro,3),
            round(buff_score,3)
        ))
        self.statistic[self.get_name_classifier(classifier)] = metrics_stack
        self.score_list[self.get_name_classifier(classifier)] = buff_score
        self.f1_score_weighted_arr[self.get_name_classifier(classifier)] =
f1_score_weighted
        self.f1_score_macro_arr[self.get_name_classifier(classifier)] = f1_score_macro
```

Рисунок 23 – Листинг Метода show_fit_info

Метод dov_interval рассчитывает значения доверительного интервала и выводит гистограмму разброса точности согласно информации в параграфе 2.7. Листинг метода представлен на рисунке 24.

```
def dov_interval(
        self,
        clean_model,
        optimaly_params,
        dataset,
        n_iterations = 1000,
    ):
        n_size = int(len(dataset) * 0.50)
        stats = []
        print(optimaly_params)
        for _ in range(n_iterations):
            train = resample(dataset, n_samples=n_size)
            test = np.array([x for x in dataset if x.tolist() not in
train.tolist()])
            model = clean_model()
            model.set_params(**optimaly_params)
            model.fit(train[:,:-1], train[:,-1])
            predictions = model.predict(test[:,:-1])
            score = accuracy_score(test[:,-1], predictions)
            stats.append(score)
        plt.hist(stats)
        plt.show()
        alpha = 0.95
        p = ((1.0-alpha)/2.0) * 100
        lower = max(0.0, np.percentile(stats, p))
        p = (alpha+((1.0-alpha)/2.0)) * 100
        upper = min(1.0, np.percentile(stats, p))
```

Рисунок 24 – Листинг метода dov_interval

Метод calculate_roc рассчитывает значения ROC кривой и выводит график. Информация об анализе классификатора с помощью ROC описана в параграфе 2.7. Листинг метода представлен на рисунке 25.

```
def calculate_roc(self, expected, predicted):
    fpr, tpr, _ = metrics.roc_curve(expected,  predicted)
    auc = metrics.roc_auc_score(expected, predicted)
    plt.plot(fpr,tpr,label="ROC curve {auc}".format(
                auc= round(auc, 3)))
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend(loc=4)
    plt.show()
```

Рисунок 25 – Листинг Метода calculate_roc

Методы, представленные в классе TistingClassifications иллюстрирующими эффективность работы алгоритмов классификации, рисуют графики, собирают и хранят статистику обучения по каждому классификатору. Данный программный модуль необходим для анализа обучения классификаторов вручную, но также и для работы по крон функции создавая ансамбль классификаторов согласно установленному лимиту.

Выводы к главе 3

В третьей главе был осуществлен выбор средств разработки подсистемы обучения классификаторов. Основываясь на данных представленных сервисом GitHab выбран язык программирования Python.

Данный язык чаще всего используется в задачах машинного обучения и имеет огромное количество открытых библиотек в области ML, активно поддерживаемых сообществом.

На основе математической модели и псевдокода сформулированных в главе 2, был реализован класс Classifications, включающий в семя метод уменьшения размерности входных данных, функции нормализации, обучения и тестирования классификаторов. Так же Classifications проводит сравнение обученных классификаторов и создание на их основе ансамбля по методу взвешенного голосования. Данный класс является частью подсистема обучения классификаторов.

Глава 4 Тестирование разработанных программных модулей

4.1 Создание наборов данных

Для создания и предобработки набора данных был реализован программный модуль CreateDataSet, приложение А. С помощью него в спектре был выделен информативный диапазон, часть спектра с индуцированным лазером пиком была удалена, рисунок 26.

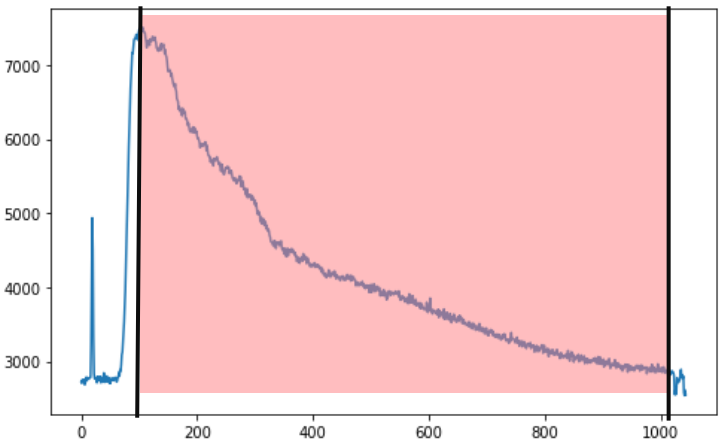

Рисунок 26 – Необработанный спектр с выделенной информативной частью

Каждый спектр был предобработан с помощью фильтра Савицкого-Голея с параметрами: размер окна – 15, порядок полинома – 0. Результат представлен на рисунке 27.

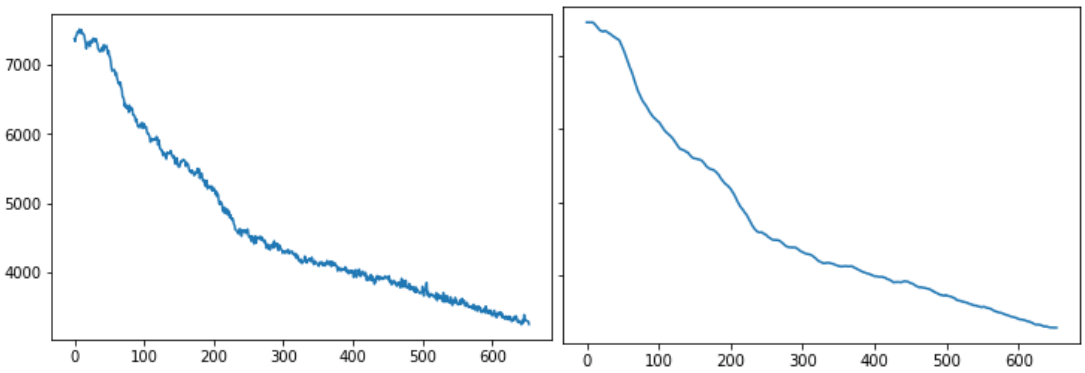

Рисунок 27 – Сглаживание спектра методом Савицкого-Голея

Помимо этого, набор данных был стандартизирован. В качестве алгоритма стандартизации использовался метод скалярной стандартизации, результат представлен на рисунке 28.

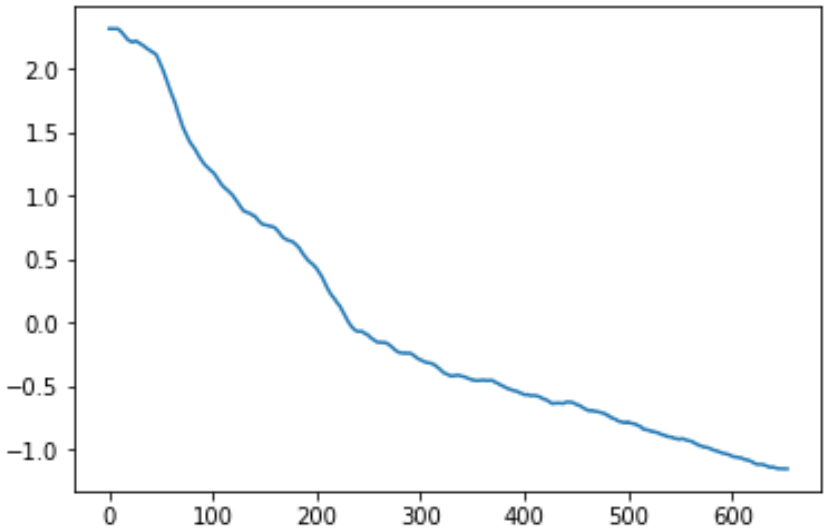

Рисунок 28 – Стандартизированный спектр

Созданный набор данных сохраняется отдельно и используется в дальнейшем для обучения классификаторов.

4.2 Уменьшение размерности входного вектора признаков.

Для уменьшения векторов признаков в разработанном программном модуле реализован метод chosen_best_dimensional_reduction_algorithm.

Главной проблемой метода является поиск оптимального количества характеристик при котором сохраняется информативность спектра. Для решения этой задачи использовался метод поиска по сетке(GridSearchCV) перебирающий параметры и сравнивающий обученные модели., представленный в библиотеке scikit-learn [11].

Принимая на входе список методов уменьшения размерности и классификатор для проверки информативности нового набора данных, метод возвращает новый набор данных. На экран выводится статистика по обучению рисунок 29.

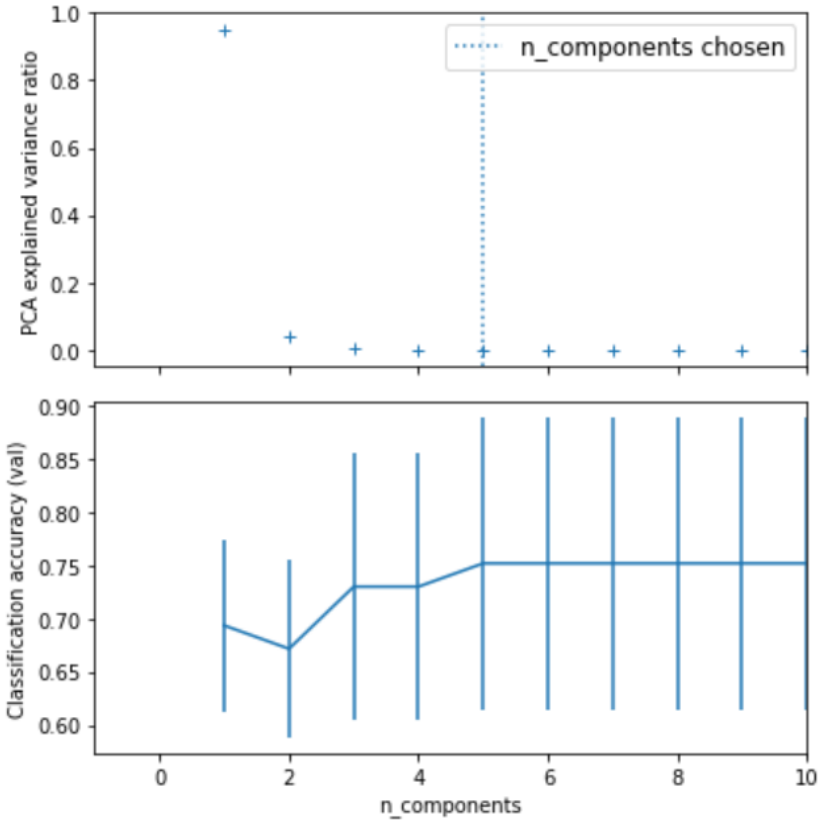

Рисунок 29 – Графики зависимость кроссвалидационной оценки обучения логической регрессии относительно количества компонентов

При тестировании алгоритма PCA совместно с другими алгоритмами значении кроссвалидационной проверки колебалось в пределах 0.55-0.85. При количестве компонентов от 2-8. Оптимальным стало использование пяти компонентов. Информация о тестировании представлена в таблице 5.

Таблица 5 – Сравнительный анализ зависимости кроссвалидационной оценки и размерности входного вектора признаков

Алгоритм	Кроссвалидационная оценка	Количество компонентов
kNN	0.58	2
SVM	0.82	5
LogisticRegression	0.75	5
Perceptron	0.63	8

В качестве алгоритма, проверяющего эффективность уменьшения размерности вектора признаков, был выбран алгоритм линейной регрессии. Данный алгоритм показывает сравнительно высокую точность работы с набором данных. Сравним его эффективность с алгоритмом k-NN, рисунок 30.

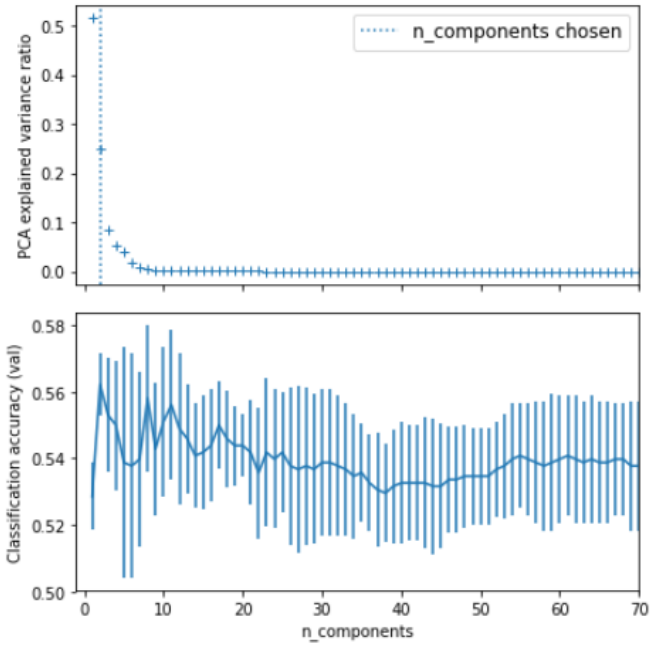

Рисунок 30 – Графики зависимость кроссвалидационной оценки обучения логической регрессии относительно количества компонентов

Алгоритм остановился на значении 2 при значении кроссвалидационной проверки 0.58. При работе с логической регрессией оптимальным размером стал вектор пятью компонентов при значении кроссвалидационной проверки 0.75

4.3 Обучение классификаторов

Используя разработанный программный модуль для динамического обучения классификаторов предоставляемых библиотекой Sclearn, были обучены алгоритмы: деревья принятий решений, метод опорных векторов,

логическая регрессия, k-NN. Данные обучения были помещены в пункты параграфа 4.3.

4.3.1 Деревья принятия решений

Осуществлялся поиск оптимальных параметров с помощью алгоритма GridSearchCV. Оптимальными параметрами для работы классификатора являются maximum depth 5, maximum features 2.

Оценка эффективности классификатора проводилась с помощью составления доверительного интервала (рисунок 31).

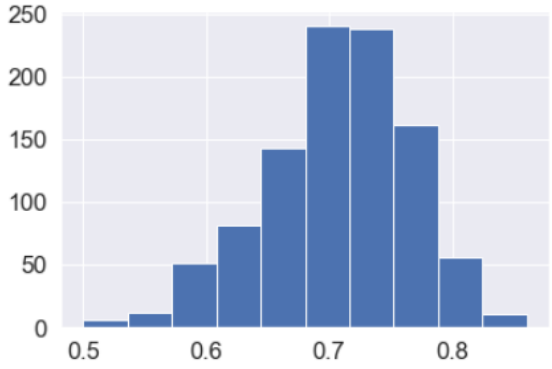

Рисунок 31 – Доверительный интервал оценки классификатора «Дерево принятия решений»

На рисунке 32 представлена тепловая карта распределения ошибок по классам, а также ROC кривая для классификатора «Деревья принятия решений».

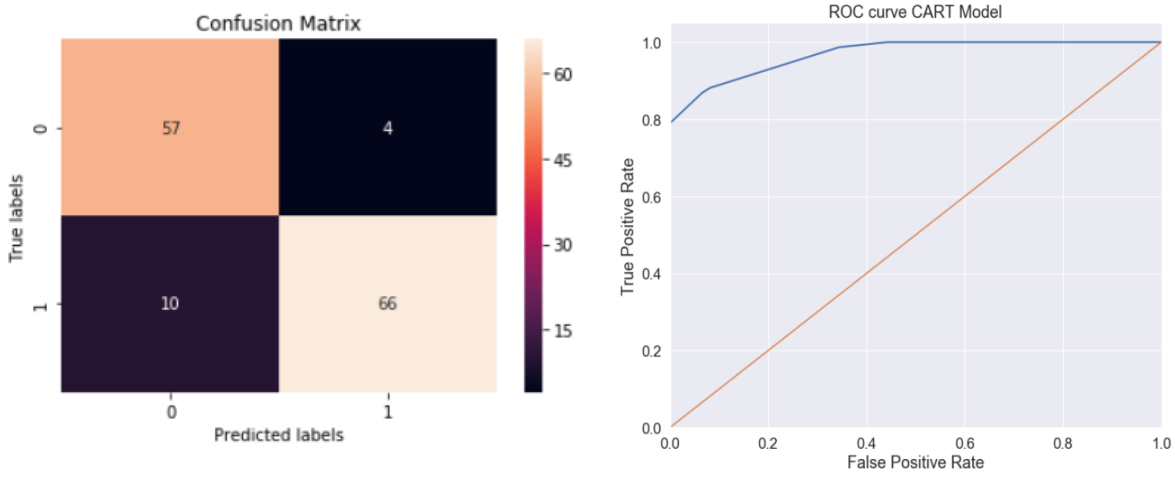

Рисунок 32 – Оценка работы классификатора «Дерево принятия решений»

Обученный классификатор обладает следующими характеристиками: f1 score – 89%, доверительный интервал - 58.5% – 81.2%, ROC – 90.1%.

4.3.2 Метод опорных векторов

Осуществлялся поиск оптимальных параметров с помощью алгоритма GridSearchCV. Оптимальными параметрами для работы классификатора являются gamma=0.001; kernel='rbf'. С их помощью проводилось обучение классификатора.

Оценка эффективности классификатора проводилась с помощью составления доверительного интервала (рисунок 33).

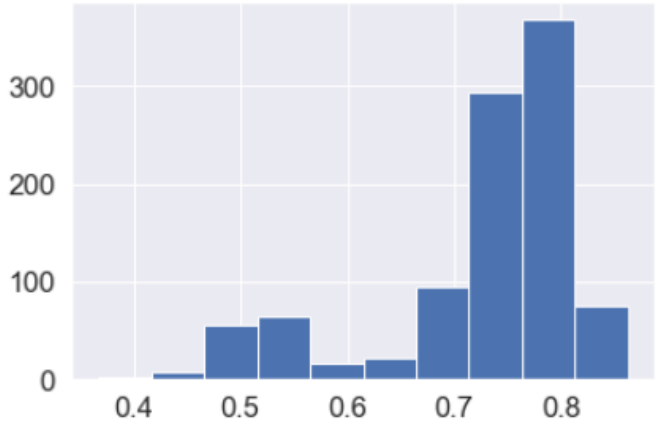

Рисунок 33 – Доверительный интервал оценки классификатора «Метод опорных векторов»

На рисунке рисунок 34 представлена тепловая карта распределения ошибок по классам, а также ROC кривая для классификатора «Деревья принятия решений».

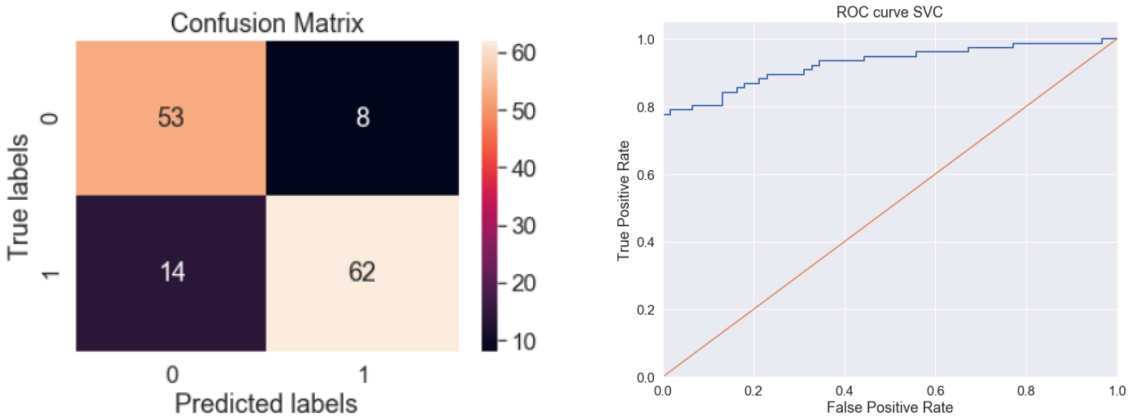

Рисунок 34 – Оценка работы классификатора «Метод опорных векторов»

Обученный классификатор обладает следующими характеристиками: f1 score – 82%, доверительный интервал: 48.8% – 82.7%, ROC – 84.2.

4.3.3 Логическая регрессия

Осуществлялся поиск оптимальных параметров с помощью алгоритма GridSearchCV. Оптимальными параметрами для работы классификатора являются maximum depth 5, maximum features 2. С их помощью проводилось обучение классификатора.

Оценка эффективности классификатора проводилась с помощью составления доверительного интервала (рисунок 35).

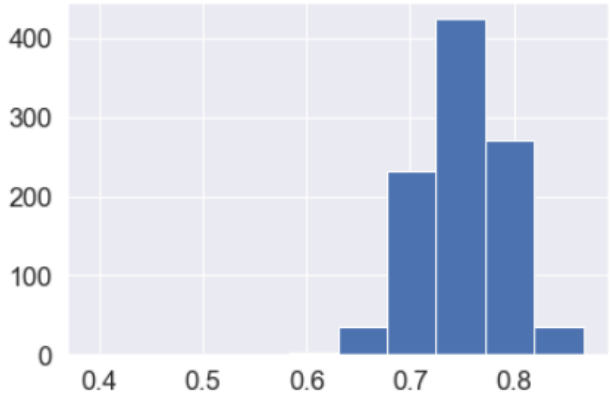

Рисунок 35 – Доверительный интервал оценки классификатора «Логическая регрессия»

На рисунке рисунок 36 представлена тепловая карта распределения ошибок по классам, а также ROC кривая для классификатора «Логическая регрессия».

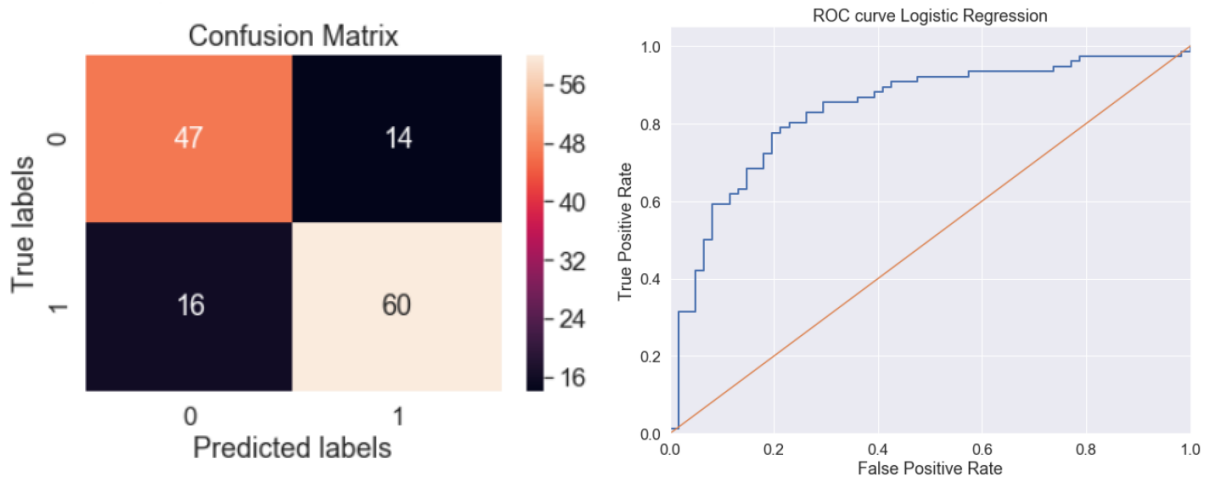

Рисунок 36 – Оценка работы классификатора «Логическая регрессия»

Обученный классификатор обладает следующими характеристиками: f1 score - 81%, Доверительный интервал – 66.7% - 82.5%, ROC 78.0%.

4.3.4 Алгоритм k-NN

Осуществлялся поиск оптимальных параметров с помощью алгоритма GridSearchCV. Оптимальными параметрами для работы классификатора являются maximum depth 5, maximum features 2. С их помощью проводилось обучение классификатора.

Оценка эффективности классификатора проводилась с помощью составления доверительного интервала (рисунок 37).

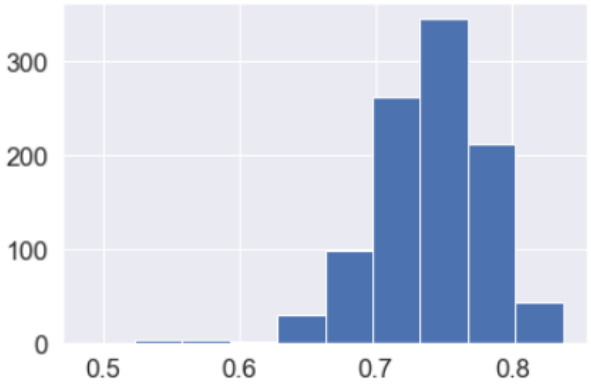

Рисунок 37 – Доверительный интервал оценки классификатора «k-NN»

На рисунке рисунок 38 представлена тепловая карта распределения ошибок по классам, а также ROC кривая для классификатора «k-NN».

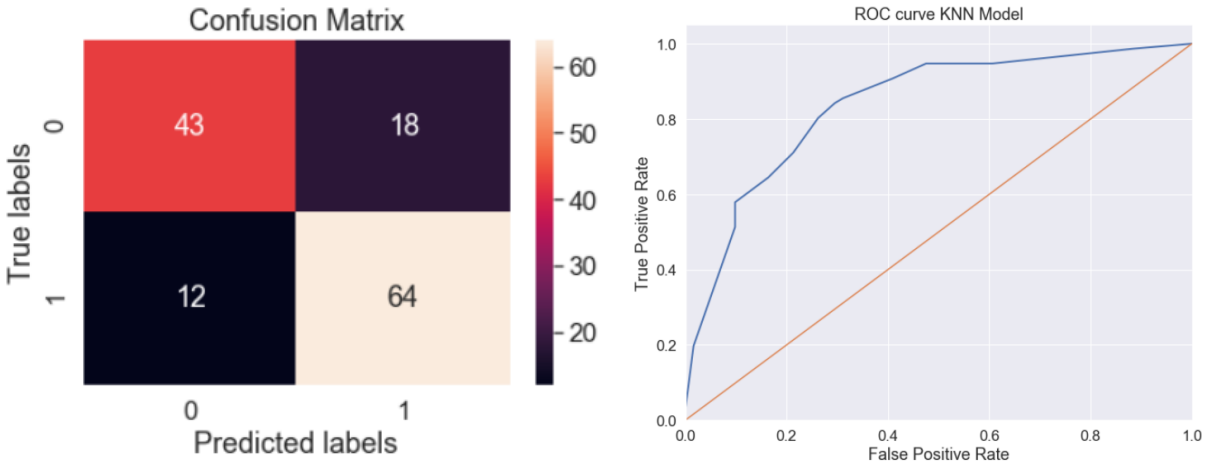

Рисунок 38 – Оценка работы классификатора «k-NN»

Обученный классификатор обладает следующими характеристиками: f1 score – 90%, Доверительный интервал: 65.5% – 81.0%, ROC 67,5% – 83,5%.

4.4 Обучение ансамбля

Из обученных выше классификаторов, опираясь на установленный лимит качества классификации в 80% f1-score, составлен ансамбль из алгоритмов: деревья принятия решений, метод опорных векторов, логическая регрессия, k-NN.

Оценка эффективности классификатора проводилась с помощью составления тепловая карты распределения ошибок по классам (Рисунок 39), построения ROC кривой(Рисунок 40), доверительного интервала (Рисунок 41).

Качество обучения ансамбля:

- f1 score – 90%,
- специфичность – 93%,
- чувствительность – 88%,
- доверительный интервал 67,5% – 83,5%.

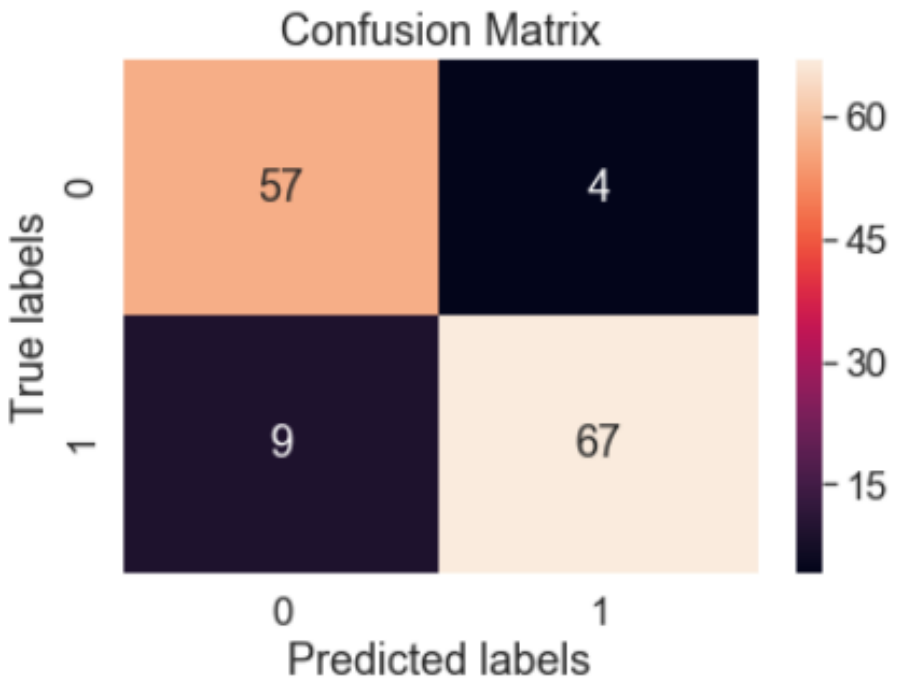

Рисунок 39 – Оценка работы классификатора - тепловая карта

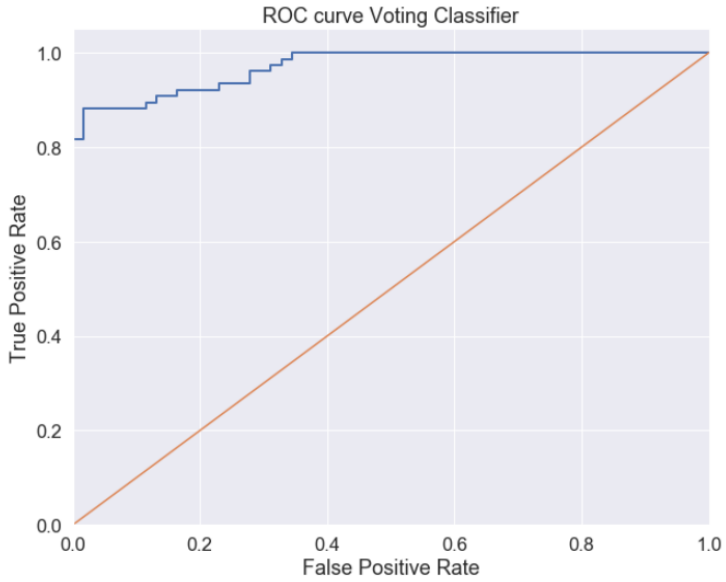

Рисунок 40 – Оценка работы классификатора - ROC кривая

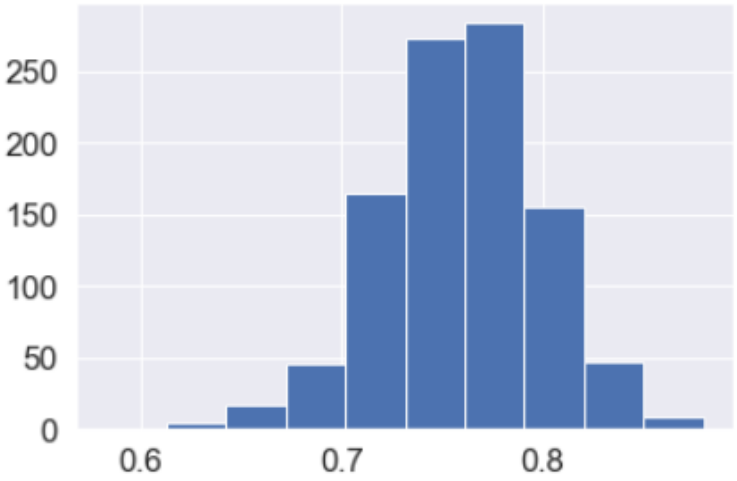

Рисунок 41 – Оценка работы классификатора – доверительный интервал

Полученный ансамбль классификаторов является основным. На его основе проводится дальнейшая работа со спектрами. Созданный данным программным модулем классификатор сериализуется и сохраняется отдельным файлом пригодным для загрузки в систему.

Выводы к главе 4

Проводилось тестирование разработанного программного кода. Были созданы и предварительно обработан обучающий набор данных. Каждый спектр был обрезан, нормирован и сглажен. Так же с помощью алгоритма уменьшения размерности удалось преобразовать спектр в пятикомпонентный вектор признаков.

Полученный набор данных использовался в обучение следующих алгоритмов: деревья принятия решений, метод опорных векторов, Логическая регрессия, k-NN. Значение f-меры для каждого алгоритма превышало установленный порог в 80% значения f1-score. Согласно данному условию все классификаторы смогли использоваться в составе ансамбля.

С помощью алгоритма мягкого взвешенного голосования удалось достичь значения f1-score – 91%, специфичность – 93%, чувствительность – 88%, доверительный интервал 67,5% – 83,5% в задаче дифференциации кожных патологий от здоровой кожи.

Заключение

В процессе работы над магистерской диссертацией были описаны актуальность и проблематика рассматриваемой темы, выявлен объект, предмет, поставлена цель и сформулированы задачи по теме исследования.

Проанализированы виды кожно-раковых патологий. Проведено сравнение подходов к анализу патологий. Выявлены перспективы использования спектрального анализа в работе с кожно-раковыми патологиями. Сформулированы требования и задачи для выполнения решения проблемы исследования.

Рассмотрен обучающий набор данных. Сформированы рекомендации по предобработке обучающего набора. Выбран подход к реализации задачи дифференциации патологий. Произведен анализ основных алгоритмов машинного обучения применяемые в данного рода задачах: k-NN, наивный байесовский алгоритм, логическая регрессия, деревья принятия решений, опорные вектора. Рассмотрена возможность использования ансамблей для оптимизации работы классификаторов. Опираясь на рассмотренные методы предобработки данных, алгоритмы построения классификаторов и ансамблей составлена математическая модель обучения системы дифференциации патологий. Написан псевдокод подробно описывающий принцип работы математической модели.

Выбраны средства разработки программного кода. Язык программирования Python, для работы с алгоритмами машинного обучения была выбрана библиотека Scikit-learn.

Реализован программный модулю, включающий в себя помимо логики математической модели, функции предобработки обучающего набора и алгоритмы уменьшения размерности, тестирования и анализа процесса обучения классификаторов. Разработка программного кода выполнена с учетом возможной масштабируемости системы.

Проведено тестирование работоспособности программного модуля. Созданы обучающие наборы для обучения классификаторов. Проведено обучение классификаторов. Сформирован и обучен ансамбль классификаторов.

С помощью алгоритма мягкого взвешенного голосования удалось достичь значения f1-score - 91%, специфичность - 93%, чувствительность - 88%, значение доверительного интервала 67,5% - 83,5%.

Разработанный в процессе диссертационной работы программный код поможет быть интегрирован в систему диагностирования патологий. Возможность использования различных алгоритмов машинного обучения делает программный код более гибким в работе с различными наборами данных. Качество работы классификатора может быть улучшено за счет увеличения набора данных.

Так же классификатор может быть применен для совместного использования с классификаторами обученными на других видах данных, фотографии, результаты дермоскопии, УЗИ и тд.

Всесторонний анализ патологий с использованием различных типов данных имеет огромный потенциал к поиску скрытых зависимостей для получения высокой точности дифференциации патологий.

Список используемых источников

Нормативно-правовые акты

1. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения [Текст]. Введ. 1992-01-01. – М.: Изд-во стандартов, 1992. – 14 с.

2. ГОСТ 2.105–95. Общие требования к текстовым документам [Текст]. – Введ. 1995-04-26. – М. : Госстандарт России: Изд-во стандартов, 1996 – 29 с.

3. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления [Текст]. – Введ. 2004-07-01. – М. : Госстандарт России: Изд-во стандартов, 2004.

4. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. Структура и правила оформления [Текст]. Введ. 2002-07-01. – М. : Госстандарт России: Изд-во стандартов, 2002.

5. ГОСТ 7.82-2001. Библиографическая запись. Библиографическое описание электронных ресурсов. [Текст]. Введ. 2001-05-01. – М. : Госстандарт России: Изд-во стандартов, 2001.

Научная и методическая литература

6. Жерон О. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем [Текст] / О Жерон, – [2018] – 199-222 с.

7. Никонов А.В. Фильтрация методом Савицкого-Голея спектральных характеристик чувствительности матричных фотоприемных устройств: журнал / А. В. Никонов, Р. В. Давлетшин, Н. И. Яковлева, П. С. Лазарев - АО «НПО «Орион»-2016. – С. 198-205.

8. Потекаев Н.Н. Современные диагностические технологии в дерматовенерологии / Н.Н. Потекаев, Н.В. Фриго, О.Л. Новожилова, Л.С. Круглова - 2018 - 104-113с.

9. Сержантов К.А. Дифференциация онкологических патологий с использованием алгоритмов машинного обучения [Текст] / К.А. Сержантов, М.Г. Лисовская; сборник статей «Информационные технологии в моделировании и управлении: подходы, методы, решения». – Тольятти, 2019. – 564-570 с.

10. Сержантов К.А. Разработка системы анализа результатов диагностики кожных патологий для выявления злокачественных новообразований на основе нейронных сетей [Текст] / К.А. Сержантов; выпускная квалификационная работа. – [2018] – 10 с.

11. Сержантов К.А. Реализация метода предобработки спектров комбинационного рассеяния для дифференциации кожных патологий ансамблем алгоритмов машинного обучения [Текст] / К.А. Сержантов, М.Г. Лисовская; сборник статей «Информационные технологии в моделировании и управлении: подходы, методы, решения». – Тольятти, 2020.

12. Холлок Г., Проспективное исследование точности диагностики хирурга при 2000 иссеченных опухолях кожи / Г. Холлок, Д.А.Лутц, Plast Reconstr Surg – 1998 – 1255–1261 с.

Электронные ресурсы

13. About GitHub [Электронный ресурс] / GitHub – Электрон. дан. – [2020]. – Режим доступа: https://github.com/about.

14. Automated detection of nonmelanoma skin cancer using digital images: a systematic review. – Электрон. дан. – [2019] – Режим доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6394090/.

15. Chan S., Machine Learning in Dermatology: Current Applications, Opportunities, and Limitations. – Электрон. дан. – [2020] – Режим доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7211783/.

16. Decision Trees [Электронный ресурс] / Scikit-learn – Электрон. дан. – [2020] – Режим доступа: https://scikit-learn.org/dev/modules/tree.html.

17. GitHub is how people build software / Wikipedia– Электрон. дан. – [2020] – Режим доступа: https://ru.wikipedia.org/wiki/GitHub.

18. Heath N. GitHub: The top 10 programming languages for machine learning/N. Heath. – Электрон. дан. – [2020] – Режим доступа: https://www.techrepublic.com/article/github-the-top-10-programming-languages-for-machine-learning/.

19. Curry JL, Pathology of Dysplastic (Atypical) Melanocytic Nevi . – Электрон. дан. – [2015] – Режим доступа: https://emedicine.medscape.com/article/1960604-overview.

20. Ngan V, Writer S., Amanda O., Atypical melanocytic naevus / DermNet NZ – Электрон. дан. – [2003] – Режим доступа: https://dermnetnz.org/topics/atypical-melanocytic-naevus/.

21. scipy.signal.savgol_filter [Электронный ресурс] / Scipy– Электрон. дан. – [2020] – Режим доступа: https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.signal. savgol_filter.html.

22. Ultraviolet (UV) radiation and skin cancer filter [Электронный ресурс] / World Health Organization. – Электрон. дан. – [2019] – Режим доступа: https://www.who.int/news-room/q-a-detail/ultraviolet-(uv)-radiation-and-skin-cancer.

23. Взвешенное голосование [Электронный ресурс] / Студопедия – Электрон. дан. – [2020] – Режим доступа: https://studopedia.ru/5_164917_vzveshennoe-golosovanie.html.

24. Меланома кожи / Московский клинический научный центр – Электрон. дан. – [2019] – Режим доступа: https://mknc.ru/album_view.php?album_id=9682&dir=melanoma-koji-chto-eto-za-opuhol-i-kak-vovremya-ee-raspoznat.

25. Методы классификации и прогнозирования. Деревья решений [Электронный ресурс] / ИНТУИТ – Электрон. дан. – [2020]. – Режим доступа:

https://www.intuit.ru/studies/professional_skill_improvements/1210/courses/6/lecture/174?page=3.

26. Назван самый смертельный вид рака в России. / Наши издания – Электрон. дан. – [2019]. – Режим доступа: https://rg.ru/2019/06/21/nazvan-samyj-smertelnyj-vid-raka-v-rossii.html.

27. C++ CART algorithm. / Github. – Электрон. дан. – [2020]. – Режим доступа: https://github.com/KirSerz/machine_learning/blob/master/CART.cpp.

28. Топ 10 библиотек Python для машинного обучения [Электронный ресурс] / kverner – Электрон. дан. – [2020]. – Режим доступа: https://www.kverner.ru/top-10-bibliotek-python-dlya-mashinnogo-obucheniya/.

29. Формула Байеса – Электрон. дан. – [2015]. – Режим доступа: https://nsu.ru/mmf/tvims/chernova/tv/lec/node15.html.

Литература на иностранном языке

30. Geraud C. Re: Deep learning outperformed 11 pathologists in the classification of histopathological melanoma images/ C. Geraud, KG. Griewank, - 2019.

31. Hekler A. Deep learning outperformed 11 pathologists in the classification of histopathological melanoma images / A. Hekler; European Journal of Cancer – 2019.

32. Khristoforova Y, “In vivo Raman and autofluorescence study of the pigmented skin neoplasms / Y. Khristoforova, I. Bratchenko, S. Konovalov, A. Andreeva, A. Moryatov, D. Kassirov, A. Orlov, S. Kozlov, V. Zakharov; Journal of Physics: Conference Series – 2019.

33. Khristoforova, Y. Optical diagnostics of malignant and benign skin neoplasms / Y. Khristoforova, L. Bratchenko, D.N. Artemyev, A Moryatov, O.O. Myakinina, A.A. Moryatovb, O.I. Kaganovb, S.V. Kozlovb, V.P. Zakharova; Information Technology and Nanotechnology – [2011] – 141-148 pg.

34. Lodha S, Discordance in the histopathologic diagnosis of difficult melanocytic neoplasms in the clinical setting. J Cutan Pathol./ S Lodha, S Saggar, JT Celebi, DN Silvers; Journal of Cutaneous Pathology – 2008.

35. Powers D.M.W., Evaluation: From Precision, Recall And F-Measure To Roc, Informedness, Markedness & Correlation / D.M.W Powers; Journal of Machine Learning Technologies 2 – 2011.

36. Powers D.M.W., Recall & Precision versus The Bookmaker / D.M.W Powers; International Conference on Cognitive Science International Conference on Cognitive Science – 2003.

37. Serzhantov K.A. Comparison testing of machine learning algorithms separability on Raman spectra of skin cancer / K.A. Serzhantov, O.O. Myakinin, M.G. Lisovskaya, I.A. Bratchenko, A.A. Moryatov, S.V. Kozlov, V.P. Zakharov; Proc. SPIE 11359, Biomedical Spectroscopy, Microscopy, and Imaging, – 2020.

38. Tran, B.X. Global Evolution of Research in Artificial Intelligence in Health and Medicine / B.X. Tran, G.T. Vu, G.H. Ha; Journal of Clinical Medicine – 2019.

Приложение А

Листинг класса CreateDataSet

```
class CreateDataSet ():
    path = None
    index_1 = None
    index_2 = None
    window = None
    polinom = None
    def __init__(self, path, index_1, index_2, window, polinom):
        self.path = path
        self.index_1 = index_1
        self.index_2 = index_2
        self.window = window
        self.polinom = polinom
    def create_data (self, dataFrame, index_1, index_2):
        dataset =np.array( dataFrame.iloc[:,
list(dataFrame.columns).index(index_1):list(dataFrame.columns).index(index_1)])
        target = dataFrame.loc[:,['label']]
        target = np.array(target.replace(value=[1, 0]))
        target =target.astype('int32')
        target.transpose()
        x=np.where(np.isnan(dataset))[0]
        x=np.unique(x)
        i=0
        for item in x:
            dataset=np.delete(dataset, item-i, axis=0)
            target=np.delete(target, item-i)
            i+=1
        return dataset, target
    def save_in_xlsx(self, data,name_file = 'sav_filter_data'):
    """сохранение нового датасета"""
        wb = openpyxl.Workbook()
        wb.create_sheet(title = 'Первый лист', index = 0)
        sheet = wb['Первый лист']
        for i, item in enumerate(data):
            for i_2, item_2 in enumerate(item):
                value =item_2
                cell=sheet.cell(row=i+1,column=i_2+1)
                cell.value = value
        wb.save(name_file+self.window +'.xlsx')
    def run(self):
        dataFrame = pd.read_excel(path)
        clear_data,y=self.create_data(dataFrame)
        sav_filter_data=[]
        for item in clear_data:
            sav_filter_data.append(savgol_filter(item,self.window, self.polinom))
        self.save_in_xlsx(sav_filter_data,name_file = 'sav_filter_data')
```

Приложение Б

Листинг класса Classification

```
Class Classification(
    TestClassifications,
    AnsamblesFit
):

    scaler_data = None
    all_target = None
    dimensional_reduction_algorithm = None

    def __init__(
        self,
        path_load_dataset = None,
        dict_classifications = None,
        save_path = None,
        stack_algorithms_temp = None
    ):
        # path_load_dataset = "C:\\Users\\Admin\\Documents\\new_dataset_sglaz.xlsx"
        self.save_path = save_path
        self.path_load_dataset = path_load_dataset
        self.scaler_model = StandardScaler()
        self.dict_classifications = self._init_classifications_(dict_classifications)
        self.classifier_temp = SVC()
        if stack_algorithms_temp is None:
            self.stack_algorithms_temp = [
                {
                    'model': PCA(),
                    'params': [
                        ('n_components', [item for item in range(1,5,1)])
                    ]
                }
            ]
        else:
            self.stack_algorithms_temp = stack_algorithms_temp

    def chosen_best_dimensional_reduction_algorithm(self, classifier, stack_algorithms):
        """
            return best minimized demention algorithm
        """
        print("---chosen_best_dimensional_reduction_algorithm---")
        result = {}
        for dimensional_reduction_algorithm in stack_algorithms():
            pipe = Pipeline(steps=[
                (
                    self.get_name_classifier(dimensional_reduction_algorithm['model']), dimens
ional_reduction_algorithm['model']
                ),
                (
```

```
                        self.get_name_classifier(classifier), classifier
                    )
                ])
                param_grid = {}
                for params in dimensional_reduction_algorithm['params']:
                    param_grid[
                        '__'.join([ self.get_name_classifier(dimensional_reduction_algorithm['model'
]),params[0]])
                        ] = params[1]
                param_grid['__'.join([ self.get_name_classifier(classifier),'C'])] =  np.logspace(-
2, 2, num=3),
                param_grid['__'.join([ self.get_name_classifier(classifier),'gamma'])] = np.logspac
e(-3, 2, num=3)
                search = GridSearchCV(pipe, param_grid, n_jobs=-1)
                search.fit(self.scaler_data, self.all_target)
                result[search.best_score_] = dimensional_reduction_algorithm['model']
            print("Best parameter (CV score=%0.3f):" % search.best_score_)
            print(search.best_params_)
            self.dimensional_reduction_algorithm = result[max(result.keys())]
            self.dimensional_reduction_algorithm.fit(self.scaler_data)
        def _init_classifications_(self, dict_classifications:tuple):
            print("---_init_classifications_---")
            if dict_classifications is None:
                return (
                    {
                        'model':KNeighborsClassifier,
                        'classifier':KNeighborsClassifier(),
                        'params_grid':(
                            (
                                'n_neighbors', list(range(1,50))
                            ),
                        ),
                    },
                    {
                        'model':DecisionTreeClassifier,
                        'classifier':DecisionTreeClassifier(),
                        'params_grid':(
                            (
                                'max_depth', np.arange(1,11)
                            ),
                            (
                                'max_features', np.arange(1,5)
                            ),
                        ),
                    },
                    {
                        'model':LogisticRegression,
                        'classifier':LogisticRegression(),
                        'params_grid':(
                            (
                                'C', list(range(1,150))
                            ),
```

```
                    ),
                },
                {
                    'model':SVC,
                    'classifier':SVC(probability=True),
                    'params_grid':(
                        (
                            'C', np.logspace(-4, 4, 4)
                        ),
                        (
                            'gamma', ['auto','scale']
                        ),
                    ),
                },
                {
                    'model':RandomForestClassifier,
                    'classifier':RandomForestClassifier(),
                    'params_grid':(
                        (
                            'n_estimators', np.arange(10,100)
                        ),
                        (
                            'max_features', np.arange(1,5)
                        ),
                    ),
                }
            )
        else:
            return dict_classifications
    def _load_dataset_(self, test_size = 0.1):
        print("---_load_dataset_---")
        dataFrame = pd.read_excel(self.path_load_dataset)
        dataset =np.array(dataFrame.iloc[:, list(dataFrame.columns).index(803.01):list(dataF
rame.columns).index(994.54)])
        all_target = dataFrame.loc[:,['label']]
        all_target = np.array(all_target)
        all_target =all_target.astype('int32')
        all_target.transpose()
        self.all_target_column = all_target
        self.all_target = all_target.ravel()
        self.scaler_data = self.scaler_model.fit_transform(dataset)
    def save_classifier(self,preprocessing_model,classifer,name):
        pickle.dump(Classifer(
            preprocessing_model,
            classifer
        ))
    def classifier_fit(
            self,
            classifier,
            params_grid,
            train_dataset,
            train_target
```

```
            ):
            print("---classifier_fit---")
            pipe = Pipeline(steps=[
                (
                    self.get_name_classifier(classifier),
                    classifier
                )
            ])
            if params_grid is not None:
                stack_param_grid = {'__'.join([self.get_name_classifier(classifier), item[0]]): item
[1] for item in params_grid}
            else:
                stack_param_grid = {}
            search = GridSearchCV(pipe, stack_param_grid, n_jobs=-1)
            search.fit(train_dataset, train_target)
            print("Best parameter (CV score=%0.3f):" % search.best_score_)
            classifier.fit(train_dataset, train_target)
        def get_name_classifier(self, classifier):
            return str(classifier).split('(')[0]
        def run(self):
            self._load_dataset_()
            self.chosen_best_dimensional_reduction_algorithm(self.classifier_temp, self.stack_a
lgorithms_temp)
            dimensional_reduction_dataset = self.dimensional_reduction_algorithm.transform(s
elf.scaler_data)
            train_dataset,  test_dataset,  train_target,  test_target = train_test_split(
                dimensional_reduction_dataset,
                self.all_target,
                test_size=0.1,
                stratify=self.all_target
            )
            for classifier_data in self.dict_classifications:
                self.classifier_fit(
                    classifier_data['classifier'],
                    classifier_data['params_grid'],
                    train_dataset,
                    train_target
                )
                self.test_(
                    classifier_data['model'],
                    train_dataset,
                    train_target,
                    test_dataset,
                    test_target,
                    classifier_data['classifier'],
                    dimensional_reduction_dataset,
                    self.all_target_column
                )
```

Приложение В

Листинг класса TestClassifications

```
# -*- coding: utf-8 -*-
"""
   @author: Admin
"""
class TestClassifications():

   dimensional_reduction_dataset = None
   all_target = None
   statistic = {}
   score_list = {}
   f1_score_weighted_arr = {}
   f1_score_macro_arr = {}


   def get_name_classifier(self, classifier):
      return str(classifier).split('(')[0]

   def calculate_roc(self, expected, predicted):
      fpr, tpr, _ = metrics.roc_curve(expected,  predicted)
      auc = metrics.roc_auc_score(expected, predicted)
      plt.plot(fpr,tpr,label="ROC curve {auc}".format(
                  auc= round(auc, 3)))
      plt.xlabel('False Positive Rate')
      plt.ylabel('True Positive Rate')
      plt.legend(loc=4)
      plt.show()

   def test_(
      self,
      clean_model,
      train_dataset,
      train_target,
      test_dataset,
      test_target,
      classifier,
      dimensional_reduction_dataset,
      all_target_column
   ):
      dataset = np.concatenate((dimensional_reduction_dataset, all_target_column), axis=
1)
      np.random.shuffle(dataset)
      self.dov_interval(
         clean_model,
         classifier.get_params(),
         dataset
      )
      self.show_fit_info(
```

```
            train_dataset,
            train_target,
            classifier,
            self.statistic,
            self.score_list,
            self.f1_score_weighted_arr,
            self.f1_score_macro_arr
        )

    def dov_interval(
        self,
        clean_model,
        optimaly_params,
        dataset,
        n_iterations = 1000,
    ):
        print("---dov_interval---")
        n_size = int(len(dataset) * 0.50)
        stats = []
        print(optimaly_params)
        for _ in range(n_iterations):
            train = resample(dataset, n_samples=n_size)
            test = np.array([x for x in dataset if x.tolist() not in train.tolist()])
            model = clean_model()
            model.set_params(**optimaly_params)
            model.fit(train[:,:-1], train[:,-1])
            predictions = model.predict(test[:,:-1])
            score = accuracy_score(test[:,-1], predictions)
            stats.append(score)

        plt.hist(stats)
        plt.show()
        alpha = 0.95
        p = ((1.0-alpha)/2.0) * 100
        lower = max(0.0, np.percentile(stats, p))
        p = (alpha+((1.0-alpha)/2.0)) * 100
        upper = min(1.0, np.percentile(stats, p))
        print('%.1f confidence interval %.1f%% and %.1f%%' % (alpha*100, lower*100, up
per*100))


    def show_fit_info(
        self,
        dataset,
        target,
        classifier,

        statistic,
        score_list,
        f1_score_weighted_arr,
        f1_score_macro_arr,
    ):
```

```
        print("---show_fit_info---")
        predicted = classifier.predict(dataset)
        # Отрисовка confusion_matrix
        cm = metrics.confusion_matrix(target, predicted)
        metrics_stack = metrics.classification_report(target, predicted)
        ax = plt.subplot()
        sns.heatmap(cm, annot=True, ax = ax)
        ax.set_xlabel('Predicted labels')
        ax.set_ylabel('True labels')
        ax.set_title('Confusion Matrix')
        plt.show()
        f1_score_weighted=f1_score(
            target,
            predicted,
            average='weighted'
        )
        f1_score_macro=f1_score(
            target,
            predicted,
            average='macro'
        )
        buff_score = classifier.score(
            dataset,
            target
        )
        print("""
            f1_score_weighted: {0};
            f1_score_macro: {1};
            buff_score: {2};
        """.format(
            round(f1_score_weighted,3),
            round(f1_score_macro,3),
            round(buff_score,3)
        ))

        self.statistic[self.get_name_classifier(classifier)] = metrics_stack
        self.score_list[self.get_name_classifier(classifier)] = buff_score
        self.f1_score_weighted_arr[self.get_name_classifier(classifier)] = f1_score_weighte
d
        self.f1_score_macro_arr[self.get_name_classifier(classifier)] = f1_score_macro
```

Приложение Г

Листинг класса AnsamblesFit

```
class AnsamblesFit():
    def create_ansabler_voting_classifier(
        self,
        all_models,
        score_limite = 0.75
    ):
        """[summary]
        Arguments:
            all_models list -- список обученных классификаторов
        Keyword Arguments:
            score_limite float -- ограничение точности (default: {0.75})
        Returns:
            VotingClassifier object -- ансамбль классификаторов
        """
        estimators = []
        for key, score in self.f1_score_weighted_arr.items():
            if score > score_limite and all_models.get(key):
                estimators.append((key, all_models[key]))
        return VotingClassifier(estimators=estimators, voting='soft',
flatten_transform=True)
    def stacking_ansamble(
        self
    ):
        dataset_for_stacking = None
        for sample in train_dataset:
            new_semple = None
            for name, classifer in new_stack_classifer.items():
                if name in ("DecisionTreeClassifier"):
                    temp_sample = classifer.predict_proba(sample.reshape(1, -1))
                    if new_semple is None:
                        new_semple = temp_sample
                    else:
                        new_semple = np.hstack((new_semple, temp_sample))
                else:
                    temp_sample = classifer.decision_function(sample.reshape(1, -1))
                    if new_semple is None:
                        new_semple = temp_sample
                    else:
                        new_semple = np.hstack((new_semple, temp_sample))

            if dataset_for_stacking is None:
                dataset_for_stacking = new_semple
            else:
                dataset_for_stacking = np.vstack((dataset_for_stacking, new_semple))
        dataset_for_stacking = np.array(dataset_for_stacking)
```

Приложение Д

Листинг класса Classifer

```
class Classifer():
    preprocessing_model = None
    classifer = None

    def __init__(self, preprocessing_model,classifer):
        self.preprocessing_model = preprocessing_model
        self.classifer = classifer

    def predicted(self,specter):
        preprocessing_specter = self.preprocessing_model(specter)
        return self.classifer.predict(preprocessing_specter)
```