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*)
Кафедра «Прикладная математика и информатика»
(*наименование кафедры*)

БАКАЛАВРСКАЯ РАБОТА

на тему **Анализ спектров комбинационного рассеяния с помощью алгоритмов машинного обучения**

Студент В. А. Нейман _____
(И.О. Фамилия) (личная подпись)
Руководитель С. В. Баумгертнер _____
(И.О. Фамилия) (личная подпись)
Консультанты Н.В. Андрюхина _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2019

АННОТАЦИЯ

Тема выпускной квалификационной работы: «Анализ спектра комбинационного рассеяния с помощью алгоритмов машинного обучения».

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

В выпускной квалификационной работе рассматривается проблема диагностики онкологических заболеваний тканей кожи человека.

Задачи выпускной квалификационной работы – подготовка входных данных, тестирование моделей машинного обучения на задаче классификации результатов спектроскопии комбинационного рассеяния, разработка ансамбля алгоритмов машинного обучения, тестирование ансамбля на результатах спектроскопии комбинационного рассеяния.

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

Предмет исследования – алгоритмы машинного обучения.

В ходе выполнения выпускной квалификационной работы была проанализирована проблема ранней диагностики онкологических заболеваний, сделаны выводы на основе данных о статистике заболеваний за 2017-2018 года, проанализирован метод спектроскопии комбинационно рассеяния, как способ диагностики онкологии кожи. Далее была проведена предобработка входных данных и анализ моделей машинного обучения для задачи классификации. После сравнительного анализа были отобраны наиболее эффективные алгоритмы классификации и разработан ансамбль на их основе. Результаты тестирования ансамбля из алгоритмов машинного обучения показали, что значение точности классификации, в отличие от лучшего классификатора, не возросло, однако метрики качества классификации показывают, что модель является более стабильной и устойчивой. При разработке модели машинного обучения использовались такие инструменты разработки, как объектно-ориентированный язык

программирования Python 3.7.3, библиотека для машинного обучения с открытым исходным кодом Scikit-learn v0.21.2, среда разработки Announcing Anaconda Enterprise 5.3 и облачный сервис Google Colaboratory с интерактивной оболочкой Jupyter Notebook.

Тестирование ансамбля алгоритмов машинного обучения на данных спектроскопии комбинационного рассеивания показало следующие результаты: точность классификации – 87%, полнота – 85%, гармоническое среднее между точностью и полнотой (мера F1) – 86%.

Анализ кривых валидации и обучения указал на малый размер тренировочных данных и, для некоторых случаев, высокую сложность модели, что привело к падению точности классификации.

Данная выпускная квалификационная работа состоит из пояснительной записки на 53 страницы, введения – 3 страницы, включая 27 рисунков, 6 таблиц, списка литературы из 23 источников, в том числе 12 источников на иностранном языке и 2 приложений.

ABSTRACT

Subject of final qualification work: "Analysis of a range of combinational dispersion by means of algorithms of machine learning".

The final qualification work purpose is – maximizing differentiation quality of skin's new growths by results of combinational spectroscopy dispersion.

In final qualification work is considered the problem of oncological disease's diagnosis of person's skin tissues.

Tasks of final qualification work – development and testing of ensemble of algorithms of machine learning on results of spectroscopy of combinational dispersion.

Research object – process of differentiation of samples of biofabric by results of spectroscopy of combinational dispersion.

Object of research – algorithms of machine learning.

During execution of final qualification work the problem of early diagnosis of oncological diseases was analyzed, conclusions on the basis of data on statistics of diseases for 2017-2018 are drawn, the spectroscopy method kombinatsionno of dispersion as a way of diagnostics of oncology of skin is analyzed. Further preprocessing of input data and the analysis of models of machine learning for a problem of classification was carried out. After contrastive analysis the most effective algorithms of classification were selected and the ensemble on their basis is developed. Results of testing of ensemble from algorithms of machine learning showed that the value of accuracy of classification, unlike the best qualifier, did not increase, however metrics of quality of classification show that the model is stabler and steady. When developing model of machine learning such building tools as an object-oriented programming language of Python 3.7.3, library for machine learning open source Scikit-learn v0.21.2, the Announcing development environment were used Anaconda Enterprise 5.3 and cloud service of Google Colaboratory with an interactive cover of Jupyter Notebook.

This final qualification work consists of the explanatory note on 53 pages, introduction - 3 pages, including 27 drawings, 6 tables, the list of references from 23 sources, including 12 sources in a foreign language and 2 applications.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 7
1 АНАЛИЗ ПРОБЛЕМЫ ОНКОЛОГИЧЕСКИХ ЗАБОЛЕВАНИЙ И МЕТОДОВ ДИАГНОСТИКИ 10
1.1 Онкологические заболевания. Классификация рака. 10
1.2 Рак кожи. Базально-клеточная карцинома, меланома, сквамозно-клеточная карцинома. 11
1.3 Статистика заболеваний за 2017-2018 год 13
1.4 Методы диагностики онкологических заболеваний кожи 15
1.5 Спектроскопия комбинационного рассеяния, как метод диагностики онкологических заболеваний кожи 18
1.6 Анализ разработанных решений 20
2 АНАЛИЗ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ 21
2.1 Формальное определение задачи классификации 21
2.2 Предобработка входных данных 22
2.3 Алгоритмы машинного обучения 24
2.4 Сравнительный анализ моделей машинного обучения 33
2.5 Подбор параметров моделей машинного обучения 37
3 ФОРМИРОВАНИЕ И ТЕСТИРОВАНИЕ АНСАМБЛЯ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ 42
3.1 Метрики анализа качества классификации 42
3.2 Формирование и тестирование ансамбля алгоритмов машинного обучения на входных данных 46
ЗАКЛЮЧЕНИЕ 61
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 62
ПРИЛОЖЕНИЕ А 65
ПРИЛОЖЕНИЕ Б 74

ВВЕДЕНИЕ

За 2017 год, как сообщает Министерство здравоохранения, самым частым онкологическим заболеванием стал рак кожи.

Общее число онкологических заболеваний выросло на 3% за 2017 год (с 599,3 тыс. до 617,2 тыс.) Диагноз – новообразование кожи не первый раз становится лидером, такая же ситуация складывалась в 2014-2016 годах. [9]

Рак кожи – это обобщенное название большого количества разновидностей злокачественных опухолей. Каждая опухоль имеет свои специфические биологические особенности: клиническое проявление, тканевую структуру, метастазирование и т.д.

Существует три основных типа рака кожи: базальноклеточный рак, плоскоклеточный рак и злокачественная меланома. Среди всех раков кожи злокачественная меланома кожи является наиболее опасным видом рака, так как в подавляющем большинстве случаев приводит к гибели пациентов, особенно при обнаружении патологии на поздней стадии. При этом заболеваемость и смертность от меланомы кожи увеличивается в большинстве стран по всему миру.

Трудности диагностики меланом врачами общей практики связаны со сложностью в интерпретации клинических признаков опухоли и невозможностью отличить меланому от доброкачественных пигментных образований на ранней стадии развития. Также при подозрении на наличие меланомы медицинский персонал лишён возможности использования инвазивных методов исследования, таких как биопсия с гистологическим или цитологическим исследованием, в связи с повышенным риском прогрессирования поражений. В этой связи оптические методы имеют огромный потенциал для неинвазивного выявления и определения конкретного типа опухолевых образований в тканях кожи с применением инструментальных методов.

Наиболее широко развивающимися в этой области являются методы оптической спектроскопии, они позволяют неинвазивно диагностировать

раковые опухоли. Сегодня в арсенале ученых существует несколько способов спектроскопического анализа биологических сред - спектроскопия обратного рассеяния, автофлуоресценция, спектроскопия комбинационного рассеяния (КР), отражательная спектроскопия и другие. [14]

При выполнении бакалаврской работы использовались объектно-ориентированный язык программирования Python 3.7.3, библиотека для машинного обучения с открытым исходным кодом Scikit-learn v0.21.2, среда разработки Announcing Anaconda Enterprise 5.3 и облачный сервис Google Colaboratory с интерактивной оболочкой Jupyter Notebook.

Проект разрабатывается при поддержке Самарского национального исследовательского университета имени академика С.П. Королева, материалы для исследования предоставлены ГБУЗ Самарским областным клиническим онкологическим диспансером.

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

Задачи выпускной квалификационной работы – подготовка входных данных, тестирование моделей машинного обучения на задаче классификации результатов спектроскопии комбинационного рассеяния, разработка ансамбля алгоритмов машинного обучения, тестирование ансамбля на результатах спектроскопии комбинационного рассеяния.

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

Предмет исследования – алгоритмы машинного обучения.

Выпускная квалификационная работа состоит из введения, трех глав и заключения.

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

В первой главе обосновывается актуальность проделанной работы, анализируется проблема онкологии, представляется статистика заболеваний за 2017-2018 года, проанализированы текущие исследования и разработки в данной области.

Во второй главе описываются отобранные алгоритма машинного обучения, так же подготавливаются наборы данных для их обучения. Происходит обучение моделей и сравнительный анализ точности классификации, а также отбор лучших моделей классификации.

В третьей главе происходит формирование ансамбля из отобранных алгоритмов классификации и его тестирование на основе подготовленных наборов данных.

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

1 АНАЛИЗ ПРОБЛЕМЫ ОНКОЛОГИЧЕСКИХ ЗАБОЛЕВАНИЙ И МЕТОДОВ ДИАГНОСТИКИ

1.1 Онкологические заболевания. Классификация рака.

Онкологические заболевания представляют из себя обширный класс заболеваний, являющимися одной из основных причин смерти на планете. Смертность от рака занимает второе место в мире — после сердечно-сосудистых заболеваний.

Каждый год во всем мире врачи обнаруживают 12,3 миллиона образование новых раковых заболеваний. Ежегодно именно рак становится причиной смерти 7,6 миллиона больных. Наибольшее кол-во смертей регистрируется в странах с низким уровнем жизни. Прогнозируется рост онкологических заболеваний и достижения отметки в 17 миллионов больных к 2030 году.

Российская федерация организовывает две программы по борьбе с раком. Разработка научных подходов к лечению, финансирование ученых, врачей, специалистов данной области, закупка необходимого оборудования – главные цели данных программ.

Каждый вид онкологического заболевания имеет свои особенности локализации, течения, метастазирования, поэтому универсального лекарства, избавившего бы человечество от всех видов рака, не удается найти. Следовательно, к лечению онкологии необходимо подходить одновременно комплексно и индивидуально.

В большинстве случаем вся ответственность за здоровье и жизнь человека лежит на плечах врачей, однако существуют случаи, когда пациент обращается за помощью слишком поздно, что приводит к неисправимым последствиям. Поэтому именно профилактика онкологических заболеваний может снизить кол-во заболеваний и выявить новообразования на ранней стадии. [15]

Современная медицина разработала мероприятия по профилактике онкологических заболеваний, которые предполагают устранение 80% причин и факторов заболеваний.

1.2 Рак кожи. Базально-клеточная карцинома, меланома, сквамозно-клеточная карцинома.

Несмотря на то, что рак кожи развивается на эпителии организма, его диагностика является сложным процессом, как для начинающих онкологов, так и для врачей других специальностей, например, дерматологов. Постоянный рост базально-клеточной и сквамозно-клеточной карциномы приводит к осложнениям в ранней диагностике заболевания, так как параллельно с ними существует большое многообразие доброкачественных новообразований кожи.

Одной из главных причин возникновения онкологических заболеваний кожи является ультрафиолетовое воздействие. В первую очередь, ультрафиолетовое воздействие влияет на клетки дезоксирибонуклеиновой кислоты (ДНК). Прямое воздействие УФВ характеризуется онкогенной трансформацией клеток и эпигеномным механизмом воздействия канцерогенов. Степень влияния УФВ на кожу во многом зависит от фототипа.

Сквамозно-клеточная карцинома – злокачественное новообразование эпителия, образующееся из 90% клеток эпидермиса – кератиноцитов.

Сквамозно-клеточная карцинома чаще всего возникает благодаря некоторым патологическим состояниям организма: рубцы после ожога, лучевые повреждения покровных тканей, псориаза, профессиональных заболеваний кожи и другие.

Вопрос клинической онкологии всегда обусловлен не высокой эффективность своевременной диагностики, первичной и вторичной профилактики.

Базально-клеточная карцинома характеризуется наличием агрессивных клеток, потенциально злокачественных, в периферическом клеточном слое. Ядра в этом случае располагаются в виде частокола. В данном случае отношение между площадями цитоплазмы и ядра живой клетки, то есть ядерно-цитоплазматическое отношение (ЯЦО), значительно больше, чем у нормальных клеток. Базально-клеточная карцинома (базалиома) состоит из клеток, ядра которых имеют круглую, бобовидную или овоидную форму. Ядра, в большинстве своем, состоят из матрикса и небольшого количества цитоплазмы. [9]

Карцинома (плоскоклеточный рак кожи) в начале своего развития внешне очень схожа с базально-клеточной карциномой (базалиомой). Однако скорость развития карциномы выше, она быстрее увеличивается в размерах.

Меланоме характерны некоторые факторы риска, которые способствуют злокачественному превращению маленоцитов. Данные факторы разделяют на экзогенные (факторы окружающей среды) и эндогенные (образованные организмом человека).

К факторам окружающей среды (экзогенным) относится:

- ионизирующая радиация;
- хроническая травматизация кожи;
- ультрафиолетовое излучение (УФ);
- флюоресцентное освещение;
- электромагнитное излучение.

Несмотря на прямое воздействие УФ на кожу человека, меланома чаще всего появляются в областях кожи, которые обычно скрыты под одеждой. Это означает, что процесс возникновение новообразования имеет некоторые особенности. Даже кратковременное УФ-излучение может привести к высокому канцерогенному эффекту.

Однако более частой причиной возникновения меланомы является травматизация невусов. Невус (родинка, родимое пятно) - врождённые или приобретённые пигментированные образования на коже, имеющие

различные цвета. Однократные и многократные травмы предшествующих невусов является наиболее частым экзогенным фактором риска развития меланомы кожи. [18]

Существует две группы эндогенных факторов, к первой относится расовая и этническая предрасположенность, наследственные факторы, иммунные нарушения, уровень пигментации организма и эндокринные факторы. Все эти факторы повышают риск развития меланомы.

1.3 Статистика заболеваний за 2017-2018 год

За 2017 год, как сообщает Министерство здравоохранения, самым частым онкологическим заболеванием стал рак кожи.

Общее число онкологических заболеваний выросло на 3% за 2017 год (с 599,3 тыс. до 617,2 тыс.) Диагноз – новообразование кожи не первый раз становится лидером, такая же ситуация складывалась в 2014-2016 годах.

Рак груди занимает второе место по частоте обнаружения (70,6 тыс.). За последнее время данное онкологическое заболевание стали выявлять на много чаще.

Онкология дыхательных путей занимает третье место - 62,2 тыс. случаев.

Полная статистика показана на рисунке 1.1.

Рисунок 1.1 – Статистика онкологических заболеваний за 2017 год

В 2017 году кол-во умерших от онкологических заболеваний приблизилось к рекордному значению, которое было достигнуто в 2015 году. Практически четверть пациентов умерли в первый год после обнаружения диагноза.

В 2017 году число поставленных диагнозов в Самарской области выросло на 11%. По данным Самарастата в 2017 году доля умерших от новообразований кожи составила 15%.

Каждый год в регионе регистрируют около 13 тысяч случаев заболеваний раком. На учете в онкологическом диспансере состоит каждый 40-ой житель области. [6]

По сравнению с другими регионами России, в Самарской области максимальные показатели. Похожая ситуация в Новгородской, Орловской, Ярославской, Рязанской и Курской областях.

Среди онкозаболеваний на первом месте в Самарской области - рак кожи, на втором месте - заболевания прямой и слепой кишки, на третьей строчке - злокачественные новообразования в молочных железах у женщин, в предстательной железе у мужчин, а также в, желудке, крови. Смертельных исходов больше всего среди больных раком легких.

В 2018 году в Самарской области так же сохраняется высокий уровень заболеваемости различными видами рака. Показатель за 2018 год составляет 508,5 случая на 100 тысяч населения.

По итогам 2018 года в регионе зарегистрировали 16 тысяч новых пациентов. Несмотря на такие неблагоприятные цифры, зафиксировано снижение смертности. Эта положительная тенденция появилась благодаря скрининговым программам, которые лежат в основе диспансеризации. Также больше возможностей открывается благодаря химиотерапии. [6]

1.4 Методы диагностики онкологических заболеваний кожи

Существует прямая зависимость между ранним выявлением злокачественного заболевания и онкологической насторожённостью врачей общей практики. Врачи первичного звена, к которым в первую очередь обращаются пациенты при различных жалобах, обязаны уже при осмотре исключить симптомы, указывающие на онкологию.

Именно первичные рекомендации пациенту являются определяющим шагом к лечению.

Пациент может обратиться к терапевту или врачу семейной медицины с жалобами, к примеру, на воспалительный процесс в организме, при этом нет абсолютных гарантий, что в этот момент он не страдает онкологическим заболеванием, которое на начальном этапе может себя не проявлять. Соответственно грамотный специалист может наряду с лечением ОРЗ параллельно выявить онкологию. Прием у стоматолога также может привести к онкологу, если врач заметил подозрительные образования

полости рта. Гинеколог, в свою очередь, должен обязательно осматривать молочные железы пациентки на предмет выявления опухолевой патологии.

На раннем этапе диагностики обычно определяются новообразования, которые видны визуально, например, на коже, молочной железе, шейки матки, полости рта и т.д. Именно визуальные новообразования должен обнаружить врач, впервые контактирующий с пациентом.

В случае обнаружения новообразования пациент отправляется на консультацию к онкологу, задача которого разобраться в причине возникновения новообразования и направить на соответствующее лечение.

Дальнейшие действия пациента зависят от вида новообразования. Если новообразование является доброкачественным, то в лечении пациента применяется хирургический подход. Если это злокачественный процесс, который требует специальных методов лечения, пациента отправляют в онкодиспансер.

Очень важно, чтобы врачи общей практики и население в целом были проинформированы о факторах риска возникновения злокачественных новообразований.

Существует большое количество методов диагностики онкологических заболеваний. К ним относится визуальный осмотр, дерматоскопия, термометрический анализ, цитологическая диагностика, биопсия и другие.

Самый распространенный неинвазивный диагностический метод новообразований кожи – дерматоскопия. Данный метод позволяет исследовать как поверхность кожи, так и более глубокие слои. Любые изменения структуры кожи записываются в атлас для дальнейшего сравнения с новообразованиями.

Благодаря развитию цифрового ультразвукового изображения появилась возможность использовать частоты 20-100 МГц для одного из метода диагностики – УЗИ. Данный метод так же чаще применяется для диагностики новообразований кожи без ее повреждения. [20]

Еще один неинвазивный метод, диагностирующий новообразования кожи, который позволяет получить изображения дермы, близкие к оптической микроскопии называется конфокальной лазерной микроскопией.

Соскоб с поверхности дермы. Особенность данного способа заключается в том, что проводить данное исследование разрешено лишь с кожным покровом не имеющих наружных и внутренних патологий. В случае, если диагноз подтвердить или опровергнуть не удается, назначается повторное исследование.

Скрининг. Направлен на выявление, в основном, раковых заболеваний на начальной стадии поражения, он быстротечен, его результаты могут служить направлением на диагностирование и лечение.

Эффективность программ скрининга при выявлении определенных видов рака достигается за счет использования соответствующего тестирования, его эффективного использования, связи с другими этапами процесса скрининга и контроля качества. В общем, программа скрининга представляет собой более сложное медицинское вмешательство.

Одним из самых точных методов диагностики на данный момент является цитологическое исследование. Данный метод позволяет морфологически проверить поставленный диагноз.

Тотальная ножевая эксцизионная биопсия. Болезненная диагностика, исследование проводится только под общим наркозом.

1. Биопсия назначается только пациентом с запущенной степенью онкозаболеваия, когда другие варианты исследования были испробованы и не принесли свои плоды.

2. Биопсия проводится при опухолях не более 15 мм.

3. Биопсия показана, в случае достоверности поставленного диагноза.

4. Биопсия проводится, при ампутации конечностей пациента.

1.5 Спектроскопия комбинационного рассеяния, как метод диагностики онкологических заболеваний кожи

Спектроскопия комбинационного рассеяния света (Рамановская спектроскопия) на данный момент широко применяется во множестве научных и промышленных направлениях. Рамановская спектроскопия подразумевает облучение вещества (твердого, жидкого, газообразного) монохроматическим пучком света и запись спектра, рассеянного веществом под излучением. Выделяется, что спектроскопия комбинационного рассеяния света является эффективным методом химического анализа, изучения состава и строения вещества.

Процесс можно описать следующим образом: квант направленного излучения оказывает влияние на молекулу, которая в свою очередь находится в основном или активном состоянии. Если взаимодействие является упругим, то энергетическое состояние молекулы не меняется, и частота рассеянного излучения будет такая же, как падающего. В противном случае происходит обмен энергией между квантом излучения и молекулой, следовательно, возникает рассеянное излучение, которое может быть большей или меньшей частоты. Поскольку частоты колебаний определяются характером внутри- и межмолекулярных связей, то спектр уникален для каждого химического соединения и зависит от его аллотропной формы и агрегатного состояния. Таким образом формируется спектр Рамановской спектроскопии.

Выделяют несколько существенных преимуществ метода КРС. Во-первых, нет нужды изменять физическую и химическую структуру образца, то есть необходимо лишь направить излучение на вещество. Так же метод КРС предоставляет большой объем данных, в который входит как количественная, так и качественная информация об образце.

Еще одна важная особенность КРС – диапазон спектра не зависит от колебательных особенностей, то есть КРС позволяет выбрать наиболее подходящий диапазон волн для определенного вещества и получить наилучшие результаты.

Рамановская спектроскопия широко применяется в различных направлениях, таких как медицина – анализ поведения живых клеток при взаимодействии с препаратами, биология – классификация клеток и микроорганизмов, фармацевтика – сертификация конечного продукта, химическая промышленность – контроль входного сырья, геология и минералогия – анализ происхождения минералов, криминалистика - идентификации взрывчатых и наркотических веществ, и другие.

Характерные нормированные спектры комбинационного рассеяния для новообразований и нормальной кожи представлены на рисунке 1.2.

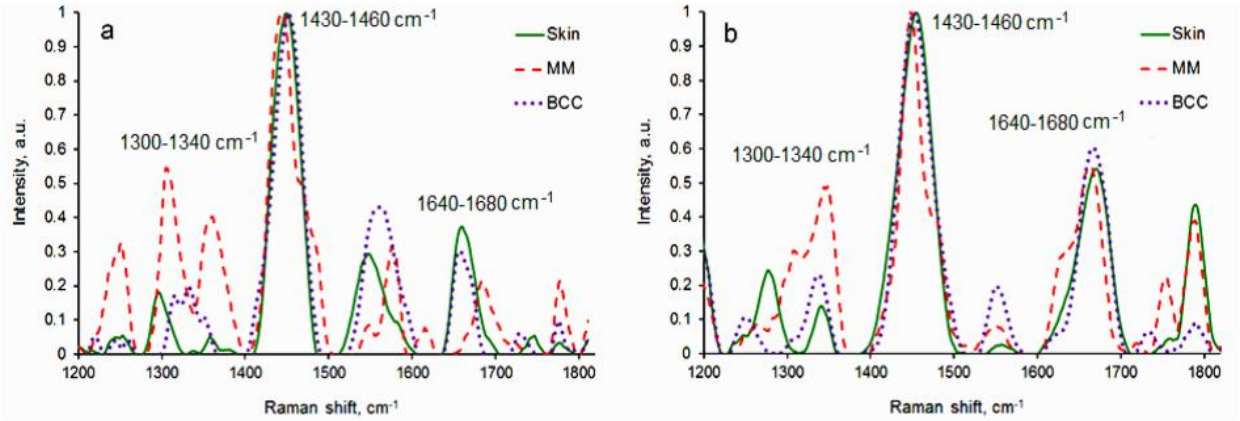

Рисунок 1.2 - Нормализованные спектры комбинационного рассеяния здоровой кожи, меланомы и базальноклеточного рака (a – ex vivo, b – in vivo) в области 1200-1800 $см^{-1}$ [2]

Одной из главных особенностей использования Рамановской спектроскопии является неинвазивный подход диагностики онкологического заболевания кожи.

Опухоль при своем развитии вносит химические и структурные изменения в молекулярный уровень кожи, соответственно, спектр КР может отобразить данные с помощью изменения длины волны падающего света.

Сравнительные анализ спектров здоровой кожи и различных типов опухолей доказал, что дифференциальная диагностика новообразований возможна. [14]

1.6 Анализ разработанных решений

Для подтверждения актуальности исследуемой проблемы изучим разработанные решения в данной области и проанализируем их эффективность.

В статье «Оптическая диагностика злокачественных и доброкачественных новообразований кожи» Ю.А. Христофоровой, И.А. Братченко, Д.Н. Артемьева, О.О. Мякинина, А.А. Морятова, О.И. Каганова, С.В. Козлова, В.П. Захарова исследуется возможность использования автофлуоресцентного анализа для диагностики новообразований кожи. В статье указывается, что «дифференциация различных типов новообразований проводилась с использованием бинарной логистической регрессии. Описанные в статье методы анализа спектральных данных позволяют достичь точности дифференциации образований кожи на уровне 85%».

S.W. Menzies, C. Ingwar, W. McCarthy в своей публикации под названием «A sensitivity and specificity analysis of the surface microscopy features of invasive melanoma» исследуют эпилюминесцентную микроскопию. Для эксперимента были произведены фотографии с использованием иммерсионного масла. Общее число образцов – 164, 62 из них злокачественная меланома, 119 – немеланомые новообразования. Результаты были получены следующие: чувствительность – 92%, специфичность – 71%.

С.В. Козлов, Е.Ю. Неретин, В.В. Куколкина в своей публикации «Диагностика меланомы кожи с использованием экспертной системы» говорят об использовании экспертной системе ПКАД для диагностики новообразований кожи. Результаты были получены следующие: «Чувствительность диагностики составила 93%, специфичность – 79%».

Проанализировав выше указанную литературу, приходим к выводу об актуальности данного исследования. Стоит изучить возможности дифференциации образцов биоткани по результатам спектроскопии комбинационного рассеяния с использованием алгоритмов машинного обучения.

2 АНАЛИЗ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

2.1 Формальное определение задачи классификации

При классификации каждая единица наблюдения относится определённой группе или номинальной категории на основе некоторого качественного свойства. Формально задача классификации ставится следующим образом:

Пусть X— множество описаний объектов, Y— конечное множество номеров (имён, меток) классов. Существует неизвестная целевая зависимость — отображение $y^*: X \to Y$, значения которой известны только на объектах конечной обучающей выборки

$$X^m = \{ x_1, y_1 , \dots, (x_m, y_m)\} \quad (2.1)$$

Требуется построить а лгоритм $a: X \to Y$, способный классифицировать произвольный объект $x \in X$.

Объект — это сущность, представленная набором признаков. Таким образом, объект может быть представлен вектором:

$$x_r = (x_{r1}, x_{r1}, \dots, x_{rn}) \quad (2.2)$$

Все признаки можно поделить на несколько групп по типу их значения:

1) булевые, область значений которых — $\{0, 1\}$;

2) номинальные, область значений которых — конечное подмножество N;

3) порядковые, представляющие собой номинальные признаки, для которых определен линейный порядок;

4) количественные, значение которых — вещественное число.

В математической статистике задачи классификации называются также задачами дискриминантного анализа.

В машинном обучении задача классификации решается с использованием обучения с учителем, поскольку классы определяются заранее и для примеров обучающего множества метки классов заданы. Аналитические модели, решающие задачу классификации, называются классификаторами. [3]

Задача классификации представляет собой одну из базовых задач прикладной статистики и машинного обучения, а также искусственного интеллекта в целом. Это связано с тем, что классификация является одним из наиболее понятных и простых для интерпретации технологий анализа данных, а классифицирующие правила могут быть сформулированы на естественном языке.

2.2 Предобработка входных данных

Исследование тканей кожи проводилось в Самарском национальном исследовательском университете имени академика С.П. Королева при поддержке ГБУЗ Самарского областного клинического онкологического диспансера с использованием лабораторной установки, включающей в себя лазерный источник LML-785.0RB-04 с центральной длиной волны 785 нм (ширина спектральной линии 0,1 нм), спектрометр QE6500.

Важной особенность данной лабораторной установки в том, что происходит одновременная регистрация АФ и КР сигналов по средствам фильтрации оптического пробника. Для регистрации спектров тканей кожи была установлена мощность лазерного излучения 200 мВт. Спектральное разрешение 0,2 нм. Время накопления сигнала 30 с. Регистрация спектров проводилась в диапазоне 780-1000 нм.

Каждый участок кожи подвергался трем независимым измерениям.

Самарским национальным исследовательским университетом имени академика С.П. Королева были проведены исследования порядка 400 пациентов с новообразованиями кожи. Любые исследования не проводились без согласия пациентов. Исследования были одобрены этическим комитетом Самарского государственного медицинского университета. [2]

Данные были получены в виде текстовых файлов, содержащих идентификатор и информацию о пациенте, а также спектр комбинационно рассеяния в виде двух значений – длина волны и полученный спектр (рисунок 2.1).

```
781.11  2.00
781.34  4.33
781.58  3.67
781.81  4.33
782.05  4.67
782.28  4.67
782.52  7.33
782.75  7.67
782.98  8.67
783.22  10.67
783.45  9.00
783.69  13.67
783.92  15.00
```

Рисунок 2.1 - Пример полученных данных (спектр КР)

Так как работу алгоритмов машинного обучение необходимо строить на подготовленных, структурированных данных, была реализована программа на языке программирования Python для подготовки входных данных, а в частности перевода данных в табличный вид формата xlsx. Пример преобразованных данных представлен на рисунке 2.2, где последний столбец – метка класса (О – опухоль, К – кожа), то есть входные данные содержат всего два класса.

2769,67	2616	2587,67	2616,67	2598,67	О
99,67	47	32,33	52,33	35,67	К
133	22,33	35,67	47	28,67	О
6426	2632,33	2614,67	2657,67	2633,33	К
3240	2643,67	2609,67	2671,67	2628,67	О
211,67	72,33	42,33	72,67	61	К
351,33	73	23	88,33	37	О
324	68	63	68,67	82	К
317,33	60	81,67	72	78	О
190,67	55,67	75,67	46,33	81,67	О
287,33	76	78,33	75,33	110,67	К

Рисунок 2.2 - Пример обработанных данных в табличном виде

Далее данные необходимо нормализовать, так как мы наблюдаем разные диапазоны значений. Процесс нормализации данных подразумевает замену номинальных признаков так, чтобы значение каждого из них находилось в диапазоне от 0 до 1. Библиотека Sklearn позволяет нормализовать данные с помощью встроенной функции sklearn.preprocessing.normalize.

Следующим этапом предобработки данных является разделение датасета на тестовый и тренировочный, это позволит трезво оценить работу алгоритмов машинного обучения. Библиотека Sklearn так же позволяет сделать это с помощью встроенной функции train_test_split, в который укажем размер тестовых данных – 20%.

2.3 Алгоритмы машинного обучения

Машинное обучение - это категория алгоритмов, которая позволяет программным приложениям становиться более точными в прогнозировании результатов в каких-либо направлениях деятельности. Основная задача машинного обучения заключается в создании алгоритмов, которые могут получать входные данные и использовать статистический анализ для прогнозирования выходных данных, а также в ситуациях обновления выходных данных по мере поступления новых входных значений.

Процессы машинного обучения аналогичны процессам интеллектуального анализа данных и прогнозного моделирования. Данные методы требуют наличие начальных данных для поиска шаблонов и соответствующей настройки дальнейших действий программы.

Алгоритмы машинного обучения часто классифицируются, как контролируемые или неконтролируемые. Контролируемые алгоритмы требуют, чтобы эксперт или аналитик данных указал алгоритму правильные связи между входными и выходными значениями, также предоставляя обратную связь о точности прогнозов во время обучения алгоритмов, то есть обучение с учителем. Эксперт, анализирующий данные, определяет какие параметры или особенности обработки данных должна использовать модель машинного обучения. После процесса обучения алгоритм применит «полученные знания» на новых входных данных. [7]

Неконтролируемые алгоритмы не нуждаются в обучении с данными о результатах. Вместо этого они используют итеративный подход, называемый глубоким обучением, для анализа данных и получения выводов, то есть

обучение без учителя. Неконтролируемые алгоритмы обучения, называемые нейронными сетями, используются для более сложных задач обработки, чем контролируемые системы обучения, включая распознавание изображений, преобразование речи в текст и генерацию естественного языка. Нейронные сети обрабатывают миллионы входных значений, формируя при этом ассоциации и корреляции. После самостоятельного обучения алгоритм может применить «свои знания» на новых входных значениях. [22]

Scikit-learn - бесплатная библиотека машинного обучения для языка программирования Python. Библиотека включает в себя различные алгоритмы классификации, регрессии и кластеризации. Scikit-learn организован на применении стека SciPy (Scientific Python), включающий в себя:

- NumPy: многомерные массивы и матрицы;
- SciPy: библиотека, предназначенная для выполнения научных и инженерных расчётов;
- Matplotlib: библиотека на языке программирования Python для визуализации данных с помощью 2D и 3D графики;
- IPython: интерактивная оболочка, которая предоставляет дополнительный командный синтаксис, подсветку кода и автоматическое дополнение;
- Sympy: библиотека символьных вычислений;
- Pandas: высокоуровневая Python библиотека для анализа данных. [17]

Для обучения алгоритмов классификации воспользуемся облачный сервис Google Colaboratory с интерактивной оболочкой Jupyter Notebook и библиотекой Scikit-learn v0.21.2.

Google Colaboratory — облачный сервис, направленный на упрощение исследований в области машинного и глубокого обучения. Используя Colaboratory, можно получить удаленный доступ к машине с подключенной видеокартой и тензорным процессором, что, соответственно, ускоряет работу алгоритмов.

Разберем наиболее популярные модели машинного обучения.

Алгоритм k-ближайших соседей (KNN).

Алгоритм направлен на отбор k-ближайших объектов среди всего множества. Важным фактором является проблема выбора коэффициента ближайших соседей, то есть элементов, которые считаются близкими.

Алгоритм состоит из нескольких шагов, первый – выбор значения количества ближайших соседей. Второй шаг демонстрирует определенное количество записей с наименьшим расстоянием до вектора. Расстояние между объектами получаем с помощью Евклидова расстояния:

$$D_E = \sqrt{\sum_i^n (x_i - y_i)^2}, \qquad (2.3)$$

где n – количество атрибутов.

Графическое отображение работы алгоритма k-ближайших соседей на примере датасета Iris можно увидеть на рисунке 2.3.

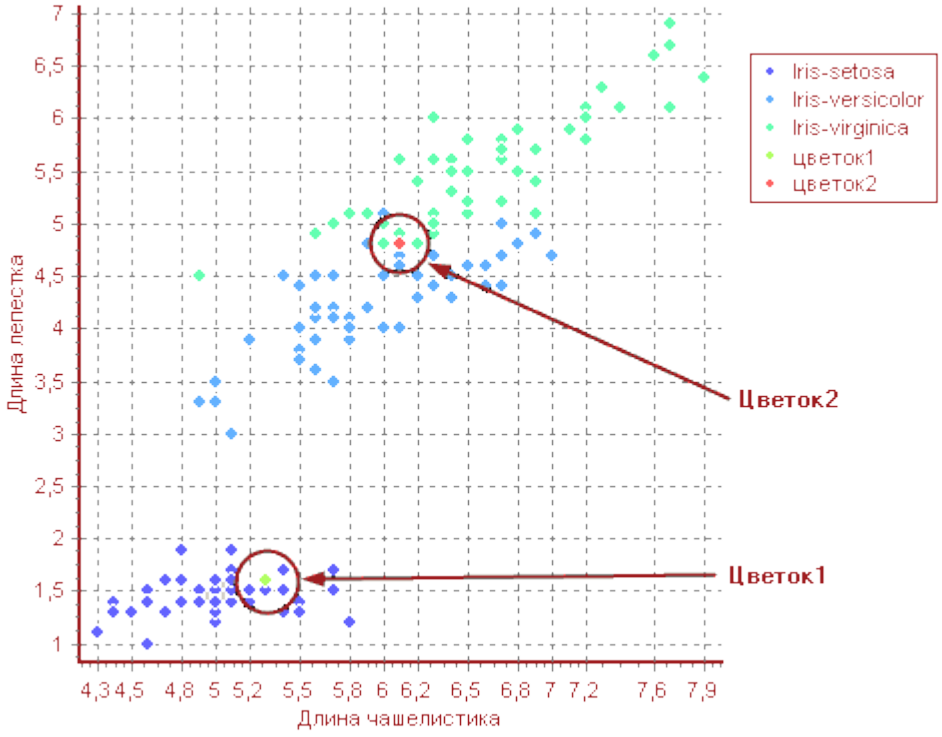

Рисунок 2.3 – Пример классификации методом k-ближайших соседей

Отбор значения k-ближайших соседей вызывает двойственную ситуацию. Если параметр увеличивать, то мы будем наблюдать рост качества

классификации, однако разделения между классами становятся нечеткими. Выбор значения k чаще всего подбирают методом перекрёстной проверки.

Метод k-ближайших соседей вполне может продемонстрировать высокие результаты, даже несмотря на свою алгоритмическую простоту. Однако есть и минус у данного алгоритма – его вычислительная трудоемкость. [10]

Библиотека Sklearn позволяет воспользоваться алгоритмом k-ближайших соседей (KNN) с помощью KNeighborsClassifier, подключаемый с помощью команды from sklearn.neighbors import KNeighborsClassifier. Далее создается классификатор, в который передается тренировочные данные.

Классификатор Роше (Rocchio) — один из самых простых алгоритмов классификации.

Введем следующие обозначения:

1. POS_i – множество элементов, принадлежащих классу c_i

$$POS_i = \{d_j \in Tr | \Phi\ d_j, c_j\ = 1\} \quad (2.4)$$

2. NEG_i – множество элементов, не принадлежащих классу c_i

$$NEG_i = \{d_j \in Tr | \Phi\ d_j, c_j\ = 0\} \quad (2.5)$$

3. $c_\iota -$ взвешенный центроид для класса c_i

$$c_\iota = < w_{1i}, \dots, w_{|T|i} > \quad (2.6)$$

Для каждого класса вычисляется взвешенный центроид по формуле:

$$c_\iota = \alpha \frac{1}{|POS_i|} \sum_{d \in POS_i} d - y \frac{1}{|NEG_i|} \sum_{d \in NEG_i} d \quad (2.7)$$

Здесь α и y — главные параметры, отражающие на сколько важно влияние положительных и отрицательных примеров.

Алгоритм подразумевает нахождение центроидов для всех классов и вычисление степени принадлежности элементов к данным центроидам. Функция расстояния в данном случае – косинус между векторами.

Тогда классифицирующая функция определяется как величина, обратная расстоянию элемента $d \in D$ до центроида класса c_i:

$$CSV\ \ d, c_i\ \ = \frac{1}{d \cdot c_i} \tag{2.8}$$

Преимущества метода:

1. Метод просто реализуем и существует много готовых реализаций

2. При увеличении числа примеров центроид легко можно пересчитать. Данная возможность используется в задачах фильтрации, когда можно указывать системе какие элементы классифицированы правильно.

3. Данный метод дает положительные результаты в особенности в близких по расстоянию предметов.

Недостатки метода:

Если расстояние между элементами внутри класса велико, тогда классификатор любому классифицируемому элементу будет присваивать малое значение, и не один элемент не попадет в эту категорию. Расстояние между элементами внутри класса также может быть большим. [16]

Деревья решений. Алгоритм состоит в разделении исходных данных на группы, пока не будут получены однородные (или почти однородные) множества.

В методе деревьев решений применяется принцип «рекурсивного деления». По-другому эту стратегию называют «Разделяй и властвуй». Суть данного метод, состоит в выборке корня элемента, процесс завершается пока не будут достигнуты критерии остановки. В узлах из корня выбирается элемент, значение которого используется для разделения всех данных на 2 класса. Процесс продолжается до тех пор, пока не будет достигнут критерий остановки.

В результате алгоритм предоставит нам «чистые» группы, которые состоят из объектов одного класса. Чистота группы определяется с помощью энтропии.

Мера неопределенности в смыслах информации и мат. статистики называется энтропией. Энтропия вычисляется по формуле:

$$Entropy\ S\ \ = \ \ \sum_{i=1}^{c} -p_i \log_2(p_i) \tag{2.9}$$

Здесь p_i – вероятность нахождения системы в состоянии i.

На следующем шаге необходимо выбрать признак, который позволит получить наиболее чистую группу (при разбиении). Для этого вычисляется значение усиления информации (InfoGain).

$$InfoGain\ F\ = Entropy\ S_1\ - Entropy(S_2) \qquad (2.10)$$

Здесь $Entropy(S_2)$ – сумма энтропии групп после разбиения.

Стоит упомянуть, что наибольшее значение InfoGain означает, что данный признак подходит для разбиения и приведет к получению наиболее чистой группы. [11]

Библиотека Sklearn позволяет воспользоваться алгоритмом деревьев решений с помощью DecisionTreeClassifier, подключаемый с помощью команды from sklearn.tree import DecisionTreeClassifier. Далее создается классификатор, в который передается тренировочные данные.

Случайный лес — алгоритм машинного обучения, заключающийся в использовании комитета (ансамбля) решающих деревьев. Особенность алгоритма заключается, в том, что по одному данные ансамбли не дают высоких результатов, но их объединение показывает высокое качество классификации [12]

Наиболее распространённый способ построения деревьев комитета следующий (бэггинг):

1. Сгенерируем случайную подвыборку с повторениями размером N из обучающей выборки. (Таким образом, некоторые образцы попадут в неё два или более раза, некоторые образцы не попадут совсем).

2. Далее строим решающее дерево, при создании узла выберем признаки, по которым произведем разбиение, выберем лучшие признаки по критерию Джини или критерию прироста информации.

3. Строим дерево до конца данных, отсечение не проводим.

Объект отправляется в тот класс, за который проголосовало наиболее кол-во деревьев.

Главная цель при оптимизации числа деревьев – минимизация ошибки классификации на тестовых данных. Если ошибки нет, то тоже самое проделываем с объектами, не попавшими в обучающие данные из-за повторений.

Библиотека Sklearn позволяет воспользоваться алгоритмом случайного леса с помощью RandomForestClassifier, подключаемый с помощью команды from sklearn.ensemble import RandomForestClassifier. Далее создается классификатор, в который передается тренировочные данные.

Логистическая регрессия.

Данный метод подразумевает, что функция возвращает вероятность принадлежности объекта к классу, в отличие от обычной регрессии.

Основная идея логистической регрессии заключается в том, что пространство исходных значений может быть разделено линейной границей (т.е. прямой) на две соответствующих классам области.

Классификацию можно проводить, используя данную функцию:

$$Q\ w\ =\ \ {}^{L}_{i=1}\ a\ w, x_i\ \neq y\ x_i\ \ \rightarrow min \qquad (2.11)$$

Можно так же получить функцию эмпирического риска, если воспользоваться линейной классификацией:

$$Q\ w\ =\ \ {}^{L}_{i=1}\ f\ \ \ {}^{k}_{j=0}\ w_j \cdot x_j\ \ \neq y\ x_i\ \ \rightarrow min \qquad (2.12)$$

Расстояние от гиперплоскости до объекта в линейной классификации называют отступом. Отступ будет положительным, если объект классифицирован правильно, отрицательным, если классификация неверна. Формулы определения отступа в бинарных классификациях:

$$M\ w, x_i, y_i\ \ =\ y_i \cdot (\ \ {}^{k}_{j=0}\ w_j \cdot x_j) \qquad (2.13)$$

Библиотека Sklearn позволяет воспользоваться алгоритмом логистической регрессии с помощью LogisticRegression, подключаемый с помощью команды from sklearn.linear_model import LogisticRegression. Далее создается классификатор, в который передается тренировочные данные.

К этой же группе линейных классификаторов относится метод опорных векторов (SVM).

Данный метод так же называют методом с максимальным зазором, так как происходит постоянное уменьшение эмпирической ошибки и увеличение зазора.

Метод опорных векторов основан на переходе от исходного вектора в пространство высокой размерности, где проводится поиск гиперплоскости с максимальным зазором. Подразумевается, что по сторонам найденной гиперплоскости строятся еще две параллельные гиперплоскости, и чем больше расстояние между этими гиперплоскостями, тем меньше ошибка. [16]

Метод SVM работает с абстрактной векторной моделью предметной области. Это позволяет применять SVM для решения различных задач машинного обучения. SVM используется для задач классификации текстов, распознавания образов и речи.

Преимущества метода:

1. Метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет единственное решение

2. Метод находит разделяющую полосу максимальной ширины, что позволяет в дальнейшем осуществить более уверенную классификацию

Недостатки метода:

1. Метод чувствителен к шумам и стандартизации данных

2. Не существует общего подхода к автоматическому выбору ядра в случае линейной неразделимости классов

Еще один из наиболее известных и простых алгоритмов машинного обучения – Наивный байесовский классификатор.

В зависимости от точной природы вероятностной модели, наивные байесовские классификаторы могут обучаться очень эффективно. Во многих практических приложениях для оценки параметров для наивных байесовых моделей используют метод максимального правдоподобия; другими словами,

можно работать с наивной байесовской моделью, не веря в байесовскую вероятность и не используя байесовские методы.

Достоинством наивного байесовского классификатора является малое количество данных необходимых для обучения, оценки параметров и классификации.

Бустинг — это техника построения ансамблей, в которой предсказатели построены не независимо, а последовательно.

Это техника использует идею о том, что следующая модель будет учится на ошибках предыдущей. Они имеют неравную вероятность появления в последующих моделях, и чаще появятся те, что дают наибольшую ошибку. Предсказатели могут быть выбраны из широкого ассортимента моделей, например, деревья решений, регрессия, классификаторы и т.д. Из-за того, что предсказатели обучаются на ошибках, совершенных предыдущими, требуется меньше времени для того, чтобы добраться до реального ответа. Но мы должны выбирать критерий остановки с осторожностью, иначе это может привести к переобучению. [8]

Градиентный бустинг — это техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений.

Давайте обратимся к математике градиентного бустинга. Пусть, например, в качестве функции потерь будет среднеквадратичная ошибка (MSE):

$$Loss = MSE = \quad y_i - y_i^p \ ^2 \tag{2.14}$$

Мы хотим, чтобы построить наши предсказания таким образом, чтобы MSE была минимальна. Используя градиентный спуск и обновляя предсказания, основанные на скорости обучения (learning rate), ищем значения, на которых MSE минимальна.

$$y_i^p = y_i^p + \alpha * \frac{\delta \quad y_i - y_i^p \ ^2}{\delta y_i^p} \tag{2.15}$$

$$y_i^p = y_i^p - \alpha * 2 * \quad y_i - y_i^p \tag{2.16}$$

Итак, мы просто обновляем предсказания таким образом, что сумма наших отклонений стремилась к нулю и предсказанные значения были близки к реальным.

Еще один известный и эффективный алгоритм машинного обучения – AdaBoost. AdaBoost – это алгоритм усиления классификаторов.

Усиление – это ансамблевый алгоритм обучения, который берет множество алгоритмов обучения, например, деревья решений, и объединяет их. Целью является взять набор или группу слабых классификаторов и объединить их в один сильный. [13]

Слабый классификатор производит классификацию с точностью чуть выше шанса. Популярный пример слабого классификатора – это так называемый «решающий пень» – одноуровневое дерево решений.

Сильный классификатор, наоборот, имеет гораздо большую точность. Самым распространенным примером сильного классификатора является SVM.

Этот метод требует обучения, поскольку на каждой итерации по размеченному набору дан данных тренируются более слабые классификаторы.

Это отличный способ автоматической настройки классификатора, поскольку каждая успешная итерация AdaBoost корректирует вес лучшего классификатора. Нам необходимо только указать количество итераций.

И наконец, AdaBoost очень гибкий и легко приспосабливающийся к условиям задачи алгоритм. AdaBoost может работать с любыми обучающимися алгоритмами и с большим количеством разнообразных данных.

2.4 Сравнительный анализ моделей машинного обучения

В данной работе было использовано 16 моделей классификаторов из библиотеки Sklearn.

- LogisticRegression,

- SVC,
- NuSVC,
- LinearSVC,
- GaussianNB,
- BernoulliNB,
- KNeighborsClassifier,
- NearestCentroid,
- GaussianProcessClassifier,
- DecisionTreeClassifier,
- BaggingClassifier,
- RandomForestClassifier,
- ExtraTreesClassifier,
- AdaBoostClassifier,
- GradientBoostingClassifier,
- MLPClassifier,

Далее проанализируем работу моделей машинного обучения.

Исходные данные представлены в табличном виде, данные содержат в себе 1734 записи, каждая запись относится к одному из двух классов.

Для работы с данными создадим датасет, где dataset =np.array(dataFrame.iloc[:, 0:1044]), а target = dataFrame.loc[:,['label']] (полный код программы указан в Приложении А), то есть последний столбец мы будет использовать в качестве метки класса, укажем в обучающей выборке какие элементы относятся к классам «Кожа» и «Опухоль». На тестовых данных метка классу будет отсутствовать.

Далее нормализуем и разделим данные на тренировочные и тестовые. Размер тестовых данных составляет 20% от общей выборки.

Далее для удобства работы создадим два списка из наименований всех классификаторов.

Обучение классификаторов происходит с помощью встроенной функции библиотеки Sklearn fit().

Так же воспользуемся встроенной функцией для начальной оценки точности классификации score(), далее воспользуемся иными метриками оценки качества классификации, о которых говорили ранее.

Обратим внимание, что все классификаторы на данном этапе обучаются на стандартных параметрах, которые указаны в библиотеке Sklearn.

Код программы обучения классификаторов и записи точности выглядит так:

```
for classifier in classifiers:
    model = classifier.fit(x_train, y_train)
    scores.append(model.score(x_test, y_test))
```

После обучения классификаторов выведем значения точности, для начальной оценки алгоритмов машинного обучения:

```
for i, score in enumerate(scores):
    print(names[i], 'score =', score)
```

Полный код программы указан в Приложении А.

Полученные результаты точности отобразим в таблице 2.1.

Таблица 2.1 – Результаты точности классификации моделей со стандартными параметрами

Модель машинного обучения	Точность на стандартных параметрах
LogisticRegression	0.6685878962536023
SVC	0.5302593659942363
NuSVC	0.5504322766570605
LinearSVC	0.6945244956772334
GaussianNB	0.5734870317002881
BernoulliNB	0.5360230547550432
KNeighborsClassifier	0.7089337175792507
NearestCentroid	0.6109510086455331
GaussianProcessClassifier	0.6685878962536023

DecisionTreeClassifier	0.8443804034582133
BaggingClassifier	0.8645533141210374
RandomForestClassifier	0.8040345821325648

Продолжение таблицы 2.1

Модель машинного обучения	Точность на стандартных параметрах
ExtraTreesClassifier	0.861671469740634
AdaBoostClassifier	0.7435158501440923
GradientBoostingClassifier	0.829971181556196
MLPClassifier	0.6253602305475504

Для более удобного анализа полученных результатов составим диаграмму, отображающую разницу точности классификаторов (рисунок 2.4).

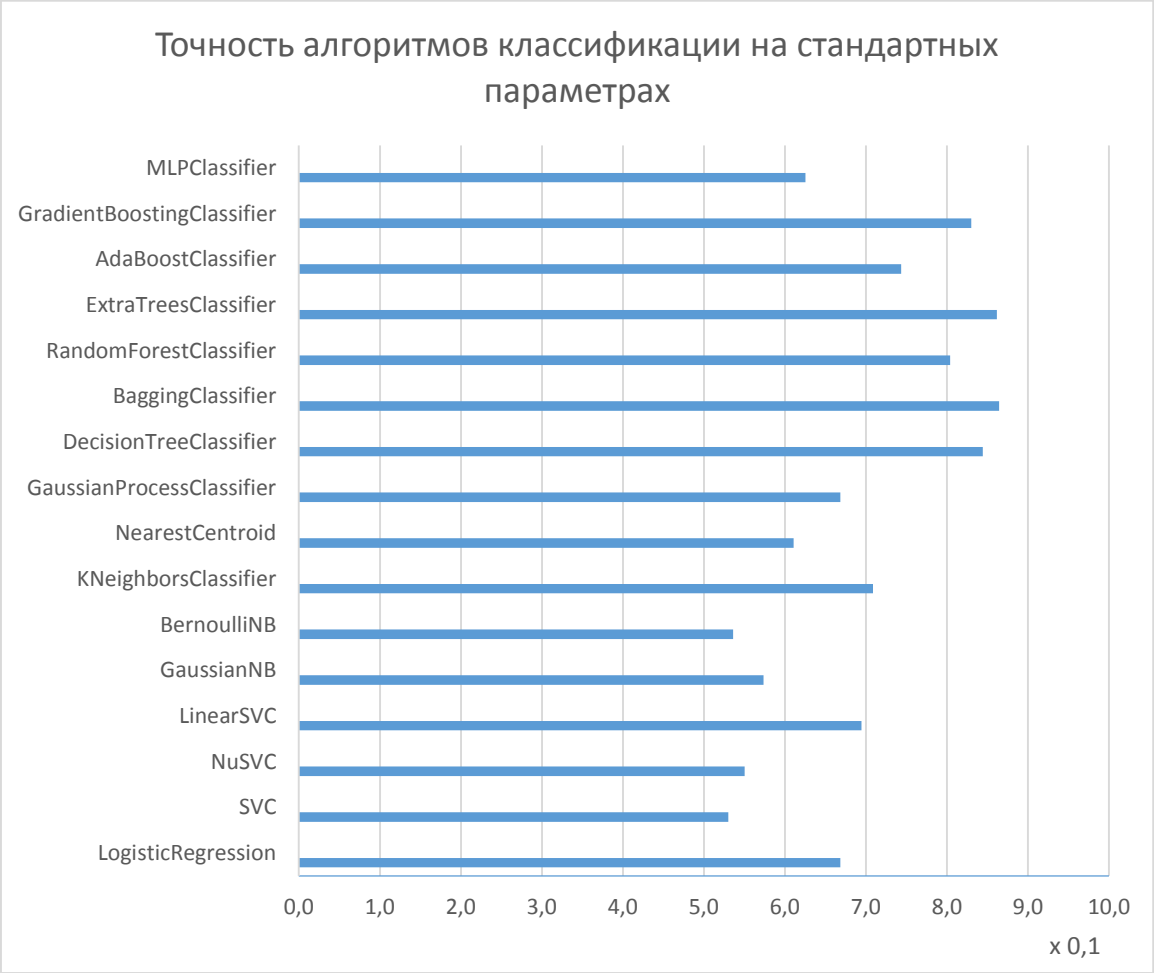

Рисунок 2.4 – Диаграмма точности классификации моделей со стандартными параметрами

Проанализировав данные показатели, можно сказать, что больше половины классификаторов показывают точность менее 70%, в частности, это LogisticRegression, SVC, NuSVC, LinearSVC, GaussianNB, BernoulliNB, NearestCentroid, GaussianProcessClassifier и MLPClassifier. Установим пороговое значение в 70%, чтобы провести отбор классификаторов в следующий этап.

Использование классификаторов с точность менее 70% не рационально, так как случайное распределение элементов по классам может предоставить точность близкую к этому порогу.

К моделям машинного обучения, показавшим высокий результат точности, относятся DecisionTreeClassifier, BaggingClassifier, RandomForestClassifier, ExtraTreesClassifier, AdaBoostClassifier и GradientBoostingClassifier. Данные классификаторы уже на стандартных параметрах показывают высокое качество классификации.

Наиболее точным оказался BaggingClassifier. Можно предположить, что это связано с тем, что при бэггинге используется целая композиция алгоритмов, в которой элементарные классификаторы обучаются и работаю независимо друг от друга. Они не исправляют ошибки друг друга, а компенсируют их при голосовании. Бэггинг в свою очередь может снизить процент ошибки классификации, если высока дисперсия ошибки базового метода.

2.5 Подбор параметров моделей машинного обучения

Следующий этап – подбор параметров для улучшения качества классификации и перекрестная проверка.

Для того, чтобы процесс обучения модели прошел максимально успешно, необходимо чтобы классы в обучающем множестве были в одинаковой пропорции, так же необходимо иметь достаточный объем данных, иначе один из классов окажется доминирующим. Чтобы избежать

подобных ситуаций, воспользуемся перекрестной проверкой (кросс-валидация). [17]

Кросс-валидация подразумевает разделение исходных данных на несколько равных блоков. Если разбиение произвели на 5 блоков, то обучение проходит на 4-х блоках, а на 5-ом блоке проводим тестирование. Данная операция проводится 5 раз, причем каждый раз выбирается новый блок для тестирования (рисунок 2.5).

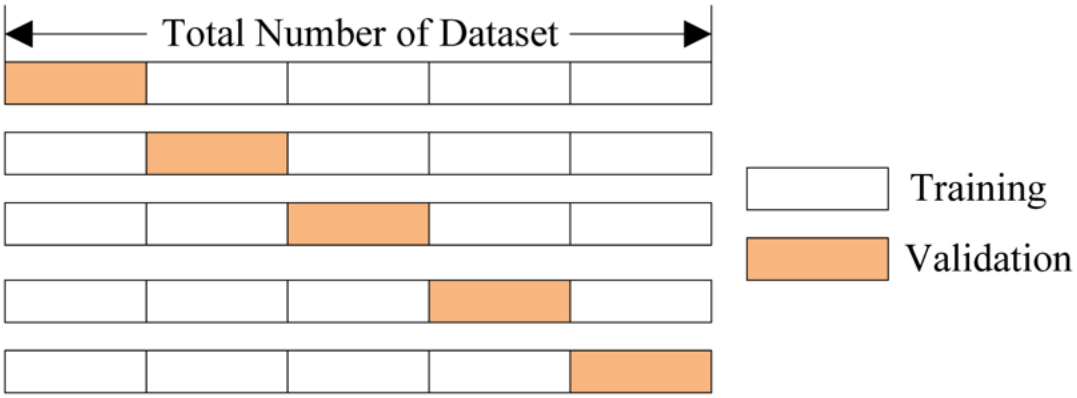

Рисунок 2.5 – Схема работы перекрестной проверки

Перекрестная проверка имеет два основных преимущества перед применением одного множества для обучения и одного для тестирования модели:

- Распределение классов оказывается более равномерным, что улучшает качество обучения.

- Если при каждом проходе оценить выходную ошибку модели и усреднить ее по всем проходам, то полученная оценка будет более достоверной.

В машинном обучении оптимизация или настройка алгоритма - задача выбора набора оптимальных гиперпараметров для обучения. Гиперпараметр - это параметр, значение которого используется для управления процессом обучения.

Одна и та же модель машинного обучения может иметь различные ограничения, веса и скорости обучения. Эти меры называются гиперпараметрами и должны быть настроены таким образом, чтобы модель

могла оптимально решить поставленную задачу. Для оптимизации используют кортеж, содержащий в себе вариации гиперпараметров. Целевая функция принимает кортеж и возвращает набор гиперпараметров, при которых достигается максимальная эффективность. Библиотека Sklearn содержит в себе подобный алгоритм под названием Grid Search [17]

Для отобранных моделей машинного обучения зададим множество параметром для поиска по сетке и запустим процедуру поиска лучших параметров с помощью встроенной функции GridSearchCV.

Например, для алгоритма DecisionTreeClassifier укажем функции качества оценки предсказательных моделей – Коэффициент Джини, Энтропия, максимальную глубину дерева, минимальное количество выборок, необходимых для разделения внутреннего узла, и минимальное кол-во объектов в листе.

Таким же образом, опираясь на специфичность входных данных, поберем множество параметров для оставшихся классификаторов.

Далее в таблице 2.2 отобразим множество параметров, которые мы использовали для поиска, и параметры, которые были выбраны наилучшими.

Таблица 2.2 – Множество параметров для поиска по сетке и множество параметров, выбранных наилучшими.

Модель машинного обучения	Множество параметров для поиска по сетке	Лучшие параметры
KNeighborsClassifier	'weights' : ('uniform', 'distance'), 'n_neighbors' : (1, 3, 5, 7, 9),	'n_neighbors': 9, 'weights': 'distance'
DecisionTreeClassifier	'criterion' : ('gini', 'entropy'), 'max_depth' : (10, 30, 50, 70, 90, None), 'min_samples_split' : (2, 3,	'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2

	4, 5, 6), 'min_samples_leaf' : (1, 2, 3, 4, 5),	

Продолжение таблицы 2.2

BaggingClassifier	'n_estimators' : (5, 10, 15, 20, 25), 'max_samples' : (0.2, 0.4, 0.6, 0.8, 1.0), 'max_features' : (0.2, 0.4, 0.6, 0.8, 1.0),	'max_features': 0.2, 'max_samples': 1.0, 'n_estimators': 25
RandomForestClassifier	'n_estimators' : (5, 10, 15, 20, 25), 'criterion' : ('gini', 'entropy'), 'min_samples_split' : (2, 3, 4, 5, 6), 'min_samples_leaf' : (1, 2, 3, 4, 5),	'criterion': 'gini', 'min_samples_leaf': 3, 'min_samples_split': 4, 'n_estimators': 25}
ExtraTreesClassifier	'criterion' : ('gini', 'entropy'), 'max_depth' : (10, 30, 50, 70, 90, None), 'min_samples_split' : (2, 3, 4, 5, 6), 'min_samples_leaf' : (1, 2, 3, 4, 5),	'criterion': 'gini', 'max_depth': 30, 'min_samples_leaf': 1, 'min_samples_split': 2
AdaBoostClassifier	'n_estimators' : (10, 30, 50, 70, 90),	'learning_rate': 1.3, 'n_estimators': 90

	'learning_rate' : (0.3, 0.6, 1, 1.3, 1.6),	

Далее воспользуемся отобранными классификаторами с новыми параметрами для построения ансамбля алгоритмов машинного обучения с помощью встроенной функции VotingClassifier.

3 ФОРМИРОВАНИЕ И ТЕСТИРОВАНИЕ АНСАМБЛЯ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

3.1 Метрики анализа качества классификации

Для оценки качества работы алгоритмов машинного обучения, а, следовательно, оценки качества классификации воспользуемся несколькими численными метриками.

Оценка качества классификации (работы алгоритмов машинного обучения) используют несколько численных метрик.

Accuracy - доля элементов классификации, по которым классификатор принял правильное решение.

Расчет метрики производится по формуле:

$$Accuracy = \frac{P}{N} \quad (3.1)$$

где, P – количество правильно классифицируемых, N – размер обучающей выборки.

Однако Accuracy имеет некоторую особенность. Если классификация смещена в сторону одного из классов, метрика будет устанавливать всем элементам одинаковый вес, что, соответственно, приведет к некорректной оценке классификации.

Для решения данной ситуации использую сбалансированные массивы данных, однако это сказывается на относительной частоте элементов, а данная информация важна для классификации. [23]

Чаще всего для оценки качества классификации используют точность (precision) и полноту (recall).

Точность рассчитывается в пределах одного класса, это отношение элементов, принадлежащих данному классу к общему числу элементов классификации.

Однако для отображения информации о том, все ли правильные ответы возвратил классификатор показывает мера полноты (Recall). Данная мера

описывает то, как эффективно классификатор может «угадать» наибольшее кол-во положительных ответов.

Для расчета данных значений можно воспользоваться таблицей контингентности (таблица 3.1).

Таблица 3.1 - Таблицей контингентности

Категория i		Экспертная система	
		Положительная	Отрицательная
Оценка системы	Положительная	TP	FP
	Отрицательная	FN	TN

Из таблицы можно получить значение, описывающее кол-во верных и неверных решений для элементов данного класса.

- TP — истинное положительное решения;
- TN — истинное отрицательное решение;
- FP — ложное положительное решение;
- FN — ложное отрицательное решение.

В таком случае можно получить формулы определения полноты и точности:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (3.2)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad (3.3)$$

Матрица ошибок (Confusion Matrix) - матрица размера N на N, где N — это количество классов. В данной матрице столбцы - экспертные решения, строки - решения классификатора. Когда мы классифицируем документ из тестовой выборки мы инкрементируем число, стоящее на пересечении строки класса, который вернул классификатор и столбца класса, к которому действительно относится документ.

Используя данную матрицу можно вычислить значения точности и полноты. Точность - отношению соответствующего диагонального элемента матрицы и суммы всей строки класса. Полнота – отношению диагонального элемента матрицы и суммы всего столбца класса:

$$\text{Precision}_c = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{c,i}} \quad (3.4)$$

$$\text{Recall}_c = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{i,c}} \quad (3.5)$$

F-мера. Понятно, что чем выше точность и полнота, тем лучше. Но в реальной жизни максимальная точность и полнота не достижимы одновременно и приходится искать некий баланс. Поэтому, хотелось бы иметь некую метрику, которая объединяла бы в себе информацию о точности и полноте нашего алгоритма. Именно такой метрикой является F-мера.

Precision и Recall дают довольно исчерпывающую характеристику классификатора, причем «с разных углов». Обычно при построении подобного рода систем приходится все время балансировать между двумя этими метриками. Если вы пытаетесь повысить Recall, делая классификатор более «оптимистичным», это приводит к падению Precision из-за увеличения числа ложно-положительных ответов. Если же вы подкручиваете свой классификатор, делая его более «пессимистичным», например, строже фильтруя результаты, то при росте Precision это вызовет одновременное падение Recall из-за отбраковки какого-то числа правильных ответов. Поэтому удобно для характеристики классификатора использовать одну величину, так называемую метрику F1. [7]

F-мера представляет собой гармоническое среднее между точностью и полнотой. Она стремится к нулю, если точность или полнота стремится к нулю.

$$F = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.6)$$

Данная формула придает одинаковый вес точности и полноте, поэтому F-мера будет падать одинаково при уменьшении и точности и полноты. Возможно рассчитать F-меру придав различный вес точности и полноте, если вы осознанно отдаете приоритет одной из этих метрик при разработке алгоритма.

$$F = (\beta^2 + 1)\frac{\text{Precision} \times \text{Recall}}{\beta^2\text{Precision} + \text{Recall}} \quad (3.7)$$

где β принимает значения в диапазоне $0<\beta<1$ если вы хотите отдать приоритет точности, а при $\beta>1$ приоритет отдается полноте. При $\beta=1$ формула сводится к предыдущей, и вы получаете сбалансированную F-меру (также ее называют F1). [13]

F-мера является хорошим кандидатом на формальную метрику оценки качества классификатора. Она сводит к одному числу две других основополагающих метрики: точность и полноту. Имея в своем распоряжении подобный механизм оценки вам будет гораздо проще принять решение о том являются ли изменения в алгоритме в лучшую сторону или нет.

Для оценки качества обучения модели так же необходимо построить кривые валидации и обучения. Такие графики позволят ответить на следующие вопросы:

Является ли достаточной сложность модели, или же ее необходимо упростить?

Необходимо ли добавить больше признаков для обучения?

Удовлетворительно ли кол-во данных для обучения, или же выборку необходимо увеличить?

То есть с помощью кривых обучения и валидации можно сделать вывод о переобучении или недообучении модели.

Для простых моделей тренировочная и валидационная ошибка находятся где-то рядом, и они велики. Это говорит о том, что модель недообучилась: то есть она не имеет достаточное кол-во параметров.

Для сильно усложненных моделей тренировочная и валидационная ошибки значительно отличаются. Это можно объяснить переобучением: когда параметров слишком много либо не хватает регуляризации, алгоритм может "отвлекаться" на шум в данных и упускать основной тренд.

3.2 Формирование и тестирование ансамбля алгоритмов машинного обучения на входных данных

VotingClassifier – модуль из библиотеки Scikit-learn, позволяющий использовать сразу несколько, не похожих между собой моделей классификаторов, объединяй их в один классификатор (рисунок 3.1).

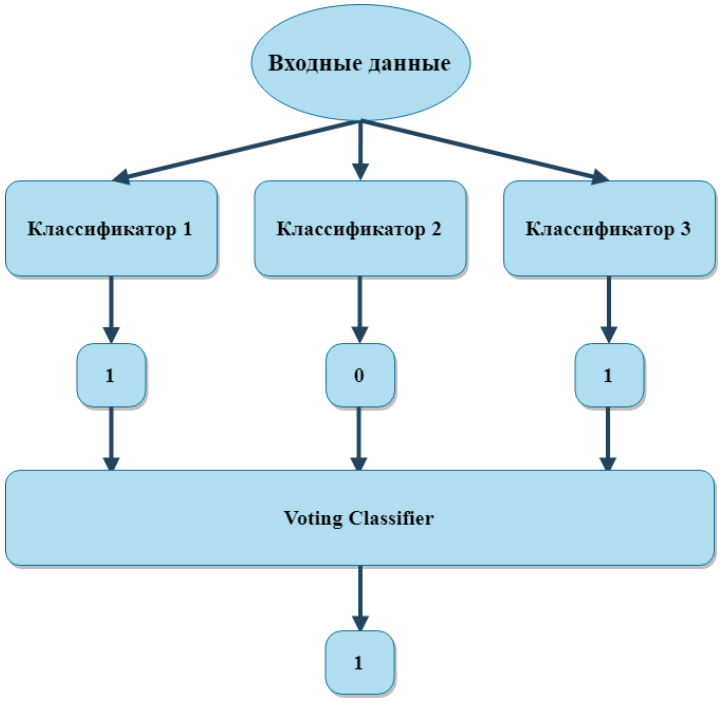

Рисунок 3.1 – Схема работы ансамбля

Использование данного модуля позволит снизить риск переобучения, а также неправильной интерпретации результатов какой-либо одной отдельно взятой модели.

Для формирования ансамбля воспользуемся отобранными классификаторами, для которых уже подобраны лучшие параметры. К моделям машинного обучения, показавшим высокий результат точности, относятся DecisionTreeClassifier, BaggingClassifier, RandomForestClassifier, ExtraTreesClassifier, AdaBoostClassifier и GradientBoostingClassifier.

Проведем тестирование данных классификаторов на входных данных. Результаты тестирования отобразим в таблице 3.2

Таблица 3.2 – Результаты тестирования отобранных моделей с лучшими гиперпараметрами

Модель машинного обучения	Score	Accuracy	Precision	Recall	F1
KNeighborsClassifier	0.88472	0.81385	0.83497	0.80978	0.82202
DecisionTreeClassifier	0.89048	0.80158	0.81241	0.81247	0.81821
BaggingClassifier	0.88760	0.82107	0.84295	0.81797	0.83511
RandomForestClassifier	0.88184	0.81312	0.84032	0.81120	0.83027
ExtraTreesClassifier	0.86743	0.80805	0.83095	0.79343	0.82433
AdaBoostClassifier	0.83573	0.79076	0.80674	0.79896	0.80231
GradientBoostingClassifier	0.91066	0.83838	0.85549	0.84242	0.84536

Теперь построим графики, отображающие зависимость показателя AUC, площадь, ограниченная ROC-кривой, от размера тренировочной выборки.

На рисунке 3.2 отобразим график классификатора KNeighborsClassifier.

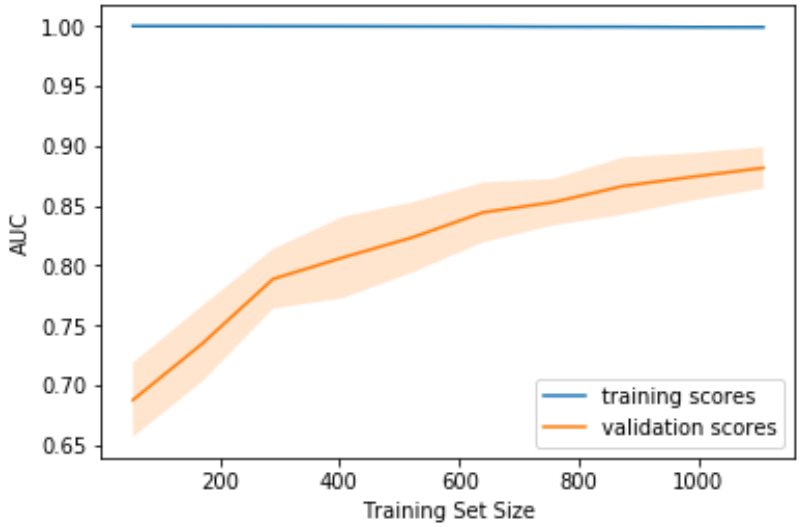

Рисунок 3.2 – график зависимости показателя AUC от размера тренировочной выборки для классификатора KNeighborsClassifier

На рисунке 3.3 отобразим график классификатора DecisionTreeClassifier.

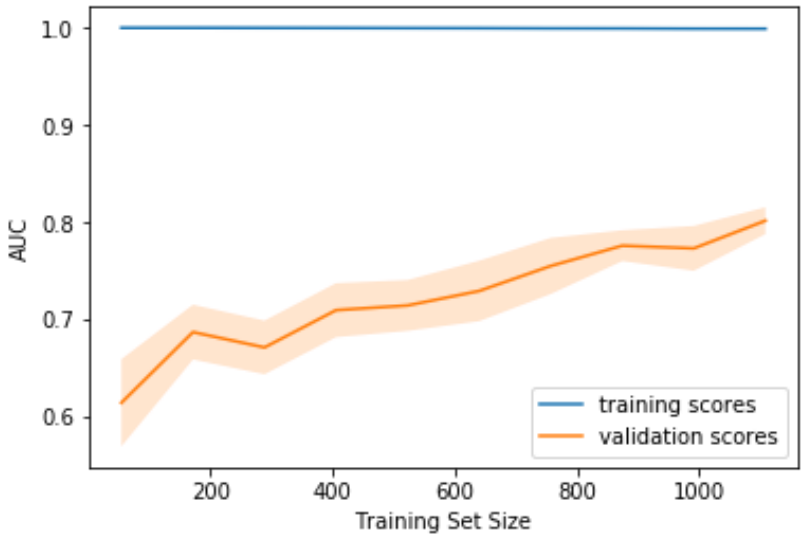

Рисунок 3.3 – график зависимости показателя AUC от размера тренировочной выборки для классификатора DecisionTreeClassifier

На рисунке 3.4 отобразим график классификатора BaggingClassifier.

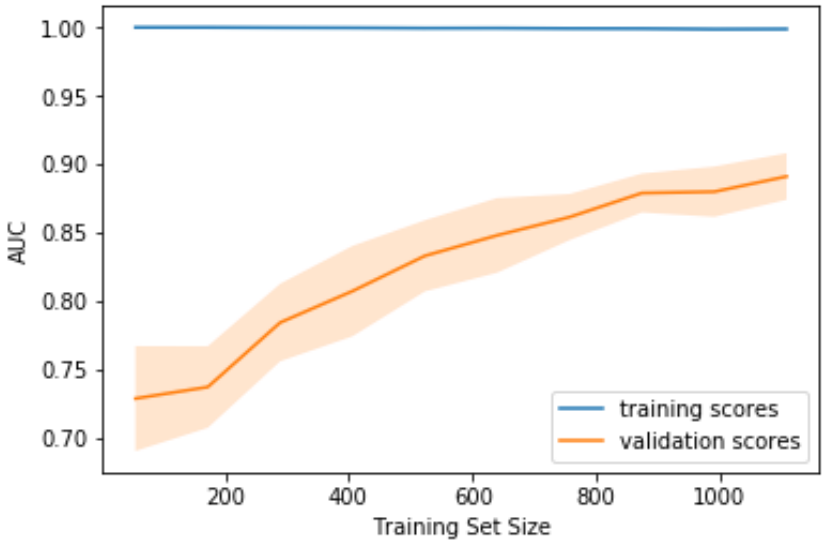

Рисунок 3.4 – график зависимости показателя AUC от размера тренировочной выборки для классификатора BaggingClassifier

На рисунке 3.5 отобразим график классификатора RandomForestClassifier.

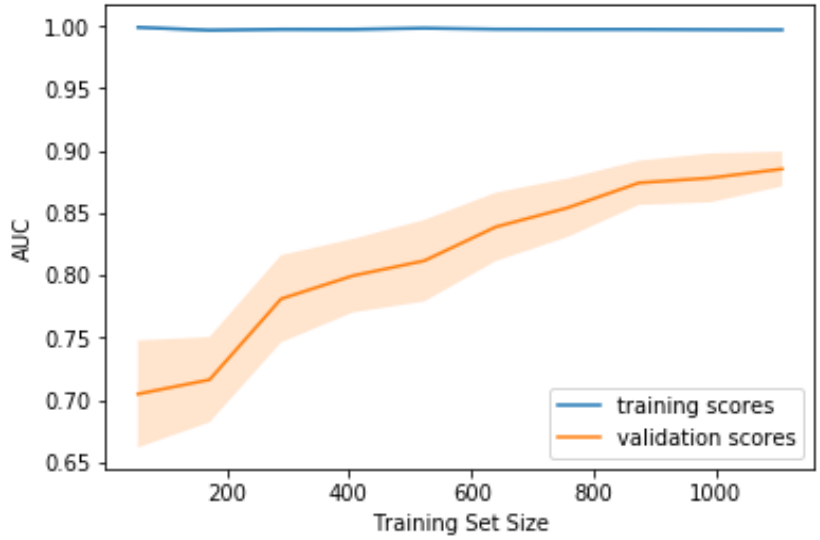

Рисунок 3.5 – график зависимости показателя AUC от размера тренировочной выборки для классификатора RandomForestClassifier

На рисунке 3.6 отобразим график классификатора ExtraTreesClassifier.

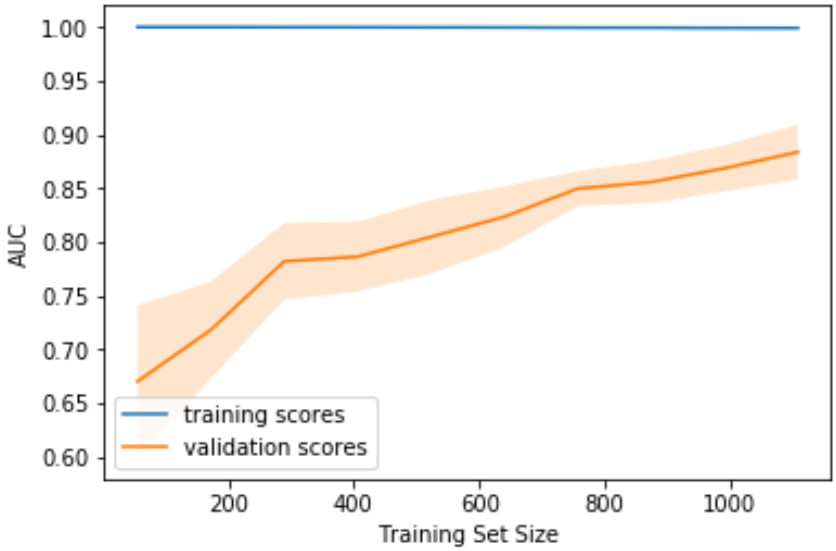

Рисунок 3.6 – график зависимости показателя AUC от размера тренировочной выборки для классификатора ExtraTreesClassifier

На рисунке 3.7 отобразим график классификатора AdaBoostClassifier.

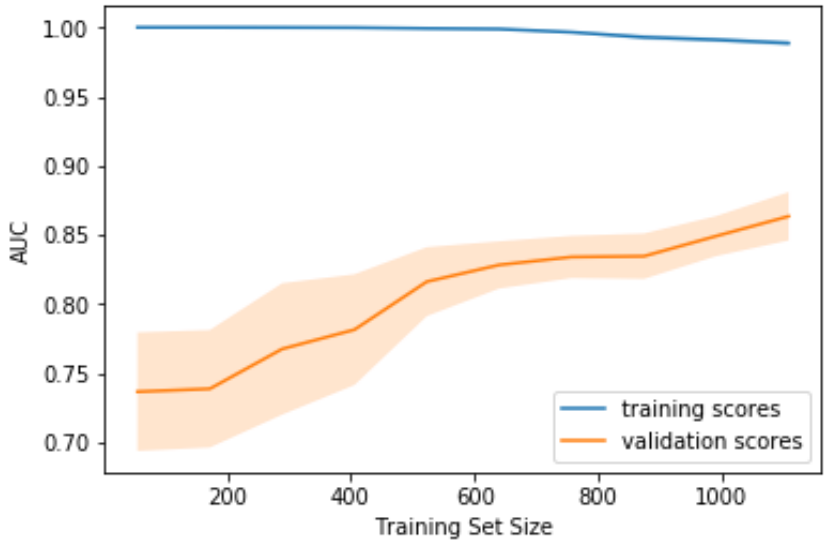

Рисунок 3.7 – график зависимости показателя AUC от размера тренировочной выборки для классификатора AdaBoostClassifier

На рисунке 3.8 отобразим график классификатора GradientBoostingClassifier.

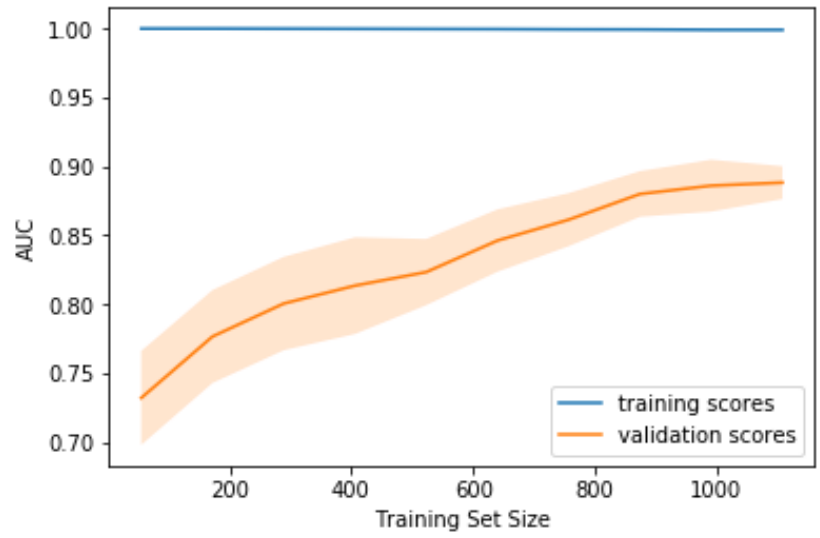

Рисунок 3.8 – график зависимости показателя AUC от размера тренировочной выборки для классификатора GradientBoostingClassifier

Если еще раз обратиться к понятию кривой обучения, то есть график, показывающий результаты на валидации и тренировочной подвыборке в зависимости от количества наблюдений, и взглянуть на полученные графики, то можно с уверенностью сказать, что данных для обучения недостаточно, так как мы явно наблюдаем процесс схождения данных кривых. Увеличение количества входных данных может благоприятно сказаться на качестве

обучения моделей, так как полная сходимость кривых указывает на высокий результат.

Код программы построения кривых обучения указан в приложении Б.

Далее построим валидационные кривые, показывающие, как качество (ROC AUC) на обучающей и проверочной выборке меняется с изменением параметра модели.

Отобразим валидационные кривые для модели KNeighborsClassifier на рисунке 3.9

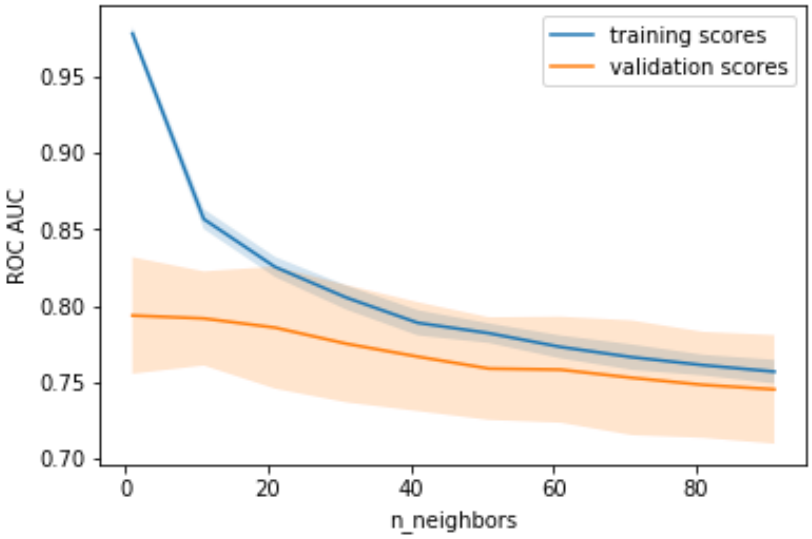

Рисунок 3.9 - Валидационные кривые модели KNeighborsClassifier

Отобразим валидационные кривые для модели DecisionTreeClassifier на рисунке 3.10

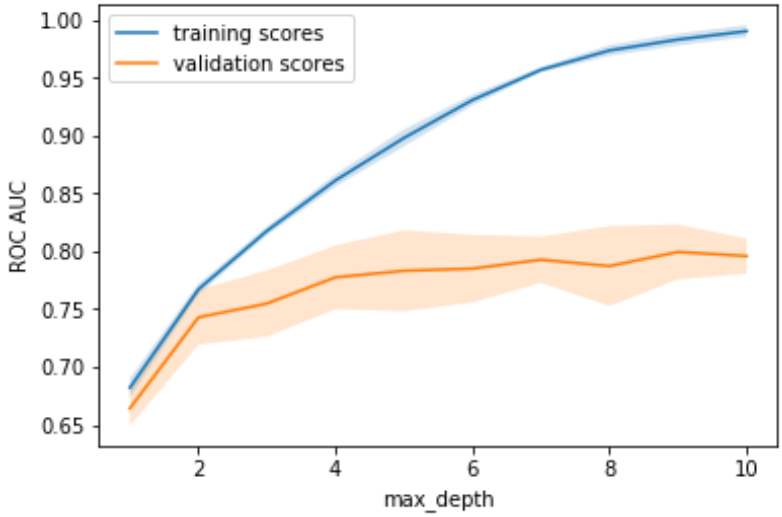

Рисунок 3.10 - Валидационные кривые модели DecisionTreeClassifier

Отобразим валидационные кривые для модели BaggingClassifier на рисунке 3.11

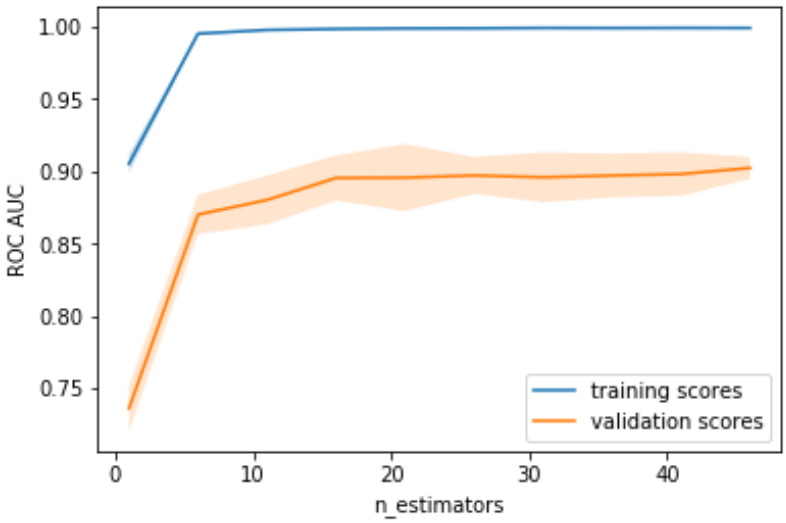

Рисунок 3.11 - Валидационные кривые модели BaggingClassifier

Отобразим валидационные кривые для модели RandomForestClassifier на рисунке 3.12

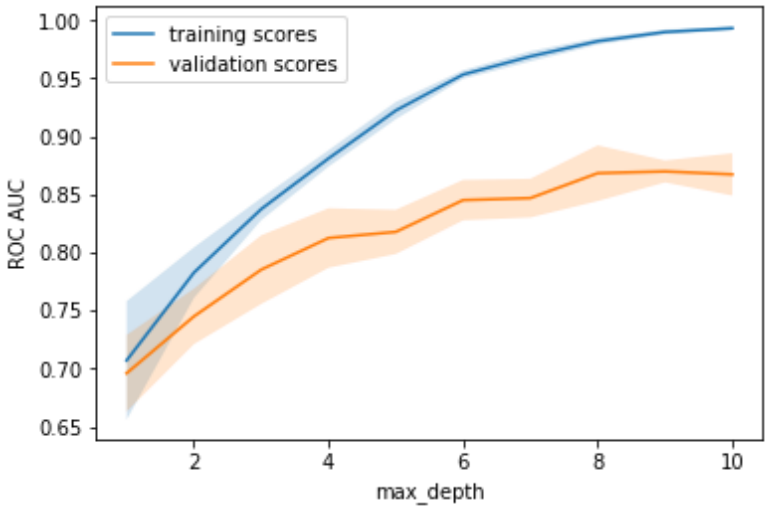

Рисунок 3.12 - Валидационные кривые модели RandomForestClassifier

Отобразим валидационные кривые для модели ExtraTreesClassifier на рисунке 3.13

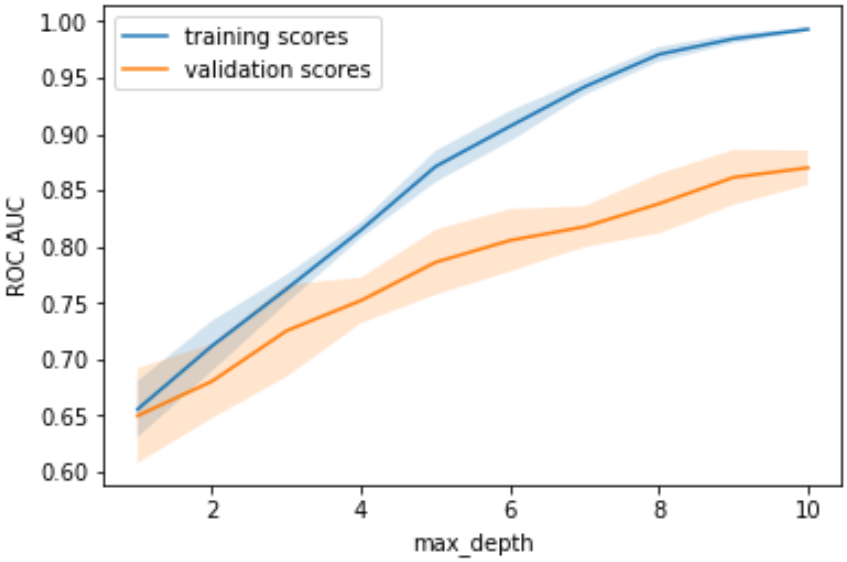

Рисунок 3.13 - Валидационные кривые модели ExtraTreesClassifier

Отобразим валидационные кривые для модели AdaBoostClassifier на рисунке 3.14

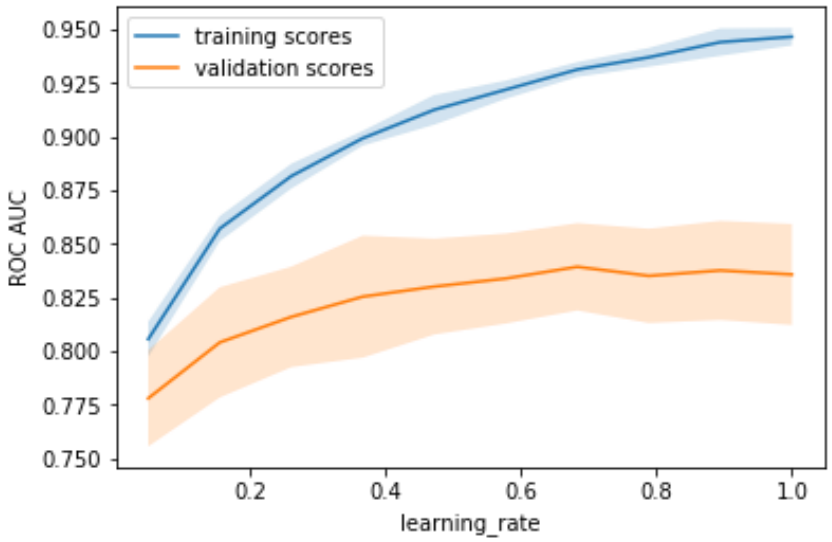

Рисунок 3.14 - Валидационные кривые модели AdaBoostClassifier

Отобразим валидационные кривые для модели AdaBoostClassifier (параметр learning_rate) на рисунке 3.14

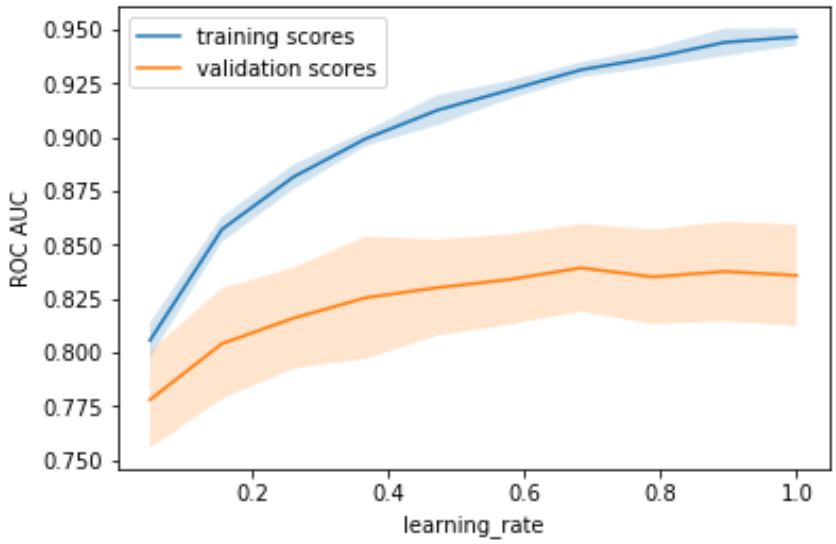

Рисунок 3.14 - Валидационные кривые модели AdaBoostClassifier (параметр learning_rate)

Отобразим валидационные кривые для модели AdaBoostClassifier (параметр n_estimators) на рисунке 3.15

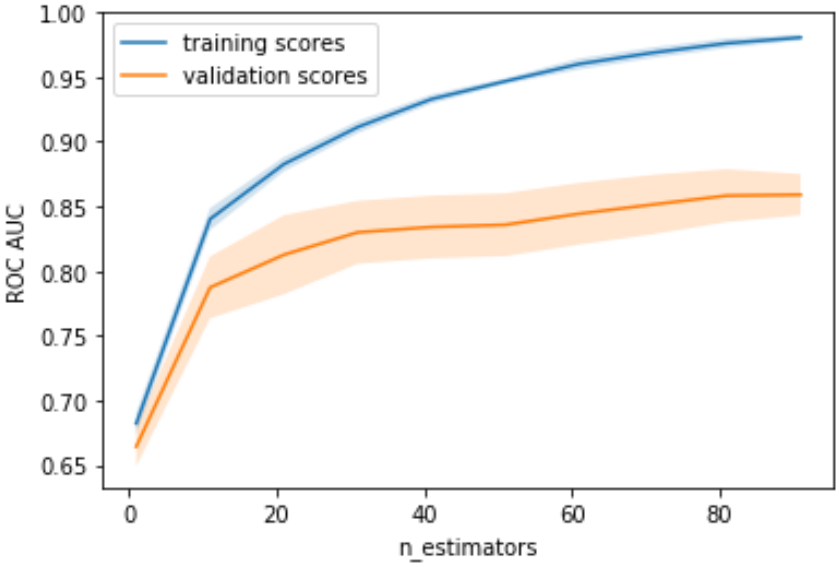

Рисунок 3.15 - Валидационные кривые модели AdaBoostClassifier (параметр n_estimators)

Отобразим валидационные кривые для модели GradientBoostingClassifier (параметр learning_rate) на рисунке 3.16

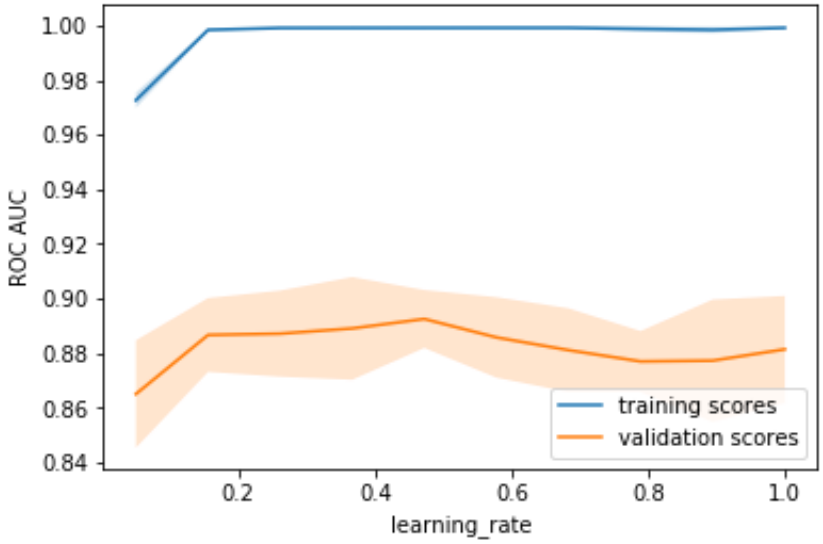

Рисунок 3.16 - Валидационные кривые модели GradientBoostingClassifier (параметр learning_rate)

Отобразим валидационные кривые для модели GradientBoostingClassifier (параметр n_estimators) на рисунке 3.17

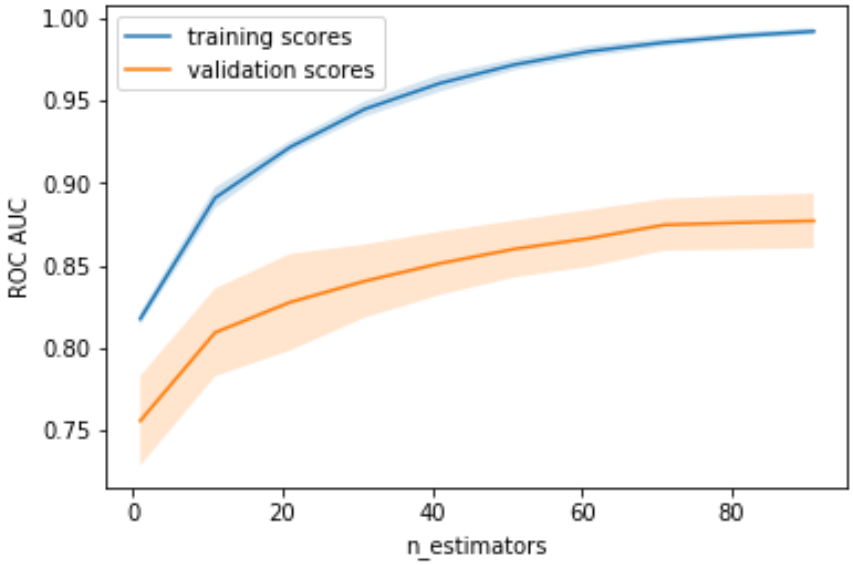

Рисунок 3.17 - Валидационные кривые модели GradientBoostingClassifier (параметр n_estimators)

Если вспомнить определение валидационной кривой, то есть график, показывающий результат на тренировочной и валидационной выборке в зависимости от сложности модели, и посмотреть на полученные кривые, то можно сказать, то практически все валидационные кривые далеки друг от друга.

Для сильно усложненных моделей тренировочная и валидационная ошибки значительно отличаются. Это можно объяснить переобучением: когда параметров слишком много либо не хватает регуляризации, алгоритм может "отвлекаться" на шум в данных и упускать основной тренд.

Классификатор KNeighborsClassifier показывает обратное - тренировочная и валидационная ошибка находятся рядом, и они велики. Это говорит о том, что модель недообучилась, то есть она не имеет достаточное кол-во параметров.

Код программы построения валидационных кривых указан в приложении Б.

Далее необходимо подобрать параметры запуска модуля VotingClassifier. Важным параметром является – voting, имеющий два значения 'hard' и 'soft'.

При параметре hard итоговый ответ объединенного классификатора будет соответствовать «мнению» большинства входящих в него членов. Например, три классификатора из четырех отправят элемент в первый класс, а, соответственно, четвертый классификатор посчитает, что элемент принадлежит второму классу. В такой ситуации модуль выберет первый класс, так как за него проголосовало большинство. [17]

Если установить значение soft, то будет проводится полноценное «голосование» и взвешивание предсказаний. Это значит, что итоговый ответ объединенного классификатор — argmax суммы предсказанных вероятностей.

Для полноценной оценки качества работы ансамбля алгоритмов машинного обучения проведем два тестирования, одно – с использование параметра hard, другое – с параметром soft.

Результат точности работы ансамбля с данными параметрами отобразим в таблице 3.3

Таблица 3.3 - Результат точности работы ансамбля на тестовой выборке

Параметр voting	Значение точности классификации на тестовой выборки
Hard	0.8962536023054755
Soft	0.8933717579250721

Значения точности классификации ансамбля на тестовой выборке в зависимости от выбранного параметра, показывают малый разрыв, однако ансамбль с параметром Hard является более точным.

Так же отобразим график зависимости показателя AUC от размера тренировочной выборки для ансамбля VotingClassifier на рисунке 3.18

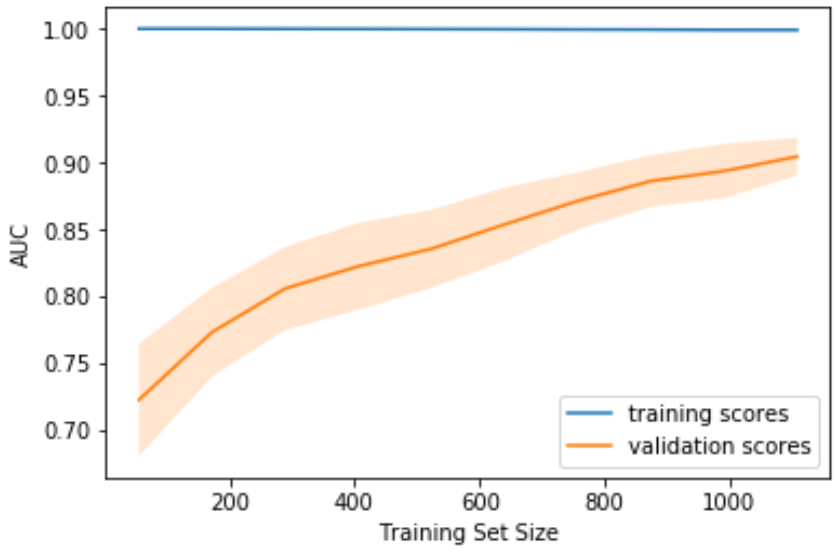

Рисунок 3.18 – график зависимости показателя AUC от размера тренировочной выборки для классификатора VotingClassifier

Кривая обучения VotingClassifier так же указывает на нехватку тренировочных данных, однако, кривая является более сглаженной, по сравнению с кривыми обучения других моделей.

Можно заметить, что не обязательно значение точности ансамбля должно превышать точность лучшего входящего в него классификатора. Стоит обратить внимание на другие метрики, которые указывают на

стабильность модели, то есть снижают риск переобучения, подгонку под обучающую выборку, и других проблем классификации.

Сформируем ансамбль и протестируем на входных данных.

Код программы формирования ансамбля указан в Приложении А.

Далее приведем таблицу для сравнения значений метрик отобранных моделей и ансамбля (таблица 3.4).

Таблица 3.4 – Значения метрик моделей машинного обучения

Модель машинного обучения	Score	Accuracy	Precision	Recall	F1
KNeighborsClassifier	0.88472	0.81385	0.83497	0.80978	0.82202
DecisionTreeClassifier	0.89048	0.80158	0.81241	0.81247	0.81821
BaggingClassifier	0.88760	0.82107	0.84295	0.81797	0.83511
RandomForestClassifier	0.88184	0.81312	0.84032	0.81120	0.83027
ExtraTreesClassifier	0.86743	0.80805	0.83095	0.79343	0.82433
AdaBoostClassifier	0.83573	0.79076	0.80674	0.79896	0.80231
GradientBoostingClassifier	0.91066	0.83838	0.85549	0.84242	0.84536
VotingClassifier	0.89625	0.87188	0.87398	0.85252	0.86389

По полученным данным построим диаграмму (рисунок 3.19)

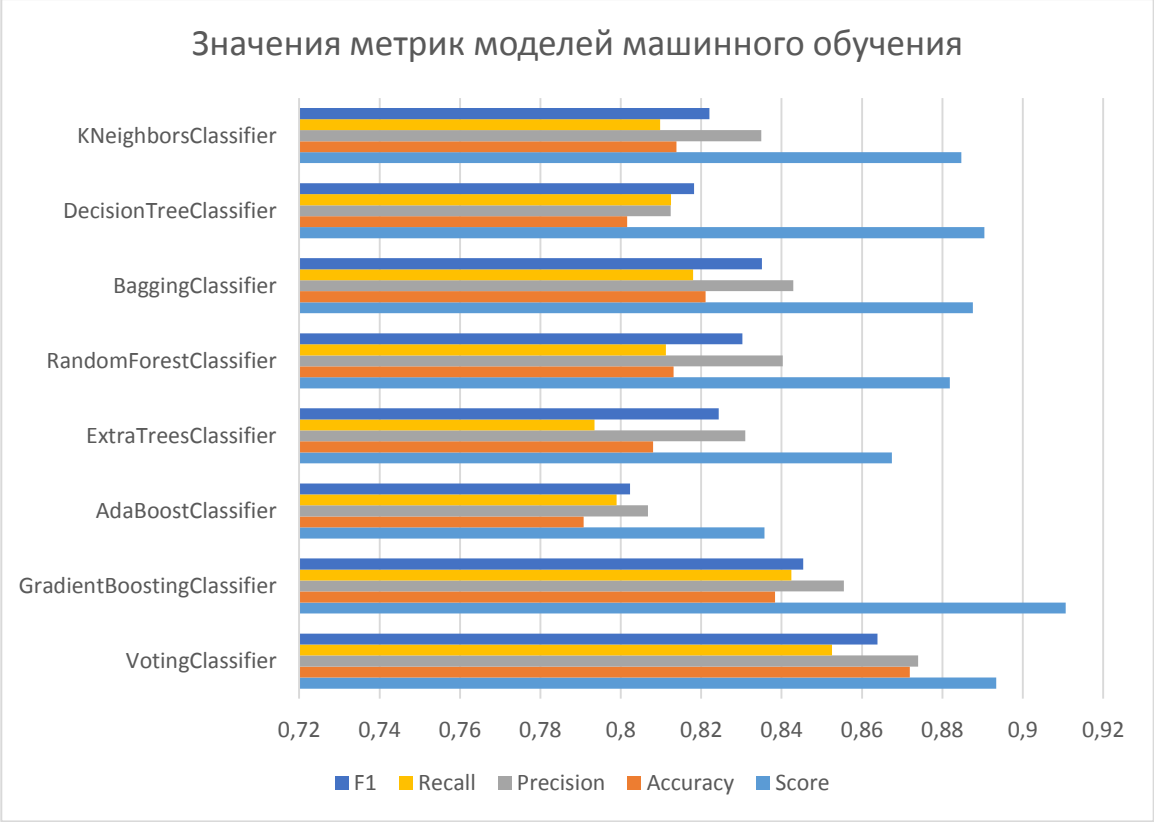

Рисунок 3.19 – Диаграмма значения метрик моделей машинного обучения

Проанализировав полученные данные, можно сказать, что, да, действительно значение точности на тестовых данных (0.89337 против 0.91066) ансамбля не превысило значения точности лучшего классификатора – GradientBoostingClassifier, однако, значения метрик у ансамбля более стабильны, что может означать, что на новых тестовых данных, ансамбль будет показывать более стабильный результат в отличие от других моделей машинного обучения, указанных в данной таблице. Данный вывод сделан на основании значений метрики Precision, которая характеризует долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, и метрики Recall, отображающую, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

Precision и recall, как говорилось ранее, в пункте 3.1, не зависят, в отличие от accuracy, от соотношения классов и потому применимы в условиях несбалансированных выборок. VotingClassifier имеет значения precision равное 0.87398 (87%) и recall - 0.85252 (85%), что говорит о более стабильной работе алгоритма в отличие от других классификаторов.

Так как F-мера напрямую зависит от значений метрик precision и recall, она так же может показать более сбалансированную характеристику, в данном случае VotingClassifier имеет значение 0.86389 (86%), что является наиболее высоким показателем среди исследуемых моделей.

После анализа графиков кривых обучений можно сказать, что данных для обучения недостаточно, так как мы явно наблюдаем процесс схождения кривых. Увеличение количества входных данных может благоприятно сказаться на качестве обучения моделей, так как полная сходимость кривых указывает на высокий результат.

По полученным валидационным кривым можно сказать, что практически все кривые далеки друг от друга.

Для сильно усложненных моделей тренировочная и валидационная ошибки значительно отличаются. Это можно объяснить переобучением: когда параметров слишком много либо не хватает регуляризации, алгоритм может "отвлекаться" на шум в данных и упускать основной тренд. Но есть и модель, у которой тренировочная и валидационная ошибка находятся рядом, и они велики. Это говорит о том, что модель недообучилась, то есть она не имеет достаточное кол-во параметров.

В заключение можно сказать, что полученный алгоритм с помощью ансамбля семи классификаторов имеет незначительное падение точности классификации на тестовой выборке в отличие от GradientBoostingClassifier, но высокие показатели метрик Precision, recall и F1, указывают на стабильную работу алгоритма «в бою», то есть высокое качество классификации на будущих новых входных данных, а кривые обучения и валидацинные кривые отобранных классификаторов показывают, что данных для обучения мало, а высокая сложность модели не всегда ведет к качественной классификации.

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы была описана актуальность рассматриваемой темы, определены объект и предмет выпускной квалификационной, поставлена цель и выявлены задачи. Так же, было рассмотрено два различных подхода к дифференциации патологи.

Для решения задачи был привлечен один из методов диагностики онкологии – спектроскопия комбинационного рассеяния. Полученные данные, то есть спектр КР, были преобразованы в необходимую форму для использования в алгоритмах машинного обучения.

Были поставлены и выполнены следующие задачи – подготовка входных данных, тестирование моделей машинного обучения на задаче классификации результатов спектроскопии комбинационного рассеяния, разработка ансамбля алгоритмов машинного обучения, тестирование ансамбля на результатах спектроскопии комбинационного рассеяния.

В качестве инструментов решения поставленной задачи были использованы объектно-ориентированный язык программирования Python 3.7.3, библиотека для машинного обучения Scikit-learn v0.21.2, среда разработки Announcing Anaconda Enterprise 5.3 и облачный сервис Google Colaboratory с интерактивной оболочкой Jupyter Notebook.

На начальном этапе было проведено обучение 16 моделей классификации, проанализирована точность работы алгоритмов, проведен отбор классификаторов с наиболее высокой оценкой разбиения данных по классам с последующим подборов лучших параметров классификации.

Заключительным этапом был анализ кривых валидации и обучения отобранных классификаторов, формирование ансамбля и его тестирование на входных данных. Результаты: точность классификации – 87%, полнота – 85%, гармоническое среднее между точностью и полнотой (мера F1) – 86%.

Анализ кривых валидации и обучения указал на малый размер тренировочных данных и, для некоторых случаев, высокую сложность модели, что привело к падению точности классификации.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Nilsson, R., Pena, J.M., Bjorkegren, J., Tegner, J.: Consistent feature selection for pattern recognition in polynomial time. The Journal of Machine Learning Research 8, 2017. - 687 с.
2. Сержантов К.А. Автофлуоресцентный анализ кожных патологий, с использованием нейросетевого алгоритма: науч. работа /К.А. Сержантов, М.Г. Лисовская, В.П. Захаров, А.А. Морятов, С.В. Козлов / сборник статей «Информационные технологии в моделировании и управлении: подходы, методы, решения», 2017. – 256-263 с.
3. Chollet, F. Deep Learning with Python / Francois Chollet: – Manning Publications - December 22, 2016.
4. Guyon, I., Elisseeff, A.: An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3, 1157–1182, 2017.
5. X. Pan, Y. Luo, Y. Xu, “K-nearest neighbour based structural twin support vector machine,” KnowledgeBased Systems, vol. 88, pp. 34-44, 2015.
6. Газета «Комсомольская правда» Айна Утибаева [Электронный ресурс] – «Тревога: каждый 40-й житель Самарской области - на учете в онкологическом диспансере» от 09.10.16. URL: https://www.samara.kp.ru/daily/26592.7/3606520/ (дата обращения: 08.03.2019)
7. V. Utkin, Y. A. Zhuk, “An one-class classification support vector machine model by interval-valued training data,” Knowledge-Based Systems, vol. 120, pp. 43-56, 2015.
8. M. Baig, M .M. Awais, E. M. El-Alfy, “AdaBoost-based artificial neural network learning,” Neurocomputing, vol. 16, pp. 22 – 41, 2017.
9. Падило Л.П. ОНКОЛОГИЧЕСКИЕ ЗАБОЛЕВАНИЯ: ПРИЧИНЫ, ВИДЫ, ПРОФИЛАКТИКА, ЛЕЧЕНИЕ [Электронный ресурс]// Молодежный научный форум: Естественные и медицинские науки: электр. сб. ст. по мат. XXIV междунар. студ. науч.-практ. конф. №

5(23). URL: https://nauchforum.ru/archive/MNF_nature/5(23).pdf (дата обращения: 10.04.2019)

10. L. D. Miller and L. K. Soh, "Cluster-Based Boosting," IEEE Transactions on Knowledge and Data Engineering, vol. 27, pp. 1491-1504, 2015.
11. A. C. Bahnsen, D. Aouada, B. Ottersten, “Exampledependent cost-sensitive decision trees,” Expert Systems with Applications, vol. 42, pp. 6609-6619, 2015.
12. Y. Lertworaprachaya, Y. Yang, R. John, “Interval-valued fuzzy decision trees with optimal neighbourhood perimeter,” Applied Soft Computing, vol. 24, pp. 851- 866, 2014.
13. Holzinger, D. Blanchard, M. Bloice, K. Holzinger, V. Palade, R. Rabadan, “Darwin, Lamarck, or Baldwin: Applying Evolutionary Algorithms to Machine Learning Techniques”, International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, 2014.
14. Ищеряков, С. Н. Развитие паллиативной помощи больным злокачественными новообразованиями в системе здравоохранения Самарской области / С. Н. Ищеряков, Э. М. Гимранова, А. А. Саланов // Управление качеством медицинской помощи. - 2012. - № 2. - С. 18-24.
15. Каприн А. Д., Старинский В. В., Петрова Г. В., ред. Злокачественные новообразования в России в 2012 году (заболеваемость и смертность). М.: ФГБУ «МНИОИ им П. А. Герцена» Минздрава России; 2015. 249 с
16. Steinwart Ingo, Christmann Andreas. Support Vector Machines. — 2st edition. — Springer Publishing Company, Incorporated, 2018.
17. Scikit-learn: Machine Learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort et al. // Journal of Machine Learning Research. — 2017. — Vol. 12. — P. 2825–2830.
18. Zalaudek I., Argenziano G., Soyer H.P. et al. Three-point checklist of dermoscopy: an open internet study // Brit. J. Dermatol. 2015. Vol. 154. P. 431.

19. Корси Л.В., Соколов В.Г. Лазерный способ фотохимической деструкции опухолей без экзогенных сенсибилизаторов // В сб. «Лазерно-оптические системы и технологии» ФГУП «НПО Астрофизика». М., 2015. С.101-106.
20. Рапаков Г. Г., Горбунов В. А. Интеллектуальный анализ медико-социологических данных с использованием метода Decision Trees // Вестник Череповецкого государственного университета. 2019, №1
21. Денисова О.О. и Мухутдинов Э.А. «Большие данные - это не только размер данных» // Вестник технологического университета, т. 18, № 4, p. 5, 2015.
22. Зарубина Т.В. Актуальные вопросы внедрения информационных технологий в здравоохранении. Вестник Росздравнадзора. 2018; (3): 20-6
23. Рыков В.П. Модульный принцип обучения искусственных нейронных сетей на примере известных нейросетевых топологий // Вестник Тамбовского университета. Серия: Естественные и технические науки. - 2015. - Т. 19. № 2. С. 583-586.

ПРИЛОЖЕНИЕ А

Полный листинг кода программы.

```
import numpy as np


def loadData():
    import pandas as pd
    data_path = "ramanfull.xlsx"
    return pd.read_excel(data_path)


def createDataSet(dataFrame):
    dataset =np.array( dataFrame.iloc[:, 0:1044])
    target = dataFrame.loc[:,['label']]
    target = np.array(target.replace(to_replace=['O', 'K'], 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 target,dataset


target, dataset = createDataSet(loadData())


from sklearn import preprocessing
dataset = preprocessing.normalize(dataset)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(dataset, target, test_size=0.2)
y_train = y_train.reshape(-1)
y_test = y_test.reshape(-1)

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.svm import NuSVC
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors.nearest_centroid import NearestCentroid
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier

classifiers = [
    LogisticRegression(),
    SVC(),
    NuSVC(),
    LinearSVC(),
    GaussianNB(),
```

```
    BernoulliNB(),
    KNeighborsClassifier(),
    NearestCentroid(),
    GaussianProcessClassifier(),
    DecisionTreeClassifier(),
    BaggingClassifier(),
    RandomForestClassifier(),
    ExtraTreesClassifier(),
    AdaBoostClassifier(),
    GradientBoostingClassifier(),
    MLPClassifier(),
]

names = [
    'LogisticRegression',
    'SVC',
    'NuSVC',
    'LinearSVC',
    'GaussianNB',
    'BernoulliNB',
    'KNeighborsClassifier',
    'NearestCentroid',
    'GaussianProcessClassifier',
    'DecisionTreeClassifier',
    'BaggingClassifier',
    'RandomForestClassifier',
    'ExtraTreesClassifier',
    'AdaBoostClassifier',
```

```
    'GradientBoostingClassifier',
    'MLPClassifier',
]

scores = []
for classifier in classifiers:
    model = classifier.fit(x_train, y_train)
    scores.append(model.score(x_test, y_test))

for i, score in enumerate(scores):
    print(names[i], 'score =', score)

param_grids = [
    {
        'weights' : ('uniform', 'distance'),
        'n_neighbors' : (1, 3, 5, 7, 9),
    }, #KNeighborsClassifier

    {
        'criterion' : ('gini', 'entropy'),
        'max_depth' : (10, 30, 50, 70, 90, None),
        'min_samples_split' : (2, 3, 4, 5, 6),
        'min_samples_leaf' : (1, 2, 3, 4, 5),
    }, #DecisionTreeClassifier

    {
        'n_estimators' : (5, 10, 15, 20, 25),
        'max_samples' : (0.2, 0.4, 0.6, 0.8, 1.0),
```

```
        'max_features' : (0.2, 0.4, 0.6, 0.8, 1.0),
    }, #BaggingClassifier

    {
        'n_estimators' : (5, 10, 15, 20, 25),
        'criterion' : ('gini', 'entropy'),
        'min_samples_split' : (2, 3, 4, 5, 6),
        'min_samples_leaf' : (1, 2, 3, 4, 5),
    }, #RandomForestClassifier

    {
        'criterion' : ('gini', 'entropy'),
        'max_depth' : (10, 30, 50, 70, 90, None),
        'min_samples_split' : (2, 3, 4, 5, 6),
        'min_samples_leaf' : (1, 2, 3, 4, 5),
    }, #ExtraTreesClassifier

    {
        'n_estimators' : (10, 30, 50, 70, 90),
        'learning_rate' : (0.3, 0.6, 1, 1.3, 1.6),
    }, #AdaBoostClassifier

    {
        'loss' : ('deviance', 'exponential'),
        'learning_rate' : (0.05, 0.1, 0.2, 0.25, 0.3),
        'n_estimators' : (30, 60, 100, 130, 160),
    }, #GradientBoostClassifier
]
```

```
from sklearn.model_selection import GridSearchCV

classifiers = [
    KNeighborsClassifier(),
    DecisionTreeClassifier(),
    BaggingClassifier(),
    RandomForestClassifier(),
    ExtraTreesClassifier(),
    AdaBoostClassifier(),
    GradientBoostingClassifier(),
]
func_names = [
    'KNeighborsClassifier',
    'DecisionTreeClassifier',
    'BaggingClassifier',
    'RandomForestClassifier',
    'ExtraTreesClassifier',
    'AdaBoostClassifier',
    'GradientBoostingClassifier',
    'VotingClassifier',
]
clf = []

for i in range(len(classifiers)):
  clf.append(GridSearchCV(classifiers[i], param_grids[i], cv=5))
  clf[-1].fit(x_train, y_train)
```

```
def call_func(func_name, args):
    tmp = []
    for a, b in args.items():
      if type(b) == str: tmp.append(a+'='+"'"+b+"', ")
      else: tmp.append(a+'='+str(b)+', ')
    args_string = ''.join(tmp)
    call_string = '%s(%s)' % (func_name, args_string)
    return eval(call_string)

from sklearn.ensemble import VotingClassifier

clf_best = []
for i,x in enumerate(clf):
  d = x.best_params_
  clf_best.append(call_func(func_names[i], d))
lst = []
for i in range(len(clf)):
  lst.append((func_names[i], classifiers[i]))
clf_best.append(VotingClassifier(estimators=lst, voting='hard'))

from sklearn.model_selection import cross_val_score

for i in range(len(clf_best)):
  clf_best[i].fit(x_train, y_train)
  print(i)

scores = []
for i in range(len(clf_best)):
```

```
  scores.append(clf_best[i].score(x_test, y_test))


acc = []
for i in range(len(clf_best)):
  acc.append(cross_val_score(clf_best[i], x_train, y_train, cv=5,
scoring='accuracy').mean())


pre = []
for i in range(len(clf_best)):
  pre.append(cross_val_score(clf_best[i], x_train, y_train, cv=5,
scoring='precision').mean())


rec = []
for i in range(len(clf_best)):
  rec.append(cross_val_score(clf_best[i], x_train, y_train, cv=5,
scoring='recall').mean())


fm = []
for i in range(len(clf_best)):
  fm.append(cross_val_score(clf_best[i], x_train, y_train, cv=5,
scoring='f1').mean())


for i in range(8):
  print(func_names[i], 'score =', scores[i])
  print(func_names[i], 'accuracy =', acc[i])
  print(func_names[i], 'precision =', pre[i])
  print(func_names[i], 'recall =', rec[i])
  print(func_names[i], 'f1 =', fm[i])
  print()
```

```
np.savetxt('x_train.txt', x_train.flatten(), header=str(x_train.shape))
np.savetxt('y_train.txt', y_train.flatten(), header=str(y_train.shape))
np.savetxt('x_test.txt', x_test.flatten(), header=str(x_test.shape))
np.savetxt('y_test.txt', y_test.flatten(), header=str(y_test.shape))

file = open('params.txt', 'w')
for i, x in enumerate(clf):
  d = x.best_params_
  file.write(str(d) + ' #' + func_names[i] + '\n')
file.close()
```

ПРИЛОЖЕНИЕ Б

Код программы построения кривых обучения и валидационных кривых.

```
import numpy as np
import re

with open('x_train.txt') as f:
    shape = tuple(int(num) for num in re.findall(r'\d*', f.readline()) if len(num))
x_train = np.loadtxt('x_train.txt').reshape(shape)
with open('y_train.txt') as f:
    shape = tuple(int(num) for num in re.findall(r'\d*', f.readline()) if len(num))
y_train = np.loadtxt('y_train.txt').reshape(shape)
with open('x_test.txt') as f:
    shape = tuple(int(num) for num in re.findall(r'\d*', f.readline()) if len(num))
x_test = np.loadtxt('x_test.txt').reshape(shape)
with open('y_test.txt') as f:
    shape = tuple(int(num) for num in re.findall(r'\d*', f.readline()) if len(num))
y_test = np.loadtxt('y_test.txt').reshape(shape)

y_train = y_train.astype('int32')
y_test = y_test.astype('int32')

from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier

params = []
```

```
with open('params.txt') as f:
    for line in f:
        params.append(eval(line))

def call_func(func_name, args):
    tmp = []
    for a, b in args.items():
      if type(b) == str: tmp.append(a+'='+"'"+b+"', ")
      else: tmp.append(a+'='+str(b)+', ')
    args_string = ''.join(tmp)
    call_string = '%s(%s)' % (func_name, args_string)
    print(call_string)
    return eval(call_string)

func_names = [
    'KNeighborsClassifier',
    'DecisionTreeClassifier',
    'BaggingClassifier',
    'RandomForestClassifier',
    'ExtraTreesClassifier',
    'AdaBoostClassifier',
    'GradientBoostingClassifier',
]

clf_best = []
for i in range(len(func_names)):
  d = params[i]
  if i == 0 or i == 2 or i == 3:
    d['n_jobs'] = -1
  clf_best.append(call_func(func_names[i], d))
```

```
for i,x in enumerate(clf_best):
    x.fit(x_train, y_train)
    print(func_names[i], 'score =', x.score(x_test, y_test))

from __future__ import division, print_function
import warnings
warnings.filterwarnings('ignore')
get_ipython().run_line_magic('matplotlib', 'inline')
from matplotlib import pyplot as plt
import seaborn as sns

import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV,
SGDClassifier
from sklearn.model_selection import validation_curve
from sklearn.model_selection import learning_curve
from sklearn.svm import SVC

def plot_with_err(x, data, **kwargs):
    mu, std = data.mean(1), data.std(1)
    lines = plt.plot(x, mu, '-', **kwargs)
    plt.fill_between(x, mu - std, mu + std, edgecolor='none',
facecolor=lines[0].get_color(), alpha=0.2)

def roc_auc(clf):
```

```
    train_sizes = np.linspace(0.05, 1, 10)
    N_train, val_train, val_test = learning_curve(clf,
                    x_train, y_train, train_sizes=train_sizes, cv=5, scoring='roc_auc')
    plot_with_err(N_train, val_train, label='training scores')
    plot_with_err(N_train, val_test, label='validation scores')
    plt.xlabel('Training Set Size'); plt.ylabel('AUC')
    plt.legend()
    plt.show()

for i,x in enumerate(clf_best):
  print(func_names[i])
  roc_auc(x)

from sklearn.ensemble import VotingClassifier

lst = []
for i,x in enumerate(clf_best):
    lst.append((func_names[i], x))
vclf_hard = VotingClassifier(estimators=lst, voting='hard')

vclf_hard.fit(x_train, y_train)
print(vclf_hard.score(x_test, y_test))

scores = []
for x in clf_best:
  scores.append(x.score(x_test, y_test))

vclf_soft = VotingClassifier(estimators=lst, voting='soft', weights=scores)

vclf_soft.fit(x_train, y_train)
```

```
print(vclf_soft.score(x_test, y_test))

print('VotingClassifier - soft voting')
roc_auc(vclf_soft)

from sklearn.model_selection import validation_curve

def val_curve(clf, param_name, params):
    val_train, val_test = validation_curve(clf, x_train, y_train, param_name,
                                           params, cv = 5, scoring='roc_auc')
    plot_with_err(params, val_train, label='training scores')
    plot_with_err(params, val_test, label='validation scores')
    plt.xlabel(param_name); plt.ylabel('ROC AUC')
    plt.legend();

# KNeighborsClassifier n_neighbors validation curve
val_curve(KNeighborsClassifier(), 'n_neighbors', range(1, 101, 10))

# DecisionTreeClassifier max_depth validation curve
val_curve(DecisionTreeClassifier(), 'max_depth', range(1, 11, 1))

# BaggingClassifier n_estimators validation curve
val_curve(BaggingClassifier(), 'n_estimators', range(1, 51, 5))

# RandomForestClassifier max_depth validation curve
val_curve(RandomForestClassifier(), 'max_depth', range(1, 11, 1))

# ExtraTreesClassifier max_depth validation curve
val_curve(ExtraTreesClassifier(), 'max_depth', range(1, 11, 1))
```

```
# AdaBoostClassifier learning_rate validation curve
val_curve(AdaBoostClassifier(), 'learning_rate', np.linspace(0.05, 1, 10))
val_curve(AdaBoostClassifier(), 'n_estimators', np.arange(1, 100, 10))

val_curve(GradientBoostingClassifier(), 'learning_rate', np.linspace(0.05, 1, 10))
val_curve(GradientBoostingClassifier(), 'n_estimators', np.arange(1, 100, 10))
```