Федеральное государственное автономное образовательное учреждение высшего образования

«Российский университет дружбы народов»

АННОТАЦИЯ

выпускной квалификационной работы

Мохаммед Маусара Абдулсаттар Мохаммед

(фамилия, имя, отчество)

на тему:

Использование машинного обучения для анализа выдыхаемого воздуха с помощью масс-спектометрии у пациентов с сердечно-сосудистыми и онкологическими заболеваниями

Объектом исследования выпускной квалификационной работы являются алгоритмы, позволяющие классифицировать пациентов с сердечно-сосудистыми и онкологическими заболеваниями по данным масс-спектрометра. В процессе написания выпускной квалификационной работы мною рассматриваются преимущества и недостатки подходящих для данной задачи методов. Для проведения экспериментов мною был разработан программный комплекс на языке Python с использованием различных фреймворков. Были реализованы различные алгоритмы градиентного бустинга деревьев решений и разновидностей реккурентных нейронных сетей. Было произведено сравнение результатов их работы, таких как качество классификации, время обучения и другие параметры.

Содержание

Введение

В связи с набирающими популярность методами анализа данных в медицине, становятся целесообразным попытки применения методов анализа больших данных в дополнение к стандартным методам диагностики и выявления заболеваний. Это привело к появлению большого количества задач, таких как анализ рентгенограмм с помощью компьютерного зрения, выявление аномалий на временных рядах в электрокардиографии.

Хроническая сердечная недостаточность (ХСН) является одной из самых массовых сердечных заболеваний и, согласно статистике РОССТАТа, за последние 20 лет количество больных пациентов увеличилось на 3%. Согласно статистике, в России годовая смертность от ХСН составляет около 850,000 человек.

Классическими инструментальными методами диагностики ХСН являются эхокардиография (Эхо-КГ) и электрокардиография (ЭКГ). Обязательным условием для выполнения Эхо-КГ является высокий профессионализм и компетентность специалиста. Одним из основных современных методов диагностики ХСН является анализ выдыхаемого воздуха методом протонной масс-спектрометрии (PTR-MS). Суть данной технологии заключается в диагностике за счёт обнаружения летучих органических соединений (ЛОС) в реальном времени на следовых уровнях в газообразных средах. Этот метод был впервые применен учеными из Института ионной физики Университета Леопольда-Франценса в Инсбруке в середине 1990-х годов.

—

Основными плюсами данного метода можно назвать неинвазив-

ность процедуры и большая точность диагностики. Однако, существенными минусами является высокая стоимость исследования и необходимость привлечения высокопрофессионального специалиста.

Актуальность Учитывая, что диагностика с помощью данного метода является довольно трудоёмким процессом, применение современных средств автоматизации, таких как алгоритмы машинного обучения и нейронные сети, является актуальной задачей.

Цель работы Целью данной работы является разработка алгоритмов, позволяющих классифицировать пациентов с хронической сердечной недостаточностью и онкологическими заболеваниями на базе имеющихся данных.

Задачи работы Задачами работы являются:

1. Изучение бустинговых алгоритмов.
2. Изучение реккурентных нейронных сетей.
3. Изучение методов и инструментов для анализа временных рядов.
4. Разработка программного комплекса на языке Python с использованием различных фреймворков.
5. Проведение сравнительного анализа различных методов для решения задачи классификации пациентов на основе данных масс-спектрометра. Оценка качества решаемой задачи.

Методы исследования Для реализации алгоритмов градиентного бустинга деревьев решений используется библиотеки CatBoost, XGBoost, LightGBM. (TODO: может быть стоит сделать главу, где эти библиотеки будут сравниваться). Реккурентные нейронные сети были написаны с использованием библиотек PyTorch и Keras. Тренировочными и тестовыми данными являются база записей дыхания пациентов, собранная

медицинскими работниками на аппаратах масс-спектрометрии и обработанная в библиотеке Ionicon 1000. (TODO: write about it). Был проведён исследовательский анализ этих данных (EDA, exploratory data analysis). EDA включает в себя извлечение наиболее полезной и актуальной информации и исследование распределений данных, поиск зависимой и анализ корреляций. Данное исследование проводилось с помощью библиотек pandas, numpy и seaborn.

Структура работы Сначала мы представим анализ данных. После будет дан краткий эскурс в теорию используемых в работе алгоритмов машинного обучения и нейронных сетей, обученных на основе имеющихся данных. Будет представлено описание процесса обучение моделей, а также показан анализ их результатов. В заключении работы будет дан вывод, сделанный после проведённых экспериментов.

1 Данные

1.1 Литературный анализ

За последние десятилетия метод получил большую популярность в задачах поиска ЛОС в выдыхаемом воздухе, которые можно использовать в качестве возможных диагностических и неинвазивных маркеров болезни. Так в 2016 г. Копылов Ф.Ю., Сыркин Ф.Ю. и др. показали, что анализ выдыхаемого воздуха, является одним из перспективных направлений в поиске новых биомаркеров хронической сердечной недостаточности (ХСН)[5]. Особое внимание уделялось концентрации ацетона, дальнейший анализ показал, что ацетон может быть использован в качестве маркера ХСН[6]. Ткаченко В.И в 2015 году изучил уровень СО в выдыхаемом воздухе у пациентов с сахарным диабетом 2-го типа и было установлено, что СО в составе выдыхаемого воздуха является неинвазивным методом и может использоваться для анализа развития ангиопатий и сердечнососудистых осложнений в общеврачебной практике[7]. Также в 2015 г. Bodelier AG et al. в 2015 г. проводили поиск ЛОС в выдыхаемом воздухе как возможных диагностических и неинвазивных маркеров у лиц с болезнью Крона (БК) и вместе с тем пытались выяснить, есть ли разница между этими показателями во время активной стадии БК и ее ремиссией. Найденные ими ЛОС были объединены в один показатель активности заболевания, который смог помочь дифференцировать воспалительный процесс у 60% ранее не обследуемых больных [8]. Zuo W et al. в 2019 г., проводя исследование выдыхаемых газов, собранных мешком для отбора проб, и детектирования с помощью электро-распылительной ионизационной масс-спектрометрии, пытались выяснить, могут ли неко-

торые ЛОС служить специфическими диагностическими маркерами рака легких. Результаты их работы показали, что бутадиен, оротовая кислота, тетрагидробиоптерин и N-фенилацетилглутамин у пациентов с раком легких значительно отличаются от таковых в контрольной и легочной инфекционных группах [9].

В диссертационной работе Малиновской Л.М.[1] на основе анализа данных масс-спектрометрии был выявлен класс веществ, названных наиболее перспективными веществами для классификации групп онкобольных пациентов и пациентов с сердечной недостаточностью. К классу этих веществ были отнесены вещества, занесённые в таблицу ниже.

Название вещества	Химическая формула	Молярная масса [г/моль]
Ацетон	C_3H_{6O}	58.080
Уксусная кислота	CH_3COOH	60.052
Пропилен	C_3H_6	42.081
Формальдегид	CH_2O	30.026
Этанол	C_2H_6O	46.069
Ксилол	C_8H_{10}	106.16

Таблица 1. Вещества, наиболее перспективные для классификации

В соответствии с этим, была произведена аугментация всех признаков, соотвествующих данным веществам, а именно: были добавлены колонки максимального значения, среднего значения, минимального значения, кумулятивный суммы и медианного значения. Эксперименты показали, точность моделей, обученных на данных с аугментацией перечисленных признаков существенно превосходит точности моделей, обучен-

ных на исходных данных, что позволяет делать вывод о подтверждении результатов работы [1].

Согласно выводу той же работы, анализ выдыхаемого воздуха методом протонной масс-спектрометрии может использоваться в качестве неинвазивного диагностического метода и может быть рекомендован для диагностики хронической сердечной недостаточности. Данный вывод согласуется с результатом экспериментов, представленных в этой работе, показывающих о наличии корреляции между распределением концентраций веществ при анализах методом масс-спектрометрии и заболеванием пациента.

1.2 Анализ данных

С помощью аппарата PTR были собраны анализы дыхания пациентов четырех разных групп. Эти данные представляют собой некий файл, который содержит в себе записанные масс-спектры для конкретного пациента. Особая роль в таких исследованиях выделяется обработке сырых данных, полученных с эксперементального аппарата. Эти данные были обработаны стандартной библиотекой, рекомендованной производителем аппарата, Ionicon 1000. После обработки, на выходе получалась таблица, состоящая из 3000 колонок, где каждая колонка соответствует значению m/z (отношение массы к заряду), а строка соответствует секунде эксперимента. В дальнейшем, все эксперементы проводились на основании полученной таблицы данных. Первом делом было принято решение проанализировать показания капнографа.

1.3 Все что касается

Также в исходной таблице дополнительно присутствуют данные полученные с датчика капнографии, который показывают содержание CO_2 в выдохе пациента. рис. (1.3).

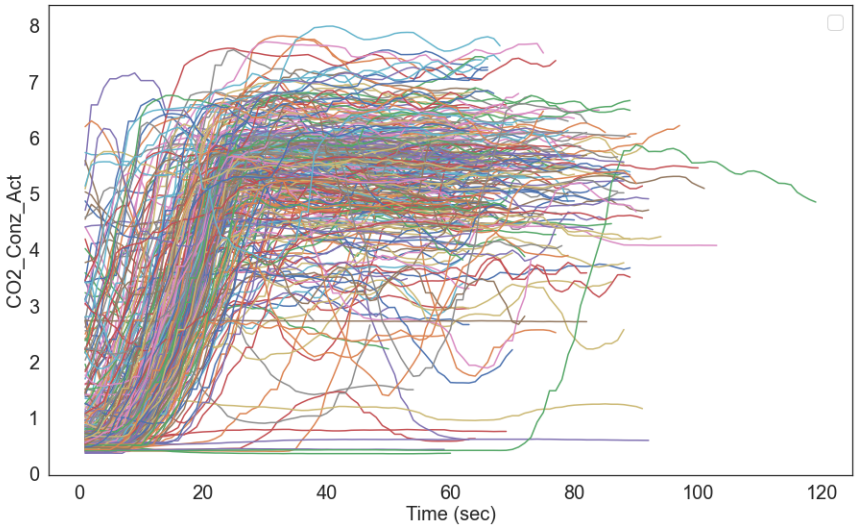

Рис. 1: Накапливание концентрации CO_2 пациентов по датчику капнографии

На графике заметен уровень плато, на который выходят показания прибора, когда в него дышит человек. По этому плато можно отличать, когда пациент действительно дышит аппаратом, а когда прибор работает в холостую и фиксирует внешний шум. Так же можно заметить, что данные далеки от идеальных: в них присутствуют холостые образцы, некоторые записи начинались спустя какое-то время после начала эксперимента, длина записисей не одинакова.

В ходе анализа данных стало понятно, что показания капностата, отличаются от показаний самого масс-спектрометра. На рис. (1.3) показана зависимость показаний капностата от того, к какому классу относиться больной.

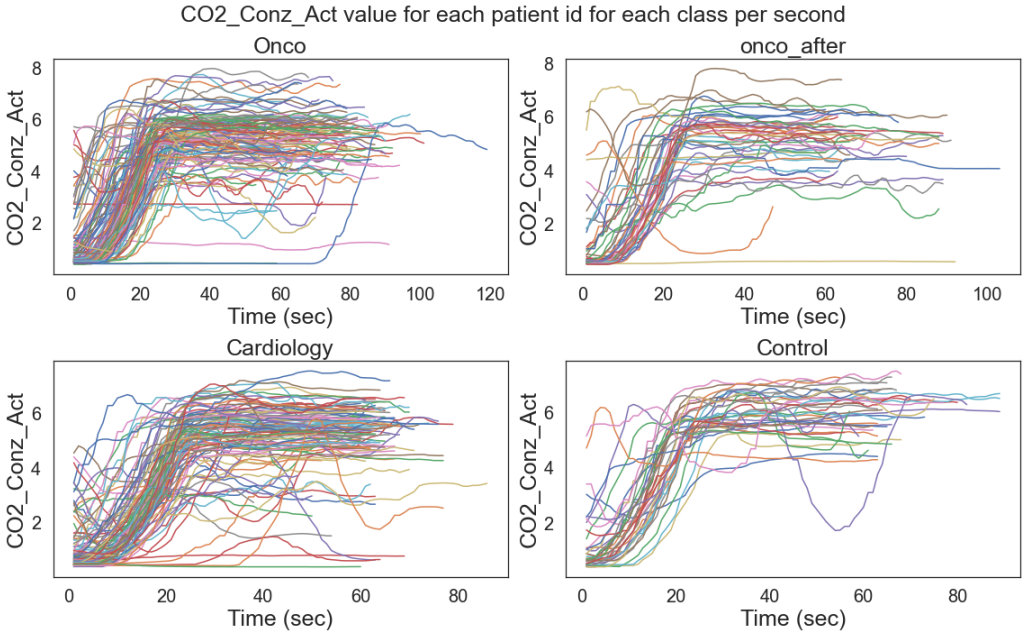

Рис. 2: Накапливание концентрации CO_2 пациентов по датчику капнографии, в зависимости от класса.

Сделать какие-то выводы. Стоить заметить, что подобный рост концентрации вещества, который наблюдается в показаниях капнографа, не наблюдается в других веществах. Для примера, посмотрим на значения вещества $(H2O)H+$:

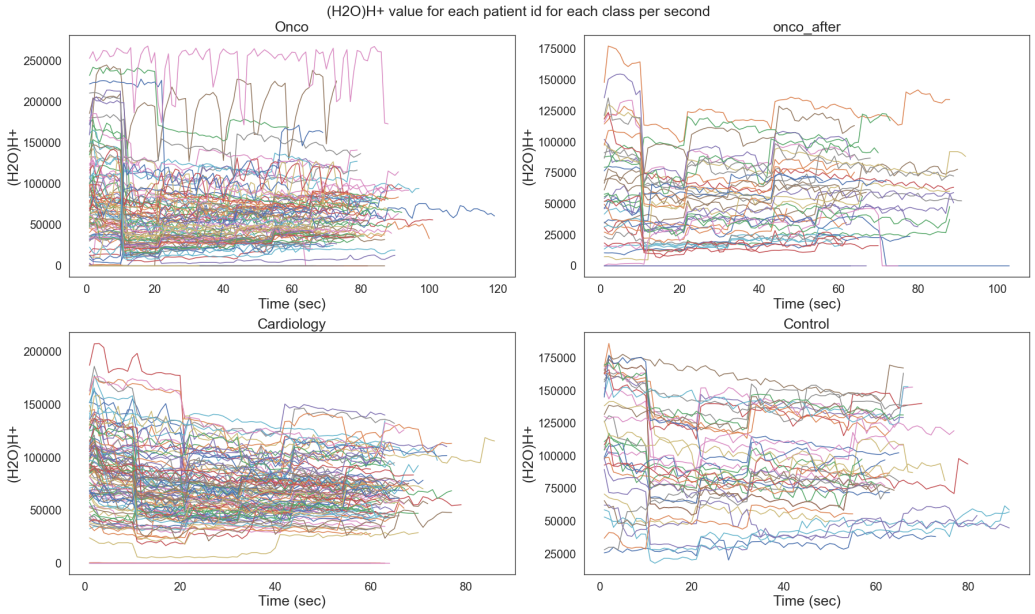

Рис. 3: Значения концентрации $H2OH$ пациентов в зависимости от класса.

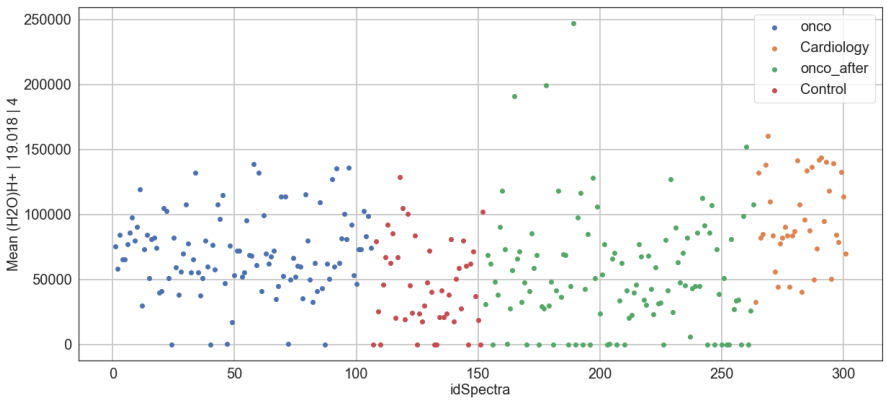

Рис. 4: Среднее значения концентрации иона $(H_2O)H+$ для каждого пациента

1.3.1 Все что касается (H2O)H+

Проводя более подробный анализ данных (EDA), можно посмотреть на среднее значения концентрации иона (H2O)H+ для каждого пациента в зависимости от группы больных, к которой он принадлежит. Таким образом дальнейшее преобразование данных потребовало добавления статистических колонок, колонок отражающих зависимость значений от секунды эксперимента.

В результате анализа имеющихся данных было выявлено, что представленные 4 класса пациентов не сбалансированы. Их пропорции также были проанализированы и учтены.

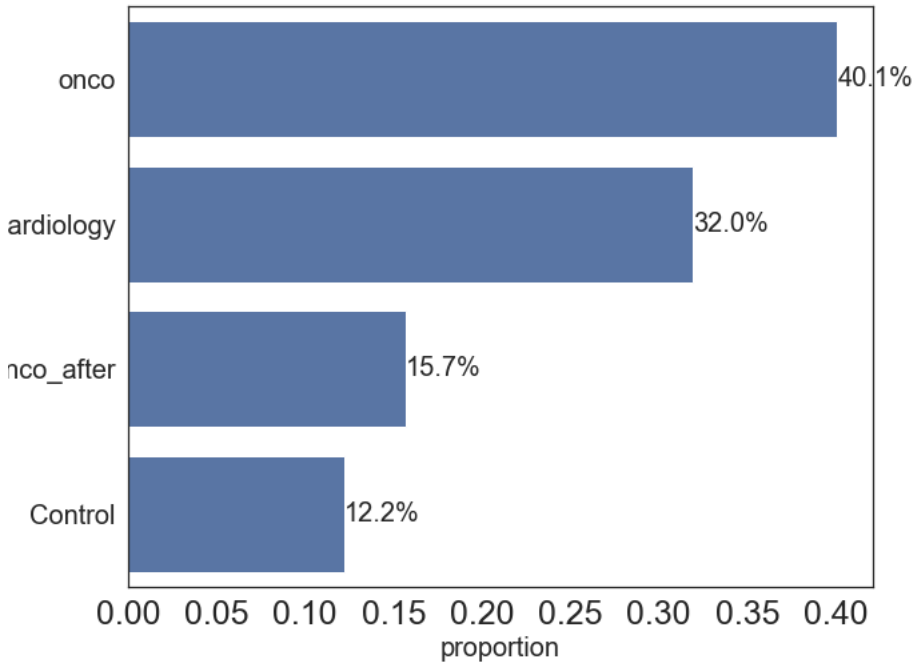

Рис. 5: Распределение пациентов по классам

Данные были кластеризованы на предмет поиска возможностей их

классификации. Стандартный алгоритм PCA не дал каких-либо существенных результатов. Более комплексные алгоритмы так же не показали никаких зависимостей.

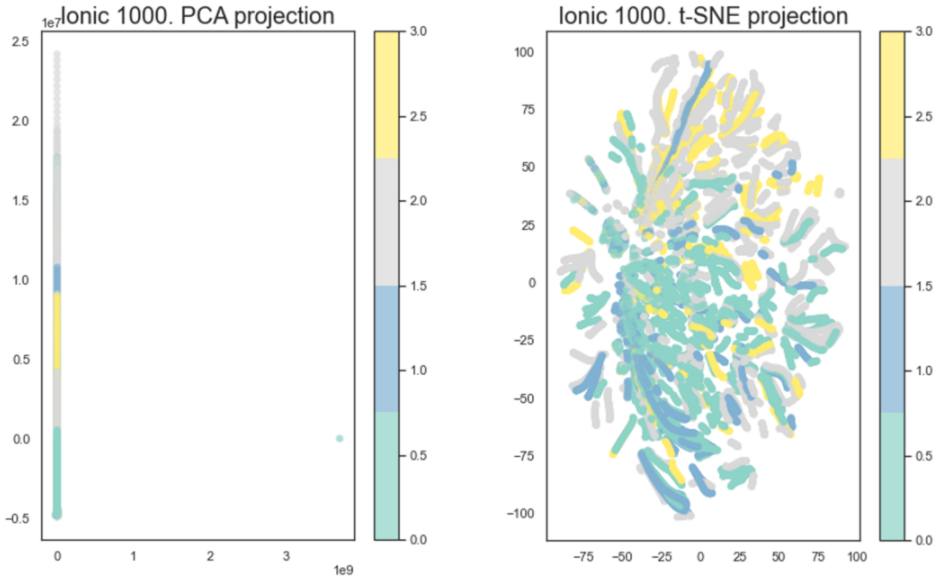

Рис. 6: Карта кластеризации пациентов

После анализа данных, было принято решение, что добавление статистических колонок, отражающих как максимальную концентрацию вещества за время дыхания конкретного пациента, так и множество других, позволяют решать данную задачу построчно. Не смотря на то, что представленные данные имеют явную временную зависимость, они отражают лишь накопление концентраций веществ в аппарате за время дыхания пациента. В добавок к этому, группы людей с онкологическими заболеваниями и недавно излечившиеся были объединены в одну группу ввиду их существенного сходства.

В полученной таблице данных всем пациентам был поставлен в соответствие уникальный идентификатор их состояния в столбце "target":

здоровые пациенты и пациенты, больные онкологическим заболеванием, либо же обладающие сердечной недостаточностью. Набор данных был проверен на наличие дубликатов и NaN, все данные, не являющиеся числовыми, были приведены к числовому виду.

Все вещества были представлены в трёх состояния - *Raw*, *Corr* и *Conc*. Представление данных *Conc* содержало в себе поправленные нормализованные данные, при анализе с поправкой на масштаб была выявлена их полная корреляция.

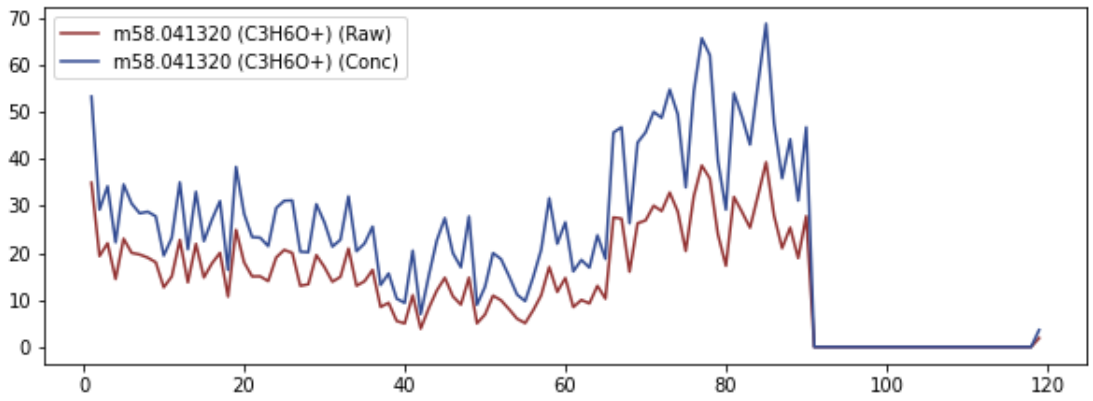

Рис. 7: Корреляция данных *Raw* и *Conc* на примере C_3H_6O

Соответственно, было принято решение отказаться от всех представлений данных, кроме *Conn*.

Максимальное время проведения эксперимента составило 119 секунд, минимкальное - 22 секунды, медианное - около 66 секунды, среднее - около 63 секунды.

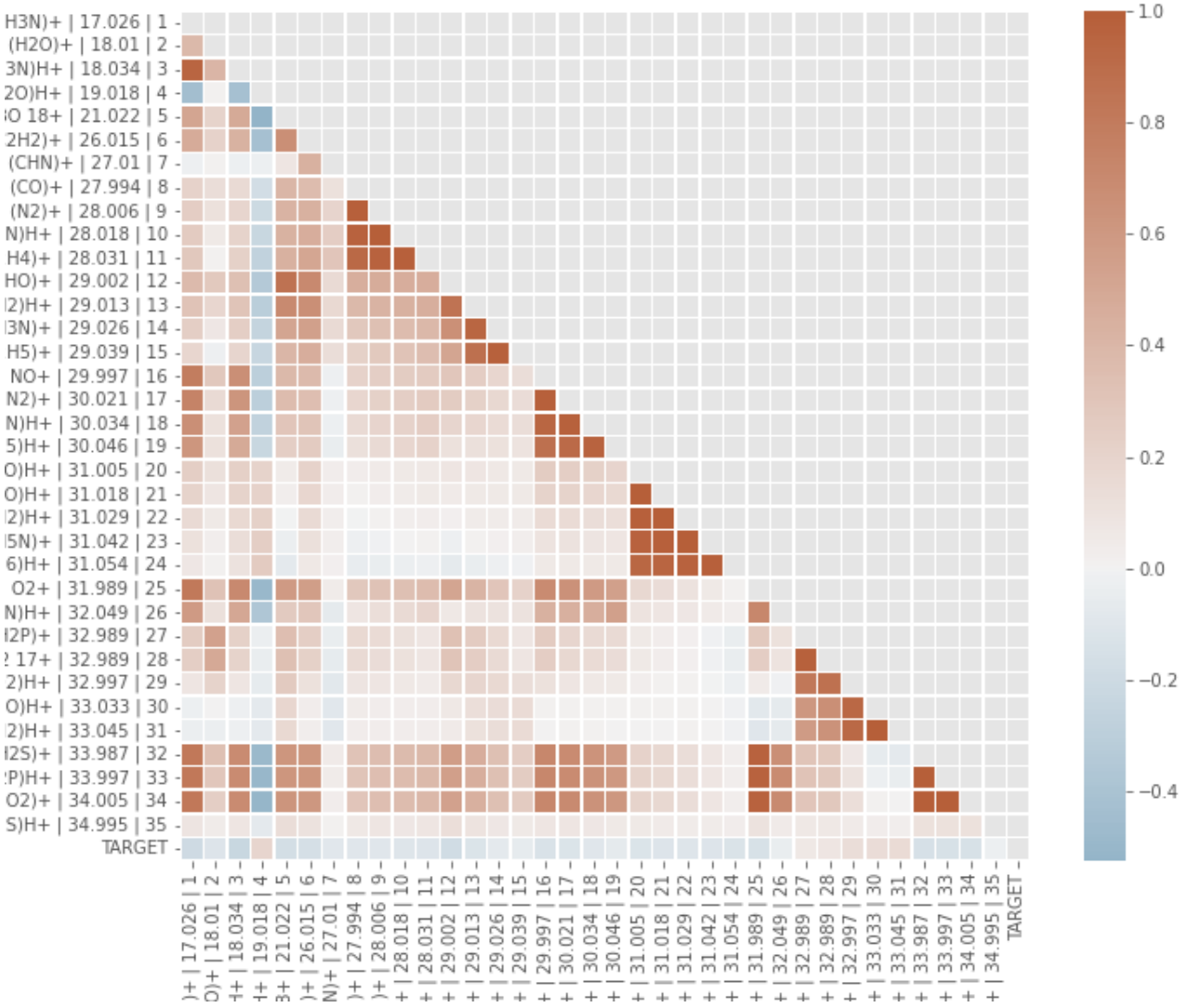

Рис. 8: Корреляционная матрица

Была построена и проанализирована корреляционная матрица набора данных (см. рис. 1). Был сделан вывод, что данные обладают низкой корреляцией, следовательно, классификация состояния пациентов на основе малого количества коррелируемых столбцов невозможна.

Не смотря на это, при анализе зависимостей различий концентраций вещества при разных состояниях пациентов, был сделан вывод о том, что такие зависимости имеют место быть, что показано на рисунке 2. Это дает основание высказывать гипотезу о том, что автоматическая классификация на основе данного набора данных возможна.

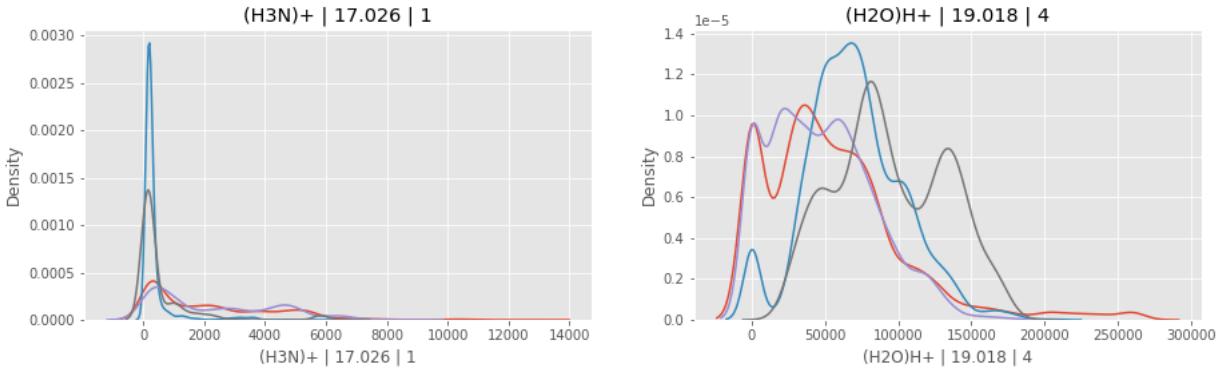

Рис. 9: Различие концентраций вещества при разных состояниях пациентов

Пациент 1	X_{11}	⋯	$X_{1,\ d*p}$	T_1
Пациент 2	X_{21}	⋯	$X_{2,\ d*p}$	T_2
⋯	⋯	⋯	⋯	⋯
Пациент n	X_{n1}	⋯	$X_{n,\ d*p}$	T_n

Рис. 10: Все d*p признаков i-го пациента хранятся в i-ой строке при анализе d секунд дыхния пациента

Пациент 1	t_1	$x_{1,\,1}$	⋯	$x_{1,\,1*p}$	T_1
⋯	⋯	⋯	⋯	⋯	---
Пациент 1	t_d	$x_{1,\,d*1}$	⋯	$x_{1,\,d*p}$	T_1
⋯	⋯	⋯	⋯	⋯	⋯
Пациент n	t_1	$x_{n,\,1}$	⋯	$x_{n,\,1*p}$	T_n
⋯	⋯	⋯	⋯	⋯	⋯
Пациент n	t_d	$x_{n,\,d*1}$	⋯	$x_{n,\,d*p}$	T_n

Рис. 11: Каждая строка хранит признаки о соответствующей секунде соотвествующего пациента

2 Теория методов и инструментов, используемых в работе

В данной главе будут рассмотрены некоторые методы, используемые в данной работе в ходе экспериментов и непосредственно влияющие на их качество.

2.1 K-fold валидация

Кросс-валидация или скользящий контроль - это статистический метод, необходимый для оценивания обобщающей способности алгоритмов. Данный метод заключается в фиктивном представлении наличия тестовой выборки без её создания.

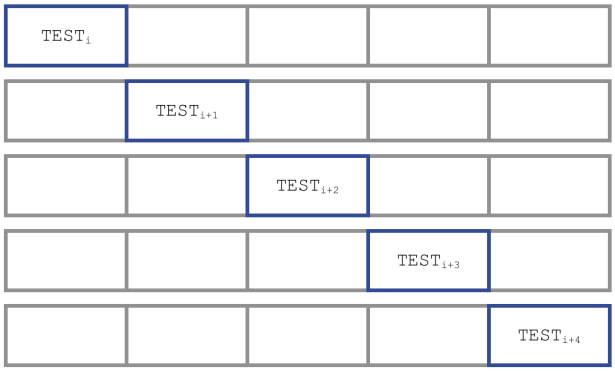

Рис. 12: Каждый фолд рассматривается как тестовая выборка на текущей итерации

Обучающая выборка автоматически делится на k непересекающихся и одинаковых по объему частей. Далее, модель обучается на k-1 частей

и тестируется на i-части данных, не задействованных в обучении. Далее начинается следующая итерация и модель обучается на следующих k-1 частей, причём i-ая часть участвует в обучении, а тестирование происходит на j-й части. Этот процесс повторяется k раз и позволяет оценивать модель намного более объективно, чем без данной методики.

В данной работе метод K-fold валидации будет использован в экспериментах с деревьями принятия решений.

2.2 Подбор гиперпараметров

Как и в абсолютном большинстве случаев, наша модель будет иметь некоторое количество параметров, значения которых не будут изменяться в результате её обучения, но будут прямо влиять на результат, выдаваемый ею. Эти параметры будут зафиксированы при инициализации модели. Такие параметры называются гиперпараметрами. Поскольку, как было сказано выше, гиперпараметры напрямую влияют на работу модели, а следовательно и на её точность, встаёт вопрос о выборе их значений при инициализации.

Одним из возможных решений этой проблемы является алгоритм поиска по сетке (grid search).

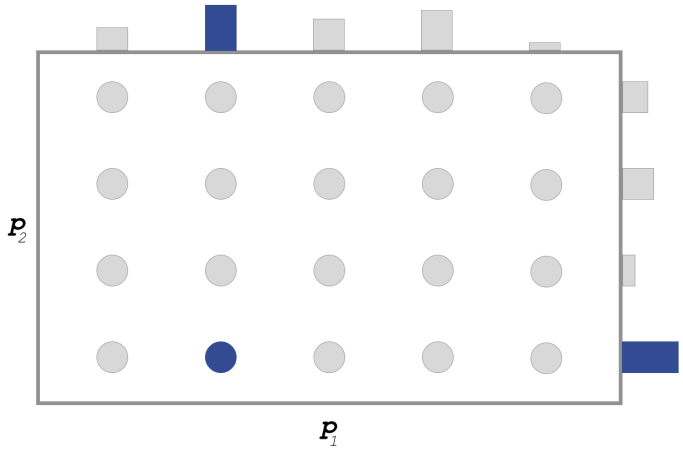

Рис. 13: Поиск наилучших гиперпараметров

Его суть состоит в том, что исходя из всевозможных значений разных параметров конструируется сетка их различных комбинаций. Далее вся совокупность данных разделяется на три выборки: обучающую, валидационную и тестовую. Инициализируясь на каждой из комбинаций гиперпараметров из сетки, модель обучается на обучающей выборке с последующей проверкой на валидационной выборке. Для выявления переобучения используется тестовая выборка.

Мы будем использовать алгоритм случайного поиска по сетке RandomizedSearchCV из библиотеки scikit-learn. Данный метод представляет из себя улучшение стандартного метода поиска по сетке. Главным его отличием от первоначального подхода является механизм случайного выбора комбинаций гиперпараметров и отбрасывания неперспективных значений гиперпараметров, за счёт чего метод работает значительно быстрее. Число проанализированных нами комбинаций гиперпараметров будет равняться 4,000.

2.3 Анализ важности признаков

При анализе имеющегося массива данных, довольно естественным видется предположение о более высокой значимости одних признаков над другими. В некоторых случаях, такой анализ может показать даже почти полное отсутствие влияния признака на результат работы модели.

Такой анализ может быть полезен в двух вещах. Во-первых, за счёт уменьшения количества признаков может быть уменьшено количество параметров модели, а следовательно, может быть увеличена скорость её работы и снижен её размер. Во-вторых, информация о признаках, не влияющих на работу модели, может быть использована при анализе данных и вынесении некоторых гипотез, объясняющих этот феномен.

Очевидно, что подобный анализ также может выявить признаки, имеющие сильное влияние на точность модели.

Мы будем использовать метод Permutation Feature Importance. Суть метода заключается в том, что мы обучаем некоторую модель на оригинальном наборе данных и берём её точность в виде точки отсчёта. После этого, итерируясь по признакам, мы перемешиваем их значения на каждой итерации и обучаем модель на получившемся наборе данных, после чего сравниваем точность модели с точностью исходной модели.

$$\mathcal{L}(\mathcal{M}(x_0, x_1, ..., x_n, x_{n+1}), y_0, y_1, ..., y_n, y_{n+1}) \approx \mathcal{L}(\mathcal{M}(x_0, \epsilon, ..., x_n, x_{n+1}), y_0, y_1, ..., y_n, y_{n+1})$$

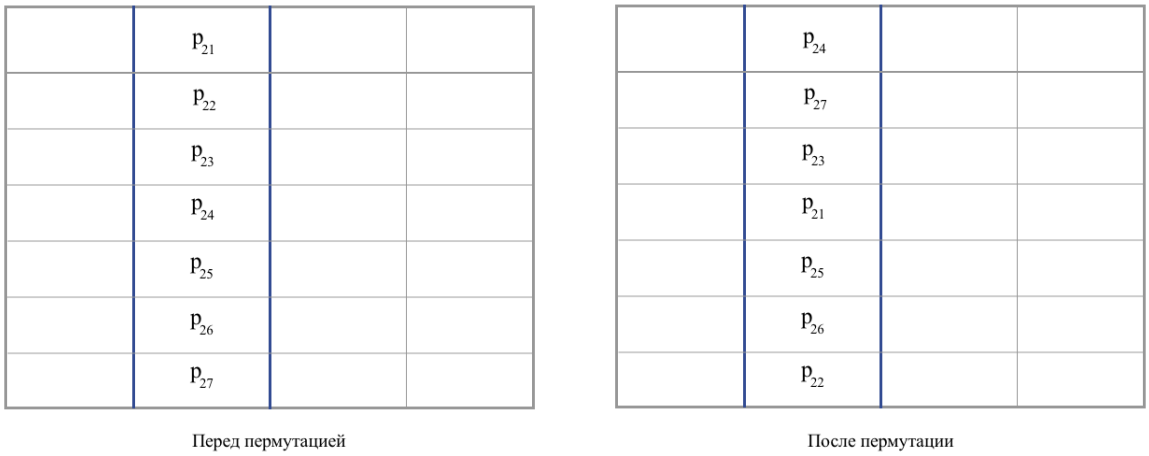

Рис. 14: Permutation Feature Importance

На основании этого сравнения мы и делаем вывод о важности признаков.

2.4 Регуляризация

Регуляризация представляет собой метод добавления некоторых дополнительных ограничений к условию с целью решить некорректно поставленную задачу для предотвращения переобучения.

Регуляризационная компонента как правило накладывает некий штраф за сложность модели, либо напрямую ограничевает её структуру: это может быть ограничение гладкости результирующей функции, ограничения по норме векторного пространства, ограничение на глубину решающих деревьев и т.д. С байесовской точки зрения многие методы регуляризации соответствуют добавлению некоторых априорных распределений на параметры модели.

Рассмотрим некоторые виды регуляризации, используемые в данной работе, которые позволили улучшить качество и объективность мо-

дели за счёт ввода дополнительных ограничений.

- L_1-регуляризция

  Одним из разновидностей регуляризации является следующий функционал:

  $$L_1 = \Sigma_i (y_i - \hat{y_i})^2 + \lambda ||w||_1$$

  Учитывая рассматриваемую метрику как Манхэттенское расстояние, можно заметить что происходит прибавление к функции потерь дополнительного члена в виде суммы весов в модуле.

- L_2-регуляризция

  Ещё одним путём модификации задачи с помощью добавления регуляризации является следующий функционал:

  $$L_2 = \Sigma_i (y_i - \hat{y_i})^2 + \lambda ||w||_2^2$$

  С помощью данной модификации мы переходим от поиска максимума функции правдоподобия к поиску максимума некой новой функции. Проэкспонируем эту новую функцию издержек:

  $$exp(J) = [\prod_{n=1}^{N} exp[-(y_n - w^T x_n)^2]] exp[-\lambda w^T w].$$

  $$P(Y|X, w) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} exp[-1 \frac{1}{2\sigma^2} (y_n - w^T x_n)^2]$$

  Таким образом, мы приходим к двух гауссианам: первоначальному и новому, учитывающему регуляризационную компоненту. В новом

гауссиане w является случайной величиной и имеет среднее значение, равное нулю, и дисперсию, равную $\frac{1}{\lambda}$. Второй гауссиан будем называть приором. Легко убедиться, что приор описывает w независимо от данных, так как, очевидно, он не зависит от X или Y.

$$J_{\text{old}} \sim lnP(Y|X,w)$$

$$J_{\text{new}} \sim lnP(Y|X,w) - lnP(w)$$

$$P(w|Y,X) = \frac{P(Y|X,w)P(w)}{P(Y|X)}$$

$$P(w|Y,X) \sim P(Y|X,w)P(w)$$

Таким образом, в приведённых выкладках угадывается правило Байеса. $P(w|Y,X)$ называется апостериором, а такой метод – методом максимизации апостериора. После приведённых преобразований, мы приходим к задаче поиска максимума апостериора, а не функции правдоподобия, как было ранее.

Итак, выписывая функцию затрат в матричной форме, получим:

$$J = (Y - Xw)^T(Y - Xw) + \lambda w^T w$$

$$J = Y^T Y. - 2Y^T Xw + w^T X^T Xw + \lambda w^T w$$

Беря от данного выражения производную и приравнивая её к нулю, получим:

$$\frac{\partial J}{\partial w} = -2X^T Y 2X^T Xw + 2\lambda w = 0$$

Решая относительно w приходим к

$$(\lambda I + X^T X)w = X^T Y$$

и наконец

$$w = (\lambda I + X^T X)^{-1} X^T Y$$

Очевидно, что полученный результат эквивалентен прежнему результату, за исключением наличия λ.

- Dropout

  Данный метод регуляризации нейронных сетей предназначен для уменьшения переобучения сети за счет предотвращения сложных коадаптаций отдельных нейронов на тренировочных данных во время обучения.

  Суть этого метода состоит в выбрасывании некоторого процента (например, 20%) случайных нейронов в процессе обучения модели.

  В дополнении к увеличению качества сети на тестовых данных при применении указанного метода, существуют также косвенные положительные эффекты, такие как, например, существенное увеличение скорости обучения нейронной сети.

- Регуляризция деревьев

  Деревья довольно сильно подвержены проблеме переобучения, поэтому процесс ветвления в какой-то момент нуждается в остановке.

Есть несколько способов делать это: можно ограничивать максимальную глубину дерева; можно ограничивать минимальное количество объектов в листе; ограничивать максимальное количество листьев в дереве. Также есть методики, согласно которым случайном образом вырезаются некоторое количесво вершин так, чтобы не наблюдалось сильного падения в качестве модели.

2.5 Логистическая регрессия

Одним из базовых методов в задаче классификации является статистическая модель логистической регрессии. Сформулируем решение поставленной задачи, основываясь на данном алгоритме: Пусть дана выборка $S = (x_i, y_i), i = 1, ..., m$, состоящая из m объектов, то есть записей эксперимента в каждый момент времени. Пусть каждый объект описывается n признаками $x_i \in R^n$ и принадлежит одному из трёх целевых классов: онкологические заболевания, сердечные заболевания и контрольная выборка. Для того чтобы разобраться в математике многоклассового алгоритма, сначала опишем работу для бинарного случая:

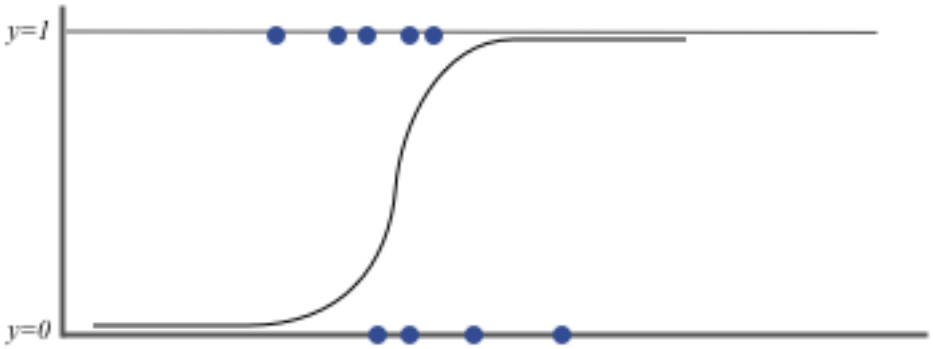

Рис. 15: Кривая логистической регрессии

У нас есть некая логитическая функция $f(\mathbf{x}; \mathbf{w})$, зависящая от признаков - $\mathbf{x}$ и параметров модели (коэффициентов регрессии) - $\mathbf{w}$.

$$f(\mathbf{x}; \mathbf{w}) = \sigma\left(\mathbf{w}^\top \mathbf{x}\right) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}, \quad \text{где} \quad \left(\mathbf{w}^\top \mathbf{x}\right) = \mathbf{w}_0 + \sum_{i=1}^{n} \mathbf{w}_i \cdot \mathbf{x}_i, \quad (1)$$

и так как задача бинарная, значение целовой переменной $y \in \{0, 1\}$. Предположим, что вероятность того, что $y = 1$ записывается следующим образом $P(y = 1 \mid \mathbf{x}; \mathbf{w}) = f(\mathbf{x}; \mathbf{w})$, тогда из свойств вероятностей получим, что $P(y = 0 \mid \mathbf{x}; \mathbf{w}) = 1 - f(\mathbf{x}; \mathbf{w})$.

Для многоклассового случая наша целевая переменная будет принимать значения $y \in \{1, 2, \ldots, K\}$. Тут применяется очень понятная и простая идея, вместо того, чтобы модель выводила вероятность принадлежности нашего объекта к первому классу, будем выводить вектор, элементы котрого показывают некую оценку принадлежности объекта к каждому из классов - $[\mathbf{w}_1^\top \mathbf{x}, ..., \mathbf{w}_k^\top \mathbf{x}]^\top$. И чтобы сделать из этого вектора вероятности принадлежности нашего объекта к каждому из классов, применем экспоненту$[e^{(\mathbf{w}_1^\top \mathbf{x})}, ..., e^{(\mathbf{w}_K^\top \mathbf{x})}]$, не забыв при этом все отнормировать, так как сумма вероятностей должна равняться 1:

$$P(y = k \mid \mathbf{x}; \mathbf{w}_k) = \frac{e^{\mathbf{w}_K^\top \mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}}}$$

Таким образом, ответом алгоритма будет являться следующий вектор:

$$f(\mathbf{x}; \mathbf{W}) = \begin{bmatrix} P(y = 1 \mid \mathbf{x}; \mathbf{w}_1) \\ P(y = 2 \mid \mathbf{x}; \mathbf{w}_2) \\ \vdots \\ P(y = K \mid \mathbf{x}; \mathbf{w}_K) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}}} \begin{bmatrix} e^{(\mathbf{w}_1^\top \mathbf{x})} \\ e^{(\mathbf{w}_2^\top \mathbf{x})} \\ \vdots \\ e^{(\mathbf{w}_K^\top \mathbf{x})} \end{bmatrix}, \text{где} \quad \mathbf{W} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix}, \tag{2}$$

то задача будет из себя представлять ничто иное как функцию $softmax(\mathbf{W}x)$. Остается только разобраться, как обучать такую модель. Для этого используем метод максимального правдоподобия, так же как и в бинарной задаче. В таком случае, задача логистической регрессии будет сформулирована так: требуется минимизировать следующую функцию ошибки

$$\begin{aligned} J(\mathbf{W}) &= -\log L(\mathbf{W}) = -\sum_{m=1}^{M} \log P\left(y^{(m)} \mid \mathbf{x}^{(m)}; \mathbf{W}\right) \\ &= -\sum_{m=1}^{M} \sum_{k=1}^{K} \mathbb{I}\left\{y^{(m)} = k\right\} \log \frac{e^{\mathbf{w}_k^\top \mathbf{x}^{(m)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}^{(m)}}} \end{aligned} \tag{3}$$

которая эквивалентна минимизации логорифмической вероятности или

отрицательной функции правдоподобия. В формуле (3) $\mathbb{I}\left\{y^{(m)} = k\right\}$ - переменная только с двумя возможными значениями, равная 1 когда $y^{(m)} = k$ и равная 0 во всех остальных случаях, таким образом, эта переменная будет равняться 1 только если объект из обучающей выборки действительно принадлежит классу k.

Частная роизводная при этом будет вычислятся по следующей формуле

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_k} = -\sum_{m=1}^{M} \left(\mathbb{I}\left\{y^{(m)} = k\right\} - f_k\left(\mathbf{x}^{(m)}; \mathbf{w}_k\right)\right) \mathbf{x}^{(m)} \tag{4}$$

А раз есть частная производная по параметрам модели, то возможно осуществить градиент спуск по пространству коэффециентов регрессии, минимизируя тем самым функцию ошибки.

Так же стоит упомянуть про one-hot-encoding, который целесообразно использовать в данной задаче. Данный метод позволяет представить нашу целевую переменную, которая имела значения $y \in \{1, 2, \ldots, K\}$ в виде one-hot вектора:

$$y^{(m)} = [0, 0, ..., 0, 1, 0, ..., 0]^\top,$$

у такого вектора единица будет означать класс, к которому принадлежит объект $x^{(m)}$ с целевой переменной $y_k^{(n)}$, а все остальные значения будут нулями. Это удобное представление для целевого признака, поскольку оно позволяет нам векторизовать алгоритм. И мы можем записать функцию потерь и градиент как:

$$J(\mathbf{W}) = -\sum_{m=1}^{M}\sum_{k=1}^{K} y_k^{(m)} \log \frac{e^{\mathbf{w}_k^\top \mathbf{x}^{(m)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}^{(m)}}}$$

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_k} = -\sum_{m=1}^{M} \left(y_k^{(m)} - f_k\left(\mathbf{x}^{(m)}; \mathbf{w}_k\right)\right) \mathbf{x}^{(m)}$$

В методе градиентного спуска будет использоваться случайная инициализация весов модели из нормального распределения с нулевым средним и дисперсией равной $1/N$, где N - это размерность входного вектора x:

$$w_i \sim \mathcal{N}\left(\mu = 0, \sigma^2 = \frac{1}{M}\right),$$

и начинается итеративный процесс. На каждой итерации вычисляется вектор градиента нашей функции потерь по обучаемым параметрам и вычитаем его, домноженного на α - скорость обучения, из текущих значений параметров.

$$\mathbf{W}_k = \mathbf{W}_{k-1} - \alpha \nabla_{\mathbf{W}} J\left(\mathbf{W}_{k-1}\right),$$

таким образом получаются новые параметры, которые лучше минимизируют нашу функцию потерь. Метод прекращает работу, когда точность на валидационной выборке перестает расти.

Также следует упомянуть, что логистическая регрессия приветствует нормализацию входных данных.

2.6 Деревья и бустинги

2.6.1 Решающие деревья

Рассмотрим алгоритм, называющийся решающие деревья (decision trees). Решающее дерево предсказывает значение целевой переменной с помощью применения последовательности простых решающих правил, которые также часто называют предикатами.

Пусть объекты некоторого набора даных имеют n признаков с, положим, вещественными значениями. Решение о том, к какому классу будет отнесён текущий объект выборки в соответствии с сутью рассматриваемого алгоритма будет приниматься с помощью прохода от корня дерева к некоторому листу. В каждом узле дерева находится предикат. Если предикат верен для текущего примера из выборки, мы переходим дальше в правого потомка, если нет — в левого. Иными словами, каждый предикат порождает разделение текущего подмножества пространства признаков на две части. Стоит иметь ввиду, что чаще всего предикаты - лишь взятие порога по значению какого-либо признака. В самих же листьях хранятся некоторые значения предсказаний: в нашем случае - метки классов состояния здоровья пациентов.

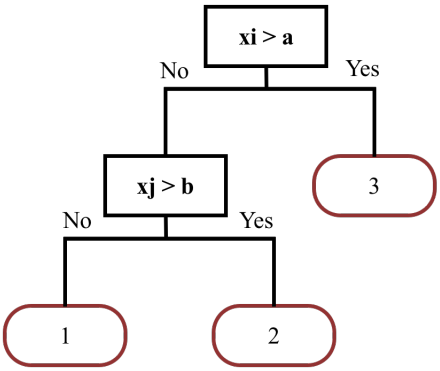

Рис. 16: Пример построенного дерева решений

Таким образом, решающее дерево - это бинарное дерево, в котором:
- каждой внутренней вершине v приписан предикат $B_v : X \to 0, 1$;
- каждой листовой вершине v приписан прогноз $c_v \in Y$, где Y — область значений целевой переменной (в случае классификации листу может быть также приписан вектор вероятностей классов).

В ходе предсказания осуществляется проход по этому дереву к некоторому листу. Для каждого объекта выборки x движение начинается из корня. В очередной внутренней вершине v проход продолжится вправо, если $B_v(x) = 1$, и влево, если $B_v(x) = 0$. Проход продолжается до момента, пока не будет достигнут некоторый лист, и ответом алгоритма на объекте x считается прогноз c_v, приписанный этому листу.

Предикат B_v может иметь, вообще говоря, произвольную структуру, но, как правило, на практике используют просто сравнение с порогом $t \in R$ по какому-то j-му признаку: $B_v(x, j, t) = [x_j \leq t]$.

При проходе через узел дерева с данным предикатом объекты будут отправлены в правое поддерево, если значение j-го признака у них меньше либо равно t, и в левое — если больше.

2.6.2 Бэггинг

Теперь введём операцию бэггинга (bagging, bootstrap aggregation). Идея метода заключается в следующем. Выберем из обучающей выборки n примеров. Будем выбирать примеры равновероятно, с повторением. Получим новую выборку X_1, в которой некоторых элементов исходной выборки не будет, а какие-то могут войти несколько раз. С помощью некоторого алгоритма b обучим на этой выборке модель $b_1(x) = b(x, X_1)$.

Повторим процедуру: сформируем вторую выборку X_2 из n элементов с возвращением и с помощью того же алгоритма обучим на ней модель $b_2(x) = b(x, X_2)$. Повторив k раз, получим k моделей, обученных на k выборках. Чтобы получить одно предсказание, усредним предсказания всех моделей:

$$a_x = \frac{1}{k} * (b_1(x) + ... + b_k(x))$$

Результат такой обладает намного меньшей дисперсией, чем каждый из отдельных результатов моделей $b_{i,i=1,...,k}$.

2.6.3 Случайный лес

Построим ансамбль алгоритмов, где базовый алгоритм — это решающее дерево. Будем строить по следующей схеме:

1. Для построения i-го дерева: Сначала, как в обычном бэггинге, из обучающей выборки X выбирается с возвращением случайная подвыборка X_i того же размера. Затем случайно выбираются $n < N$ признаков, где N — полное число признаков (то есть, мы используем метод случайных подпространств). Так же, как и подвыборка для каждого дерева, набор признаков свой для каждого из деревьев. Такой приём позволя-

ет управлять степенью скоррелированности базовых алгоритмов. На n выбранных признаках подвыборки X_i строится i-е дерево.

2. Чтобы получить предсказание ансамбля на тестовом объекте, усредняем отдельные ответы деревьев (для регрессии) или берём самый популярный класс (для классификации).

3. Таким образом строится модель Random Forest (случайный лес) – комбинация бэггинга и метода случайных подпространств над решающими деревьями.

Ошибка модели (на которую мы можем повлиять) состоит из смещения и разброса. Разброс мы уменьшаем с помощью процедуры бэггинга. На смещение, которое должно быть минимально, бэггинг не влияет. Поэтому смещение должно быть небольшим у самих деревьев, из которых строится ансамбль. У неглубоких деревьев малое число параметров, то есть дерево способно запомнить только верхнеуровневые статистики обучающей подвыборки. Они во всех подвыборках будут похожи, но будут не очень подробно описывать целевую зависимость. Поэтому при изменении обучающей подвыборки предсказание на тестовом объекте будет стабильным, но не точным (низкая дисперсия, высокое смещение). Наоборот, у глубоких деревьев нет проблем запомнить подвыборку подробно. Поэтому предсказание на тестовом объекте будет сильнее меняться в зависимости от обучающей подвыборки, зато в среднем будет близко к истине (высокая дисперсия, низкое смещение). Вывод: используем глубокие деревья.

Ограничивая число признаков, которые используются в обучении одного дерева, мы также управляем качеством случайного леса. Чем больше признаков, тем больше корреляция между деревьями и тем меньше чувствуется эффект от ансамблирования. Чем меньше признаков, тем

слабее сами деревья. Обычно берётся корень из числа всех признаков для классификации и треть признаков для регрессии.

Выше было показано, что увеличение числа элементарных алгоритмов в ансамбле не меняет смещения и уменьшает разброс. Так как число признаков и варианты подвыборок, на которых строятся деревья в случайном лесе, ограничены, уменьшать разброс до бесконечности не получится. Поэтому имеет смысл построить график ошибки от числа деревьев и ограничить размер леса в тот момент, когда ошибка перестанет значимо уменьшаться.

Вторым практическим ограничением на количество деревьев может быть время работы ансамбля. Однако есть положительное свойство случайного леса: случайный лес можно строить и применять параллельно, что сокращает время работы, если у нас есть несколько процессоров. Но процессоров, скорее всего, всё же сильно меньше числа деревьев, а сами деревья обычно глубокие. Поэтому на большом числе деревьев Random Forest может работать дольше желаемого и количество деревьев можно сократить, немного пожертвовав качеством.

2.6.4 Бустинг

Бустинг (boosting) – это ансамблевый метод, в котором так же, как и в методах выше, строится множество базовых алгоритмов из одного семейства, объединяющихся затем в более сильную модель. Отличие состоит в том, что в бэггинге и случайном лесе базовые алгоритмы учатся независимо и параллельно, а в бустинге – последовательно.

Пусть задан некий набор данных и некоторая дифференциируемая функция потерь

$$(x_i, y_i)_{i=1,\dots,n}, \text{где L(y, f) - функция потерь.} \tag{5}$$

Пусть есть некая оптимальная модель на нашей обучающей выборке

$$\hat{f}(x) = \arg\min L(y, f(x)) = \arg\min \mathbb{E}_{x,y}[L(y, f(x))] \tag{6}$$

которая принадлежит некоторому параметрическкому семейству

$$\hat{f}(x) = f(x, \hat{\theta}), \tag{7}$$

$$\hat{\theta} = \arg\min \mathbb{E}_{x,y}[L(y, \theta)]$$

Тогда задача поиска оптимальной модели сводится к задаче поиска оптимальных параметров

$$\hat{f}(x) = \Sigma_{i=0}^{t-1} \hat{f}_i(x) \tag{8}$$

Будем строить следующую модель так, чтобы она минимизировала ошибку всего предыдущего ансамбля. Соответственно, на этом шаге необходимо решить задачу следующего вида:

$$(\rho_t, \theta_t) = \arg\min \mathbb{E}_{x,y}[L(y, \hat{f}(x))] + \rho * h(x, \theta) \tag{9}$$

То есть модель, которую мы добавляем на $t-$м шаге представляется в виде

$$\hat{f}(x) = \rho_t * h(x, \theta_t) \tag{10}$$

То есть нужно минимизировать функцию ошибки между истинными значениями и результатом предыдущего ансамбля в совокупности с

$t-$й моделью с неким коэффициентом ρ. Иными словами, каждый следующий базовый алгоритм в бустинге обучается так, чтобы уменьшить общую ошибку всех своих предшественников.

$$\hat{\theta} = \arg\min \Sigma_{i=1}^{n} (r_i t - h(x_i, \theta))^2 \tag{11}$$

Как следствие, итоговая композиция будет иметь меньшее смещение, чем каждый отдельный базовый алгоритм. Поскольку основная цель бустинга – уменьшение смещения, в качестве базовых алгоритмов часто выбирают алгоритмы с высоким смещением и небольшим разбросом. В нашем случае в качестве базовых классификаторов выступают деревья и их глубина должна быть сравнительно небольшой.

Ещё одной важной причиной для выбора моделей с высоким смещением в качестве базовых является то, что такие модели, как правило, быстрее учатся. Это важно для их последовательного обучения, которое может стать очень дорогим по времени, если на каждой итерации будет учиться сложная модель.

На текущий момент основным видом бустинга с точки зрения применения на практике является градиентный бустинг, о котором подробно рассказывается в соответствующей главе.

Хотя случайный лес – мощный и достаточно простой для понимания и реализации алгоритм, на практике он чаще всего уступает градиентному бустингу. Поэтому градиентный бустинг сейчас – одно из основных решений в случае, если работа происходит с табличными данными.

2.7 LSTM

Пытаясь решить проблему восприятия информации нейронной сетью в некотором контексте, были предложены различные классы архи-

тектур, основным из которых являются реккурентные нейронные сети. Реккурентные нейронные сети имеют обратные связи, которые позволяют передать информацию о прошлых токенах в последовательности, тем самым обогатив токен информацией о его контексте. Одним из примеров реккурентных нейронных сетей является архитектура LSTM (долгая краткосрочная память). Данная архитектура была представлена Зеппом Хохрайтер и Юргеном Шмидхубером в 1997 году и развивается по сей день. В отличии от простых RNN, сети LSTM позволяют усваивать зависимости между более удалёнными друг от друга токенами.

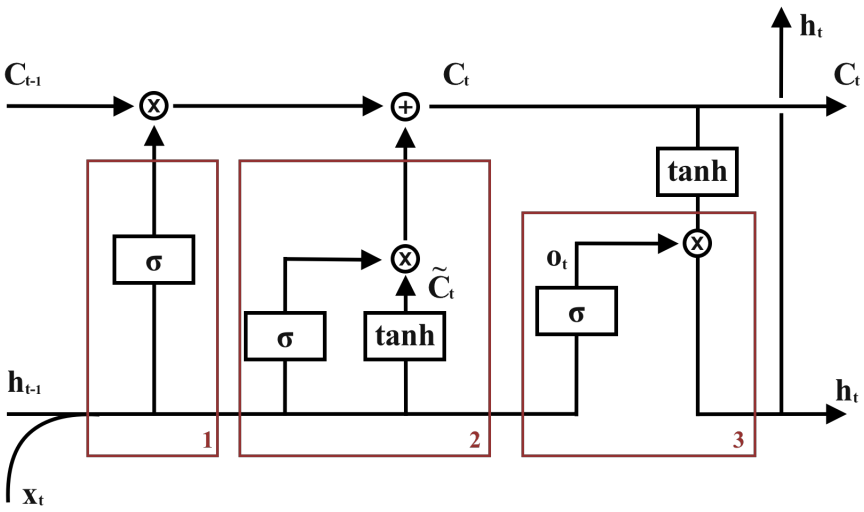

Рис. 17: Архитектура ячейки в LSTM сети

Основной компонент LSTM – это состояние ячейки - горизонтальная линия, проходящая через всю ячейку слева направо. Информация, которая течёт по линии состоянии ячейки, подвергается лишь двум линейным преобразованиям. Также сеть содержит три фильтра, которые позволяют пропускать информацию на основании некоторых условий. Они состоят из слоя сигмоидальной нейронной сети и операции поточеч-

ного умножения. Первый фильтр - это "фильтр слоя забывания" (forget gate layer, участок один на изображении). Он позволяет "забывать" некоторый контекст, который в моменте передачи информации через фильтр более не является актуальным.

Следующий фильтр (участок два на изображении) помогает решить, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей. Сначала сигмоидальный слой под названием “слой входного фильтра"(input layer gate) определяет, какие значения следует обновить. Затем tanh-слой строит вектор новых значений-кандидатов $\sim C_t$, которые можно добавить в состояние ячейки. Итак, чтобы заменить старое состояние ячейки C_{t-1} на новое состояние C_t, сигнал сначала проходит через forget gate layer, после чего поэлементно складывается с вектором $i_t^{\sim} C_t$ новой информации.

Третий фильтр - выходной (output gate layer, участок 3 на изображении). Он позволяет решить, какую именно информацию мы хотим получать на выходе ячейки. Сначала мы применяем сигмоидальный слой, который решает, какую информацию из состояния ячейки мы будем выводить. Затем значения состояния ячейки проходят через tanh-слой, чтобы получить на выходе значения из диапазона от -1 до 1, и перемножаются с выходными значениями сигмоидального слоя, что позволяет выводить только требуемую информацию.

3 Результаты

3.1 Логистическая регрессия

Была построена модель, реализующая в себе данный функционал. Модель была обучена на данных с предобработкой, рассмотренной в предыдущих главах. Точность модели равняется 71.26%.

	precision	recall	f1-score	support
Cardiology	0.49	0.72	0.58	473
Control	0.59	1.00	0.75	129
Onco	0.88	0.68	0.77	1395
accuracy			0.71	1997
macro avg	0.65	0.80	0.70	1997
weighted avg	0.77	0.71	0.72	1997

Результаты модели на оригинальных данных

После анализа полученных результатов, была высказана гипотеза об избыточной структуре наборе данных и о возможности решения данной задачи на гораздо меньшем количестве значений концентраций. Чтобы проверить эту гипотезу, был применён критерий хи-квадрат.

TO-DO: хи-квадрат Критерий показал верность высказанной гипотезы. В качестве проверки, была обучена та же модель на 50 полученных признаках. Модель показала точность в 69.4%.

	precision	recall	f1-score	support
Cardiology	0.50	0.74	0.59	473
Control	0.48	1.00	0.65	129
Onco	0.88	0.65	0.75	1395

Результаты модели на данных с 50 колонками

Принимая во внимание близкие по значению метрики оценки модели, такие как метрика полноты и точности, можно с уверенностью говорить об избытке полученного в ходе экспериментов набора данных. Впоследствии планируется изучение фундаментальных причин данного результата с целью усовершенствования проведений экспериментов и построение более обоснованных моделей.

3.2 Бустинг

Мы использовали модель LGBMClassifier из фреймворка LightGBM. Данный фреймворк обрядает рядом положительных черт - он хорошо задокументирован, имеет большое распространение и достаточно прост.

Модель была обучена на предобработанных данных. Для валидации модели был применён алгоритм k-fold с размером 8. Параметр learning rate был задан 0.3; boosting_type = "gbdt¨ , поскольку мы используем деревья принятия решений. Также в качестве фактора регуляризации было ограничено количество листьев в каждом дереве.

Для оценки обобщающей способности модели мы использовали алгоритм K-Fold. Данный метод заключается в фиктивном представлении наличия тестовой выборки без её создания. Обучающая выборка автоматически делится на k непересекающихся и одинаковых по объему частей. Далее, модель обучается на k-1 частей и тестируется на i-части данных, не задействованных в обучении. Далее начинается следующая итерация и модель обучается на следующих k-1 частей, причём i-ая часть участвует в обучении, а тестирование происходит на j-й части. Этот процесс повторяется k раз и позволяет оценивать модель намного более объективно, чем без данной методики.

Для инициализации параметров модели, не участвующих в обучении, мы использовали метод случайного поиска по сетке. Суть этого метода состоит в составлении всевозможных комбинаций гиперпараметров, случайном выборе комбинаций из всевозможных вариантов и попытке отбрасывания бесперспективных пар гиперпараметров для уменьшения времени поиска лучшей комбинации. Также мы использовали регуляризацию L2, позволяющую увеличить способность модели к обобщению.

В результате проведения экспериментов, нами была получена модель, достигающая усреднённой точности на тестовой выборке в **77.3%**.

	precision	recall	f1-score	support
Cardiology	0.52	0.63	0.57	473
Control	1.00	1.00	1.00	129
Onco	0.87	0.80	0.83	1395

Результаты модели на оригинальных данных

Из анализа метрик полноты, точности и F-меры видно, что модель отлично классифицирует пациентов из контрольной выборки, удовлетворительно выявляет людей из выборки онко-заболеваний и значительно хуже определяет пациентов с кардиоваскулярным заболеванием.

3.3 LSTM

Для обучения модели были использованы предобработанные данные. Каждой матрице признаков, формирующих данные записи одного человека, был поставлен в соответствие вектор ответов той же высоты, то есть нейронная сеть была обучена на каждой из строк исходного массива данных.

В результате обучения сети на данных, указанных выше, нейросетевая модель LSTM показала результат точностью в **0.8954%**. Согласно графикам, представленным ниже, для достижения стабильного результата в процессе обучения требуется всего около 50 эпох, что свидетельствует о нормальном обучении нейронной сети.

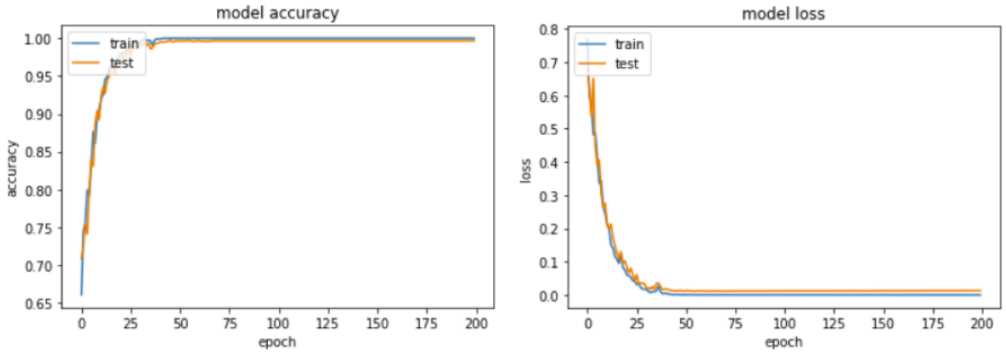

Рис. 18: Бла-бла-бла

Анализируя точность (precision), полноту (recall) и f1-меру (f1-score) модели-классификатора, можно сделать вывод о том, что TODO

	precision	recall	f1-score	support
Cardiology	0.90	0.84	0.87	594
Control	0.77	0.90	0.83	228
onco	0.93	0.93	0.93	918
accuracy			0.90	1740
macro avg	0.87	0.89	0.88	1740
weighted avg	0.90	0.90	0.90	1740

Анализируя графики, можно с уверенностью утверждать об отсутствии переобучения модели. На графике изменения величины функции

потерь в зависимости от количества эпох обучения модели можно увидеть

Заключение

В данной работе были рассмотрены алгоритмы, позволяющие классифицировать пациентов с сердечно-сосудистыми и онкологическими заболеваниями по данным масс-спектрометра. Были качественно изучены отрицательные и положительные стороны, этих алгоритмов, был разработан программный комплекс на языке Python с использованием различных фреймворков.

Результаты экспериментов были оценены с помощью метрик успеха, позволяющих утверждать, что поставленная задача была решена значительным способом с помощью нейронной сети класса LSTM. Эксперименты также показали, что поставленная задача может, хоть и менее успешно, решаться с помощью алгоритмов градиентного бустинга деревьев решений. Было произведено сравнение результатов их работы, таких как качество классификации, время обучения и другие параметры. Результаты этого сравнения можно наблюдать в таблице 1.

Название метода	Результаты
Логистическая регрессия	...
Бустинг	...
LSTM	...

Имя таблицы

По данным результатам был сделан вывод о подтверждении гипотезы возможной автоматизации процесса классификации юольных людей с помощью технологии масс-спектрометрии. Учитывая возможность улучшения качества данных, ровно как и их количества, полученные результаты можно считать успешными и наглядными, а данный метод - заслуживающим внимания и дальнейшего исследования.

Литература

Список литературы

[1] *Малиновская Людмила Кирилловна N.* Применение метода протонной масс-спектрометрии выдыхаемого воздуха в диагностике хронической сердечной недостаточности // City: СЕЧЕНОВСКИЙ УНИВЕРСИТЕТ, 2020.

[2] *Ralf C. Staudemeyer and Eric Rothstein Morris N.* Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks // City: arXiv, 2019.

[3] *Chen, Tianqi and Guestrin, Carlos N.* XGBoost // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016. City: arXiv, 2016.

[4] *Terence Shin N.* An Extensive Step by Step Guide to Exploratory Data Analysis // https://towardsdatascience.com/an-extensive-guide-to-exploratory-data-analysis-ddd99a03199e, 2021.

Приложение

```
# LSTM model

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.optimizers.schedules import ExponentialDecay
from sklearn.metrics import mean_absolute_error as mae
from sklearn.preprocessing import RobustScaler, normalize
from sklearn.model_selection import train_test_split, GroupKFold, KFol

model = keras.models.Sequential([
    keras.layers.Input(shape=train.shape[-2:]),
    keras.layers.Bidirectional(keras.layers.LSTM(300, return_sequences=
    keras.layers.Bidirectional(keras.layers.LSTM(250, return_sequences=
    keras.layers.Bidirectional(keras.layers.LSTM(150, return_sequences=
    keras.layers.Bidirectional(keras.layers.LSTM(100, return_sequences=
    keras.layers.Dense(50, activation='selu'),
    keras.layers.Dense(1),
])
model.compile(optimizer="adam", loss="mae")
scheduler = ExponentialDecay(1e-3, 400*((len(train)*0.8)/BATCH_SIZE),
lr = LearningRateScheduler(scheduler, verbose=1)
es = EarlyStopping(
    monitor="val_loss", patience=15,
```

```
    verbose=1, mode="min",
    restore_best_weights=True
)

history = model.fit(
    train, y_train, epochs=EPOCH,
    batch_size=BATCH_SIZE, callbacks=[lr],
    validation_split = 0.1,
    validation_data=(test, y_test)
)
```