Санкт-Петербургский Государственный Университет

Факультет прикладной математики – процессов управления

Кафедра технологии программирования

**Вопилова Ольга Анатольевна**

**Выпускная квалификационная работа**

**Применение методов глубокого обучения в медицинской диагностике**

Уровень образования: бакалавриат

Направление: 01.03.02 «Прикладная математика и информатика»

Основная образовательная программа: «Прикладная математика, фундаментальная информатика и программирование»

Профиль: «Математическое и программное обеспечение вычислительных машин»

Научный руководитель:

кандидат физ.-мат. наук,

доцент кафедры управления

медико-биологическими системами

Платонов Алексей Викторович

Рецензент:

кандидат физ.-мат. наук,

доцент кафедры теории

систем управления

электрофизической аппаратурой

Головкина Анна Геннадьевна

Санкт-Петербург

2021 год

Оглавление

[Введение 3](#_Toc72969921)

[Постановка задачи 5](#_Toc72969922)

[Обзор литературы 8](#_Toc72969923)

[Используемые методы 10](#_Toc72969924)

[Линейные модели 10](#_Toc72969925)

[Логистическая регрессия 10](#_Toc72969926)

[K ближайших соседей 11](#_Toc72969927)

[Деревья решений 12](#_Toc72969928)

[Ансамбли деревьев решений 14](#_Toc72969929)

[Случайный лес 14](#_Toc72969930)

[Градиентный бустинг деревьев 15](#_Toc72969931)

[Полносвязные нейронные сети прямого распространения 18](#_Toc72969932)

[SuperTML 20](#_Toc72969933)

[NODE Нейронные ансамбли небрежных решающих деревьев 21](#_Toc72969934)

[Методы борьбы с несбалансированностью классов 25](#_Toc72969935)

[SMOTE 25](#_Toc72969936)

[Undersampling 26](#_Toc72969937)

[Отбор наиболее информативных признаков 26](#_Toc72969938)

[Отбор на основе модели случайного леса 26](#_Toc72969939)

[Вычислительная часть 28](#_Toc72969940)

[Используемый язык программирования 28](#_Toc72969941)

[Предобработка данных 28](#_Toc72969942)

[Распределение классов 30](#_Toc72969943)

[Применение методов машинного обучения 31](#_Toc72969944)

[Анализ результатов 43](#_Toc72969945)

[Выводы 45](#_Toc72969946)

[Интерфейс 46](#_Toc72969947)

[Заключение 48](#_Toc72969948)

[Список литературы 49](#_Toc72969949)

# 

# Введение

Инфаркт миокарда*–* это острое сердечно-сосудистое заболевание, обусловленное резким прекращением кровотока в одной или нескольких корональных артериях, что приводит к гибели части сердечной ткани. Это одна из основных причин инвалидности во взрослом возрасте.

Основными осложнениями после инфаркта миокарда являются кардиогенный шок, острая левожелудочковая недостаточность вплоть до отека легких, жизнеугрожающие нарушения ритма сердца, снижение артериального давления, внезапная смерть.

К факторам риска развития инфаркта миокарда относятся курение, ожирение, недостаток двигательной активности, наследственность. Смертность от инфаркта миокарда среди всех заболевших составляет 10-12%.

Как известно, прогноз пациентов, переносящих инфаркт миокарда, неодинаков и зависит от множества факторов. Прогнозирование исхода для таких пациентов, а также выявление важнейших из признаков, достоверно влияющих на развитие в дальнейшем осложнений и на летальный исход, являются одними из важнейших задач кардиологии.

Задачу прогнозирования летального исхода после инфаркта миокарда можно решать с помощью машинного обучения. Это научная область, находящаяся на пересечении статистики, искусственного интеллекта и компьютерных наук. Для применения методов машинного обучения необходим некоторый набор данных, с помощью которого можно обучить различные алгоритмы и, сравнив результаты, полученные разными моделями, выбрать наилучший из алгоритмов для решения задачи.

Для обучения моделей машинного обучения имеется база данных, предоставленная военно-медицинской академией имени С. М. Кирова. В базе имеются сведения о 1040 пациентах, перенёсших инфаркт миокарда. Задача состоит в выявлении наиболее значимых в плане прогноза переменных и построении прогностической модели, оценивающей влияние различных факторов на исход при инфаркте миокарда.

Для решения этой задачи использовались различные методы машинного обучения, такие как, например, алгоритм случайного леса [1], градиентный бустинг деревьев решений [2], логистическая регрессия [3], а также некоторые алгоритмы глубокого обучения. Затем с помощью различных метрик эффективности выбиралась наилучшая в плане прогноза модель. Кроме того, проводился анализ на выявление наиболее информативных факторов. В связи с несбалансированностью классов в имеющемся наборе данных применялись разные методы борьбы с данной проблемой. [4, 5]

# 

# Постановка задачи

В имеющейся базе данных представлены в основном вещественные и категориальные факторы, представляющие собой различные данные о пациентах. Все факторы можно условно разделить на 12 блоков: первый блок – “Паспортная часть”. Тут представлена информация о возрасте пациента, его росте, массе, диагнозе и т.д. Второй блок содержит информацию о жалобах пациента, перенёсшего инфаркт миокарда: длительность, характер и интенсивность боли, одышка, количество приступов и другие. Третий блок – анамнез – содержит сведения о человеке, полученные в результате опроса пациента врачом. В четвертом блоке находятся факторы риска: наследственность, курение, степень ожирения и другие. В пятом блоке представлены данные осмотра и физикального обследования. Шестой блок включает в себя результаты лабораторных исследований. В следующем блоке описаны нарушения сердечного ритма. Далее представлены результаты количественного анализа ЭКГ. В девятом блоке содержится информация об эхокардиографии. Затем идут данные об имеющихся осложнениях: отек легких, психические нарушения, аритмия и т.д. Далее идут результаты гармонического анализа ЭКГ, а в конце представлены данные о результатах суточного мониторирования ЭКГ.

Поскольку в таблице имеются столбцы с пустыми ячейками, то есть имеются факторы, информация о которых дана не для всех пациентов, то значения таких признаков заполнялись в случае вещественных факторов средним арифметическим значением, а в случае категориальных факторов – наиболее часто встречающимся значением признака. При этом предварительно исходный набор данных был разделён на две группы в зависимости от того, был исход летальным или благоприятным. Поэтому теперь в наборе данных пропуски заполнены средним или наиболее частым значением по классу, который необходимо научиться предсказывать.

Вещественные данные необходимо стандартизировать. Данный процесс приводит значения всех признаков из набора данных, независимо от их начальных распределений и единиц измерения, к набору значений из распределения с нулевым средним и [стандартным отклонением](https://wiki.loginom.ru/articles/mean-square-deviation.html), равным 1.

Далее уже преобразованный набор данных необходимо разделить на 3 части: тренировочный, тестовый и валидационный, поскольку для проверки эффективности модели нельзя брать данные, которые использовались при обучении. Таким образом, на тренировочном наборе тренируется каждая из моделей, на тестовом проверяется их обобщающая способность и сравниваются различные алгоритмы машинного обучения, а на валидационном сравниваются различные методы по борьбе с несбалансированностью классов.

На этом препроцессинг данных заканчивается, теперь можно приступать к обучению моделей и выбору лучшей из них.

Очевидно, с точки зрения машинного обучения данная задача относится к классу задач обучения с учителем [3], поскольку в наборе данных имеются пары объект-ответ. То есть помимо самих данных о пациентах известен и их исход: летальный или благоприятный.

Машинное обучение с учителем рассматривает две основные задачи: классификация, которая в свою очередь подразделяется на бинарную (разделение на два класса) и мультиклассовую, и регрессия (прогнозирование непрерывного числа) [3]. Данная задача принадлежит к классу задач бинарной классификации, поскольку прогнозируемая переменная может принимать только два значения: 1, если исход летальный, и 0 – в противном случае.

К основным алгоритмам классификации относятся линейные модели, куда входит, например, логистическая регрессия, а также деревья решений и ансамбли деревьев решений, куда входят такие алгоритмы машинного обучения как алгоритм случайного леса, который решает проблему склонности к переобучению деревьев решений, и градиентный бустинг деревьев. Кроме того, иногда для решения задач классификации применяют нейронные сети, но, как правило, на табличных данных они дают более плохое качество, чем ансамбли деревьев, и подходят больше для неструктурированных данных. Однако в последние несколько лет были предложены некоторые методы глубокого обучения [6, 7], которые подходят для применения на табличных данных и во многих случаях превосходят по качеству градиентный бустинг деревьев решений.

Цель исследования: предсказание исхода после инфаркта миокарда при помощи различных методов машинного и глубокого обучения.

Основные задачи:

* Предобработка данных (обработка пропущенных значений, стандартизация, кодирование категориальных признаков, разделение на тренировочный, тестовый и валидационный наборы).
* Отбор наиболее информативных признаков.
* Применение разных техник борьбы с несбалансированностью классов.
* Использование как классических методов машинного обучения, так и недавно разработанных для работы с табличными данными методов глубокого обучения, а также выбор наилучшей модели путем сравнения полученных результатов.

# Обзор литературы

Для решения данной задачи применяются в том числе методы, уже зарекомендовавшие себя в аналогичных работах. Чаще всего в подобных задачах используют деревья решений [8, 9, 10], ансамбли деревьев решений [8, 9, 11, 12, 13], логистическую регрессию [8, 9, 10, 11, 12] и алгоритмы глубокого обучения [8, 10, 12]. Модели машинного обучения превосходят традиционные статистические методы [8]. Случайный лес деревьев решений и алгоритм опорных векторов обычно показывают более высокую результативность, чем логистическая регрессия, деревья решений и нейронные сети в аналогичных задачах. Для того чтобы нейронная сеть давала сравнимый со случайным лесом результат, необходимы большие наборы данных для обучения [8]. Одним из основных преимуществ случайного леса является способность обрабатывать большие входные данные, нелинейные зависимости и избегать переобучения [11]. Поскольку логистическая регрессия является линейным классификатором, она показывает лучшую способность при обработке линейных зависимостей [11]. Таким образом, ансамбли деревьев оказываются наиболее эффективным методом решения задач, аналогичных рассматриваемой в данной работе [8, 9, 11, 12].

Из-за большой размерности медицинских данных иногда применяют отбор наиболее информативных признаков. Обычно в числе наиболее важных факторов содержатся такие признаки как возраст, класс по классификации Killip, показатели креатинина, систолическое давление, индекс массы тела [8, 9]. Для выбора наиболее информативных признаков применяют в том числе отбор на основе случайного леса [13], который далее будет использоваться в данной работе.

Поскольку в медицинских данных очень часто возникает проблема несбалансированности классов, иногда применяются такие методы как undersample [13]. В данной работе он также будет применяться как один из способов.

Для ознакомления с основными понятиями машинного обучения, типами задач, метрическими и линейными алгоритмами классификации, полносвязными нейронными сетями использовалась книга [3].

При изучении различных методов машинного обучения использовались в основном оригинальные статьи [1, 2, 6, 7]. Источники [4, 5] применялись для ознакомления с методами борьбы с проблемой несбалансированности классов. Также использовалась документация к используемым в данной работе программным пакетам языка Python:

* [14] – для разработки классификаторов,
* [15] – для работы с несбалансированными выборками,
* [16] - для реализации полносвязных нейронных сетей,
* [17] - для работы с методом NODE [7],
* [18] - для разработки метода SuperTML [6],
* [19] - для работы с изображениями при реализации метода SuperTML.

# Используемые методы

## Линейные модели

В случае бинарной классификации в линейных моделях прогноз получают при помощи формулы

где n – размерность признакового пространства, – значение j-го признака для объекта x из выборки, – вес j-го признака – параметр модели, который оценивается в ходе обучения, , , …, ) – вектор весов. Если добавить признак, который на каждом объекте принимает значение 1, то из предыдущей формулы можно убрать неоднородность, и она запишется следующим образом:

где – скалярное произведение вектора параметров на признаковое описание объекта. Прогнозирование того или иного класса происходит в зависимости от положительности или отрицательности значения функции

Таким образом, линейный бинарный классификатор разделяет два класса при помощи гиперплоскости.

### Логистическая регрессия

К алгоритмам линейной классификации относится логистическая регрессия [3]. Задача обучения линейного классификатора заключается в том, чтобы по выборке настроить вектор весов w. В случае логистической регрессии для нахождения весов решается задача [минимизации эмпирического риска](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%8D%D0%BC%D0%BF%D0%B8%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D1%80%D0%B8%D1%81%D0%BA%D0%B0) (средней величины ошибки алгоритма на обучающей выборке) с функцией ошибки вида

где m – количество объектов в выборке, вектор значений признаков i-го объекта выборки, – значение целевого признака на этом объекте.

Когда решение w найдено, можно не только вычислять значение  для произвольного объекта x, но и оценивать апостериорные вероятности его принадлежности к каждому из классов при помощи формулы

где  — сигмоидная функция, x – объект из выборки, w – найденный вектор весов. Таким образом, для каждого объекта x можно получить две вероятности: и – вероятности того, что объект x принадлежит классу 0 или 1 соответственно.

## K ближайших соседей

Метрические алгоритмы — это методы, которые предполагают, что в пространстве признаков введено понятие метрики. Алгоритм k ближайших соседей [3] относит новую точку к какому-то классу, руководствуясь тем, как объект расположен относительно уже известных из обучающей выборки точек. Самый простой алгоритм смотрит, какая точка из обучающей выборки ближе, и относит новую точку к тому же классу. Этот метод называется алгоритмом ближайшего соседа.

Метод k ближайших соседей смотрит на k ближайших точек, затем определяет, какой класс среди них доминирует, и относит новую точку к этому классу.

В этот алгоритм иногда добавляют веса объектов. Веса могут зависеть от номера соседа, а могут зависеть от расстояния до соседа. Чтобы отнести новую точку к тому или иному классу алгоритм считает сумму весов для одного класса, сумму весов для другого, после чего принимает решение на основе того, какая из сумм получилась больше.

## Деревья решений

Решающие деревья [20] представляют собой логические схемы, позволяющие получить окончательное решение о классификации объекта с помощью ответов на иерархически организованную систему вопросов, которая приводит к истинному ответу максимально коротким путём. Чтобы построить дерево, алгоритм перебирает все возможные тесты и находит тот, который является наиболее информативным с точки зрения прогнозирования значений целевой переменной. Рекурсивное разбиение данных повторяется до тех пор, пока все точки данных в каждой области разбиения не будут принадлежать одному и тому же классу целевой переменной.

В основе алгоритмов построения дерева решений лежит принцип жадной максимизации прироста информации, то есть на каждом шаге выбирается тот признак, при разделении по которому происходит больший прирост информации. Для разбиения вершин дерева используются условия вида

,

где – j-ый признак объекта, а t – некоторый порог. Если неравенство выполняется, то объект отправляется в одну сторону, если нет – в другую.

Рассмотрим произвольную вершину дерева g. Путь там находится набор объектов . Обозначим критерий ошибки за , тогда параметры j и t выбираются путём перебора с целью минимизации этой ошибки:

Обозначим выборки, которые на данном шаге m отправляются в левое и правое поддерево за и соответственно, тогда критерий ошибки определяется следующим образом:

где H измеряет, насколько сильный разброс ответов получается при попадании в соответствующее поддерево. Чем меньше разброс ответов, тем меньше значение H.

Введём вспомогательную величину , которая показывает, какая доля объектов класса k находится в произвольной выборке X:

где – значение целевого признака на i-м объекте, k принимает значения из множества классов (в случае бинарной классификации классов всего два – 0 и 1), а функция принимает значение 1, если условие в квадратных скобках выполняется, и 0 – в противном случае.

Для задач классификации имеют место два различных критерия информативности, которые можно задавать через параметр criterion при использовании метода решающих деревьев.

Пусть K – максимальное значение целевого признака (в нашем случае 1). Первый критерий – критерий Джини, где функция H вычисляется по формуле

где определяются формулами выше, X – произвольная выборка объектов. В частности, из формулы понятно, что если , то H = 0, то есть если в выборке присутствуют элементы только одного из классов, то функция H достигает своего минимального значения.

Второй критерий – энтропийный, где H определяется по формуле

при этом считается, что 0ln0 = 0.

Как правило, такое построение дерева приводит к получению моделей, которые являются очень сложными и переобученными. Один из способов исправления данной проблемы, который далее будет использоваться в работе – остановка процесса построения дерева на определенной глубине. Это приводит к более низкой правильности на обучающем наборе, но улучшает правильность на тестовом наборе. При использовании остановки класс для новой точки данных, оказавшейся после разбиений в вершине g, определяется по формуле:

где – значение целевого признака на i-м объекте из выборки , Y – множество всех возможных значений целевого признака, y принимает значения из множества Y, – функция, результат которой – аргумент, при котором достигается наибольшее значение функции, то есть это такой y, при котором сумма из формулы принимает наибольшее значение,или, простыми словами, тот класс, который наиболее популярен в данной выборке.

## Ансамбли деревьев решений

### Случайный лес

Основная идея случайного леса [1] заключается в использовании большого ансамбля [решающих деревьев](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9), каждое из которых само по себе даёт не очень высокое качество классификации, но за счёт их большого количества результат получается хорошим. Случайный лес решает проблему склонности деревьев решений к переобучению.

Для построения модели случайного леса необходимо определиться с количеством деревьев, которые будут строиться независимо друг от друга. При построении каждого из деревьев сначала генерируется случайная выборка с повторениями такого же размера, что и исходный набор данных. Таким образом, некоторые образцы попадут в неё два или более раза. Данный процесс называется бэггинг. Кроме того, алгоритм случайным образом отбирает признаки для построения каждого дерева, чтобы получить ещё более непохожие друг на друга деревья. За количество отбираемых признаков отвечает задаваемый параметр. Таким образом, если в случае решающих деревьев при минимизации критерия ошибки номер признака j выбирался из всего множества признаков, то в случайных лесах он берётся из случайного подмножества заданного размера. В случае задач классификации за размер данного подмножества обычно берут величину

,

где n – исходное количество признаков.

Алгоритм относит классифицируемый объект к тому классу, к которому его отнесла большая часть входящих в композицию деревьев.

### Градиентный бустинг деревьев

Бустинг представляет собой построение ансамбля, при котором входящие в него деревья построены не независимо, как, например, в случайном лесе, а последовательно [2]. Каждое дерево пытается исправить ошибки предыдущего. Здесь в основном используются деревья небольшой глубины. Это делает модель эффективнее с точки зрения памяти и скорости вычисления прогнозов.

Обозначим за L(y,z) функцию потерь, которую необходимо минимизировать, где y – истинный ответ, а z – результат работы алгоритма, то есть прогноз.

Композицию N базовых алгоритмов (в данном случае деревьев) обозначим

где N – количество алгоритмов в композиции, – алгоритмы, входящие в композицию, – результат работы (прогноз) j-го алгоритма.

Для начала построим первый базовый алгоритм . В качестве такого алгоритма можно, например, взять самый популярный класс на обучающем наборе данных:

где – значение целевого признака на i-м объекте из обучающего набора, Y – множество всех возможных значений целевого признака, y принимает значения из множества Y, m – количество объектов в выборке.

Пусть уже построили N-1 базовых алгоритмов. Теперь необходимо выбрать следующий алгоритм так, чтобы минимизировать ошибку композиции L на обучающем наборе:

где – вектор признаков i-го объекта из обучающего набора, – значение целевого признака на этом объекта, m – количество объектов в обучающем наборе.

Для начала решим более простую задачу: попытаемся понять, какие конкретные значения должен принимать новый алгоритм на объектах обучающей выборки чтобы функция потерь принимала наименьшее значение. Таким образом, предыдущая формула перепишется в виде

где – вектор сдвигов, который необходимо найти. Очевидно, что в качестве вектора s нужно брать антиградиент, так как он направлен в сторону наискорейшего убывания функции:

По этой формуле можно найти все элементы вектора s, поэтому теперь известно, какие значения должен принимать новый алгоритм на объектах набора, чтобы минимизировать ошибку. Тогда обучение нового алгоритма представляет собой задачу обучения на размеченных данных, в которой – обучающая выборка и используется, например, квадратичная функция ошибки:

где s = ( – найденный выше вектор сдвигов, – функция, результат которой – аргумент, при котором достигается наименьшее значение функции, то есть это такое b, при котором сумма из формулы принимает наименьшее значение. Так градиентный бустинг последовательно строит композицию.

Чтобы обеспечить аккуратное движение к минимуму функции ошибки, вводят такую величину как размер шага, то есть при добавлении найденного алгоритма происходит лишь небольшое смещение в сторону этого вектора:

*,*

где λ – длина шага. Понятно, что чем меньше размер шага, тем больше времени занимает процесс, но тем лучшего качества можно достичь.

В задаче бинарной классификации с классами {+1, -1} в качестве функции потерь используется логистическая функция потерь вида

где a(x) – оценка принадлежности положительному классу: при положительном значении классификатор относит объект к классу +1, в противном случае – к классу -1.

## Полносвязные нейронные сети прямого распространения

Один нейрон определяется выражением

где σ — функция активации, n – количество признаков, w = ( — вектор весов, x = ( – вектор значений признаков объекта x из выборки. Функция активации должна быть непрерывной, монотонной и, желательно, дифференцируемой функцией. То есть один нейрон является моделью логистической регрессии, описанной выше, если в качестве функции активации выбрана сигмоидная функция активации.

Имеется много различных функций активации. На последнем выходном слое в данной работе будет использоваться именно сигмоидная функция активации, поскольку множество ее значений находится в промежутке от 0 до 1. В этом случае при результате выше 0,5 объект относится к классу 1, а в противном случае — к классу 0. На всех остальных слоях будет использоваться функция активации ReLU, которая определяется следующей формулой:

где

В большинстве случаев на практике применяются многослойные нейронные сети. Многослойными называются [нейронные сети](https://wiki.loginom.ru/articles/neural-network.html), в которых [нейроны](https://wiki.loginom.ru/articles/artificial-neuron.html) сгруппированы в слои. При этом каждый нейрон предыдущего слоя связан со всеми нейронами следующего слоя, а внутри слоёв связи между нейронами отсутствуют. Для обучения многослойных нейронных сетей используется [обучение с учителем](https://wiki.loginom.ru/articles/supervised-learning.html). Наиболее популярным алгоритмом обучения для них является алгоритм [обратного распространения ошибки](https://wiki.loginom.ru/articles/back-propagation-algorithm.html).

**Алгоритм обратного распространения ошибки:**

Для каждого нейрона сети и одного объекта выборки x вычисляется:

1. , где w – вектор весов на данном слое
2. Значение на выходе: , где – функция активации на данном слое
3. Производная функции ошибки Q

Далее можно вычислить градиент функции ошибки Q по вектору весов w, используя формулы

При помощи этих градиентов на каждом шаге обучения нейронной сети происходит оптимизация весов методом градиентного спуска.

Для того чтобы задать нейронную сеть, требуется указать число слоев, число нейронов в каждом слое, тип функции активации в каждом слое и вид функции ошибки. Также желательно, чтобы подготовленная выборка не содержала пропусков, а признаки были нормированы.

## SuperTML

Идея метода SuperTML, описанного в [6], состоит в представлении каждого объекта табличных данных изображением (как показано на рисунке 1 [6]), которое затем передается на вход предобученным моделям сверточных нейронных сетей для классификации. Процесс использования предобученных моделей также известен как трансферное обучение [21].

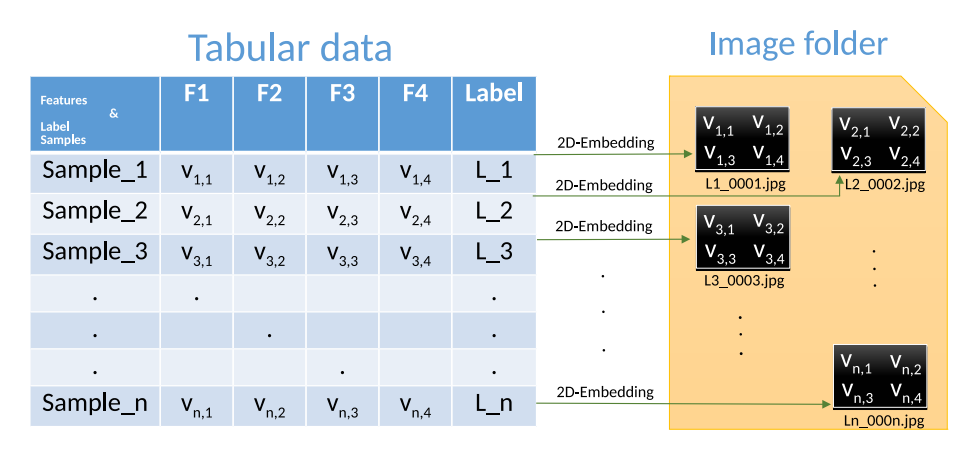


Рисунок 1 – пример представления объекта табличных данных рисунком

**Алгоритм:** для каждого объекта данных рассматривается набор значений признаков, каждое из которых изображается на картинке так, чтобы значения всех признаков занимали как можно больше места на изображении, но при этом не перекрывались. Все значения изображаются одинаковым шрифтом. Затем тренировочный набор данных подается на вход предобученным моделям (в данной работе использовалась модель ResNet18 [22]) для классификации по классам.

Авторы статьи экспериментально показали, что этот метод хорошо работает как с большими, так и с маленькими наборами данных, что является большим преимуществом применительно к имеющемуся набору данных.

На рисунке 2 [6] показано сравнение точностей, полученных градиентным бустингом деревьев решний и методом SuperTML на трех табличных наборах данных. Видно, что SuperTML показал себя лучше в сравнении с xgboost.

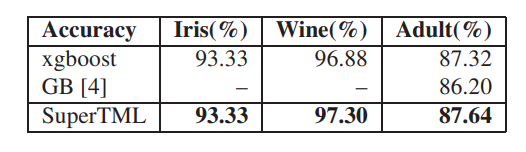


Рисунок 2 – сравнение градиентного бустинга и метода SuperTML

## NODE Нейронные ансамбли небрежных решающих деревьев

Чтобы показать идею метода, описанного в [7], необходимо ввести некоторые понятия. Небрежные решающие деревья – это обычные деревья решений глубины d, в которых на всех узлах одной глубины используется один и тот же признак для разделения, а также одинаковый порог. Из-за этих ограничений небрежные деревья значительно слабее обучаются в сравнении с деревьями решений без ограничений, но они менее подвержены переобучению.

Существенным недостатком обычных деревьев решений является то, что они не допускают оптимизации и используют жадные процедуры для построения дерева. Эта проблема иногда решается «смягчением» функции принятия решений, чтобы сделать ее дифференцируемой. Такие деревья называются дифференцируемыми. В данной работе для этой цели была использована функция entmax [23]. Entmax способна создавать разреженные распределения вероятностей, где большинство из вероятностей точно равны 0.

Основной блок предложенной в статье модели – NODE слой. Он состоит из m дифференцируемых небрежных решающих деревьев одинаковой глубины d.

По своей сути небрежное решающее дерево – это разбиение по d признакам. При прогнозировании дерево сравнивает каждый признак объекта с порогом. Затем дерево возвращает один из 2d возможных вариантов в зависимости от результата сравнения. Таким образом, каждое небрежное решающее дерево может быть представлено в виде

где – функция Хевисайда, R – d-мерная матрица ответов размерности .

Чтобы сделать выход дерева дифференцируемым, функции заменяются взвешенной суммой признаков с весами, вычисляемыми при помощи функции entmax, применяемой к обучаемой матрице выбора признаков :

Кроме того, функция Хевисайда заменяется двухклассной функцией entmax, которая обозначается следующим образом:

Поскольку разные признаки могут иметь разные масштабы, используется также масштабированная версия:

где - обучаемые параметры, отвечающие за порог и масштаб соответственно.

Основываясь на значениях определяется матрица выбора размерность которой равна размерности матрицы ответов R, по следующей формуле:

.

Таким образом, матрица выбора C содержит вероятности каждого из 2d возможных вариантов. Финальное предсказание вычисляется как взвешенная линейная комбинация матрицы ответов R с весами из элементов матрицы выбора C:

При этом в случае недифференциального дерева entmax всегда возвращает ненулевой вес только для одного из признаков, а всегда возвращает только нули или единицы.

Таким образом, одно дерево в NODE слое выглядит так, как показано на рисунке 3 [7].

В случае классификации дерево выдает , где |C| - количество классов. Выход NODE слоя состоит из конкатенации выходных данных m деревьев:

Слой NODE, описанный выше, можно обучать отдельно или в составе сложной структуры, например, полностью связанных слоев, которые могут быть организованы в многоуровневые архитектуры.

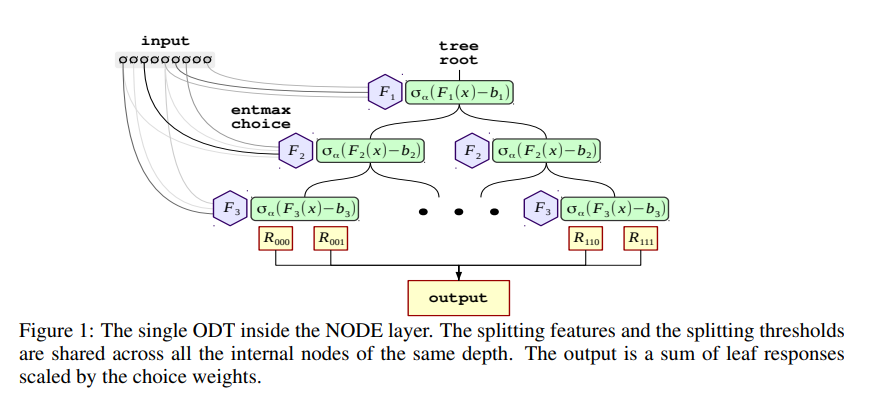


Рисунок 3 – структура одного дерева в NODE слое

Авторы статьи используют архитектуру, представляющую собой последовательность из k NODE слоев, где каждый уровень использует конкатенацию всех предыдущих слоев в качестве входных данных. На первый уровень подаются значения признаков объекта x.

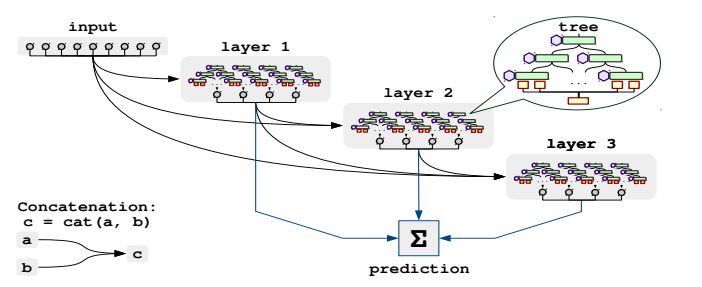


Рисунок 4 – архитектура из нескольких NODE слоев

Результирующий прогноз представляет собой среднее всех деревьев решений со всех уровней. При этом в многоуровневой архитектуре, описанной выше, выходные значения дерева из ранних слоев используются в качестве входных данных для последующих. Поэтому авторы не ограничивают размерность количеством классов, а вводят дополнительную произвольную размерность , что соответствует (d+1)-мерной матрице ответов R: . При усреднении прогнозов по всем слоям только первые |C| координаты используются для задач классификации, и только первая - для задач регрессии. Таким образом, - дополнительный гиперпараметр. Многослойная архитектура представлена на рисунке 4 [7].

## Методы борьбы с несбалансированностью классов

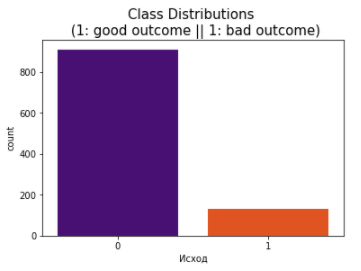


Рисунок 5 – распределение классов в исходном наборе данных

На рисунке 5 показана диаграмма распределения классов в имеющемся наборе данных. Видно, что классы сильно не сбалансированы, поэтому помимо тренировки на исходном наборе данных вышеописанные методы машинного обучения применялись на других наборах данных, созданных при помощи двух методов борьбы с несбалансированностью классов.

### SMOTE

Метод SMOTE [4] основан на идее генерации искусственных примеров, которые были бы похожи на имеющиеся в миноритарном классе, но при этом не совпадали с ними. Алгоритм выбирает примеры, которые находятся близко в пространстве признаков, рисует линию между этими объектами и создает новый образец данных в точке вдоль этой линии. Для этого сначала выбирается случайный пример из класса меньшинства, затем для этого примера находятся k ближайших соседей (обычно k = 5), после чего выбирается случайным образом сосед , соединяются и , чтобы сформировать линейный сегмент в пространстве признаков, и создается синтетический пример в случайно выбранной точке между двумя примерами.

Данный подход является эффективным, потому что создаются новые синтетические примеры из класса меньшинства, которые являются правдоподобными, то есть они относительно близки в пространстве признаков к существующим в миноритарном классе объектам.

### Undersampling

Суть метода Undersampling [5] заключается в удалении некоторых элементов из набора данных, чтобы классы стали более сбалансированными. В данной работе удалялось такое количество примеров мажоритарного класса, чтобы соотношение классов стало 1 к 1.

## Отбор наиболее информативных признаков

Методы, позволяющие определять, насколько полезен каждый признак, дают возможность понижать размерность признакового описания объектов и, как следствие, получать более простые модели с лучшей обобщающей способностью.

Существуют множество методов, решающих данную задачу. В данной работе использовался отбор на основе модели. В качестве самой модели использовался случайный лес деревьев решений.

### Отбор на основе модели случайного леса

Для начала рассмотрим, как такую задачу можно решать при помощи деревьев решений. При построении деревьев на каждом этапе вершина разбивается на две. Для этого выбирается признак, по которому происходит разбиение, и порог, с которым сравнивается данный признак. Как уже указывалось ранее, выбор признака и порога осуществляется жадным образом с целью минимизации функции ошибки на каждом шаге.

Пусть вершина разбивается по признаку j, тогда чем сильнее уменьшилось значение критерия информативности H(X), тем лучше признак. Таким образом, можно оценивать информативность признака на основе того, как сильно он уменьшает значение H.

Рассмотрим вершину g, которая разбивает выборку объектов , содержащуюся в данной вершине, признаком j на две выборки: и Тогда уменьшение dec критерия информативности в ней будет определяться по формуле

Далее можно просуммировать эту величину по всем вершинам, где разбиение происходило по j-му признаку. Очевидно, что чем больше эта сумма, тем важнее признак. Обозначим эту сумму за и рассмотрим далее отбор признаков, где в качестве модели используется случайный лес деревьев. В данном случае важность признака можно определять через сумму величин по всем деревьям, входящим в лес. Чем больше эта сумма, тем более информативен данный признак.

# Вычислительная часть

## Используемый язык программирования

Программа была написана на языке Python, поскольку там имеется библиотека pandas [24], где реализован большой набор инструментов для обработки и анализа данных, а также имеются библиотеки scikit-learn [14], keras [16] и torch [18], где реализованы различные алгоритмы машинного обучения, а также приведена подробная документация.

## Предобработка данных

Первым делом данные считываются при помощи библиотеки pandas и перемешиваются, затем из всех признаков выбираются только те 49, которые были указаны врачами как наиболее сильно влияющие на целевой признак, и разделяются на категориальные, которые впоследствии разделяются на бинарные и остальные, и вещественные, поскольку они требуют разной предобработки. После вывода описания данных видно, что и в категориальных, и в бинарных, и в вещественных данных имеются пропуски. Чтобы их заполнить, данные сначала разделяются на два набора: где целевой признак принимает значение 1 и 0. Заполнение пропусков реализуется при помощи функции SimpleImputer из библиотеки scikit-learn [25]. За стратегию заполнения отвечает такой параметр этой функции как strategy. Пропуски в категориальных и бинарных данных заполняются наиболее часто встречающимся значением в классе, что соответствует значению “most\_frequent” параметра strategy, а в вещественных данных – средним значением каждого конкретного признака по классу – значение “mean” этого параметра.

После заполнения пропущенных значений необходимо избавиться от категориальных данных, поскольку некоторые методы машинного обучения с ними не работают. Для этого применяется унитарное кодирование, где из одного категориального признака образуется несколько бинарных, отвечающих за каждое отдельное значение из возможных для данного признака. Эта процедура осуществлялась при помощи функции OneHotEncoder из библиотеки scikit-learn [26]. При использовании этой функции названия исходных признаков теперь имеют другой вид, по которому сложно понять, за что конкретно отвечал ранее этот признак. Например, первый из образованных бинарных признаков имеет теперь название ‘x0\_1.0’. Первая цифра отвечает за порядковый номер преобразуемого категориального признака, а вторая – за порядковый номер уникального значения. Чтобы по названию признаков далее можно было понимать, о каком конкретно свойстве идет речь, названия новых бинарных признаков менялись так: каждое название состоит теперь из исходного названия категориального признака и порядкового номера уникального значения, например: “Килип1”.

Вещественные данные необходимо стандартизировать. Для этого необходимо вычислить вспомогательные величины: средние значения признаков и стандартные отклонения, которые соответственно определяются следующими формулами:

*,*

где m – число объектов в обучающей выборке, – значение признака j на i-м объекте выборки. Стандартизация вещественных данных осуществлялась при помощи метода StandardScaler из библиотеки scikit-learn [27].

Далее все три полученных предобработанных набора данных, разделенных изначально по типу признаков, соединяются в один при помощи библиотеки pandas, и на этом препроцессинг данных заканчивается. В итоговом наборе данных 104 признака без учета целевого – увеличение за счет создания большого количества бинарных признаков из категориальных.

## Распределение классов

После получения готового набора данных выводилась диаграмма распределения классов, которая представлена на рисунке 6. В наборе наблюдается сильная несбалансированность классов: 12,4% и 87,6% для класса 1 и 0 соответственно. Поэтому было решено попробовать разные техники борьбы с несбалансированностью классов, так что далее методы машинного обучения будут последовательно применяться и сравниваться на трех различных наборах: первый из них – набор, для которого применялась техника undersample, второй – к которому применялся метод SMOTE и третий – это исходный набор с несбалансированными данными. Для того чтобы в самом конце сравнить разные техники борьбы с несбалансированностью данных, из набора была извлечена небольшая выборка val из 40 экземпляров, в которой 20 из них относятся к классу 1, а 20 – к классу 0. Для сравнения различных алгоритмов машинного обучения каждый из трех наборов разделялся на тренировочный, на котором происходит обучение алгоритмов, и тестовый, на котором считается точность алгоритмов и сравниваются различные методы машинного обучения.

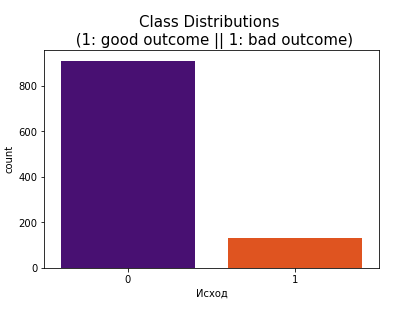


Рисунок 6 - распределение классов в исходном наборе данных

## Применение методов машинного обучения

Сначала из 104 имеющихся признаков из набора данных undersample, в котором соотношение классов 1:1, (диаграмма распределения классов этого набора представлена на рисунке 7) отбирались 16 наиболее информативных при помощи метода ExtraTreesClassifier из библиотеки scikit-learn [28, 29], в котором реализован отбор признаков на основе случайного леса деревьев решений, описанного в разделе “Используемые методы”. Отобранные признаки, которые представлены на рисунке 8, соответствуют результатам, полученным в [8], где среди восьми наиболее информативных признаков тоже есть такие как возраст, класс по Killip (все 4 класса имеются среди отобранных 16 признаков), показатель креатинина, систолическое давление (соответствует признаку “Рсрлао(Адср)1”).

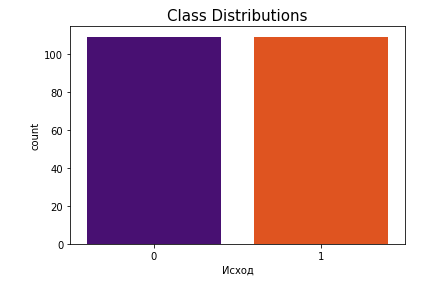


Рисунок 7 - распределение классов в наборе undersample

Если посмотреть на матрицу корреляции признаков, которая представлена на рисунке 9, то видно, что отобранные признаки действительно коррелируют с целевым. Для обучения использовался далее набор с этими 16 признаками. После этого последовательно применялись логистическая регрессия, метод k ближайших соседей, деревья решений, случайный лес деревьев решений, полносвязная нейронная сеть прямого распространения, градиентный бустинг деревьев решений, SuperTML и NODE. Подбор гиперпараметров происходил следующим образом: составлялся массив из нескольких параметров, для каждого из них происходила тренировка и считалась точность, затем выводился график, основанный на результатах, и по нему выбирался параметр, который даст более хороший результат.

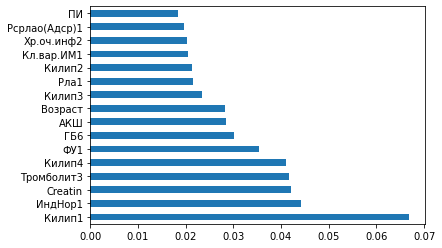


Рисунок 8 - отобранные наиболее информативные признаки

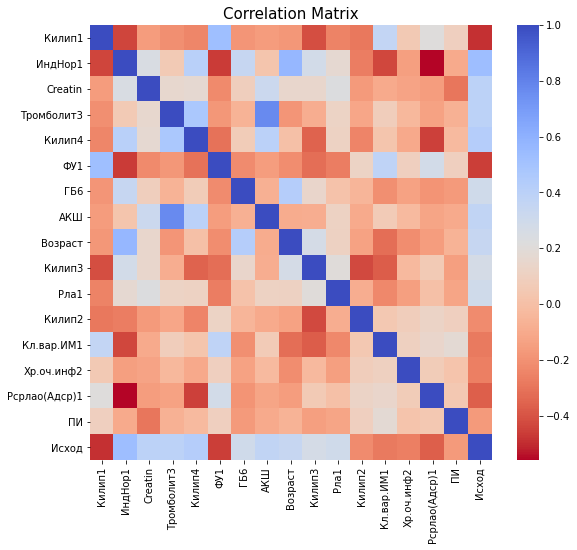


Рисунок 9 - матрица корреляции признаков

В качестве примера на рисунке 10 приведен график подбора гиперпараметра, отвечающего за количество деревьев в случайном лесе. Видно, что оптимальное число деревьев – 14.

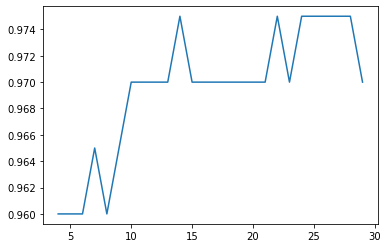


Рисунок 10 - пример подбора гиперпараметров

После обучения логистической регрессии на основании полученных коэффициентов выводилось полученное уравнение.

Для наглядного понимания работы метода деревьев решений выводилась схема полученного дерева.

Выбранная архитектура полносвязной нейронной сети показана на рисунке 11. Первый слой – входной – это значения 16 признаков, которые подаются на вход нейронной сети. На первом скрытом слое 10 нейронов, на втором и третьем по три нейрона, на четвертом 1 нейрон, значение которого – результат работы нейронной сети. Было сделано 100 шагов оптимизации, в качестве оптимизатора был выбран adam [30].

При использовании метода SuperTML в данной работе использовались два различных возможных положения значений признаков объекта на изображении, которые представлены на рисунке 12. Вещественные значения округлялись с точностью до двух знаков, чтобы 16 значений хорошо помещались на изображении.

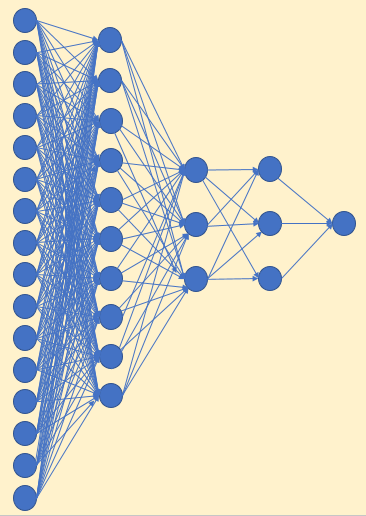


Рисунок 11 - архитектура используемой полносвязной нейронной сети

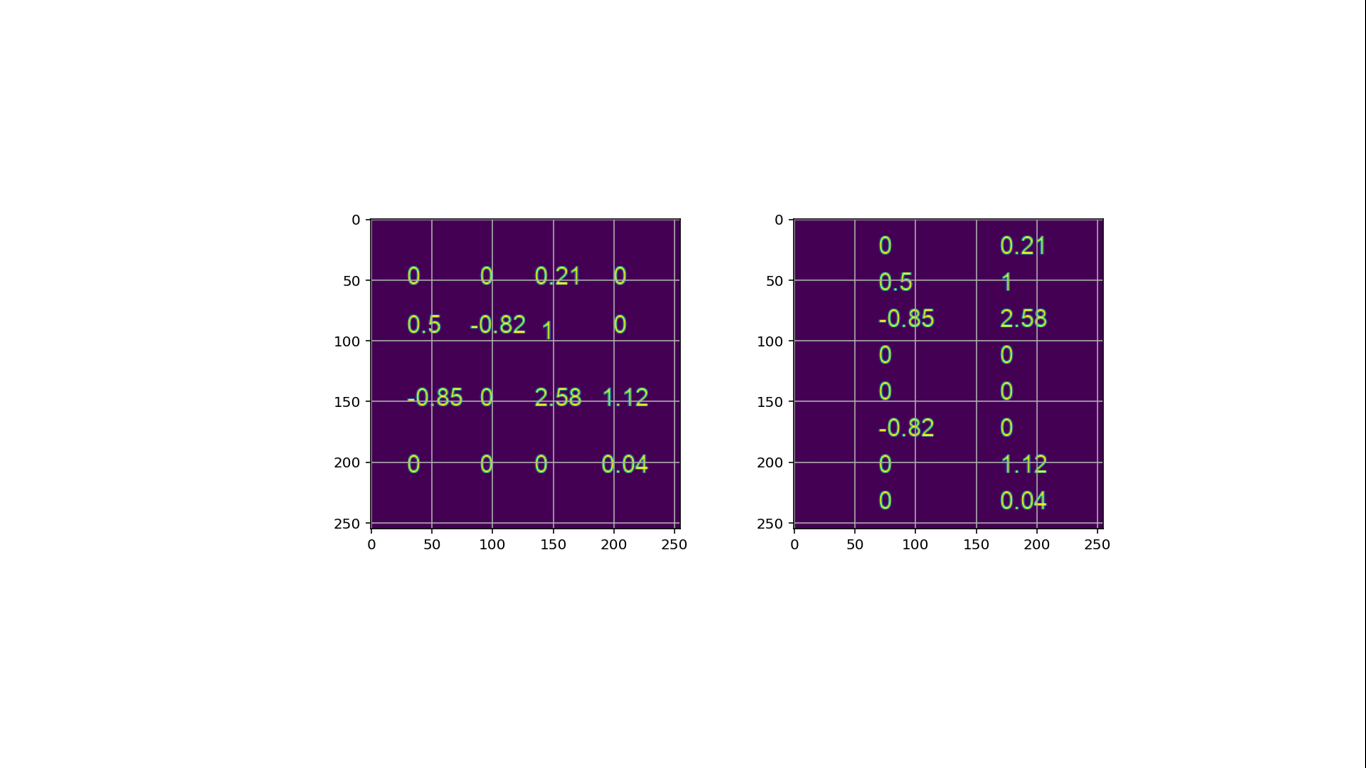


Рисунок 12 – используемые положения значений на изображении

При использовании метода NODE в данной работе использовались и сравнивались две различные архитектуры: первая из них содержит два NODE слоя, на каждом из которых по 1024 небрежных дифференциальных деревьев, а вторая из архитектур содержит три NODE слоя с тем же числом деревьев на каждом из них.

При сравнении различных алгоритмов машинного обучения использовались две метрики: accuracy и f-мера [31]. Метрика accuracy присваивает всем объектам одинаковый вес, что может быть некорректно в случае, если распределение объектов в обучающей выборке, как и в нашем случае, сильно смещено в сторону одного из классов. В этом случае у классификатора есть больше информации по этому классу, и, соответственно, он будет принимать более правильные решения на объектах данного класса. Таким образом, в этом случае алгоритм имеет хорошую точность, но на миноритарном классе классификатор работает плохо. В таких случаях целесообразно посмотреть на другие метрики, такие как, например, f-мера, которая представляет собой [гармоническое среднее](http://bazhenov.me/blog/2012/05/05/harmonic-mean.html) между точностью и полнотой:

где precision – точность и recall – полнота определяются следующими формулами:

где TP – количество истинно положительных, FP – ложноположительных, а FN – ложноотрицательных прогнозов. То есть точность показывает, какую часть объектов, которые классификатор распознал как объекты положительного класса, он предсказал верно, полнота показывает, какую долю объектов, на самом деле относящихся к положительному классу, классификатор предсказал правильно, а f-мера учитывает обе эти метрики.

Для обучения логистической регрессии, деревьев решений, метода k ближайших соседей и случайного леса деревьев решений использовались методы LogisticRegression, DecisionTreeClassifier, KNeighborsClassifier и RandomForestClassifier из библиотеки scikit-learn соответственно, а для реализации градиентного бустинга деревьев решений применялась библиотека XGBoost [32].

Полносвязная нейронная сеть реализовывалась при помощи библиотеки keras [16]. Для обучения алгоритма SuperTML использовались библиотеки PyTorch [18] и Pillow [19]. Для того чтобы рисовать значения признаков на изображении, сначала значения преобразовывались в строковый формат данных, после чего использовалась функция ImageDraw.Draw.text из библиотеки Pillow.

Для реализации алгоритма NODE использовался pytorch-tabular [17] – фреймворк, выпущенный в начале 2021 года, предназначенный для использования различных нейросетевых архитектур на табличных данных.

Диаграммы, изображенная на рисунке 13, отражают результаты работы различных алгоритмов, обученных на наборе данных undersample, полученные на тестовом наборе данных. В таблице 1 также представлены значения метрик для валидационного набора данных.

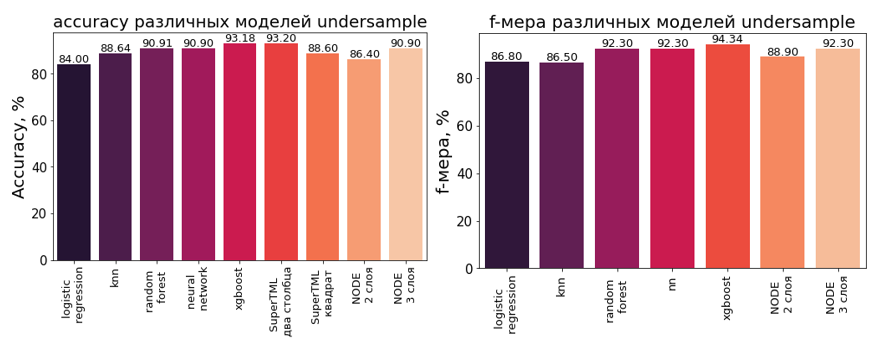


Рисунок 13 – результаты для набора данных undersample

Уравнение логистической регрессии получилось следующим:

где x[j] – значение признака j на объекте x из выборки.

Полученная схема дерева решений глубины 3 представлена на рисунке 14.

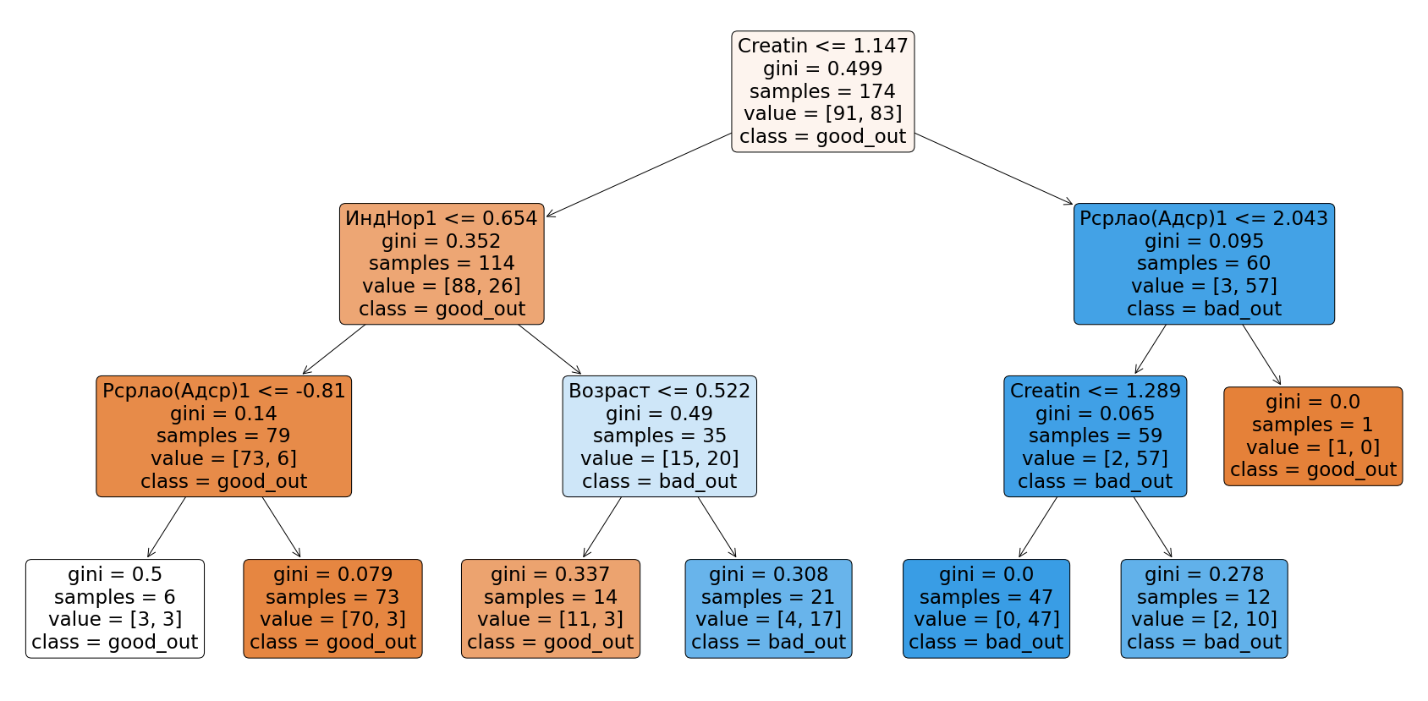


Рисунок 14 - схема дерева, полученного для набора undersample

Таблица 1 - результаты для набора данных undersample

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Тестовый набор | | Валидационный набор | |
|  | Точность | f-мера | Точность | f-мера |
| Логистическая регрессия | 84 | 86,8 | 85 | 85,7 |
| K ближайших соседей | 88,64 | 86,5 | 80 | 80,95 |
| Случайный лес деревьев решений | 90,91 | 92,3 | 87,5 | 87,8 |
| Полносвязная нейронная сеть | 90,9 | 92,3 | 85 | 85,7 |
| Градиентный бустинг деревьев решений | 93,18 | 94,34 | 90 | 90 |
| SuperTML 1 вариант (два столбца) | 93,2 |  | 90 | 90 |
| SuperTML 2 вариант  (квадратная матрица) | 88,6 |  | 87,5 | 88,4 |
| NODE 2 слоя | 86,4 | 88,9 | 85 | 85,7 |
| NODE 3 слоя | 90,9 | 92,3 | 89,5 | 90 |

Затем набор данных oversample, в котором соотношение классов тоже 1:1 было достигнуто при помощи метода SMOTE, также разделялся на тренировочный и тестовый наборы. На тренировочном наборе обучались последовательно те же самые алгоритмы машинного обучения. На рисунке 15 представлены значения метрик эффективности, полученные на тестовом наборе, а таблица 2 содержит, кроме того, результаты работы алгоритмов на валидационной выборке.

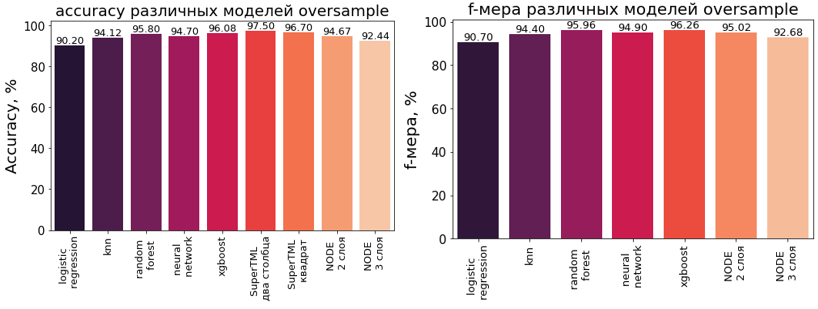


Рисунок 15 - результаты для набора данных oversample

Полученное уравнение логистической регрессии:

Схема дерева решений представлена на рисунке 16.

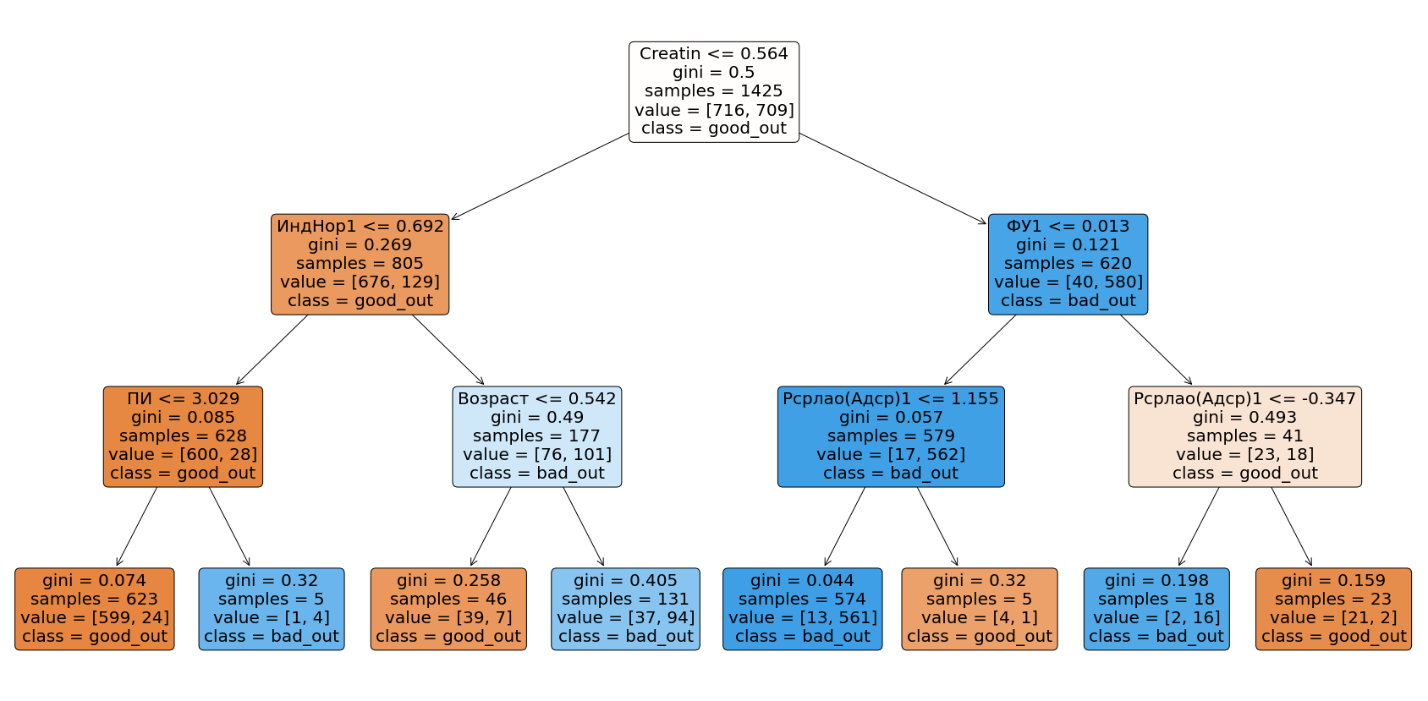


Рисунок 16 - схема дерева, полученного для набора ovesample

Таблица 2 - результаты для набора данных oversample

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Тестовый набор | | Валидационный набор | |
|  | Точность | f-мера | Точность | f-мера |
| Логистическая регрессия | 90,2 | 90,7 | 85 | 85,7 |
| K ближайших соседей | 94,12 | 94,4 | 85 | 84,2 |
| Случайный лес деревьев решений | 95,8 | 95,96 | 90 | 89,5 |
| Полносвязная нейронная сеть | 94,7 | 94,9 | 90 | 89,47 |
| Градиентный бустинг деревьев решений | 96,08 | 96,26 | 92,5 | 92,31 |
| SuperTML 1 вариант  (два столбца) | 97,5 |  | 95 | 95 |
| SuperTML 2 вариант  (квадратная матрица) | 96,7 |  | 90 | 90 |
| NODE 2 слоя | 94,67 | 95,02 | 94,73 | 95 |
| NODE 3 слоя | 92,44 | 92,68 | 94,8 | 94,7 |

Аналогичные шаги применялись к исходному набору данных с несбалансированными классами. Полученные результаты отражены на рисунке 17 и в таблице 3.

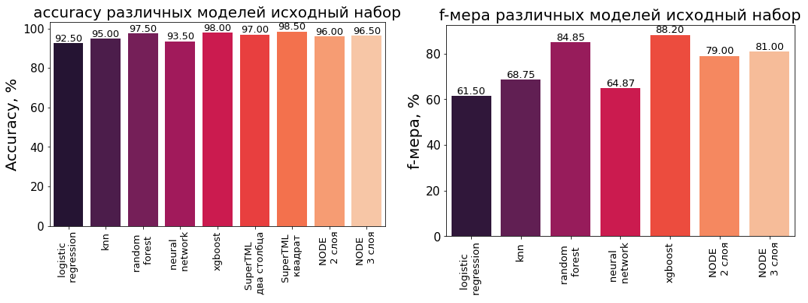


Рисунок 17 - результаты для исходного набора данных

Уравнение логистической регрессии получилось следующим:

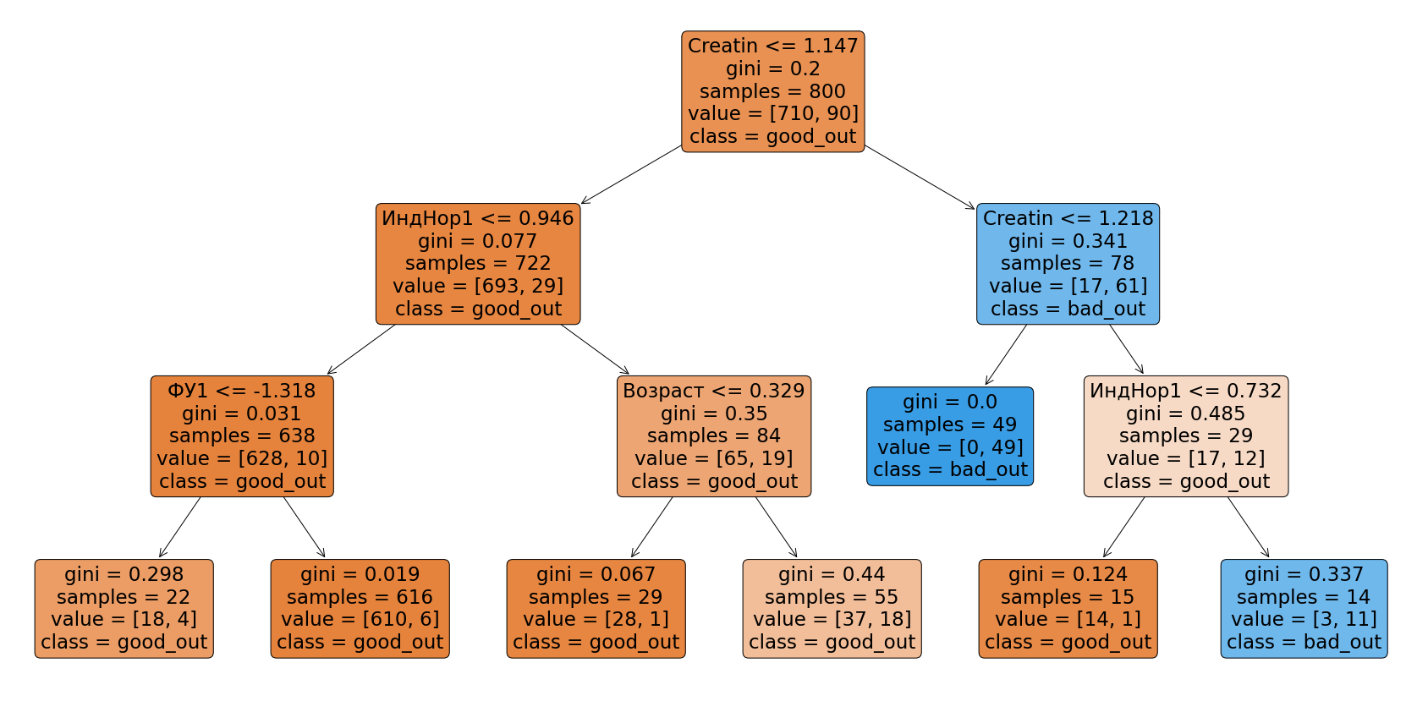


Рисунок 18 - схема дерева, полученного для исходного набора

Таблица 3 - результаты для исходного набора данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Тестовый набор | | Валидационный набор | |
|  | Точность | f-мера | Точность | f-мера |
| Логистическая регрессия | 92,5 | 61,5 | 82,5 | 78,79 |
| K ближайших соседей | 95 | 68,75 | 85 | 82,35 |
| Случайный лес деревьев решений | 97,5 | 84, 85 | 87,5 | 86,48 |
| Полносвязная нейронная сеть | 93,5 | 64,87 | 85 | 84,2 |
| Градиентный бустинг деревьев решений | 98 | 88,2 | 90 | 88,9 |
| SuperTML  (два столбца) | 97 |  | 92,5 | 92,3 |
| SuperTML  (квадратная матрица) | 98,5 |  | 92,5 | 91,9 |
| NODE 2 слоя | 96 | 79 | 86,84 | 85,7 |
| NODE 3 слоя | 96,5 | 81 | 84,2 | 82,35 |

# Анализ результатов

Из приведенных выше диаграмм (рисунки 13, 15, 17) и таблиц (1-3) видно, что использование методов борьбы с несбалансированностью классов помогло улучшить результаты работы алгоритмов. Видно, что метод SMOTE по всем значениям показал себя гораздо лучше, чем метод undersample.

В некоторых случаях точность алгоритмов на исходном наборе данных была выше, чем на других, но это связано лишь с тем, что классы сильно не сбалансированы: даже классификатор, который всегда будет предсказывать только один класс, может получить точность 87,6% (поскольку именно столько от всего исходного набора данных составляет класс 0). Это подтверждается очень низкими значениями f-меры. С этим связаны и более плохие результаты работы алгоритмов, обученных на исходном наборе данных, при прогнозировании на валидационной выборке: там соотношение классов 1:1, а алгоритмы плохо предсказывают ответы для миноритарного класса.

Если посмотреть на разницу между точностью и f-мерой на исходном наборе данных, можно сделать вывод, что ансамбли деревьев решений менее чувствительны к несбалансированным выборкам.

На всех наборах данных градиентный бустинг продемонстрировал более высокое качество в сравнении с логистической регрессией, методом k ближайших соседей, случайным лесом деревьев решений и используемой архитектурой полносвязной нейронной сети.

Лучшее качество показал метод SuperTML, который превзошел градиентный бустинг деревьев решений на всех трех выборках. На наборе undersample расположение значений признаков на изображении в виде матрицы оказалось более эффективным, чем другой вариант расстановки, но на двух других наборах данных расположение в два столбца оказалось гораздо более результативным.

Трехслойная архитектура NODE превзошла двуслойную на наборе oversample и исходном наборе данных, а на undersample они показали практически идентичную результативность. Алгоритм NODE не показал более хороший результат в сравнении с градиентным бустингом деревьев решений и методом SuperTML, однако он оказался более результативным, чем полносвязная нейронная сеть.

# Выводы

* Была произведена предобработка данных. Категориальные признаки заполнялись наиболее часто встречающимся значением в классе, после чего для них производилось унитарное кодирование. Вещественные факторы заполнялись средним значением по классу, а потом стандартизировались.
* Признаки, полученные в результате отбора наиболее информативных факторов, в основном соответствуют литературным данным.
* Внедрение методов борьбы с несбалансированностью классов помогло улучшить результаты работы алгоритмов машинного обучения. Метод SMOTE по всем показателям оказался значительно лучше, чем метод undersample.
* Среди примененных методов машинного обучения наилучшим на всех выборках оказался метод SuperTML, использующий трансферное обучение. Он превзошел градиентный бустинг, зарекомендовавший себя в решении задач на табличных данных.

# Интерфейс

Поскольку модель SuperTML, обученная на наборе данных oversample, показала себя наилучшим образом среди всех остальных моделей, то для прогнозирования новых данных будет использоваться далее именно эта натренированная модель. Кроме того, сохранялся натренированный StandardScaler, чтобы новые данные были стандартизированы точно так, как данные, на которых обучалась модель. Затем создавалось веб-приложение, которое может использовать сохраненную модель, чтобы генерировать прогнозы по новым данным в режиме реального времени. Для создания веб-интерфейса применялся фреймворк Flask. Есть две части этого приложения: front-end, который разрабатывается с использованием HTML и css, и back-end, который создается с использованием Flask в Python.



Рисунок 19 - часть реализованного веб-приложения

Подразумевается, что врач вводит данные пациента в 16 полей, которые соответствуют 16 отобранным наиболее информативным признакам. Эти 16 значений сохраняются в список, преобразуются сохраненной моделью стандартизации, затем эти значения накладываются на изображение аналогичным образом, как это происходило при обучении модели, и отправляются на вход сохраненной модели SuperTML, которая делает предсказание. Таким образом, после заполнения всех необходимых полей, пользователь нажимает кнопку “Подтвердить” и на открывшейся странице появляется результат работы программы: “благоприятный исход” или “неблагоприятный исход”.  Вид реализованного интерфейса представлен на рисунке 19.

# Заключение

В данной работе произведен сравнительный анализ методов машинного обучения для задачи прогнозирования исхода после инфаркта миокарда. Были использованы как классические методы, такие как, например, логистическая регрессия и градиентный бустинг деревьев решений, так и недавние разработки, связанные с применением глубокого обучения для прогнозирования на табличных данных. Кроме того, решалась проблема несбалансированности классов путем применения методов undersample и SMOTE.

Результатом данной работы является программа [33] с реализованным интерфейсом, способная помогать врачам в предсказании исхода для пациентов, перенесших инфаркт миокарда.

# Список литературы

[1] Breiman L. Random forests // Mach. Learn. 2001. Т. 45. № 1. С. 5–32.

[2] Mason L. и др. Boosting Algorithms as Gradient Descent // 2000.

[3] Воронцов К.В. Математические методы обучения по прецедентам (теория обучения машин).

[4] Chawla N. V и др. SMOTE: Synthetic Minority Over-sampling Technique. , 2002. 321–357 с.

[5] Кузьмишкина А. и др. Выбор объектов для обучения в условиях сильной несбалансированности классов.

[6] Sun B. и др. SuperTML: Two-Dimensional Word Embedding for the Precognition on Structured Tabular Data. , 2019.

[7] Popov S., Morozov S., Babenko A. Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data // 2019.

[8] Lee H. C. и др. Prediction of 1-Year Mortality from Acute Myocardial Infarction Using Machine Learning // Am. J. Cardiol. 2020. Т. 133. С. 23–31.

[9] Shouval R. и др. Machine learning for prediction of 30-day mortality after ST elevation myocardial infraction: An Acute Coronary Syndrome Israeli Survey data mining study // Int. J. Cardiol. 2017. Т. 246. С. 7–13.

[10] Piros P. и др. Comparing machine learning and regression models for mortality prediction based on the Hungarian Myocardial Infarction Registry // Knowledge-Based Syst. 2019. Т. 179. С. 1–7.

[11] Mansoor H. и др. Risk prediction model for in-hospital mortality in women with ST-elevation myocardial infarction: A machine learning approach // Hear. Lung J. Acute Crit. Care. 2017. Т. 46. № 6. С. 405–411.

[12] Bansal A. и др. MACHINE LEARNING TECHNIQUES TO PREDICT IN-HOSPITAL CARDIOVASCULAR OUTCOMES IN ELDERLY PATIENTS PRESENTING WITH ACUTE MYOCARDIAL INFARCTION // J. Am. Coll. Cardiol. 2020. Т. 75. № 11. С. 3603.

[13] Ibrahim K. S. и др. Preliminary Study on Application of Machine Learning Method in Predicting Survival Versus Non-Survival after Myocardial Infarction in Malaysian Population // Int. J. Cardiol. 2018. Т. 273. С. 8.

[14] Documentation scikit-learn: machine learning in Python — scikit-learn 0.21.3 documentation [Электронный ресурс]. URL: https://scikit-learn.org/0.21/documentation.html (дата обращения: 26.09.2020).

[15] imbalanced-learn documentation — Version 0.8.0 [Электронный ресурс]. URL: https://imbalanced-learn.org/stable/ (дата обращения: 13.01.2021).

[16] Keras: the Python deep learning API [Электронный ресурс]. URL: https://keras.io/ (дата обращения: 12.12.2020).

[17] pytorch-tabular · PyPI [Электронный ресурс]. URL: https://pypi.org/project/pytorch-tabular/ (дата обращения: 10.05.2021).

[18] PyTorch documentation — PyTorch 1.8.1 documentation [Электронный ресурс]. URL: https://pytorch.org/docs/stable/index.html (дата обращения: 12.03.2021).

[19] Pillow — Pillow (PIL Fork) 8.2.0 documentation [Электронный ресурс]. URL: https://pillow.readthedocs.io/en/stable/ (дата обращения: 29.04.2021).

[20] Quinlan J. R. Simplifying decision trees // Int. J. Man. Mach. Stud. 1987. Т. 27. № 3. С. 221–234.

[21] Pan S. J., Yang Q. A survey on transfer learning // IEEE Trans. Knowl. Data Eng. 2010. Т. 22. № 10. С. 1345–1359.

[22] He K. и др. Deep Residual Learning for Image Recognition.

[23] Peters B., Niculae V., Martins A. F. T. Sparse Sequence-to-Sequence Models. : Association for Computational Linguistics. 1504–1519 с.

[24] pandas documentation — pandas 1.2.4 documentation [Электронный ресурс]. URL: https://pandas.pydata.org/docs/ (дата обращения: 01.11.2020).

[25] sklearn.impute.SimpleImputer — scikit-learn 0.24.2 documentation [Электронный ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html (дата обращения: 24.03.2021).

[26] sklearn.preprocessing.OneHotEncoder — scikit-learn 0.24.2 documentation [Электронный ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html (дата обращения: 13.01.2021).

[27] sklearn.preprocessing.StandardScaler — scikit-learn 0.24.2 documentation [Электронный ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (дата обращения: 22.02.2021).

[28] sklearn.ensemble.ExtraTreesClassifier — scikit-learn 0.24.2 documentation [Электронный ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html#sklearn.ensemble.ExtraTreesClassifier (дата обращения: 22.02.2021).

[29] Geurts P., Ernst D., Wehenkel L. Extremely randomized trees // Mach. Learn. 2006. Т. 63. № 1. С. 3–42.

[30] Kingma D. P., Ba J. L. Adam: A method for stochastic optimization // 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings.

[31] F-Score Definition | DeepAI [Электронный ресурс]. URL: https://deepai.org/machine-learning-glossary-and-terms/f-score (дата обращения: 13.01.2021).

[32] XGBoost Documentation — xgboost 1.5.0-SNAPSHOT documentation [Электронный ресурс]. URL: https://xgboost.readthedocs.io/en/latest/ (дата обращения: 22.02.2021).

[33] ovopilova/Predicting-outcome-after-myocardial-infarction-using-machine-learning [Электронный ресурс]. URL: https://github.com/ovopilova/Predicting-outcome-after-myocardial-infarction-using-machine-learning.