**СОДЕРЖАНИЕ**

[СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОПРЕДЕЛЕНИЙ 7](#_Toc159170518)

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 8](#_Toc159170519)

[ВВЕДЕНИЕ 9](#_Toc159170520)

[1 ТЕОРЕТИЧЕСКИЙ ОБЗОР 11](#_Toc159170521)

[1.1 Алгоритмы на табличных данных 11](#_Toc159170522)

[1.1.1 Классические алгоритмы 11](#_Toc159170523)

[1.1.2 Алгоритмы глубинного обучения 14](#_Toc159170524)

[1.1.3 Итог анализа алгоритмов 16](#_Toc159170525)

[1.2 Существующие инструменты для анализа таблиц 17](#_Toc159170526)

[1.2.1 RATH: Autopilot for exploratory data analysis 17](#_Toc159170527)

[1.2.2 ChatGPT for Excel 17](#_Toc159170528)

[1.2.3 Pandas GPT / LangChain / ChatGPT Advanced Data Analysis 17](#_Toc159170529)

[1.2.4 Итог анализа инструментов 18](#_Toc159170530)

[2 ПРОЕКТИРОВАНИЕ 20](#_Toc159170531)

[3 РЕАЛИЗАЦИЯ 21](#_Toc159170532)

[ЗАКЛЮЧЕНИЕ 22](#_Toc159170533)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc159170534)

[ПРИЛОЖЕНИЯ 24](#_Toc159170535)

# СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОПРЕДЕЛЕНИЙ

ИИ – искусственный интеллект

Нейросеть – нейронная сеть

GBDT – градиентные бустинги на решающих деревьях

CNN – свёрточная нейронная сеть

MLP – многослойный перцептрон

1D – одномерный

NODE – нейронный забывающий дифференцируемый ансамбль

API – прикладной программный интерфейс

# ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Нейронная сеть – упрощенная модель биологической нейронной сети, представляющая собой совокупность искусственных нейронов, взаимодействующих между собой

Датасет – набор данных

Табличные данные – множество гетерогенных векторов одного типа, хранящегося в структурированном виде

Таблица – множество гетерогенных векторов одного типа, хранящегося в структурированном виде

Промпт – текст на естественном языке, подающийся на вход нейросети

# ВВЕДЕНИЕ

Табличные данные и временные ряды входят в число самых практичных и часто используемых видов данных, применяющихся в деятельности человека. Широкий спектр организаций от крупных машиностроительных заводов до больниц и школ используют их для решения своих задач. Важно, что большинство связанных с такими данными процессов, таких как составление расписания, учёт и планирование ресурсов, хранение исторических данных, мониторинг состояния систем, являются ключевыми в рамках своих организаций. Без преуменьшения можно утверждать, что табличные данные и временные ряды играют ключевую роль в современной экономике [1].

Широкое распространение табличного формата в работе разнообразных систем приводит к тому, что часто с таблицами приходится работать специалистам, не имеющим профессиональных компетенций в области анализа данных и статистики. Такое положение неизбежно приводит к неэффективным затратам рабочего времени, лишней когнитивной нагрузке на сотрудников и ошибкам в принятии важных решений на основании данных. Сегодня это становится особенно критично, когда данные и культура работы с ними являются залогом экономического развития большинства компаний.

В то же время внимание специалистов машинного обучения на фоне всплеска популярности генеративных моделей всё больше привлекают обработка естественных языков и изображений. Ситуация ведёт к тому, что область табличных данных несмотря на свою практическую важность теряет темпы развития. Стоит упомянуть, что на сегодняшний день не существует нейросетевой архитектуры, которая демонстрирует уверенное превосходство над классическими алгоритмами на табличных данных [2]. Библиотеки же классических алгоритмов поддерживаются всё неохотнее, а созданные для нейронных сетей инфраструктурные возможности, которые могли бы быть полезны и для классических решений, в эти библиотеки не внедряются.

Таким образом, накапливается разрыв между качеством продуктов в области анализа языка и инструментов для анализа табличных данных. Функционал современных табличных движков гораздо менее автоматизирован, чем даже сложные инструменты генерации кода или системы распознавания образов для роботов. Множество основанных на больших языковых моделях решений, недавно появившихся в области анализа таблиц, частично закрывает упомянутый разрыв, но не может претендовать на полное решение проблемы, так как языковые модели не созданы для работы с таблицами, они могут лишь дать подсказки по правильному формированию запроса на соответствующем языке. Эти решения скорее показывают потребность общества в решении выявленной проблемы, чем полностью решают её.

Целью данной работы является сокращение ошибок и временных затрат при работе специалистов различных организаций с табличными данными через создание приложения, решающего задачи пользователя, оформленных в виде текстовых запросов (промптов). Будет рассмотрено несколько примеров подобных задач и результат применения на них разработанного решения.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Провести исследование текущего состояния предметной области и протестировать наиболее подходящие существующие алгоритмы работы с табличными данными и временными рядами,
2. Поиск данных для проведения экспериментов и оценки качества алгоритмов,
3. Проектирование системы для анализа табличных данных на основе выбранных алгоритмов,
4. Реализация системы в коде,
5. Проверка результатов работы системы на основе выбранных данных и наборов задач.

# 1 ТЕОРЕТИЧЕСКИЙ ОБЗОР

В данной главе приведён анализ текущего состояния предметной области алгоритмов для табличных данных и временных рядов и инструментов на их основе. Исследованы различие классические и нейросетевые подходы к задачам регрессии и классификации на табличных данных, методы получения скрытых состояний и поиска по таблицам, подходы к решению поставленных задач в существующих инструментах. На основании анализа были выбраны наиболее перспективные для дальнейшего исследования подходы.

## Алгоритмы на табличных данных

Для получения способного решать задачи по таблицам и рядам инструмента, необходимо понимать не только архитектуру решающих похожие задачи инструментов, но и базовые алгоритмы. Это позволит решить, какие алгоритмы для каких задач применять, оценить способность каждого из них работать с выбранным классом данных.

### 1.1.1 Классические алгоритмы

Данный класс алгоритмов очень богат, так как развивался очень долго и включает в себя хорошо проверенные временем решения. Самыми универсальными классическими алгоритмами для поставленной задачи являются деревья и ансамбли на их основе, в особенности – градиентные бустинги.

Принцип работы решающих деревьев заключается в нахождении кусочно-постоянной аппроксимации исходной функции через итеративное построение древовидной структуры. Каждый узел этой структуры – точка разделения пространства признаков. Поданная на вход алгоритму точка пространства признаков проверяется на принадлежность одной из непересекающихся областей и спускается по дереву по соответствующему этой области ребру. Листья такого дерева – промежутки постоянства аппроксимирующей функции, им в соответствие поставлены значения.

Алгоритм решающих деревьев хорошо адаптирован к гетерогенным данным, так как предикат разбиения пространства признаков и критерий выбора признака для разбиения в каждом узле могут быть адаптированы под любой тип колонки: числовой, категориальный, порядковый. Кроме того, к плюсам этого алгоритм относятся широкие возможности для настройки: можно разными способами регулировать глубину дерева и критерии разбиения в узлах. Тем не менее данный подход имеет очевидные недостатки: склонность к переобучению и неустойчивость предсказаний. Модель машинного обучения должна уметь выдавать точные результаты на данных, которые при обучении не видела, то есть обладать обобщающей способностью. Для получения подобного свойства у решающего дерева надо тщательно подбирать его параметры: построение слишком глубокого дерева ведёт к простому запоминанию всех данных из обучающей выборки, а построение слишком короткого дерева вызывает недообучение. Баланс между слишком глубоким переобученным деревом и слишком коротким недообученным найти удаётся не для всех видов данных в силу сложной формы целевой функции. Из этого следует, что сами по себе решающие деревья являются интересными с точки зрения работы с таблицами моделями, но имеют значительные недостатки, которые не позволяют применять их в чистом виде. Чтобы обойти эти недостатки, деревья объединяют в ансамбли.

Ансамбли моделей – модели машинного обучения, которые агрегирует результаты своих базовых алгоритмов для получения предсказания. В данной работе из этого класса моделей рассмотрены случайный лес и градиентные бустинги, как наиболее популярные.

Алгоритм случайного леса строит множество решающих деревьев и усредняет их данные для получения предсказания. За счёт неглубоких базовых деревьев уменьшается вероятность переобучения ансамбля, а их большое количество уменьшает неустойчивость ансамбля в целом.

Принцип алгоритма градиентного бустинга – создание линейной комбинации базовых алгоритмов. В алгоритме происходит итеративное уменьшение значения функции ошибок ансамбля через добавление к нему предсказания нового базового алгоритма, обученного предсказывать ошибку предыдущего ансамбля:

|  |  |  |
| --- | --- | --- |
|  | *,* | (1) |

где – алгоритм бустинга, а – базовый алгоритм, добавленный на итерации .

На шаге к ансамблю добавляется базовый алгоритм , обученный предсказывать антиградиент функции потерь от результатов .

Алгоритм бустингов позволяет последующим алгоритмам компенсировать ошибки предыдущих и значительно повышать точность итоговых предсказаний. Если в качестве базового алгоритма используется дерево, то получается градиентный бустинг на решающих деревьях (GBDT) – на данный момент один из самых точных алгоритмов для таблиц [3].

Несмотря на свою точность, градиентные бустинги могут переобучаться. Эффективное средство борьбы с проблемой – разнообразные регуляризации. Регуляризация – добавление к функции потерь функции, которая задаёт некоторые ограничения для весов модели. Самые популярные регуляризации L1 и L2, например, построены таким образом, чтобы ограничить рост весов модели во избежания переобучения весов и возникновения «градиентного взрыва». Регуляризация помогает бустингам не просто выучивать тренировочные данные, а обретать обобщающую способность.

Подводя промежуточные итоги, ключевым преимуществом классических моделей машинного обучения является их заточенность под гетерогенные данные. Ансамблевые алгоритмы на базе деревьев устойчиво опережают по качеству все иные известные алгоритмы. Важной особенностью деревьев и их ансамблей является интерпретируемость и простота. На руку классическим методам играет и частота их применения на практике, огромное количество связанных с ними библиотек и исследований.

### 1.1.2 Алгоритмы глубинного обучения

Нейронные сети на табличных данных – развивающийся раздел исследований глубинного обучения. В работе были рассмотрены архитектуры ансамбля дифференцируемых деревьев, многослойных перцептронов, свёрточных сетей и трансформеров.

Ансамбли дифференцируемых деревья – архитектура, использующая метод обратного распространения ошибки для обучения ансамблей деревьев. Для применения метода обратного распространения ошибки необходима дифференцируемость, которой решающее дерево не обладает. Как было упомянуто ранее, деревья строят кусочно-постоянную аппроксимацию, то есть производная деревьев может быть определена только на участках постоянства. Для решения этой проблемы в узлах дерева оператор выбора фичи для разбиения и предикат сравнения заменяются своими непрерывными аналогами. В случае архитектуры NODE (нейронный дифференцируемый забывающий ансамбль) дифференцируемость узлов дерева обеспечивается преобразованием . На некоторых датасетах архитектура NODE опережает бустинги, на других – нет.

Многослойный перцептрон (MLP) – многослойная (слоёв не менее 3) полносвязная нейронная сеть с нелинейными активациями. Данный тип нейронных сетей способен работать с нелинейными целевыми функциями, хоть и является простейшем видом нейронных сетей. К минусам MLP относится неумение извлекать признаки, являющиеся произведениями исходных. Данная проблема частично решается созданием синтетических признаков-произведений исходных. Простота модели делает её хорошим кандидатом для анализа заведомо несложных функций, но в общем случае MLP довольно ограничен. Большей описательной способностью обладают модификации MLP, такие как GrowNet (градиентный бустинг над MLP), DCN V2 (модель, добавляющая фичи-произведения в обучение).

Свёрточные нейронные сети слабо адаптированы под табличные данные, так как ключевое свойство CNN – извлечение признаков из пространственных отношений, которых в таблицах нет. Существуют адаптации архитектуры ResNet и метод 1D CNN, но исследования показали, что точность их ниже точности классических подходов на большинстве наборах данных.

Трансформерные архитектуры, как и CNN, в чистом виде плохо адаптированы к гетерогенным данным, которые трудно представить в виде последовательностей гомогенных (а именно с такими данными работают трансформеры). Поэтому общий принцип работы большинства трансформеров на табличных данных базируется на создании скрытых представлений фичей, с которыми удобно работать трансформеру. Эти скрытые представления представляются в виде последовательности, трансформер же должен предсказать последний токен последовательности. Этот токен – скрытое представление ответа, которое на последних слоях превращается в значение целевой переменной. Один из примеров такой архитектуры – FTTransformer. В исследованиях данная модель показала неплохие результаты, на некоторых датасетах оказалась лучше бустингов, однако устойчивого опережения не продемонстрировала. Полезная свойство трансформеров – механизм внимания над признаками. Оно позволяет выделять самые важные фичи. Интересны трансформеры и с точки зрения генерации скрытых представлений: их можно использовать в системах семантического поиска данных по таблице.

Лучшая по качеству найденная в рамках исследования архитектура – гибридный метод TabR [4]. TabR совмещает в себе идеи алгоритма K-ближайших соседей и многослойных перцептронов. Относится к классу retrieval-augmented алгоритмов, то есть делает предсказания не только на основании фичей поданной на вход точки пространства признаков, но и с использованием ранее увиденных моделью данных, сохранённых в некотором аналоге памяти и полученном оттуда наборе наиболее релевантных точек. По результатам исследований показал себя лучшей среди всех рассмотренных архитектурой глубинного обучения на табличных данных. Более того, TabR продемонстрировал стабильное превосходство над классическими алгоритмами. Из этого факта следует, что retrieval-augmented подходы стоят рассмотрения, как очень перспективный класс алгоритмов. Однако, TabR не лишён и недостатков: операция поиска похожих объектов в базе увеличивает время работы на порядки по сравнению с бустингами.

В итоге нейронные сети отлично показывают себя на текстовых данных и изображениях, однако пока ещё не демонстрируют таких же широких возможностей на табличных данных. Дело в фундаментальных основах большинства сегодняшних нейросетевых архитектур, заточенных под обработку гомогенных данных, такие как пиксели на изображениях или токены в текстах. В данный момент ведутся активные исследования возможностей переноса способностей глубоких моделей на гетерогенные данные. Самыми перспективными себя показали подходы токенизации признаков, механизмы внимания и подходы, дополненные поиском по уже собранной коллекции данных. Также видно, что лучшими алгоритмами являются гибридные, такие как бустинги над MLP или же совмещение архитектуры MLP с поиском ближайших соседей.

### 1.1.3 Итог анализа алгоритмов

Вывод: среди всех алгоритмов не найдено такого, который бы побеждал все остальные на всех наборах данных. Выбор алгоритма сильно зависит от конкретных данных, универсального решения на данный момент не существует. Однако в большинстве случаев лучше всего работают ансамбли (особенно бустинги) и гибриды. Самыми перспективными для дальнейшего исследования подходами себя показали кодирование табличных признаков в вектора скрытых представлений, применение механизма внимания к признакам, использование retrieval-augmented подходов, создание эффективных методов регуляризации и создание гибридных алгоритмов, совмещающих глубинные и классические подходы машинного обучения,

## Существующие инструменты для анализа таблиц

На рынке уже существует линейка продуктов, решающих задачи, подобные поставленной.

### 1.2.1 RATH: Autopilot for exploratory data analysis

Данный инструмент является большим комбайном разнообразных полезных инструментов для анализа данных. На автоматической основе, используя разнообразные статистики и большие языковые модели, он генерирует наиболее удобные визуализации данных, а также предлагает режим CoPilot для введения запросов к таблицам. Данное решение рассчитано скорее для профессионалов в области анализа данных, а не как универсальная API для полной автоматизации работы с данными.

### 1.2.2 ChatGPT for Excel

ChatGPT for Excel – это плагин для табличного редактора Excel, основная задача которого – помогать пользователю писать формулы. Инструмент действительно упрощает работу с таблицами, но является лишь посредником между пользователем и табличным движком. Плагин не знает контекста данных, не понимает задачи пользователя. Работа происходит не в виде «естественный язык – таблица – естественный язык», а в виде «естественный язык – язык Excel – таблица».

### 1.2.3 Pandas GPT / LangChain / ChatGPT Advanced Data Analysis

На рынке существует пласт похожих по принципу работы решений, такие как библиотеки Pandas GPT и LangChain, сервис ChatGPT в режиме Advanced Data Analysis, интересных в рамках данной работы с точки зрения своего интерфейса. Эти инструменты на вход получают текст на естественном языке, а на выходе отдают также текст на естественном языке, возможно, с добавлением графиков. Такой интерфейс является наиболее похожим на требуемый в поставленной задаче. Результаты работы данных решений действительно впечатляют. К примеру, с помощью библиотеки Pandas GPT можно задать вопрос о Pandas-датасете на естественном языке вида «Сколько дней в году был дождь?» и получить правильный ответ. Принцип работы данных инструмент состоит в преобразовании запроса с естественного языка на язык запросов соответствующего хранилища данных с помощью большой языковой модели (в случае Pandas GPT, например, запрос пользователя с помощью API OpenAI преобразуется в код на языке Python, осуществляющий запрос в Pandas-датасет), выполнения этого запроса и преобразование его результатов в текст опять же с помощью большой языковой модели. Недостатки таких решений – ограниченность понимания контекста данных. В данный момент нельзя попросить модель найти наиболее похожий элемент для новой точки пространства признаков или дать его текстовое описание.

### 1.2.4 Итог анализа инструментов

Все рассмотренные решения с продуктовой точки зрения делятся на 2 группы: плагины к существующим средствам анализа данных и независимые приложения. И та, и другая группа обладает своими преимуществами. Плагины легче в использовании и имеют гораздо больший охват аудитории, так как пишутся для популярных аналитических движков, как Excel или Tableau. Независимые же приложения более гибки, позволяют отображать больше информации и визуализации и предоставлять более удобные формы интерфейса, как, например, вопросно-ответная система по таблице в рамках ChatGPT.

В архитектуре большинства подобных систем используются 2 базовые идеи:

1. Генерация кода или SQL-запросов для обращения к хранилищу данных,
2. Использование статистики и агрегатов, упрощение и сжатие данных.

Каждый из перечисленных подходов имеет свои недостатки.

Генерация кода влечёт создание лишней прослойки между системой и таблицей, которая понижает качество работы качество, так как операция поиска данных в такой схеме не имеет семантических свойств, данные рассматриваются в отрыве от своего контекста, как набор чисел. Такой подход скорее удобен, как инструмент ускорения разработки программного кода, чем как универсальный инструмент анализа табличных данных.

Использование статистики и агрегатов сильно ограничивает функционал приложения заранее заданными разработчиками статистиками и алгоритмами, которые работают с этими статистиками. К примеру, если разработчик делает приложение, дающее информацию на основании данных о среднем по каждому столбику, то пользователь лишён аналитических инструментов для работы с медианами. Прописать все необходимые пользователям сценарии не представляется возможным. Кроме того, агрегаты и статистики вызывают сильное сжатие реальных данных. Это уменьшает точность и не даёт возможности работать, например, с выбросами, которые могут представлять для аналитика сильный интерес.

В итоге анализ существующих решений позволил понять основные существующие подходы к поставленной задаче. На основании анализа подходов делается вывод о том, что наиболее удобным для пользователей является система плагинов, но такая система сильно ограничена в возможностях базовой платформой, а таже технически реализуется сложнее. Сделан вывод о том, что систему необходимо проектировать, как отдельное приложение, но учитывать подходы существующих решений для обеспечения удобства пользования.

Также анализ показал, что на данный момент решения поставленной проблемы существуют, но не обладают достаточным уровнем качества. Внимание специалистов в данной области смещено к использованию больших языковых моделей, которые не являются подходящим выбором для табличных данных, так как слабо учитывают контекст данных, их семантические свойства, рассматривая таблицу, как набор чисел в хранилище, к которому можно делать запросы.

# 2 ПРОЕКТИРОВАНИЕ

# 3 РЕАЛИЗАЦИЯ

# ЗАКЛЮЧЕНИЕ

В рамках данной работы были достигнуты следующие результаты:

1. Изучена предметная область алгоритмов анализа табличных данных и существующих инструментов для конченых пользователей, упрощающих этот анализ. Было проведено исследование наиболее эффективных из имеющихся алгоритмов, на основании которого был выбран набор подходов для дальнейшего исследования,
2. Проведён поиск необходимых существующих и разметка собственных датасетов для сравнения и тестирования выбранных моделей,
3. На основании выбранных моделей созданы и исследованы различные алгоритмы интерполяции, экстраполяции и создания скрытого представления табличных данных и временных рядов, проведены тесты и выбраны наиболее подходящие,
4. РеализованоAPI, решающее произвольные задачи пользователя по произвольным табличным данным,
5. На основании API создан набор приложений для решения конкретных задач, исследована работоспособность и практическая применимость полученной системы,
6. Проведено тестирование на заранее заготовленных данных, проверено качество ответов системы,
7. Проведено тестирование системы на пользователях со сравнительными замерами скорости работы, получена обратная связь от специалистов и оценка экономии времени от использования системы,
8. Код приложения и результаты исследования опубликованы в открытом доступе.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шпилькина Т.А., Ляшкова О.В. Роль Big Data в деятельности корпораций // Экономика и бизнес: теория и практика. 2020. №4-3.
2. Borisov V., Leemann T., Seßler K., Haug J., Pawelczyk M., Kasneci G. Deep Neural Networks and Tabular Data: A Survey //arXiv preprint arXiv :2110.01889v3. – 2022.
3. Shwartz-Ziv R., Armon A. TABULAR DATA: DEEP LEARNING IS NOT ALL YOU NEED //arXiv preprint arXiv:2106.03253v2. – 2021.
4. Gorishniy Y., Rubachev I., Kartashev N., Shlenskii D., Kotelnikov A., Babenko A. TabR: Tabular Deep Learning Meets Nearest Neighbors in 2023 //arXiv preprint arXiv:2307.14338. – 2023.

# ПРИЛОЖЕНИЯ