

Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет социально-экономических и компьютерных наук

Чепоков Елизар Сергеевич

**РАЗРАБОТКА MVP СИСТЕМЫ МАШИННОГО ОБУЧЕНИЯ ДЛЯ
ПРОГНОЗИРОВАНИЯ РЕНТАБЕЛЬНОСТИ КИНОБИЗНЕСА**

Выпускная квалификационная работа

студента образовательной программы «Программная инженерия»
по направлению подготовки 09.03.04 Программная инженерия

Рецензент

Руководитель

Профессор кафедры
информационных
технологий в бизнесе,
Доктор технических наук

Л. Н. Ясницкий

Пермь, 2022 год

Аннотация

Название: Разработка MVP системы машинного обучения для прогнозирования рентабельности кинобизнеса.

Автор: Чепоков Елизар Сергеевич, студент четвертого курса образовательной программы «Программная инженерия».

Руководитель: Ясницкий Леонид Нахимович, профессор кафедры информационных технологий в бизнесе, доктор технических наук.

Данная выпускная квалификационная работа посвящена разработке системы машинного обучения для прогнозирования рентабельности кинобизнеса. Предназначением системы является снижение производственных и инвестиционных рисков при производстве кинокартин как для инвесторов, так и для кинокомпаний, а также прогнозирование, на ранних стадиях кинопроизводства, коммерческого потенциала проекта на основании статистических данных.

Работа состоит из 3 глав. В первой главе проанализированы методы прогнозирования и выявлены взаимосвязи рентабельности кинобизнеса, а также выдвинуты требования, предъявляемые системе. Вторая глава описывает процесс проектирования алгоритмической части системы и анализ данных для машинного обучения. Третья глава посвящена реализации конечной системы. В заключении описаны итоги выполненной работы.

Работа включает 163 страниц формата А4, из них в основной части 87 страниц.

Основная часть работы включает в себя 57 иллюстраций и 2 таблицы.

Библиографический список состоит из 71 публикации.

Работа включает в себя 11 приложений.

Оглавление

Введение	5
Глава 1 Аналитический обзор.....	9
1.1 Выявление особенностей прогнозирования кассовых сборов.....	10
1.2 Анализ особенностей систем машинного обучения	16
1.2.1 Анализ архитектур построения нейронных сетей.....	22
1.2.2 Анализ алгоритмов машинного обучения.....	32
1.3 Сравнительный анализ работ по прогнозированию рентабельности кинобизнеса	35
1.4 Обзор существующих программных решений.....	41
1.5 Формирование требований к разрабатываемой системе	44
1.6 Выводы по первой главе	48
Глава 2 Проектирование разрабатываемой системы.....	49
2.1 Описание поведения системы	50
2.2 Проектирование модуля сбора и систематизации обучающих данных.....	51
2.3 Проектирование нейронной сети	60
2.4 Проектирование графического интерфейса.....	63
2.5 Выводы по второй главе	64
Глава 3 Реализация программного решения	65
3.1 Обоснование выбора средств разработки	65
3.2 Реализация модуля сбора и систематизации обучающих данных	66
3.2.1 Реализация алгоритма первичного сбора данных	67
3.2.2 Реализация алгоритма сбора параметров обучающего множества	68
3.2.3 Реализация алгоритма формирования обучающего множества	77
3.3 Реализация модуля нейронной сети	80
3.4 Реализация графического интерфейса.....	82
3.5 Выводы по третьей главе	84

Заключение.....	85
Список сокращений и условных обозначений	87
Библиографический список.....	88
ПРИЛОЖЕНИЕ А Иллюстрации к введению	95
ПРИЛОЖЕНИЕ В Сравнение типов нейронных сетей	98
ПРИЛОЖЕНИЕ С Сравнение алгоритмов машинного обучения	101
ПРИЛОЖЕНИЕ Д Техническое задание на разрабатываемую систему	112
ПРИЛОЖЕНИЕ Е Описание прецедентов.....	124
ПРИЛОЖЕНИЕ F Диаграмма последовательности обучения	129
ПРИЛОЖЕНИЕ G Блок-схемы алгоритма сбора данных	130
ПРИЛОЖЕНИЕ Н Листинг модуля сбора и систематизации данных.....	135
ПРИЛОЖЕНИЕ Ј Листинг модуля нейронной сети	152
ПРИЛОЖЕНИЕ К Листинг графического интерфейса.....	154
ПРИЛОЖЕНИЕ L Дополнительные иллюстрации к третьей главе.....	160

Введение

Киноиндустрия вносит огромный вклад как в общую социальную жизнь человека, так и в развитие экономики стран, она стимулирует многочисленные внешние факторы: потребление связанных с кинокартиной товаров, посещение представленных в кинокартинах стран, развитие индустрии игр и т.п. По данным отчета «Motion picture association» [63] Американской ассоциации кинокомпаний, за 2019 год суммарные кассовые сборы всех вышедших в мировой кинотеатральный прокат фильмов составили \$98,3 млрд., что на 6,8% больше, чем в 2018 году. Однако в 2020 году из-за сложившейся эпидемиологической ситуации во всем мире кассовые сборы резко упали и составили \$80,8 млрд., что на 17.8% меньше чем в 2019 году, количество проданных билетов в 2020 году, по данным сайта «The Numbers» [67] составило 223 млн., против 1,3 млрд в 2019 году и 496 млн. в 2021 году (Рисунок А.3 и А.4 ПРИЛОЖЕНИЯ А). Так же стоит обратить внимание, что доля кинотеатрального проката с каждым годом становится меньше, когда доля цифрового просмотра на стриминговых сервисах¹ растет, так, в 2020 году стриминговые сервисы собрали \$61,8 млрд (76% от суммарных кассовых сборов), что на \$14,6 млрд больше чем в 2019 году (\$47,2 млрд). Результаты отчета МРА 2021 года, представленные на рисунках А.1 и А.2 в «ПРИЛОЖЕНИЕ А» показывают, что в течении года, из-за снятия ограничений на посещение общественных мест выручка кинопроката возросла до значения в \$21,3 млрд., но все еще не достигла значений 2019 года, в свою очередь выручка цифрового проката в 2021 году выросла на 58%. На долю США и Канады приходится 40% всех кассовых сборов (\$32,2 млрд). Россия же находится на 8-ом месте по кассовым сборам, суммарные кинотеатральные сборы в 2020 году в России составили лишь \$192 млн (Рисунки А.6 и А.7 ПРИЛОЖЕНИЯ А).

Производство кинофильмов, было и остается одним из наиболее рискованных видов предпринимательства, особенно во времена ограничений на посещение общественных мест. Затрачивая огромные бюджеты в несколько сотен миллионов долларов на производство фильма кинокомпании ожидают соответствующие

¹ Стриминговые сервисы – сервисы для передачи контента от провайдера к пользователю (онлайн кинотеатры), прим.: «NETFLIX».

доходы от показа фильма в кинотеатрах и продаж на стриминговых сервисах, но инвесторы отмечают, что предсказать финансовый успех фильма крайне сложно даже для опытных участников рынка. По данным сайта «The numbers» [67] лишь около 30% фильмов окупает затраты на их производство и приносит прибыль, так как кинокартина считается прибыльной только при превышении кассовых сборов больше чем в два раза поставленного бюджета. Расширение знаний относительно факторов успеха фильмов позволит уменьшить долю неудачных инвестиций в кинопроизводство. Предварительная оценка будущего коммерческого успеха кинофильмов может позволить инвесторам сузить круг проектов, требующих более тщательного последующего анализа их инвестиционной привлекательности.

Актуальность данной выпускной квалификационной работы заключается в том, что выпуск кинокартины в наши дни является большим риском, не только для киностудий и режиссеров, но и для инвесторов, система прогнозирования способна снизить риски неудачных инвестиций в кинопроизводство на ранних этапах создания кинокартины, а также спрогнозировать целесообразность выпуска кинокартины на экраны кинотеатров с сопутствующими затратами на рекламу и аренду кинозалов или ограничить выпуск лишь в цифровом формате. В настоящее время решения о целесообразности реализации кинопроектов принимаются в большинстве случаев исключительно на основе интуиции и предыдущего опыта продюсерских организаций, что способствует неоправданному росту рисков инвесторов, вкладывающих значительные финансовые средства в кинопроизводство. Хотя, интуиция и опыт являются важнейшими профессиональными качествами продюсера, для роста обоснованности принимаемых решений необходимо, чтобы они основывались на комплексном анализе сильных и слабых сторон кинопроекта до его запуска в производство.

Таким образом, актуальность исследования основывается на необходимости решения важных задач, таких как, научное и статистическое обоснование и создание эффективно функционирующего механизма прогнозирования коммерческого потенциала кинопроекта, принятия рациональных управлеченческих решений о целесообразности его реализации, определение направлений и принципов эффективного управления кинематографическим бизнес-процессом, в частности, на ранних стадиях создания кинофильмов.

Объектом исследования в данной работе является алгоритм системы машинного обучения. **Предметом** исследования выступает разработка алгоритма системы машинного обучения.

Целью данной работы является разработка системы машинного обучения для прогнозирования рентабельности кинобизнеса.

Задачи для достижения поставленной цели:

1. Провести анализ систем машинного обучения и методов прогнозирования.
 - 1.1. Провести анализ особенностей кинобизнеса.
 - 1.2. Провести анализ особенностей построения систем машинного обучения.
 - 1.3. Формализовать существующие методы прогнозирования рентабельности кинобизнеса.
 - 1.4. Провести обзор существующих программных решений.
 - 1.5. Конкретизировать требования к разрабатываемой системе.
2. Спроектировать систему.
 - 2.1. Описать поведение системы и ее компонентов.
 - 2.2. Спроектировать модуль сбора и систематизации данных.
 - 2.3. Спроектировать модуль нейронной сети.
 - 2.4. Спроектировать графический интерфейс разрабатываемой системы.
3. Реализовать систему машинного обучения для прогнозирования рентабельности кинобизнеса.
 - 3.1. Обосновать выбор технологий для разработки.
 - 3.2. Реализовать модуль сбора и систематизации данных и провести его тестирование.
 - 3.3. Реализовать модуль нейронной сети и провести его тестирование.
 - 3.4. Реализовать графический интерфейс разрабатываемой системы и провести его тестирование.

Методы исследования. При проектировании и разработке системы машинного обучения для прогнозирования рентабельности кинобизнеса были использованы такие общенаучные методы исследования, как анализ и синтез, системный подход, статистический анализ, методы сравнений, методы объектно-

ориентированного проектирования и объектно-ориентированного программирования.

Степень разработанности проблемы. Одними из первых исследователей, применивших в кинобизнесе метод экономико-математического моделирования, были J. Prag и J. Casavant, которые в 1994 опубликовали статью с сообщением о создании регрессионной модели на основе выборки из 625 американских фильмов [46]. В 2006 году для прогнозирования кассовых сборов фильмов, американскими учеными R. Sharda и D. Delen был впервые применен аппарат нейронных сетей, а также построены модели на основе дискриминантного анализа и логистической регрессии [51]. Среди русских исследователей следует выделить работу Л.Н. Ясницкого и Н.О. Белобородовой 2017 года [12], а также работы [13, 56] в которых были расширены параметры обучения и проведены анализы зависимостей результата от выбранных параметров.

Практическая значимость данной работы заключается в возможности применения разрабатываемой системы для представления методических рекомендаций по рационализации продюсерской деятельности, обеспечивающих достижение высоких экономических результатов кино-отрасли и отдельных кинокартин.

Научная новизна работы состоит в разработке моделей, подходов и методик, а также системы, способствующих анализу коммерческого успеха кинофильмов, позволяющего сделать количественно обоснованные выводы об их коммерческом потенциале.

Глава 1 Аналитический обзор

Для реализации разрабатываемой системы и для решения поставленной задачи, необходимо провести аналитический обзор выбранной предметной области с целью формирования теоретической базы для дальнейшей работы над проектом. Аналитический обзор включает в себя нахождение особенностей прогнозирования кассовых сборов в кинобизнесе, определения архитектуры и алгоритма реализуемой нейронной системы, а также обзор существующих программных решений и сравнительный анализ предыдущих работ в данной предметной области. В главе решаются следующие задачи:

1. Определение особенностей технического процесса кинобизнеса и построение взаимосвязей результатов кассовых сборов от переменных, присущих процессу создания и публикации кинокартины, а также определение методов сбора данных о кинокартинах из разносторонних ресурсов.
2. Утверждение архитектуры нейронной машины и алгоритмов нейронной сети для обучения и тестирования разрабатываемой системы.
3. Обзор существующих исследований и классификация методологий и параметров, необходимых для решения поставленной задачи.
4. Обзор существующих программных решений с целью выделения сильных и слабых сторон по отношению к решаемой проблеме и обоснования необходимости разработки нового средства, подходящего под регламент задач.
5. Формирование и детализация требований, выдвигаемых разрабатываемой системе, написание технического задания и построение диаграммы прецедентов.

Результатом написания первой главы будет разработано техническое задание, в котором будут отражены функциональные и нефункциональные требования к разрабатываемому программному средству.

1.1 Выявление особенностей прогнозирования кассовых сборов

Первым и обязательным этапом в процессе разработки любого программного обеспечения, является изучение теоретической базы предметной области, для которой средство разрабатывается. В данном разделе необходимо выделить термины и понятия предметной области для которой ведется разработка, а именно кинопроизводство.

Кинопроизводство процесс создания кинокартин от первоначального замысла до непосредственного показа готового продукта аудитории в кинотеатрах, телевидении или стриминговых сервисах. Как было сказано ранее публикация фильмов тесно соприкасается с экономическими, социальными и политическими сферами жизнедеятельности человеческого общества, а успешные фильмы могут увеличить количество туристов в том или ином регионе, а также поднять экономику, представленной в кинокартинах страны. Как правило, кинопроизводство требует вовлечения большого количества людей, больших временных и материальных затрат, наиболее сложными, долгосрочными и затратными в производстве считаются художественные игровые фильмы.

Кинематограф зарождался в конце XIX века. Из-за высокого спроса среди зрителей, запечатленные на кинопленке движущиеся картинки привели к возникновению нового вида искусства и закреплению популярности в обществе. Регулярное производство кинофильмов появилось в конце 1890-х годов и отличалось коротким периодом съемки и выпуска на экраны, художественная составляющая фильма находилась на втором плане, а упор ставился на эффектное зрелище. Первые фильмы длились не больше 10 минут и занимали не более одной части. Первые сформированные этапы кинопроизводства:

1. Написание сценария.
2. Подбор творческой группы (актеры, художник, оператор).
3. Съемочный процесс.
4. Монтажный период.
5. Тиражирование.

Одним из важнейших аспектов кинопроизводства, как разновидности коммерции является возможность получения прибыли и размеры вложенных средств. Основной задачей продюсера или продюсерской компании является поиск

финансирования проекта среди частных фирм, частных инвесторов или государственных структур, которые будут заинтересованы в данных инвестициях. Зачастую продюсеры сами выбирают или заказывают сценаристов, приглашают режиссеров и участвуют в подборе актеров. Стратегия производства и продвижения фильма основывается на изучении зрительского интереса. Контроль производственного процесса выполняется продюсером до самого конца, включая рекламу фильма и его прокат.

Отслеживание кассовых сборов имеет ключевое значение для современного коммерческого кинематографа, отображая экономическую успешность той или иной киноленты. Наиболее часто отслеживаются кассовые сборы за первые выходные дни показа фильма, которые можно приблизительно просчитать, основываясь на информации о забронированных и выкупленных местах до начала показа фильма, данную информацию предоставляют системы предварительных продаж. Уровень дохода от фильма за первые выходные позволяет судить о том, насколько фильм успешен, и планировать продолжительность дальнейшего проката. Также ведется отслеживание общих сборов за время всего показа, одним из агрегаторов² информации о кассовых сборах фильмов выступает сайт «Box Office Mojo» [58], на котором собрана информация о кассе фильмов, путем отслеживания опубликования сводных данных прокатчиками, а также представлена общая тенденция сборов фильмов. В России отслеживанием кассовых сборов занимается «Фонд Кино» на основе данных Единой информационной системы сведений о показах фильмов в кинозалах (ЕАИС).

Продвижение фильма начинается почти одновременно с написанием сценария, для чего съемочной группой подготавливается стандартный комплект рекламных материалов, в который входят:

- Расширенная аннотация.
- Список основного состава творческой группы.
- Фотографии наиболее выразительных кадров фильма, снятых во время съемок на натуре и в павильоне.

² Агрегатор – сервис, который собирает информацию из разных источников, сайтов или баз данных в один источник

После чего готовятся рекламные ролики, из дублей, не вошедших в фильм или отснятых специально для этапа продвижения. Рекламная работа ведется по следующим направлениям:

- Сети кинотеатров (рекламные ролики).
- Интернет (репортажи со съемок).
- Теле-радио вещание (рекламные ролики).
- Печатные издания (фотографии и интервью).

В обязанности продюсера входит просчитывание: сколько кинотеатров купят фильм; сколько зрителей его посмотрят; насколько он окупится и какую принесет прибыль. Составление прогнозов относительно прибыли фильма является одной из самых важных частей создания фильмов, оцениваются риски и выявляются элементы на которые необходимо сфокусировать внимание. Одним из вариантов прогнозирования является проведение тестовых показов для настоящей аудитории кино. Так же учитывается сезонность релиза фильма в кинопрокат, на которую опираются продюсеры и кинокомпании для достижения поставленного результата. Традиционно выделяют следующие сезоны:

- Апрель – май: выход в прокат «блокбастеров», обычно с наличием франшизы, предыдущие фильмы которых набрали популярность и зарекомендовали себя среди зрителей.
- Август: выход «пробных» кинокартин.
- Сентябрь: фильмы, претендующие на «Оскар».
- Ноябрь – декабрь: семейные фильмы.

Помимо вышеперечисленных факторов, огромное значение для последующих сборов имеют такие факторы как бюджет самого фильма, имена режиссера, участвующих звезд и самой киностудии и другие факторы, прямо или косвенно способствующие привлечению зрителей в кинотеатры.

Таким образом, основываясь на представленной выше информации можно выделить критерии, влияющие на кассовые сборы фильма:

- Бюджет фильма.
- Сезон выхода фильма.
- Режиссер, его популярность среди зрителей и рейтинг предыдущих, его работ.

- Актерский состав и популярность звезд на главных ролях.
- Зрительский интерес к кинокартинам со схожим жанром.
- Комментарии зрителей перед выходом фильма, на основе рекламы.
- Результаты тестового проката.
- Количество забронированных и купленных билетов до старта показа.

В своих исследованиях 1994 года J. Prag и J. Casavant [46], как уже описывалось ранее, выделили следующие критерии:

- Критические обзоры.
- Наличие звезд.
- Наличие франшизы.
- Наличие премий.
- Жанр.
- Возрастное ограничение.

Соответственно, для оценки и прогнозирования кассовых сборов следует опираться на описанные выше критерии, позволяющие выявить и классифицировать входные данные проектируемой системы, а также удовлетворить требования конечных пользователей, предъявляемые системе.

Для сбора данных требуется выделить агрегаторы предоставляющие информацию о фильмах и возможности скраппинга³/парсинга⁴ выбранной информации.

КиноПоиск [60] – русскоязычный интернет сервис о кино, на момент написания данной работы база данных сайта насчитывает около 3 млн фильмов. Сайт предоставляет полную информацию, а фильме: кассовые сборы, творческая группа, рейтинг, кассовые сборы и т.д.

Преимущества:

- Наличие ссылок-переходов на другие агрегаторы.
- Отображение всей основной информации на странице фильма.

Недостатки:

³ Сcrapпинг (scraping) – поверхностное извлечение данных из предварительно собранных URL-адресов и сохранение в хранилище без какой-либо обработки.

⁴ Парсинг (parsing) – автономный алгоритм для обработки и анализа данных, в отличии от скраппинга, используется для извлечения информации из неструктурированных данных.

- Отсутствие API для автоматического сбора информации.
- Постоянное изменение структуры сайта, для предотвращения скрапинга/парсинга данных с сайта.
- Наличие защиты от ботов.
- Завышение рейтинга фильмов, по просьбе прокатчика или киностудии.
- У фильмов, не пользующихся популярностью частично или полностью отсутствуют данные.

IMDb – «Интернет-база фильмов» [59], крупнейшая база данных о фильмах и кинематографе, на момент написания работы база данных сайта насчитывает около 5 млн фильмов. Сайт предоставляет полную информацию со ссылками на источники о фильме, а также предоставляет информацию о каждом участнике творческого состава, его рейтинге, наградах и т.д.

Преимущества:

- Наличие API для автоматического сбора информации.
- Самая обширная база данных на данный момент.
- Стабильная структура сайта.
- Достоверные данные о бюджете и кассовых сборах предоставляемые сайтом «Box Office Mojo».

Недостатки:

- API предоставляется по подписке, без возможности получить доступ бесплатно в рамках создания какого либо проекта.
- Частичное отсутствие данных у разных фильмов, из-за наличия модерации страницы фильма любым пользователем.

Rotten Tomatoes [66] – сайт-агрегатор рецензий на кинофильмы, предоставляет расширенную информацию о оценке и рецензиях фильмов от пользователей, а так же от профессиональных критиков.

Преимущества:

- Наличие на сайте информации для семантического анализа рецензий кинофильма.
- Достоверные данные о рейтинге фильма.
- Наличие бесплатного API для автоматического сбора информации.

Недостатки

- Отсутствие большого количества данных о фильмах.

Metacritic [62] – сайт-агрегатор собирающий отзывы о фильмах, играх и прочем. Предоставляет информацию о оценке фильмов от пользователей и изданий.

Преимущества:

- Предоставление данных об оценке фильма популярным изданием.

Недостатки:

- Отсутствие API для автоматического сбора данных.
- Меньшее количество предоставляемой информации по сравнению с «Rotten Tomatoes».

Box Office Mojo [58] – сайт-агрегатор кассовых сборов и бюджета фильма. Международный раздел еженедельно обновляет статистику истории сборов для отдельных фильмов из более 150 стран. На сайте учитываются ежегодные и вневременные особенности наблюдаемых стран.

Преимущества:

- Официальная информация о бюджетах и кассовых сборах фильмов, получаемая от компаний прокатчиков и кинематографических студий.
- Разделение данных на группы по странам и регионам.

Недостатки:

- Отсутствие API для автоматического сбора информации.

Таким образом сбор информации о фильме и о его кассовых сборах, которые будут использоваться для тренировки и тестирования нейронной сети будут производится на сайтах-агрегаторах: «IMDb» [59], «Rotten Tomatoes» [66] и «Box Office Mojo».

Сайт «IMDb» будет использоваться для поиска сбора основной информации о фильме, такой как: жанр фильма, творческий состав, возрастном рейтинге, длительности и других параметров. С сайта «Rotten Tomatoes» будет собираться информация о рейтинге фильма и рецензиях, для последующего семантического анализа. С сайта «Box Office Mojo» будут собираться данные о бюджете фильма и его кассовых сборах.

1.2 Анализ особенностей систем машинного обучения

В данной главе рассмотрены основные принципы, термины и понятия систем машинного обучения, нейронных сетей и искусственного интеллекта. Целью аналитического обзора особенностей систем машинного обучения является определение существующих типов построения нейронных сетей и их алгоритмов обучения. Результатом главы выступает выбранный тип нейронной сети на основе которой будет простроена нейронная сеть для создания разрабатываемой системы, а также выбранный алгоритм обучения нейронной сети для обучения и тестирования данных системы.

Искусственный интеллект – свойство интеллектуальных систем выполнять функции, которые свойственны лишь человеку. Существующие на данный момент интеллектуальные системы приспособлены использоватьсь лишь в достаточно узких областях, например, системы, обученные под игры в шахматы или логические игры, не могут отвечать на вопросы человека и не могут предсказывать погоду, как делают системы, заточенные на прогнозирование погодных условий.

Впервые термин «Искусственный интеллект» был использован в преамбуле, данным Джоном Маккартни на семинаре в Дартмутском университете [7] в 1956 году. Данное определение не было напрямую связано с пониманием интеллекта у человека, согласно Маккарти, ИИ-исследователи вольны использовать методы, которые не наблюдаются у людей, если это необходимо для решения конкретных проблем. Поясняя своё определение, Джон Маккарти указывает: «Проблема состоит в том, что пока мы не можем в целом определить, какие вычислительные процедуры мы хотим называть интеллектуальными. Мы понимаем некоторые механизмы интеллекта и не понимаем остальные. Поэтому под интеллектом в пределах этой науки понимается только вычислительная составляющая способности достигать целей в мире» [69].

Под каждую задачу искусственного интеллекта принято подбирать необходимую технологию решения данной задачи. В рамках разработки системы «Прогнозирование рентабельности кинобизнеса» таковой выступает – задача прогнозирования, для чего принято использовать искусственные нейронные сети.

Искусственные нейронные сети (ИНС) в свою очередь, с точки зрения искусственного интеллекта являются основой философского течения

коннекционизма и основным направлением в структурном подходе по изучению возможности построения и моделирования естественного интеллекта с помощью компьютерных алгоритмов. С точки зрения развития вычислительной техники и программирования, нейронная сеть является способом решения проблемы эффективного параллелизма [3].

Первой попыткой создания нейронной сети считается созданная сеть У. Маккалока и У. Питтса в 1943 году за 13 лет до определения термина «Искусственный интеллект», данной нейронной сетью авторы фармализируют понятие «Нейронная сеть» в своей статье [5]. Первый алгоритм обучения нейронных сетей был предложен Д. Хеббом в 1949 году. Основным деятелем, продвигавшим разработку нейронных сетей и нейросетевых алгоритмов, стал Ф. Розенблatt, после изобретения им однослойного перцептрона в 1958 [48], для решения задач классификации. Нейронные сети представляют собой частный случай методов распознавания образов и дискриминантного анализа для машинного обучения.

Как уже было описано ранее нейронные сети используются для решения некоторого набора задач, которые возможны к решения таким способом. Среди известных применений нейронных сетей подразделяют следующие:

- Распознавание образов и классификация – при обучении предлагаются различные образцы образов с указанием того, к какому классу они относятся. Образец, как правило, представляется как вектор значений признаков. При этом совокупность всех признаков должна однозначно определять класс, к которому относится образец.
- Принятие решений и управление – аналогично задаче классификации, при которой классификации подлежат ситуации, характеристики которых поступают на вход нейронной сети. На выходе сети при этом должен появиться признак решения, которое она приняла.
- Кластеризация – разбиение множества входных сигналов на классы, при том, что ни количество, ни признаки классов заранее не известны. После обучения такая сеть способна определять, к какому классу относится входной сигнал.
- Прогнозирование – следует из способности к обобщению и выделению скрытых зависимостей между входными и выходными данными

нейронной сети. После обучения сеть способна предсказать будущее значение некой последовательности на основе нескольких предыдущих значений и/или каких-то существующих в настоящий момент факторов.

- Аппроксимация – возможность получить устройство из произвольного нелинейного элемента, вычисляющее любую непрерывную функцию с некоторой наперёд заданной точностью, с помощью линейных операций и каскадного соединения.
- Сжатие данных и ассоциативная память – выражение данных большой размерности более компактно, если данные тесно взаимосвязаны друг с другом или восстановление исходного набора данных из части информации.
- Анализ данных.
- Оптимизация.

Машинное обучение является подразделом искусственного интеллекта и науки о данных, специализирующийся на использовании данных и алгоритмов для имитации процесса наработки опыта человеком с постепенным повышением точности. Обучение характеризуется не прямым решением поставленной задачи, а обучением системы за счет применения решений множества сходных задач. Различают дедуктивный и индуктивный типы обучения.

Дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы данных содержащей информацию о знаниях и опыте в необходимой предметной области и правилах вывода, позволяющих делать автоматические умозаключения об уже имеющихся или вновь вводимых фактах и тем самым производить семантическую обработку информации. Онтологию таких знаний принято называть «База знаний».

Обучение по прецедентам или индуктивное обучение, основано на выявлении общих закономерностей по частным эмпирическим данным или выборке. Выборка состоит из прецедентов генеральной совокупности, выбранных разными методами классификации данных. По каждому прецеденту в выборке собирается совокупность данных, образующих описание конкретного прецедента [1].

Совокупность описаний всех прецедентов выборки, сгруппированных попарно в виде (объект, ответ), является входной информацией или обучающей

выборкой для машинного обучения или статистического и интеллектуального анализа данных [6]. На основе этих данных требуется построить алгоритм, способный для любого входного объекта выдать достаточно точный, классифицирующий ответ с выявлением общей зависимости, закономерности, взаимосвязи, присущие не только конкретной выборке, но и всем, статистически похожим данным. Важной особенностью такого обучения является способность обучаемой системы к обобщению, отклику на данные, выходящие за пределы обучающего множества.

Для обучения по прецедентам фиксируется модель восстанавливаемой зависимости по эмпирическим данным [2]. Затем вводится функционал качества, значение которого показывает насколько хорошо модель описывает наблюдаемые данные. Алгоритм обучения ищет такой набор параметров модели, при котором функционал качества на заданной обучающей выборке принимает оптимальное значение [8, 43].

Различают следующие архитектуры нейронных сетей:

- Персепtron.
- Сверточная сеть.
- Рекурсивная сеть.
- Рекуррентная сеть.
- LSTM.

Данные архитектуры подразделяются на типы нейронных сетей, для которых существуют следующие алгоритмы машинного обучения, основанные на применении нейронных сетей:

- Обучение с учителем.
- Обучение без учителя.
- Частичное обучение.
- Трансдуктивное обучение.
- Обучение с подкреплением.
- Динамическое обучение.
- Метаобучение.

Для создания полноценно обученной нейронной сети и ее обучения так же стоит выделить и рассмотреть подробнее некоторые этапы решения задачи, для которой обучение производится. Всего выделяют 9 этапов решения задач:

1. Сбор данных для обучения.
2. Подготовка и нормализация данных.
3. Выбор топологии сети.
4. Экспериментальный подбор характеристик сети.
5. Экспериментальный подбор параметров обучения.
6. Обучение.
7. Проверка адекватности обучения.
8. Корректировка параметров, окончательное обучение.
9. Вербализация сети с целью дальнейшего использования.

Сбор данных для обучения – является самым сложным этапом решения задачи, включает в себя выбор данных для обучения (обучающего множества) и их обработку. Набор данных для обучения должен удовлетворять нескольким критериям:

- Репрезентативность – данные должны иллюстрировать истинное положение вещей в предметной области.
- Непротиворечивость – противоречивые данные в обучающей выборке приведут к плохому качеству сети.

Исходные данные преобразуются к виду, в котором их можно подать на входы сети. Каждая запись в файле данных называется обучающей парой или обучающим вектором. Обучение сети на неподготовленных данных, как правило, не даёт качественных результатов. Существует ряд способов улучшить «восприятие» сети:

- Нормировка – выполняется, когда на различные входы подаются данные разной размерности. При отсутствии нормировки значения на втором входе будут всегда оказывать существенно большее влияние на выход сети, чем значения на первом входе. При нормировке размерности всех входных и выходных данных сводятся воедино.
- Квантование – выполняется над непрерывными величинами, для которых выделяется конечный набор дискретных значений.
- Фильтрация – выполняется для «зашумленных» данных.

Выбор топологии сети выбирается исходя из постановки задачи и имеющихся данных для обучения. Например, для обучения с учителем требуется наличие для каждого элемента выборки итоговый ответ, если массив данных слишком велик и получение такового ответа невозможно, то используется обучение без учителя.

Подбор характеристик сети необходим для подбора параметров сети после выбора общей структуры. Для сетей подобных перцептрону выбором параметров является: выбор количества слоев, наличие или отсутствие необходимых соединений, придаточные функции нейронов. При выборке количества слоев и нейронов в них следует исходить из того что «способности сети к обобщению тем выше, чем больше суммарное число связей между нейронами».

Подбор параметров обучения особенно важен при выборе обучения с учителем. От данного этапа зависит насколько быстро ответы сети будут сходиться к правильным ответам и форма поверхности ошибки. Значения параметров возможно выбрать лишь путем экспериментального подбора, руководствуясь критерием завершения обучения (минимизация ошибки или ограничение времени обучения).

Обучение сети является основным этапом решения задачи, при котором возможно появление совершенно непредсказуемых ошибок, как переобучение⁵, паралич сети, попадание сети в локальный минимум поверхности ошибок. Во время обучения сети обучающую выборку делят на три части в соотношении 70/30%, где 70% - обучающее множество, 30% - тестовое множество. Тестовое множество так же разделяют на валидирующее (используемое для обобщения) и проверочное множество в соотношении 15/15%.

Проверка адекватности обучения. Даже в случае успешного, на первый взгляд, обучения сеть не всегда обучается именно тому, чего от неё хотел создатель. Тестирование качества обучения нейронной сети необходимо проводить на примерах, которые не участвовали в её обучении. При этом число тестовых примеров должно быть тем больше, чем выше качество обучения.

⁵ Переобучение (overfitting) – наиболее распространенное явление в машинном обучении, при котором построенная модель хорошо объясняет примеры из обучающей выборки, но плохо на примерах, не участвовавших в ней.

1.2.1 Анализ архитектур построения нейронных сетей

Персепtron или перцепtron - математическая или компьютерная модель восприятия информации мозгом, предложенная Фрэнком Розенблаттом в 1958 году и впервые реализованная в виде электронной машины. Персепtron стал одной из первых моделей нейронных сетей, а «Марк-1» — первым в мире нейрокомпьютером [48].

Персептроны позволяют создать набор «ассоциаций» между входными стимулами и необходимой реакцией на выходе. В биологическом плане это соответствует преобразованию, например, зрительной информации в физиологический ответ от двигательных нейронов. Элементарный персепtron состоит из элементов трёх типов:

А-элементы называются ассоциативными, так как каждому такому элементу, соответствует набор-ассоциация S-элементов. Данный элемент даёт выходной сигнал $+1$, когда алгебраическая сумма его входных сигналов превышает некоторую пороговую величину θ (элемент активный), в противном случае выход равен 0. Сигналы от возбудившихся А-элементов, передаются в сумматор R, сигнал от i ассоциативного элемента передаётся с коэффициентом w_i – весом А-R связи [9, 48], схема связей представлена на рисунке 1.1.

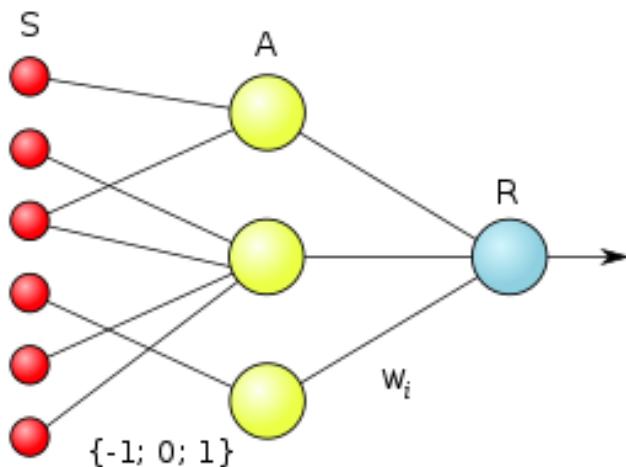


Рисунок 1.1 – Схема работы персептрана

S-элементы являются слоем сенсоров или рецепторов (например, светочувствительная сетчатка глаза) элементы которого вырабатывают сигнал от воздействия какого-либо из видов энергии. Каждый рецептор может находиться в одном из двух состояний (покой, возбуждение), если входной сигнал превышает некоторый порог θ , на выходе элемента получаем $+1$, в противном случае 0 [48, 49].

R-элемент является действующим и подсчитывает сумму значений входных сигналов, помноженных на веса, R-элемент который является выводом персептрана, выдаёт сигнал «1», если линейная форма превышает порог θ , иначе на выходе будет «−1» [49]:

$$f(x) = \text{sign}(\sum_{i=1}^n w_i x_i - \theta) \quad (1.1)$$

Обучение элементарного персептрана состоит в изменении весовых коэффициентов w_i связей A-R. Веса связей S-A, принимающих значения {−1; 0; +1}. Значения порогов А-элементов выбираются случайным образом в самом начале и затем не изменяются.

После обучения персептран готов работать в режиме распознавания или обобщения [39]. В этом режиме персептрану предъявляются ранее неизвестные ему объекты, при предъявлении объекта А-элементы передают сигнал R-элементу, равный сумме соответствующих коэффициентов w_i , если эта сумма положительна, то принимается решение, что данный объект принадлежит к первому классу, а если она отрицательна — то ко второму. На рисунке 1.2 представлена архитектура многослойного персептрана.

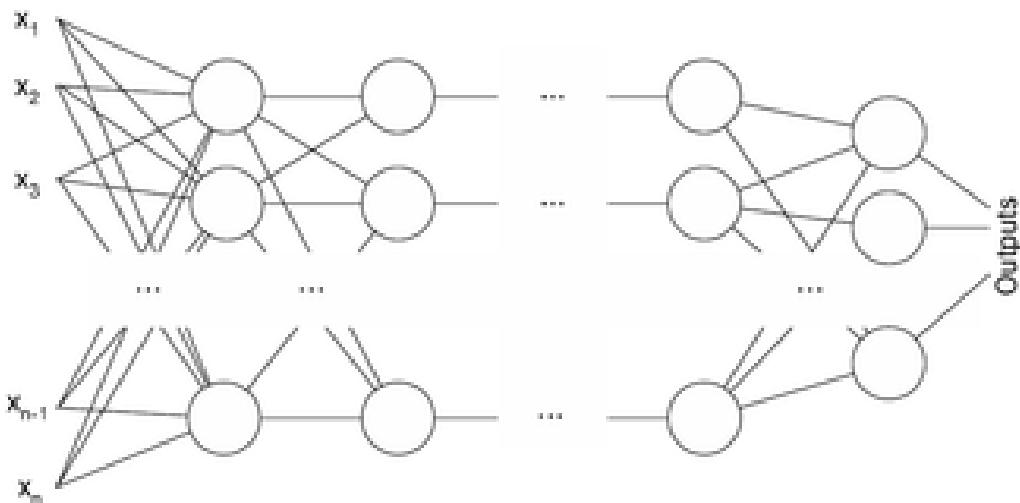


Рисунок 1.2 – Архитектура многослойного персептрана

Нейронная сеть Хопфилда (Рисунок 1.3) – полносвязная нейронная сеть с симметричной матрицей связей. В процессе работы динамика таких сетей сходится к одному из положений равновесия, которые определяются заранее в процессе обучения, они являются локальными минимумами функционала, называемого

«энергией» сети. Такая сеть может быть использована как автоассоциативная память, как фильтр, а также для решения некоторых задач оптимизации.

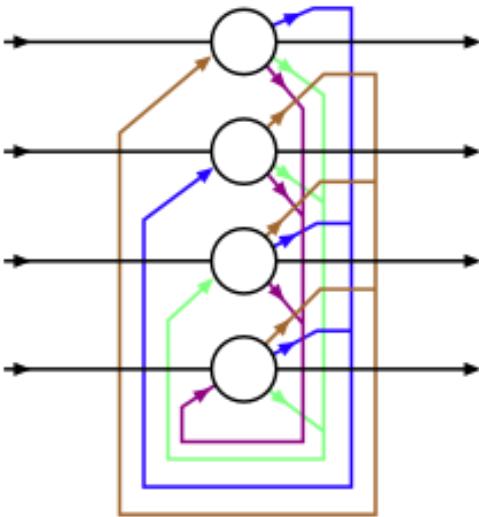


Рисунок 1.3 – Схема сети Хопфилда с четырьмя нейронами

В отличие от многих нейронных сетей, работающих до получения ответа через определённое количество тактов, сети Хопфилда работают до достижения равновесия, когда следующее состояние сети в точности равно предыдущему: начальное состояние является входным образом, а при равновесии получают выходной образ [31]. Формула обучения:

$$\Delta w_{ij} = s_i * s_j \quad (1.2)$$

где w_{ij} – сила связи между нейронами i и j ,

s_i – состояние, $s_i \in \{0, 1\}$, нейрона i ,

s_j – состояние, $s_j \in \{0, 1\}$, нейрона j .

Сеть Хопфилда однослойная и состоит из N искусственных нейронов. Каждый нейрон системы может принимать на входе и на выходе одно из двух состояний (1, -1). Каждый нейрон связан со всеми остальными нейронами.

Взаимодействие нейронов в сети описывается выражением:

$$E = \frac{1}{2} \sum_{i,j=1}^N w_{ij} x_i x_j \quad (1.3)$$

где E – «энергия» сети,

$w_{i,j}$ – элемент матрицы взаимодействий W ,

x_i – состояние нейрона $i \in \{0, 1\}$,

x_j – состояние нейрона $j \in \{0, 1\}$.

Машина Больцмана (Рисунок 1.4) – вид стохастической рекуррентной нейронной сети, изобретенной G. Hinton и T. Sejnowski в 1985 году [16]. Машина Больцмана может рассматриваться как стохастический генеративный вариант сети Хопфилда. Эта сеть использует для обучения алгоритм имитации отжига и способна обучаться внутренним представлениям.

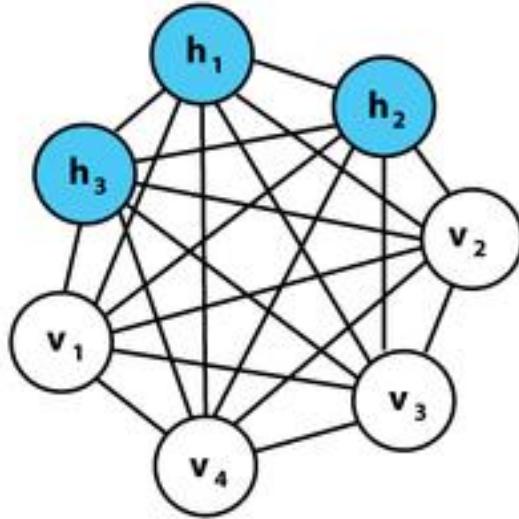


Рисунок 1.4 – Графическое представление машины Больцмана

Несмотря на это, из-за ряда проблем, машины Больцмана с неограниченной связностью не могут использоваться для решения практических проблем. Если же связность ограничена, то обучение может быть достаточно эффективным для использования на практике. Формула обучения:

$$\Delta w_{ij} = e * (p_{ij} - p'_{ij}) \quad (1.4)$$

где w_{ij} – сила связи между нейронами i и j ,

e – скорость обучения,

p_{ij} – фактическое распределение,

p'_{ij} – прогнозируемое распределение.

Как и сеть Хопфилда, машина Больцмана является сетью нейронов с определенной для неё понятием «энергии». Расчет глобальной энергии производится идентичным по форме с сетью Хопфилда образом [4]:

$$E = - \sum_{i < j} w_{ij} s_i s_j - \sum_i \theta_i s_i \quad (1.5)$$

где E – «энергия» сети,

$w_{i,j}$ – сила связи между нейронами i и j ,

s_i – состояние, $s_i \in \{0, 1\}$, нейрона i ,

s_j – состояние, $s_j \in \{0, 1\}$, нейрона j .

Ограничения:

$w_{ii} = 0$: \forall_i – нейрон не может иметь связь с самим собой,

$w_{ij} = w_{ji}$: $\forall_{i,j}$ – все связи являются симметричными.

Ограниченная машина Больцмана или RBM (Рисунок 1.5) – вид генеративной стохастической нейронной сети, которая определяет распределение вероятности на входных образцах данных, является модификацией обычной машины Больцмана, в которой нейроны разделили на видимые и скрытые, а связи допустимы только между нейронами разного типа [53].

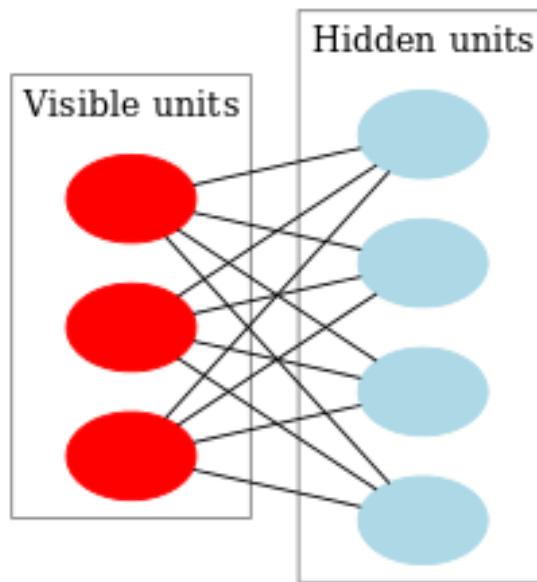


Рисунок 1.5 – Схема сети ограниченной машины Больцмана

Объединение нескольких каскадов ограниченных машин Больцмана формирует глубокую сеть доверия, особый вид многослойных нейронных сетей, которые могут самообучаться без учителя при помощи алгоритма обратного распространения ошибки [26]. Вид обучения:

$$\Delta w_{ij} = e * (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{equilibrium}) \quad (1.6)$$

где w_{ij} – сила связи между нейронами i и j ,

e – скорость обучения,

$\langle v_i h_j \rangle$ – прогноз

В ограниченной машине Больцмана нейроны образуют двудольный граф, с одной стороны графа находятся видимые нейроны (вход), а с другой стороны – скрытые, причём перекрёстные связи устанавливаются между каждым видимым и

каждым скрытым нейроном. Такая система связей позволяет применить при обучении сети метод градиентного спуска с контрастивной дивергенцией [42].

Ограниченнная машина Больцмана базируется на бинарных элементах с распределением Бернулли, составляющие видимый и скрытый слои сети:

$$E(v, h) = - \sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j \quad (1.7)$$

где E – «энергия» сети (v, h) ,

a_i – смещения видимого слоя,

v_i – видимый слой,

b_j – смещения скрытого слоя,

h_j – скрытый слой,

$w_{i,j}$ – матрица весов W .

Особенностью ограниченных машин Больцмана является возможность проходить обучение без учителя в виде сети глубоких убеждений или DBN [22, 30], которую можно рассматривать как совокупность простых обучающих модулей, каждый из которых представляет собой ограниченный тип машины Больцмана, который содержит слой видимых единиц, представляющих данные, и слой скрытых единиц, которые учатся представлять функции, которые фиксируют корреляции более высокого порядка в данных (Рисунок 1.6). Скрытый слой машины представляет собой глубокие признаки в данных, которые выявляются в процессе обучения.

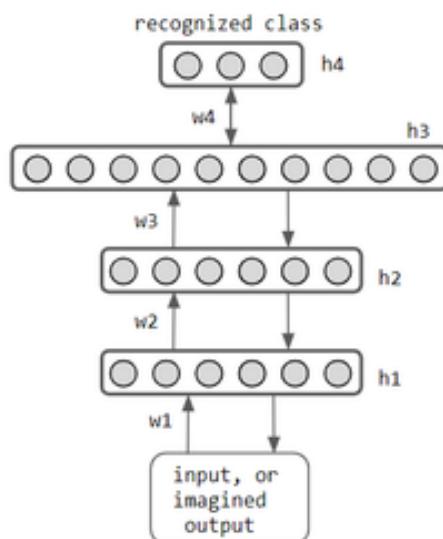


Рисунок 1.6 – Схема сети глубоких убеждений с тремя слоями (h) и симметричными весами (w)

Машина Гельмгольца (Рисунок 1.7) – тип искусственной нейронной сети, которая может учитывать скрытую структуру набора данных путем обучения для создания генеративной модели исходного набора данных, базовая структура генеративной модели должна разумно приближаться к скрытой структуре набора данных, изучая экономические представления данных [20]. Нейронная сеть состоит из двух сетей, объединенных в одну: сеть распознавания снизу-вверх, которая принимает данные в качестве входных данных и производит распределение по скрытым переменным, и "генеративную" сеть сверху вниз, которая генерирует значения скрытых переменных и самих данных, стохастический бинарный нейрон b^G_k выдает вероятность того, что его состояние равно 0 или 1.

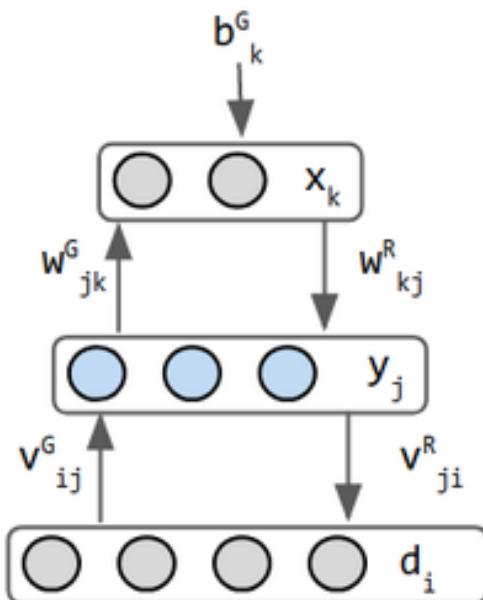


Рисунок 1.7 – Схема машины Гельмгольца

Машины Гельмгольца обычно обучаются с использованием неконтролируемого алгоритма обучения, такого как алгоритм бодрствования-сна [27]. Они являются предшественниками вариационных автокодеров, которые вместо этого обучаются с использованием обратного распространения. Машины Гельмгольца также могут использоваться в приложениях, требующих контролируемого алгоритма обучения (например, распознавание символов или распознавание объекта в поле, не зависящее от положения).

Автокодировщик (Рисунок 1.8 и 1.9) - архитектура искусственных нейронных сетей, позволяющая применять обучение без учителя при использовании метода обратного распространения ошибки [10, 38]. Простейшая архитектура автокодировщика – сеть прямого распространения, без обратных связей, наиболее

схожая с персептроном и содержащая входной слой, промежуточный слой и выходной слой. В отличие от персептрана, выходной слой автокодировщика должен содержать столько же нейронов, сколько и входной слой. Основной принцип работы и обучения сети автокодировщика – получить на выходном слое отклик, наиболее близкий к входному [49]. Для улучшения результатов на нейронную сеть накладывают ограничения: уменьшение размерности промежуточного слоя (промежуточный < входной) или искусственное ограничение единовременно активных нейронов промежуточного слоя, из-за чего нейронная сеть автоматически обучается выделять из входных данных общие признаки, которые кодируются в значениях весов искусственной нейронной сети. Так, при обучении сети на наборе различных входных изображений, нейронная сеть может самостоятельно обучаться распознавать линии и полосы под различными углами [40, 54].

Чаще всего автокодировщики применяют каскадно для обучения глубоких

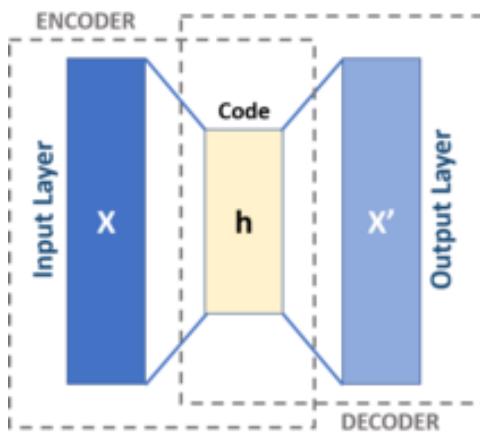


Рисунок 1.8 – Визуализация структуры автокодировщика

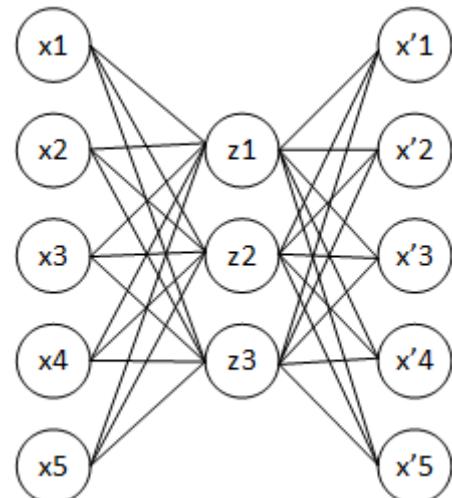
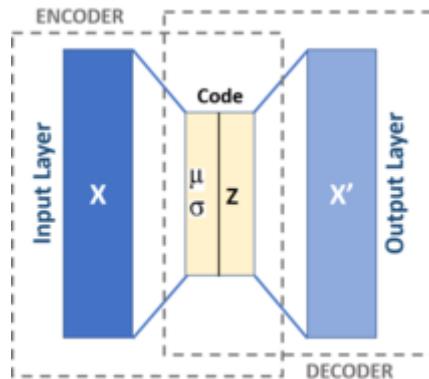


Рисунок 1.9 – Нейронная сеть автокодировщика

(многослойных) сетей. Автокодировщики применяют для предварительного обучения глубокой сети без учителя. Основными практическими приложениями автокодировщиков остаются уменьшение шума в данных, а также уменьшение размерности многомерных данных для визуализации [18, 21].

Вариационный автокодировщик или VAE (Рисунок 1.10) – архитектура искусственной нейронной сети, представленная K. Diederik и W. Max, принадлежащая к семействам вероятностных графических моделей и вариационных байесовских методов [36]. Архитектурно близок к простому автокодировщику, но с различиями в цели математической формулировки [28, 38]. Вариационные автокодировщики позволяют переписывать задачи статистического вывода как

задачи статистической оптимизации (т.е. находить значения параметров, которые минимизируют некоторую целевую функцию) [32]. Они предназначены для сопоставления входной переменной с многомерным скрытым распределением. Хотя этот тип модели изначально был разработан для обучения без учителя [22, 30], его эффективность была доказана в других областях машинного обучения: частичное обучение [23, 55] или обучение с учителем [33].



В вариационном автоэнкодере входные данные отбираются из параметризованного распределения (предшествующего, в терминах байесовского

Рисунок 1.10 – Визуализация структуры вариационного автокодировщика
вывода), а кодер и декодер обучаются совместно таким образом, чтобы выходные данные сводили к минимуму ошибку восстановления в смысле расхождения Кулбека–Лейблера между параметрическим задним и истинным задним (Рисунок 1.11) [17, 34, 37].

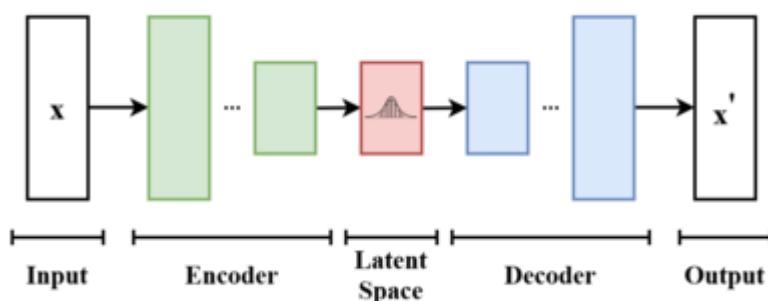


Рисунок 1.11 – Схема работы вариационного автокодировщика

С формальной точки зрения, учитывая входной набор данных, характеризующийся неизвестным распределением вероятностей, цель обучения состоит в том, чтобы смоделировать или приблизить истинное распределение данных с использованием параметризованного распределения.

$$p_{\theta}(x) = \int_z p_{\theta}(x, z) dz \quad (1.8)$$

Где p_θ – параметризованное распределение,

x – набор данных (датасет⁶),

z – скрытая кодировка.

Так же существуют и другие виды нейронных сетей, являющиеся расширением или совокупность перечисленных выше сетей, подробнее все виды нейронных сетей обозрены в «ПРИЛОЖЕНИЕ В», некоторые из этих сетей:

- Нейронная сеть Коско.
- Нейронная сеть Джордана.
- Нейронная сеть Элмана.
- Нейронная сеть Хэмминга.
- Рекуррентные нейронные сети.
- Свёрточная нейронная сеть.
- Байесовская сеть.
- Когнитрон.

По итогам сравнения видов нейронных сетей «см. ПРИЛОЖЕНИЕ В» была составлена таблица 1.1, содержащая нейронные сети, способные решить задачу разрабатываемой системы, а именно – прогнозирование рентабельности кинобизнеса.

Таблица 1.1 – Выбранные нейронные сети

Название	Сфера применения	Сильные стороны	Слабые стороны
<i>Персепtron</i>	Прогнозирование, управление агентами, слабая возможность классификации	Способность к обучению по простому и эффективному алгоритму	Качество прогноза и точность построенной модели зависит от числа знаний, используемых при построении модели
<i>Нейронная сеть Джордана</i>	«То же что и в Персептроне»	Нейроны имеют обратную связь	–

⁶ Датасет (data set) – коллекция из логических записей, хранящихся в виде кортежа.

Название	Сфера применения	Сильные стороны	Слабые стороны
<i>Нейронная сеть Коско</i>	Выявление ассоциаций	Адаптивность	Малый объём памяти
<i>МашинаГельмгольца</i>	Создание представлений, мимикрия	Анатомический, анализируемый с теорией информации	–

1.2.2 Анализ алгоритмов машинного обучения

Раздел машинного обучения, образовался в результате разделения науки о нейросетях на методы обучения сетей и виды топологий их архитектуры, а также вобрал в себя методы математической статистики. Базовые виды нейронных сетей, как перцептрон, многослойный перцептрон и их модификации, могут обучаться как с учителем, так и без учителя, частично или с подкреплением. Но некоторые нейронные сети и большинство статистических методов можно отнести только к одному из способов обучения. Поэтому, если нужно классифицировать методы машинного обучения в зависимости от способа обучения, то будет некорректным относить нейронные сети к определенному виду и правильнее типизировать алгоритмы обучения нейронных сетей. В «ПРИЛОЖЕНИЕ С», детально описаны все наиболее распространенные алгоритмы машинного обучения.

Из всех алгоритмов обучения стоит выделить несколько основных, которыми являются: «Обучение с учителем», «Обучение без учителя» и «Частичное обучение». Для каждой задачи на обучение, принято подбирать необходимый алгоритм и необходимый метод решения задачи для максимизации точности результата.

Обучение с учителем – наиболее распространённый случай машинного обучения, при котором нейронная сеть должна выявить неизвестную функциональную зависимость между ответами и изначальными данными, основываясь на конечной совокупности попарных прецедентов (изначальные данные, ответ) и построить алгоритм, принимающий на входе описание объекта и

выдающий на выходе ответ [50]. Под учителем понимается либо сама обучающая выборка, либо тот, кто указал на заданных объектах правильные ответы.

Одной из разновидностей обучения с учителем является мета-обучение, при котором прецедентами являются ранее решённые задачи обучения, для сведения задачи выбора алгоритма к задаче обучения с учителем, задачи описываются мета-признаками.

Обучение без учителя – один из способов машинного обучения, при котором испытуемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора. В отличие от обучения с учителем, где данные помечаются экспертом, алгоритм, с помощью мимикии адаптируется и создает внутреннее представление мира, после чего генерирует результат, неконтролируемые методы демонстрируют самоорганизацию, которая фиксирует закономерности в виде плотности вероятности или комбинации предпочтений нейронных функций [22, 30].

Частичное обучение (Обучение с частичным привлечением учителя) – способ машинного обучения, при котором входные данные могут быть как помечены учителем, так и быть неразмеченными [23, 55]. Такой вид обучения занимает промежуточную позицию между обучением без учителя (без привлечения каких-либо размеченных данных для тренировки) и обучением с учителем (с привлечением лишь размеченных данных). При обучении обычно использую небольшое количество размеченных данных и большое количество неразмеченных.

Задание размеченных данных для задачи обучения часто требует квалифицированного человека (например, для перевода звуковой дорожки в текст) или физического эксперимента (например, для определения 3D структуры белка или выявления наличия нефти в определенном регионе). Поэтому затраты на разметку данных могут сделать процесс обучения с использованием лишь размеченных данных невыполнимым, в то время как процесс задания неразмеченных данных не является очень затратным. В таких ситуациях, полуавтоматическое обучения может иметь большое практическое значение. Такое обучение также представляет интерес в сфере машинного обучения и как модель для человеческого обучения. Разновидностью частичного обучения выступает «Трансдуктивное обучение» и «Обучение с подкреплением».

При трансдуктивном обучении прогноз предполагается делать только для прецедентов из тестовой выборки. Необходимость такого обучения возникает при необходимости поиска кластеров в нескольких группах данных в случае двоичной классификации, а также при необходимости приближения, когда точный ответ вычислительно невозможен.

Для обучения с подкреплением характерно взаимодействие с некоторой средой, в котором в котором агенты обучения пытаются адаптироваться или найти более оптимальный результат. Обучение с подкреплением является частным примером генетического алгоритма, при котором обучение производится по эпохам, в которых сохраняются самые успешные результаты и применяются как пример обучения для следующих эпох. Из-за того, что откликом среды на принятые решения являются сигналы подкрепления, то данный вид обучения можно отнести как к обучению с учителем, так и к обучению без учителя, так как некоторые правила подкрепления базируются на неявных учителях, например, в случае искусственной нейронной среды, на одновременной активности формальных нейронов.

По результатам описания алгоритмов машинного обучения «см. ПРИЛОЖЕНИЕ С» был выбран алгоритм, подходящий для решения задачи, разрабатываемой системы. Так как решаемая задача является задачей прогнозирования и/или задачей классификации, то для решения данной задачи необходимо составить нейронную сеть с входными данными и ответами, а соответственно выбрать алгоритм «Обучения с учителем». Входными данными будут являться признаковые описания кинофильмов.

1.3 Сравнительный анализ работ по прогнозированию рентабельности кинобизнеса

Для выбора необходимых методов машинного обучения и набора входных данных, необходимо провести анализ предыдущих работ и выделить совпадения и различия в формулировках и построениях нейросетевых алгоритмов для решения задачи выбранной темы дипломной работы. Результатом сравнительного анализа выступает список входных параметров и набор нейросетевых алгоритмов, на основе которых производилось обучение и тестирование нейронных сетей в исследованиях, представленных ниже.

Одними из первых исследователей, применивших в кинобизнесе метод экономико-математического моделирования, были J. Prag и J. Casavant опубликовавшие статью [46] с сообщением о создании регрессионной модели, как уже было описано ранее (п. 1.1 текущей главы), исследователями были выявлены следующие входные параметры:

- Критические обзоры;
- Наличие звезд;
- Наличие франшизы;
- Наличие премий;
- Жанр фильма;
- Возрастное ограничение.

В 2006 году американскими учеными R. Sharda и D. Delen в работе [51] был впервые применен аппарат нейронных сетей для прогнозирования кассовых сборов фильмов, так же были построены модели на основе логистической регрессии, дискриминантного анализа, классификационного и регрессионного дерева. Во входные параметры входило:

- Возрастное ограничение;
- Конкуренция в период релиза;
- Наличие звезд;
- Жанр фильма;
- Качество съемки;
- Наличие франшизы;

- Количество премьерных показов.

Нейронная сеть (Рисунок 1.12) была построена на основе многослойного персептрона с 1 входным слоем, имеющим 26 нейронов, 2 скрытыми слоями: 18 нейронов в первом слое и 16 нейронов во втором и 1 выходным слоем с 9 нейронами.

В 2010 году эти же авторы в работе [52] улучшили прогнозирование, включив деревья решений и более полную выборку фильмов. Входные значения и нейронная машина остались, неизменными. Данная сеть смогла спрогнозировать результат с погрешностью в 54%.

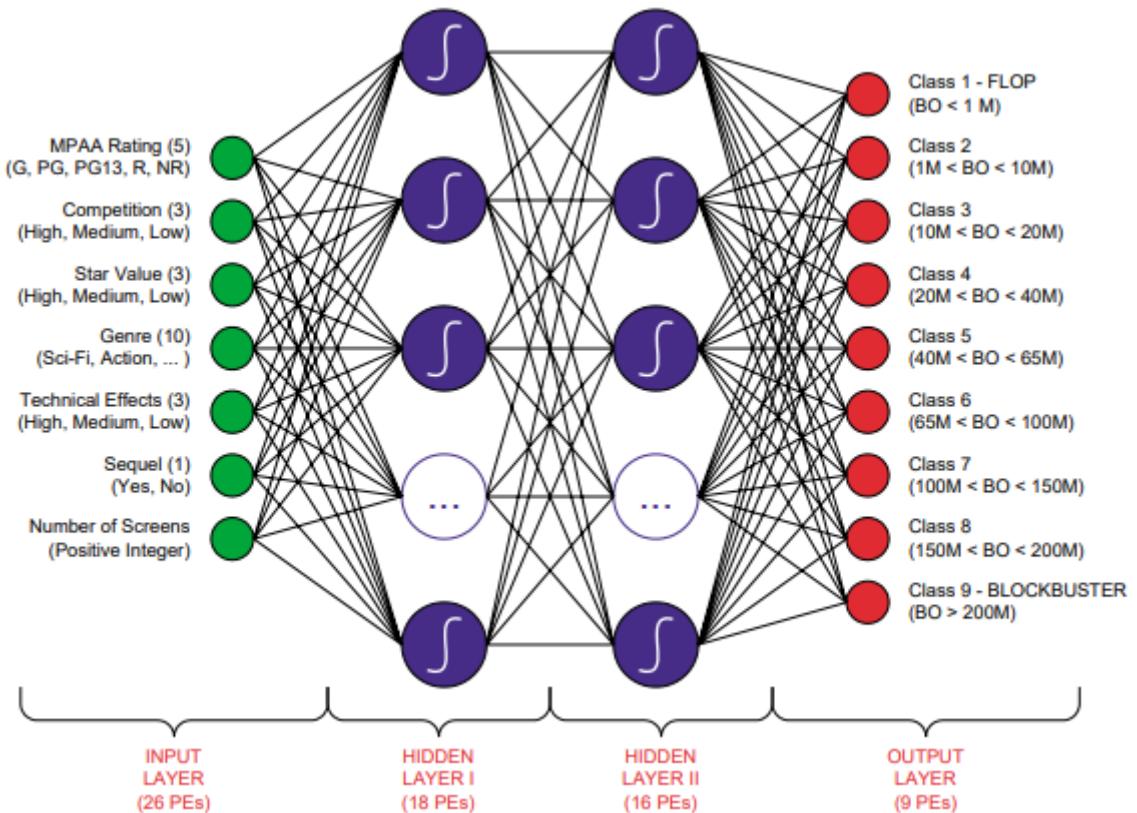


Рисунок 1.12 – Нейронная сеть R. Sharda и D. Delen

В 2009 году W. Chang и K.J. Lee предложили использовать байесовскую сеть убеждений для выявления причинно-следственных связей при прогнозировании кассового успеха корейских фильмов [19]. В качестве показателя успеха было выбрано количество зрителей, разделенных на группы. По сравнению с нейронными сетями и деревьями решений этот подход оказался более точным.

Также были проведены дальнейшие исследования с целью выявления новых причинно-следственных связей между кассовым успехом фильмов и различными переменными. Варианты зависимости активности пользователей в социальных сетях, таких как Twitter [68] и YouTube [71], от рейтинга и популярности фильма

были предложены исследователями A. Oghina, M. Breuss и др. в 2012 году в работе [45]. Кроме того, исследователи провели анализ настроений комментариев пользователей Twitter [68] и их причинно-следственной связи с рейтингом фильма на платформе IMDb [59].

В работе [47] 2016 года исследователи T. Rhee и F. Zulkernine использовали нейронную машину из 1 входного слоя, состоящего из 14 нейронов, 1 скрытого слоя, состоящего из 25 нейронов и 1 выходного слоя, состоящего из 2 нейронов. Входные данные были выбраны следующие:

- Рейтинг актеров;
- Рейтинг режиссера;
- Конкуренция в период релиза;
- Месяц выхода фильма;
- Возрастное ограничение;
- Наличие франшизы;
- Бюджет фильма;
- Рейтинг с сайта «IMDb»;
- Количество положительных оценок с сайта «IMDb»;
- Рейтинг пользователей с сайта «Metascore» [62];
- Рейтинг с сайта «Rotten Tomatoes» [66];
- Рейтинг критиков с сайта «Rotten Tomatoes»;
- Оценка обзоров пользователей с сайта «Rotten Tomatoes»;
- Рейтинг пользователей с сайта «Rotten Tomatoes».

Так как выходными данными был бинарный ответ в виде Успех/Провал, то нейронной машине удалось достичь точности 94%.

Исследователи M. Galvão, R. Henriques в работе 2018 года [24] расширили набор входных параметров, выбранный ими набор включает в себя:

- Наличие франшизы;
- Возрастное ограничение;
- Жанр фильма;
- Бюджет;
- Престижные награды у съемочной группы;

- «Звездный» режиссер;
- Наличие звезд;
- Сезон выхода;
- Оценка критиков.

Процент ошибки, построенной исследователями нейронной сети, составил 40%.

В работе [12] 2017 года Л.Н. Ясницким, Н.О. Белобородовой и Е.Ю. Медведевой была построена нейронная система, на основе персептрана, состоящую из 1 входного слоя, включающего 20 нейронов, 1 скрытого слоя из 6 нейронов и 1 выходного слоя из 1 нейрона, нейронная сеть смогла достичь результата 13,8% погрешности. Входные параметры нейронной сети:

- Год выпуска фильма;
- Страна производитель;
- Пол режиссера;
- Основа сценария;
- Бюджет фильма;
- Возрастное ограничение;
- Наличие вымышленных персонажей;
- Наличие в фильме злодея;
- Длительность фильма;
- Наличие у режиссера успешных киноработ;
- Возраст режиссера на момент создания фильма;
- Наличие у режиссера премий «Оскар»;
- Наличие у режиссера премии «Золотая малина»;
- Наличие у режиссера номинаций на премии «Оскар» и/или «Золотой глобус»;
- Наличие у актеров номинаций на премии «Оскар» и/или «Золотой глобус»
- Жанр фильма;
- Является ли фильм приключенческим;
- Является ли фильм фантастическим;
- Наличие франшизы;

- Является ли фильм частью трилогии.

В работе [13] 2017 года Л.Н. Ясницкий и Д.И. Плотников использовали нейронную систему, на основе персептрана, включающую в себя 1 входной слой из 11 нейронов, 1 скрытый слой из 2 нейронов и 1 выходной слой из 1 нейрона. Результат погрешности данной нейронной машины составил 11%. Обучение происходило на выборке из 120 фильмов. Для обучения использовались следующие входные параметры:

- Бюджет фильма;
- Год выпуска фильма;
- Произведение, ставшее основой сценария;
- Возрастное ограничение;
- Наличие главного героя, заметно превосходящего в силе стальных
- Наличие номинаций у актеров;
- Доминирующий жанр в фильме;
- Номинации за лучший фильм (в своём жанре);
- Наличие у режиссера номинаций;
- Наличие спецэффектов;
- Наличие франшизы.

В работе [56] 2019 года Л.Н. Ясницкий, И.А. Метрофанов и др. разработали нейронную машину, на основе персептрана, включающую в себя 1 входной слой из 56 нейронов, 1 скрытый слой из 13 нейронов и 1 выходной слой из 1 нейрона, обучение производилось на выборке из 4329 фильмов. Погрешность нейронной машины составила 9.17%. Входные данные нейронной сети:

- Год выпуска;
- Страна выпуска;
- Количество наград «Оскар» у съемочной группы;
- Жанр фильма;
- Бюджет фильма;
- Длительность фильма.

На основе анализа вышеперечисленных работ выделены следующие входные данные, которые будут рассмотрены и отсортированы во 2 главе данной работы:

- Год выпуска фильма;

- Бюджет фильма;
- Страна производитель;
- Доминирующий жанр фильма (Является ли фильм приключенческим, является ли фильм фантастическим);
- Длительность фильма;
- Возрастное ограничение;
- Произведение, ставшее основой сценария;
- Наличие франшизы (Является ли фильм частью трилогии);
- Критические обзоры (Оценка критиков);
- Наличие звезд;
- Конкуренция в период релиза;
- Качество съемки;
- Количество премьерных показов;
- Количество наград у съемочной группы (Наличие премий);
- Рейтинг актеров;
- Возраст режиссера на момент создания фильма;
- Рейтинг режиссера («Звездный» режиссер, Наличие у режиссера успешных киноработ);
- Количество внушительных наград у режиссера (Наличие у режиссера премий «Оскар», Наличие у режиссера премии «Золотая малина», Наличие у режиссера номинаций на премии «Оскар» и/или «Золотой глобус»);
- Количество внушительных наград у актеров (Наличие у актеров номинаций на премии «Оскар» и/или «Золотой глобус»);
- Сезон выхода (Месяц выхода фильма);
- Пол режиссера;
- Наличие вымышленных персонажей;
- Наличие в фильме злодея;
- Наличие главного героя, заметно превосходящего в силе стальных;
- Наличие спецэффектов;
- Рейтинг с сайта «IMDb» [59];

- Количество положительных оценок с сайта «IMDb» [59];
- Рейтинг пользователей с сайта «Metascore» [62];
- Рейтинг с сайта «Rotten Tomatoes» [66];
- Рейтинг критиков с сайта «Rotten Tomatoes»;
- Оценка обзоров пользователей с сайта «Rotten Tomatoes»;
- Рейтинг пользователей с сайта «Rotten Tomatoes».

1.4 Обзор существующих программных решений

В данном разделе необходимо провести аналитический обзор существующих программных решений задач данной дипломной работы, выделить их недостатки и достоинства, и на основе этого анализа выдвинуть требования к разрабатываемой программе, обеспечивающие её конкурентоспособность на рынке.

На момент написания аналитического обзора, данной дипломной работы, после сравнительного анализа предыдущих исследований в выбранной предметной области, было выявлено отсутствие общедоступных программных решений для прогнозирования кассовых сборов фильма. Единственным, найденным решением, является разработанное веб приложение, в рамках исследования схожей темы, а именно «Прогнозирование кассовых сборов фильма» в «Пермской научной школе искусственного интеллекта». Изображение данного решения представлено на рисунке 1.13.

Прогноз кассовых сборов фильма

Бюджет фильма (доллары США):
Возраст, от которого допускается просмотр фильма (согласно российской возрастной классификации информационной продукции):
Продолжительность фильма (минут):
Наличие у режиссера номинаций/наград на престижные кинопремии («Оскар», «Золотой глобус», SAAG, Critics' Choice Awards):
Наличие у актеров номинаций/наград на престижные кинопремии («Оскар», «Золотой глобус», SAAG, Critics' Choice Awards):
Основной жанр:
Наличие спецэффектов:
Является ли фильм продолжением какого-либо фильма (т.е. является ли фильм следующей серией франшизы – сиквелом):
Выход фильма в период высокой посещаемости кинотеатров (новогодние праздники, летние каникулы и т.д.):

© 2022 - Нейросетевые проекты

Рисунок 1.13 – Веб приложение для прогнозирования кассовых сборов фильмов

Данная программа предоставляет небольшой функционал в виде: задача – ответ, и позволяет пользователю вводить и выбирать параметры кинокартины, для

выведения результата. Результатом работы приложения является кассовые сборы в приблизительном диапазоне (Рисунок 1.14).

Сборы фильма в мире, в долларах США

от 1 000 000 000 до 1 499 999 999

[Изменить введенные данные](#)
[Попробовать с новыми данными](#)
[Перейти к списку проектов](#)

© 2022 - Нейросетевые проекты

Рисунок 1.14 – Результат работы веб приложения

Из преимуществ стоит выделить удобство использования и легкодоступный и простой дизайн приложения. К недостаткам стоит отнести нехватку входных данных, отсутствие решений для улучшения результата кассовых сборов кинокартины, невозможность ввода пустых значений где это возможно.

«Нейросимулятор 5.0» – в программа-симулятор разработанная и зарегистрированная Ф.М. Черепановым и Л.Н. Ясницким в 2014 году [11]. Программа предназначена для создания небольших нейронных сетей для решения любых задач, таким образом программа лишь косвенно подходит под выбранную предметную область.

Интерфейс программы (Рисунок 1.15) является узконаправленным и сложным для использования среднестатистическим пользователем. В окне программы представлена возможность выбора необходимого количества скрытых слоев нейронной сети и нейронов в каждом слое, а так же внесение необходимых данных для обучения и тестирования нейронной системы.

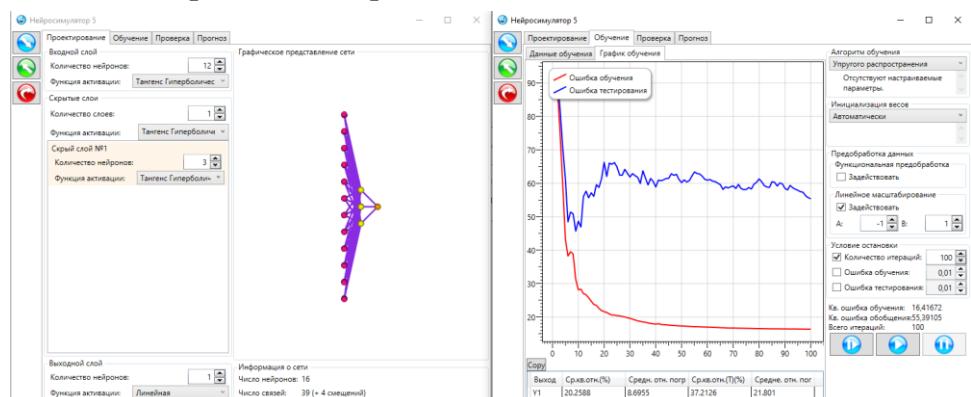


Рисунок 1.15 – Интерфейс программы «Нейросимулятор 5.0»

Внесение данных производится путем экспорта базы данных из файла «Excel», после чего производится обучение нейронной сети на основе этих данных. Нейронная сеть программы представлена в виде перцептрана, а обучение производится на основе «Обучения с учителем».

Результатом работы программы является нейронная сеть, на которой можно проводить проверку данных путем импорта/экспорта данных из/в файл «Excel». Так же программа предоставляет возможность отобразить функциональную зависимость ответа от входных данных (Рисунок 1.16).

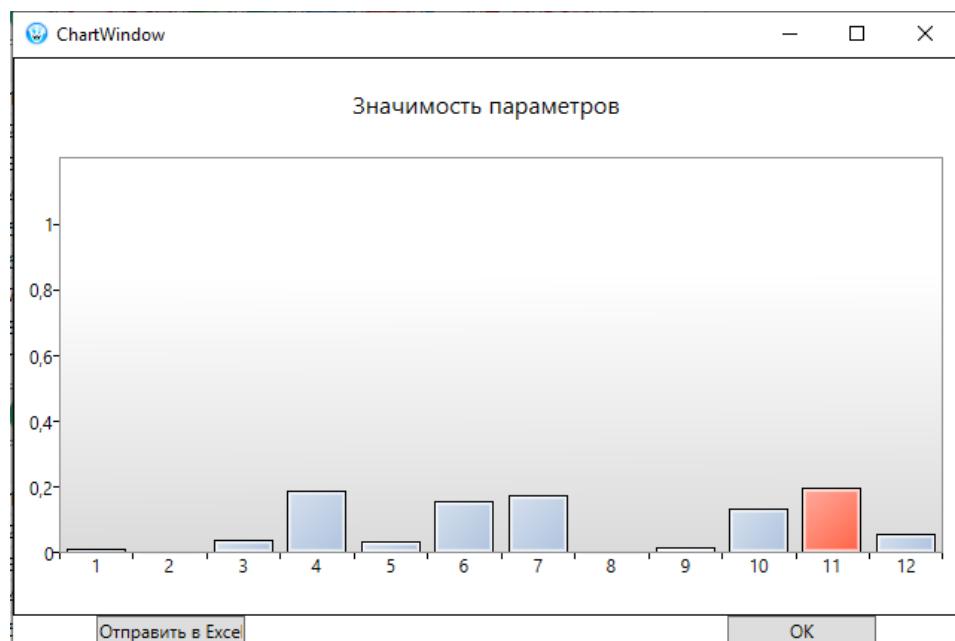


Рисунок 1.16 – Результат отображения значимости параметров

Преимущества данной программы состоят в функционале и гибкости системы, с возможностью подстраивать нейронную сеть программы под разные задачи. К минусам следует отнести сложность программы, как в виде интерфейса так и в виде обучения самой нейронной системы.

Так же стоит учесть что данная программа полноценно не подходит под аналог разрабатываемой системы, но подходит для первичного обучения и тестирования входного множества и нейронной сети.

1.5 Формирование требований к разрабатываемой системе

В результате аналитического обзора предметной области: анализа процесса создания и продвижения кинокартин и анализа особенностей строения и обучения нейронных систем, можно сделать выводы о необходимости создания системы «Прогнозирование рентабельности кинобизнеса» в связи с отсутствием наличия программ для решения задачи, также в результате были сформулированы требования к разрабатываемой системе с точки зрения программной части.

Реализация программного продукта для прогнозирования рентабельности кинобизнеса будет проводится в несколько этапов:

1. Сбор данных для обучения.
2. Подготовка и нормализация данных.
3. Выбор топологии сети.
4. Экспериментальный подбор характеристик сети.
5. Экспериментальный подбор параметров обучения.
6. Тестовое обучение в программе «Нейросимулятор 5.0» [11].
7. Обучение с использованием языка программирования.
8. Проверка адекватности обучения.
9. Корректировка параметров, окончательное обучение.
10. Вербализация нейронной сети с целью дальнейшего использования.

Для чего необходимо разработать следующие модули:

- Модуль сбора данных обучающего множества (этап сбора данных).
- Модуль очистки, систематизации и нормализации данных (этап подготовки и нормализации данных).
- Модуль нейронной сети (этап обучения, этап проверки адекватности обучения, этап корректировки параметров).
- Пользовательский модуль программного продукта (этап вербализации нейронной сети).

Требования к модулю сбора данных обучающего множества:

- Собираемые данные должны быть репрезентативными.
- Собираемые данные должны быть непротиворечивыми.
- Собираемые данные должны подходить под критерии алгоритма обучения с учителем.

- Данные должны собираться автоматически.
- Данные должны собираться с сайтов-агрегаторов (п. 1.1 текущей главы).
- Набор данных должен включать и/или заменять данные представленные в работе (п. 1.3 текущей главы).
- Модуль должен иметь возможность повторного использования.

Требования к модулю очистки, систематизации и нормализации данных:

- Данные в наборе должны быть нормализованы в рамках: 0-1000.
- Данные не должны содержать выбросов.
- Данные не должны содержать пустых строк и значений.
- Данные не должны быть отсортированы по порядку вне зависимости от критерия.
- Данные должны проходить очистку несколько раз, для сужения границ выбросов.
- Очистка, систематизация и нормализация данных должна производится автоматически.
- Модуль должен иметь возможность на модификацию.
- Модуль должен иметь возможность повторного использования.

Требования к модулю нейронной сети

- Нейронная сеть должна иметь архитектуру перцептрона или многослойного перцептрона.
- Обучение должно производится по критериям алгоритма обучения с учителем.
- Модуль должен иметь возможность повторного обучения.
- Модуль должен иметь возможность корректировки параметров.
- Результат обучения должен примерно совпадать с результатом тестового обучения в программе «Нейросимулятор 5.0».
- Ошибка обучения не должна превышать 40%.

Требования к пользовательскому модулю

- Интерфейс программы должен содержать поля для ввода данных пользователем.
- Модуль должен работать на обученной нейронной сети.

- При срабатывании программы должен выводится результат.
- Модулю должен передавать данные пользователя обученной нейронной сети.

Входные данные:

- Для модуля очистки и модуля нейронной сети:
 - Собранные данные модулем сбора данных в одном из форматов: «XSLX», «XLS», «CSV», «TXT».
- Для пользовательского модуля
 - Обученная нейронная сеть в формате «HDF5».
 - Текстовые данные, введенные и/или выбранные пользователем.
 - Измененные текстовые данные введенные пользователем (для последующего обзора).

Выходные данные:

- Для модуля очистки:
 - Файл в одном из форматов: «XSLX», «XLS», «CSV», «TXT».
- Для модуля нейронной сети:
 - Файл в формате «HDF5».
- Для пользовательского модуля:
 - Данные отображаемые пользователю:
 - Результат в формате: успешно/не успешно.
 - Числовые данные о примерных кассовых сборах.
 - Текстовые данные с анализом входных данных.
 - Диаграмма кассовых сборов от изменения данных.

В результате описания требований к разрабатываемой системы был сформирован список детализированных и уточненных требований, представленный в виде технического задания в «ПРИЛОЖЕНИЕ D». Техническое задание выполнено в соответствии с требованиями ГОСТ 34.602-89 [15].

Так же была разработана диаграмма прецедентов в нотации UML [65] (Рисунок 1.17). В диаграмме присутствует 2 актора: пользователь, который имеет доступ пользовательскому интерфейсу и может вводить данные и нейронная сеть, которая принимает введенные данные пользователем, обрабатывает их и выводит результат. В диаграмме также присутствует артефакт «Обученная выборка», как файл, с которым взаимодействует нейронная сеть для корректной работы внутри системы. Описание прецедентов в полном виде представлено в «ПРИЛОЖЕНИЕ Е».

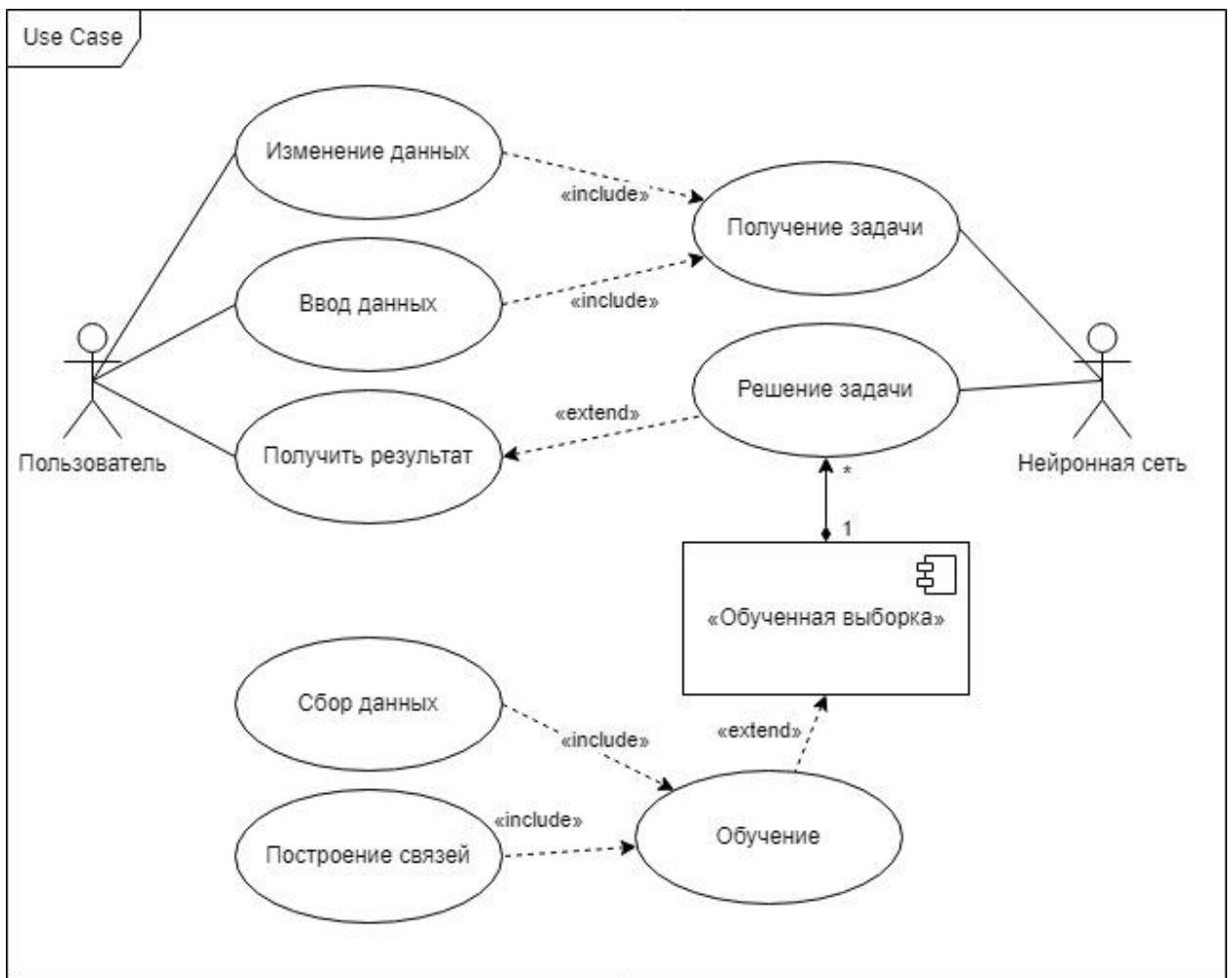


Рисунок 1.17 – Диаграмма вариантов использования

1.6 Выводы по первой главе

В результате аналитического обзора был приведен подробный анализ предметной области и решены задачи, поставленные в начале главы:

1. Определены особенности технического процесса кинобизнеса и взаимосвязей результатов кассовых сборов от переменных. Определены методов и ресурсов сбора данных о кинокартинах (п. 1.1 текущей главы).
2. Проанализированы существующие архитектуры построения нейронных сетей (п. 1.2.1 текущей главы). и создана их сравнительная таблица «см. ПРИЛОЖЕНИЕ В» а также утверждена необходимая архитектура для создания разрабатываемой системы. Проанализированы «см. ПРИЛОЖЕНИЕ С» и выбраны алгоритмы машинного обучения нейронной сети (п. 0 текущей главы). для обучения и тестирования разрабатываемой системы.
3. Рассмотрены предыдущие работы и исследования в выбранной предметной области и составлен набор входных параметров для обучения нейронной сети (п. 1.3 текущей главы).
4. Рассмотрены существующие программные решения (п. 1.4 текущей главы).
5. Сформированы и детализированы требования к разрабатываемой системе (п. 1.5 текущей главы). и сформировано техническое задание «см. ПРИЛОЖЕНИЕ D», также построена диаграмма прецедентов (Рисунок 1.17) и описаны ее прецеденты «см. ПРИЛОЖЕНИЕ Е».

В результате работы над главой сформировано направление для решения задачи «Прогнозирование рентабельности кинобизнеса».

Глава 2 Проектирование разрабатываемой системы

В данной главе описывается процесс проектирования разрабатываемой системы, охватывающий планирование взаимодействия пользователя с программой путём построения диаграмм активности и последовательности, выбор программных структур для реализации нейронной сети, а также проектирование непосредственно внутренней архитектуры системы путём создания диаграмм классов и компонентов. В главе решаются следующие задачи:

1. Описание внутрипрограммного взаимодействия системы и ее компонентов, основываясь на выдвинутых требованиях в предыдущей главе, с помощью диаграмм последовательностей и диаграмм компонентов.
2. Проектирование алгоритмов сбора данных для обучения, опираясь на сформированный, в предыдущей главе, список входных параметров, путем разбора каждого выбранного параметра, нахождения их сильных и слабых сторон и определения необходимости его использования.
3. Проектирование нейронной сети путем тестового построения ее архитектуры и тестового обучения в программе «Нейросимулятор 5.0» [11], а также корректировки ее параметров, с последующим переносом на язык программирования.
4. Проектирование графического интерфейса пользовательского приложения и описание взаимодействия с ним.

Итогом второй главы является спроектированная схема взаимодействия пользователя и разрабатываемой системы и архитектура самого программного решения и нейронной сети.

2.1 Описание поведения системы

Первоначальным этапом проектирования системы является описания поведения системы и взаимодействия с ней. Основываясь на диаграмме прецедентов и выделенных, в предыдущей главе, модулей системы можно описать работу системы и взаимодействие пользователя с ней с помощью диаграммы последовательности (Рисунок 2.1) в нотации «UML» [65]. Взаимодействие компонентов системы описывается с помощью диаграммы компонентов (Рисунок 2.2) в той же нотации.

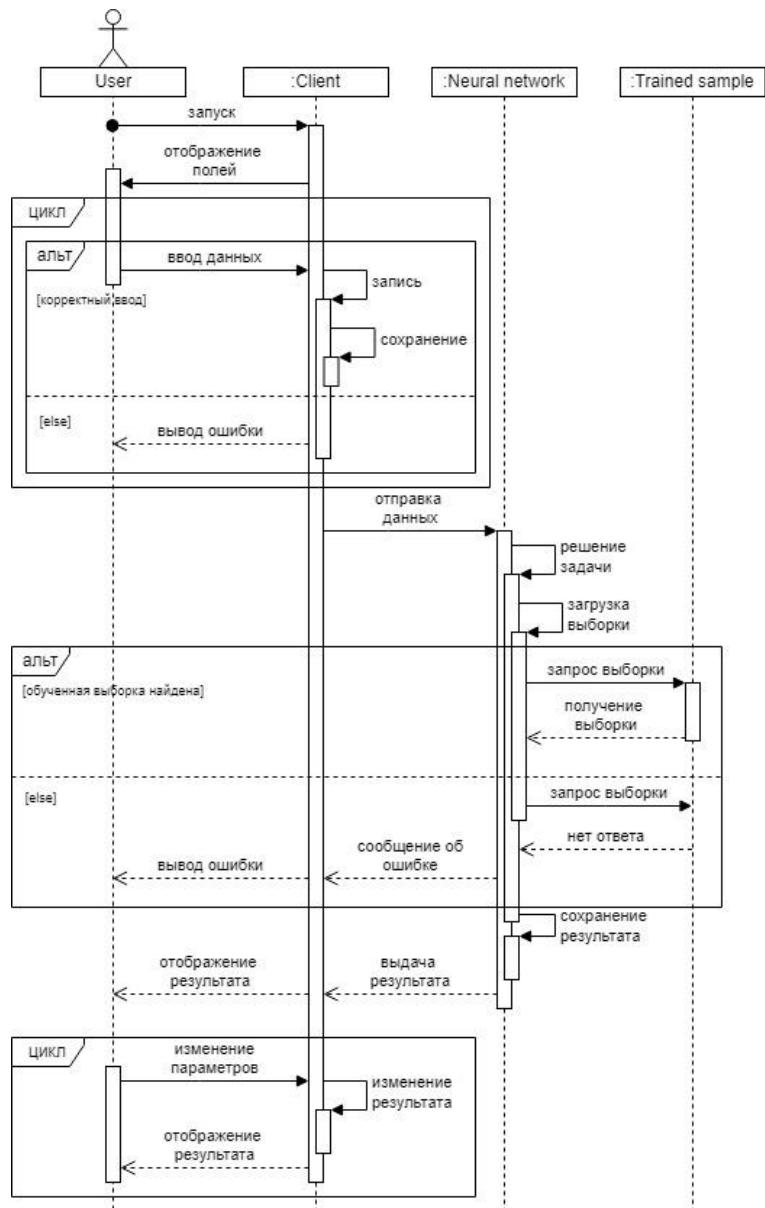


Рисунок 2.1 – Диаграмма последовательности

Таким образом система соответствует минимальным требованиям функционала в рамках разработки «MVP» системы. Взаимодействие с системой полностью происходит через клиентское приложение, в котором пользователь

вводит данные для определения прогнозируемых кассовых сборов кинокартины и получает результат проведенного прогнозирования нейронной сетью. На основании введенных данных, система создает файл для его обработки нейронной сетью и обрабатывает его, подгружая обученную выборку с весами входных параметров, при их наличии.

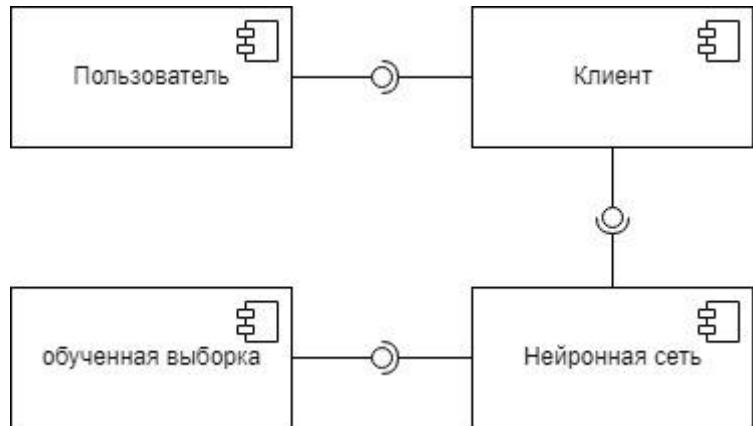


Рисунок 2.2 – Диаграмма компонентов

Так же была построена диаграмма последовательности для системы, на этапе его обучения «см. ПРИЛОЖЕНИЕ F». Как видно из диаграммы, обучение производится в несколько, идущих последовательно, этапов, результатом чего является обученная выборка.

2.2 Проектирование модуля сбора и систематизации обучающих данных

Последующим этапом процесса проектирования разрабатываемой системы является проектирование алгоритма сбора обучающих данных. Таким образом необходимо рассмотреть каждый входной параметр из собранного списка (п. 1.3 первой главы) и выявить необходимость его включения. Также необходимо спроектировать возможности очистки и нормализации данных.

Год выпуска является первым выбранным параметром, данный параметр отображает состояние кинобизнеса в течении года, так как год выпуска кинокартины может сыграть огромную роль для кассовых сборов каждой, отдельно выбранной картины, то несомненно данный параметр является весомым для обучения. Однако так как разрабатываемая система предназначена для предсказания успеха, то и использоваться будет на ранних этапах кинопроизводства, соответственно обучающие данные должны содержать кинокартины того же года, что и

прогнозируема кинокартина, что является невозможным, тем более если дата выхода фильма планируется только через несколько лет от начала старта съемок. Соответственно данный параметр является не актуальным.

Бюджет фильма – необходимый параметр, так как от данного параметра напрямую зависят сборы кинофильма. Так как численные значения параметра намного превышают значения других параметров, то эти значения нуждаются в нормализации, таким образом бюджет в 1000000 должен быть нормализован до 1 (млн.). Также необходимо избавиться от выбросов с помощью диаграммы размаха. Сбор параметра, может производится как с сайта «IMDb» [59], так и с сайта «Box Office Mojo» [58], из-за использования одного источника.

Страна производитель – так как в некоторых странах культура кинопроизводства развита больше чем в других, то, соответственно, от страны, в которой был снят фильм, достаточно сильно зависит итоговая прибыль, поэтому данный параметр также стоит включить в данные для обучения. Из-за того, что название страны является строковыми данными, то необходимо закодировать названия страны, с нормализацией, например, 1-Россия, 2-США и т.д. К сожалению данный параметр не подойдет для статистического анализа.

Доминирующий жанр фильма – важный параметр, потому что некоторые кинолюбители предпочитаю один жанр другому. Так же, как и со страной, необходимо закодировать наиболее популярные жанры. Параметр можно использовать для статистического анализа.

Длительность фильма – параметр средней важности, так как большинство кинокартин имеют примерно схожую продолжительность, из-за чего включение данного параметра может как навредить обучению, так и помочь ему. Длительность будет переводиться в минуты, для записи в обучающее множество. Также параметр можно использовать для статистического анализа.

Возрастное ограничение – отображает с какого возраста доступен просмотр кинокартины. Из-за низкого возрастного ограничения повышается посещаемость семей с детьми и, соответственно, увеличиваются продажи, но в то же время уменьшается посещаемость взрослых, без детей, что может оказаться негативно на кассовых сборах, потому параметр стоит включить в обучающую выборку. Также параметр можно использовать для статистического анализа. Так как не все

кинокартины ограничиваются по правилам российский возрастных ограничений, то строковые данный другого формата (PG, G, R и т.д.) будут закодированы и приравнены к возрастным ограничениям, принятым в России.

Произведение, ставшее основой сценария – параметр отображает, на основании какого произведения, игры или истории была снята кинокартина. Таким образом кинокартины, снятые по популярной книге, ожидаю соберут больше продаж, чем по никому не известной книге. Данные будут закодированы в зависимости от источника. Сбор параметра, может производится с сайта «IMDb» [59].

Наличие франшизы – наличие успешной первой части, несомненно поднимут шансы собрать больше кассовых сборов, поэтому данный параметр необходим для включения в обучение.

Критические обзоры – несомненно, параметр значительно влияет на кассовые сборы, но, из-за того, что не всегда критики в основном не пишут обзоры задолго до выхода фильма, то и включение данного параметра не является необходимостью, данный параметр возможен к рассмотрению при модификации разрабатываемого прототипа, до полноценного приложения.

Наличие звезд – отображает значимость актеров, снимающихся в кинокартинах, но из-за неоднозначности понятия «Звезда», сложно определить, как нужно оценивать данный параметр, соответственно параметр не будет включен в обучающую выборку и будет заменен рейтингом актеров.

Конкуренция в период релиза – параметр, отображающий количество фильмов, особенно от мировых кинокомпаний, которые будут выходить в месяц выпуска фильма. Так как данный параметр является сложным для определения, то он не будет входить в обучающую выборку, но возможен при модификации системы.

Качество съемки – критерий, имеющий малую значимость, так как в современном кинематографе приняты стандарты качества съемки, из-за чего качество кинокартин приблизительно одинаковое и равняется широкоэкранным 4K (4096x1716 пикселей) с соотношением 2.39:1. Параметр не будет включен в обучающую выборку.

Количество премьерных показов – от количества премьерных показов может зависеть примерный ажиотаж на покупку билетов, что позволяет спрогнозировать количество проданных билетов в первую неделю проката. Параметр сложен для сбора и рекомендуется добавить параметр при модификации системы.

Количество наград у съемочной группы – отображает совокупное количество наград у всей съемочной группы. Сбор параметра возможен с сайта «IMDb» [59].

Рейтинг актеров – параметр, отображает среднее значение рейтингов предыдущих работ у актеров. Параметр является сложным для записи, из-за большого количества актеров на одной кинокартине, возможно уменьшение количества актеров, до 3 самых значимых, но в таком случае параметр будет являться необъективным. Параметр не будет использоваться для обучения.

Возраст режиссера на момент создания фильма – неоднозначный параметр, у которого сложно проследить влияние на обучение. Параметр не будет входить в обучающую выборку, но возможен к рассмотрению при модификации системы.

Рейтинг режиссера – как и с рейтингом актеров, параметр, отображает среднее значение рейтингов предыдущих работ у режиссера. В данном случае параметр объективен, из-за наличия всего одного режиссера на кинокартину и соответственно может выступать в качестве параметра для обучения. Сбор параметра возможен с сайта «IMDb».

Количество внушительных наград у режиссера (Наличие у режиссера премий «Оскар», Наличие у режиссера премии «Золотая малина», Наличие у режиссера номинаций на премии «Оскар» и/или «Золотой глобус») – параметр, отображающий наличие перечисленных выше наград у режиссера. Данные награды могут оказать сильное влияние на оценочное суждение смотрящих и, соответственно, увеличить кассовые сборы. Параметр является суммой всех полученных наград, номинации на эти награды будут суммированы в соотношении 4 номинации как 1 награда. Сбор параметра возможен с сайта «IMDb».

Количество внушительных наград у актеров (Наличие у актеров номинаций на премии «Оскар» и/или «Золотой глобус») – аналогично «Количество

наград у режиссера», но для актеров. Как и с «Рейтинг актеров», параметр не будет включен в обучение, но возможен к рассмотрению при модификации системы.

Сезон выхода (Месяц выхода фильма) – от параметра может зависеть посещаемость фильма, так как летом посещаемость возрастает, а весной наоборот падает, соответственно параметр должен оказать сильное влияние на результат. Также возможно выделение отдельного параметра со значением выхода фильма в праздничные дни. Значения параметра будут закодированы в соответствии с порядковым номером месяца, где зима – 1. Сбор параметра возможен с сайта «IMDb» [59].

Пол режиссера – влияние данного параметра сложно предсказать, поэтому при разработке «MVP» системы, параметр не будет применяться, но возможен для статистического исследования и применения при модификации системы.

Наличие вымышленных персонажей – параметр может быть заменен другим, а точнее жанром «Фантастика», так как в других жанрах крайне низок шанс появления вымышленных персонажей, по данной причине параметр не будет входить в обучающее множество.

Наличие в фильме злодея – наличие злодея является клише, присутствующее в наибольшей части кинофильма, параметр возможен к замене на «Наличие в фильме антигероя», что только начало обретать популярность. Параметр не будет применяться при обучении, но возможен к рассмотрению при модификации системы.

Наличие главного героя, заметно превосходящего в силе стальных – как и «Наличие вымышленных персонажей», параметр может быть заменен жанром «Фантастика», соответственно, не будет входить в обучающее множество.

Наличие спецэффектов – параметр, имеющий не однозначное влияние. В современном мире, крайне мало кинокартин не имеют спец эффектов, как минимум потому, что большинство сцен записываются в павильонах с зеленым фоном, который после заменяется на полноценное окружение, в данном случае, авторы работы [13], возможно имели в виду наличие спец эффектов, не свойственных реальности (эффекты космоса, лазеры и т.д.) из-за чего параметр можно заменить жанром «Фантастика». В рамках разработки «MVP» системы параметр сложен к

сбору и внесению в обучающую выборку, потому не будет собираться, но возможен к рассмотрению при модификации.

Рейтинг с сайта «IMDb» [59] – как и последующие параметры: Количество положительных оценок с сайта «IMDb»; Рейтинг пользователей с сайта «Metascore» [62]; Рейтинг с сайта «Rotten Tomatoes» [66]; Рейтинг критиков с сайта «Rotten Tomatoes»; Оценка обзоров пользователей с сайта «Rotten Tomatoes»; Рейтинг пользователей с сайта «Rotten Tomatoes», сложны или не являются возможными для сбора на ранних стадиях кинопроизводства, так как публикуются не за долго до премьера картины. Также параметр является сложным для сбора в рамках реализации «MVP» системы и соответственно не будет включен в обучение, но возможен к рассмотрению при модификации системы.

В результате описания всех вышеперечисленных входных параметров, собранных из работ (п. 1.3 первой главы), был составлен список тех которые будут использоваться для обучения и представлен в таблице 2.1. Выходным параметром будут являться кассовые сборы кинокартины.

Таблица 2.1 – Входные параметры обучения

№	Параметр	Изначальные значения	Кодировка
X1	Бюджет	100000 – 100000000	1 – 1000
X2	Продолжительность	50 мин – 5 ч	50 – 300
X3	Страна	США; Канада; Россия; Англия; Франция; Италия; Китай; Индия;	1 – 8
X4	Жанр	Боевик; Приключение; Драма; Комедия; Криминальный; Мистика; Ужасы; Вестерн; Исторический; Биография; Анимация; Фантастика; Триллер; Мюзикл; Нуар	1 – 14
X5	Возрастное ограничение	0+; 6+; 12+; 16+; 18+; G; PG; PG-13; R; NC-17	1 – 5
X6	Основа сценария	Книга; Бестселлер; История; Игра	1 – 4
X7	Франшиза	Да; Нет	0 – 1
X8	Сезон выхода	Зима; Весна; Лето; Осень	1 – 4
X9	Период высокой посещаемости	Да; Нет	0 – 1
X10	Рейтинг режиссера	0 – 10	0 – 10
X11	Наличие у режиссера престижных наград	Да; Нет	0 – 1

№	Параметр	Изначальные значения	Кодировка
X12	Наличие у сценаристов престижных наград	Да; Нет	0 – 1
X13	Наличие у 3-х актеров на главных ролях престижных наград	Да; Нет	0 – 1
X14	Суммарное количество «оскаров» у съемочной группы	0 – 20	0 – 20

В рамках разработки «MVP» системы, являющейся минимальной тестовой версией полноценного продукта, для ускорения процесса обучения и сбора обучающих данных были выбраны кинокартини из списка «TOP-1000» и «BOTTOM-1000» по оценкам зрителей сайта «IMDb» [59], пример отображения списков на сайте отображен на рисунке 2.3. Данный набор был выбран основываясь на необходимости в усреднении средних значений всего списка фильмов. Общее количество входящих в обучающую выборку кинокартин уменьшено, в связи с временными рамками разработки системы и отсутствии необходимости полноценного, тщательного обучения в рамках разработки «Минимально жизнеспособного продукта». При увеличении количества данных, сбор данных и обучение нейронной сети будут занимать больше времени, до нескольких дней на один запуск.

IMDb "Bottom 1000" (Sorted by IMDb Rating Descending)



Рисунок 2.3 – Отображение рейтинга «TOP-1000» сайта «IMDb»

Для корректной и более быстрой работы модуля сбора обучающих данных, а также для экономии ресурсов, необходимо разделить модуль на несколько частей, для предотвращения выполнения одной и той же, избыточной, задачи при каждом

запуске. Таким образом модуль будет включать алгоритм первичного сбора данных (скрапинга) и алгоритм сбора обучающих данных.

Алгоритм первичного сбора данных предназначен для сбора ссылок на кинокартинны, так как нет необходимости при каждом запуске заново собирать эту информацию, из-за низкого времени обновления страниц рейтинга. Алгоритм будет принимать на вход страницы с рейтингом, представленные выше и в качестве результата формировать файл со списком ссылок на кинокартинны для последующей обработки. Примерное описание работы алгоритма представлено (Рисунок 2.4) в виде блок-схемы, составленной в соответствии с требованиями ГОСТ 19.701-90 [14].

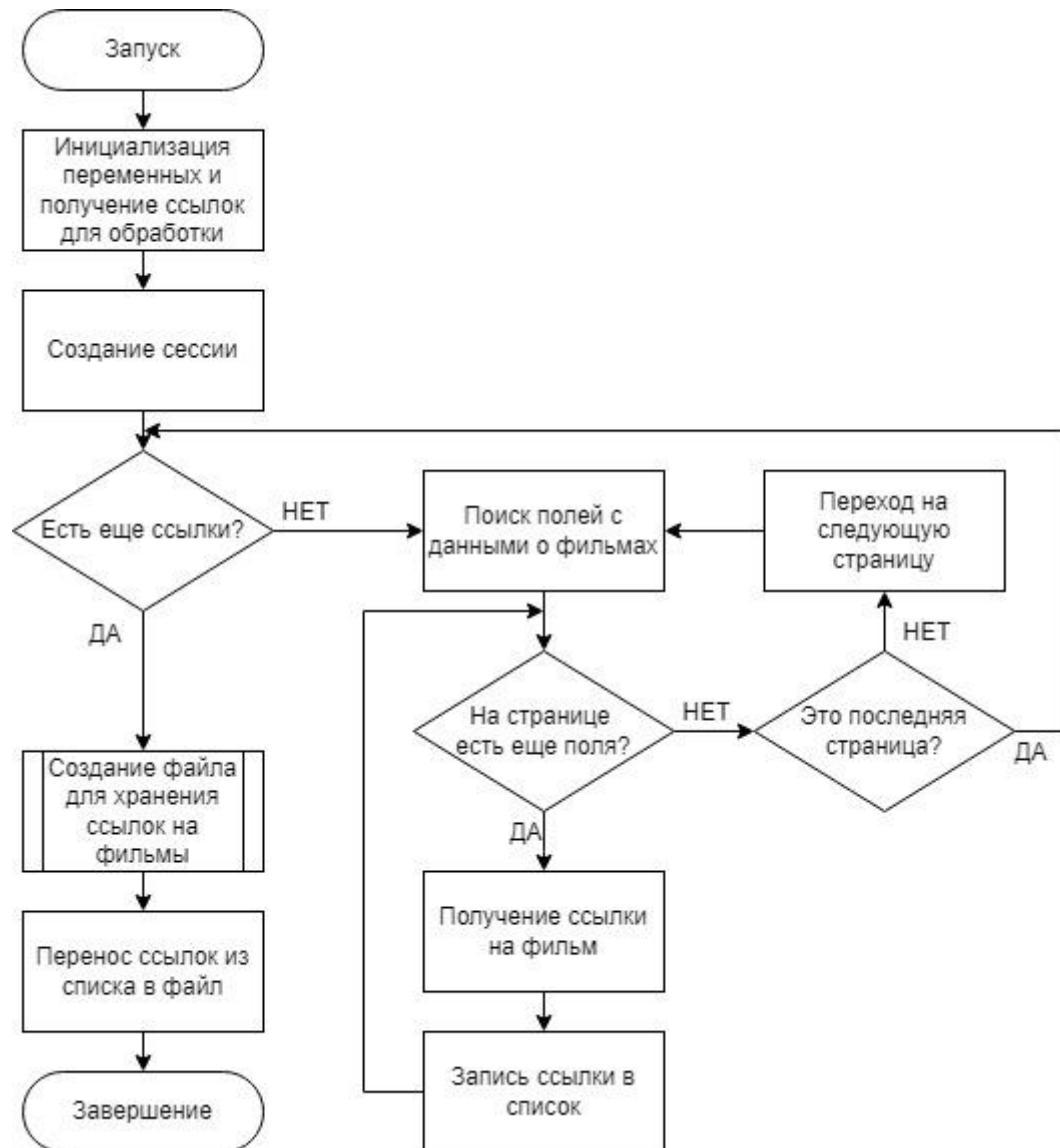


Рисунок 2.4 – Алгоритм первичного сбора данных

Алгоритм сбора обучающих данных будет принимать сформированный предыдущим алгоритмом файл и формировать файл для обучения. Для сбора обучающих данных, в отличии от первичного сбора данных, алгоритм будет более глубоко углубляться в структуру сайта, а также переходить на другие ресурсы сбора данных, для этого алгоритм дополнительно будет разделен на под процессы в соответствии с методологией «ООП». Описание работы алгоритма (Рисунок 2.5) в виде блок-схемы, составленной в соответствии с требованиями ГОСТ 19.701-90 [14]. Под процессы модуля в развертке, представлены в «ПРИЛОЖЕНИИ G».



Рисунок 2.5 – Основной алгоритм сбора обучающих данных

2.3 Проектирование нейронной сети

Для проектирования итоговой нейронной сети, разрабатываемой системы, в соответствии с требованиями (п. 1.5 первой главы) данной работы, необходимо провести тестовое построение нейронной сети в программе «Нейросимулятор 5.0» [11] для поиска оптимальной структуры нейронной сети и подбора критериев обучения на основе результата тестового обучения. Предварительная архитектура, разработанной в программе, нейронной сети представлена на рисунке 2.7.

Построенная нейронная сеть состоит из 3 слоев:

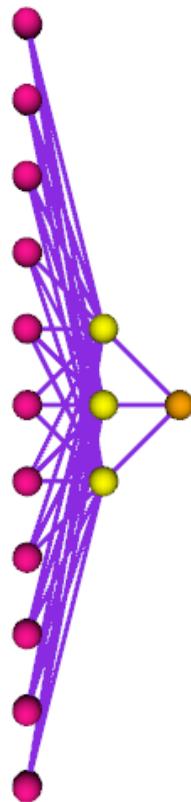


Рисунок 2.7 – Графическое представление нейронной сети

- Входной слой с 11 нейронами. Активационная функция: сигмоида.
- Скрытый слой с 3 нейронами. Активационная функция: сигмоида.
- Выходной слой с 1 нейроном. Активационная функция: линейная.

Для обучения было проведено 700 итераций. Количество построенных связей между нейронами составило: 36 связей.

Результат обучения (Рисунок 2.8):

- Среднее квадратичное отношение – 17.9%
- Средняя относительная погрешность – 14.5%
- Средняя квадратичная относительная ошибка тестирования – 23.0%

- Средняя относительная погрешность тестирования – 18.6%

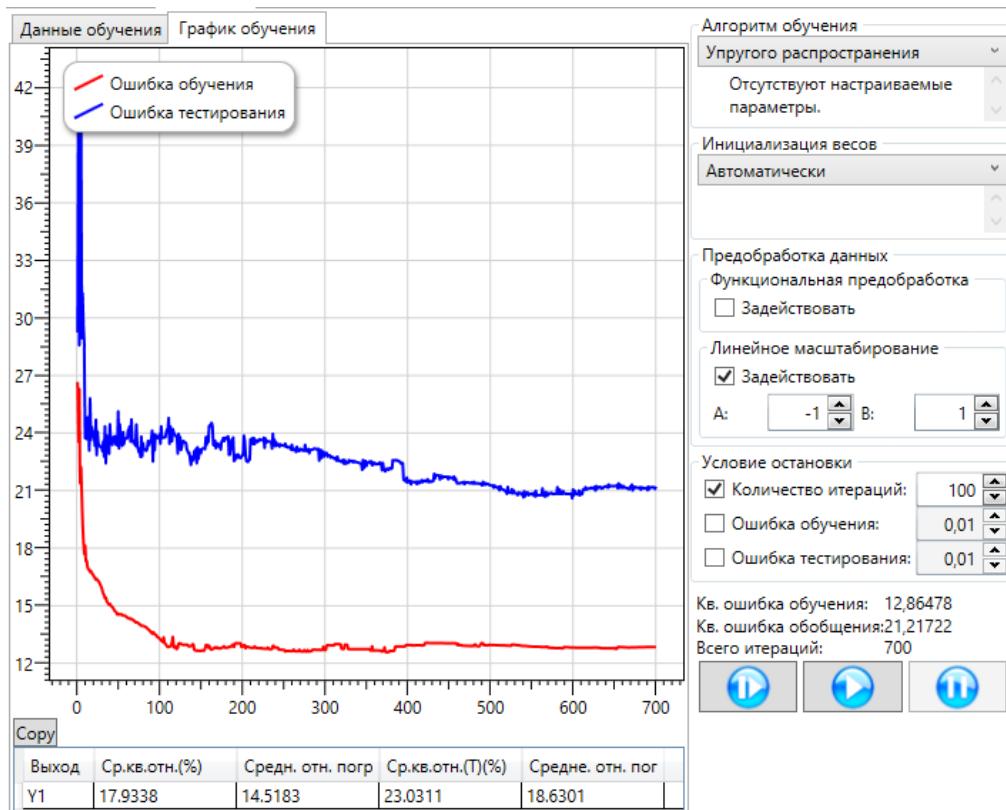


Рисунок 2.8 – Результат тестового обучения

Прогностические свойства сети так же проверялись на примерах тестирующего множества, не участвовавшего в процессе обучения. За функционал качества, характеризующий среднюю ошибку алгоритма на выборке, была выбрана «Средняя квадратичная относительная ошибка тестирования», составившая 23% и вычисляемая по формуле:

$$E = \sqrt{\frac{\sum_{n=1}^N (d_n - y_n)^2}{N}} \cdot 100\% \quad (2.1)$$

где d_n – фактические кассовые сборы n -го фильма,

y_n – прогнозируемая величина кассовых сборов,

N – количество элементов выборки.

Эта же формула будет применяться в конечном варианте нейронной сети, как функционал качества для получения точности искомого результата.

Оптимизационным алгоритмом нейронной сети будет выступать стохастический градиентный спуск – «Adam», так как этот метод «эффективен в вычислительном отношении, требует мало памяти, инвариантен к диагональному

масштабированию градиентов и хорошо подходит для задач, которые являются большими с точки зрения данных/параметров» согласно [35].

На рисунке 2.9 представлена гистограмма, демонстрирующая разницу между фактическими и прогнозируемыми, нейронной сетью, кассовыми сборами, среди 35, случайно отобранных, кинокартин из тестирующего множества.

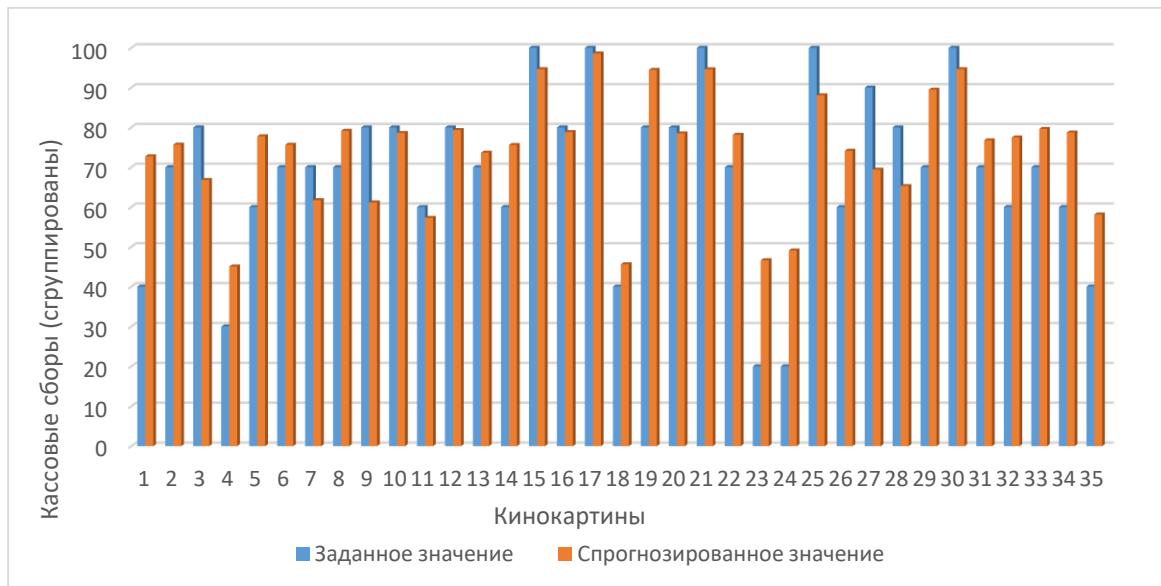


Рисунок 2.9 – Результат работы нейронной сети

Значимость параметров (Рисунок 2.10), предоставляемая программой, отображает зависимость итогового результата от каждого, отдельно взятого, параметра.

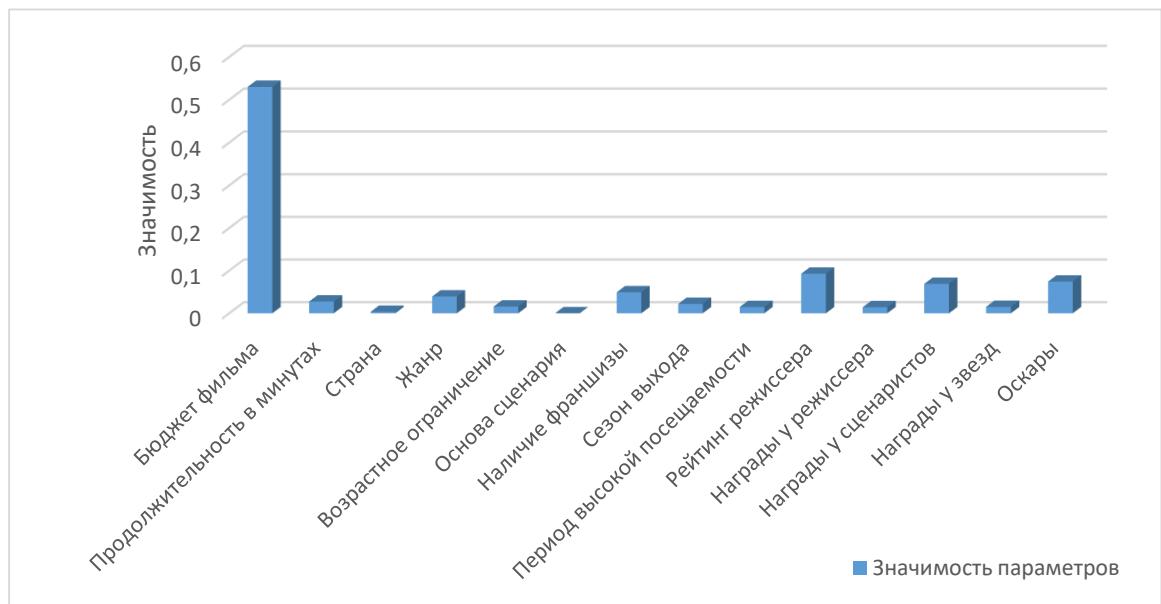


Рисунок 2.10 – Значимость входных параметров нейросетевой модели

Самая низкая значимость среди параметров у «страны» и «основы сценария», 0.3% и 0.01%. Такая низкая значимость была вызвана малой выборкой обучающих

данных, так как из 2000 собранных данных, после очистки осталось всего 635 примера, с одинаковыми значениями по этим параметрам. Такие параметры не стали значимы в тестовом обучении, но могут навредить при разработке конечной нейронной сети, из-за возможности отсутствия данных по странам, которые будет вводить пользователь, соответственно, желательно убрать выбранные параметры из дальнейшего обучения, но при модификации системы и увеличения обучающей выборки, необходимо добавить эти параметры обратно.

Так же были протестированы другие варианты построения нейронной сети, с большим и меньшим количеством скрытых слоев и нейронов в них, в рамках возможности программы. К сожалению выбранная программа может осуществить обучение лишь относительно небольших нейронных сетей, до 1500 связей, после чего программа перестает принимать команды. Представленная выше нейронная сеть показала наиболее оптимальный результат при оптимальном времени обучения и вывела наименьшую ошибку прогнозирования, но при разработке планируется также протестировать многослойные варианты нейронной сети, для снижения процента ошибок.

2.4 Проектирование графического интерфейса

В рамках реализации «MVP» системы, графический интерфейс не является основной задачей разработки, а лишь вспомогательным инструментом для демонстрации работы основных функций системы. Таким образом графический интерфейс разрабатываемой системы, не должен занимать большую часть разработки и соответственно может быть примитивным и выполнять лишь поставленные задачи.

Таким образом интерфейс будет соответствовать интерфейсу (п. 1.4 первой главы), а точнее интерфейсу, представленного в этой главе веб-приложения.

Для выполнения работы основных функций системы интерфейс, должен предоставлять возможность вводить значения в необходимые поля, получать сообщения об ошибках и получать результат по нажатию соответствующей кнопки.

Вводимые значения могут быть числовыми и логическими (да/нет) и иметь проверку ввода. Далее программа будет автоматически конвертировать значения в

необходимый формат для получения результата. Основываясь на этом должны присутствовать поля ввода и поля с выпадающим списком, где необходимо.

Результатом работы системы должен выступать числовой результат предсказания системы и дополнительные комментарии при необходимости. Пример графического интерфейса представлен ниже (Рисунок 2.11).

Бюджет фильма (доллары США):
Возраст, от которого допускается просмотр фильма (согласно российской возрастной классификации информационной продукции):
Продолжительность фильма (минут):
Наличие у режиссера номинаций/наград на престижные кинопремии («Оскар», «Золотой глобус», SAAG, Critics' Choice Awards):
Наличие у актеров номинаций/наград на престижные кинопремии («Оскар», «Золотой глобус», SAAG, Critics' Choice Awards):
Основной жанр:
Наличие спецэффектов:
Является ли фильм продолжением какого-либо фильма (т.е. является ли фильм следующей серией франшизы – сиквелом):
Выход фильма в период высокой посещаемости кинотеатров (новогодние праздники, летние каникулы и т.д.):

Получить результат

Рисунок 2.11 – Пример проектируемого интерфейса

2.5 Выводы по второй главе

В результате проектирования системы были созданы необходимые диаграммы и алгоритмы, которые будут использованы при реализации системы, а также выполнены поставленные задачи:

1. Описано взаимодействие системы внутри программы и ее компонентов, с помощью диаграмм последовательностей «см. ПРИЛОЖЕНИЕ F» и диаграммы компонентов (п. 2.1 текущей главы).
2. Спроектированы алгоритмы сбора данных обучения в виде блок-схем «см. ПРИЛОЖЕНИЕ G» и определены параметры обучающего множества (п. 2.2 текущей главы).
3. Спроектирована нейронная сеть и ее архитектура, а также скорректированы параметры обучения (п. 2.3 текущей главы).
4. Спроектирован необходимый графический интерфейс в соответствии с требованиями и ролью пользователей системы (п. 2.4 текущей главы).

Глава 3 Реализация программного решения

Данная глава посвящена и обозревает реализацию спроектированной в предыдущей главе систему и включает в себя выбор и обоснование средств и инструментов разработки, написание программного кода и выделение используемых структур и решений, связанных с реализацией системы в выбранной предметной области, создание пользовательского графического интерфейса, тестирование и отладка разработанного решения, а также написание необходимого набора программной документации. В главе решаются следующие задачи:

1. Определение и обоснование языка разработки, наиболее подходящих в разрабатываемой системе, используемых библиотек и набор используемых инструментов, для непосредственной реализации.
2. Написание и структурное описание программного кода для модуля сбора и очистки данных, основываясь на функциональных требованиях, выдвинутых в первой главе и проектировании второй главы.
3. Написание, тестирование и структурное описание программного кода для модуля нейронной сети основываясь на функциональных требованиях, выдвинутых в первой главе и проектировании второй главы.
4. Реализация внешнего вида системы, с привязкой ко всем разработанным функциям и требованиям, приведенным в техническом задании.

Результатом главы является проведение оценки полученных результатов, разработанное программное решение, а также необходимая документация для пользователей системы.

3.1 Обоснование выбора средств разработки

Языком программирования для реализации конечной системы был выбран ЯП «Python». Выбор обусловлен более высокой производительностью в задачах, требующих обработку данных, не смотря на то, что выбранный ЯП является наиболее высокоуровневым, в отличии от группы других объектно-ориентированных языков, позволяющих работать с машинным обучением, а также наличием библиотек для облегчения построения нейронной сети в сжатые сроки, что крайне полезно при разработке «MVP» системы, так как для построения необходимой архитектурной структуры, требуется большое количество времени и

человеческих ресурсов, что невозможно во временных рамках разработки поставленной задачи. Также, на данный момент, наибольшее количество решений, использующих нейронные сети, реализованы именно на языке «Python». Ниже представлен список средств, которые будут применяться для сбора, анализа и обработки данных:

- Сбор данных: BeautifulSoup, Selenium, Regex;
- Анализ данных: Numpy, Openpyxl, Matplotlib;
- Обучение нейронной сети: Pandas, Keras, Tensorflow;

В качестве инструмента для реализации пользовательского интерфейса была выбрана библиотека PyQT, из-за возможности создания кроссплатформенных приложений.

В качестве среды разработки используется IDE «PyCharm», разработчика «JetBrains», выбор обосновывается сотрудничеством разработчика с НИУ «ВШЭ», а также наибольшим функционалом среди других программных решений и удобностью использования.

3.2 Реализация модуля сбора и систематизации обучающих данных

Для обучения нейронной сети, первоначальной задачей, является сбор обучающего множества, на основе которого будет работать, обученная система. Основываясь на результатах проектирования (п. 2.2 второй главы) был составлен список параметров обучения, определен источник сбора этих параметров и составлены алгоритмы работы модуля. Также было принято решение для разделения модуля на отдельные алгоритмы, для более энергозатратной и эффективной работы. Таким образом было сформировано 3 отдельных алгоритма:

Скрэппер – выступает процессом сбора первичных URL-адресов на кинокартины для последующей отправки этих адресов в парсер. Так как процесс занимает время, а обновление списка кинокартин производится не регулярно, то и нет необходимости в работе алгоритма при каждом запуске программы. На вход будут поступать, определенные во время проектирования, со списками из кинофильмов, откуда необходимо собрать персональные адреса каждой кинокартине. Результатом работы процесса будет список URL-адресов кинокартин, который будет являться входными данными для парсинга.

Парсер – основной процесс сбора обучающих параметров. Причиной отделения послужило долгое время работы программы и при необходимости изменения структуры обучающего множества и/или изменении его параметров, срабатывание процесса заново является абсолютно не оптимальным решением. Результатом работы будет служить список данных о фильме с их параметрами в виде словаря ключ-значение.

Алгоритм очистки и сохранения – предназначен для финальной обработки полученных параметров, очистки от выбросов и сохранение в конечное обучающее множество.

Листинг исходного кода для модуля сбора и систематизации обучающих данных в полном объеме представлен в «ПРИЛОЖЕНИИ Н».

3.2.1 Реализация алгоритма первичного сбора данных

Как уже было описано ранее, входными данными алгоритма являются URL-адреса (п. 2.2 второй главы), с рейтингами «TOP-1000» и «BOTTOM-1000» кинокартин, по оценкам зрителей сайта «IMDb» [59]. Данные представлены в виде списков таблиц с небольшой выдержкой основной информации о кинокартинах и адресом на каждую отдельную кинокартину. Таким образом алгоритм обрабатывает каждую страницу (Рисунок 3.1) по указанному во входных данных адресу и находит ID страниц каждого кинокартины, как видно на изображении, на обработку всех 2000 ссылок, затрачивается порядка 1 минуты после чего следует перенос списков в один файл, который и является результатом работы алгоритма и хранит URL-адреса на кинокартины. (Рисунок 3.2).

```
Run: scrapper
bottom_1000 / 772. id 1520470
bottom_1000 / 993: id 3595298
bottom_1000 / 994: id 6910020
bottom_1000 / 995: id 2112131
bottom_1000 / 996: id 5456546
bottom_1000 / 997: id 3477064
bottom_1000 / 998: id 1188982
bottom_1000 / 999: id 6439558
bottom_1000 / 1,000: id 1373215
--- END IN 55.476 seconds ---
```

Рисунок 3.1 – Процесс работы алгоритма

```
https://www.imdb.com/title/tt6710474/
https://www.imdb.com/title/tt1745960/
https://www.imdb.com/title/tt1877830/
https://www.imdb.com/title/tt10872600/
https://www.imdb.com/title/tt0068646/
https://www.imdb.com/title/tt0499549/
https://www.imdb.com/title/tt1160419/
```

Рисунок 3.2 – Результат работы алгоритма

Алгоритм является наиболее простым для реализации среди всех алгоритмов системы и алгоритмически включает в себя подключение и загрузку страницы, проверку на окончание списка, проверку на окончание страниц и сохранение результата. Ниже представлен листинг алгоритма (Рисунок 3.3).

```

def scrapper(url, rat):
    data = []          # Переменная для хранения id фильмов
    start = 0          # Счетчик обработанных фильмов
    iteration = 0      # Счетчик первой итерации

    while start < 1000:
        request = requests.get(url, headers=headers)           # Передаем ссылку в обработчик
        soup = BeautifulSoup(request.content, 'html.parser')   # Подключаемся к странице

        # Находим список фильмов на странице
        title_block = soup.find('div', class_='lister-list').find_all('div', class_='lister-item mode-advanced')
        for item in title_block:                                # Для каждого элемента в списке
            title = item.find('h3', class_='lister-item-header').find('a')          # Находим элемент
            title_id = title.get('href').replace('/title/tt', '').replace('/ref_=adv_li_tt', '') # Находим id
            data.append(title_id)                                # Записываем id фильма в список
            print(
                f'{rat} / '
                f'{item.find("span", class_="lister-item-index unbold text-primary").get_text().replace(".", ":")}: '
                f'id \033[32m{title_id}\033[0m')

        if start + 50 <= len(data):    # Переходим на след страницу
            if iteration == 0:         # Если первая итерация то добавляем необходимый текст к ссылке
                url = url + f'&start={str(start + 50 + 1)}&ref_=adv_nxt'
                iteration += 1
            else:                      # Иначе заменяем параметр страницы в запросе ссылки
                url = url.replace(f'&start={str(start + 1)}', f'&start={str(start + 50 + 1)}')
                start += 50
        else:
            # Иначе вывод ошибки
            # print('сломалось')
            break

    return data # Возвращаем список id фильмов

```

Рисунок 3.3 – Листинг алгоритма первичного сбора данных

Для работы со страницами алгоритм использует 2 библиотеки:

- **Requests** – вспомогательная библиотека, которая позволяет отправлять HTTP запросы и получать ответы от сервера веб-страницы для взаимодействия с контентом, представленным на этих страницах.
- **BeautifulSoup4** [57] – библиотека позволяющая собирать HTML разметку веб-страниц и взаимодействовать с ней. Необходима при получении данных с сайтов.

3.2.2 Реализация алгоритма сбора параметров обучающего множества

Как и предыдущий алгоритм, также разрабатывается на языке программирования «Python». Алгоритм предназначен для сбора выделенных в проектировании параметров, их обработки и составления словаря ключ-значение для последующей очистки и сохранения их в датасет. Также алгоритм выступает тестом для предыдущего алгоритма.

Получение всех данных о кинокартинах занимает достаточно большое время, так как приходится ждать загрузки страницы и создавать множество

дополнительных запросов на смежные веб-страницы, из-за чего время работы только увеличивается.

Сбор данных происходит в фоновом режиме с помощью двух библиотек:

BeautifulSoup4 – как уже описывалось в предыдущем алгоритме, библиотека используется для быстрого взаимодействия с HTML разметкой страницы и получения информации из нее, но не позволяет получать данные, обработанные с помощью «JavaScript».

Selenium – библиотека, предназначенная, для взаимодействия напрямую с браузером, в отличии от «BeautifulSoup» библиотека позволяет настраивать веб-драйвер, для имитации человеческого взаимодействия с веб-ресурсом, таким образом сайт или веб-страница воспринимает загрузку библиотекой, как загрузку настоящим пользователем, что позволяет загрузить дополнительные алгоритмы и проявить информацию, которая загружается со временем. Из-за этих преимуществ, библиотека становится более требовательной к ресурсам и времени, затрачиваемого для загрузки веб-страницы, также необходимо наличие дополнительного пакета с самим драйвером в папке с программой, а загрузка каждой страницы занимает больше времени чем при использовании «BeautifulSoup». В качестве веб-драйвера используется «Chromium», являющийся 62 стандартным для RPi. Используемый веб-драйвер и браузер для использования библиотеки должны быть одной и той же версии.

Основная часть алгоритма состоит из последовательного сбора данных с сайта «IMDb» [59]. Алгоритм реализован по построенной блок-схеме (п. 2.2 второй главы) и блок схемам в «ПРИЛОЖЕНИИ Н». При запуске алгоритма проверяется наличие файла с URL-адресами на кинокартины, после чего инициализируется список, заполненный этими адресами и запускается основная функция сбора данных. Ниже представлен тестовый пример работы алгоритма (Рисунок 3.4)

```
Run: parser x
P:\.Projects\GitHub\HSE-University-projects\Graduate-work\src\parser\venv\Scr:
--- TEST PROCESSING ---
0/2000 Progress (0038650): [processing] https://www.imdb.com/title/tt0038650/
0/2000 Progress (0038650): [done] https://www.imdb.com/title/tt0038650/
31;130;0;1;0;4;1;7;1;1;1;11;61
--- END IN 79.835 sec ---
```

Рисунок 3.3 – Процесс тестовой работы алгоритма

Как было представлено выше, на обработку одной кинокартины уходит большое количество времени, а точнее от 60 до 110 секунд, в зависимости от свободного объема оперативной памяти компьютера, что несопоставимо долго и на обработку 2000 таких адресов потребуется от 33 до 61 часа, что занимает более суток. Из-за такой большой длительности было принято решение о включении многопоточности, с помощью подключения библиотеки «Multiprocessing».

Multiprocessing [64] – библиотека, позволяющая клонировать процессы программы, для одновременной работы сразу нескольких копий, количество создаваемых процессов, зависит от максимального количества возможных потоков процессора компьютера. Таким образом удалось достичь 12 одновременно запущенных процессов, что сократило обработку 2000 адресов до ~3 часов.

Для получения данных алгоритм открывает в фоновом режиме страницу (Рисунок L.1-L.2 «см. ПРИЛОЖЕНИЕ L») и проверяет наличие элемента с атрибутом названия (цифра 1 рисунка L.1) кинокартины. При ее нахождении, считается что страница существует и алгоритм продолжает работу, а при ее отсутствии ставит ключевое значение равное 0, при сохранении алгоритм будет пропускать строки с нулевым ключевым значением. При продолжении работы алгоритма выполняются последующие функции, для сбора параметров:

Получение длительности. Для уменьшения количества обрабатываемых данных, было принято решение совместить получение длительности и возрастного ограничения из одного элемента. Для этого алгоритм находит список элементов с классом «ipc-inline-list» (цифра 2 рисунка L.1) и в цикле обрабатывает их значения (Рисунок 3.4). Для корректной обработки значений была добавлена библиотека «re», для поиска регулярных выражений и, соответственно, сами регулярные выражения:

- '\b\w{1,2}[h]\b' – Количество часов.
- '\b\w{1,2}[m]\b' – Количество минут.

```
# region С помощью regex выбираем значения длительности и возрастного ограничения
items_list = soup.find('ul', class_='ipc-inline-list').find_all('li', class_='ipc-inline-list__item')
for item in items_list:
    hour = re.findall(r'\b\w{1,2}[h]\b', item.get_text()) if not hour else hour      # поиск часов
    minute = re.findall(r'\b\w{1,2}[m]\b', item.get_text()) if not minute else minute  # поиск минут
    if item.find('a', class_='ipc-link') is not None:
        year = re.findall(r'\d{4}', item.get_text()) if not year else year          # поиск года выхода
        mpaa = re.findall(r'\b([0-9]{1,2})+(PG-13|NC-17|PG|R|G|X)\b',
                           item.get_text()) if not mpaa else mpaa # поиск возрастного рейтинга
# endregion
```

Рисунок 3.3 – Листинг сбора элементов

После нахождения значений вычисляется итоговая длительность фильма по формуле:

$$D = 60h + m \quad (3.1)$$

где h – часы,

m – минуты.

Получение возрастного ограничения. Так же, как и с получением длительности используется регулярное выражение вида:

– `'\b([0-9]{1,2}\+)|(PG-13|NC-17|PG|R|G|X)'`

После чего очищаются от лишних символов и кодируются в соответствии со списком кодировки:

- $(G / 0+) - 1;$
- $(PG / 6+) - 2;$
- $(PG-13 / 12+ / 14+) - 3;$
- $(NC-17 / 16+) - 4;$
- $(R / 18+) - 5.$

Получение жанра. Алгоритм находит поле с информацией о жанре (цифра 8 рисунка L.1) по атрибуту «genres» и получает первое значение, так как оно является основным жанром фильма, после чего, полученное значение кодируется в соответствии со списком кодировки (Рисунок 3.4).

```
# Функция получения жанра
def parse_genre(request):
    # Список кодировок жанров
    genre_format = {'Action': 1, 'Adventure': 2, 'Drama': 3, 'Romance': 3, 'Comedy': 4,
                    'Crime': 5, 'Mystery': 6, 'Horror': 7, 'Western': 8, 'History': 9, 'Documentary': 9,
                    'Biography': 10, 'Animation': 11, 'Fantasy': 12, 'Sci-Fi': 12,
                    'Thriller': 13, 'Music': 14, 'Musical': 14, 'Film-Noir': 15, 'War': 16,
                    'Family': 17, 'Short': 18, 'Sport': 19, 'Reality-TV': 20,
                    'Game-Show': 21, 'Talk-Show': 22, 'News': 23, 'Adult': 24}

    genre = request.find('div', attrs={'data-testid': 'genres'})           # Поиск поля по атрибуту
    if genre is not None:                                                 # Проверка на пустоту
        genre = genre.find('span', class_='ipc-chip__text')
        if genre is not None:
            genre = genre_format[genre.get_text()]                           # Получение текста

    return genre
```

Рисунок 3.4 – Листинг функции получения жанра

Получение наличия франшизы. Проведя поиски возможностей сбора данного параметра, было выяснено, что ни один из сайтов-агрегаторов (п. 1.1 первой главы) работы не содержат напрямую информацию касательно франшизы

кинокартин. Единственное найденное возможно решение, это узнавать данные о франшизе через интернет ресурс «Wikipedia» [70], но на оригинальной английской версии (Рисунок 3.5) нет данной информации и такая возможность присутствует только на российской версии ресурса (Рисунок 3.6, цифра 3). Также, только в русской версии, присутствует возможность поиска страницы с кинокартиной по ID с сайта «IMDb» [59] (цифра 4 рисунка 3.6).

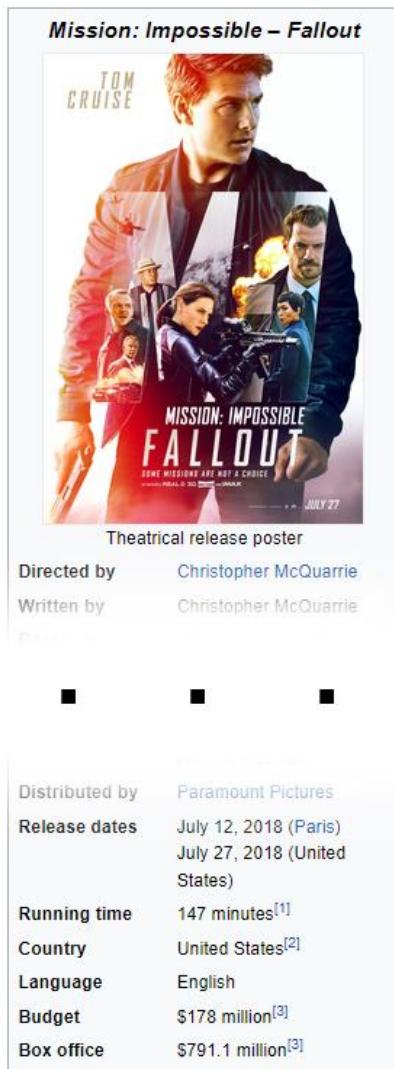


Рисунок 3.5 – Пример страницы кинокартины на английской версии Википедии

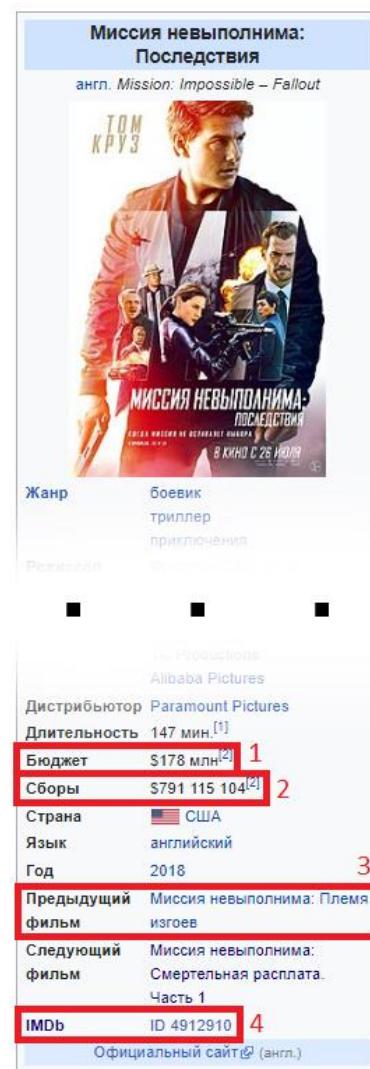


Рисунок 3.6 – Пример страницы кинокартины на русской версии Википедии

Наличие франшизы будет считаться в том случае если кинокартина выступает продолжением, а не является самым первым фильмом во франшизе, так как в таком случае кинокартина не успела себя зарекомендовать как продолжение популярной франшизы и не сможет привлечь зрителей по знакомому названию.

Поиск и сбор данных производится в несколько этапов, для чего сначала с помощью библиотеки «Selenium» производится поиск кинокартины на википедии

(Рисунок 3.7) с задержкой в 1 секунду для загрузки «JavaScript» алгоритмов и отображения результата, после чего сохраняется первая найденная страница и загружается для обработки с помощью более быстрой библиотеки «BeautifulSoup». После загрузки нужной страницы проверяется совпадение искомого ID с тем, что представлен на странице (цифра 4 рисунка 3.6) и ищется элемент с атрибутом «P155», при его отсутствии считается, что франшизы нет.

The screenshot shows the IMDb search interface. In the search bar, the text "IMDb ID 4912910" is entered. Below the search bar are two dropdown menus: "Расширенный поиск:" (Advanced search) set to "Сортировать по релевантности" (Sort by relevance), and "Поиск по:" (Search by) set to "(Основное)" (Main). A blue "Найти" (Find) button is located at the top right.

Миссия невыполнима: Последствия

Предыдущий фильм Миссия невыполнима: Племя изгоев Следующий фильм

Миссия невыполнима: Смертельная расплата. Часть 1 IMDb ID 4912910

Официальный сайт (англ.)

19 Кб (1011 слов) - 00:15, 25 мая 2022

Рисунок 3.7 – Поиск кинокартины на Википедии

При отсутствии результатов поиска выводится одна из ошибок: «NoSuchElementException», «StaleElementReferenceException» и считается что страница не существует. Основываясь на результатах, можно сделать вывод что практически для всех кинокартин, являющихся частью франшизы, существует страница на сайте «Wikipedia» и соответственно информация о предыдущих и последующих фильмах этой франшизы. Часть алгоритма представлена ниже (Рисунок 3.8), а так же, полный алгоритм представлен в «ПРИЛОЖЕНИИ Н».

```
try:
    # вводим строку поиска id на сайте imdb
    driver.get(f'https://ru.wikipedia.org/w/index.php?search=IMDb%09ID+{str(child_id)}&ns0=1')
    time.sleep(2) # ждем загрузки страницы с результатами поиска

    search_result = driver.find_element(By.XPATH, '//div[@class="mw-search-result-heading"]/a')
    if search_result is not None: # проверка на пустоту
        search_result = search_result.get_attribute('href') # Находим ссылку на страницу

    url = str(search_result) #
    session = requests.Session()
    request = session.get(url, headers=headers)
    soup = BeautifulSoup(request.content, 'lxml') # Загружаем найденную страницу в более быстрый bs4

    # region Франшиза
    franchise1 = soup.find('span', attrs={'data-wikidata-property-id': 'P155'}) # узнаем есть ли предыдущие фильмы
    if franchise1 is not None: # Если значение есть, то есть и франшиза
        wiki_info[2] = 1
    else: # Если нет то и франшизы нет
        wiki_info[2] = 0
    # endregion
```

Рисунок 3.8 – Алгоритм поиска данных на Википедии

Получение даты релиза. Для получения сезона выхода алгоритм находит элемент (цифра 9 рисунка L.2) и сохраняет ссылку на информацию по выходу фильма, после чего переходит на нужную страницу (Рисунок 3.9). Так как в каждой отдельной стране дата выхода отличается, то алгоритм находит наиболее упоминаемую дату, с помощью которой определяет период выхода фильма и сезон выхода.

UK	13 July 2017
Ireland	16 July 2017
Belgium	19 July 2017
Finland	19 July 2017

Рисунок 3.9 – Список дат выхода кинокартины

После получения самой часто встречающейся даты, с помощью библиотеки «DateTime» определяется сезон со сдвигом в неделю, таким образом зимний сезон начинается не с 01.12 а с 23.11 и заканчивается 21.03. Далее производится кодировка значений: Зима – 1, Весна – 2, Лето – 3, Осень – 4. Так же, аналогично, если дата находится в рамках летних каникул или праздников, то дата выхода находится в период высокой посещаемости.

Получение данных о съемочной группе. В данную функцию входит поиск наличия о наличии влиятельных наград у съемочной группы, определение общего количества оскаров и рейтинг режиссера, для чего изначально собираются списки, отдельные для каждой должности, состоящие из ID персон съемочной группы (цифры 5-7 рисунка L.1), после чего для каждого списка собирается внутренняя информация со страницы персоны (Рисунок L.3 «см. ПРИЛОЖЕНИЕ L»).

Для сбора информации о престижных наградах производится поиск элемента с престижными номинациями и наградами (цифра 2 рисунка L.3), который имеет вариативное отображение, в зависимости от наград (Рисунок 3.10 и 3.11), откуда удаляются лишние символы и суммируются награды.

Won 1 Oscar. Another 75 wins &

Рисунок 3.10 – Отображение получения оскара

Nominated for 1 BAFTA Film Award. Anot

Рисунок 3.11 – Отображение номинации BAFTA

Для сбора рейтинга режиссера на странице с персоной производится поиск таблицы с работами человека как режиссера кинокартин (цифра 3 рисунка L.3), откуда собираются ID кинокартин, после чего выполняется переход по каждому ID и собирается рейтинг каждой кинокартини, если рейтинга нет или картина не выпущена (тоже отсутствие рейтинга). То возвращается значение -1 и данные по этой кинокартине не учитываются. Ниже представлен алгоритм работы (Рисунок 3.12).

```

works_urls, works_num, works_sum = [], 0, 0
if director == 1:                                # Если получали данные о директорах
    # Поиск атрибутов по регулярному выражению
    works = soup.find_all('div', attrs={'id': re.compile("director-tt")})
    if works is not None:                          # Проверка на пустоту
        for work in works:                        # Для каждого фильма в списке
            # Находим фильм и получаем его id
            work = (work.find('b').find('a')).get('href')
            .replace('/title/tt', '').replace('/?ref_=nm_flmg_dr_1', '')
            works_urls.append(work)
    for url in works_urls:                         # для каждого id в списке
        info = parse_director(url)
        if info != -1:                             # Если вернулся не -1 то учитываем это значение
            works_num += float(info)              # Суммируем рейтинги
            works_sum += 1                         # Суммируем количество фильмов
    output[3] = round(works_num / works_sum)       # Делим сумму рейтингов фильмов на количество фильмов

```

Рисунок 3.12 – Алгоритм получения рейтинга режиссера

Так же рассматривалась возможность получения рейтинга каждого члена команды по результатам оценки пользователями сайта «IMDb» [59] (цифра 1 рисунка L.3), но данный параметр был бы не объективен так как является оценочным суждением, так же сложен в использовании при наличии персон, не состоящих в списке сайта. То же самое для остальных наград, для получения оптимального значения, нужно составить полноценный рейтинг и вес для каждой отдельной награды.

Получение бюджета и кассовых сборов. Данные о бюджете и кассовых сборах, представленные на сайте «IMDb» (цифры 11-12 рисунка L.2), предоставляются напрямую с другого сайта-агрегатора «BoxOfficeMojo» [58], из-за чего для оптимизации времени работы, было принято решение собирать данные с сайта «IMDb», при их отсутствии на сайте, так же принято решение для дополнительного сбора информации о бюджете и сборах с сайта «Wikipedia» [70], аналогично как и со сбором наличия франшизы (цифры 1-2 рисунка 3.6). Получение всех данных только с Википедии не является возможным, так как не для каждого

фильма существует страница на ресурсе и работает, только для популярных фильмов и франшиз. Поэтому при отсутствии данных кинокартина не записывается в окончательную обучающую выборку.

Алгоритмы сбора бюджета и кассовых сборов идентичны между собой (Рисунок 3.13). Сначала производится поиск секции с данными, после чего производится поиск элементов секции, после получения элементов производится очистка данных и конвертация разных валют в доллары.

```
# region Значения бюджета и кассовых сборов
boxoffice_section = soup.find('div', attrs={'data-testid': 'title-boxoffice-section'})
if boxoffice_section is not None:          # Проверка на пустоту секции кассовых сборов
    budget = boxoffice_section.find('li', attrs={'data-testid': 'title_boxoffice-budget'})
    if budget is not None:                  # Проверка на пустоту элемента с бюджетом
        budget = budget.find('span', class_='ipc-metadata-list-item__list-content-item')
        if budget is not None:              # Дополнительная проверка на пустоту
            # Очистка значений
            budget = replace_scrap(budget.get_text()).replace(',', '').replace('\xa0', '')
            budget = convert_currency(budget) # Конвертация в доллары
        else:                                # Если информации нет то присвоение данных из википедии
            budget = wiki_info[0]
    else:
        budget = wiki_info[0]
# все то же самое что и сверху но для кассовых сборов
boxoffice = boxoffice_section.find('li', attrs={'data-testid': 'title-boxoffice-cumulativeworldwidegross'})
if boxoffice is not None:                  # Проверка на пустоту элемента с кассовыми сборами
    boxoffice = boxoffice.find('span', class_='ipc-metadata-list-item__list-content-item')
    if boxoffice is not None:              # Дополнительная проверка на пустоту
        # Очистка значений
        boxoffice = replace_scrap(boxoffice.get_text()).replace(',', '').replace('\xa0', '')
        boxoffice = convert_currency(boxoffice) # Конвертация в доллары
    else:
        boxoffice = wiki_info[1]           # Если информации нет то присвоение данных из википедии
else:
    boxoffice = wiki_info[1]
# endregion
```

Рисунок 3.13 – Алгоритм получения бюджета и кассовых сборов

Получение страны. Получение данных оказалось трудно доступным, так как на сайте «IMDb» [59] отображены данные о всех странах, участвовавших в производстве (цифра 10 рисунка L.2) и если собирать только первую страну, то в 90% случаев это будет США, из-за чего обучение может быть не верным. При сборе данных с Википедии, результат аналогичен, описанному в сборе бюджета и кассовых сборах и потому, тоже не является оптимальным. Из-за этих причин, после нескольких попыток собрать данные о стране, было принято решение убрать данный параметр из обучающей выборки.

Получение основы сценария. Получение данного параметра оказалось затруднительным, так как нет ресурсов, позволяющих автоматически собирать этот параметр и единственным вариантом остается введение этого параметра вручную для каждой отдельной кинокартины, что трудозатратно и невозможно при увеличении итоговой обучающей выборки.

Результатом работы алгоритма выступает файл, хранящий в себе список в формате ключ-значение: {'id': '10872600', 'name': 'Человек-паук: Нет пути домой', 'url': 'https://www.imdb.com/title/tt10872600/', 'year': '2021', 'budget': 2000, 'duration': 148, 'genre': 1, 'age-limit': 3, 'franchise': 0, 'release-season': 4, 'holiday': 1, 'director-rating': 7, 'directors-awards': 0, 'writers-awards': 1, 'stars-awards': 1, 'oscars': 0, 'writers': 56, 'stars': 11, 'box-office': 18000}. Всего было собрано 1285 строк данных из изначальных 2000 URL-адресов. Тестируением алгоритма является алгоритм формирования обучающего множества.

3.2.3 Реализация алгоритма формирования обучающего множества

Для составления итогового обучающего множества необходимо очистить значения от выбросов и пустых значений, а также распределить значения на обучающие и тестирующие. Очистка производится в несколько этапов: приведение значений к нужному формату, нахождение дубликатов, определение нетипичных значений и сама очистка. При реализации алгоритма, для визуализации значений использовались библиотеки:

- **MatPlotLib** [61] – библиотека предназначенная для визуализации данных двумерной графикой в виде диаграмм с данными.
- **Seaborn** – библиотека, расширяющая возможности «MatPlotLib», используется для отображения тепловых карт.

Первым этапом выступает приведение значений в необходимый формат, для чего создаются внутренние файлы, для временного хранения данных и взаимодействия с ними. Алгоритм сохранения (Рисунок 3.14) и результат сохранения (Рисунок 3.15) представлены ниже.

```
with open(f'./input_dataset.txt', 'a', encoding='utf-8') as file:
    for row in data:
        file.write(f'{row["budget"]};{row["duration"]};{row["genre"]};{row["age-limit"]};{row["franchise"]};'
                  f'{row["release-season"]};{row["holiday"]};{row["director-rating"]};'
                  f'{row["directors-awards"]};{row["writers-awards"]};{row["stars-awards"]};'
                  f'{row["oscars"]};{row["box-office"]}\n')
```

Рисунок 3.14 – Алгоритм сохранения значений

```
230;88;1;4;0;4;0;7;0;1;1;2;1000
450;187;1;3;0;1;0;8;1;1;1;0;990
130;202;2;4;0;3;0;6;1;1;1;16;470
290;123;4;4;0;3;0;6;1;1;0;0;1100
250;130;1;4;0;4;0;8;0;1;0;0;560
```

Рисунок 3.15 – Результатом сохранения значений

Далее следует нахождение дубликатов значений, для чего алгоритм создает временный список, с которым производится сравнение основного списка. При отсутствии совпадения данные записываются во временный список. После прохождения цикла основной список перезаписывается временным. Алгоритм представлен ниже (Рисунок 3.16).

```
def del_copy(obj): # удаляем копии из списка
    n = []
    for i in obj:
        if i not in n:
            n.append(i)
    return n
```

Рисунок 3.16 – Алгоритм очистки дубликатов

Следующим, наиболее важным для обучения, этапом является определение нетипичных значений (выбросов), для чего алгоритм производит определение квантилей нормального распределения: квантиль 0.25 (нижний quartиль) и квантиль 0.75 (верхний quartиль). После чего все значения не попавшие в интерквартильный заменяются пустыми значениями (NaN) и в последствии вся строка с пустым значением удаляется. Так как остальные значения кроме бюджета и кассовых сборов были в рамках нормального распределения, из-за кодировки и нормализации, то основными данными для поиска выбросов являются именно они. До определения выбросов (Рисунок 3.17) кассовые сборы содержали достаточно много нетипичных данных, при увеличении выборки этих данных может быть меньше, но в рамках разработки «MVP» системы, для более точного обучения, от этих данных необходимо избавиться. Результат очистки нетипичных значений (Рисунок 3.18) и алгоритм очистки (Рисунок 3.19) представлен ниже.

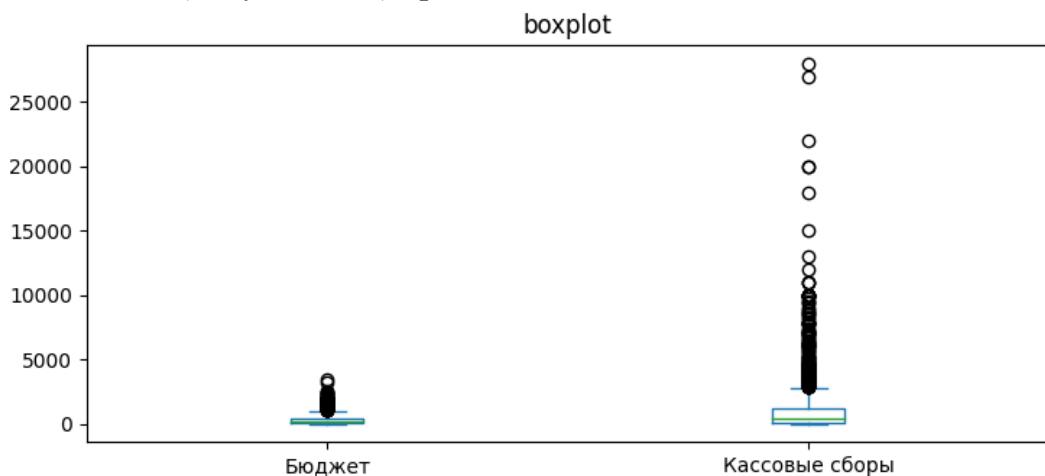


Рисунок 3.17 – Коробчатая диаграмма до удаления выбросов

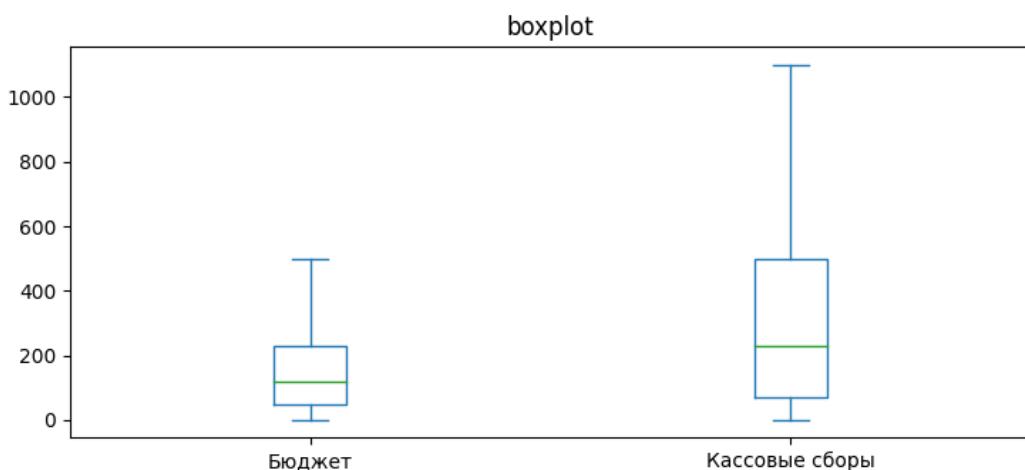


Рисунок 3.18 – Коробчатая диаграмма после удаления выбросов

```

for x in ['budget', 'boxoffice']:
    q75, q25 = numpy.percentile(df.loc[:, x], [75, 25])      # Находим перцентели
    intr_qr = q75 - q25

    maxq = q75 + (1.5 * intr_qr)                            # Определяем квантили
    minq = q25 - (1.5 * intr_qr)

    df.loc[df[x] < minq, x] = numpy.nan                      # Заменяем на NaN
    df.loc[df[x] > maxq, x] = numpy.nan

#print(df.isnull().sum())
df.isnull().sum()                                         # Удаляем NaN
df = df.dropna(axis=0)

```

Рисунок 3.19 – Листинг алгоритма очистки нетипичных значений

Так же была построена тепловая карта (Рисунок 3.20), отображающая равномерность распределения значений. Для оптимального обучения необходимо, чтобы значения были распределены хаотично, так как при повторах нейронная сеть может построить не нужную закономерность. Как представлено ниже основные значения (бюджет и кассовые сборы), распределены достаточно хаотично.

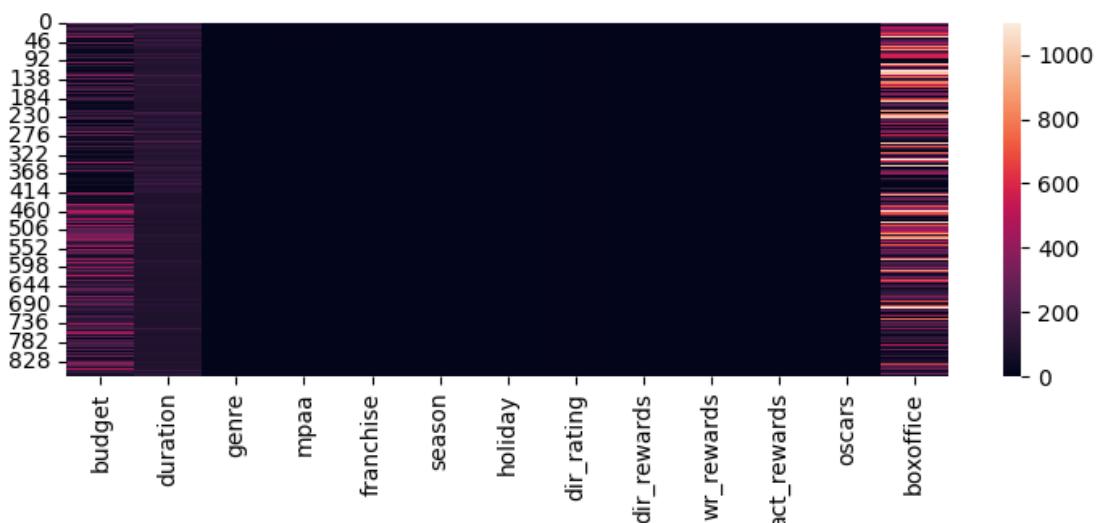


Рисунок 3.20 – Тепловая карта значений

Таким образом после очистки было оставлено 866 строк значений из 1285, которые были распределены алгоритмом (Рисунок 3.21) в процентном соотношении 70/15/15% на обучающее, валидирующее и тестирующее множества по формуле:

$$\sum_n \left\lfloor \frac{r_n}{\frac{R_{sum}}{100} * 15} \right\rfloor = 0 \quad (3.2)$$

где r_n – строка значений,

R_{sum} – общее количество строк.

```
row_counter = 0
dt_val, dt_test, dt_temp, dt_atemp = [], [], [], []
with open(f'./input_dataset.txt', 'r', encoding='utf-8') as mfile:
    for mrow in mfile:
        if row_counter != 0:
            if row_counter % (round(row_sum / ((row_sum * 15) / 100))) == 0:
                dt_val.append(mrow)
            else:
                dt_temp.append(mrow)
        row_counter += 1
mfile.close()
```

Рисунок 3.21 – Листинг алгоритма распределения на множества

Результатом работы программы является файл в формате «XLSX», содержащий: 660 строк данных для обучения и 206 строк для тестирования (разделенных 50/50 на валидирующие и тестирующие). Полный листинг алгоритма представлен в «ПРИЛОЖЕНИИ Н».

3.3 Реализация модуля нейронной сети

На основе тестового обучения (п. 2.3 второй главы) работы была составлена нейронная сеть на языке программирования «Python» с использованием библиотек «Tensorflow» и «Keras», архитектура и алгоритм обучения нейронной сети соответствуют представленной в приведённой выше главе.

- **Tensorflow** – библиотека для машинного обучения, разработанная компанией «Google» для решения задач построения и тренировки нейронных сетей.
- **Keras** – библиотека, представляющая собой надстройку над фреймворком «Tensorflow».

Алгоритм обучения (Рисунок 3.22) построен с помощью метода стохастического градиентного спуска «Adam» (Рисунок 3.22). Ниже представлен процесс обучения (Рисунок 3.23) и результат обучения (Рисунок 3.24). Аргументы алгоритма:

- Скорость обучения (learning_rate) – 0.006
- Скорость затухания для оценок первого момента (beta_1) – 0.9
- Скорость затухания для оценок второго момента (beta_2) – 0.999
- Вычислительная стабильность (epsilon) – 1e07
- Дополнительная скользящая средняя (amsgrad) – False
- Имя операций (name) – «Adam»

```
keras.backend.clear_session()
opt = tensorflow.keras.optimizers.Adam(
    learning_rate=0.006,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    name="Adam")

model = Sequential()
model.add(Dense(3, activation="relu"))
model.add(Dense(1, activation="linear"))
model.compile(optimizer=opt, loss="mse", metrics=["mae"])

history = model.fit(x_train, y_train, validation_split=0.1, verbose=1, epochs=100, shuffle=True)
model.summary()
```

Рисунок 3.22 – Листинг алгоритма обучения

```
Epoch 99/100
13/13 [=====] - 0s 4ms/step - loss: 530.5214 - mae: 19.0040 - val_loss: 381.1562 - val_mae: 16.8285
Epoch 100/100
13/13 [=====] - 0s 4ms/step - loss: 555.2369 - mae: 19.3069 - val_loss: 516.5276 - val_mae: 19.0390
Model: "sequential"

Layer (type)          Output Shape         Param #
=====
dense (Dense)         (None, 3)            36
dense_1 (Dense)       (None, 1)             4
=====
Total params: 40
Trainable params: 40
Non-trainable params: 0
```

Рисунок 3.23 – Процесс обучения нейронной сети

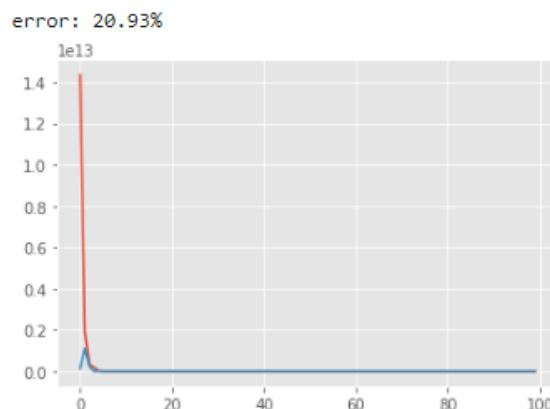


Рисунок 3.24 – Результат обучения нейронной сети

Таким образом средняя квадратичная относительная ошибка тестирования составила 20.93%, данное значение высчитывалось в соответствии с формулой 2.1. Тестирование нейронной сети производилось на подготовленном тестовом множестве. Результат тестирования представлен ниже (Рисунок 3.25), для читабельности данные гистограммы ограничены первыми 31 тестовыми значениями.

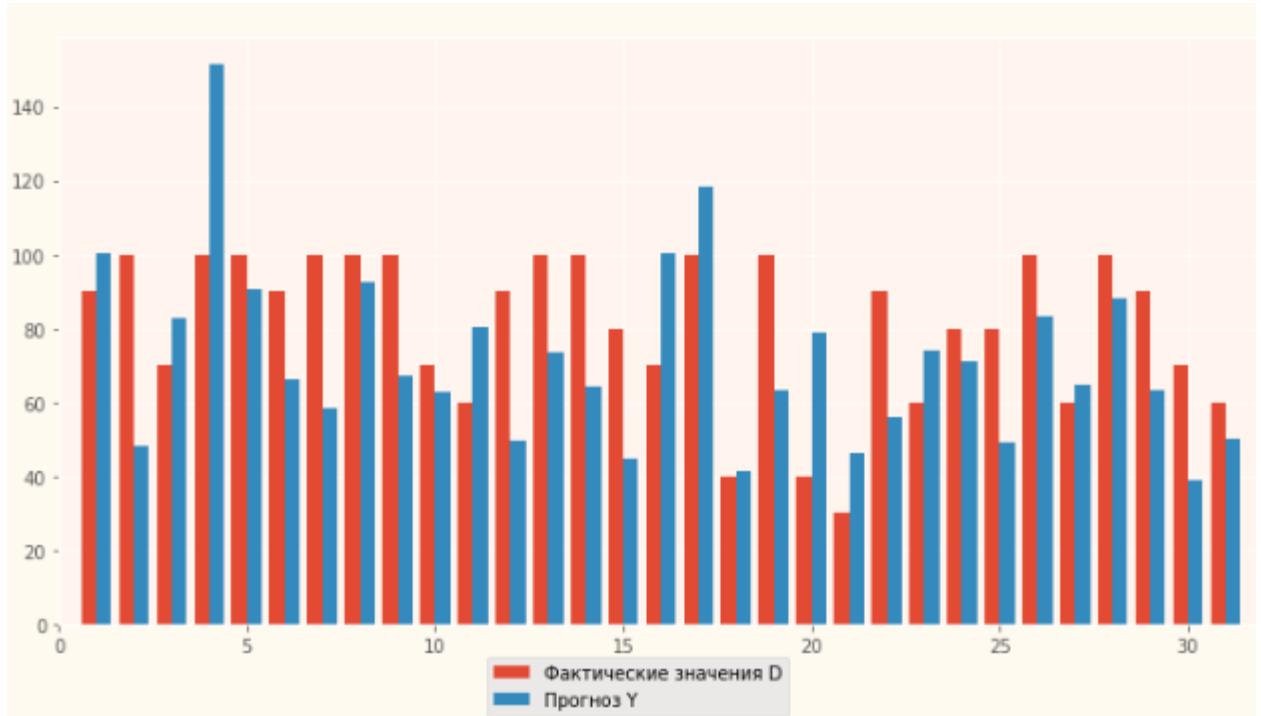


Рисунок 3.25 – Результат тестирования нейронной сети

Для последующего использования нейронная сеть была сохранена в формате «HDF5». Алгоритм сохранения представлен ниже (Рисунок 3.26).

```
# Сохранение сети
model.save('NeuroSet', save_format='h5')
```

Рисунок 3.26 – Листинг сохранения обучения нейронной сети

3.4 Реализация графического интерфейса

Пользовательский графический интерфейс был реализован в соответствии с проектированием второй главы и представляет собой минимальный набор необходимых функций, а именно поля для ввода информации пользователем и поле вывода результата. Ниже представлен сам графический интерфейс системы (Рисунок 3.27).

Рисунок 3.27 – Отображение интерфейса

Как представлено на рисунке при некорректном вводе данных программа выводит предупреждение и продолжает работу. При правильно введенных данных программа конвертирует данные в нужный формат и загружает обученную нейронную сеть (Рисунок 3. 28). Результат работы программы представлен ниже (Рисунок 3.29).

```

self.lblOutput.setText('')
budget = int(data[0])
duration = int(data[1])
genre = genre_format[data[2]]
mpaa = agelimit_format[data[3]]
franchise = yesno_format[data[4]]
season = season_format[data[5]]
holiday = yesno_format[data[6]]
dirating = int(data[7])
dirawards = yesno_format[data[8]]
wrawards = yesno_format[data[9]]
stawards = yesno_format[data[10]]
oscars = int(data[11])

# Формирование датасета для предсказания
output = numpy.array([[mpaa, duration, season, holiday, dirawards, wrawards,
                      stawards, oscars, genre, franchise, budget],
                      [mpaa, duration, season, holiday, dirawards, wrawards,
                      stawards, oscars, genre, franchise, int(budget - ((budget/100)*5))]])

predictions = model.predict(output) # Загрузка обученной нейронной сети

```

Рисунок 3.28 – Листинг алгоритма конвертации

Рисунок 3.29 – Результат работы программы

Программа регулярно тестировалась в процессе разработки, в целях обнаружения и устранения банальных критических ошибок из-за которых не будут работать какие-либо функции или приложение в целом. Данное «пассивное» тестирование очевидно не подвергалось документированию, так как по сути оно является регулярным этапом разработки любого программного средства.

Так как разработанное программное средство необходимо протестировать по стандарту ГОСТ 19.301-79 ЕСПД, на основе сформулированных функциональных и нефункциональных требований к системе, а также диаграммы прецедентов и её описании. Помимо этого, проводилось тестирование программы по критериям тестов «чёрного ящика». Данное тестирование проводилось неоднократно до тех пор, пока все тесты не оказались пройденными успешно.

3.5 Выводы по третьей главе

В главе приведены основные этапы реализации системы и подкреплены в виде фрагментов программного кода системы, а также решены поставленные задачи:

1. Обоснованы выбранные средства разработки и используемые библиотеки (п. 3.1 текущей главы).
2. Реализован модуль сбора и очистки данных (п. 3.2 текущей главы), на основании проектирования второй главы. Приведен листинг модуля «см. ПРИЛОЖЕНИЕ Н».
3. Реализован и протестирован модуль нейронной сети (п. 3.3 текущей главы), на основании проектирования второй главы. Приведен листинг модуля «см. ПРИЛОЖЕНИЕ І».
4. Реализован графический интерфейс системы (п. 3.4 текущей главы), на основании проектирования второй главы. Приведен листинг модуля «см. ПРИЛОЖЕНИЕ К».

Результатом главы является проведение оценки полученных результатов, разработанное программное решение, а также необходимая документация для пользователей системы.

Заключение

В ходе проделанного исследования выявлены наиболее значимые критерии, влияющие на успех фильма. Так же продемонстрированы способы использования созданной модели для получения различных статистических данных способствующих повышению качества фильма.

Практическая ценность исследования основывается на необходимости решения важных задач, таких как, научное обоснование и создание эффективно функционирующего механизма прогнозирования коммерческого потенциала кинопроекта, принятия рациональных управленческих решений о целесообразности его реализации, определение направлений и принципов эффективного управления кинематографическим бизнес-процессом, в частности, на ранних стадиях создания кинофильмов.

Экономическая ценность работы основывается на необходимости снижения инвестиционных рисков и денежных расходов на производство кинокартин как для инвесторов, так и для компаний, занимающихся кинобизнесом.

Результатом выполнения работы является программная системы, разработанная с использованием объектно-ориентированного языка программирования «Python» и нейронной сети на основе машинного обучения.

Для достижения цели был проведен аналитический обзор информации по теме работы, включающий определение особенностей кинобизнеса и тонкостей оценки кассовых сборов, особенностей архитектур нейронных сетей и их алгоритмов и методов машинного обучения, а также сравнительный анализ предыдущих работ, исследований и программных решений по теме дипломной работы. По итогам аналитического обзора, в рамках технического задания был сформирован список требований к разрабатываемому программному средству.

На этапе проектирования программной системы, были разработаны схемы взаимодействия пользователя с системой и взаимодействия компонентов системы между друг другом в виде диаграмм последовательности и активности в нотации UML.

В рамках этапа реализации системы, были выбраны средства, методы и необходимые компоненты разработки программного кода, реализованы

спроектированные модули, а также построена и обучена нейронная сеть. Нейронная сеть тестировалась на основе собранных тестовых значений.

Так как программное средство находилось в рамках разработки «MVP» системы, оно может быть усовершенствовано, расширив выборку данных обучения и увеличив возможности пользовательского интерфейса.

Список сокращений и условных обозначений

API – Application Programming Interface.

DBM – Deep Boltzmann machine.

LSTM – Long short-term memory.

MVP – Minimum Viable Product.

RBM – Restricted Boltzmann machine.

UML – Unified Modeling Language.

UI – User Interface.

VAE – Variational autoencoder.

ВКР – Выпускная Квалификационная Работа.

ГОСТ – Межгосударственный стандарт.

ПО – Программное Обеспечение.

ПрО – Предметная Область.

ТЗ – Техническое Задание.

Библиографический список

1. *Айвазян С.А.* Прикладная статистика: основы моделирования и первичная обработка данных. / С.А. Айвазян, И.С. Енюков, Л.Д. Мешалкин // М.: Финансы и статистика. – 1983.
2. *Вапник В.Н.* Восстановление зависимостей по эмпирическим данным // М.: Наука. – 1979.
3. *Горбань А.Н.* Нейроинформатика: кто мы, куда мы идём, как путь наш измерить // Вычислительные технологии. М.: Машиностроение. – 2000. – № 4. – С. 10-14.
4. *Лоскутов А.Ю.* Введение в синергетику. / А.Ю. Лоскутов, А.С. Михайлов // М.: Наука. – 1990. – С. 233-237.
5. *Мак-Каллок У.С.* Логическое исчисление идей, относящихся к нервной активности / У.С. Мак-Каллок, В. Питтс // Автоматы. – 1956. – С. 363-384.
6. *Маталыцкий М.А.* Теория вероятностей, математическая статистика и случайные процессы. / М.А. Маталыцкий, Г.А. Хацкевич // Минск: Вышэйшая школа. – 2012. – 720 с.
7. *Осипов Г.С.* Искусственный интеллект: состояние исследований и взгляд в будущее // – 2011.
8. *Флах П.* Машинное обучение. / П. Флах // М.: ДМК Пресс. – 2015. – 400 с.
9. *Фомин С.В.* Математические проблемы в биологии / С.В. Фомин, М.Б. Беркинблит //
10. *Хуриудов А.А.* Обучение многослойного разреженного автокодировщика на изображениях большого масштаба // Вестник компьютерных и информационных технологий 02.2014 pp.027-030
11. *Черепанов Ф.М.* Нейросимулятор 5.0 / Ф.М. Черепанов, Л.Н. Ясницкий // Свидетельство о регистрации программы для ЭВМ RUS 2014618208 – № 2014618208 – 12.07.2014 г.
12. *Ясницкий Л.Н.* Методика нейросетевого прогнозирования кассовых сборов кинофильмов / Л.Н. Ясницкий, Н.О. Белобородова, Е.Ю.

Медведева // Финансовая аналитика: проблемы и решения – 2017. – Т. 10. – № 4 (334). – С. 449-463.

13. Ясницкий Л.Н. Экономико-математическая нейросетевая модель для оптимизации финансовых затрат в кинобизнесе / Л.Н. Ясницкий, Д.И. Плотников // Фундаментальные исследования. – 2016. – № 11-2. – С. 339-342.
14. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения [Электронный ресурс]. Режим доступа: <https://files.stroyinf.ru/Data/283/28346.pdf>, свободный (дата обращения: 05.04.2022)
15. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы [Электронный ресурс]. Режим доступа: <http://docs.cntd.ru/document/gost-34-602-89>, свободный (дата обращения: 05.04.2022).
16. Ackley D.H. A Learning Algorithm for Boltzmann Machines. / D.H. Ackley, G.E. Hinton, T.J. Sejnowski // Cognitive Science. – 1985. – Vol. 9 (1). – C. 147-169.
17. An, J. Variational autoencoder based anomaly detection using reconstruction probability / J. An, S. Cho // Special Lecture on IE. – 2015 – Vol. 2(1).
18. Buades A. A Review of Image Denoising Algorithms, with a New One / A. Buades, B. Coll, J.M. Morel // Multiscale Modeling & Simulation. – 2005. – Vol. 4 (2). – P. 490-530.
19. Chang W. Bayesian belief network for box-office performance: A case study on Korean movies, / W. Chang, K.J. Lee // Expert Systems with Applications. – Elsevier, 2009. – Vol. 36(1). – P. 280-291.
20. Dayan P. The Helmholtz machine / P. Dayan, G.E. Hinton, N.M. Radford, Z.S. Richard // Neural Computation. 1995 Vol. 7 (5). P. 889-904.
21. Devansh A. Why Regularized Auto-Encoders learn Sparse Representation? / A. Devansh, Z. Yingbo, N. Hung, Venu G. // – 2015. –arXiv:1505.05561 [stat.ML].

22. *Dilokthanakul N.* Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders / *N., Dilokthanakul, P.A.M., Mediano, M. Garnelo, M., Lee, H. Salimbeni, K. Arulkumaran, M. Shanahan* // – 2017. arXiv:1611.02648 [cs.LG].
23. *Ehsan A.M.* Infinite Variational Autoencoder for Semi-Supervised Learning / *A.M. Ehsan, A. Dick, A. Hengel* // – 2017 – P. 5888-5897.
24. *Galvão M* Forecasting Movie Box Office Profitability / *M. Galvão, R. Henriques* // INFORM SYSTEMS ENG. – 2018. – Vol. 3(3). – 22 p.
25. *Ghiassi M.* Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network / *M. Ghiassi, J. Skinner, D. Zimbra* // Expert Systems with Applications. – 2013. – Vol. 40(16). – P. 6266-6282.
26. *Hinton G.E.* Deep belief networks // Scholarpedia. – 2009. – Vol. 4(5). – 5947 p.
27. *Hinton G.E.* The wake-sleep algorithm for unsupervised neural networks / *G.E. Hinton, P. Dayan, F.J. Brendan, N. Radford* // Science. – 1995. – Vol. 268(5214). – P. 1158-1161.
28. *Hinton G.E.* Reducing the Dimensionality of Data with Neural Networks / *G.E. Hinton, R.R. Salakhutdinov* // Science. – 2006. – Vol. 313(5786). – P. 504-507.
29. *Hinton G.E.* Unsupervised Learning: Foundations of Neural Computation / *G.E. Hinton, T. Sejnowski* // MIT Press. – 1999. ISBN 978-0262581684
30. *Hsu WN.* Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation / *WN. Hsu, Y. Zhang, J. Glass* // 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). – 2017. – P. 16-23.
31. *Hopfield J.J.* Neural networks and physical systems with emergent collective computational abilities // Proceedings of National Academy of Sciences. – 1982. – Vol. 79(8) – P. 2554-2558.
32. *Jang E.* A Beginner's Guide to Variational Methods: Mean-Field Approximation // – 2016

33. *Kameoka H.* Supervised Determined Source Separation with Multichannel Variational Autoencoder / *H. Kameoka, L. Li, S. Inoue, S. Makino* // Neural Computation. – 2019. – Vol. 31(9). – P. 1891–1914.
34. *Khobahi S.* Model-Aware Deep Architectures for One-Bit Compressive Variational Autoencoding / *S. Khobahi, M. Soltanalian* // – 2019. – arXiv:1911.12410 [eess.SP].
35. *Kingma D.P.* Adam: A Method for Stochastic Optimization” / *D.P. Kingma, J. Ba* // – 2014. – <i>arXiv e-prints</i>
36. *Kingma D.P.* Auto-Encoding Variational Bayes / *D.P. Kingma, M. Welling* // – 2014. arXiv:1312.6114 [stat.ML].
37. *Kingma D.P.* An Introduction to Variational Autoencoders / *D.P. Kingma, M. Welling* // Foundations and Trends in Machine Learning. – 2019 – Vol. 12(4). – P. 307–392.
38. *Kramer M.A.* Nonlinear principal component analysis using autoassociative neural networks // AIChE Journal. – 1991 – Vol. 37(2). – P. 233–243.
39. *Kussul E.* Rosenblatt Perceptrons for Handwritten Digit Recognition / *E. Kussul, T. Baidyk, L. Kasatkina, V. Lukovich* // IEEE. – 2001. – C. 1516-1520.
40. *Lovedeep G.* Medical Image Denoising Using Convolutional Denoising Autoencoders // 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). – Spain, Barcelona, 2016. – P. 241-246.
41. *Luttrell S.P.* A Bayesian analysis of self-organizing maps // Neural Computation. – 1994 – Vol. 6(5). – P. 767-794.
42. *Miguel Á.* On contrastive divergence learning / *Á. Miguel, G. Hinton* // Artificial Intelligence and Statistics. – 2005.
43. *Mitchell T.* Machine Learning. // McGraw-Hill Science/Engineering/Math. – 1997. ISBN 0-07-042807-7.
44. *Mohri M.* Foundations of Machine Learning / *M. Mohri, A. Rostamizadeh, A. Talwalkar* // The MIT Press. – 2012. ISBN 9780262018258.
45. *Oghina A.* Predicting IMDB movie ratings with social media / *A. Oghina, M. Breuss, M. Tsagkias, M. Rijke* // 34th European Conference on IR Research ECIR 2012: Advances in Information Retrieval. – 2012. – P. 503-507.

46. *Prag J.* An empirical study of the determinants of revenues and marketing expenditures in the motion picture industry / *J. Prag, J. Casavant* // Journal of Cultural Economics. – 1994. – Vol. 18(3). – P. 217–235.
47. *Rhee, T.* (2016). Predicting Movie Box Office Gross: A Neural Network Approach / *T. Rhee, F. Zulkernine* // 15th IEEEInternational Conference on Machine Learning and Applications. – 2016. – P. 665-670.
48. *Rosenblatt F.* The Perceptron: A Probabilistic Model for Information Storage and Organization // the Brain, Cornell Aeronautical Laboratory, Psychological Review. – 1958. – Vol. 65, No. 6. – P. 386-408.
49. *Rosenblatt F.* Principles of Neurodynamic: Perceptrons and the Theory of Brain Mechanisms // Washington, DC: Spartan Books. – 1965. – 480 p.
50. *Russell S. J.* Artificial Intelligence: A Modern Approach, Third Edition / *S.J. Russell, P. Norvig* // Prentice Hall. – 2010. ISBN 9780136042594.
51. *Sharda R.* Predicting box-office success of motion pictures with neural networks / *R. Sharda, D. Delen* // Expert Systems with Applications. – 2006. – Vol. 30. – P. 243–254.
52. *Sharda R.* Predicting the financial success of Hollywood movies using an information fusion approach / *R. Sharda, D. Delen* // Industrial Engeneering Journal. – 2010. – Vol. 21. – P. 30–38.
53. *Smolensky P.* Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory // MIT Press. – 1986. – P. 194-281.
54. *Tzu-Hsi S.* Hybrid deep autoencoder with Curvature Gaussian for detection of various types of cells in bone marrow trephine biopsy images / *S. Tzu-Hsi, V. Sanchez, E. Hesham, M. Nasir* // 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI). – 2017. – P. 1040–1043.
55. *Weidi X.* Variational Autoencoder for Semi-Supervised Text Classification / *X. Weidi, S. Haoze, D. Chao, T. Ying* // Proceedings of the AAAI Conference on Artificial Intelligence. – 2017. – Vol. 31(1).
56. *Yasnitsky, L.N.* Intelligent System for Prediction Box Office of the Film / *L.N. Yasnitsky, I.A. Mitrofanov, M.V. Immis* // Lecture Notes in Networks and Systems. – 2020. – Vol. 78. – P. 18-25.

57. BeautifulSoup 4.9.0 documentation [Электронный ресурс]. Режим доступа: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>, свободный (дата обращения 10.05.2022).
58. Box Office Mojo 2022. [Электронный ресурс]. Режим доступа: <https://www.boxofficemojo.com/>, свободный (дата обращения 20.03.2022).
59. Internet Movie Database, 2022. [Электронный ресурс]. Режим доступа: <https://www.imdb.com/>, свободный (дата обращения 20.03.2022).
60. Kinopoisk, 2022. [Электронный ресурс]. Режим доступа: <https://www.kinopoisk.ru/>, свободный (дата обращения 20.03.2022).
61. Matplotlib: Visualization with Python 3.5.2 Documentation [Электронный ресурс]. Режим доступа: <https://matplotlib.org/stable/index.html>, свободный (дата обращения 20.05.2022).
62. Metacritic, 2022. [Электронный ресурс]. Режим доступа: <https://www.metacritic.com/>, свободный (дата обращения 20.03.2022).
63. Motion Picture Association (MPA). 2021 THEME Report [Электронный ресурс]. Режим доступа: <https://www.motionpictures.org/wp-content/uploads/2022/03/MPA-2021-THEME-Report-FINAL.pdf>, свободный (дата обращения 10.03.2022).
64. Python Multiprocessing Documentation [Электронный ресурс]. Режим доступа: <https://docs.python.org/3/library/multiprocessing.html>, свободный (дата обращения 20.05.2022).
65. OMG Unified Modeling Language (OMG UML). Version 2.5.1. OMG Document Number: formal/2017-12-05 [Электронный ресурс]. Режим доступа: <https://www.omg.org/spec/UML/2.5.1/PDF>, свободный (дата обращения 18.01.2022).
66. Rotten Tomatoes, 2022. [Электронный ресурс]. Режим доступа: <https://www.rottentomatoes.com/>, свободный (дата обращения 20.03.2022).
67. The Numbers. Domestic Movie Theatrical Market Summary 1995 to 2022 [Электронный ресурс]. URL: <https://www.the-numbers.com/market/>, (дата обращения 10.03.2022)
68. Twitter, 2022. [Электронный ресурс]. Режим доступа: <https://twitter.com/>, свободный (дата обращения 20.03.2022).

69. What is Artificial Intelligence? FAQ от Джона Маккарти, 2007. [Электронный ресурс]. Режим доступа: <http://www-formal.stanford.edu/jmc/whatisai/whatisai.html>, свободный (дата обращения 20.05.2022).
70. Wikipedia, the free encyclopedia, 2022. [Электронный ресурс]. Режим доступа: <https://wikipedia.org/>, свободный (дата обращения 20.05.2022).
71. YouTube, 2022. [Электронный ресурс]. Режим доступа: <https://www.youtube.com/>, свободный (дата обращения 20.03.2022).

ПРИЛОЖЕНИЕ А

Иллюстрации к введению

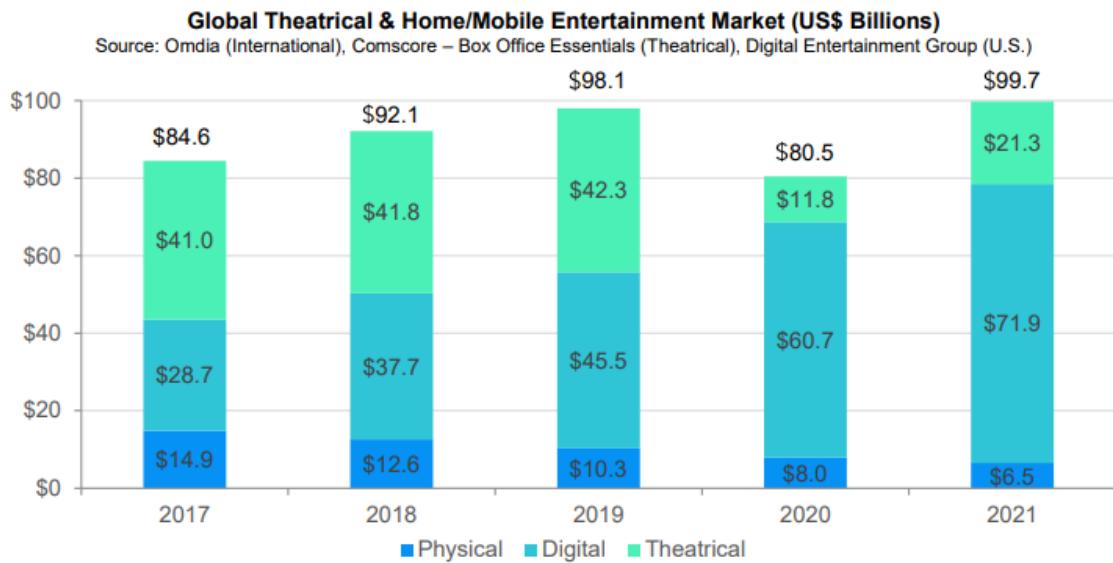


Рисунок А.1 – Суммарная прибыль мирового рынка кинопроката



Рисунок А.2 – Доли Театрального, Цифрового и Физического кинопроката

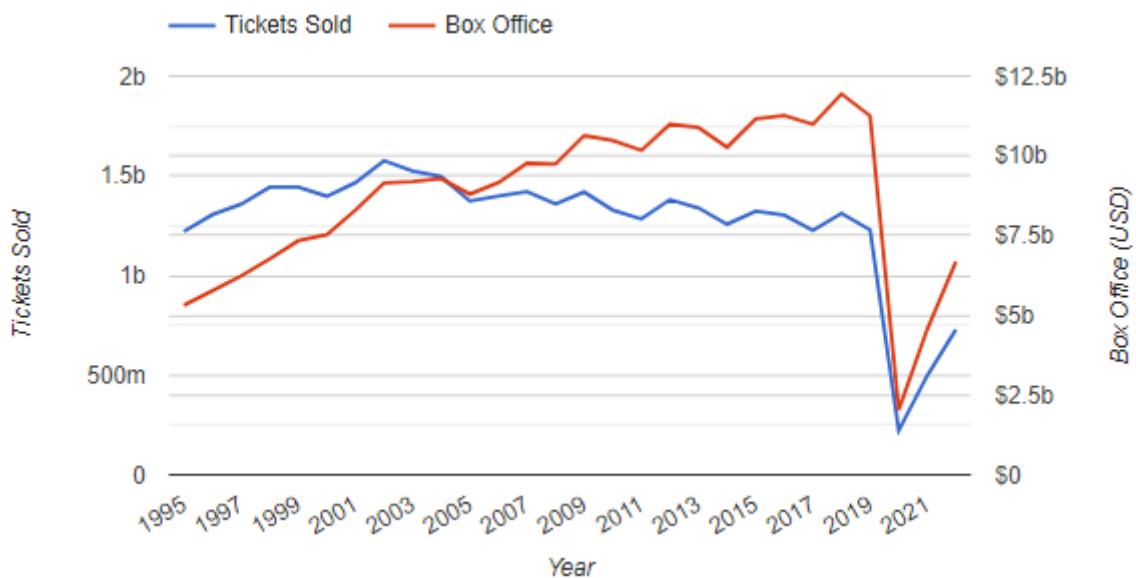


Рисунок А.3 – Ежегодные продажи билетов

Year	Tickets Sold	Total Box Office	Total Inflation Adjusted Box Office	Average Ticket Price
2022	728,275,407	\$6,678,285,481	\$6,678,285,481	\$9.17
2021	497,907,558	\$4,565,813,637	\$4,565,813,740	\$9.17
2020	221,762,724	\$2,033,566,047	\$2,033,566,047	\$9.17
2019	1,228,763,382	\$11,255,475,182	\$11,267,760,303	\$9.16
2018	1,311,300,934	\$11,945,954,034	\$12,024,629,572	\$9.11
2017	1,225,639,761	\$10,993,991,460	\$11,239,116,609	\$8.97
2016	1,302,556,379	\$11,267,115,924	\$11,944,442,006	\$8.65
2015	1,323,267,005	\$11,155,143,861	\$12,134,358,439	\$8.43
2014	1,257,220,182	\$10,271,492,042	\$11,528,709,079	\$8.17
2013	1,339,168,926	\$10,887,446,341	\$12,280,179,054	\$8.13
2012	1,380,921,942	\$10,992,141,616	\$12,663,054,206	\$7.96
2011	1,282,915,169	\$10,173,519,704	\$11,764,332,102	\$7.93
2010	1,328,549,022	\$10,482,254,025	\$12,182,794,533	\$7.89
2009	1,418,567,388	\$10,639,257,284	\$13,008,236,683	\$7.50
2008	1,358,042,073	\$9,750,744,148	\$12,453,245,819	\$7.18
2007	1,420,036,680	\$9,769,854,914	\$13,021,736,356	\$6.88
2006	1,398,738,283	\$9,161,738,221	\$12,826,430,051	\$6.55
2005	1,372,980,280	\$8,800,805,718	\$12,590,229,167	\$6.41
2004	1,495,651,298	\$9,287,996,519	\$13,715,122,406	\$6.21
2003	1,524,589,620	\$9,193,277,289	\$13,980,486,821	\$6.03
2002	1,575,789,488	\$9,155,338,716	\$14,449,989,603	\$5.81
2001	1,465,874,205	\$8,296,849,636	\$13,442,066,488	\$5.66
2000	1,397,460,079	\$7,532,311,479	\$12,814,708,924	\$5.39
1999	1,444,664,086	\$7,338,894,852	\$13,247,569,672	\$5.08
1998	1,443,848,522	\$6,771,650,563	\$13,240,090,953	\$4.69
1997	1,357,375,176	\$6,230,352,944	\$12,447,130,358	\$4.59
1996	1,305,221,290	\$5,769,078,886	\$11,968,833,756	\$4.42
1995	1,221,731,527	\$5,314,532,839	\$11,203,278,099	\$4.35

Рисунок А.4 – Ежегодные продажи билетов в виде таблицы

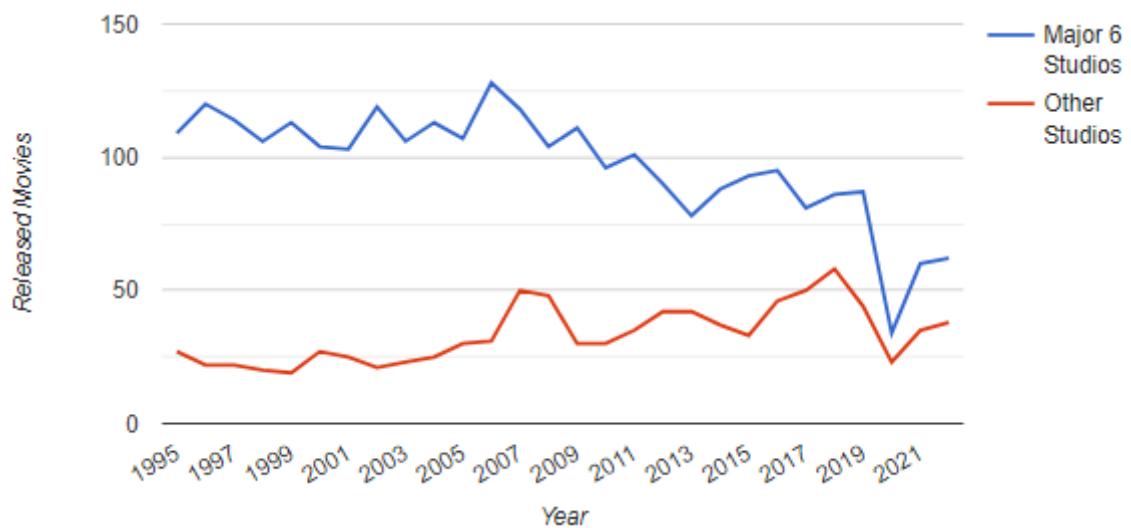


Рисунок А.5 – Количество ежегодно выпускаемых кинокартин

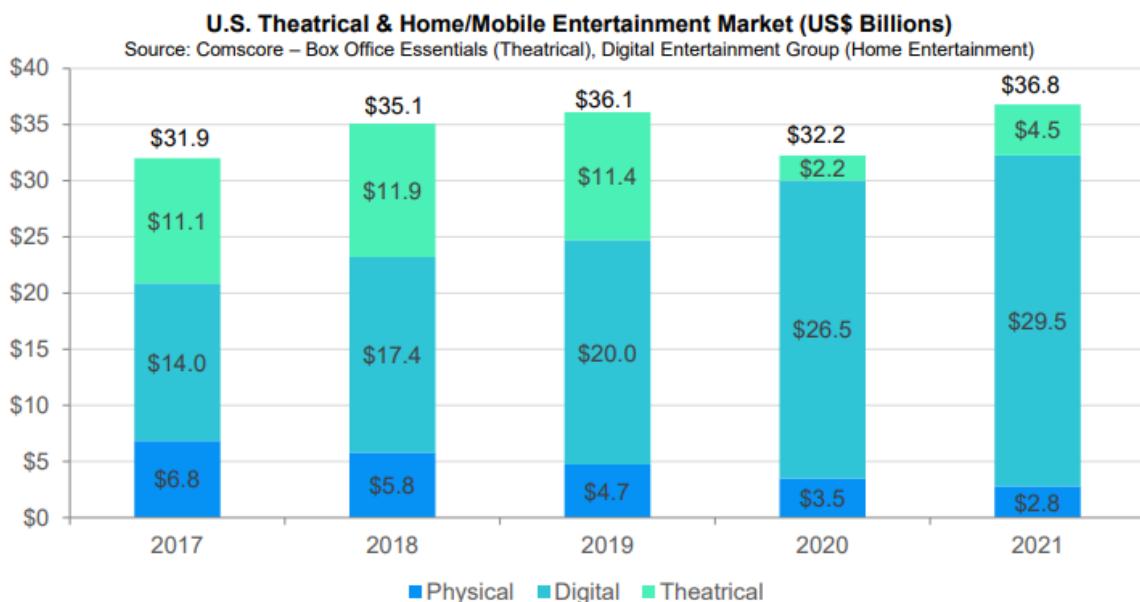


Рисунок А.6 – Суммарная прибыль рынка кинопроката в США

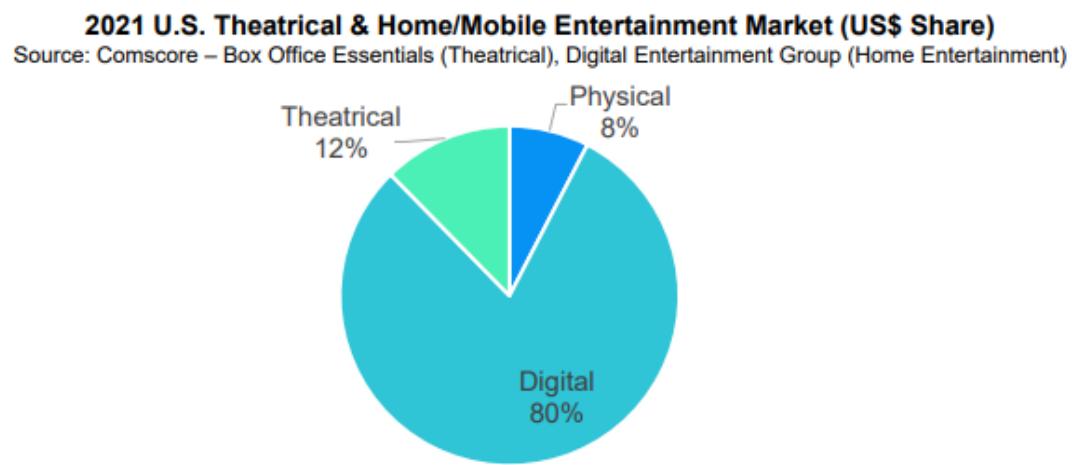


Рисунок А.7 – Доли Театрального, Цифрового и Физического кинопроката в США

Таблица В.1 – Сравнение вариантов построения нейронных сетей 1 части

Название	Сфера применения	Вид нейронов	Вид связей	Обучение	Возможности модели	Ограничения модели
<i>Персепtron</i>	Прогнозирование, управление агентами, слабая возможность классификации	Бинарное состояние (0, 1)	A-R связи	Изменение весовых коэффициентов связей A-R {−1; 0; +1}	Способность обучения простому алгоритму	Качество прогноза и точность построенной модели зависит от числа знаний, используемых при построении модели
<i>Нейронная сеть Хопфилда</i>	Автоматизация (САМ), проблема коммивояжера, оптимизация	Бинарное состояние { 0 (или -1), если x отрицательно, в противном случае 1 }	Первый слой с симметричными весами без самостоятельных связей	Отображено в формуле (1.2)	Наследует уравнения физических систем	Относительно небольшой объём памяти
<i>МашинаБольцмана</i>	Автоматизация (САМ), оптимизация	«То же что и в нейронной сети Хопфилда»	2-слойный симметричными весами. 1 скрытый и 1 видимый	Отображено в формуле (1.4)	«То же что и в нейронной сети Хопфилда»	Боковые соединения усложняют тренировку
<i>RBM</i>	Распознавание образов (цифр), распознавание речи	«То же что и в нейронной сети Хопфилда»	«То же что и в машине Больцмана без боковых соединений»	Отображено в формуле (1.6)	«То же что и в машине Больцмана с более быстрым обучением»	Увеличение кол-ва итераций из-за равновесия

Таблица В.2 – Сравнение вариантов построения нейронной сети 2 части

Название	Сфера применения	Вид нейронов	Вид связей	Обучение	Возможности модели	Ограничения модели
DBN	Распознавание и творческое воображение	«То же что и в нейронной сети Хопфилда»	Верхний слой симметричный остальные нет	То же что и в RBM	Быстрая тренировка. Иерархия уровня функций	Сложность обучения из-за вещественных нейронов
Машинна Гельмгольца	Создание представлений, мимикрия	«То же что и в нейронной сети Хопфилда»	3 слоя: асимметричные веса. 2 сети объединены в одну	Тренировка 2 фазы бодрствования-сна	Анатомический –	–
Автокодировщик	Перевод, улучшение размытых изображений, уменьшение шума данных	Локальные восприимчивые поля	Трехслойный, повторяющиеся слои для NLP.	Метод обратного распространения ошибки восстановления	–	–
VAE	Генерация реалистичных данных	средний слой нейронов кодирует средние значения и отклонения для функции Гаусса.	3 слоя: вход, кодировщик, декодер	Изменение параметров скрытого состояния для метода обратного распространения ошибки	–	–

Таблица В.3 – Сравнение вариантов построения нейронной сети 3 части

Название	Сфера применения	Вид нейронов	Вид связей	Обучение	Возможности модели	Ограничения модели
<i>Нейронная сеть Коско</i>	Выявление ассоциаций	«То же что и в нейронной сети Хопфилда»	«То же что и в нейронной сети Хопфилда» с возможностью обобщения	«То же что и в нейронной сети Хопфилда»	Адаптивность	–
<i>Нейронная сеть Джордана</i>	«То же что и в Персептроне»	«То же что и в Персептроне», добавление задержки в выходном векторе	«То же что и в Персептроне»	«То же что и в Персептроне»	Нейроны имеют обратную связь	–
<i>Нейронная сеть Хэмминга</i>	Классификация бинарных векторов, распознавание изображений	«То же что и в нейронной сети Хопфилда»	Трехслойная нейронная сеть с обратной связью	«То же что и в нейронной сети Хопфилда»	–	–
<i>Нейронная сеть Элмана</i>	«То же что и в нейронной сети Хопфилда»	«То же что и в нейронной сети Хопфилда»	Нейроны имеют обратную связь	«То же что и в нейронной сети Хопфилда»	Запоминание последовательностей	–
<i>Свёрточная нейронная сеть</i>	Распознавание и классификация изображений	«То же что и в Персептроне»	3 слоя: слой свертки, слой активации, слой субдискретизации	Метод обратного распространения ошибки	Малое количество настраиваемых весов	Слишком много варьируемых параметров сети

ПРИЛОЖЕНИЕ С

Сравнение алгоритмов машинного обучения

Обучение с учителем

Обучение с учителем – один из способов машинного обучения, в ходе которого испытуемая система принудительно обучается с помощью попарных прецедентов (данные, ответ). Существует неизвестная зависимость между ответами и объектами, но она неизвестна, известна только конечная совокупность прецедентов [50]. Под учителем понимается либо сама обучающая выборка, либо тот, кто указал на заданных объектах правильные ответы. Требуется найти функциональную зависимость ответов от описаний объектов и построить алгоритм, принимающий на входе описание объекта и выдающий на выходе ответ. Для измерения точности ответов, может вводиться функционал качества, который обычно определяется как средняя ошибка ответов, выданных алгоритмом, по всем объектам выборки [44].

С точки зрения кибернетики, является одним из видов кибернетического эксперимента. Данный эксперимент представляет собой частный случай кибернетического эксперимента с обратной связью. Постановка данного эксперимента предполагает наличие экспериментальной системы, метода обучения и метода испытания системы или измерения характеристик [33].

Экспериментальная система в свою очередь состоит из испытываемой «используемой» системы, пространства стимулов, получаемых из внешней среды, и системы управления подкреплением «регулятора внутренних параметров». В качестве системы управления подкреплением может быть использовано автоматическое регулирующее устройство или учитель, способный реагировать на реакции испытываемой системы и стимулы внешней среды путём применения особых правил подкрепления, изменяющих состояние памяти системы.

Различают два варианта:

- Реакция испытуемой системы не изменяет состояние внешней среды
(Рисунок С.1).



Рисунок С.1 – Визуальное описание обучения 1 вариант

- Реакция системы изменяет стимулы внешней среды (Рисунок С.2).

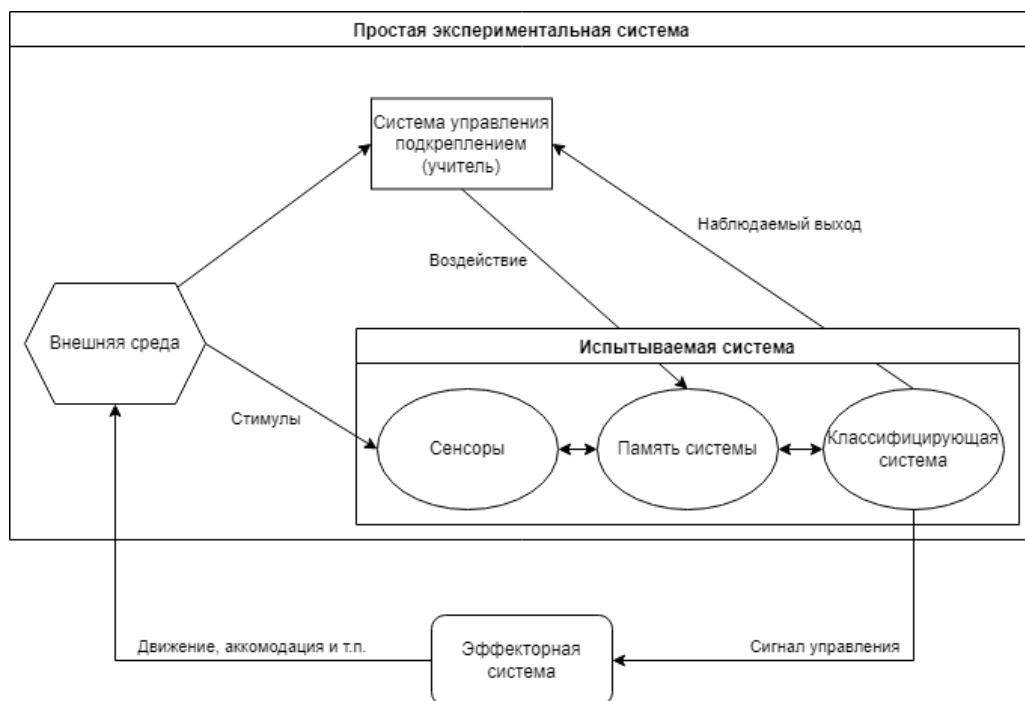


Рисунок С.2 – Визуальное описание обучения 2 вариант

Эти схемы указывают принципиальное сходство такой системы общего вида с биологической нервной системой.

Типы входных данных:

- Признаковое описание или матрица объекты-признаки – каждый объект описывается набором своих характеристик-признаков, которые могут быть числовыми или нечисловыми.

- Матрица расстояний между объектами – каждый объект описывается расстояниями до всех остальных объектов обучающей выборки. С этим типом входных данных работают немногие методы, в частности, метод ближайших соседей, метод парзеновского окна, метод потенциальных функций.
- Временной ряд или сигнал – последовательность измерений во времени, где каждое измерение может представляться числом, вектором, а в общем случае – признаком описанием исследуемого объекта в данный момент времени.
- Изображение или видеоряд.

Типы откликов:

- Задача классификации и распознавания образов – множество допустимых ответов конечно.
- Задача регрессии и аппроксимации – множество возможных ответов бесконечно (ответы являются действительными числами или векторами).
- Задача ранжирования – ответы надо получить сразу на множестве объектов, после чего отсортировать их по значениям ответов. Часто применяется в информационном поиске и анализе текстов.
- Задача прогнозирования – ответы характеризуют будущие поведения процесса или явления. Объектами являются отрезки временных рядов, обрывающиеся в тот момент, когда требуется сделать прогноз на будущее.

Методы решения:

- Кластеризация:
 - Линейный дискриминант Фишера.
 - Метод парзеновского окна.
 - Разделение смеси вероятностных распределений или ЕМ-алгоритм.
 - Метод потенциальных функций или метод радиальных базисных функций.
 - Метод ближайших соседей.
- Нейронная сеть:
 - Персепtron.
 - Многослойный персепtron.

- Сети векторного квантования, обучаемые с учителем (Learning Vector Quantization).
- Гибридная сеть встречного распространения.
- Линейный разделитель:
 - Линейный дискриминант Фишера.
 - Однослойный персепtron.
 - Логистическая регрессия.
 - Машина опорных векторов.
- Индукция правил:
 - Решающее дерево.
 - Решающий список.
 - Решающий лес.
 - Тестовый алгоритм.
 - Алгоритм вычисления оценок.
- Алгоритмическая композиция:
 - Взвешенное голосование.
 - Бустинг.⁷
 - Бэггинг.⁸
 - Метод комитетов.
 - Смесь экспертов.
- Сокращение размерности:
 - Селекция признаков.
 - Метод главных компонент.
 - Метод независимых компонент.
 - Многомерное шкалирование.
- Выбор модели:
 - Минимизация эмпирического риска.
 - Структурная минимизация риска.

⁷ Бустинг – метаалгоритм машинного обучения, применяется, главным образом, для уменьшения смещения (погрешности оценки), а также дисперсии в обучении с учителем.

⁸ Бэггинг – технология классификации, где в отличие от бустинга все элементарные классификаторы обучаются и работают параллельно

- Минимум длины описания.
- Скользящий контроль.
- Извлечение признаков.
- Самоорганизация моделей.
- Случайный поиск с адаптацией.
- Генетический алгоритм.

Обучение без учителя

Обучение без учителя – один из способов машинного обучения, при котором испытуемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора. Алгоритм изучает шаблоны из неотмеченных данных и с помощью мимикрии создает компактное внутреннее представление своего мира, а затем генерирует из него образный контент [22, 30]. В отличие от контролируемого обучения, где данные помечаются экспертом, например, неконтролируемые методы демонстрируют самоорганизацию, которая фиксирует закономерности в виде плотности вероятности или комбинации предпочтений нейронных функций. Как правило, это пригодно только для задач, в которых известны описания множества обучающей выборки и требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами.

Обучение без учителя часто противопоставляется обучению с учителем, когда для каждого обучающего объекта принудительно задаётся «правильный ответ», и требуется найти зависимость между стимулами и реакциями системы [29].

Типы входных данных:

- Признаковое описание объектов – объект описывается набором своих характеристик, называемых признаками. Признаки могут быть числовыми или нечисловыми.
- Матрица расстояний между объектами – объект описывается расстояниями до всех остальных объектов обучающей выборки.

Типы откликов:

- Задача кластеризации – выборка объектов разбивается на непересекающиеся подмножества, называемые кластерами, так, чтобы

каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались. Исходная информация представляется в виде матрицы расстояний.

- Задачи обобщения – пример спонтанного обобщения, при котором критерии подобия не вводятся извне или не навязываются экспериментатором. Математически может быть сформулировано как кластеризация.
- Задачи обнаружения аномалий – обнаружение участков данных, на которых поведение объекта значительно отличается от ожидаемого поведения.
- Задачи поиска правил ассоциации – нахождение наборов признаков, и значений этих признаков, которые особенно часто встречаются в признаковых описаниях объектов. Исходная информация представляется в виде признаковых описаний.
- Заполнение пропущенных значений – заполнение отсутствующих значений построением алгоритма, заполняющего пропущенные значения прогнозами в зависимости от других признаков. Исходная информация представляется в виде признаковых описаний.
- Задачи сокращения размерности – представление данных в пространстве меньшей размерности, по возможности, минимизировав потери информации. Исходная информация представляется в виде признаковых описаний, причём число признаков может быть достаточно большим.
- Задачи визуализации данных - отображение многомерных данных в виде плоских графиков, что способствует лучшему пониманию данных и самой сути решаемой задачи.

Методы решения:

- Кластеризация
 - Графовые алгоритмы кластеризации.
 - Статистические алгоритмы кластеризации.
 - Иерархическая кластеризация или таксономия.
 - Нейронная сеть Кохонена.
 - Нейронная сеть встречного распространения.

- Метод радиальных базисных функций.
- Ассоциация:
 - Анализ рыночных корзин.
- Сокращение размерности:
 - Метод главных компонент.
 - Метод независимых компонент.
 - Многомерное шкалирование.
- Визуализация:
 - Дендрограмма.
 - Самоорганизующаяся карта Кохонена.
 - Упругие карты.
 - Карта сходства.

Частичное обучение

Частичное обучение (Обучение с частичным привлечением учителя) – способ машинного обучения, занимающий промежуточную позицию между обучением с учителем и обучением без учителя, использует небольшое количество размеченных данных и большое количество неразмеченных данных для тренировки. Обоснование способа основано на обнаружении исследователей машинного обучения, в которых неразмеченные данные, при использовании в сочетании с небольшим количеством размеченных данных, могут значительно улучшить точность обучения [23, 55].

Задание размеченных данных для задачи обучения часто требует квалифицированного человека (например, для перевода звуковой дорожки в текст) или физического эксперимента (например, для определения 3D структуры белка или выявления наличия нефти в определенном регионе). Поэтому затраты на разметку данных могут сделать процесс обучения с использованием лишь размеченных данных невыполнимым, в то время как процесс задания неразмеченных данных не является очень затратным. В таких ситуациях, полуавтоматическое обучения может иметь большое практическое значение. Такое обучение также представляет интерес в сфере машинного обучения и как модель для человеческого обучения.

Типы входных данных:

- Признаковое описание объектов – объект описывается набором своих характеристик, называемых признаками. Признаки могут быть числовыми или нечисловыми.
- Изображение или видеоряд.

Типы откликов:

- Задача плавности – нахождение отсутствующих элементов в области высокой плотности, лежащих близко друг к другу.
- Задача кластеризации – выборка объектов разбивается на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались.
- Задачи сокращения размерности – представление данных в пространстве меньшей размерности, по возможности, минимизировав потери информации.

Методы решения:

- Метод опорных векторов (SVM)
- Метод полуавтоматических опорных векторов (S3VM)
- Метод самообучения
- Генеративные модели

Трансдуктивное обучение

Трансдуктивное обучение – полу-контролируемое обучение, разновидность частичного обучения, когда прогноз предполагается делать только для прецедентов из тестовой выборки.

В логике и статистическом методе – трансдуктивное умозаключение или трансдуктивный метод являются выводами о наблюдаемых частных случаях тестовых данных на основании частных тестовых случаев данных обучения. Напротив, индуктивное умозаключение приводит наблюдаемые частные случаи обучения к общим правилам, которые затем применяются к тестовым случаям. Различие наиболее интересно в тех случаях, когда прогнозы трансдуктивной модели не достижимы ни одной индуктивной моделью, такие ситуации происходят, когда в

результате трансдуктивного вывода на различных испытательных выборках получаются взаимопротиворечивые прогнозы.

Примером обучения, не являющегося индуктивным, может быть случай двоичной классификации, в котором входные данные склонны разделяться на две группы. Большой объём контрольных данных может помочь в поиске кластеров, давая полезную информацию о метках классов. Примером алгоритма этой категории может послужить трансдуктивная машина опорных векторов (Transductive Support Vector Machine, TSVM). Еще одна причина, ведущая к трансдукции, возникает при необходимости в приближении. Если построение точного ответа вычислительно невозможно, то можно по крайней мере попытаться убедиться в том, что приближения хороши на тестовых данных. В этом случае тестовые данные могут иметь произвольное распределение, что недопустимо в случае частичного обучения. Примером алгоритма, подпадающего под эту категорию, может являться Машина Байесовых Комитетов (Bayesian Committee Machine, BCM).

Обучение с подкреплением

Обучение с подкреплением – один из способов машинного обучения, в ходе которого испытуемая система (агент) обучается, взаимодействуя с некоторой средой. С точки зрения кибернетики, является одним из видов кибернетического эксперимента. Откликом среды на принятые решения являются сигналы подкрепления, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или её модель. Также нужно иметь в виду, что некоторые правила подкрепления базируются на неявных учителях, например, в случае искусственной нейронной среды, на одновременной активности формальных нейронов, из-за чего их можно отнести к обучению без учителя.

При обучении с подкреплением, в отличии от обучения с учителем, не представляются верные пары (данные, ответ), а принятие субоптимальных решений (дающих локальный экстремум) не ограничивается явно. Обучение с подкреплением пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний. Баланс изучения-применения при обучении с подкреплением исследовался в задаче многорукого бандита.

Методы решения:

Генетический алгоритм – алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений, с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как наследование, мутации, отбор и кроссинговер. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

Динамическое обучение

Динамическое обучение – может быть, как обучением с учителем, так и без учителя. Специфика в том, что прецеденты поступают потоком. Требуется немедленно принимать решение по каждому прецеденту и одновременно доучивать модель зависимости с учётом новых прецедентов. Как и в задачах прогнозирования, здесь существенную роль играет фактор времени.

Мета-обучение

Мета-обучение – отличается тем, что прецедентами являются ранее решённые задачи обучения. Основная идея мета-обучения – свести задачу выбора алгоритма к задаче обучения с учителем, задачи описываются мета-признаками. Мета-признак описывает свойство задачи — например, разрежен ли датасет или нет, число категориальных или численных признаков объектов в датасете, число возможных меток, размер датасета и многое другое. Требуется определить, какие из используемых в них эвристик работают более эффективно. Конечная цель – обеспечить постоянное автоматическое совершенствование алгоритма обучения с течением времени.

Иногда к метаобучению ошибочно относят построение алгоритмических композиций, в частности, бустинг, однако в композициях несколько алгоритмов решают одну и ту же задачу, тогда как метаобучение предполагает, что решается много разных задач.

Методы решения:

- Многозадачное обучение (multi-task learning) – набор взаимосвязанных или схожих задач обучения решается одновременно, с помощью различных алгоритмов обучения, имеющих схожее внутренне представление. Информация о сходстве задач между собой позволяет более эффективно совершенствовать алгоритм обучения и повышать качество решения основной задачи.
- Индуктивный перенос (inductive transfer) – опыт решения отдельных частных задач обучения по прецедентам переносится на решение последующих частных задач обучения. Для формализации и сохранения этого опыта применяются реляционные или иерархические структуры представления знаний.

ПРИЛОЖЕНИЕ D

Техническое задание на разрабатываемую систему

СИСТЕМА МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПРОГНОЗИРОВАНИЯ РЕНТАБЕЛЬНОСТИ КИНОБИЗНЕСА

Техническое задание

Инв. № полз.	Подпись и дата	Взам. инв. №	Инв. № дубл.	Подпись и дата

Руководитель разработки
_____ Ясницкий Л.Н.
“10” марта 2022

Ответственный исполнитель
_____ Чепоков Е.С.
“10” марта 2022

1. Общие сведения

Наименование программы – «Система машинного обучения для прогнозирования рентабельности кинобизнеса». Программа является алгоритмом, позволяющим обучить нейронную сеть на примерах для прогнозирования кассовых сборов кинокартин. Система позволяет внести свои данные, не представленные в обучающем множестве, для нахождения оптимального ответа. Использование машинного обучения позволяет снизить инвестиционные риски при производстве кинофильмов.

Целью создания системы является получение инструмента с возможностью статистического прогнозирования кассовых сборов выпускаемых кинокартин и исследования результата от вариации входных данных, для пользователей, не являющихся аналитиками и/или ИТ-специалистами. Разрабатываемый инструмент должен использовать эффективные алгоритмы для повышения скорости выполнения операций и иметь сравнительно малую погрешность прогнозирования.

2. Основание для разработки

Основанием для разработки являются:

- Рабочий учебный план по направлению 09.03.04 «Программная инженерия» факультета экономики, менеджмента и бизнес-информатики, НИУ ВШЭ-Пермь на 4 курс, 2021/2022 учебный год, утвержденный проректором НИУ ВШЭ С. Ю. Рошиным от 16.04.2021.
- Правила подготовки выпускной квалификационной работы студентов основной образовательной программы бакалавриата «Программная инженерия» по направлению подготовки 09.03.04. Программная инженерия, утвержденные протоколом ученого совета НИУ ВШЭ – Пермь от 19.11.2020 № 8.2.1.7-10/10.

3. Назначение разработки

Система предназначена для прогнозирования кассовых сборов фильма, а также исследования вариации результата от входных параметров.

Функциональным назначением программы является предоставление пользователю прогнозируемой информации касательно кассовых сборов кинокартин в зависимости от переменных предоставленных пользователем о рассматриваемой кинокартине.

Программа предназначена для эксплуатации на ранних этапах процесса создания и аналитического обзора кинокартин. Пользователями программы могут быть как разработчики и IT-специалисты, так и эксперты предметных областей, а также обычные пользователи.

4. Требования к программе или программному изделию

Для разрабатываемой системы определены функциональные требования, требования к надежности, требования к условиям эксплуатации, к составу и параметрам технических средств. Требования к маркировке и упаковке, а также требования к транспортированию и хранению не предъявляются, так как поставка системы предполагается в виде открытого исходного кода, доступного для скачивания через веб-сервис для хостинга IT-проектов GitHub.

4.1. Требования к функциональным характеристикам

Все функции системы можно разделить на следующие группы в зависимости от их назначения:

1. Взаимодействие с обучающим множеством:
 - a. Функции сбора данных с сайтов-агрегаторов;
 - b. Функции очистки выбросов из собранных данных;
 - c. Функции нормализации собранных данных;
 - d. Функции систематизации данных;
 - e. Функции графического отображения данных;
 - f. Функции разделения множества данных, на обучающее и тестирующее подмножества;
 - g. Функции преобразования собранных данных в базу данных для последующего экспорта;
 - h. Функции экспорта собранных обучающих данных, в форматах: «XSLX», «XLS», «CSV», «TXT».

2. Взаимодействие с нейронной сетью:
 - a. Функции создания нейронной сети;
 - b. Функции импорта и обработки обучающих и тестирующих данных;
 - c. Функции модификации параметров обучения нейронной сети;
 - d. Функции графического отображения результатов обучения нейронной сети;
 - e. Функции группировки моделей, реализация переиспользования элементов обучения;
 - f. Функции экспорта обученной нейронной сети в формате «HDF5».
3. Управление обученной нейронной сетью:
 - a. Функции хранения результатов обучения нейронной сети;
 - b. Функции импорта/экспорта данных в форматах: «XSLX», «XLS», «CSV», «TXT».
4. Выполнение работы нейронной сетью:
 - a. Функции записи вводимых пользователем значений;
 - b. Функции заполнения данными, для вариативного анализа;
 - c. Функции хранения собранных данных;
 - d. Функции экспорта данных в обученную нейронную сеть;
 - e. Функции графического отображения результатов работы нейронной сети.

Ввод и вывод данных должен осуществляться с использованием пользовательского визуального интерфейса или напрямую с нейронной сетью, через использование консоли и файлов в форматах: «XSLX», «XLS», «CSV», «TXT». Импортируемые и экспортируемые входные данные и результаты работы предоставляются и хранятся в виде отдельных файлов, размещаемых на локальных или съемных носителях в перечисленных выше форматах. Файлы указанного формата должны размещаться на носителях, отформатированных согласно требованиям операционной системы.

4.2. Требования к надежности

Надежность ИС зависит от надежности используемых технических средств и операционной системы. Устойчивое функционирование программы должно быть

обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

- 1) Организацией бесперебойного питания технических средств;
- 2) Использованием лицензионного программного обеспечения, необходимого для запуска приложения, включая лицензионную операционную систему;
- 3) Регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
- 4) Регулярным выполнением требований ГОСТ 51188-98 «Защита информации. Испытания программных средств на наличие компьютерных вирусов»;
- 5) Резервным копированием данных программы, включая версии моделей, метамоделей, правила трансформации и объединяющие их проекты, онтологии базы знаний, а также возможностью использования системы управления версиями (VCS).

Пользователю, работающему с программой через веб-браузер, должен быть предоставлен непрерывный доступ к веб-приложению, расположенному по определённому URL-адресу. Веб-сервис не должен непредвиденно прерывать свою работу.

Контроль входной и выходной информации осуществляется посредством их валидации и верификации. Так, осуществляются следующие проверки:

- 1) Проверка корректности введенных пользователем данных и их полноты: если хотя бы одно, обязательное к заполнению, поле осталось незаполненным, выполнение работы нейронной сети блокируется, а пользователю предлагается изменить введенные данные, чтобы они соответствовали необходимому вводу;
- 2) Проверка корректности данных переданных автозаполнителем для сравнительного анализа: при несоответствии формата введенных данных с необходимым, система переписывает данные автозаполнителя, при

повторном срабатывании, система заменяет предыдущие данные на новые, близкие по значению;

- 3) Проверка корректности файлов нейронной сети, формата «HDF5»: в случае некорректности, повреждения или отсутствия файла, пользователь извещается о невозможности импорта данного файла, после чего система старается загрузить ранее сохраненную, не использовавшуюся копию файла, в случае повторного срабатывания пользователю предоставляется возможность скачать файл заново;
- 4) Проверка корректности импортируемых и экспортируемых файлов, включая файлы с данными для работы нейронной сети и файлы с результатами срабатывания нейронной сети, в необходимых форматах («XSLX», «XLS», «CSV», «TXT»): в случае если файл поврежден или некорректен, пользователь извещается о невозможности импорта данного файла, после чего пользователю предлагается перезапись файла.

Осуществляемые проверки корректности должны предотвращать возможные отказы системы вследствие некорректных действий пользователя при взаимодействии с системой через пользовательский интерфейс. Система должна отслеживать работоспособность нейронной сети и предотвращать действия, которые могут повлечь нарушение полноты или корректности моделей.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск системы, при условии соблюдения условий эксплуатации технических и программных средств. Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств, переустановки программных средств и запуска системы.

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии со средствами, через которые осуществляется взаимодействие с системой. Во избежание возникновения отказов программы по указанным причинам для пользователей системы должно быть проведено обучение

по использованию как самой разрабатываемой системы, так и средств, которые необходимы для обращения к ней.

Отказы как самой системы, так и ее отдельных функций, могут привести к аварийному завершению работы программы, однако при перезапуске программы ее функциональность не должна пострадать. Среди возможных последствий вышеперечисленных отказов можно выделить нарушение целостности файлов нейронной сети и отдельных компонентов системы, и частичную потерю информации, вносившейся непосредственно перед отказом.

4.3. Условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

Пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы. Пользователем системы является только «Конечный пользователь» – заполняет данные о кинофильме и отправляет на автоматическую обработку нейронной сетью.

Доступ к функциям системы не зависит от роли пользователя, а также от делегируемых ему полномочий для использования определенных систем.

Программа не требует проведения каких-либо видов обслуживания кроме тех, что описаны в требованиях к надежности.

4.4. Требования к составу и параметрам технических средств

Требования к составу и параметрам технических средств предъявляются к персональным компьютерам клиентов, на которых осуществляется работа с системой.

К персональным компьютерам клиентов предъявляются следующие требования:

- Процессор с тактовой частотой не ниже 2 ГГц, при количестве ядер не менее 2;
- ОЗУ не менее 2 Гб;

- Не менее 400 Мб свободного места на жестком диске для хранения файлов системы;
- Видеоадаптер с минимальным разрешением 1024 на 1024 пикселя;
- Клавиатура;
- Компьютерная мышь.

4.5. Требования к информационной и программной совместимости

Исходные коды программы должны быть реализованы на языках программирования Python, для нейронной сети и модуля сбора обучающих данных, и C#, для пользовательского приложения, на платформе .NET Core, что позволит сделать систему кроссплатформенной. В качестве интегрированной среды разработки программы должна быть использована среда PyCharm или аналогичная ему и Microsoft Visual Studio 2019 как реализующая актуальную версию .NET Core. Система разрабатывается под любые 64х операционные системы, которые поддерживаются платформой .NET Core.

Для защиты информации и программ должна быть реализована система аутентификации пользователя для предотвращения доступа злоумышленника к программе. Система разделения ролей позволяет определить функции системы, доступные каждой роли и каждой группе пользователей, что должно предотвратить осуществление пользователем действий, на выполнение которых у пользователя не хватает полномочий. Защита информации и программ также частично обеспечивается организационно-техническими мероприятиями, описанными в требованиях к надежности.

Информационная структура файлов импортируемых и экспортируемых файлов определяется спецификацией «XSLX», «XLS», «CSV», «TXT» форматов, для хранения данных работы нейронной системы. Структура нейронной сети определяется спецификацией «HDF5»-формата.

Система должна быть разбита на слои и модули, между которыми определена слабая связанность, что должно позволить заменить один модуль другим без потери работоспособности всей системы.

4.6. Требования к маркировке и упаковке

Программа поставляется посредством веб-сервиса для хостинга IT-проектов GitHub и должна включать исходный код системы и исполняемые файлы, примеры использования системы, программную документацию и прочие необходимые для работы системы файлы, а также презентация проекта.

Помимо этого, должна быть включена маркировка с обозначением наименования Системы, ФИО исполнителя и руководителя разработки и года выпуска программы.

5. Требования к программной документации

Документы к программе должны быть выполнены в соответствии с ГОСТ 19.106–78 и ГОСТами к каждому виду документа (таблица D.1).

Таблица D.1 – Выбранные нейронные сети

Название документа	Краткое содержание
Текст программы (ГОСТ 19.401–78)	Программный код всех модулей программы с необходимыми комментариями
Программа и методика испытаний (ГОСТ 19.301–79)	Требования, подлежащие проверке при испытании программы, а также порядок и методы их контроля
Техническое задание (ГОСТ 19.201–78)	Назначение и область применения программы, технические, технико-экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний.
Руководство программиста (ГОСТ 19.504–79)	Объединенное: руководство системного программиста и руководство программиста. Содержит сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения, а также сведения для эксплуатации программы. В руководстве также описаны основные алгоритмы, используемые в данной системе.
Руководство пользователя (ГОСТ 19.505–79)	Сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе

Название документа	Краткое содержание
	выполнения программы. Описание приводится отдельно для каждой роли пользователя системы.

Каждый документ должен быть представлен в формате «PDF» или «DOCS» и расположен в архиве формата «ZIP» или «RAR».

6. Технико-экономические показатели

В рамках данной работы расчет экономической эффективности не предусмотрен, однако использование разрабатываемой системы позволит:

- Спрогнозировать кассовые сборы кинокартин, что позволит рассчитать затраты на рекламу и аренду кинозалов;
- Снизить инвестиционные риски при создании кинокартин;
- Составить анализ аспектов кинокартины основываясь на статистике, для возможной замены аспектов или избавления;
- Сократить длительность этапов кастинга и выбора сюжета, что способствует уменьшению затрат на данные этапы и уменьшению бюджета;
- Определить целесообразность выпуска кинокартин в театральный прокат
- Определить оптимальный сезон премьеры кинокартин и шанс получения кинопремии.

Выпуск кинокартины в наши дни является большим риском, не только для киностудий и режиссеров, но и для инвесторов, система прогнозирования способна снизить риски неудачных инвестиций в кинопроизводство на ранних этапах создания кинокартины, а также спрогнозировать целесообразность выпуска кинокартины на экраны кинотеатров с сопутствующими затратами на рекламу и аренду кинозалов.

В настоящее время решения о целесообразности реализации кинопроектов принимаются в большинстве случаев исключительно на основе интуиции и предыдущего опыта продюсерских организаций, что способствует неоправданному росту рисков инвесторов, вкладывающих значительные финансовые средства в кинопроизводство. Хотя, интуиция и опыт являются важнейшими профессиональными качествами продюсера, для роста обоснованности

принимаемых решений необходимо, чтобы они основывались на комплексном анализе сильных и слабых сторон кинопроекта до его запуска в производство.

Таким образом, создание системы основывается на необходимости решения важных задач, таких как, научное и статистическое обоснование и создание эффективно функционирующего механизма прогнозирования коммерческого потенциала кинопроекта, принятия рациональных управленческих решений о целесообразности его реализации, определение направлений и принципов эффективного управления кинематографическим бизнес-процессом, в частности, на ранних стадиях создания кинофильмов.

7. Стадии и этапы разработки

В рамках исследовательского прототипа предлагается реализация базовой функциональности. Стадии и этапы разработки представлены в таблице D.2.

Таблица D.2 – Таблица этапов разработки

№	Содержание работы	Срок	Исполнитель этапа
1	Постановка задачи	20.11.2021	Чепоков Е.С.
2	Описание общих сведений выбранной предметной области	22.12.2021	Чепоков Е.С.
3	Изучение нейронных сетей и их алгоритмов работы	15.01.2022	Чепоков Е.С.
4	Изучение исследований и работ в выбранной предметной области	24.01.2022	Чепоков Е.С.
5	Изучение систем аналогов нейронных сетей	05.02.2022	Чепоков Е.С.
6	Определение требований к системе, согласование и утверждение ТЗ	15.02.2022	Чепоков Е.С., Ясницкий Л.Н.
7	Определение архитектуры нейронной сети и алгоритма обучения	20.02.2022	Чепоков Е.С.
8	Определение необходимых данных для обучения нейронной сети	22.02.2022	Чепоков Е.С.
9	Проектирование модулей для обучения нейронной сети	25.02.2022	Чепоков Е.С.
10	Разработка модуля сбора обучающих данных	01.03.2022	Чепоков Е.С.
11	Разработка модуля очищения собранных данных	10.03.2022	Чепоков Е.С.
12	Тестовое обучение нейронной сети	14.03.2022	Чепоков Е.С.
13	Корректировка параметров обучения и изменение модуля сбора обучающих данных	16.03.2022	Чепоков Е.С.
14	Проведение приемо-сдаточных испытаний обученной нейронной сети	20.03.2022	Чепоков Е.С.
15	Проектирование остальных модулей, не вошедших в прототип	10.09.2022	Чепоков Е.С.

№	Содержание работы	Срок	Исполнитель этапа
16	Реализация проектных решений и отладка разработанных модулей	20.03.2023	Чепоков Е.С.
17	Интеграция различных модулей системы	01.04.2023	Чепоков Е.С.
18	Разработка необходимой документации	01.05.2023	Чепоков Е.С.
19	Проведение окончательных приемо-сдаточных испытаний	25.06.2023	Чепоков Е.С.

8. Порядок контроля и приемки

Проведение тестирования исследовательского прототипа включает модульное и интеграционное тестирование для основных разработанных функций, которые описаны выше. Помимо этого, необходимо тестирование и калибровка базовых функций нейронной сети.

Осуществление приемо-сдаточных испытаний для всей системы осуществляется на основе Программы и методики испытаний и включает:

- функциональное тестирование,
- тестирование стабильности работы,
- тестирование удобства эксплуатации,
- тестирование защиты и надежности,
- тестирование производительности системы.

ПРИЛОЖЕНИЕ Е

Описание прецедентов

Таблица Е.1 – Описание прецедента «Изменение данных»

Идентификатор и название	UC-1. «Изменение данных»
Акторы	Пользователь.
Описание	Пользователь изменяет введенные данные.
Триггер	Ввод данных пользователем.
Предварительные условия	PRE-1. Программа запущена.
Выходные условия	POST-1. Файл с набором данных, введенных пользователем.
Основной поток	1.0 Пользователь вводит данные: 1. Система проверяет правильность введенных данных. 2. Система отправляет данные на обработку.
Альтернативные потоки	-
Исключения	1.0.E1. Введенные пользователем данные не соответствуют требованиям: 1. Система выводит предупреждение о неправильности данных.
Приоритет	Низкий.
Точка расширения	UC-4. «Преобразование данных».

Таблица Е.2 – Описание прецедента «Ввод данных»

Идентификатор и название	UC-2. «Ввод данных»
Акторы	Пользователь.
Описание	Пользователь вводит данные.
Триггер	Ввод данных пользователем.
Предварительные условия	PRE-1. Программа запущена.
Выходные условия	POST-1. Файл с набором данных, введенных пользователем.
Основной поток	1.0 Пользователь вводит данные: 1. Система проверяет правильность введенных данных. 2. Система отправляет данные на обработку.
Альтернативные потоки	-
Исключения	1.0.E1. Введенные пользователем данные не соответствуют требованиям: 1. Система выводит предупреждение о неправильности данных.
Приоритет	Высокий.
Точка расширения	UC-4. «Преобразование данных».

Таблица Е.3 – Описание прецедента «Получение результата»

Идентификатор и название	UC-3. «Получение результата»
Акторы	Пользователь.
Описание	Пользователь получает результат работы нейронной сети.
Триггер	Нажатие кнопки «Получить результат».
Предварительные условия	PRE-1. Пользователь отправил данные. PRE-2. Нейронная система завершила работу над данными.
Выходные условия	POST-1. Отображение результата пользователю.
Основной поток	1.0 Пользователь нажимает на кнопку получить результат: 1. Система принимает полученные от нейронной сети данные. 2. Программа выводит результат на экран (см. 1.0.E1 и 1.1.E1).
Альтернативные потоки	1.1 Завершение работы программы.
Исключения	1.0.E1 Система не получает данные от нейронной сети: 1. Ожидание отклика нейронной сети несколько раз. 1.1.E1 Данные невозможно вывести: 2. Система сообщает об ошибке. 3. Система завершает работу (см. 1.1).
Приоритет	Высокий.
Точка расширения	-

Таблица Е.4 – Описание прецедента «преобразование данных»

Идентификатор и название	UC-4. «Преобразование данных»
Акторы	Пользователь.
Описание	Система принимает введение данные пользователем и формирует файл для отправки в нейронную сеть.
Триггер	Нажатие пользователем кнопки «Получить результат».
Предварительные условия	PRE-1. Пользователь ввел данные.
Выходные условия	POST-1. Формирование файла для нейронной сети.
Основной поток	1.0 Формирование файла с введенными данными: 4. Проверка полноты данных (см. 1.0.E1). 5. Заполнение смежными данными. 6. Проверка целостности нейронной сети (см. 1.1.E1). 7. Формирование файла для экспорта.
Альтернативные потоки	1.1 Завершение работы программы
Исключения	1.0.E1 Данные не являются полными: 1. Вывод сообщения о пропущенных данных. 2. Автоматическое заполнение пропущенных значений. 1.1.E1 Целостность нейронной сети повреждена: 1. Сообщение об ошибке с вариантами устранения. 2. Завершение работы программы (см. 1.1).
Приоритет	Высокий.
Точка расширения	UC-5. «Обработка задачи»

Таблица Е.5 – Описание прецедента «Обработка задачи»

Идентификатор и название	UC-5. «Обработка задачи»
Акторы	Нейронная сеть.
Описание	Нейронная сеть обрабатывает введенные данные.
Триггер	Система сформировала файл с данными.
Предварительные условия	PRE-1. Наличие файла с данными. PRE-2. Целостность файла с нейронной сетью.
Выходные условия	POST-1. Отправка данных на решение.
Основной поток	1.0 Нейронная сеть обрабатывает файл и отправляет на решение (см. 1.0.E1).
Альтернативные потоки	1.1 Завершение работы программы.
Исключения	1.0.E1 Ошибка возникшая во время работы: 1. Формирование файла с ошибкой. 2. Завершение работы программы (см. 1.1).
Приоритет	Высокий.
Точка расширения	-

Таблица Е.1 – Описание прецедента «Решение задачи»

Идентификатор и название	UC-6. «Решение задачи»
Акторы	Нейронная сеть.
Описание	Нейронная сеть находит решения для введенных данных на основе обучения.
Триггер	Нейронная сеть обработала данные.
Предварительные условия	PRE-1. Обработанные данные.
Выходные условия	POST-1. Результат работы нейронной сети.
Основной поток	1.0 Нейронная сеть находит результат на основе обучения (см. 1.0.E1): 1. Система загружает результаты обучения. 2. Система формирует результат.
Альтернативные потоки	1.1 Завершение работы программы.
Исключения	1.0.E1 Ошибка возникшая во время работы: 1. Формирование файла с ошибкой. 2. Завершение работы программы (см. 1.1).
Приоритет	Высокий.
Точка расширения	UC-3. «Получение результата»

Таблица Е.7 – Описание прецедента «Обучение»

Идентификатор и название	UC-7. «Обучение»
Акторы	Нейронная сеть.
Описание	Система обучает нейронную сеть.
Триггер	Начат процесс обучения нейронной сети.
Предварительные условия	PRE-1. Собраны обучающие данные. PRE-2. Построены связи нейронов.
Выходные условия	POST-1. Сформированный файл с обученной нейронной сетью.
Основной поток	1.0 Система обучается (см. 1.0.E1): 1. Система строит отношения связей нейронов. 2. Система повторяет процесс необходимое количество раз.
Альтернативные потоки	1.1 Завершение работы системы.
Исключения	1.0.E1 Ошибка возникшая во время работы: 1. Формирование файла с ошибкой. 2. Завершение работы системы (см. 1.1).
Приоритет	Высокий.
Точка расширения	Формирование артефакта в виде файла обученной нейронной сети

Таблица Е.1 – Описание прецедента «Сбор данных»

Идентификатор и название	UC-8. «Сбор данных»
Акторы	Нейронная сеть.
Описание	Сбор данных для обучения нейронной сети.
Триггер	Система запущена.
Предварительные условия	-
Выходные условия	POST-1. Набор данных для обучения.
Основной поток	1.0 Система собирает данные: 1. Система проходит по веб страницам собирая данные (см. 1.0.E1). 2. Система проверяет полноту данных. 3. Система удаляет пустые данные. 4. Система очищает данные. 5. Система нормализует данные. 6. Система формирует файлы для обучения.
Альтернативные потоки	1.1 Завершение работы системы.
Исключения	1.0.E1 Возникновение сбоя во время сбора данных: 1. Вывод ошибки о сбое. 2. Завершение работы системы (см. 1.1).
Приоритет	Высокий.
Точка расширения	UC-7. «Обучение»

Таблица Е.1 – Описание прецедента «Построение связей»

Идентификатор и название	UC-9. «Построение связей»
Акторы	Нейронная сеть.
Описание	Построение архитектуры нейронной сети.
Триггер	Система запущена.
Предварительные условия	-
Выходные условия	POST-1. Построенная структура нейронной сети.
Основной поток	1.0 Система формирует структуру на основании выбора разработчика.
Альтернативные потоки	-
Исключения	-
Приоритет	Средний.
Точка расширения	UC-7. «Обучение»

ПРИЛОЖЕНИЕ F

Диаграмма последовательности обучения

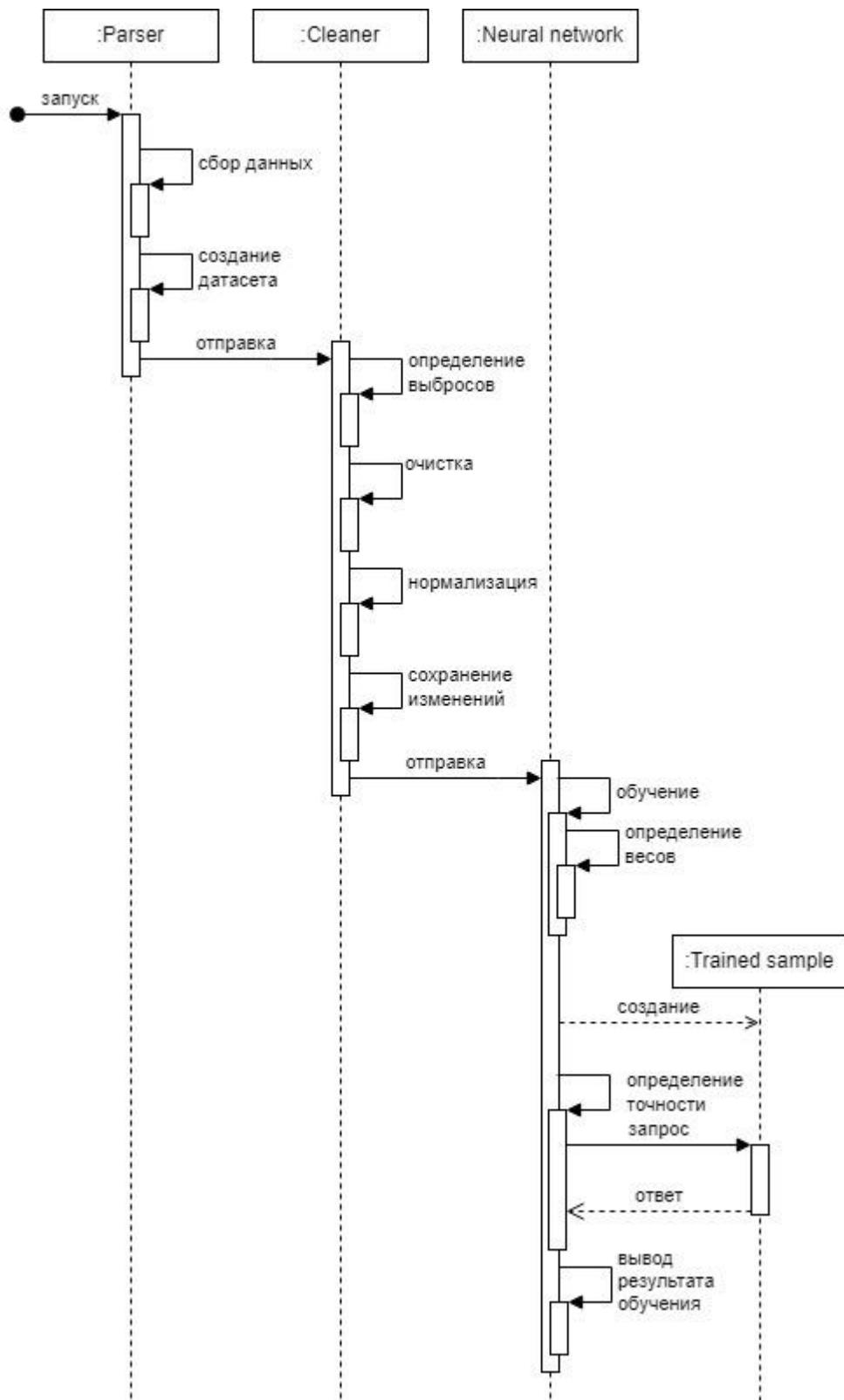


Рисунок F.1 – Диаграмма последовательности обучения нейронной сети

ПРИЛОЖЕНИЕ G

Блок-схемы алгоритма сбора данных

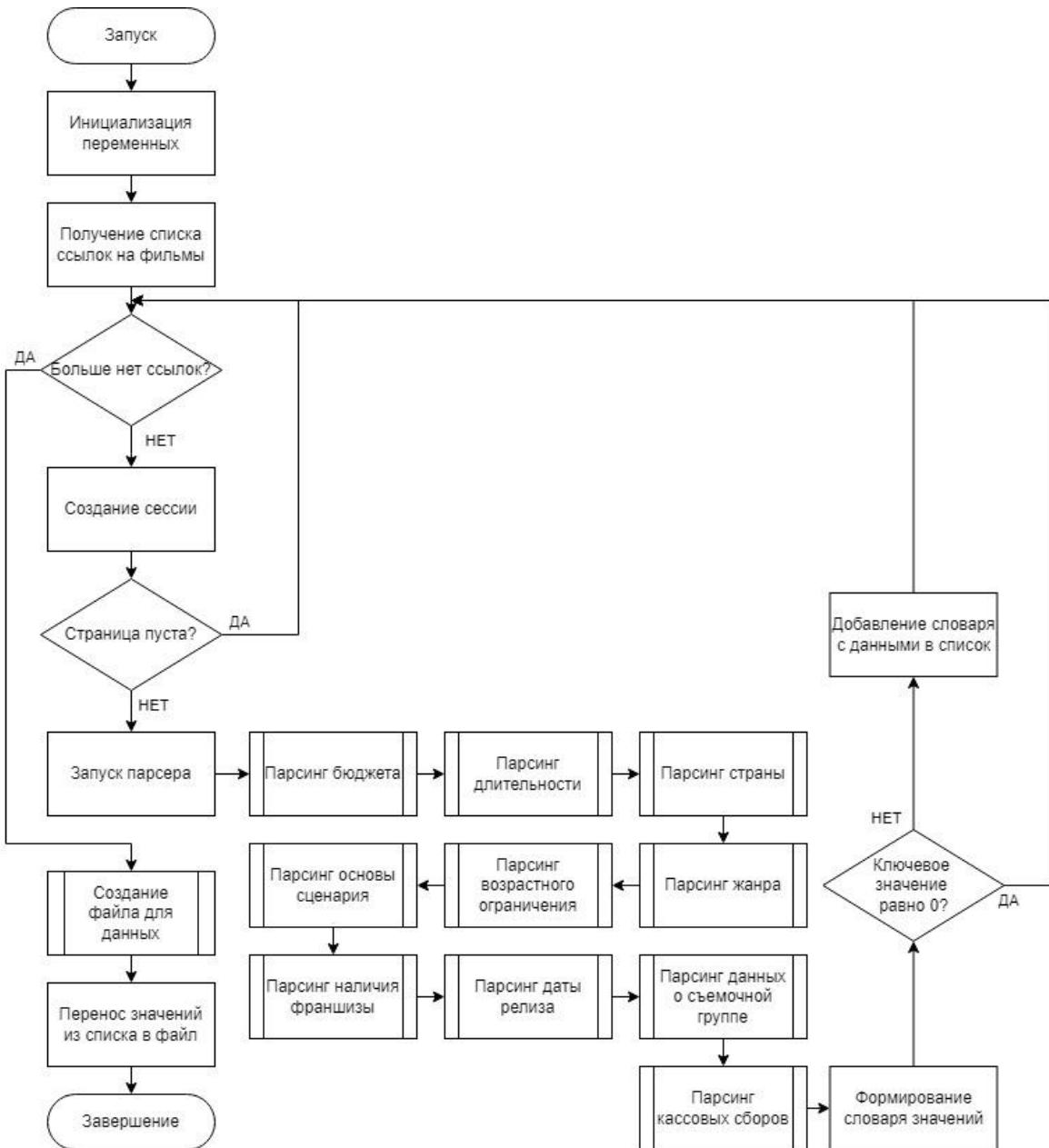


Рисунок G.1 – Основной алгоритм сбора обучающих данных



Рисунок G.2 – Под процесс сбора информации о бюджете

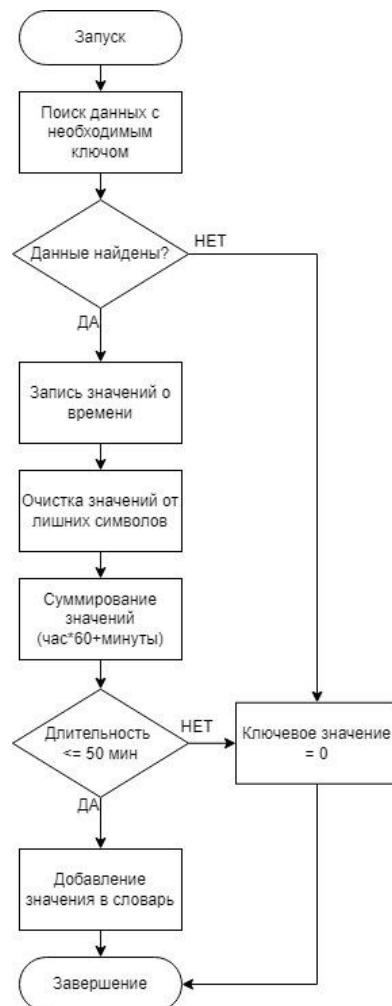


Рисунок G.3 – Под процесс сбора информации о длительности



Рисунок G.4 – Под процесс сбора информации о стране



Рисунок G.5 – Под процесс сбора информации о жанре

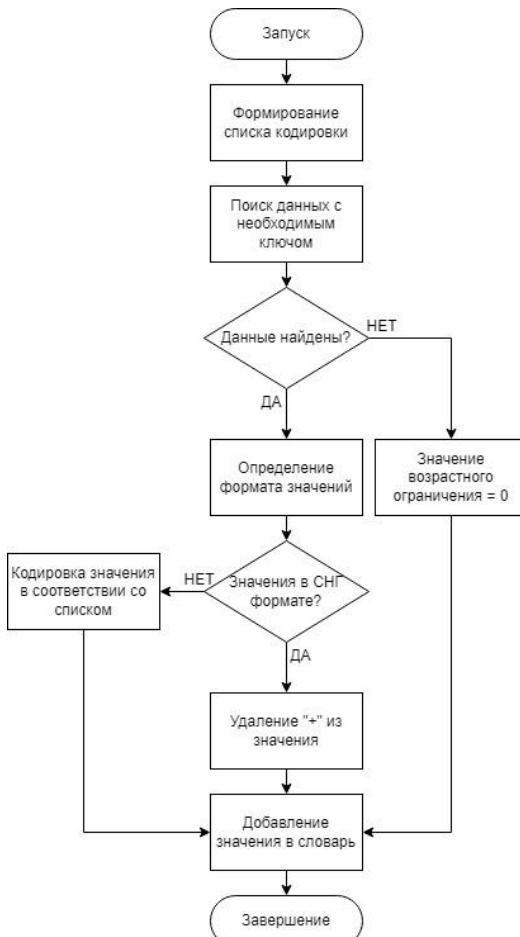


Рисунок G.6 – Под процесс сбора информации о возрастном ограничении

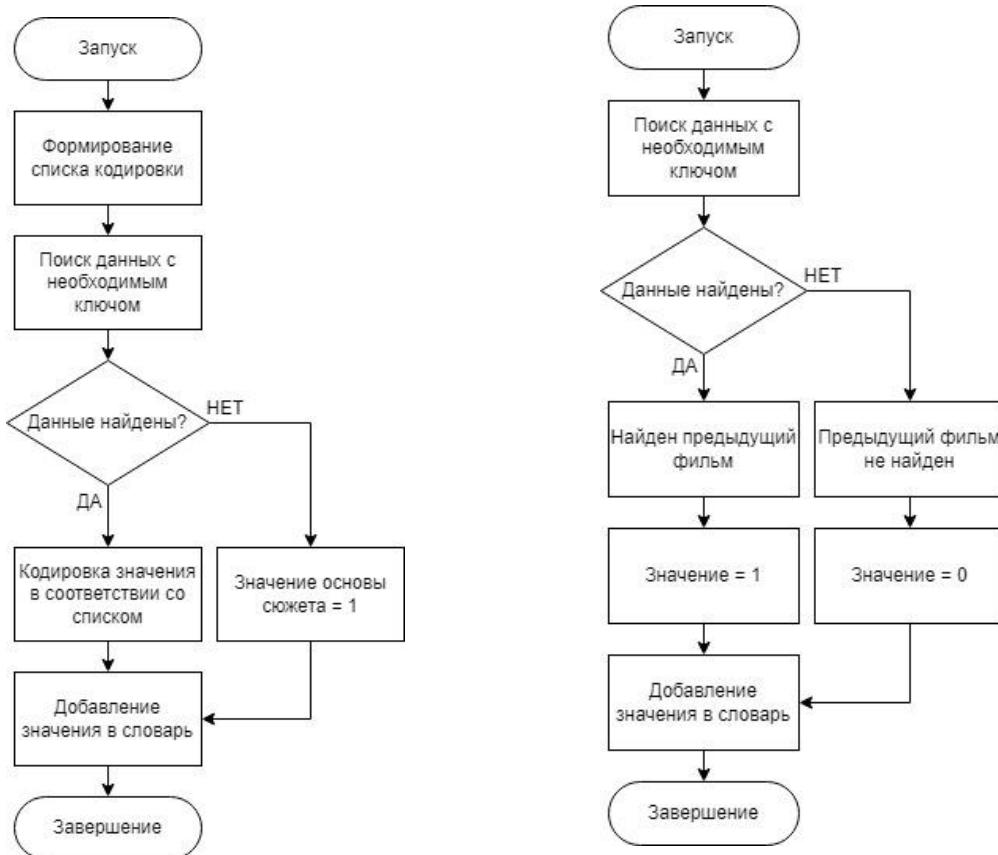


Рисунок G.7 – Под процесс сбора информации о основе сюжета

Рисунок G.8 – Под процесс сбора информации о наличии франшизы

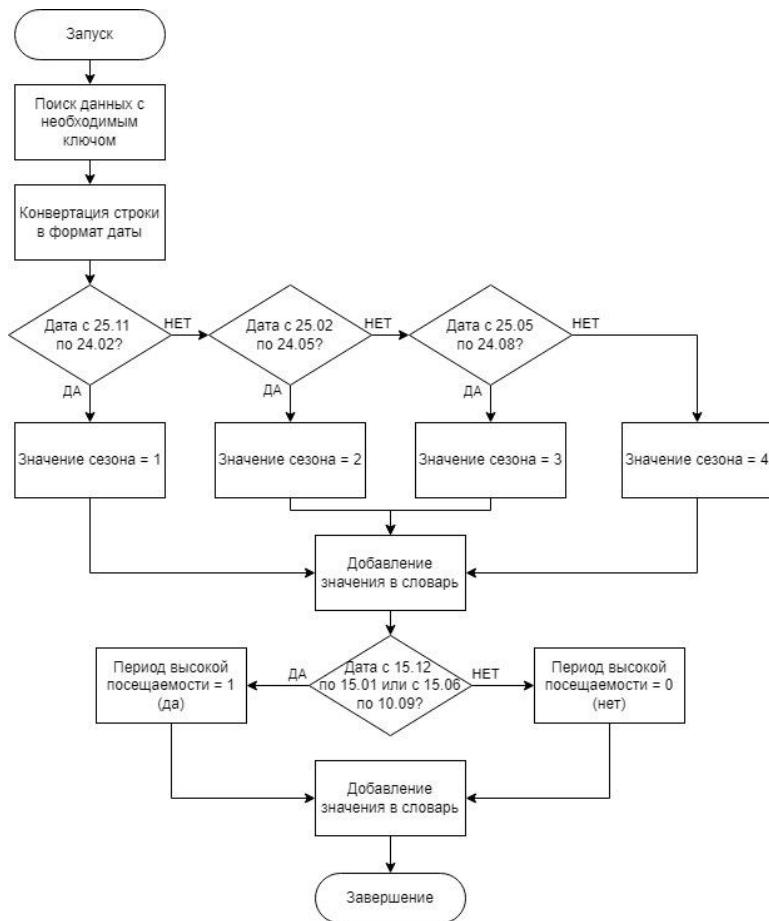


Рисунок G.9 – Под процесс сбора информации о сезоне выхода

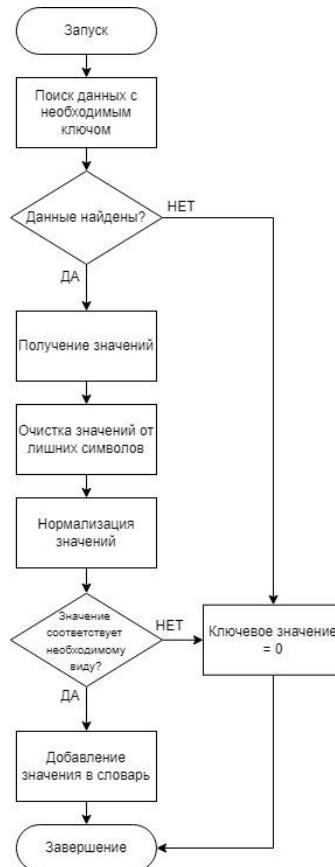


Рисунок G.10 – Под процесс сбора информации о кассовых сборах

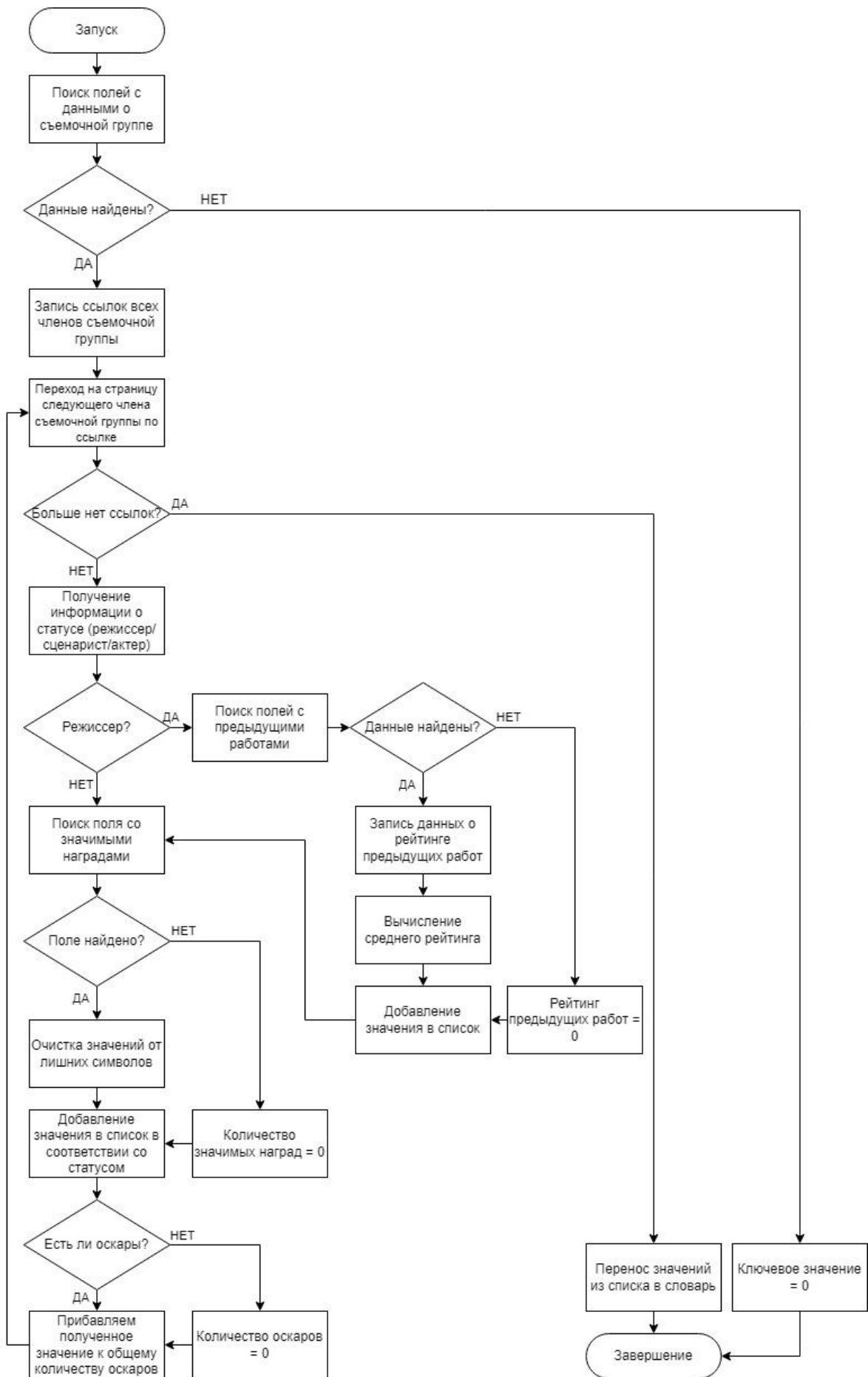


Рисунок G.11 – Под процесс сбора информации о съемочной группе

ПРИЛОЖЕНИЕ Н

Листинг модуля сбора и систематизации данных

Алгоритм первичного сбора данных

```
import requests
import time
from bs4 import BeautifulSoup

headers = {'accept': '*/*',
           'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36'}

top_1000 = 'https://www.imdb.com/search/title/?groups=top_1000'
bottom_1000 = 'https://www.imdb.com/search/title/?groups=bottom_1000'

def scrapper(url, rat):
    data = []          # Переменная для хранения id фильмов
    start = 0          # Счетчик обработанных фильмов
    iteration = 0      # Счетчик первой итерации

    while start < 1000:
        request = requests.get(url, headers=headers)           # Передаем ссылку в обработчик
        soup = BeautifulSoup(request.content, 'html.parser')   # Подключаемся к странице

        # Находим список фильмов на странице
        title_block = soup.find('div', class_='lister-list').find_all('div', class_='lister-item mode-advanced')
        for item in title_block: # Для каждого элемента в списке
            title = item.find('h3', class_='lister-item-header').find('a') # Находим элемент
            title_id = title.get('href').replace('/title/tt', '').replace('/?ref_=adv_li_tt',
') # Находим id
            data.append(title_id) # Записываем id фильма в список
            print(
                f'{rat} / '
                f'{item.find("span", class_="lister-item-index unbold text-primary").get_text().replace(".", ":"}) '
                f'id \033[32m{title_id}\033[0m')

        if start + 50 <= len(data): # Переходим на след страницу
            if iteration == 0: # Если первая итерация то добавляем необходимый текст к ссылке
                url = url + f'&start={str(start + 50 + 1)}&ref_=adv_nxt'
                iteration += 1
            else: # Иначе заменяем параметр страницы в запросе ссылки
                url = url.replace(f'&start={str(start + 1)}', f'&start={str(start + 50 + 1)}')
                start += 50
            else:
                # Иначе вывод ошибки
                # print('сломалось')
                break

    return data # Возвращаем список id фильмов

if __name__ == '__main__':
    start_time = time.time()
    ids = []
    raw_dataset = [scrapper(top_1000, 'top_1000'),
                  scrapper(bottom_1000, 'bottom_1000')]

    for k in raw_dataset[0]:
        ids.append(k) # записываем id top 1000 лучших фильмов
```

```

for k in raw_dataset[1]:
    ids.append(k) # записываем id из 1000 худших фильмов

with open(f'./ids.txt', 'w+', encoding='utf-8') as id_file:
    for row in ids:
        id_file.write(f'{row}\n') # записываем в файл

print(f'\n\033[7m--- END IN {round((time.time() - start_time), 3)} sec ---\033[0m')

```

Алгоритм сбора параметров обучающего множества

```

import datetime # Библиотека для работы с датами
import lxml # Библиотека для работы со структурой веб страницы
import os # Библиотека для работы с файлами компьютера
import re # Библиотека для поиска регулярных значений
import requests # Библиотека для формирования запросов к сайтам
import time # Библиотека для работы со временем
from bs4 import BeautifulSoup # Библиотека для парсинга данных
from multiprocessing.dummy import Pool # Библиотека для мультипоточности
from selenium import webdriver # Библиотека для парсинга, глубже и дальше чем beautifulsoup
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import StaleElementReferenceException
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options

wiki_info = [0, 0, 0] # Инициализация переменной для хранения данных с википедии
checked_counter = 0 # Счетчик количества обработанных ссылок
headers = {'accept': '*/*',
           'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36'}
```

Функция проверки на вещественное число, не путать с .isdigit (проверкой на целое число)

```

def is_number(v): # проверяем цифра или нет
    try:
        float(v) # проверка для всех вещественных чисел
        return True
    except ValueError:
        return False

```

Функция замены мусора на пустоту

```

def replace_scrap(data): # Вызывается нужно много символов заменять
    clean_data = (data # замена символов
                  .replace(' ', '')
                  .replace('U', '')
                  .replace('долл.', '')
                  .replace('долларов', '')
                  .replace('(в США)', '')
                  .replace('.', '')
                  .replace('(estimated)', '')
                  .replace('~', ''))
    return clean_data

```

Функция конвертации валют в доллары

```

def convert_currency(num):
    # Заменяем иконки символов на курс к доллару со знаком умножения
    temp_num = (num
                .replace('$', '1*')
                .replace('£', '1.26*')
                .replace('€', '1.07*')
                .replace('₩', '0.00079*')
                .replace('¥', '0.0078*')
                .replace('₹', '0.013*'))

```

```

.replace('£', '1.26*')
.replace('DEM', '0.5414*')
.replace('FRF', '1.07*')
.replace('R', '0.064*'))
temp_num = temp_num.split('*') # Разделяем строковое значение на 2 значения по знаку
умножения
if len(temp_num) > 1: # Если список содержит больше 1 значения (операция успешна)
    try: # Пробуем выполнить умножение
        # Умножаем значения переводя их в вещественные и округляем до целого
        currency = round(float(temp_num[0]) * float(temp_num[1]))
        return currency
    except Exception as ex:
        print(ex) # Выбрасываем ошибку если попался необработанный символ
        return 0
else: # Иначе возвращаем что получили
    return num

# Функция получения данных с википедии, если не было на imdb
def parse_wiki(child_id):
    global wiki_info # Ссылаемся на глобальную переменную
    wiki_info = [0, 0, 0] # Обнуляем глобальную переменную

def calculate(cal): # получаем цифру и меняем млн на 1000000 и тд
    reg = re.findall(r'\[\d*', cal) # Вводим в замену регулярное выражение
    temp = (replace_scrap(cal)) # Так как данные с википедии не возможно конвертировать
        .replace(''.join(reg), '') # То просто их удаляем
        .replace('£', '') # Тем более в 90% случаев показатели указываются в долларах
        .replace('€', '')
        .replace('₩', '')
        .replace('¥', '')
        .replace('₹', '')
        .replace('£', '')
        .replace('DEM', '')
        .replace('FRF', '')
        .replace('R', '')
    temp = (temp.replace('тыс', '*тыс')) # Добавляем знак умножения, чтобы понять где
разделять
        .replace('млн', '*млн')
        .replace('млрд', '*млрд'))
    temp = temp.split('*') # разделяем вход на число и (млн, млрд, тыс)
    temp[0] = temp[0].replace(',', '.').split('-')[0] # если данные в формате 100-150 то
выбираем нижний порог
    if len(temp) > 1: # если в списке больше 1 значения
        if len(temp[0]) > 4: # если длина цифры больше 4 символов (1 234 567)
            return ''.join(temp[0]).replace(' ', '').replace('\xa0', '')
        else: # иначе если (1,234)
            temp[1] = (temp[1] # заменяем второе значение на численное
                .replace('тыс', '1000')
                .replace('млн', '1000000')
                .replace('млрд', '1000000000')
                .replace(',', '.'))
            if is_number(temp[0]) is True and is_number(temp[1]): # если оба значения
цифры
                return round((float(temp[0]) * float(temp[1]))) # возвращаем
перемноженный вариант
            # Вхождение: $1,234 млн
            # Выход: 1234000
            # Вхождение: $100-150 млн
            # Выход: 100000000
            # Вхождение: $1 234 000
            # Выход: 1234000
            # ошибка: Вхождение: много денег и др
        else:
            return 0
    else:
        return 0

try:

```

```

# Вводим строку поиска id на сайте imdb

driver.get(f'https://ru.wikipedia.org/w/index.php?search=IMDb%09ID+{str(child_id)}&ns0=1')
time.sleep(2) # Ждем загрузки страницы с результатами поиска

search_result = driver.find_element(By.XPATH, '//div[@class="mw-search-result-heading"]//a')
if search_result is not None: # Проверка на пустоту
    search_result = search_result.get_attribute('href') # Находим ссылку на страницу

url = str(search_result) #
session = requests.Session()
request = session.get(url, headers=headers)
soup = BeautifulSoup(request.content, 'lxml') # Загружаем найденную страницу в более быстрый bs4

# region Бюджет
temp_budget = soup.find('span', attrs={'data-wikidata-property-id': 'P2130'}) # Ищем данные бюджета
if temp_budget is not None: # Проверка на пустоту
    temp_budget = temp_budget.get_text() # Получаем текст
    wiki_info[0] = calculate(temp_budget) # Переводим в числовой формат
else:
    wiki_info[0] = 0
# endregion
# region Касса
temp_box = soup.find('span', attrs={'data-wikidata-property-id': 'P2142'}) # Ищем кассовые сборы
if temp_box is not None: # Проверка на пустоту
    temp_box = temp_box.get_text() # Получаем текст
    wiki_info[1] = calculate(temp_box) # Переводим в числовой формат
else:
    wiki_info[1] = 0
# endregion
# region Франшиза
franchise1 = soup.find('span', attrs={'data-wikidata-property-id': 'P155'}) # узнаем есть ли предыдущие фильмы
if franchise1 is not None: # Если значение есть, то есть и франшиза
    wiki_info[2] = 1
else: # Если нет то и франшизы нет
    wiki_info[2] = 0
# endregion

except (NoSuchElementException, StaleElementReferenceException):
    wiki_info = [0, 0, 0] # Если вылетела ошибка возвращаем нули
    pass

# Функция получения бюджета (не актуально)
# Перенесено в основную функцию, иначе много вызовов и увеличивается время обработки
def parse_budget(request):
    budget = request.find('li', attrs={'data-testid': 'title-boxoffice-budget'})
    if budget is not None:
        budget = budget.find('span', class_='ipc-metadata-list-item__list-content-item')
        if budget is not None:
            budget = convert_currency(budget.get_text())
            budget = replace_scrap(budget).replace(',', '').replace('\xa0', '')
    else:
        budget = wiki_info[0]
    else:
        budget = wiki_info[0]

# Функция получения длительности
def parse_duration(hour, minute):
    hour = hour[0].replace('h', '') if hour else 0 # Часы длительности
    minute = minute[0].replace('m', '') if minute else 0 # Минуты длительности
    duration = int(hour) * 60 + int(minute) # Час * на 60 + минуты
    return duration

```

```

# Функция получения страны (не актуально)
def parse_country():
    smth = 0
    return smth

# Функция получения жанра
def parse_genre(request):
    # Список кодировки жанров
    genre_format = {'Action': 1, 'Adventure': 2, 'Drama': 3, 'Romance': 3, 'Comedy': 4,
                    'Crime': 5, 'Mystery': 6, 'Horror': 7, 'Western': 8, 'History': 9,
                    'Documentary': 9, 'Biography': 10, 'Animation': 11, 'Fantasy': 12, 'Sci-Fi': 12,
                    'Thriller': 13, 'Music': 14, 'Musical': 14, 'Film-Noir': 15, 'War': 16,
                    'Family': 17, 'Short': 18, 'Sport': 19, 'Reality-TV': 20,
                    'Game-Show': 21, 'Talk-Show': 22, 'News': 23, 'Adult': 24}

    genre = request.find('div', attrs={'data-testid': 'genres'}) # Поиск поля по атрибуту
    if genre is not None: # Проверка на пустоту
        genre = genre.find('span', class_='ipc-chip__text')
        if genre is not None:
            genre = genre_format[genre.get_text()] # Получение текста

    return genre

# Функция получения возрастного ограничения
def parse_age_limit(mpaa):
    # Список кодировки возрастного ограничения
    agelimit_format = {'G': 1, 'PG': 2, 'PG-13': 3, 'NC-17': 4, 'R': 5, 'X': 6,
                        '0': 1, '6': 2, '12': 3, '14': 3, '16': 4, '18': 5}

    # Поиск регулярного выражения
    if re.search(r'\b([0-9]{1,2}\+)|(PG-13|NC-17|PG|R|G)', str(mpaa)) is not None:
        if str(mpaa).replace('+', '') in agelimit_format:
            age_limit = agelimit_format[str(mpaa).replace('+', '')]
        else:
            age_limit = 1
    else:
        age_limit = 1

    return age_limit

# Функция конвертации даты в сезон выхода и нагрузженности на кинозалы
def parse_date(child_id):
    url = f'https://www.imdb.com/title/tt{str(child_id)}/releaseinfo'
    session = requests.Session()
    request = session.get(url, headers=headers)
    soup = BeautifulSoup(request.content, 'lxml') # Создаем сессию

    parsed_date = [0, 0] # Инициализация списка возвращаемых значений
    date = None

    # Список кодировки месяцев
    month_format = {'December': 12, 'January': 1, 'February': 2,
                    'March': 3, 'April': 4, 'May': 5,
                    'June': 6, 'July': 7, 'August': 8,
                    'September': 9, 'October': 10, 'November': 11}

    data = []
    dates_list = soup.find('table', class_='release-dates-table-test-only')
    dates_list = dates_list.find_all('tr', class_='ipl-zebra-list__item release-date-item')
    for item in dates_list: # Проверка всех дат выхода
        data.append(item.find('td', class_='release-date-item__date').get_text()) # Получения списка дат

```

```

qty_most_common = 0
for j in set(data):
    qty = data.count(j)
    if qty > qty_most_common:
        qty_most_common = qty # Поиск самой часто появляющейся даты
        date = j

date = date.split(' ') # Делим дату на день месяц и год

day = date[0]
if len(str(date[1])) > 3 and not date[1].isdigit():
    month = month_format[str(date[1])]
else:
    parsed_date = [0, 0]
    return parsed_date

temp_season = datetime.datetime(2020, int(month), int(day))
if datetime.datetime(2019, 11, 20) < temp_season < datetime.datetime(2020, 2, 20):
    parsed_date[0] = 4 # с 20 ноября до 20 февраля зимний сезон или 4
elif datetime.datetime(2020, 2, 20) < temp_season < datetime.datetime(2020, 5, 20):
    parsed_date[0] = 1 # аналогично для всех ниже
elif datetime.datetime(2020, 5, 20) < temp_season < datetime.datetime(2020, 8, 20):
    parsed_date[0] = 2
elif datetime.datetime(2020, 8, 20) < temp_season < datetime.datetime(2020, 11, 20):
    parsed_date[0] = 3
elif datetime.datetime(2020, 11, 20) < temp_season < datetime.datetime(2021, 2, 20):
    parsed_date[0] = 4
else:
    parsed_date[0] = 4

if datetime.datetime(2019, 12, 15) < temp_season < datetime.datetime(2020, 1, 15):
    parsed_date[1] = 1 # если вышел в зимние каникулы
elif datetime.datetime(2020, 12, 15) < temp_season < datetime.datetime(2021, 1, 15):
    parsed_date[1] = 1
elif datetime.datetime(2020, 6, 15) < temp_season < datetime.datetime(2020, 9, 10):
    parsed_date[1] = 1 # если вышел в летние каникулы
else:
    parsed_date[1] = 0

return parsed_date

```

```

# Функция получения данных о съемочной группе
def parse_crew(child_id, director):
    def parse_director(work_id):
        sub_url = f'https://www.imdb.com/title/tt{str(work_id)}/?ref_=nm_flmg_dr_1'
        sub_session = requests.Session()
        sub_request = sub_session.get(sub_url, headers=headers)
        sub_soup = BeautifulSoup(sub_request.content, 'lxml') # Создаем сессию

        # Находим рейтинг фильма
        rating = sub_soup.find('div', attrs={'data-testid': 'hero-rating-bar__aggregate-rating-score'})
        if rating is not None: # Проверка на пустоту
            rating = rating.find('span', class_='sc-7ab21ed2-1').get_text() # Получаем текст
            if rating == 0:
                rating = -1 # Возвращаем -1 если данных нет или рейтинг равен 0
            else:
                rating = -1

        return rating

    score, temp = 0, 0
    output = [0, 0, 0, 0] # Инициализация списка с возвращаемыми значениями

    for item in child_id:
        url = f'https://www.imdb.com/name/nm{str(item)}/'
        session = requests.Session()
        request = session.get(url, headers=headers)

```

```

soup = BeautifulSoup(request.content, 'lxml') # Создаем сессию

works_urls, works_num, works_sum = [], 0, 0
if director == 1: # Если получали данные о директорах
    # Поиск атрибутов по регулярному выражению
    works = soup.find_all('div', attrs={'id': re.compile("director-tt")})
    if works is not None: # Проверка на пустоту
        for work in works: # Для каждого фильма в списке
            # Находим фильм и получаем его id
            work = (work.find('b').find('a').get('href')
                     .replace('/title/tt', '')).replace('/?ref_=nm_flmg_dr_1', ''))
            works_urls.append(work)
    for url in works_urls: # для каждого id в списке
        info = parse_director(url)
        if info != -1: # Если вернулся -1 то не учитываем это значение
            works_num += float(info) # Суммируем рейтинги
            works_sum += 1 # Суммируем количество фильмов
    output[3] = round(works_num / works_sum) # Делим сумму рейтингов фильмов на
    количества фильмов

    temp = soup.find('a', attrs={'id': 'meterRank'})
    if temp is not None: # Проверка на пустоту
        temp = temp.get_text() # Если не пусто то страница существует
    else:
        output = [0, 0, 0, 0] # Иначе возвращаем нули
        return output

    awards = soup.find('div', class_='article highlighted') # Поиск поля с наградами
    if awards is not None: # Проверяем на пустоту
        awards = awards.find_all('span', class_='awards-blurb') # Поиск наград в поле
        for span_aw in awards: # Для каждой награды в списке
            if awards is not None: # Проверка на пустоту
                awards = span_aw.get_text().replace('.', '').replace('\n', '') # заменяем
                ненужные символы
                if awards.find('Oscar') != -1 or awards[0].find('Oscars') != -1: # если
                    есть информация о оскарах
                    if awards.find('Nominated') != -1: # если номинирован
                        temp_aw = (awards.replace(' ', '') # Удаляем ненужные слова
                                    .replace('Nominated', '')
                                    .replace('for', '')
                                    .replace('Oscar', '')
                                    .replace('s', ''))
                        temp_aw = int(temp_aw) if str(temp_aw).isdigit() else 1 #
                    заменяя числом
                    if temp_aw > 2:
                        output[0] += 1 # если номинаций больше 2 считаем за получение
                        1 оскара
                    else:
                        output[0] += 0
                    else:
                        temp_aw = (awards.replace(' ', '')
                                    .replace('Won', '')
                                    .replace('Oscar', '')
                                    .replace('s', ''))
                        # считаем количество полученных оскаров
                        output[0] += int(temp_aw) if str(temp_aw).isdigit() else 1
                else:
                    output[0] += 0 # иначе записываем на возврат ноль
            if awards.find('wins') != -1:
                output[1] = 1 # 1-если были полученные награды
            else:
                output[1] = 0 # 0-если полученных наград не было
            if awards.find('&') != -1:
                awards = (awards.split('&')[0]
                          .replace(' ', '')
                          .replace('Another', '')
                          .replace('wins', ''))

```

```

        else:
            awards = (awards.replace(' ', '')
                      .replace('Another', '')
                      .replace('wins', ''))

    if temp == 'SEE RANK': # считаем рейтинг
        score += 10 / 2 # если не входит в топ 5000
    elif temp in ('Top 5000', 'Top 2500', 'Top 1000'):
        score += 20 / 2 # если входит в топ 5000 но не в топ 500
    elif temp == 'Top 500':
        score += 30 / 2 # если входит в топ 500
    elif 50 < int(temp) <= 100:
        score += 40 / 2 # если входит в топ 100
    elif 0 < int(temp) <= 50:
        score += 50 / 2 # если входит в топ 50
    else:
        output = [0, 0, 0, 0]
        return output

    score += int(awards) if str(awards).isdigit() else 1 # прибавляем количество наград
    # score += ((100 - int(temp)) / 2) + 30 / 2

num = len(child_id) if len(child_id) != 0 else 1
output[2] = round(score / num) # делим количество наград на количество человек из списка
return output

# Основная функция парсера
def parser(title_id):
    global checked_counter, wiki_info # Определяем глобальные переменные
    # Инициализация переменных
    directors, writers, stars = [], [], []
    director_rating, directors_awards, writers_awards, stars_awards = '', '', '', ''
    year, mpaa, duration, genre, season, popularity, budget, boxoffice, franchise = '', '', '',
    hour, minute, age_limit, oscars, holiday = '', '', '', '', ''
    url = f'https://www.imdb.com/title/tt{str(title_id)}/' # Передаем ссылку
    session = requests.Session()
    request = session.get(url, headers=headers) # Создаем сессию
    soup = BeautifulSoup(request.content, 'lxml') # Читаем страницу

    print(f'{checked_counter}/2000 Progress ({title_id}): \033[31m[processing]\033[0m {url}')
    # print(f'{checked_counter} Progress ({title_id}): [##] 1/8 [wiki] {url}')
    parse_wiki(title_id) # отправляем поиск фильма в Вики
    # Поиск на Вики обязательен так как там информация о франшизе

    success = soup.find('li', class_='ipc-inline-list__item') # Проверяем наличие страницы
    title = soup.find('h1', attrs={'data-testid': 'hero-title-block__title'})
    if title is not None:
        title = title.get_text() # название фильма

    if success is not None: # страница не пустая или не принадлежит не фильму
        # region С помощью regex выбираем значения длительности и возрастного ограничения
        items_list = soup.find('ul', class_='ipc-inline-list').find_all('li', class_='ipc-
        inline-list__item')
        for item in items_list:
            hour = re.findall(r'\b\w{1,2}[h]\b', item.get_text()) if not hour else hour # поиск часов
            minute = re.findall(r'\b\w{1,2}[m]\b', item.get_text()) if not minute else minute
        # поиск минут
            if item.find('a', class_='ipc-link') is not None:
                year = re.findall(r'\d{4}', item.get_text()) if not year else year # поиск года выхода
                mpaa = re.findall(r'\b([0-9]{1,2}\+)|(PG-13|NC-17|PG|R|G)|(TV-
                (Y7|Y|PG|G|14|MA))\b',
                                  item.get_text()) if not mpaa else mpaa # поиск возрастного
                    рейтинга
        # endregion

```

```

        # print(f'\r{checked_counter} Progress ({title_id}): [#####] 2/8 [duration]
{url}')
        # region Длительность фильма
        duration = parse_duration(hour, minute)
        # endregion

        # print(f'\r{checked_counter} Progress ({title_id}): [#####] 3/8 [age
limit] {url}')
        # region Если получили возрастной рейтинг в формате МПАА конвертируем в рус вариант
        while type(mpaa) not in (str, int):
            mpaa = mpaa[0] if mpaa else 0

        age_limit = parse_age_limit(mpaa)
        # endregion

        # print(f'\r{checked_counter} Progress ({title_id}): [#####] 4/8 [genre]
{url}')
        # region Жанр фильма
        genre = parse_genre(soup)
        # endregion

        # print(f'\r{checked_counter} Progress ({title_id}): [#####] 5/8 [release
date] {url}')
        # region Дата выхода фильма
        release_date = parse_date(title_id) # получаем список
        season = release_date[0] # 0-сезон выхода
        holiday = release_date[1] # выход в каникулы да/нет
        # endregion

        # print(f'\r{checked_counter} Progress ({title_id}): [#####] 6/8 [crew]
{url}')
        # region Данные о съемочной группе
        temp = 1
        # Получаем список съемочной группы
        credits_list = soup.find('div', attrs={'data-testid': 'title-pc-expanded-section'})
        if credits_list is not None:
            credits_list = credits_list.find_all('li', attrs={'data-testid': 'title-pc-
principal-credit'})
        else:
            credits_list = soup.find('div', attrs={'data-testid': 'title-pc-wide-screen'})
            if credits_list is not None:
                credits_list = credits_list.find_all('li', attrs={'data-testid': 'title-pc-
principal-credit'})
            else:
                title_id = 0

        for credits_item in credits_list:
            list_items = credits_item.find_all('li', class_='ipc-inline-list__item')
            for li in list_items:
                if li.find('a', class_='ipc-metadata-list-item__list-content-item') is not
None:
                    # добавляем в список id съемочной группы со страницы фильма
                    if temp == 1:
                        directors.append(li.find('a', class_='ipc-metadata-list-item__list-
content-item')
                                         .get('href')
                                         .replace('/name/nm', '')
                                         .replace('/?ref_=tt_ov_dr', ''))

                    elif temp == 3:
                        writers.append(li.find('a', class_='ipc-metadata-list-item__list-
content-item')
                                      .get('href')
                                      .replace('/name/nm', '')
                                      .replace('/?ref_=tt_ov_wr', ''))

                    else:
                        stars.append(li.find('a', class_='ipc-metadata-list-item__list-
content-item')
                                     .get('href'))

```

```

        .replace('/name/nm', '')
        .replace('/?ref_=tt_ov_st', ''))

    temp += 1

directors = parse_crew(directors, 1) # получаем наличие наград режиссеров
writers = parse_crew(writers, 0) # получаем наличие наград сценаристов
stars = parse_crew(stars, 0) # получаем наличие наград 3х звезд в главных ролях

director_rating = directors[3]
oscars = directors[0] + writers[0] + stars[0] # количество оскаров у съемочной группы

directors_awards = directors[1] # наличие наград у режиссеров
writers_awards = writers[1] # наличие наград у сценаристов
stars_awards = stars[1] # наличие наград у звезд

directors = directors[2] # рейтинг режиссеров рейтинг в топ 5000 на imdb + количество
полученных наград
writers = writers[2] # рейтинг сценаристов рейтинг в топ 5000 на imdb + количество
полученных наград
stars = stars[2] # рейтинг звезд рейтинг в топ 5000 на imdb + количество полученных
наград
# endregion

# region Популярность на imdb
popularity = soup.find('div', attrs={'data-testid': 'hero-rating-
bar__popularity__score'})
if popularity is not None:
    popularity = popularity.get_text().replace(',', '')
else:
    popularity = 5000
# endregion

# print(f'\r{checked_counter} Progress ({title_id}): [#####] 7/8 [budget]
{url}')
# region Значения бюджета и кассовых сборов
boxoffice_section = soup.find('div', attrs={'data-testid': 'title-boxoffice-section'})
if boxoffice_section is not None: # проверка на пустоту
    budget = boxoffice_section.find('li', attrs={'data-testid': 'title-boxoffice-
budget'})
    if budget is not None:
        budget = budget.find('span', class_='ipc-metadata-list-item__list-content-
item')
        if budget is not None:
            budget = replace_scrap(budget.get_text()).replace(',', '').replace('\xa0',
)
            budget = convert_currency(budget)
        else:
            budget = wiki_info[0]
    else:
        budget = wiki_info[0]
# Все то же самое что и сверху но для кассовых сборов
boxoffice = boxoffice_section.find('li', attrs={'data-testid': 'title-boxoffice-
cumulativeworldwidegross'})
if boxoffice is not None:
    boxoffice = boxoffice.find('span', class_='ipc-metadata-list-item__list-
content-item')
    if boxoffice is not None:
        boxoffice = replace_scrap(boxoffice.get_text()).replace(',', '',
).replace('\xa0', '')
        boxoffice = convert_currency(boxoffice)
    else:
        boxoffice = wiki_info[1]
    else:
        boxoffice = wiki_info[1]
# endregion
else:
    title_id = 0 # если страницы нет или не фильм сохраняем id как пропуск значения
franchise = wiki_info[2]

```

```

# region Переводим значения в int
age_limit = int(age_limit) if age_limit else 0
duration = int(duration) if duration else 0
season = int(season) if season else 0
director_rating = int(director_rating) if str(director_rating).isdigit() else 0
writers = int(writers) if writers else 0
stars = int(stars) if stars else 0
genre = int(genre) if genre else 0
franchise = int(franchise) if franchise else 0
budget = int(budget) if str(budget).isdigit() else 0 # сохраняем число если строка может
быть числом иначе 0
boxoffice = int(boxoffice) if str(boxoffice).isdigit() else 0
# endregion

if int(duration) <= 50:
    title_id = 0 # если длительность меньше 40 (короткометражка) ставим id как пропуск

if budget in (0, None, '') or boxoffice in (0, None, ''):
    title_id = 0 # если бюджет = 0 ставим id как пропуск
else:
    if budget < 100000 or boxoffice < 100000:
        title_id = 0 # если бюджет меньше 100 000 ставим id как пропуск
    else:
        budget = int((str(budget)[0] + str(budget)[1]) + ("0" * (len(str(budget)) - 2)))
        budget = int(int(budget) / 100000)
        boxoffice = int((str(boxoffice)[0] + str(boxoffice)[1]) + ("0" *
(len(str(boxoffice)) - 2)))
        boxoffice = int(int(boxoffice) / 100000)

# if (budget * 4) > boxoffice < (budget / 4):
#     title_id = 0

print(f'{checked_counter}/2000 Progress ({title_id}): \033[32m[done]\033[0m {url}')
data = {'id': title_id,
        'name': title,
        'url': url,
        'year': year,
        'budget': budget,
        'duration': duration,
        'genre': genre,
        'age-limit': age_limit,
        'franchise': franchise,
        'release-season': season,
        'holiday': holiday,
        'director-rating': director_rating,
        'directors-awards': directors_awards,
        'writers-awards': writers_awards,
        'stars-awards': stars_awards,
        'oscars': oscars,
        'writers': writers,
        'stars': stars,
        'box-office': boxoffice}

checked_counter += 1
return data

# Функция для проведения теста
def test():
    # for item_id in ('0468569'): #ids: '10366460',
    test_set = parser('0038650')
    print(f'{test_set["budget"]};{test_set["duration"]};{test_set["genre"]};{test_set["age-
limit"]};'
          f'{test_set["franchise"]};{test_set["release-
season"]};{test_set["holiday"]};{test_set["director-rating"]};'
          f'{test_set["directors-awards"]};{test_set["writers-awards"]};{test_set["stars-
awards"]};'
          f'{test_set["oscars"]};{test_set["box-office"]}\n')

```

```

if __name__ == '__main__':
    start_time = time.time()
    pool = Pool() # создаем пул потоков для мультипоточности
    chrome_options = Options()
    chrome_options.add_argument("--disable-blink-features=AutomationControlled")
    # chrome_options.add_argument(r"user-data-dir=C:\\Users\\\" + WINUSER +
    "\\AppData\\Local\\Google\\Chrome\\User Data")
    chrome_options.add_argument("--headless")
    driver = webdriver.Chrome(options=chrome_options) # задаем значения для selenium

    # очистка предыдущего парсинга если надо
    open(f'./raw_dataset.txt', 'w').close()

if not os.path.exists('./ids.txt'):
    raise SystemExit

# test()

counter = 0 # счетчик с какой строки начинать парсить
while counter <= 2000:
    output_dataset, temp_dataset, temp_ids = [], [], []
    with open(f'./ids.txt', 'r', encoding='utf-8') as temp_file:
        for index, row_idx in enumerate(temp_file, 1):
            if counter <= index < counter + 100:
                temp_ids.append(row_idx.replace('\n', ''))

    try:
        # запускаем функцию parser в несколько потоков с входными данными из файла с id
        temp_dataset = pool.map(parser, temp_ids)
    except (TimeoutError, requests.exceptions.ConnectionError) as e:
        print(e) # пишем ошибку
        pass

    for temp_row in temp_dataset:
        if temp_row["id"] in (0, '0', '', None): # если id фильма 0 то пропускаем
            pass
        else: # иначе записываем в dataset
            output_dataset.append(temp_row)

    with open(f'./raw_dataset.txt', 'a', encoding='utf-8') as dataset:
        for subrow in output_dataset:
            dataset.write(f'{subrow}\n')

    counter += 100

    temp_file.close()
    dataset.close()

print(f'\n\033[7m--- END IN {round((time.time() - start_time), 3)} sec ---\033[0m')

```

Алгоритм формирования обучающего множества

```

import numpy
import os
import pandas
from openpyxl import Workbook
import matplotlib.pyplot as plt
import seaborn as sns
# plt.style.use('ggplot')

open('./dataset.xlsx', 'w').close()
open('./sample.xlsx', 'w').close()

```

```

indicator = 1
# files = ['all',
#           '0-10',
#           '10-40',
#           '40-inf']

def del_copy(obj): # удаляем копии из списка
    n = []
    for i in obj:
        if i not in n:
            n.append(i)
    return n

def convert_dataset(data):
    converted = []
    for row in data:
        converted_row = []

    return converted

# Удаляем выбросы
def del_ejects(filename):
    global indicator
    df = pandas.read_csv(f'./{filename}.txt', sep=';')

    plt.rcParams["figure.figsize"] = [7.50, 3.50]
    plt.rcParams["figure.autolayout"] = True

    if indicator in (1, 10):
        # plt.clf()
        print(filename, indicator)
        data = pandas.DataFrame({'Бюджет': df['budget'], 'Кассовые сборы': df['boxoffice']})
        ax = data[['Бюджет', 'Кассовые сборы']].plot(kind='box', title='boxplot')
        # plt.show()

    for x in ['budget', 'boxoffice']:
        q75, q25 = numpy.percentile(df.loc[:, x], [75, 25]) # Находим квантили
        intr_qr = q75 - q25

        maxq = q75 + (1.5 * intr_qr)
        minq = q25 - (1.5 * intr_qr)

        df.loc[df[x] < minq, x] = numpy.nan
        df.loc[df[x] > maxq, x] = numpy.nan

    #print(df.isnull().sum())
    df.isnull().sum()
    df = df.dropna(axis=0)

    df.budget = df.budget.astype(numpy.int32)
    df.boxoffice = df.boxoffice.astype(numpy.int32)
    df.to_csv(f'./{filename}.txt', index=False, sep=';')
    indicator += 1

def show_heatmap(filename):
    subrow_sum = 0
    with open(f'./{filename}.txt', 'r', encoding='utf-8') as lfile:
        for lrow in lfile:
            subrow_sum += 1

    plt.clf()
    df = pandas.read_csv(f'./{filename}.txt', sep=';')
    cols = df.columns[:round(subrow_sum/5)]
    sns.heatmap(df[cols].isnull())

```

```

plt.show()

plt.clf()
sns.heatmap(df[cols])
plt.show()

# plt.clf()
# df_m = df.copy()
# df_m['budget'] = [i.budget for i in df.index]
# df_m['boxoffice'] = [i.boxoffice for i in df.index]
# df_m = df_m.groupby(['budget', 'boxoffice']).mean()
# fig, ax = plt.subplots(figsize=(11, 9))
# sns.heatmap(df_m)
# plt.show()

plt.clf()
gr = df.groupby(['budget', 'boxoffice'])
gr = gr.nunique()
gr.head()
# dt_tweet_cnt = gr.loc[:, :, 'realDonaldTrump'].reset_index().pivot(index='hour_utc',
columns='minute_utc',
#                                         values='id')
# dt_tweet_cnt.fillna(0, inplace=True)
# dt_tweet_cnt = dt_tweet_cnt.reindex(range(0, 24), axis=0, fill_value=0)
# dt_tweet_cnt = dt_tweet_cnt.reindex(range(0, 60), axis=1, fill_value=0).astype(int)
# sns.heatmap(df[cols])
plt.show()

def save(data):
    header = f'X1;X2;X3;X4;X5;X6;X7;X8;X9;X10;X11;X12;D1\n'
    sample_frow = (f'Бюджет;'
                   f'Длительность;'
                   f'Жанр;'
                   f'Возрастное ограничение;'
                   f'Наличие франшизы;'
                   f'Сезон выхода;'
                   f'Период высокой посещаемости;'
                   f'Рейтинг режиссера;'
                   f'Режиссер престижные награды;'
                   f'Сценаристы престижные награды;'
                   f'Актеры престижные награды;'
                   f'Оскары;'
                   f'Кассовые сборы\n')
    with open(f'./input_dataset.txt', 'w', encoding='utf-8') as file_dataset:
        file_dataset.write(f'budget;duration;genre;mpaa;franchise;season;holiday;' + '\n' + header + sample_frow)

    f'dir_rating;dir_rewards;wr_rewards;act_rewards;oscars;boxoffice\n')
    with open(f'./dataset.txt', 'w', encoding='utf-8') as file_sdataset:
        file_sdataset.write(header)
    with open(f'./dataset_sample.txt', 'w', encoding='utf-8') as file_sdataset:
        file_sdataset.write(header)
        file_sdataset.write(sample_frow)
    with open(f'./temp_dataset.txt', 'w', encoding='utf-8') as temp_dataset:
        temp_dataset.write(header)
    with open(f'./temp_dataset_vt.txt', 'w', encoding='utf-8') as temp_tvdataset:
        temp_tvdataset.write(header)
    with open(f'./temp_dataset_valid.txt', 'w', encoding='utf-8') as temp_vdataset:
        temp_vdataset.write(header)
    with open(f'./temp_dataset_test.txt', 'w', encoding='utf-8') as temp_tdataset:
        temp_tdataset.write(header)

    with open(f'./input_dataset.txt', 'a', encoding='utf-8') as file:
        for row in data:
            file.write(f'{row["budget"]};{row["duration"]};{row["genre"]};{row["age-limit"]};{row["franchise"]};' +
                      f'{row["release-season"]};{row["holiday"]};{row["director-rating"]};' +
                      f'{row["directors-awards"]};{row["writers-awards"]};{row["stars-awards"]};'

```

```

                f'{row["oscars"]};{row["box-office"]}\n')
row_sum = 0
input_dataset = []
with open(f'./input_dataset.txt', 'r', encoding='utf-8') as lfile:
    for lrow in lfile:
        input_dataset.append(lrow)
        row_sum += 1

with open(f'./dataset.txt', 'a', encoding='utf-8') as file:
    for row in input_dataset:
        file.write(row)

with open(f'./dataset_sample.txt', 'a', encoding='utf-8') as file:
    for row in input_dataset:
        file.write(row)

row_counter = 0
dt_val, dt_test, dt_temp, dt_atemp = [], [], [], []
with open(f'./input_dataset.txt', 'r', encoding='utf-8') as mfile:
    for mrow in mfile:
        if row_counter != 0:
            if row_counter % (round(row_sum / ((row_sum * 15) / 100))) == 0:
                dt_val.append(mrow)
            else:
                dt_temp.append(mrow)
        row_counter += 1
lfile.close()

row_counter = 0
row_sum = round((row_sum * 75) / 100)
for nrow in dt_temp:
    if row_counter != 0:
        if row_counter % (round(row_sum / ((row_sum * 15) / 100)) + 2) == 0:
            dt_test.append(nrow)
        else:
            dt_atemp.append(nrow)
    row_counter += 1

with open(f'./temp_dataset.txt', 'a', encoding='utf-8') as ofile:
    for orow in dt_atemp:
        ofile.write(orow)
with open(f'./temp_dataset_vt.txt', 'a', encoding='utf-8') as pfile:
    for prow in dt_val:
        pfile.write(prow)
    for prow in dt_test:
        pfile.write(prow)
with open(f'./temp_dataset_valid.txt', 'a', encoding='utf-8') as qfile:
    for qrow in dt_val:
        qfile.write(qrow)
with open(f'./temp_dataset_test.txt', 'a', encoding='utf-8') as rfile:
    for rrow in dt_test:
        rfile.write(rrow)
ofile.close()
pfile.close()

show_heatmap(f'input_dataset')

# region Делим созданные файлы на группы и записываем в excel
group_dataset = pandas.read_csv(f'./dataset.txt', sep=';')
group_sample = pandas.read_csv(f'./dataset_sample.txt', sep=';')
group_data = pandas.read_csv(f'./temp_dataset.txt', sep=';')
group_vt = pandas.read_csv(f'./temp_dataset_vt.txt', sep=';')
group_valid = pandas.read_csv(f'./temp_dataset_valid.txt', sep=';')
group_test = pandas.read_csv(f'./temp_dataset_test.txt', sep=';')

# Подготавливаем листы к записи
sheets_sample = {'SAMPLE': group_sample, 'DATA': group_data,
                 'VT': group_vt, 'VALID': group_valid, 'TEST': group_test}

```

```

writer = pandas.ExcelWriter(f'./dataset.xlsx', engine='openpyxl')

for sheet_name in sheets_sample.keys(): # Записываем каждый документ в отдельный лист
    sheets_sample[sheet_name].to_excel(writer, sheet_name=sheet_name, engine='openpyxl',
index=False)

writer.save()

sheets = {'DATA': group_dataset}

writer = pandas.ExcelWriter(f'./sample.xlsx', engine='openpyxl')

for sheet_name in sheets.keys():
    sheets[sheet_name].to_excel(writer, sheet_name=sheet_name, engine='openpyxl',
index=False)

writer.save()
# endregion

# region Удаляем временные txt файлы
if os.path.exists(f'./dataset.txt'):
    os.remove('./dataset.txt')
if os.path.exists(f'./dataset_sample.txt'):
    os.remove('./dataset_sample.txt')
if os.path.exists(f'./temp_dataset.txt'):
    os.remove('./temp_dataset.txt')
if os.path.exists(f'./temp_dataset_vt.txt'):
    os.remove('./temp_dataset_vt.txt')
if os.path.exists(f'./temp_dataset_valid.txt'):
    os.remove('./temp_dataset_valid.txt')
if os.path.exists(f'./temp_dataset_test.txt'):
    os.remove('./temp_dataset_test.txt')
# endregion

def save_v2():
    header = (f'X1;X2;X3;X4;X5;X6;X7;X8;X9;X10;X11;X12;X13;X14;X15;X16; '
              f'X17;X18;X19;X20;X21;X22;X23;X24;X25;X26;X27;X28;X29;X30;D1\n')
    sample_frow = (f'Бюджет; '
                  f'Длительность; '
                  f'Боевик; '
                  f'Приключение; '
                  f'Драма; '
                  f'Комедия; '
                  f'Криминальная; '
                  f'Мистика; '
                  f'Ужасы; '
                  f'Исторический; '
                  f'Анимация; '
                  f'Файнтастика; '
                  f'Триллер; '
                  f'Мюзикл; '
                  f'0+; '
                  f'6+; '
                  f'12+; '
                  f'16+; '
                  f'18+; '
                  f'Наличие франшизы; '
                  f'Зима; '
                  f'Весна; '
                  f'Лето; '
                  f'Осень; '
                  f'Период высокой посещаемость; '
                  f'Рейтинг режиссера; '
                  f'Режиссер престижные награды; '
                  f'Сценаристы престижные награды; '
                  f'Актеры престижные награды; '
                  f'Оскары; '
                  f'Кассовые сборы\n')

```

```

with open(f'./dataset_sample.txt', 'w', encoding='utf-8') as f:
    f.write(header)
    f.write(sample_frow)

if __name__ == '__main__':
    versions = ['D1', 'D2', '']
    dataset, converted_dataset = [], []

    with open(f'./raw_dataset.txt', 'r', encoding='utf-8') as temp_fdata:
        for irow in temp_fdata:
            dataset.append(eval(irow.replace('\n', '')))

    dataset = del_copy(dataset) # удаляем копии
    converted_dataset = convert_dataset(dataset)

    # записываем заголовок
    with open(f'./raw_data.txt', 'w', encoding='utf-8') as set_file:
        set_file.write(f'url;id;name;year;duration;imdb-popularity;budget;box-office\n')

    with open(f'./raw_data.txt', 'a', encoding='utf-8') as j_file:
        for jrow in dataset:
            j_file.write(f'{jrow["url"]};{jrow["id"]};{jrow["name"]};{jrow["duration"]};'
                        f'{jrow["genre"]};{jrow["age-limit"]};{jrow["franchise"]};'
                        f'{jrow["release-season"]};{jrow["holiday"]};{jrow["director-'
                        'rating"]};'
                        f'{jrow["directors-awards"]};{jrow["writers-awards"]};{jrow["stars-'
                        'awards"]};'
                        f'{jrow["oscars"]};{jrow["budget"]};{jrow["box-office"]}\n')

    save(dataset)

    # temp_fdata.close()
    # j_file.close()

    # os.remove('./raw_data.txt')

```

ПРИЛОЖЕНИЕ J

Листинг модуля нейронной сети

```
import keras
import matplotlib
import numpy
import os
import pandas
import tensorflow
import matplotlib.pyplot as plt
from keras import layers
from keras import optimizers
from keras.datasets import cifar10
from keras.layers import Dense, Flatten, DropoutWrapper, BatchNormalization
from keras.layers.convolutional import Conv1D, MaxPooling1D
from keras.models import Sequential
from keras.utils import np_utils
from keras import regularizers
from math import sqrt
from matplotlib.pyplot import figure
from sklearn.metrics import mean_squared_error
from sklearn.metrics import median_absolute_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import max_error
from sklearn.metrics import r2_score
plt.style.use('ggplot')

%matplotlib inline
batch_size = 10
LR = 1e-3
epochs = 100

df_train = pandas.read_excel("datasetD1.xlsx", sheet_name="DATA")
df_test = pandas.read_excel("datasetD1.xlsx", sheet_name="TEST")

df_train = df_train.drop(0, axis=0)
df_test = df_test.drop(0, axis=0)

df_train.boxplot('X11')
df_train.boxplot('D1')

train = df_train.values
test = df_test.values

x_train = train[:, 0:11]
y_train = train[:, 11]
x_test = test[:, 0:11]
y_test = test[:, 11]

keras.backend.clear_session()
opt = tensorflow.keras.optimizers.Adam(
    learning_rate=0.006,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    name="Adam")

model = Sequential()
model.add(Dense(3, activation="relu"))
model.add(Dense(1, activation="linear"))
model.compile(optimizer=opt, loss="mse", metrics=["mae"])

history = model.fit(x_train, y_train, validation_split=0.1, verbose=1, epochs=100, shuffle=True)
```

```

model.summary()

scores = model.evaluate(x_test, y_test)
print('\nMAPE: %.2f%%' % (scores[1]))

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.grid(True)
plt.show()

predictions = model.predict(x_test)
def GlDiagram(n):
    # построение графика сравнения прогнозируемых значений и тестовых (фактических) значений температуры
    # Входные данные: y_test, predition
    # n - количество выводимых значений

    %matplotlib inline
    import numpy as np
    import matplotlib.pyplot as plt

    # Задаем смещение равное половине ширины прямоугольника:
    x1 = np.arange(1, n) - 0.2
    x2 = np.arange(1, n) + 0.2

    #y1 = Ytest.to_numpy().flatten()[:n-1]
    #y2 = predition.flatten()[:n-1]

    y1 = y_test.flatten()[:n-1]
    y2 = predictions.flatten()[:n-1]

    fig, ax = plt.subplots()

    ax.bar(x1, y1, width = 0.4)
    ax.bar(x2, y2, width = 0.4)

    ax.set_facecolor('seashell')
    fig.set_figwidth(12)      # ширина Figure
    fig.set_figheight(6)       # высота Figure
    fig.set_facecolor('floralwhite')
    fig.suptitle("Сравнение прогнозного значения и фактического значения результатов выборов")
    fig.legend(["Фактические значения D", "Прогноз Y"], loc = "lower center")
    plt.xlim([0,n])
    plt.show()

# Запуск процедуры:
GlDiagram(n=32)

predictions = model.predict(x_test)

print(f'r2 = {r2_score(y_test, predictions)}')
print(f'max_error = {max_error(y_test, predictions)} [{y_test.shape} {predictions.shape}]')
print(f'mean_absolute_error = {mean_absolute_error(y_test, predictions)}')
print(f'median_absolute_error = {median_absolute_error(y_test, predictions)}')
print(f'MSE = {mean_squared_error(y_test, predictions)}')
#print(f'RMSE = {sqrt(mean_squared_error(y_test, predictions))}')

```

ПРИЛОЖЕНИЕ К

Листинг графического интерфейса

```
# -*- coding: utf-8 -*-

import numpy
import sys
import tensorflow
import time
import keras
from PyQt5 import QtCore, QtGui, QtWidgets
from keras.models import load_model

model = load_model('NeuroSet')

class Ui_MainWindow(object):
    def __init__(self):
        self._mutex = QtCore.QMutex()

    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(793, 367)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayoutWidget = QtWidgets.QWidget(self.centralwidget)
        self.gridLayoutWidget.setGeometry(QtCore.QRect(10, 10, 471, 351))
        self.gridLayoutWidget.setObjectName("gridLayoutWidget")
        self.gridLayout = QtWidgets.QGridLayout(self.gridLayoutWidget)
        self.gridLayout.setContentsMargins(0, 0, 0, 0)
        self.gridLayout.setObjectName("gridLayout")
        self.lblOscars = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblOscars.setObjectName("lblOscars")
        self.gridLayout.addWidget(self.lblOscars, 11, 0, 1, 1)
        self.lblWrAwards = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblWrAwards.setObjectName("lblWrAwards")
        self.gridLayout.addWidget(self.lblWrAwards, 9, 0, 1, 1)
        self.lblStAwards = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblStAwards.setObjectName("lblStAwards")
        self.gridLayout.addWidget(self.lblStAwards, 10, 0, 1, 1)
        self.lblDuration = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblDuration.setObjectName("lblDuration")
        self.gridLayout.addWidget(self.lblDuration, 1, 0, 1, 1)
        self.lblMpaa = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblMpaa.setObjectName("lblMpaa")
        self.gridLayout.addWidget(self.lblMpaa, 3, 0, 1, 1)
        self.lblGenre = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblGenre.setObjectName("lblGenre")
        self.gridLayout.addWidget(self.lblGenre, 2, 0, 1, 1)
        self.lblDirAwards = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblDirAwards.setObjectName("lblDirAwards")
        self.gridLayout.addWidget(self.lblDirAwards, 8, 0, 1, 1)
        self.lblBudget = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblBudget.setMinimumSize(QtCore.QSize(300, 0))
        self.lblBudget.setBaseSize(QtCore.QSize(0, 0))
        self.lblBudget.setObjectName("lblBudget")
        self.gridLayout.addWidget(self.lblBudget, 0, 0, 1, 1)
        self.lblFranchise = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblFranchise.setTextFormat(QtCore.Qt.AutoText)
        self.lblFranchise.setObjectName("lblFranchise")
        self.gridLayout.addWidget(self.lblFranchise, 4, 0, 1, 1)
        self.lblSeason = QtWidgets.QLabel(self.gridLayoutWidget)
        self.lblSeason.setObjectName("lblSeason")
        self.gridLayout.addWidget(self.lblSeason, 5, 0, 1, 1)
        self.lblHoliday = QtWidgets.QLabel(self.gridLayoutWidget)
```

```

self.lblHoliday.setObjectName("lblHoliday")
self.gridLayout.addWidget(self.lblHoliday, 6, 0, 1, 1)
self.lblDirRating = QtWidgets.QLabel(self.gridLayoutWidget)
self.lblDirRating.setObjectName("lblDirRating")
self.gridLayout.addWidget(self.lblDirRating, 7, 0, 1, 1)
self.tbBudget = QtWidgets.QLineEdit(self.gridLayoutWidget)
self.tbBudget.setObjectName("tbBudget")
self.gridLayout.addWidget(self.tbBudget, 0, 1, 1, 1)
self.tbDuration = QtWidgets.QLineEdit(self.gridLayoutWidget)
self.tbDuration.setObjectName("tbDuration")
self.gridLayout.addWidget(self.tbDuration, 1, 1, 1, 1)
self.tbOscars = QtWidgets.QLineEdit(self.gridLayoutWidget)
self.tbOscars.setObjectName("tbOscars")
self.gridLayout.addWidget(self.tbOscars, 11, 1, 1, 1)
self.cbGenre = QtWidgets.QComboBox(self.gridLayoutWidget)
self.cbGenre.setObjectName("cbGenre")
self.cbGenre.addItem("")
self.gridLayout.addWidget(self.cbGenre, 2, 1, 1, 1)
self.cbMpaa = QtWidgets.QComboBox(self.gridLayoutWidget)
self.cbMpaa.setObjectName("cbMpaa")
self.cbMpaa.addItem("")
self.cbMpaa.addItem("")
self.cbMpaa.addItem("")
self.cbMpaa.addItem("")
self.cbMpaa.addItem("")
self.gridLayout.addWidget(self.cbMpaa, 3, 1, 1, 1)
self.cbFranchise = QtWidgets.QComboBox(self.gridLayoutWidget)
self.cbFranchise.setObjectName("cbFranchise")
self.cbFranchise.addItem("")
self.cbFranchise.addItem("")
self.gridLayout.addWidget(self.cbFranchise, 4, 1, 1, 1)
self.cbSeason = QtWidgets.QComboBox(self.gridLayoutWidget)
self.cbSeason.setObjectName("cbSeason")
self.cbSeason.addItem("")
self.cbSeason.addItem("")
self.cbSeason.addItem("")
self.cbSeason.addItem("")
self.gridLayout.addWidget(self.cbSeason, 5, 1, 1, 1)
self.cbHoliday = QtWidgets.QComboBox(self.gridLayoutWidget)
self.cbHoliday.setObjectName("cbHoliday")
self.cbHoliday.addItem("")
self.cbHoliday.addItem("")
self.gridLayout.addWidget(self.cbHoliday, 6, 1, 1, 1)
self.cbDirRating = QtWidgets.QComboBox(self.gridLayoutWidget)
self.cbDirRating.setObjectName("cbDirRating")
self.cbDirRating.addItem("")
self.cbDirRating.addItem("")
self.cbDirRating.addItem("")
self.cbDirRating.addItem("")
self.cbDirRating.addItem("")
self.cbDirRating.addItem("")
self.cbDirRating.addItem("")
self.cbDirRating.addItem("")
self.cbDirRating.addItem("")
self.gridLayout.addWidget(self.cbDirRating, 7, 1, 1, 1)

```

```

        self.cbDirAwards = QtWidgets.QComboBox(self.gridLayoutWidget)
        self.cbDirAwards.setObjectName("cbDirAwards")
        self.cbDirAwards.addItem("")
        self.cbDirAwards.addItem("")
        self.gridLayout.addWidget(self.cbDirAwards, 8, 1, 1, 1)
        self.cbWrAwards = QtWidgets.QComboBox(self.gridLayoutWidget)
        self.cbWrAwards.setObjectName("cbWrAwards")
        self.cbWrAwards.addItem("")
        self.cbWrAwards.addItem("")
        self.gridLayout.addWidget(self.cbWrAwards, 9, 1, 1, 1)
        self.cbStAwards = QtWidgets.QComboBox(self.gridLayoutWidget)
        self.cbStAwards.setObjectName("cbStAwards")
        self.cbStAwards.addItem("")
        self.cbStAwards.addItem("")
        self.gridLayout.addWidget(self.cbStAwards, 10, 1, 1, 1)
        self.btnExit = QtWidgets.QPushButton(self.centralwidget)
        self.btnExit.setGeometry(QtCore.QRect(490, 10, 75, 23))
        self.btnExit.setObjectName("btnExit")
        self.lblOutput = QtWidgets.QLabel(self.centralwidget)
        self.lblOutput.setGeometry(QtCore.QRect(490, 40, 291, 311))
        self.lblOutput.setText("")
        self.lblOutput.setObjectName("lblOutput")
        MainWindow.setCentralWidget(self.centralwidget)

    self.retranslateUi(MainWindow)
    QtCore.QMetaObject.connectSlotsByName(MainWindow)

    self.callPredict()

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Предсказание рентабельности
кинокартин"))
    self.lblOscars.setText(_translate("MainWindow", "Суммарное количество оскаров у
съемочной группы"))
    self.lblWrAwards.setText(_translate("MainWindow", "<html><head><body><p>Имеются ли у
сценариста престижные награды<br>(«Оскар», «Золотой глобус», SAAG, Critics' Choice
Awards)<br></body></html>"))
    self.lblStAwards.setText(_translate("MainWindow", "Имеются ли у актеров престижные
награды<br>(«Оскар», «Золотой глобус», SAAG, Critics' Choice Awards)<br>"))
    self.lblDuration.setText(_translate("MainWindow", "Продолжительность фильма (в
минутах)"))
    self.lblMpaa.setText(_translate("MainWindow", "Возрастное ограничение фильма"))
    self.lblGenre.setText(_translate("MainWindow", "Основной жанр фильма"))
    self.lblDirAwards.setText(_translate("MainWindow", "Имеются ли у режиссера престижные
награды<br>(«Оскар», «Золотой глобус», SAAG, Critics' Choice Awards)<br>"))
    self.lblBudget.setText(_translate("MainWindow", "Бюджет фильма (доллары США)"))
    self.lblFranchise.setText(_translate("MainWindow", "<html><head><body><p>Является ли
фильм продолжением кинофраншизы<br>(Является следующей частью уже
вышедшего)<br></p></body></html>"))
    self.lblSeason.setText(_translate("MainWindow", "Планируемый сезон выхода"))
    self.lblHoliday.setText(_translate("MainWindow", "Планируется ли выход в период
высокой посещаемости"))
    self.lblDirRating.setText(_translate("MainWindow", "Рейтинг предыдущих работ
режиссера"))

    self.cbGenre.setItemText(0, _translate("MainWindow", "Боевик (Action)"))
    self.cbGenre.setItemText(1, _translate("MainWindow", "Приключение (Adventure)"))
    self.cbGenre.setItemText(2, _translate("MainWindow", "Драма"))
    self.cbGenre.setItemText(3, _translate("MainWindow", "Комедия"))
    self.cbGenre.setItemText(4, _translate("MainWindow", "Криминальный фильм"))
    self.cbGenre.setItemText(5, _translate("MainWindow", "Мистика"))
    self.cbGenre.setItemText(6, _translate("MainWindow", "Хоррор"))
    self.cbGenre.setItemText(7, _translate("MainWindow", "Исторический"))
    self.cbGenre.setItemText(8, _translate("MainWindow", "Документальный"))
    self.cbGenre.setItemText(9, _translate("MainWindow", "Анимация"))
    self.cbGenre.setItemText(10, _translate("MainWindow", "Фантастика"))
    self.cbGenre.setItemText(11, _translate("MainWindow", "Триллер"))
    self.cbGenre.setItemText(12, _translate("MainWindow", "Мюзикл"))
    self.cbMpaa.setItemText(0, _translate("MainWindow", "0+"))

```

```

self.cbMpaa.setItemText(1, _translate("MainWindow", "6+"))
self.cbMpaa.setItemText(2, _translate("MainWindow", "12+"))
self.cbMpaa.setItemText(3, _translate("MainWindow", "16+"))
self.cbMpaa.setItemText(4, _translate("MainWindow", "18+"))
self.cbFranchise.setItemText(0, _translate("MainWindow", "Нет"))
self.cbFranchise.setItemText(1, _translate("MainWindow", "Да"))
self.cbSeason.setItemText(0, _translate("MainWindow", "Зима"))
self.cbSeason.setItemText(1, _translate("MainWindow", "Весна"))
self.cbSeason.setItemText(2, _translate("MainWindow", "Лето"))
self.cbSeason.setItemText(3, _translate("MainWindow", "Осень"))
self.cbHoliday.setItemText(0, _translate("MainWindow", "Нет"))
self.cbHoliday.setItemText(1, _translate("MainWindow", "Да"))
self.cbDirRating.setItemText(0, _translate("MainWindow", "0"))
self.cbDirRating.setItemText(1, _translate("MainWindow", "1"))
self.cbDirRating.setItemText(2, _translate("MainWindow", "2"))
self.cbDirRating.setItemText(3, _translate("MainWindow", "3"))
self.cbDirRating.setItemText(4, _translate("MainWindow", "4"))
self.cbDirRating.setItemText(5, _translate("MainWindow", "5"))
self.cbDirRating.setItemText(6, _translate("MainWindow", "6"))
self.cbDirRating.setItemText(7, _translate("MainWindow", "7"))
self.cbDirRating.setItemText(8, _translate("MainWindow", "8"))
self.cbDirRating.setItemText(9, _translate("MainWindow", "9"))
self.cbDirRating.setItemText(10, _translate("MainWindow", "10"))
self.cbDirAwards.setItemText(0, _translate("MainWindow", "Нет"))
self.cbDirAwards.setItemText(1, _translate("MainWindow", "Да"))
self.cbWrAwards.setItemText(0, _translate("MainWindow", "Нет"))
self.cbWrAwards.setItemText(1, _translate("MainWindow", "Да"))
self.cbStAwards.setItemText(0, _translate("MainWindow", "Нет"))
self.cbStAwards.setItemText(1, _translate("MainWindow", "Да"))
self.btnExit.setText(_translate("MainWindow", "Рассчитать"))

def callPredict(self):
    self.btnExit.clicked.connect(lambda: self.write_data([
        self.tbBudget.text(),
        self.tbDuration.text(),
        self.cbGenre.currentText(),
        self.cbMpaa.currentText(),
        self.cbFranchise.currentText(),
        self.cbSeason.currentText(),
        self.cbHoliday.currentText(),
        self.cbDirRating.currentText(),
        self.cbDirAwards.currentText(),
        self.cbWrAwards.currentText(),
        self.cbStAwards.currentText(),
        self.tbOscars.text()]))

```

```

def write_data(self, data):
    self.lblOutput.setText('')

    genre_format = {'Боевик (Action)': 1, 'Приключение (Adventure)': 2,
                    'Драма': 3, 'Комедия': 4, 'Криминальный фильм': 5,
                    'Мистика': 6, 'Хоррор': 7, 'Исторический': 8,
                    'Документальный': 9, 'Анимация': 10, 'Фантастика': 11,
                    'Триллер': 12, 'Мюзикл': 13}
    agelimit_format = {'0+'.format(): 1, '6+'.format(): 2, '12+'.format(): 3, '16+'.format(): 4, '18+'.format(): 5}
    yesno_format = {'Нет': 0, 'Да': 1}
    season_format = {'Зима': 4, 'Весна': 1, 'Лето': 2, 'Осень': 3}

    data1, data2, data3 = False, False, False
    budget, duration, genre, mpaa, franchise, season, holiday = 0, 0, 0, 0, 0, 0, 0
    dirrating, dirawards, wrawards, stawards, oscars = 0, 0, 0, 0, 0

    if str(data[0]).isdigit():
        if 100000 <= int(data[0]):
            data1 = True

```



```

[4 if mpaa is not 4 else 5, duration, season, holiday,
dirwards, w rewards,
stowards, oscars, genre, franchise, budget]])
# print(budget, duration, genre, mpaa, franchise, season, holiday,
#        dirrating, dirwards, rewards, stowards, oscars)

result = 0
self.btnPredict.setText('Ожидание')
predictions = model.predict(output)
time.sleep(4)
self.btnPredict.setText('Рассчитать')
pred = round((predictions[0, 0]) / 10) * 10
if pred < 10:
    result = 500000
elif 10 < pred < 20:
    result = 1000000
elif 20 < pred < 30:
    result = 2500000
elif 30 < pred < 40:
    result = 5000000
elif 40 < pred < 50:
    result = 7500000
elif 50 < pred < 60:
    result = 10000000
elif 60 < pred < 70:
    result = 15000000
elif 70 < pred < 80:
    result = 20000000
elif 80 < pred < 90:
    result = 30000000
elif 90 < pred:
    result = 40000000
self.lblOutput.setText(f'Прогнозируемые кассовые сборы: ~${result}')
# self.lblOutput.setText(f'Прогнозируемые кассовые сборы{}')

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

```

ПРИЛОЖЕНИЕ L

Дополнительные иллюстрации к третьей главе

The screenshot shows the IMDb movie page for 'Dunkirk'. At the top, the title 'Dunkirk' is displayed with a red border around it. Below the title, the release year '2017 - PG-13 - 1h 46m' is shown. To the right, there are sections for 'IMDb RATING 7.8/10 634K', 'YOUR RATING', and 'POPULARITY 341 + 35'. Below these are links for 'Cast & crew', 'User reviews', 'Trivia', 'IMDbPro', and 'All topics'. A large thumbnail image of the movie's poster is on the left, showing a close-up of a soldier's face. In the center, a video player shows a scene of many soldiers in green uniforms and helmets looking out over water. A 'Play trailer 0:31' button is visible. To the right, there are sections for '13 VIDEOS' and '99+ PHOTOS'. Below the main poster, there are tabs for 'Action', 'Drama', and 'History'. A summary text reads: 'Allied soldiers from Belgium, the British Commonwealth and Empire, and France are surrounded by the German Army and evacuated during a fierce battle in World War II.' Below the summary, three sections are highlighted with red boxes: '5 Director Christopher Nolan', '6 Writer Christopher Nolan', and '7 Stars Fionn Whitehead · Barry Keoghan · Mark Rylance'. At the bottom, there are sections for 'Won 3 Oscars 65 wins & 235 nominations total', 'Videos', 'More to explore', 'Storyline', and 'Plot summary'. There are also sections for 'Taglines', 'Genres Action · Drama · History · Thriller · War', 'Motion Picture Rating (MPAA) Rated PG-13 for intense war experience and some language', and 'Parents guide'.

Рисунок L.1 – Визуальное отображение страницы с фильмом часть 1

Did you know

[Edit](#)

Trivia

According to Sir Kenneth Branagh, roughly thirty Dunkirk survivors, who were in their mid-90s, attended the premiere in London, England. When asked about the movie, they felt that it accurately captured the event, but that the soundtrack was louder than the actual bombardment, a comment that greatly amused writer, producer, and director Christopher Nolan.

Goofs

This film contains several errors, including the lack of a visible American flag on the beach.



FAQ >

If there were literally 1000's of armed soldiers on the beach, why wasn't it possible to all shoot at one plane at a time as it approached? Out of a few thousand bullets surely the chances of hitting the plane... >

Is the story line based on the real-life experiences of Commander Charles Lightoller at Dunkirk? >

Details

[Edit](#)

9 **Release date** July 21, 2017 (United States) >

10 **Countries of origin** United Kingdom · Netherlands · France · United States >

Official sites [Official Facebook](#) · [Official Instagram](#) >

Languages English · French · German

Also known as Bodega Bay >

Filming locations Urk, Flevoland, Netherlands >

Production companies Syncopy · Warner Bros. · Dombey Street Productions >

See more company credits at IMDbPro



Box office

[Edit](#)

11 **Budget** \$100,000,000 (estimated) >

Opening weekend US & Canada
\$50,513,488 · Jul 23, 2017

IMDbPro See detailed box office info on IMDbPro

Gross US & Canada

\$189,740,665

Gross worldwide

\$527,016,307

12

Technical specs >

[Edit](#)

Runtime 1 hour 46 minutes

Color Color

Sound mix Sonics-DDP · Dolby Digital

Aspect ratio 2.20 : 1

Related news >



Inside the Messy Music of 'Top Gun: Maverick': Lady Gaga Gets First Scoring Credit, but Song Selection Goes Sideways
May 27 · Variety Film + TV



The Julia Roberts Cannes Interview: 'History Gets So Easily Rewritten in the Masculine Form'
May 24 · Variety Film + TV

Contribute to this page

Suggest an edit or add missing content

IMDb Answers: Help fill gaps in our data >

Learn more about contributing >

[Edit page](#)

Рисунок L.2 – Визуальное отображение страницы с фильмом часть 2

Christopher Nolan (1)

Writer | Producer | Director

1 Top 500

1:50 | Clip 22 VIDEOS | 161 IMAGES

Best known for his cerebral, often nonlinear, storytelling, acclaimed writer-director Christopher Nolan was born on July 30, 1970, in London, England. Over the course of 15 years of filmmaking, Nolan has gone from low-budget independent films to working on some of the biggest blockbusters ever made. At 7 years old, Nolan began making short movies ... See full bio »

Born: July 30, 1970 in London, England, UK

More at IMDbPro » Contact Info: View agent, publicist, legal on IMDbPro

2 Nominated for 5 Oscars. Another 141 wins & 231 nominations. See more awards »

Photos

Filmography

Jump to: Writer | Producer | Director | Cinematographer | Editor | Art department | Composer | Thanks | Self | Archive footage

Writer (19 credits) **Producer** (19 credits) **Director** (17 credits) **Cinematographer** (5 credits) **Editor** (4 credits) **Art department** (1 credit) **Composer** (1 credit) **Thanks** (28 credits) **Self** (91 credits) **Archive footage** (16 credits)

3 **Director (17 credits)**

Оппенгеймер (post-production)	2023
Довод (directed by)	2020
Дюнкерк (directed by)	2017
Кузь (Documentary short)	2015
Интерстеллар (directed by)	2014
Темный рыцарь: Возрождение легенды (directed by)	2012
Начало (directed by)	2010
Темный рыцарь	2008
Престигж	2006
Бэтмен: Начало (directed by)	2005
Синема16: Британские короткометражки (Video)	2003
Бессонница	2002
Помни	2000
Преследование	1998
Жук-скакун (Short) (as Chris Nolan)	1997
Воровство (Short)	1996
Tarantella (TV Short)	1989

Related Videos

Quick Links

- Biography
- Awards
- Photo Gallery
- Filmography (by job)
- Trailers and Videos

Explore More

Star Siblings: Famous Brothers and Sisters

Keep it in the family with a look at some of our favorite Hollywood siblings. See the entire gallery »

Share this page: [Facebook](#) [Twitter](#) [Email](#)

2022 TV Guide: The Best Shows Coming This Year

08.21.2022

The Julia Roberts Cannes Interview: 'History Gets So Easily Rewritten in the Masculine Form' 24 May 2022 | Variety

See all related articles »

Projects In Development

Untitled Keith Gordon Project Details only on IMDbPro »

Editorial Lists

Related lists from IMDb editors

 Top 100 Stars of 2020 a list of 100 people updated 01 Dec 2020
 2018 Oscars: Nominees and Stars a list of 112 people updated 05 Mar 2018
 Golden Globe Nominees: Actors a list of 76 people updated 08 Jan 2018
 Top 100 Stars of 2017 a list of 100 people updated 08 Dec 2017
 Slamdance Alumni a list of 15 images updated 18 Oct 2016

User Lists

Create a list »

Related lists from IMDb users

 directors in 2022 a list of 28 people created 4 months ago

Рисунок L.3 – Визуальное отображение страницы с персоной