|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  Факультет экономики, менеджмента и бизнес-информатики |
|  |
| 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. 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 \_\_ страниц формата А4, из них в основной части \_\_ страниц.

Основная часть работы включает в себя \_\_ иллюстрацию и \_\_ таблиц.

Библиографический список состоит из \_\_ публикаций.

Работа включает в себя \_\_ приложений.

**Оглавление**

[Введение 5](#_Toc99492421)

[Глава 1 Анализ 9](#_Toc99492422)

[1.1 Выявление особенностей прогнозирования кассовых сборов 9](#_Toc99492423)

[1.2 Анализ особенностей систем машинного обучения 15](#_Toc99492424)

[1.2.1. Анализ типов нейронных сетей 16](#_Toc99492425)

[1.2.2. Анализ способов машииного обучения 27](#_Toc99492426)

[1.2.3. Анализ методов машинного обучения 31](#_Toc99492427)

[1.3 Анализ предыдущих работ 31](#_Toc99492431)

[1.4 Выводы по первой главе 32](#_Toc99492432)

[Глава 2 Проектирование 33](#_Toc99492433)

[2.1 Проектирование модуля сбора и систематизации данных 33](#_Toc99492434)

[2.1 Проектирование системы машинного обучения 34](#_Toc99492435)

[2.2 Проектирование графического интерфейса 34](#_Toc99492436)

[2.3 Выводы по второй главе 34](#_Toc99492437)

[Глава 3 Разработка и тестирование 35](#_Toc99492438)

[3.1 Обоснование выбора средств разработки 35](#_Toc99492439)

[3.2 Реализация модуля сбора и систематизации данных 35](#_Toc99492440)

[3.3 Реализация системы машинного обучения 35](#_Toc99492441)

[3.4 Реализация графического интерфейса 37](#_Toc99492442)

[3.5 Выводы по третьей главе 37](#_Toc99492443)

[Заключение 38](#_Toc99492444)

[Список сокращений и условных обозначений 39](#_Toc99492445)

[Библиографический список 40](#_Toc99492446)

[ПРИЛОЖЕНИЕ А Сравнение нейронных сетей 41](#_Toc99492447)

[ПРИЛОЖЕНИЕ Б Сравнение алгоритмов машинного обучения 44](#_Toc99492448)

[Обучение с учителем 44](#_Toc99492449)

[Обучение без учителя 48](#_Toc99492450)

[Частичное обучение 49](#_Toc99492451)

[Трансдуктивное обучение 50](#_Toc99492452)

[Обучение с подкреплением 50](#_Toc99492453)

[Динамическое обучение 51](#_Toc99492454)

[Метаобучение 51](#_Toc99492455)

[ПРИЛОЖЕНИЕ В Листинг алгоритма сбора данных 53](#_Toc99492456)

[ПРИЛОЖЕНИЕ Г Листинг нейронной сети 70](#_Toc99492457)

Введение

Киноиндустрия вносит огромный вклад как в общую социальную жизнь человека, так и в развитие экономики стран, она стимулирует многочисленные внешние эффекты: потребление определенных товаров, посещение представленных в кинокартинах стран, развитие индустрии игр и т.п. По данным отчета «Motion picture association» 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., на 17.8% меньше чем в 2019 году, количество проданных билетов составило 223 млн., против 1,3 млрд в 2019 году. Так же стоит обратить внимание, что доля кинотеатрального проката с каждым годом становится меньше, когда доля онлайн просмотра на стриминговых сервисах 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). 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.

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 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» 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. 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 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.

***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. Провести анализ особенностей кинобизнеса.
   2. Провести анализ особенностей построения систем машинного обучения.
   3. Формализовать существующие методы прогнозирования рентабельности кинобизнеса.
   4. Конкретизировать требования к разрабатываемой системе.
2. Спроектировать систему.
   1. Спроектировать модуль сбора и систематизации данных.
   2. Спроектировать системы машинного обучения.
   3. Спроектировать графический интерфейс разрабатываемой системы.
3. Реализовать систему машинного обучения для прогнозирования рентабельности кинобизнеса.
   1. Обосновать выбор технологий для разработки.
   2. Реализовать модуль сбора и систематизации данных.
   3. Реализовать систему машинного обучения.
   4. Реализовать графический интерфейс разрабатываемой системы.
   5. Провести тестирование системы.

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.

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 [6]. Рассматриваемый ими набор входных переменных включал производственный бюджет, критические обзоры, наличие звезд, наличие франшизы, наличие премий, жанр и рейтинг. Аппарат нейронных сетей для прогнозирования кассовых сборов фильмов был впервые применен в 2002 году американскими учеными R.Sharda и D.Delen. В 2006 году эти же авторы [7] построили модели на основе логистической регрессии, дискриминантного анализа, классификационного и регрессионного дерева, а также нейронной сети, показавшей наилучший результат.

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

1. Анализ

В данной главе будут рассмотрены особенности кинобизнеса и существующих методов прогнозирования. Будут выбраны критерии отбора данных для последующего анализа системами машинного обучения, выявления их преимуществ, недостатков и особенностей. Результатом первой главы являются поставленные требования к разрабатываемой системе.

* 1. Выявление особенностей прогнозирования кассовых сборов

Кинопроизводство процесс создания кинофильмов от первоначального замысла дор 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 кинокартине 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 предпродаж. 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 агрегаторов информации о кассовых сборах фильмов выступает сайт «Box Office Mojo», 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.

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, как уже описывалось ранее, выделили следующие критерии:

* Критические обзоры
* Наличие звезд
* Наличие франшизы
* Наличие премий
* Жанр
* Рейтинг

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.

КиноПоиск – 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:

* 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», 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 – сайт-агрегатор рецензий на кинофильмы, предоставляет расширенную информацию о оценке и рецензях 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 – 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 – 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 сборов 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 тринеровки и тестирования нейронной сети будут производится на сайтах-агрегаторах: «IMDb», «Rotten Tomatoes» и «Box Office Mojo».

Сайт «IMDb» будет использоваться для поиска с 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. Анализ особенностей систем машинного обучения

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 [ ].

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 [ ]. 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 [ ]. 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 [ ].

Различают следующие архитектуры нейронных сетей:

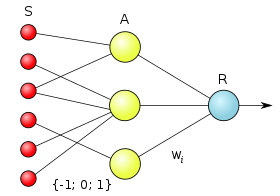
* Персептрон.
* Сверточная сеть.
* Рекурсивная сеть.
* Рекуррентная сеть.
* LSTM.

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

* Обучение с учителем.
* Обучение без учителя.
* Частичное обучение.
* Трансдуктивное обучение.
* Обучение с подкреплением.
* Динамическое обучение.
* Метаобучение.
  + 1. Анализ типов нейронных сетей

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

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 персептрон состоит из элементов трёх типов:

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

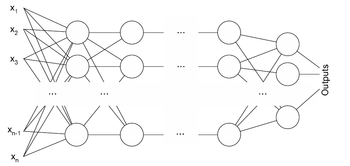
**Рисунок 1.1 – Схема работы персептрона**

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 , на выходе элемента получаем , в противном случае [ ].

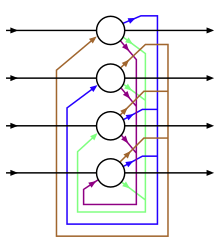
R-элемент является действующим и подсчитывает сумму значений входных сигналов, помноженных на веса, R-элемент который является выводом персептрона, выдаёт сигнал «», если линейная форма превышает порог , иначе на выходе будет «» [ ]:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Обучение элементарного персептрона состоит в изменении весовых коэффициентов связей A-R. Веса связей S-A, принимающих значения . Значения порогов A-элементов выбираются случайным образом в самом начале и затем не изменяются.

После обучения персептрон готов работать в режиме распознавания или обобщения [ ]. В этом режиме персептрону предъявляются ранее неизвестные ему объекты, при предъявлении объекта A-элементы передают сигнал R-элементу, равный сумме соответствующих коэффициентов , 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 персептрона.

**Рисунок 1.2 – Архитектура многослойного персептрона**

**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 [ ]. Формула обучения:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

где – сила связи между нейронами ,

– состояние, , нейрона ,

– состояние, , нейрона .

Сеть Хопфилда однослойная и состоит из 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:

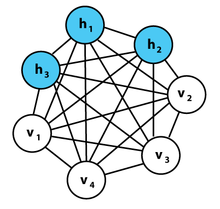
|  |  |  |
| --- | --- | --- |
|  |  | (3) |

где – «энергия» сети,

– элемент матрицы взаимодействий ,

– состояние нейрона ,

– состояние нейрона .

**Машина Больцмана** (Рисунок 1.4) – вид стохастической рекуррентной нейронной сети, изобретенной G. Hinton и T. Sejnowski в 1985 году [ ]. 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:

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

где – сила связи между нейронами ,

– скорость обучения,

– фактическое распределение,

– 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 [ ]:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

где – «энергия» сети,

– сила связи между нейронами ,

– состояние, , нейрона ,

– состояние, , нейрона .

Ограничения:

: – нейрон не может иметь связь с самим собой,

: – все связи являются симметричными.

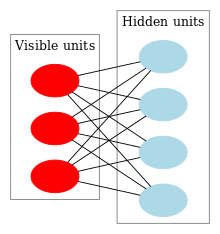
**Ограниченная машина Больцмана или RBM** (Рисунок 1.5) – вид генеративной стохастической нейронной сети, которая определяет распределение вероятности на входных образцах данных, является модификацией обычной машины Больцмана, в которой нейроны разделили на видимые и скрытые, а связи допустимы только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

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

где – сила связи между нейронами ,

– скорость обучения,

- прогноз

В ограниченной машине Больцмана нейроны образуют двудольный граф, с одной стороны графа (Рисунок 1.3) 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 [ ].

**Рисунок 1.5 – Схема сети ограниченной машины Больцмана**

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

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

где – «энергия» сети ,

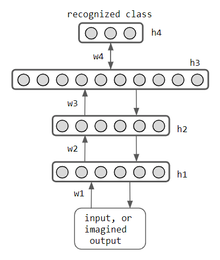
– смещения видимого слоя,

– видимый слой,

– смещения скрытого слоя,

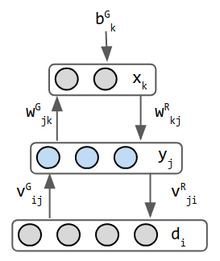
– скрытый слой,

– матрица весов .

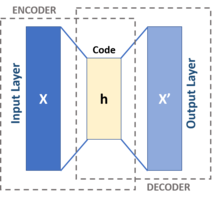
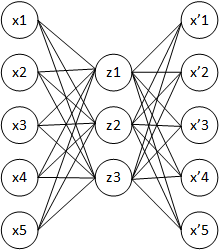
Особенностью ограниченных машин Больцмана является возможность проходить обучение без учителя в виде сети глубоких убеждений или DBN [ ], которую можно рассматривать как совокупность простых обучающих модулей, каждый из которых представляет собой ограниченный тип машины Больцмана, который содержит слой 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 [ ]. 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 выдает вероятность того, что его состояние равно 0 или 1 [ ].

Машины Гельмгольца обычно обучаются с использованием неконтролируемого алгоритма обучения, такого как алгоритм бодрствования-сна [ ]. 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).

**Рисунок 1.7 – Схема машины Гельмгольца**

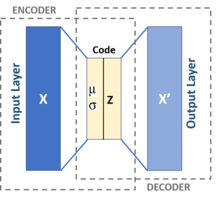
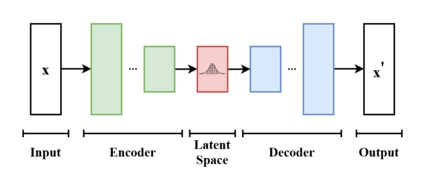
**Автокодировщик** (Рисунок 1.8 и 1.9) - архитектура искусственных нейронных сетей, позволяющая применять обучение без учителя при использовании метода обратного распространения ошибки [ ]. Простейшая архитектура автокодировщика – сеть прямого распространения, без обратных связей, наиболее схожая с персептроном и содержащая входной слой, промежуточный слой и выходной слой. В отличие от персептрона, 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 [ ]. Для улучшения результатов на нейросеть накладывают ограничения: уменьшение размерности промежуточного слоя (промежуточный входной) или искусственное ограничение единовременно активных нейронов промежуточного слоя, из-за чего нейросеть 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, нейросеть может самостоятельно обучиться распознавать линии и полосы под различными углами [ ].

**Рисунок 1.9 – Нейронная сеть автокодировщика**

**Рисунок 1.8 – Визуализация структуры автокодировщика**

Чаще всего автокодировщики применяют каскадно для обучения глубоких (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 [ ].

**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 [ ]. Архитектурно близок к простому автокодировщику, но с различиями в цели математической формулировки [ ]. Вариационные автокодировщики позволяют переписывать задачи статистического вывода как задачи статистической оптимизации (т.е. находить значения параметров, которые минимизируют некоторую целевую функцию) [ ]. Они предназначены для сопоставления входной переменной с многомерным скрытым распределением. Хотя этот тип модели изначально был разработан для обучения без учителя [ ], его эффективность была доказана в других областях машинного обучения: частичное обучение [ ] или обучение с учителем [ ].

В вариационном автоэнкодере входные данные отбираются из параметризованного распределения (предшествующего, в терминах байесовского 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) [ ].

**Рисунок 1.11 – Схема работы вариационного автокодировщика**

**Рисунок 1.10 – Визуализация структуры 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.

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Где – параметризованное распределение,

– набор данных (датасет),

– скрытая кодировка.

Так же существуют и другие виды нейронных сетей, являющиеся расширением или совокупность перечисленных выше сетей, подробнее все виды нейронных сетей обозрены в приложении (см. ПРИЛОЖЕНИЕ А), 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 А) 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** | **Сферы применения** | **Сильные стороны** | **Слабые стороны** |
| --- | --- | --- | --- |
| *Персептрон* | 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 |
| *Нейронная сеть Джордана* | То же что и в Персептроне | Нейроны имеют обратную связь | – |
| *Нейронная сеть Коско* | Выявление ассоциаций | Адаптивность | Малый объём памяти |
| *Машина Гельмгольца* | Создание представлений, мимикрия | Анатомический, анализируемый с теорией информации | – |

* + 1. Анализ способов машииного обучения

**Обучение с учителем** – наиболее распространённый случай. Существует некоторая зависимость между ответами и объектами, но она неизвестна, известна только конечная совокупность попарных прецедентов (объект, ответ) [ ]. Требуется найти функциональную зависимость ответов от описаний объектов и построить алгоритм, принимающий на входе описание объекта и выдающий на выходе ответ. Под учителем понимается либо сама обучающая выборка, либо тот, кто указал на заданных объектах правильные ответы. Для измерения точности ответов, так же как и в обучении на примерах, может вводиться функционал качества, который обычно определяется как средняя ошибка ответов, выданных алгоритмом, по всем объектам выборки [ ].

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 [ ].

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 регулирующие устройство (например, термостат) или человек-оператор (учитель), способный реагировать на реакции испытываемой системы и стимулы внешней среды путём применения особых правил подкрепления, изменяющих состояние памяти системы.

**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, которая является важным способом обучения у людей, машина вынуждена создавать компактное внутреннее представление своего мира, а затем генерировать из него образный контент [ ]. В отличие от контролируемого обучения, где данные помечаются экспертом, например, как "мяч" или "рыба", 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 объектов (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.

**Частичное обучение** – способ машинного обучения, разновидность обучения с учителем, которое также использует неразмеченные данные для тренировки — обычно небольшое количество размеченных данных и большое количество неразмеченных данных [ ].

Обучение с частичным привлечением учителя занимает промежуточную позицию между обучением без учителя (без привлечения каких-либо размеченных данных для тренировки) и обучением с учителем (с привлечением лишь размеченных данных).

Многие исследователи машинного обучения обнаружили, что неразмеченные данные, при использовании в сочетании с небольшим количеством размеченных данных, могут значительно улучшить точность обучения.

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.

**Трансдуктивное обучение** – полу-контролируемое обучение (частичное обучение), обучение с частичным привлечением учителя, 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.

**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 (а не специальной системы управления подкреплением, как это происходит в обучении с учителем) 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.

**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.

**Метаобучение** – отличается тем, что прецедентами являются ранее решённые задачи обучения. Требуется определить, какие из используемых в них эвристик работают более эффективно. Конечная цель – обеспечить постоянное автоматическое совершенствование алгоритма обучения с течением времени.

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.

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.

* + 1. Анализ методов машинного обучения

Метод коррекции ошибки

Метод обратного распространения ошибки

Метод опорных векторов

Метод ближайших соседей

1. 3. Анализ предыдущих работ

В 2002 году американские ученые Р.Шарда и Д.Делен [ ] разработали аналогичное исследование с использованием нейронных сетей для прогнозирования кассовых сборов фильмов. В 2006 году те же авторы построили модели, основанные на логистической регрессии, дискриминантном анализе, классификации и деревьях регрессии, а также преобразовали задачу прогнозирования в задачу классификации. В 2009 году [ ] они также решили улучшить прогнозирование, включив деревья решений и более полную выборку фильмов.

В 2009 году У. Чанг и К.Дж. Ли [ ] предложили использовать байесовскую сеть убеждений для выявления причинно-следственных связей при прогнозировании кассового успеха корейских фильмов. 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 [ ] и YouTube [ ], от рейтинга и популярности фильма были предложены исследователями А. Огиной, М. Бреуссом и др. [ ] в 2012 году. Кроме того, исследователи провели анализ настроений комментариев пользователей Twitter и их причинно-следственной связи с рейтингом фильма на платформе IMDb [ ].

В статьях, описанных выше, мало внимания уделяется применению совокупности всех накопленных знаний, а также недостаточному значению социальных сетей и рекламы. В этой статье будет расширена выборка для прогнозирования, наряду с использованием технологий и корреляций, используемых в описанных работах, а также будут предложены новые соотношения для прибыльности кинобизнеса, извлеченные из переменных.

* 1. Выводы по первой главе

В результате анализа особенностей кинобизнеса и существующих методов прогнозирования были выбраны критерии отбора данных для последующего анализа системами машинного обучения. Были выдвинуты функциональные требования, предъявляемые к системе. Проведен краткий обзор и сравнение систем и способов написания нейронных сетей. Рассмотрены плюсы и минусы аналогов и определены преимущества разрабатываемой системы.

1. Проектирование

В представленной главе спроектированы и рассмотрены методы и алгоритмы, которые потребуются для реализации системы и для удовлетворения требований, выявленных в ходе анализа. Результатом данной главы является спроектированная система и её интерфейс.

* 1. Проектирование модуля сбора и систематизации данных

1. При построении нейросетевой модели в качестве входных данных было решено выбрать параметры, представленные в таблице 1.

**Таблица 2.1. Входные параметры нейронной сети**

|  |  |  |
| --- | --- | --- |
| Номер | Название | Расшифровка, комментарии |
| X1 | Возрастное ограничение фильма | Целое число  (закодированы в числа от 1 до 5  0+: 1; 6+: 2; 12+: 3; 16+: 4; 18+: 5) |
| X2 | Длительность фильма в минутах | Целое число |
| X3 | Сезон выхода фильма | Целое число (Закодированы в числа от 1 до 4: 1-весна, 2-лето, 3-осень, 1-зима) |
| X4 | Выход фильма в период высокой посещаемости кинотеатров | Шкала 0/1, описывает вышел ли фильм в период летних или зимних каникул или иных длинных праздников |
| X5 | Наличие у режиссеров престижных наград | Шкала 0/1, описывает наличие у режиссера наличие таких наград как «Оскар», «Золотой глобус», SAAG и др. |
| X6 | Наличие у сценаристов престижных наград | Шкала 0/1, описывает наличие у сценаристов наличие таких наград как «Оскар», «BAFTA» и др. |
| X7 | Наличие у 3-х звезды на главных ролях престижных наград | Шкала 0/1, описывает наличие у каждого из 3х звезд на главных ролях наличие таких наград как «Оскар», «Золотой глобус», SAAG и др. |
| X8 | Количество оскаров у съемочной группы | Целое число, суммарное количество полученных премий «Оскар» у звезд, режиссера и сценаристов |
| X9 | Основной жанр фильма | Целое число (Закодированы в числа от 1 до 25: 1-Action, 2-Adventure, 3-Drama и т.д.) |
| X10 | Является ли фильм частью франшизы | Шкала 0/1, описывает является ли фильм продолжением франшизы |
| X11 | Бюджет фильма | Положительное число (в долларах США) |

Выходной параметр (Y1) – Кассовые сборы фильма (сгруппированные по множествам $1-2,5 млн, $2,5-5 млн и т. д.).

* 1. Проектирование системы машинного обучения
  2. Проектирование графического интерфейса
  3. Выводы по второй главе

На основании результатов, полученных из первой главы, была спроектирована алгоритмическая часть системы.

1. Разработка и тестирование

Данная глава обозревает реализацию и тестирование конечного продукта. В данной главе рассматриваются все основы создания системы и ее тестирование. Итоговым результатом третьей главы является конечный продукт.

* 1. Обоснование выбора средств разработки
  2. Реализация модуля сбора и систематизации данных

Обучающее множество было собрано с помощью программы-парсера с интернет-ресурса IMDb [ ]. Всего были собраны данные о 1000 фильмах из рейтинга топ-500 и низ-500, после чего была произведена очистка данных от выбросов, пустых и нестандартных данных, таким образом в итоговое множество вошли 632 примера. Данное множество было разделено на обучающее и тестирующее в соотношении 80% к 20%.

Полный листинг программы приведен в приложении В.

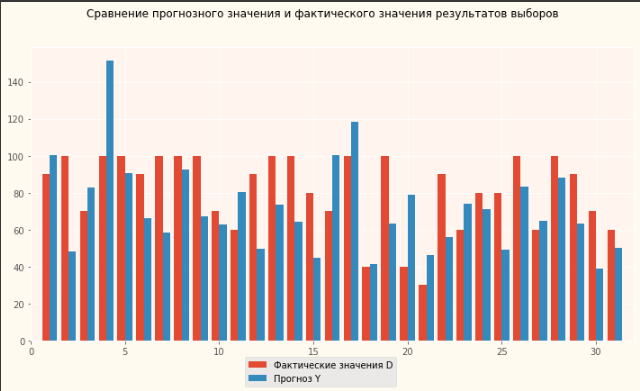
* 1. Реализация системы машинного обучения

Полный листинг программы приведен в приложении Г.

**Рисунок 3.1 – Результат тестирования нейронной сети**

Как видно из рисунка 3.2, наиболее значимым являются бюджет фильма. На втором и третьем месте по значимости являются количество оскаров у съемочной группы и Наличие наград у сценаристов.

**Рисунок 3.2 – Значимость входных параметров нейросетевой модели**

Далее был разработан нейросетевой симулятор с применением языка программирования «Python», с использованием библиотеки «Keras». Средняя ошибка тестирования с помощью разработанной нейросети составила 24.2%. На рисунке 3.3 представлена гистограмма, демонстрирующая разницу между фактическими и прогнозируемыми нейросетью кассовыми сборами.

**Рисунок 3.3 – Результат тестирования нейронной сети**

Далее было проведено исследование кинорынка. Исследование производилось с помощью метода «замораживания» [ ], суть которого заключается в варьировании значения одного параметра и фиксировании значений всех других параметров. Данный метод позволяет выявить влияние исследуемого параметра на выходной. Для этого были выбраны 3 случайных фильма. Результаты представлены в виде графика на риунке 3.4.

**Рисунок 3.4 Зависимость кассовых сборов от бюджета фильма**

Опираясь на данные результаты, можно с уверенностью сказать, что бюджет влияет на итог кассовых сборов, но данное влияние просматривается по-разному для каждого фильма, в зависимости от совокупности побочных входных данных. Полученные результаты исследования не противоречат действительности, что подтверждает, что спроектированную нейронную сеть можно считать пригодной для прогнозирования кассовых сборов фильмов.

* 1. Реализация графического интерфейса
  2. Выводы по третьей главе

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.

Список сокращений и условных обозначений

API (Application Programming Interface) – набор классов, процедур, функций, структур или констант, которыми одна компьютерная программа может взаимодействовать с другой программой.

Dataset – набор данных, также иногда используется транслитерация «Датасет»

DBM (deep Boltzmann machine) – глубокая машина Больцмана

LSTM (Long short-term memory) – сети долгой краткосрочной памяти, разновидность архитектуры рекуррентных нейронных сетей

MVP (minimum viable product) – минимально жизнеспособный продукт, продукт обладающий минимальными, но достаточными для удовлетворения первых потребителей функциями.

RBM (restricted Boltzmann machine) – ограниченная машина Больцмана

UML (Unified Modeling Language) – унифицированный язык моделирования, система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования

UI (user interface) – интерфеейс пользователя, он же пользовательский интерфейс, интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы

VAE (variational autoencoder) – вариационный автокодировщик, разновидность автокодировщика

Агрегатор – тот, кто собирает и группирует объекты

Парсинг (parsing) – процесс сопоставления линейной последовательности слов естественного или формального языка с его формальной грамматикой.

ПО – программное обеспечение.

Скрапинг или веб-скрапинг (web scraping) – технология получения веб-данных путем извлечения их со страниц веб-ресурсов

Стриминговый сервис – платформа, которая подбирает фильмы, сериалы, игровые стримы или ТВ-каналы под пользовательские интересы (пример: «Netflix», «КинопоискHD», «Disney+» и др.).

Библиографический список

1. Черепанов Ф.М., Ясницкий Л.Н. Симулятор нейронных сетей «Нейросимулятор 1.0». // Свидетельство об отраслевой регистрации разработки № 8756. Зарегистрировано в Отраслевом фонде алгоритмов и программ 12.07.2007.
2. Ясницкий Л.Н. Введение в искусственный интеллект: учеб. пособие для студ. высш. учеб. заведений/ Л.Н. Ясницкий. – 3-е изд. – М.: Издательский центр «Академия», 2010. – 176с.
3. Ясницкий Л.Н. Пермская научная школа искусственного интеллекта и ее инновационные проекты / Л.Н.Ясницкий, В.В.Бондарь, С.Н.Бурдин и др.; под ред. Л.Н.Ясницкого. – 2-е изд. – Москва-Ижевск: НИЦ «Регулярная и хаотическая динамика», 2008. – 75 с.
4. Б. Н. Коноплёв. Основы фильмопроизводства / В. С. Богатова. — 2-е изд.. — М.: «Искусство», 1975. — 448 с.
5. Internet Movie Data Base – [Электронный ресурс]/ Режим доступа: <http://imdb.com>
6. Kinopoisk – [Электронный ресурс]/ Режим доступа: <http://kinopoisk.ru>
7. Prag J. and Casavant J. An Empirical Study of the Determinants of Revenue and Marketing Expenditures in the Motion Picture Industry // Journal of Cultural Economics. – 1994. – No 18(3). – Pp. 217-235.
8. Sharda R., Delen D. Predicting box-office success of motion pictures with neural networks // Expert Systems with Applications. – 2006. – No 30. – Pp. 243-254.

ПРИЛОЖЕНИЕ А   
Сравнение нейронных сетей

**Таблица A.1** **– Сравнение вариантов построения нейронных сетей 1 часть**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Сферы применения** | **Вид нейронов** | **Вид связей** | **Обучение** | **Возможности модели** | **Ограничения модели** |
| *Персептрон* | Прогнозирование, управление агентами, слабая возможность классификации | Бинарное состояние (0, 1) | A-R связи | Изменение весовых коэффициентов связей A-R {−1; 0; +1} | Способность к обучению по простому и эффективному алгоритму | Качество прогноза и точность построенной модели зависит от числа знаний, используемых при построении модели |
| *Нейронная сеть Хопфилда* | Автоматизация (CAM), 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 (2) | Наследует уравнения физических систем | Относительно небольшой объём памяти |
| *Машина Больцмана* | Автоматизация (CAM), оптимизация | То же что и в нейронной сети Хопфилда | 2-слойный с симметричными весами. 1 скрытый и 1 видимый | Отображено в формуле (4) | То же что и в нейронной сети Хопфилда | Боковые соединения усложняют тренировку |
| *RBM* | 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 (6) | То же что и в машине Больцмана с более быстрым обучением | Увеличение кол-ва итераций из-за равновесия |

**Таблица А.2 – Сравнение вариантов построения нейронной сети 2 часть**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Сферы 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 |
| *Машина 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 А.3 – Сравнение вариантов построения нейронной сети 3 часть**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Сферы применения** | **Вид нейронов** | **Вид связей** | **Обучение** | **Возможности модели** | **Ограничения модели** |
| *Нейронная сеть Коско* | 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 | – |
| *Нейронная сеть Джордана* | 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 | – |
| *Нейронная сеть Хэмминга* | 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 | – | – |
| *Нейронная сеть Элмана* | 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 | – |
| *Свёрточная нейронная сеть* | 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 Б   
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 (объект, ответ). Существует неизвестная зависимость между ответами и объектами, но она неизвестна, известна только конечная совокупность прецедентов [ ]. 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 [ ].

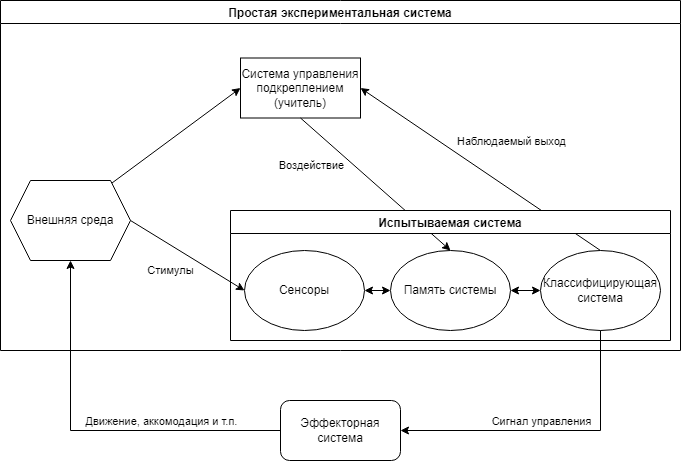
С точки зрения кибернетики, является одним из видов кибернетического эксперимента. Данный эксперимент представляет собой частный случай кибернетического эксперимента с обратной связью. Постановка данного эксперимента предполагает наличие экспериментальной системы, метода обучения и метода испытания системы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

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 регулирующие 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 испытываемой системы не изменяет состояние внешней среды (Рисунок А.1).

**Рисунок А.1 – Визуальное описание обучения 1 вариант**

* Реакция системы изменяет стимулы внешней среды (Рисунок А.2).

**Рисунок А.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 – признаковым 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:
  + персептрон;
  + многослойный персептрон;
  + сети векторного квантования, обучаемые с учителем (Learning Vector Quantization);
  + гибридная сеть встречного распространения;
* Линейный разделитель:
  + линейный дискриминант Фишера;
  + однослойный персептрон;
  + логистическая регрессия;
  + машина опорных векторов.
* Индукция правил:
  + решающее дерево;
  + решающий список;
  + решающий лес;
  + тестовый алгоритм;
  + алгоритм вычисления оценок.
* Алгоритмическая композиция:
  + взвешенное голосование;
  + бустинг;
  + бэггинг;
  + метод комитетов;
  + смесь экспертов.
* Сокращение размерности:
  + селекция признаков;
  + метод главных компонент;
  + метод независимых компонент;
  + многомерное шкалирование.
* Выбор модели:
  + минимизация эмпирического риска;
  + структурная минимизация риска;
  + 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 [ ]. 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 [ ].

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** – 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 [ ].

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:

Типы откликов:

Методы решения:

Трансдуктивное обучение

**Трансдуктивное обучение** – полу-контролируемое обучение (частичное обучение), обучение с частичным привлечением учителя, 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.

Типы входных данных:

Типы откликов:

Методы решения:

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 (а не специальной системы управления подкреплением, как это происходит в обучении с учителем) 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.

Типы входных данных:

Типы откликов:

Методы решения:

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.

Типы входных данных:

Типы откликов:

Методы решения:

Метаобучение

**Метаобучение** – отличается тем, что прецедентами являются ранее решённые задачи обучения. Требуется определить, какие из используемых в них эвристик работают более эффективно. Конечная цель – обеспечить постоянное автоматическое совершенствование алгоритма обучения с течением времени.

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.

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.

Типы входных данных:

Типы откликов:

Методы решения:

ПРИЛОЖЕНИЕ В   
Листинг алгоритма сбора данных

import datetime  
import lxml  
import numpy  
import os  
import pandas  
import re  
import requests  
import time  
from bs4 import BeautifulSoup  
from multiprocessing.dummy import Pool  
from openpyxl import Workbook  
from selenium import webdriver  
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  
  
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'  
  
checked\_counter = 0  
  
open('./raw\_data.xlsx', 'w').close()  
open('./dataset.xlsx', 'w').close()  
open('./datasetD1.xlsx', 'w').close()  
open('./datasetD2.xlsx', 'w').close()  
  
files = ['all',  
 '0-10',  
 '10-40',  
 '40-inf']  
  
  
# region Функция сбора первичных ссылок  
def scrapper(url):  
 data = []  
 start = 0  
 iteration = 1  
  
 while start < 1000:  
 request = requests.get(url, headers=headers)  
 soup = BeautifulSoup(request.content, 'html.parser')  
  
 # находим список фильмов на странице  
 title\_block = soup.find('div', class\_='lister-list').find\_all('div', class\_='lister-item mode-advanced')  
 for it in title\_block:  
 title = it.find('h3', class\_='lister-item-header').find('a')  
 title\_id = title.get('href').replace('/title/tt', '').replace('/?ref\_=adv\_li\_tt', '')  
 data.append(title\_id) # записываем id фильма в список  
 print(  
 f'{url.replace("https://www.imdb.com/search/title/?groups=", "").replace("&start=", "/").replace("&ref\_=adv\_nxt", "")}: '  
 f'{it.find("span", class\_="lister-item-index unbold text-primary").get\_text()} {title\_id}')  
  
 if start + 50 <= len(data): # переходим на след страницу  
 if iteration == 1:  
 url = url + f'&start={str(start + 50 + 1)}&ref\_=adv\_nxt'  
 else:  
 url = url.replace(f'&start={str(start + 1)}', f'&start={str(start + 50 + 1)}')  
 iteration += 1  
 start += 50  
 else:  
 break  
  
 return data # возвращаем список id фильмов  
# endregion  
  
  
# region Функция конвертации даты в сезон выхода и нагруженности на кинозалы  
def convert\_date(child\_id):  
 data = []  
 release\_date = [0, 0, 0]  
 date = None  
 qty\_most\_common = 0  
 month\_format = {'January': 1, # заменяем текстовые значения числовыми  
 'February': 2,  
 'March': 3,  
 'April': 4,  
 'May': 5,  
 'June': 6,  
 'July': 7,  
 'August': 8,  
 'September': 9,  
 'October': 10,  
 'November': 11,  
 'December': 12}  
 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')  
  
 dates\_list = soup.find('table', class\_='release-dates-table-test-only') \  
 .find\_all('tr', class\_='ipl-zebra-list\_\_item release-date-item')  
 for item in dates\_list:  
 data.append(item.find('td', class\_='release-date-item\_\_date').get\_text()) # получения списка дат  
  
 set\_data = set(data)  
  
 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:  
 release\_date[0] = 0  
 release\_date[1] = 0  
 return release\_date  
 year = date[2]  
  
 temp\_season = datetime.datetime(2020, int(month), int(day))  
 if datetime.datetime(2019, 11, 20) < temp\_season < datetime.datetime(2020, 2, 20):  
 release\_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):  
 release\_date[0] = 1 # аналогично для всех ниже  
 elif datetime.datetime(2020, 5, 20) < temp\_season < datetime.datetime(2020, 8, 20):  
 release\_date[0] = 2  
 elif datetime.datetime(2020, 8, 20) < temp\_season < datetime.datetime(2020, 11, 20):  
 release\_date[0] = 3  
 elif datetime.datetime(2020, 11, 20) < temp\_season < datetime.datetime(2021, 2, 20):  
 release\_date[0] = 4  
 else:  
 release\_date[0] = 0  
  
 # получаем день недели  
 release\_date[1] = datetime.datetime.strptime(f'{year}-{month}-{day}', '%Y-%m-%d').isoweekday()  
  
 if datetime.datetime(2019, 12, 15) < temp\_season < datetime.datetime(2020, 1, 15):  
 release\_date[2] = 1 # если вышел в зимние каникулы  
 elif datetime.datetime(2020, 12, 15) < temp\_season < datetime.datetime(2021, 1, 15):  
 release\_date[2] = 1 # если вышел в летние каникулы  
 elif datetime.datetime(2020, 6, 15) < temp\_season < datetime.datetime(2020, 9, 10):  
 release\_date[2] = 1  
 else:  
 release\_date[2] = 0  
  
 return release\_date  
  
  
# endregion  
  
  
# region Функция получения данных о съемочной группе  
def take\_person\_score(child\_id):  
 score, temp = 0, 0  
 total = [0, 0, 0]  
  
 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')  
  
 temp = soup.find('a', attrs={'id': 'meterRank'})  
 if temp is not None:  
 temp = temp.get\_text()  
 else:  
 return 0  
  
 awards = soup.find('div', class\_='article highlighted')  
 if awards is not None: # проверяем на пустоту  
 awards = awards.find\_all('span', class\_='awards-blurb')  
 for span\_a in awards:  
 if awards is not None:  
 awards = span\_a.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(' ', '')  
 .replace('Nominated', '')  
 .replace('for', '')  
 .replace('Oscar', '')  
 .replace('s', ''))  
 temp\_aw = int(temp\_aw) if str(temp\_aw).isdigit() else 1 # заменяем числом  
 if temp\_aw > 2:  
 total[0] += 1 # если номинация больше 2 считаем за получение 1 оскара  
 else:  
 total[0] += 0  
 else:  
 temp\_aw = (awards.replace(' ', '')  
 .replace('Won', '')  
 .replace('Oscar', '')  
 .replace('s', ''))  
 # считаем количество полученных оскаров  
 total[0] += int(temp\_aw) if str(temp\_aw).isdigit() else 1  
 else:  
 total[0] += 0 # иначе записываем на возврат ноль  
  
 if awards.find('wins') != -1:  
 total[1] = 1 # 1-если были полученные награды  
 else:  
 total[1] = 0 # 0-если полученных наград небыло  
  
 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 ретинг  
 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:  
 return 0  
  
 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  
 total[2] = round(score / num) # делим количество наград на количество человек из списка  
 return total  
# endregion  
  
  
# region Функция получения данных с википедии, если небыло на imdb  
def take\_box\_wiki(child\_id):  
 def is\_number(v): # проверяем цифра или нет  
 try:  
 float(v)  
 return True  
 except ValueError:  
 return False  
  
 def calculate(cal): # получаем цифру и меняем млн на 1000000 и тд  
 reg = re.findall(r'\[\d\*]', cal)  
 temp = (cal.replace('$', '') # замена символов  
 .replace('£', '')  
 .replace('€', '')  
 .replace('₩', '')  
 .replace('¥', '')  
 .replace('₹', '')  
 .replace('DEM', '')  
 .replace('FRF', '')  
 .replace('R', '')  
 .replace('U', '')  
 .replace(''.join(reg), '')  
 .replace(' долл.', '')  
 .replace(' долларов', '')  
 .replace(' (в США)', '')  
 .replace('.', '')  
 .replace(' ', ''))  
 temp = temp.replace(''.join(reg), '').replace('тыс', '\*тыс').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 выбераем 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  
  
 budget = [0, 0, 0]  
  
 try:  
 # вврдим 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 прогрузки 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')  
  
 url = str(search\_result)  
 session = requests.Session()  
 request = session.get(url, headers=headers)  
 soup = BeautifulSoup(request.content, 'lxml') # загружаем найденную страницу в более быстрый и bs4  
  
 temp\_budget = soup.find('span', attrs={'data-wikidata-property-id': 'P2130'})  
 if temp\_budget is not None:  
 temp\_budget = temp\_budget.get\_text()  
 budget[0] = calculate(temp\_budget) # получаем бюджет в числовом формате  
 else:  
 budget[0] = 0  
  
 temp\_box = soup.find('span', attrs={'data-wikidata-property-id': 'P2142'})  
 if temp\_box is not None:  
 temp\_box = temp\_box.get\_text()  
 budget[1] = calculate(temp\_box)  
 else:  
 budget[1] = 0  
  
 franchise1 = soup.find('span', attrs={'data-wikidata-property-id': 'P155'}) # узнаем есть ли предыдущие фильмы  
 if franchise1 is not None:  
 budget[2] = 1  
 else:  
 budget[2] = 0  
 except (NoSuchElementException, StaleElementReferenceException): # если вылетела ошибка возвращаем нули  
 budget[0] = 0  
 budget[1] = 0  
 budget[2] = 0  
 return budget  
 pass  
  
 return budget  
  
  
# endregion  
  
  
def parser(title\_id):  
 global checked\_counter  
 directors, writers, stars = [], [], []  
 directors\_awards, writers\_awards, stars\_awards = '', '', ''  
 year, mpaa, duration, genre, season, popularity, budget, boxoffice, franchise = '', '', '', '', '', '', '', '', ''  
 hour, minute, day, age\_limit, oscars, holiday = '', '', '', '', '', ''  
  
 agelimit\_format = {'PG-13': 3, 'NC-17': 5, 'PG': 2, 'R': 4, 'G': 1, 'X': 6,  
 '0': 1, '6': 2, '12': 3, '14': 4, '16': 5, '18': 6}  
 genre\_format = {'Action': 1, 'Adventure': 2, 'Drama': 3, 'Comedy': 4, 'Crime': 5,  
 'Romance': 6, 'Mystery': 7, 'Horror': 8, 'History': 9, 'Western': 10,  
 'Music': 11, 'Biography': 12, 'Musical': 13, 'Film-Noir': 14, 'Animation': 15,  
 'Fantasy': 16, 'Sci-Fi': 17, 'Thriller': 18, 'Family': 19, 'Short': 20,  
 'Sport': 21, 'War': 22, 'Reality-TV': 23, 'Game-Show': 24, 'Documentary': 25,  
 'Talk-Show': 26, 'News': 27, 'Adult': 28}  
  
 url = f'https://www.imdb.com/title/tt{str(title\_id)}/'  
 session = requests.Session()  
 request = session.get(url, headers=headers)  
 soup = BeautifulSoup(request.content, 'lxml')  
  
 boxoffice\_wiki = take\_box\_wiki(title\_id) # отправляем поиск фильма в вики  
  
 success = soup.find('li', class\_='ipc-inline-list\_\_item')  
 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  
  
 # region Длительность фильма  
 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  
 # endregion  
  
 # region Если получили возрастной рейтинг в формате MPAA конвертируем в рус вариант  
 while type(mpaa) not in (str, int):  
 mpaa = mpaa[0] if mpaa else 0  
 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 = 0  
 else:  
 age\_limit = 0  
 # endregion  
  
 # region Жанр фильма  
 genre = soup.find('div', attrs={'data-testid': 'genres'})  
 if genre is not None: # проверка наличия на странице  
 genre = genre.find('span', class\_='ipc-chip\_\_text')  
 if genre is not None:  
 genre = genre\_format[genre.get\_text()]  
 # endregion  
  
 # region Дата выхода фильма  
 release\_date = convert\_date(title\_id) # получаем список  
 season = release\_date[0] # 0-сезон выхода  
 day = release\_date[1] # 1-день выхода от 1 до 7  
 holiday = release\_date[2] # выход в каникулы да/нет  
 # endregion  
  
 # 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 = take\_person\_score(directors) # получаем наличие наград режиссеров  
 writers = take\_person\_score(writers) # получаем наличие наград сценаристов  
 stars = take\_person\_score(stars) # получаем наличие наград 3х звезд в главных ролях  
  
 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  
  
 # 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 = (budget.get\_text() # удаляем ненужные символы  
 .replace('$', '') # да, можно сделать через список  
 .replace('£', '') # иногда через список не работает и вызывает ошибку  
 .replace('€', '')  
 .replace('₩', '')  
 .replace('¥', '')  
 .replace('₹', '')  
 .replace('DEM', '')  
 .replace('FRF', '')  
 .replace('R', '')  
 .replace('U', '')  
 .replace(',', '')  
 .replace('.', '')  
 .replace('~', '')  
 .replace('\xa0', '')  
 .replace(' (estimated)', '')  
 .replace(' долл.', ''))  
 else:  
 budget = boxoffice\_wiki[0]  
 else:  
 budget = boxoffice\_wiki[0]  
 # все то же самое что и сверху но для кассовых сборов  
 boxoffice = boxoffice\_section.find('li', attrs={'data-testid': 'title-boxoffice-cumulativeworldwidegross'})  
 if boxoffice is not None:  
 boxoffice = boxoffice.find('span', class\_='ipc-metadata-list-item\_\_list-content-item')  
 if boxoffice is not None:  
 boxoffice = (boxoffice.get\_text()  
 .replace('$', '')  
 .replace('£', '')  
 .replace('€', '')  
 .replace('₩', '')  
 .replace('¥', '')  
 .replace('₹', '')  
 .replace('DEM', '')  
 .replace('FRF', '')  
 .replace('R', '')  
 .replace('U', '')  
 .replace(',', '')  
 .replace('.', '')  
 .replace('~', '')  
 .replace('\xa0', '')  
 .replace(' (estimated)', '')  
 .replace(' долл.', ''))  
 else:  
 boxoffice = boxoffice\_wiki[1]  
 else:  
 boxoffice = boxoffice\_wiki[1]  
 # endregion  
 else:  
 title\_id = 0 # если страницы нет или не фильм сохраняем id как пропуск значения  
  
 franchise = boxoffice\_wiki[2]  
  
 print(checked\_counter, url)  
 checked\_counter += 1  
  
 # region Переводим значения в int  
 year = int(year[0]) if year else 0  
 age\_limit = int(age\_limit) if age\_limit else 0  
 duration = int(duration) if duration else 0  
 season = int(season) if season else 0  
 day = int(day) if str(day).isdigit() else 0  
 directors = int(directors) if directors 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  
 popularity = int(popularity) if popularity else 5000  
 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) <= 40:  
 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:  
 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)))  
  
 if (budget \* 4) > boxoffice < (budget / 4):  
 title\_id = 0 # если кассовые сборы в 4 раза больше или меньше бюджета считаем ненормальными данными  
  
 if boxoffice < 10000:  
 title\_id = 0 # если кассовые сборы меньше 10 000 ставим id как пропуск  
  
 if boxoffice >= (budget \* 2):  
 profitable = 100 # если сборы в 2 раза больше то окупаемость полная  
 elif budget <= boxoffice <= (budget \* 2):  
 profitable = 50 # если сборы больше бюджета то окупаемость частичная  
 else:  
 profitable = 0 # иначе не окупился  
  
 # if (budget \* 4) > boxoffice < (budget / 4):  
 # title\_id = 0  
  
 # region Приравниваем значения  
 if boxoffice <= 500000:  
 boxoffice = 0  
 elif 500001 < boxoffice <= 1000000:  
 boxoffice = 10  
 elif 1000001 < boxoffice <= 2500000:  
 boxoffice = 20  
 elif 2500001 < boxoffice <= 5000000:  
 boxoffice = 30  
 elif 5000001 < boxoffice <= 7500000:  
 boxoffice = 40  
 elif 7500001 < boxoffice <= 10000000:  
 boxoffice = 50  
 elif 10000001 < boxoffice <= 25000000:  
 boxoffice = 60  
 elif 25000001 < boxoffice <= 50000000:  
 boxoffice = 70  
 elif 50000001 < boxoffice <= 75000000:  
 boxoffice = 80  
 elif 75000001 < boxoffice <= 100000000:  
 boxoffice = 90  
 elif 100000001 < boxoffice <= 11500000:  
 boxoffice = 100  
 else:  
 boxoffice = 100  
 # endregion  
  
 print(title\_id, title, url)  
 data = {'id': title\_id,  
 'name': title,  
 'url': url,  
 'year': year,  
 'age-limit': age\_limit,  
 'duration': duration,  
 'release-season': season,  
 'release-day': day,  
 'holiday': holiday,  
 'directors': directors,  
 'directors-awards': directors\_awards,  
 'writers': writers,  
 'writers-awards': writers\_awards,  
 'stars': stars,  
 'stars-awards': stars\_awards,  
 'oscars': oscars,  
 'genre': genre,  
 'franchise': franchise,  
 'imdb-popularity': popularity,  
 'budget': budget,  
 'box-office': boxoffice,  
 'profitable': profitable}  
 print(data)  
 return data  
  
  
def del\_copy(obj): # удаляем копии из списка  
 n = []  
 for i in obj:  
 if i not in n:  
 n.append(i)  
 return n  
  
  
# region Удаляем выбросы  
def del\_ejects(filename):  
 print(filename)  
 df = pandas.read\_csv(f'./dataset\_{filename}.txt', sep=';')  
  
 for x in ['D1']:  
 q75, q25 = numpy.percentile(df.loc[:, x], [75, 25])  
 intr\_qr = q75 - q25  
  
 max = q75 + (1.5 \* intr\_qr)  
 min = q25 - (1.5 \* intr\_qr)  
  
 df.loc[df[x] < min, x] = numpy.nan  
 df.loc[df[x] > max, x] = numpy.nan  
  
 for x in ['X11']:  
 q75, q25 = numpy.percentile(df.loc[:, x], [75, 25])  
 intr\_qr = q75 - q25  
  
 max = q75 + (1.5 \* intr\_qr)  
 min = q25 - (1.5 \* intr\_qr)  
  
 df.loc[df[x] < min, x] = numpy.nan  
 df.loc[df[x] > max, x] = numpy.nan  
  
 print(df.isnull().sum())  
 df.isnull().sum()  
 df = df.dropna(axis=0)  
  
 df.to\_csv(f'./dataset\_{filename}.txt', index=False, sep=';')  
# endregion  
  
  
def write\_file(dset, title, ver, state):  
 # region Записываем заголовки в зависимости от версии  
 sample\_frow = (f'Возрастное ограничение (1- 0+, 2- 6+ и тд);'  
 f'Длительность фильма в минутах;'  
 f'Сезон выхода (0-зима, 1-весна и тд);'  
 f'Вышел ли фильм в период высокой посещаемости кинотеатров (0-нет, 1-да);'  
 f'Имеют ли режиссеры награды (0-нет, 1-да);'  
 f'Имеют ли сценаристы награды (0-нет, 1-да);'  
 f'Имеют ли 3-х главные звезды награды (0-нет, 1-да);'  
 f'Количество оскаров у съемочной группы;'  
 f'Основной жанр фильма (1-Action, 2-Adventure, 3-Drama и тд);'  
 f'Является ли фильм частью франшизы;'  
 f'Бюджет фильма;')  
 if ver == 'D1':  
 header = f'X1;X2;X3;X4;X5;X6;X7;X8;X9;X10;X11;D1\n'  
 frow = (f'{sample\_frow}'  
 f'Кассовые сборы фильма\n')  
 elif ver == 'D2':  
 header = f'X1;X2;X3;X4;X5;X6;X7;X8;X9;X10;X11;D1\n'  
 frow = (f'{sample\_frow}'  
 f'Окупаемость фильма (0-не окупился, 50-собрал бюджет, 100-окупился)\n')  
 else:  
 header = f'X1;X2;X3;X4;X5;X6;X7;X8;X9;X10;X11;D1;D2\n'  
 frow = (f'{sample\_frow}'  
 f'Кассовые сборы фильма;'  
 f'Окупаемость фильма (0-не окупился, 50-собрал бюджет, 100-окупился)\n')  
 # endregion  
  
 if state == 1:  
 with open(f'./dataset\_sample.txt', 'w', encoding='utf-8') as f:  
 f.write(header)  
 f.write(frow)  
 for files\_item\_add in files:  
 with open(f'./dataset\_{files\_item\_add}.txt', 'w', encoding='utf-8') as f:  
 f.write(header)  
 with open(f'./dataset\_{files\_item\_add}\_test.txt', 'w', encoding='utf-8') as f:  
 f.write(header)  
  
 # region Записываем данные в зависимости от версии  
 if ver == 'D1':  
 with open(f'./dataset\_{title}.txt', 'a', encoding='utf-8') as f:  
 for span in dset:  
 f.write(f'{span["age-limit"]};{span["duration"]};'  
 f'{span["release-season"]};{span["holiday"]};'  
 f'{span["directors-awards"]};{span["writers-awards"]};{span["stars-awards"]};'  
 f'{span["oscars"]};{span["genre"]};{span["franchise"]};{span["budget"]};'  
 f'{span["box-office"]}\n')  
 elif ver == 'D2':  
 with open(f'./dataset\_{title}.txt', 'a', encoding='utf-8') as f:  
 for span in dset:  
 f.write(f'{span["age-limit"]};{span["duration"]};'  
 f'{span["release-season"]};{span["holiday"]};'  
 f'{span["directors-awards"]};{span["writers-awards"]};{span["stars-awards"]};'  
 f'{span["oscars"]};{span["genre"]};{span["franchise"]};{span["budget"]};'  
 f'{span["profitable"]}\n')  
 else:  
 with open(f'./dataset\_{title}.txt', 'a', encoding='utf-8') as f:  
 for span in dset:  
 f.write(f'{span["age-limit"]};{span["duration"]};'  
 f'{span["release-season"]};{span["holiday"]};'  
 f'{span["directors-awards"]};{span["writers-awards"]};{span["stars-awards"]};'  
 f'{span["oscars"]};{span["genre"]};{span["franchise"]};{span["budget"]};'  
 f'{span["box-office"]};'  
 f'{span["profitable"]}\n')  
 # endregion  
  
 if state == 2:  
 # region Разделяем данные на обучающие и тестирующие  
 for ctr in files:  
 lrow\_counter = 0  
 temp\_dataset, dataset\_test = [], []  
 with open(f'./dataset\_{ctr}.txt', 'r', encoding='utf-8') as lfile:  
 for lrow in lfile:  
 if lrow\_counter != 0:  
 if lrow\_counter % 8 == 0:  
 dataset\_test.append(lrow)  
 else:  
 temp\_dataset.append(lrow)  
 lrow\_counter += 1  
 lfile.close()  
 with open(f'./dataset\_{ctr}\_test.txt', 'a', encoding='utf-8') as mfile:  
 for mrow in dataset\_test:  
 mfile.write(mrow)  
 open(f'./dataset\_{ctr}.txt', 'w').close()  
 with open(f'./dataset\_{ctr}.txt', 'a', encoding='utf-8') as nfile:  
 nfile.write(header)  
 for orow in temp\_dataset:  
 nfile.write(orow)  
 mfile.close()  
 nfile.close()  
 f.close()  
 # endregion  
  
 for repeat in range(5): # 5 раз удаляем выбросы  
 for item\_files in files:  
 del\_ejects(f'{item\_files}')  
 del\_ejects(f'{item\_files}\_test')  
  
 # region Делим созданные фалы на группы и записываем в excel  
 group0 = pandas.read\_csv(f'./dataset\_sample.txt', sep=';')  
 group1 = pandas.read\_csv(f'./dataset\_all.txt', sep=';')  
 group1\_test = pandas.read\_csv(f'./dataset\_all\_test.txt', sep=';')  
 group2 = pandas.read\_csv(f'./dataset\_0-10.txt', sep=';')  
 group2\_test = pandas.read\_csv(f'./dataset\_0-10\_test.txt', sep=';')  
 group3 = pandas.read\_csv(f'./dataset\_10-40.txt', sep=';')  
 group3\_test = pandas.read\_csv(f'./dataset\_10-40\_test.txt', sep=';')  
 group4 = pandas.read\_csv(f'./dataset\_40-inf.txt', sep=';')  
 group4\_test = pandas.read\_csv(f'./dataset\_40-inf\_test.txt', sep=';')  
  
 # Подготавливаем листы к записи  
 sheets = {'SAMPLE': group0,  
 'DATA': group1, 'TEST': group1\_test,  
 'DATA\_0-10': group2, 'TEST\_0-10': group2\_test,  
 'DATA\_10-40': group3, 'TEST\_10-40': group3\_test,  
 'DATA\_40-inf': group4, 'TEST\_40-inf': group4\_test}  
  
 writer = pandas.ExcelWriter(f'./dataset{ver}.xlsx', engine='openpyxl')  
  
 # Записываем каждый документ в отдельный лист  
 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  
 os.remove('./dataset\_sample.txt')  
 for files\_item\_del in files:  
 if os.path.exists(f'./dataset\_{files\_item\_del}.txt'):  
 os.remove(f'./dataset\_{files\_item\_del}.txt')  
 if os.path.exists(f'./dataset\_{files\_item\_del}\_test.txt'):  
 os.remove(f'./dataset\_{files\_item\_del}\_test.txt')  
 # endregion  
  
  
# region Создаем файл с id фильмов  
def first\_part():  
 raw\_dataset.append(scrapper(top\_1000))  
 raw\_dataset.append(scrapper(bottom\_1000))  
  
 for k in raw\_dataset[0]:  
 ids.append(k) # записываем id топ 1000 лучших фильмов  
  
 for k in raw\_dataset[1]:  
 ids.append(k) # записываем id топ 1000 худших фильмов  
  
 print(ids)  
 with open(f'./ids.txt', 'w+', encoding='utf-8') as id\_file:  
 for item in ids:  
 id\_file.write(item + '\n') # записываем в файл  
# endregion  
  
  
def second\_part():  
 counter = 0 # счетчик с какой строки начинать парсить  
 # очистка предыдущего парсинга если надо  
 open(f'./tempdata.txt', 'w').close()  
  
 # 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')  
  
 while counter <= 2000:  
 output\_dataset, temp\_dataset, temp\_ids = [], [], []  
 print(1, counter)  
  
 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 после парсинга 0 то пропускаем  
 pass  
 else:  
 output\_dataset.append(temp\_row) # иначе записываем в dataset  
  
 with open(f'./tempdata.txt', 'a', encoding='utf-8') as temp\_data:  
 for subrow in output\_dataset:  
 temp\_data.write(f'{subrow}\n')  
  
 counter += 100  
  
 temp\_file.close()  
 temp\_data.close()  
  
  
def third\_part():  
 versions = ['D1', 'D2', '']  
 dataset = []  
  
 with open(f'./tempdata.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  
  
 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"]};'  
 f'{jrow["name"]};{jrow["year"]};{jrow["duration"]};'  
 f'{jrow["imdb-popularity"]};{jrow["budget"]};{jrow["box-office"]}\n')  
  
 dataset0\_10, dataset10\_40, dataset40\_inf = [], [], []  
  
 # сортируем данные по бюджету  
 for krow in dataset:  
 if krow["budget"] < 10000000:  
 dataset0\_10.append(krow)  
 if 10000001 < krow["budget"] < 40000000:  
 dataset10\_40.append(krow)  
 if 40000001 < krow["budget"]:  
 dataset40\_inf.append(krow)  
  
 # region Отправляем данные на запись  
 write\_file(dataset, 'sample', 'D1', 1)  
 write\_file(dataset, 'all', 'D1', 0)  
 write\_file(dataset0\_10, '0-10', 'D1', 0)  
 write\_file(dataset10\_40, '10-40', 'D1', 0)  
 write\_file(dataset40\_inf, '40-inf', 'D1', 2)  
  
 write\_file(dataset, 'sample', 'D2', 1)  
 write\_file(dataset, 'all', 'D2', 0)  
 write\_file(dataset0\_10, '0-10', 'D2', 0)  
 write\_file(dataset10\_40, '10-40', 'D2', 0)  
 write\_file(dataset40\_inf, '40-inf', 'D2', 2)  
  
 write\_file(dataset, 'sample', '', 1)  
 write\_file(dataset, 'all', '', 0)  
 write\_file(dataset0\_10, '0-10', '', 0)  
 write\_file(dataset10\_40, '10-40', '', 0)  
 write\_file(dataset40\_inf, '40-inf', '', 2)  
 # endregion  
  
 temp\_fdata.close()  
 j\_file.close()  
  
 df0 = pandas.read\_csv('./raw\_data.txt', sep=';')  
 df0.to\_excel('./raw\_data.xlsx', 'DATA', engine='openpyxl', index=False)  
  
 os.remove('./raw\_data.txt')  
  
  
# region Функция для тестирования парсера на одной или нескольких страниц  
def test():  
 # for item\_id in ('0468569'): #ids:'10366460',  
 test\_set = parser('0468569')  
 print(f'{test\_set["age-limit"]};{test\_set["duration"]};'  
 f'{test\_set["release-season"]};{test\_set["release-day"]};{test\_set["holiday"]};'  
 f'{test\_set["directors"]};{test\_set["writers"]};{test\_set["stars"]};'  
 f'{test\_set["directors-awards"]};{test\_set["writers-awards"]};{test\_set["stars-awards"]};'  
 f'{test\_set["oscars"]};'  
 f'{test\_set["genre"]};{test\_set["franchise"]};{test\_set["imdb-popularity"]};'  
 f'{test\_set["budget"]};{test\_set["profitable"]}\n')  
# endregion  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 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  
  
 raw\_dataset = []  
 ids = []  
  
 # test()  
 # first\_part()  
 # second\_part()  
 third\_part()

ПРИЛОЖЕНИЕ Г   
Листинг 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)

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)

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)}')