



Московский государственный университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра суперкомпьютеров и квантовой информатики

Вартанов Дмитрий Александрович

**Предсказание оценки игры
по её описанию
в игровом магазине Steam**

Работа по дисциплине "прикладное машинное обучение"

Москва, 2022

Содержание

1 Введение	4
2 Информация об исходных данных и их сборе	6
2.1 Откуда берутся исходные данные?	6
2.2 Какие данные собираются из страницы самой игры?	6
2.3 Какие игры считаются хорошими?	10
2.4 Отбор объектов из генеральной совокупности	11
2.4.1 Про качество данных	12
2.4.2 Про репрезентативность	13
2.5 Хранение данных	14
3 Формальная постановка задачи	16
3.1 Постановка задачи	16
3.2 Показатели качества	16
4 Описание предобработки данных	17
4.1 Приведение к структуированному виду	17
4.1.1 TF-IDF для текстов и их предобработка	18
4.1.2 Обработка картинок	19
4.2 Описание систем решения	20
4.2.1 Аугментация данных	20
4.2.2 Жесткие и мягкие метки	20
4.2.3 Режимы обучения	20
4.2.4 Итоговые системы решения	20
4.3 Итоговый вектор признаков	21
4.4 Подробные итоговые архитектуры	22
4.5 Интерпретируемость признаков	23
4.5.1 Все признаки	23
4.5.2 Отдельные признаки	23
4.5.3 Микровывод	25

5 Сравнение систем и окончательные выводы	26
5.1 Время обучения и работы алгоритмов	26
5.2 Качество распознавания	26
6 Архитектура моделей в подробностях и цифрах	28
6.1 CNN	28
6.2 MLP	28
6.3 KNN	28
6.4 CatBoost	28
6.5 ContrastiveLoss	29
7 Заключение	30

1 Введение

Представим, что мы являемся работниками маленькой инди-студией по производству игр *HuratanoGames*. Основной жанр выпускаемых игр для нас - *Action – roguelike* (по-русски будем называть далее "экшен-рогалик").

И вот была разработана игра, которую планируется выложить на специальную игровую платформу *Steam*, где покупатели смогут ознакомиться с новым шедевром игровой индустрии. Однако недостаточно просто сделать хороший продукт, но необходимо и подать его максимально качественно. С этой целью было предложено понять, как страница с описанием игры влияет на оценку игры.

Таким образом можно задать следующую **основную цель: по описанию страницы предсказать, получит ли игра хорошие отзывы после выпуска**. В случае неуспеха бизнес рискует заработать плохую репутацию и меньшие продажи.

Итак, сформулируем набор пунктов:

Что предлагается решить: задачу предсказания игрового отзыва на платформе Steam на основе оформления её страницы на торговой площадке.

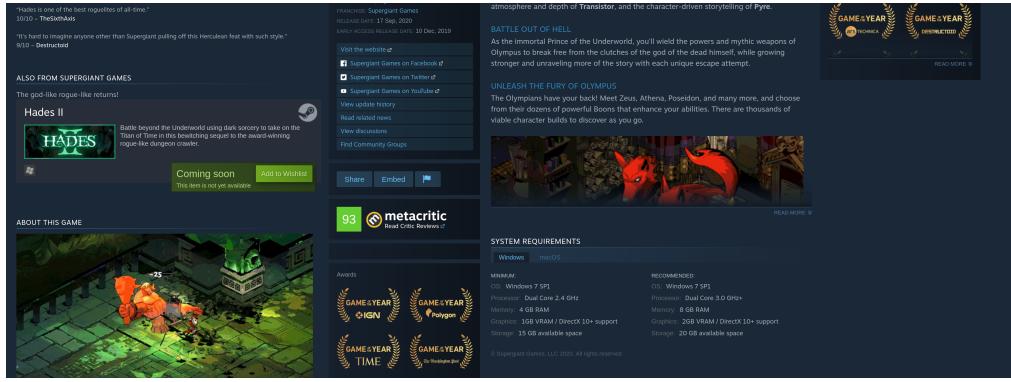
Вход: возможное заполнение полей страницы игры.

Выход: предсказание того, будет ли игра высоко оценена.

Цель: получить классификатор, позволяющий как можно точнее оценить качество предложенного оформления страницы.



Рис. 1: Пример страницы игры Hades



(a)

(b)

Рис. 2: Пример страницы игры Hades

2 Информация об исходных данных и их сборе

2.1 Откуда берутся исходные данные?

Исходные данные берутся из напрямую из магазина Steam. Для этого при помощи языка программирования Python происходит следующая последовательность обращений:

1. Обращение к странице Steam, содержащей список игр жанра "экшен-рогалик". На главной странице жанра содержится список игр, каждая из которых подкреплена ссылкой на её страницу в магазине;
2. По каждой ссылки из п.1 происходит переход на страницу игры, где и собираются основные данные.

Ссылка на Steam: <https://store.steampowered.com/?l=russian>

Ссылки на пример страницы с игрой: <https://store.steampowered.com/app/1145360/Hades/>

2.2 Какие данные собираются из страницы самой игры?

Посмотрим еще раз на страницу игры (Рис.3, вся страница, кроме куска с обзорами, который нам, очевидно не нужен).

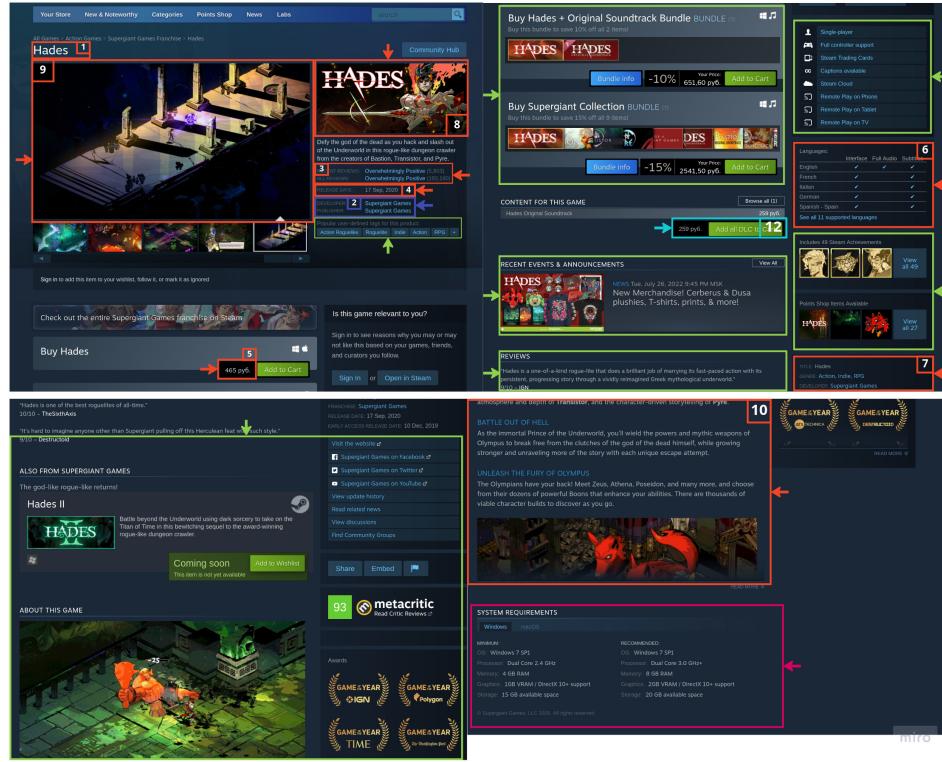


Рис. 3: Страница игры с разбором полей

Поля страницы размечены цветами, которые означают следующее:

- **Красный.** Данные страницы, которые можно использовать для предсказаний. Полный их список будет приведен ниже;
- **Розовый и темно-синий.** Данные, которые можно было бы использовать для улучшения предсказаний, но крайне затруднительно. К ним относятся изображения с моментами из игры, системные требования и название издателя. Конечно, попытаться связать отзывы на игру с системными требованиями и разградировать множество конфигураций устройств на мощные, средние, слабые и так далее. Или же пытаться искать информацию о разработчиках игры по сайтам и на этом составлять какие-то дополнительные признаки. Но это всё требует огромных дополнительных усилий (которые далеко не обязательно выльются во что-то хорошее), поэтому в рамках данной работы было решено не трогать эти поля;
- **Зеленый.** Поля, которые не могут появиться у игры во время ее выхода и в ближайшее время после него. К ним относятся, например, доступные DLC, отзывы

критиков, жанры, проставленные самими игроками и другие;

- **Голубой.** Вспомогательные поля, которые помогут отсеять некорректные примеры. К ним относится, например, поле-флажок того, является ли данная игра на самом деле лишь DLC.

Поля, которые собираются со страницы (нумерация соответствует Рис.3):

1. **Название игры.** Формат: строка;
2. **Разработчики игры.** Формат: список строк с названиями студий;
3. **Отзывы игроков.** Основная величина, которую мы и хотим предсказать. Steam предоставляет её в трех различных вариантах:
 - (a) В виде строки (например, *[Mixed, Positive]*). Такой формат больше предназначен для красивого оформления торговой площадки;
 - (b) В виде рейтинга Steam. Рейтинг Steam представляет собой грубую агрегацию отзывов игроков и представляет собой целое число от 0 до 10;
 - (c) В виде процента положительных отзывов. Такой является наиболее удобным и используется далее. Также далее под отзывами игроков по умолчанию подразумевать отношение количества положительных отзывов к количеству всех отзывов.
4. **Дата выхода.** Дата выхода игры в строковом формате. Важно понимать, что для некоторых игр страницы заводятся заранее. На момент исследования диапазон дат релиза от начала 2005го года до начала 2024го года. Формат: дата в виде строки;
5. **Цена игры.** Логично, что человек, сыгравший в посредственную игру будет куда более недоволен ей, если стоимость сильно превысила ожидания. Цены представляют собой целые неотрицательные числа;
6. **Доступные языки.** Пользователи могут быть недовольны, если игра не переведена на их родной язык. Языки считаются в список доступных для игры языков в виде строк;

7. **Список жанров от разработчиков.** Проставленные разработчиком жанры игры. Считываются аналогично языкам;
8. **Список тегов от разработчиков.** Проставленные разработчиком жанры игры. Считываются аналогично языкам;
9. **Картинка-баннер игры.** Основной баннер игры, считывается в jpg изображения;
10. **Картинка с геймплеем игры.** Первое изображение из вкладки со скриншотами геймплея;
11. **Описание игры.** Слабоструктурированная часть, содержащая в себе основную информацию об игре. Формат: строка;
12. **Поле DLC.** Если страница посвящена DLC, указанное поле будет иметь специальную пометку.

Все поля строкового формата содержат только английские слова.

Название поля	Интерпретация	Тип
score_review	оценка игры пользователями (целевой)	<i>int</i> [0, 100]
steam_rating	оценка игры пользователями (метрика steam)	<i>int</i> [0, 10]
vote_number	число оценок	<i>int</i> [0, + inf]
name	название игры	<i>string</i>
developers	список разработчиков	<i>list</i> [<i>string</i>]
release_date	дата выхода	<i>string</i>
price	цена	<i>int</i> [0, + inf]
languages	список языков	<i>list</i> [<i>string</i>]
dev_genres	жанры от разработчиков	<i>list</i> [<i>string</i>]
dev_tags	теги от разработчиков	<i>list</i> [<i>string</i>]
image_banner	изображение-баннер игры	<i>RGB</i> , 460 × 215
first_screen	скриншот игры	<i>RGB</i> , 600 × 338
description	описание игры	<i>string</i>
is_dlc	является ли игры DLC	<i>bool</i>

Таблица 1: Описание полей.

2.3 Какие игры считаются хорошими?

Выше было определено, что отзывы игроков считаются в процентном соотношении.

Для игры будем требовать не менее 85% положительных отзывов. Такое число было взято на основе построения гистограммы отзывов игр и нахождения медианы.

Медиана отзывов равна 84%. Так как мы хотим, чтобы наш продукт был успешен и запомнился потребителям особо сильно, логично запросить минимальную оценку "выше среднего". Более строгие требования ставить не совсем целесообразно ввиду того, что мы являемся маленькой инди-компанией и объективно не можем довести все аспекты игры до уровня крупных компаний разработки.

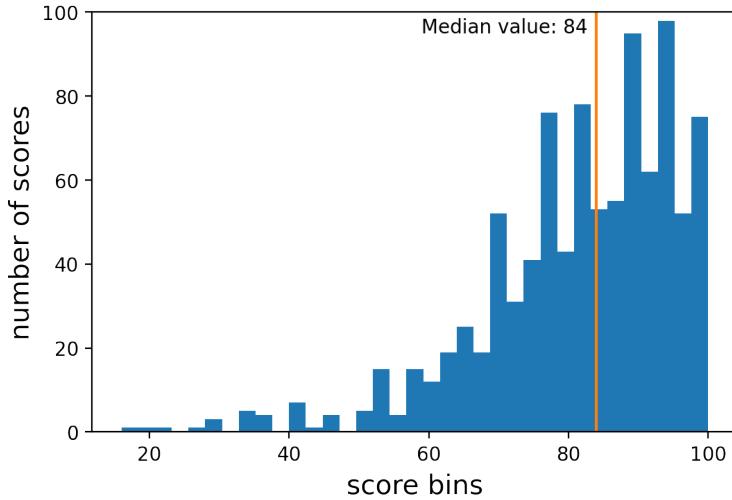


Рис. 4: Гистограмма отзывов игроков

2.4 Отбор объектов из генеральной совокупности

Еще раз отметим, что планируется выпускать игры в жанре "экшен-рогалик" на платформе Steam. Поскольку сбор данных происходил на основе Steam (см. 2.1) и отбирались только игры нужного жанра, то можно сказать, что все существующие на данный момент объекты были учтены при самой первоначальной обработке (парсинге страниц Steam). Всего удалось получить информацию о 1737 играх (т.е. о всех играх нужного жанра в рамках изучаемой платформы). Из них:

- Все игры (кроме 1-2 выбросов) имеют имя (name), доступные языки (languages), разработчиков (developers), пометку о DLC (is_dlc), жанры разработчиков (dev_genres), описание (description), теги разработчиков (dev_tags);
- Все игры (кроме примерно 10 выбросов) имеют дату релиза (release_date). Все игры с будущими релизами не имеют отзывов, поэтому автоматически отсеиваются;
- Всего 1433 игры имеет информацию о количестве голосов (vote_number) и оценки Steam (steam_rating от 0 до 10) и цену (price);
- 953 игры имеют отзыв пользователей (score_review). Из них 488 с оценкой $\geq 85\%$, 465 с оценкой $< 85\%$;

Название поля	name	languages	developers	is_dlc	dev_tags	dev_genres	description
Количество	1737	1734	1735	1737	1737	1734	1737

Таблица 2: Сколько раз поля были заполнены.

Название поля	release_date	steam_rating	price	vote_number	score_review
Количество	1735	1440	1542	1440	953

Таблица 3: Сколько раз поля были заполнены.

Название поля	name	languages	developers	is_dlc	dev_tags	dev_genres	description
Количество	1737	1734	1735	1737	1737	1734	1737

Таблица 4: Сколько раз поля были заполнены.

Оценка ниже 85%	Оценка выше 85%	score_review
465	488	953

Таблица 5: Разбиение примеров по целевому признаку

2.4.1 Про качество данных

Данные содержат в себе всю информацию о "экшен-рогаликах" представленных в Steam. Много игр не имеет голосов и оценок. Это связано с некоторыми факторами:

1. 162 игры еще не были выпущены -> и отзывов у них нет;
2. остальные игры, не имеющие голоса, просто мало оценены, из-за чего оценка составлена быть не могла;

В случае цены ситуация куда более вариативная и связана как с банальным непростоявлением цен на игру издателем, так и невозможностью купить игру для конкретного

региона.

Но разница между количеством score_review и steam_rating колоссальна, почему тогда не использовать второй показатель в качестве основной метки? На самом деле это связано с механикой проставления оценок Steam: если имеется несколько отзывов на игру, то steam_rating автоматически проставляется равным 0 и остается таким до 50 голосов от игроков. После этого он высчитывается как $np.round(score_review/10)$. Из-за этого steam_rating и аналогичен score_review.

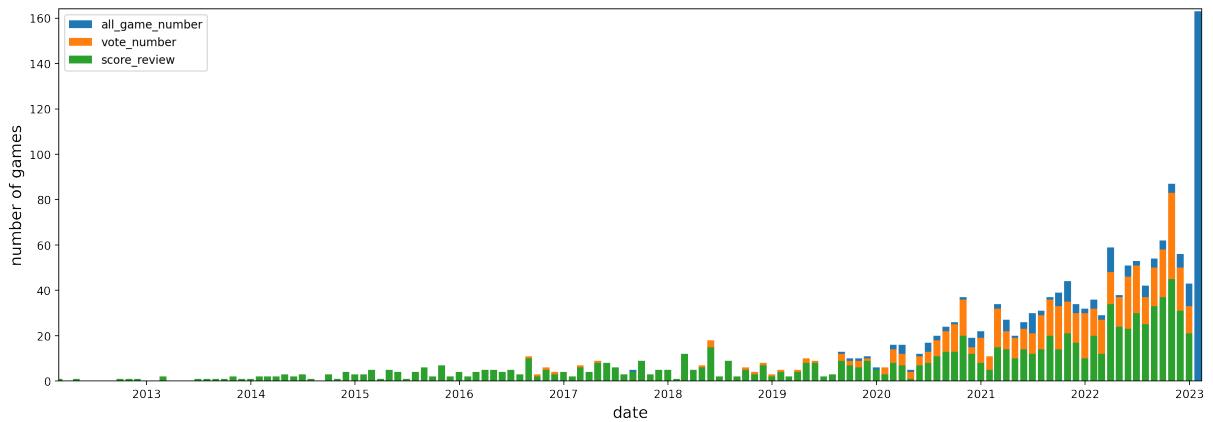


Рис. 5: Гистограмма отзывов игроков (обрезана до 2012 года)

Все значения являются правдивыми и не выходят за пределы ожидаемых значений. В итоге можно сказать, что только 928 примеров из 1737 обладают всеми необходимыми свойствами (имеют все заполненные поля и не являются DLC).

2.4.2 Про репрезентативность

За генеральную совокупность будем принимать непосредственно множество всех игр жанра "экшен-рогалик" только на платформе Steam по следующим причинам:

1. Steam является крупнейшей игровой торговой площадкой. Особенно, если речь идет о маленьких проектах. Существует множество других онлайн-магазинов, но они, либо принадлежат большими компаниями (Ubisoft, Origin, ...) и продают свои немногочисленные проекты, которые также представлены в Steam, либо не предлагают список уникальных игр, которые не были бы представлены в Steam;

2. Игра планируется для запуска в Steam. Сообщество, которому предстоит оценивать нашу игру, до этого оценивало игры именно из Steam;
3. Большое разнообразие. Только игр жанра "экшен-рогалик" в Steam насчитывается 1735. Это колоссальное число по меркам игровой индустрии и покрывает самые различные условия создания и последующей оценки (хорошо спонсированные и не очень, знаменитые разработчики или нет, дорогие или дешевые, большое число отзывов и маленькое, смешанные по жанру или нет и т.д.)

Именно из-за этого и можно назвать нашу выборку репрезентативной. Она вбирает в себя максимальное количество примеров, содержит самые разные игры в зависимости от исследуемых показателей, обладает чистыми и понятными данными, которые отражают текущую ситуацию на рынке.

2.5 Хранение данных

После парсинга данные складываются в *JSON* файлы вида Рис. 6. Пустые поля никак не заполняются, их отсутствие обрабатывается в коде по ходу считывания файлов.

```
{
  "url": "https://store.steampowered.com/app/448810/Torgars_Quest/?snr=1_7_7_240_150_19",
  "name": "Torgar's Quest",
  "score_review": "75",
  "vote_number": "12",
  "steam_rating": "7",
  "user_tags": [
    "Action Roguelike",
    "Traditional Roguelike",
    "Turn-Based Tactics",
    "Hack and Slash",
    "Mystery Dungeon",
    "Dungeon Crawler",
    "Roguelite",
    "Perma Death",
    "Turn-Based",
    "Action-Adventure",
    "Roguelike",
    "RPG",
    "2D",
    "Top-Down",
    "Lore-Rich",
    "Underground",
    "Procedural Generation",
    "Turn-Based Combat",
    "Action",
    "Casual"
  ],
  "developers": [
    "Tagunda LLC"
  ],
  "is_dlc": false,
  "release_date": "31 May, 2016",
  "dev_tags": [
    "Single-player",
    "Steam Achievements",
    "Full controller support",
    "Steam Leaderboards"
  ],
  "dev_genres": [
    "Action",
    "Adventure",
    "Indie",
    "RPG"
  ],
  "languages": [
    "English"
  ]
}
```

Рис. 6: Пример JSON файла

Изображения image_banner, first_screen хранятся в отдельных папках в виде JPG файлов.

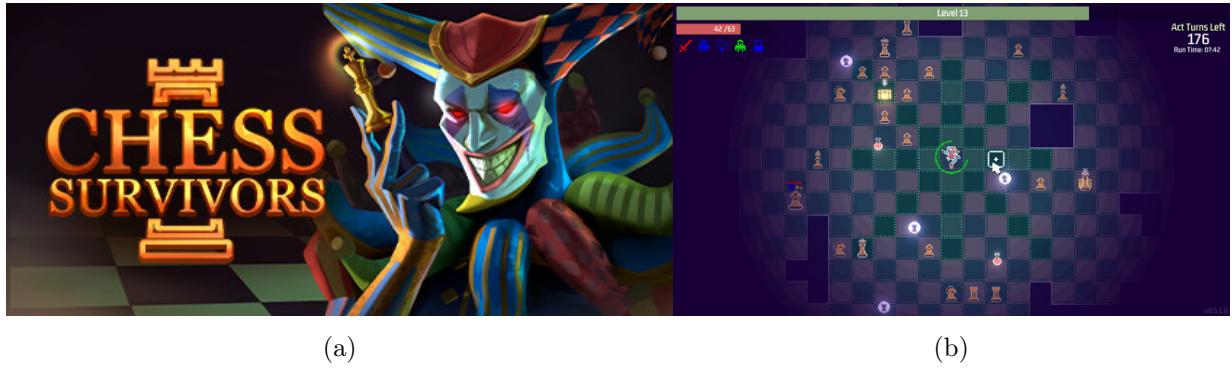


Рис. 7: Пример баннера и скриншота игры

3 Формальная постановка задачи

3.1 Постановка задачи

На вход подается вектор признаков, содержащий описание заполняемых на странице полей X^n .

На выходе необходимо получить метку класса $C \in \{0, 1\}$, где:

- 0 - игра будет иметь оценку ниже 85%;
- 1 - игра будет иметь оценку не ниже 85%.

3.2 Показатели качества

Так как исходная выборка сбалансирована в силу специфики определения задачи и решается задачи классификации, то целесообразно взять классическую метрику классификации, а именно *Accuracy*. Определяется как количество верно предсказанных объектов к количеству всех.

4 Описание предобработки данных

4.1 Приведение к структурированному виду

Посмотрим еще раз на форматы данных:

Название поля	Интерпретация	Тип
score_review	оценка игры пользователями (целевой)	$int[0, 100]$
steam_rating	оценка игры пользователями (метрика steam)	$int[0, 10]$
vote_number	число оценок	$int[0, +\infty]$
name	название игры	$string$
developers	список разработчиков	$list[string]$
release_date	дата выхода	$string$
price	цена	$int[0, +\infty]$
languages	список языков	$list[string]$
dev_genres	жанры от разработчиков	$list[string]$
dev_tags	теги от разработчиков	$list[string]$
image_banner	изображение-баннер игры	$RGB, 460 \times 215$
first_screen	скриншот игры	$RGB, 600 \times 338$
description	описание игры	$string$
is_dlc	является ли игры DLC	$bool$

Таблица 6: Описание полей.

Поскольку первые три поля из таблицы нам недоступны (являются целевыми или связаны с ними напрямую), то обработать необходимо только все остальные + score_review.

1. score_review. Преобразуем в булевское значение (метку класса 0 или 1) в зависимости от преодоления порога в 85%;
2. name. Анализируем на основе TF-IDF векторизатора (возьмем 20 признаков);
3. release_date. Переводим исходное строковое представление в $unsigned int$ значение (в виде количества дней с 01.01.2005). Дата выпуска игры может влиять на

настроения игроков;

4. price. Оставляем в исходном формате;
5. languages. Формируем кодирование языков при помощи массива из 0 и 1. Длина массива будет соответствовать ;
6. dev_genres. Аналогично languages, только по жанрам;
7. dev_tags. Аналогично languages, только по тегам;
8. description. Применим TF-IDF векторизатор для создания вектора признаков текста (возьмем 300 признаков);
9. is_dlc. В предобратке используется лишь для того, чтобы убрать DLC страницы;
10. image_banner. Предобрабатываем с помощью нейронной сети, выдающей набор признаков;
11. first_screenshot. Предобрабатываем с помощью нейронной сети, выдающей набор признаков;
12. developers. Не будем использовать, так как разработчик является сложной меткой, выходящей за пределы данной работы (связь с другими играми и оценка популярности компании).

4.1.1 TF-IDF для текстов и их предобработка

Сама по себе аббревиатура TF-IDF расшифровывается как TF — term frequency, IDF — inverse document frequency, то есть отношение частоты употребления слова в отдельном тексте к частоте употребления слова во всех документах. Построенная на основе такой меры модель прекрасно подходит для поиска похожих текстов, поскольку позволяет сравнивать совокупные меры текстов между собой, строя матрицу похожести. Source <https://habr.com/ru/post/542048/>

В данном случае TF-IDF является одновременно простым и мощным методом анализа текстов. Он, конечно, не дотягивает до современных решений по анализу текста, но и лучше самых примитивных вариантов по типу обычновенного мешка слов. Более того,

нам очень важно свойство idf, так как у одного и того же жанра игра наверняка имеется огромное количество тематических однотипных слов, которые на самом деле не воспринимаются пользователем как что-то особое.

Перед использованием TF-IDF принемим к текстам описания игры очистку от "шума". К нему относится пунктуация, а также "слова-шум" (например, We, manage, are, am ...). В итоге для TF-IDF текст уже будет очищен, что позволит получить более хорошее качество. Такая обработка основана на библиотеке *nltk*, позволяющей определять "шумовые" слова английского языка.

TF-IDF реализация была позаимствована из *sklearn*. Все параметры были оставлены по умолчанию, кроме количества выдаваемых признаков: 300 для описания и 20 для оглавлений. Такие значения были подобраны экспериментальным путем.

4.1.2 Обработка картинок

Обрабатывать изображения будем при помощи классических простейших сверточных нейронных сетей. Они отлично показывают себя в задачах классификации изображений и являются относительно простым методом анализа картинок.

Построим нейронную сеть с двумя входами (один для баннера, а второй для скриншота игры), которая будет выдавать уверенность в том, что игра будет оценена хорошо. Далее эти признаки будем использовать для окончательного решения.

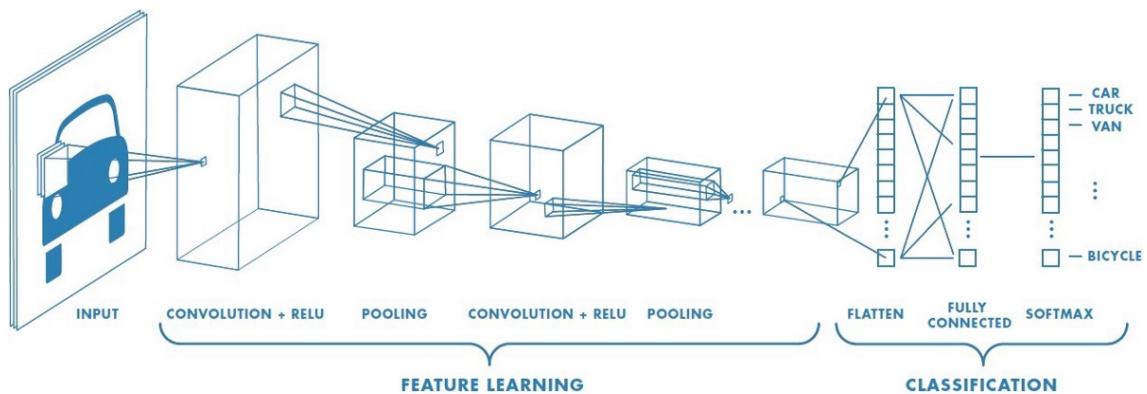


Рис. 8: Пример сверточной нейронной сети

Реализация всех нейронных сетей проводится на *PyTorch*.

4.2 Описание систем решения

Всего наличие аугментации, жесткие/мягкие метки и режимы обучения дают нам 8 систем, которые можно применить в данной задаче.

4.2.1 Аугментация данных

Аугментировать будем слабоструктурированную часть - описание игры. В качестве аугментации будем изменять очередное слово в описании игры на его синоним с вероятностью $p = 0.2$. Такая вероятность была выбрана, исходя из визуальной оценки различия измененных текстов и оригинальных. К сожалению, более точного способа определения параметра p найти не удалось, так как описание игр специфично и содержит много уникальных слов.

Аугментации проводятся при помощи библиотеки *nltk*, содержащей словари синонимов.

4.2.2 Жесткие и мягкие метки

С жесткими метками все понятно: 0, если оценка $< 85\%$, и 1 иначе.

В случае с мягкими метками можно подойти по-разному. В текущей работе будем использовать следующую стратегию:

$$y_{soft} = \begin{cases} 0 & \text{если оценка } < 70\%, \\ \frac{1}{2} + \frac{y_{init}-85}{30} & \text{иначе} \end{cases}$$

Такое решение связано с желанием сбалансировать вклад близких к 85% игр. Совсем плохие игры (с оценкой $< 70\%$ так и так должны отсеиваться с большой уверенностью).

4.2.3 Режимы обучения

Будем использовать в данной работе обучение через *CrossEntropyLoss*, а также *ContrastiveLoss*.

4.2.4 Итоговые системы решения

Итоговые системы решения следующие:

1. *rossentropyLoss* в сочетании с *CatBoost*. Мягкие/жесткие метки и аугментации данных;

2. *ContrastiveLoss* в сочетании с *MLP* (многослойный персепtron) и *KNN* (алгоритм К ближайших соседей). Мягкие/жесткие метки и аугментации данных. В основе *ContrastiveLoss* берем косинусную схожесть.

Стоит отметить, что озвученные системы решения оказались лучшими среди остальных опробованных по метрикам качества. Поэтому и были выбраны именно они для представления в отчёте.

4.3 Итоговый вектор признаков

В итоге после применения всех преобразований получаем для каждой страницы вектора признаков длины 473. Первый этап предобработки существенно уменьшил количество исходных признаков. Признаки нового "высокогоуровневого"вектора можно описать так:

- Цена игры (1 признак);
- Дата выхода игры (переведенная в число дней с 01.01.2005, 1 признак);
- Количество слов в оглавлении игры (1 признак);
- Количество слов в описании игры (1 признак);
- Количество слов в описании игры после удаление шумовых слов (1 признак);
- Признаки, построенные CNN, для изображений с сайта (2 признака);
- Жанры в виде one-hot кодирования (всего 12 признаков);
- Теги в виде one-hot кодирования (всего 31 признак);
- Языки в виде one-hot кодирования (всего 103 признака);
- Оглавление игры, преобразованное при помощи TF-IDF (20 признаков);
- Описание игры, преобразованное при помощи TF-IDF (300 признаков)

4.4 Подробные итоговые архитектуры

На схеме ниже представлены итоговые системы решения и весь процесс от получения данных до их полной обработки.

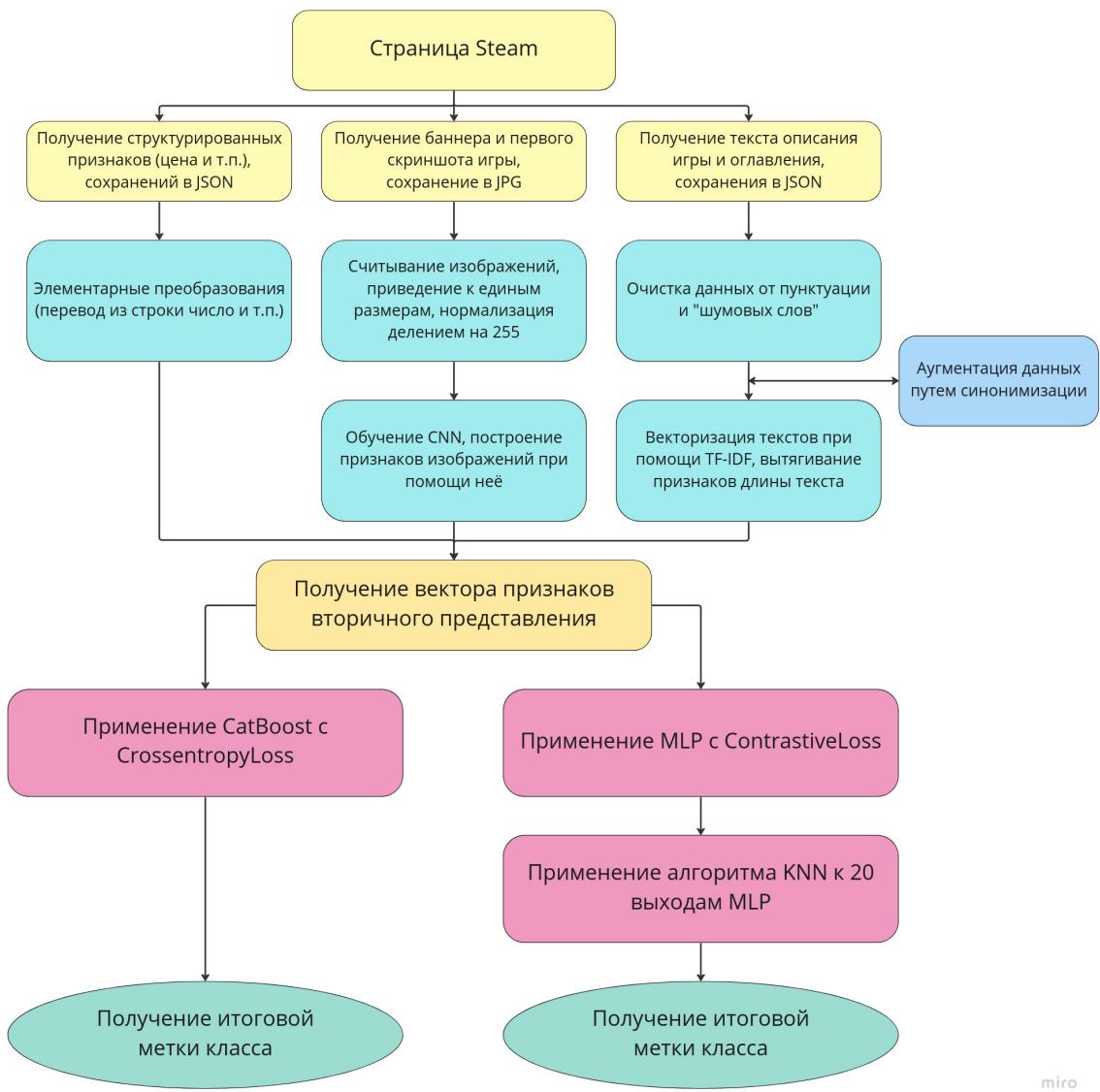


Рис. 9: Системы решения

4.5 Интерпретируемость признаков

4.5.1 Все признаки

Для интерпретируемости признаков воспользуемся РСА. Именно им, так как, кажется, некоторое небольшое множество признаков (опять же, изображения) могут вносить куда больший вклад в различие векторов.

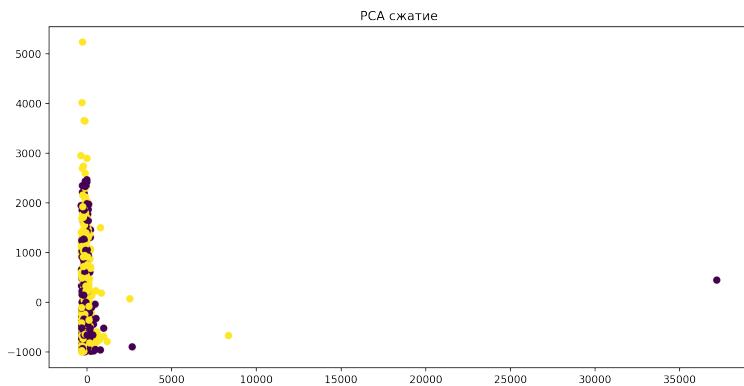


Рис. 10: PCA сжатие до 2 компонентов

Как можно заметить, интерпретировать такие данные нельзя. Видимо, это связано с тем, что заданные признаки не совсем хорошо отвечают на вопрос различия плохих страниц от хороших.

4.5.2 Отдельные признаки

Поскольку признаков даже во вторичном представлении получается очень много, рассмотреть все не получится. Но изучим хотя бы влияние, например, признаков изображений (которые оказывают наибольшее влияние на модель CatBoost в целом).

В случае изображений посмотрим на следующий их набор (слева указаны самые наиболее вовлеченны, а справа наоборот).

Можно увидеть, что левые объекты обладают куда большей детализацией маленьких объектов. У всех рисунков есть сильные контрастные маленькие объекты, и сцены выглядят более хаотично. Справа же цвета менее токсичные, сцены более мягкие.



(a)



(b)



(c)



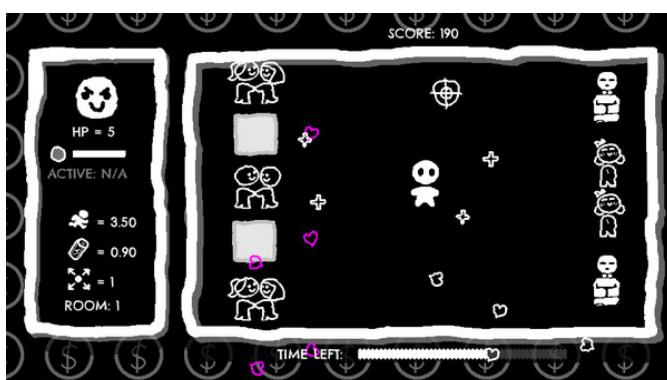
(d)



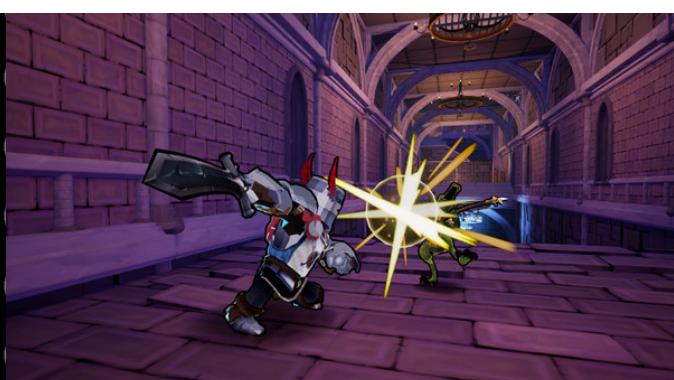
(e)



(f)



(g)



(h)

4.5.3 Микровывод

Обогащения выборки и изменения алгоритма обучения почти никак не влияют на интерпретацию данных.

5 Сравнение систем и окончательные выводы

Для удобства далее будем обозначать наши системы в виде тройки $Method\text{-}Augmentation_{state}\text{-}Label_{state}$. Например, для модели CatBoost без аугментации и с жесткими метками сокращение будет такое: *CatBoost-NoAug-Hard*.

5.1 Время обучения и работы алгоритмов

Все предложенные методы работают примерно за одинаковое количество времени. Это связано с единой тяжелой общей частью в виде CNN для обработки картинок. Всего обучение занимает примерно минуту, откуда более 50% уходит на сверточную сеть. Время тестирования крайне мало (порядка миллисекунд).

5.2 Качество распознавания

Рассмотрим поподробнее таблицу с результатами методов. Замеры производились с использованием метода кросс-валидации при разбиении выборки на 4 части.

Название метода	Худший, train	Худший, val
<i>CatBoost-NoAug-Hard</i>	1	0.59
<i>CatBoost-NoAug-Soft</i>	0.88	0.63
<i>CatBoost-Aug-Hard</i>	1	0.6
<i>CatBoost-Aug-Soft</i>	1	0.59
<i>MLP-NoAug-Hard</i>	0.67	0.63
<i>MLP-NoAug-Soft</i>	0.69	0.62
<i>MLP-Aug-Hard</i>	0.66	0.6
<i>MLP-Aug-Soft</i>	0.7	0.64

Таблица 7: Результаты работы систем. Метрика качества - *Accuracy*. Выбирались худшие результаты на основе кросс-валидации по 4 подвыборкам.

Результаты, конечно, далеко не самые лучшие. Видимо, задача довольно сложна, и итоговое ощущение от игры гораздо сильнее перевешивает то, как игра представлена в

магазине.

Все методы оказались примерно одинаково плохими, но лучшим по принятой метрике качества можно назвать MLP+KNN с аугментациями описаний игры и мягкими метками. На контрольной выборке *MLP-Aug-Soft* выдал 0.62 точности.

6 Архитектура моделей в подробностях и цифрах

Все архитектуры были подобраны экспериментальным путем.

6.1 CNN

CNN состоит из двух последовательных сверточных нейронных сетей, на вход которым подаются изображения баннера и первого скриншоты. Структуры подсетей аналогичны (веса при этом не общие). Один элементарный блок CNN состоит из операции свертки Conv2d, функции нелинейной активации ReLU() и пулинга MaxPool2d(). Всего таких блоков три, а в конце стоит аддитивным пулинг с последующими линейными слоями. На вход одной подсети подаются изображения размера (230, 108, 3), а другой - (300, 169, 3). Такие входные размеры обусловлены исходными размерами картинок (в два раза меньше, чем исходный размер).

Всего в сети **299578** обучаемых параметров.

6.2 MLP

Многослойный перспектор состоит из набора элементарных блоков, состоящих из линейного слоя, функции активации ReLU и Dropout. Всего таких блоков 4, в конце стоит еще один линейный слой, возвращающий в итоге вектор из 20 признаков, который после подается в KNN.

На вход сети подается обработанный вектор признаков (размера 473). Этот вектор и строится на этапе первичной обработки.

Всего в сети **195920** обучаемых параметров.

6.3 KNN

В качестве KNN используется готовая реализация *sklearn.neighbors.KNeighborsClassifier*. Подобранные число соседей составляет 15.

6.4 CatBoost

В качестве catboost используется готовое решение *CatBoostClassifier* с функцией потерь *logloss*. Так как количество параметров у *CatBoostClassifier* очень велико, на-

страивались только параметры, предлагаемые в официальной документации https://catboost.ai/en/docs/concepts/python-reference_catboost_grid_search.

В итоге были методом *grid_search* были подобраны параметры iterations=1000, learning_rate=0.0, depth=3

6.5 ContrastiveLoss

В качестве ContrastiveLoss использовалась готовая реализация PyTorch *CosineEmbeddingLoss*. Она основана на косинусной схожести векторов.

$$\text{loss}(x, y) = \begin{cases} 1 - \cos(x_1, x_2), & \text{if } y = 1 \\ \max(0, \cos(x_1, x_2) - \text{margin}), & \text{if } y = -1 \end{cases}$$

Рис. 13: Формула подсчета ContrastiveLoss. $y = 1$, если метки объектов совпадают, и -1 иначе

7 Заключение

Была проведена работа по предсказанию оценок игры в Steam (низкие - ниже 85%, высокие - иначе). По результатам работы было проделано следующее:

1. Предварительный анализ исследуемого объекта (страницы Steam). Анализ полей с отсеиванием и выявлением потенциально хороших признаков;
2. Извлечение данных из торговой площадки Steam при языке Python и библиотеки requests;
3. Предобработка данных разных форматов и типов: простые числовые признаки (цена, отзывы, и др.), обработка one-hot признаков (список доступных языков, жанров, тегов разработчика), обработка слабоструктурированного текста (с приведением его к безпунктуационному виду, удалением шумовых слов и прогоном через векторизатор TF-IDF), обработка изображений (при помощи нейронной сети CNN), учет пропущенных значений;
4. Оценка репрезентативности данных в рамках задачи;
5. Составление полного описания входных данных до и после обработки;
6. Анализ интерпретируемости данных на вторичном представлении глобально и в зависимости от признаков;
7. Составление 8 систем, решающих поставленную задачу (с применением CatBoost, MLP и KNN, аугментаций слабоструктурированной части, мягких/жестких методов);
8. Оценка качества работы систем с применением кросс-валидации и метрики Accurасy;
9. Принятие окончательного решения по поводу лучшего решения и проверка его на контрольной выборке.