МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по курсу

«Data Science»

Тема: «Прогнозирование потенциальной прибыли компьютерных игр»

Слушатель Быков Илья Павлович

Москва, 2023

Содержание

Введение

По прогнозам ведущих аналитиков, мировая игровая индустрия достигнет 256 млрд. долларов к 2025 году. Более чем 3.2 миллиарда людей во всем мире играют в игры. Прибыль от одних только мобильных игр оценивается как имеющая потенциал превысить 100 млрд. долларов уже в 2023 году. Одни только Соединенные Штаты Америки отвечают за более чем 40 млрд. долларов прибыли от игровой индустрии.

Благодаря ситуации с пандемией, количество игроков в игры по всему миру сейчас как никогда велико за всю историю, более 150 млн. человек в одних только США играют в мобильные игры на регулярной основе. И эта цифра прогнозируется стать больше еще на 30 млн. к 2027 году, демография игроков также включает в себя не только молодых но и пожилых граждан, излюбленным жанром которых являются, так называемые, мобильные, казуальные игры. Один только жанр головоломок может достичь 29 млрд. долларов прибыли к 2026 году. Более того, мировая индустрия кибер-спорта привлекла более чем 500 миллионов человек в качестве аудитории через стриминговые платформы, такие как Twitch в 2022 году.

Весьма примечателен факт того, что средний возраст игрока в видео-игры в 2022 году повысился до 31 года. Наиболее играющая возрастная категория теперь между 18 и 34 лет, и составляет 38% от общего числа игроков, достигая, по численности, 3.8 млрд человек.

Наиболее актуальным трендом в области предоставления гражданам услуг в области видеоигр продолжают оставаться электронные платформы по распространению и продаже игр без необходимости в приобретении физической копии, такие как платформа Steam.

Начав свою историю с 2003 года, как платформа, разработанная компанией Valve для того, чтобы позволить им автоматически обновлять этой проблемы обычно используются два способа: физические тесты образцов материалов или

свои игры, сейчас Steam является крупнейшей в мире платформой по продаже игр для персональных компьютеров, с 120 млн. активных игроков, зарегистрированных на платформе и до 25 млн. пользователей, одновременно на ней находящихся, которые используют платформу, как для игры, так и для простого просмотра страниц.

Широко известно, насколько сложны игры в разработке. Даже тривиальные вещи вроде создания работающей исправно двери или лифта в игре, опытные разработчики называют настоящим испытанием. Не говоря уже о сложнейших системах ИИ для врагов, или персонажей игр, графики, оптимизации, игровых системах и так далее. Все это требует значительных людских ресурсов, времени, а, следовательно, и денег, чтобы не только разработать, но так же и, в дальнейшем, поддерживать, исправно выпуская патчи на вновь и вновь возникающие баги, а так же, дополнительный продаваемый контент.

Самый эффективный способ уменьшить риск, что все затраты на разработку не будут потрачены зря, это принять верное решение в области того какую именно игру, вы собираетесь делать, еще на этапе концепции или идеи. Также, учитывая, что большинство игр не достигают ошеломительного успеха, более того, ситуация настолько плоха, что те игры, которые достигают, статистически могут считаться выбросами, данная задача может быть отнесена к задаче несбалансированной бинарной классификации.

Целью данной квалификационной работы стало исследование вопроса о том, возможно ли, используя методы машинного обучения, предсказать шансы на успех для игры, на основе «тагов», меток которые используются для классификации игр на платформе Steam. Для этого были разработаны несколько моделей для бинарной классификации, а так же нейросеть, на основе которых, было разработано простое приложение, которое получает на вход 3 тага игры, и выдает метрику ее, потенциального, успеха. Данной метрикой является бинарный класс.

.

Аналитическая часть

1.1.Постановка задачи

Для исследовательской работы был использован файл: steam_games.csv (с данными о различных видео-играх на платформе Steam, состоящий из 121904 строки и 22 столбцов.

title	Apex Legends™	God of War	ELDEN RING	
url	https://store.steampowered.com/app/1172470/Ape...	https://store.steampowered.com/app/1593500/God...	https://store.steampowered.com/app/1245620/ELD...	https://store.steampowe
image	https://cdn.akamai.steamstatic.com/steam/apps/...	https://cdn.akamai.steamstatic.com/steam/apps/...	https://cdn.akamai.steamstatic.com/steam/apps/...	https://cdn.akamai.st
release_date	4 Nov, 2020	14 Jan, 2022	24 Feb, 2022	
platforms	Windows	Windows	Windows	
discount_rate	NaN	-20%	NaN	
original_price	Free to Play	Rp 729 000	Rp 599 000	
discounted_price	NaN	Rp 583 200	NaN	
developer	Respawn Entertainment	Santa Monica Studio	FromSoftware Inc.	
publisher	Electronic Arts	PlayStation PC LLC	FromSoftware Inc.,Bandai Namco Entertainment	
overall_reviews	Very Positive	Overwhelmingly Positive	Very Positive	
recent_reviews	- 81% of the 15,998 user reviews in the last 3...	- 96% of the 1,056 user reviews in the last 30...	- 92% of the 14,027 user reviews in the last 3...	- 90% of the 15,02
whole_reviews	- 86% of the 469,045 user reviews for this gam...	- 97% of the 34,533 user reviews for this game...	- 90% of the 381,880 user reviews for this gam...	- 85% of the 1,226
description	Apex Legends is the award-winning, free-to-pla...	His vengeance against the Gods of Olympus year...	THE NEW FANTASY ACTION RPG. Rise, Tarnished, a...	Grand Theft Auto V
tags	Free to Play,Battle Royale,Multiplayer,Shooter...	Action,Adventure,Singleplayer,Story Rich,Mytho...	Souls-like,Relaxing,Dark Fantasy,RPG,Difficult...	Open World,Action,Mu
genre	Action,Adventure,Free to Play	Action,Adventure,RPG	Action,RPG	

Рисунок 1 – выборка первых 3-х строк данных из steam_games.csv

Цель данной исследовательской работы заключается в том, чтобы выяснить, возможно ли путем разработки и применения моделей машинного обучения получить удовлетворительный результат прогнозирования потенциальной прибыли видео-игр на платформе Steam, использую исключительно метки этих игр.

Для достижения поставленной цели, необходимо провести разведочный анализ данных, удалить, или заполнить пропуски, дубликаты, определиться с критерием деления игр на 2 класса, преобразовать данные в форму подходящую для подачи в модели машинного обучения.

Затем, необходимо обучить модели машинного обучения на тренировочной выборке, найти для ряда моделей оптимальные гиперпараметры при помощи метода кросс-валидации. После этого нужно оценить результат работы моделей на тестовой выборке, по ряду метрик, использующихся для оценки ре-

зультата бинарной классификации, в частности для несбалансированных, по целевым классам выборок данных.

Наиболее эффективную модель необходимо сохранить, и создать приложение на основе фреймворка Flask, которое будет принимать на вход 3 метки игры, и выдавать результат обозначающий принадлежности игры с такими метками к целевому классу.

1.2. Описание используемых методов

Так как наша задача является задачей бинарной классификации с несбалансированными целевыми классами, были применены модели классификации наиболее лучше соответствующие поставленным условиям:

1) логистическая регрессия;
2) дерево решений;
3) гауссовский байесовский классификатор;
4) метод K-ближайших соседей;
5) метод опорных векторов SVC;
6) случайный лес.

Логистическая регрессия – это модель линейной классификации, которая использует логистическую функцию для определения вероятности результата классификации, например сигмоиду или «логит» функцию.

Реализация логистической регрессии библиотеки scikit-learn, поддерживает как бинарную так и полиномиальную логистическую регрессию с регуляризацией по умолчанию.

Простая логистическая регрессия бинарной классификации использующая сигмоиду в качестве логистической функции напоминает линейную регрессию, однако график ее функции — сигмоида, а не прямая линия, и на выходе бинарное значение а не любое численное значение. Значение логистической функции изменяется согласно ее определению, например по сигмоиде, и если

оно превысит заданный порог, то будет определен другой класс для этого значения.

Однако, это значение никогда не достигает точного значения заданного максимума кривой, только приближенного к ним, принадлежность же к классу определяется порогом. На рисунке ниже проиллюстрирована функция сигмоиды.

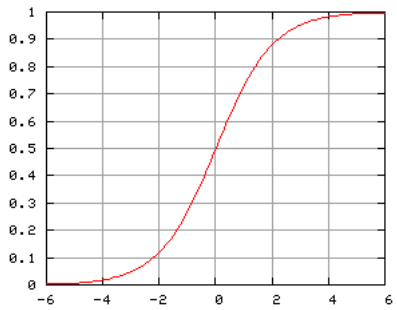


Рисунок 3 — логистическая функция сигмоиды

Достоинства логистической регрессии, которые побудили избрать ее для поставленной задачи:

1) хорошо подходит для бинарной классификации;

2) имеет неплохую точность на маленьких датасетах;

3) выдает не только класс, но также и вероятность принадлежности к нему;

4) может быть использована с несбалансированными датасетами;

5) не так подвержена переобучению с маломерными датасетами;

Недостатки, особенно критичные для нашей проблемы:

1) чувствительность к выбросам;

2) проблемы с распознанием сложных взаимосвязей между признаками;

Дерево решений – алгоритм, использующий простую систему правил для репрезентации данных, и предсказания значений. Каждое звено дерева решений представляет собой тест или вопрос о данных, с целью выяснения их связи с целевым значением или классом, например мы можем задать вопрос, больше ли возраст ученика 15 лет или меньше?

Затем на основе других признаков в датасете, таких как возраст родителей, класс, любимой еды, мы определяем новые звенья деревьев связанные между собой ответами на вопросы, которые эти звенья представляют. Затем звенья-ответы, в зависимости от их точности, которая определяется по формуле, например, gini impurity, либо становятся новым звеном с уточняющим вопросом, либо становится «листом», конечным звеном дерева с идеальной точностью. Чтобы дерево не переобучилось, например когда количество ответов на одном листе всего одно, можно обрезать листы, и подняться вверх к менее точному звену, и взять большинство ответов там за правильный ответ, либо ограничить количество возможных звеньев в иерархии, ограничив «глубину дерева».

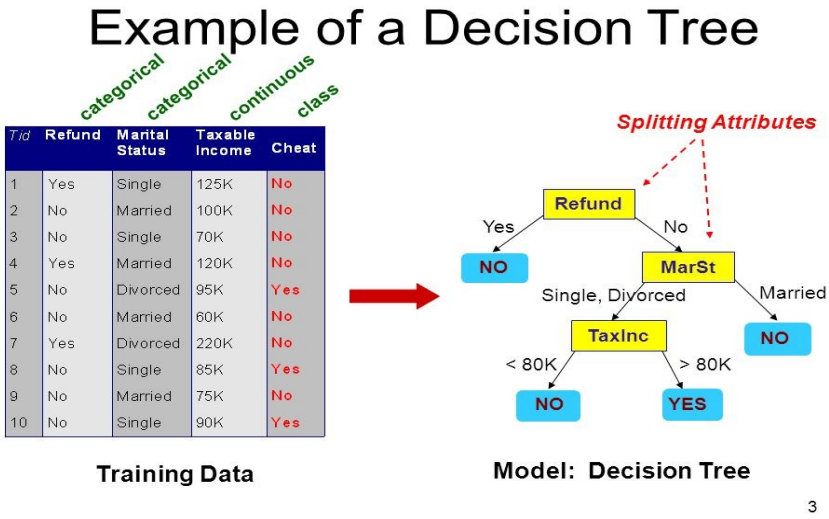


Рисунок 4 — пример простого дерева решений

В соответствии с рисунком 4, можно увидеть, что после обучения дерева и выстраивания иерархии звеньев, можно подать строку с данными на вход и следуя по иерархии признаков вниз, прийти к листу с предсказанным классом по поданным данным.

Достоинства:

1) просты в понимании и интерпретации и визуализации;

2) не требовательны к предобработке данных.

Недостатки:

1) необходимо активно следить за тем чтобы модель не переобучилась, путем применения обрезания листьев, ограничения минимума количества ответов на листе, и ограничения максимальной глубины дерева;

2) плохая устойчивость к дисбалансу классов.

Гауссовский байесовский классификатор — алгоритм классификации, использующий теорему Байеса. Вероятность признаков считается Гауссовской.

Достоинства:

1) хорошо работают на кейсах из реальной жизни;

2) требует небольшого количество данных для обучения;

Недостатки:

1) плох в оценке вероятностей классов.

Метод K-ближайших соседей — классифицирует объекты на основе их близости друг к другу. Параметр k регулирует количество ближайших элементов, которые будут учитываться для классификации. Если расстояние между ними и классифицируемым элементом достаточно близкое, то наибольшее количество ближайших элементов в k определенного класса - определит класс нового элемента.

Достоинства:

1) прост в освоении и понимании, однако эффективен в боевых проблемах;

2) не параметричен, а значит что структура модели определяется датасетом;

3) способен на нелинейное разбиение классов;

Недостатки:

1) плох в оценке вероятностей классов;

2) медлителен в работе с большими датасетами;

3) чувствителен к диапазону и качеству данных;

4) «ленивый», не обучается на данных, а просто запоминает их и использует эту информацию для классификации новых данных.

Алгоритм K-ближайших соседей использует заученную и сохраненную в памяти информацию, чтобы классифицировать новые инстанции, и так как ему приходится рассчитать все расстояния для каждого элемента тренировочной выборки, он требует существенных вычислительных мощностей, и объема памяти, если датасет большой. На изображении снизу, проиллюстрировано как происходит процесс предсказания.

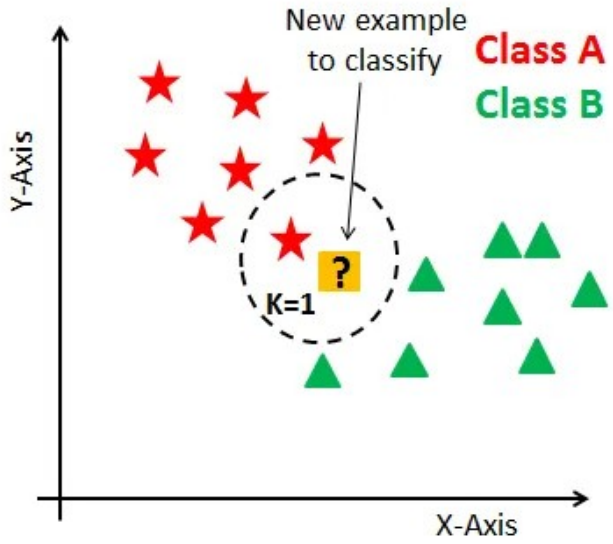


Рисунок 5 — K-nearest neighbor

В соответствие с рисунком 5, можно увидеть визуализацию работы алгоритма ближайших соседей.

Метод опорных векторов – в основе данного линейного классификатора лежит концепция о гиперплоскости, разделяющей элементы на классы. Причем это может быть как линия, так и многомерная гиперплоскость. Располагается эта линия на границе между крайними элементами классов, однако существует также буферная зона за границами этой линии с обоих сторон, попав в которую элементу противоположного класса разрешается быть ошибочно классифицированным, так как это позволяет гиперплоскости оставаться на разумном месте в целом для разделения двух классов, и не сместиться слишком сильно, сохранив баланс двух сил, которые контролируют всю сферу машинного обучения, «variance» и «bias».

Если проблема линейно неразделима, тогда утилизируется повышение размерности данных по специальным формулам, функциям ядер. Например, полиномиальным ядром. На картинке снизу можно увидеть иллюстрацию работы метода опорных векторов на различных типов ядер. LinearSVC это версия метода SVC библитеки scikit-learn, которая использует линейное ядро. Аналог метода SVC с параметром kernel установленным как linear, однако использующим другую библиотеку линейной классификации, более оптимизированную для больших датасетов.

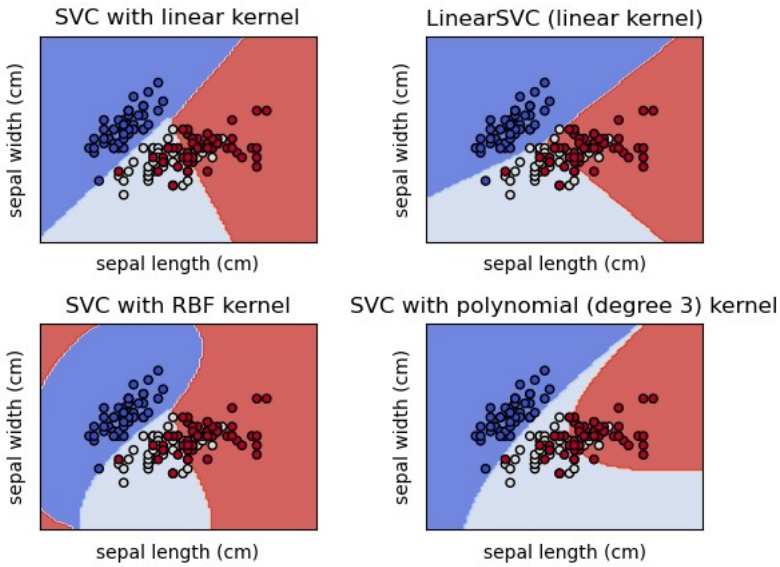


Рисунок 6 — иллюстрация работы методов SVC и LinearSVC

В соответствии с рисунком 6, можно увидеть, что на нем проиллюстрированы визуализации результатов работы методов SVC с различными типами ядер и LinearSVC.

Достоинства:

1) хорошие результаты при серьезном дисбалансе классов;

2) быстрая выдача предсказаний;

3) имеет высокую гибкость в настройках, благодаря множеству ядер.

Недостатки:

1) плохо справляется с наложенными классами;

2) плох для больших датасетов;

3) чувствителен к шуму, особенно среди классов.

Случайный лес — это ансамблевый метод, состоящий, однако, исключительно из деревьев решений. Алгоритм создает множество деревьев, из датасета состоящего из случайным образом выбранных записей из исходного датасета, методом bootstrapping, то есть, с разрешением дубликатов. Затем, для каждого дерева случайным образом выбираются признаки из нового датасета, и лучший становится корнем дерева. Затем признак, ставший корнем исключается из нового датасета, и из оставшихся признаков вновь случайно выбираются новые признаки, для того чтобы разделить уже новые звенья дерева, отходящие от корня, и так для всех звеньев, пока дерево не будет построено полностью. Таким же образом, будут построены еще, например, 100 деревьев. В реализации алгоритма библиотеки scikit-learn, количество деревьев определяется параметром n_estimators.

Затем из записей не попавших в bootstrap датасет будет составлена валидационная выборка, которая будет пропущена через лес, в результате чего, при помощи пропорции правильных к неправильным ответов будет оценена точность случайного леса.

Затем мы повторяем процесс, но в этот раз меняя количество признаков мы случайным выбираем создавая новое дерево. В конце всех итераций, мы выбираем лес с наилучшей точностью.

Предсказанный сэмпл будет пропущен через все деревья, и результат получивший больше всего голосов станет выходом случайного леса.

Достоинства:

1) устойчивость к выбросам;

2) благодаря генерализации выхода множества деревьев, решается проблема склонности к переобучению дерева решений;

3) устойчивы к пропускам;

4) очень адаптируемый и точный алгоритм.

Недостатки:

1) долгий и затратный процесс обучения;

2) тяжел и сложен для понимания;

3) долгий процесс выдачи предсказаний;

4) низкая гибкость настройки;

5) затратен с точки зрения производительности для больших датасетов.

1.3.Разведочный анализ данных

Так как мы поставили перед собой задачу бинарной классификации исключительно использую не более 3 меток игры, наш разведочный анализ преимущественно состоял из извлечения и обработки меток игры, удалении пропусков там, где метки совпадали, определения на основе имеющихся данных целевых классов, и построение графиков распределения целевой переменной, которое помогло бы нам лучше задать критерии определения этих классов.

Важнейшей задачей стало определении тактики работы с дубликатами. Было принято решение удалить все записи в выборке данных, у которых отсутствует, либо является меньше заданного количества, количество меток. Так же было принято решение использовать поле, описывающее общее количество отзывов за игру, и поле с датой выхода игры, для расчета нашей целевой переменной, которая в дальнейшем помогла рассчитать классы бинарной классификации.

Остановимся подробнее на дубликатах, так как это стало ключевым аспектом разведочного анализа. Сначала мы удалили все записи, где первые 3 метки и вычисленный нами параметр обозначающий общее количество отзывов, совпадали. Хотя другие могли отличатся, мы игнорируем их и считаем такие записи дубликатами.

	reviews_count	dupl_check
2794	203	StrategyWarWorld War II
3577	203	StrategyWarWorld War II
3669	112856	StrategyTurn-BasedTurn-Based Strategy
5316	112856	StrategyTurn-BasedTurn-Based Strategy
5283	74015	StrategyTower DefenseZombies
...	...	...
4008	54	2D PlatformerActionPixel Graphics
3726	279	2.5DFPSOld School
4365	279	2.5DFPSOld School
3115	1132	1990'sAdventurePixel Graphics
4151	1132	1990'sAdventurePixel Graphics

Рисунок 8 - дубликаты

На рисунке 13 видно, что мы определяем как дубликаты записи с совпадающими первыми 3-мя метками, и совпадающими значениями поля reviews_count, которое обозначает в нашей работе общее количество отзывов на игру. Все они были удалены.

Однако, как видно на рисунке 14, у нас все еще очень много игр, у которых совпадают первые 3 метки, но не совпадает количество отзывов. Так как из общего количества отзывов мы в дальнейшем рассчитаем как мы будем определять классы, получается что одни и те же данные связаны у нас с разными значениями целевой переменной. Это определенно негативно скажется на работе наших моделей.

	reviews_count	days_since_release	dupl_check
1639	5275	784	StrategyTurn-BasedTurn-Based Tactics
626	3188	1296	StrategyTurn-BasedTurn-Based Tactics
4633	56957	2329	StrategyTurn-BasedTurn-Based Strategy
2077	3875	5718	StrategyTurn-BasedTurn-Based Strategy
3435	112856	4291	StrategyTurn-BasedTurn-Based Strategy
...	...	...	...
2109	146	733	2D FighterActionFighting
867	4333	265	2D FighterActionAnime
1345	2566	944	2D FighterActionAnime
4632	4687	830	2D FighterActionAnime
561	1787	97	2D FighterActionAnime

2336 rows × 3 columns

Рисунок 9 - проверка выборки на наличие дубликатов исключительно по первым трём меткам

Чтобы разрешить эту проблему было решено удалить все дубликаты игр с одинаковыми первыми 3-мя метками, и оставить только одно, уникальное зна-

чение для каждой игры в датасете. Таким образом, все комбинации первых трёх меток в нашем датасете уникальны.

Что удаляет огромное количество информации и вскрывает серьезную проблему в самой постановке задачи, но мы продолжим решать поставленную задачу все равно, а отсутствующую информацию мы возместим тем, что заменим значение количества отзывов для всех оставшихся значений комбинаций меток, на среднее между ними и значений удаленных дубликатов.

В рамках разведочного анализа, для определения категорий классов на которые мы будем в дальнейшем прогнозировать, было решено в качестве целевой переменной создать новое значение, которое является средним количеством отзывов игра получила в месяц. Для этого значения даты были преобразованы в количество дней с момента выхода игры, а затем поделены на количество отзывов, которые игра получила за все время после своего выхода.

Так как единственный способ предсказать прибыль игры на платформе Steam это умножить количество отзывов на предполагаемое число владельцев игр на отзыв, которое примерно от 30 до 100, мы будем использовать эту метрику, чтобы определить потенциальный успех игры. Из-за сильного разброса, ее было решено логарифмировать, гистограмма логарифма этой переменной представлена на рисунке 15.

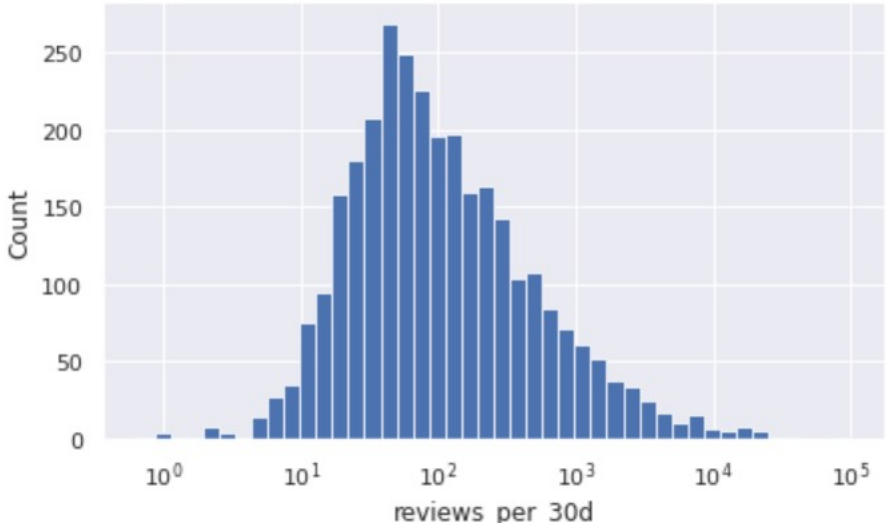


Рисунок 10 - распределение логарифма целевой переменной

Как можно заметить, ее распределение напоминает распределение Пуассона.

Далее были протестированы различные способы избавления от выбросов и построены диаграммы «Ящик с усами» .

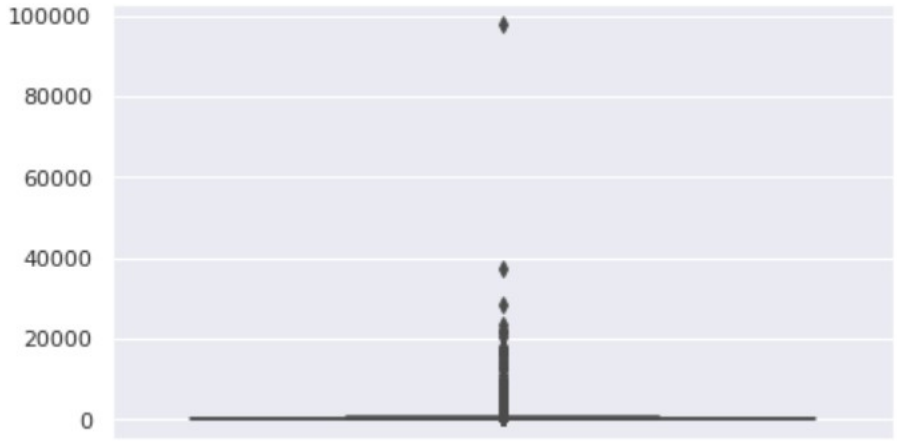


Рисунок 11 - диаграмма «Ящик с усами» целевой переменной без логарифма

Как видим из рисунка 16, без логарифма наша переменная имеет слишком большой диапазон значений, и выбросов. На рисунке 17 можно увидеть гистограммы нашей целевой переменной вместе с диаграммами «ящик с усами» для ее логарифма, и ее логарифма с примененными к ней методами исключения выбросов «zscore» и «quantile». В общем итоге, было решено не использовать ни один из них, по причине того, что они удаляют слишком много полезной информации о супер успешных играх.

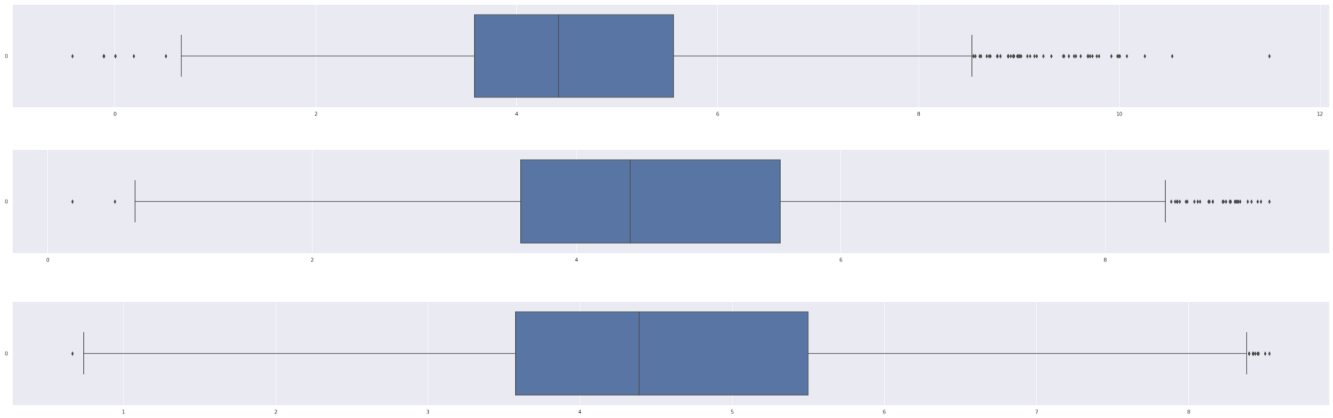


Рисунок 12 - сравнение методов избавления от выбросов на примере диаграммы «ящик с усами»

На этом графике можно увидеть диаграмму «ящик с усами» логарифма целевой переменной:

1) без применения методов избавления от выбросов;
2) с применением метода избавления от выбросов «zscore»;
3) с применением метода избавления от выбросов «quantile»;

В соответствии с рисунком 17, оба метода избавляются от слишком большого количества ценной информации справа графика, где расположены наши игры-хиты. Взглянем теперь на графики их гистограмм.

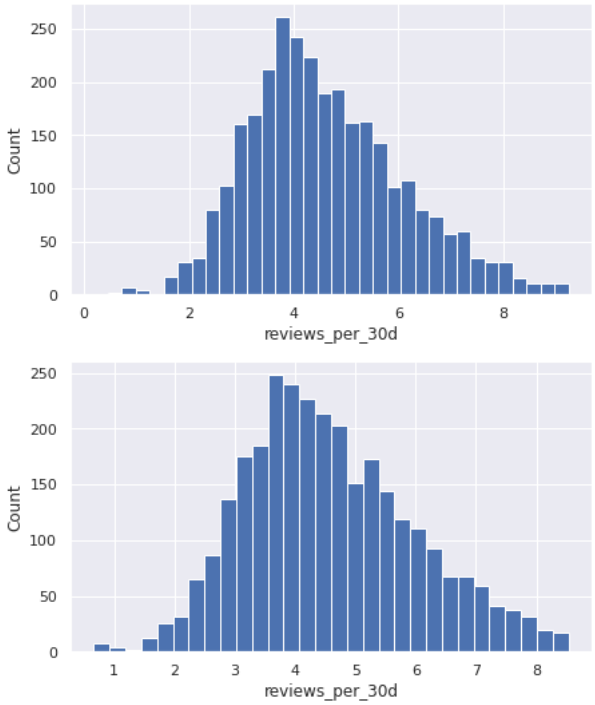


Рисунок 13 - гистограммы после применения методов избавления от выбросов

В соответствии с рисунком 18, на графиках мы видим гистограмму целевой переменной после применения метода «zscore» и, затем, гистограмму целевой переменной после применения метода «quantile»;

Как мы видим, в соответствии с рисунком 18, после применения избавления от выбросов распределение начинает напоминать нормальное. Однако данные методы было решено не использовать.

Для определения двух классов, на которые мы будем классифицировать наши записи, было решено разбить все переменные на десять квантилей и по-

строить график «ящик с усами» для каждого квантиля, чтобы лучше увидеть распределение значений в целевой переменной.

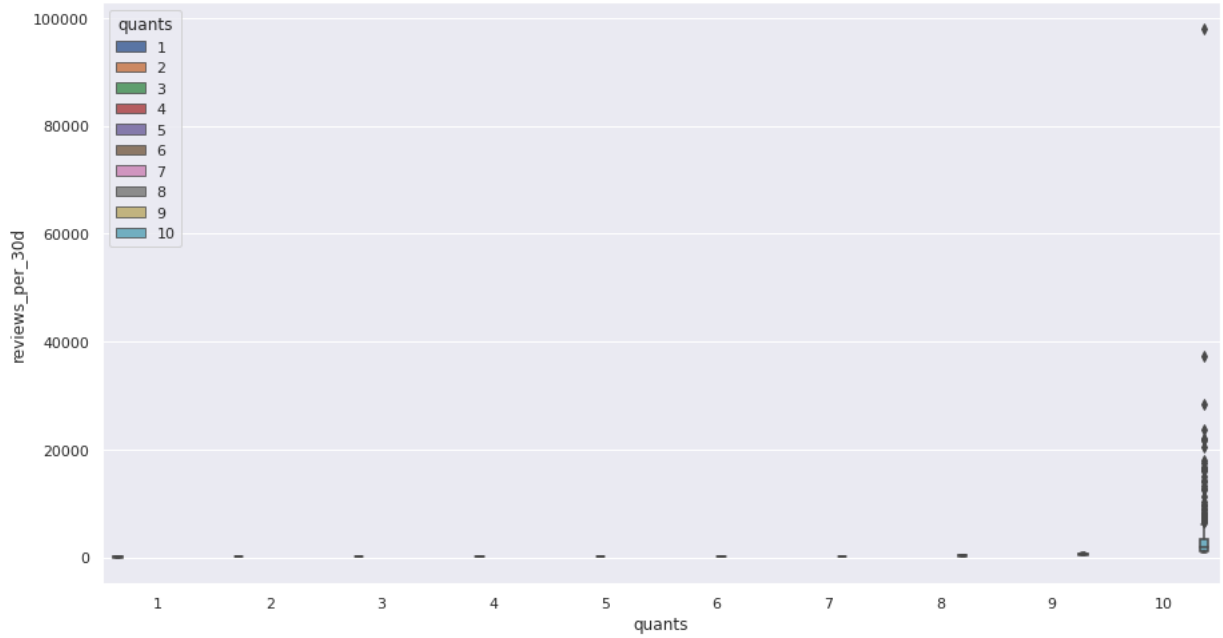


Рисунок 14 - график «ящик с усами» для десяти квантилей целевой переменной

Как мы видим в соответствии с с рисунком 19, 10 квантиль содержит большую часть наших высоких значений, и наша целевая переменная не равномерно возрастает. Только небольшое число игр добивается успеха на платформе steam, например, только 98% всех игр достигают отметки в количество владельцев больше миллиона.

Такое распределение нашей целевой переменной позволяет предположить, что возможно создание целевого класса «супер успеха», и применение бинарной классификации по разделению всех игр на, условно, «супер успешные» и «не успешные». Давайте взглянем на график диаграммы «ящик с усами» разбитый по квантилям для логарифмированной целевой переменной, и также с применением методов избавления от выбросов «zscore» и «quantile».

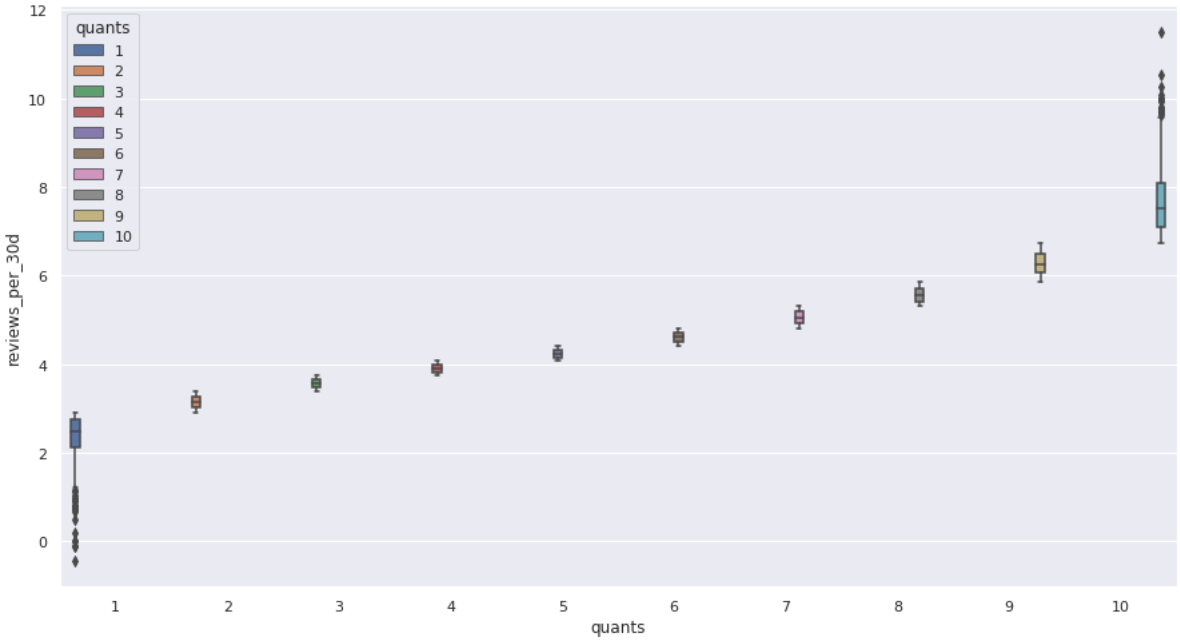
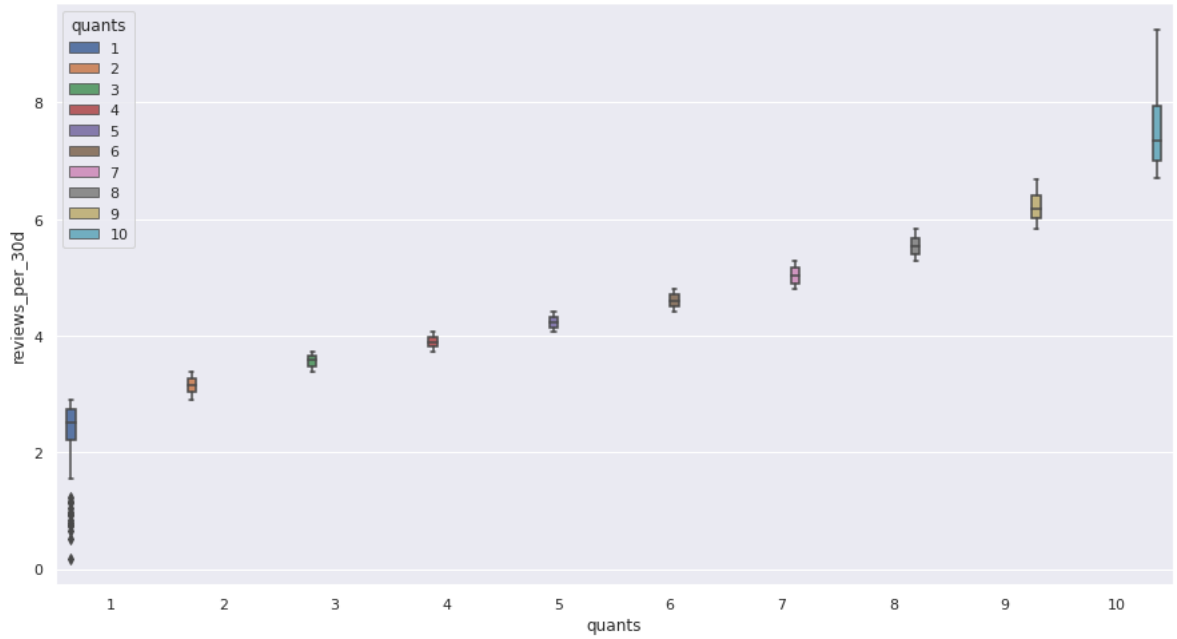


Рисунок 15 - график квантилей для логарифмированной переменной

В соответствии с рисунком 20 мы видим, что большая часть данных имеют слабую волатильность. Посмотрим на такие же графики но с применением методов «zscore» и «quantile».



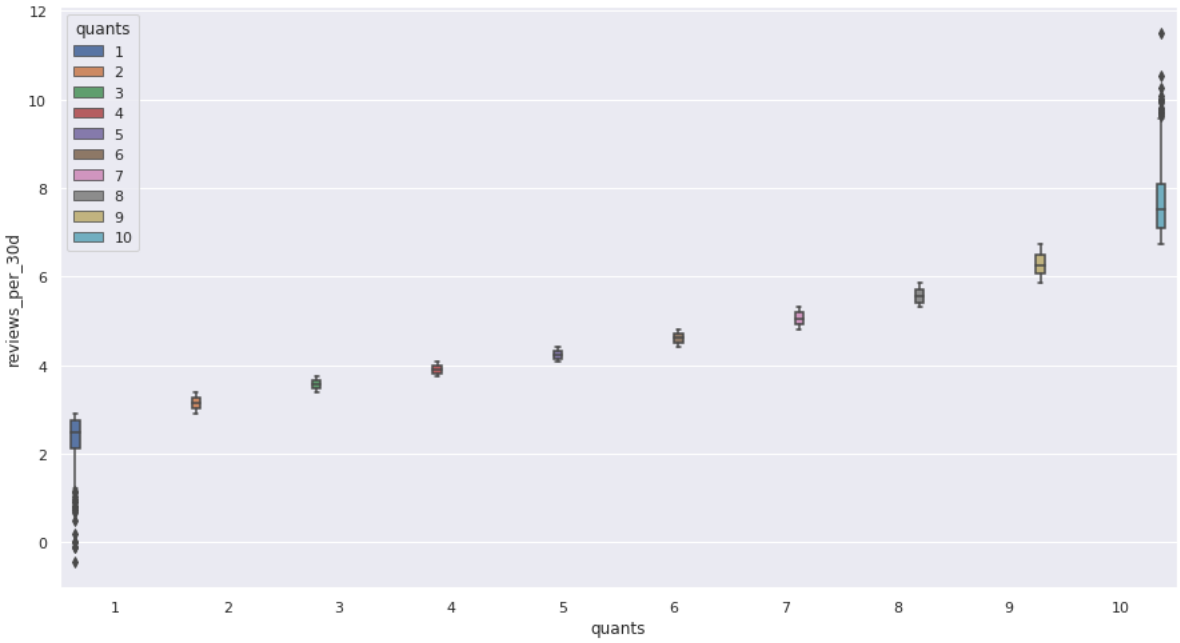


Рисунок 16 — квантильный график после применения «zscore» и «quantile»

В соответствии с рисунком 16, мы видим что «zscore» вырезал больше крайних высоких значений чем «quantile», что делает его не подходящим для наших целей поиска супер успешных игр и классификации их в соответствующую категорию.

В общем итоге были выбраны последние 2 квантиля, как принадлежащие к классу «супер успешных» и все остальные записи как «не успешные». И на их основе было совершенно разбили на 2 класса, которые мы будем предсказывать. В итоге, мы будем решать задачу обучения с учителем на несбалансированном датасете, при помощи бинарной классификации.

```
quants = dst.reviews_per_30d.quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
dst["rating"] = pd.cut(dst.reviews_per_30d, quants, labels = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], right=True, include_lowest=True)
```

```
#Используем последние 2 квантиля, как бинарный класс определяющий "успех"
dst['is_successful'] = dst.rating > 8
dst['is_successful'].sum()
```

Рисунок 17 — создание классов для бинарной классификации

В соответствии с рисунком 17, мы видим, что было создано 2 класса для бинарной классификации, те которые принадлежат двум последним квантилям, и все остальные.

Практическая часть

2.1.Предобработка данных

Нормализация в нашем случае не нужна, так как на взоде у нас одни категориальные переменные, поэтому вместо нее пред нами стояла проблема кодирования этих переменных. Ниже выведен наш датасет до обработки на входе на модели.

	tag1	tag2	tag3
0	Free to Play	Battle Royale	Multiplayer
1	Action	Adventure	Singleplayer
2	Souls-like	Relaxing	Dark Fantasy
3	Open World	Action	Multiplayer
4	Racing	Open World	Driving

Рисунок 18 — датасет подаваемый на модель до обработки

В соответствии с рисунком 18, мы видим что наш датасет состоит из трёх колонок, каждая из которых является строкой, обозначающей нашу метку. Перед тем как подавать ее в модели, эти метки нужно привести к числовому значению. Для того чтобы это сделать, был выбран класс TargetEncoder библиотеки category_encoders. Он заменит наши строки с метками на число, отражающее смесь апостериорной вероятности целевой переменной с заданным конкретным категориальным значением и априорной вероятности цели по всем обучающим данным. На рисунке ниже, будет изображена наша итоговая выборка, преобразованная в числовые значения.

	tag1	tag2	tag3
177	0.210210	0.255002	0.429923
3007	0.270230	0.325690	0.126232
2955	0.345940	0.389890	0.244699
2389	0.261258	0.130153	0.211574
2602	0.244699	0.272970	0.171630

Рисунок 19 — итоговый датасет предобработанных данных

В наших классах есть дисбаланс, который будет виден на картинках снизу, он был сохранен как в тренировочной так и в тестовой выборке наших лейблов, путем использования стратифицированного разделения на тренировочную и тестовую выборки.

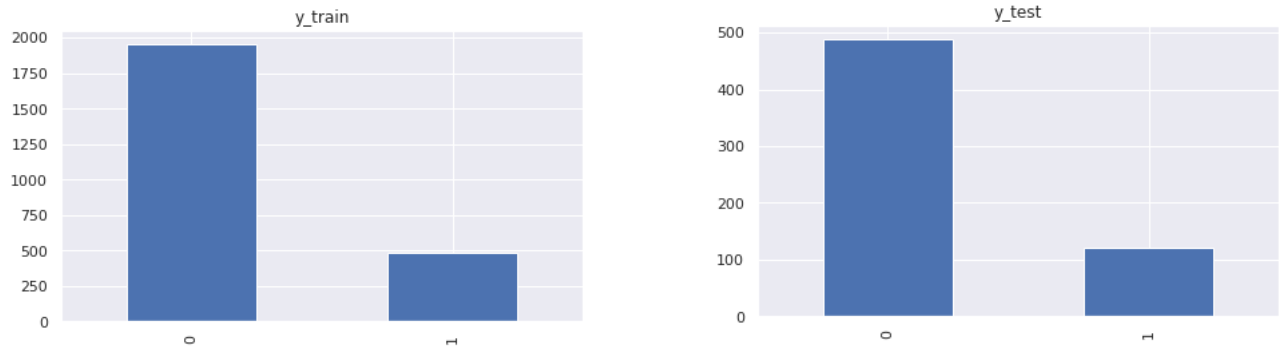


Рисунок 20 — дисбаланс классов итоговых выборок лейблов

2.2. Разработка и обучение модели

Перед тем как показать список моделей, которые будут использоваться для прогнозирования нашего класса «успех», давайте расскажем подробнее о метриках, которые будут использоваться для оценки результатов работы алгоритмов.

f1_score – среднее гармоническое между точностью и полнотой. Соотношение между точностью и полнотой в этом показателе относительно равно. Преимущество данной метрики в том, что она хорошо работает как с сбалансированными так и с несбалансированными датасетами. В отличие от метрики accuracy. В рисунке внизу показаны примерные оценки показателя f1_score.

F1 score	Interpretation
> 0.9	Very good
0.8 - 0.9	Good
0.5 - 0.8	OK
< 0.5	Not good

Рисунок 21 — примерные оценки показателей метрики f1_score

В соответствии с рисунком 21, мы можем увидеть примерные оценки метрики f1_score.

Precision – отношение правильно классифицированных позитивных классов к правильно и неправильно классифицированным позитивным классам. Как точно модель может распознавать позитивные классы.

Recall – отношение правильно классифицированных позитивных классов к правильным позитивным классам и неправильно классифицированным позитивным классам. Как много позитивных классов в целом смогла распознать модель.

matthews_corr_coef – отражает корреляцию между предсказанными результатами и правдой. -1 означает полное несоответствие между предсказанными классами и правдой, 1 на полное соответствие и 0 означает полностью случайное угадывание. Особенно хорошо работает с несбалансированными данными.

Таблица 1, где перечислены модели, которые будут использоваться для прогноза нашего класса «успех».

Таблица 1. Список моделей которые будут использоваться

Название модели	Реализация в библиотеке scikit_learn
0	1
Логистическая регрессия	Logistic_regression
Дерево решений	DecisionTreeClassifier
Гауссовский байесовский классификатор	GaussianNB
Метод K-ближайших соседей	KNeighborsClassifier
Метод опорных векторов SVC	SVC
Случайный лес	RandomForestClassifier

2.3. Тестирование модели

Протестируем наши модели и посмотрим на результаты. Были протестированы все методы из списка на входе получив массив с закодированными категориальными переменными и лейблами, состоящими из значений 1 и 0. 1 обозначает игры, которые попали в последние 2 квантиля целевой переменной, которая обозначает среднее количество отзывов игра получила за 30 дней с момента ее выхода.

Для оценки моделей используются метрики, которые мы описали выше. Рисунок с итоговыми метриками моделей представляет из себя pandas DataFrame с столбцами, обозначающими метрики и названия моделей. Также алгоритмы по возможности были подготовлены для работы с несбалансированными данными.

	model_name	f1_score	precision	recall	matthews_corr_coef
0	Logistic regression	0.509202	0.406863	0.680328	0.366585
1	Decision Tree Classifier	0.355731	0.343511	0.368852	0.187627
2	Gaussian Naive Bayes	0.341969	0.464789	0.270492	0.240256
3	K-Nearest neighbors	0.445652	0.661290	0.336066	0.387900
4	SVC	0.474138	0.500000	0.450820	0.351781
5	Random forest classifier	0.464000	0.453125	0.475410	0.326104
6	Neural network	0.391061	0.614035	0.286885	0.332317

Рисунок 22 — результат тестирования моделей.

В соответствии с рисунком, видно, что наилучшие показатели у алгоритмов логистической регрессии и алгоритма K-ближайших соседей, однако наибольший показатель matthews_corr_coef, который как раз показывает точную работу с несбалансированными данными, было принято решения выбрать алгоритм K-ближайших соседей как лучший. Учитывая что лучший продукт с подобными характеристиками на рынке имеет matthews_corr_coef 0.4, результат не самый плохой.

2.4. Написать нейронную сеть.

На картинке снизу — то, как выглядит нейронная сеть для бинарной классификации, и ее график ошибки.

```
model = Sequential()
model.add(Dense(512, activation='relu', input_dim=3))
model.add(Dense(512, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy')
```

Рисунок 23 — архитектура нейронной сети

В соответствии с рисунком 23, видно, что архитектура у нас простая, два полносвязных слоя по 512 нейронов, с активационной функцией relu, и один выходной слой с одним нейроном и активационной функцией сигмоиды. Функ-

ция ошибки у нас, так как классификация бинарная, бинарная кросс-энтропия. Далее, рисунок с графиками ошибки и матрицы неточностей.

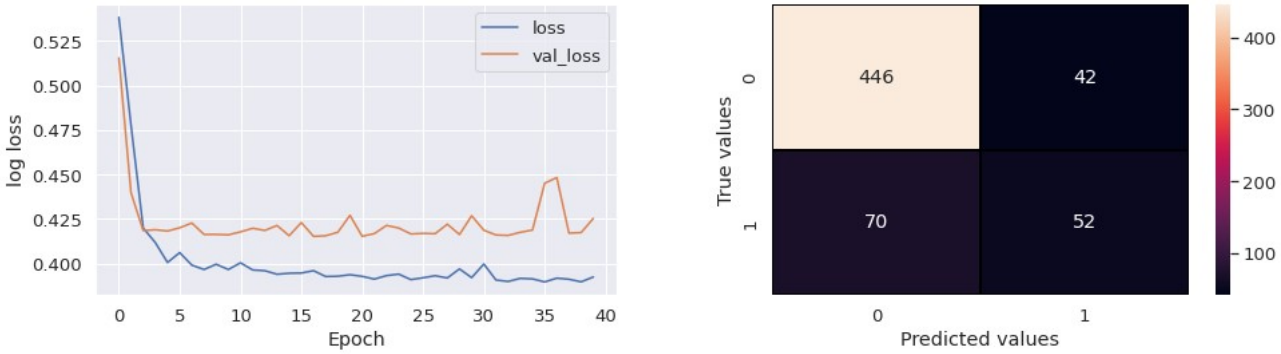


Рисунок 24 — график ошибки и матрицы неточностей нейронной сети

В соответствии с рисунком 24, можно сказать, что большое количество эпох роли не играет на уменьшение ошибки. Результаты на матрице неточностей весьма посредственные. Однако видно, что точность у нейросети немного лучше чем у большинства наших моделей.

2.5. Разработка приложения

Было разработано приложение на фреймворке Flask с графическим интерфейсом, которое имеет три поля, каждое из них должно быть щаполнено уникальной меткой для игры. После нажатия на копку «Отправить», приложение выдаст свою оценку потенциальной успешности игры, на основе этих меток. Интерфейс приложения на рисунке ниже.

Рисунок 25 — графический интерфейс приложения

В соотвествии с рисунком 25, можно увидеть графический интерфейс приложения.

Для запуска приложения, нужно запустить его командой python3 app.py, находясь в папке с приложением, после чего, по адресу в строке браузера, являющимся http://127.0.0.1:5000/index, появится приложение.

2.6. Создание удалённого репозитория и загрузка

Был создан репозиторий на Github, и там размещены файлы ВКР.

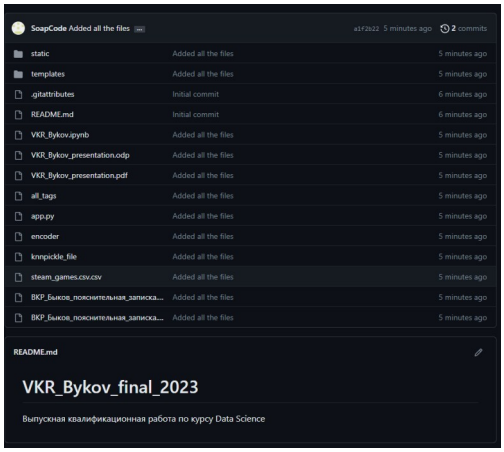


Рисунок 25 — Файлы ВКР на Github

В соотвествии с рисунком можно увидеть, что файлы ВКР размещены на Github, по ссылке: https://github.com/SoapCode/VKR_Bykov_final_2023

2.7. Список используемой литературы и веб ресурсы.

1 Amita Kapoor, Antonio Gulli, Sujit Pal - Deep Learning with TensorFlow and Keras_ Build and deploy supervised, unsupervised, deep, and reinforcement learning models, 3rd (2022, Packt Publishing)

2 Andriy Burkov - The Hundred-Page Machine Learning Book (2019, Andriy Burkov)

3 Christopher M. Bishop - Pattern Recognition and Machine Learning (2007, Springer) - libgen.lc)

4 Elder, John Fletcher_ Miner, Gary_ Nisbet, Robert_ Peterson, And - Handbook of statistical analysis and data mining applications (2018, Elsevier,Academic Press)

5 George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, Greta M - Time Series Analysis_ Forecasting and Control (2015, Wiley)

6 Generative-Deep-Learning-2nd-Edition

7 Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2019) /Aurélien Géron.- 856

8 Joel Grus - Data Science from Scratch_ First Principles with Python (2019, O'Reilly Media)

9 Jan Erik Solem - Programming Computer Vision with Python_ Tools and algorithms for analyzing images (2012, O'Reilly Media)

10 Николенко С. И., Кадурин А. А., Архангельская Е. О. - Глубокое обучение (2018, Питер)

11 Рутковская Д., Пилиньский М., Рутковский Л. - Нейронные сети, генетические алгоритмы и нечеткие системы_ Пер.с польск.И.Д.Руди

12 Stuart J. Russell, Peter Norvig - Artificial Intelligence_ A Modern Approach, Global Edition (2021, Pearson).

13 Ajitesh Kumar, Micro-average & Macro-average Scoring Metrics – Python: - Режим доступа: https://vitalflux.com/micro-average-macro-average-scoring-metrics-multi-class-classification-python/. (дата обращения 19.03.2023).

14 Arun Kumar, Thomas Parashos, Vraj Shah, How do Categorical Duplicates Affect ML? A New Benchmark and Empirical Analyses : - Режим доступа: https://adalabucsd.github.io/papers/TR_2022_CategDedup.pdf. (дата обращения 03.02.2023).

15 Creating Steamml: - Режим доступа: https://substantial.com/insights/creating-steamml-predicting-game-success. (дата обращения: 09.03.2023).

16 Dinesh Yadav, Weighted Logistic Regression for Imbalanced Dataset – Режим доступа: https://towardsdatascience.com/weighted-logistic-regression-for-imbalanced-dataset-9a5cd88e68b. (дата обращения 20.03.2023).

17 Документация по библиотеке category_encoders: - Режим доступа: https://contrib.scikit-learn.org/category_encoders/. (дата обращения 22.03.2023).

18 Документация библиотеки imbalanced-learn – Режим доступа: https://imbalanced-learn.org/stable/index.html. (дата обращения 20.03.2023).

19 24 Evaluation Metrics for Binary Classification (And When to Use Them): - Режим доступа: https://neptune.ai/blog/evaluation-metrics-binary-classification. (дата обращения 1.03.2023).

20 How Much Is the Gaming Industry Worth in 2023? [+25 Powerful Stats]: - Режим доступа: https://techjury.net/blog/gaming-industry-worth/. (дата обращения 05.03.2023).

21 Johny's Machine Learning Blog, Cost-Sensitive Decision Trees for Imbalanced Classification – Режим доступа: https://johdev.com/jupyter/2020/02/26/Decision_Tree_Imbalance.html#Decision-Trees-for-Imbalanced-Classification. (дата обращения 12.03.2023).

22 Jeff Prosise, ML & AI for Software Developers – Part 18, Binary classification with neural networks: - Режим доступа: https://www.atmosera.com/blog/binary-classification-with-neural-networks/. (дата обращения 11.03.2023).

23 Logistic Regression with Imbalanced Data: - Режим доступа: https://chandlerzuo.github.io/blog/2015/03/weightedglm. (дата обращения 09.02.2023).

24 96 Steam Statistics You Must Know: 2023 Market Share Analysis & Data: - Режим доступа: https://financesonline.com/steam-statistics/. (дата обращения 04.02.2023).

25 Stack exchange, how-f1-score-is-good-with-unbalanced-dataset: - Режим доступа: https://datascience.stackexchange.com/questions/105089/how-f1-score-is-good-with-unbalanced-dataset. (дата обращения 25.03.2023).

26 Target encoding: - Режим доступа: https://brendanhasz.github.io/2019/03/04/target-encoding.html. (дата обращения

09.02.2023).

27 Top 10 Binary Classification Algorithms [a Beginner's Guide] – Режим доступа: https://towardsdatascience.com/top-10-binary-classification-algorithms-a-beginners-guide-feeacbd7a3e2 (дата обращения 26.03.2023).

28 Turns Out The Hardest Part of Making a Game is...Everything: - Режим доступа: https://in.ign.com/absolver/164976/feature/turns-out-the-hardest-part-of-making-a-game-iseverything. (дата обращения 09.02.2023)
доступа: https://numpy.org/doc/1.22/user/index.html#user. (дата обращения: 03.06.2023).

29 Vsmolyakov, Imbalanced data: - Режим доступа: https://github.com/vsmolyakov/experiments_with_python/blob/master/chp01/imbalanced_data.ipynb. (дата обращения 026.03.2023).

30 Zach, F1 Score vs. Accuracy: Which Should You Use? – Режим доступа: https://www.statology.org/f1-score-vs-accuracy/. (дата обращения 20.03.2023).