

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**РАЗРАБОТКА МЕТОДОВ АНАЛИЗА ЗАПИСЕЙ МАТЧЕЙ В
МНОГОПОЛЬЗОВАТЕЛЬСКИХ ИГРАХ ОТ ПЕРВОГО ЛИЦА**

Автор: Глухов Евгений Дмитриевич _____

Направление подготовки: 01.03.02 Прикладная
математика и информатика

Квалификация: Бакалавр

Руководитель ВКР: Ульянов В.И., канд. техн. наук _____

Санкт-Петербург, 2020 г.

Обучающийся Глухов Евгений Дмитриевич
Группа М3437 Факультет ИТиП

Направленность (профиль), специализация
Математические модели и алгоритмы в разработке программного обеспечения

ВКР принята « ____ » _____ 20 ____ г.

Оригинальность ВКР ____ %

ВКР выполнена с оценкой _____

Дата защиты « ____ » _____ 20 ____ г.

Секретарь ГЭК Павлова О.Н. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

УТВЕРЖДАЮ

Руководитель ОП
проф., д.т.н. Парфенов В.Г. _____
« ____ » _____ 20 ____ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающийся Глухов Евгений Дмитриевич

Группа М3437 **Факультет** ИТиП

Квалификация: Бакалавр

Направление подготовки: 01.03.02 Прикладная математика и информатика

Направленность (профиль) образовательной программы: Математические модели и алгоритмы в разработке программного обеспечения

Тема ВКР: Разработка методов анализа записей матчей в многопользовательских играх от первого лица

Руководитель Ульянов В.И., канд. техн. наук, доцент ФИТиП

2 Срок сдачи студентом законченной работы до: « ____ » _____ 20 ____ г.

3 Техническое задание и исходные данные к работе

Требуется разработать и реализовать метод, который позволит вычислять реакцию игрока в многопользовательской онлайн игре от первого лица Counter-Strike: Global Offensive. Алгоритм должен анализировать записи матчей и рассматривать реакцию в ситуациях, когда игрок одной команды видит игрока другой команды. Реакцией в данном случае является время между моментом, когда противник появился на экране, до того момента, когда игрок начал направлять прицел в сторону противника с целью атаки.

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

Описание предметной области и существующих решений для анализа записей матчей. Разработка и реализация алгоритма, позволяющего анализировать запись матча с целью получения реакции игрока. Разработка и получение тестовых данных, для проверки работоспособности разработанного алгоритма.

5 Перечень графического материала (с указанием обязательного материала)

Графические материалы и чертежи работой не предусмотрены

6 Исходные материалы и пособия

7 Дата выдачи задания « ____ » _____ 20 ____ г.

Руководитель ВКР _____

Задание принял к исполнению _____ « ____ » _____ 20 ____ г.

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся: Глухов Евгений Дмитриевич

Наименование темы ВКР: Разработка методов анализа записей матчей в многопользовательских играх от первого лица

Наименование организации, в которой выполнена ВКР: Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: Разработка и реализация метода, который позволит вычислять реакцию игрока в многопользовательской онлайн игре от первого лица Counter-Strike: Global Offensive.

2 Задачи, решаемые в ВКР:

- а) Разработка признаков и их извлечение из записей матча.
- б) Разработка и реализация алгоритма для вычисления реакции игрока.
- в) Тестирование алгоритма на полученных данных

3 Число источников, использованных при составлении обзора: 17

4 Полное число источников, использованных в работе: 35

5 В том числе источников по годам:

Отечественных			Иностранных		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	1	3	2	14

6 Использование информационных ресурсов Internet: да, число ресурсов: 15

7 Использование современных пакетов компьютерных программ и технологий:

Пакеты компьютерных программ и технологий	Раздел работы
Компьютерная игра Counter-Strike: Global Offensive	2.1, 2.3
Среда разработки Visual Studio Code	2.2, 2.4, 2.5
Среда разработки PyCharm	2.6, 3.1, 3.2, 3.3, 3.4
Язык программирования Golang	2.2, 2.4, 2.5
Язык программирования Python 3	2.6, 3.1, 3.2, 3.3, 3.4
Пакет с алгоритмами машинного обучения Scikit-learn	2.6, 3.2, 3.3, 3.4

8 Краткая характеристика полученных результатов

Описаны алгоритмы извлечения данных из записей матчей, также разработаны признаки, описывающие состояние рассматриваемого случая. Разработаны алгоритмы, позволяющие, на основе полученных данных, вычислить значение времени реакции. Проведено их тестирование, которое показывает, что применение методов машинного обучения улучшает качественную работу алгоритма.

9 Гранты, полученные при выполнении работы

Нет

10 Наличие публикаций и выступлений на конференциях по теме выпускной работы

Нет

Обучающийся Глухов Е. Д. _____

Руководитель ВКР Ульянцев В.И. _____

« _____ » _____ 20__ г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ОЦЕНКИ СПОСОБНОСТЕЙ ИГРОКОВ	9
1.1. Описание используемой видеоигры	9
1.2. Внутриигровые статистические данные	10
1.3. Записи матчей	11
1.4. Статистические показатели основанные на комбинации внутриигровых данных	11
1.4.1. Rating 1.0	11
1.4.2. Rating 2.0	12
1.5. Оценка способностей игроков на основе результатов матчей	13
1.6. Использование методов машинного обучения для предсказания способностей игрока	15
1.7. Постановка цели и задач ВКР	16
Выводы по главе 1	16
2. ОПИСАНИЕ РАЗРАБОТАННЫХ АЛГОРИТМОВ	18
2.1. Описание ситуации и реакции в ней	18
2.2. Извлечение данных	19
2.3. Алгоритм разметки данных	19
2.4. Разработка и реализация наивного алгоритма, позволяющего вычислить время реакции игрока	20
2.5. Извлечение признаков для объектов	25
2.6. Улучшение алгоритма, путем применения методов машинного обучения	25
2.6.1. Использование алгоритмов, решающих задачу классификации	26
2.6.2. Объединение машинного обучения и наивного алгоритма	27
Выводы по главе 2	28
3. ТЕСТИРОВАНИЕ И ВАЛИДАЦИЯ АЛГОРИТМА И ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ	29
3.1. Тестирование наивного алгоритма	29
3.2. Тестирование алгоритмов решающих задачу классификации	30
3.3. Применение метода отбора признаков	37

3.4. Тестирование алгоритма объединения наивного решения и решения, полученного методом машинного обучения	45
3.5. Общая сравнительная таблица.....	48
Выводы по главе 3	49
ЗАКЛЮЧЕНИЕ	51
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	53
ПРИЛОЖЕНИЕ А. Описание извлеченных признаков для каждого тика .	56
ПРИЛОЖЕНИЕ Б. Признаки, полученные путем применения методов отбора	61

ВВЕДЕНИЕ

Киберспорт – это вид спорта, в котором соревнования проводятся на основе видеоигр [1]. По популярности он не уступает многим привычным спортивным дисциплинам таким, как футбол, хоккей, волейбол и другим. Прямые трансляции киберспортивных соревнований собирают миллионы пользователей в сети, турниры проводятся на больших стадионах, а призовые фонды составляют десятки миллионов долларов. Основой большой популярности является то, что каждый зритель, как правило, это такой же человек, который играет в видеоигры. Он имеет очень хорошее понимание всех правил и нюансов игры, что позволяет лучше понимать все действия, которые киберспортсмены предпринимают во время матча, в отличие от других спортивных дисциплин, где, как правило, зрители редко занимаются тем видом спорта, за которым наблюдают.

Киберспортивные соревнования обычно можно разделить двумя способами: по видеоигре, которая лежит в основе турнира, и по характеру, где выступает либо один игрок, либо команда. Профессиональная киберспортивная команда, как правило, состоит из игроков, капитана, тренера и менеджера. Также, часто в состав входят: психолог, массажист, повар и другой персонал. Они позволяют киберспортсменам сконцентрироваться на игре и не думать о других проблемах, которые решают люди, находящиеся рядом с ними. Уровень подготовки не хуже, чем в футболе, хоккее и других видах спорта.

Основная часть подготовительной работы в любом виде спорта лежит на тренере. Он просматривает матчи соперника, анализирует его игру и придумывает стратегию, которая приведет к победе. Также тренер уделяет внимание ошибкам, которые совершают его игроки и работает над их устранением. Немаловажную роль в его работе играет статистика. Она дает более объективную информацию, чем тренер получает исходя из своих наблюдений. Эти данные, как правило, получаются ручным трудом или с помощью различных датчиков. Например, компания InStat, которая занимается спортивной аналитикой, имеет штат сотрудников, которые просматривают матчи и фиксируют статистические показатели [2]. Это очень трудоемкий процесс, который требует высокой концентрации и понимания игры, что делает дорогой ее получение. Но в киберспорте этот процесс

удобнее, так как видеоигра является компьютерной программой, которая имеет доступ ко всем данным во время и после матча. Это позволяет быстро получать нужную информацию.

В киберспорте все статистические данные предназначены, в основном, для оценки способностей игрока – скилла. Существует несколько различных методов для оценки этого показателя. Для каждой игры он имеет различное значение и алгоритм вычисления. Как правило, он базируется на количестве сделанных убийств, смертей и помощи союзникам. То есть он основан на том, какой вклад в матч вносит игрок. Например, отношение совершенных убийств к количеству смертей является одним из базовых показателей, который входит во многие более сложные оценки. Если этот показатель меньше единицы, то это значит, что игрок чаще умирает, чем убивает, тем самым во многих ситуациях дает численное преимущество команде соперника. Если больше либо равно единицы, то этот игрок делает большой вклад, помогая команде получить численное преимущество, что повышает шансы на победу. Все подобные оценки ничего не говорят о физическом состоянии игроков. Например, в случае проигрышного матча, по этим показателям трудно понять в чем именно была проблема: в тактике, в физических кондициях спортсмена или простом везении. В современных спортивных дисциплинах, таких как футбол, хоккей, баскетбол и других, этому уделяется большое внимание. Так как от физического состояния спортсменов зависит исход матча. Над этим работает целый тренерский штаб: тренеры по физической подготовке, врач-физиотерапевт, который следит за физическими показателями и решает все возникающие с ними проблемы.

Существует много различных физических показателей, например, выносливость, скорость, ловкость, но в киберспорте основным является реакция. Особенно это очень важно в шутерах, где нужно наводить прицел, зачастую с помощью компьютерной мыши, на противника и выстреливать. Вследствие чего, было решено разработать метод для оценки реакции игроков в видеоигре Counter-Strike: Global Offensive [3].

В основу работы была взята именно эта игра, потому что она является одной из самых популярных многопользовательских онлайн игр от первого лица в жанре шутер, а также имеет возможность записывать матчи, которые впоследствии будут использоваться для анализа.

В первой главе рассмотрены: видеоигра, на основе которой будет проводиться работа, структура записи матча, а также существующие методы оценки способностей игроков как в Counter-Strike: Global Offensive, так и в других играх. Во второй главе описаны методы извлечения данных, а также алгоритмы для вычисления времени реакции. В третьей главе описаны результаты тестирования всех разработанных алгоритмов и сделаны выводы на их основе.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ОЦЕНКИ СПОСОБНОСТЕЙ ИГРОКОВ

Скилл – это показатель, который определяет уровень мастерства игроков [4]. На его основе проще всего сравнивать способности игроков между собой. Также он часто используется для подбора равных по силе команд, чтобы у них была одинаковая вероятность победы. Скилл может выражаться числом, целым или вещественным, или классом, например, новичок или опытный. Все существующие методы оценивания способностей игрока основаны на внутриигровом вкладе. И они лишь косвенно учитывают физические показатели. Например, если представить ситуацию, при которой два игрока одновременно увидели друг друга, то вероятность выиграть в этой дуэли больше у того игрока, который быстрее среагирует. Давайте рассмотрим видеоигру, на основе которой будет проводиться исследование, а также те способы и оценки, которые применяются в настоящее время для оценивания способностей игрока, а также способы их получения.

1.1. Описание используемой видеоигры

Counter-Strike: Global Offensive – это многопользовательская командная онлайн игра в жанре тактический шутер от первого лица [5]. Эта игра является очень популярной, так как серия игр Counter-Strike существует с 1999 года, в которую играли несколько поколений.

В данной игре есть много режимов, но самым популярным является состязательный, который присутствует во всех вышедших частях серии. Матч, в этом режиме, состоит из нескольких раундов. В основное время игры проводится не более 30. Для победы нужно выиграть 16 раундов, после чего игра завершается. Если в основное время матч закончился с равным счетом, то есть каждая команда выиграла по 15 раундов, то играют дополнительные шесть раундов, где нужно выиграть четыре из них. При повторении ничейного результат играется еще шесть и так до тех пор, пока одна из команд не выиграет. Матч проходит между двумя командами из пяти человек. Одна из команд называется спецназ, другая террористы. Для победы нужно убивать противников используя огнестрельное и метательное оружия. У террористов есть бомба, после установки которой начинается отсчет до ее взрыва. В таком случае спецназ должен ее обезвредить. Если они это делают, то одерживают победу в раунде. Иначе раунд переходит террористам. Также матч ограничен

по времени и если команда террористов не успевает установить бомбу или убить противников, то победу в раунде одерживает спецназ.

1.2. Внутриигровые статистические данные

Видеоигра вычисляет некоторые статистические данные в реальном времени и предоставляет к ним доступ. Они позволяют киберспортсменам во время матча оценивать ситуацию и делать выводы по возможной смене тактики игры. Такими показателями являются количество убийств, количество смертей, процент попадания в голову, отношение количества убийств к количеству сыгранных раундов, средний урон за раунд и многие другие. На рисунке 1 показано внутриигровое окно с основными статистическими показателями. Это неплохие метрики для оценки способностей в большинстве случаев. Но рассмотрим такую ситуацию, одна из команд проиграла в матче. Но показатель количества убийств к количеству смертей у одного из игроков этой команды лучше, чем у любого игрока победившей команды. Такое возможно, если игрок старался просто не умирать и прятался от противников, вследствие чего проигрывал раунд. При этом такое поведение не является признаком хорошей игры. Что нельзя отследить по показателю отношения количества убийств к количеству смертей.



Рисунок 1 – Окно в игре, отображающее основные внутриигровые показатели

Также подобные данные не позволяют получить оценку каких-либо физических способностей игрока, так как они основаны на внутриигровом влиянии.

1.3. Записи матчей

Данные, описанные в прошлой главе, можно получить как во время игры, так и после нее, если эта игра записывается. Демо файл [6] хранит информацию о событиях, произошедших в матче, что позволяет просматривать его. Это происходит следующим образом, видеоигра считывает произошедшие события и моделирует их.

Демо состоит из заголовка и последовательности фреймов, в которых хранятся команды, описывающие происходящие события. Заголовок содержит данные о названии сервера и клиента, карты, на которой проходит матч, длительности матча в секундах, количестве фреймов, количестве тиков. Тики – это минимальная единица времени в демо формате. Как правило, в одной секунде либо 64, либо 128 тиков. Фрейм хранит данные о номере фрейма на сервере и на клиента, количестве пакетов в фрейме и сами пакеты. Пакет содержит номер команды, тик, в который эта команда исполняется и сами данные для команды.

Для удобного извлечения данных из демо файла, разработчики опубликовали собственный синтаксический анализатор [7], который позволяет получать последовательность команд из записи. Но энтузиасты реализовали более удобные средства [8–10], которые позволяют получать данные в более понятном и удобном формате. Были протестированы различные парсеры и лучшим оказался синтаксический анализатор на языке программирования Golang [11], так как он показал лучшую производительность и удобство использования среди других.

1.4. Статистические показатели основанные на комбинации внутриигровых данных

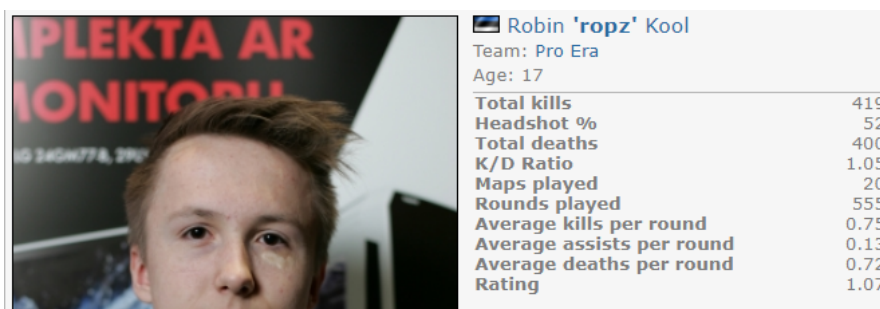
Показатели такого вида базируются на внутриигровых данных и позволяют лучше оценить способности, так как происходит комбинирование простых данных.

1.4.1. Rating 1.0

Одним из популярных показателей является rating 1.0 [12] от сайта hltv.org [13]. Он основан на комбинации трех показателей: kill rating, survival rating и rounds with multiple kills rating. Kill rating – это отношение KPR к AverageKPR. KPR (Kills per round) – это отношение количества совершенных

убийств к количеству сыгранных раундов. AverageKPR – это среднее значение KPR, которое ожидается от игрока. Survival rating – отношение SPR к AverageSPR. SPR (Survived rounds per round) – это отношение разности количества раундов и количества смертей к количеству раундов. AverageSPR – это среднее значение SPR, которое ожидается от игрока. Rounds with multiply kill rating - отношение RWMK к AverageRWMK. RWMK (Rounds with multiply kills) – это отношение взвешанной суммы количества раундов с несколькими убийствами к количеству сыгранных раундов. За один раунд игрок может совершить от нуля до пяти убийств. Раунд с нулем убийств имеет вес ноль, с одним убийством – один, с двумя убийствами – четыре, с тремя – девять, с четырьмя – 16, с пятью – 25. AverageRWMK – среднее значение RWMK, которое ожидается от игрока. Итого, рейтинг вычисляется по следующей формуле:

$$\text{Rating} = \frac{\text{KillRating} + 0.7 \cdot \text{SurvivalRating} + \text{RoundsMultipleKillsRating}}{2.7}.$$



Robin 'ropz' Kool	
Team: Pro Era	
Age: 17	
Total kills	419
Headshot %	52
Total deaths	400
K/D Ratio	1.05
Maps played	20
Rounds played	555
Average kills per round	0.75
Average assists per round	0.13
Average deaths per round	0.72
Rating	1.07

Рисунок 2 – Статистика киберспортсмена Робина «ropz» Кула в 2017 году на сайте hltv.org

На рисунке 2 можно увидеть статистику киберспортсмена Робина «ropz» Кула на сайте hltv.org в 2017 году. Здесь указаны, описанные выше, внутриигровые статистические показатели, а также rating 1.0, который является более сложным за счет комбинирования стандартных метрик, что позволяет давать оценку лучше, чем внутриигровые статистические данные, но он также не позволяет оценить физические способности игрока.

1.4.2. Rating 2.0

Rating 2.0 [14] является новой версией вышеописанного рейтинга, в который были добавлены новые показатели, такие как количество первых

убийств в раунде, количество урона, наносимое за раунд, и многие другие. Он также основан на средних ожидаемых значениях, описанных выше, величин и показывает насколько способности игрока лучше или хуже, чем ожидалось. Новые показатели, вошедшие в этот рейтинг, позволяют оценивать вклад игрока не только за счет совершаемых убийств, но и за счет наносимого урона, первого убийство в раунде, которое дает численное преимущество, что делает этот рейтинг еще честнее чем предыдущая версия. На рисунке 3 приведена статистика киберспортсмена Робина «ropz» Кула на сайте hltv.org. Указаны значения второй версии рейтинга, а также некоторые другие показатели, которые входят в его основу.

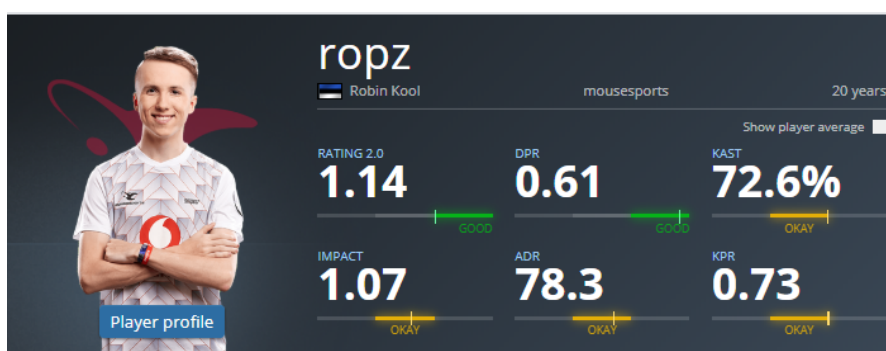


Рисунок 3 – Статистика киберспортсмена Робина «ropz» Кула в 2020 году на сайте hltv.org

Его создатели не раскрывают зависимости, между описанными величинами, но утверждают, что он является более корректным в сравнении способностей двух игроков.

Улучшение предыдущей версии рейтинга, также не позволяют получить оценку каких-либо физических показателей игрока.

1.5. Оценка способностей игроков на основе результатов матчей

Рейтинг Эло [15] – это один из первых методов для оценки способностей игроков, который был разработан Арпадом Эло для шахматистов. Он заключается в том, что игра имеют нулевую сумму, то есть сколько один выиграет – столько другой проиграет. Рейтинг каждого игрока меняется по результатам матча. Присваивается либо победа, либо поражение. Причем количество единиц рейтинга, которое потеряет проигравший и соответственно получит победивший, зависит от их значений до матча. Если играют два игрока с разными рейтингами, то при победе аутсайдера, он получит больше очков,

чем теряет за поражение. Это связано с тем, что у сильного игрока больше шансов на победу и в этом не будет ничего удивительного, в отличии от поражения.

Рейтинг Эло также применяется и для того, чтобы оценивать игроков в Counter-Strike: Global Offensive. Например, его использует игровая платформа Faceit [16] – это независимая игровая платформа, которая предоставляет игрокам сервера и не только для более комфортной игры. Каждый зарегистрированный игрок имеет свой skill level это число от единицы до десяти. Этот показатель определяет уровень игроков, против которых вы будете соревноваться. Skill level зависит от значения рейтинга Эло, при изменении которого skill level также будет меняться. Здесь применяется метод, описанный выше, но с той поправкой, что Counter-Strike: Global Offensive – командная игра, и у каждого ее члена рейтинг изменяется на одинаковое значение. На рисунке 4 приведена статистика киберспортсмена команды Team Spirit Бориса «magixx» Воробьева, который стал лучшим игроком мая 2020 года в закрытой лиге для профессионалов на платформе FACEIT. Здесь указаны skill level, эло рейтинг и некоторые стандартные статистические показатели.

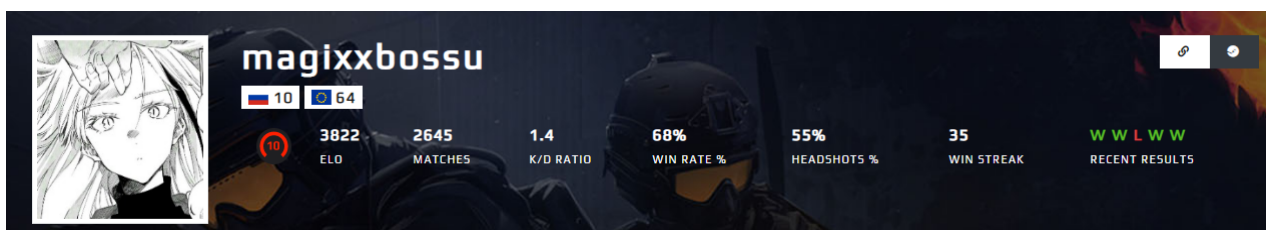


Рисунок 4 – Статистика лучшего игрока мая 2020 года, выступающего в закрытой лиге на платформе FACEIT

Этот рейтинг пользуется популярностью, поскольку используется на платформе FACEIT, где ежедневно играет около 200 000 человек, для формирования равных команд, которые будут играть друг против друга, но у него есть несколько недостатков. Один из них – это то, что рейтинг отдельно взятого игрока очень сильно зависит от команды. Если она проиграет, то как бы ты хорошо не играл, твой рейтинг все равно уменьшится. Что, на мой взгляд, не является полностью честным по отношению к игроку, так как не рассматривается даже тот вклад, который он внес в игру. А также этот рейтинг не позволяет оценить физические способности игрока, но он довольно прост

в реализации, так как не требует доступа к внутриигровым данным, а также заставляет игроков играть командой, так как при поражении все теряют очки.

1.6. Использование методов машинного обучения для предсказания способностей игрока

Все рассмотренные до этого момента способы описывают простую зависимость между некоторыми статистическими данными. Для получения более сложных зависимостей можно воспользоваться методами машинного обучения.

Рассмотрим следующую работу [17], в которой автор рассказывает о быстром способе, позволяющем предсказывать способности игрока на основе входных данных, то есть данных с мыши и клавиатуры, а также различным внутриигровым событиям. Скилл рассматривается не как число, а как класс. Автор разделяет игроков на 4 категории: новичок, любитель, продвинутый, профессионал.

Эта работа строится на предположении о том, что взаимодействие опытного игрока с элементами управления отличается от того, как это делает новичок. Игрок со стажем использует устройства ввода данных более сложным образом. То есть входные данные новичка являются более предсказуемыми, что позволяет использовать алгоритмы сжатия данных, такие как алгоритм Хаффмана [18] и LZW [19], или подсчета энтропии для вычисления сложности входных данных, и в последствии оценки способностей игрока. Если данные хорошо сжимаются, то между ними простая зависимость и этого игрока можно считать новичком. В противном случае, зависимость сложная и игрока можно считать опытным.

Признаки, описывающие данные, были разбиты на несколько групп в зависимости от устройства, типа и контекста. Первая группа содержит данные о мыши клавиатуре, вторая – данные о передвижении, сложности ввода и частоте происходящих событий с устройствами ввода, а третья – собранные данные, зависящие и независимые от количества сыгранных матчей или знаний особенностей игры, имеющиеся у опытных игроков.

Для предсказания рейтинга игрока используется случайный лес [20]. Он обучался на различных, описанных выше, группах и лучшие показатели были получены при использовании признаков на основе данных клавиатуры,

что говорит о том, что они лучше коррелируют с навыком игрока, а данные, полученные с мыши, оказались самыми худшими, потому что они зашумлены сильнее, так как игрок с помощью мыши осматривается и может совершать действия, которые не нужны при оценке способностей игрока.

Так же подобный алгоритм сложно использовать в реальном мире, так как данное исследование проводилось на данных, полученных с мыши и клавиатуры, которые можно получить только устанавливая дополнительно программное обеспечение на компьютер игрока, или встраивая соответствующий код в видеоигру, что невозможно, так как все популярные игры в основном имеют закрытый код.

Этот метод отличается от остальных тем, что он основан не на внутриигровом влиянии, а на входных данных игрока, но эти данные используются для оценки скилла с точки зрения внутриигрового вклада, что также не позволяет оценивать физические способности.

1.7. Постановка цели и задач ВКР

Целью работы является разработка и реализация алгоритма для вычисления реакции игрока в многопользовательской онлайн игре от первого лица Counter-Strike: Global Offensive, на основе записей матчей. Алгоритм будет рассматривать ситуации, в которых два игрока начинают видеть друг друга. И реакцией в такой ситуации является время, которое понадобится игроку от начала такой ситуации, до момента, когда игрок начнет перемещение прицела в сторону противника.

Для достижения поставленной цели необходимо решить следующие задачи:

- а) Разработка и реализация алгоритма для извлечения данных из демо файла, хранящего запись матчей.
- б) Извлечение данных и разработка признаков, которые потребуются для вычисления реакции.
- в) Разработка и реализация алгоритма для вычисления реакции.
- г) Тестирование алгоритма на полученных данных.

Выводы по главе 1

В данной главе проведен обзор существующих способов оценивания способностей игрока. Все описанные методы не позволяют оценить

физические способности игрока, а лишь внутриигровой вклад. Некоторые из них, примитивные и не позволяют получить правильную информацию. Также отмечены некоторые проблемы этих методов, что приводит к необходимости разработки и реализации нового решения.

ГЛАВА 2. ОПИСАНИЕ РАЗРАБОТАННЫХ АЛГОРИТМОВ

Для разработки алгоритма вычисления реакции игрока, нужно в первую очередь описать ситуации, которые будут рассмотрены, определить что именно будет считаться реакцией, понять, какие данные можно получить из демо файла и какие из них будут полезны.

2.1. Описание ситуации и реакции в ней

Наиболее интересными, для вычисления реакции, являются случаи, когда у игрока появляется противник на экране, и спустя время человек, за которым ведется наблюдение, начинает перемещать прицел в сторону противника с целью его атаки. Они интересны тем, что отражают время, нужное игроку для реакции на случившееся событие, а именно появление противника. Разберем более подробно на примере.

Рассмотрим случай, когда игрок выходит из-за угла, где его ожидает противник. Для большей наглядности данная ситуация был схематично изображена. На рисунках синим цветом помечен игрок, от лица которого ведется наблюдение, красным – противник, пунктирная линия – направление прицела, сплошная линия – стена.

На рисунке 5а изображен момент времени, когда игрок еще не видит противника и начинает выходить из-за угла.

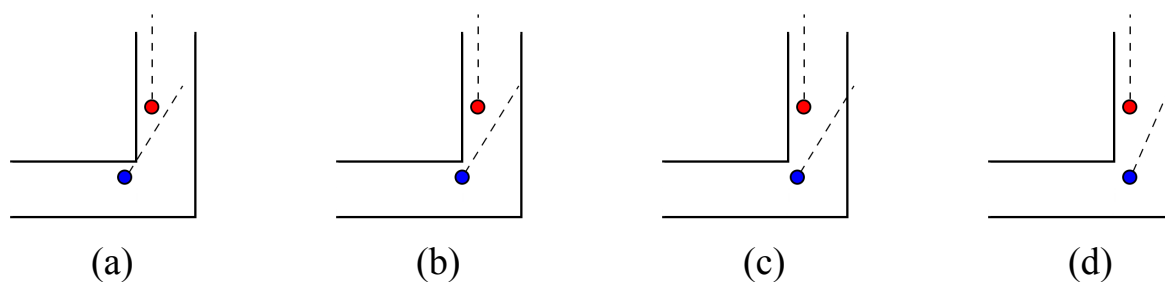


Рисунок 5 – Пример рассматриваемой ситуации, вид сверху. (a) Атакующий игрок начинает выходить из-за угла; (b) В поле зрения атакующего игрока появился противник; (c) Атакующий игрок перемещается не замечая противника; (d) Атакующий игрок заметил противника и изменил положение прицела

Далее он перемещается вперед, не изменяя направления прицела, например, потому что он не ожидает увидеть противника, либо проверяет позицию, в которой, в данный момент, не находится оппонент. И в это время, на его экране появляется противник и человеческому мозгу нужно время,

чтобы обработать эту информацию и начать реагировать. Этот момент будем считать началом отсчета времени реакции. Он изображен на рисунке 5b. Последующее перемещение без изменения положения прицела представлено на рисунке 5с.

После того, как прошло некоторое время, которое понадобилось для обработки ситуации, игрок начинает перемещать прицел в сторону противника, чтобы атаковать его. Данный случай изображен на рисунке 5d. В этот момент игрок отреагировал.

Поэтому реакцией будем считать время, прошедшее от момента изображенного на рисунке 5b до момента представленного на рисунке 5d.

2.2. Извлечение данных

Для извлечения данных нужно сначала найти все ситуации, в которых нужно будет вычислить реакцию. Для этого будем последовательно, на каждом тике, рассматривать игроков обеих команд и проверять, кого из противников он видит. Парсер имеет методы, которые позволяют узнать такую информацию. Далее находим все последовательности тиков максимально возможной длины, на которых один игрок видит другого. Тем самым, получаем для каждого киберспортсмена отрезки времени, когда он видит противника.

К сожалению, парсер не дает возможности получить реакцию игрока, поэтому воспользуемся вышеописанным алгоритмом для того, чтобы извлечь все рассматриваемые ситуации и вручную разметим каждую из них, то есть для каждой ситуации найдем реакцию.

2.3. Алгоритм разметки данных

Разметка извлеченных ситуаций происходила по следующему алгоритму. Запись матча запускается в игре. Далее переходим к просмотру каждого рассматриваемого случая. Последовательно просматривается каждый тик, начиная с момента когда противник появился на экране выбранного игрока, и ищем первый тик, в который наблюдаемый нами человек начнет перемещение прицела, которое приведет к атаке противника. Но бывают моменты, когда наблюдаемый игрок выходя из-за угла хочет заранее проверить позицию, которую часто занимает противник, при этом оппонент, в данном случае, не занимал ее, но все равно находится в зоне

видимости игрока. В таком случае, нужно дождаться, когда игрок закончит маневр с проверкой часто занимаемой позиции и продолжить просмотр по описанному выше алгоритму. Каждая такая ситуации рассматривается от лица каждого игрока, который принимает в ней участие, но лишь в случае, если он видел противника.

Тем самым, получаем время в тиках между моментом когда игрок появился на экране, до того как он смог обработать это событие и начал предпринимать действия, например, перемещение прицела с целью атаки.

2.4. Разработка и реализация наивного алгоритма, позволяющего вычислить время реакции игрока

Реакцией считаем время, которое прошло с того момента, как игрок увидел противника, до того момента, как игрок начал передвигать прицел в сторону противника. Можно считать, что игрок до того момента, как он увидел противника, мог не перемещать прицел, либо перемещать ее с небольшой скоростью или ускорением. Но как только он увидит противника и отреагирует, произойдет резкое изменение положения прицела, то есть скорость и ускорение будут иметь большее значение, чем было до момента реакции. На этой гипотезе и будем строить алгоритм.

В демо файле, положение прицела определяется двумя числами: отклонением по вертикали и горизонтали. Единицей измерения являются градусы. По вертикали самая верхняя точка это 90 градусов, нижняя 270 градусов. По горизонтали от 0 до 360 градусов. На основе этих данных будем получать значение скорости и ускорения перемещения прицела за один тик. Обозначим за d_x – отклонение по горизонтали, а d_y – по вертикали. Можно считать, что перемещение прицела – это перемещение точки по поверхности сферы, поэтому перейдем к сферической системе координат. Каждая точка в ней состоит из трех компонент (r, θ, ϕ) , где r – расстояние от центра системы координат до точки, θ – отклонение по вертикали от самой верхней точки, ϕ – отклонение по горизонтали. Расстояние до точки будем считать равным единице, так как прицел перемещается по поверхности сферы. Отклонение прицела по горизонтали эквивалентно координате ϕ . Угол по вертикали

преобразуем следующим образом:

$$\theta = \begin{cases} 90 - d_y & 0 \leq d_x \leq 90 \\ 90 + (360 - d_y) & \text{иначе} \end{cases}.$$

Скорость будем вычислять как разницу между положениями прицела за один тик. Положение прицела в первый момент времени имеет координаты (r, θ_1, ϕ_1) , через один тик – (r, θ_2, ϕ_2) . Не забываем, что описанные точки находятся на сфере радиуса $r = 1$. Изменением положения прицела будем считать угол между двумя векторами. Первый – от начала координат до точки (r, θ_1, ϕ_1) , второй – от начала координат до точки (r, θ_2, ϕ_2) . Обозначим их как a и b соответственно. Угол будем находить из скалярного произведения этих векторов:

$$a \cdot b = |a| |b| \cos \alpha = a_x b_x + a_y b_y + a_z b_z.$$

Для нахождения координат вектора нужно перейти из сферической системы координат в декартову. Делать это будем по следующим формулам:

$$\begin{cases} x = r \sin \theta \cos \phi \\ y = r \sin \theta \sin \phi \\ z = r \cos \theta \end{cases}.$$

Имея в виду то, что модуль векторов эквивалентен r , которое равно 1, нужный нам угол между векторами будет вычисляться следующим образом:

$$\begin{aligned} \alpha &= \arccos (\sin \theta_1 \cos \phi_1 \sin \theta_2 \cos \phi_2 + \sin \theta_1 \sin \phi_1 \sin \theta_2 \sin \phi_2 + \cos \theta_1 \cos \theta_2) \\ &= \arccos (\sin \theta_1 \sin \theta_2 (\cos \phi_1 \cos \phi_2 + \sin \phi_1 \sin \phi_1) + \cos \theta_1 \cos \theta_2) \\ &= \arccos (\sin \theta_1 \sin \theta_2 \cos (\phi_1 - \phi_2) + \cos \theta_1 \cos \theta_2). \end{aligned}$$

Так как скорость – изменение положения прицела за один тик, и за один тик прицел перемещается на угол α , то будем считать, что скорость – $v = \alpha$, где α – угол в радианах.

Ускорение за один тик, это изменение скорости за один тик. Будет выражаться в следующем виде: $a = v_2 - v_1$, где a – ускорение, v_2 – скорость в один тик времени, v_1 – скорость в предыдущий тик времени.

Исходя из соображений гипотезы, можно сделать следующие выводы, что скорость и ускорение в момент реакции игрока, должны быть больше, чем среднее их значение за несколько тиков до этого. Для реализации этой идеи следует извлечь данные об изменении скорости и ускорения игрока на каждом тике, для всех рассматриваемых моментов. Далее для каждого тика, вычислим среднюю скорость и среднее ускорение на последних трех, пяти и десяти тиках. Были выбраны именно такие значения для того, чтобы лучше отслеживать изменение показателей скорости и ускорения. Большой промежуток будет лучше сглаживать моменты резкого изменения показателей. Далее реализовали алгоритм исходя из соображений гипотезы. Псевдокод алгоритма представлен на листинге 1.

Данный алгоритм на вход принимает два массива одинаковой длины, количество тиков, на протяжении которых будем высчитывать средние значения, а также максимальные значения разницы между текущими скоростью и ускорением, и средними их значениями. В первом массиве хранятся скорости на каждом тике, во втором ускорения. Алгоритм обходит последовательно каждый тик и вычисляет среднюю скорость и ускорение. И если абсолютная разница между текущей скоростью и ее средней превышает максимальную разницу для нее, а также абсолютная разница между текущим ускорением и ее средним превышает максимальную разницу для нее то в этот тик произошла реакция – это и является ответом.

Численные значения, количества тиков для вычисления среднего значения и максимальных значений разницы, были получены на основе просмотра нескольких размеченных ситуаций. Для количества тиков было выбрано значение – десять. Это не очень большое значение и при этом не очень маленькое, что не позволяет сглаживать пиковые показатели скорости или ускорения. В качестве максимальных значений разницы были выбраны, такие минимальные значения, которые удовлетворяют большинству размеченных ситуаций. Такими значениями оказались 0,002 для скорости и 0,0005 для ускорения.

Листинг 1 – Псевдокод наивного алгоритма, позволяющего вычислить время реакции игрока

```

function CalculateReaction(speeds, accelerances, lastTicks, maxSpeedDiff,
maxAcceleranceDiff)
    ticksCount  $\leftarrow$  Len(speed)
    for i  $\leftarrow$  [1...ticksCount] do
        avrSpeed  $\leftarrow$  0
        for j  $\leftarrow$  [Max(0, i – lastTicks)..i – 1] do
            avrSpeed  $\leftarrow$  avrSpeed + speeds[j]
        end for
        avrSpeed  $\leftarrow$  avrSpeed / (i – Max(0, i – lastTicks))
        avrAccelerance  $\leftarrow$  0
        for j  $\leftarrow$  [Max(0, i – lastTicks)..i – 1] do
            avrAccelerance  $\leftarrow$  avrAccelerance + accelerances[j]
        end for
        avrAccelerance  $\leftarrow$  avrAccelerance / (i – Max(0, i – lastTicks))
        if Abs(avrSpeed – speeds[i]) > maxSpeedDiff then
            if Abs(avrAccelerance – accelerances[i]) > maxAcceleranceDiff
then
                return i + 1
            end if
        end if
    end for
    return i + 1
end function

```

Теперь проведем тестирование данного алгоритма на небольшом количестве размеченных данных. Оно выявило одну неточность, а именно неправильная обработка моментов, в которых игрок на протяжении рассматриваемой ситуации меняет оружие с огнестрельного на холодное или метательное. Рассмотрим конкретный случай, когда игрок перемещается по карте, держа в руках нож, с ним игровой персонаж перемещается быстрее, чем с любым другим видом оружия. И при подходе на позицию, прыжками начинает проверять подошел ли оппонент. При этом двигая прицел лишь в целях получения информации о нахождении противника. Как только его оппонент появится в поле зрения, подобные перемещения прицела могут расцениваться как реакция, которую мы хотим вычислить. Простым решением было бы отбросить все ситуации, в которых игрок пользуется не огнестрельным оружием. Но такие моменты нельзя опускать, так как перед

началом реакции игрок мог сменить оружие, а уже потом начать реакцию. Например он мог идти на позицию держа в руках нож, так как перемещение с ним происходит быстрее, и при подходе на позицию сменил оружие, но противник появился чуть раньше, чем игрок начал смену. Для решения подобных ситуаций будем на каждом тике хранить тип оружия, которое находится в руках у игрока, и добавим дополнительное условие, при поиске тика реакции, о необходимости огнестрельного оружия. Псевдокод алгоритма представлен на листинге 2.

Листинг 2 – Псевдокод наивного алгоритма, позволяющего вычислить время реакции игрока с улучшениями

```

function CalculateReaction(speeds, accelerances, weaponTypes, lastTicks,
maxSpeedDiff, maxAcceleranceDiff)
    ticksCount  $\leftarrow$  Len(speed)
    for i  $\leftarrow$  [1...ticksCount] do
        avrSpeed  $\leftarrow$  0
        for j  $\leftarrow$  [Max(0, i – lastTicks)..i – 1] do
            avrSpeed  $\leftarrow$  avrSpeed + speeds[j]
        end for
        avrSpeed  $\leftarrow$  avrSpeed / (i – Max(0, i – lastTicks))
        avrAccelerance  $\leftarrow$  0
        for j  $\leftarrow$  [Max(0, i – lastTicks)..i – 1] do
            avrAccelerance  $\leftarrow$  avrAccelerance + accelerances[j]
        end for
        avrAccelerance  $\leftarrow$  avrAccelerance / (i – Max(0, i – lastTicks))
        if Abs(avrSpeed – speeds[i]) > maxSpeedDiff then
            if Abs(avrAccelerance – accelerances[i]) > maxAcceleranceDiff
then
                if weaponTypes[i] == 1 then
                    return i + 1
                end if
            end if
        end if
    end for
    return i + 1
end function

```

Тестирование также показало, что алгоритм дает очень точные результаты в простых ситуациях, например, когда игрок не передвигает прицел пока не увидит противника, при этом часто дает плохие и неточные

результаты, в сложных ситуациях, например когда игрок передвигал прицел в одну сторону, но увидев соперника сменил направление движения. Для поставленной задачи такие показатели не очень хороши, так как при получении статистических показателей ожидается, что их значения не будут иметь большое отклонение.

Это означает что данные имеют более сложную зависимость, и для ее нахождения можно воспользоваться методами машинного обучения.

2.5. Извлечение признаков для объектов

Вышеописанное решение использовало малое количество признаков на каждом тике, при этом вполне возможно, что существуют такие признаки, которые позволят получить лучшее решение. Поэтому извлечем из записи матчей все возможные данные, которые будут полезны для поставленной задачи.

На каждом тике были извлечены различные данные, которые каким-то образом имеют отношение к игрокам, которые участвуют в рассматриваемых ситуациях, а также некоторое состояние рассматриваемого случая. Каждый из них получен с помощью интерфейсов используемого синтаксического анализатора. Всего было извлечено 118 признаков каждый из них описан в приложении А. Среди них также присутствуют те, которые используются наивным решением. Они были добавлены на основе первоначальной гипотезы, а также для сохранения совместимости данных с наивным решением.

2.6. Улучшение алгоритма, путем применения методов машинного обучения

Решение, описанное в разделе 2.4, оказалось слишком простым как в плане реализации, что является плюсом, так и в плане анализа данных. Для получения лучшего решения следует использовать более сложные зависимости между данными. В связи с этим, было решено применить методы машинного обучения. Они должны дать результат не хуже, так как наивный алгоритм вполне можно осуществить, например, с помощью дерева решений [21].

Перед тем, как применять методы машинного обучения нужно понять, как будут выглядеть объекты, каким образом будет происходить обучение, и какие метрики использовать для оценки качества модели.

Начнем с данных. На этом этапе они представляют собой отрезки времени, в которые один игрок видит противника, и количество тиков, которое понадобилось игроку, чтобы среагировать на появление оппонента. На каждом тике рассматриваемого отрезка времени были получены признаки описанные в разделе 2.5.

Далее рассмотрим алгоритмы машинного обучения, которые будем применять для получения лучшего решения.

2.6.1. Использование алгоритмов, решающих задачу классификации

С первого взгляда кажется, что решаемая задача, это задача регрессии [22], так как результатом обработки объекта должно быть число – время реакции. Но все рассматриваемые ситуации разного размера и соответственно содержат разное количество тиков, что затрудняет построение модели. Поэтому сведем эту задачу к задаче классификации [23]. Сделаем это следующим образом, будем последовательно рассматривать каждый объект и назначать ему класс. Все тики, которые произошли до момента реакции пометим классом «0», а несколько тиков, которые произошли в момент реакции и позже пометим классом «1». Остальные отбрасываем, так как в эти моменты времени происходит события, которые не нужно рассматривать, они не принесут пользы и даже могут ухудшить результаты. Классом «1» помечаем не один тик, а несколько, потому что данные размечены вручную и не являются гарантированно точными, также в некоторых сложных ситуациях, другой человек мог разметить по-другому. Кроме того, это нужно для того, чтобы не было большого количества элементов класса «0», из наблюдений размеченных данных, можно заметить, что количество таких элементов в одной рассматриваемой ситуации в среднем будет равно 30. Иначе возможна ситуация, при которой модель будет помечать классом «0» любой, входящий в нее, объект, так как их количество во много раз превышает объекты класса «1».

Опишем алгоритм на основе которого будет работать модель. Обучение происходит следующим образом, модель принимает на вход набор размеченных ситуаций. Затем извлекается последовательность тиков и им назначаются классы описанным выше способом. Получившиеся наборы размеченных объектов объединим в одну последовательность. Для каждого из этих тиков получим 118 признаков, описанных в разделе 2.5. Таким

образом, имеем последовательность помеченных объектов. На этом наборе и производим обучение модели, решающей задачу классификации.

Вычисление реакции не размеченного случая будет происходить следующим образом, для начала, извлечем последовательность тиков и признаки, описанные выше. Получаем набор объектов, состоящих из 118 признаков. Далее, отдаем эту последовательность на обработку модели, которая классифицирует каждый из объектов. Первый тик, который будет помечен классом «1» и является тиком реакции. Таким образом получается модель, которая решает задачу регрессии путем многократного решения задачи классификации.

Также возможны ситуации, в которых игрок не отреагировал, в таких реакциях каждый тик будем помечать классом «0». Такие реакции безусловно нужны для обучения модели, поэтому значением реакции в данном случае будет число на единицу большее, чем количество тиков в ситуации.

2.6.2. Объединение машинного обучения и наивного алгоритма

Так как наивный алгоритм хорошо работает в очевидных моментах и плохо в сложных, то это наталкивает на мысль о том, чтобы полученные ими результаты объединить для получения более точного результата.

При тестировании на некотором небольшом подмножестве датасета было замечено, что бывают случаи, в которых наивный алгоритм получает немного меньше ожидаемого, а метод машинного обучения немного больше и наоборот. Поэтому было решено объединить оба решения.

Происходить это будет следующим образом: при обучении, будет задействован алгоритм машинного обучения, так как наивному оно не требуется. Обучение будет происходить также как описано в подразделе 2.6.1. При вычислении реакции будем действовать следующим образом, каждый из алгоритмов классифицирует входящий поток тиков. После чего будут получены значения реакции, вычисленные обоими методами. Ответом будем считать число на отрезке от первого до второго ответов. Для того, чтобы выбрать решение, на получившемся отрезке введем следующий параметр – коэффициент доверия к классификатору. Его значения находятся в диапазоне от нуля до единицы. Если этот параметр равен одному, то это значит, что мы хотим полностью учитывать ответ классификатора, для которого этот параметр задан. Если значение равно нулю, то мы полностью не доверяем

ответу данного классификатора и предпочитаем решение другого. Так как данный параметр представлен для обоих алгоритмов, то итоговый ответ будет вычислен по следующей формуле:

$$r = \frac{(a + (b - a)(1 - \alpha)) + (b + (a - b)(1 - \beta))}{2}.$$

В данной формуле r – итоговая реакция, a – это реакция посчитанная первым алгоритмом, b – реакция подсчитанная вторым, α – коэффициент для первого алгоритма, β – коэффициент для второго. После вычисления реакции, каждый тик, произошедший до нее, должен быть помечен классом ноль, остальные классом один.

Также нужно применить методы ансамблирования, например адаптивный бустинг, который будет в качестве базовых моделей иметь наивное решение и некоторый набор алгоритмов машинного обучения, решающих задачу классификации, что возможно позволит получить более лучшее решение.

Выводы по главе 2

В данной главе был описан тип рассматриваемых ситуаций и наглядно изображен ее пример, также представлен алгоритм, который с помощью синтаксического анализатора извлекает из записи матча рассматриваемые случаи. Далее, для каждого извлеченного случая были вручную вычислены реакции путем просмотра записи матча. Разработаны признаки, описывающие состояние ситуации и игроков в ней на каждом тике. Далее был разработан наивный алгоритм, основанный на гипотезе, что при возникновении реакции произойдет резкое изменение положения прицела, а также предложены его улучшения с использованием методов машинного обучения.

ГЛАВА 3. ТЕСТИРОВАНИЕ И ВАЛИДАЦИЯ АЛГОРИТМА И ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

Для тестирования нужно большое количество данных, поэтому было загружено 4 демо файла с сайта hltv.org. Из каждого матча были извлечены рассматриваемые ситуации. Их количество равно 1064. Каждая из этих ситуаций была вручную просмотрена и для нее вычислена реакция. Каждый тик представляет из себя объект со 118 признаками.

Были рассмотрены и протестированы различные алгоритмы для решения задачи классификации, такие как: метод k-ближайших соседей (KNN) [24], метод опорных векторов (SVM) [25], деревья решений, случайный лес, адаптивный бустинг [26]. Реализации данных алгоритмов были взяты из библиотеки для языка программирования Python 3 – `scikit-learn` [27].

Для оценки качества работы того или иного алгоритма нужно вычислять различные метрики. Для оценки полученного результата а именно реакции, будем использовать следующие: средняя абсолютная ошибка [28], средняя квадратичная ошибка [29]. Для оценки работы классификаторов, воспользуемся такими метриками: `f1-score`, `accuracy`, `precision`, `recall` [30].

Алгоритмы машинного обучения имеют одну плохую особенность – переобучение, когда алгоритм на тестовых данных получает значение гораздо меньшее, чем на тренировочных данных. Поэтому воспользуемся валидацией, для оценки обобщающей способности алгоритма. В качестве такого алгоритма был выбран довольно популярный алгоритм кросс-валидации, с делением множества данных на пять равных частей [31]. Также будем проводить несколько тестирований каждого алгоритма для вычисления дисперсии рассматриваемых метрик, причем каждая из них будет средним значением из полученных. Дисперсия будет записываться вторым значением в ячейке таблицы.

3.1. Тестирование наивного алгоритма

Первым алгоритмом для тестирования будет наивный алгоритм. Результаты приведены в таблице 1 и таблице 2.

Данный алгоритм является дискретным и при каждом тестировании возвращает одинаковые результаты, поэтому дисперсия метрик для данного алгоритма равна 0. Также заметим, что алгоритм имеет довольно большое значение средней квадратичной ошибки. Глядя на таблицу 2 можно увидеть,

Таблица 1 – Результаты тестирования наивного алгоритма

Алгоритм	MAE	MSE	f1-score	accuracy	precision	recall
Наивный	3,92	101,53	0,62	0,75	0,49	0,83

Таблица 2 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для наивного алгоритма

Алгоритм	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
Наивный	625	162	148	74	32	8	6	1	8

что 277 ситуаций имеют отклонение больше 3, а 129 - больше 10. Из за этих данных значение средней квадратичной ошибки оказывается таким большим. Это еще раз показывает, что алгоритм довольно хорошо работает в простых ситуациях, выдавая близкий к точному результат, но при этом есть некоторое количество сложных ситуаций, где данный алгоритм показывает плохие показатели. Для того, чтобы уменьшить количество ошибок на такие большие значения воспользуемся методами машинного обучения.

3.2. Тестирование алгоритмов решающих задачу классификации

Далее протестируем классические алгоритмы машинного обучения, решающие задачу классификации. Для начала нужно нормализовать данные, так как они разного типа и диапазона значений. Для это воспользуемся мин-макс алгоритмом [32], который представит каждый параметр вещественным числом от 0 до 1. Теперь каждый объект будет восприниматься классификатором равноценно, что позволит получить более корректный результат.

Первым протестируем метод k-ближайших соседей при различных значениях гиперпараметра k. В качестве реализации взят класс KNeighborsClassifier из библиотеки для машинного обучения scikit-learn. Далее приведены результаты тестирования алгоритма при наилучших значения этого параметра в таблицах 3 и 4.

Можно заметить, что с увеличением количества рассматриваемых соседей возрастают значения средней абсолютной и средней квадратичной ошибок, а также показатели accuracy и precision, при этом уменьшаются значения других метрик классификации, а именно f1-score и recall. Неплохое значение accuracy и плохие значения других метрик классификации, говорят о том, что алгоритм хорошо распознает объекты класса ноль, при этом очень

Таблица 3 – Результаты тестирования метода k-ближайших соседей

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
1	9,15 0,011	298,83 3,47	0,37 9,15e-06	0,58 1,26e-05	0,29 3,39e-06	0,52 7,87e-05
2	9,16 0,006	299,30 2,87	0,37 3,82e-06	0,59 2,37e-05	0,29 5,52e-06	0,50 4,96e-05
9	9,04 3,59e-05	291,79 11,73	0,37 1,84e-05	0,59 1,87e-05	0,29 1,57e-05	0,52 1,65e-05
40	9,77 0,005	356,64 6,15	0,22 3,21e-05	0,73 4,99e-05	0,34 0,0002	0,16 8,93e-05
70	9,75 0,003	350,91 0,82	0,13 4,60e-05	0,75 2,71e-05	0,39 0,002	0,08 1,71e-05

Таблица 4 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для метода k-ближайших соседей

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
1	440	61	236	172	74	26	18	11	23
2	438	60	240	172	73	27	17	12	23
9	427	67	245	178	68	26	17	11	22
40	427	59	241	188	49	34	18	15	29
70	420	54	248	198	48	33	17	15	28

плохо объекты класса один, которые означают тик реакции, в следствии чего получаются значения с большим отклонением от ожидаемого, что и отражает таблица 4. Видим, что число правильно угаданных ситуаций около 430, а это лишь 40 % от общего числа рассматриваемых ситуаций и при этом около 340 с отклонением больше либо равным десяти.

Также данный алгоритм работает хуже, чем наивный, что показывают значения средней абсолютной ошибки, среднеквадратичной ошибки, f1-score и других рассматриваемых метрик. Также это прослеживается из таблицы отклонений вычисленного значения от ожидаемого.

Лучшей конфигурацией, среди протестированных, является третья, поскольку она имеет минимальное значение средних абсолютной и квадратичной ошибок, причем дисперсия всех метрик, кроме MSE, является очень маленьким значением, но правая граница получаемого интервала для среднего квадратичного отклонения примерно равна правой границе первой

и второй конфигураций, при этом левая граница меньше, что говорит о том, что значения MSE получаемый данным алгоритм лучшие.

Рассмотрим следующий алгоритм машинного обучения – метод опорных векторов. Возьмем реализацию SGDClassifier из библиотеки для машинного обучения scikit-learn. Она основана на стохастическом градиентном спуске, который позволяет быстрее проводить тестирование. Данный алгоритм имеет следующие гиперпараметры, которые мы будем изменять: метод регуляризации, коэффициент регуляризации и количество итераций алгоритма. Далее приведены результаты тестирования в таблицах 5 и 6.

Таблица 5 – Результаты тестирования метода опорных векторов

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
100 l1 1e-09	7,63 0,08	239,80 2,75	0,39 0,0005	0,73 0,001	0,43 0,002	0,38 0,005
100 l1 0,001	7,82 0,009	271,53 9,73	0,28 7,36e-05	0,79 9,25e-07	0,70 2,12e-05	0,17 4,42e-05
100 l2 0,01	9,88 0,002	353,86 6,63	0,01 2,47e-06	0,76 8,96e-09	0,73 0,008	0,00 6,37e-07
1000 l1 1e-09	7,63 0,08	239,80 2,75	0,39 0,0005	0,73 0,001	0,43 0,002	0,38 0,005
1000 l1 0,001	7,82 0,009	271,53 9,73	0,28 7,36e-05	0,79 9,25e-07	0,70 2,12e-05	0,17 4,42e-05
1000 l2 0,01	9,88 0,002	353,86 6,63	0,01 2,47e-06	0,76 8,96e-09	0,73 0,008	0,00 6,37e-07

Таблица 6 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для метода опорных векторов

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
100 l1 1e-09	440	119	242	147	53	21	11	8	20
100 l1 0,001	464	98	235	150	46	24	12	10	21
100 l2 0,01	411	57	244	207	49	32	17	16	29
1000 l1 1e-09	440	119	242	147	53	21	11	8	20
1000 l1 0,001	464	98	235	150	46	24	12	10	21
1000 l2 0,01	411	57	244	207	49	32	17	16	29

Из данных таблицы 5 можно заметить, что увеличение количества итераций не влияет на результат, каждый из алгоритмов при 100 и 1000

итераций показывают одинаковые результаты. Также при увеличении параметра регуляризации, значение средней абсолютной ошибки увеличивается, что также прослеживается по таблице 6. Значения первых столбцов, которые обозначают количество элементов, ошибка на которых составила соответствующее столбцу значение, уменьшается, а значения столбцов, которые отражают большую ошибку, увеличиваются.

Лучшей конфигураций метода опорных векторов, среди принимавших участие в тесте, является первая, поскольку имеет оптимальные значения средних абсолютной и квадратичной ошибок и f1-меры, а также приемлемые показатели дисперсии рассматриваемых величин. Из таблицы 6 видно, что для лучшей конфигурации число правильно угаданных ситуаций – 440, что составляет около 41 % от общего числа рассматриваемых ситуаций, и при этом 260 с отклонением больше либо равным десяти.

Метод опорных векторов показывает результаты лучше, чем метод k-ближайших соседей, но все же хуже, чем наивный, что можно увидеть глядя на значения средней абсолютной ошибки, среднеквадратичной ошибки, f1-score. Также увеличилось число ситуаций, в которых не было допущено ошибок и уменьшилось тех, в которых допущена ошибка больше или равная десяти, что подтверждает уменьшение значений средней абсолютной ошибки и средней квадратичной ошибки.

Протестируем следующий алгоритм – дерево решений. Его реализацию также возьмем из библиотеки для машинного обучения scikit-learn. Этот метод реализует класс DecisionTreeClassifier. Гиперпараметром данного алгоритма, который мы будем выбирать, является высота дерева. Далее приведены результаты тестирования в таблицах 7 и 8.

По результатам тестирования деревьев решений видно, что данный метод показывает результаты лучше всех предыдущих протестированных алгоритмов машинного обучения. Значения средней абсолютной ошибки и средней квадратичной ошибки приближаются к показателям наивного алгоритма, но все еще хуже их. Также из таблицы 8 можно заметить, что большая часть объектов сконцентрирована в левой половине. Это говорит о том, что случаев, когда отклонение от правильного значения большое, становится меньше.

Таблица 7 – Результаты тестирования дерева решений

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
8	5,18 0,002	165,02 3,26	0,58 5,44e-05	0,81 2,49e-06	0,60 3,46e-06	0,56 0,0002
9	5,38 0,002	169,10 12,68	0,57 1,79e-05	0,80 1,07e-05	0,59 0,0001	0,56 0,0004
21	5,93 0,004	180,31 7,97	0,54 4,66e-07	0,76 1,40e-05	0,50 4,01e-05	0,60 5,93e-05
31	6,04 0,001	184,44 11,98	0,54 2,92e-05	0,76 1,05e-06	0,49 2,64e-06	0,60 0,0002
32	6,04 0,001	184,44 11,98	0,54 2,92e-05	0,76 1,05e-06	0,49 2,64e-06	0,60 0,0002

Таблица 8 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для деревьев решений

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
8	472	248	183	86	33	18	8	5	9
9	454	257	187	87	35	17	10	6	9
21	415	246	213	105	41	17	11	4	10
31	409	247	213	106	44	17	11	4	10
32	409	247	213	106	44	17	11	4	10

Первая конфигурация является лучшей, поскольку имеет оптимальные показатели рассматриваемых метрик качества. А также одни из минимальных значений дисперсий этих величин. Из таблицы 8 следует, что эта конфигурация обрабатывает без ошибок 472 объекта, что составляет около 44 % от общего количества рассматриваемых ситуаций, и при этом 259 объектов с отклонением большим либо равным 10 от ожидаемого. Это также показывает что данный алгоритм работает лучше, чем ранее рассмотренные модели.

Для получения лучших показателей воспользуемся методами ансамблирования. Они могут позволить получить более точный ответ используя простые модели, которые тестировались выше.

Рассмотрим первый такой алгоритм – случайный лес. Для тестирования воспользуемся реализацией из библиотеки для машинного обучения scikit-learn. Этот метод реализован в классе RandomForestClassifier. Гиперпараметрами данного алгоритма являются максимальная высота

дерева, а также их количество в ансамбле. Далее приведены результаты тестирования в таблицах 9 и 10.

Таблица 9 – Результаты тестирования случайного леса

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
100 7	4,60 0,009	135,94 1,21	0,61 2,18e-05	0,84 1,84e-06	0,70 6,38e-06	0,54 4,11e-05
100 10	4,59 0,0005	134,76 0,16	0,63 1,25e-05	0,84 6,30e-06	0,70 8,82e-05	0,58 4,73e-06
100 11	4,60 0,0034	135,26 10,35	0,64 6,48e-06	0,84 2,40e-06	0,70 3,53e-05	0,58 1,21e-05
500 8	4,59 0,012	135,62 13,61	0,62 3,20e-06	0,84 1,82e-06	0,70 4,26e-05	0,56 1,86e-05
500 11	4,54 0,002	133,97 0,31	0,64 1,42e-05	0,84 4,59e-06	0,70 5,82e-05	0,59 1,84e-05
1000 9	4,58 0,0059	136,20 8,25	0,63 7,86e-06	0,84 2,24e-06	0,70 4,51e-05	0,57 3,14e-05
1000 10	4,56 0,005	135,73 13,16	0,63 1,28e-05	0,84 4,01e-06	0,70 5,31e-05	0,58 1,73e-05

Таблица 10 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для случайного леса

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
100 7	518	250	150	80	27	15	5	6	10
100 10	520	253	144	79	29	16	5	6	9
100 11	517	257	145	77	28	16	5	6	9
500 8	520	249	151	77	27	16	5	6	9
500 11	528	252	140	78	29	15	5	6	9
1000 9	526	249	145	78	29	15	5	6	10
1000 10	528	248	143	78	28	15	5	6	10

По результатам тестирования случайный лес имеет лучшие значения показателей средних абсолютной и квадратичной ошибок и метрик классификации по сравнению с предыдущими алгоритмами.

Лучшие значения метрик достигаются при 500 деревьях в лесу с максимальной высотой равной 11. Данная конфигурация показывает как оптимальные значения вычисленных показателей качества, так и хорошие значения их дисперсий. Количество ситуаций, которые вычисляются в точности, для случайного леса с выбранными гиперпараметрами, равно 528,

что больше, чем у оптимальной конфигурации дерева решений. Посмотрим на количество элементов, отклонение которых от ожидаемого больше либо равно 10. Их 142, что меньше, чем у деревьев. При этом, у данного алгоритма значения f1-score, accuracy и precision лучше, чем у наивного решения, но показатели ошибки хуже, что можно увидеть сравнив таблицы 10 и 2.

Протестируем следующий алгоритм ансамблирования – адаптивный бустинг или AdaBoost. Воспользуемся реализацией из библиотеки для машинного обучения scikit-learn. Этот метод реализован в классе AdaBoostClassifier. Гиперпараметрами данного алгоритма являются базовая модель – алгоритм классификации, и количество моделей в ансамбле. В качестве базового классификатора возьмем дерево решений, так как этот алгоритм показал лучшие показатели, в сравнении с остальными. Далее приведены результаты тестирования в таблицах 11 и 12.

Таблица 11 – Результаты тестирования адаптивного бустинга

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
50 3	4,98 0,044	148,17 11,43	0,59 4,12e-05	0,80 2,78e-06	0,57 7,52e-06	0,61 0,0001
50 12	4,85 0,008	140,46 1,67	0,57 5,41e-05	0,81 5,84e-07	0,61 3,31e-06	0,54 0,0002
100 11	4,78 0,0004	138,84 5,74	0,59 4,92e-05	0,82 4,60e-06	0,63 2,261e-05	0,55 0,0001
500 3	5,19 0,005	152,08 4,56	0,59 7,81e-06	0,79 3,23e-06	0,54 1,14e-05	0,64 1,53e-05
500 10	4,63 0,0005	135,56 2,82	0,60 7,40e-06	0,82 1,66e-06	0,64 2,83e-05	0,57 5,09e-05
500 12	4,62 7,79e-05	136,14 0,57	0,60 1,37e-05	0,82 2,65e-06	0,65 1,51e-05	0,57 1,25e-05

По результатам тестирования, адаптивный бустинг деревьев решений показал неплохие значения используемых метрик, в сравнении с предыдущими алгоритмами. Оптимальная конфигурация достигается при 500 деревьях в ансамбле с максимальной высотой 10. Дисперсии рассматриваемых величин также являются очень хорошими. Показатели данной модели обходят все другие рассмотренные методы машинного обучения, кроме случайного леса, у которого они немного лучше. Можно отметить, что оптимальная конфигурация безошибочно обрабатывает 522 ситуации, что составляет около

Таблица 12 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для адаптивного бустинга

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	≥ 60
50 3	476	251	184	82	35	13	8	3	9
50 12	483	257	176	80	27	16	6	6	9
100 11	494	261	162	80	27	15	6	6	9
500 3	483	240	176	85	39	16	8	5	8
500 10	522	247	149	79	27	17	5	5	9
500 12	522	244	153	80	25	16	5	6	9

49 % от общего числа рассматриваемых моментов. Количество объектов, на которых была допущена ошибка больше либо равная 10 – 141.

Из проведенных выше экспериментов видно, что модели на основе случайного леса показывают лучшие значения метрик качества, и их величина приближается к наивному решению.

3.3. Применение метода отбора признаков

Каждый новый рассматриваемый выше метод, улучшал получаемые результаты, но получилось лишь приблизиться, но не обойти наивное решений. Также в ходе тестирования было замечено, что работа алгоритмов занимает продолжительное время. Для улучшения полученных результатов и времени работы, было решено применить методы отбора признаков. Они позволят значительно уменьшить размер данных, что поспособствует уменьшению времени работы алгоритмов, а также может улучшить качественную их работу, так как некоторые признаки, своим наличием, могут лишь ухудшить качество работы классификатора.

В качестве алгоритма отбора признаков был выбран алгоритм фильтрации, где функциями, используя которые проводится селекция, являются chi-square test [33] и ANOVA F-test [34]. Будем выбирать лучшие признаки основываясь на описанных функциях, количество оставшихся признаков будем перебирать. Далее будут приведены результаты тестирования рассмотренных выше алгоритмов машинного обучения с применением описанных алгоритмов отбора признаков. Реализации использующихся функций взяты из библиотеки для машинного scikit-learn, chi2 и f_classif соответственно. В каждой приведенной ниже таблице результатов тестирования, в столбце параметры, будет указан используемый метод

отбора, который позволит добиться приведенного в ней качества работы. В приложении Б приведены отобранные признаки для разных конфигураций алгоритма.

Первым будет метод k-ближайших соседей, при различных значениях гиперпараметра k. Результаты тестирования приведены в таблицах 13 и 14.

Таблица 13 – Результаты тестирования алгоритма k-ближайших соседей с применением отбора признаков

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
5	5,29	131,88	0,54	0,80	0,58	0,51
f_classif 30	0,007	18,97	1,13e-05	1,91e-06	2,01e-05	5,15e-05
4	5,39	134,63	0,50	0,80	0,61	0,43
f_classif 30	0,002	4,62	3,42e-07	1,69e-06	3,10e-05	8,81e-06
9	4,97	139,17	0,57	0,81	0,62	0,53
f_classif 10	0,003	3,22	9,19e-06	1,91e-06	1,52e-05	1,55e-05
10	5,35	139,64	0,53	0,81	0,66	0,44
f_classif 30	0,02	16,09	4,42e-05	4,12e-06	4,59e-05	6,70e-05
7	4,97	145,58	0,58	0,82	0,64	0,54
f_classif 20	0,0006	4,58932	2,50e-05	4,54e-06	3,55e-05	3,96e-05
15	5,29	162,45	0,54	0,81	0,64	0,47
chi2 5	0,0002	0,92	6,52e-06	5,34e-07	2,41e-06	9,09e-06
15	5,29	162,45	0,54	0,81	0,64	0,47
f_classif 5	0,0002	0,92	6,53e-06	5,34e-07	2,41e-06	9,09e-06

Полученные данные показывают, что применение метода отбора признаков пошло на пользу. Значения рассматриваемых метрик улучшились, что можно увидеть сравнив таблицы 3 и 13. Посмотрим на таблицы 4 и 14, заметим, что с применением метода отбора признаков количество точно угаданных ситуаций увеличилось, а также уменьшилось количество объектов, на которых алгоритм ошибается на большие значения. Это безусловно хорошо, но метод k-ближайших соседей все еще имеет худшие показатели рассматриваемых метрик, в сравнении с наивным решением.

Лучшей конфигурацией, по приведенным выше результатам, является вторая, хоть и значение среднеквадратичной ошибки больше, чем у первой модели, но значение дисперсии намного меньше, что говорит о большей ее стабильности при использовании.

Таблица 14 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для метода k-ближайших соседей с применением отбора признаков

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
5 f_classif 30	475	215	197	95	46	13	6	7	8
4 f_classif 30	464	227	187	103	45	14	6	6	9
9 f_classif 10	471	243	199	80	36	14	6	4	9
10 f_classif 30	485	205	196	95	42	15	5	6	11
7 f_classif 20	502	220	182	90	31	16	8	3	10
15 chi2 5	484	236	185	82	30	18	7	6	14
15 f_classif 5	484	236	185	82	30	18	7	6	14

Также можно увидеть, что для метода k-ближайших соседей лучшим оказался метод отбора признаков на основе функции ANOVA F-test с выбором лучших 30 признаков.

Далее рассмотрим метод опорных векторов с применением алгоритма отбора признаков. Данные о тестировании приведены в таблицах 15 и 16.

Сравнив результаты тестирования, полученные без применения метода отбора признаков, представленные в таблице 5, и с его применением, находящиеся в таблице 15, можно заметить, что происходит небольшое улучшение качества рассматриваемых метрик. Но при этом, они все еще остаются хуже, чем показывает, например, наивное решение. Сравнивая таблицы 6 и 16 можно увидеть, что число объектов, на которых допускается меньшая ошибка, увеличилось, а число объектов, где допускается большая – уменьшилась. Вследствие этого показатели данных конфигураций показывают лучшие метрики.

Лучшей конфигурацией, среди представленных является первая, поскольку она имеет оптимальные значения средних абсолютной и квадратичной ошибок и f1-score, а также их дисперсий.

Таблица 15 – Результаты тестирования метода опорных векторов используя отбор признаков

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
1000 11 1e-06 f_classif 50	6,92 0,10	202,15 7,21	0,46 0,0004	0,75 0,001	0,49 0,006	0,44 0,001
5000 11 1e-06 f_classif 50	6,92 0,10	202,15 7,21	0,46 0,0004	0,75 0,001	0,49 0,006	0,44 0,001
100 11 0,0001 chi2 50	6,27 0,0001	214,24 14,78	0,44 9,96e-05	0,81 4,99e-08	0,76 0,0002	0,31 0,0001
1000 11 0,0001 chi2 50	6,27 0,0001	214,24 14,78	0,44 9,96e-05	0,81 4,99e-08	0,76 0,0002	0,31 0,0001
100 11 0,0001 f_classif 30	6,44 0,001	217,72 1,22	0,42 0,0001	0,81 1,5678e-06	0,77 0,0001	0,29 0,0001
1000 11 1e-05 chi2 20	6,60 0,003	231,76 14,05	0,44 8,42e-05	0,81 1,59e-06	0,75 0,0004	0,31 0,0001
100 11 1e-09 f_classif 20	8,20 0,20	243,45 4,02	0,38 0,0004	0,65 0,00540797	0,34 0,001	0,46 0,018

Таблица 16 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для метода опорных векторов с применением отбора признаков

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
1000 11 1e-06 f_classif 50	455	147	219	136	51	25	8	6	15
5000 11 1e-06 f_classif 50	455	147	219	136	51	25	8	6	15
100 11 0,0001 chi2 50	498	157	199	121	39	16	8	7	17
1000 11 0,0001 chi2 50	498	157	199	121	39	16	8	7	17
100 11 0,0001 f_classif 30	487	157	202	127	36	19	8	7	18
1000 11 1e-05 chi2 20	483	167	197	120	40	20	8	8	19
100 11 1e-09 f_classif 20	412	106	248	165	68	27	10	8	17

Оптимальный набор признаков, для метода опорных векторов, получается при выборе лучших 50 с помощью метода ANOVA F-test.

Следующим на очереди будет дерево решений с применением алгоритма отбора признаков. Результаты тестирования представлены в таблицах 17 и 18.

Таблица 17 – Результаты тестирования дерева решений с применением отбора признаков

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
6	4,62	125,46	0,59	0,82	0,64	0,55
chi2 50	0,03	6,02	7,02e-05	6,67e-06	0,0002	0,0005
5	4,73	136,43	0,59	0,82	0,63	0,55
chi2 30	0,03	11,34	0,0002	7,70e-06	2,13e-07	0,0008
17	5,69	169,48	0,59	0,80	0,58	0,60
f_classif 20	0,003	1,50	2,42e-05	5,11e-06	2,25e-05	4,52e-05
18	5,82	174,58	0,53	0,76	0,50	0,57
chi2 50	0,02	6,86	2,28e-05	2,18e-05	7,06e-05	1,43e-06
2	5,53	187,69	0,56	0,79	0,56	0,56
chi2 10	0,002	10,71	0,0001	4,06e-06	3,42e-05	0,0005
4	5,63	192,70	0,56	0,82	0,68	0,47
f_classif 30	0,01	6,27	0,0001	8,83e-06	0,0001	0,0003
3	5,78	201,95	0,55	0,80	0,59	0,51
chi2 5	0,0008	2,36	3,93e-05	1,94e-06	6,06e-05	0,0003

Таблица 18 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для дерева решений с применением отбора признаков

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
6	531	215	173	75	33	14	7	4	9
chi2 50									
5	522	216	180	77	34	14	6	5	8
chi2 30									
17	433	258	190	93	47	17	8	7	8
f_classif 20									
18	423	237	223	95	44	14	7	7	10
chi2 50									
2	532	181	178	88	38	17	7	7	13
chi2 10									
3	444	258	193	91	32	16	7	5	15
f_classif 30									
3	511	195	182	89	34	20	7	7	17
chi2 5									

Данное тестирование показало, что качество работы дерева решений лучше при применении метода отбора признаков. Сравнении таблиц 7 и

17 показывает это, так как каждая из рассматриваемых метрик улучшилась. Также, глядя на таблицы 8 и 18, можно заметить, что с применением метода отбора признаков увеличилось количество объектов в первой части таблицы, и соответственно уменьшилось во второй, что говорит о том, что величина ошибок, допущенных алгоритмом, уменьшается. Но, к сожалению, этот метод также не дает решения лучше, чем наивное.

Оптимальной конфигурацией является дерево решений, построенное на датасете, на котором был произведен отбор лучших 50 признаков с использованием chi-square test, с максимальной высотой 6. Она имеет минимальные, среди представленных, значения средних абсолютной и квадратичной ошибки с маленькими значениями дисперсии этих величин.

Далее рассмотрим случайный лес с применением метода отбора признаков. Результаты проведенного тестирования представлены в таблицах 19 и 20.

Таблица 19 – Результаты тестирования случайного леса с применением отбора признаков

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
100 10	4,38	117,42	0,63	0,84	0,69	0,58
chi2 50	0,006	7,78	3,61e-05	5,79e-06	3,04e-05	4,10e-05
500 8	4,46	138,86	0,64	0,85	0,75	0,56
f_classif 30	0,0007	4,64	3,67e-06	1,21e-06	1,96e-05	1,66e-06
100 6	4,53	143,26	0,63	0,85	0,75	0,55
f_classif 30	0,0003	1,06	4,46e-06	1,75e-06	4,23e-05	8,63e-06
100 8	4,55	144,20	0,64	0,84	0,69	0,59
chi2 20	0,0016	4,94	1,36e-05	2,21e-06	1,68e-05	2,44e-05
500 13	4,57	145,11	0,63	0,84	0,69	0,57
f_classif 10	0,014	11,95	8,12e-06	3,48e-06	4,04e-05	2,73e-07
100 10	4,55	147,64	0,65	0,85	0,73	0,58
f_classif 50	0,007	0,65	9,46e-06	2,49e-06	3,45e-05	1,11e-05
100 7	4,64	149,09	0,64	0,85	0,75	0,56
f_classif 20	0,0008	6,68	7,38e-06	1,92e-06	2,40e-05	2,07e-06

Проанализируем полученные данные. Сравнив таблицы 9 и 19 можно увидеть, что лишь первая конфигурация, с применением метода отбора признаков, обходит другие конфигурации случайного леса. Она и является оптимальной, по приведенным выше результатам. Эта конфигурация имеет более лучшее значение средних абсолютной и квадратичной ошибок, что

Таблица 20 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для случайного леса с применением отбора признаков

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
100 10 chi2 50	554	217	152	73	32	16	5	4	8
500 8 f_classif 30	525	252	145	79	29	12	5	5	9
100 6 f_classif 30	525	247	149	79	30	13	4	5	10
100 8 chi2 20	550	223	147	74	33	14	5	5	10
500 13 f_classif 10	525	245	150	78	30	13	5	5	10
100 10 f_classif 50	522	258	145	73	30	14	5	5	9
100 7 f_classif 20	528	242	148	80	28	14	5	4	11

также можно проследить, сравнив таблицы 10 и 20. Количество безошибочно угаданных объектов увеличилось и составляет около 52 % от общего числа объектов. Уменьшилась величина случаев, на которых допускается отклонение от одного до трех, но при этом немного увеличилось для ошибок больше либо равных четырех.

Лучшей конфигурацией случайного леса является первая, поскольку имеет превосходящее значение средней квадратичной ошибки по сравнению с другими. Также данная модель обучена на 50 лучших признаков, отобранных с помощью chi-square test.

Протестируем следующий алгоритм – это адаптивный бустинг деревьев решений с применением алгоритма отбора признаков. Результаты тестирования приведены в таблицах 21 и 22.

Рассмотрим полученные результаты. Сравним показатели из таблиц 11 и 21. Показатели средних абсолютной и квадратичной ошибок, f1-score, accuracy и precision у адаптивного бустинга деревьев решений с применением метода отбора признаков лучше, чем при использовании всего множества признаков. Сравним таблицы 12 и 22. Видно, что количество безошибочно обработанных ситуаций осталось на примерно том же уровне. Число случаев, отклонение в которых составило от одного до трех и от четырех до

Таблица 21 – Результаты тестирования метода адаптивного бустинга деревьев решений с применением отбора признаков

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
100 1	4,51	115,97	0,63	0,83	0,66	0,60
f_classif 50	0,0005	2,68	5,67e-05	1,99e-05	0,0002	2,40e-05
100 2	4,68	118,47	0,60	0,82	0,63	0,58
f_classif 10	0,0001	7,92	2,03e-05	5,79e-06	3,77e-05	1,07e-05
100 2	4,49	121,16	0,63	0,83	0,66	0,60
f_classif 30	0,007	4,78	6,34e-05	1,02e-05	4,43e-05	8,83e-05
500 2	4,91	128,77	0,60	0,81	0,59	0,61
chi2 30	0,024	7,06	0,0002	7,54e-05	0,00042	3,34e-05
500 1	4,73	131,55	0,61	0,81	0,61	0,60
chi2 50	0,002	1,04	1,26e-05	7,53e-06	5,53e-05	1,99e-05
500 6	4,92	135,09	0,61	0,82	0,61	0,60
chi2 20	0,01	3,79	5,29e-06	9,34e-08	4,09e-06	4,11e-05
500 10	4,82	139,01	0,61	0,83	0,68	0,56
f_classif 30	0,004	5,23	1,10e-05	2,49e-06	1,89e-05	6,65e-06

Таблица 22 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для метода адаптивного бустинга деревьев решений с применением отбора признаков

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
100 1	480	276	171	74	31	12	6	5	7
f_classif 50									
100 2	484	246	185	82	32	14	5	5	8
f_classif 10									
100 2	483	270	174	73	34	10	5	3	8
f_classif 30									
500 2	483	231	191	85	40	13	8	3	7
chi2 30									
500 1	522	214	174	82	38	14	5	4	8
chi2 50									
500 6	493	237	181	73	41	16	7	4	9
chi2 20									
500 10	507	242	164	81	28	18	7	6	9
f_classif 30									

девяти, для большинства конфигураций немного увеличилось, вследствие чего уменьшилось количество ошибок на большие значения. Поэтому значения

средних абсолютной и квадратичной ошибок, f1-score, accuracy и precision улучшилось.

Лучшей конфигурацией является адаптивный бустинг 100 деревьев решений с максимальной высотой дерева равной единице при использовании датасета, где у каждого объекта 50 признаков, отобранных с помощью ANOVA F-test.

По приведенным результатам можно заметить, что применение алгоритма отбора признаков улучшило качество работы каждого рассматриваемого метода машинного обучения.

3.4. Тестирование алгоритма объединения наивного решения и решения, полученного методом машинного обучения

Протестируем алгоритм описанный в подразделе 2.6.2. В данном методе, в качестве первого алгоритма будем наивный, описанный в разделе 2.4, а вторым случайный лес, адаптивный бустинг или дерево решений, поскольку они показали наилучшие показатели по результатам, проведенных выше, тестов. В качестве гиперпараметров данного алгоритма выступают, модель, с которой объединяется наивное решение, а также коэффициенты доверия для обоих методов.

Проведем два тестирования, первое – без использования алгоритма отбора признаков и второе с его применением, поскольку, проведенное выше, тестирование показало увеличение качества работы при уменьшении количества используемых признаков. Результаты первого тестирования приведены в таблицах 23 и 24.

Проанализируем полученные результаты. Глядя на таблицу 23 можно заметить, что каждая из представленных конфигураций имеет лучшее значение средней квадратичной ошибки, по сравнению с остальными, протестированными, моделями. При этом показатели классификации остаются на среднем уровне. Рассмотрим таблицу 24 и сравним ее с соответствующими таблицами выше. Можно заметить, что количество безошибочно вычисленных реакций в некоторых случаях гораздо меньше обычного, но при этом основная часть объектов находится в первых трех столбцах, что говорит о том, что количество больших ошибок уменьшилось.

Модель, объединяющая наивное решение и адаптивный бустинг 100 деревьев с максимальной высотой 10 и коэффициентами доверия равными 0,5

Таблица 23 – Результаты тестирования алгоритма без использования метода отбора признаков

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
RF(100 1) 1 1	4,03 0,002	77,35 2,68	0,55 3,54e-05	0,79 1,02e-05	0,55 6,10e-05	0,55 0,0002
RF(100 3) 0,9 0,3	4,07 1,23e-05	93,57 0,04	0,61 2,09e-07	0,77 2,32e-07	0,51 4,84e-07	0,76 1,59e-07
DT(8) 1 0,4	4,04 0,0003	96,26 2,19	0,61 9,50e-08	0,75 2,20e-07	0,49 3,31e-07	0,80 2,07e-06
Ada(500 10) 0,5 0,7	3,87 0,0004	92,23 0,004	0,60 3,98e-07	0,77 2,20e-07	0,51 4,55e-07	0,75 1,43e-06
RF(100 9) 0,9 1	3,88 0,003	94,01 8,51	0,61 7,19e-06	0,77 7,85e-06	0,51 1,75e-05	0,75 4,94e-06
RF(1000 11) 0,5 0,7	3,86 0,007	95,20 4,88	0,61 5,54e-05	0,78 4,70e-05	0,52 0,0001	0,75 2,82e-06
RF(100 9) 0,5 1	3,93 0,005	99,63 9,02	0,60 1,72e-05	0,77 1,64e-05	0,51 3,88e-05	0,73 1,00e-05

Таблица 24 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для алгоритма без применения метода отбора признаков

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
RF(100 1) 1 1	317	296	369	55	10	4	2	1	7
RF(100 3) 0,9 0,3	398	382	167	75	16	10	4	1	8
DT(8) 1 0,4	451	320	171	71	27	9	5	1	7
Ada(500 10) 0,5 0,7	461	303	192	58	27	9	4	2	6
RF(100 9) 0,9 1	475	291	188	60	26	9	4	2	6
RF(1000 11) 0,5 0,7	482	290	184	59	26	7	4	2	7
RF(100 9) 0,5 1	486	291	180	56	27	9	5	2	6

и 0,7, является оптимальной, поскольку имеет одно из минимальных значений средней абсолютной ошибки, имеет довольно низкое значение средней квадратичной ошибки и минимальные значения дисперсий рассматриваемых величин. Также количество объектов, ошибка на которых не больше трех равно 764, что составляет около 70 % от общего числа рассматриваемых ситуаций.

Далее применим метод отбора признаков, для улучшения качества работы моделей. Результаты второго тестирования приведены в таблицах 25 и 26.

Таблица 25 – Результаты тестирования алгоритма с применением метода отбора признаков

Параметры	MAE	MSE	f1-score	accuracy	precision	recall
RF(100 1) 1 1 chi2 30	3,90 2,38e-05	59,67 0,001	0,54 1,12e-06	0,80 3,71e-06	0,61 2,43e-06	0,48 1,99e-06
RF(100 10) 1 1 chi2 50	3,73 0,001	87,06 0,67	0,63 1,44e-05	0,78 9,64e-06	0,53 2,18e-05	0,77 5,21e-06
RF(500 9) 0,1 0,4 chi2 30	3,72 6,73e-05	85,67 0,28	0,62 2,78e-06	0,79 2,79e-06	0,54 8,35e-06	0,73 5,69e-07
RF(100 6) 0,7 1 f_classif 20	3,72 0,003	85,67 0,34	0,63 2,15e-06	0,79 1,53e-06	0,55 4,94e-06	0,73 1,17e-05
Ada(500 2) 1 1 chi2 30	4,06 0,004	91,78 4,76	0,59 3,11e-05	0,74 2,31e-05	0,47 3,39e-05	0,79 2,21e-05
DT(5) 0,8 0,2 chi2 30	3,91 0,001	88,27 19,26	0,62 3,25e-06	0,77 1,31e-06	0,50 2,42e-06	0,80 3,03e-05
RF(100 1) 0,7 1 chi2 30	4,12 2,34e-05	63,18 0,005	0,44 5,99e-06	0,80 1,45e-06	0,67 4,91e-06	0,33 8,73e-07

Таблица 26 – Количество ситуаций, имеющие соответствующие отклонения от ожидаемого результата для алгоритма с применением метода отбора признаков

Параметры	0	1-3	4-9	10-19	20-29	30-39	40-49	50-59	>=60
RF(100 1) 1 1 chi2 30	315	246	435	48	8	3	3	1	5
RF(100 10) 1 1 chi2 50	466	313	180	59	24	7	5	1	7
RF(500 9) 0,1 0,4 chi2 30	480	290	193	56	23	7	5	1	7
RF(100 6),0,7 1 f_classif 20	480	288	196	56	22	7	5	1	7
Ada(500 2) 1 1 chi2 30	426	313	210	68	26	6	4	1	6
DT(5) 0,8 0,2 chi2 30	464	319	167	66	25	8	5	1	6
RF(100 1) 0,7 1 chi2 30	338	187	473	46	9	3	1	1	6

При анализе таблицы 25 можно заметить, что по регрессионным метрикам, а именно средним абсолютной и квадратичной ошибкам, рассматриваемый алгоритм, во многих случаях, опережает наивный алгоритм, а также наибольшую часть других рассматриваемых алгоритмов машинного

обучения. Также применение метода отбора признаков улучшило работу алгоритма, что можно увидеть, сравнив данные результаты, с показателями прошлого тестирования. Из таблицы 26 видно, что допускается меньше ошибок на большие значения, что и привело к снижению показателей средних абсолютной и квадратичной ошибок.

Первая и последняя конфигурации имеют наименьшие значения средней квадратичной ошибки, но при этом значение средней абсолютной ошибки довольно большое, это происходит по той причине, что количество больших ошибок уменьшилось, но вместе с тем количество точно угаданных реакций также снизилось, вследствие чего увеличилось количество объектов отклонение на которых составляет от четырех до девяти тиков. По этой причине такие конфигурации нельзя считать оптимальными. При использовании адаптивного бустинга и деревьев решений, в качестве второй модели, алгоритм показывает одни из худших значений регрессионных метрик, а также значение дисперсии средней квадратичной величины являются больше, чем у остальных алгоритмов. Среди оставшихся трех алгоритмов лучшими являются третий и четвертый, поскольку они имеют лучшие показатели рассматриваемых метрик. Для данных моделей количество объектов, на которых ошибка не меньше десяти, примерно равно 99, что составляет около 9 % от общего числа объектов, что говорит о том, что количество больших ошибок уменьшилось. Число объектов с отклонением не большим трех равно 770, что составляет около 72 %.

3.5. Общая сравнительная таблица

Протестировали все алгоритмы, которые планировалось рассмотреть, поэтому теперь нужно выбрать лучший из них. В таблице 27, приведены значения метрик лучших конфигураций для каждого рассмотренного алгоритма и метода машинного обучения.

Из таблицы 27 видно, что лучшей моделью является метод объединения, описанный в подразделе 2.6.2, с применением метода отбора признаков. Данный алгоритм имеет лучшие значения средних абсолютной и квадратичной ошибок, это говорит о том, что он допускает меньше больших ошибок, чем другие представленные модели, что однозначно является хорошим показателем. Если быть точным, данная конфигурация лишь на 9

Таблица 27 – Результаты тестирования лучших конфигураций рассмотренных алгоритмов и методов машинного обучения

Модели	MAE	MSE	f1-score	accuracy	precision	recall
Наивный алгоритм	3,92	101,53	0,62	0,75	0,49	0,83
KNN(9)	9,04 3,59e-05	291,79 11,73	0,37 1,84e-05	0,59 1,87e-05	0,29 1,57e-05	0,52 1,65e-05
SVM(100 11 1e-09)	7,63 0,08	239,80 2,75	0,39 0,0005	0,73 0,001	0,43 0,002	0,38 0,005
DT(8)	5,18 0,002	165,02 3,26	0,58 5,44e-05	0,81 2,49e-06	0,60 3,46e-06	0,56 0,0002
RF(500 11)	4,54 0,002	133,97 0,31	0,64 1,42e-05	0,84 4,59e-06	0,70 5,82e-05	0,59 1,84e-05
Ada(500 10)	4,63 0,0005	135,56 2,82	0,60 7,40e-06	0,82 1,66e-06	0,64 2,83e-05	0,57 5,09e-05
KNN(4) f_classif 30	5,39 0,002	134,63 4,62	0,50 3,42e-07	0,80 1,69e-06	0,61 3,10e-05	0,43 8,81e-06
SVM(1000 11 1e-06) f_classif 50	6,92 0,10	202,15 7,21	0,46 0,0004	0,75 0,001	0,49 0,006	0,44 0,001
DT(6) chi2 50	4,62 0,03	125,46 6,02	0,59 7,02e-05	0,82 6,67e-06	0,64 0,0002	0,55 0,0005
RF(100 10) chi2 50	4,38 0,006	117,42 7,78	0,63 3,61e-05	0,84 5,79e-06	0,69 3,04e-05	0,58 4,10e-05
Ada(100 1) f_classif 50	4,51 0,0005	115,97 2,68	0,63 5,67e-05	0,83 1,99e-05	0,66 0,0002	0,60 2,40e-05
Ada(500 10) 0,5 0,7	3,87 0,0004	92,23 0,004	0,60 3,98e-07	0,77 2,20e-07	0,51 4,55e-07	0,75 1,43e-06
RF(500 9) 0,1 0,4 chi2 30	3,72 6,73e-05	85,67 0,28	0,62 2,78e-06	0,79 2,79e-06	0,54 8,35e-06	0,73 5,69e-07

% объектов допускает ошибку больше либо равную 10, а на 72 % объектов ошибка составляет не более трех.

Выводы по главе 3

В данной главе представлены результаты проведенных тестов. Были протестированы различные алгоритмы машинного обучения, решающие задачу классификации, также для улучшения качества их работы были использованы методы ансамблирования. Для уменьшения времени работы моделей, а также повышения их качества был применен метод отбора признаков, который сократил количество признаков. Во многих случаях использование алгоритма отбора улучшило показатели, тестируемых методов.

Также были рассмотрены алгоритмы описанные в разделе 2.4 и подразделе 2.6.2. При этом второй является наилучшим, при использовании в качестве модели случайного леса, состоящего из 500 деревьев, максимальная высота которых равна 9, с коэффициентами доверия равными 0,1 и 0,4 и выбором лучших 30 признаков с помощью chi-square test.

ЗАКЛЮЧЕНИЕ

В рамках данной работы был проведен обзор методов, которые используются с целью оценки способностей игрока. Описаны способы и случаи их применения, а также их хорошие качества и недостатки. Вследствие чего, была поставлена задача, заключающаяся в извлечении из записей матчей по компьютерной игре Counter-Strike: Global Offensive времени реакции игроков.

При разработке алгоритмов была описана структура демо файла, также разработан алгоритм, находящий случаи, время реакции в которых необходимо вычислить. Предложен первоначальный алгоритм, основанный на гипотезе о том, что в момент реакции происходит резкое изменение положения прицела. Затем рассмотрены способы улучшения данного алгоритма с помощью методов машинного обучения, а также приведены признаки, которые необходимо извлечь для получения датасета.

Алгоритм был реализован частично на языке Golang, с помощью которого происходит извлечений данных из демо файла, а обработка этих данных на языке Python 3 с использованием библиотек для машинного обучения. Тестирование производилось при использовании различных методов из библиотеки scikit-learn. Алгоритм показывает хорошие результаты и применение методов машинного обучения улучшает его качество.

Таким образом был получен алгоритм, который принимает на вход демо файл с записью матча, находит и извлекает все рассматриваемые случаи. Далее происходит их обработка с помощью алгоритмов машинного обучения и на выходе получаем список рассматриваемых случаев и значение реакции игрока в них.

Полученный алгоритм имеет среднюю абсолютную ошибку равную 3,72 тика что составляет около 29 миллисекунд. Каждое официальное соревнование проходит с задержкой не более 50 миллисекунд, что больше, чем получившееся значение. Также по данным сайта humanbenchmark.com [35] среднеарифметическое значение реакции человека – 284 миллисекунд, и получившееся отклонение алгоритма около 10 %, что является хорошим показателем.

Результаты, выдаваемые алгоритмом, можно использовать не просто как сырые данные о реакции игроков в конкретном матче. С их помощью можно

выявлять игроков, которые нарушают правила. Например, если у человека в матче среднее время реакции намного ниже, чем обычно, то возможно он использует специальную программу, которая автоматически наводит его прицел на противника, и его стоит проверить. Также время реакции можно использовать как дополнительный параметр рейтинговых систем, описанных в главе 1, что сделает их еще более честными, так как они будут учитывать физические данные игроков.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 *Hamari J., Sjöblom M.* What is eSports and why do people watch it? // Internet research. — 2017.
- 2 Найди меня, или Главный инструмент футбольного скаута [Электронный ресурс]. — URL: <https://www.sports.ru/tribuna/blogs/tsaplind/927166.html> (дата обращения 01.11.2019).
- 3 Counter-Strike: Global Offensive [Электронный ресурс]. — URL: <https://blog.counter-strike.net/> (дата обращения 11.11.2019).
- 4 Trueskill through time: Revisiting the history of chess / P. Dangauthier [et al.] // Advances in neural information processing systems. — 2008. — P. 337–344.
- 5 About CS:GO [Электронный ресурс]. — URL: <https://blog.counter-strike.net/index.php/about/> (дата обращения 11.11.2019).
- 6 DEM format [Электронный ресурс]. — URL: https://developer.valvesoftware.com/wiki/DEM_Format (дата обращения 10.12.2019).
- 7 Counter-Strike: Global Offensive - Demo Parsing Tool [Электронный ресурс]. — URL: <https://github.com/ValveSoftware/csgo-demoinfo> (дата обращения 10.12.2019).
- 8 C# library for reading CS:GO-Demos [Электронный ресурс]. — URL: <https://github.com/StatsHelix/demoinfo> (дата обращения 10.12.2019).
- 9 High performance CS:GO demo parser for Go [Электронный ресурс]. — URL: <https://github.com/markus-wa/demoinfocs-golang> (дата обращения 10.12.2019).
- 10 Node.js library for parsing Counter-Strike: Global Offensive demo files [Электронный ресурс]. — URL: <https://github.com/saul/demofile> (дата обращения 10.12.2019).
- 11 The Go programming language [Электронный ресурс]. — URL: <https://golang.org/> (дата обращения 14.12.2019).

- 12 What is that rating thing in stats? [Электронный ресурс]. — 2010. — URL: <https://www.hltv.org/news/4094/what-is-that-rating-thing-in-stats> (дата обращения 04.12.2019).
- 13 CS:GO News & Coverage [Электронный ресурс]. — URL: <https://www.hltv.org> (дата обращения 04.12.2019).
- 14 Introducing rating 2.0 [Электронный ресурс]. — 2017. — URL: <https://www.hltv.org/news/20695/introducing-rating-20> (дата обращения 04.12.2019).
- 15 *Elo A. E.* The rating of chessplayers, past and present. — Arco Pub., 1978.
- 16 FACEIT [Электронный ресурс]. — URL: <https://www.faceit.com/en> (дата обращения 20.12.2019).
- 17 *Buckley D., Chen K., Knowles J.* Rapid skill capture in a first-person shooter // IEEE Transactions on Computational Intelligence and AI in Games. — 2015. — Vol. 9, no. 1. — P. 63–75.
- 18 *Huffman D. A.* A method for the construction of minimum-redundancy codes // Proceedings of the IRE. — 1952. — Vol. 40, no. 9. — P. 1098–1101.
- 19 *Welch T. A.* A technique for high-performance data compression // Computer. — 1984. — No. 6. — P. 8–19.
- 20 *Ho T. K.* Random decision forests // Proceedings of 3rd international conference on document analysis and recognition. Vol. 1. — IEEE. 1995. — P. 278–282.
- 21 *Quinlan J. R.* Induction of decision trees // Machine learning. — 1986. — Vol. 1, no. 1. — P. 81–106.
- 22 *Draper N. R., Smith H.* Applied regression analysis. Vol. 326. — John Wiley & Sons, 1998.
- 23 Прикладная статистика: Классификация и снижение размерности / С. Айвазян [и др.]. — 1989.
- 24 *Larose D. T., Larose C. D.* Discovering knowledge in data: an introduction to data mining. Vol. 4. — John Wiley & Sons, 2014.
- 25 An introduction to support vector machines and other kernel-based learning methods / N. Cristianini, J. Shawe-Taylor, [et al.]. — Cambridge university press, 2000.

- 26 *Freund Y., Schapire R. E.* A decision-theoretic generalization of on-line learning and an application to boosting // European conference on computational learning theory. — Springer. 1995. — P. 23–37.
- 27 Scikit-learn: machine learning in Python [Электронный ресурс]. — URL: <https://scikit-learn.org/stable/index.html> (дата обращения 04.02.2020).
- 28 *Willmott C. J., Matsuura K.* Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance // Climate research. — 2005. — Vol. 30, no. 1. — P. 79–82.
- 29 *Battaglia G. J.* Mean square error // AMP Journal of Technology. — 1996. — Vol. 5, no. 1. — P. 31–36.
- 30 *Powers D. M.* Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. — 2011.
- 31 *Rodriguez J. D., Perez A., Lozano J. A.* Sensitivity analysis of k-fold cross validation in prediction error estimation // IEEE transactions on pattern analysis and machine intelligence. — 2009. — Vol. 32, no. 3. — P. 569–575.
- 32 *Patro S., Sahu K. K.* Normalization: A preprocessing stage // arXiv preprint arXiv:1503.06462. — 2015.
- 33 *Pearson K. X.* On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling // The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. — 1900. — Vol. 50, no. 302. — P. 157–175.
- 34 *Freedman D. A.* Statistical models: theory and practice. — cambridge university press, 2009.
- 35 Reaction Time Statistics [Электронный ресурс]. — URL: <https://humanbenchmark.com/tests/reactiontime/statistics> (дата обращения 15.03.2020).

ПРИЛОЖЕНИЕ А. ОПИСАНИЕ ИЗВЛЕЧЕННЫХ ПРИЗНАКОВ ДЛЯ КАЖДОГО ТИКА

Для простоты понимания разобьем признаки на группы, и для каждого из них предъявим тип возможных значений. Также каждому из признаков назначим сокращенное название, на английском языке, для удобства использования.

Условно признаки можно поделить на три группы:

- а) Общие для обоих участников.
- б) Связанные только с атакующим.
- в) Связанные только с защищающимся.

Первый тип состоит из признаков, которые описывают общее состояние рассматриваемой ситуации либо связаны имеют отношение к обоим игрокам одновременно:

- а) Tick – глобальный номер тика, может принимать значение от 0 до самого последнего тика в записи. Тип – целочисленное положительное.
- б) TickIndex – номер тика относительно начала рассматриваемой ситуации. Также является целочисленным положительным значением.
- в) DistanceBetweenPlayers – расстояние между игроками в данный момент времени. Значения этого показателя являются положительными вещественными числами.
- г) DistanceBetweenPlayersDecrease – уменьшилось ли расстояние между двумя игроками, по сравнению с предыдущим тиком. Этот показатель может принимать два значения ноль или один, так как расстояние между игроками может либо уменьшиться (значение один), либо увеличиться или не измениться (значение ноль).

Последние две группы являются одинаковыми и отличаются тем, кого из игроков они определяют, поэтому признаки, описывающие атакующего, будут иметь префикс Attacker, а противника – Victim. Для удобства, при их описании будем разделять признаки на подгруппы, по типу принимаемых значений. Выглядят они так:

- а) Вещественные значения:

- 1) PositionX, PositionY, PositionZ – положение игрока в пространстве. Представляется как три координаты x, y и z, и используются они как три разных параметра.

- 2) ViewX, ViewY – положение прицела. Это две координаты x и y, которые измеряются в градусах, и также каждая из координат используется как отдельный признак.
- 3) ViewXDiff, ViewYDiff – изменение координат прицела относительно предыдущего тика, также две координаты x и y, причем каждая является отдельным признаком.
- 4) ViewXDeviationFromVictim – разница между x координатой прицела и положением противника в градусах относительно текущего игрока, то есть высчитывается какое было бы значение координаты x прицела, если бы игрок в данный момент был наведен на противника, тем самым получаем значение, на которое нужно изменить координату x прицела для того, чтобы навести его на оппонента.
- 5) CurSpeed – значение скорости изменения положения прицела на текущем тике.
- 6) PrevSpeed – значение скорости изменения положения прицела на предыдущем тике.
- 7) CurAvgSpeedForLast3Ticks – среднее значение скорости изменения положения прицела на последних трех тиках, учитывая текущий тик.
- 8) CurAvgSpeedForLast5Ticks – тоже что и предыдущее, только на пяти тиках.
- 9) CurAvgSpeedForLast10Ticks – тоже что и предыдущее, только на десяти тиках.
- 10) PrevAvgSpeedForLast3Ticks – среднее значение скорости изменения положения прицела на последних трех тиках, не учитывая текущий тик.
- 11) PrevAvgSpeedForLast5Ticks – тоже что и предыдущее, только на пяти тиках.
- 12) PrevAvgSpeedForLast10Ticks – тоже что и предыдущее, только на десяти тиках.
- 13) CurSpeedPrevSpeedDiff – это разность между CurSpeed и PrevSpeed.
- 14) CurSpeedPrevAvgSpeedForLast3TicksDiff – это разность между CurSpeed и PrevAvgSpeedForLast3Ticks.

- 15) CurSpeedPrevAvgSpeedForLast5TicksDiff – это разность между CurSpeed и PrevAvgSpeedForLast5Ticks.
- 16) CurSpeedPrevAvgSpeedForLast10TicksDiff – это разность между CurSpeed и PrevAvgSpeedForLast10Ticks.
- 17) CurSpeedCurAvgSpeedForLast3TicksDiff – это разность между CurSpeed и CurAvgSpeedForLast3Ticks.
- 18) CurSpeedCurAvgSpeedForLast5TicksDiff – это разность между CurSpeed и CurAvgSpeedForLast5Ticks.
- 19) CurSpeedCurAvgSpeedForLast10TicksDiff – это разность между CurSpeed и CurAvgSpeedForLast10Ticks.
- 20) CurAcc – значение ускорения изменения положения прицела на текущем тике.
- 21) PrevAcc – значение ускорения изменение положения прицела на предыдущем тике.
- 22) CurAvgAccForLast3Ticks – среднее значение ускорения изменения положения прицела на последних треках тиках, учитывая текущий.
- 23) CurAvgAccForLast5Ticks – тоже что и предыдущее, только на пяти тиках.
- 24) CurAvgAccForLast10Ticks – тоже что и предыдущее, только на десяти тиках.
- 25) PrevAvgAccForLast3Ticks – среднее значение ускорения изменения положения прицела на последних треках тиках, не включая текущий.
- 26) PrevAvgAccForLast5Ticks – тоже что и предыдущее, только на пяти тиках.
- 27) PrevAvgAccForLast10Ticks – тоже что и предыдущее, только на десяти тиках.
- 28) CurAccPrevSpeedDiff – это разность между CurAcc и PrevAcc.
- 29) CurAccPrevAvgAccForLast3TicksDiff – это разность между CurAcc и PrevAvgAccForLast3Ticks.
- 30) CurAccPrevAvgAccForLast5TicksDiff – это разность между CurAcc и PrevAvgAccForLast5Ticks.
- 31) CurAccPrevAvgAccForLast10TicksDiff – это разность между CurAcc и PrevAvgAccForLast10Ticks.

- 32) CurAccCurAvgAccForLast3TicksDiff – это разность между CurAcc и CurAvgAccForLast3Ticks.
- 33) CurAccCurAvgAccForLast5TicksDiff – это разность между CurAcc и CurAvgAccForLast5Ticks.
- 34) CurAccCurAvgAccForLast10TicksDiff – это разность между CurAcc и CurAvgAccForLast10Ticks.

б) Целочисленные значения:

- 1) SpottersCount – количество противников, которые видят текущего игрока.
- 2) SpottedCount – количество противников, которое видит текущий игрок.
- 3) HP – количество оставшихся единиц здоровья у текущего игрока. Принимает значение от 0 до 100.
- 4) Armor – количество оставшихся единиц бронежилета у текущего игрока. Принимает значение от 0 до 100.
- 5) CurEquipVal – значение отражающее качество имеющейся, на текущий момент игроком амуниции. То есть это сумма денег, которую нужно потратить, для покупки всех вещей.
- 6) InitEquipVal – тоже что и предыдущее, но не на текущий момент, а перед началом раунда.
- 7) Money – количество денег у игрока в данный момент времени. Значение не может быть меньше нуля.
- 8) InitEquipCurEquipDiff – разница между InitEquipVal и CurEquipVal.
- 9) CashSpentThisRound – количество денег, которое потратил игрок в этом раунде для покупки амуниции.
- 10) CashSpentTotal – общее количество денег, которое потратил игрок к этому времени

в) Целочисленные признаки имеющие только два возможных значения – ноль или один:

- 1) WeaponClass – огнестрельное ли оружие в руках у текущего игрока, ноль, если не имеется, и один в противном случае.
- 2) MoveXViewToVictim – стал ли прицел игрока ближе к противнику, по сравнению с предыдущим моментом времени, то есть равное единице, если отклонение текущего положения прицела от

положения противника меньше, чем было в предыдущий момент, иначе значение ноль.

- 3) HasDefuseKit – имеется ли в амуниции игрока приспособление, позволяющее быстро разминировать заложенную бомбу. Этот параметр имеет значение один, если это правда, иначе ноль. Только игрок команды спецназа может иметь значение этого параметра равное единице, так как только они имеют возможность разминировать бомбу.
- 4) HasHelmet – имеется ли у игрока шлем. Имеет значение один если он есть, иначе ноль. Каждый игрок имеет возможность купить шлем вместе с бронежилетом, либо только бронежилет. Шлем повышает вероятность того, что противник не сможет убить вас с одного попадания в голову, так как она является самым уязвимым местом.
- 5) Team – означает за какую из двух сторон выступает игрок, а именно спецназ (значение один), или террористы (значение ноль).
- 6) IsAirborne – находится ли в данный момент игрок в воздухе. Значение один, если да, иначе нет. Это может быть прыжок вверх либо падение вниз, с какой-либо возвышенности.
- 7) IsBlinded – является ли игрок ослепленным в данный момент. Значение один, если да, иначе ноль. Ослепить игрока можно с помощью специальной светошумовой гранаты. В этот момент экран игрока ослепляется белым светом, он перестает видеть и слышать все, что происходит вокруг.
- 8) IsScoped – использует ли оптический прицел оружия. Значение один, если да, иначе нет. Оптический прицел имеют только снайперские винтовки.
- 9) IsDucking – перемещается ли игрок в положении согнутых коленей. Значение один если да, иначе ноль. Этот способ позволяет перемещаться бесшумно, что не позволит противникам услышать ваше приближение.

Итого получается 118 различных признаков.

ПРИЛОЖЕНИЕ Б. ПРИЗНАКИ, ПОЛУЧЕННЫЕ ПУТЕМ ПРИМЕНЕНИЯ МЕТОДОВ ОТБОРА

Далее будут приведены используемые методы отбора признаков и признаки, которые были им выбраны.

а) Лучшие 5 признаков отобранные с помощью ANOVA F-test:

- 1) AttackerCurSpeed
- 2) AttackerPrevSpeed
- 3) AttackerCurAvgSpeedForLast3Ticks
- 4) AttackerPrevAvgSpeedForLast3Ticks
- 5) AttackerCurAvgSpeedForLast5Ticks

б) Лучшие 5 признаков отобранные с помощью chi-square test:

- 1) AttackerCurSpeed
- 2) AttackerPrevSpeed
- 3) AttackerCurAvgSpeedForLast3Ticks
- 4) AttackerPrevAvgSpeedForLast3Ticks
- 5) AttackerCurAvgSpeedForLast5Ticks

в) 10 признаков отобранные с помощью ANOVA F-test:

- 1) DistanceBetweenPlayers
- 2) AttackerWeaponClass
- 3) AttackerCurSpeed
- 4) AttackerPrevSpeed
- 5) AttackerCurAvgSpeedForLast3Ticks
- 6) AttackerPrevAvgSpeedForLast3Ticks
- 7) AttackerCurAvgSpeedForLast5Ticks
- 8) AttackerPrevAvgSpeedForLast5Ticks
- 9) AttackerCurAvgSpeedForLast10Ticks
- 10) AttackerPrevAvgSpeedForLast10Ticks

г) Лучшие 10 признаков отобранные с помощью chi-square test

- 1) DistanceBetweenPlayers
- 2) AttackerCurSpeed
- 3) AttackerPrevSpeed
- 4) AttackerCurAvgSpeedForLast3Ticks
- 5) AttackerPrevAvgSpeedForLast3Ticks
- 6) AttackerCurAvgSpeedForLast5Ticks

- 7) AttackerPrevAvgSpeedForLast5Ticks
- 8) AttackerCurAvgSpeedForLast10Ticks
- 9) AttackerPrevAvgSpeedForLast10Ticks
- 10) VictimIsScoped

д) Лучшие 20 признаков отобранные с помощью ANOVA F-test:

- 1) TickIndex
- 2) DistanceBetweenPlayers
- 3) AttackerWeaponClass
- 4) AttackerCurSpeed
- 5) AttackerPrevSpeed
- 6) AttackerCurAvgSpeedForLast3Ticks
- 7) AttackerPrevAvgSpeedForLast3Ticks
- 8) AttackerCurAvgSpeedForLast5Ticks
- 9) AttackerPrevAvgSpeedForLast5Ticks
- 10) AttackerCurAvgSpeedForLast10Ticks
- 11) AttackerPrevAvgSpeedForLast10Ticks
- 12) AttackerCurSpeedPrevAvgSpeedForLast5TicksDiff
- 13) AttackerCurSpeedPrevAvgSpeedForLast10TicksDiff
- 14) AttackerCurSpeedCurAvgSpeedForLast5TicksDiff
- 15) AttackerCurSpeedCurAvgSpeedForLast10TicksDiff
- 16) AttackerCurAvgAccForLast3Ticks
- 17) AttackerPrevAvgAccForLast3Ticks
- 18) AttackerCurAvgAccForLast5Ticks
- 19) AttackerPrevAvgAccForLast5Ticks
- 20) AttackerCurAvgAccForLast10Ticks

е) Лучшие 20 признаков отобранные с помощью chi-square test:

- 1) TickIndex
- 2) DistanceBetweenPlayers
- 3) AttackerViewY
- 4) AttackerSpottedCount
- 5) AttackerWeaponClass
- 6) AttackerCurSpeed
- 7) AttackerPrevSpeed
- 8) AttackerCurAvgSpeedForLast3Ticks

- 9) AttackerPrevAvgSpeedForLast3Ticks
- 10) AttackerCurAvgSpeedForLast5Ticks
- 11) AttackerPrevAvgSpeedForLast5Ticks
- 12) AttackerCurAvgSpeedForLast10Ticks
- 13) AttackerPrevAvgSpeedForLast10Ticks
- 14) AttackerTeam
- 15) AttackerIsAirborne
- 16) VictimViewY
- 17) VictimCurAvgSpeedForLast3Ticks
- 18) VictimCurAvgSpeedForLast5Ticks
- 19) VictimTeam
- 20) VictimIsScoped

ж) Лучшие 30 признаков отобранные с помощью ANOVA F-test:

- 1) TickIndex
- 2) DistanceBetweenPlayers
- 3) AttackerWeaponClass
- 4) AttackerCurSpeed
- 5) AttackerPrevSpeed
- 6) AttackerCurAvgSpeedForLast3Ticks
- 7) AttackerPrevAvgSpeedForLast3Ticks
- 8) AttackerCurAvgSpeedForLast5Ticks
- 9) AttackerPrevAvgSpeedForLast5Ticks
- 10) AttackerCurAvgSpeedForLast10Ticks
- 11) AttackerPrevAvgSpeedForLast10Ticks
- 12) AttackerCurSpeedPrevAvgSpeedForLast3TicksDiff
- 13) AttackerCurSpeedPrevAvgSpeedForLast5TicksDiff
- 14) AttackerCurSpeedPrevAvgSpeedForLast10TicksDiff
- 15) AttackerCurSpeedCurAvgSpeedForLast5TicksDiff
- 16) AttackerCurSpeedCurAvgSpeedForLast10TicksDiff
- 17) AttackerCurAvgAccForLast3Ticks
- 18) AttackerPrevAvgAccForLast3Ticks
- 19) AttackerCurAvgAccForLast5Ticks
- 20) AttackerPrevAvgAccForLast5Ticks
- 21) AttackerCurAvgAccForLast10Ticks

- 22) AttackerPrevAvgAccForLast10Ticks
- 23) VictimCurSpeed
- 24) AttackerPrevSpeed
- 25) VictimCurAvgSpeedForLast3Ticks
- 26) VictimPrevAvgSpeedForLast3Ticks
- 27) VictimCurAvgSpeedForLast5Ticks
- 28) VictimPrevAvgSpeedForLast5Ticks
- 29) VictimCurAvgSpeedForLast10Ticks
- 30) VictimPrevAvgSpeedForLast10Ticks

и) Лучшие 30 признаков отобранные с помощью chi-square test:

- 1) TickIndex
- 2) DistanceBetweenPlayers
- 3) AttackerViewY
- 4) AttackerSpottedCount
- 5) AttackerWeaponClass
- 6) AttackerCurSpeed
- 7) AttackerPrevSpeed
- 8) AttackerCurAvgSpeedForLast3Ticks
- 9) AttackerPrevAvgSpeedForLast3Ticks
- 10) AttackerCurAvgSpeedForLast5Ticks
- 11) AttackerPrevAvgSpeedForLast5Ticks
- 12) AttackerCurAvgSpeedForLast10Ticks
- 13) AttackerPrevAvgSpeedForLast10Ticks
- 14) AttackerHasHelmet
- 15) AttackerTeam
- 16) AttackerIsAirborne
- 17) VictimViewX
- 18) VictimViewY
- 19) VictimSpottersCount
- 20) VictimCurSpeed
- 21) AttackerPrevSpeed
- 22) VictimCurAvgSpeedForLast3Ticks
- 23) VictimPrevAvgSpeedForLast3Ticks
- 24) VictimCurAvgSpeedForLast5Ticks

- 25) VictimPrevAvgSpeedForLast5Ticks
- 26) VictimCurAvgSpeedForLast10Ticks
- 27) VictimPrevAvgSpeedForLast10Ticks
- 28) VictimTeam
- 29) VictimIsAirborne
- 30) VictimIsScoped

к) Лучшие 50 признаков отобранные с помощью ANOVA F-test:

- 1) TickIndex
- 2) DistanceBetweenPlayers
- 3) AttackerViewX
- 4) AttackerViewY
- 5) AttackerSpottedCount
- 6) AttackerWeaponClass
- 7) AttackerCurSpeed
- 8) AttackerPrevSpeed
- 9) AttackerCurAvgSpeedForLast3Ticks
- 10) AttackerPrevAvgSpeedForLast3Ticks
- 11) AttackerCurAvgSpeedForLast5Ticks
- 12) AttackerPrevAvgSpeedForLast5Ticks
- 13) AttackerCurAvgSpeedForLast10Ticks
- 14) AttackerPrevAvgSpeedForLast10Ticks
- 15) AttackerCurSpeedPrevAvgSpeedForLast3TicksDiff
- 16) AttackerCurSpeedPrevAvgSpeedForLast5TicksDiff
- 17) AttackerCurSpeedPrevAvgSpeedForLast10TicksDiff
- 18) AttackerCurSpeedCurAvgSpeedForLast3TicksDiff
- 19) AttackerCurSpeedCurAvgSpeedForLast5TicksDiff
- 20) AttackerCurSpeedCurAvgSpeedForLast10TicksDiff
- 21) AttackerPrevAcc
- 22) AttackerCurAvgAccForLast3Ticks
- 23) AttackerPrevAvgAccForLast3Ticks
- 24) AttackerCurAvgAccForLast5Ticks
- 25) AttackerPrevAvgAccForLast5Ticks
- 26) AttackerCurAvgAccForLast10Ticks
- 27) AttackerPrevAvgAccForLast10Ticks

- 28) AttackerHasHelmet
- 29) AttackerInitEquipCurEquipDiff
- 30) AttackerTeam
- 31) AttackerCashSpentThisRound
- 32) AttackerIsAirborne
- 33) VictimViewX
- 34) VictimViewY
- 35) VictimWeaponClass
- 36) VictimViewDeviationFromVictimX
- 37) VictimCurSpeed
- 38) AttackerPrevSpeed
- 39) VictimCurAvgSpeedForLast3Ticks
- 40) VictimPrevAvgSpeedForLast3Ticks
- 41) VictimCurAvgSpeedForLast5Ticks
- 42) VictimPrevAvgSpeedForLast5Ticks
- 43) VictimCurAvgSpeedForLast10Ticks
- 44) VictimPrevAvgSpeedForLast10Ticks
- 45) VictimCurSpeedPrevAvgSpeedForLast10TicksDiff
- 46) VictimCurSpeedCurAvgSpeedForLast10TicksDiff
- 47) VictimHasHelmet
- 48) VictimInitEquipCurEquipDiff
- 49) VictimTeam
- 50) VictimIsScoped

л) Лучшие 50 признаков отобранные с помощью chi-square test:

- 1) TickIndex
- 2) DistanceBetweenPlayers
- 3) DistanceBetweenPlayersDecrease
- 4) AttackerViewX
- 5) AttackerViewY
- 6) AttackerSpottedCount
- 7) AttackerWeaponClass
- 8) AttackerMoveXViewToVictim
- 9) AttackerCurSpeed
- 10) AttackerPrevSpeed

- 11) AttackerCurAvgSpeedForLast3Ticks
- 12) AttackerPrevAvgSpeedForLast3Ticks
- 13) AttackerCurAvgSpeedForLast5Ticks
- 14) AttackerPrevAvgSpeedForLast5Ticks
- 15) AttackerCurAvgSpeedForLast10Ticks
- 16) AttackerPrevAvgSpeedForLast10Ticks
- 17) AttackerCurSpeedPrevAvgSpeedForLast10TicksDiff
- 18) AttackerCurSpeedCurAvgSpeedForLast10TicksDiff
- 19) AttackerArmor
- 20) AttackerCurEquipVal
- 21) AttackerHasHelmet
- 22) AttackerInitEquipVal
- 23) AttackerInitEquipCurEquipDiff
- 24) AttackerTeam
- 25) AttackerCashSpentThisRound
- 26) AttackerIsAirborne
- 27) AttackerIsScoped
- 28) AttackerIsDucking
- 29) VictimPositionY
- 30) VictimViewX
- 31) VictimViewY
- 32) VictimSpottersCount
- 33) VictimWeaponClass
- 34) VictimViewDeviationFromVictimX
- 35) VictimMoveViewToVictim
- 36) VictimCurSpeed
- 37) AttackerPrevSpeed
- 38) VictimCurAvgSpeedForLast3Ticks
- 39) VictimPrevAvgSpeedForLast3Ticks
- 40) VictimCurAvgSpeedForLast5Ticks
- 41) VictimPrevAvgSpeedForLast5Ticks
- 42) VictimCurAvgSpeedForLast10Ticks
- 43) VictimPrevAvgSpeedForLast10Ticks
- 44) VictimHasDefuseKit

- 45) VictimHasHelmet
- 46) VictimInitEquipCurEquipDiff
- 47) VictimTeam
- 48) VictimIsAirborne
- 49) VictimIsBlinded
- 50) VictimIsScoped