Санкт-Петербургский государственный университет

БЕРДИН Егор Максимович

Выпускная квалификационная работа

Анализ социального взаимодействия на примере повторяющейся игры «Дилемма заключенного»

Уровень образования: магистратура
Направление: 01.04.02 «прикладная математика и информатика»
Основная образовательная программа: ВМ.5504 «Исследование операций и системный анализ»

Научный руководитель:
профессор кафедры МТИСР, доктор физ.-мат. наук,
Парилина Е.М.

Рецензент: директор департамента, Акционерное общество «Группа компаний «Эталон»,
Марковкин М.В.

Санкт-Петербург
2020 год

Содержание

Введение

В повседневной жизни люди довольно часто взаимодействуют между собой. Это может быть общение в кругу семьи, с коллегами на работе или соседями по дому. Но как происходит взаимодействие? Есть ли какие-нибудь закономерности, согласно которым мы общаемся с одними, а другим попросту не доверяем? Если постараться описать поведение каждого человека с помощью теории игр, мы сможем ответить на этот вопрос.

Целью данной ВКР является выявление тех факторов, которые могут влиять на взаимодействие людей. Сначала мы формализуем самые распространенные модели поведения. Затем, с помощью повторяющейся игры «дилемма заключенного», будут проведены эксперименты непосредственно с людьми. В данном эксперименте двум игрокам будут предложены два варианта взаимодействия: довериться (T) или обмануть (C).

После проведения эксперимента проводится запись результатов и анкетирование испытуемых. На основе полученных анкет необходимо обнаружить факторы, которые влияют на модель поведения. На основе данных, полученных в ходе проведения игр, необходимо выявить стратегии, которые мы уже формализовали и найти новые.

На заключительном этапе можно запустить симуляцию, в которой модели поведения будут соревноваться друг с другом. Эта модель будет носить эволюционный характер, то есть менее удачные модели будут заменяться на те, которые показали наилучший результат. В результате эволюции можно будет выявить наилучшую модель поведения.

Обзор литературы

При написании данной дипломной работы была использована научная литература, Интернет-ресурсы и статьи для поиска необходимой информации.

Для того, чтобы подойти к проведению эксперимента с точки зрения теории игр, необходимо было изучить источники [1-5]. Благодаря данным ресурсам стало возможно математически формализовать выявленные модели поведения игроков.

В процессе работы понадобилось написать бота на платформе «Telegram». Было решено писать программу на языке Python. На Интернет-ресурсе [15] были подробно описаны начальные этапы подготовки к написанию бота на данном языке. В источниках [16-18] были продемонстрированы минимальные примеры рабочего бота, который умел отвечать на стандартные сообщения и выполнять простые команды. С помощью советов и рекомендаций из данного источника была написана первая рабочая версия бота. Для улучшения работоспособности кода был изучен ресурс [19]. Эксперимент по проведению игры подразумевает хранение большого количества данных. Для этой цели было решено воспользоваться бесплатной онлайн базой данный MongoDB [20]. Инструкция по работе с данной базой была так же изучена на ресурсе [19]. Благодаря статье на Интернет-ресурсе [21], было решено воспользоваться сервисом Windscribe-VPN [22] для стабильного запуска и работы бота в ходе проведения эксперимента.

В конце эксперимента испытуемые должны пройти анкетирование. Для этой цели лучше всего подойдет Интернет-ресурс Google Формы [23]. Благодаря удобной системе хранения, анкетные данные могут быть сохранены в таблицах Excel, с которыми удобно взаимодействовать, используя различные методы, реализованные на языке программирования Python.

В процессе работы возникла необходимость в кластерном анализе. Чтобы более детально разобраться в кластерном анализе, были разобраны источники [6-9]. Изучив их, стало ясно, что для текущих задач подойдет наивный классификатор Байеса. В источнике [12] хорошо описан наивный классификатор Байеса, который решено было применить в данной работе.

Чтобы установить связь между факторами и поведением игроков, была изучена литература [10,14]. В условиях данной работы и имеющих-

ся факторов было решено воспользоваться сопряженными таблицами и критерием Хи-квадрат, который подробно описан в источнике [11].

Чтобы провести эксперимент наилучшим образом, дополнительно была изучена статья [13], в которой описывался примерно похожий эксперимент. Это было сделано для того, чтобы понять, на какие аспекты эксперимента стоит обратить наибольшее внимание.

§1. Формализация стратегий поведения

Имеем игру $G = (N, (S_i)_{i \in N}, (u_i)_{i \in N})$, где N – набор игроков, S_i - набор действий для i-ого игрока, $u_i : S \to R$ – выигрыш игрока i, где $S = S_1 \times S_2 \times \ldots \times S_n$ [3,4].

Смешанное расширение игры G - это игра $= (N, (\Sigma_i)_{i \in N}, (U_i)_{i \in N})$, где $\Sigma_i = \Delta(S_i)$ - смешанные стратегии i-ого игрока. Функция выигрыша

$$U_i(\sigma) = \sum_{(s_1, s_2, ..., s_n) \in S} u_i(s_1, s_2, ..., s_n)\sigma_1(s_1)\sigma_2(s_2)...\sigma_n(s_n),$$

где $\sigma = (\sigma_1, \sigma_2, ..., \sigma_n) \in \Sigma = (\Sigma_1, \Sigma_2, ..., \Sigma_n)$.

Поскольку игроки встречаются друг с другом неоднократно в повторяющихся играх, они собирают информацию по ходу игры. Информация, доступная каждому игроку на этапе $t + 1$, является действиями всех игроков на первых t этапах игры. Поэтому мы определим, для каждого $t > 0$ множество историй как

$$H(t) := S^t = \underbrace{S \times S \times ... \times S}_{t\ times}.$$

В начале игры при $t = 0$ определим историю как $H(0) = \{\varnothing\}$.

Стратегия поведения для игрока i – это план действий, который инструктирует игрока, какое смешанное действие следует выполнять после каждой возможной истории. Стратегия поведения для игрока i в T-шаговой игре - это функция, связывающая смешанное действие с каждой историей, длина которой меньше T:

$$\tau_i : \bigcup_{t=0}^{T-1} H(t) \to \Sigma_i.$$

Пусть τ_e - стратегия поведения противника, $\mathbb{T}$ - вариант довериться, C - вариант обмануть, а под символом «$\cdot$» будем подразумевать любую историю. Обозначим распространённые модели поведения:

- Доверчивый: $\tau(\cdot) = \mathbb{T}$;
- Обманщик: $\tau(\cdot) = C$;
- Имитатор:
  - $\tau(\varnothing) = \mathbb{T}$,
  - $\tau(H(t)) = \mathbb{T}$, если $\tau_e(H(t-1)) = \mathbb{T}, t \geqslant 1$,
  - $\tau(H(t)) = C$, если $\tau_e(H(t-1)) = C, t \geqslant 1$;
- Хитрый:
  - $\tau(\varnothing) = \mathbb{T}$,
  - $\tau(H(1)) = C$,
  - $\tau(H(2)) = \mathbb{T}$,
  - Для $3 \leqslant t < T$:
    - $\tau(H(t)) = C$, если $\tau_e(H(t)) = \mathbb{T}$ или $\tau_e(H(t)) = C, t \in \{0, 1, 2\}$,
    - Иначе ведет себя как имитатор,
  - $\tau(H(\mathbb{T})) = C$;
- Злопамятный:
  - $\tau(\varnothing) = \mathbb{T}$,
  - $\tau(H(\mathbb{T})) = \mathbb{T}$, если $\tau_e(H(t-1)) = \tau(H(t-1)) = \mathbb{T}$, $t \geqslant 1$
  - $\tau(H(\mathbb{T})) = C$, иначе.

Доверчивый — игрок, который всегда доверяется. Обманщик — который всегда обманывает. Имитатор повторяет предыдущий ход противника. Хитрый игрок пытается установить поведение своего противника: если он понимает, что перед ним доверчивый или злопамятный человек, то он обманывает. Иначе он будет вести себя как имитатор. Злопамятный — это

такой игрок, который будет всегда обманывать, если его хоть раз до этого обманули.

Если мы рассматриваем задачу социального взаимодействия, то должны иметь ввиду тот факт, что в социуме возможно некоторое недопонимание при взаимодействии. Поэтому имеет смысл добавить случайную составляющую в рассмотренные модели. Обозначим за α – вероятность совершения ошибки, $\alpha \in [0, 1]$. С учетом этого переобозначим наши модели:

- Доверчивый: $\tau(\cdot) = [(1-\alpha)(\mathbb{T}), \alpha(C)]$;
- Обманщик: $\tau(\cdot) = [(1-\alpha)(C), \alpha(\mathbb{T})]$;
- Имитатор:
  - $\tau(\varnothing) = [(1-\alpha)(\mathbb{T}), \alpha(C)]$,
  - $\tau(H(t)) = [(1-\alpha)(\mathbb{T}), \alpha(C)]$, если $\tau_e(H(t-1)) = \mathbb{T}, t \geqslant 1$,
  - $\tau(H(t)) = [(1-\alpha)(C), \alpha(\mathbb{T})]$, если $\tau_e(H(t-1)) = C, t \geqslant 1$;
- Хитрый:
  - $\tau(\varnothing) = [(1-\alpha)(\mathbb{T}), \alpha(C)]$,
  - $\tau(H(1)) = [(1-\alpha)(C), \alpha(\mathbb{T})]$,
  - $\tau(H(2)) = [(1-\alpha)(\mathbb{T}), \alpha(C)]$,
  - Для $3 \leqslant t < T$:
    - $\tau(H(t)) = [(1-\alpha)(C), \alpha(\mathbb{T})]$, если $\tau_e(H(t)) = \mathbb{T}$ или $\tau_e(H(t)) = C$, $t \in \{0, 1, 2\}$,
    - Иначе ведет себя как имитатор,
  - $\tau(H(\mathbb{T})) = [(1-\alpha)(C), \alpha(\mathbb{T})]$;
- Злопамятный:
  - $\tau(\varnothing) = [(1-\alpha)(\mathbb{T}), \alpha(C)]$,
  - $\tau(H(\mathbb{T})) = [(1-\alpha)(\mathbb{T}), \alpha(C)]$, если $\tau_e(H(t-1)) = \tau(H(t-1)) = \mathbb{T}$, $t \geqslant 1$
  - $\tau(H(\mathbb{T})) = [(1-\alpha)(C), \alpha(\mathbb{T})]$, иначе.

§2. Реализация и проведение эксперимента

Для следующего шага исследования предлагается провести множество игр «дилемма заключенного». Данная игра может быть представлена в виде следующей матрицы:

$$\begin{array}{c} \\ \mathbb{T} \\ C \end{array} \begin{array}{c} \begin{array}{cc} \mathbb{T} & C \end{array} \\ \begin{pmatrix} (2,2) & (-1,3) \\ (3,-1) & (0,0) \end{pmatrix} \end{array},$$

где «$\mathbb{T}$» - вариант довериться, а «C» - обмануть.

Суть проводимого эксперимента заключается в следующем. Двум испытуемым предлагается очно сыграть в игру «дилемма заключенного», но с небольшим изменением. Если игрок хочет совершить одно из двух доступных действий (довериться или обмануть), то с вероятностью α его выбор меняется на противоположный. Параметр α - есть вероятность ошибки. Интерпретация данного параметра следующая. В социальной жизни между живыми людьми возможно недопонимание и неверная интерпретация поступков. Параметр α как раз и характеризует это недопонимание. Его значение можно установить, например, 0.1.

Сама игра реализована с помощью языка Python в среде PyCharm. Будет проведено 2 тура. В каждом туре разыгрывается 6 раундов (то есть всего каждый игрок сыграет 12 раундов). Каждому игроку должны будут выдаваться контроллеры с двумя кнопками: довериться или обмануть. Результаты игр (счет и ходы игроков) записываются в таблицу Excel.

Рис. 1: Примерная схема проведения эксперимента

Ниже перечислены необходимые условия для проведения эксперимента:

- Игроки должны находится рядом друг с другом (то есть необходимо, чтобы игрок знал своего оппонента). Если игроки знают друг друга, то это может повлиять на поведение того или иного игрока;
- Запрещено всякое взаимодействие с оппонентом (жесты; сигналы; физические, зрительные и вербальные контакты);
- Запрещено эмоционально реагировать на любые ситуации во время проведения эксперимента;
- Игроку разрешено только смотреть в монитор и совершать выбор одной из двух кнопок;
- В конце каждого раунда на экран выводится совершенное действие каждого игрока и выигрыш за этот раунд;
- В конце каждого тура игрокам показывается их счет, а так же максимально и минимально возможное количество очков, которое можно было получить в этом туре;
- Всё выше перечисленное контролируется организатором, который присутствует в одной комнате с испытуемыми.

Перед началом проведения эксперимента было проведено несколько пробных игр, в ходе которых был выявлен ряд проблем, что в дальнейшем было исправлено:

1. Изначально планировалось проводить 3 тура по 8 раундов. Но игроки после второго тура жаловались, что игра идет достаточно долго. И в третьем туре они особо не задумывались, какое действие в игре выбрать. В связи с этим было принято решение о сокращении длительности игры. После нескольких попыток было установлено, что оптимально проводить 2 тура по 6 раундов в каждом.
2. Когда срабатывала ошибка, то есть с вероятностью α ответ менялся на противоположный, игроки иногда бурно реагировали, что ставило

под вопрос чистоту эксперимента. Поэтому игроки должны были не только молчать, но и не подавать никаким другим способом сигнал оппоненту о той или иной ситуации в игре.

3. По общим жалобам и проблемам во время проведения пробного эксперимента был сделан более читаемый интерфейс, который включал в себя более интуитивный вывод на экран возможных ходов для игроков, выигрыши после каждого раунда и суммарные выигрыши после каждого тура.

Но, в связи со сложившейся ситуацией в мире из-за коронавирусной инфекции, провести игру очно оказалось невозможно по разным причинам: многие из студентов уехали домой, доступ в общежития частично ограничен и введен режим самоизоляции. Поэтому в данной ситуации было принято решение перенести эксперимент в онлайн-формат. Реализовать игры решено было с помощью бота на платформе «Telegram».

Чтобы начать работу по созданию бота, необходимо написать другому боту, который именуется «BotFather», в мессенджере «Telegram» [15-18]. На данном этапе мы даем имя новому боту и получаем так называемый токен - уникальный ключ для доступа к созданному боту:

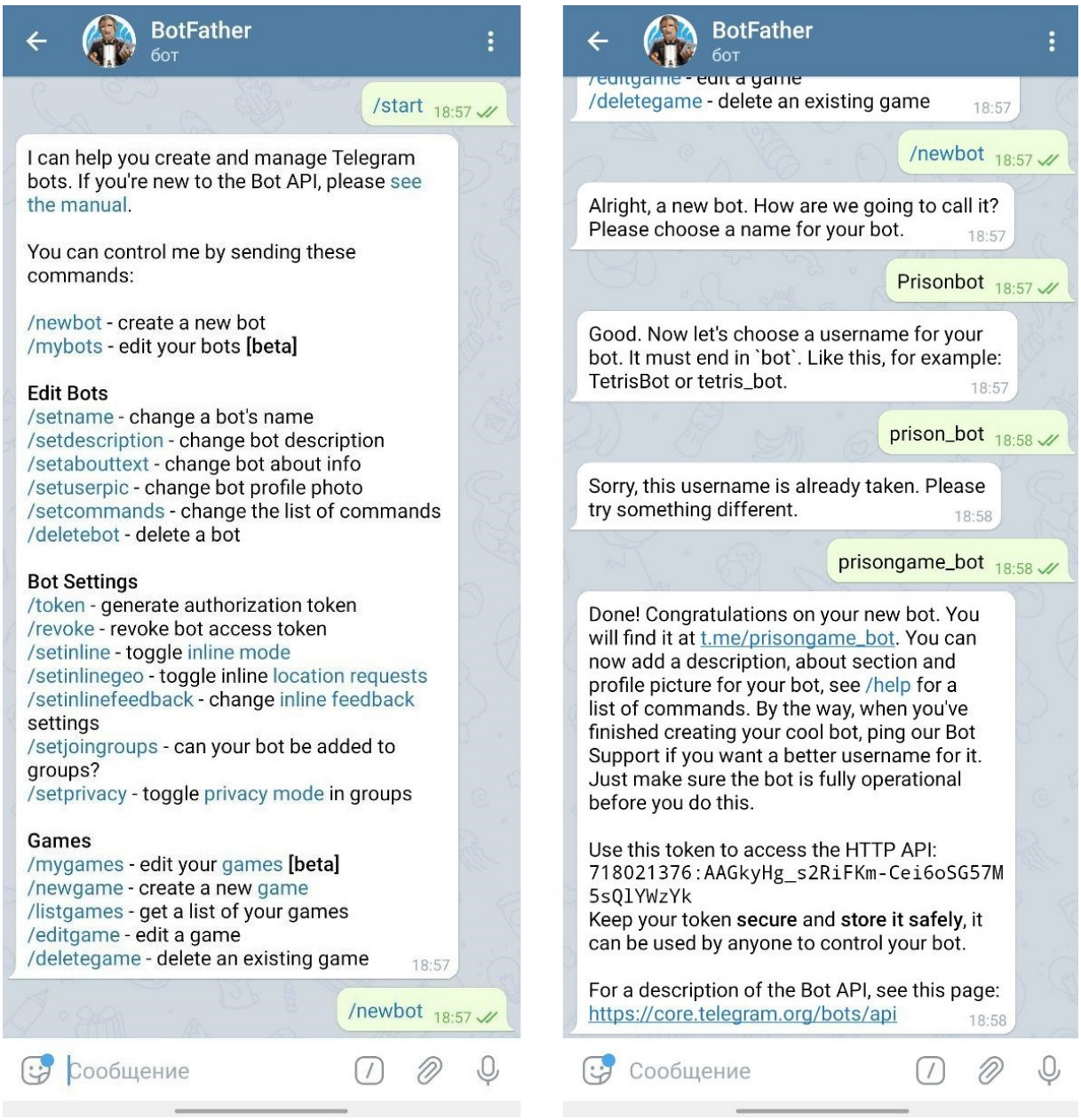

Рис. 2: Создание бота

Как можно видеть, бот имеет имя «prisongame_bot», а его уникальный ключ: «718021376:AAGkyHg_s2RiFKm-Cei6oSG57M5sQlYWzYk». После этого необходимо настроить бота. Редактируется бот в среде PyCharm на языке Python.

Принцип работы бота следующий. Потенциальным игрокам присылается ссылка на бота. При переходе по ссылке открывается следующее диалоговое окно:

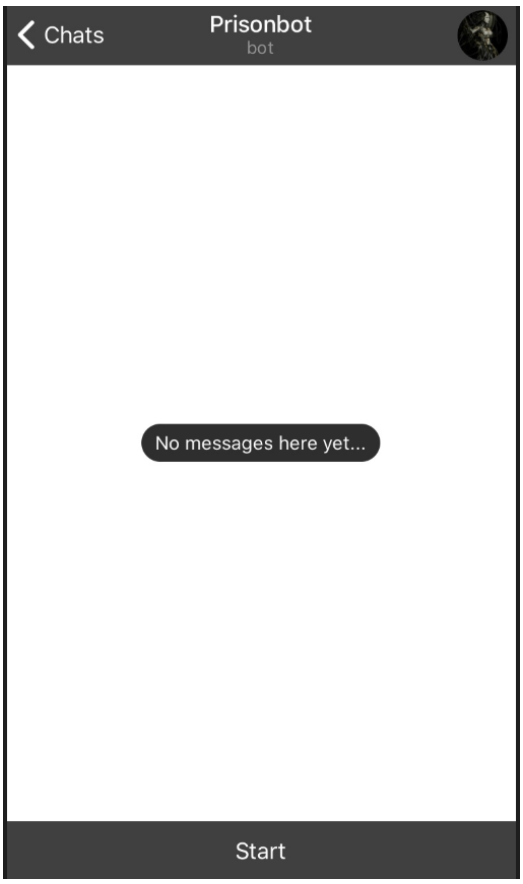

Рис. 3: Запуск бота

Чтобы бот начал работать, необходимо запустить написанный код в PyCharm. Запуск кода происходит на ноутбуке, и бот будет функционировать до тех пор, пока код запущен (то есть в данном случае ноутбук играет роль сервера). Чтобы бот стабильно функционировал, необходимо воспользоваться VPN-сервисом, так как на территории Российской Федерации возможности мессенджера «Telegram» ограничены. Было решено воспользоваться сервисом Windscribe-VPN ввиду того, что данный сервис является бесплатным на первые 2 гигабайта интернет-трафика, чего вполне достаточно для проведения игр [21].

Подход с использованием бота отвечает всем требованиям проведения эксперимента, которые были сформулированы ранее, кроме первого пункта: «Игроки должны находится рядом друг с другом». Данный пункт необходим, чтобы игроки понимали, против кого они играют (так как в дальнейшем будет проверена зависимость ходов от того, как близко игроки знакомы друг с другом). Чтобы удовлетворить этому требованию, боту была добавлена функция «Заполнить анкету». Благодаря этому игроки будут понимать, против кого они играют. Заполнение анкеты предлагается игроку сразу

после того, как он нажмет кнопку старт:

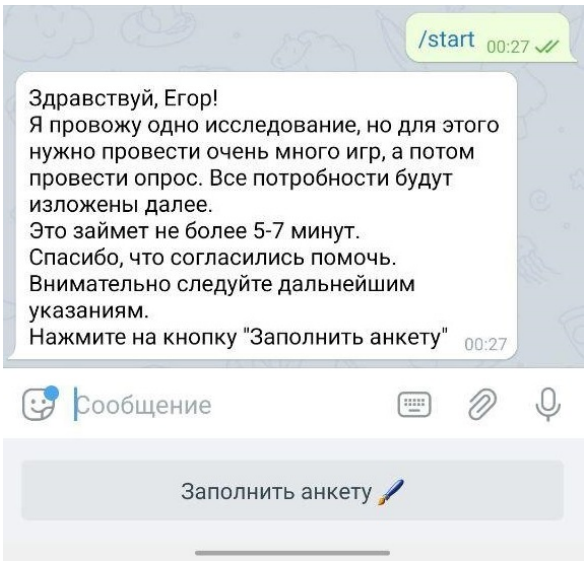

Рис. 4: Начало работы с ботом

После заполнения анкеты, где игроки вводят свое реальное имя, им предлагается ознакомиться с правилами игры. Текст с описанием игры приведен ниже:

- *"Ты будешь играть в игру со случайным человеком. Тебе будет предложено сыграть 2 тура по 6 раундов (т.е. всего 12 раундов). В каждом раунде у тебя (как и у твоего противника) всего две опции: довериться или обмануть. Если вы оба доверяетесь, то получаете выигрыш +2. Если вы оба обманываете, то вы ничего не получаете (т.е. +0). Но если кто-то из вас решил обмануть, а другой довериться, то обманувший получает +3, а доверчивый теряет 1 балл (т.е. -1). В конце каждого тура (т.е. после 6-ти раундов) ты увидишь свой счет и счет противника за этот тур. Твоя цель: набрать как можно больше баллов, но больше, чем противник. И последнее. В игру встроена вероятность ошибки 0.1. Это означает, что с вероятностью 0.1 твой выбор (как и выбор соперника) поменяется на противоположный. Поэтому если тебя вдруг обманут, подумай, может это не обман, а ошибка. Если все понятно, то нажми кнопку 'Игра'".*

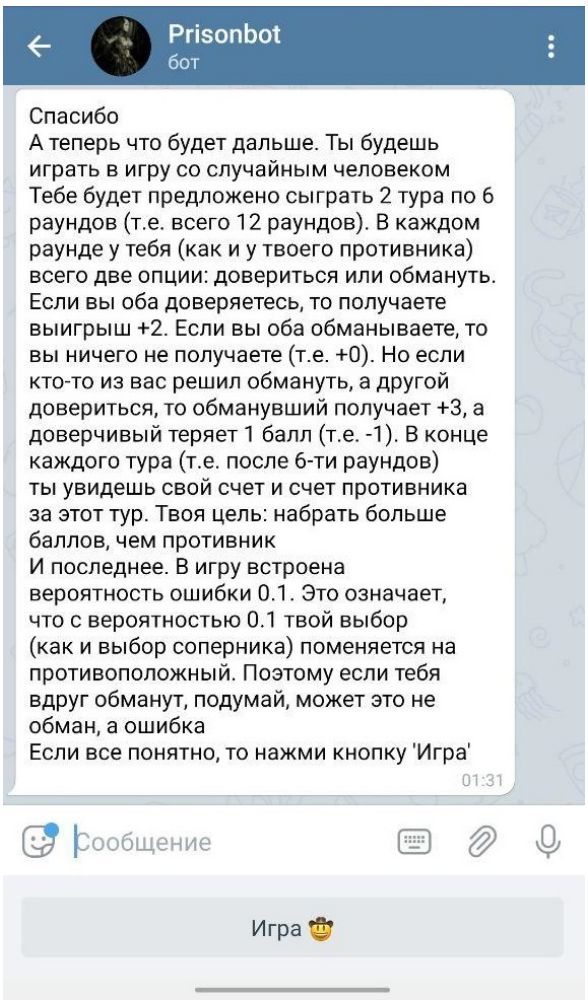

Рис. 5: Описание игры

Механизм проведения ботом игры между двумя игроками состоит в следующем:

- Каждый игрок, запустив бота, начал с ним чат. У каждого чата есть свой индивидуальный номер, так называемый id_chat;
- После заполнения анкеты игроком бот так же сохраняет id_chat для этого игрока;
- После того, как игрок нажал на кнопку «Игра», происходит один из двух следующий вариантов:
    1. Бот напишет игроку немного подождать своего соперника. Это означает, что на данный момент нет свободных противников и игрок пока остается без пары;

2. Бот сразу образует пару игроков из двух участников. Первым участником является тот, кто только что нажал кнопку «Игра». Вторым участником будет игрок, кто ожидал своего соперника из первого пункта. Игроки буду связаны друг с другом посредством id_chat;

- Имена игроков, записи всех ходов и результаты игр будут храниться в онлайн базе данных «MongoDB»;

- Сыграть в эту игру возможно только один раз. Если участники эксперимента попробуют написать боту «Игра», то бот не позволит им сыграть еще раз и напишет: «Извините, но вы уже играли».

Когда бот образовал пару игроков, то каждому из них присылается в чат имя противника, и игра начинается. Интерфейс интуитивно понятен - внизу чата всегда присутствуют две кнопки, которые соответствуют выбору «довериться» или «обмануть». Ниже представлен пример того, как проходит игра:

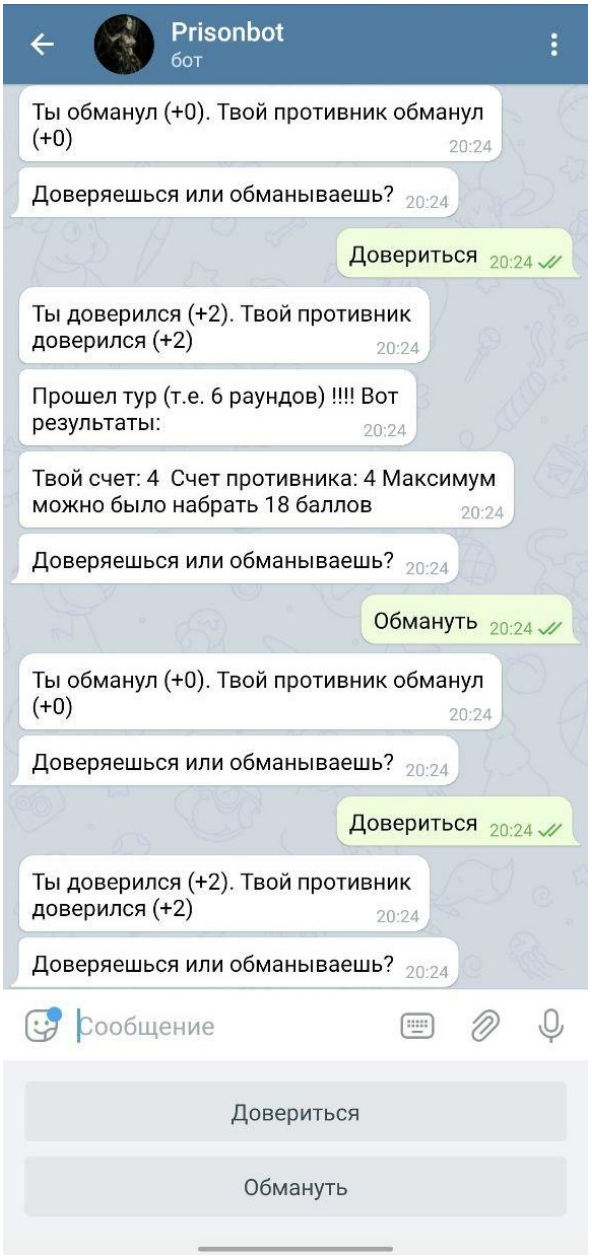

Рис. 6: Пример игры

В играх приняло участие более 130 студентов факультета прикладной математики - процессов управления (это более 65 игр, так как в игре принимают участие 2 игрока). К сожалению, некоторые результаты невозможно было интерпретировать, в связи с чем количество игр, которое поддавалось анализу, сократилось до 51.

§3. Проверка гипотез и сопоставление стратегий

После проведения эксперимента испытуемым предлагалось пройти анкетирование, которое включало в себя следующие пункты:

- *Пол;*
- *Возраст;*
- *Религия;*
- *Показатель измерения черт личности (интроверт, экстраверт или амбиверт);*
- *Знаком ли испытуемый со своим оппонентом;*
- *Сколько раз в месяц испытуемый собирался с друзьями;*
- *Наличие братьев/сестер;*
- *Наличие домашнего животного;*
- *Живет ли испытуемый один;*
- *Численность населения родного города;*
- *Любимый жанр кино;*
- *Доволен ли испытуемый текущей политической ситуацией в стране;*
- *Степень внимательности при прочтении договоров;*
- *Отдает ли испытуемый деньги в долг;*
- *Оправдывает ли испытуемый ложь во благо.*

Вот некоторые данные, собранные из анкет:

- *В эксперименте приняло участие 59 женщин и 43 мужчины;*
- *Средний возраст испытуемых составил 20 лет;*
- *Религиозными считают себя 27 человек, атестами - 31 человек, агностиками - 44 человека;*
- *24 человека считают себя экстравертами, 19 - интровертами, 59 - амбивертами;*
- *Большинство испытуемых (64 человека) не знало своего оппонента.*
- *32 человека не имеют братьев/сестер;*
- *Домашнее животное есть у 45 испытуемых;*
- *Только 29 человек проживали одни на момент эксперимента. У остальных испытуемых имелся сожитель.*

Так же в анкете обязательным пунктом был вопрос: "Опиши впечатление от этой небольшой игры, как ты старался(-лась) играть и прочее? Пиши все, что считаешь нужным". Благодаря этому вопросу, а так же записанным ходам во время проведения игры можно было установить тип поведения игрока.

На первом этапе данной работы предполагается проверить зависимость первого хода от факторов, представленных в анкете. Это имеет смысл, так как до первого хода игрока у игры не было истории, и этот первый ход может зависеть от индивидуальных особенностей игрока. Для этой цели было решено построить таблицы сопряженности между факторами, представленными в анкете, и решениями «Довериться» и «Обмануть». После этого необходимо воспользоваться критерием Хи-квадрат, чтобы определить наличие связи [10,11]. Было проверено 17 гипотез.

Зависимость «Решение - Пол». Проверим гипотезу зависимости первого хода от пола игрока. Гипотезу, как говорилось ранее, будем проверять с помощью критерия Хи-квадрат. Нулевая гипотеза: H_0 - признаки

являются независимыми. В противовес к нулевой гипотезе сформулируем альтернативную: H_1 - имеется статистически значимая связь между факторами. Гипотезы будем проверять на уровне значимость $\alpha = 0.05$. Составим таблицу сопряженности по результатам проведения игр:

Решение \ Пол	Муж	Жен	Всего
Довериться	29	42	71
Обмануть	14	17	31
Всего	43	59	102

Таблица 1: Таблица сопряженности "Решение - Пол"

Посчитав критерий Хи-квадрат для данной таблицы, получаем следующее: $\chi^2 = 0.0354$, $p-value = 0.8508$. Для данной таблицы $p-value > 0.05$, поэтому нет оснований отвергать нулевую гипотезу H_0. Следовательно, статистически значимой связи между решением и полом нет.

Проверим так же гипотезы о наличии связи между решением и полом противника. Для женщин будет построена следующая таблица:

Решение \ Пол оппонента	Жен	Муж	Всего
Довериться	24	18	42
Обмануть	6	11	17
Всего	30	19	59

Таблица 2: Таблица сопряженности "Решение - Пол оппонента"(Жен.)

Для мужчин имеем следующую таблицу:

Решение \ Пол оппонента	Муж	Жен	Всего
Довериться	8	21	29
Обмануть	6	8	14
Всего	14	29	43

Таблица 3: Таблица сопряженности "Решение - Пол оппонента"(Муж.)

Для первой таблицы имеем: $\chi^2 = 1.5199$, $p - value = 0.2176$. Для второй: $\chi^2 = 0.4279$, $p - value = 0.51$. В обоих случаях мы принимаем нулевую гипотезу H_0 об отсутствии статистически значимой связи между факторами.

Зависимость «Решение - Наличие домашнего животного». Построим таблицу сопряженности по данном факторам:

Решение \ Дом. животное	Есть	Нет	Всего
Довериться	31	40	71
Обмануть	14	17	31
Всего	45	57	102

Таблица 4: Таблица сопряженности "Решение - Дом. животное"

Для данной таблицы имеем: $\chi^2 = 2.0544 * 10^{-30}$, $p - value = 1$. По критерию Хи-квадрат мы делаем вывод о независимости факторов с большим значение вероятности p-value.

Зависимость «Решение - Черта личности». Построим таблицу сопряженности, благодаря которой проверим зависимость между первым

ходом и тем, является ли человек экстравертом, интровертом или амбивертом:

Черта Личности / Решение	Амбиверт	Интроверт	Экстраверт	Всего
Довериться	41	14	16	71
Обмануть	18	5	8	31
Всего	59	19	24	102

Таблица 5: Таблица сопряженности "Решение - Черта личности"

Для данной таблицы имеем: $\chi^2 = 0.2476$, $p-value = 0.8835$. По критерию Хи-квадрат мы делаем вывод об отсутствии статистической значимости между данными факторами.

Зависимость «Решение - Наличие братьев или сестер». Данный пункт при прохождении анкеты игроками предлагал выбрать так же и уровень взаимоотношений с братьями/сестрами. Были доступны следующие варианты ответов: «хорошие отношения», «нейтральные отношения» или «мы никогда не ладили». Из 102 игроков только один выбрал последний вариант, поэтому данного игрока мы не будем вносить в таблицу. Составим таблицу по имеющимся данным:

Братья/сестры / Решение	Нет	Да, хорошие отношения	Да, нейтральные отношения	Всего
Довериться	20	36	14	70
Обмануть	12	12	7	31
Всего	32	48	21	101

Таблица 6: Таблица сопряженности "Решение - Братья/сестры"

Для данной таблицы имеем: $\chi^2 = 1.4972$, $p - value = 0.473$. По критерию Хи-квадрат мы делаем вывод об отсутствии статистической значимости между данными факторами. Так же имеет смысл проверить зависимость решения от наличия братьев/сестер, не учитывая взаимоотношения:

Решение \ Братья/сестры	Есть	Нет	Всего
Довериться	51	20	71
Обмануть	19	12	31
Всего	70	32	102

Таблица 7: Таблица сопряженности "Решение - Братья/сестры"

Для данной таблицы имеем: $\chi^2 = 0.6778$, $p - value = 0.4104$. Даже если убрать взаимоотношения, что мы так же не получаем зависимости между факторами.

Зависимость «Решение - Внимательное чтение документов». Данный пункт при прохождении анкеты игроками предлагал выбрать, как они подписывают документы. Было предложено три варианта: «Внимательно прочитать все пункты», «Прочитать только основное» или «Подписать сразу». Получаем следующую таблицу:

Решение \ Документ	Читать все	Читать главное	Не читать	Всего
Довериться	44	3	24	71
Обмануть	15	3	13	31
Всего	59	6	37	102

Таблица 8: Таблица сопряженности "Решение - Документ"

Для данной таблицы имеем: $\chi^2 = 2.1723$, $p - value = 0.3375$. Статистически значимая связь отсутствует.

Зависимость «Решение - Религия». В данном пункте анкеты у игроков спрашивали, являются ли они верующими. Можно было выбрать один из вариантов ответа: «Да», «Нет, я атеист» и «Нет, я агностик» или написать свой вариант. Просмотрев полученные ответы, можно составить следующую таблицу сопряженности:

Религия / Решение	Верующий	Агностик	Атеист	Всего
Довериться	23	34	14	71
Обмануть	4	10	17	31
Всего	22	46	34	102

Таблица 9: Таблица сопряженности "Решение - Религия"

Получаем следующие значения для данной таблицы: $\chi^2 = 13.076$, $p - value = 0.0014$. Мы получили значение $p - value < 0.05$, поэтому мы отклоняем нулевую гипотезу об отсутствии связи и принимаем альтернативную, что связь между факторами имеется. В самом деле, если посмотреть на значения в этой таблице, то можно увидеть, что верующие и агностики на первом ходу склонны доверять, когда как для атеиста нет четкого преобладания одного решения над другим.

Зависимость «Решение - Степень знакомства». В данном пункте анкеты у игроков спрашивали, как хорошо они знают своего противника (оценка от 1 до 5). Таблица сопряженности выглядит следующим образом:

Степень знакомства / Решение	1	2	3	4	5	Всего
Довериться	45	6	9	2	9	71
Обмануть	19	5	3	3	1	31
Всего	64	11	12	5	10	102

Таблица 10: Таблица сопряженности "Решение - Степень знакомства"

По данной таблице мы не можем ничего сказать, так как в некоторых ячейках таблицы присутствуют слишком маленькие значения. Поэтому проверка гипотезы на проверку связи между факторами по данной таблице лишена смысла.

Зависимость «Решение - Дать в долг». У игроков спрашивали, дают ли они друзьям деньги в долг. Предлагались следующие варианты ответов: «Даю спокойно», «Даю, но осторожно (сохраняю переписку или записи звонков для страховки)» и «Никому не даю в долг».

Деньги в долг / Решение	Даю спокойно	Даю с осторожностью	Не даю	Всего
Довериться	12	55	4	71
Обмануть	9	18	4	31
Всего	21	73	8	102

Таблица 11: Таблица сопряженности "Решение - Деньги в долг"

Для данной таблицы имеем: $\chi^2 = 4.131$, $p - value = 0.1268$. Значение $p - value$ недостаточно низкое, чтобы отклонить нулевую гипотезу, поэтому мы заключаем об отсутствии зависимости.

Зависимость «Решение - Ложь во благо». В данном пункте анкеты у игроков спрашивали, оправдывают ли они ложь во благо, где 1 - «Однозначно нет», а 5 - «Однозначно да». Построим таблицу сопряженности.

Решение \ Ложь во благо	1	2	3	4	5	Всего
Довериться	3	9	35	20	4	71
Обмануть	2	4	13	9	3	31
Всего	5	13	48	29	7	102

Таблица 12: Таблица сопряженности "Решение - Ложь во благо"

Для данной таблицы имеем: $\chi^2 = 0.9872$, $p-value = 0.9117$. Статистически значимая связь отсутствует. Проверять значимость связи с помощью коэффициента ранговой корреляции не имеет смысла, так как порядок рангов у табличных значений будет совпадать.

Зависимость «Решение - Политическая ситуация». В данном пункте анкеты у игроков спрашивали, довольны ли они политической ситуацией в стране, где 1 - «Крайне недоволен», а 5 - «Полностью доволен». Построим таблицу сопряженности.

Решение \ Полит. ситуация	1	2	3	4	5	Всего
Довериться	15	35	15	3	3	71
Обмануть	12	17	2	0	0	31
Всего	27	52	17	3	3	102

Таблица 13: Таблица сопряженности "Решение - Полит. ситуация"

Для такой таблицы мы не можем посчитать значение Хи-квадрат, так как у нас если маленькие и нулевые значения. Предлагается объединить колонки 1-2 и колонки 3-5. Первая колонка отражает недовольных игроков, вторая - нейтральных и положительно относящихся к власти. Получаем следующую таблицу:

Полит. ситуация / Решение	1-2	3-5	Всего
Довериться	50	21	29
Обмануть	29	2	14
Всего	79	23	43

Таблица 14: Новая таблица сопряженности "Решение - Полит. ситуация"

Для данной таблицы применим точный критерий Фишера. Получаем значение $p - value = 0.01$, что говорит нам о наличии связи. В самом деле, судя по таблице, если человек доволен политической ситуацией в стране, то он не склонен к обману.

Зависимость «Решение - Население родного города». В последнем случае мы проверим зависимость решения от населения родного города игроков. В анкете предлагалось ответить на вопрос «Размер вашего родного города» выбрав один из четырех вариантов: «Малый город (до 20 тыс.)», «Средний город (20 - 100 тыс.)» «Крупный город (100-500 тыс.)» и «Крупнейший город (от 500 тыс.)». Получаем следующую таблицу:

Решение \ Население	до 20 тыс	20-100 тыс	100-500 тыс	от 500 тыс	Всего
Довериться	30	28	9	4	71
Обмануть	11	6	8	6	31
Всего	41	34	17	10	102

Таблица 15: Таблица сопряженности "Решение - Население"

Получаем следующие значения для данной таблицы: $\chi^2 = 9.2326$, $p - value = 0.02635$. Мы получили значение $p - value < 0.05$, что говорит о наличии статистически значимой связи. В данном случае, чтобы лучше увидеть закономерность, предлагается объединить колонки 1 и 2, а так же 3 и 4. Получаем следующую таблицу:

Решение \ Население	от 100 тыс	до 100 тыс	Всего
Довериться	58	13	71
Обмануть	17	14	31
Всего	75	27	102

Таблица 16: Новая таблица сопряженности "Решение - Население"

В данном случае получаем следующие значения: : $\chi^2 = 6.67$, $p - value = 0.01$. Из таблицы можно заключить, что люди, приехавшие из крупных городов склонны к доверию на первом ходу, когда как для игроков из маленьких городов нет однозначного хода.

В итоге только 3 теста наличие связи между фактором и ответом на первом ходу. Факторы, которые влияют на первый ход - это «Религия», «Отношение к политической ситуации в стране» и «Население родного города».

§4. Определение стратегий игроков

Чтобы проверить зависимость стратегий от факторов, необходимо сначала определить, какими стратегиями пользовались игроки. Для этого выпишем пары игроков из каждой игры и проанализируем из ходы:

13	[2,1,-2,2,1,1,-1,-1,2,1,1,2]	[2,1,-2,2,1,1]	[-1,-1,2,1,1,2]	+	Балансировщик
14	[1,1,1,1,-1,1,1,1,1,1,1,2]	[1,1,1,1,-1,1]	[1,1,1,1,1,-1]	+	Доверие
15	[1,1,1,1,1,1,1,1,1,2,1,1]	[1,1,1,1,1,1]	[1,1,1,-1,1,1]	+	Доверие
16	[1,1,2,1,2,2,1,1,2,2,2,2]	[1,1,2,1,2,2]	[1,1,2,2,2,2]	+	Балансировщик
17	[1,1,1,1,1,1,1,1,2,1,1,1]	[1,1,1,1,1,1]	[1,1,2,1,1,1]	+	Око за око
18	[1,1,1,1,1,1,1,-1,1,1,1,-1]	[1,1,1,1,1,1]	[1,-1,1,1,1,-1]	+	Доверие
19	[1,1,2,2,1,2,2,2,2,2,2,2]	[1,1,2,2,1,2]	[2,2,2,2,2,2]	+	Балансировщик
20	[2,2,2,2,2,2,2,-2,2,-2,2,2]	[2,2,2,2,2,2]	[2,-2,2,-2,2,2]	+	Обман
21	[1,1,1,1,1,1,-2,2,2,1,1,1]	[1,1,1,1,1,1]	[-2,2,2,1,1,1]	+	Доверие(?)
22	[1,1,1,1,1,1,1,1,1,1,1,1]	[1,1,1,1,1,1]	[1,1,1,1,1,1]	+	Доверие
23	[1,2,2,1,2,2,1,1,2,2,1,1]	[1,2,2,1,2,2]	[1,1,2,2,-2,1]	+	Балансировщик
24	[1,1,1,2,2,2,-2,2,2,2,1,2]	[1,1,-2,2,2,2]	[-2,2,2,2,-2,2]	+	Злопамятный(2)

Рис. 7: Пример распечатки ходов

Всего приняло участие в эксперименте 102 человека. В каждой игре было 2 тура, в которых игроки могли придерживаться одной и той же стратегии, а могли использовать разные. Всего получилось 204 стратегии (Количество игроков * количество туров). Помимо стратегий, описанных в параграфе 1, были выявлены новые стратегии:

- Балансировщик: если оппонент доверяется, то игрок его обманывает, но с некоторым шансом иногда доверяет противнику, чтобы не потерять его расположение. Если же противник обманывает игрока, то он иногда доверяется, чтобы противник тоже начал доверять. Такая стратегия не зависит от ходов оппонента. Ее можно задать одним параметром : $p \in [0, 1]$ - доля доверия. Если $p = 0.5$ - то это стратегия случайного;

- Попытка договориться: после взаимных обманов игрок доверяется в надежде на то, что его оппонент тоже решит довериться. Задается параметр $q \in [0, 1]$ - вероятность довериться и пойти на сговор с противником;

- Неожиданный обман: после того, как игроки несколько раз взаимно доверялись друг другу, данное поведение используется под конец игры, чтобы выровнять счет или выйти вперед по очкам. Является вариацией стратегии полного доверия.

Все эти стратегии имеют место, так как либо в анкете (где игроки могли рассказать, почему они играли именно так), либо в личном разговоре игроки описывали свое поведение в играх, которое совпадало либо с уже известными стратегиями, либо с новыми, которые описаны выше:

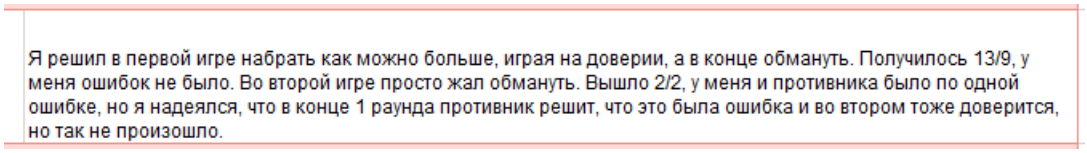
Я решил в первой игре набрать как можно больше, играя на доверии, а в конце обмануть. Получилось 13/9, у меня ошибок не было. Во второй игре просто жал обмануть. Вышло 2/2, у меня и противника было по одной ошибке, но я надеялся, что в конце 1 раунда противник решит, что это была ошибка и во втором тоже доверится, но так не произошло.

Рис. 8: Пример комментария к игре

В результате анализа ходов и комментариев игроков получаем следующее распределение стратегий в первой и второй играх (в ячейках таблицы записано то количество тех или иных стратегий, которые были применины в соответствующих играх):

Номер игры / Стратегия	№1	№2	Всего
Имитатор	13	15	28
Доверчивый	15	9	24
Обманщик	10	21	31
Хитрый	3	3	6
Злопамятный	9	9	18
Обманщик + попытка договориться	7	10	17
Доверчивый + неожиданный обман	7	7	14
Балансировщик	36	25	61
Случайный	2	3	5

Таблица 17: Стратегии в первой и во второй играх

Формализуем новые модели поведения, где $p \in [0, 1]$ - доля доверия, а $q \in [0, 1]$ - вероятность довериться и пойти на сговор с противником:

- Балансировщик: $\tau(\cdot) = [((1 - \alpha) \cdot p + \alpha \cdot (1 - p))(\mathbb{T}), (\alpha \cdot p + (1 - \alpha) \cdot (1 - p))(C)]$;
- Неожиданный обман под конец игры:

- $\tau(H(t)) = [(1-\alpha)(\mathbb{T}), \alpha(C)], t < T$;
- $\tau(H(T)) = [(1-\alpha)(C), \alpha(\mathbb{T})]$;

- Неожиданный обман с постепенным увеличением вероятности обмануть:
  - $\tau(\varnothing) = [(1-\alpha)(\mathbb{T}), \alpha(C)]$,
  - $\tau(H(t)) = [((1-\alpha)\cdot\frac{T-t}{T} + \alpha\cdot\frac{t}{T})(\mathbb{T}), (\alpha\cdot\frac{T-t}{T} + (1-\alpha)\cdot\frac{t}{T})(C)], t < T$;

- Обман с попыткой договориться:
  - $\tau(\varnothing) = [(1-\alpha)(C), \alpha(\mathbb{T})]$,
  - $\tau(H(1)) = [((1-\alpha)\cdot(1-q) + \alpha\cdot q)(C), (\alpha\cdot(1-q) + q\cdot(1-\alpha))(\mathbb{T})]$,
  - Для $2 \leqslant t \leqslant T$:
    - $\tau(H(t)) = [(1-\alpha)(\mathbb{T}), \alpha(C)]$, если $\tau(H(t-1)) = \mathbb{T}$ и $\tau(H(t-2)) = C$ или $\tau(H(t-1)) = \mathbb{T}$, $\tau(H(t-2)) = \mathbb{T}$ и $\tau_e(H(t-1)) = \mathbb{T}$;
    - Иначе: $\tau(H(t)) = [((1-\alpha)\cdot(1-q) + \alpha\cdot q)(C), (\alpha\cdot(1-q) + q\cdot(1-\alpha))(\mathbb{T})]$.

Стратегию с неожиданным обманом было принято разделить на две: когда обман происходит непосредственно на последнем ходу, и когда вероятность обмануть возрастает к концу игры (по сути игрок из доверчевого превращается в обманщика).

Обман с попыткой обмануть можно описать следующим образом. Сначала игрок ведет себя как обманщик. Но потом он может довериться своему оппоненту, тем самым приглашая его к сотрудничеству. Если игрок на предыдущем ходу доверился, то он смотрит на свой ход, который случился двумя ходами ранее. Если два хода назад он обманул, то ход назад он только предложил пойти на доверие с оппонентом, поэтому в таком случае игрок должен довериться. Если же два хода назад игрок доверялся, то мы смотрим на реакцию противника: если противник доверился ход назад, ты игрок тоже доверяется, иначе игрок возвращается к своей обычной стратегии.

По результатам определения стратегий у игроков можно сказать, что самой популярной моделью поведения является «Балансировщик». Действительно, анализируя анкетные данные можно увидеть комментарии следующего типа:

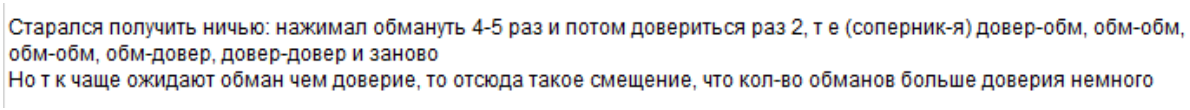
Старался получить ничью: нажимал обмануть 4-5 раз и потом довериться раз 2, т е (соперник-я) довер-обм, обм-обм, обм-обм, обм-довер, довер-довер и заново
Но т к чаще ожидают обман чем доверие, то отсюда такое смещение, что кол-во обманов больше доверия немного

Рис. 9: Пример комментария к игре

Среди остальных моделей поведения можно заметить низкую популярность поведения «Хитрый» и «Случайный». Действительно, очень мало игроков сказали, что пытались отгадать стратегию противника. Модель поведения «Случайный» является частным случаем модели «Балансировщик» с параметром $p = 0.5$.

§5. Кластеризация стратегий

Для того, чтобы попытаться предсказать поведение того или иного игрока, попробуем воспользоваться классификатором. В качестве факторов будут выступать те же анкетные данные, а кластеризовать мы будем стратегии игроков в первой игре. Для такой задачи можно воспользоваться наивным Байесовским классификатором [7,12]. Байесовский алгоритм — это такой алгоритм классификации, который основан на теореме Байеса с допущением о независимости признаков. Сама теорема Байеса для нашей задачи будет записана следующим образом:

$$P(c|X) = \frac{P(X|c) \cdot P(c)}{P(X)},$$

где

- $X = \{x_1, x_2, \ldots, x_n\}$ — набор n факторов, в качестве которых выступают анкетные данные;
- c — тип поведения, к которому мы будем относить игрока с набором факторов X, $c \in C$;
- $P(c|X)$ — апостериорная вероятность того, что игрок придерживается типа поведения c при данном наборе факторов X;
- $P(c)$ — априорная вероятность того, что игрок придерживается типа поведения c;
- $P(X|c)$ — правдоподобие, т.е. вероятность такого набора факторов X при данном типе поведения c;
- $P(X)$ — априорная вероятность иметь такой набор факторов X.

В нашем случае анкетные данные можно считать независимыми, поэтому можно переписать вероятность $P(c|X)$:

$$P(c|X) = P(c|x_1) \cdot P(c|x_1) \cdot \ldots \cdot P(c|x_n).$$

Знаменатель не будет влиять на итоговый результат. В связи с этим итоговая формула примет вид:

$$P(c|X) = P(c) \cdot \prod_{i=1}^{n} P(c|x_i).$$

Чтобы произвести классификацию объекта, необходимо, имея набор факторов X, высчитать указанные вероятности при всех возможных $c \in C$. Объект будет отнесен к такому классу c^*, для которого вероятность $P(c|X)$ будет наивысшая. Решающее правило имеет вид:

$$c^* = \underset{c \in C}{\operatorname{argmax}} \; P(c) \cdot \prod_{i=1}^{n} P(c|x_i).$$

Построим классификатор по нашим данным и проверим его точность. Наивысшую точность показали те же самые факторы, которые имели влияние на первый ход игрока: «Религия», «Отношение к политической ситуации в стране» и «Население родного города». Точность классификатора, построенного на каждом из этих трех факторов записана в следующей таблице:

Фактор	Точность
Религия	$35,29\%$
Полит. ситуация	$37,25\%$
Население города	$35,29\%$

Таблица 18: Точность классификатора с одним фактором

Точность данных классификаторов очень невысокая. Попробуем увеличить количество факторов в классификаторе. Для этого попробуем различные комбинации тех трех факторов, которые мы рассмотрели ранее. Получим следующие результаты:

Фактор	Точность
Религия и Полит. ситуация	$37,25\%$
Полит. ситуация и Население города	$38,24\%$
Население города и Религия	$33,33\%$
Население города, Полит. ситуация, Религия	$39,22\%$

Таблица 19: Точность классификатора с двумя и тремя факторами

Можем заметить, что, используя все три фактора, классификатор имеет наибольшую точность. К сожалению, точность продолжает оставаться невысокой. Это связанно с тем, что мы имеем всего 102 наблюдения, и построить точный классификатор по такому объему выборки достаточно сложно.

§6. Эволюция стратегий

Попробуем ответить на вопрос, какая модель поведения может считаться лучшей. Для этого в среде PyCharm напишем программу на языке Python, последовательно выполнив следующие этапы:

1. Создадим 8 классов, которые соответствуют рассмотренным ранее моделям поведения (Имитатор, Доверчивый, Обманщик, Хитрый, Злопамятный, Обманщик с попыткой договориться, Доверчивы с неожиданным обманом и Балансировщик);

2. Задаем начальные условия: количество экземпляров каждого класса. Таким образом мы получаем игроков с заданными моделями поведения;

3. Назовем эволюцией следующую последовательность действий:

   (a) Все игроки играют со всеми игроками по несколько раундов (можно, например, взять 6 раундов, как было сделано в эксперименте со студентами). Каждый игрок имеет общую сумму очков, накопленную за эти игры;

   (b) По результатам очков выбираются несколько игроков, которые показали наихудший результат, а так же игрок (или игроки) с наилучшим результатом;

   (c) Игроки с наихудшим результатом меняют свою модель поведения на модель поведения игрока с наилучшим результатом (если игроков с наилучшим результатом несколько, то модель поведения наихудших игроков меняется случайно на одну из моделей поведения наилучших игроков);

   (d) После этого сумма очков сбрасывается и эволюция начинается заново.

4. Проводим несколько этапов эволюции, в результате которой остаются игроки с наилучшими моделями поведения. Назовем такую последовательность эволюций эпохой;

5. Так как каждая модель поведения содержит элемент случайности, то необходимо провести моделирование нескольких эпох. Каждая эпоха включает в себя:

   (a) Создание игроков по начальным условиям;

   (b) Проведение нескольких этапов эволюции;

   (c) Подсчет наилучших стратегий.

6. После моделирования последней эпохи для каждой модели поведения подсчитывается доля эпох, в которой эти модели поведения были признаны лучшими.

Для процесса эволюции возьмем 20 штук каждой модели поведения (то есть всего получится 160 игроков). В каждом этапе эволюции, как уже говорилось ранее, будет проводиться 6 раундов между всеми игроками. Заменять будем модель поведения у четырех наихудших игроков. Всего будет проведено 150 эпох. В конце каждой эпохи мы смотрим на пять игроков с наибольшим счетом - модели их поведения признаются лучшими в данной эпохе. После работы программы получаем следующий результат:

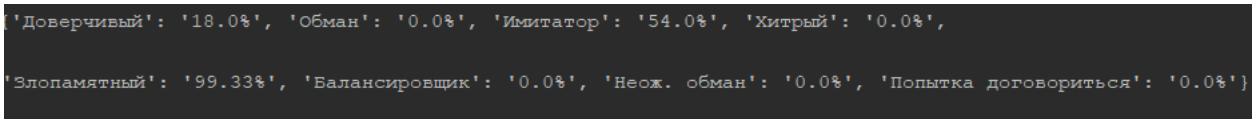

Рис. 10: Пример работы программы

Получается, что почти во всех эпохах стратегия злопамятного игрока оказывалась в пятерке лучших. Так же примерно в половине эпох лучшей признавалась модель имитатора. И в 18% случаях в пятерке лучших оказывалась модель поведения доверчивого игрока. Остальные модели поведения ни разу не вошли в пятерку лучших. Рассмотрим, как проходила эволюция в одну из случайных эпох:

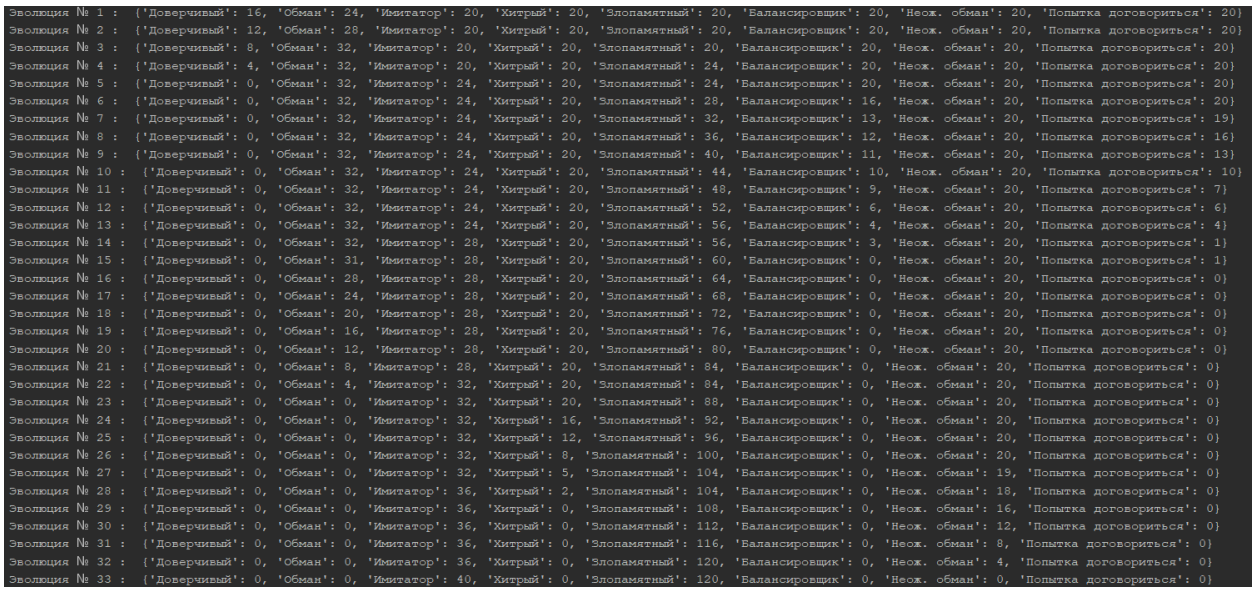

Рис. 11: Пример одной эпохи

На первых этапах эволюции видим, что популяция обманщиков увеличивается за счет доверчивых игроков. Действительно, за счет доверчивых обманщики довольно сильно увеличивают свой игровой счет в каждой эпохе. Но после того, как доверчивых не остается, обманщики не могут набрать достаточное количество очков, чтобы признаться лучшей моделью поведения. Теперь увеличивается число имитаторов и злопамятных. Их стратегии не такие агрессивные, как у обманщиков или балансировщиков, но и не наивные, как у доверчивых. Поэтому число злопамятных и имитаторов увеличивается за счет оставшихся игроков. Так как в игре присутствует ошибка, то злопамятные будут находится в большем выигрыше по очкам, в отличии от имитаторов. Если рассматривать другие эпохи, то картина эволюции будет примерно одинаковой, но иногда популяция доверчивых все-таки доживает до конца эпохи. Если бы в игре отсутствовала ошибка, которая меняет ответ на противоположный, то доверчивые, имитаторы и злопамятные, играя только между собой, всегда бы доверяли друг другу. Но имея в игре ошибку, модель поведения злопамятного обеспечивает наилучшее "выживание" среди агрессивных моделей поведения на первых этапах эволюции.

Данную симуляцию можно проводить с различным числом параметров. Например, изменив количество игр между игроками. Или можно изменять количество тех или других моделей поведения на начальном этапе. Так же можно изменять количество игроков, которые будут изменять свои

модели поведения в конце каждого этапа эволюции. Попробуем уменьшить количество каждых моделей поведения в 2 раза, но увеличим количество игр, которые игроки играют между собой до 12. Так же уменьшим количество игроков, которые будут менять свои стратегии, до двух. Посмотрим на результат работы данной симуляции:

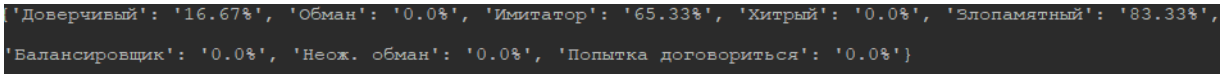

Рис. 12: Пример работы программы

Можем увидеть, что модель поведения злопамятного по-прежнему является наилучшей, несмотря на то, что она попала в список лучших моделей только в 83% случаях. В остальном картина слабо поменялась.

Резюмируя все выше сказанное можно заключить, что модель поведения злопамятного является лучшей с точки зрения получения наибольшего выигрыша, если модели поведения распределены пропорционально.

§7. Поиск причины смены стратегии

В проведенном эксперименте со студентами не все игроки придерживались одной и той же модели поведения в течении двух игр. Кто-то выбирал более агрессивную стратегию в следующей игре. Другие же отдавали предпочтение более мягкой модели поведения. Попробуем проанализировать данный факт.

Рассмотрим модели поведения в каждой игре. Посчитаем, сколько игроков изменяли свои модели поведения, когда играли второй раз. Запишем полученные результаты в следующую таблицу:

Поведение в первой игре	Изменили поведение	Не изменили поведение
Имитатор	9	4
Доверчивый	9	6
Обманщик	0	10
Хитрый	3	0
Злопамятный	6	3
Обманщик + попытка договориться	6	1
Доверчивый + неожиданный обман	5	2
Балансировщик	22	14
Случайный	0	2

Таблица 20: Изменение моделей поведения после первой игры

Исходя из полученных результатов мы можем видеть, что примерно $60-70\%$ игроков каждой модели меняют свое поведение, кроме нескольких случаев. Игроки, которые использовали модель обмана в первой игре решили использовать ее и во второй игре. Про модели поведения «Хитрый» и «Случайны» мы ничего сказать не можем, так как выборка для этих двух моделей очень мала.

Попробуем найти закономерность в изменении модели поведения, рассмотрев каждую игру в отдельности. Построим таблицу, в которой будет показано, сколько игроков меняло или не меняло свою стратегию в зависимости от счета в первой игре:

Счет игрока / Решение	Больше, чем у противника	Меньше, чем у противника	Такой же счет
Смена стратегии	28	18	14
Оставить стратегию	11	21	10

Таблица 21: Таблица сопряженности "Решение-Счет игрока"

Как мы можем заметить, если счет противника меньше или такой же, как у игрока, то решение о смене стратегии происходит случайно. Но если у противника счет больше, то есть большая вероятность, что игрок поменяет свою стратегию. Для данной таблицы имеем: $\chi^2 = 5.2962$, $p-value = 0.0707$. Если мы возьмем уровень значимости $\alpha = 0.05$, то мы принимаем нулевую гипотезу об отсутствии связи между факторами. Но при $\alpha = 0.1$ мы имеем право отклонить данную гипотезу.

Зависимость смены стратегий от факторов в анкете мы исследовать не сможем. Так как мы выявили 9 моделей поведения, то возможных пар моделей после прохождения двух игр каждым игроком может быть $9^2 = 81$. Выборка из 102 игроков очень мала для изучения этого вопроса, так как это более детальный анализ, по сравнению с предыдущим.

Заключение

В данной работе был проведен анализ социального взаимодействия на примере повторяющейся игры «Дилемма заключенного». Было решено проводить данное исследование среди студентов факультета прикладной математики - процессов управления (ПМ-ПУ). На начальном этапе данной работы необходимо было провести как можно больше игр среди студентов. Изначально предполагалось проводить все игры очно в здании факультета, но в связи с распространением коронавирусной инфекции это было невозможно сделать. В связи с этим было принято решение написать бота в мессенджере «Telegram» на языке Python. Дополнительно понадобилось воспользоваться сервисом Windscribe-VPN для обеспечения стабильной работы бота. Так же была необходима база данных, в которой хранились бы все необходимые данные, полученные в ходе игр. Для этой задачи была выбрана база данных «MongoDB».

После того, как бот был написан, необходимо было привлечь как можно больше студентов к участию в данном эксперименте. За сутки в эксперименте приняло участие около 130 студентов факультета прикладной математики - процессов управления. После игры студентам предлагалось пройти анкету, результаты которой будут использованы в анализе поведения каждого игрока. После прохождения анкеты было отобрано 102 игрока и 51 игра, в которой они участвовали. Фильтрация была необходима, так как некоторые студенты несерьезно отнеслись к эксперименту, поэтому данные таких игроков (ходы в игре и информация из анкет) не могли в полной мере поддаваться анализу.

После проведения эксперимента и фильтрации наблюдений было проведено исследование зависимости первого хода от факторов, представленных в анкете. С помощью критерия Хи-квадрат была установлена связь между первым ходом и следующими факторами: «Религия», «Население родного города» и «Отношение к политической ситуации в стране».

В начале данной работы были описаны распространенные модели поведения (имитатор, доверчивый, обманщик, злопамятный, хитрый). Но в ходе анализа ходов игроков и их анкетных данных были выявлены еще 4 модели поведения: балансировщик, обман с попыткой договориться, доверчивый с неожиданным обманом и случайный. С помощью наивного

классификатора Байеса была предпринята попытка кластеризовать модели поведения, используя анкетные данные. Наилучший классификатор имел точность 39.22%. Такой результат нельзя назвать удовлетворительным. Это связано с тем, что мы имеем всего 102 наблюдения, и построить точный классификатор по такому объему выборки достаточно сложно.

Далее была поставлена задача по поиску наилучшей модели поведения. Для ответа на этот вопрос была построена симуляция, в которой игроки с разными моделями поведения играют друг с другом. Затем наихудшие модели поведения заменялись одной из наилучших. Данный процесс повторялся многократно. В результате работы симуляции было установлено, что наилучшей моделью поведения с точки зрения максимизации итогового счета в конце игры является модель поведения злопамятного игрока. Так же можем заметить, что поиск наилучшей модели поведения можно рассматривать при других различных параметрах симуляции (первоначальное число игроков, количество игр, итераций, число игроков, меняющие модель поведения).

В конце работы было проведено исследование на установление причины смены модели поведения игроков. Было установлено, что игроки охотнее меняют свою модель поведения на более агрессивную после игры, в которой они набрали меньше очков, чем противник.

В заключении можно сказать, что полученные новые стратегии поведения игроков можно в дальнейшем исследовать в теории, а именно, изучить, какими свойствами они обладают при многократном и бесконечном повторении дилеммы заключенного.

Литература

1. R. Axelrod. The Evolution of Cooperation. New York: Basic Books, 2006. P. 254.

2. J. Smith. Evolution and the Theory of Games. Cambridge: Cambridge University Press, 1982. P. 234.

3. Л. А. Петросян, Н. А. Зенкевич, Е. В. Шевкопляс. Теория игр: учебник. СПб.: БХВ-Петербург, 2012. 432 с.

4. Michael Maschler, Eilon Solan, Shmuel Zamir. Game Theory. Cambridge: Cambridge University Press, 2013. P. 1003.

5. George J. Mailath, Larry Samuelson. Repeated Games and Reputations: Long-Run Relationships. Oxford: Oxford University Press, 2006. P. 672.

6. С. А. Айвазян, В.М. Бухштабер, И. С Енюков, Л. Д. Мешалкин. Прикладная статистика. Классификация и снижение размерности. М.: Финансы и статистика, 1989. 608 с.

7. R. O. Duda, P.E. Hart, D.G. Stork. Pattern Classification, 2nd edition. Chichester: Wiley, 2000. P. 680.

8. A. R. Webb, K.D. Copsey. Statistical Pattern Recognition, 3rd edition. Chichester: Wiley, 2011. P. 666.

9. Т. Андерсон. Введение в многомерный статистический анализ. М.: Физматгиз, 1963. 500 с.

10. А. Бююль, П. Цеффель. SPSS: Искусство обработки информации, М.: ООО ДиаСофтЮП, 2005, 608 с.

11. А. Наследов. IBM SPSS Statistics 20 и AMOS: профессиональный статистический анализ данных, СПб.: Питер, 2013, 416 с.

12. Хабр. https://habr.com/ru/post/120194/

13. I. Brocas, J. D. Carrillo, J. Tarrasó. How long is a minute? // Games and Economic Behavior. 2018. №111. P. 305-322.

14. Айвазян С.А., Мхитарян В.С. Прикладная статистика и основы эконометрии. М.: ЮНИТИ, 1998. 1022 с.

15. vc.ru. https://vc.ru/selectel/22593-howto-bot-selectel

16. Хабр. https://habr.com/ru/post/262247/

17. Tproger — типичный программист. https://tproger.ru/translations/telegram-bot-create-and-deploy/

18. GitHub. https://mastergroosha.github.io/telegram-tutorial/

19. Яндекс Дзен. https://zen.yandex.ru/id/5d947dd28d5b5f00b14d62d6

20. MongoDB. https://cloud.mongodb.com/

21. Uguide. https://uguide.ru/rejting-luchshie-vpn-servisy

22. Windscribe https://rus.windscribe.com/

23. Google https://docs.google.com/forms/

Приложение

Листинг 1. Игра «Дилемма заключенного», реализованная на языке Python.

```
import random
import getpass
import numpy as np
import pyexcel
import pandas as pd
import xlwt
Player1 = input("Первый игрок:")
Player2 = input("Второй игрок:")

filename = "C:\Exp\\"+Player1+'_и_'+Player2+'.xls'

Pl1='"'+Player1+'"'
Pl2='"'+Player2+'"'

Rounds = 6

TotalPO1 = TotalPO2 = 0
Games = 3

DataPl1 = [[0] * Rounds for i in range(Games)]
DataPl2 = [[0] * Rounds for i in range(Games)]

R = np.zeros(Games)

PO = [2, 3, -1, 0]
mis = 0.1

# эта функция, которая с некоторой вероятностью меняет ответ игрока
def mistake(dis):
    r = random.random()
    if r < mis:
        return(str(int(dis)+2))
    else:
        return(dis)

# основная функция - реализация игры
def payoff(pl1, pl2):
    global PayOff1
    global PayOff2
    global i
    if (pl1 == '1' or pl1 == '4') and (pl2 == '1' or pl2 == '4'):
        print("Игрок", Pl1, "ДОВЕРИЛСЯ (+",PO[0],")
            Игрок", Pl2, "ДОВЕРИЛСЯ (+",PO[0],") ")
        PayOff1+=PO[0]
        PayOff2+=PO[0]

    elif (pl1 == '2' or pl1 == '3') and (pl2 == '2' or pl2 == '3'):
        print("Игрок", Pl1, "ОБМАНУЛ (+",PO[3],")
            Игрок", Pl2, "ОБМАНУЛ (+",PO[3],") ")
        PayOff1+=PO[3]
        PayOff2+=PO[3]
```

```
    elif (pl1 == '1' or pl1 == '4') and (pl2 == '2' or pl2 == '3'):
        print("Игрок", Pl1, "ДОВЕРИЛСЯ (",PO[2],")
            Игрок", Pl2, "ОБМАНУЛ (+",PO[1],") ")
        PayOff1+=PO[2]
        PayOff2+=PO[1]

    elif (pl1 == '2' or pl1 == '3') and (pl2 == '1' or pl2 == '4'):
        print("Игрок", Pl1, "ОБМАНУЛ (+",PO[1],")
            Игрок", Pl2, "ДОВЕРИЛСЯ (",PO[2],") ")
        PayOff1+=PO[1]
        PayOff2+=PO[2]

    else:
        print("!!!!!!Неверные команды!!!!!!")
        i-=1

j=0

while j<Games:
    print()
    print("                  !!!!! Тур", j+1," / 3 !!!!!")
    PayOff1 = PayOff2 = 0
    i = 0
    R[j] = Rounds
    while i<Rounds:
        print()
        print("              !!! Раунд", i+1, " / 6 !!!")
        key1 = getpass.getpass("Ходит первый игрок")
        key1 = mistake(key1)
        key2 = getpass.getpass("Ходит второй игрок")
        key2 = mistake(key2)

        #if (key1 == '1' or key1 == '2') and (key2 == '1' or key2 == '2'):
        DataPl1[j][i] = key1
        DataPl2[j][i] = key2
        payoff(key1, key2)
        i+=1

    print()
    print("Выигрыш игрока", Pl1, ": ", PayOff1, "
        Выигрыш игрока", Pl2, ": " ,PayOff2)
    print("Максимальный возможный выигрыш: ", Rounds*PO[1],
        "    Минимальный возможный выигрыш: ", Rounds*PO[2])
    TotalPO1+=PayOff1
    TotalPO2+=PayOff2
    j+=1

print()
print("Итоговый выигрыш игрока", Pl1, ": ", TotalPO1,
    "   Итоговый выигрыш игрока", Pl2, ": " ,TotalPO2)
print("Максимальный возможный выигрыш: ", np.sum(R)*PO[1],
    "    Минимальный возможный выигрыш: ", np.sum(R)*PO[2])

print(DataPl1)
print(DataPl2)

filename = "C:\Exp\\"+Player1+'_и_'+Player2+'.xls'
```

```
df1 = pd.DataFrame(DataPl1)
df2 = pd.DataFrame(DataPl2)

with pd.ExcelWriter(filename) as writer:
    df1.to_excel(writer, sheet_name=Player1)
    df2.to_excel(writer, sheet_name=Player2)
```

Листинг 2. Игра «Дилемма заключенного», реализованная на языке Python с помощью бота на платформе «Telegram».

```
File: bot.py

import logging

from telegram.ext import CommandHandler
from telegram.ext import MessageHandler
from telegram.ext import Updater
from telegram.ext import Filters

from settings import TG_TOKEN
from settings import TG_API_URL
from handlers import *

logging.basicConfig(format='%(asctime)s - %(levelname)s - %(message)s',
                    level=logging.INFO,
                    filename='bot.log'
                    )



def main():
    my_bot = Updater(TG_TOKEN,
                     #TG_API_URL,
                     use_context=True)
    logging.info('Start bot')
    my_bot.dispatcher.add_handler(CommandHandler('start', sms))
    my_bot.dispatcher.add_handler(MessageHandler(Filters.regex('Начать'),
    sms))

    my_bot.dispatcher.add_handler(
        ConversationHandler(entry_points=[MessageHandler
        (Filters.regex('Заполнить анкету'), anketa_start)],
                            states={
                                "user_name": [MessageHandler
                                (Filters.text, anketa_get_name)],
                                "user_age": [MessageHandler
                                (Filters.text, anketa_get_age)],
                                "evaluation": [MessageHandler
                                (Filters.regex('1|2|3|4|5'),
                                anketa_get_evaluation)],
                                "comment": [MessageHandler
                                (Filters.regex('Пропустить'),
                                anketa_exit_comment),
                                            MessageHandler
                                            (Filters.text,
```

```
                                        anketa_comment)],
                            },
                            fallbacks=[MessageHandler
                            (
                                Filters.text | Filters.video |
                                Filters.photo | Filters.document,
                                dontknow)]
                            )
    )
    my_bot.dispatcher.add_handler(
        ConversationHandler(entry_points=[MessageHandler
        (Filters.regex('Игра'), find_enemy)],
                            states={
                                "Game": [MessageHandler
                                (Filters.text, Game)],
                            },
                            fallbacks=[MessageHandler(
                                Filters.text | Filters.video |
                                Filters.photo | Filters.document,
                                dontknow)]
                            )
    )
    my_bot.start_polling()
    my_bot.idle()


if __name__ == "__main__":
    main()
```

File: settings.py

```
from pymongo import MongoClient

TG_TOKEN = "718021376:AAGkyHg_s2RiFKm-Cei6oSG57M5sQlYWzYk"
TG_API_URL = "https://telegg.ru/orig/bot"
MONGODB_LINK =
"mongodb+srv://BotFromPrison:2688737
@playersdb-vdzvd.mongodb.net/test?retryWrites=true&w=majority"
MONGO_DB = "PlayersDB"

mdb = MongoClient(MONGODB_LINK)[MONGO_DB]
```

File: mongodb.py

```
from pymongo import MongoClient
from settings import MONGO_DB
from settings import MONGODB_LINK
import requests
from telegram import ReplyKeyboardMarkup

mdb = MongoClient(MONGODB_LINK)[MONGO_DB]
keyboard_ready = ReplyKeyboardMarkup([['Готов']], resize_keyboard=True)
keyboard_ans = ReplyKeyboardMarkup([['Довериться'], ['Обмануть']],
```

```
resize_keyboard=True)

def search_or_save_user(mdb, effective_user, message):
    user = mdb.users.find_one({"user_id": effective_user.id})
    if not user:
        user = {
            "user_id": effective_user.id,
            "first_name": effective_user.first_name,
            "last_name": effective_user.last_name,
            "chat_id": message.chat.id,
            "moves_in_a_game": [],
            "turns": 0,
            "score": 0,
            "scoreGame1": 0,
            "scoreGame2": 0,
            'GamePast': 0
        }
        mdb.users.insert_one(user)
    return user

def save_user_anketa(mdb, user, user_data):
    mdb.users.update_one(
        {'_id': user['_id']},
        {'$set': {'anketa': {'name': user_data['name'],
                             'age': user_data['age'],
                             'evaluation': user_data['evaluation'],
                             'comment': user_data['comment']
                             }
                  }
         }
    )
    print(user_data['name'])
    return user

def send_mess(id, text):
    requests.get('https://api.telegram.org/bot{}/sendMessage'.
    format("718021376:AAGkyHg_s2RiFKm-Cei6oSG57M5sQlYWzYk"),
    params=dict(
        chat_id=id,
        text=text
    ))

def save_user_enemy(mdb, ID, bot):

    user = mdb.users.find_one({"user_id": bot.effective_user.id})
    enemy = mdb.users.find_one({"user_id": ID})

    mdb.users.update_one(
        {'_id': user['_id']},
        {'$set': {'game': {'game_enemy': ID,
                           'game_enemy_name': enemy['anketa']['name']}
                  }
         }
    )

    mdb.users.update_one(
        {'_id': enemy['_id']},
```

```
        {'$set': {'game': {'game_enemy': bot.effective_user.id,
                           'game_enemy_name': user['anketa']['name']
                           }
                  }
         }
    )

    enemy_info = "Твой противник: " + enemy['anketa']['name'] +
    "\n" + "Выберите первый ход"
    user_info = "Твой противник: " + user['anketa']['name']
    send_mess(ID, user_info)
    bot.message.reply_text(enemy_info,
                           reply_markup=keyboard_ans)

    print("Получилась пара: ", user['anketa']['name'], "  и  ",
    enemy['anketa']['name'])
```

File: handlers.py

```
from telegram import ParseMode
from telegram import ReplyKeyboardMarkup
from telegram import ReplyKeyboardRemove
from telegram.ext import ConversationHandler
import random

from mongodb import mdb, search_or_save_user, save_user_anketa,
save_user_enemy, send_mess

keyboard_ans = ReplyKeyboardMarkup([['Довериться'], ['Обмануть']],
resize_keyboard=True)
keyboard_ready = ReplyKeyboardMarkup([['Готов']], resize_keyboard=True)

keyboard_anketa = ReplyKeyboardMarkup([["Заполнить анкету"]],
resize_keyboard=True)
keyboard_game = ReplyKeyboardMarkup([['Игра']], resize_keyboard=True)

ans0_0 = "Ты доверился (+2). Твой противник доверился (+2)"
ans0_1 = "Ты доверился (-1). Твой противник обманул (+3)"
ans1_0 = "Ты обманул (+3). Твой противник доверился (-1)"
ans1_1 = "Ты обманул (+0). Твой противник обманул (+0)"

text = "Извините, но вы уже заполняли анкету"
text2 = "Извините, но вы уже играли"

def sms(bot, update):
    user = search_or_save_user(mdb, bot.effective_user, bot.message)
    print(user)
    print('Кто-то отправил команду /start. Что мне делать?')
    bot.message.reply_text('Здравствуй, {}! \n'
                   'Я провожу одно исследование, но для этого нужно
                   провести очень много игр, а потом провести опрос. Все
                   потробности будут изложены далее.\n'
                    'Это займет не более 5-7 минут. \n'
                   'Спасибо, что согласились помочь. Внимательно
                   следуйте дальнейшим указаниям. \n'
```

```
                    'Нажмите на кнопку "Заполнить анкету" '
                    .format(bot.message.chat.first_name),
                    reply_markup=keyboard_anketa)

def anketa_start(bot, update):
    user = search_or_save_user(mdb, bot.effective_user, bot.message)
    if 'anketa' in user:
        bot.message.reply_text(
            text,
            reply_markup=ReplyKeyboardRemove())
        return ConversationHandler.END
    else:
        bot.message.reply_text(
            'Напиши, пожалуйста, Имя и Фамилию?',
            reply_markup=ReplyKeyboardRemove())
        return "user_name"

def anketa_get_name(bot, update):
    update.user_data['name'] = bot.message.text
    bot.message.reply_text("Сколько вам лет?")
    return "user_age"

def anketa_get_age(bot, update):
    update.user_data['age'] = bot.message.text
    reply_keyboard = [["1", "2", "3", "4", "5"]]
    bot.message.reply_text(
        "Оцени свое настроение от 1 до 5",
        reply_markup=ReplyKeyboardMarkup(
            reply_keyboard, resize_keyboard=True, one_time_keyboard=True))
    return "evaluation"

def anketa_get_evaluation(bot, update):
    update.user_data['evaluation'] = bot.message.text
    reply_keyboard = [["Пропустить"]]
    bot.message.reply_text("Можешь написать моему
    создателю или нажмите кнопку
    пропустить этот шаг.",
                           reply_markup=ReplyKeyboardMarkup(
                               reply_keyboard, resize_keyboard=True,
                               one_time_keyboard=True))
    return "comment"


def anketa_comment(bot, update):
    update.user_data['comment'] = bot.message.text
    user = search_or_save_user(mdb, bot.effective_user, bot.message)
    anketa = save_user_anketa(mdb, user, update.user_data)
    text = """Результат опроса:
    <b>Имя:</b> {name}
    <b>Возраст:</b> {age}
    <b>Оценка:</b> {evaluation}
    <b>Комментарий:</b> {comment}
    """.format(**update.user_data)
    bot.message.reply_text(text, parse_mode=ParseMode.HTML)  # текстовое
    сообщение с форматированием HTML
    bot.message.reply_text("Спасибо \n"
                    "А теперь что будет дальше. Ты будешь играть в игру
```

```
                    со случайным человеком\n"
                    "Тебе будет предложено сыграть 2 тура по 6 раундов
                    (т.е. всего 12 раундов). В каждом раунде "
                    "у тебя (как и у твоего противника) всего две опции:
                    довериться или обмануть. "
                    "Если вы оба доверяетесь, то получаете выигрыш +2.
                    Если вы оба обманываете, то вы ничего не получаете
                    (т.е. +0). "
                    "Но если кто-то из вас решил обмануть, а другой
                    довериться, то обманувший получает +3, а доверчивый "
                    "теряет 1 балл (т.е. -1). В конце каждого тура (т.е.
                    после 6-ти раундов) ты увидешь свой счет и счет
                    противника за этот тур. Твоя цель: набрать больше
                    баллов, чем противник \n"
                    "И последнее. В игру встроена вероятность ошибки 0.1.
                    Это означает, что с вероятностью 0.1 твой выбор (как
                    и выбор соперника) поменяется на противоположный. "
                    "Поэтому если тебя вдруг обманут, подумай, может это
                    не обман, а ошибка\n"
                    "Если все понятно, то нажми кнопку 'Игра'",
                           reply_markup=keyboard_game)
    return ConversationHandler.END


def anketa_exit_comment(bot, update):
    update.user_data['comment'] = None
    user = search_or_save_user(mdb, bot.effective_user, bot.message)
    save_user_anketa(mdb, user, update.user_data)
    text = """Результат опроса:
    <b>Имя:</b> {name}
    <b>Возраст:</b> {age}
    <b>Оценка:</b> {evaluation}""".format(**update.user_data)
    bot.message.reply_text(text, parse_mode=ParseMode.HTML)
    bot.message.reply_text("Спасибо \n"
                    "А теперь что будет дальше. Ты будешь играть в игру
                    со случайным человеком\n"
                    "Тебе будет предложено сыграть 2 тура по 6 раундов
                    (т.е. всего 12 раундов). В каждом раунде "
                    "у тебя (как и у твоего противника) всего две опции:
                    довериться или обмануть. "
                    "Если вы оба доверяетесь, то получаете выигрыш +2.
                    Если вы оба обманываете, то вы ничего не получаете
                    (т.е. +0). "
                    "Но если кто-то из вас решил обмануть, а другой
                    довериться, то обманувший получает +3, а доверчивый "
                    "теряет 1 балл (т.е. -1). В конце каждого тура (т.е.
                    после 6-ти раундов) ты увидешь свой счет и счет
                    противника за этот тур. Твоя цель: набрать больше
                    баллов, чем противник \n"
                    "И последнее. В игру встроена вероятность ошибки 0.1.
                    Это означает, что с вероятностью 0.1 твой выбор (как
                    и выбор соперника) поменяется на противоположный. "
                    "Поэтому если тебя вдруг обманут, подумай, может это
                    не обман, а ошибка \n"
                    "Если все понятно, то нажми кнопку 'Игра'",
                           reply_markup=keyboard_game)
    return ConversationHandler.END
```

```
ID = 0
def find_enemy(bot, update):
    print("Начали")

    global ID
    user = mdb.users.find_one({"user_id": bot.effective_user.id})

    if user['GamePast'] == 1:
        bot.message.reply_text(text2, reply_markup=ReplyKeyboardRemove())
        return ConversationHandler.END

    else:
        if ID == 0:
            ID = bot.effective_user.id
            bot.message.reply_text("Ничего (!!!) не нажимайте,
            ожидайте противника!(Это может занять некоторое время)
            Выберите действие,
            когда увидите имя своего оппонента",
            reply_markup=keyboard_ans)
        else:
            IDbuf = ID
            ID = 0
            save_user_enemy(mdb, IDbuf, bot)
        return "Game"

mis = 0.1
def Game(bot, update):
    user = mdb.users.find_one({"user_id": bot.effective_user.id})
    id_enemy = user['game']['game_enemy']
    enemy = mdb.users.find_one({"user_id": id_enemy})

    if random.random() > mis:

        if bot.message.text == 'Довериться':

            mdb.users.update({'_id': user['_id']}, {'$push':
            {'moves_in_a_game': 1}})

        elif bot.message.text == 'Обмануть':
            mdb.users.update({'_id': user['_id']}, {'$push':
            {'moves_in_a_game': 2}})

        else:
            print("Что-то пошло не так")

    else:

        if bot.message.text == 'Довериться':

            mdb.users.update({'_id': user['_id']}, {'$push':
            {'moves_in_a_game': -1}})

        elif bot.message.text == 'Обмануть':
            mdb.users.update({'_id': user['_id']}, {'$push':
            {'moves_in_a_game': -2}})
```

```
        else:
            print("Что-то пошло не так")

    mdb.users.update_one(
        {'_id': user['_id']},
        {'$set': {'turns': user['turns']+1}}
    )

    GameResult(bot, update)
    if (user['turns'] == 11):
        print(user['anketa']['name'], " вышел")
        mdb.users.update({'_id': user['_id']}, {'$set': {'GamePast': 1}})
        return ConversationHandler.END


def GameResult(bot, update):
    user = mdb.users.find_one({"user_id": bot.effective_user.id})
    id_enemy = user['game']['game_enemy']
    enemy = mdb.users.find_one({"user_id": id_enemy})

    if user['turns']!=enemy['turns']:
        if user['turns'] != 12:
            bot.message.reply_text("Ничего(!!!) не нажимай, подожди, твой
            противник еще не походил",
                                   reply_markup=keyboard_ans)
        else:
            bot.message.reply_text("Ничего(!!!) не нажимай, подожди, твой
            противник еще не походил",
                                   reply_markup=ReplyKeyboardRemove())

    else:
        if (user['moves_in_a_game'][-1] == 1 or
        user['moves_in_a_game'][-1] ==
        -2) and (enemy['moves_in_a_game'][-1] == 1 or
        enemy['moves_in_a_game'][-1] == -2):

            mdb.users.update({'_id': user['_id']}, {'$set': {'score':
            user['score']+2}})
            mdb.users.update({'_id': enemy['_id']}, {'$set': {'score':
            enemy['score']+2}})

            bot.message.reply_text(ans0_0)
            send_mess(id_enemy, ans0_0)

            if (user['turns'] != 6) and (user['turns'] != 12):
                bot.message.reply_text("Доверяешься или обманываешь?",
                                       reply_markup=keyboard_ans)
                send_mess(id_enemy, "Доверяешься или обманываешь?")

        elif (user['moves_in_a_game'][-1] == 1 or
        user['moves_in_a_game'][-1] ==
        -2) and (enemy['moves_in_a_game'][-1] == 2 or
        enemy['moves_in_a_game'][-1] == -1):

            mdb.users.update({'_id': user['_id']}, {'$set': {'score':
            user['score']-1}})
```

```
            mdb.users.update({'_id': enemy['_id']}, {'$set': {'score':
            enemy['score']+3}})

            bot.message.reply_text(ans0_1)
            send_mess(id_enemy, ans1_0)

            if (user['turns'] != 6) and (user['turns'] != 12):
                bot.message.reply_text("Доверяешься или обманываешь?",
                                       reply_markup=keyboard_ans)
                send_mess(id_enemy, "Доверяешься или обманываешь?")

        elif (user['moves_in_a_game'][-1] == 2 or
        user['moves_in_a_game'][-1] ==
        -1) and (enemy['moves_in_a_game'][-1] == 1 or
        enemy['moves_in_a_game'][-1] == -2):

            mdb.users.update({'_id': user['_id']}, {'$set': {'score':
            user['score']+3}})
            mdb.users.update({'_id': enemy['_id']}, {'$set': {'score':
            enemy['score']-1}})

            bot.message.reply_text(ans1_0)
            send_mess(id_enemy, ans0_1)

            if (user['turns'] != 6) and (user['turns'] != 12):
                bot.message.reply_text("Доверяешься или обманываешь?",
                                       reply_markup=keyboard_ans)
                send_mess(id_enemy, "Доверяешься или обманываешь?")

        elif (user['moves_in_a_game'][-1] == 2 or
        user['moves_in_a_game'][-1]
        == -1) and (enemy['moves_in_a_game'][-1] == 2 or
        enemy['moves_in_a_game'][-1] == -1):
            bot.message.reply_text(ans1_1)
            send_mess(id_enemy, ans1_1)

            if (user['turns'] != 6) and (user['turns'] != 12):
                bot.message.reply_text("Доверяешься или обманываешь?",
                                       reply_markup=keyboard_ans)
                send_mess(id_enemy, "Доверяешься или обманываешь?")

    if (user['turns'] == enemy['turns']) and ((user['turns'] == 6) or
    (user['turns'] == 12)):
        final_round(bot, update)

def mistake(dis):
    r = random.random()
    if r < mis:
        return(int(dis)+2)
    else:
        return(dis)

def final_round(bot, update):
    user = mdb.users.find_one({"user_id": bot.effective_user.id})
    id_enemy = user['game']['game_enemy']
    enemy = mdb.users.find_one({"user_id": id_enemy})
```

```
    bot.message.reply_text("Прошел тур (т.е. 6 раундов)
    !!!! Вот результаты:")
    send_mess(id_enemy, "Прошел тур (т.е. 6 раундов)
    !!!! Вот результаты:")

    sc_user = "Твой счет: " + str(user['score']) + "  Счет противника: " +
    str(enemy['score']) + str(" Максимум можно было набрать 18 баллов")
    sc_enemy = "Твой счет: " + str(enemy['score']) + "  Счет противника: " +
    str(user['score']) + str(" Максимум можно было набрать 18 баллов")
    bot.message.reply_text(sc_user)
    send_mess(id_enemy, sc_enemy)

    if (user['turns'] == 6):
        mdb.users.update({'_id': user['_id']}, {'$set': {'scoreGame1':
        user['score']}})
        mdb.users.update({'_id': enemy['_id']}, {'$set': {'scoreGame1':
        enemy['score']}})

    if (user['turns'] == 12):
        mdb.users.update({'_id': user['_id']}, {'$set': {'scoreGame2':
        user['score']}})
        mdb.users.update({'_id': enemy['_id']}, {'$set': {'scoreGame2':
        enemy['score']}})

    mdb.users.update({'_id': user['_id']}, {'$set': {'score': 0}})
    mdb.users.update({'_id': enemy['_id']}, {'$set': {'score': 0}})

    if (user['turns'] == 6):

        bot.message.reply_text("Доверяешься или обманываешь?",
                               reply_markup=keyboard_ans)
        send_mess(id_enemy, "Доверяешься или обманываешь?")
    else:

        print("КОНЕЦ!!!")
        bot.message.reply_text("Спасибо большое:)
        Пройди, пожалуйста, эту
        анкету, она буквально на 2 минуты. Она тоже нужна для диплома\n"
                               "https://docs.google.com/forms/d/
                               1fheS-j8_Rc8BHLxm8Y8GLApGrqv7WZGiXfKerfZqK1E/
                               edit",
                               reply_markup=ReplyKeyboardRemove())
        send_mess(id_enemy, "Спасибо большое:)
        Пройди, пожалуйста, эту анкету,
        она буквально на 2 минуты.
        Она тоже нужна для диплома\n"
                            "https://docs.google.com/forms/d/
                            1fheS-j8_Rc8BHLxm8Y8GLApGrqv7WZGiXfKerfZqK1E/
                            edit")


def dontknow(bot, update):
    bot.message.reply_text("Я вас не понимаю, выберите оценку на клавиатуре!")
```