ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОЕКТУ ПО ТЕМЕ:

“Анализ шахматных партий на использование компьютерных подсказок”

Содержание

1. Общие сведения о проблеме ………………………………………..……...2
2. Выгрузка данных и вычисление необходимых метрик…………...………3
3. Статистический анализ данных……………………………………………7
4. Заключение ………………………………………………………………. 19

**1. Общие сведения о проблеме**

Одной из главных проблем в современных онлайн-шахматах является использование многими игроками подсказок компьютерных программ (далее – шахматных движков или просто движков) во время игры. Любой игрок может прямо во время онлайн-партии открыть на втором мониторе или просто в другой вкладке браузера шахматный движок – программу, рассчитывающую более 40 миллионов шахматных позиций в секунду, и способную подсказывать игроку наилучшие ходы в любой шахматной позиции.

Уже в начале 2000-х годов шахматные движки были намного сильнее любого игрока на планете. После этого разрыв между возможностями шахматных движков и игрой живых игроков многократно вырос, поэтому в партии против игрока, использующего подсказки компьютера, даже у сильнейших шахматистов планеты нет никаких шансов.

Особенно большие проблемы с нечестными игроками возникают в турнирах с призовым фондом, т.е. в турнирах, в которых за хороший результат можно получить денежное вознаграждение. Для того, чтобы недопустить занятия призовых мест мошенниками, которые пользуются подсказками движка, необходима тщательная проверка подозрительных игроков (далее – проверка на читерство).

Современные методы проверки игроков на читерство включают в себя:

1. Анализ конкретных партий, в которых игрок сыграл подозрительно сильно
2. Анализ движения глаз игрока во время партии
3. Попытка нахождения прямых доказательств мошенничества.

Но самым мощным инструментом анализа выступления игрока в турнире является статистический анализ, в котором разбираются данные из прошлых партий игрока и сравниваются с его выступлением в данном турнире. В данном проекте представлен 1 из способов реализации программы, анализирующего выступление игрока за некоторый промежуток времени (например, за данный турнир) и за некоторый более длительный промежуток времени (например, за последние 6 месяцев перед этим турниром).

**2. Выгрузка данных и вычисление необходимых для анализа метрик**

В данном проекте будут рассматриваться партии, сыгранные на сайте lichess.org. Это 2-ой по популярности шахматный сайт, на котором играют сотни тысяч игроков.

Для выгрузки партий с данного сайта необходимо зарегистрироваться в нём и получить персональный токен доступа к его API. Это можно сделать, зайдя в настройки сайта —> Токены доступа к API —> Новый персональный токен доступа к API, после чего выбрать необходимо поставить все галочки и получить персональный токен доступа (крайне важно никому его не сообщать и удалить после использования, чтобы ни у кого не было доступа к вашей учетной записи на сайте).

В python есть библиотека berserk, позволяющая направлять запросы на сайт, используя персональный токен доступа к API, полученный на предыдущем шаге. Благодаря библиотеке berserk создаются 2 основных объекта:

# Объект, который управляет HTTP-сессией и добавляет ваш API-токен к каждому запросу.

session = berserk.TokenSession(token)

# Объект TokenSession, который будет использоваться для запросов.

client = berserk.Client(session)

При использовании функции berserk.TokenSession может возникнуть ошибка: TypeError: JSONDecoder.\_\_init\_\_() got an unexpected keyword argument 'encoding'

Она возникает из-за того, что в библиотеке berserk данные скачиваются в формате json, а функция berserk.TokenSession при декодировании пытается подать аргумент 'encoding', которого нет в JSONDecoder, начиная с версии python 3.9, поэтому для работы с данными необходимо виртуальное окружение с python версией 3.8 и ниже. В файле c кодом есть полная настройка для работы с программой в среде выполнения Google Colab.

Первым шагом к анализу шахматных партий является их выгрузка. В коде она реализована функцией

get\_player\_games(username, days=365, perf\_type="blitz", end\_date=None, token='lip\_XdtZYxfUqOjOV0zNOSck')

которая берет на вход никнейм рассматриваемого игрока, тип партий (пуля, блиц, рапид или классика), API-токен, временной интервал, в котором будут рассмотрены партии в днях, конечная дата интервала (по умолчанию в коде она задается как день, в который код был запущен) и сохраняет партии в формате PGN (формат файла для сохранения шахматных партий).

Затем для анализа партий необходимо скачать движок, например Stockfish.

!sudo apt-get install stockfish

!pip install chess-engine

Анализ партий с использованием движка реализован в функции

analyze\_games\_with\_engine(games, engine\_path='/usr/games/stockfish', depth=20)

которая принимает список партий, полученный в ходе работы предыдущей функции, расположение скачанного движка и заданную глубину движка (на сколько ходов вперед движок считает партии в каждой позиции) и возвращает список all\_games\_scores, каждый элемент которого - список оценок позиций после каждого хода игроков.

После получения оценок движка на каждом этапе каждой партии необходимо их проанализировать. Шахматные движки оценивают позиции в эквиваленте числа пешек, которое могло бы уравнять позицию, для белых это число положительно, а для черных – отрицательно. Например, оценка +2 означает, что данная позиция эквивалентна тому, если бы у белых было бы преимущество в 2 пешки перед черными. Из оценки позиции в пешках можно получить ключевую для анализа партий на читерство величину – потерю сантипешек за ход, которая показывает, насколько в пешечном эквиваленте данный ход ухудшает позицию (или насколько сделанный игроком ход хуже оптимального с точки зрения компьютера). Для анализа партий используется средняя потеря сантипешек (Average centipawns loss) игроком за всю партию:

Где – pawn loss, потеря пешек за i-й ход

n – количество ходов, сделанных игроком в партии

Ходы, в которых было потеряно более 50 сантипешек называют неточностями (Inaccuracies), более 100 – ошибками (Mistakes), более 300 – зевками (Blunders).

Для анализа партии на читерство сантипешек зачастую недостаточно, ведь в разных позициях разные потери сантипешек могут внести разный вклад в партию. Например, при равной позиции потеря 1.5 пешек является ошибкой, которая серьезно снижает шансы на победу, а в позиции с тремя лишними фигурами потеря одной из этих фигур практически никак не уменьшит шанс на победу. Поэтому многие сайты, в том числе lichess используют другую метрику – точность (Accuracy). Для ее нахождения необходимо на каждом ходу для игрока вычислить следующую величину:

Где – текущая оценка позиции.

С помощью этой величины вычисляется:

Где - точность данного хода

*–* Win% before. Win% до хода

*–* Win% after. Win% после хода

Для анализа партий на читерство используется суммарная точность всех ходов в партии, она может принимать значения от 0% до 100%.

За подсчет потерянных сантипешек отвечает функция

analyze\_player\_losses(all\_games\_scores)

Ее аргументом является список оценок каждого этапа каждой партии, получаемый в результате работы предыдущей описанной функции analyze\_games\_with\_engine(). А на выходе функция возвращает списки с потерями сантипешек в каждом ходу в каждой партии.

Для подсчета Win% и Accuracy% в каждом ходу вводятся вспомогательные функции:

calculate\_win\_percent(centipawns) и calculate\_accuracy(win\_diff)

Для выгрузки и полного анализа партий на читерство вводится функция

analyze\_player\_performance(username, engine\_path='/usr/games/stockfish', token='lip\_XdtZYxfUqOjOV0zNOSck',  end\_date=None, days=365, perf\_type="blitz", depth=20)

Эта функция сначала выгружает партии указанного игрока за указанный промежуток времени, используя функцию get\_player\_games(), анализирует их с помощью движка, используя analyze\_games\_with\_engine(), после чего на каждом этапе партии рассчитывает потерянные игроком сантипешки с помощью analyze\_player\_losses(), после чего на каждом ходу рассчитывает Win% и Accuracy% и, наконец, рассчитывает 5 основных метрик, которые будут использоваться для проведения анализа партий:

1) Средние потери сантипешек в партии

2) Общая точность партии

3) Число зевков в партии

4) Число ошибок в партии

5) Число неточностей в партии

Помимо этих параметров можно использовать и другие. Например, анализ времени, потраченного на ходы игрока, может выявить неопытного читера, так как компьютер для расчета позиции на заданной глубине на каждом ходу тратит примерно одинаковое время и некоторые игроки сильно выдают себя, когда делают хорошие ходы ровно раз в 5 секунд.

**3. Статистический анализ данных**

После загрузки и обработки данных их необходимо проанализировать. Ключевым вопросом данного проекта является: Использовал ли игрок компьютерные подсказки в рассматриваемых партиях, если у нас есть статистика по его предыдущим партиям). Ключевой проверяемой величиной будут средние значения описанных в предыдущей главе метрик. Тогда со статистической значимостью 5% можно сформулировать следующие гипотезы для точности:

*H0 – в сыгранных на турнире партиях нет статистической разницы в точности с остальными сыгранными игроком партиями*

*H1 – в сыгранных на турнире партиях показана более высокая точность, чем в остальных сыгранных игроком партиях*

Для потерь сантипешек, зевков, ошибок и неточностей альтернативная гипотеза будет левосторонней:

*H0 – в сыгранных на турнире партиях нет статистической разницы в средних потерях сантипешек с остальнми сыгранными игроком партиями*

*H1 – в сыгранных на турнире партиях показана более низкие средние потери сантипешек, чем в остальных сыгранных игроком партиях*

Чтобы проанализировать поставленные гипотезы сначала необходимо провести описательную статистику указанных величин. Это можно сделать с помощью метода describe(), примененного к датафреймам players\_stats и tournament\_stats, полученных при анализе партий подозреваемого игрока до турнира и в самом турнире. Описание этих таблиц приведено на рисунках 1 и 2.

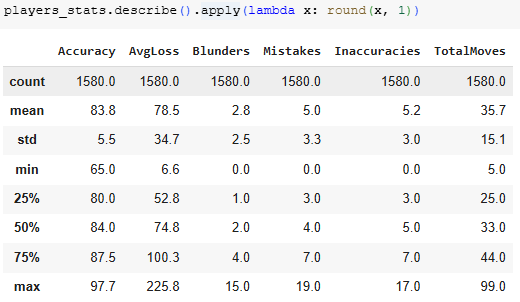


Рисунок 1. Описание общей статистики игрока

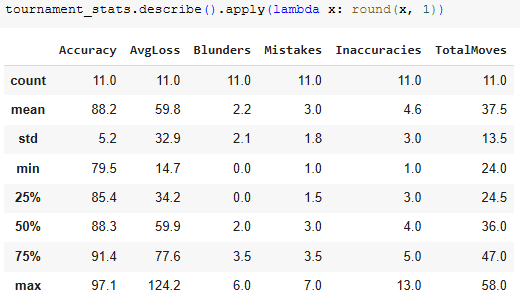


Рисунок 2. Описание статистики игрока за турнир

Видно, что средняя и медианная точность в сыгранном турнире у игрока выше, чем обычно, а среднее и медианное значение других показателей – наоборот ниже. Также стоит отметить, что выборка турнирных партий значительно ниже общей выборки (11 партий против 1580)

На этом какие-либо выводы делать очень рано. Для начала построим графики распределения всех 5 метрик для датафрейма players\_stats. Для этого была написана функция

plot\_all\_metrics(players\_stats)

Она строит гистограммы для каждой из рассматриваемых величин и красную пунктирную линию, показывающую среднее значение данной метрики. Результат действия этой функции для общего набора данных (т.е. сыгранных до турнира партий) приведен на рисунке 3. Для турнирных партий строить гистограммы смысла не имеет, так как их выборка равна всего 11, поэтому такие гистограммы не внесут ничего информативного.

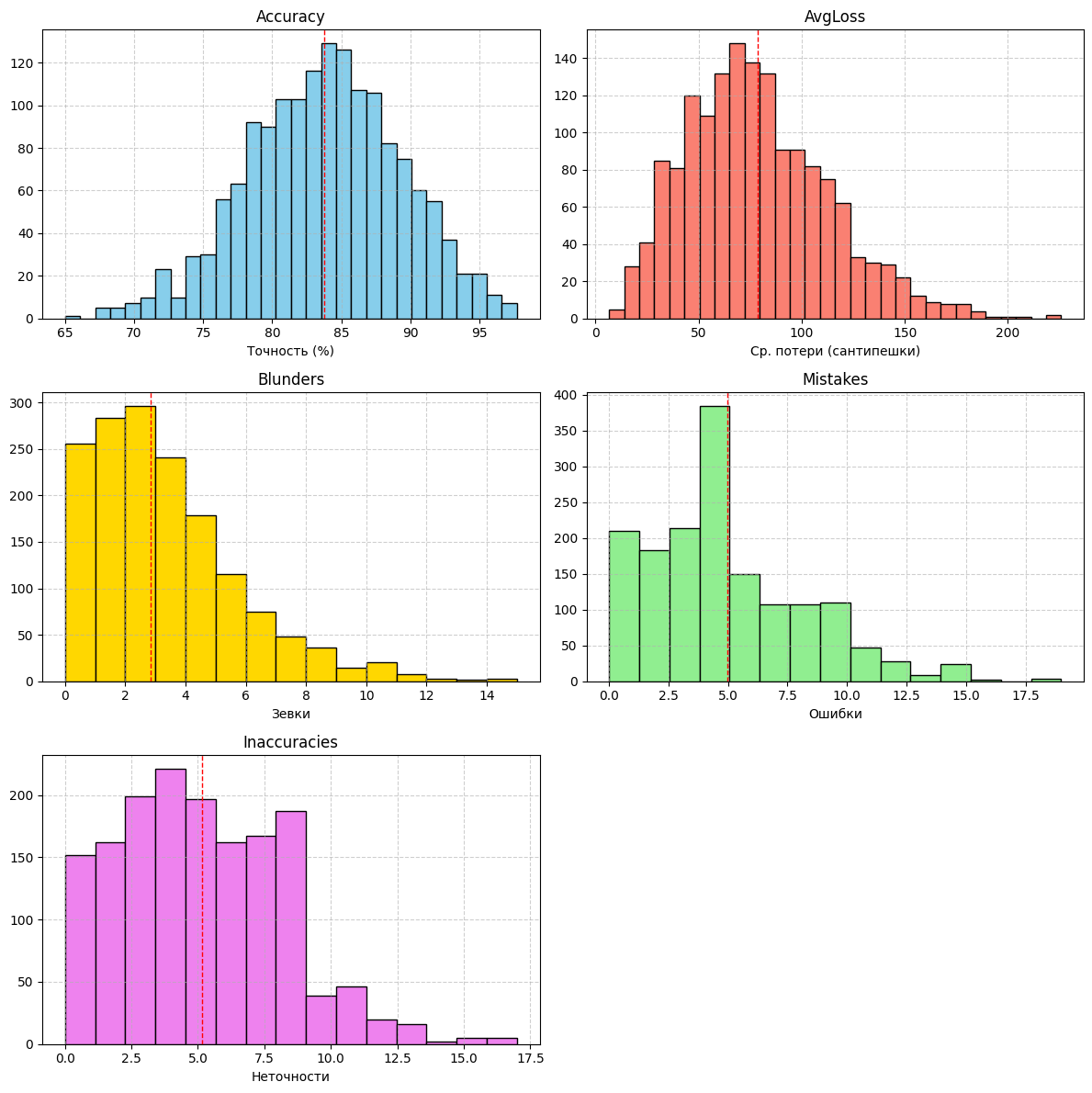


Рисунок 3. Гистограммы основных метрик для общего набора партий

Далее необходимо проверить данные на нормальность. Визуально графики точности и потерь сантипешек напоминают нормальные распределения, а остальные графики нет. Помимо визуального анализа для проверки данных на нормальность необходимо учесть следующие факторы:

1) Насколько близки друг к другу среднее значение, медиана и мода.

2) Насколько близки ассиметрия и коэффициент эксцесса к 0

3) Насколько QQ-plot (Quantile-Quantile plot) близок к линейной зависимости

4) Значение и p-value критериев, определяющих нормальность распределения. Если это значение будет >0.05, гипотезу можно считать верной. В качестве таких критериев можно использовать критерий Шапиро-Уилка, применимый для выборок объемом меньше 5000 и реализованный в функции stats.shapiro() в python, а также критерий К-квадрат Д'Агостино, проверяющий ассиметрию и эксцесс и реализованный в функции stats.normaltest() в python.

В этом исследовании написана функция, проверяющая данные на нормальность:

normality\_check (data)

Эта функция строит гистограмму распределения, вычисляет некоторые статистические характеристики, проверяет гипотезы о нормальности распределения и строит QQ-plot. Результаты использования этой функции для определения нормальности распределения потерь сантипешек в общей статистике приведены на рисунке 4.

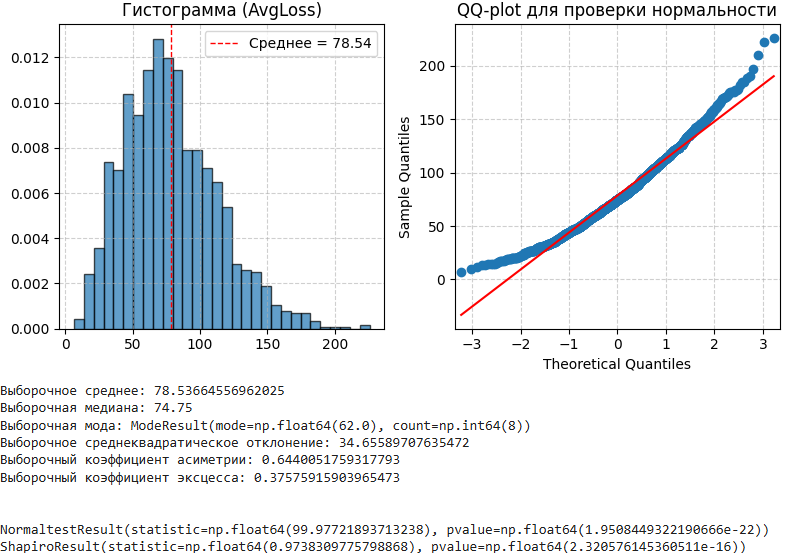


Рис.4. Оценка нормальности распределения потерь сантипешек

Исходные данные о средних потерях сантипешек(AvgLoss) имеют правостороннюю асимметрию (коэффициент = 0.64) и отклонение от нормальности (p-value Шапиро-Уилка < 0.001). Для нормализации можно применить квадратное преобразование f(x)= . Результат проверки нормализованного распределения показан на рисунке 5.

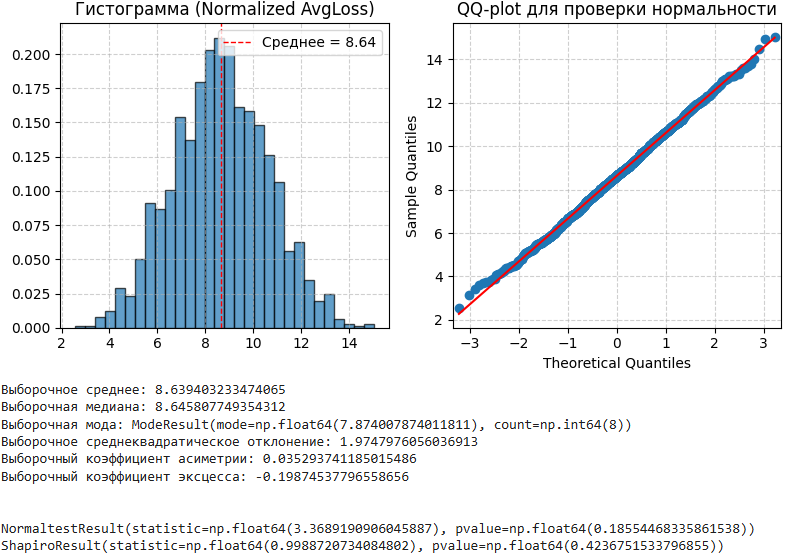


Рисунок 5. Проверка нормализованных данных о потерях сантипешек

Как видно из рис.5, после преобразования квадратным корнем распределение не отличается от нормального (p-value = 0.424, асимметрия = 0.035). Все дальнейшие анализы будут проведены с преобразованными данными.

По аналогии со средними потерями сантипешек, необходимо проверить гипотезу о нормальности распределения значений точности. На рисунке 6 приведен результат проверки этой гипотезы.

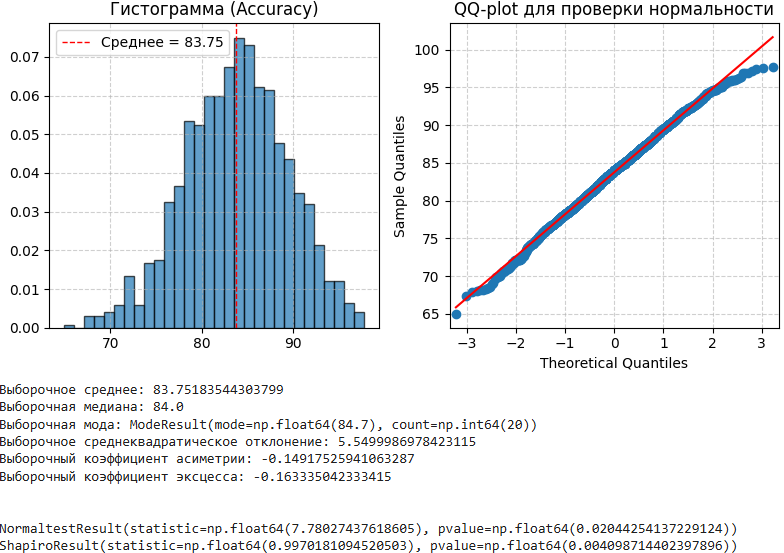


Рисунок 6. Оценка нормальности распределения точностей

Исходные данные о точности (Accuracy) имеют левостороннюю асимметрию (коэффициент = -0.149) и отклонение от нормальности (p-value Шапиро-Уилка = 0.004). Если руководствоваться критерием К-квадрат Д'Агостино, то с уровнем значимости в 1% гипотезу о нормальности данных можно принять, так как p-value = 0.02, но с уровнем значимости в 5% - принять ее нельзя. Для нормализации в таких случаях можно применить квадратичное преобразование f(x)=. На рисунке 7 указано распределение квадратов точностей.

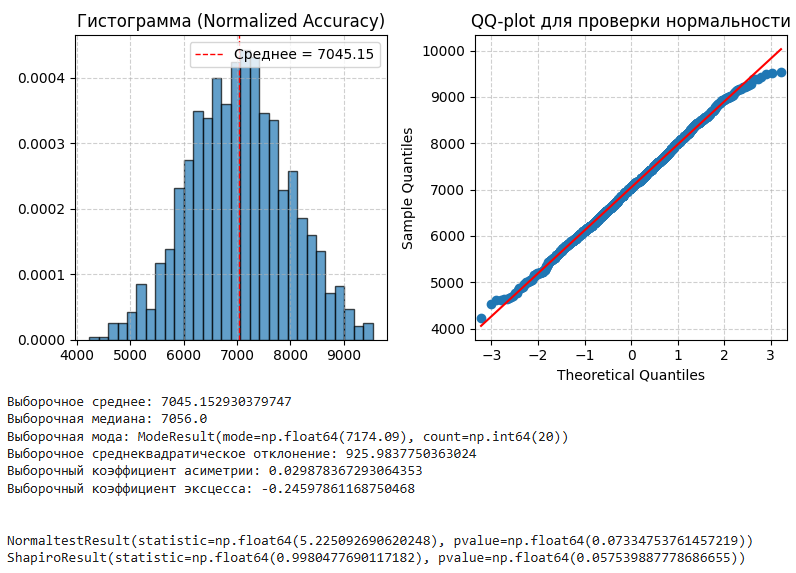


Рисунок 7. Проверка нормализованных данных о точностях

После преобразования распределение не отличалось от нормального (p-value = 0.858, асимметрия = 0.029). Метрику можно трактовать как **"**интенсивность точности", где: Разница между 80% и 90% (6400→8100) значимее, чем между 90% и 100% (8100→10000), при этом метрику все еще корректно использовать для анализа мастерства игрока.

Для статистики игрока за турнир проверка аналогично нормализованных значений точности и потерь сантипешек дала положительный результаты, изображенные на рисунках 8, 9. Стоит понимать, что проверка эксцесса для маленьких выборок можен дать неточный результат p-value, но QQ-plot и критерий Шапиро также подтверждают нормальность данных.

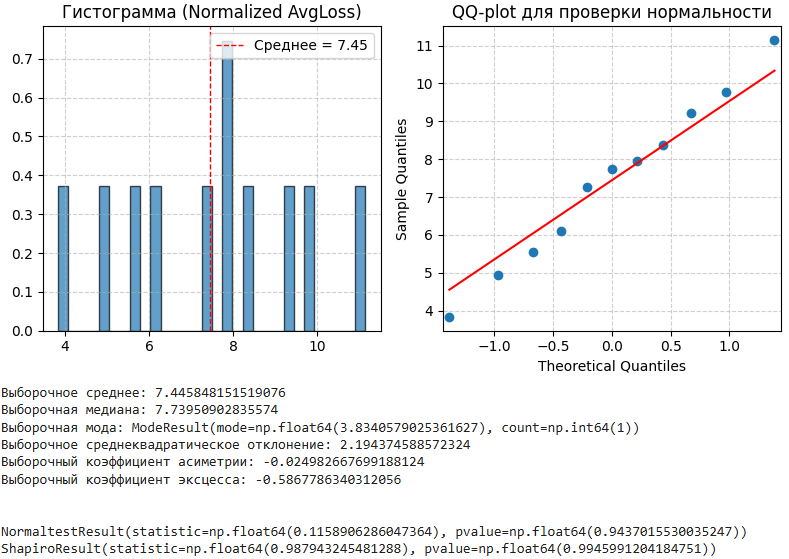


Рис.8. Проверка нормализованных данных о потерях сантипешек в турнире

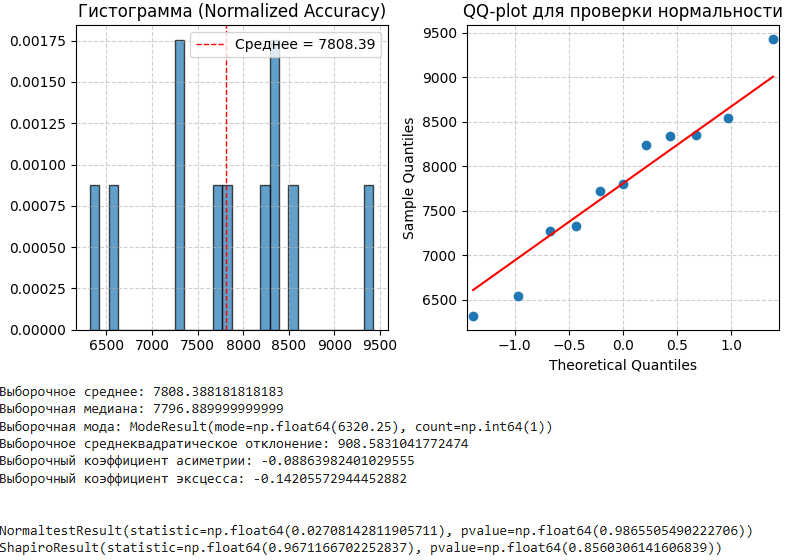


Рис.9 Проверка нормализованных данных о точности в турнире

Как было видно на рис.3, распределения числа зевков, ошибок и неточностей нормальным не является. Эти параметры могут принимать только целые и положительные значения, а зависят они в первую очередь от числа ходов в партии (так как чем больше ходов, тем больше возможностей сделать плохие ходы). Кроме того, из-за крайне ограниченной глубины анализа партии движком, значения числа неточностей и ошибок крайне неточны. Например, ход, при котором на данной глубине анализа было потеряно 70-80 сантипешек (что классифицируется как неточность), на большей глубине может быть учтен как потеря < 50 или > 100 сантипешек (что перестает классифицировать его как неточность или начнет классифицировать как ошибку). Поэтому в дальнейшем анализе число ошибок и неточностей учитываться не будут.

Осталось 3 величины, 2 из которых распределены нормально – корень от потерь сантипешек и квадрат точности. Для того, чтобы выбрать критерий, по которому будет проверяться гипотеза о равенстве их средних значений сначала необходимо проверить равенство их дисперсий.

Для проверки равенства дисперсий нормально распределенных величин можно использовать критерий Фишера или критерий Бартлетта. В данном анализе очень сильно различаются размеры выборок – в общей статистике данные почти по 1600 партиям, а в данном турнире было сыграно всего 11 партий. В таких условиях предпочтительным критерием является критерий Бартлетта. В python этот критерий реализован в функции stats.bartlett().

В свою очередь, для данных по числу зевков, которые практически никогда не будут распределены нормально, разумно использовать критерий Левена с медианным значением. В python этот критерий можно проверить с помощтю функции stats.levene().

Для проверки равенства дисперсий в коде написана функция, которая выводит, равны ли диспресии каждого параметра и возвращает значения p-value для каждой из проверок равенства дисперсий

check\_dispersion\_equality(data1, data2)

Заключительным шагом проверки гепотезы о читерстве является проверка равенства средних значений. Если выборки являются нормальными, а их дисперсии равны, то для этой цели разумно использовать критерий Стьюдента. Если же дисперсии не равны, то для проверки равенства средних значений двух нормально распределенных выборок нужно воспользоваться критерием Уэлча. Оба эти критерия реализованы в функции

stats.ttest\_ind(equal\_var=True)

Где при equal\_var=True будет применен критерий Стьюдента, а при equal\_var=False – критерий Уэлча.

Для ненормально распределенных зевков можно использовать непараметрический критерий Манна-Уитти, реализованый в функции

stats.mannwhitneyu().

Для финальной стадии анализа была написана функция, которая сначала проверяет равенство дисперсий для всех метрик, затем проверяет гипотезу о равенстве средних одним из трех описанных выше методов.

detecting\_cheaters(data1, data2)

На выходе эта функция выводит, равны ли дисперсии всех трех метрик в обоих выборках и есть ли основания для опровержения гипотезы о равенстве средних. На рисунке 10 показан результат работы функции.

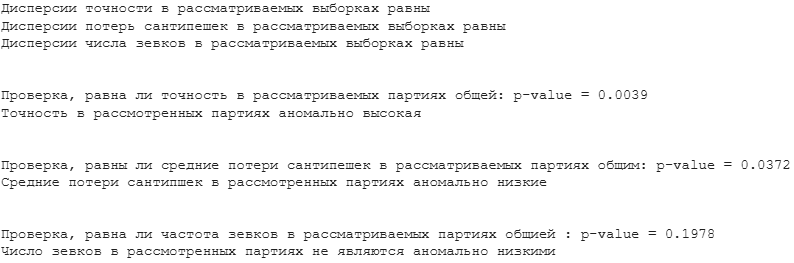


Рис.10 Результат проверки игрока на читерство

Как видно из рис.10, два главных параметра, определяющих вероятность читерства являются аномальными относительно общей статистики игрока. При этом стоит отметить, что для потерь сантипешек при статистической значимости 1% гипотезу о честности игрока отвергать было бы нельзя, так как p-value > 0.01. Частота зевков в турнире и вовсе не является аномалией и вполне соответствует общей статистике игрока.

Такую статистику можно считать крайне подозрительной, но не дающей стопроцентных оснований называть игрока нечестным.

Для подтверждения корректности выводов можно попробовать проверить ненормализованные данные. Для этого для всех трех параметров необходимо использовать непараметрический тест Манна-Уитти, проверяющий равенство медиан значений, которые, как видно из тестов на нормальность, обычно равны средним значениям. В коде эта проверка реализована в функции

detecting\_cheaters\_if\_distributions\_are\_not\_normal(data1, data2)

Эта функция не проверяет равенство дисперсий, в остальном работает аналогично detecting\_cheaters(data1, data2). Результат ее работы изображен на рисунке 11.

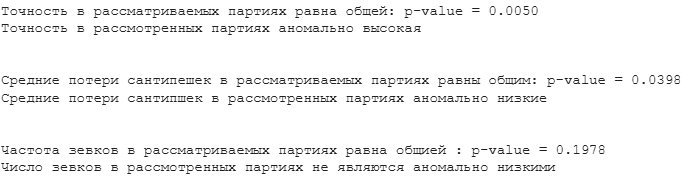


Рис.11 Анализ статистики с использованием непарамтрических критериев.

Результат этого анализа практически не отличается от анализа с помощью параметрических методов, из чего можно сделать вывод о корректности проведения статистического анализа.

**Заключение**

Исходя из анализа рассмотренных выше партий результаты игрока в рассмотренном турнире являются очень подозрительными. Значения точности игры и средних потерь сантипешек являются аномальными по сравнению с обычной игрой этого игрока.

Для точности значение p-value, равное 0.004 слишком низкое, чтобы называть это случайностью или везением. В свою очередь, значение p-value для потерь сантипешек, равное 0.037 ниже установленной статистической значимости в 0.05, но оно не принимает экстремально низких значений. Такие показатели могут свидетельствовать об “умном читерстве”, при котором игрок пользуется подсказками только в важных позициях и иногда делает плохие ходы, приводящие к значительным потерям сантипешек, но сильно не снижающих шансов на победу в партии.

В защиту игрока выступает частота зевков, статистически равная частоте зевков при его обычной игре.

Исходя из проведенного анализа можно сделать вывод, что скорее всего игрок действительно пользовался компьютерными подсказками на турнире. Для более точного анализа можно проверить среднее время, которое игрок тратил на ходы, а также проанализировать игру других игроков на предмет того, не играли ли они аномально слабо в этом турнире.