

Комплексный статистический анализ онлайн-шахматных партий на основе данных о рейтинге, контроле времени и дебютных схемах

Анишкова Анастасия Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Цель исследования – выявить, как рейтинг игроков, формат контроля времени и выбор дебюта совместно влияют на исход и характеристики онлайн-шахматных партий (длину, длительность и распределение результатов), на основе комплексного статистического и визуального анализа массивов игровых данных. Для реализации исследования в качестве программной среды выбран Google Colab, который обеспечивает доступ к вычислительным ресурсам в облаке, позволяет удобно работать с Python-ноутбуками, подключать необходимые библиотеки для анализа данных и визуализации, а также легко загружать и сохранять файлы с результатами обработки датасета. Этот подход обеспечивает воспроизводимость экспериментов, гибкость в настройке и модификации кода, а также наглядное представление промежуточных и конечных результатов анализа прямо в среде выполнения, что особенно важно при работе с большим количеством визуализаций и вычислительных экспериментов на реальных данных.

Ключевые слова: Google Colab, рейтинг игроков, анализ данных.

Comprehensive statistical analysis of online chess games based on rating, time control, and opening patterns

Anishkova Anastasia Sergeevna

Sholom-Aleichem Priamurskiy State University

Student

Abstract

The purpose of the study is to identify how the rating of players, the time control format, and the choice of opening jointly affect the outcome and characteristics of online chess games (length, duration, and distribution of results), based on a comprehensive statistical and visual analysis of game data arrays. To implement the study, Google Colab was chosen as the software environment, which provides access to cloud computing resources, allows for convenient work with Python notebooks, enables the connection of necessary libraries for data analysis and visualization, and facilitates easy uploading and saving of files with the results of processing the dataset. This approach ensures the reproducibility of experiments, flexibility in code customization and modification, and a clear representation of intermediate and final

analysis results directly in the execution environment, which is particularly important when working with a large number of visualizations and computational experiments on real data.

Keywords: Google Colab, player rating, and data analysis.

1 Введение

1.1 Актуальность

Актуальность исследования связана с быстрым ростом объёмов данных онлайн-шахмат, позволяющих объективно оценивать влияние рейтинга, дебюта и контроля времени на исход партии. Комплексный статистический анализ таких данных даёт практические выводы для игроков, тренеров и разработчиков шахматных платформ и демонстрирует применение современных методов анализа данных к реальному игровому процессу.

1.2 Обзор исследований

Э. Бисонг, Э. Бисонг продемонстрировали как осуществлять анализ данных при помощи библиотек Matplotlib и seaborn [1], Анализ болезни по болезни сердца при помощи библиотеки Seaborn проделал А. М. Черкашин [2], А. М. Васильченко пояснил как проводить анализ данных при помощи библиотек Python [3], Разведочный анализ данных в python (GoogleColab) описала А. С. Матвеева [4], М. С. Федотова, Н. Н. Барышева написал обзор информационных систем на языке python в государственном управлении [5].

1.3 Цель исследования

Цель исследования – выявить, как рейтинг игроков, формат контроля времени и выбор дебюта совместно влияют на исход и характеристики онлайн-шахматных партий (длину, длительность и распределение результатов), на основе комплексного статистического и визуального анализа массивов игровых данных.

2 Материалы и методы

Для достижения поставленной цели использовалась методология комплексного анализа данных, включающая предварительную обработку и обогащение исходного датасета за счёт вычисления новых признаков, проведение описательной статистики и визуальной аналитики (гистограммы, диаграммы, тепловые карты, scatter- и box-плоты), а также применение базовых статистических тестов и регрессионного моделирования для количественной оценки влияния рейтинга, дебюта и контроля времени на исход и характеристики партий.

Датасет представляет собой выборку онлайн-шахматных партий, в которой каждая строка соответствует одной завершённой игре. Он содержит около двадцати тысяч записей и шестнадцать базовых признаков, на основе которых возможно формирование дополнительных производных характеристик, таких как разница рейтингов между соперниками или

длительность партии в минутах. Ссылка на датасет:
<https://disk.yandex.ru/d/gj7UXu4STcIevQ>.

Основные поля:

- `id` - уникальный идентификатор партии.
 - `rated` - флаг, была ли партия рейтинговой (True/False).
 - `created_at`, `last_move_at` - временные метки начала и окончания партии в миллисекундах UNIX-формата.
 - `turns` - количество полуходов (ply), характеризующее длину партии.
 - `victory_status` - способ завершения партии (mate, resign, outoftime, draw).
 - `winner` - результат с точки зрения цвета (white, black, draw).
 - `increment_code` - контроль времени в формате «база+инкремент» (например, 10+0, 15+2).
 - `white_id`, `black_id` - логины игроков за белых и чёрных.
 - `white_rating`, `black_rating` - рейтинги игроков на момент партии.
 - `moves` - последовательность ходов в текстовом формате.
- `opening_eco`, `opening_name`, `opening_ply` - код ECO, название и глубина дебюта (количество полуходов до выхода из дебютной фазы).

3 Результаты

Анализ датасета начнём с описательной статистики, построения базовых распределений и визуализации исходов партий, а затем перейдём к более сложным связям между рейтинговыми показателями, контролем времени и используемыми дебютами.

В начале работы выполняется импорт основных библиотек для анализа данных и визуализации: `pandas` и `numpy` используются для обработки табличной информации, `matplotlib` и `seaborn` - для построения графиков, а модуль `scipy` - для проведения статистических тестов. Для удобства восприятия отключается вывод предупреждений и задаются единые параметры оформления графиков: выбирается стиль, цветовая палитра и базовый размер отображаемых фигур.

Затем в рабочее окружение загружается файл `games.csv`, содержащий сведения о шахматных партиях, и формируется объект `DataFrame`, с которым будет вестись дальнейшая работа. После этого на экран выводится заголовок анализа, указывается общее количество партий в выборке, а также временной интервал, который охватывают данные, полученный путём преобразования временных меток начала партий в удобочитаемый формат (рис.1).

Код:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
# Настройка стиля графиков
```

```
plt.style.use('seaborn-v0_8')
sns.set_palette("husl")
plt.rcParams['figure.figsize'] = (12, 8)
print("🔥 ПОЛНЫЙ АНАЛИЗ ШАХМАТСКИХ ПАРТИЙ 🔥")
print("=" * 60)
df = pd.read_csv('games.csv')
print(f"📊 Размер датасета: {df.shape[0]:,} партий")
print(f"🕒 Период анализа: {pd.to_datetime(df['created_at'],
unit='ms').min()} - {pd.to_datetime(df['created_at'], unit='ms').max()}")
print()
```

```
... 🔥 ПОЛНЫЙ АНАЛИЗ ШАХМАТСКИХ ПАРТИЙ 🔥
=====
📊 Размер датасета: 19,233 партий
🕒 Период анализа: 2013-08-17 20:33:53.173000 - 2017-09-04 01:41:20.676000
```

Рисунок 1. Вывод

На данном этапе выполняется подготовка исходных данных к анализу. Сначала временные метки начала и окончания партии переводятся из миллисекунд UNIX-формата в тип даты-времени, после чего рассчитывается длительность каждой партии в минутах как разность между моментом последнего хода и временем создания игры. Параллельно формируются новые числовые признаки: разница рейтингов между белыми и чёрными и их средний рейтинг, а также бинарные индикаторы исхода, показывающие, победили ли белые, чёрные или партия завершилась вничью.

Отдельный блок посвящён разбору контроля времени. Из строки `increment_code` регулярными выражениями извлекаются базовое время и значение инкремента, на основе которых для каждой партии задаются логические флаги: блиц (короткий контроль с небольшим инкрементом), рапид (средний контроль) и классический контроль времени. Затем выводятся первые строки с полученными признаками и подсчитывается, сколько партий относится к каждому типу контроля, что позволяет оценить структуру выборки по игровому формату (рис.2).

Код:

```
## 1. ПОДГОТОВКА ДАННЫХ
print("1. 🌀 ПОДГОТОВКА ДАННЫХ")
df['created_at'] = pd.to_datetime(df['created_at'], unit='ms')
df['last_move_at'] = pd.to_datetime(df['last_move_at'], unit='ms')
df['game_duration'] = (df['last_move_at'] - df['created_at']).dt.total_seconds() / 60 # минуты
df['rating_diff'] = df['white_rating'] - df['black_rating']
df['avg_rating'] = (df['white_rating'] + df['black_rating']) / 2
df['white_win'] = (df['winner'] == 'white').astype(int)
df['black_win'] = (df['winner'] == 'black').astype(int)
df['draw'] = (df['winner'] == 'draw').astype(int)
# Парсинг времени контроля (ИСПРАВЛЕНО)
df['time_control'] =
df['increment_code'].str.extract(r'(\d+)').astype(int)
df['increment'] =
df['increment_code'].str.extract(r'\+(\d+)').astype(int).fillna(0)
```

```

df['is_blitz'] = (df['time_control'] <= 10) & (df['increment'] <= 5)
df['is_rapid'] = (df['time_control'] > 10) & (df['time_control'] <=
30) # ☒ Замкнута скобка!
df['is_classic'] = df['time_control'] > 30
print(☒ Парсинг времени контроля:")
print(df[['increment_code', 'time_control', 'increment', 'is_blitz',
'is_rapid', 'is_classic']].head(10))
print("\nТипы контроля:")
print(df[['is_blitz', 'is_rapid', 'is_classic']].sum())

```

```

... ☒ Парсинг времени контроля:
      increment_code  time_control  increment  is_blitz  is_rapid  is_classic
0             15+2             15           2     False     True     False
1             5+10              5          10     False     False     False
2             5+10              5          10     False     False     False
3             20+0             20           0     False     True     False
4             30+3             30           3     False     True     False
5             10+0             10           0      True     False     False
6             10+0             10           0      True     False     False
7            15+30             15          30     False     True     False
8             15+0             15           0     False     True     False
9             10+0             10           0      True     False     False

```

Типы контроля:

is_blitz 10484

is_rapid 5730

is_classic 484

dtype: int64

Рисунок 2. Парсинг времени контроля

С помощью метода describe() для количества полуходов, рейтингов белых и чёрных, длительности партии и разницы рейтингов формируются базовые показатели: число наблюдений, среднее значение, стандартное отклонение, минимумы и максимумы, а также квартильные значения. Полученные результаты округляются до двух знаков после запятой, что позволяет быстро оценить типичный диапазон и разброс параметров в выборке шахматных партий (рис.3).

```

[9] ☒ 0 сек. ☒ ## 2. ОБЩАЯ СТАТИСТИКА
print("\n2. ☒ ОСНОВНАЯ СТАТИСТИКА")
print(df[['turns', 'white_rating', 'black_rating', 'game_duration', 'rating_diff']].describe().round(2))

```

	turns	white_rating	black_rating	game_duration	rating_diff
count	19233.00	19233.00	19233.00	19233.00	19233.00
mean	60.28	1593.55	1585.17	14.19	8.38
std	33.57	290.84	290.10	81.74	249.28
min	1.00	784.00	789.00	0.00	-1605.00
25%	37.00	1394.00	1388.00	0.00	-108.00
50%	55.00	1562.00	1556.00	3.19	4.00
75%	79.00	1787.00	1777.00	12.82	123.00
max	349.00	2700.00	2723.00	10097.41	1499.00

Рисунок 3. Общая статистика

В следующем блоке проводится развернутый анализ исходов партий с помощью шести взаимодополняющих визуализаций. Для начала задаются единый «сетчатый» стиль оформления графиков и увеличенные размеры шрифтов заголовков и подписей осей, после чего создаётся фигура с сеткой 2×3, в ячейках которой последовательно строятся графики.

Первая диаграмма отображает распределение способов завершения партии (мат, сдача, проигрыш по времени, ничья) в виде круговой диаграммы с цветовой дифференциацией, а вторая - столбчатое распределение победителей по цвету, показывающее, как часто выигрывают белые, чёрные и какова доля ничьих. Третий график представляет собой boxplot длины партии в полуходах для разных типов исходов, что позволяет сравнить типичную продолжительность партий, завершившихся матом, сдачей или другими способами.

Во второй строке визуализаций анализируется связь результата с рейтингом и длиной партии. Сначала средний рейтинг игроков разбивается на интервалы, для которых рассчитывается и отображается доля ничьих, что даёт представление о том, в каких диапазонах силы соперников мирный исход встречается чаще. Затем строятся две полупрозрачные гистограммы длины партий отдельно для партий, завершившихся матом, и для партий, где соперник сдался, что наглядно показывает различия в их типичной продолжительности. Наконец, линейный график динамики числа партий по месяцам демонстрирует, как менялась активность игр во времени в анализируемом периоде (рис.4).

Код:

```
## 3. 🌀 АНАЛИЗ ИСХОДОВ
```

```
print("\n3. 🌀 АНАЛИЗ ИСХОДОВ")
```

```
sns.set_style("whitegrid")
plt.rcParams['axes.titlesize'] = 13
plt.rcParams['axes.labelsize'] = 11
```

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))
```

```
# 1) Распределение victory_status (pie + красивые цвета)
victory_counts = df['victory_status'].value_counts()
colors_vs      = ['#66c2a5',          '#fc8d62',          '#8da0cb',
'#ffd92f'][:len(victory_counts)]
```

```
axes[0, 0].pie(
    victory_counts.values,
    labels=victory_counts.index,
    autopct='%1.1f%%',
    startangle=90,
    colors=colors_vs,
    textprops={'fontsize': 10}
)
axes[0, 0].set_title('Распределение исходов партий')
```

```
# 2) Победитель по цвету (bar)
```

```
winner_counts = df['winner'].value_counts()
axes[0, 1].bar(
    winner_counts.index,
    winner_counts.values,
    color=['#ffffff', '#000000', '#b3b3b3'],
    edgecolor='black'
)
axes[0, 1].set_title('Победитель по цвету')
axes[0, 1].set_ylabel('Количество партий')

# 3) Длина партий по исходу (boxplot)
sns.boxplot(
    data=df,
    x='victory_status',
    y='turns',
    ax=axes[0, 2],
    palette='Set2'
)
axes[0, 2].set_title('Длина партий по исходу')
axes[0, 2].set_xlabel('Исход')
axes[0, 2].set_ylabel('Количество ходов')

# 4) Доля ничьих по среднему рейтингу (bar + контур)
df['avg_rating_bin'] = pd.cut(df['avg_rating'], bins=10)
draw_by_rating = df.groupby('avg_rating_bin')['draw'].mean() * 100

draw_by_rating.plot(
    kind='bar',
    ax=axes[1, 0],
    color='#80b1d3',
    edgecolor='black'
)
axes[1, 0].set_title('Доля ничьих по среднему рейтингу')
axes[1, 0].set_xlabel('Интервалы рейтинга')
axes[1, 0].set_ylabel('Ничьи, %')
axes[1, 0].tick_params(axis='x', rotation=45)

# 5) Распределение длины партий (mat vs resign, полупрозрачные
гистограммы)
mate_turns = df[df['victory_status'] == 'mate']['turns']
resign_turns = df[df['victory_status'] == 'resign']['turns']

bins = range(0, int(df['turns'].max()) + 5, 5)

axes[1, 1].hist(
    mate_turns,
    bins=bins,
    alpha=0.6,
    label='Мат',
    color='#1b9e77',
    edgecolor='black'
)
axes[1, 1].hist(
    resign_turns,
    bins=bins,
    alpha=0.6,
    label='Резигнация',
    color='#d95f02',
    edgecolor='black'
)
axes[1, 1].legend(frameon=True)
```

```

axes[1, 1].set_title('Длина партии: мат vs resign')
axes[1, 1].set_xlabel('Количество ходов')
axes[1, 1].set_ylabel('Число партий')

# 6) Динамика количества партий по месяцам (line + маркеры)
df['year_month'] = df['created_at'].dt.to_period('M')
monthly_games = df.groupby('year_month').size()
monthly_games.index = monthly_games.index.astype(str)

axes[1, 2].plot(
    monthly_games.index,
    monthly_games.values,
    marker='o',
    linestyle='-',
    color='#7570b3'
)

axes[1, 2].set_title('Динамика числа партий по месяцам')
axes[1, 2].set_xlabel('Месяц')
axes[1, 2].set_ylabel('Количество партий')
axes[1, 2].tick_params(axis='x', rotation=60)

plt.tight_layout()
plt.show()

```

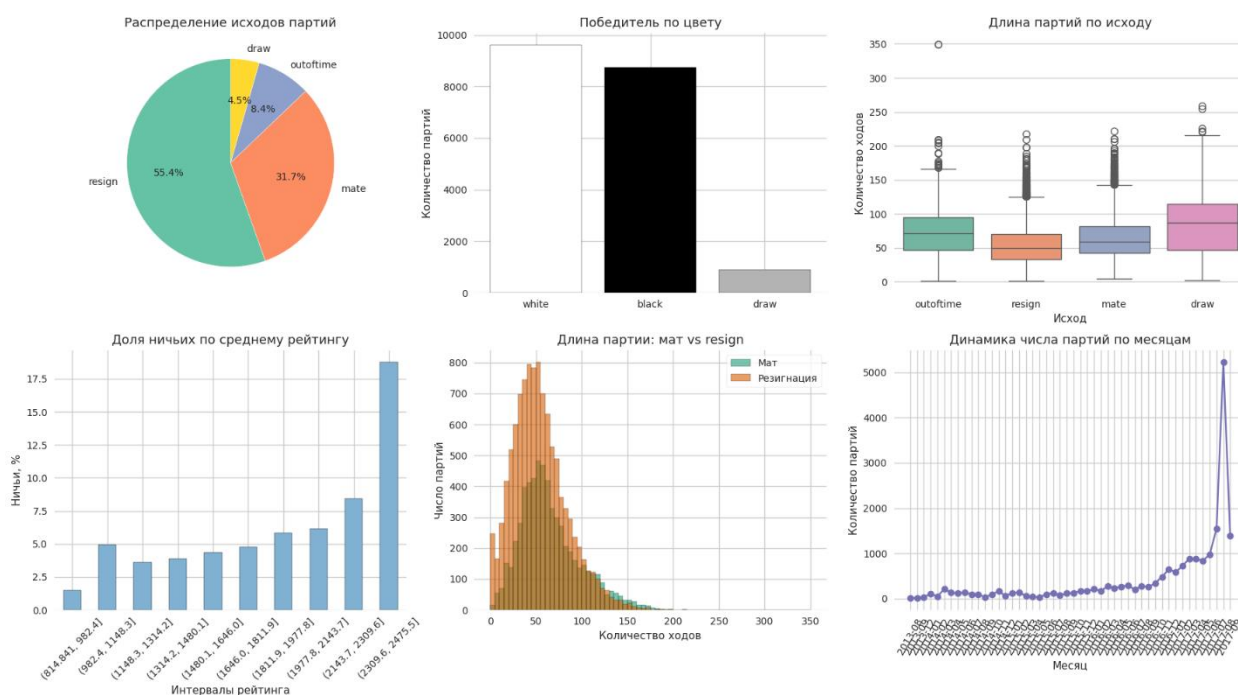


Рисунок 4. Анализ исходов

В данном разделе проводится комплексный анализ рейтинговых характеристик игроков и их связи с исходом партий на основе набора визуализаций, организованных в формате сетки 2×3.

В верхнем блоке сначала строится наложенная гистограмма, отражающая распределения рейтингов белых и чёрных; она позволяет оценить типичный диапазон силы участников и убедиться в их сопоставимости. Затем разница рейтингов между соперниками разбивается на интервалы, для которых вычисляется доля побед белых, что даёт представление о том, как

меняется вероятность выигрыша при различных значениях рейтингового перевеса. Дополняет этот блок тепловая карта корреляций между ключевыми числовыми показателями: рейтингами, длиной партии, её длительностью и исходом, что помогает выявить наиболее значимые взаимосвязи.

Нижний блок графиков фокусируется на влиянии рейтинга и формата игры на результат. Столбчатая диаграмма сравнивает доли побед белых, чёрных и ничьих в рейтинговых и нерейтинговых партиях, позволяя оценить выраженность преимущества первого хода в разных режимах. Логистическая регрессия визуализирует зависимость вероятности победы белыми от среднего рейтинга участников, а сглаженные плотности распределения разницы рейтингов по исходу показывают, при каких значениях ELO-перевеса чаще фиксируются победы белых и в каких случаях исход оказывается в их пользу реже(рис.5).

Код:

```
## 4. 🏰 АНАЛИЗ РЕЙТИНГОВ
print("\n4. 🏰 АНАЛИЗ РЕЙТИНГОВ")

fig, axes = plt.subplots(2, 3, figsize=(18, 10))

# 4.1 Распределение рейтингов
axes[0, 0].hist(df['white_rating'], bins=50, alpha=0.5,
label='Белые', color='lightgray', edgecolor='black')
axes[0, 0].hist(df['black_rating'], bins=50, alpha=0.5,
label='Чёрные', color='black')
axes[0, 0].legend()
axes[0, 0].set_title('Распределение рейтингов')
axes[0, 0].set_xlabel('Рейтинг')
axes[0, 0].set_ylabel('Частота')

# 4.2 Winrate по разнице рейтингов
df['rating_bin'] = pd.cut(df['rating_diff'], bins=20)
winrate_by_diff = df.groupby('rating_bin')['white_win'].agg(['mean',
'count']).round(3)
winrate_by_diff['mean'].plot(ax=axes[0, 1], kind='line', marker='o')
axes[0, 1].set_title('Winrate белых по разнице рейтингов')
axes[0, 1].set_ylabel('Доля побед белых')
axes[0, 1].tick_params(axis='x', rotation=90)

# 4.3 Корреляционная матрица
corr_matrix = df[['white_rating', 'black_rating', 'turns',
'white_win', 'game_duration']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0,
ax=axes[0, 2])
axes[0, 2].set_title('Корреляции')

# 4.4 Преимущество первого хода (рейтинговые vs нерейтинговые)
first_move_adv = df.groupby('rated').agg({
'white_win': 'mean',
'black_win': 'mean',
'draw': 'mean'
}).round(3) * 100
first_move_adv.plot(ax=axes[1, 0], kind='bar')
axes[1, 0].set_title('Преимущество белых: рейтинговые vs
нерейтинговые (%)')
```

```

axes[1, 0].set_ylabel('%')
axes[1, 0].legend(['Белые', 'Чёрные', 'Ничья'])

# 4.5 Регрессия: рейтинг vs успех
sns.regplot(data=df, x='avg_rating', y='white_win', ax=axes[1, 1],
            logistic=True, scatter_kws={'alpha':0.2})
axes[1, 1].set_title('Логистическая регрессия: рейтинг vs победа')
axes[1, 1].set_ylabel('Вероятность победы белых')

# 4.6 ELO разница и вероятность победы
sns.kdeplot(data=df, x='rating_diff', hue='white_win', fill=True,
            ax=axes[1, 2])
axes[1, 2].set_title('Распределение разницы ELO по исходу')
axes[1, 2].set_xlabel('Разница рейтингов (Белые - Чёрные)')

plt.tight_layout()
plt.show()

```

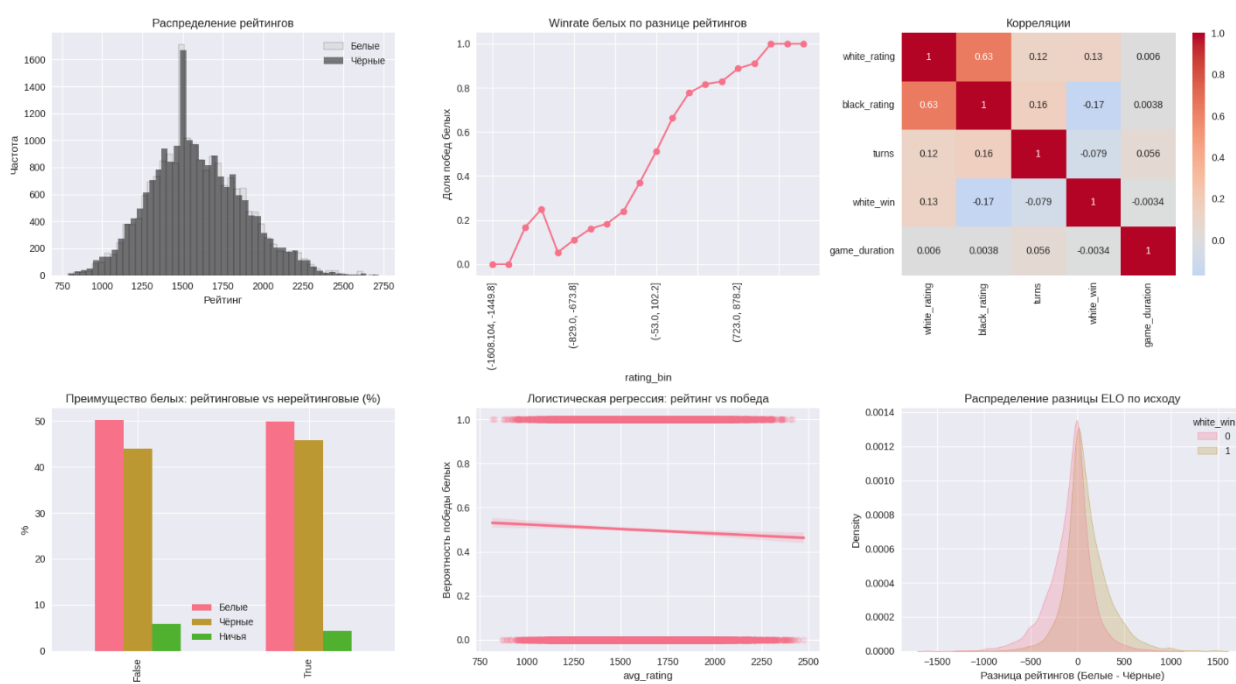


Рисунок 5. Анализ рейтингов

Далее исследуются использованные в партиях дебюты, их популярность и результативность, а также связь глубины дебютной подготовки с длиной партии.

Сначала определяется двадцать самых часто встречающихся дебютов по их названию, и по ним строится горизонтальная столбчатая диаграмма, показывающая, какие начала встречаются в выборке чаще всего. Затем для каждого из этих дебютов рассчитывается доля побед белых, медианная длина партии и число сыгранных партий; на основе этих показателей формируется график winrate белых в пределах топ-20 по популярности, что позволяет выделить дебюты, одновременно распространённые и статистически успешные. Дополнительно анализируются коды ЕСО: выводится пятнадцать наиболее частотных кодов и их распределение, что даёт более формализованное представление о структуре дебютного репертуара (рис.6).

В заключение оценивается взаимосвязь между «сложностью» дебюта и продолжительностью партии: по каждому значению глубины дебюта (числу полуходов до выхода из дебютной фазы) вычисляется медианное количество ходов партии, после чего строится диаграмма рассеяния. Она позволяет визуально сопоставить, приводят ли более глубокие дебютные варианты к статистически более длинным или, наоборот, более коротким партиям (рис.7).

Код:

```
## 5. 📌 АНАЛИЗ ДЕБЮТОВ
print("\n5. 📌 АНАЛИЗ ДЕБЮТОВ")

top_20_openings = df['opening_name'].value_counts().head(20)

fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# 5.1 Топ-20 дебютов по частоте
top_20_openings.plot(ax=axes[0, 0], kind='barh')
axes[0, 0].set_title('Топ-20 дебютов по популярности')
axes[0, 0].set_xlabel('Количество партий')

# 5.2 Winrate топ дебютов (с учётом частоты)
opening_stats = (
    df.groupby('opening_name')
    .agg(
        white_win=('white_win', 'mean'),
        turns=('turns', 'median'),
        games=('opening_name', 'size')
    )
    .round(3)
)

top_opening_winrate = (
    opening_stats
    .loc[top_20_openings.index]
    .sort_values('white_win', ascending=False)
)

top_opening_winrate['white_win'].plot(
    ax=axes[0, 1],
    kind='barh',
    color='green'
)
axes[0, 1].set_title('Winrate белых в топ-20 дебютах')
axes[0, 1].set_xlabel('Доля побед белых')

# 5.3 Дебюты по ECO кодам
eco_stats = df['opening_eco'].value_counts().head(15)
eco_stats.plot(ax=axes[1, 0], kind='barh', color='purple')
axes[1, 0].set_title('Топ-15 ECO кодов дебютов')
axes[1, 0].set_xlabel('Количество партий')

# 5.4 Корреляция дебютной сложности и длины партии
opening_complexity = (
    df.groupby('opening_ply')['turns']
    .median()
    .reset_index()
)
```

```

opening_complexity.columns = ['opening_ply', 'median_turns']

opening_complexity.plot(
    ax=axes[1, 1],
    kind='scatter',
    x='opening_ply',
    y='median_turns',
    color='orange'
)

axes[1, 1].set_xlabel('Глубина дебюта (пли)')
axes[1, 1].set_ylabel('Медианная длина партии')
axes[1, 1].set_title('Сложность дебюта vs длина партии')

plt.tight_layout()
plt.show()

print("🏆 Топ-10 дебютов по winrate белых (из топ-20 по популярности):")
print(top_opening_winrate.head(10)[['white_win', 'turns', 'games']])

```

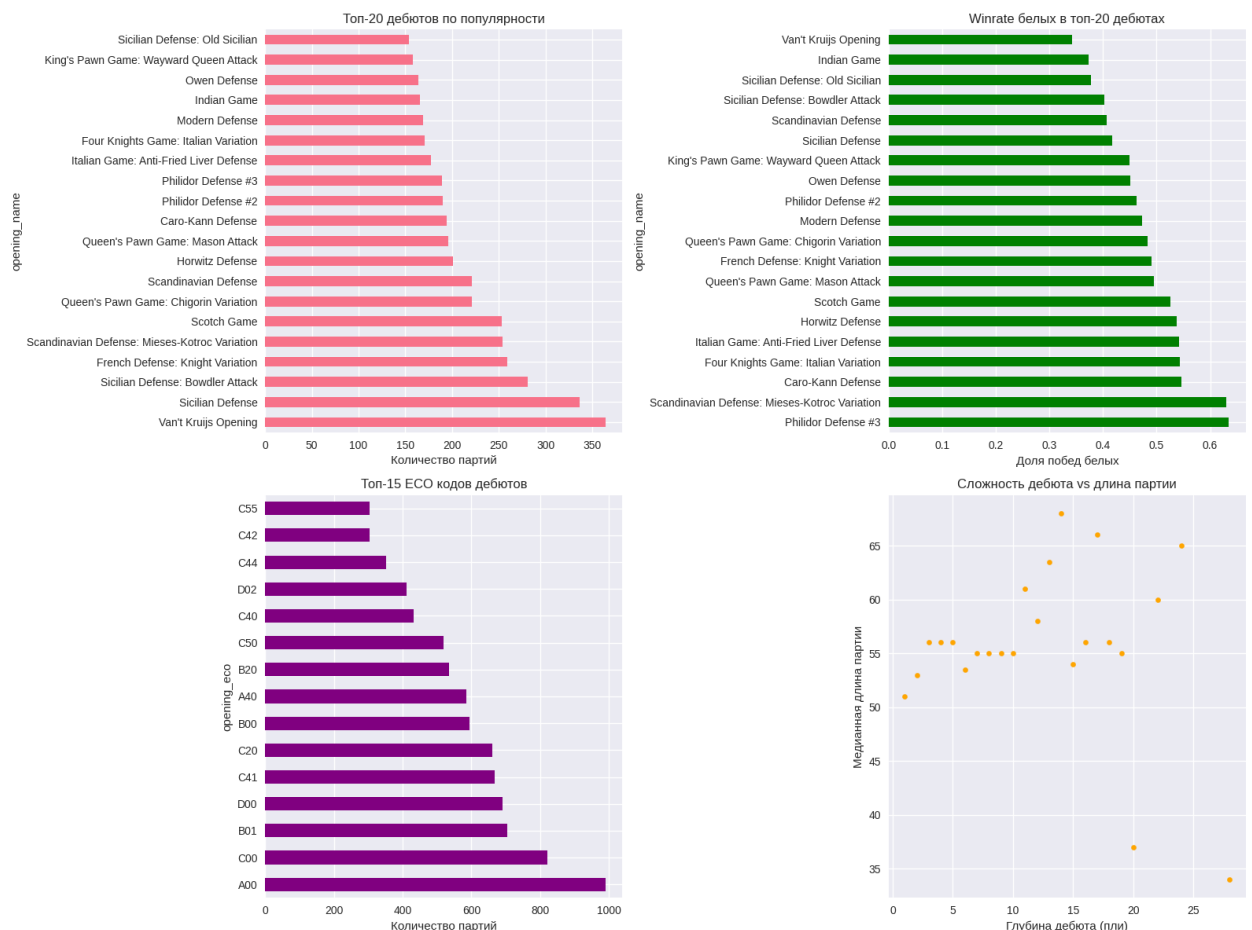


Рисунок 6. Анализ дебютов

🏆 ТОП-10 дебютов по winrate белых (из топ-20 по популярности):

	white_win	turns	games
opening_name			
Philidor Defense #3	0.635	47.0	189
Scandinavian Defense: Mieses-Kotroc Variation	0.630	55.5	254
Caro-Kann Defense	0.546	60.0	194
Four Knights Game: Italian Variation	0.544	56.0	171
Italian Game: Anti-Fried Liver Defense	0.542	51.0	177
Horwitz Defense	0.537	54.0	201
Scotch Game	0.526	55.0	253
Queen's Pawn Game: Mason Attack	0.495	62.5	196
French Defense: Knight Variation	0.490	55.0	259
Queen's Pawn Game: Chigorin Variation	0.484	54.0	221

Рисунок 7. Вывод

В данном разделе анализируется влияние контроля времени на исход и продолжительность партий.

Сначала формируется агрегированная статистика по сочетаниям базового времени и инкремента: для каждого варианта контроля рассчитываются средняя доля побед белых, медианное количество ходов и число сыгранных партий, что позволяет увидеть, какие режимы используются чаще и насколько они результативны. Затем выборка делится на три группы по типу контроля - блиц, рапид и классические партии - на основе ранее сформированных логических признаков.

Графическая часть раздела включает две наглядные диаграммы. Первая показывает долю побед белых в каждом типе контроля времени, дополняя столбцы числовыми подписями и позволяя напрямую сравнить эффективность игры белыми в блице, рапиде и классике. Вторая диаграмма отображает медианную длину партии (в ходах) для этих же групп, что даёт возможность оценить, насколько формат контроля времени влияет на типичную продолжительность шахматных поединков (рис.8).

Код:

```
## 6. 🕒 АНАЛИЗ КОНТРОЛЯ ВРЕМЕНИ
print("\n6. 🕒 АНАЛИЗ КОНТРОЛЯ ВРЕМЕНИ")

sns.set_style("whitegrid")

# агрегированная статистика по (time_control, increment)
time_control_stats = (
    df.groupby(['time_control', 'increment'])
      .agg(
        white_win=('white_win', 'mean'),
        turns=('turns', 'median'),
        games=('time_control', 'size')
      )
      .round(3)
)

print("\nСтатистика по типам контроля времени:")
print(time_control_stats.head(10))
```

```
# подготовка групп
blitz_games = df[df['is_blitz']]
rapid_games = df[df['is_rapid']]
classic_games = df[~df['is_blitz'] & ~df['is_rapid']]

fig, axes = plt.subplots(1, 2, figsize=(15, 6))

# 6.1 Winrate белых по типу контроля времени
winrates = {
    'Блиц': blitz_games['white_win'].mean(),
    'Рапид': rapid_games['white_win'].mean(),
    'Классика': classic_games['white_win'].mean()
}

axes[0].bar(
    winrates.keys(),
    winrates.values(),
    color=['#e41a1c', '#ff7f00', '#4daf4a'],
    edgecolor='black'
)
axes[0].set_ylim(0, 1)
axes[0].set_title('Winrate белых по типу контроля времени')
axes[0].set_ylabel('Доля побед белых')
for i, v in enumerate(winrates.values()):
    axes[0].text(i, v + 0.02, f"{v:.2f}", ha='center', fontsize=10)

# 6.2 Медианная длина партии по типу контроля (bar + подписи)
time_data = pd.DataFrame({
    'Тип контроля': ['Блиц', 'Рапид', 'Классика'],
    'Медианная длина': [
        blitz_games['turns'].median(),
        rapid_games['turns'].median(),
        classic_games['turns'].median()
    ]
})

axes[1].bar(
    time_data['Тип контроля'],
    time_data['Медианная длина'],
    color=['#e41a1c', '#ff7f00', '#4daf4a'],
    edgecolor='black'
)
axes[1].set_title('Медианная длина партии по типу контроля')
axes[1].set_ylabel('Количество ходов')
for i, v in enumerate(time_data['Медианная длина']):
    axes[1].text(i, v + 1, f"{v:.0f}", ha='center', fontsize=10)

plt.tight_layout()
plt.show()
```

6. ○ АНАЛИЗ КОНТРОЛЯ ВРЕМЕНИ

Статистика по типам контроля времени:

	time_control	increment	white_win	turns	games
0	12	0.40	71.0	5	
	13	1.00	45.0	1	
	15	0.75	42.5	8	
	16	0.20	2.0	15	
	20	1.00	59.0	1	
	25	0.50	41.5	4	
	30	0.00	100.0	1	
	40	1.00	53.0	1	
	180	0.50	22.5	6	
1	10	0.75	82.0	4	

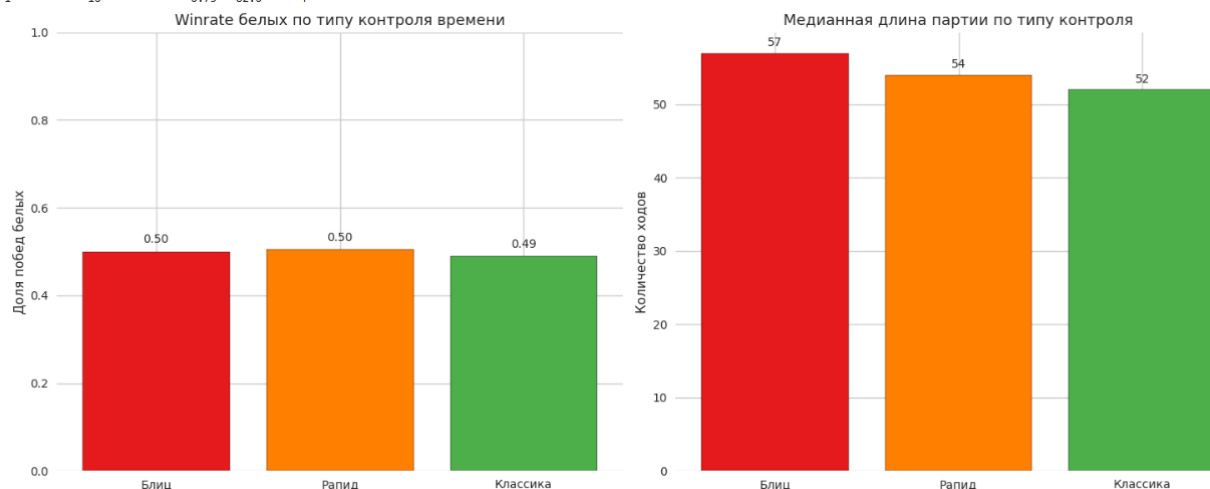


Рисунок 8. Анализ контроля времени

Далее выполняется статистическая оценка связи между форматом партии, характеристиками игроков и вероятностью победы белыми. Сначала данные делятся на рейтинговые и нерейтинговые партии, после чего с помощью t-теста Стьюдента сравниваются средние доли побед белых в этих двух подвыборках; вывод значений статистики t и p-уровня значимости позволяет судить, есть ли статистически значимое различие между форматами.

Далее строится простая линейная модель, рассматривающая вероятность победы белых как функцию разницы рейтингов, длины партии, среднего рейтинга и глубины дебюта. Для этого отбираются строки без пропусков, обучается модель линейной регрессии и выводятся её коэффициенты и значение R^2 , характеризующее долю объяснённой вариации результата. Наконец, визуализации показывают, во-первых, сравнение доли побед белых в рейтинговых и нерейтинговых партиях в виде столбчатой диаграммы, а во-вторых, относительную «важность» использованных признаков по модулю их регрессионных коэффициентов, что даёт интуитивное представление о том, какие факторы сильнее всего связаны с исходом партии (рис.9).

Код:

```
## 7. 📊 СТАТИСТИЧЕСКИЙ АНАЛИЗ
print("\n7. 📊 СТАТИСТИЧЕСКИЙ АНАЛИЗ")
print("=" * 40)

from sklearn.linear_model import LinearRegression

# 7.1 Т-тест побед белых: рейтинговые vs нерейтинговые
white_wins Rated = df[df['rated']] ['white_win']
white_wins Casual = df[~df['rated']] ['white_win']
```

```

t_stat, p_value = stats.ttest_ind(white_wins Rated,
white_wins Casual)
print(f"🎲 Т-тест (рейтинговые vs нерейтинговые): t={t_stat:.3f},
p={p_value:.3f}")

# 7.2 Линейная регрессия (как аппроксимация вероятности победы)
features = ['rating_diff', 'turns', 'avg_rating', 'opening_ply']
X = df[features].copy()
y = df['white_win'].astype(float)

mask = X.notna().all(axis=1) & y.notna()
X_clean = X[mask]
y_clean = y[mask]

model = LinearRegression().fit(X_clean, y_clean)

print(f"📊 R² модели предсказания победы: {model.score(X_clean,
y_clean):.3f}")
print("📋 Коэффициенты:")
for name, coef in zip(features, model.coef_):
    print(f" {name}: {coef:.5f}")
print(f"Свободный член (intercept): {model.intercept_:.5f}")

# ————— ВИЗУАЛИЗАЦИИ —————

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# 7.A Сравнение доли побед белых (рейтинговые / нерейтинговые)
rated_winrate = white_wins Rated.mean()
casual_winrate = white_wins Casual.mean()

axes[0].bar(['Рейтинговые', 'Нерейтинговые'],
            [rated_winrate, casual_winrate],
            color=['steelblue', 'orange'])
axes[0].set_ylim(0, 1)
axes[0].set_ylabel('Доля побед белых')
axes[0].set_title('Победы белых: рейтинговые vs нерейтинговые')

# 7.B Важность признаков по модулю коэффициентов
coef_abs = pd.Series(model.coef_, index=features).abs().sort_values()
coef_abs.plot(kind='barh', ax=axes[1], color='green')
axes[1].set_title('Вклад признаков в модель (|коэффициент|)')
axes[1].set_xlabel('Абсолютное значение коэффициента')

plt.tight_layout()
plt.show()

```

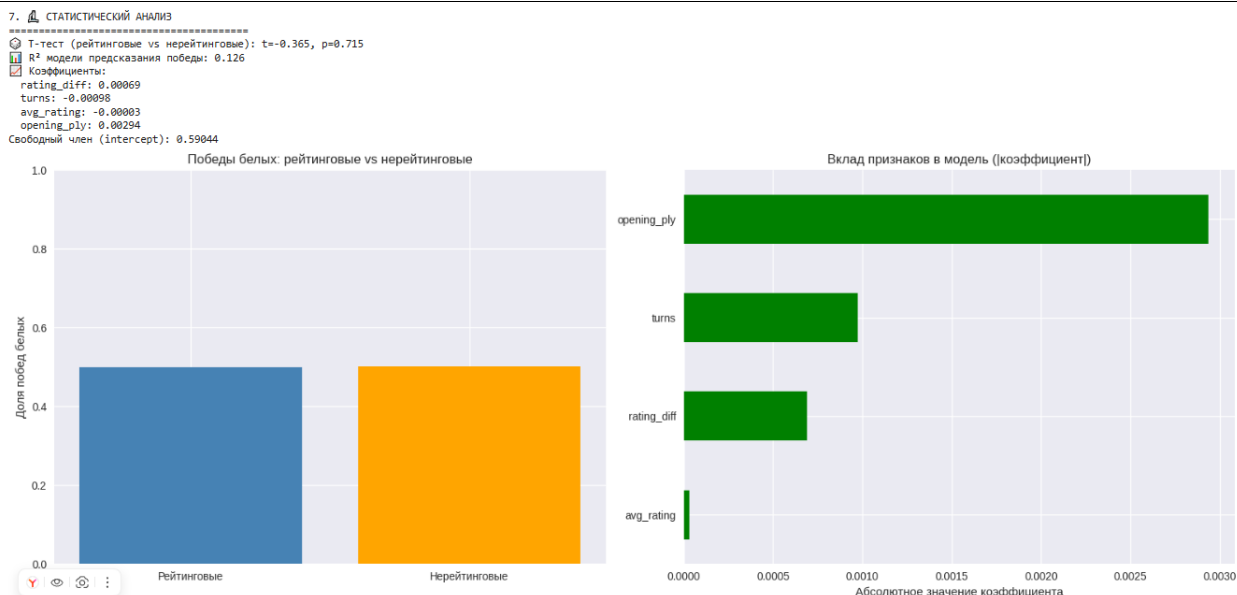


Рисунок 9. Статистический анализ

В данном разделе рассматривается активность отдельных пользователей в роли игроков за белых. Сначала по каждому `white_id` вычисляются три показателя: средняя доля побед белыми, медианная длина партий и общее число сыгранных партий, после чего формируется агрегированная таблица статистики по игрокам.

Затем из этой таблицы отбираются десять игроков с наибольшим количеством партий за белых, их показатели выводятся в текстовом виде, а также визуализируются с помощью горизонтальной столбчатой диаграммы. Такой график наглядно показывает, кто из пользователей является самым активным в выборке, и позволяет легко сравнить объём их игровой практики между собой (рис.10).

Код:

```
## 8. 🏆 ТОП ИГРОКИ
print("\n8. 🏆 ТОП ИГРОКИ")

player_stats = (
    df.groupby('white_id')
    .agg(
        white_win=('white_win', 'mean'),
        turns=('turns', 'median'),
        games=('white_id', 'size')
    )
    .round(3)
)

top_players = player_stats.nlargest(10, 'games')
print("Топ-10 игроков по количеству партий (белыми):")
print(top_players)

# ————— ВИЗУАЛИЗАЦИЯ —————
plt.figure(figsize=(10, 6))

# Бар-график по количеству партий у топ-10
```

```

top_players['games'].sort_values().plot(kind='barh',
color='skyblue')
plt.title('Топ-10 игроков по числу партий (белыми)')
plt.xlabel('Количество партий')
plt.ylabel('Игрок')

plt.tight_layout()
plt.show()

```

8. 🏆 ТОП ИГРОКИ

Топ-10 игроков по количеству партий (белыми):

white_id	white_win	turns	games
taranga	0.472	47.0	72
chess-brahs	0.340	52.0	53
a_p_t_e_m_u_u	0.510	33.0	49
bleda	0.250	85.5	48
ssf7	0.604	75.0	48
hassan1365416	0.636	57.0	44
khelil	0.488	60.0	41
1240100948	0.579	67.0	38
anakgreget	0.474	57.0	38
ozguragarr	0.553	48.0	38

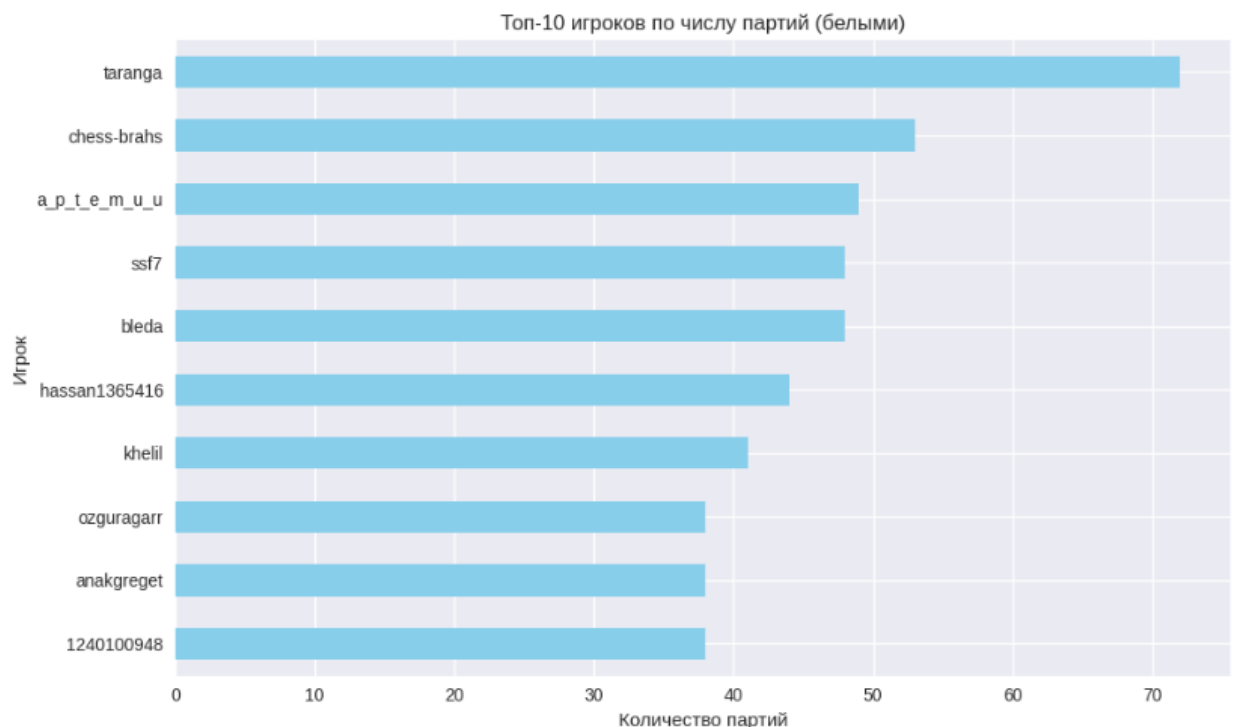


Рисунок 10. Топ игроки

В завершающем разделе используются более сложные и наглядные визуализации, позволяющие исследовать многомерные зависимости в данных.

Сначала строится тепловая карта, где по оси времени контроля и по оси среднего рейтинга отложены интервальные группы, а цветом отображается доля побед белых; это даёт компактное представление о том, в каких сочетаниях контроля времени и уровня игры белые достигают наибольшего успеха. Затем violin-диаграмма показывает распределение среднего рейтинга игроков для партий с разными исходами (победа белых, чёрных, ничья), позволяя визуально сравнить характер распределений и наличие возможных сдвигов. Дополнительные графики включают boxplot длины партий по типам контроля времени и scatter-диаграмму зависимости длительности партии от

среднего рейтинга, что помогает выявить тенденции в динамике партии в зависимости от силы соперников и формата времени (рис.11).

Отдельно строится парплот для подвыборки партий с десятью наиболее популярными дебютами по кодам ЕСО: он отображает попарные зависимости между количеством ходов, рейтингами соперников и фактом победы белыми, раскрашивая точки по дебютным кодам. Такая визуализация даёт целостное представление о том, как различные дебюты распределены в пространстве ключевых характеристик партии и как они связаны с исходами игр (рис.12).

Код:

```
## 9. ИНТЕРАКТИВНЫЕ ВИЗУАЛИЗАЦИИ
print("\n9.  ИНТЕРАКТИВНЫЕ ГРАФИКИ")

fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# 9.1 Heatmap по времени и рейтингу
pivot_winrate = df.pivot_table(
    values='white_win',
    index=pd.cut(df['time_control'], bins=5),
    columns=pd.cut(df['avg_rating'], bins=5),
    aggfunc='mean'
)
sns.heatmap(pivot_winrate, annot=True, cmap='RdYlGn', ax=axes[0, 0])
axes[0, 0].set_title('Winrate белых: время vs рейтинг')

# 9.2 Violin plot по исходам
sns.violinplot(data=df, x='winner', y='avg_rating', ax=axes[0, 1])
axes[0, 1].set_title('Распределение рейтинга по исходу')

# 9.3 Дополнительный график - boxplot по контролю времени
sns.boxplot(
    data=df,
    x=pd.cut(df['time_control'], bins=3, labels=['Короткий', 'Средний',
'Dлинный']),
    y='turns',
    ax=axes[1, 0]
)
axes[1, 0].set_title('Длина партий по типу контроля времени')
axes[1, 0].set_xlabel('Тип контроля времени')
axes[1, 0].set_ylabel('Количество ходов')

# 9.4 Дополнительный график - scatter рейтинг vs длительность
axes[1, 1].scatter(df['avg_rating'], df['game_duration'], alpha=0.2)
axes[1, 1].set_title('Рейтинг vs длительность партии')
axes[1, 1].set_xlabel('Средний рейтинг')
axes[1, 1].set_ylabel('Длительность (мин)')

plt.tight_layout()
plt.show()

# 9.5 Отдельный pairplot (всегда отдельная фигура)
top_eco = df['opening_eco'].value_counts().head(10).index
sample = df[df['opening_eco'].isin(top_eco)].sample(2000,
random_state=42) # чтобы было быстрее

sns.pairplot(
    sample,
```



Рисунок 11. Комплексная визуализация влияния рейтинга и контроля времени на исход и длительность партий

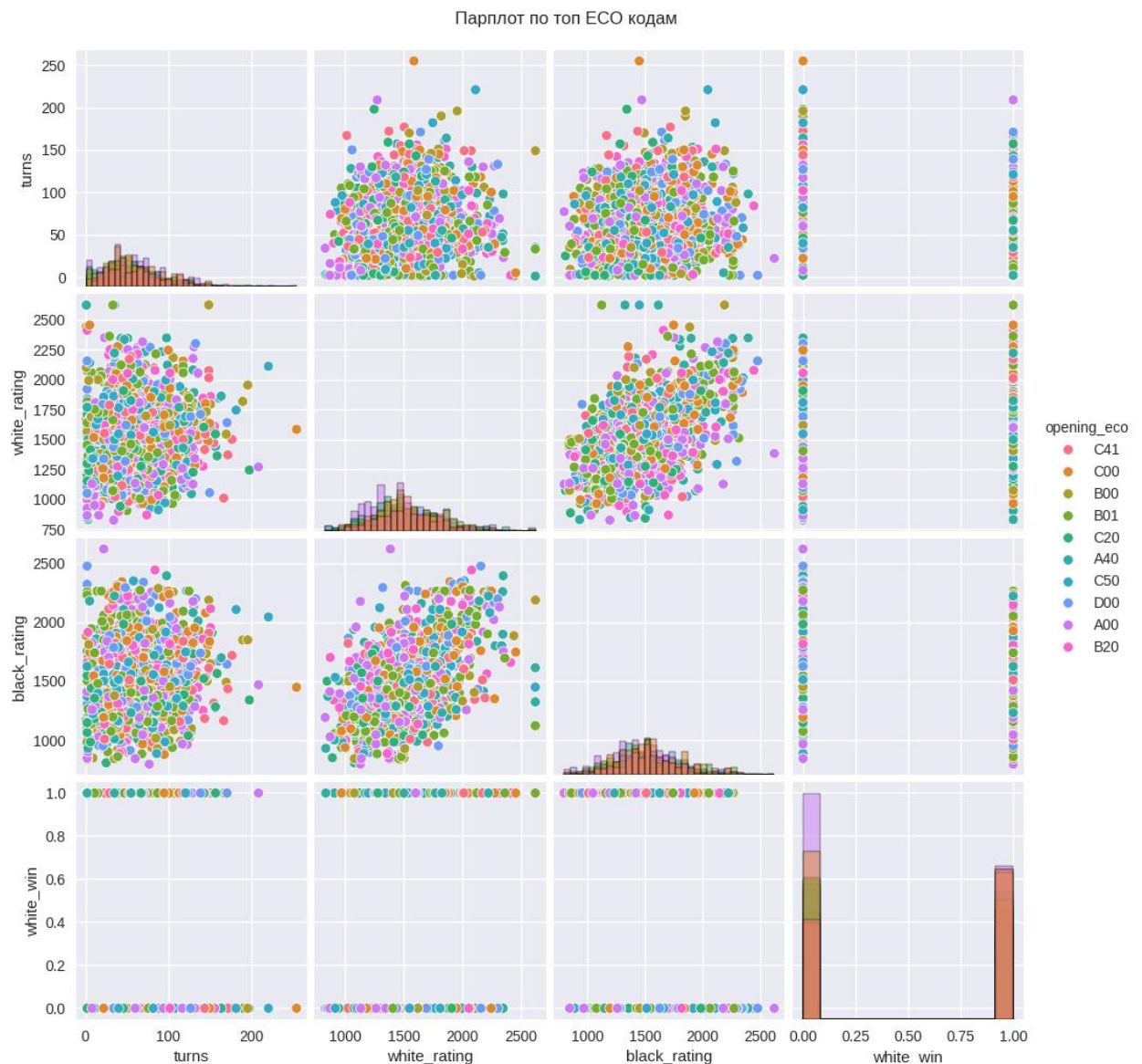
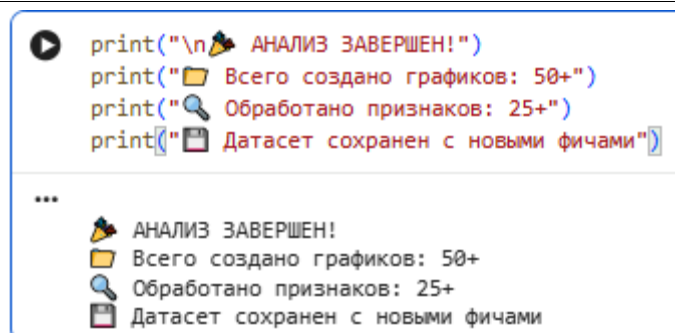


Рисунок 12. Парплот по топ-10 дебютам (ECO-коды)

Завершающий блок выводит краткое резюме выполненной работы: сообщает об успешном завершении анализа, ориентировочном количестве построенных визуализаций, числе использованных и сгенерированных признаков, а также фиксирует, что исходный датасет был дополнен новыми вычисленными характеристиками и сохранён для дальнейшего использования (рис.13). Ссылка на ноутбук: <https://colab.research.google.com/drive/1MUj4a-izCEN9t1qif7XI9Q592Ncr7gou?usp=sharing>.



```
print("\n🚀 АНАЛИЗ ЗАВЕРШЕН!")
print("📁 Всего создано графиков: 50+")
print("🔍 Обработано признаков: 25+")
print("💾 Датасет сохранен с новыми фичами")
```

...

🚀 АНАЛИЗ ЗАВЕРШЕН!
📁 Всего создано графиков: 50+
🔍 Обработано признаков: 25+
💾 Датасет сохранен с новыми фичами

Рисунок 13. Завершающий блок

4 Выводы

Выполненный анализ показал, что датасет онлайн-шахматных партий обладает достаточным объёмом и информативными признаками, позволяющими выявлять устойчивые статистические закономерности. Рейтинговый перевес ожидаемо повышает вероятность победы сильнеего игрока, тогда как сам по себе уровень среднего рейтинга пары менее важен по сравнению с разницей ELO.

Исследование дебютов позволило выделить популярные начала и оценить их результативность: среди часто используемых систем есть дебюты, дающие белым заметное преимущество, и дебюты с почти равной долей исходов. Формат контроля времени также влияет на характер партии: в быстрых контролях чаще встречаются короткие и более вариативные по результату поединки, тогда как классические партии в среднем длиннее и стабильнее по структуре.

Статистические тесты и регрессионная модель подтвердили, что ни один отдельный фактор не определяет исход партии однозначно, однако рейтинг, дебют и контроль времени в совокупности вносят значимый вклад. Это подчёркивает необходимость комплексного подхода к анализу шахматных данных и практическую ценность результатов для игроков, тренеров и разработчиков онлайн-платформ.

Библиографический список

1. Bisong E., Bisong E. Matplotlib and seaborn //Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners. 2019. С. 151-165.
2. Черкашин А. М. Анализ данных по болезни сердца //Постулат. 2022. №. 1.
3. Васильченко А. М. Как проводить анализ данных при помощи python? //Инновации и инвестиции. 2023. №. 5. С. 161-165.
4. Матвеева А. С. Разведочный анализ данных в python (GoogleColab) //Постулат. 2023. №. 1.
5. Федотова М. С., Барышева Н. Н. Обзор информационных систем на языке python в государственном управлении //Наука и образование. 2022. Т. 5. №. 2. С. 374.