

Финальная работа по курсу «Аналитик данных с нуля 2.0»

Описание кейса

Проведение A/B-теста

Представьте, что вы работаете в компании, которая разрабатывает игры. Ваш основной хит — бесплатный командный онлайн-шутер. В игре есть внутриигровая валюта, которую вы можете выигрывать, побеждая в матчах, а можете покупать за настоящие деньги.

На днях в игре прошёл A/B-тест — некоторые игроки могли приобрести премиумную броню по скидке. Ваше руководство хочет узнать, как это повлияло на ARPU (средняя прибыль на игрока), ARPPU (средняя прибыль на платящего игрока) и траты внутриигровой валюты.

Цель работы

Выяснить, стоит ли проводить акцию в дальнейшем. Если игроки, участвовавшие в акции, принесли больше денег, чем игроки, у которых акции не было, то стоит повторять акцию и при этом уже на всех игроках.

Папки с данными

- Money — таблица с платежами.
- Cheaters — таблица с обнаруженными читерами.
- Platforms — таблица с игровыми платформами (PC, PS4, Xbox).
- Cash — таблица с тратами внутриигровой валюты.
- ABgroup — таблица с распределением игроков по группам теста.

Некоторые детали

- Среди игроков есть читеры — игроки, которые с помощью взлома игры начисляют себе большие объёмы внутриигровой валюты. У вас есть список известных вам читеров, но есть и ещё не пойманные читеры, чьи результаты могут повлиять на выводы. Попробуйте найти их.

- Чтобы сравнить результаты тестовой и контрольной групп, вам нужно сравнить средние по группам, а также построить доверительные интервалы от средних значений с точностью 95%. Если доверительные интервалы пересекаются, то это означает, что результаты случайны и акция не принесла результатов.

- Для разных платформ результаты могут быть разными. Структура работы
Формат финальной работы — текстовый документ с подробным описанием проекта, расчётами и выводами. Укажите ссылки на используемые матери

Раздел 1. Цель проекта

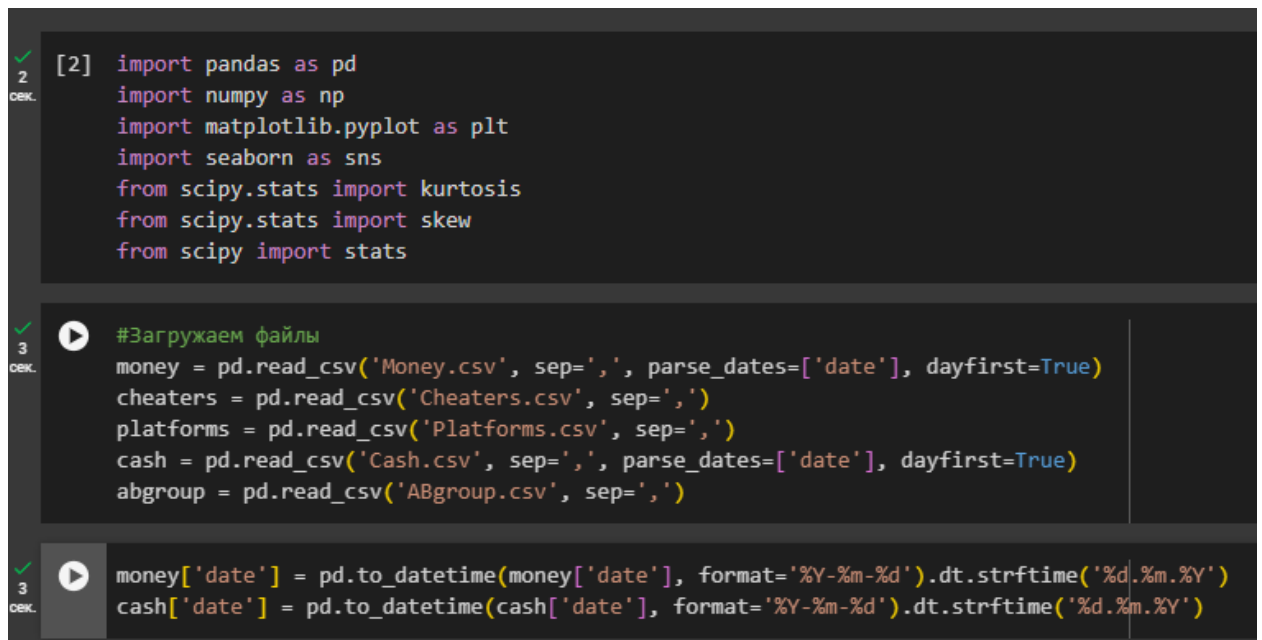
Для анализа результатов А/В-тестирования необходимо сравнить показатели контрольной и тестовой групп: среднюю прибыль на игрока (ARPU) и среднюю прибыль на платящего игрока (ARPPU), а также сравнить Paying Share (ARPU/ARPPU).

Если акция была успешной, только в том случае, когда результаты метрик в тестовой группе участников должно быть больше, чем в контрольной.

Раздел 2. Анализ источников

Поскольку данные представлены в виде CSV-файлов, удобно использовать язык программирования Python на платформе Jupyter Notebook.

Для работы с этими данными я выбрал Python, так как это мощный инструмент для анализа данных и построения графиков. Ниже приведён исходный код для загрузки данных (Рисунок 1).



```
[2] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import kurtosis
from scipy.stats import skew
from scipy import stats

#Загружаем файлы
money = pd.read_csv('Money.csv', sep=',', parse_dates=['date'], dayfirst=True)
cheaters = pd.read_csv('Cheaters.csv', sep=',')
platforms = pd.read_csv('Platforms.csv', sep=',')
cash = pd.read_csv('Cash.csv', sep=',', parse_dates=['date'], dayfirst=True)
abgroup = pd.read_csv('ABgroup.csv', sep=',')

money['date'] = pd.to_datetime(money['date'], format='%Y-%m-%d').dt.strftime('%d.%m.%Y')
cash['date'] = pd.to_datetime(cash['date'], format='%Y-%m-%d').dt.strftime('%d.%m.%Y')
```

Рисунок 1 – Исходный код для загрузки данных

Раздел 3. Очистка данных

Чтобы обнаружить читеров, которые ранее не были замечены, необходимо проанализировать информацию о тех, кого уже поймали (Рисунок 2). Для этого следует изучить их расходы на игровую валюту и платежи, а затем сравнить полученные данные с показателями обычных игроков (Рисунок 3).

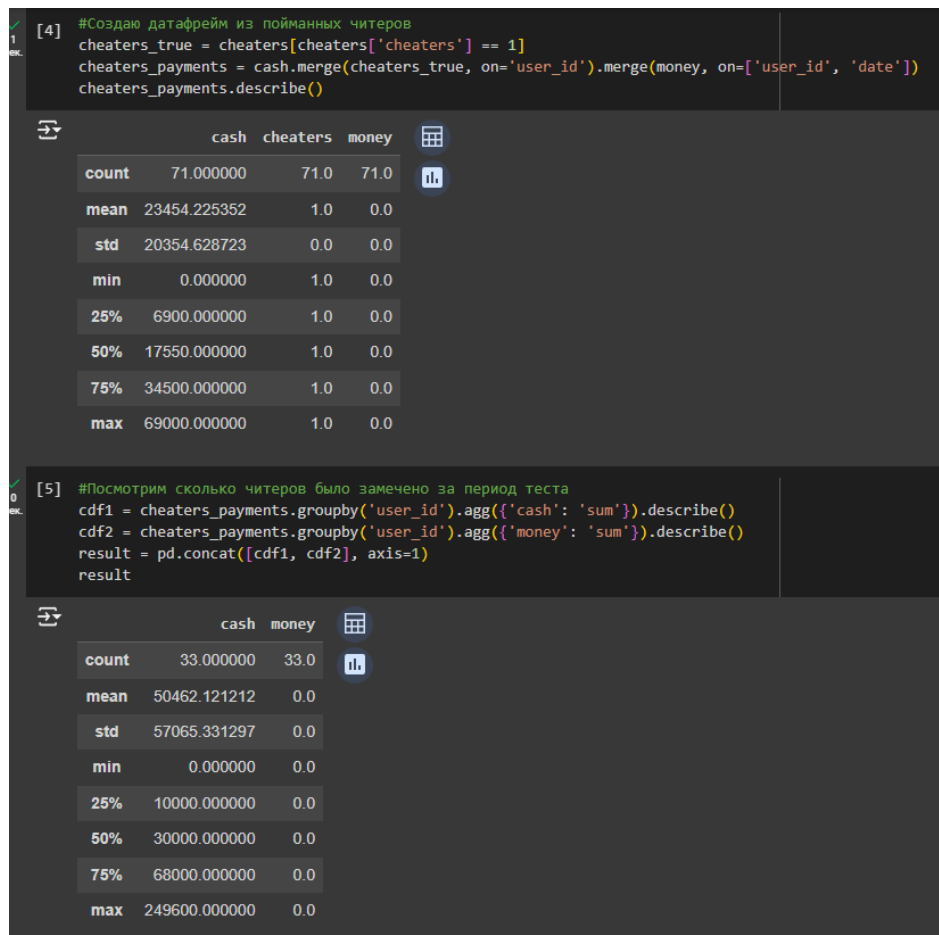


Рисунок 2 – Анализ пойманных читеров

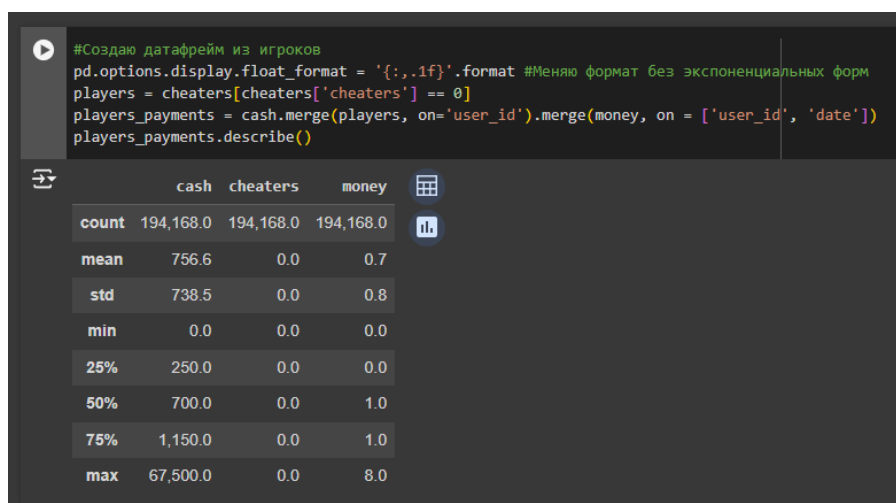


Рисунок 3 – Анализ данных игроков

Метод заключается в том, что читеры не тратят деньги непосредственно на игровую валюту, но их общие расходы значительно превышают затраты обычных игроков (Рисунок 4-5).

```

players_payments_grouped = players_payments.groupby('user_id').agg({'cash': 'sum'})
player_level_95 = np.percentile(players_payments_grouped['cash'], 95)

suspicious_player = (players_payments['cash'] > player_level_95) & (players_payments['money'] == 0)
suspicious_player_filtered = players_payments[suspicious_player]

suspicious_player_filtered

```

	user_id	date	cash	cheaters	money
4135	GP90LE-1HZR7A	16.07.2021	23,750.0	0.0	0.0
4136	GP90LE-1HZR7A	16.07.2021	23,750.0	0.0	0.0
8162	4BD9PQ-TPSMQZ	12.07.2021	23,200.0	0.0	0.0
18839	U4AFEA-8H7Q3K	10.07.2021	7,500.0	0.0	0.0
18840	U4AFEA-8H7Q3K	10.07.2021	7,500.0	0.0	0.0
19006	ZWHACF-Z2J5IA	10.07.2021	31,900.0	0.0	0.0
28170	5JFM8H-U3T8AI	16.07.2021	9,900.0	0.0	0.0
28171	5JFM8H-U3T8AI	16.07.2021	9,900.0	0.0	0.0
28922	4DBX3L-2LHKON	11.07.2021	9,450.0	0.0	0.0
28923	4DBX3L-2LHKON	11.07.2021	9,450.0	0.0	0.0
28924	4DBX3L-2LHKON	11.07.2021	9,450.0	0.0	0.0

Рисунок 4 – Метод поиска непойманных читеров

```

[8] # Ловим читеров
suspicious_player_filtered.reset_index(inplace=True)
cheaters.loc[cheaters['user_id'].isin(suspicious_player_filtered['user_id']), 'cheaters'] = 1
players = cheaters[cheaters['cheaters'] == 0].merge(abgroup, on = 'user_id')

# Проверка
cheaters.loc[cheaters['user_id'].isin(suspicious_player_filtered['user_id'])]

```

	user_id	cheaters
1577	XKTIX5-M7HG8T	1.0
32080	4HCVCA-V3M19Y	1.0
88353	4HCVCA-V3M19Y	1.0
101066	0MPTOG-ZCT5IV	1.0
191700	J5XJ9D-93L10T	1.0
203943	UEOJ79-4NVE5I	1.0
225473	ZWHACF-Z2J5IA	1.0
257731	XKTIX5-M7HG8T	1.0
265069	XKTIX5-M7HG8T	1.0
270830	U4AFEA-8H7Q3K	1.0
369905	173ARI-2L41OB	1.0
456544	KQ8EQQ-DEAKPK	1.0

Рисунок 5 – Добавление читеров в черный список

Раздел 4. Использование статистических методов

Чтобы найти ARPU (Рисунок 10) и ARPPU (Рисунок 12), нужно определить прибыль по каждой группе, разделив её на количество пользователей в этой группе. ARPPU рассчитывается аналогичным образом, но учитываются только те пользователи, которые оплачивали подписку в течение тестового периода (Рисунок 11).

Для определения доверительных интервалов мы будем использовать библиотеку `scipy`. В качестве исходных данных мы возьмём колонку «money» и «cash», которая содержит информацию о сумме платежей пользователей за период тестирования. Доверительная вероятность составляет 95%.

Для применения этого метода нам понадобятся следующие значения (Рисунок 6): среднее и стандартное отклонение. Среднее значение можно найти с помощью метода `df[column].mean()`, а стандартное отклонение — с помощью метода `df[column].std()`.

```
def researh(df, column, color = 'purple', text = 'График распределения'):
    print("Базовые метрики :")
    print(f"{df[column].describe()}\n")
    print(f'Медиана > {df[column].median()}\n')

    print(f"Топ 5 метриков :")
    print(f"{df[column].value_counts().nlargest(5)}\n")

    print(f"Экцесс > {kurtosis(df[column])}")
    print(f"Ассиметрия > {skew(df[column])}\n")

    plot = plt.hist(df[column], color = color, bins=12);
    plt.tight_layout()
    plt.show()
```

Рисунок 6 – Функция для анализа данных

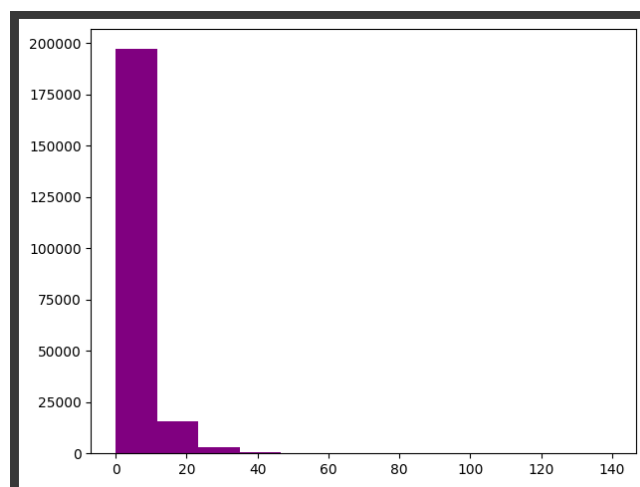


Рисунок 7 – График распределения игроков тестовой группы

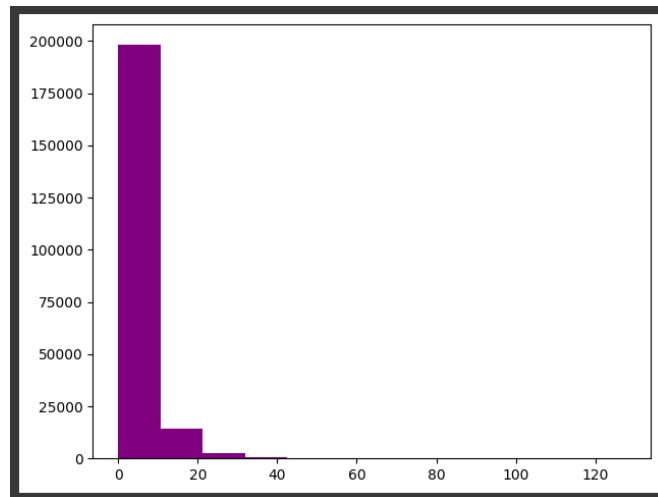


Рисунок 8 – График распределения игроков контрольной группы

```
#Поиск суммы платежей
print(f'Сумма платежей тестовой группы > {test_group.money.sum()}')
print(f'Сумма платежей контрольной группы > {control_group.money.sum()}')

Сумма платежей тестовой группы > 833987.6600000001
Сумма платежей контрольной группы > 783245.8000000004
```

Рисунок 9 – Поиск суммы платежей

```
[16] # Создаю функцию для поиска ARPU или ARPPU
def aru(df, column):
    ARU = df[column].sum() / df.drop_duplicates('user_id').count()[0]
    return ARU

#Поиск ARPU
ARPU_test = aru(test_group, 'money')
ARPU_control = aru(control_group, 'money')

print(f'ARPU тестовой группы > {ARPU_test}\nARPU контрольной группы > {ARPU_control}')

ARPU тестовой группы > 3.8554136540986343
ARPU контрольной группы > 3.6311307677686466

#Разница ARPU между тестовым и контрольной групп
print(f'ARPU тестовой группы больше контрольной на {round((ARPU_test - ARPU_control) / ARPU_control * 100)} %')

ARPU тестовой группы больше контрольной на 6 %
```

Рисунок 10 – Поиск ARPU

```
#Отделяем платящих игроков
df = test_group.groupby('user_id').agg({'money': 'sum'}).reset_index()
test_group_paying = df[df['money'] > 0]

df = control_group.groupby('user_id').agg({'money': 'sum'}).reset_index()
control_group_paying = df[df['money'] > 0]
```

Рисунок 11 – Поиск платящих игроков

```
[24] #ARPPU
ARPPU_test = aru(test_group_paying, 'money')
ARPPU_control = aru(control_group_paying, 'money')
print(f'ARPPU тестовой группы > {ARPPU_test}\nARPPU контрольной группы > {ARPPU_control}')
```

ARPPU тестовой группы > 5.552662254653919
ARPPU контрольной группы > 5.3890216800489865

```
[25] #Создаем функцию создания расчета доверительного интервала
def my_norm_confidence(df, column, alpha=0.95):
    interval = stats.norm.interval(alpha, loc=df[column].mean(), scale=df[column].std())
    return interval
```

```
[26] test_paying_confidence = my_norm_confidence(df = test_group_paying.groupby('user_id').agg({'money': 'sum'}), column = 'money')
print(f'Доверительный интервал тестовой группы > {test_paying_confidence}')
control_paying_confidence = my_norm_confidence(df = control_group_paying.groupby('user_id').agg({'money': 'sum'}), column = 'money')
print(f'Доверительный интервал контрольной группы > {control_paying_confidence}')
```

Доверительный интервал тестовой группы > (-5.859513767100298, 16.964838276408138)
Доверительный интервал контрольной группы > (-5.445034998529283, 16.223078358627255)

```
#Разница ARPPU между тестовым и контрольной групп
print(f'ARPPU тестовой группы больше контрольной на {round((ARPPU_test - ARPPU_control) / ARPPU_control * 100)} %')
```

ARPPU тестовой группы больше контрольной на 3 %

Рисунок 12 – Поиск ARPPU

Вывод: согласно результатам исследования, добавление акций оказывает незначительное влияние на прибыль и активность игроков.

Раздел 5. Отчет

ARPU = трата валюты (или прибыль) / кол-во игроков

ARPPU = трата валюты (или прибыль) / кол-во платящих игроков

Траты валют и прибыль считаются суммарно за тестовый период (Рисунок 13 - 15).

	date	paying_users	money	cash	users	ARPU	ARPPU	ARSU	ARSPU
0	10.07.2021	2954	11,813.5	12,500,800.0	15884	0.7	4.0	787.0	4,231.8
1	11.07.2021	3055	12,208.6	13,149,150.0	15884	0.8	4.0	827.8	4,304.1
2	12.07.2021	3055	12,228.4	13,150,100.0	15884	0.8	4.0	827.9	4,304.5
3	13.07.2021	2976	12,089.7	12,785,050.0	15884	0.8	4.1	804.9	4,296.1
4	14.07.2021	2928	11,249.7	11,952,550.0	15884	0.7	3.8	752.5	4,082.2
5	15.07.2021	2707	10,507.5	10,525,850.0	15884	0.7	3.9	662.7	3,888.4
6	16.07.2021	2467	8,801.0	8,245,400.0	15884	0.6	3.6	519.1	3,342.3
7	17.07.2021	2991	19,588.2	17,035,750.0	15884	1.2	6.5	1,072.5	5,695.7

Рисунок 13 – Сравнение метрик

2				
3	Названия строк	user.sum()	money.sum()	ARPU.sum()
4	PC	694957	760230,03	0,914140421
5	control	346402	357353,72	0,969353278
6	test	348555	402876,31	0,865166284
7	PS4	695368	763258,44	0,911051832
8	control	347436	371200,8	0,935978586
9	test	347932	392057,64	0,887451141
10	XBox	698992	786554,13	0,888676282
11	control	350091	392487,71	0,891979522
12	test	348901	394066,42	0,885386276
13	Общий итог	2089317	2310042,6	0,904449554
14				

Рисунок 14 – Сводная таблица в Excel с ARPU по группам и платформам

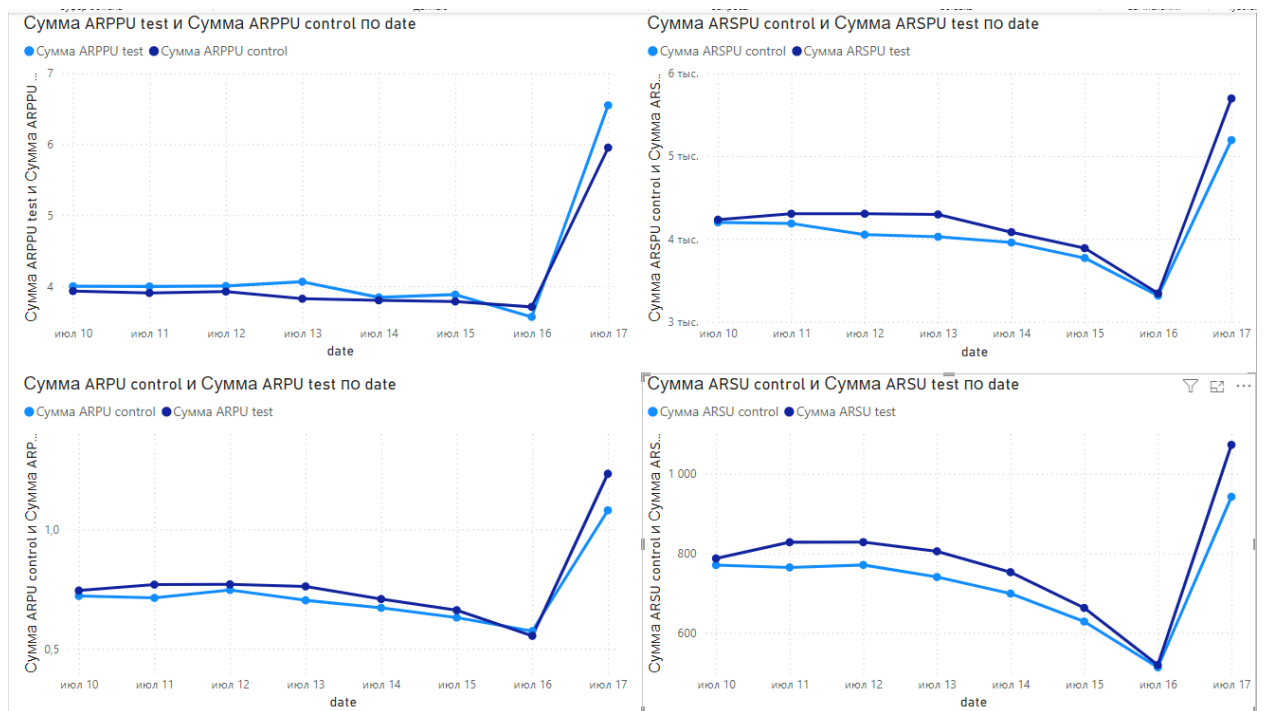


Рисунок 15 – Графики сравнения метрик по дням в Power BI

Ссылки на источники

1. Основа pandas: <https://education.yandex.ru/handbook/python/article/modul-pandas>
2. Основы numpy: <https://education.yandex.ru/handbook/python/article/modul-math-i-numpy>
3. Основа Matplotlib: <https://skillbox.ru/media/code/biblioteka-matplotlib-dlya-postroeniya-grafikov/>
4. Основа Seaborn: <https://habr.com/ru/companies/otus/articles/540526/>
5. Основа SciPy: <https://habr.com/ru/articles/701016/>