Финальная работа по курсу:

"Аналитик данных с нуля 2.0"

Автор:

Сашина Юлия Борисовна

Город Тюмень, 2025 г.

Содержание

Раздел 1. Цель проекта

Какую задачу необходимо решить?

В игре (бесплатный командный онлайн-шутер) есть внутриигровая валюта, которую игроки могут выигрывать, побеждая в матчах, а могут покупать за настоящие деньги.

Необходимо выяснить, стоит ли проводить акцию по премиальной броне по скидке в дальнейшем через анализ влияния на ARPU (средняя прибыль на игрока), ARPPU (средняя прибыль на платящего игрока) и траты внутриигровой валюты. 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.

Как подходить к её решению?

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

Чтобы сравнить результаты тестовой и контрольной групп, 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.

Раздел 2. Анализ источников

Какой вариант работы с данными выбран: БД (SQL) / Python. Описание, почему был выбран используемый вариант.

Python и SQL являются двумя самыми популярными средствами для анализа данных и управления данными. SQL и Python используются для разных целей: SQL используется для управления данными, а Python — для разработки, машинного обучения, анализа данных и так далее. Поскольку задачей проекта является анализ исходных данных в формате csv, Python отлично подойдет для данной работы.

Python — это универсальный язык, который можно использовать для решения широкого спектра задач, что делает его подходящим для различных проектов и отраслей. Используя обширные библиотеки и фреймворки, мы можем анализировать данные и манипулировать ими, а также получать мощные возможности с помощью таких библиотек, как Pandas, Numpy. Python — простой в освоении и читаемый язык. У него понятный синтаксис, который помогает легче писать код.

Исходники кодов по загрузке данных

Обработка данных проводилась в облачной среде Google Colab на языке Python. Код для загрузки библиотек:

```
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
```

```
import scipy.stats as st
from scipy import stats
```

Код для загрузки файлов csv с локального компьютера в Google Colab:

```
from google.colab import files
uploaded = files.upload()
```

Выгрузка списка имен столбцов каждой таблицы с помощью кода:

```
df = pd.read_csv('ABgroup.csv')#название каждой из 5 таблиц
header = list(df.columns)
print("Список имён столбцов:", header)
```

Исходные данные содержатся в 5 таблицах в формате csv со следующими именами столбцов:

- ABgroup ['user_id', 'group']

- Cash ['user_id', 'date', 'cash']

- Cheaters ['user_id', 'cheaters']

- Money ['user_id', 'date', 'money']

- Platforms ['user_id', 'platform']

Код для загрузки данных в DataFrame:

```
df_cash = pd.read_csv('Cash.csv') #название таблицы
```

Раздел 3. Очистка данных

Описание, как вычислять читеров. Каким способом пользоваться, чтобы убрать данные по тем читерам, которые не были обнаружены на момент теста

Код для создания DataFrame cash и money и подготовки данных к работе над задачей:

```
df_cash = pd.read_csv('Cash.csv')
del df_cash['date'] #удаляем столбец с датой
df_cash = df_cash.groupby(['user_id'])['cash'].sum().reset_index()
#объединяем одинаковые id, суммируем cash, возвращаем в таблицу
df_cash.head()
df_money = pd.read_csv('Money.csv')
del df_money['date']
df_money = df_money.groupby(['user_id'])['money'].sum().reset_index()
df_money.head()
```

Объединим cash и money в общий DataFrame:

```
df_merge = df_cash.merge(df_money, on='user_id')
df_unique = df_merge.drop_duplicates()
df_unique.head()
```

```
df_unique.shape[0] #число юзеров в датафрейме 1 080 000
```

Нужно очистить данные – удалить читеров из файла cheaters.csv, используя данные из столбца cheaters (1 – читер, 0 – не читер). Таким образом будут удалены выявленные читеры.

Код для создания DataFrame cheaters и удаления выявленных читеров:

```
df_chiters = pd.read_csv('Cheaters.csv')
df_chiters = df_chiters.loc[(df_chiters['cheaters'] == 1)] #фильтруем по 1

list_of_chiters = df_chiters['user_id'].tolist() #создадим список с id
```

```
print(len(set(list_of_chiters))) #в списке 353 выявленных читера
```

Код для удаления читеров из списка читеров из объединенного DataFrame cash и money:

```
df_unique = df_unique[~df_unique['user_id'].isin(list_of_chiters)]
df_unique.head()
```

```
df_unique.shape[0] #количество оставшихся юзеров 1 079 647
```

Невыявленных читеров будем искать путем 99 квантиля. Создадим DataFrame с выбором по условиям: (cash для читеров больше 99 квантиля и юзеры с money = 0):

```
quantile = df_unique['cash'].quantile(0.99)
quantile #получилось 9350
```

```
df_ch = df_unique.loc[(df_unique['cash'] > 9350) & (df_unique['money'] ==
0)]
df_ch.head()
```

```
df_ch.shape[0] #получилось 344 невыявленных читера
```

Удалим найденных на последнем шаге читеров из списка читеров:

```
chiter_list = df_ch['user_id'].tolist() #из столбца df список читеров
создаем список с читерами
df_unique = df_unique[~df_unique['user_id'].isin(chiter_list)] #удаляем,
если противоположно {id юзера есть в списке}, т.е. не в списке
df_unique.shape[0] #количество оставшихся юзеров - не читеров 1 079 303
```

Загрузим данные файла Platforms.csv в DataFrame и добавим в таблицу с юзерами столбец с названием платформы:

```
df_platform = pd.read_csv('Platforms.csv')

df_unique = df_unique.merge(df_platform, on='user_id', how='left')
df_unique = df_unique.drop_duplicates()
df_unique.head()
```

Загрузим данные файла ABgroup.csv в DataFrame и добавим в таблицу столбец с типом группы (тестовая или контрольная):

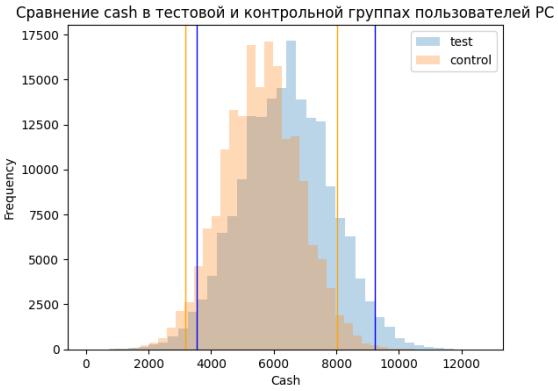
```
df_group = pd.read_csv('ABgroup.csv')
df_unique = df_unique.merge(df_group, on='user_id', how='left')
df_unique = df_unique.drop_duplicates()
df_unique.head()
```

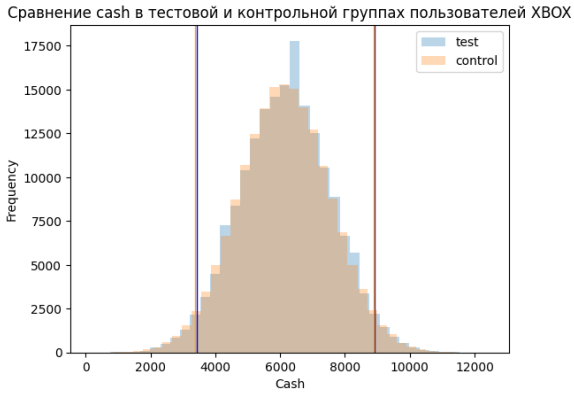
Раздел 4. Использование статистических методов

Процесс построения доверительных интервалов для каждой метрики

Разделим данные на 3 группы платформ: PC, XBox и PS4, а затем по каждой платформе на тестовую и контрольную группы.

Посчитаем средние cash (траты внутриигровой валюты) для каждой платформы для тестовой группы и для контрольной группы, сравним средние по группам, вычислим доверительный интервал, построим графики и сравним их.





Исходники кода

Разделим юзеров по 3 платформам путем создания DataFrame для каждой платформы:

```
df_pc = df_unique[df_unique['platform'] == 'PC']
df_xbox = df_unique[df_unique['platform'] == 'XBox']
df_ps4 = df_unique[df_unique['platform'] == 'PS4']
```

Разделим юзеров каждой платформы на тестовую и контрольную группу (на примере платформы PC):

```
df_pc_test = df_pc[df_pc['group'] == 'test']
df_pc_control = df_pc[df_pc['group'] == 'control']
```

Посчитаем средние cash (траты внутриигровой валюты) для каждой платформы для тестовой группы и для контрольной группы и вычислим доверительный интервал (на примере платформы PC):

```
df_pc_test_mean = df_pc_test['cash'].mean()
df_pc_test_std = df_pc_test['cash'].std()

ci_test = stats.norm.interval(0.95, loc=df_pc_test_mean,
scale=df_pc_test_std)
print(df_pc_test_mean)
print(df_pc_test_std)
print(f'95% доверительный интервал: {ci_test}')
df_pc_control_mean = df_pc_control['cash'].mean()
df_pc_control_std = df_pc_control['cash'].std()

ci_control = stats.norm.interval(0.95, loc=df_pc_control_mean,
scale=df_pc_control_std)
print(df_pc_control_mean)
print(df_pc_control_std)
print(f'95% доверительный интервал: {ci_control}')
```

Построим графики (на примере платформы PC):

```
plt.hist(df_pc_test['cash'], bins=40, alpha=0.3, label='test')
plt.hist(df_pc_control['cash'], bins=40, alpha=0.3, label='control')

plt.axvline(ci_test[0], color='blue', linewidth=1)
plt.axvline(ci_test[1], color='blue', linewidth=1)
plt.axvline(ci_control[0], color='orange', linewidth=1)
plt.axvline(ci_control[1], color='orange', linewidth=1)

plt.title('Сравнение cash в тестовой и контрольной группах пользователей
PC')
plt.ylabel('Frequency')
plt.xlabel('Cash')
plt.legend(loc='upper right')
plt.savefig('pc.png')
plt.show()
```

Раздел 5. Формирование отчёта

Сравнение метрик ARPU (средняя прибыль на игрока), ARPPU (средняя прибыль на платящего игрока) и траты внутриигровой валюты между различными группами игроков

Сравнение ARPU (средняя прибыль на игрока):

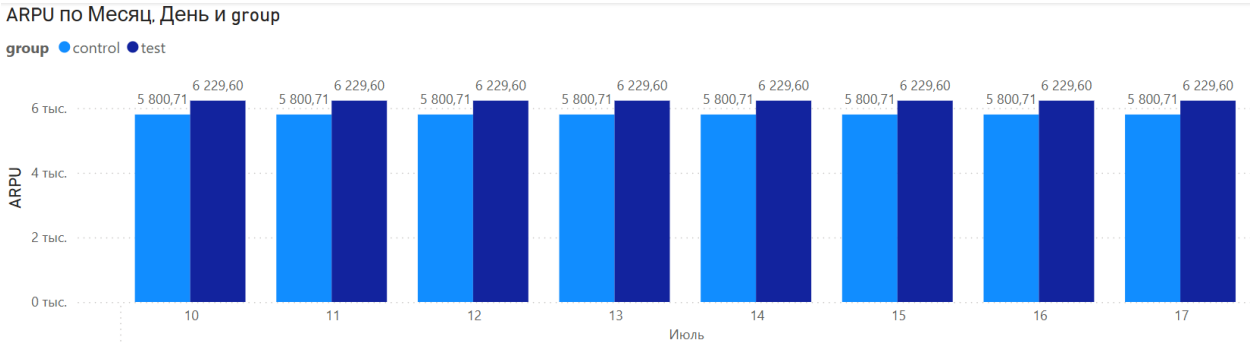
PC		XBox		PS4	
test	control	test	control	test	control
6 382	5 587	6 176	6 132	6 130	5 682

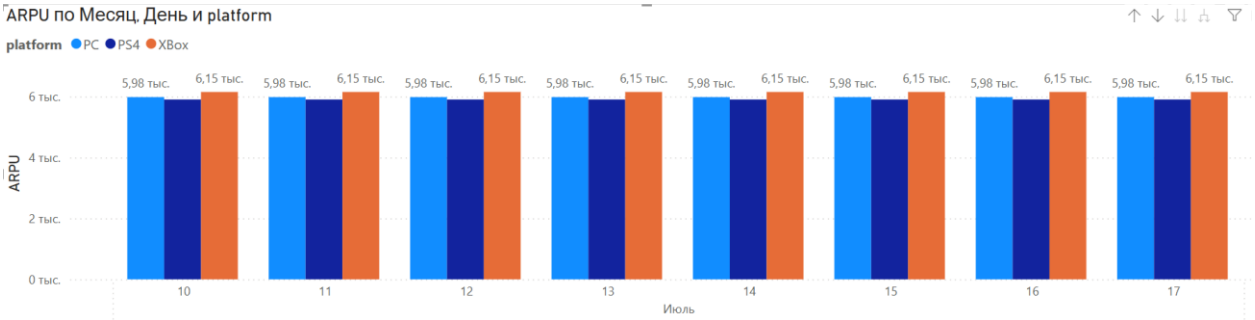
Сравнение ARPPU (средняя прибыль на платящего игрока):

PC		XBox		PS4	
test	control	test	control	test	control
6,27	5,65	6,13	6,10	6,08	5,74

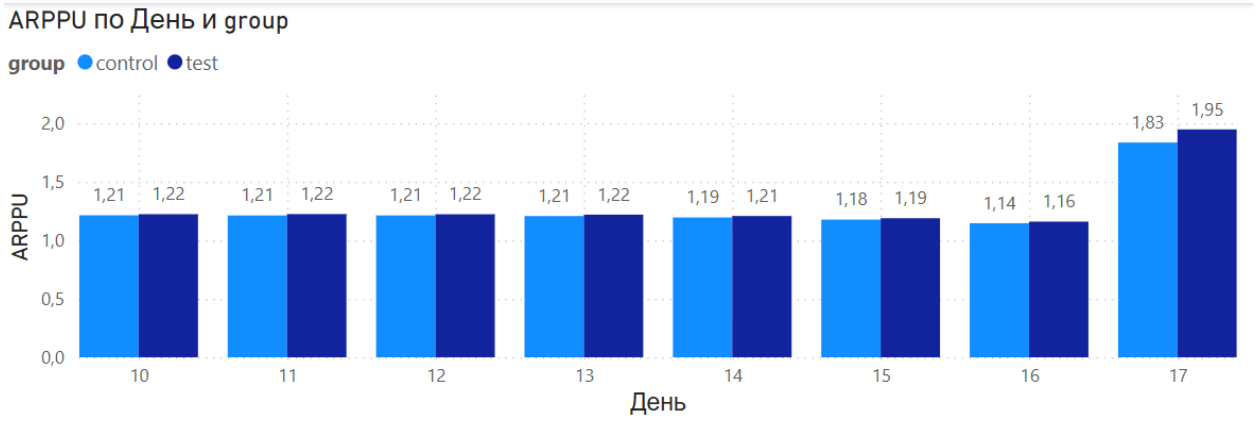
Графики сравнения метрик по дням (Power BI)

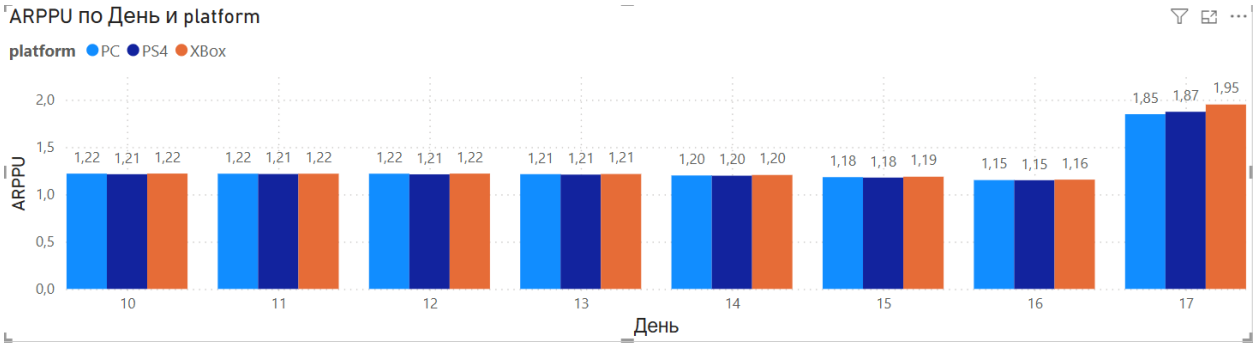
ARPU по дням (по группам и платформам):





ARPPU по дням (по группам и платформам):





Исходники программы Python для построения графиков и таблиц

Считаем ARPU и ARPPU для тестовой и контрольной группы по каждой платформе (на примере тестовой группы платформы PC):

```
df_pc_test_mean
```

```
df_pc_test_money_mean = df_pc_test['money'.mean()
df_pc_test_money_mean
```

Сохраняем DataFrame таблицы money, cash, unique в файлы формата csv, предварительно отфильтровав юзеров от читеров (на примере cash):

```
df_cash = pd.read_csv('Cash.csv')
df_cash = df_cash[~df_cash['user_id'].isin(list_of_chiters)] #удаляем
выявленных читеров
df_cash = df_cash[~df_cash['user_id'].isin(chiter_list)] #удаляем
невыявленных читеров
df_cash.to_csv('Cash_withoutCheaters.csv', sep=';')
```

Сводная таблица в Excel с ARPU по группам и платформам

В MS Excel сохранить файл df_unique с объединенными данными ARPU по группам и платформам не получается в связи с очень большим объемом данных. Её можно заменить матрицей MS Power BI:

group	Среднее значение cash	Среднее значение money
control	**5 800,71**	**5,83**
PC	5 587,10	5,65
PS4	5 681,94	5,74
XBox	6 131,79	6,10
test	**6 229,60**	**6,16**
PC	6 381,75	6,27
PS4	6 130,49	6,08
XBox	6 176,40	6,13
Всего	**6 015,02**	**6,00**

Раздел 6. Выводы

Анализ по платформам:

1. Xbox

Графики практически слились: доверительные интервалы для тестовой и контрольной групп практически пересекаются. Разница между группами статистически незначима: результаты следует считать случайными. Проводить акцию повторно не нужно.

2. PS4

График тестовой группы смещен чуть правее: средние траты немного выше в тестовой группе, чем в контрольной. Доверительные интервалы имеют небольшое расхождение. Вероятно, есть незначительный положительный эффект от акции. Лучше не распространять акцию на всех игроков, а провести повторную акцию на тестовой группе с новой выборкой.

3. PC

График тестовой группы смещен значительно правее, чем в тестах 2 предыдущих платформ, хотя доверительные интервалы по-прежнему значительно пересекаются. Акция оказала положительный эффект, пусть и не особо значительный. Можно провести повторную акцию на тестовой группе, например, с новой расширенной выборкой, или же рекомендовать повторить акцию.