**Финальная работа по курсу «Аналитик данных с**

**нуля 2.0»**

**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.

# Раздел 1. Цель проекта.

Необходимо проанализировать данные полученные во время A/B тестирования и определить – имеется ли эффект от проведения акции. Для этого нужно сравнить метрики контрольной и тестовой группы, а именно ARPU(средняя прибыль на игрока) и ARPPU(средняя прибыль на платящего игрока) также можно сравнить Paying Share(ARPU/ ARPPU) и понять – появилось ли больше платящих игроков.

При благоприятном исходе значения этих метрик должны быть больше чем у контрольной группы.

# Раздел 2. Анализ источников

Так как данные представляют собой csv файлы, удобно будет использовать jupyter notebook

Код загрузки выглядит так:

ABgroup = pd.read\_csv('ABgroup.csv', sep=',')

cash = pd.read\_csv('Cash.csv', sep=',', parse\_dates=['date'], dayfirst=True)

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=',')

# Раздел 3. Очистка данных.

Имеется список с обнаруженными читерами, их мы просто удалим из выборки. Однако возможно существуют и другие, не пойманные читеры, для того чтобы их обнаружить нужно сначала проанализировать уже обнаруженных читеров и посмотреть на их траты валюты и платежи, сравнить с обычными игроками.

Читеры тратят значительно больше валюты чем обычные игроки, при этом саму валюту не покупают, это и будет методом поиска читеров.

# Раздел 4. Использование статистических методов

Необходимо найти ARPU и ARPPU, для этого найдем прибыль по каждой группе и разделим на количество пользователей в этой группе, а ARPPU найдем также, но учувствуют пользователи, которые платили за валюту.

Доверительные интервалы найдем с помощью библиотеки scipy, колонка money и cash в которой сумма платежей пользователя за период теста. Доверительная вероятность 95%. Этому методу нужны µ или среднее, и σ то есть стандартное отклонение, также нужно значение доверительной вероятности. Среднее и сигму найдем методами пандаса, df[column].mean(), df[column].std()

Функция для анализа данных:

def researh(df, column):

print("Базовые метрики")

print(df[column].describe())

print('Медиана - ', df[column].median())

print("------------------------------------")

print("Самые популярные значения метрики, топ 5")

print(df[column].value\_counts().nlargest(5))

print("------------------------------------")

print("Эксцесс ", kurtosis(df[column]))

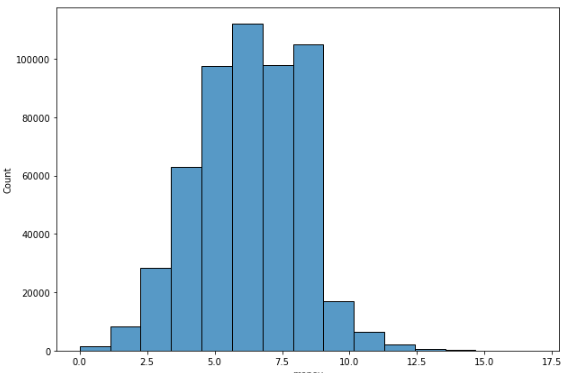
print("Ассиметрия ", skew(df[column]))

sb.histplot(df[column], bins = 15)

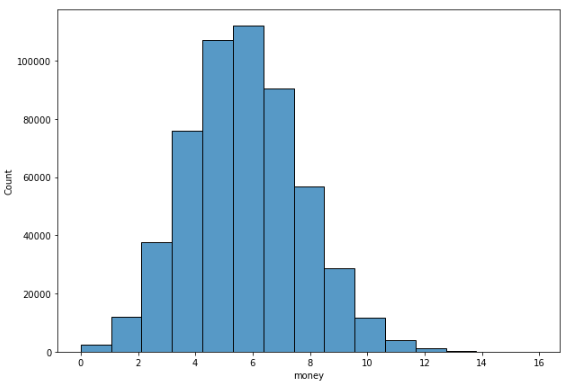
Код для столбчатых диаграмм:

sb.histplot(df[column], bins = 15)

Графики распределения сумм платежей всех игроков:



Тестовая группа



Контрольная группа

Код:

# разделим пользователей на группы и сразу добавим их платежи и платформу

test\_group = players[players['group'] == 'test']\

.merge(money, on = ['user\_id'])\

.merge(platforms, on = ['user\_id'])

control\_group = players[players['group'] == 'control']\

.merge(money, on = ['user\_id'])\

.merge(platforms, on = ['user\_id'])

# найдем суммы платежей для каждой группы

print('Сумма платежей тестовой группы - ', test\_group\_sum := test\_group.money.sum())

print('Сумма платежей контрольной группы - ', control\_group\_sum := control\_group.money.sum())

# найдем ARPU

ARPU\_test = test\_group\_sum / test\_group.drop\_duplicates('user\_id').count()[0]

ARPU\_control = control\_group\_sum / control\_group.drop\_duplicates('user\_id').count()[0]

print('ARPU тестовой группы - ', ARPU\_test, '\n', 'ARPU контрольной группы - ', ARPU\_control)

ARPU тестовой группы - 6.162275602990467

ARPU контрольной группы - 5.829478250563439

# также найдем их доверительные интервалы возьмем доверительную вероятность 95%

def confidence(df, column, alpha = 0.95):

interval = stats.norm.interval(alpha, loc=df[column].mean(), scale=df[column].std())

return interval

test\_confidence = confidence(test\_group.groupby('user\_id').agg('sum'), 'money')

print('Доверительный интервал тестовой группы', test\_confidence)

control\_confidence = confidence(control\_group.groupby('user\_id').agg('sum'), 'money')

print('Доверительный интервал контрольной группы', control\_confidence)

print('ARPU тестовой группы больше контрольной на ', round((ARPU\_test - ARPU\_control) / ARPU\_control \* 100, 1), ' %')

ARPU тестовой группы больше контрольной на 5.7 %

df = test\_group.groupby('user\_id').agg('sum').reset\_index()

test\_group\_paying = df[df['money'] > 0]

df = control\_group.groupby('user\_id').agg('sum').reset\_index()

control\_group\_paying = df[df['money'] > 0]

# найдем ARPPU

ARPPU\_test = test\_group\_sum / test\_group\_paying.drop\_duplicates('user\_id').count()[0]

ARPPU\_control = control\_group\_sum / control\_group\_paying.drop\_duplicates('user\_id').count()[0]

print('ARPPU тестовой группы - ', ARPU\_test, '\n', 'ARPPU контрольной группы - ', ARPU\_control)

ARPPU тестовой группы - 6.162275602990467

ARPPU контрольной группы - 5.829478250563439

test\_paying\_confidence = confidence(test\_group\_paying.groupby('user\_id').agg('sum'), 'money')

print('Доверительный интервал тестовой группы', test\_confidence)

control\_paying\_confidence = confidence(control\_group\_paying.groupby('user\_id').agg('sum'), 'money')

print('Доверительный интервал контрольной группы', control\_confidence)

Доверительный интервал тестовой группы (2.417077142534608, 9.907474063461883)

Доверительный интервал контрольной группы (2.1572457079550493, 9.501710793182074)

print('ARPPU тестовой группы больше контрольной на ', round((ARPPU\_test - ARPPU\_control) / ARPPU\_control \* 100, 1), ' %')

ARPPU тестовой группы больше контрольной на 5.7 %

# Раздел 5. Отчёт

ARPU = прибыль / кол-во игроков

ARPPU = прибыль / кол-во платящих игроков

ARSU = трата валюты / кол-во игроков

ARSPU = трата валюты / кол-во платящих игроков

Траты и прибыль считаются суммарно за период теста.

Таблица 1. Сравнение метрик

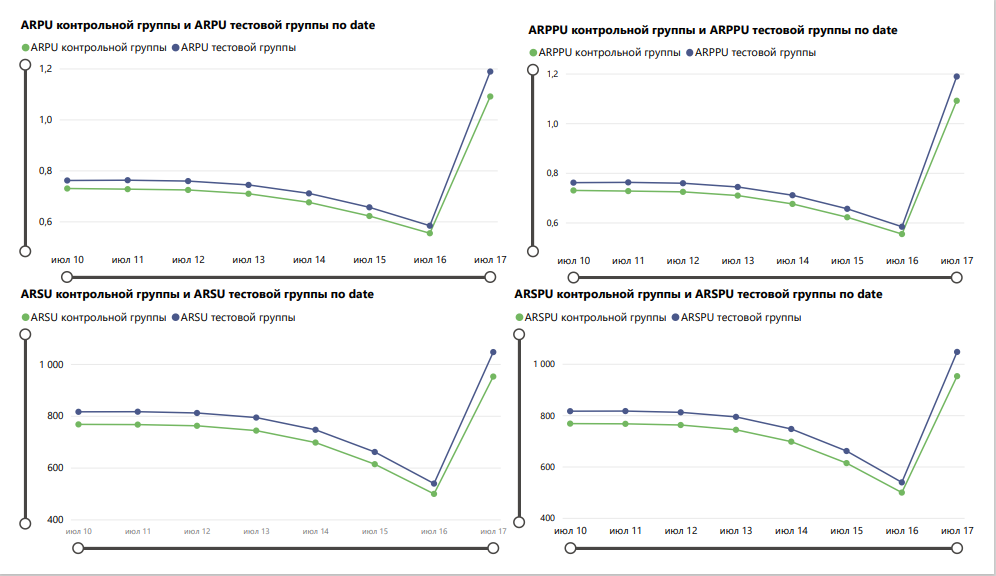
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Платформа | Группа | ARPU | ARPPU | ARSU | ARSPU |
| Все | Тестовая | 6.162275 | 6.163144 | 6229.59 | 6229.99 |
| Все | Контрольная | 5.829478 | 5.831108 | 5800.70 | 5801.40 |
| ПК | Тестовая | 6.269046 | 6.269743 | 6381.74 | 6382.06 |
| ПК | Контрольная | 5.646201 | 5.648306 | 5587.10 | 5587.96 |
| PS4 | Тестовая | 6.084882 | 6.085866 | 6130.49 | 6130.92 |
| PS4 | Контрольная | 5.737646 | 5.739466 | 5681.93 | 5682.68 |
| Xbox | Тестовая | 6.132765 | 6.133685 | 6176.40 | 6176.84 |
| Xbox | Контрольная | 6.103520 | 6.104434 | 6131.79 | 6132.22 |

По результатам анализа по каждым платформам было выявлено, что акция наиболее эффективна если ее проводить для игроков на ПК и ПС4, для игроков на иксбокс результат акции практически отсутствует.

Остальные таблицы в файлах отчет - сравнение метрик по дням.pbix

и table.xlsx

Графики из PowerBI



# Ссылки:

<https://apptractor.ru/measure/user-analytics/arpu-i-arppu-odna-bukva-i-printsipialnyie-otlichiya.html>

<https://pandas.pydata.org/docs/reference/>

<https://seaborn.pydata.org/>

<https://ppt-online.org/566565>

<https://mse.msu.ru/wp-content/uploads/2020/03/%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F-5-%D0%B4%D0%BE%D0%B2%D0%B5%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5-%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB%D1%8B.pdf>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html>