Конспект по теме «Анализ результатов A/B-теста»

Проверка гипотезы о равенстве долей

Одна из типичных задач статистики — проверка гипотез для пропорции, или доли. Если некоторая доля генеральной совокупности обладает каким-то признаком, а другая её часть — нет, по выборке из неё можно судить об этой доле. Как и в случае со средним, выборочные доли будут нормально распределены вокруг настоящей доли. Стандартного теста в базовых библиотеках Python нет, напишем его сами.

Исследуемой статистикой будет разница между пропорциями, наблюдаемыми на выборках. Можно доказать, что она будет распределена нормально:

$$Z \cong \frac{(P_1 - P_2) - (\pi_1 - \pi_2)}{\sqrt{P(1-P)(1/n_1 + 1/n_2)}} \sim N(0,1)$$

Величина `Z` — стандартная для критерия со стандартным нормальным распределением: со средним, равным нулю и стандартным отклонением, равным единице. Это указано в правой части формулы.

В формуле `n₁ и n₂` — размеры двух сравниваемых выборок, то есть количества наблюдений в них; `P₁, P₂` — пропорции, наблюдаемые в выборках; `P` — пропорция в выборке, скомбинированной из двух наблюдаемых; `π₁, π₂` — настоящие пропорции в сравниваемых генеральных совокупностях.

В A/B-тестировании чаще всего проверяют гипотезу о равенстве `π₁` и `π₂`. Тогда при верной нулевой гипотезе выражение `(π₁ - π₂)` в числителе будет равно нулю, и критерий можно рассчитывать только по выборочным данным.

Поскольку полученная статистика будет распределена нормально, так можно проводить двусторонние и односторонние тесты. При той же нулевой гипотезе о равенстве пропорций двух генеральных совокупностей можно проверить альтернативные гипотезы о том, что 1) они просто неравны, или 2) одна пропорция больше или меньше другой.

```
from scipy import stats as st
import numpy as np
import math as mth

alpha = .05 # критический уровень статистической значимости

successes = np.array([78, 120])
trials = np.array([830, 909])

# пропорция успехов в первой группе:
p1 = successes[0]/trials[0]

# пропорция успехов во второй группе:
p2 = successes[1]/trials[1]

# пропорция успехов в комбинированном датасете:
p_combined = (successes[0] + successes[1]) / (trials[0] + trials[1])

#разница пропорций в датасетах
difference = p1 - p2
```

Считаем статистику в стандартных отклонениях стандартного нормального распределения:

```
#считаем статистику в ст.отклонениях стандартного нормального распределения
z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/trials[0] + 1/trials[1]))

#задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
distr = st.norm(0, 1)
```

Если бы пропорции были равны, разница между ними была бы равна нулю. Посчитаем, как далеко статистика уехала от нуля: какова вероятность получить такое отличие или больше. Так как распределение статистики нормальное, применим метод `cdf()`. Саму статистику возьмём по модулю методом `abs()` — чтобы получить правильный результат независимо от её знака. Это возможно, потому что тест двухсторонний. По этой же причине удваиваем результат:

```
#считаем статистику в ст.отклонениях стандартного нормального распределения
z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/trials[0] + 1/trials[1]))

#задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
distr = st.norm(0, 1)

p_value = (1 - distr.cdf(abs(z_value))) * 2

print('p-значение: ', p_value)

if (p_value < alpha):
    print("Отвергаем нулевую гипотезу: между долями есть значимая разница")
else:
    print("Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными")
```

Проверка данных на нормальность. Критерий Шапиро-Уилка

На практике многие переменные далеки от нормального распределения: в них есть выбросы, игнорировать которые нельзя.

Центральная предельная теорема гласит, что выборочные средние будут нормально распределены вокруг настоящего среднего генеральной совокупности (а пропорции, взятые из выборок — вокруг пропорции). Это верно и для распределений, содержащих сильные выбросы. Однако хорошо работает такой подход, если взять не один десяток выборок и делать выводы по ним. Если же выборка из интересующей совокупности одна, в неё может закрасться выброс, который существенно сдвинет картину и повлияет на результаты.

Сперва нужно научиться проверять по выборке гипотезу о том, что она взята из нормально распределённой генеральной совокупности.

Простой критерий — χ^2 (хи-квадрат). Находят сумму квадратов разниц между наблюдаемыми и ожидаемыми значениями, и делят их на ожидаемые значения:

$$\frac{\sum(O_i - E_i)^2}{E_i}$$

В этой формуле O — наблюдаемые значения, E — ожидаемые. Предполагается, что разница между ожидаемыми и наблюдаемыми значениями распределена нормально вокруг нуля: вероятность отклонений убывает в обе стороны от ожидаемых значений. Соответственно, такой критерий оказывается распределён как сумма из *n* квадратов стандартных нормальных распределений, где n — число наблюдений в выборке. Такое распределение и носит название **хи-квадрат**.

Хи-квадрат — распространённый критерий, однако для проверки распределения на нормальность лучше применить другой — **критерий Шапиро-Уилка**. Его плюс в том, что при

фиксированном уровне значимости он обладает бóльшей мощностью, чем хи-квадрат: чаще обнаруживает различия между распределениями, если они и правда есть.

Расчёт критерия Шапиро-Уилка встроен в стандартную библиотеку `scipy.stats`. роверим можно ли считать эту переменную `sample_1` нормально распределенной методом **st.shapiro(x)**:

```
from scipy import stats as st

alpha = .05 # критический уровень статистической значимости

results = st.shapiro(sample_1)
p_value = results[1] # второе значение в массиве результатов (с индексом 1) - p-value

print('p-значение: ', p_value)

if (p_value < alpha):
    print("Отвергаем нулевую гипотезу: распределение не нормально")
else:
    print("Не получилось отвергнуть нулевую гипотезу, всё нормально")
```

Непараметрический тест Уилкоксона-Манна-Уитни

В случае, когда в данных есть большие (по сравнению с нормальным распределением) выбросы, алгебраические метрики работают плохо. Они учитывают все значения, но при этом одно выбивающееся значение существенно влияет на результат.

Алгебраические критерии **критерий хи-квадрат** и **критерий Шапиро-Уилка**, проверяющие предположение о нормальности распределения исходных данных, называют **параметрическими**, так как они по выборке оценивают **параметры предполагаемого распределения**: среднее значение, например.

Рассмотрим тест, основанный на **структурном подходе**, или **непараметрический**. Метод, который будем применять для A/B-тестирования, называется **st.mannwhitneyu()** (U-критерий Манна-Уитни). Ключевая идея — проранжировать две выборки по порядку от меньшего к большему и сравнить ранги одних и тех же значений, попавших в обе выборки. **Ранг** — это место в упорядоченной выборке. Разница между рангами одних и тех же значений может одинакова, и такой сдвиг называют типичным. Значит, просто добавились значения, сдвинувшие все остальные. А нетипичные сдвиги по рангу рассматриваются как изменения. Сумма рангов таких сдвигов и выступает значением критерия. Чем он выше — тем больше вероятность, что для этих выборок интересующая нас величина различается.

Вероятности получения разных значений критерия Манна-Уитни рассчитаны теоретически, что даёт возможность делать вывод о различии или его отсутствии для любого заданного уровня значимости.

Главное отличие непараметрических методов в том, что они работают с рангами, — номерами значений в упорядоченном ряду, — никак не учитывая сами значения. Поэтому к ним прибегают тогда, когда работа с самими значениями невозможна из-за выбросов, сильно сдвигающих параметрические результаты.

```
from scipy import stats as st

alpha = .05 # критический уровень статистической значимости

results = st.mannwhitneyu(data1, data2)

print('p-значение: ', results.pvalue)
```

```
if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу: разница статистически значима")
else:
    print("Не получилось отвергнуть нулевую гипотезу, вывод о различии сделать нельзя")
```

Стабильность кумулятивных метрик

Чтобы исключить проблему подсматривания, анализируют графики метрик.

Изучают графики **кумулятивных**, или накапливаемых данных. Например, тест шёл 14 дней. Если построить график по кумулятивным данным, в точке первого дня будут значения метрик за этот день, в точке второго дня — сумма метрик за два дня, в точке третьего — сумма за три. Так отслеживают изменения результатов эксперимента на каждый день тестирования.

По центральной предельной теореме значение кумулятивных метрик часто сходится и устанавливается около некоторого среднего. Тогда по графику кумулятивных метрик определяют, стоит останавливать тест или нет.

Чтобы нагляднее увидеть разницу между группами, строят **график относительного различия**. Каждую его точку рассчитывают так: `кумулятивная метрика группы B / кумулятивная метрика группы A - 1`.

Для расчёта кумулятивных данных пригодятся функции **np.logical_and()**, **np.logical_or()**, **np.logical_not()**. Она позволяет применить булевы операции к объектам *Series*. Это удобно, когда из таблицы нужно взять подмножество строк, удовлетворяющее нескольким условиям.

```
np.logical_and(first_condition, second_condition)
np.logical_or(first_condition, second_condition)
np.logical_not(first_condition)
```

Пусть в нашем распоряжении имеются датафреймы `orders` и `visitors`, содержащие информацию о заказах и посетителях интернет-магазина, полученные в рамках A/B-тестирования. Получим новый датасет, содержащий кумулятивные данные

```
# получаем агрегированные кумулятивные по дням данные о заказах
ordersAggregated = datesGroups.apply(lambda x: orders[np.logical_and(orders['date'] <=
x['date'], orders['group'] == x['group'])].agg({'date' : 'max', 'group' : 'max',
'orderId': pd.Series.nunique, 'userId' : pd.Series.nunique, 'revenue' : 'sum'}),
axis=1).sort_values(by=['date','group'])

# получаем агрегированные кумулятивные по дням данные о посетителях интернет-магазина
visitorsAggregated = datesGroups.apply(lambda x: visitors[np.logical_and(visitors['date']
<= x['date'], visitors['group'] == x['group'])].agg({'date' : 'max', 'group' : 'max',
'visitors' : 'sum'}), axis=1).sort_values(by=['date','group'])

# объединяем кумулятивные данные в одной таблице и присваиваем ее столбцам понятные
# названия
cummulativeData = ordersAggregated.merge(visitorsAggregated, left_on=['date', 'group'],
right_on=['date', 'group'])
cummulativeData.columns = ['date', 'group', 'orders', 'buyers', 'revenue', 'visitors']
```

Построим графики кумулятивной выручки по дням и группам A/B-тестирования:

```
import matplotlib.pyplot as plt

# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе A
cummulativeRevenueA = cummulativeData[cummulativeData['group']=='A'][['date','revenue', 'orders']]

# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе B
cummulativeRevenueB = cummulativeData[cummulativeData['group']=='B'][['date','revenue', 'orders']]
```

```
# Строим график выручки группы A
plt.plot(cummulativeRevenueA['date'], cummulativeRevenueA['revenue'], label='A')

# Строим график выручки группы B
plt.plot(cummulativeRevenueB['date'], cummulativeRevenueB['revenue'], label='B')

plt.legend()
```

Построим графики среднего чека по группам — разделим кумулятивную выручку на кумулятивное число заказов:

```
plt.plot(cummulativeRevenueA['date'], cummulativeRevenueA['revenue']/cummulativeRevenueA['orders'], label='A')
plt.plot(cummulativeRevenueB['date'], cummulativeRevenueB['revenue']/cummulativeRevenueB['orders'], label='B')
plt.legend()
```

Построим график относительного различия для среднего чека. Добавим горизонтальную ось методом **axhline()**:

```
# собираем данные в одном датафрейме
mergedCummulativeRevenue = cummulativeRevenueA.merge(cummulativeRevenueB, left_on='date',
right_on='date', how='left', suffixes=['A', 'B'])

# строим отношение средних чеков
plt.plot(mergedCummulativeRevenue['date'], (mergedCummulativeRevenue['revenueB']/mergedCu
mmulativeRevenue['ordersB'])/(mergedCummulativeRevenue['revenueA']/mergedCummulativeRevenue['ordersA'])-1)

# добавляем ось X
plt.axhline(y=0, color='black', linestyle='--')
```

Анализ выбросов и всплесков: крайние значения данных

Одна из трудностей, возникающих при анализе результатов A/B-теста — аномалии и выбросы, которые могут искажать результаты. **Аномалия** — такое значение, которое бывает в генеральной совокупности редко, но его попадание в выборку способно внести погрешность.

Для анализа аномалий полезно изучить гистограмму и диаграмму рассеивания.

Если делить упорядоченную выборку не на 4 части (как мы делали для получения квартиля), а на 100 — получим **перцентиль**. Аналогично квартилю N-й выборочный перцентиль — значение, больше которого определённая доля элементов выборки. В статистике N-й персентиль — значение, которое случайная величина не превышает с заданной вероятностью.

Для подсчёта перцентилей применяют метод **percentile()** библиотеки *Numpy*:

```
# values - диапазон значений
# percentiles - массив перцентилей, которые нужно вычислить

import numpy as np
np.percentile(values, percentiles)
```

Для определения аномалий нужно проанализировать значения 95, 97.5 и 99 перцентилей.

Основные ошибки при анализе A/B-тестов

Некорректное деление трафика теста

Распространённая проблема. Например, пользователи распределились между сегментами неравномерно или не в соответствии с указанными долями, либо когда структура трафика различная.

Забывание про статистическую значимость

Часто решение о различии в результатах теста принимают только на основе относительного изменения. Все неверные решения в долгосрочной перспективе ведут к потере или недополучению денег бизнесом.

"Подглядывание"

О ней мы говорили раньше. Не допускайте её и другим не давайте. К аналитикам часто приходят с промежуточными результатами и требуют принять решение — не поддавайтесь!

Слишком маленькая выборка

Приходят и просят запустить A/B-тест на 10-20 пользователей. Можно, конечно, согласиться. Только доверять результатам будет нельзя. Ни значимости, ни точности — слишком велико влияние каждого отдельного наблюдения.

Слишком малое время теста

Вот вы посчитали калькулятором необходимый размер выборки, запустили тест, собрали выборку и приняли решение. Вы уже видели, как сильно могут колебаться результаты теста в реальной среде. Принимайте решение, только когда уверены в результатах.

Слишком длительный тест

Не уходите и в другую крайность: иногда результаты ещё не установились, колеблются, но при этом дальнейшее проведение эксперимента не отразится на принятии решения, и не имеет смысла.

Отсутствие анализа аномалий

Помните об аномалиях и анализируйте результаты только с их учётом.

В реальных данных аномалий значительно больше, чем вы увидели в курсе. С опытом вы соберёте их коллекцию и научитесь их мастерски отыскивать.

Пренебрежение поправками к статистической значимости при множественном сравнении

Чем больше групп в тесте, тем чаще хотя бы при одном сравнении получается ложнопозитивный результат.