KARPOV.COURSES >>>
КОНСПЕКТ

> Конспект > 11 урок > СТАТИСТИКА

> Оглавление

> Критерий Хи-квадрат для одной случайной величины

Критерий Хи-квадрат (или критерий согласия Пирсона) используется для проверки того, соответствует ли категориальная случайна величина выбранному распределению. Есть всего два важных условия:

1. Все наблюдения независимы
2. Количество наблюдений в каждой ячейке больше 5

Рассмотрим пример с определением того, честная ли у нас игральная кость. Пусть мы сделали 600 бросков игральной кости и записали в таблице под номером каждой грани количество раз, которое она выпала:

номер грани	1	2	3	4	5	6
кол-во выпадений	70	110	20	300	40	60

Чтобы использовать критерий, для начала необходимо сформулировать предположение о честности игральной кости (гипотезу H_0) и выразить его в виде вероятностного распределения. Данное предположение может быть описано равномерным распределением - то есть таким, где все исходы равновероятны.

Теперь необходимо посчитать, сколько выпадений каждой грани мы ожидаем при справедливости нашего предположения. Если вероятность получить каждую грань одинакова, то и ожидаем мы в итоге получить одинаковое количество выпадений каждой грани. То есть:

номер грани	1	2	3	4	5	6
кол-во выпадений	100	100	100	100	100	100

После того как ожидаемое количество выпадений посчитано, мы можем посчитать статистику критерия Хи-квадрат:

$$\chi^2 = \sum_{i=1}^{n} \frac{(N_i - N_{p_i})^2}{N_{p_i}}$$

В данном выражении n - это количество граней, N_i - это количество выпадений i-той грани, N - это общее количество бросков, а p_i - это вероятность выпадения грани i при справедливости нашего предположения. Соответственно, N_{p_i} - это ожидаемое количество выпадений i-той грани.

Данная величина принадлежит к распределению Хи-квадрат с $n-1$ степенью свободы. Что это такое? Такое распределение возникает на основе суммы квадратов n независимых величин из стандартного нормального распределения (да, всё настолько запущено).

Стоит напомнить, что количество степеней свободы в нашем случае не 6, а 5, так как, зная количество бросков и то, сколько раз выпала каждая из граней, мы можем без труда восстановить, сколько раз выпала 6 грань.

Осталось довести задачу до конца. Считаем статистику Хи-квадрат:

$$\chi^2 = \frac{(70-100)^2}{100} + \frac{(110-100)^2}{100} + \frac{(20-100)^2}{100} + \frac{(300-100)^2}{100} + \frac{(40-100)^2}{100} + \frac{(60-100)^2}{100} = 9 + 1 + 64 -$$

По таблице значений (или через Python, об этом ниже) ищем значение под количество степеней свободы и уровень значимости, который мы хотим проверить. В нашем случае уровень значимости 0.99 и соответствующее значение квантили - 23.2. Мы видим, что значение статистики очень сильно превосходит значение 99-процентной квантили, а значит, наверняка с костью что то не так.

Реализация в Python

В scipy функция `chisquare` проводит этот тест, принимая на вход наблюдаемые и ожидаемые в соответствии с нулевой гипотезой значения величины. В случае с игральной костью наблюдаемые значения - значения из первой таблицы, а ожидаемые - из второй. Документация

Также в Python можно посчитать просто значение функции распределения Хи-квадрат в заданной точке при заданном количестве степеней свободы - делается это вот этой функцией.

> Таблицы сопряжённости

Пусть у нас есть приложение для заказа еды, которое работает в Москве и в Питере. Мы хотим посмотреть на состав нашей аудитории по городам и по операционной системе телефона. На основе наших данных мы можем построить следующую таблицу:

	iOS	Android	Всего:
Москва	1000	4000	5000
Санкт-Петербург	700	1200	1900
Всего:	1700	5200	6900

Такая таблица называется **таблицей сопряженности**. Попробуем ее проанализировать. Пусть мы хотим проверить, зависит ли доля пользователей iOS от города. Наша гипотеза H_0: распределение не зависит от города.

Вопрос: Какие будут ожидаемые значения для нашей нулевой гипотезы?

Ответ: Давайте рассчитаем. Нулевая гипотеза утверждает, что доля iOS-пользователей не зависит от города, следовательно, доля iOS-пользователей в отношении ко всем пользователям города должна быть одинаковой. Доля пользователей iOS всего составляет $1700/6900 = 24.64\%$, а доля пользователей Android - $5200/6900 = 75.36\%$. Следовательно, в Москве должно быть $5000 * 0.2464 = 1232$ пользователя iOS и $5000 * 0.7536 = 3768$ пользователей Android. Проделывая аналогичные вычисления для Питера, получаем таблицу с ожидаемыми значениями:

	iOS	Android
Москва	1232	3768
Санкт-Петербург	468.12	1431.84

Далее мы просто считаем статистику Хи-квадрат по формуле из прошлой секции.

$$\chi^2 = \frac{(1000-1232)^2}{1232} + \frac{(4000-3768)^2}{3768} + \frac{(700-468.12)^2}{468.12} + \frac{(1200-1431.84)^2}{1431.84} = 48.69 + 14.28 + 114.81 -$$

Вопрос: Сколько у распределения нашей статистики степеней свободы? Иными словами, сколько независимых случайных величин в таблице сопряженности (за случайные величины мы считаем только те, что представлены в четырех центральных ячейках таблицы)?

Ответ: Всего одна степень свободы, так как, зная значение любой из четырех центральных ячеек таблицы, мы можем восстановить остальные 3. Более общая формула для расчета степеней свободы выглядит так:

$$df = (n-1)(m-1)$$

где n - это количество строк, а m - это количество столбцов.

Данное значение не укладывается в 99-процентный доверительный интервал для данного распределения, следовательно, можно сделать вывод, что распределение операционных систем по городам статзначимо различается.

Реализация в Python

Помимо того, что это можно было сделать формулой одновыборочного критерия хи-квадрат, есть специальная функция `chi2_contingency`. Она принимает на вход табличку сопряжённости (можно посчитать её самостоятельно, а можно воспользоваться функцией pd.crosstab). Документацию можно прочесть тут.

Ну и также аналоги этого есть в библиотеке `pingouin` ;) Раздел Contingency как раз посвящён таблицам сопряжённости.

> Кодирование фиктивными переменными

По уроку с регрессионным анализом вы можете помнить, что в минипроекте мы преобразовывали категориальные переменные странной функцией pd.get_dummies(). В лекции была демонстрация похожего подхода силами библиотеки scikit-learn, и там оно носило название One-hot Encoding. Но что это за подход такой и зачем он вообще нужен? Давайте посмотрим поподробнее.

Представьте себе, что у нас есть данные по кошечкам и собачкам. У нас есть переменная, обозначающая их принадлежность тому или иному виду, и выглядит она так:

Животное
"кошка”
"собака”
"собака”
"кошка”

One-hot Encoding сделает из этой переменной вот такое:

кошка	собака
1	0
0	1
0	1
1	0

Что здесь произошло:

1. На каждую категорию создалось по своей переменной
2. Если данное животное было кошкой, то в столбце "кошка" ставится 1, а в столбце "собака" ставится 0 (и наоборот)

То есть у нас создались новые переменные, обозначающие принадлежность (1) или не-принадлежность (0) объекта к определённой категории! Их называют "фиктивными" или dummy-переменными.

Для большего числа категорий идея та же самая:

Животное
"кошка”
"собака”

Животное
"крыса”
"кошка”

И её "фиктивная" репрезентация:

кошка	собака	крыса
1	0	0
0	1	0
0	0	1
1	0	0

> Ловушка фиктивных переменных

Давайте ещё раз взглянем на нашу табличку:

кошка	собака
1	0
0	1
0	1
1	0

Какие из этих переменных поместить в регрессию? Первая мысль, которая приходит в голову - поместить обе, нам же нужна информация и о собаках, и о кошках! Но присмотритесь внимательно: **информация дублируется**! Чтобы понять, где у нас кошки, а где собаки, нам нужна лишь одна из этих двух переменных:

кошка
1
0
0
1

Либо:

собака
0
1
1
0

Как видите, мы и так прекрасно можем понять, где какое животное! Соответственно, для модели нам нужна лишь одна из этих переменных. Более того: если мы включим обе переменные в модель, то она просто сломается! В этом заключается **ловушка фиктивных переменных**. Почему так происходит?

Давайте вспомним, в чём суть множественной регрессии. С помощью неё мы по нескольким признакам пытаемся предсказать значение зависимой переменной. При этом регрессия пытается оценить уникальный вклад каждой переменной в это предсказание - и если переменные не коррелируют друг с другом, ей это сделать легко.

Если переменные сильно коррелируют, то они ведут себя похожим образом. В результате регрессии становится сложнее оценить, какой вклад принадлежит какой переменной! Вы можете помнить, что такая ситуация называется **мультиколлинеарностью** и приводит к завышению стандартных ошибок регрессии - и, соответственно, плохой оценке доверительных интервалов и статзначимости.

Что же с нашими фиктивными переменными? Между ними **идеальная отрицательная корреляция** - в результате возникает **идеальная мультиколлинеарность**. В такой ситуации регрессия просто неспособна оценить уникальный вклад

каждой переменной, ведь они двойники друг друга! Соответственно, регрессия либо откажется считаться в принципе, либо выдаст нам абсолютно случайные значения коэффициентов. Ни то, ни другое нам не нужно.

Поэтому стандартной практикой является выкидывание одной из получившихся фиктивных переменных. Это и делала опция `drop_first=True` в `pd.get_dummies()` - она выкидывала первую из фиктивных переменных. Но по факту выкинуть можно любую. Как это повлияет на интерпретацию - расскажем в следующем шаге.

Аналогично работает и для большего числа категорий. Вот так у нас снова будет идеальная мультиколлинеарность:

кошка	собака	крыса
1	0	0
0	1	0
0	0	1
1	0	0

А вот так всё посчитается нормально:

собака	крыса
0	0
1	0
0	1
0	0

> Как интерпретировать фиктивные переменные?

Давайте вспомним стандартную формулу линейной регрессии:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + ...\beta_n X_n$$

Перепишем её для ситуации с тремя животными - кошкой, собакой и крысой. Предположим, что мы пытаемся предсказать **вредность животного** по тому, к какому виду оно относится. Тогда у нас может выйти следующее уравнение:

$$\text{вредность} = 20.15 - 10.8(\text{собака}) + 2.5(\text{крыса})$$

Что будет, если нам досталась собака? Подставляем соответствующие значения фиктивных переменных:

$$\text{вредность} = 20.15 - 10.8 * 1 + 2.5 * 0 = 20.15 - 10.8 = 10.7$$

Обратите внимание, что у нас фактически остался только "собачий" коэффициент, а всё остальное убралось благодаря нулю! В результате мы получили **среднее значение вредности для собак**. Аналогично для крыс:

$$\text{вредность} = 20.15 - 10.8 * 0 + 2.5 * 1 = 20.15 + 2.5 = 22.65$$

Вопрос: а что тогда отражает у нас коэффициент β_0, равный 20.15? А это **среднее значение вредности для кошек**. Как видите, выкидывание "кошачьей" переменной не привело к потере информации - просто эта информация оказалась в коэффициенте β_0.

Соответственно, коэффициенты при фиктивных переменных надо интерпретировать следующим образом: **как изменится среднее значение переменной относительно "выкинутой" категории**. В нашем случае средняя вредность собак составляет **-10.8 единиц относительно средней вредности кошек**, а средняя вредность крыс составляет **2.5 единиц относительно вредности кошек**. Если бы мы выкинули не кошек, а собак или крыс - уравнение и его интерпретация изменились бы соответственно.

В принципе, мы можем и не считать коэффициент β_0, как это было сделано в лекции. Интерпретация будет той же самой, просто в таком случае нам остаётся довольствоваться относительными значениями коэффициентов - посчитать средние для каждой категории у нас не выйдет.

NB! Если вы дочитали до этой части конспекта, то вы могли бы задуматься: а разве это не похоже на **дисперсионный анализ**? Если бы у нас были только кошки и собаки, не похоже ли это было бы на **t-тест**?

Ответ простой: да, похоже! Более того - регрессия с фиктивными переменными и дисперсионный анализ/t-тест **математически эквивалентны**. Это главная тайна прикладной статистики, часто скрываемая от студентов - но дающая нам невероятную гибкость анализа, когда мы понимаем идущие от неё следствия.

Есть классический пост, демонстрирующий эту эквивалентность (либо близость к ней) для целого ряда тестов - и да, хи-квадрат тоже сюда попадает! Изначально эта демонстрация сделана на R, но есть её порт на Python. Рекомендуем взглянуть на оба для полноты взгляда ;)

> Дополнительные материалы

1. Применение Хи-квадрата в А/Б тестировании для расчета конверсии
2. Применение Хи-квадрата в А/Б тестировании
3. Материалы по непараметрической статистике