МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М. В. ЛОМОНОСОВА

Механико-математический факультет

экономический поток

Статистический практикум

3 курс, группа 332

6 семестр

Лектор и семинарист
А.А. Муромская
«___» ________ 2021 г.

Москва, 2021 г.

Техническая информация

Данный PDF содержит основную информацию и факты весеннего семестра 3 курса по предмету "Статистический практикум".

Собрали и напечатали по мотивам лекций и семинаров студенты 3-го курса Конов Марк и Гащук Елизавета.

Добавления и исправления принимаются на почты vkonov2@yandex.ru и gashchuk2011@mail.ru.

ПРИЯТНОГО ИЗУЧЕНИЯ

Содержание

1

Базовые понятия

1.1 Python

1.1.1 Преамбула

import numpy as np
import scipy as sp
import scipy.stats as st
from scipy.stats import norm (нормальное распределение)
from scipy.stats import uniform (равномерное распределение)
from scipy.stats import expon (экспоненциальное распределение)
from scipy.stats import beta (бета-распределение)
from scipy.stats import cauchy (распределение Коши)
from scipy.stats import t (распределение Стьюдента)

1.1.2 Функции

Функция плотности - pdf (pdf - probability density function)
Плотность в точке $x = 0.1$ распределения $N(2, 9)$:
norm.pdf(0.1, 2, 3)

Функция распределения - cdf (cdf - cumulative density function)
Функция распределения в точке $x = 0.3$ распределения $N(2, 9)$:
norm.cdf(3.5, 2, 3)

Квантиль - ppf (ppf - pension protection fund)
Квантиль уровня 0.5 распределения $N(2, 9)$:
norm.ppf(0.5, 2, 3)

Выборочное среднее - mean (mean - среднее)
Выборочное среднее выборки:

sample.mean()

Выборочное среднее - mean (std - standart deviation)
Среднеквадртическое отклонение выборки:
sample.std()
np.std(data, ddof = 1) - исправленное ср. кв. отклонение

Дисперсия - mean (var - variance)
Дисперсия выборки:
sample.var()

Логарифм плотности - logpdf (logpdf - logarithm of probability density function)
Логарифм функции плотности распределения $N(0, 1)$:
norm.logpdf()

1.1.3 Генерация выборок

Генерация выборок из $N(0, 1)$
Выборка объема 1000 из $N(0, 1)$:

- **sample = np.random.randn(1000)**
- **sample = norm.rvs(size=1000)**

Генерация выборок из $N(a, \sigma^2)$
Выборка объема 1000 из $N(2, 9)$:

- **sample = np.random.randn(1000)*3+2**
- **sample = norm.rvs(2, 3, size=1000)**

Генерация выборок из $R[0, 1]$
Выборка объема 1000 из $R[0, 1]$:

- **sample = np.random.rand(1000)**
- **sample = uniform.rvs(size=1000)**

Генерация выборок из $R[loc, loc + scale]$
Выборка объема 1000 из $R[loc, loc + scale]$:

- **sample = uniform.rvs(loc=1, scale=3, size=1000)**

Генерация выборок из $Exp[1]$
Выборка объема 1000 из $Exp[1]$:

- **sample = expon.rvs(size = 10000)**

Генерация выборок из Бета-распределения
Выборка объема 1000 из Бета-распределения(6,2):

- alpha = 6, bheta = 2
  sample = beta.rvs(alpha, bheta, size = 1000)

Генерация выборок из распределения Коши
Выборка объема 1000 из распределения Коши:

- **sample = cauchy.rvs(size = 1000)**

Генерация выборок из распределения Стьюдента
Выборка объема 1000 из распределения Стьюдента:

- **sample = t.rvs(size = 1000)**

1.1.4 Обработка

Чтение из файла
Считывание данных из файла File.txt в DataFrame data:

- **data = pd.read_csv(«File.txt»)**

Вывод нескольких строк DataFrame
Вывод каждой второй строки в диапазоне от 30 до 35 строки DataFrame data:

- **data.loc[30:35:2]**

Получение информации о DataFrame
Получение информации о DataFrame data:

- **data.describe**

Выделение одного столбца DataFrame
Выделение столбца x из DataFrame data в массив x:

- **x = data['x']**

Удаление элементов из массива
Удаление из массива x элементов, равных нулю:

- **x = x[x != 0]**

Удаление из массива x элементов, меньших 113:

- **x = x[x > 113]**

1.2 Статистика

1.2.1 Проверка гипотез

Дана выборка $X = (X_1, \dots, X_n)$. Параметр θ неизвестен.
$H_0 : \theta = \theta_0$ - гипотеза.

$$H_1 : \begin{cases} \theta < \theta_0 \text{ - левосторонняя альтернатива} \\ \theta > \theta_0 \text{ - правосторонняя альтернатива} \\ \theta \neq \theta_0 \text{ - двусторонняя альтернатива} \\ \theta = \theta_1 \text{ - простая альтернатива} \end{cases}$$

Статистика $T(X)$ зависит от θ_0, должны знать распределение T, если верна H_0. $C_{\text{кр}}$ строится по квантилям распределения T, если верна H_0.

- левостороння альтернатива $\Rightarrow C_{\text{кр}} = (-\infty; X_\alpha)$
- правостороння альтернатива $\Rightarrow C_{\text{кр}} = (X_{1-\alpha}; +\infty)$
- двустороння альтернатива $\Rightarrow C_{\text{кр}} = (-\infty; X_{\frac{\alpha}{2}}) \bigcup (X_{\frac{1-\alpha}{2}}; +\infty)$

Правило.

$$\begin{cases} T_{\text{реал}} \in C_{\text{кр}} \;\Rightarrow\; \textit{отвергаем } H_0,\ \textit{принимаем } H_1 \\ T_{\text{реал}} \notin C_{\text{кр}} \;\Rightarrow\; \textit{принимаем } H_0 \end{cases}$$

- левостороння альтернатива $\Rightarrow pvalue = P(T \leq T_{\text{реал}} | H_0)$
- правостороння альтернатива $\Rightarrow pvalue = P(T \geq T_{\text{реал}} | H_0)$
- двустороння альтернатива $\Rightarrow pvalue = 2min\left(P(T \leq T_{\text{реал}} | H_0), P(T \geq T_{\text{реал}} | H_0)\right)$

Правило.

$$\begin{cases} pvalue < \alpha \;\Rightarrow\; \textit{отвергаем } H_0,\ \textit{принимаем } H_1 \\ pvalue > \alpha \;\Rightarrow\; \textit{принимаем } H_0 \end{cases}$$

2
Проверка гипотез о параметрах одной выборки

2.1 Гипотеза о мат. ожидании нормального распределения

2.1.1 Z-test (1 sample) (известная дисперсия)

Теория

Применяется в предположении, что выборка нормальна:
$X = (X_1, \ldots, X_n),\ X_i \sim N(a, \sigma^2)$
Дисперсия σ^2 известна, строим гипотезу про a: $H_0 : a = a_0$, альтернатива любая.

$$Z = \frac{\overline{X} - a_0}{\frac{\sigma}{\sqrt{n}}} \overset{\mathrm{H}_0}{\sim} N(0, 1)$$

Критическое множество зависит от альтернативы.

2.1.2 t-test (1 sample) (неизвестная дисперсия)

Теория

Применяется в предположении, что выборка нормальна:
$X = (X_1, \ldots, X_n),\ X_i \sim N(a, \sigma^2)$
Дисперсия σ^2 неизвестна, строим гипотезу про a: $H_0 : a = a_0$, альтернатива любая.

$$t = \frac{\overline{X} - a_0}{\frac{S}{\sqrt{n}}} \overset{\mathrm{H}_0}{\sim} t(n-1), \quad S = \sqrt{S^2} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2}$$

Критическое множество зависит от альтернативы.

Python

```
alpha = .05
n = len(data)
data_mean = np.mean(iq)
data_s = np.std(data, ddof=1)
t_stat = (data_mean - 110) * np.sqrt(n) / data_s
iq_crit = t.ppf(1-alpha, n-1)
t_p_value = 1 - t.cdf(t_stat, n-1)
print ("{0:.7f}".format(t_p_value))
```

Встроенная реализация Параметр **popmean** $= a_0$. Тест для двусторонней альтернативы, чобы получить односторонний pvalue, делим полученный pvalue на 2.

```
ans = ttest_1samp(iq, popmean=110)
pvalue = ans[1]
```

Пример

Задача 2.1. *Оцениваем средний уровень IQ профессоров университета города N. Можно ли утверждать на уровне значимости 10%, что средний уровень IQ профессоров выше 110 баллов? Данные в файле IQ.txt (решить задачу в предположении нормальности данных).*

Решение.

1 способ

```
import numpy as np
import pandas as pd
from scipy.stats import t
iq_data = pd.read_csv("IQ.txt")
iq = iq_data['iq']
alpha = .1
n = len(iq)
iq_mean = np.mean(iq)
iq_s = np.std(iq, ddof=1)
t_stat = (iq_mean - 110) * np.sqrt(n) / iq_s
iq_crit = t.ppf(1-alpha, n-1)
t_p_value = 1 - t.cdf(t_stat, n-1)
print("{0:.7f}".format(t_p_value))

```

```
import numpy as np
import pandas as pd
from scipy.stats import ttest_1samp
iq_data = pd.read_csv("IQ.txt")
iq = iq_data['iq']
ans = ttest_1samp(iq, popmean=110)
pvalue = ans[1]/2
print("{0:.7f}".format(pvalue))

```

■

Задача 2.2. *В городе Ивановск проведено выборочное исследование доходов жителей. По выборке из 500 человек получено среднее 23800 руб. и среднее квадратическое отклонение 400 руб. Можно ли утверждать на уровне значимости 5%, что средний доход жителей составляет менее 25000 руб? Решить задачу в предположении нормальности данных.*

Решение.

```python
import numpy as np
import pandas as pd
from scipy.stats import t
alpha = .05
n = 500
mean_ivan = 23800
s_ivan = 400 * np.sqrt(n) / np.sqrt(n-1)
t_statistics = (mean_ivan - 25000) * np.sqrt(n) / s_ivan
crit_ivan = t.ppf(alpha, n-1)
ivan_p_value = t.cdf(t_statistics, n-1)
print("{0:.7f}".format(ivan_p_value))

```

■

2.2 Гипотеза о дисперсии нормального распределения

2.2.1 Известное м.о.

Теория

Применяется в предположении, что выборка нормальна:
$X = (X_1, \ldots, X_n),\ X_i \sim N(a, \sigma^2)$
М.о. a известно, строим гипотезу про σ^2: $H_0 : \sigma = \sigma_0$, альтернатива любая.

$$T = \frac{\sum\limits_{i=1}^{n}(X_i - a)^2}{\sigma_0^2} \overset{\mathrm{H_0}}{\sim} \chi^2(n)$$

Критическое множество зависит от альтернативы.

2.2.2 Неизвестное м.о.

Теория

Применяется в предположении, что выборка нормальна:
$X = (X_1, \ldots, X_n),\ X_i \sim N(a, \sigma^2)$

М.о. a неизвестно, строим гипотезу про σ^2: $H_0 : \sigma = \sigma_0$, альтернатива любая.

$$T = \frac{(n-1)S^2}{\sigma_0^2} \overset{H_0}{\sim} \chi^2(n-1), \quad S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2$$

Критическое множество зависит от альтернативы.

Пример

Задача 2.3. *Партия изделий принимается, если дисперсия размеров не превышает 0.2. Исправленная выборочная дисперсия для 30 изделий оказалась равной 0.3. Можно ли принять партию на уровне значимости 5%? Решить задачу в предположении нормальности данных.*

Решение.

```
import numpy as np
import pandas as pd
from scipy.stats import chi2
alpha = .05
n = 30
s2_izd = 0.3
T = (n-1) * s2_izd / (0.2)
crit_izd = chi2.ppf(1 - alpha, n-1)
p_value_izd = 1 - chi2.cdf(T, n-1)
print("{0:.7f}".format(p_value_izd))

```

■

2.3 Гипотеза о параметрах гамма-распределения

2.3.1 Критерий Вальда

Теория

Применяется в предположении, что имеем выборку с гамма-распределение:
$X = (X_1, \ldots, X_n), \quad X_i \sim Pois(\lambda)$
Строим гипотезу про $EX_i = \lambda$: $H_0 : \lambda = \lambda_0$, альтернатива любая. По ЦПТ получаем:

$$T = \frac{\sum_{i=1}^{n} X_i - n\lambda_0}{\sqrt{n\lambda_0}} \overset{H_0}{\sim} N(0,1)$$

Критическое множество зависит от альтернативы.

Существует асимптотический критерий. **Строим гипотезу про $EX_i = \mu$**: $H_0 : \mu = \mu_0$, альтернатива любая. По лемме Слуцкого получаем:

$$T = \frac{\sum\limits_{i=1}^{n} X_i - n\mu_0}{\sqrt{nS^2}} \overset{\mathrm{H}_0}{\sim} N(0,1), \quad S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2$$

Критическое множество зависит от альтернативы.

Если $|T(X)| > Z_{1-\alpha/2}$, то гипотеза H_0 отвергается, иначе H_0 принимается.

Пример

Задача 2.4. *Рассмотрим гамма-распределение с плотностью* $f(x) = \frac{\theta^{-\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{\frac{-x}{\theta}}$, $x \geq 0$. *Гипотеза* H_0: $\alpha\theta = E[X_1] = 1$. *Альтернатива* H_1: $E[X_1] \neq 1$.

Решение. Асимтотическое решение:

```
import numpy as np
import pandas as pd
from scipy.stats import gamma, norm
alpha = .05

def calc_wald_statistics(X, assumed_mean):
    X = np.array(X)
    n = len(X)
    return (X.sum() - n * assumed_mean) / np.sqrt(n * X.var(ddof=1))

norm_threshold = norm.ppf(1.0 - 0.5 * alpha)
h0_cws = calc_wald_statistics (gamma.rvs(a = 1, scale = 1, size=2000), 1)
p_value = 2*np.min([ norm.cdf(h0_cws), 1 - norm.cdf(h0_cws) ])
print("{0:.7f}".format(p_value))

```

P.S. a - это параметр альфа, scale - параметр тета

Точное решение:

```
import numpy as np
import pandas as pd
from scipy.stats import gamma, norm
alpha = .05
samples_count = 1000
iters_count = 10000
h0_a = 2.0
h0_scale = 0.5
h0_samples = gamma.rvs(a = h0_a, scale = h0_scale, size=(iters_count,
samples_count))

def calc_wald_statistics_multirow(X, samples_count, assumed_mean):
    X = np.array(X)[:, : samples_count]
    n = X.shape[1] (n=samples_count
    return (X.sum(axis=1) - n * assumed_mean) /np.sqrt(n * X.var(ddof=1, axis
=1))

```

```
    h0_stat_values = calc_wald_statistics_multirow(h0_samples, 1000, h0_a *
  h0_scale)
    ans = np.sum(np.abs(h0_stat_values) > norm_threshold) / float(iters_count)
    print("{0:.7f}".format(ans))

```

■

2.4 Гипотеза о параметрах распределения Коши

2.4.1 Метод выборочных квантилей

Теория

не нашел у себя

Пример

Задача 2.5. *Рассмотрим распределение Коши с плотностью* $f(x) = \frac{1}{\pi(1+(x-x_0)^2)}$. *Гипотеза* H_0*:* $x_0 = 0$*. Альтернатива* H_1*:* $x_0 \neq 0$*.*

Решение. Мы не можем здесь рассматривать статистику Вальда. Но знаем, что при H_0 статистика

$$T(X) = \frac{\sqrt{n}\widehat{z}_{0.5}}{\frac{\pi}{2}}$$

стремится к $N(0,1)$ с ростом количества элементов выборки. Тест снова устроен следующим образом:

Если $|T(X)| > z_{1-\alpha/2}$, то гипотеза H_0 отвергается, иначе H_0 принимается. Проведем тест с уровнем значимости $\alpha = 0.05$.

```
import numpy as np
import pandas as pd
from scipy.stats import cauchy

def calc_statistics(X):
    X = np.array(X)
    n=len(X)
    return 2 * np.sqrt(n) * np.percentile(X, 50, interpolation='lower') / (np
  .pi)

alpha = 0.05
norm_threshold = norm.ppf(1.0 - 0.5 * alpha)
h0_sample = cauchy.rvs(size = 1000)
h0_cs = calc_statistics (h0_sample)
h1_sample = cauchy.rvs(size = 1000, loc=2)
h1_cs = calc_statistics (h1_sample)

```

■

2.5 Непараметрические критерии о математическом ожидании

2.5.1 Одновыборочный критерий знаков

Теория

Применяется в предположении, что имеем выборку $Z = (Z_1, \dots, Z_n)$, удовлетворяющую следующим условия:

2 способ 1) все Z_i независимы

2) все Z_i получены из непрерывной совокупности с медианой θ:

$$P(Z_i < \theta) = P(Z_i > \theta) = \frac{1}{2},\ i = 1, 2, \dots, n$$

Строим гипотезу про медиану θ: $H_0 : \theta = \theta_0$, альтернатива любая. Модифицируем Z_i: $\tilde{Z}_i = Z_i - \theta_0$. Если $\tilde{Z}_i = 0$, то обрасываем этот элемент и уменьшаем n. Рассматриваем $\psi_i = \begin{cases} 1, & \tilde{Z}_i > 0 \\ 0, & \tilde{Z}_i < 0 \end{cases}$. $B = \sum\limits_{i=1}^{n} \psi_i \overset{\mathrm{H}_0}{\sim} Bin(n, \frac{1}{2})$ - статистика (число успехов в n испытаниях). В качестве квантилей для левосторонней и правосторонней альтернатив равны $X_\alpha - 1$ и $X_{1-\alpha} + 1$ соответственно.

Существует асимптотический критерий. Гипотезы и альтернативы аналогичные.

$$B^* = \frac{B - \frac{n}{2}}{\sqrt{\frac{n}{4}}} \overset{\mathrm{H}_0}{\sim} N(0, 1)$$

Пример

Задача 2.6. *В городе N проведены выборочные обследования доходов жителей. Проверить на уровне значимости 3% утверждение о том, что средняя зарплата жителей в городе N менее 40000 руб. Данные в файле City.txt.*

Решение.

```
import numpy as np
import pandas as pd
from scipy.stats import binom_test
from scipy.stats import binom
city = pd.read_csv("City.txt")
city = city['City']
z = city - 40000
z = z[z!=0]
b = sum(z > 0)
n = len(z)
sign_res=binom_test(b, n,  p=0.5)
```

```
    ans = sign_res/2.0
    ans = binom.cdf(b, n, 0.5)

```

Асимтотический критерий:

```
    import numpy as np
    import pandas as pd
    from scipy.stats import binom_test
    from scipy.stats import binom
    from scipy.stats import norm
    city = pd.read_csv("City.txt")
    city = city['City']
    z = city - 40000
    z = z[z!=0]
    b = sum(z > 0)
    n = len(z)
    b_star = (b - n*0.5) / np.sqrt(n*0.25)
    ans = norm.cdf(b_star, loc=0, scale=1)

```

■

2.5.2 Одновыборочный знако-ранговый критерий Вилкоксона

Теория

Wilcoxon signed-rank test

Применяется в предположении, что имеем выборку $Z = (Z_1, \ldots, Z_n)$, удовлетворяющую следующим условия:

1) все Z_i независимы

2) все Z_i получены из непрерывной и симметричной относительно θ совокупности

Строим гипотезу про медиану θ: $H_0 : \theta = \theta_0$, альтернатива любая. Модифицируем Z_i: $\tilde{Z}_i = Z_i - \theta_0$. Если $\tilde{Z}_i = 0$, то обрасываем этот элемент и уменьшаем n. Сортируем $|\tilde{Z}_1|, |\tilde{Z}_2|, \ldots, |\tilde{Z}_n|$ по возрастанию и присваиваем ранги, равные порядковому номеру элемента в последовательности. Если соседние элементы равны, то присваиваем им ранги, равные друг другу и среднему арифметическому их порядковых номеров. Рассматриваем $\psi_i = \begin{cases} 1, & \tilde{Z}_i > 0 \\ 0, & \tilde{Z}_i < 0 \end{cases}$. $T^+ = \sum\limits_{i=1}^{n} R_i \cdot \psi_i$

Существует асимптотический критерий. Гипотезы и альтернативы аналогичные. Применяется для $n > 20$.

$$T^* = \frac{T^+ - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}} \overset{\mathrm{H_0}}{\sim} N(0,1)$$

Пример

Задача 2.7. *В городе N проведены выборочные обследования доходов жителей. Проверить на уровне значимости 3% утверждение о том, что средняя зарплата жителей в городе N менее 40000 руб. Данные в файле City.txt.*

Решение. Асимтотический критерий:

```python
import numpy as np
import pandas as pd
from scipy.stats import wilcoxon
city = pd.read_csv("City.txt")
city = city['City']
z = city - 40000
z = z[z!=0]
signed_rank_res = wilcoxon(z)
ans = signed_rank_res.pvalue/2.0

```

■

3 Критерии согласия

3.1 Проверка произвольного распределения

3.1.1 Тест Колмогорова-Смирнова

Теория

Предполагается выборка $X = (X_1, \dots, X_n)$ с непрерывной функцией распределения F. Тест применяется при $n \geq 20$.
$H_0 : F = F_0$ - простая гипотеза. $H_1 : F \neq F_0$.

$$T(x) = \sqrt{n} \sup_x |\hat{F}_n(x) - F_0(x)| \xrightarrow[n\to\infty, H_0]{d} \xi$$

где ξ имеет распределение Колмогорова:

$$\text{ф.р. } K(t) = \begin{cases} \sum\limits_{j=-\infty}^{+\infty} (-1)^j e^{-2j^2t^2}, \ t > 0 \\ 0, t \leq 0 \end{cases}$$

$$P(\sqrt{n} \sup_x |\hat{F}_n(x) - F_0(x)| \leq t) \xrightarrow[n\to\infty]{H_0} K(t)$$

$$C_{\text{кр}} = [K_{1-\alpha}, +\infty]$$

Пример

Проверка на $N(0,1)$

```python
import numpy as np
import pandas as pd
import scipy.stats as stats
norm_sample = stats.norm.rvs(size=10000)
stats.kstest(norm_sample, stats.norm.cdf)
stats.kstest(norm_sample, 'norm')
```

Проверка на $N(1,4)$

```python
import numpy as np
import pandas as pd
```

```python
import scipy.stats as stats
def N_1_4_cdf(x):
    return stats.norm.cdf(x, loc=1, scale=2)

stats.kstest(norm_sample, N_1_4_cdf)
stats.kstest(norm_sample, lambda x: stats.norm.cdf(x, loc=1, scale=2))
```

3.1.2 Тест Андерсона-Дарлинга

Теория

(Ω^2-критерий)

Тест работает для $N(0,1)$, $Exp(1)$ и еще пары распределений. Тест выдает значение статистики, набор квантилей вида $x_{1-\alpha}$ и набор соответствующих значений α (в %).

$X = (X_1, \ldots, X_n)$ - непрерывная функция распределения F. $H_0 : F = F_0$ - простая гипотеза.

$$\Omega^2 = \int\limits_{-\infty}^{+\infty} \frac{\left(\hat{F}_n(x) - F_0(x)\right)^2}{F_0(x)(1 - F_0(x))} dF_0(x)$$

$$T(x) = n \cdot \Omega^2 \xrightarrow[H_0]{d} \xi,\ P(n\Omega^2 \leq x) \xrightarrow[H_0]{} A(x),\ A(Z_p) = p$$

Пример

Проверка на $N(0,1)$

```python
import numpy as np
import pandas as pd
import scipy.stats as stats
norm_sample = stats.norm.rvs(size=10000)
stats.anderson(norm_sample, 'norm')
```

Проверка на $Exp(1)$

```python
import numpy as np
import pandas as pd
import scipy.stats as stats
norm_sample = stats.norm.rvs(size=10000)
stats.anderson(exp_sample, 'expon')
```

3.1.3 Критерий Пирсона (хи-квадрат)

Теория

Полиномиальная схема: n независимых испытаний, r исходов $A_1, \ldots, A_r$. H_0 : вероятности исходов $p_1^0, \ldots, p_r^0$. Пусть исходы встретилиись $m_1, \ldots, m_r$ раз. Статисика:

$$\hat{\chi^2} = T(x) = \sum_{i=1}^{r} \frac{(m_i - np_i^0)^2}{np_i^0} \xrightarrow[H_0]{d} \chi^2(r-1)$$

$$C_{\text{кр}} = [\chi^2_{1-\alpha}(r-1), +\infty)$$

Тест применяется при $n \geq 50$ и $m_i \geq 5$.

Критерий χ^2-Фишера:

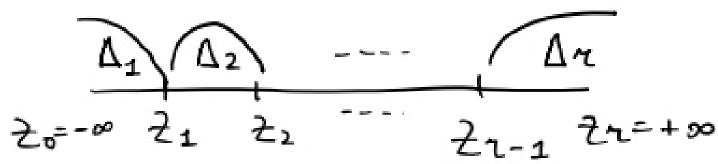
$$p_i^0(\theta_1, \dots, \theta_k),\ \theta = (\theta_1, \dots, \theta_k) \in \Theta$$

$$\hat{\theta} = arg\ \underset{\theta \in \Theta}{max} \prod_{j=1}^{r} \left(p_j^0(\theta)\right)^{m_j} \text{ – ОМП}$$

$$\hat{\chi^2} = \sum_{i=1}^{r} \frac{\left(m_i - np_i^0(\hat{\theta})\right)^2}{np_i^0(\hat{\theta})} \xrightarrow[H_0]{d} \chi^2(r-1-k), \text{ где } k \text{ - кол-во неизвестных параметров}$$

В качестве $A_1, \dots, A_r$ берем попадание $X_1, \dots, X_n$ в некоторые множества $\triangle_1, \dots, \triangle_r$. В каждом интервале ≥ 5 попаданий.

Пример 3.1 ((критерий Пирсона))**.** $X = (X_1, \dots, X_n)$, $H_0 : F = F_0$ - *простая гипотеза.*



m_i: число X_i в $\triangle_i$.
$p_i^0 = P_0(X_i \in \triangle_i) = F_0(Z_i) - F_0(Z_{i-1})$ *(если верна H_0)*

$$\hat{\chi^2} = \sum_{i=1}^{r} \frac{(m_i - np_i^0)^2}{np_i^0} \xrightarrow[H_0]{d} \chi^2(r-1)$$

Пример 3.2 ((критерий Фишера))**.** $X = (X_1, \dots, X_n)$, $H_0 : F = F_0(\theta_1, \dots, \theta_k)$.

$$\hat{\chi^2} = \sum_{i=1}^{r} \frac{\left(m_i - np_i^0(\hat{\theta}_1, \dots, \hat{\theta}_k)\right)^2}{np_i^0(\hat{\theta}_1, \dots, \hat{\theta}_k)} \xrightarrow[H_0]{d} \chi^2(r-1-k)$$

Пример

Проверка на $N(0,1)$

```python
import numpy as np
import pandas as pd
import scipy.stats as stats

def chisquare_normality_test(d, loc=None, scale=None,
    min_bin_value=-3, max_bin_value=3, nbins=17):
    :param d: array like -- initial data
    :param loc: loc parameter of norm distribution
    if loc is None then d.mean() is used
    :param scale: scale parameter of norm distribution
    if scale is None then d.std(ddof=0) is used
    :param min_bin_value: right bound of the first bin
    :param max_bin_value: left bound of the last bin
    :param nbins: number of bins

    bins = [-np.inf] + list(np.linspace(min_bin_value, max_bin_value,
        max(nbins-1, 2))) + [np.inf]

    if loc is None and scale is None:
        degrees_of_freedom = 2
    elif loc is not None and scale is not None:
        degrees_of_freedom = 0
    else:
        degrees_of_freedom = 1

    sf = np.histogram(d, bins)[0]

    loc = loc or d.mean()
    scale = scale or d.std(ddof=0)

    tf = [stats.norm.cdf(bins[i], loc=loc, scale=scale) -

        stats.norm.cdf(bins[i-1], loc=loc, scale=scale)
                    for i in range(1, len(bins))]
    tf = np.array(tf)*len(d)

    return stats.chisquare(sf, tf, ddof=degrees_of_freedom),
        stats.chisquare(sf, tf, ddof=0)

chisquare_normality_test(t_sample, nbins=27)
chisquare_normality_test(norm_sample, loc=0, scale=2)
```

3.2 Проверка на нормальность

3.2.1 Тест Шапиро-Уилка

Теория

Это наиболее мощный критерий проверки нормальности: $P(H_1|H_1) \to max$ среди всех. Работает корректно при $n < 5000$.

$$H_0 : F \in \{N(a, \sigma^2)\}$$

$$W = \frac{\left(\sum\limits_{i=1}^{t} a_{n-i+1}(X_{(n-i+1)} - X_{(i)})\right)^2}{\sum\limits_{i=1}^{n}(X_i - \overline{X})^2}$$

$$t = \begin{cases} \frac{n}{2}, \text{ если } n\%2 = 0 \\ \frac{n-1}{2}, \text{ если } n\%2 = 1 \end{cases}$$

При H_0 W имеет табличное распределение, $C_{\text{кр}} = (-\infty, W_\alpha)$.

Пример

```
stats.shapiro(norm_sample)
```

3.2.2 Тест Харке-Бера

Теория

$$H_0 : F(x) \in \{N(a, \sigma^2)\}$$

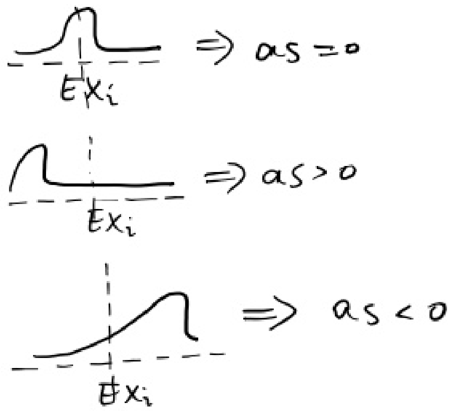
$$JB = \frac{n}{6}\left((S_k)^2 + \frac{1}{4}(K_n)^2\right)$$

$$S_k = \frac{\hat{\mu_3}}{(\hat{\mu_2})^{\frac{3}{2}}}, \text{ Skewness - коэффициент ассиметрии}$$

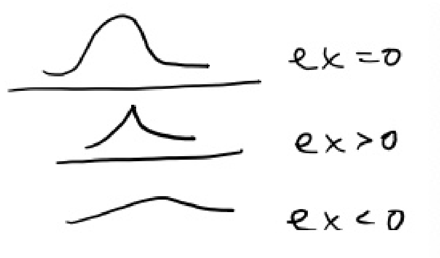
$$K_n = \frac{\hat{\mu_4}}{(\hat{\mu_2})^2} - 3, \text{ Kurtosis - коэффициент эксцесса}$$

$$\hat{\mu_j} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^j$$

$$as = \frac{\mu_3}{(\mu_2)^{\frac{3}{2}}}, \text{ степень симметричности}$$



$$ex = \frac{\mu_4}{(\mu_2^2)} - 3, \text{ степень остроконечности}$$



Тест работает корректно при $n > 2000$, тогда можно считать, что $JB \overset{H_0}{\sim} \chi^2(2)$. Если $n < 2000$, то нужно смотреть таблицу квантилей.

Пример

```
stats.jarque_bera(norm_sample)
res = stats.jarque_bera(norm_sample)
statistics = res[0]
p_value = 1 - stats.chi2.cdf(statistics, 2)
p_value
```

3.2.3 Тест Лиллиефорса

Теория

До сих пор находится в разработке, периодически выдает ошибки (чаще всего на выборках объема больше 900). Если p-value получается больше 0.2, то выдает 0.2.
$H_0 : F(x) = F_0(x)$ - функция распределения $N(a, \sigma^2)$ - сложная гипотеза, параметры a и σ неизвестны. ($H_0 : F \in \{N(a, \sigma^2)\}$)

$$T(x) = \sqrt{n}\, \underset{x}{sup}\, |\hat{F}_n(x) - F(x)|$$

$$C_{\text{кр}} = [X_{1-\alpha}, +\infty),\ n > 30$$

$F(x)$ - функция распределения $N(\overline{X}, \hat{\sigma^2})$, $\overline{X}$ - ОМП, а для $\hat{\sigma^2}$ - $\frac{1}{n}\sum\limits_{i=1}^{n}(X_i - X)^2$.

$F(x) = \Phi(\frac{X - \overline{X}}{\hat{\sigma}})$ - распределение Лиллиефорса.

Пример

```python
from statsmodels.stats.diagnostic import lilliefors
short_t_sample = stats.t.rvs(4, size=800)
lilliefors(short_t_sample)
```

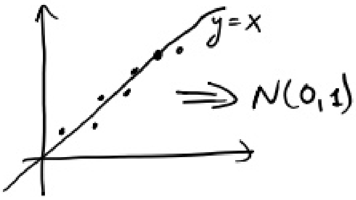
3.2.4 QQ Plot

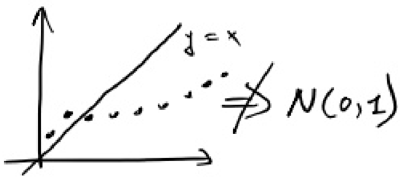
Теория

(Quantile Quantile Plot - вероятностная бумага)

Проверка на $N(0,1)$

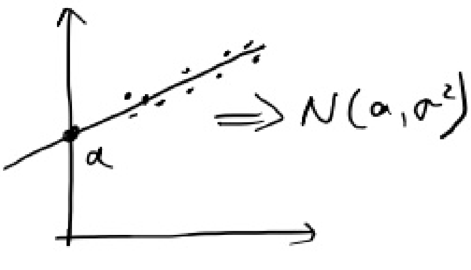
$X = (X_1, \ldots, X_n)$ из $N(0,1)$? $\Phi(x)$ - функция распределения $N(0,1)$. $\hat{F}_n(x) \approx \Phi(x)$. $\Phi^{-1}(\hat{F}_n(X_i)) \approx \Phi^{-1}(\Phi(X_i)) = X_i$. Рассмотрим точки $\left(\Phi^{-1}(\hat{F}_n(X_i)),\ \hat{F}_n^{-1}(\hat{F}_n(X_i))\right)$, где первая координата - теоретический квантиль, а вторая - выборочный квантиль.





Проверка на $N(a, \sigma^2)$

$X = (X_1, \ldots, X_n)$ из $N(a,\sigma^2)$? Вместо $y = x$ хотим увидеть $y = \sigma x + a$. $\hat{F}_n(x) \approx F_{N(a,\sigma^2)}(x)$. $\Phi^{-1}(\hat{F_N}(X)) \approx \Phi^{-1}(F_N(X)) = \Phi^{-1}(\Phi(\frac{X-a}{\sigma})) = \frac{X-a}{\sigma}$. Рассмотрим точки $\left(\frac{X_i - a}{\sigma}, X_i\right)$.



Пример

```python
stats.probplot(norm_sample, plot=plt);
```

3.2.5 Bootstrap

Теория

Асимптотический ДИ:

$$\sqrt{n}\,(\hat{\theta_n} - \theta) \xrightarrow{d} N(0, \sigma^2(\theta)) \text{ – АНО } \hat{\theta_n}$$

Проблемы:

1) $\sigma^2(\theta)$ сложно посчитать

2) не знаем распределения выборки X_i

3) n не бесконечно большое

$X = (X_1, \dots, X_n)$ - выборка с неизветсной функцией распределения F.

$$T_n(X_1, \dots, X_n) \ \leftarrow \ \hat{\theta_n}(X_1, \dots, X_n) \text{ АНО}$$

Например, хотим оценить DT_n.

Этап 1

Рассмотрим реализацию $X_1, \dots, X_n$:

X_1	$\dots$	X_n
$\frac{1}{n}$	$\dots$	$\frac{1}{n}$

- дискретное распределение, это функция распределения $\hat{F}_n(x)$ (ЭФР).

Bootstrap – создание одной или нескольких выборок $\hat{F}_n$ объема n. $\hat{F}_n(x) \approx F(x)$.

Этап 2

DT_n – ?

Сэмплируем m бутстрэпных выборок: $\{X^*_{i,1}\}_{i=1}^n, \dots, \{X^*_{i,m}\}_{i=1}^n$. Вычисляем статистику на выборках: $T^*_{n,1}, \dots, T^*_{n,m}$.

$$V_{boot}(T) = \hat{DT_n} = \frac{1}{m}\sum_{j=1}^{m}\left(T^*_{n,j} - \frac{1}{m}\sum_{l=1}^{m}T^*_{n,l}\right)^2, \ \ T^* = \frac{1}{m}\sum_{l=1}^{m}T^*_{n,l}$$

Этап 3

T_n для АДИ:

$$\left(T_n - Z_{\frac{1+\gamma}{2}} \cdot \sqrt{V_{boot}(T)},\ T_n + Z_{\frac{1+\gamma}{2}} \cdot \sqrt{V_{boot}(T)}\right)$$

Пример

```python
sample = pd.read_csv("Bootstrap_sample.txt")
sample = sample['sample']
n = len(sample)

bootstrap = np.random.choice(sample, size=(100, n))
v_boot = np.percentile(bootstrap, 50, axis=1).var()
np.percentile(bootstrap, 50, axis=1)
gamma=0.9
g = (1 + gamma)/2.0
(
  np.percentile(sample, 50) - stats.norm.ppf(g) * np.sqrt(v_boot),
  np.percentile(sample, 50) + stats.norm.ppf(g) * np.sqrt(v_boot)
)
```

4 Корреляция

4.1 Общие сведения

$$r = \frac{cov(\xi, \eta)}{\sqrt{D\xi}\sqrt{D\eta}} = \frac{E\xi\eta - E\xi E\eta}{\sqrt{D\xi}\sqrt{D\eta}} \in (-1, 1)$$

- $r = 1 \;\Rightarrow\; \xi = a\eta + b,\; a > 0$
- $r = -1 \;\Rightarrow\; \xi = a\eta + b,\; a < 0$
- $r = 0 \;\Rightarrow$ возможно ξ и η независимы, но не всегда: $\xi \sim N(0,1)$ и ξ^2 – $corr(\xi, \xi^2) = 0$, но ξ и ξ^2 не независимы.

Т.о. корреляция - мера линейной зависимости.

4.2 Выборочный коэффициент корреляции Пирсона

Теория

Тест применяется для нормальных выборок, больших выборок без тяжелых хвостов. Тест не устойчив к выбросам. Строим точечную оценку корреляции. Хорошо ловит именно линейную зависимость.

n объектов, X и Y - два признака.

$$r_{XY} = \frac{\frac{1}{n}\sum X_i Y_i - \overline{XY}}{\sigma_X \sigma_Y} \text{ - состоятельная оценка}$$

$$\sigma_X = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2},\ \sigma_Y = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})^2}$$

$$r_{XY} = \frac{\sqrt{\sum(X_i - \overline{X})(Y_i - \overline{Y})}}{\sqrt{\sum\limits_{i=1}^{n}(X_i - \overline{X})^2 \sum\limits_{i=1}^{n}(Y_i - \overline{Y})^2}}$$

$H_0 : r = 0.$

$$T = \frac{r_{XY} \cdot \sqrt{n-2}}{\sqrt{1 - r_{XY}^2}} \overset{H_0}{\sim} t(n-2)$$

Пример

```
pearsonr(x, y)[0]
pearsonr(x, y)[1]
r = (np.mean(x*y) - np.mean(x)*np.mean(y))/(np.std(x) * np.std(y))
from scipy.stats import t
n=len(x)
t_stat = (r * np.sqrt(n-2)) / np.sqrt(1 - r**2)
p_value = 2*np.min([t.cdf(t_stat, n-2), 1 - t.cdf(t_stat, n-2)])
```

4.3 Коэффициент корреляции Спирмана

Теория

Тест непараметрический (применяется, если выборка ненормальна). Ловит монотонные распределения, более устойчив к выбросам.

Имеем $X = (X_1, \ldots, X_n)$, $Y = (Y_1, \ldots, Y_n)$. R_i - ранги X_i, S_j - ранги Y_j. $\overline{R} = \overline{S} = \frac{n+1}{2}$.

$$\rho_S = \frac{\sqrt{\sum(R_i - \overline{R})(S_i - \overline{S})}}{\sqrt{\sum\limits_{i=1}^{n}(R_i - \overline{R})^2 \sum\limits_{i=1}^{n}(S_i - \overline{S})^2}}$$

$H_0 : X$ и Y независимы, т.е. $E\rho_S = 0,\ D\rho_S = \frac{1}{n-1}$.

Асимптотический критерий для $n \geq 50$:

$$\frac{\rho_S}{\sqrt{D\rho_S}} = \sqrt{n-1}\,\rho_S \xrightarrow[H_0]{d} N(0,1)$$

Пример

```
spearmanr(x, y)[0]
spearmanr(x, y)[1]
```

4.4 Коэффициент корреляции Кендала

Теория

Определение 4.1. (X_i, Y_i) *и* (X_j, Y_j) *называются* ***согласованными***, *если* $sgn(X_i - X_j) \cdot sgn(Y_i - Y_j) = 1$.

S - число согласованных пар, R - число несогласованных пар. $T = S - R$. Всего пар $C_n^2 = \frac{n(n-1)}{2}$, $-\frac{n(n-1)}{2} \leq T \leq \frac{n(n-1)}{2}$.

$$\tau = \frac{2}{n(n-1)}T,\ -1 \leq \tau \leq 1$$

$H_0 : X$ и Y независимы, т.е. $E\tau = 0,\ D\tau = \frac{2(2n+5)}{9n(n-1)}$.

Асимптотический критерий для $n \geq 50$:

$$\frac{\tau}{\sqrt{D\tau}} \xrightarrow[H_0]{d} N(0,1)$$

H_1 двусторонняя, свойства похожи на Спирмана.

Пример

```
kendalltau(x, y)[0]
kendalltau(x, y)[1]
```

4.5 Частная корреляция

Теория

$$\rho_{XY|Z} = \frac{\rho_{XY} - \rho_{XZ} \cdot \rho_{YZ}}{\sqrt{(1-\rho_{XZ}^2)(1-\rho_{YZ})^2}}$$

$M \geq 3$ - общее число признаков $X_1, \ldots, X_M$.
$\rho_{X_1X_2|X_3\ldots X_M} = -\frac{r_{12}}{r_{11}r_{22}}$, Σ - матрица выборочных корреляций, $R = \Sigma^{-1}$, $R = (r_{ij})$.

H_0 : некоррелируемость X_1 и X_2 без $(X_3, \ldots, X_M)$.

$$T = \frac{\rho \cdot \sqrt{n-M}}{\sqrt{1-\rho^2}} \overset{\mathrm{H}_0}{\sim} t(n-M),\ \rho = \rho_{X_1 X_2 | X_3 \ldots X_M}$$

Пример

```
z = st.norm.rvs(size=1000, loc=0, scale=4)
x = z + st.norm.rvs(size=1000, loc=3, scale=1)
y = z + st.norm.rvs(size=1000, loc=-2, scale=1)
def partial_corr(x, y, z, method='pearson'):
    if method == 'pearson':
        r_xy, r_xz, r_yz = pearsonr(x, y)[0], pearsonr(x, z)[0],
            pearsonr(y, z)[0]
    elif method == 'kendall':
         r_xy, r_xz, r_yz = kendalltau(x, y).correlation,
            kendalltau(x, z).correlation, kendalltau(y, z).correlation
    else:
        return None
    return (r_xy - r_xz * r_yz) / np.sqrt((1 - r_xz ** 2) *
        (1 - r_yz ** 2))
  print ("pearson partial correlation:",
      partial_corr(x, y, z, method='pearson'))
```

4.6 Таблицы сопряженности

Теория

Пусть A и B - признаки, $A_1, \ldots, A_r, B_1, \ldots, B_s$ - значения.

	B_1	B_2	$\ldots$	B_s
A_1	μ_{11}	μ_{12}	$\ldots$	μ_{1s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
A_r	μ_{r1}	μ_{r2}	$\ldots$	μ_{rs}

H_0 : A и B независимы.

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{s} \frac{\left(\mu_{ij} - n \cdot \frac{\mu_{i\cdot}}{n} \cdot \frac{\mu_{\cdot j}}{n}\right)^2}{n \cdot \frac{\mu_{i\cdot}}{n} \cdot \frac{\mu_{\cdot j}}{n}} \xrightarrow[H_0]{d} \chi^2\left((r-1)(s-1)\right)$$

$$\mu_{i\cdot} = \sum_{j=1}^{s} \mu_{ij},\ \mu_{\cdot j} = \sum_{i=1}^{r} \mu_{ij}$$

$$C_{\text{кр}} = [\chi^2_{1-\alpha}\left((r-1)(s-1)\right), +\infty)$$

Критерий Фишера:
$\tilde{r}$ - кол-во исходов

$$\hat{\chi}^2 = \sum_{i=1}^{r} \frac{\left(m_i - np_i(\hat{\theta}_1, \ldots, \hat{\theta}_k)\right)^2}{np_i(\hat{\theta}_1, \ldots, \hat{\theta}_k)} \xrightarrow[H_0]{d} \chi^2(\tilde{r} - 1 - k)$$

Пример

Задача 4.1. *Программные продукты оцениваются по шкале от 1 до 4 по качеству, и кроме того, имеется два способа написания программных продуктов: быстрый и медленный. Известно, что среди быстро написанных программных продуктов оценку 1 имеют 120 продуктов, оценку 2 – 124 продукта, 3 — 133 продукта, 4 – 106 продуктов. Среди медленно написанных продуктов оценку 1 имеют 97, 2 – 142, 3 – 129 и 4 – 149 продуктов. Выяснить, имеется ли статистически значимая связь между скоростью написания программных продуктов и их качеством (на уровне значимости 1% и на уровне значимости 3%).*

Решение.

```
table = np.array([[120, 124, 133, 106],
[97, 142, 129, 149]])

st.chi2_contingency(table)
st.chi2_contingency(table)[1]

```

■

Задача 4.2. *Проверка независимости двух признаков с помощью таблиц сопряженности*

Решение.

```
Star = pd.read_csv("Star.csv")
x = Star['x']
y = Star['y']
Star['x_bin'] = pd.cut(x, [-np.inf, -0.5, 0.5, np.inf])
Star['y_bin'] = pd.cut(y, [-np.inf, -0.5, 0.5, np.inf])

table2 = Star.pivot_table(values='x', index='x_bin', columns='y_bin', aggfunc
='count', fill_value=0)

res = st.chi2_contingency(table2)
p_value = res[1]

```

■

5 Сравнение двух выборок

5.1 Сравнение средних (медиан)

5.1.1 Парные (зависимые выборки)

Признаки:

- одинаковый размер выборок
- строгое соответствие $X_i \leftrightarrow Y_i$

Примеры:

- «до и после»
- один и тот же показатель двумя способами

Сводим к одной выборке: $Z_i = Y_i - X_i$.

t-test 2 sample

Теория Применяем, если X и Y - две нормальные выборки.

$$Z_i = Y_i - X_i$$
$$H_0 : EZ_i = 0 \;\Leftrightarrow\; H_0 : EX_i = EY_i$$

Применяем t-test:

$$t = \frac{\overline{Z}}{\frac{S_Z}{\sqrt{n}}} \overset{\mathrm{H_0}}{\sim} t(n-1)$$

Пример

Задача 5.1. *Было проведено исследование, чтобы выяснить, повлияют ли новые диетические медикаменты на женщин, желающих сбросить вес. Вес 100 пациенток был измерен до лечения и через 6 недель ежедневного применения лечения. Данные приведены в файле "Weight.txt". При уровне значимости 5% можно ли сделать вывод, что лечение уменьшает вес?*

Решение.

```
data = pd.read_csv("Weight.txt")
x = data['x']
y = data['y']
st.shapiro(x)
st.shapiro(y)
from scipy.stats import ttest_rel
ttest_rel(x, y)
t_test_res = ttest_rel(x, y)
t_test_res.pvalue/2.0
```

Вручную:

```
z = y - x
from scipy.stats import t
n = len(z)
z_mean = np.mean(z)
z_s = np.std(z, ddof=1)
t_stat = (z_mean - 0) * np.sqrt(n) / z_s
t_p_value = t.cdf(t_stat, n-1)
t_p_value
```

■

Критерий знаков

Теория Является непараметрическим критерием, применяется, когда выборки не нормальны.

$Z_i = Y_i - X_i$, $Z_i = \theta + e_i$, где θ - эффект обработки, e_i - взаимно независимые случайные величины из непрерывной совокупности такой, что $P(e_i < 0) = P(e_i > 0) = \frac{1}{2}$.

$H_0 : \theta = 0$. $\psi_i = \begin{cases} 1, & Z_i > 0 \\ 0, & Z_i < 0 \end{cases}$. Отбрасываем $Z_i = 0$ и уменьшаем n. $B = \sum\limits_{i=1}^{n} \psi_i$ - число пар таких, что $Y_i > X_i$. $B \overset{\mathrm{H_0}}{\sim} Bin(n, \frac{1}{2})$.

Асимптотический критерий: $B^* = \frac{B - \frac{n}{2}}{\sqrt{\frac{n}{4}}} \xrightarrow[H_0]{d} N(0, 1)$.

```python
    z = z[z!=0]
    b = sum(z > 0)
    n = len(z)

    from scipy.stats import binom_test
    binom_test(b, n,  p=0.5)

    sign_res=binom_test(b, n,  p=0.5)
    sign_res/2.0

    from scipy.stats import binom
    binom.cdf(b, n, 0.5)

    b_star = (b - n*0.5) / np.sqrt(n*0.25)
    from scipy.stats import norm
    norm.cdf(b_star, loc=0, scale=1)

```

Знакоранговый критерий Вилкоксона

Теория Является непараметрическим критерием, применяется, когда выборки не нормальны. Применяется при $n > 20$.

$Z_i = Y_i - X_i$, $Z_i = \theta + e_i$, где θ - эффект обработки, e_i - взаимно независимые случайные величины из непрерывной совокупности и симметричные относительно 0.

$H_0 : \theta = 0$. Сортируем $|Z_1|, \dots, |Z_n|$ по возрастанию. R_i - ранг Z_i. $\psi_i = \begin{cases} 1, & Z_i > 0 \\ 0, & Z_i < 0 \end{cases}$. Отбрасываем $Z_i = 0$ и уменьшаем n. $T^+ = \sum\limits_{i=1}^{n} R_i \psi_i$.

Асимптотический критерий: $\frac{T^* - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}} \xrightarrow[H_0]{d} N(0,1)$.

```python
    from scipy.stats import wilcoxon
    wilcoxon(x, y)
    signed_rank_res = wilcoxon(x, y)
    signed_rank_res.pvalue/2.0

```

5.1.2 Независимые выборки

Признаки:

- может быть разный размер выборок
- выборки взяты «из разных мест» независимо друг от друга

Примеры:

- врачи из разных городов

F-test 2 sample

Применяется, если выборки нормальны. Используем F-test для установления равенства или неравенства дисперсий.

t-test 2 sample

Теория Применяется, если имеем неизвестные равные дисперсии.

$X_i \sim N(a_1, \sigma_1^2)$, $Y_i \sim N(a_2, \sigma_2^2)$. После F-testa устанавливаем, что $\sigma_1^2 = \sigma_2^2$. $H_0 : EX_i = EY_j$.

$$t = \frac{\overline{X} - \overline{Y}}{\sqrt{S_p^2(\frac{1}{n} + \frac{1}{m})}} \overset{\text{H}_0}{\sim} t(n + m - 2)$$

$$S_p^2 = \frac{(n-1)S_x^2 + (m-1)S_y^2}{n + m - 2} \text{ (pooled variance estimator)}$$

Пример

Задача 5.2. *Для сравнения уровня заработной платы были отобраны в соответствии со стажем работники-мужчины и работники-женщины. В файлах "Male.txt"и "Female.txt"содержатся получившиеся данные (в тысячах рублей). Можно ли утверждать на уровне значимости 5%, что зарплата женщин ниже?*

Решение.

```
data1 = pd.read_csv("Male.txt")
male = data1['male']
data2 = pd.read_csv("Female.txt")
female = data2['female']
st.shapiro(male)
st.shapiro(female)

from scipy.stats import f
def F_test(x, y):
    x = np.array(x)
```

```python
        y = np.array(y)
        df1 = len(x) - 1
        df2 = len(y) - 1
        F_stat = np.var(x, ddof=1)/np.var(y, ddof=1)
        pv = 2*np.min([f.cdf(F_stat, df1, df2), 1 - f.cdf(F_stat, df1, df2)
])
        return pv

    F_test(male, female)

    from scipy.stats import ttest_ind
    ttest_ind(male, female, equal_var=False)
    t_res = ttest_ind(male, female, equal_var=False)
    t_res.pvalue/2.0

```

■

Критерий Аспина-Уэлса

Теория Применяется, если имеем неизвестные разные дисперсии.

$X_i \sim N(a_1, \sigma_1^2)$, $Y_i \sim N(a_2, \sigma_2^2)$. После F-testa устанавливаем, что $\sigma_1^2 \neq \sigma_2^2$. $H_0 : EX_i = EY_j$.

$$t = \frac{\overline{X} - \overline{Y}}{\sqrt{\frac{S_x^2}{n} + \frac{S_y^2}{m}}} \overset{\mathrm{H}_0}{\sim} t(d.f.)$$

$$d.f. = \frac{\left(\frac{S_x^2}{n} + \frac{S_y^2}{m}\right)^2}{\frac{\left(\frac{S_x^2}{n}\right)^2}{n-1} + \frac{\left(\frac{S_y^2}{m}\right)^2}{m-1}} \text{ (берем ближайшее целое число)}$$

При больших n и m статистика $t \overset{\mathrm{H}_0}{\sim} N(0,1)$.

Пример надо реализовать

Критерий ранговых сумм Вилкоксона

Теория Является непараметрическим тестом, применятся в случае ненормальности выборок.

$X_i = e_i, i = 1, 2, \dots, n$, $Y_j = e_{n+j} + \triangle$, где $\triangle$ - сдвиг. $e_1, \dots, e_n$ - значения X, $e_{n+1}, \dots, e_{n+m}$ - значения Y. Значения взаимно независимы и из одной непрерывной совокупности.

$H_0 : \triangle = 0$ (нет повторов). Рассмотрим $N = n + m$ наблюдений. От min к max R_j – ранг Y_j. Статистика: $W = \sum\limits_{j=1}^{m} R_j$.

Замечание: рассматриваем ранги X_i, тогда:

$$W' = \frac{(n+m)(n+m+1)}{2} - W$$

Асимптотический критерий при $n, m > 20$:

$$\frac{W - \frac{m(n+m+1)}{2}}{\sqrt{\frac{nm(n+m+1)}{12}}} \xrightarrow[H_0]{d} N(0,1)$$

Пример надо реализовать

Критерий Манна и Уитни

Теория Является непараметрическим тестом, применятся в случае ненормальности выборок.

$H_0 : \triangle = 0$. Вместо W рассмотрим:

$$U = \sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{I}(X_i < Y_j)$$

$$W = U + \frac{m(m+1)}{2}$$

Асимптотический критерий:

$$\frac{U - \frac{nm}{2}}{\sqrt{\frac{nm(n+m+1)}{12}}} \xrightarrow[H_0]{d} N(0,1)$$

```
from scipy.stats import mannwhitneyu
mannwhitneyu(female, male, alternative='less')
res = mannwhitneyu(female, male, alternative='less')
res.pvalue
```

5.2 Сравнение дисперсий

Пусть $X = (X_1, \ldots, X_n)$ и $Y = (Y_1, \ldots, Y_n)$ – две независимые выборки. $H_0 : DX_i = DY_j$. $H_1 : DX_i \neq DY_j$.

5.2.1 F-test 2 sample

Теория

Применяется, если обе выборки нормальные.

$X_i \sim N(a_1, \sigma_1^2),\ Y_j \sim N(a_2, \sigma_2^2)$ - независимые выборки. $H_0 : \sigma_1^2 = \sigma_2^2$. $H_1 : \sigma_1^2 \neq \sigma_2^2$.

$$F = \frac{S_X^2}{S_Y^2} \overset{\mathrm{H_0}}{\sim} F(n-1, m-1)$$

Пример

надо реализовать

5.2.2 Критерий Зигеля-Тьюки

Теория

Рассмотрим $N = n + m$. Сортируем от min к max, где Z - объединенные X и Y: $Z_{(1)} \leq Z_{(2)} \leq \cdots \leq Z_{(n-1)} \leq Z_{(n)}$. Присваиваем ранги двигаясь от краев к центру последовательности, чередуя начало и конец последовательности, т.е.: ранг 1 имеет $Z_{(1)}$, ранг 2 – $Z_{(n)}$, ранг 3 – $Z_{(2)}$ и т.д.

$T = \sum\limits_{i=1}^{n} \widetilde{rank}(X_i)$.

Асимптотический критерий:

$$\frac{T - \frac{n(n+m+1)}{2}}{\sqrt{\frac{nm(n+m+1)}{2}}} \xrightarrow[H_0]{d} N(0,1)$$

Пример

надо реализовать

6 Проверка на однородность

Используется только для непрерывных распределений.

6.1 Сравнение двух выборок

Пусть $X = (X_1, \dots, X_n)$ имеет распределение F, $Y = (Y_1, \dots, Y_n)$ имеет распределение G. X и Y - две независимые выборки.
$H_0 : F = G$.

6.1.1 Критерий Смирного

Теория

Является непараметрическим критерием.

$$T = \sqrt{\frac{nm}{n+m}} \cdot \underset{x \in \mathbb{R}}{sup} \, |\hat{F_n}(x) - \hat{G_m}(x)|$$

Асимптотический критерий:

$$T \xrightarrow[H_0]{d} \xi, \;\; \xi \text{ имеет распределение Колмогорова}$$

$$C_{\text{кр}} = [K_{1-\alpha}, +\infty)$$

Пример

```
from scipy.stats import norm, t
x = norm.rvs(size = 200, loc = 0, scale = 1)
y = t.rvs(size=300, df = 7)
z = norm.rvs(size = 400, loc = 0, scale = 3)
from scipy.stats import ks_2samp
s_2samp(x, y)
ks_2samp(x, z)

```

6.2 Сравнение $k \geq 2$ выборок

6.2.1 Общий критерий Андерсона-Дарлинга

Теория

Является непараметрическим критерием.

Имеем $k \geq 2$ независимых выборок. $(X_{11}, \dots, X_{1n_1})$ имеет распределение F_1, $\dots$, $(X_{k1}, \dots, X_{kn_k})$ имеет распределение F_k. $H_0: F_1 = \dots = F_k$. $\hat{F}_1, \dots, \hat{F}_k$ - ЭФР, $N = n_1 + \dots + n_k$. $\hat{H}_N(x)$ – ЭФР по всей совокупности из N наблюдений.

$$\Omega^2 = \sum_{i=1}^{k} n_i \cdot \int\limits_{\mathbb{R}} \frac{\left(\hat{F}_i(x) - \hat{H}_N(x)\right)^2}{\hat{H}_N(x)(1 - \hat{H}_N(x))} d\hat{H}_N(x)$$

Пример

В качестве результата тест выдает значение статистики, набор квантилей $x_{1-\alpha}$ для значений α вида 25%, 10%, 5%, 2.5%, 1%, 0.5%, 0.1% и p-value.

```
    from scipy.stats import anderson_ksamp
    anderson_ksamp([x, y])

```

6.2.2 Критерий Краскела-Уоллиса

Теория

Применяется в непараметрическом случае (хотя бы одна выборка ненормальна).

Имеем $k \geq 2$ выборок:

$$\begin{pmatrix} X_{11} \\ \vdots \\ X_{n_1 1} \end{pmatrix}, \begin{pmatrix} X_{12} \\ \vdots \\ X_{n_2 2} \end{pmatrix}, \dots, \begin{pmatrix} X_{1k} \\ \vdots \\ X_{n_k k} \end{pmatrix}$$

Для элемента X_{ij} i - это номер элемента в j-ой выборке, а j - это номер выборки. $N = \sum\limits_{j=1}^{k} n_j$.

$X_{ij} = \mu + \beta_j + \varepsilon_{ij}$, где β_j - эффект от воздействия фактора, ε_{ij} - случайные ошибки. Необходимо, чтобы были выполнены следующие условия:

- все ε_{ij} независимы
- все ε_{ij} имеют одинаковое непрерывное распределение

$H_0 : \beta_1 = \cdots = \beta_k$. H_1 : не все β_j равны. Строим статистику: смешиваем все N наблюдений, R_{ij} - ранг X_{ij}, $S_j = \sum\limits_{i=1}^{n_j} R_{ij}$ - сумма рангов j-ой выборки, $R_{\cdot j} = \frac{S_j}{n_j}$ - средний ранг j-ой выборки, $R_{\cdot\cdot} = \frac{1}{N}\sum\limits_{i,j} R_{ij} = \frac{N+1}{2}$ - общий средний ранг.

$$H = \frac{12}{N(N+1)} \cdot \sum_{j=1}^{k} n_j (R_{\cdot j} - R_{\cdot\cdot})^2 = \left[\frac{12}{N(N+1)} \cdot \sum_{j=1}^{k} \frac{S_j^2}{n_j}\right] - 3(N+1)$$

$$H \underset{H_0}{\longrightarrow} \text{табличное распределение}$$

$$C_{\text{кр}} = [h_{1-\alpha}, +\infty)$$

Асимптотический критерий:

$$H \xrightarrow[H_0]{d} \chi^2(k-1), \;\; C_{\text{кр}} = [\chi^2_{1-\alpha}(k-1), +\infty)$$

Пример

Задача 6.1. *В файле «Harvest.txt» представлены данные об урожае клубники (в квартах) с участков трех типов почв. Влияет ли (на уровне значимости 5%) тип почвы на урожайность?*

Решение.

```
        data = pd.read_csv("Harvest.txt")
        x = data['x']
        y = data['y']
        z = data['z']
        st.shapiro(x)
        st.shapiro(y)
        st.shapiro(z)
        from scipy.stats import kruskal
        kruskal (x, y, z)

```

■

6.2.3 Критерий Джонкхиера

Теория

$H_0 : \beta_1 = \cdots = \beta_k$, $H_1 : \beta_1 \le \beta_2 \le \cdots \le \beta_k$ - альтернатива возрастания влияния фактора. Данный тест имеет большую мощность при такой альтернативе, чем тест Краскела-Уоллеса.

Статистика: $C_k^2 = \frac{k(k-1)}{2}$ - кол-во пар выборок. Считает $\frac{k(k-1)}{2}$ значений статистики Манна-Уитни.

$$U_{rs} = \sum_{i=1}^{n_r}\sum_{j=1}^{n_s}\mathbb{I}(X_{ir} < X_{js}), \quad 1 \leq r \leq s \leq k$$

$$J = \sum_{r<s} U_{rs} = \sum_{r=1}^{k-1}\sum_{s=r+1}^{k} U_{rs} \text{ -- имеет табличное распределение}$$

$$C_{\text{кр}} = [j_{1-\alpha}, +\infty)$$

Асимптотический критерий:

$$J^* = \frac{J - EJ}{\sqrt{DJ}} \xrightarrow[H_0]{d} N(0,1)$$

$$H_0 \Rightarrow \begin{cases} EJ = \frac{1}{4}\left(N^2 - \sum\limits_{j=1}^{k} n_j^2\right) \\ DJ = \frac{1}{72}\left(N^2(2N+3) - \sum\limits_{j=1}^{k} n_j^2(2n_j+3)\right) \end{cases}$$

Пример

надо реализовать

6.2.4 Критерий Неменьи

Теория

Nemenyi test принадлежит семейству Post Hoc tests - они мощнее, чем попарный t-test, а также легки в подсчете.

$H_0 : \beta_r = \beta_s$, $H_1 : \beta_r \neq \beta_s$.

$$T = \frac{R_{\cdot r} - R_{\cdot s}}{\sqrt{\frac{N(N+1)}{24}\left(\frac{1}{n_r} + \frac{1}{n_s}\right)}} \sim \text{ табличное распределение}$$

$$|T| > q_{\text{crit}} \Rightarrow \text{ отвергаем } H_0$$

Желательно делать поправку на множественное сравнение.

Пример

```
import scikit_posthocs as sp
sp.posthoc_nemenyi([x, y, z])
nem_res = sp.posthoc_nemenyi([x, y, z])
nem_res[1][2]
```

6.2.5 Критерий Бартлетта

Теория

Применяется в предположении, что $X_{ij} \sim N(\mu_j, \sigma_j^2)$, параметры распределения неизвестны. Критерий применяется для получения информации про дисперсию.

$H_0 : \sigma_1 = \dots = \sigma_k$, H_1 : не все σ_j равны. Статистика:

$$B^* = \gamma^{-1} \cdot N \cdot \ln B, \text{ где}$$

$$\gamma = 1 + \frac{1}{3(k-1)} \left[\sum_{j=1}^{k} \frac{1}{n_j} - \frac{1}{N} \right]$$

$$B = \frac{\left(\frac{1}{N} \sum\limits_{j=1}^{k} n_j S_j^2 \right)}{\sqrt[N]{\prod\limits_{j=1}^{k} (S_j^2)^{n_j}}}$$

$$S_j^2 = \frac{1}{n_j - 1} \sum_{i=1}^{n_j} (X_{ij} - X_{\cdot j})^2$$

При $n_j \geq 3$ $B^* \overset{\mathrm{H}_0}{\sim} \chi^2(k-1)$.

При отвержении H_0 возникает проблема множественных сравнений, поэтому необходимо делать **поправку Бонферонни**: $\alpha \to \frac{\alpha}{C_k^2} = \tilde{\alpha}$.

Пример

```
from scipy.stats import bartlett
bartlett(x, y, z)
```

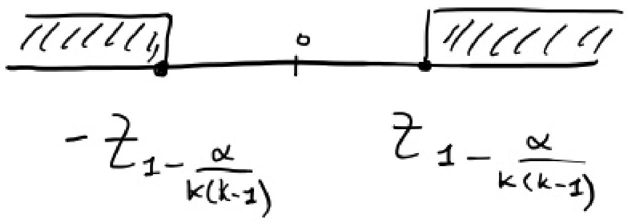
6.2.6 Критерий Данна

Теория

Является непараметрическим тестом, применяется для больших выборок, т.е. является асимптотическим.

$H_0 : \beta_r = \beta_s$, $H_1 : \beta_r \neq \beta_s$.

$$T = \frac{R_{\cdot r} - R_{\cdot s}}{\sqrt{\frac{N(N+1)}{12}\left(\frac{1}{n_r} + \frac{1}{n_s}\right)}}$$



$$1 - \frac{\alpha}{2} \to 1 - \frac{\alpha}{C_k^2 \cdot 2} = 1 - \frac{\alpha}{k(k-1)}$$

Пример

```
sp.posthoc_dunn([x, y, z], p_adjust = 'bonferroni')
dunn_res = sp.posthoc_dunn([x, y, z], p_adjust = 'bonferroni')
dunn_res[1][3]
dunn_res = sp.posthoc_dunn([x, y, z], p_adjust = None)
dunn_res[1][3]*3

```

6.2.7 ANOVA

Теория

Применяется в случае k нормальных выборок - однофакторный дисперсионный анализ: ANOVA = Analysis of Variance.

$X_{ij} \sim N(\mu_j, \sigma^2)$. Для применения необходимо выполнния следующих условий:

- нормальное распределение всех выборок
- равенство дисперсий всех выборок
- независимость наблюдений

$H_0 : \mu_1 = \dots = \mu_k$, H_1 : не все μ_j равны. Строим статистику:

$$X_{\cdot j} = \frac{1}{n_j}\sum_{i=1}^{n_j} X_{ij} = \overline{X_j} \text{ - среднее } j\text{-ой выборки}$$

$$X_{\cdot\cdot} = \frac{1}{N}\sum_{j=1}^{k}\sum_{i=1}^{n_j} X_{ij}$$

$$F = \frac{N-k}{k-1}\cdot\frac{\sum\limits_{j=1}^{k} n_j(X_{\cdot j} - X_{\cdot\cdot})^2}{\sum\limits_{j=1}^{k}\sum\limits_{i=1}^{n_j}(X_{ij} - X_{\cdot j})^2} \overset{\mathrm{H_0}}{\sim} F(k-1, N-k)$$

Если $n_1 = \dots = n_k$, то при небольшом отклонении от нормальности распределений и от равенства дисперсий ANOVA все равно можно применять.

Пример

```
from scipy.stats import f_oneway
f_oneway(x, y, z)

```

6.2.8 LSD Фишера

Теория

Fisher's least significant difference test применяется в предположении нормальности данных. $X_{ij} \sim N(\mu_j, \sigma^2)$. Дисперсии равны, выборки независимы.

$H_0 : \mu_r = \mu_s$, $H_1 : \mu_r \neq \mu_s$.

$$T = \frac{X_{\cdot r} - X_{\cdot s}}{\sqrt{MS_E\left(\frac{1}{n_r} + \frac{1}{n_s}\right)}} \overset{\mathrm{H_0}}{\sim} t(N-k), \text{ где}$$

$$X_{\cdot r} = \frac{1}{n_r}\sum_{i=1}^{n_r} X_{ir},\ MS_E = \frac{SS_E}{N_k},\ SS_E = \sum_{j=1}^{k}\sum_{i=1}^{n_j}(X_{ij} - X_{\cdot j})^2 = \sum_{j=1}^{k}(n_j - 1)S_j^2$$

Пример

```
def LSD_Fisher(i, j, samples):
    n1, n2 = len(samples[i]), len(samples[j])
    k = len(samples)
    N = np.sum([len(samples[l]) for l in range(k)])
    SSe = np.sum([np.var(samples[l], ddof=0) * len(samples[l]) for l in
range(k)])
```

```
        stat = (np.mean(samples[i]) - np.mean(samples[j]))/np.sqrt(SSe / (N - k
    ) * (1.0/n1 + 1.0/n2))
        return 2*np.min([ st.t.cdf(stat, N - k), 1 - st.t.cdf(stat, N - k)])

    LSD_Fisher(0, 2, [x,y,z])

```

6.2.9 Критерий Шеффе

Теория

Scheffe's test применяется в предположении нормальности данных.

$X_{ij} \sim N(\mu_j, \sigma^2)$. $H_0 : \sum\limits_{j=1}^{k} c_j\mu_j = 0$, где $\sum\limits_{j=1}^{k} c_j = 0$. $H_1 : \sum\limits_{j=1}^{k} c_j\mu_j \neq 0$. Статистика:

$$S = \frac{\left(\sum\limits_{j=1}^{k} c_j \cdot X_{\cdot j}\right)^2}{(k-1)MS_E \cdot \sum\limits_{j=1}^{k} \frac{c_j^2}{n_j}} \overset{H_0}{\sim} F(k-1, N-k)$$

$$C_{\text{кр}} = [f_{1-\alpha}(k-1, N-k), +\infty)$$

Пример

```
    sp.posthoc_scheffe([x, y, z])
    sch_res = sp.posthoc_scheffe([x, y, z])
    sch_res[1][2]

```

6.3 Однофакторный дисперсионный анализ для связанных выборок

$$X_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij}$$

μ – неизвестное среднее, ε_{ij} – случайные ошибки

α_i – влияние особенностей i-го объекта

β_j – влияние j-го уровня фактора

Необходимые условия:

- ε_{ij} независимы
- одинаковое непрерывное распределение

6.3.1 Критерий Фридмана

Теория

Friedman test является непараметрическим тестом.

$H_0 : \beta_1 = \dots = \beta_k$, H_1 : не все β_j равны. Строим статистику: для каждой строки i ранжируем элементы, получаем R_{ij} - ранг j-го элемента в i-ой строке. $T_j = \sum\limits_{i=1}^{n} R_{ij}$, $R_{\cdot j} = \frac{T_j}{n}$.

$$F = \left[\frac{12}{nk(k+1)} \cdot \sum_{j=1}^{k} T_j^2\right] - 3n(k+1)$$

Асимптотический критерий:

$$F \xrightarrow[H_0]{d} \chi^2(k-1), \; C_{\text{кр}} = [\chi^2_{1-\alpha}(k-1), +\infty)$$

Пример

Задача 6.2. *Несколько дегустаторов оценивают различные сорта вин. Имеют ли вина значимые отличия на уровне значимости 5%? Данные представлены в файле «Wine.csv».*

Решение.

```
        data = pd.read_csv('Wine.csv', index_col='tasters')
        data.columns.name = 'wine'
        data_ar = np.array(data)
        data_ar[0]
        data_ar.T[0]
        friedmanchisquare(*data_ar.T)

```

■

6.3.2 Критерий Пэйджа

Теория

Page's trend test является непараметрическим тестом.

$H_0 : \beta_1 = \dots = \beta_k$, $H_1 : \beta_1 \le \beta_2 \le \dots \le \beta_k$.

Статистика: $L = \sum\limits_{j=1}^{k} j \cdot T_j = T_1 + 2T_2 + \dots + kT_k$. Асимптотический критерий:

$$\frac{L - EL}{\sqrt{DL}} \xrightarrow[H_0]{d} N(0,1), \; C_{\text{кр}} = [Z_{1-\alpha}, +\infty)$$

$$H_0 \;\Rightarrow \begin{cases} EL = \frac{nk(k+1)^2}{4} \\ DL = \frac{n(k-1)k^2(k+1)^2}{144} \end{cases}$$

Пример

Задача 6.3. *Несколько дегустаторов оценивают различные вина, расположенные по увеличению стоимости за бутылку. Имеют ли вина значимые отличия на уровне значимости 5%? Данные представлены в файле «Wine_Page.csv».*

Решение.

```
data_page = pd.read_csv('Wine_Page.csv', index_col='tasters')
data_page.columns.name = 'wine'
from PageTest import Page
Page.test(np.array(data_page).tolist(), ascending=True)

```

Return values:
L float: Page's L statistic
m int: Number of replications (по строкам)
n int: Number of treatments (по столбцам)
p float: P-value ■

6.3.3 ANOVA RM

Теория

Reapeted measures ANOVA, ANOVA RM, RM ANOVA (ANOVA for correlated samples) применяется в случае нормальный наблюдений, т.е.является параметрическим тестом.

Должно быть выполнено условие сферичности: дисперсия по всех наблюдениях одинаковая. Проверка на сферичность проходит с помощью теста Маухли: смотрим C_k^2 разностей столбцов, обнулем построчные влияния α_i и применяем тесты по столбцам.

$H_0 : \beta_1 = \dots = \beta_k$, H_1 : не все β_j равны.

$$F = \frac{\left[\frac{n}{k-1} \cdot \sum\limits_{j=1}^{k} (X_{.j} - X_{..})^2\right]}{\left[\frac{1}{(n-1)(k-1)} \cdot \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{k} (X_{ij} - X_{i.} - X_{.j} + X_{..})^2\right]}$$

$$X_{i.} = \frac{1}{k}\sum_{j=1}^{k} X_{ij},\ X_{.j} = \frac{1}{n}\sum_{i=1}^{n} X_{ij},\ X_{..} = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k} X_{ij}$$

$$F \overset{\text{H}_0}{\sim} F\left(k-1, (n-1)(k-1)\right)$$

$$C_{\text{кр}} = \left[f_{1-\alpha}\left(k-1, (n-1)(k-1)\right), +\infty\right)$$

Пример

Задача 6.4. *Несколько дегустаторов оценивают различные сорта вин. Имеют ли вина значимые отличия на уровне значимости 5%? Данные представлены в файле «Wine.csv».*

Решение.

```
        data = pd.read_csv('Wine.csv', index_col='tasters')
        data.columns.name = 'wine'
        data_ar = np.array(data)
        pg.sphericity(data)
        spher_res = pg.sphericity(data)
        spher_res[4]

        resid = data.add(-data.mean(axis=0), axis='columns').add(-data.mean(axis
    =1),
            axis='rows') + data.values.mean()

        #data.add(-data.mean(axis=0), axis='columns')
        #data.add(-data.mean(axis=1), axis='rows')

        st.shapiro(resid)
        data.unstack().head()

        data.unstack().to_frame(name='score').head()

        data.unstack().to_frame(name='score').reset_index()
        data_anova = data.unstack().to_frame(name='score').reset_index()

        an_rm = AnovaRM(data_anova, depvar='score', subject='tasters', within=['
    wine'])

        res = an_rm.fit()
        pg.rm_anova(data)

        pg.rm_anova(data=data_anova, dv='score', within='wine',
         subject='tasters', detailed=False)

```

```
        rmanova_res = pg.rm_anova(data)
        rmanova_res['p-unc'][0]

```

■

7 Линейная регрессия

7.1 Общие сведения

7.1.1 Построение модели

Пусть $X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}$ - наблюдения. Имеем k неслучайных факторов:

$$Z = \begin{pmatrix} Z_{11} & \dots & Z_{1k} \\ \vdots & \ddots & \vdots \\ Z_{n1} & \dots & Z_{nk} \end{pmatrix}$$

$(Z_{ij}:\ i$ - номер эксперимента, j - номер фактора$)$

EX_i линейно зависит от факторов:

$$EX_i = \theta_1 Z_{i1} + \dots + \theta_k Z_{ik} = (Z_{i1} \dots Z_{ik}) \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_k \end{pmatrix},\ i = 1, 2, \dots, n$$

где $\theta_1, \dots, \theta_k$ - коэффициенты линейной регрессии. Пусть $\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$ - столбец ошибок. Тогда: $X = Z\theta + \varepsilon$. X_i называются откликами.

$$\left.\begin{matrix} 1)\ E\varepsilon_i = 0 \\ 2)\ D\varepsilon = \sigma^2 \mathbb{I}_n \end{matrix}\right\} \Rightarrow \text{ошибки не коррелируют и } D\varepsilon_i = \sigma^2$$

```
from statsmodels.regression.linear_model import OLS

data = pd.read_csv('Carseats.csv')
data.head()
x = add_constant(data[['CompPrice', 'Advertising', 'Price', 'Age']])
```

```
y = data['Sales']
ols = OLS(y, x)
results = ols.fit()
results.summary()
```

7.1.2 Метод наименьших квадратов

$$S(\theta) = (X - Z\theta)^T (X - Z\theta) = \varepsilon^T \varepsilon = (\varepsilon_1 \dots \varepsilon_n) \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix} = \sum_{i=1}^{n} \varepsilon_i^2$$

$$\hat{\theta} = arg\, \min_{\theta} S(\theta)$$

Теорема 7.1. *Если $A = Z^T Z$ невырождена, то $\hat{\theta} = \left(Z^T Z\right)^{-1} Z^T X$ и верны следующие свойства:*

1) $E\hat{\theta} = \theta$, $D\hat{\theta} = \sigma^2 \left(Z^T Z\right)^{-1}$

2) $\hat{\sigma^2} = \dfrac{S(\hat{\theta})}{n-k} = \dfrac{\left(X - Z\hat{\theta}\right)^T \left(X - Z\hat{\theta}\right)}{n-k} = \dfrac{\sum\limits_{i=1}^{n} \left(X_i - \hat{X}_i\right)^2}{n-k}$, *где* $RSS = S(\hat{\theta}) = \sum\limits_{i=1}^{n} \left(X_i - \hat{X}_i\right)^2$ *- сумма квадратов остатков регрессии (residual sum of squares),* $\hat{X}_i = \hat{\theta}_1 Z_{i1} + \dots + \hat{\theta}_k Z_{ik}$

Нормальная регрессия

Предположение нормальной регрессии: $\varepsilon \sim N(0, \sigma^2 \mathbb{I}_n)$, тогда $X \sim N(Z\theta, \sigma^2 \mathbb{I}_n)$.

Свойства 7.1.

1) $\hat{\theta}$ *и* $S(\hat{\theta})$ *независимы*

2) $\dfrac{\hat{\theta}_j - \theta_j}{\sigma\sqrt{a^{jj}}} \sim N(0,1)$, a^{jj} *- элементы* A^{-1}

3) $\dfrac{\hat{\theta}_j - \theta_j}{\sqrt{\frac{S(\hat{\theta})}{n-k} a^{jj}}} \sim t(n-k)$

4) $\dfrac{S(\hat{\theta})}{\sigma^2} \sim \chi^2(n-k)$

7.1.3 Доверительные интервалы

Доверительный интервал для θ_j:

$$\left(\hat{\theta}_j - t_{\frac{1+\gamma}{2}}(n-k)\sqrt{\frac{RSS}{n-k}a^{jj}}; \hat{\theta}_j + t_{\frac{1+\gamma}{2}}(n-k)\sqrt{\frac{RSS}{n-k}a^{jj}}\right)$$

Доверительный интервал для σ^2:

$$\left(\frac{RSS}{\chi^2_{\frac{1+\gamma}{2}}(n-k)}; \frac{RSS}{\chi^2_{\frac{1-\gamma}{2}}(n-k)}\right)$$

```
#### Оценки параметров регрессии
results.params
y_hat = ols.predict(results.params , x)
print(y - y_hat) #вектор остатков регрессии

#### Построение прогноза
y_hat_new = ols.predict(results.params , [1, 120, 10, 100, 50])
print(y_hat_new)

#### Оценка дисперсии
RSS = results.ssr
k = 5
n = len(data['Sales'])
sigma2_hat = RSS/(n-k)
print(sigma2_hat)

##### ДИ параметров нормальной регрессии
conf_intervals = results.conf_int(alpha=0.05)
print (conf_intervals[1][2])
```

7.1.4 Проверка значимости признаков

$H_0 : \theta_j = 0,\ H_1 : \theta_j \neq 0$ (т.е. фактор значимый).

$$T = \frac{\hat{\theta}_j}{\sqrt{\frac{RSS}{n-k}a^{jj}}} \overset{\mathrm{H_0}}{\sim} t(n-k)$$

$$C_{\text{кр}} = \big(-\infty, t_{\frac{\alpha}{2}}(n-k)\big] \bigcup \big[t_{1-\frac{\alpha}{2}}(n-k), +\infty\big)$$

```
##### Проверка значимости признаков
results.pvalues
results.tvalues
```

7.1.5 Доверительный интервал для отклика

Пусть мы оценили θ по $X_1, \ldots, X_n$ и получили $\hat{\theta}$. Тогда по $Z_{n+1} = (Z_{(n+1)1} \ldots Z_{(n+1)k})$ строим отклик $X_{n+1} = Z_{n+1}\theta + \varepsilon_{n+1}$, где $\varepsilon_{n+1} \sim N(0, \sigma^2)$. Точечная оценка: $\hat{X_{n+1}} = Z_{n+1}\hat{\theta}$. Доверительный интервал:

$$\left(\hat{X_{n+1}} - t_{\frac{1+\gamma}{2}}(n-k)\sqrt{\frac{RSS}{n-k}\left(1 + Z_{n+1}A^{-1}Z_{n+1}^T\right)};\right.$$
$$\left.\hat{X_{n+1}} + t_{\frac{1+\gamma}{2}}(n-k)\sqrt{\frac{RSS}{n-k}\left(1 + Z_{n+1}A^{-1}Z_{n+1}^T\right)}\right)$$

```
#### ДИ для отклика
new_data = np.array([1, 120, 10, 100, 50])
pred_results = results.get_prediction(new_data)
#pred_results.predicted_mean = y_hat_new
pred_results.conf_int()
```

7.1.6 Общая линейная гипотеза

$H_0 : T\theta = \tau$, где T - $m \times k$ матрица, $m \leq k$, $rkT = m$, $\theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_k \end{pmatrix}$, а $\tau = \begin{pmatrix} \tau_1 \\ \vdots \\ \tau_m \end{pmatrix}$.

$$\begin{pmatrix} T_{11} & \ldots & T_{1k} \\ \vdots & \ddots & \vdots \\ T_{m1} & \ldots & T_{mk} \end{pmatrix} \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_k \end{pmatrix} = \begin{pmatrix} \tau_1 \\ \vdots \\ \tau_m \end{pmatrix}$$

Статистика:

$$F = \frac{\left(T\hat{\theta} - \tau\right)^T B^{-1} \left(T\hat{\theta} - \tau\right)}{RSS} \cdot \frac{n-k}{m} \overset{\mathrm{H_0}}{\sim} F(m, n-k)$$
$$\text{где } B = T\left(Z^T Z\right)^{-1} T^T$$
$$C_{\text{кр}} = \left[f_{1-\alpha}(m, n-k); +\infty\right)$$

7.1.7 Критерий значимости регрессии

$$X_i = \theta_1 + Z_{i2}\theta_2 + \cdots + Z_{ik}\theta_k + \varepsilon_i$$

$H_0 : \theta_2 = \cdots = \theta_k = 0$, $H_1 : \exists i \in \{2, \ldots, k\} : \ \theta_i \neq 0$.

$$T\theta = \tau, \ \tau = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \ T = \begin{pmatrix} 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{pmatrix}$$

$$F = \frac{\sum\limits_{i=1}^{n}\left(\hat{X}_i - \overline{X}\right)^2}{\sum\limits_{i=1}^{n}\left(X_i - \hat{X}_i\right)^2} \cdot \frac{n-k}{k-1} = \frac{ESS}{RSS} \cdot \frac{n-k}{k-1} \overset{\mathrm{H}_0}{\sim} F(k-1, n-k)$$

```
##### Критерий значимости регрессии
results.f_pvalue
results.fvalue
```

Пример

Задача 7.1. *В файле "Carseats.csv"представлены данные о продажах детских кресел в различных магазинах страны: Sales – количество проданных кресел, Advertising – бюджет, выделенный на рекламу, Price – цена, CompPrice - цена основного конкурента, Age – средний возраст населения.*

Охарактеризовать линейную зависимость продаж кресел от всех перечисленных выше показателей.

```
from statsmodels.regression.linear_model import OLS

data = pd.read_csv('Carseats.csv')
data.head()
x = add_constant(data[['CompPrice', 'Advertising', 'Price', 'Age']])
y = data['Sales']
ols = OLS(y, x)
results = ols.fit()

#### Оценки параметров регрессии
results.params
y_hat = ols.predict(results.params, x)
print(y - y_hat) #вектор остатков регрессии

#### Построение прогноза
y_hat_new = ols.predict(results.params, [1, 120, 10, 100, 50])
print(y_hat_new)

#### Оценка дисперсии
RSS = results.ssr
k = 5
n = len(data['Sales'])
sigma2_hat = RSS/(n-k)
print(sigma2_hat)

##### ДИ параметров нормальной регрессии
conf_intervals = results.conf_int(alpha=0.05)
print (conf_intervals[1][2])

#### ДИ для отклика
new_data = np.array([1, 120, 10, 100, 50])
pred_results = results.get_prediction(new_data)
#pred_results.predicted_mean = y_hat_new
pred_results.conf_int()

##### Проверка значимости признаков
```

```
results.pvalues
results.tvalues
```

7.2 Коэффициент детерминации и анализ остатков

Коэффициент детерминации

Теория

Имеем: $X_i = 1 \cdot \theta_1 + \dots + Z_{ik} \cdot \theta_k + \varepsilon_i$.

$$R^2 = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS}$$

$$ESS = \sum_{i=1}^{n} \left(\hat{X}_i - \overline{X_i}\right)^2 \text{ - explained sum of squares}$$

$$TSS = ESS + RSS \text{ - total sum of squares}$$

$$\sum_{i=1}^{n} \left(X_i - \overline{X_i}\right)^2 = \sum_{i=1}^{n} \left(\hat{X}_i - \overline{X_i}\right)^2 + \sum_{i=1}^{n} \left(X_i - \hat{X}_i\right)^2$$

Пример

```
#### Коэффициент детерминации
results.rsquared
```

Анализ остатков

Теория

$$e_i = X_i - \hat{X}_i = X_i - (Z_{i1}\hat{\theta}_1 + \dots + Z_{ik}\hat{\theta}_k) \text{ - residuals}$$

$$e = \begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix} = X - Z\hat{\theta} = X - \underbrace{Z\left(Z^T Z\right)^{-1} Z^T}_{=H} X = (\mathbb{I}_n - H)\, X$$

$Ee = 0$, т.к. $\hat{\theta}$ несмещенная. $De = D\left((\mathbb{I}_n - H)\, X\right) = \sigma^2 \left(\mathbb{I}_n - H\right)$. Тогда если $\varepsilon_i \sim N(0, \sigma^2)$, то $e_i \sim N(0, \sigma^2(1 - h_{ii}))$.

Стьюдентизированные и стандартизированные остатки

$$t_i = \frac{e_i}{\sqrt{\frac{RSS}{n-k} \cdot \sqrt{1 - h_{ii}}}} \sim t(n-k)$$

Если $n >> k$, то $h_{ii} \simeq 0$ и $t(n-k) \to N(0,1)$. Тогда получаем стандартизированные остатки:

$$\tilde{e}_i = \frac{e_i}{\sqrt{RSS}n-k}$$

Пример

```
#### Анализ остатков
influence = results.get_influence()
  #Обычные"" остатки e_i
residuals = influence.resid
print(residuals[0:5])
  #Стьюдентизированные остатки
stud_residuals = influence.resid_studentized
print(stud_residuals[0:5])
  #Стандартизированные остатки
stand_residuals = residuals/np.sqrt(sigma2_hat)
print(stand_residuals[0:5])
  #Визуальный анализ остатков
plt.scatter(y_hat, stand_residuals)
from scipy.stats import probplot
probplot(stand_residuals, plot=plt);
```

7.3 Проверка гомоскедастичности

7.3.1 Общие сведения

Теория

$$X_i = \theta_1 + Z_{i2}\theta_2 + \dots + Z_{ik}\theta_k + \varepsilon_i$$

$H_0 : D\varepsilon_i = \sigma^2 \ \forall i$ (гомоскедастичность).
$H_1 : \exists i, j : \ D\varepsilon_i \neq D\varepsilon_j$ (гетероскедастичность).

7.3.2 Тест Уайта

Теория

$E\varepsilon_i^2 \xrightarrow{?} Ee_i^2$
Рассмотрим вспомогательную регрессию:

$$e_i^2 = \alpha_1 + \sum \alpha_j Z_{ij} + \sum \beta_j Z_{ij}^2 + \sum \gamma_{jl} Z_{ij} Z_{il} + u_i$$

Количество факторов вспомогательной регрессии равно $m-1$.

$$R^2 = \frac{\sum\limits_{i=1}^{n}\left(\hat{e_i^2} - \overline{e^2}\right)^2}{\sum\limits_{i=1}^{n}\left(e_i^2 - \overline{e^2}\right)^2},\ T = n \cdot R^2 \overset{\text{H}_0}{\sim} \chi^2(m-1)$$

$$C_{\text{кр}} = \left[\chi^2_{1-\alpha}(m-1); +\infty\right)$$

Пример

```
het_white(residuals, x)
het_white?
```

7.3.3 Тест Голдфельда-Квандта

Теория

Рассмотрим e_i. Гипотеза: $D\varepsilon_i$ возрастает, когда фактор возрастает. Алгоритм состоит из следующих шагов:

1. упорядочиваем $\begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}$ по росту фактора
2. делим на три группы с размерами $n_1, n_2, n_3:\ n = n_1 + n_2 + n_3,\ n_1 = n_3$

Если предположение верно, то $\dfrac{RSS_1}{n_1 - k} << \dfrac{RSS_3}{n_3 - k}$, иначе наоборот.

Можно воспользоваться F-тестом:

$$F = \frac{RSS_3 \cdot (n_1 - k)}{RSS_1 \cdot (n_3 - k)} \overset{\text{H}_0}{\sim} F(n_3 - k, n_1 - k)$$

$$C_{\text{кр}} = \left[f_{1-\alpha}(n_3 - k, n_1 - k); +\infty\right)$$

Пример

```
het_goldfeldquandt(y, x, idx = 3) #idx = 3: номер фактора
het_goldfeldquandt?
```

Задача 7.2. *В файле "House_prices.csv" представлены характеристики различных домов (стоимость, площадь, количество комнат, год постройки и тп, описание признаков можно найти по ссылке Ames Housing dataset).*

Изучить линейную зависимость стоимости домов (SalePrice) от всех остальных показателей.

```python
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as st
import seaborn as sns
sns.set()
from statsmodels.regression.linear_model import OLS
from statsmodels.tools.tools import add_constant
from scipy.stats import probplot
from scipy.stats import jarque_bera
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer

data = pd.read_csv('House_prices.csv')
```

7.4 Пропуски в данных

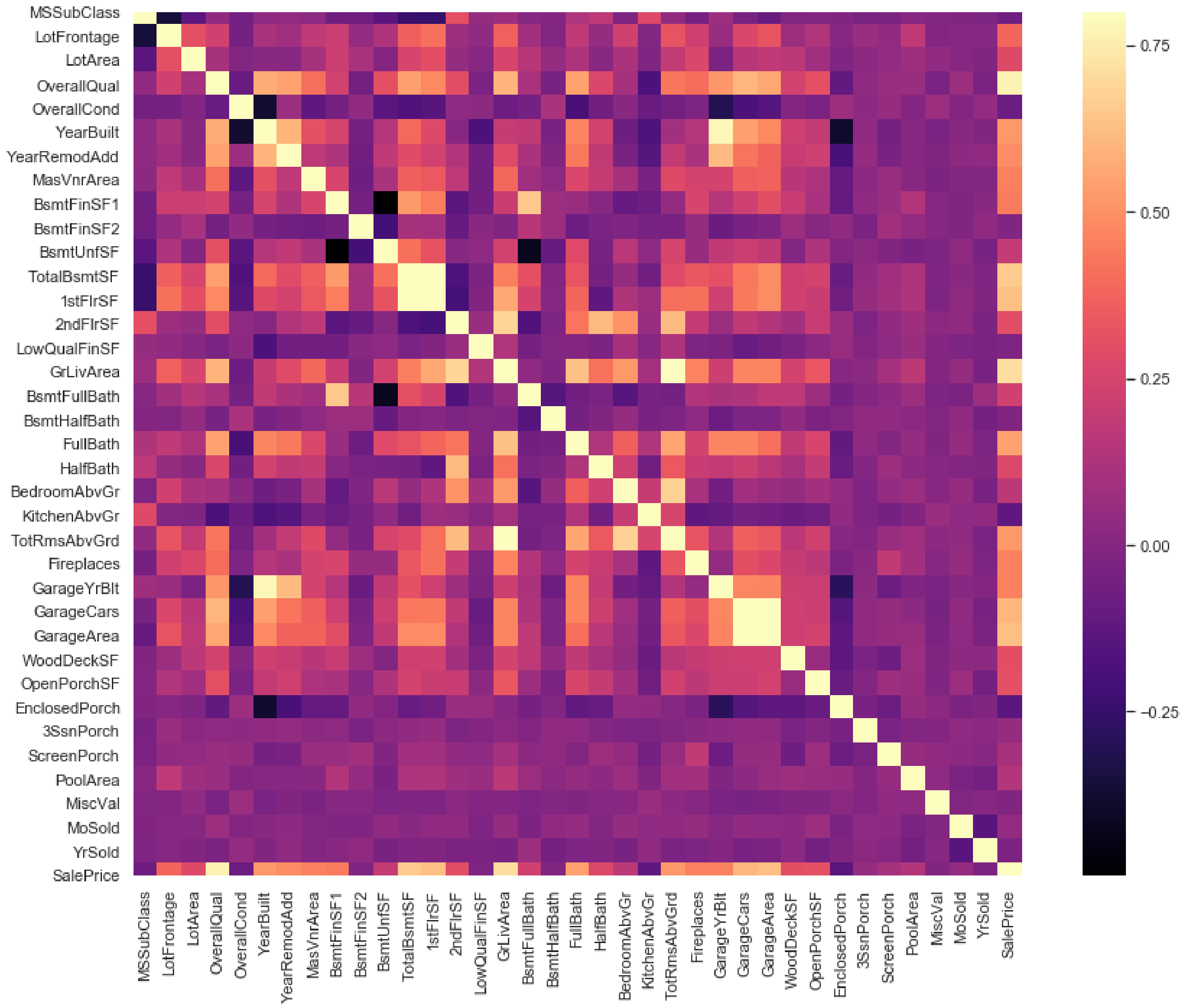
Часто в реальных данных не для всех объектов известно значение того или иного признака. Такие объекты нужно обрабатывать прежде, чем приступать к построению линейной регрессии. Для каждого признака посмотрим, в какой доле объектов отсутствует значение. Пропуски заполним медианным значением

```python
fill = data.median(axis=0) #axis=0: по столбцам
data = data.fillna(value=fill)
```

7.5 Проверка на мультиколлинеарность

Посмотрим на матрицу корреляций признаков и целевой переменной SalePrice

```python
corr = data.corr()
plt.figure(figsize=(15, 11))
sns.heatmap(corr, vmax=.8, square=True, cmap='magma');
```



Чем светлее ячейка, соответствующая паре признаков $(feature_i, feature_j)$, тем больше корелляция между ними.

Посмотим значения корелляций в численном виде.

```
corr_df = corr.unstack().to_frame().reset_index()
#corr.unstack(): сделали двухуровневый индекс, получили series (1 столбец со сложным
 индексом)
#reset_index(): превратили индекс в значения ячеек
new_coor_table = corr_df[corr_df.level_0!=corr_df.level_1].sort_values(0,
 ascending=False)
#corr_df.level_0!=corr_df.level_1 : убираем пары с одинаковыми признаками
```

Если есть сильно скоррелированны, то выбросим из каждой пары по одному признаку. Например, хотим выбросить признаки 'TotalBsmtSF', 'GarageYrBlt', 'GarageCars', 'TotRmsAbvGrd', тогда:

```
data.drop(['TotalBsmtSF', 'GarageYrBlt', 'GarageCars', 'TotRmsAbvGrd'], 1,
 inplace=True)
```

Строим регрессию.

```
x = add_constant(data.drop(['SalePrice'], 1))
y = data['SalePrice']
```

```
ols = OLS(y, x)
results = ols.fit()
RSS = results.ssr
n, k = x.shape[0], x.shape[1]
sigma2_hat = RSS/(n-k)
```

Проверим остатки на нормальность:

```
probplot(stand_residuals, plot=plt);
sns.boxplot(data=stand_residuals, orient="h");
```

Рис. 7.1: probplot

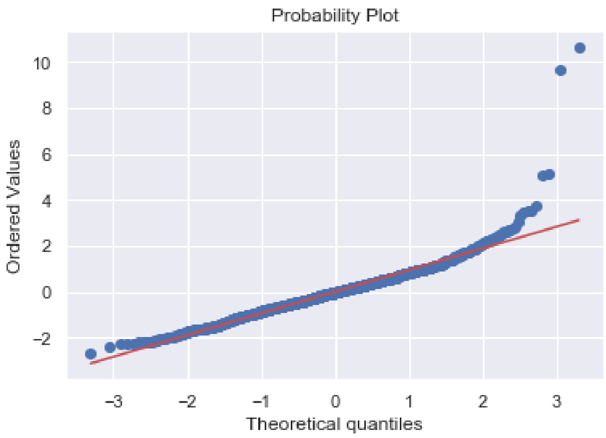
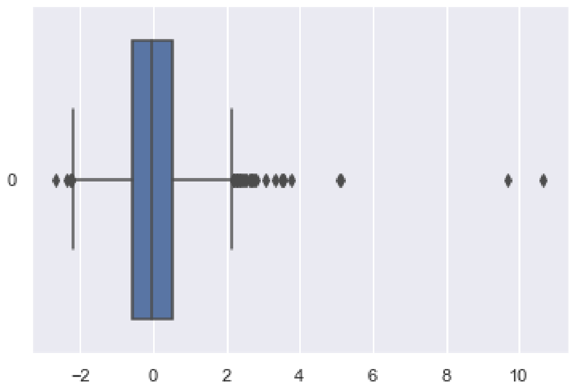


Рис. 7.2: boxplot



Видим, что не очень хорошо все ложится на прямую в случае qqplot, для boxplot есть выбросы $\Longrightarrow$ есть подозрения на невыаолнение условия нормальности остатков. Разочаруемся в предположении о нормальности до конца:

```
jarque_bera(stand_residuals)
```

Результат : $Jarque\_beraResult(statistic = 16543.863883530954, pvalue = 0.0)$, то есть остатки не есть нормальные. Если не хотим применять результаты нормальное регрессии, то это не проблема. Что делать в ином случае? Преобразуем данные: пытаемся брать логорифмы, експоненту и т.д. в надежде на то, что новая реграссия будет удовлетворять условиям нормальности. Можно применить преобразование Бокса - Кокса. Чаще всего используют логарифм.

```
ln_y = np.log(y)
ln_ols = OLS(ln_y, x)
ln_results = ln_ols.fit()
ln_RSS = ln_results.ssr
n, k = x.shape[0], x.shape[1]
ln_sigma2_hat = ln_RSS/(n-k)
ln_influence = ln_results.get_influence()
ln_residuals = ln_influence.resid
ln_stand_residuals = ln_residuals/np.sqrt(ln_sigma2_hat)
probplot(ln_stand_residuals, plot=plt);
sns.boxplot(data=ln_stand_residuals, orient="h");
jarque_bera(ln_stand_residuals)
```

Рис. 7.3: probplot

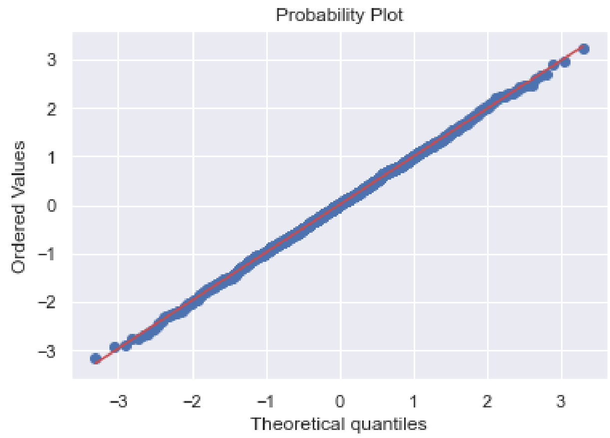
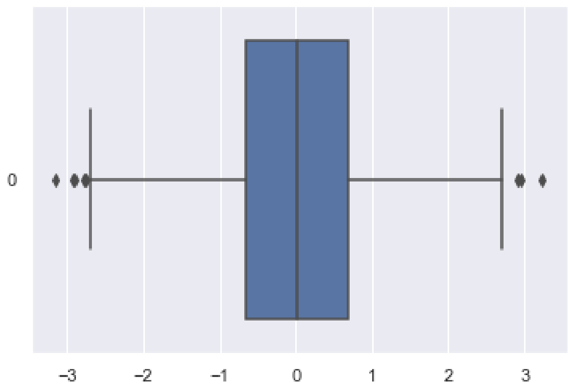


Рис. 7.4: boxplot



Результат теста Харке - Бера: $Jarque\_beraResult(statistic = 0.06510310527385516, pvalue = 0.9679725470077343)$, то есть теперь имеем нормальную регрессию.

Замечание. *Помним о том, что мы работаем с логарифмом регрессии, поэтому, когда хотим проводить какие-либо сравнения с исходными данными, необходимо произвести обратное преобразование.*

7.6 Отбор признаков. Информационные критерии AIC и BIC

Хотим отобрать признаки, для этого нам необходимо сравнивать модели, построенные на разных наборах признаков $\{x_{i_k}\}_k$ и $\{x_{i_s}\}_s$. Для этого используются информационные критерии AIC (информационный критерий Акаике) и BIC (Байесовский информационный критерий):

$$AIC = 2k + n\log(RSS),$$
$$BIC = \log(n)k + n\log(RSS),$$

где $RSS = \sum\limits_{i=1}^{n}(x_i - \hat{x}_i)^2,\ \hat{x}_i = z_i\hat{\theta},\ \hat{\theta} = (Z^TZ)^{-1}Z^TX,\ X = (x_1, \dots, x_n)$ – вектор наблюдений, $Z = (z_{ik})$ – матрица факторов (факторов k штук, имеем n наблюдений). Модель с меньшим AIC/BIC - лучшая.

Замечание. *Из формулы для подсчета AIC и BIC видно, что BIC штрафует за добавление новых признаков сильнее, чем AIC, поэтому подбор признаков, основанный на BIC, как правило, всегда исключает больше признаков, чем при подборе признаков, основанном на AIC.*

Замечание. *Данные формулы лучше использовать для нормальной регрессии.*

```python
def select_best_combination(y, x, metric): #metric: 'aic' или 'bic'
    current_factors = x.columns.to_list() #сначала создаем список всех наименований столбцов
    ols = OLS(y, x[current_factors])
    results = ols.fit()
    metric_base = getattr(results, metric)

    while 1 == 1:
        res = pd.Series(index=current_factors) #создаем Series с индексом current_factors и значениями = Nan
        for factor in current_factors:
            ols = OLS(y, x[list(set(current_factors)-{factor})]) #выкидываем по очереди один столбец
            results = ols.fit()
            res.loc[factor] = getattr(results, metric) #вместо Nan в res записываем метрику, соответствующую модели без данного столбца
        res = res.sort_values(ascending=True) #сортируем res по возрастанию
        if res.iloc[0] < metric_base:
            current_factors.remove(res.index.values[0])
            metric_base = res.iloc[0]
        else:
            break

    ols = OLS(y, x[current_factors])
    results = ols.fit()

    return current_factors, results

    current_factors, results = select_best_combination(ln_y, x, 'bic')
```

```
    set(x.columns.to_list()) - set(current_factors) #вывод факторов, основываясь на BIC
    current_factors, results = select_best_combination(ln_y, x, 'aic')
    set(x.columns.to_list()) - set(current_factors) #вывод факторов, основываясь на AIC
```

7.7 Деление выборки на обучающую и тестовую

Разделим выборку на обучающую и тестовую. Разделим случайным образом 75% на 25%:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=10)
```

У моделей из sklearn есть методы fit и predict. fit принимает на вход обучающую выборку и вектор целевых переменных и обучает модель, predict, будучи вызванным после обучения модели, возвращает предсказание на выборке.

```
lr = LinearRegression() #по умолчанию в модели регрессии есть константа
lr.fit(x_train, y_train)

y_hat_test = lr.predict(x_test)
print('Using Y:')
print('Test MSE %.3f' % mean_squared_error(y_test, y_hat_test))
print('Test R2 %.3f' % r2_score(y_test, y_hat_test))

y_hat_train = lr.predict(x_train)
print("Train MSE = %.3f" % mean_squared_error(y_train, y_hat_train))
print("Train R2 = %.3f" % r2_score(y_train, y_hat_train))
```

Вывод:
Using Y:
Test MSE 1304270005.734
Test R2 0.771
Train MSE = 1342194893.254
Train R2 = 0.813
Если будем использовать логарифмирование данных, получим

```
lr.fit(x_train, np.log(y_train))
y_hat_test = np.exp(lr.predict(x_test))
print('Using logY:')
print('Test MSE %.3f' % mean_squared_error(y_test, y_hat_test))
print('Test R2 %.3f' % r2_score(y_test, y_hat_test))
```

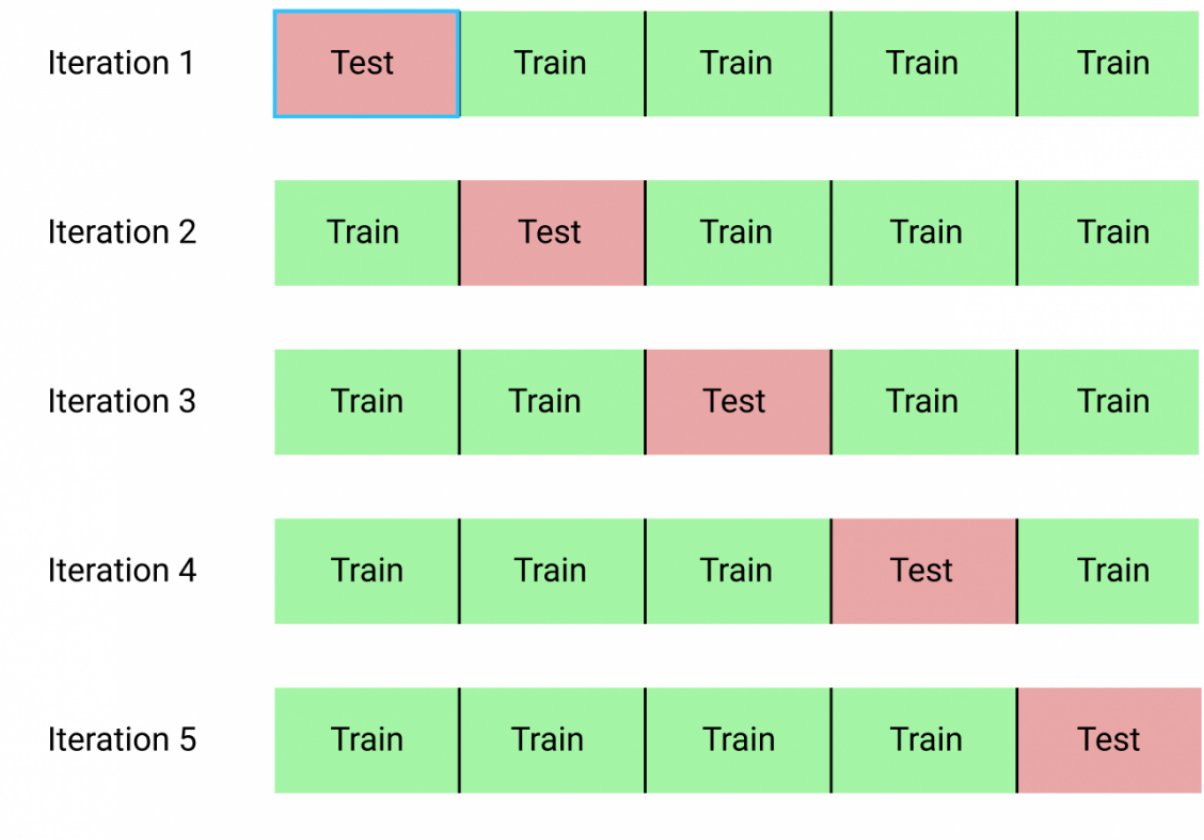
Вывод:
Using logY:
Test MSE 866920715.293
Test R2 0.848
Итак, мы обучили модель и посчитали ее качество на тестовой выборке.

7.8 Кросс-валидация

Принцип кросс-валидации изображен на рисунке. Берем данные, делим на k групп равного объема (k – количество фолдов). Делаем все то же самое, что ранее для train и test k раз: для i-го фолда $i = 1, \ldots, k$ считаем метрику $metric_i$ на $test_i$. Итоговое значение качества модели по кросс - валидации есть значение.

$$\frac{\sum_{i=1}^{k} metric_i}{k}.$$

Рис. 7.5: 5 фолдов




```
cv_scores = cross_val_score(lr, x, y, cv=10, scoring="neg_mean_squared_error")
```

Если мы выведем *cv_scores*, то результаты получились отрицательными. Это соглашение в sklearn (скоринговую функцию нужно максимизировать). Поэтому все стандартные скореры называются *neg_**, например, *neg_mean_squared_error*.

```
print("Mean CV MSE = %.4f" % np.mean(-cv_scores)) #итоговое значение качества
 модели по кросс - валидации
```

В нашем примере она равна Mean CV MSE = 1494450970.6308. Мы всегда можем определить свою метрику и использовать ее, например, в *cross_val_score*. Для этого нужно воспользоваться *sklearn.metrics.make_scorer*. Ниже показано, как можно задать свою метрику на примере метрики R^2 (на самом деле, она прописана в sklearn: #scoring="r2"в функции *cross_val_score*)

```
def r2_squared(y_true, y_pred):
    r2_coef = r2_score(y_true, y_pred)
    return r2_coef

r2_scorer = make_scorer(r2_squared, greater_is_better=True) #greater_is_better влияет на знаки метрик в cv_scores
cv_scores = cross_val_score(lr, x, y, cv=10, scoring=r2_scorer)
print("Mean CV R2 = %.4f" % np.mean(cv_scores))

cv_scores = cross_val_score(lr, x, ln_y, cv=10, scoring=r2_scorer)
print("Mean CV R2 = %.4f" % np.mean(cv_scores))
```

Вывод для не логарифма: Mean CV R2 = 0.7825. Для $\log y$: Mean CV R2 = 0.8504.

7.8.1 Отбор признаков с помощью кросс-валидации (greedy algorithm)

Вместо критериев AIC и BIC возьмем итоговое значение качества модели по кросс - валидации:

```
def calc_kfold_validation(x, y):
    lr = LinearRegression()
    cv_scores = cross_val_score(lr, x, y, cv=5, scoring="neg_mean_squared_error")
    return np.mean(-cv_scores)

def select_best_combination(x, y):
    current_factors = x.columns.to_list() #сначала создаем список всех наименований столбцов
    metric_base = calc_kfold_validation(x[current_factors], y)

    while 1 == 1:
        res = pd.Series(index=current_factors) #создаем Series с индексом current_factors и значениями = Nan
        for factor in current_factors:
            res.loc[factor] = calc_kfold_validation(x[list(set(current_factors)-{factor})], y)
                #вместо Nan в res записываем метрику, соответствующую модели без данного столбца
        res = res.sort_values(ascending=True) #сортируем res по возрастанию
        if res.iloc[0] < metric_base:
            current_factors.remove(res.index.values[0])
            metric_base = res.iloc[0]
        else:
            break

    return current_factors, calc_kfold_validation(x[current_factors], y)

ln_y_train, ln_y_test = np.log(y_train), np.log(y_test)
current_factors, _ = select_best_combination(x_train, ln_y_train) #делаем отбор признаков на train по( ln_y_train)
set(x.columns.to_list()) - set(current_factors)
```

Сравним модель до и после отбора признаков.

```
k = x_train.shape[1] + 1 #количество факторов с( учетом const)
n = x_test.shape[0]
lr = LinearRegression()
lr.fit(x_train, ln_y_train)

ln_y_hat_test = lr.predict(x_test)
print('Test R2 %.3f' % r2_score(ln_y_test, ln_y_hat_test))
print('Test AIC %.3f' % (2*k + n * np.log(np.sum((ln_y_test - ln_y_hat_test) **
    2))) )
```

Вывод:
Test R2 0.858
Test AIC 835.847

```
k = x_train[current_factors].shape[1] + 1 #количество факторов с( учетом const)
n = x_test.shape[0]
lr = LinearRegression()
lr.fit(x_train[current_factors], ln_y_train)

ln_y_hat_test = lr.predict(x_test[current_factors])
print('After greedy algorithm')
print('Test R2 %.3f' % r2_score(ln_y_test, ln_y_hat_test))
print('Test AIC %.3f' % (2*k + n * np.log(np.sum((ln_y_test - ln_y_hat_test) **
    2))) )
```

Вывод:
After greedy algorithm
Test R2 0.856
Test AIC 821.092

7.9 Гребневая регрессия

Было: $X_i = \theta_1 + Z_{i2}\theta_2 + \dots + Z_{ik}\theta_k + \varepsilon_i =< Z_i, \theta > +\varepsilon_i$.
МНК: $S(\theta) = (X - Z\theta)^T (X - Z\theta) \to \min\limits_{\theta}$.

Теорема 7.2. *Если $A = Z^T Z$ не вырождена, то $\exists!$ о.н.к. $\hat{\theta} = \left(Z^T Z\right)^{-1} Z^T X$.*

Если A необратима, то мы получаем проблему мультиколлинераности, то имеются коррелирующие (линейно зависимые) признаки. Т.е. если A не обратима, то задача $S(\theta) \to \min$ имеет бесконечное число решений, то регрессия получается переобученная.
Т.к. $\exists$ линейная зависимость признаков, то $\exists$ вектор $\lambda : < Z_i, \lambda >= 0 \ \forall i \ \Rightarrow$

$$X_i =< \theta, Z_i > +\varepsilon_i$$
$$< \hat{\theta} + c\lambda, Z_i >=< \hat{\theta}, Z_i > +c \cdot \underbrace{\sqrt{< \lambda, Z_i >}}_{=0} =< \hat{\theta}, Z_i >$$

7.10 Регуляризация

Вместо задачи минимизации $S(\theta)$ рассматривается минимизация следующего функицонала:

$$S_\alpha = S(\theta) + \alpha \cdot R(\theta) \to \min_\theta$$

где α - параметр, $R(\theta)$ - регуляризатор

$L_2: \ R(\theta) = \sum\limits_{j=2}^{k} \theta_j^2$ - получаем гребневую регрессию.

$L_1: \ R(\theta) = \sum\limits_{j=2}^{k} |\theta_j|.$

В случае L_2 регуляризации $\hat{\theta} = \left(Z^T Z + \alpha \cdot \mathbb{I}\right)^{-1} Z^T X.$

В sklearn есть несколько классов, реализующих линейную регрессию:

1. LinearRegression — «классическая» линейная регрессия с оптимизацией MSE
2. Ridge — линейная регрессия с оптимизацией MSE и l_2-регуляризацией
3. Lasso — линейная регрессия с оптимизацией MSE и l_1-регуляризацией

```
lr = LinearRegression()
lr = Ridge() #по умолчанию alpha=1.0, fit_intercept=True
lr.fit(x_train, y_train)

y_hat_test = lr.predict(x_test)
print('Test MSE %.3f' % mean_squared_error(y_test, y_hat_test))
print('Test R2 %.3f' % r2_score(y_test, y_hat_test))

#для сравнения
y_hat_train = lr.predict(x_train)
print("Train MSE = %.3f" % mean_squared_error(y_train, y_hat_train))
print("Train R2 = %.3f" % r2_score(y_train, y_hat_train))
```

7.11 Отбор признаков

Посмотрим на то, какие признаки оказались самыми "сильными". Для этого визуализируем коэффициенты регрессии, соответствующие признакам.

```
sorted(zip(a, b, c), reverse=True) #сортировка по ым1 элементам по убыванию

def show_weights(features, weights, scales):
    fig, axs = plt.subplots(figsize=(14, 10), ncols=2) #ncols=2: два графика два(
  столбца)
    sorted_weights = sorted(zip(weights, features, scales), reverse=True) #
 сортировка по весам по убыванию
    weights = [x[0] for x in sorted_weights]
    features = [x[1] for x in sorted_weights]
    scales = [x[2] for x in sorted_weights]
    sns.barplot(y=features, x=weights, ax=axs[0])
```

```
    axs[0].set_xlabel("Weight")
    sns.barplot(y=features, x=scales, ax=axs[1])
    axs[1].set_xlabel("Scale")
    plt.tight_layout()

show_weights(x_train.columns, lr.coef_, x_train.std())
```

Будем масштабировать наши признаки. Это сделает нашу регуляризацию более честной: теперь все признаки будут регуляризоваться в равной степени.

Для этого воспользуемся трансформером StandardScaler. Трансформеры в sklearn имеют методы fit и transform (а еще fittransform). Метод fit принимает на вход обучающую выборку и считает по ней необходимые значения (среднее и стандартное отклонение каждого из признаков). transform применяет преобразование к переданной выборке.

```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test) #на test делаем только transform!

lr = Ridge()
lr.fit(x_train_scaled, y_train)

show_weights(x_train.columns, lr.coef_, x_train_scaled.std(axis=0)) #type(
  x_train_scaled
```

7.12 Подбор гиперпараметров

Наряду с параметрами, которые модель оптимизирует на этапе обучения (коэффициенты регрессии), у модели есть и гиперпараметры. У нашей модели это alpha — коэффициент регуляризации.

Будем пользоваться кросс-валидацией для подбора гиперпараметров.

Подберем alpha по логарифмической сетке, чтобы узнать оптимальный порядок величины.

```
from sklearn.metrics import make_scorer

def r2_squared(y_true, y_pred):
    r2_coef = r2_score(y_true, y_pred)
    return r2_coef

r2_scorer = make_scorer(r2_squared, greater_is_better=True)

alphas = np.logspace(-2, 3, 20)
searcher = GridSearchCV(Ridge(), [{"alpha": alphas}], scoring=r2_scorer, cv=10)
    #scoring="r2"
searcher.fit(x_train_scaled, y_train)

best_alpha = searcher.best_params_["alpha"]
print("Best alpha = %.4f" % best_alpha)

plt.plot(alphas, searcher.cv_results_["mean_test_score"])
plt.xscale("log")
plt.xlabel("alpha")
```

```
plt.ylabel("CV score")

lr = Ridge(best_alpha)
lr.fit(x_train_scaled, y_train)

y_hat_test = lr.predict(x_test_scaled)
print('Test MSE %.3f' % mean_squared_error(y_test, y_hat_test))
print('Test R2 %.3f' % r2_score(y_test, y_hat_test))

lr.intercept_

lr = Ridge()
lr.fit(x_train_scaled, y_train)

y_hat_test = lr.predict(x_test_scaled)
print('Test MSE %.3f' % mean_squared_error(y_test, y_hat_test))
print('Test R2 %.3f' % r2_score(y_test, y_hat_test))

lr.intercept_ #обратим внимание на то, что константа в регрессии не регуляризуется
```

Пример

Задача 7.3. *В файле «House-prices-corrected.csv» представлены характеристики различных домов (стоимость, площадь, количество комнат, год постройки и тп, описание признаков можно найти по ссылке Ames Housing dataset).*

Изучить линейную зависимость стоимости домов (SalePrice) от всех остальных показателей.

```
#from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

data = pd.read_csv('House_prices_corrected.csv')
x = data.drop(['SalePrice'], 1)
y = data['SalePrice']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
  random_state=10)

#lr = LinearRegression()
lr = Ridge() #по умолчанию alpha=1.0, fit_intercept=True
lr.fit(x_train, y_train)

y_hat_test = lr.predict(x_test)
print('Test MSE %.3f' % mean_squared_error(y_test, y_hat_test))
print('Test R2 %.3f' % r2_score(y_test, y_hat_test))

#для сравнения
y_hat_train = lr.predict(x_train)
print("Train MSE = %.3f" % mean_squared_error(y_train, y_hat_train))
print("Train R2 = %.3f" % r2_score(y_train, y_hat_train))
```

8 Список возможных importов

import numpy as np
import scipy as sp
import pandas as pd
import scipy.stats as st
import scikit_posthocs as sp (post hoc tests)
import pingouin as pg
from scipy.stats import norm (нормальное распределение)
from scipy.stats import uniform (равномерное распределение)
from scipy.stats import expon (экспоненциальное распределение)
from scipy.stats import beta (бета-распределение)
from scipy.stats import cauchy (распределение Коши)
from scipy.stats import t (распределение Стьюдента)
from scipy.stats import f (распределение Фишера)
from scipy.stats import chi2 (распределение хи-квадрат)
from scipy.stats import gamma (гамма распределение)
from scipy.stats import binom (биномиальное распределение)
from scipy.stats import ttest_1samp (t-test 1 sample)
from scipy.stats import ttest_rel (t-test для парных выборок)
from scipy.stats import ttest_ind (t-test с поправкой на неравенство дисперсий)
from scipy.stats import binom_test (биномильный критерий)
from scipy.stats import wilcoxon (знако-ранговый критерий Вилкоксона)
from statsmodels.stats.diagnostic import lilliefors (тест Лиллиефорса)
from scipy.stats import mannwhitneyu (критерий Манна и Уитни)
from scipy.stats import ks_2samp (критерий Смирного)
from scipy.stats import anderson_ksamp (критерий Андерсона-Дарлинга)
from scipy.stats import kruskal (критерий Краскела-Уоллиса)
from scipy.stats import bartlett (критерий Бартлетта)
from scipy.stats import f_oneway (ANOVA)
from scipy.stats import friedmanchisquare (критерий Фридмана)

from statsmodels.stats.anova import AnovaRM (критерий Фишера aka ANOVA RM)

()

()

()

()

()

Литература

[1] Курс лекций и семинаров А.А.Муромской, механико-математический факультет МГУ им. М.В.Ломоносова, 2021 г.

[2]

[3]

[4]

[5]