Статистика, прикладной поток

Практическое задание 7

В данном задании вы потренируетесь над практическим применением процедуры проверки статистических гипотез на примере критериев согласия, а так же рассмотрите различные модели линейной регрессии.

Правила:

- Дедлайн **15 декабря 23:59**. После дедлайна работы не принимаются кроме случаев наличия уважительной причины.
- Выполненную работу нужно отправить на почту `mipt.stats@yandex.ru` , указав тему письма `"[applied] Фамилия Имя - задание 7"` . Квадратные скобки обязательны. Если письмо дошло, придет ответ от автоответчика.
- Прислать нужно ноутбук и его pdf-версию (без архивов). Названия файлов должны быть такими: `7.N.ipynb` и `7.N.pdf` , где `N` - ваш номер из таблицы с оценками.
- Решения, размещенные на каких-либо интернет-ресурсах не принимаются. Кроме того, публикация решения в открытом доступе может быть приравнена к предоставлении возможности списать.
- Для выполнения задания используйте этот ноутбук в качестве основы, ничего не удаляя из него.
- Никакой код из данного задания при проверке запускаться не будет.

Баллы за задание:

- Задача 1 - 8 баллов **О2**
- Задача 2 - 12 баллов **О3**
- Задача 3 - 12 баллов **О3**
- Задача 4 - 15 баллов **О2**
- Задача 5 - 10 баллов **О2**
- Задача 6 - 20 баллов **О3**

In [2]:
```
import numpy as np
import pandas as pd
import scipy.stats as sps
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.sandbox.stats.multicomp import multipletests
from tqdm import tqdm_notebook

from collections import Counter
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.datasets import load_boston
from scipy.linalg import eigvals

%matplotlib inline
```

Критерии согласия

При возникновении затруднений посмотрите в ноутбук с лекции.

Задача 1.

С помощью критерия хи-квадрат вам нужно проверить, правильно ли `scipy.stats` генерирует случайные величины.

1. Реализуйте критерий для генерации выборки $U\{1, \dots, 10\}$, аналогично разобранному на семинаре.

In [5]:
```
def is_uniform10(sample):
    f_obs = np.array(list(Counter(sample).values()))
    if f_obs.size > 10:
        return "is not U{1, .. ,10}"
    f_obs = np.concatenate((f_obs, np.zeros(10-f_obs.size)))
    f_exp = np.ones(10)*sample.size/10

    if sps.chisquare(f_obs, f_exp)[1] > 0.05:
        return "is U{1, .. ,10}"
    else:
        return "is not U{1, .. ,10}"
```

In [6]:
```
sample_size = 100

for i in [1,2,9,10,11,15]:
    sample = sps.randint(low=1, high=i+1).rvs(size=sample_size)
    print('U{1,..%d}'%(i), is_uniform10(sample))
```

```
U{1,..1} is not U{1, .. ,10}
U{1,..2} is not U{1, .. ,10}
U{1,..9} is not U{1, .. ,10}
U{1,..10} is U{1, .. ,10}
U{1,..11} is not U{1, .. ,10}
U{1,..15} is not U{1, .. ,10}
```

2. Проверьте, действительно ли код `sps.poisson(mu=5).rvs(size=1000)` генерирует выборку размера 1000 из пуассоновского распределения с параметром 5.

Указания

1. Не забудьте учесть условия применимости критерия хи-квадрат;
2. Для вычисления плотности воспользуйтесь `sps.poisson(mu=5).pmf` ;
3. Для подсчета количества элементов в разбиении на множества воспользуйтесь `np.unique(sample, return_counts=True)` ;
4. Не нужно писать сколь-либо универсальный код, который одним нажатием кнопки проверяет гипотезу с учетом всех условий применимости критерия. Расписывайте код поэтапно, на каждый логический этап свой код для *конкретной* реализации выборки. Так вам проще реализовать, а нам проще проверять.

In [7]:
```python
def is_poisson5(sample):
    k = 10 #12
    counted = np.unique(sample, return_counts=True)
    if (counted[0] < 0).any():
        return "is not poisson(5)"
    f_obs = []
    for i in range(counted[0].size):
        if counted[0][i] > k:
            f_obs[-1] += counted[1][i]
        else:
            f_obs.append(counted[1][i])

    f_exp = sps.poisson(mu=5).pmf(range(10))
    f_exp = np.append(f_exp, 1-f_exp.sum())*sample.size

    if sps.chisquare(f_obs, f_exp)[1] < 0.05:
        return "is not poisson(5)"
    else:
        return "sample is poisson(5)"

sample = sps.poisson(mu=5).rvs(size=1000)
is_poisson5(sample)
```

Out[7]:
```
'sample is poisson(5)'
```

Вывод:

Код `sps.poisson(mu=5).rvs(size=1000)` действительно генерирует выборку размера 1000 из пуассоновского распределения с параметром 5.

`sps.randint` также проходит критерий хи-квадрат.

Задача 2.

На лекциях и семинарах были разобраны следующие критерии проверки нормальности:

- Колмогорова
- Жарка-Бера
- Шапиро-Уилка

1. Данные критерии являются асимптотическими, и их реальное значение уровня значимости может отличаться от желаемого числа $\alpha = 0.05$. На семинарах был разобран метод оценки реального уровня значимости критерия. Посчитайте реальный уровень значимости этих критериев для размеров выборки от 5 до 100.

Подсказка:

Ваша реализация:

In [3]:
```
min_sample_size = 5
max_sample_size = 101
sample_count = 10000

is_reject_kolm = np.zeros((max_sample_size, sample_count))
is_reject_jb = np.zeros((max_sample_size, sample_count))
is_reject_shapiro = np.zeros((max_sample_size, sample_count))

for i in tqdm_notebook(range(min_sample_size, max_sample_size)):
    for j in range(sample_count):
        sample = sps.norm.rvs(size=i)
        is_reject_kolm[i, j] = sps.kstest(sample, 'norm')[1] < 0.05
        is_reject_jb[i, j] = sps.jarque_bera(sample)[1] < 0.05
        is_reject_shapiro[i, j] = sps.shapiro(sample)[1] < 0.05
```

100% 96/96 [34:40<00:00, 21.15s/it]

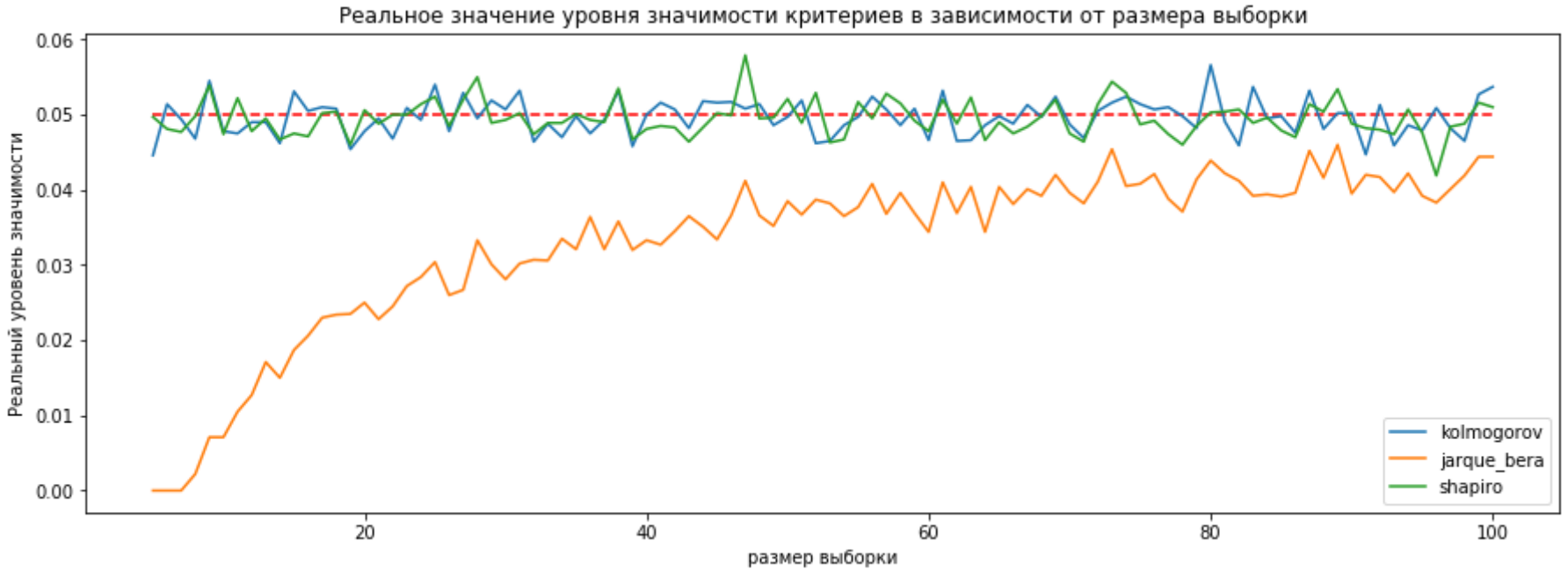
Поясните, почему вы выбрали такое значение `sample_count` .

На семинаре был разобрано, что sample_count должен иметь значени 1000000, чтобы точность была 0,001. Однако в таком случае вычисления занимали бы 5 часов. Так что я решила, что взять точность 0,01.

Нарисуйте на одном графике зависимость реального уровня значимости от размера выборки для каждого критерия. Пунктиром отметьте

уровень 0.05. Не забудьте добавить легенду и подписать оси.

In [4]:
```
grid = np.arange(min_sample_size, max_sample_size)
plt.figure(figsize=(15,5))
plt.hlines([0.05], min_sample_size, max_sample_size-1, color='r', linestyles ='dashed')
plt.plot(grid, is_reject_kolm.mean(axis=1)[min_sample_size:max_sample_size], label='kolmogorov')
plt.plot(grid, is_reject_jb.mean(axis=1)[min_sample_size:max_sample_size], label='jarque_bera')
plt.plot(grid, is_reject_shapiro.mean(axis=1)[min_sample_size:max_sample_size], label='shapiro')
plt.legend()
plt.title("Реальное значение уровня значимости критериев в зависимости от размера выборки")
plt.xlabel("размер выборки")
plt.ylabel("Реальный уровень значимости");
```



Какой можно сделать вывод?

Можно считать, что для критериев Кологорова и Шапиро реальный уровень значимости совпадает с теоретическим. Однако реальный уровень значимости критерия Жарка-Бера медленно стремится снизу к своему асимптотическому уровню значимости.

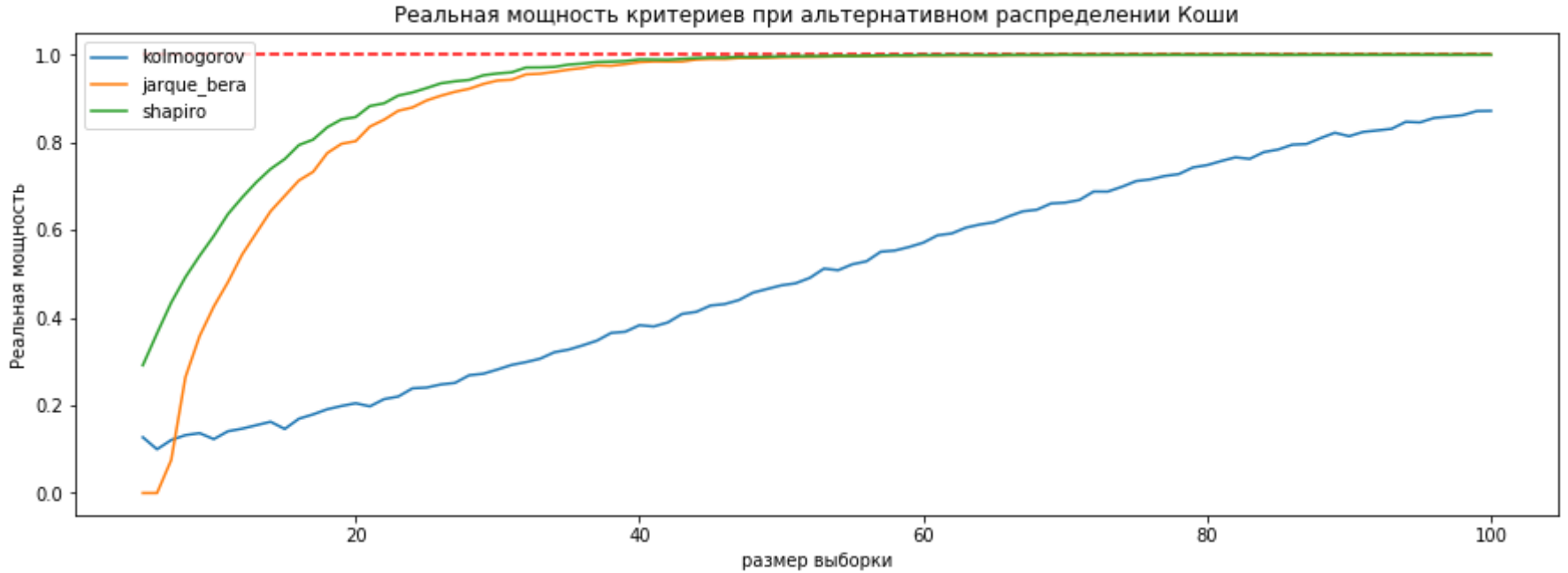
2. Аналогичным образом можно вычислять мощность критерия. Вычислите мощности критериев для размеров выборки от 5 до 100, если альтернативная гипотеза заключается в том, что выборка имеет стандартное распределение Коши. Иначе говоря, для критерия S надо приближенно посчитать $\beta_S(\mathrm{P})$, где P --- стандартное распределение Коши.

Нарисуйте на одном графике зависимость мощности критерия от размера выборки для каждого критерия. Не забудьте добавить легенду и подписать оси.

In [5]:
```
min_sample_size = 5
max_sample_size = 101
sample_count = 10000

is_reject_kolm = np.zeros((max_sample_size, sample_count))
is_reject_jb = np.zeros((max_sample_size, sample_count))
is_reject_shapiro = np.zeros((max_sample_size, sample_count))

for i in tqdm_notebook(range(min_sample_size, max_sample_size)):
    for j in range(sample_count):
        sample = sps.cauchy.rvs(size=i)
        is_reject_kolm[i, j] = sps.kstest(sample, 'norm')[1] < 0.05
        is_reject_jb[i, j] = sps.jarque_bera(sample)[1] < 0.05
        is_reject_shapiro[i, j] = sps.shapiro(sample)[1] < 0.05
```

100% 96/96 [36:08<00:00, 21.46s/it]

In [6]:
```
grid = np.arange(min_sample_size, max_sample_size)
plt.figure(figsize=(15,5))
plt.hlines([1], min_sample_size, max_sample_size-1, color='r', linestyles ='dashed')
plt.plot(grid, is_reject_kolm.mean(axis=1)[min_sample_size:max_sample_size], label='kolmogorov')
plt.plot(grid, is_reject_jb.mean(axis=1)[min_sample_size:max_sample_size], label='jarque_bera')
plt.plot(grid, is_reject_shapiro.mean(axis=1)[min_sample_size:max_sample_size], label='shapiro')
plt.legend()
plt.title("Реальная мощность критериев при альтернативном распределении Коши")
plt.xlabel("размер выборки")
plt.ylabel("Реальная мощность");
```



Сделайте вывод. Какой критерий является наиболее мощным при данной альтернативе?

Наиболее мощным является критерий Шапиро, критерий Жарка-Бера не сильно отличается от него по мощности. Однако критерий Колмогорова намного менее мощный.

3. Распределение Стьюдента является в некотором смысле обобщением нормального распределения (при бесконечном количестве степеней свободы) и распределения Коши (при одной степени свободы). Посчитайте приближенно мощности критериев для выборки

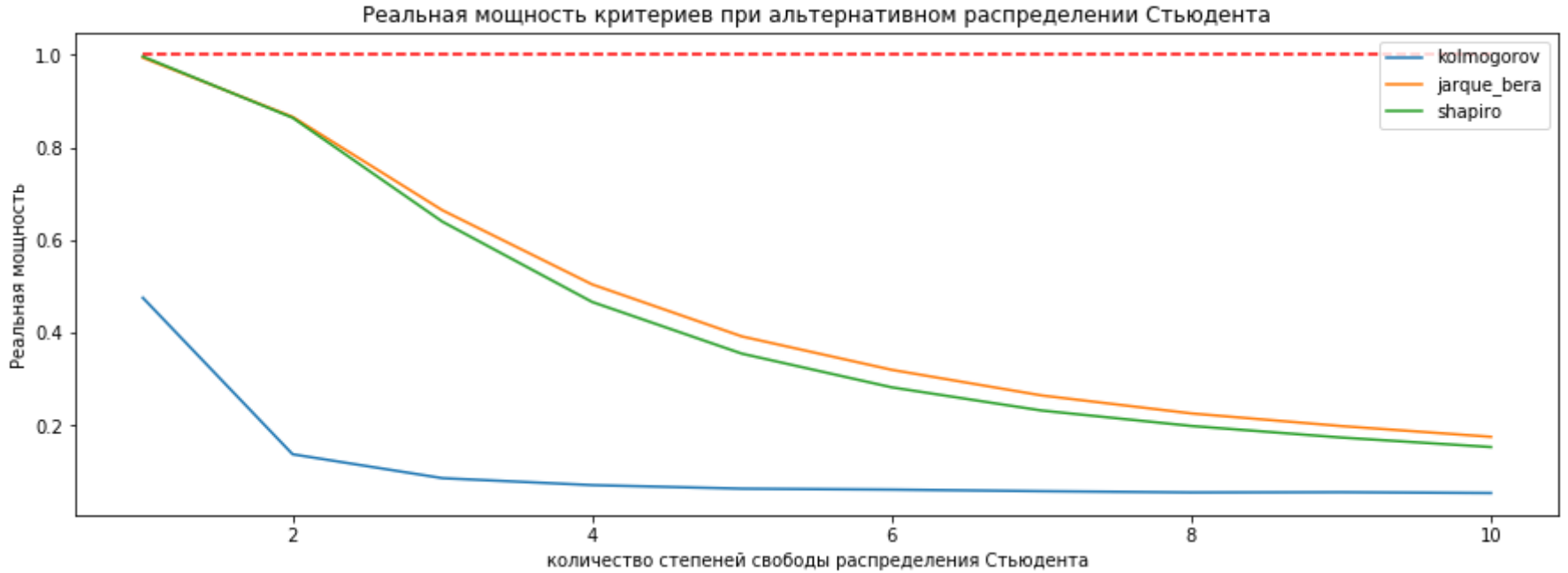
размера 50 если альтернатива заключается в том, что выборка имеет распределение Стьюдента. Поскольку мощность критерия является функцией от распределения из альтернативной гипотезы, вам нужно посчитать функцию. Посчитайте ее для целых значений степени свободы от 1 до 10.

Нарисуйте на одном графике зависимость мощности критерия от количества степеней свободы для каждого критерия. Не забудьте добавить легенду и подписать оси.

In [7]:
```
sample_size = 50
sample_count = 100000
min_dof = 1
max_dof = 11

is_reject_kolm = np.zeros((max_dof, sample_count))
is_reject_jb = np.zeros((max_dof, sample_count))
is_reject_shapiro = np.zeros((max_dof, sample_count))

for i in tqdm_notebook(range(min_dof, max_dof)):
    for j in range(sample_count):
        sample = sps.t(df=i).rvs(size=sample_size)
        is_reject_kolm[i, j] = sps.kstest(sample, 'norm')[1] < 0.05
        is_reject_jb[i, j] = sps.jarque_bera(sample)[1] < 0.05
        is_reject_shapiro[i, j] = sps.shapiro(sample)[1] < 0.05
```

100% 10/10 [1:12:13<00:00, 432.87s/it]

In [8]:
```
grid = np.arange(min_dof, max_dof)
plt.figure(figsize=(15,5))
plt.hlines([1], min_dof, max_dof-1, color='r', linestyles ='dashed')
plt.plot(grid, is_reject_kolm.mean(axis=1)[min_dof:max_dof], label='kolmogorov')
plt.plot(grid, is_reject_jb.mean(axis=1)[min_dof:max_dof], label='jarque_bera')
plt.plot(grid, is_reject_shapiro.mean(axis=1)[min_dof:max_dof], label='shapiro')
plt.legend()
plt.title("Реальная мощность критериев при альтернативном распределении Стьюдента")
plt.xlabel("количество степеней свободы распределения Стьюдента")
plt.ylabel("Реальная мощность");
```



Сделайте вывод относительно мощности критерия при разных распределениях из альтернативы.

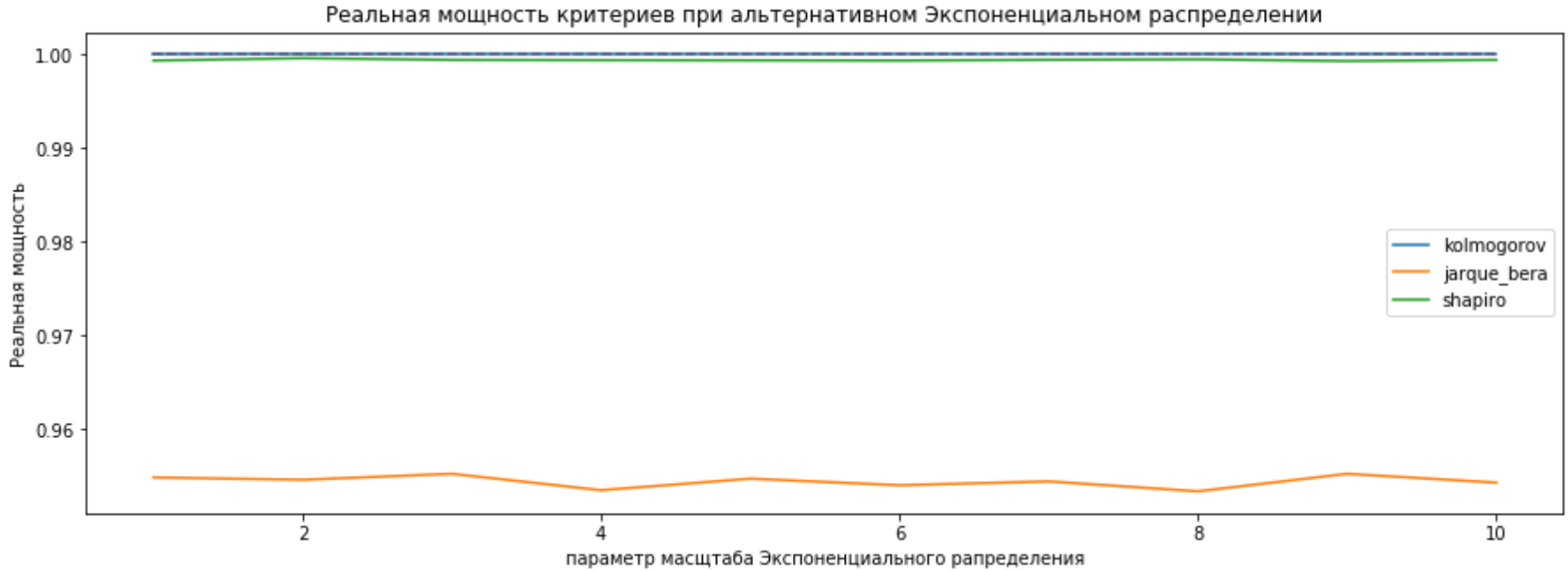
Мощность критерия при альтернативном распрделении Стьюдента понижается при увеличении числа степеней свободы. Так происходит потому что при больших значениях степеней свободы нормальное распределение и распределение Стьюдента становятся похожими, следовательно, критериям сложнее их различать. Можно заметить, что критерии Жарка-Бера и Шапиро показывают примерно одинаковые результаты, но критерий Колмогорова имеет сильно меньшую мощность.

4. Аналогичным образом посчитайте мощности критериев, если альтернативная гипотеза заключается в том, что выборка имеет экспоненциальное распределение.

In [9]:
```
sample_size = 50
sample_count = 100000
min_scale = 1
max_scale = 11

is_reject_kolm = np.zeros((max_scale, sample_count))
is_reject_jb = np.zeros((max_scale, sample_count))
is_reject_shapiro = np.zeros((max_scale, sample_count))

for i in tqdm_notebook(range(min_scale, max_scale)):
    for j in range(sample_count):
        sample = sps.expon(scale=i).rvs(size=sample_size)
        is_reject_kolm[i, j] = sps.kstest(sample, 'norm')[1] < 0.05
        is_reject_jb[i, j] = sps.jarque_bera(sample)[1] < 0.05
        is_reject_shapiro[i, j] = sps.shapiro(sample)[1] < 0.05
```

100% 10/10 [1:18:09<00:00, 467.85s/it]

In [10]:
```
grid = np.arange(min_dof, max_dof)
plt.figure(figsize=(15,5))
plt.hlines([1], min_dof, max_dof-1, color='r', linestyles ='dashed')
plt.plot(grid, is_reject_kolm.mean(axis=1)[min_dof:max_dof], label='kolmogorov')
plt.plot(grid, is_reject_jb.mean(axis=1)[min_dof:max_dof], label='jarque_bera')
plt.plot(grid, is_reject_shapiro.mean(axis=1)[min_dof:max_dof], label='shapiro')
plt.legend()
plt.title("Реальная мощность критериев при альтернативном Экспоненциальном распределении")
plt.xlabel("параметр масщтаба Экспоненциального рапределения")
plt.ylabel("Реальная мощность");
```



Сделайте вывод:

Мощность критерия при альтернативной гипотезе $Exp(\lambda)$ не зависит от λ. Критерии Колмогорова и Шапиро работают лучше, чем критерий Жарка-Бера.

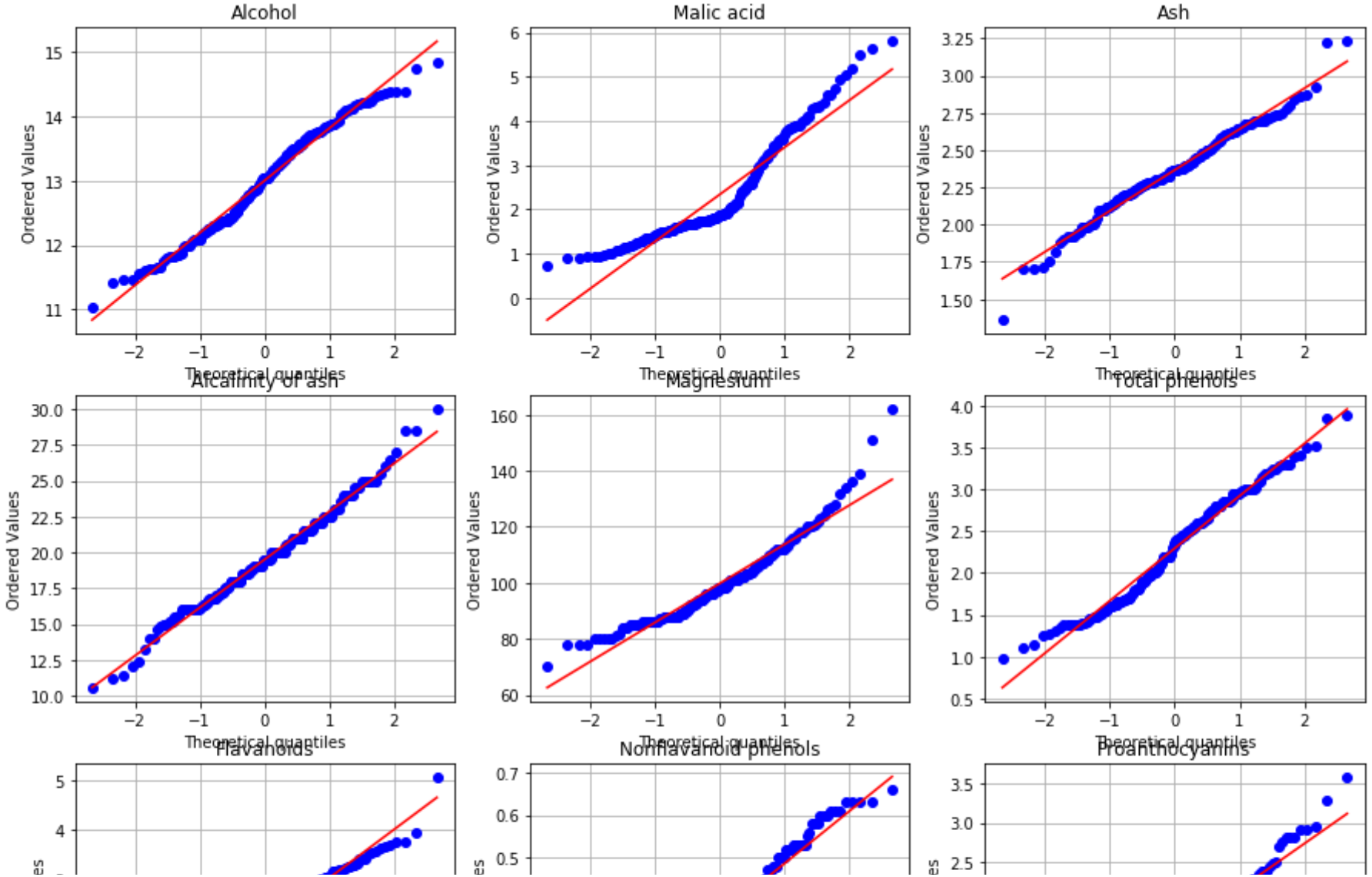
Задача 3.

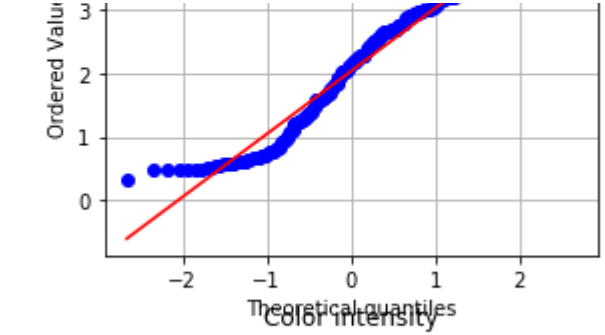
Скачайте данные `wine dataset` (http://archive.ics.uci.edu/ml/datasets/wine), взяв все колонки, кроме `Class` .

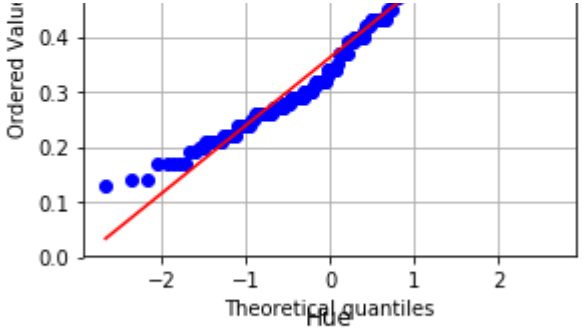
In [17]:
```
names= ['Class', 'Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Total phenols', 'Flavanoids',
'Nonflavanoid phenols', 'Proanthocyanins', 'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline']
wine = pd.read_csv('wine.data', names=names)[names[1:]]
names = names[1:]
```

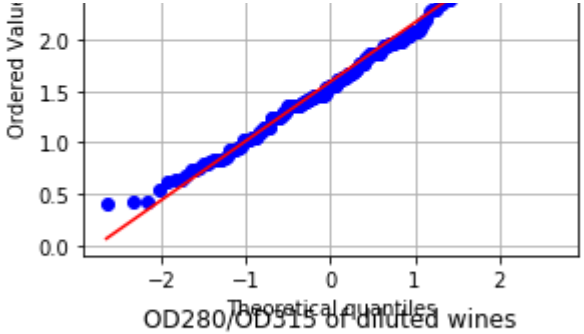
Для каждого параметра нарисуте Q-Q plot. Для наглядности и экономии места рисуйте графики в несколько столбцов, используя `plt.subplot` .

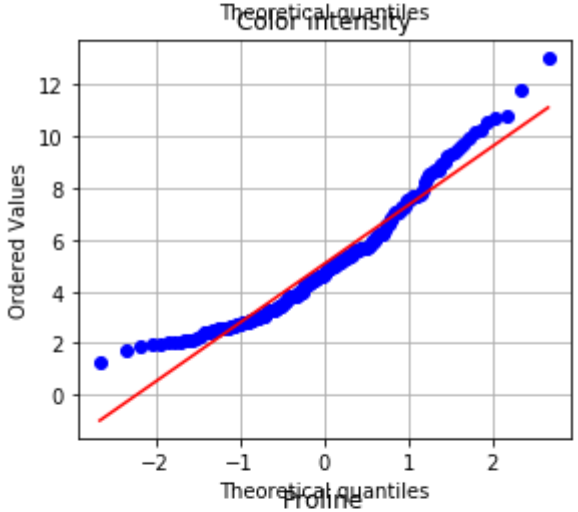
In [18]:
```
plt.figure(figsize=(15, 30))
for i, name in enumerate(names):
    ax = plt.subplot(7, 3, i+1)
    sps.probplot(wine[name], plot=ax)
    plt.title(name)
    plt.grid(True)
plt.show()
```

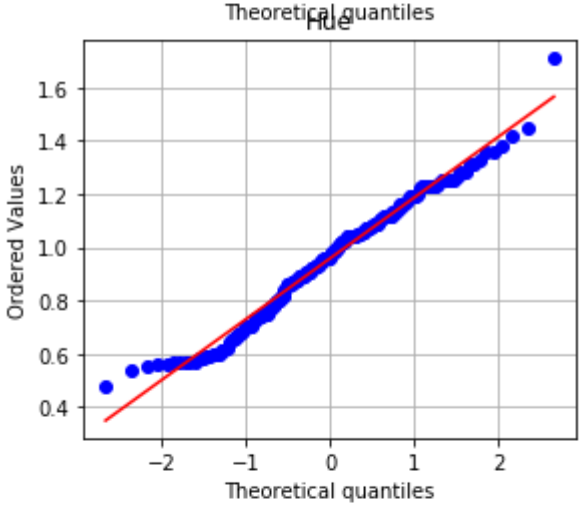


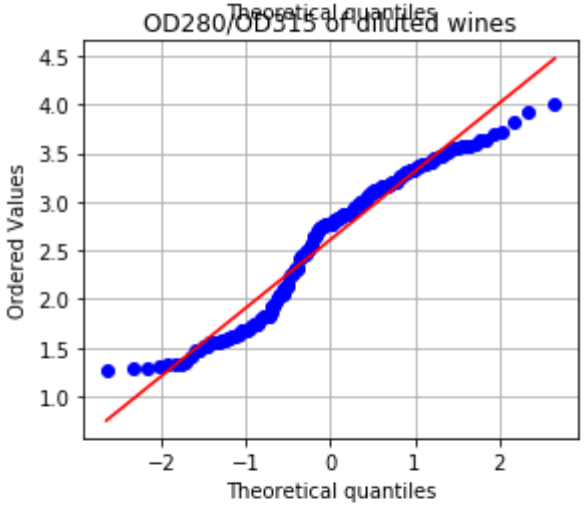


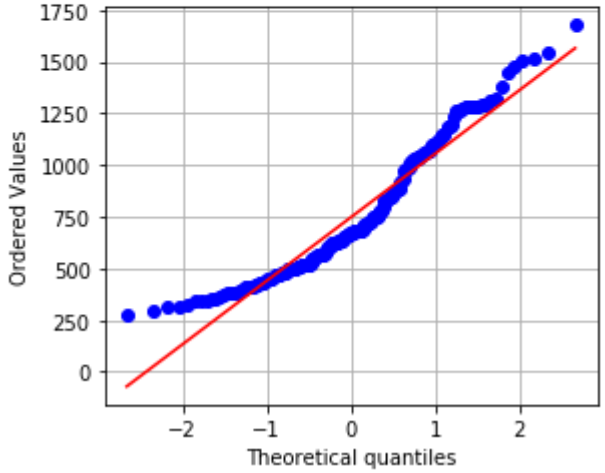












Для каких параметров можно говорить об их нормальности и почему?

Ответ: Для нормальных величин точки лежат ровно на прямой, мало отклоняясь. Такое можно заметить для Alcanity of ash и Proanthocyanins.

Проверьте нормальность каждого параметра статистическими критериями. Прежде чем выполнить следующую часть задачи, ответьте на вопросы. Помочь в этом может теоретическое домашнее задание 11, разбор которого был выложен в чате.

Для каждого параметра используйте несколько критериев проверки нормальности. Какие критерии вы будете использовать?

Ответ: Буду использовать критерии Жарка-Бера и Шапиро-Уилка, потому что они заточены специально на проверку нормальности, и критертий Колмогорова, потому что он имеет большую мощность при экспоненциальном распределении.

Результаты критериев нужно обработать с помощью *одной* процедуры множественной проверки гипотез для всех параметров и всех критериев сразу. Почему так нужно делать?

Ответ: Потому что если не пользоваться множественной проверкой гипотез, то вероятность того, что хотя бы одна верная гипотеза будет отвергнута резко повышается. Например, если вероятность ошибки первого рода для одной проверки это α, то для n проверок вероятность отвергнуть одну верную гипотезу будет $1 - (1 - \alpha)^n$

Какой метод для контроля FWER стоит применить и почему?

Ответ: Стоит применить метод Холма. Во-первых, потому что не стоит применять методы, контролирующие FDR. Во-вторых FDR контролируют Бонферрони, Холм и Шидак-Холм, однако есть теорема гласящая, что Холм мощнее любого другого критерия, если про выборку ничего дополнительно не известно.

Следовательно, следует применять метод Холма.

Как понять из результата процедуры множественной проверки гипотез, нормальность каких параметров следует отклонить?

Ответ: Если нормальность параметра была отвергнута хотя бы 1 раз, то его нормальность можоно отклонить.

Если нормальность не отклоняется, что можно сказать про выборку (ответ в презентации с лекции)?

Ответ: Если нормальность не отклоняется, значит, данных согласуются с гипотезой, о том, что они нормальные

Реализуйте данную схему. Предварительно соберите p-value всех критериев в таблицу.

Функция `multipletests` принимает только одномерные массивы, поэтому для полученной `numpy`-таблицы нужно воспользоваться методом `ravel`. Результат нужно собрать обратно в таблицу с помощью метода `reshape`, которому нужно передать размерности таблицы. Для наглядности сделайте таблицу с помощью `pandas.DataFrame`. В качестве названий строк используйте названия переменных, а названий столбцов --- используемые критерии.

In [19]:
```
table = pd.DataFrame()
for name in names:
    row = pd.Series(name=name)
    sample = wine[name]
    mean = sample.mean()
    std = sample.std()
    row['kolm'] = sps.kstest(sample, sps.norm(loc=mean, scale=std).cdf)[1]
    row['jb'] = sps.jarque_bera(sample)[1]
    row['shapiro'] = sps.shapiro(sample)[1]
    table = table.append(row)
shape = table.shape
table = multipletests(np.ravel(table), method='h')
table = pd.DataFrame(table[0].reshape(shape), index=names,
                     columns=['Метод Колмогорова отвергает нормальность',
                              'Метод Жарка-Бера отвергает номальность',
                              'Метод Шапиро отвергает нормальность'])
table['нормальность отвергается'] = table.sum(axis=1) > 0 # эта строчка делает поэлментное "или" для трех столбцов
```

In [21]:
```
table
```

Out[21]:

	Метод Колмогорова отвергает нормальность	Метод Жарка-Бера отвергает номальность	Метод Шапиро отвергает нормальность	нормальность отвергается
Alcohol	False	False	False	False
Malic acid	True	True	True	True
Ash	False	False	False	False
Alcalinity of ash	False	False	False	False
Magnesium	True	False	True	True
Total phenols	False	False	False	False
Flavanoids	False	False	True	True
Nonflavanoid phenols	False	False	True	True
Proanthocyanins	False	False	False	False
Color intensity	True	False	True	True
Hue	False	False	False	False
OD280/OD315 of diluted wines	False	False	True	True
Proline	True	False	True	True

Для каких параметров нормальность не отвергается?

Ответ: Нормальность не отвергается для Alcohol, Ash, Alcalinity of ash, Total phenols, Proanthocyanins, Hue.

Линейная регрессия

Задача 4.

По шаблону напишите класс, реализующий гауссовскую линейную регрессию. Интерфейс этого класса в некоторой степени соответствует классу `LinearRegression` (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression) из библиотеки `sklearn`.

In [268]:

```
class LinearRegression:
    def __init__(self):
        super()

    def fit(self, X, Y, alpha=0.05):
        ''' Обучение модели. Предполагается модель Y = X * theta + epsilon,
            где X --- регрессор (матрица размера n x d),
            Y --- отклик (матрица размера n x 1),
            а epsilon имеет нормальное распределение
            с параметрами (0, sigma^2 * I_n).
            alpha --- уровень значимости критерия
            (1-alpha --- ур. доверия для доверительного интервала).
        '''
        self.n, self.d = X.shape

        self.theta = (X.T*X).I*X.T*Y
        self.sigma_sq = ((Y-X*self.theta).T*(Y-X*self.theta)/(self.n-self.d))[0,0]
        self.sigma = self.sigma_sq**(1/2)
        help_vector = np.matrix(np.sqrt(np.diag((X.T*X).I))).T
        delta = sps.t(df=self.n-self.d).ppf(1-alpha/2)*self.sigma*help_vector
        # доверительные интервалы для коэффициентов (матрица размера d x 2)
        self.conf_int = np.concatenate([self.theta-delta, self.theta+delta], axis=1)
        # статистики критериев о незначимости коэффициентов (вектор размера d)
        self.t_stats = self.theta/self.sigma/help_vector
        # p-value критериев о незначимостикоэффициентов (вектор размера d)
        self.pvalue = 2*sps.t(df=self.n-self.d).sf(np.abs(self.t_stats))

        return self

    def summary(self):
        print('Linear regression on %d features and %d examples' \
              % (self.d, self.n))
        print('Sigma: %.6f' % self.sigma_sq)

        # Далее напечатайте таблицу,
        # в которой для каждого коэффициента (по строкам)
        # будут следующие столбцы:
        # оценка, границы дов. интервала (два столбца),
        # статистика критерия, p-value
        display(pd.DataFrame(np.column_stack((self.theta, self.conf_int,
                                              self.t_stats, self.pvalue)),
```

```
                                columns=['theta estimator', 'left confidence interval',
                                         'right confidence interval','statistic', 'p-value']))

    def predict(self, X):
        ''' Возвращает предсказание отклика на новых объектах X. '''

        Y_pred = X*self.theta
        return Y_pred
```

Загрузите данные о потреблении мороженного в зависимости от температуры воздуха и цены (файл `ice_cream.txt`). Примените реализованный выше класс линейной регрессии к этим данным предполагая, что модель имеет вид $ic = \theta_1 + \theta_2\, t$, где t --- температура воздуха (столбец `temp`), ic --- постребление мороженного в литрах на человека (столбец `IC`). Значения температуры предварительно переведите из Фаренгейта в Цельсий [(Фаренгейт — 32) / 1,8 = Цельсий].

К обученной модели примените фунцию `summary` и постройте график регрессии, то есть график прямой $ic = \widehat{\theta}_1 + \widehat{\theta}_2\, t$, где $\widehat{\theta}_1, \widehat{\theta}_2$ --- МНК-оценки коэффициентов. На график нанесите точки выборки.

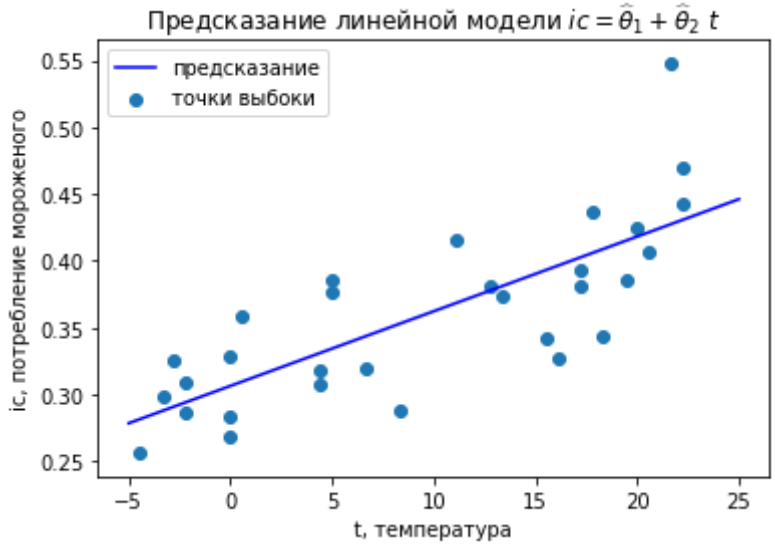
In [63]:
```
sample = pd.read_csv('ice_cream.txt', sep='\t', index_col='date')
sample['temp'] = (sample.temp-32)/1.8
```

In [64]:
```
lr = LinearRegression()
X = np.matrix([np.ones(sample.shape[0]), sample['temp']]).T
Y = np.matrix(sample.IC).T
lr.fit(X, Y)
lr.summary()
```

```
Linear regression on 2 features and 30 examples
Sigma: 0.001786
```

	theta estimator	left confidence interval	right confidence interval	statistic	p-value
0	0.306298	0.283276	0.329319	27.253231	1.060446e-21
1	0.005593	0.003831	0.007355	6.502305	4.789215e-07

In [52]:
```
grid = np.linspace(-5, 25, 200)
X = np.matrix([np.ones(200), grid]).T
plt.plot(grid, lr.predict(X), label='предсказание', color='b')
plt.scatter(sample.temp, sample.IC, label='точки выбоки')
plt.legend()
plt.xlabel('t, температура')
plt.ylabel("ic, потребление мороженого")
plt.title("Предсказание линейной модели $ic = \\widehat{\\theta}_1 + \\widehat{\\theta}_2\ t$");
```



Теперь учтите влияние года (столбец `Year`) для двух случаев:

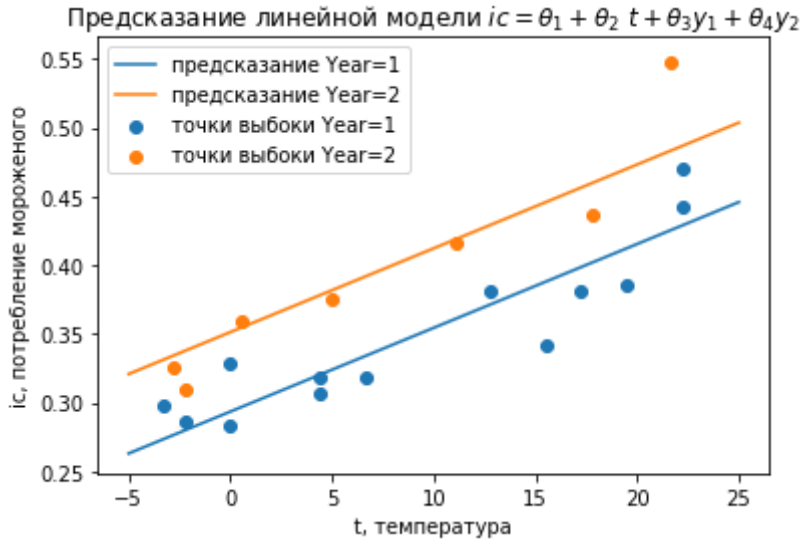
- модель $ic = \theta_1 + \theta_2\ t + \theta_3 y_1 + \theta_4 y_2$, где $y_1 = I\{1 \text{ год}\}, y_2 = I\{2 \text{ год}\}$. Поясните, почему нельзя рассматривать одну переменную y --- номер года.
- для каждого года рассматривается своя линейная зависимость $ic = \theta_1 + \theta_2\ t$.

В каждом случае нарисуйте графики. Отличаются ли полученные результаты? От чего это зависит? Как зависит потребление мороженного от года?

Первый случай:

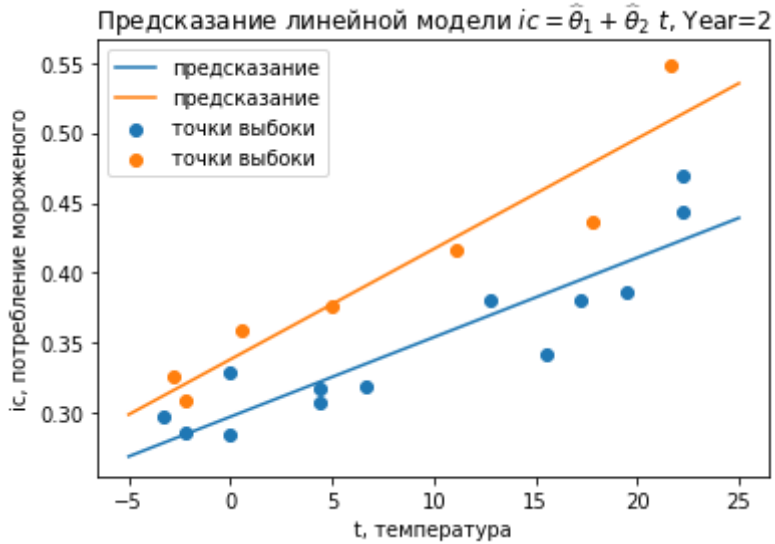
In [53]:
```
X = np.matrix([np.ones(sample.shape[0]), sample.temp, sample.Year==1, sample.Year==2]).T
Y = np.matrix(sample.IC).T
lr.fit(X, Y);
```

In [54]:
```
grid = np.linspace(-5, 25, 200)
X = np.matrix([np.ones(200), grid, np.ones(200), np.zeros(200)]).T
plt.plot(grid, lr.predict(X), label='предсказание Year=1')
X = np.matrix([np.ones(200), grid, np.zeros(200), np.ones(200)]).T
plt.plot(grid, lr.predict(X), label='предсказание Year=2')
for year in [1,2]:
    Year = sample[sample.Year==year]
    plt.scatter(Year.temp, Year.IC, label=('точки выбоки Year=%d'%year))
plt.legend()
plt.xlabel('t, температура')
plt.ylabel("ic, потребление мороженого")
plt.title("Предсказание линейной модели $ic = \\theta_1 + \\theta_2\ t + \\theta_3 y_1 + \\theta_4 y_2$");
```



Второй случай:

In [55]:
```
lr = LinearRegression()
#plt.figure(figsize=(13, 4))
for i in [1, 2]:
    year_data = sample[sample.Year==i]

    lr = LinearRegression()
    X = np.matrix([np.ones(year_data.shape[0]), year_data['temp']]).T
    Y = np.matrix(year_data.IC).T
    lr.fit(X, Y)

    #plt.subplot(1, 2, i)
    grid = np.linspace(-5, 25, 200)
    X = np.matrix([np.ones(200), grid]).T
    plt.plot(grid, lr.predict(X), label='предсказание')
    plt.scatter(year_data.temp, year_data.IC, label='точки выбоки')
    plt.legend()
    plt.xlabel('t, температура')
    plt.ylabel("ic, потребление мороженого")
    plt.title("Предсказание линейной модели $ic = \\widehat{\\theta}_1 + \\widehat{\\theta}_2\ t$, Year=%d"%i);
```



В каждом случае нарисуйте графики. Отличаются ли полученные результаты? От чего это зависит? Как зависит потребление мороженного от года?

Вывод: Полученные результаты различаются для первого и второго случаев, тем, что в первом случае две прямые получаются параллельными. Также можно заметить, что во втором случае прямые лучше подходят под точки выборки. Из графиков видно, что во втором году потребление мороженого было больше.

Наконец, обучите модель на предсказание потребления мороженного в зависимости от всех переменных. Не забудьте, что для года нужно ввести две переменных. Для полученной модели выведите `summary` .

Линейная модель $ic = \theta_0 + \theta_1 price + \theta_2 income + \theta_3 t + \theta_4 lagtemp + \theta_5 I\{Year = 1\} + \theta_6 I\{Year = 2\}$

In [68]:
```
lr = LinearRegression()
X = np.matrix([np.ones(sample.shape[0]), sample.price, sample.income, sample.temp, sample['Lag-temp'],
               sample.Year==1, sample.Year==2]).T
Y = np.matrix(sample.IC).T
lr.fit(X, Y)
lr.summary();
```

```
Linear regression on 7 features and 30 examples
Sigma: 0.001024
```

	theta estimator	left confidence interval	right confidence interval	statistic	p-value
0	0.717753	0.107657	1.327849	2.433690	0.023126
1	-0.659296	-2.467091	1.148500	-0.754431	0.458246
2	-0.003231	-0.007774	0.001311	-1.471643	0.154669
3	0.005654	0.003801	0.007507	6.311846	0.000002
4	-0.000024	-0.000886	0.000838	-0.057599	0.954566
5	0.038141	-0.000852	0.077134	2.023462	0.054797
6	0.117733	0.045224	0.190242	3.358875	0.002716

Так как для параметров $\theta_0, \theta_3, \theta_5, \theta_6$ p-value<0.05, то можно гипотеза о том, что они незначительны отвергается, для остальных не отвергается. То есть признаки $temp, Year$ скорее всего значительны.

Но это еще не все. Постройте теперь линейную регрессию для модели $ic = \theta_1 + \theta_2\, t + \theta_3\, t^2 + \theta_4\, t^3$. Выведите для нее summary и

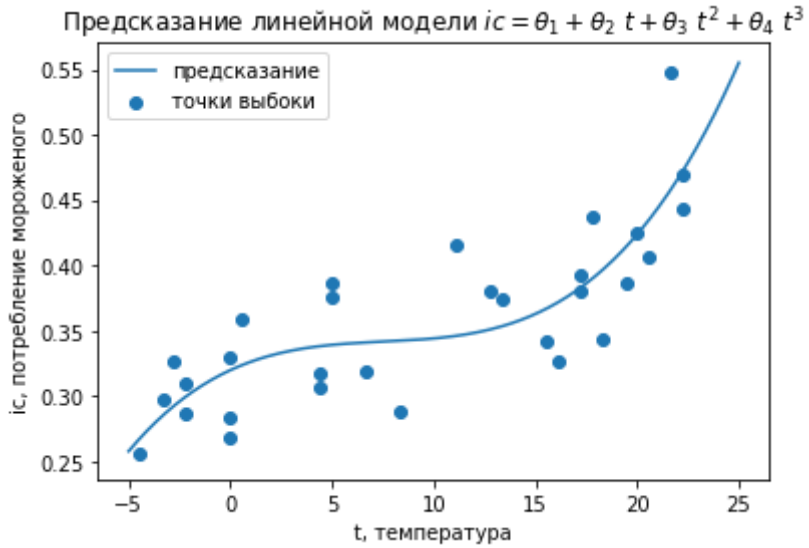
постройте график предсказания, то есть график кривой $ic = \widehat{\theta}_1 + \widehat{\theta}_2\, t + \widehat{\theta}_3\, t^2 + \widehat{\theta}_4\, t^3$. Хорошие ли получаются результаты?

In [69]:
```
lr = LinearRegression()
X = np.matrix([np.ones(sample.shape[0]), sample.temp, sample.temp**2, sample.temp**3]).T
Y = np.matrix(sample.IC).T
lr.fit(X, Y)
lr.summary();
```

```
Linear regression on 4 features and 30 examples
Sigma: 0.001529
```

	theta estimator	left confidence interval	right confidence interval	statistic	p-value
0	0.319902	0.295294	0.344510	26.721959	1.957227e-20
1	0.007200	0.000388	0.014013	2.172538	3.911764e-02
2	-0.000855	-0.001861	0.000152	-1.745679	9.267541e-02
3	0.000038	0.000002	0.000073	2.180118	3.849208e-02

In [73]:
```
grid = np.linspace(-5, 25, 200)
X = np.matrix([np.ones(200), grid, grid**2, grid**3]).T
plt.plot(grid, lr.predict(X), label='предсказание')
plt.scatter(sample.temp, sample.IC, label=('точки выбоки'))
plt.legend()
plt.xlabel('t, температура')
plt.ylabel("ic, потребление мороженого")
plt.title("Предсказание линейной модели $ic = \\theta_1 + \\theta_2\ t + \\theta_3\ t^2 + \\theta_4\ t^3$");
```



Вывод: Результаты действительно получаются лучше, чем в линейной модели $ic = \theta_1 + \theta_2\ t$

Чтобы понять, почему так происходит, выведите значения матрицы $(X^T X)^{-1}$ и посчитайте для нее индекс обусловленности $\sqrt{\lambda_{max} / \lambda_{min}}$, где $\lambda_{max}, \lambda_{min}$ --- максимальный и минимальный собственные значения матрицы $X^T X$. Собственные значения можно посчитать функцией `scipy.linalg.eigvals` (https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.eigvals.html).

Прокомментируйте полученные результаты. Помочь в этом может следующая статья (https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE_%D0%BE%D0%B1%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0

In [83]:
```
matrix = (X.T*X).I
eigvals_ = eigvals(matrix)
eigvals_
```

Out[83]:
```
array([1.47067249e-02+0.j, 1.01357067e-03+0.j, 2.66101118e-06+0.j,
       1.68932875e-10+0.j])
```

In [91]:
```
eigvals_ = eigvals(matrix)
cond = np.sqrt(max(eigvals_)/min(eigvals_))
print("Condition number: %d"%cond)
```

```
C:\Users\1\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: ComplexWarning: Casting complex values to real discards the imaginary part
  This is separate from the ipykernel package so we can avoid doing imports until

Condition number: 9330
```

Можно считать оператор плохо обусловленным. Это значит, что из-за небольших погрешностей в наблюдениях будут большие погрешности предсказаний. Значит, модели не стоит сильно доверять.

Задача 5.

В данной задаче нужно реализовать функцию отбора признаков для линейной регрессии. Более подробно, пусть у объектов есть признаки $x_1, \ldots, x_k$. Нужно определить, какое подмножество признаков $x_{j_1}, \ldots, x_{j_s}$ нужно использовать, чтобы качество полученной модели $y = \theta_0 + \theta_{j_1} x_{j_1} + \ldots + \theta_{j_s} x_{j_s}$ было максимальным.

Один из методов, решающих эту задачу описан ниже.

Сначала имеющаяся выборка случайно разделяется на обучающую выборку и тестовую (train и test). Для некоторого подмножества признаков на обучающей выборке обучается модель, после чего вычисляется её качество на тестовой выборке. Операция повторяется для всех подмножеств признаков. Лучшей считается модель с наилучшим качеством на тестовой выборке.

Иначе говоря, сначала выборка X разделяется по объектам на $X_{train} \sqcup X_{test} = X$. Далее, в цикле по всем подмножествам индексов признаков $(j_1, \ldots j_s)$ на обучающей выборке X_{train} обучается модель $y = \theta_{j_1} x_{j_1} + \ldots + \theta_{j_s} x_{j_s}$, после чего считается её качество на X_{test}.

В данной задаче под метрикой качества понимается средний квадрат ошибки (mean squared error)

$$MSE = \frac{1}{n} \sum_{x \in X_{test}} \left(\widehat{y}(x) - Y(x) \right)^2,$$

где x - объект, $Y(x)$ - значение целевой переменной (отклика) на объекте x, а $\widehat{y}(x)$ - оценка отклика на объекте x.

Заметим, что полный перебор подмножеств признаков требует большого времени. Кратко проблема отбора признаков состоит в том, что могут быть два признака, добавление каждого из которых ухудшает (не улучшает) качество, но добавление двух признаков сразу увеличивает качество.

Для выполнения задания воспользуйтесь следующими функциями:

- `sklearn.linear_model.LinearRegression` (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression) --- реализация линейной регрессии. В данной реализации свободный параметр θ_1 по умолчанию автоматически включается в модель. Отключить это можно с помощью `fit_intercept=False` , но это не нужно. *В данной задаче требуется, чтобы вы воспользовались готовой реализацией линейной регрессии, а не своей. Ведь на практике важно уметь применять готовые реализации, а не писать их самостоятельно.*
- `sklearn.model_selection.train_test_split` (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) --- функция разбиения данных на train и test. Установите параметр `test_size=0.3` и `random_state=17` .
- `sklearn.metrics.mean_squared_error` (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html) --- реализация MSE.

Для перебора реализуйте следующий класс. Данный класс частично реализует интерфейс, похожий на интерфейс `sklearn.model_selection.GridSearchCV` (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

In [3]:
```
class BestFeaturesSelection:
    def __init__(self, estimator, scoring, parameters=dict(),
                 test_size=0.3, random_state=17, minimize=True):
        """
        Отбор наилучших признаков
        estimator: конструктор класса, например, LinearRegression
        paramters: параметры, передаваемые конструктору estimator,
                    например dict(fit_intercept=False)
        scoring: функция риска, например, mean_squared_error
        minimize: минимизировать ли функционал качества
                    (иначе - максимизировать)
        """

        self.estimator = estimator
        self.parameters = parameters
        self.scoring = scoring
        self.test_size = test_size
        self.random_state = random_state
        self.minimize=minimize

    def fit(self, X, y):
        """
        Подбор лучшего подмножества признаков
        и обучение модели на нём
        """
        # разделение выборки на test и train. Не перепутайте порядок !
        X_train, X_test, y_train, y_test =  train_test_split(X, y,
                                                             test_size=self.test_size, random_state=self.random_state)
        self.results_ = []  # список пар (вектор использованных признаков,
                            # значение функции потерь)
        features_count = X.shape[1]

        for bitmask in range(1, 2 ** features_count):
            subset = [i == "1" for i in np.binary_repr(bitmask,
                                                       width=features_count)]
            # binary_repr возвращает строку длины width с двоичным
            # представлением числа и ведущими нулями
            curr_estimator = self.estimator(**self.parameters)
            curr_estimator.fit(X_train[:, subset], y_train)
            prediction = curr_estimator.predict(X_test[:, subset])
            score = self.scoring(y_test, prediction)  # вычисление качества модели
```

```
            self.results_.append((subset, score))

        self.results_.sort(key = lambda pair: pair[1],
                           reverse=not self.minimize)
        # сортируем по второму элементу в нужном порядке

        self._best_subset = self.results_[0][0]
        self._best_estimator = self.estimator(**self.parameters)
        self._best_estimator.fit(X_train[:, self._best_subset], y_train)

        return self._best_estimator

    def predict(self, X):
        """
        Предсказание модели,
        обученной на наилучшем подмножестве признаков.
        """

        return self._best_estimator.predict(X[:, self._best_subset]);
```

Примените реализованный отбор признаков к датасетам

- Yacht Hydrodynamics (http://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics) --- для парусных яхт нужно оценить остаточное сопротивление на единицу массы смещения (последний столбец) в зависимости от различных характеристик яхты.
- Boston Housing Prices (http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html#sklearn.datasets.load_boston) --- цены на дома в Бостоне в зависимости от ряда особенностей.

Посмотрите на графики зависимости целевой переменной от каждого признака. Какие бы признаки вы стали использовать? Совпадает ли ваш выбор с результатом алгоритма, описанного выше?

Yacht Hydrodynamics

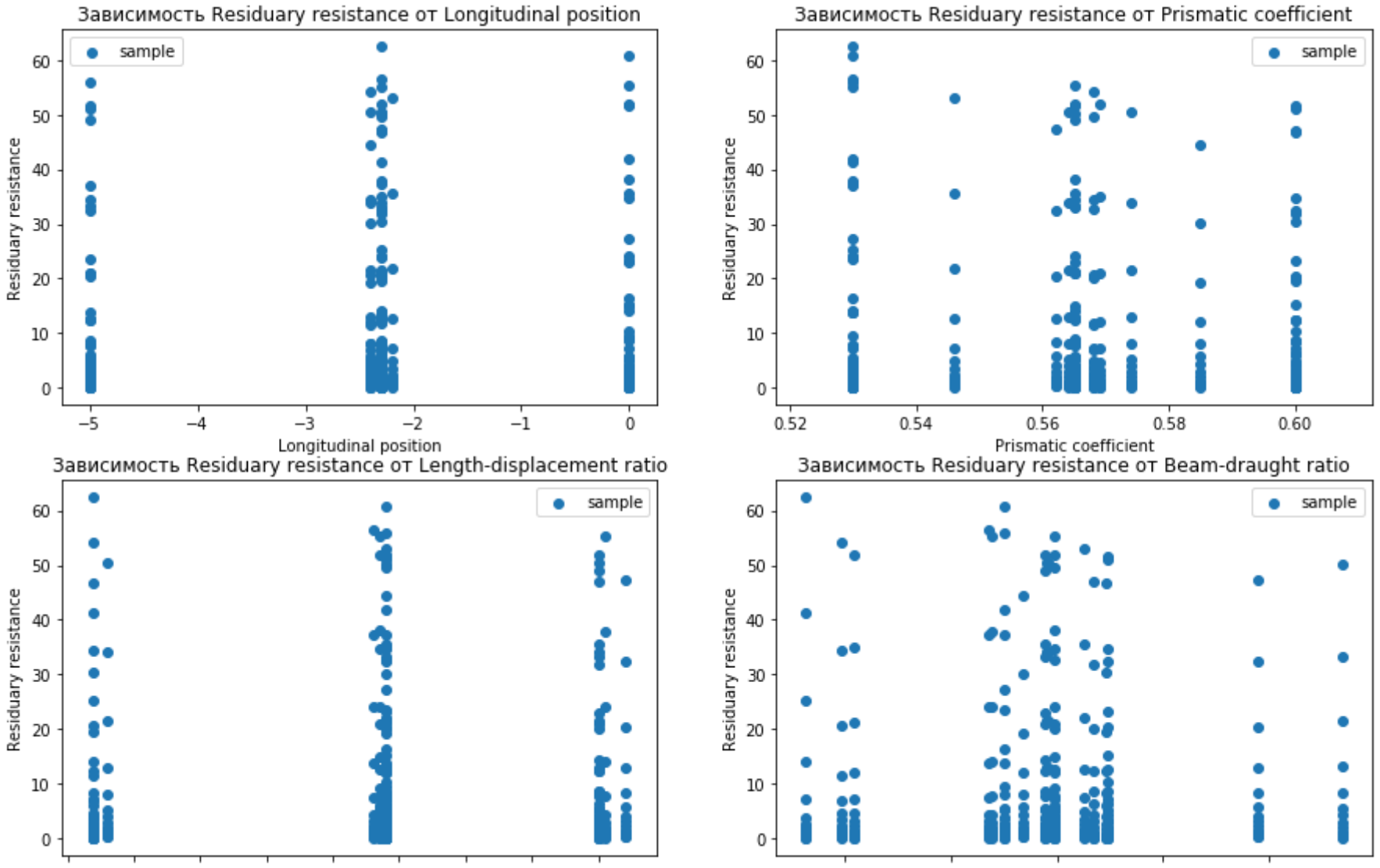
In [4]:
```
names = ["Longitudinal position",
         "Prismatic coefficient",
         "Length-displacement ratio",
         "Beam-draught ratio",
         "Length-beam ratio",
         "Froude number",
         "Residuary resistance"]

data = pd.read_csv('yacht_hydrodynamics.data', names=names, delim_whitespace=True)
data.dropna(how='any', inplace=True)
```

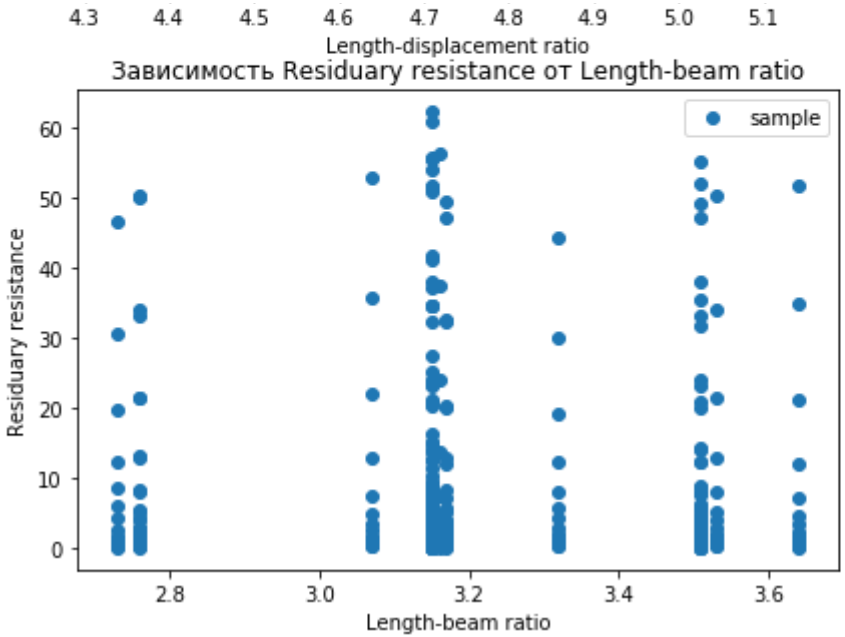
In [5]:
```
bfs = BestFeaturesSelection(LinearRegression, mean_squared_error)
X = np.matrix(data[names[:-1]])
Y = np.matrix(data[names[-1]]).T
best_estimator = bfs.fit(X, Y)
```

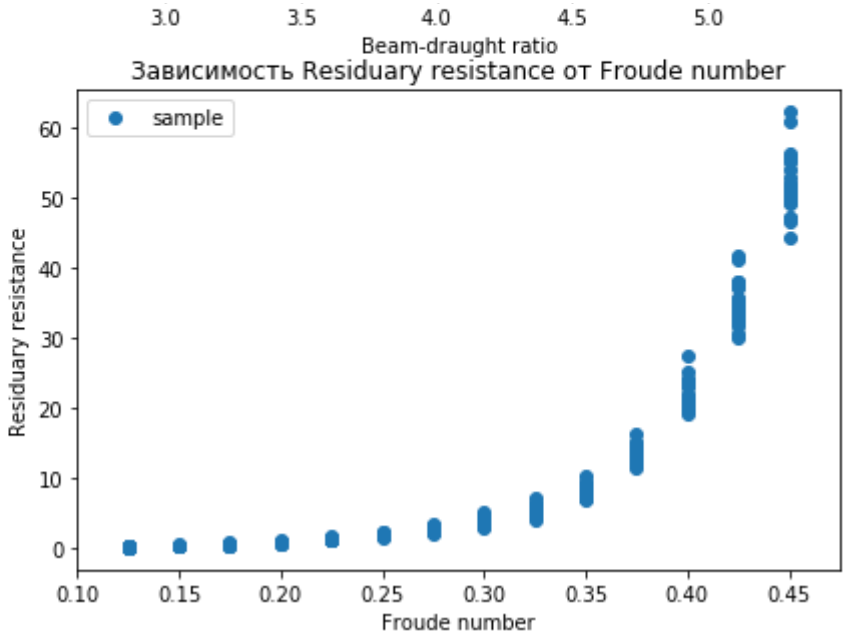
In [6]:
```
best_subset = []
for i, b in enumerate(bfs._best_subset):
    if b:
        best_subset.append(names[i])
print("Лучший набор переменных это:", best_subset)
```

```
Лучший набор переменных это: ['Prismatic coefficient', 'Froude number']
```

In [7]:
```
plt.figure(figsize=(15, 15))
for i, name in enumerate(names[:-1]):
    plt.subplot(3, 2, i+1)
    plt.scatter(data[name], data[names[-1]], label='sample')
    plt.title("Зависимость "+names[-1]+" от "+name)
    plt.xlabel(name)
    plt.ylabel(names[-1])
    plt.legend()
```







Вывод: Я бы взяла только функцию от 'Froude number', потому что в распределении других точек сложно найти линейную структуру. Однако модель дает лучшие результаты, если включить еще 'Prismatic coefficient'. Значит, сложно судить о значимости коэфициента только по графику.

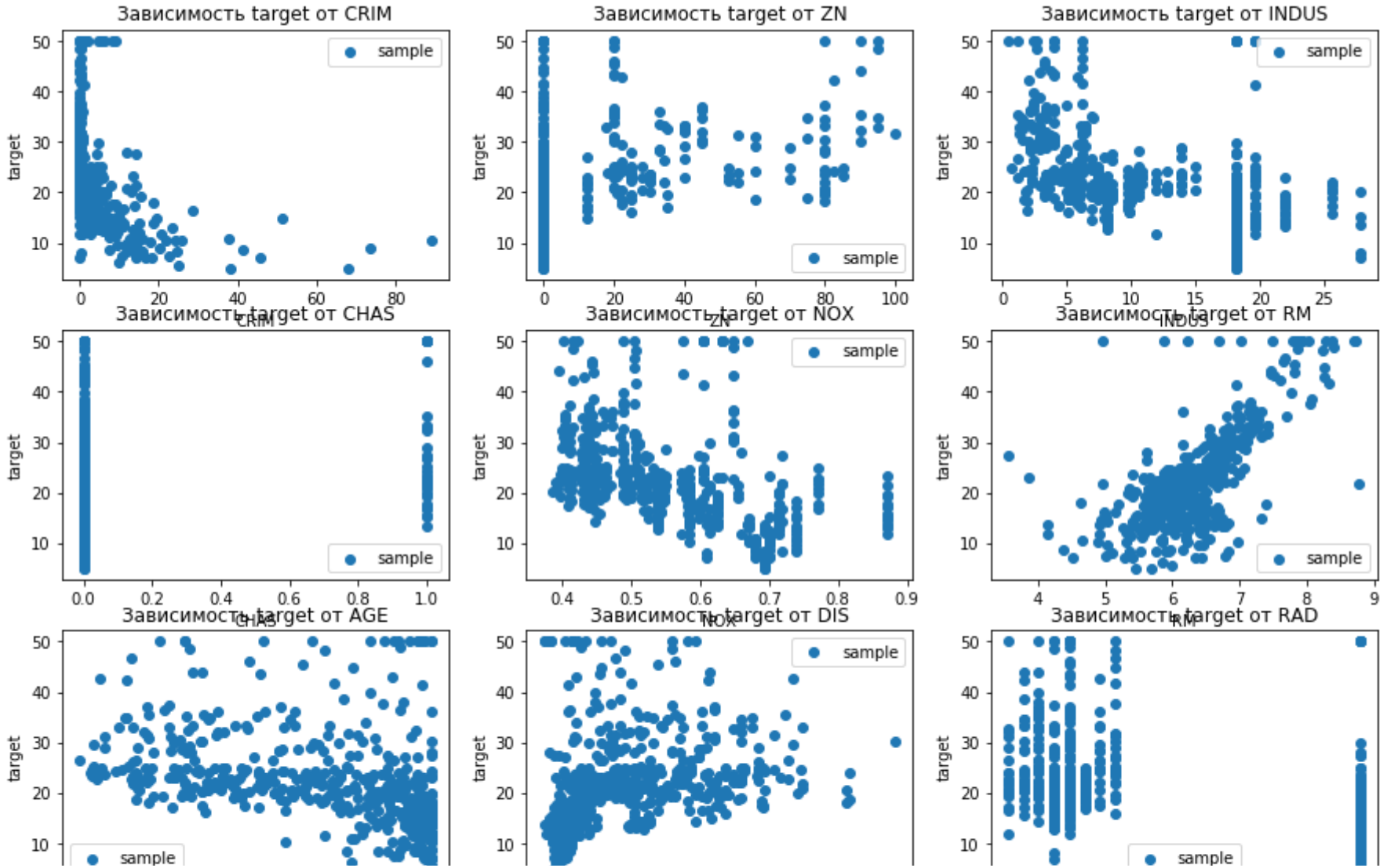
Boston Housing Prices

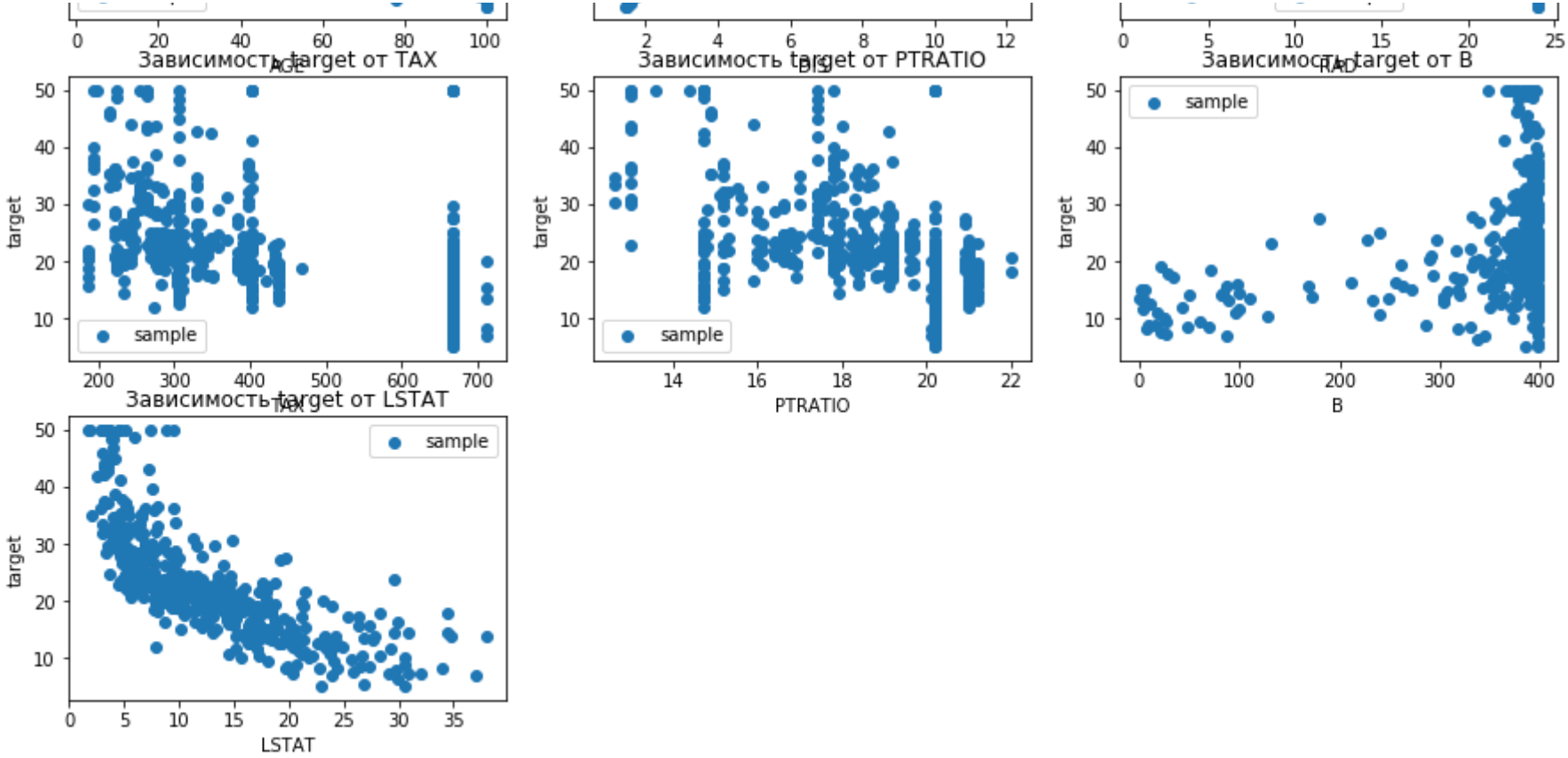
In [8]:
```
data = load_boston()
X = np.matrix(data['data'])
Y = np.matrix(data['target']).T
bfs = BestFeaturesSelection(LinearRegression, mean_squared_error)
best_estimator = bfs.fit(X, Y)
```

In [9]:
```
names = data['feature_names']
best_subset = []
for i, b in enumerate(bfs._best_subset):
    if b:
        best_subset.append(names[i])
print("Лучший набор переменных это:", best_subset)
```

```
Лучший набор переменных это: ['CRIM', 'ZN', 'NOX', 'RM', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']
```

In [10]:
```
plt.figure(figsize=(15, 17))
for i, name in enumerate(names):
    plt.subplot(5, 3, i+1)
    plt.scatter(data['data'][:,i], data['target'], label='sample')
    plt.title("Зависимость target от "+name)
    plt.xlabel(name)
    plt.ylabel('target')
    plt.legend()
```





Вывод: По графикам видно, что target линейно зависит от RM и LSTAT. Однако алгоритмом было выбрано большее множество параметров. И если присмотреться к этим параметрам, то можно заметить, что они немного линейны.

Задача 6.

Рассмотрим модель $Y = X\theta + \varepsilon$, где $X \in \mathbb{R}^{n \times k}$ --- регрессор, $Y \in \mathbb{R}^{n \times 1}$ --- отклик, а $\varepsilon \sim \mathcal{N}(0, \beta^{-1} I_n)$. При классической постановке задачи линейной регрессии оценка параметра θ находится минимизацией остаточной суммы квадратов ($RSS = ||Y - X\theta||^2$) и равна $\hat{\theta} = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}Y$. Часто также рассматривают регуляризацию, эквивалентную баесовской остановке задачи. Если в качестве априорного на θ взять $\mathcal{N}(0, \alpha^{-1} I_k)$, то оценкой параметра является математическое ожидание апостериорного распределения (которое в данном случае

она совпадает с модой) $\hat{\theta} = \left(X^{\mathrm{T}}X + \frac{\alpha}{\beta}I_k\right)^{-1} X^{\mathrm{T}}Y$. Такая постановка задачи соответствует случаю Ridge-регрессии. Если же θ имеет в качестве априорного распределение Лапласа, то оценкой параметра является мода апостерионого распределения. Данная постановка задачи соответствует случаю Лассо регрессии и не решается аналитически.

При выполнении задания воспользуйтесь готовыми реализациями:

- `sklearn.linear_model.LinearRegression` (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)
- `sklearn.linear_model.RidgeRegression` (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html)
- `sklearn.linear_model.LassoRegression` (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html)

В обоих случаях свободный параметр автоматически включается в модель, отключить это можно с помощью `fit_intercept=False` . В случае Ridge и Lasso регрессии параметр `alpha` отвечает за регуляризацию, по умолчанию он принимает значение 1.0. В обозначениях выше для Ridge-регрессии `alpha` $= \frac{\alpha}{\beta}$.

Для выполнения задания скачайте данные `cost_of_living_2018` (https://dasl.datadescription.com/datafile/cost-of-living-2018/?_sfm_cases=539+541), в которых используйте следующие столбцы:

- `Cost of Living Index` --- является относительным показателем цен на потребительские товары, включая продукты, рестораны, транспорт и коммунальные услуги. Cost of Living Index не включает расходы на проживание, такие как аренда или ипотека. Если город имеет индекс стоимости жизни 120, это означает, что Numbeo оценивает его на 20% дороже, чем Нью-Йорк.
- `Rent Index` --- это оценка цен на аренду квартир в городе по сравнению с Нью-Йорком. Если индекс арендной платы равен 80, Numbeo оценивает, что цена аренды в этом городе в среднем на 20% меньше, чем цена в Нью-Йорке.
- `Cost of Living Plus Rent Index` --- это оценка цен на потребительские товары, включая арендную плату, по сравнению с Нью-Йорком.
- `Restaurant Price Index` --- сравнение цен на блюда и напитки в ресторанах и барах по сравнению с Нью-Йорк.
- `Local Purchasing Power Index` --- показывает относительную покупательную способность при покупке товаров и услуг в данном городе за среднюю заработную плату в этом городе. Если внутренняя покупательная способность составляет 40, это означает, что жители этого города со средней зарплатой могут позволить себе покупать в среднем на 60% меньше товаров и услуг, чем жители Нью-Йорка со средней зарплатой по Нью-Йорку.
- `Groceries Index` --- это оценка цен на продукты в городе по сравнению с Нью-Йорком. Для расчета этого раздела Number использует веса товаров в разделе "Рынки" для каждого города.

Задача заключается в построении предсказания `Groceries Index` по известным значениям остальных параметров.

Для начала исследуйте зависимость значений коэффициентов от параметра регуляризации для Ridge и Lasso регрессии. Для этого в обоих случаях постройте график значений каждого из коэффициентов в зависимости от параметра регуляризации `alpha` из реализации `sklearn`. Для Ridge-регрессии в качестве значений параметра рекомендуется брать отрезок от 0 до примерно 0.3-0.5. Для Lasso от 0 до 600-800.

In [ ]:
```
<...>
```

Расчитайте индекс обусловленности для случая линейной регрессии. Исследуйте зависимость индекса обусловленности от параметра регуляризации для Ridge-регрессии.

In [ ]:
```
<...>
```

Перейдем к более практической части. Разбейте исходную выборку на обучающую и тестовую (в этом поможет функция `train_test_split` (http://scikit-learn.org/0.16/modules/generated/sklearn.cross_validation.train_test_split.html) из библиотеки `sklearn`), обучив модель на одной части, постройте предсказание для другого куска для случаев линейной регрессии, Ridge и Lasso. Посчитайте среднеквадратичную ошибку (`mean_squared_error` (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html) также из библиотеки `sklearn`).

In [ ]:
```
<...>
```

Исследуйте зависимость среднеквадратичной ошибки от параметра регуляризации для случаев Ridge и Lasso регрессии. Найдите оптимальное значение параметра для конкретного разбиения.

In [ ]:
```
<...>
```