```
# если блок падает с ошибкой, раскомментируйте эту строчку и перезапустите ноутбук
# !pip install numpy pandas matplotlib scipy sklearn

import enum

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as sts
from sklearn import linear_model, metrics, model_selection
from sklearn.datasets import fetch_california_housing
%matplotlib inline
```

Цены на жильё

Это пятая задача второго ДЗ блока по статистике. Она важна тем, что показывает, как мыслить в анализе данных (в частности в машинном обучении) статистически.

Мы будем работать с готовым датасетом `California Housing` - это данные о ценах на жильё в Калифорнии вместе с разными характеристиками жилья, например количество комнат или широта расположения.

```
data = fetch_california_housing(as_frame=True)
```

```
X = data['data']
y = data['target']
```

Здесь X - это данные о жилье (матрица признаков), y - стоимость. Обе эти переменные - это объекты класса `DataFrame` библиотеки pandas, которую очень любят аналитики. Она красиво отображает таблицы и позволяет удобно проводить операции над табличными данными.

```
X
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24

20640 rows × 8 columns

```
y
```

```
0        4.526
1        3.585
2        3.521
3        3.413
4        3.422
         ...
20635    0.781
20636    0.771
20637    0.923
20638    0.847
20639    0.894
Name: MedHouseVal, Length: 20640, dtype: float64
```

Так можно посмотреть на свойства распределений значений по колонкам - среднее, стандартное отклонение и выборочные квантили.

```
X.describe()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000

```
y.describe()
```

```
count    20640.000000
mean         2.068558
std          1.153956
min          0.149990
25%          1.196000
50%          1.797000
75%          2.647250
max          5.000010
Name: MedHouseVal, dtype: float64
```

1. Решаем регрессию

Мы построим статистическую модель на этих данных и проверим её качество. Чтобы быть уверенными, что качество репрезентативно, сразу отделим часть данных так, как будто проверяем модель "в боевых действиях" - есть данные, на которых модель строится, а есть те, на которых она проверяется (это классическое для анализа данных разделение на train и test)

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.33, random_state=42)
```

Применим к данным модель линейной регрессии. Пусть ϵ - вектор независимых случайных величин, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. X - матрица размера $N_1 \times p$, где N_1 - размер тренировочной выборки, p - количество признаков каждого элемента. Тогда,

$$y = X\beta + \epsilon.$$

1. Для начала проверим наше предположение. Графически проверьте зависимость переменной y от разных признаков X.

```
# чтобы извлечь из X отдельный признак, достаточно обратиться к нему по ключу:
ave_rooms = X_train['AveRooms']
# чтобы получить список всех признаков:
all_names = X_train.columns


# ваши графики и мысли
```

2. Вспомним решение задачи линейной регрессии методом наименьших квадратов:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

```
# вычислите оценку. помните, что для построения модели мы пользуемся только тренировочной частью выборки
# hat_beta = # используйте X_train, y_train
```

Теперь, нам приходят новые данные X_t (X_test). Предсказанием для них в нашей модели, очевидно, будет

$$\hat{y_t} = X_t \hat{\beta}.$$

Теперь сравним y_t (y_test) и $\hat{y_t}$. Для этого вычислим сумму квадратов наших ошибок

$$e = \sum_{i=1}^{n} (y_{ti} - \hat{y}_{ti})^2.$$

```
# вычислите предсказание модели и ошибку

# эта ошибка большая или маленькая? а в процентном соотношении?
```

В нашем предположении тренировочные и тестовые данные имеют одно и то же распределение, то есть если N_2 - размер тестовой выборки, а ϵ_t - вектор независимых случайных величин с распределением $\mathcal{N}(0, \sigma^2)$ размера N_2, то справедливо

$$y = X\beta + \epsilon,$$

$$y_t = X_t\beta + \epsilon_t.$$

3. Проверьте удачность выбора модели статистическим тестом. Воспользуйтесь критерием Вальда и проверьте гипотезу $H_0 : \beta = \hat{\beta}$ для тренировочной и тестовой выборки. В качестве подсказки - выборка $Z_i = y_i - X_i\beta$ имеет нормальное распределение с параметрами $0, \sigma^2$. Вместо дисперсии шума можно взять его состоятельную оценку.

Отвергается ли гипотеза? При каком уровне доверия можно сказать, что построенная модель не подходит для предсказаний на тестовой выборке?

2. Применяем регрессию

Воспользуемся готовой линейной регрессией и сравним результаты.

```
lr = linear_model.LinearRegression()
# примените регрессию и получите предсказания
# y_pred =
```

Для получения значения ошибки есть готовая функция.

```
# metrics.mean_squared_error(y_pred, y_test)
```

Какой метод оказался лучше - from scratch или готовый? Как вы думаете, почему?

3. Улучшаем результаты?

На самом деле, можно улучшить результаты применения нашей модели. Для этого нужно выяснить, являются ли признаки в данных скореллированными и исключить такие признаки, если они есть. Объяснение этого выходит за рамки нашего блока, так как требует рассмотрения самих данных X как случайных величин.

Для каждой пары признаков найдите коэффициенты корреляции Пирсона и Спирмэна. Сделать это для двух выборок X, Y можно так:

```
# corr_pearson = sts.pearsonr(X, Y)[0]
# corr_spearman = sts.spearmanr(X, Y)[0]
```

Теперь выберите пару признаков с наибольшим коэффициентом корреляции. Как вы думаете, является ли такое значение достаточным, чтобы отвергнуть гипотезу об их независимости? Отобразите пару признаков на графике, проверьте своё предположение.

```
# plt.scatter(..., ...)
```

Теперь удалите из данных признак "Label" из найденной пары, который вам кажется менее влияющим на предсказание (напомню, мы пытаемся предсказать стоимость жилья). Сделать это можно так:

```
# X_train.drop(labels='Label', axis=1, inplace=True)
# X_test.drop(labels='Label', axis=1, inplace=True)
```

```
# metrics.mean_squared_error(y_pred, y_test)
```

Снова воспользуйтесь линейной регрессией, получите предсказания и сравните ошибку с предыдущей. Если ошибка не уменьшилась, попробуйте объяснить почему. Если она осталась прежней, предположите, стоит ли при использовании модели удалять данный признак из данных.