### МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАНА

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

***Практическая работа***

***по предмету «Машинное обучение»***

Выполнил: студент гр.721-19

Хакимбеков Дониёрбек

Приняла: Маннапова Мафтуна

Ташкент 2021

Практический пример: машинное обучение без учителя, часть 2 — кластеризация методом k средних

В этом разделе будет представлен, пожалуй, самый простой из алгоритмов машинного обучения без учителя — кластеризация методом k средних. Алгоритм анализирует непомеченные образцы и пытается объединить их в кластеры. Поясним, что k в «методе k средних» представляет количество кластеров, на которые предполагается разбить данные.  
  
Алгоритм распределяет образцы на заранее заданное количество кластеров, используя метрики расстояния, сходные с метриками алгоритма кластеризации k ближайших соседей. Каждый кластер группируется вокруг центроида — центральной точки кластера. Изначально алгоритм выбирает k случайных центроидов среди образцов набора данных, после чего остальные образцы распределяются по кластерам с ближайшим центроидом. Далее выполняется итеративный пересчет центроидов, а образцы перераспределяются по кластерам, пока для всех кластеров расстояние от заданного центроида до образцов, входящих в его кластер, не будет минимизировано. В результате выполнения алгоритма формируется одномерный массив меток, обозначающих кластер, к которому относится каждый образец, а также двумерный массив центроидов, представляющих центр каждого кластера.

Набор данных Iris

Поработаем с популярным набором данных Iris, входящим в поставку scikit-learn. Этот набор часто анализируется при классификации и кластеризации. И хотя набор данных помечен, мы не будем использовать эти метки, чтобы продемонстрировать кластеризацию. Затем метки будут использованы для определения того, насколько хорошо алгоритм k средних выполняет кластеризацию образцов.  
  
Набор данных Iris относится к «игрушечным» наборам данных, поскольку состоит только из 150 образцов и четырех признаков. Набор данных описывает 50 образцов трех видов цветов ириса — Iris setosa, Iris versicolor и Iris virginica (см. фотографии ниже). Признаки образцов: длина наружной доли околоцветника (sepal length), ширина наружной доли околоцветника (sepal width), длина внутренней доли околоцветника (petal length) и ширина внутренней доли околоцветника (petal width), измеряемые в сантиметрах.

14.7.1. Загрузка набора данных Iris

Запустите IPython командой ipython --matplotlib, после чего воспользуйтесь функцией load\_iris модуля sklearn.datasets для получения объекта Bunch с набором данных:

In [1]: **from** sklearn.datasets **import** load\_iris

In [2]: iris = load\_iris()

Атрибут DESCR объекта Bunch показывает, что набор данных состоит из 150 образцов (Number of Instances), каждый из которых обладает четырьмя признаками (Number of Attributes). В наборе данных нет отсутствующих значений. Образцы классифицируются целыми числами 0, 1 и 2, представляющими Iris setosa, Iris versicolor и Iris virginica соответственно. Проигнорируем метки и поручим определение классов образцов алгоритму кластеризации методом k средних. Ключевая информация DESCR выделена жирным шрифтом:

In [3]: print(iris.DESCR)

.. \_iris\_dataset:

Iris plants dataset

--------------------

\*\*Data Set Characteristics:\*\*

:Number of Instances: 150 (50 **in** each of three classes)

:Number of Attributes: 4 numeric, predictive attributes **and** the **class**

:Attribute Information:

- sepal length **in** cm

- sepal width **in** cm

- petal length **in** cm

- petal width **in** cm

- **class**:

- Iris-Setosa

- Iris-Versicolour

- Iris-Virginica

:Summary Statistics:

============== ==== ==== ======= ===== ====================

Min Max Mean SD Class Correlation

============== ==== ==== ======= ===== ====================

sepal length: 4.3 7.9 5.84 0.83 0.7826

sepal width: 2.0 4.4 3.05 0.43 -0.4194

petal length: 1.0 6.9 3.76 1.76 0.9490 (high!)

petal width: 0.1 2.5 1.20 0.76 0.9565 (high!)

============== ==== ==== ======= ===== ====================

:Missing Attribute Values: None

:Class Distribution: 33.3% **for** each of 3 classes.

:Creator: R.A. Fisher

:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)

:Date: July, 1988

...

Проверка количества образцов, признаков и целевых значений

Количество образцов и признаков можно узнать из атрибута shape массива data, а количество целевых значений — из атрибута shape массива target:

In [4]: iris.data.shape

Out[4]: (150, 4)

In [5]: iris.target.shape

Out[5]: (150,)

Массив target\_names содержит имена числовых меток массива. Выражение target — dtype='<U10' означает, что его элементами являются строки длиной не более 10 символов:

In [6]: iris.target\_names

Out[6]: array(['setosa', 'versicolor', 'virginica'], dtype='<U10')

Массив feature\_names содержит список строковых имен для каждого столбца в массиве data:

In [7]: iris.feature\_names

Out[7]:

['sepal length (cm)',

'sepal width (cm)',

'petal length (cm)',

'petal width (cm)']

14.7.2. Исследование набора данных Iris: описательная статистика в Pandas

Используем коллекцию DataFrame для исследования набора данных Iris. Как и в случае с набором данных California Housing, зададим параметры pandas для форматирования столбцового вывода:

In [8]: **import** pandas **as** pd

In [9]: pd.set\_option('max\_columns', 5)

In [10]: pd.set\_option('display.width', None)

Создадим коллекцию DataFrame с содержимым массива data, используя содержимое массива feature\_names как имена столбцов:

In [11]: iris\_df = pd.DataFrame(iris.data, columns=iris.feature\_names)

Затем добавим столбец с названием вида для каждого из образцов. Трансформация списка в следующем фрагменте использует каждое значение в массиве target для поиска соответствующего названия в массиве target\_names:

In [12]: iris\_df['species'] = [iris.target\_names[i] **for** i **in** iris.target]

Воспользуемся pandas для идентификации нескольких образцов. Как и прежде, если pandas выводит \ справа от имени столбца, это означает, что в выводе остаются столбцы, которые будут выведены ниже:

In [13]: iris\_df.head()

Out[13]:

sepal length (cm) sepal width (cm) petal length (cm) \

0 5.1 3.5 1.4

1 4.9 3.0 1.4

2 4.7 3.2 1.3

3 4.6 3.1 1.5

4 5.0 3.6 1.4

petal width (cm) species

0 0.2 setosa

1 0.2 setosa

2 0.2 setosa

3 0.2 setosa

4 0.2 setosa

Вычислим некоторые показатели описательной статистики для числовых столбцов:

In [14]: pd.set\_option('precision', 2)

In [15]: iris\_df.describe()

Out[15]:

sepal length (cm) sepal width (cm) petal length (cm) petal width (cm)

count 150.00 150.00 150.00 150.00

mean 5.84 3.06 3.76 1.20

std 0.83 0.44 1.77 0.76

min 4.30 2.00 1.00 0.10

25% 5.10 2.80 1.60 0.30

50% 5.80 3.00 4.35 1.30

75% 6.40 3.30 5.10 1.80

max 7.90 4.40 6.90 2.50

Вызов метода describe для столбца 'species' подтверждает, что он содержит три уникальных значения. Нам заранее известно, что данные состоят из трех классов, к которым относятся образцы, хотя в машинном обучении без учителя это и не всегда так.

In [16]: iris\_df['species'].describe()

Out[16]:

count 150

unique 3

top setosa

freq 50

Name: species, dtype: object

14.7.3. Визуализация набора данных функцией pairplot

Проведем визуализацию признаков в этом наборе данных. Один из способов извлечь информацию о ваших данных — посмотреть, как признаки связаны друг с другом. Набор данных имеет четыре признака. Мы не сможем построить диаграмму соответствия одного признака с тремя другими на одной диаграмме. Тем не менее можно построить диаграмму, на которой будет представлено соответствие между двумя признаками. Фрагмент [20] использует функцию pairplot библиотеки Seaborn для создания таблицы диаграмм, на которых каждый признак сопоставляется с одним из других признаков:

In [17]: **import** seaborn **as** sns

In [18]: sns.set(font\_scale=1.1)

In [19]: sns.set\_style('whitegrid')

In [20]: grid = sns.pairplot(data=iris\_df, vars=iris\_df.columns[0:4],

...: hue='species')

...:

Ключевые аргументы:

* коллекция DataFrame с набором данных, наносимым на диаграмму;
* vars — последовательность с именами переменных, наносимых на диаграмму. Для коллекции DataFrame она содержит имена столбцов. В данном случае используются первые четыре столбца DataFrame, представляющие длину (ширину) наружной доли околоцветника и длину (ширину) внутренней доли околоцветника соответственно;
* hue — столбец коллекции DataFrame, используемый для определения цветов данных, наносимых на диаграмму. В данном случае данные окрашиваются в зависимости от вида ирисов.

Предыдущий вызов pairplot строит следующую таблицу диаграмм 4 × 4:  
  
  
Диаграммы на диагонали, ведущей из левого верхнего в правый нижний угол, показывают распределение признака, выведенного в этом столбце, с диапазоном значений (слева направо) и количеством образцов с этими значениями (сверху вниз). Возьмем распределение длины наружной доли околоцветника:  
  
  
Самая высокая закрашенная область указывает, что диапазон значений длины наружной доли околоцветника (по оси x) для вида Iris setosa составляет приблизительно 4–6 см, а у большинства образцов Iris setosa значения лежат в середине этого диапазона (приблизительно 5 см). Крайняя правая закрашенная область указывает, что диапазон значений длины наружной доли околоцветника (по оси x) для вида Iris virginica составляет приблизительно 4–8,5 см, а у большинства образцов Iris virginica значения лежат между 6 и 7 см.  
  
На других диаграммах в столбце представлены диаграммы разброса данных других признаков относительно признака по оси x. В первом столбце на первых трех диаграммах по оси y представлены ширина наружной доли околоцветника, длина внутренней доли околоцветника и ширина внутренней доли околоцветника соответственно, а на оси x — длина наружной доли околоцветника.  
  
При выполнении этого кода на экране появляется цветное изображение, показывающее отношения между разными видами ирисов на уровне отдельных признаков. Интересно, что на всех диаграммах синие точки Iris setosa четко отделяются от оранжевых и зеленых точек других видов; это говорит о том, что Iris setosa действительно является отдельным классом. Также можно заметить, что других два вида иногда можно перепутать, на что указывают перекрывающиеся оранжевые и зеленые точки. Например, по диаграмме ширины и длины наружной доли околоцветника видно, что точки Iris versicolor и Iris virginica смешиваются. Это говорит о том, что если доступны только измерения наружной доли околоцветника, то различить эти два вида будет сложно.

Вывод результатов pairplot в одном цвете

Если убрать ключевой аргумент hue, то функция pairplot использует только один цвет для вывода всех данных, потому что она не знает, как различать виды при выводе:

In [21]: grid = sns.pairplot(data=iris\_df, vars=iris\_df.columns[0:4])

Как видно из следующей диаграммы, в данном случае диаграммы на диагонали представляют собой гистограммы с распределениями всех значений этого признака независимо от вида. При изучении диаграмм может показаться, что существуют всего два кластера, хотя мы знаем, что в наборе содержатся данные трех видов. Если количество кластеров неизвестно заранее, то можно обратиться к эксперту предметной области, хорошо знакомому с данными. Эксперт может знать, что в наборе данных присутствуют три вида; эта информация может пригодиться при проведении машинного обучения с данными.

Данные, которые мы будем использовать для этого примера, - это набор данных Iris, классический набор данных в машинном обучении и статистике. Он уже включен в модуль  
datasets библиотеки scikit-learn. Мы можем загрузить его, вызвав функцию load\_iris:

[**In** 1]:

from sklearn.datasets import load\_iris

iris\_dataset = load\_iris()

Объект iris, возвращаемый load\_iris, является объектом Bunch, который очень похож на словарь. Он содержит ключи и значения:

[**In** 3]:

print("Ключи iris\_dataset: \n{}".format(iris\_dataset.keys()))

Ключи iris\_dataset:

dict\_keys(['data', 'target', 'target\_names', 'DESCR', 'feature\_names', 'filename'])

Значение ключа DESCR - это краткое описание набора данных. Здесь мы покажем начало описания (оставшуюся часть описания вы можете посмотреть самостоятельно):

[**In** 4]:

print(iris\_dataset['DESCR'][:193] + "\n...")

.. \_iris\_dataset:

Iris plants dataset

--------------------

\*\*Data Set Characteristics:\*\*

:Number **of** Instances: 150 (50 **in** each **of** three classes)

:Number **of** Attributes: 4 numeric, pre

...

Значение ключа target\_names - это массив строк, содержащий сорта цветов, которые мы хотим предсказать:

[**In** 5]:

print("Названия ответов: {}".format(iris\_dataset['target\_names']))

Названия ответов: ['setosa' 'versicolor' 'virginica']

Значение feature\_names - это список строк с описанием каждого признака:

[**In** 6]:

print("Названия признаков: \n{}".format(iris\_dataset['feature\_names']))

Названия признаков:

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

Сами данные записаны в массивах target и data. data - массив NumPy, который содержит количественные измерения длины чашелистиков, ширины чашелистиков,  
длины лепестков и ширины лепестков:

[**In** 7]:

print("Тип массива data: {}".format(type(iris\_dataset['data'])))

Тип массива data: <**class** 'numpy.ndarray'>

Строки в массиве data соответствуют цветам ириса, а столбцы представляют собой четыре признака, которые были измерены для каждого цветка:

[**In** 8]:

print("Форма массива data: {}".format(iris\_dataset['data'].shape))

Форма массива data: (150, 4)

Мы видим, что массив содержит измерения для 150 различных цветов по 4 признакам. Вспомним, что в машинном обучении отдельные элементы называются примерами (samples), а их свойства -  
характеристиками или признаками (feature). Форма (shape) массива данных определяется количеством примеров, умноженным на количество признаков.  
Это является общепринятым соглашением в scikit-learn, и ваши данные всегда будут представлены в этой форме. Ниже приведены значения признаков для первых пяти примеров:

[**In** 11]:

print("Пepвые пять строк массива data:\n{}".format(iris\_dataset['data'][:5]))

Пepвые пять строк массива data:

[[5.1 3.5 1.4 0.2]

[4.9 3. 1.4 0.2]

[4.7 3.2 1.3 0.2]

[4.6 3.1 1.5 0.2]

[5. 3.6 1.4 0.2]]

Взглянув на эти данные, мы видим, что все пять цветов имеют ширину лепестка 0.2 см и первый цветок имеет самую большую длину чашелистика, 5.1 см.

Массив target содержит сорта уже измеренных цветов, тоже записанные в виде массива NumPy:

[**In** 12]:

print("Тип массива target: {}".format(type(iris\_dataset['target'])))

Тип массива target: <**class** 'numpy.ndarray'>

target представляет собой одномерный массив, по одному элементу для каждого цветка:

[**In** 13]:

print("Форма массива target: {}".format(iris\_dataset['target'].shape))

Форма массива target: (150,)

Сорта кодируются как целые числа от 0 до 2:

[**In** 14]:

print("Ответы:\n{}".format(iris\_dataset['target']))

Ответы:

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

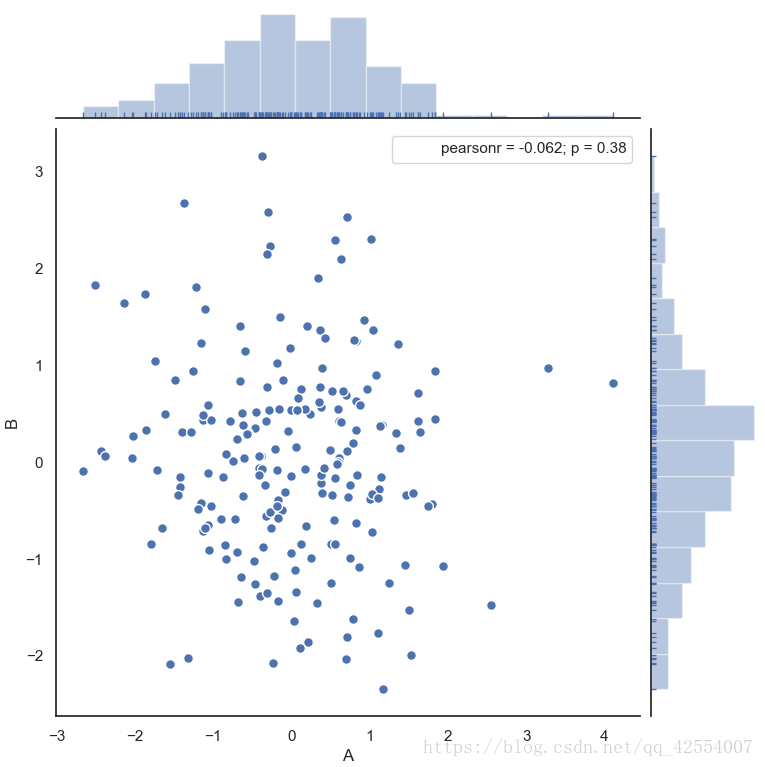
2 2]

Значения чисел задаются массивом iris['target\_names']: 0 - setosa, 1 - versicolor, а 2 - virginica.

### график рассеяния + график распределения

### Пример 1:

1. ***# Создать данные***
2. **rs = np.random.RandomState(2)**
3. **df = pd.DataFrame(rs.randn(200,2), columns = ['A','B'])**
5. **sns.jointplot (x = df ['A'], y = df ['B'], *#Set xy axis, отображать имена столбцов***
6. **data = df, *#set data***
7. **color = 'b', *#Set color***
8. **s = 50, edgecolor = 'w', ширина линии = 1, *# Установить размер разброса, цвет и ширину ребра (только для разброса)***
9. **stat\_func=sci.pearsonr,**
10. **kind = 'scatter', *# Типы наборов: 'scatter', 'reg', 'остаток', 'kde', 'hex'***
11. ***#stat\_func=<function pearsonr>,***
12. **пробел = 0,1, *# Устанавливать пространство между точечной диаграммой и макетом***
13. **размер = 8, размер *# диаграммы (автоматически настраивается на квадрат)***
14. **Соотношение = 5, *# Соотношение высот между точечной диаграммой и макетом, целое число***
15. **marginal\_kws = dict (bins = 15, rug = True), *# Установить количество блоков гистограммы, устанавливать ли коврик***
16. **)**

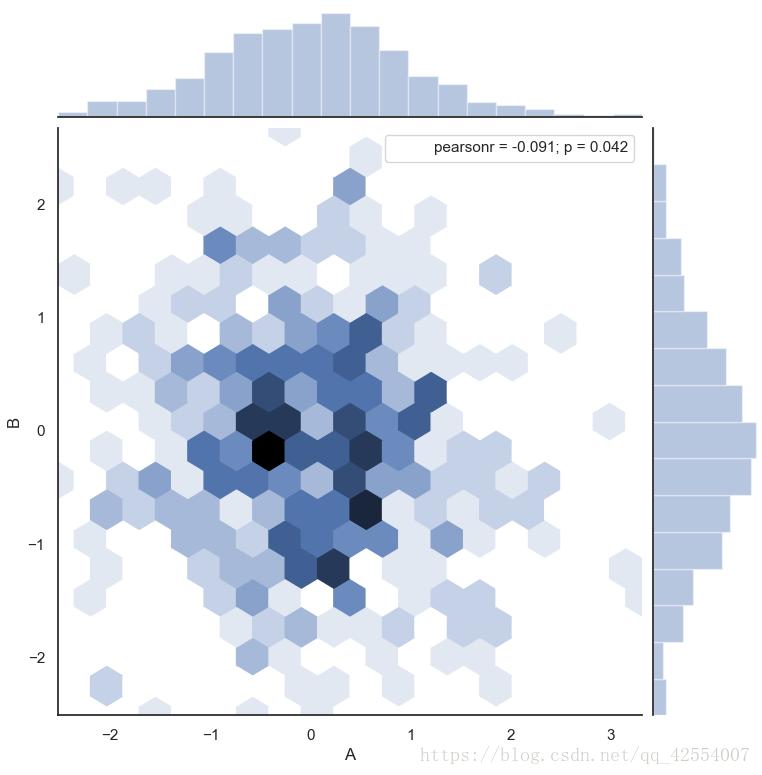


### Рисунок 1. Распределение рассеяния

### Примечание. Если значение r (pearsonr) не отображается, вы можете добавить к параметру stat\_func = sci.pearsonr.

### Пример 2: гексагональный график

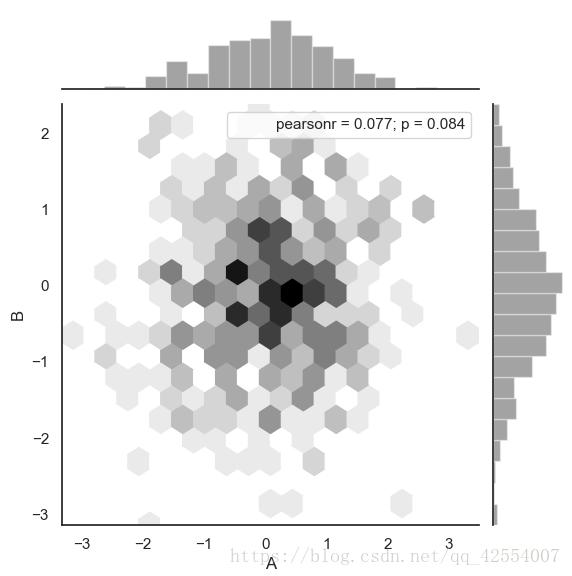
1. **sns.jointplot (x = df ['A'], y = df ['B'], *#Set xy axis, отображать имена столбцов***
2. **data = df, *#set data***
3. **color = 'b', *#Set color***
4. ***#s = 50, edgecolor = 'w', ширина линии = 1, # Установить размер разброса, цвет и ширину кромки (только для разброса)***
5. **stat\_func=sci.pearsonr,**
6. **kind = 'hex', *# Тип набора: 'scatter', 'reg', 'остаток', 'kde', 'hex'***
7. **пробел = 0,1, *# Устанавливать пространство между точечной диаграммой и макетом***
8. **размер = 8, размер *# диаграммы (автоматически настраивается на квадрат)***
9. **Соотношение = 5, *# Соотношение высот между точечной диаграммой и макетом, целое число***
10. ***# marginal\_kws = dict (bins = 15, rug = True) # Установить количество блоков гистограммы, независимо от того, нужно ли устанавливать коврик***
11. **)**



### Гексагональный граф также можно назвать графом улья

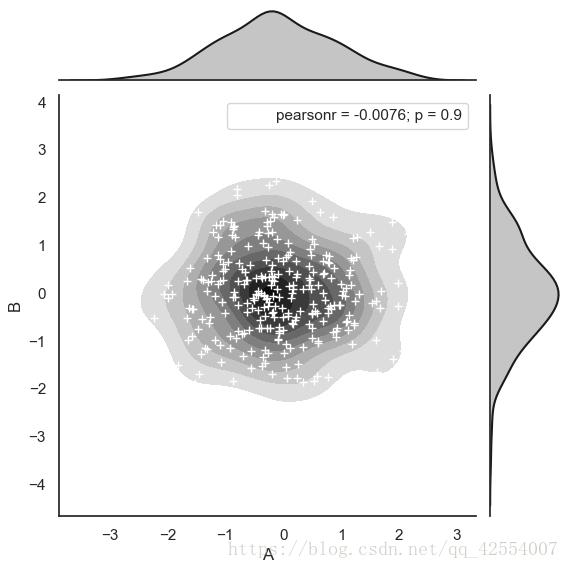
### Пример 3: гексагональный график

1. ***# Создать данные***
2. **df = pd.DataFrame(rs.randn(500,2), columns = ['A', 'B'])**
3. **with sns.axes\_style('white'):**
4. **sns.jointplot(x=df['A'], y=df['B'], data = df, kind = 'hex',**
5. **color = 'k',stat\_func=sci.pearsonr,**
6. **marginal\_kws = dict(bins = 20))**



### Пример 4: Карта плотности

1. ***# Создать данные***
2. **rs = np.random.RandomState(15)**
3. **df = pd.DataFrame(rs.randn(300,2), columns = ['A', 'B'])**
4. ***# Создать карту плотности***
5. **g = sns.jointplot(x = df['A'], y = df['B'], data = df,**
6. **kind = 'kde', color = 'k', stat\_func= sci.pearsonr,**
7. **shade\_lowest = False)**
8. ***# Добавить точечный график***
9. **g.plot\_joint(plt.scatter, c = 'w', s = 30, linewidth = 1, marker='+')**



### 1.2 Разбросанные участки

### #plot\_joint() + ax\_marg\_x.hist() + ax\_marg\_y.hist()

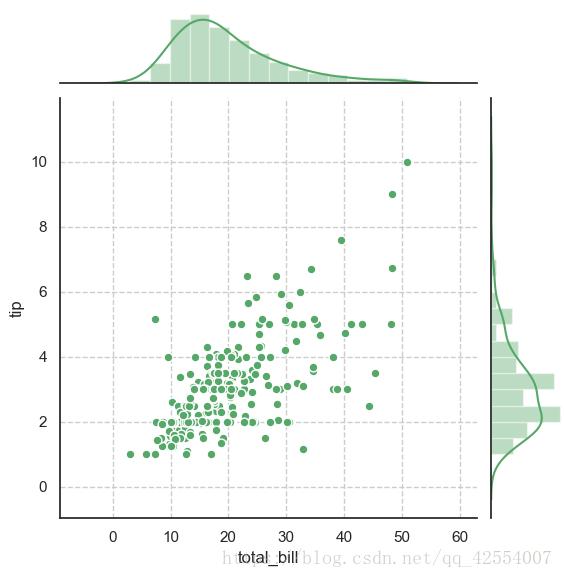
### Пример 1: Сплит-график

2. ***# Установить стиль***
3. **sns.set\_style('white')**
4. ***# Импорт данных***
5. **tips = sns.load\_dataset('tips')**
6. **print(tips.head())**
8. ***# Создать область таблицы чертежа, установить соответствующие данные по x, y***
9. **g = sns.JointGrid(x = 'total\_bill', y = 'tip', data = tips)**
11. **g.plot\_joint (plt.scatter, color = 'm', edgecolor = 'white') *# Установите диаграмму в поле, разброс***
12. **g.ax\_marg\_x.hist(tips['total\_bill'], color='b', alpha = .6,**
13. **bins = np.arange (0,60,3)) *# Установите ось x как гистограмму, обратите внимание, что bin является массивом***
14. **g.ax\_marg\_y.hist(tips['tip'], color = 'r', alpha = .6,**
15. **orientation = 'horizontal',**
16. **bins = np.arange (0,12,1)) *# Установите гистограмму оси x, обратите внимание, что параметр ориентации является обязательным***
17. **from scipy import stats**
18. **g.annotate(stats.pearsonr)**
19. ***# Установить аннотации, может быть Pearsonar, Spearmanr***
20. **plt.grid(linestyle = '--')**

### 

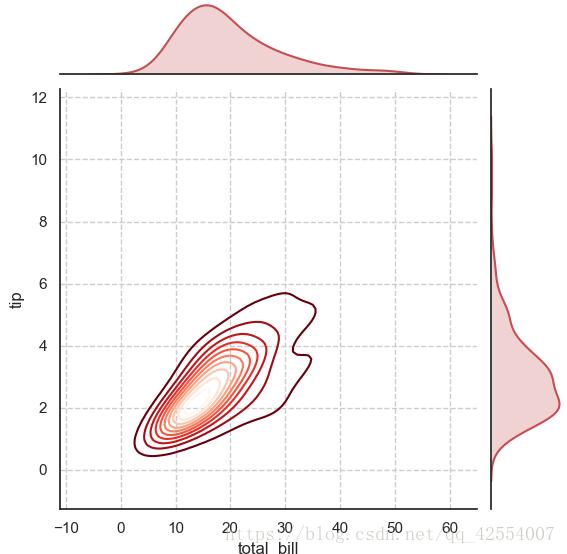
### Пример 2: две гистограммы установлены в одной функции

1. ***# Создать область таблицы чертежа, установить соответствующие данные по x, y***
2. **g = sns.JointGrid(x = 'total\_bill', y = 'tip', data = tips)**
3. **g = g.plot\_joint (plt.scatter, color = 'g', s = 40, edgecolor = 'white') *# Создание точечной диаграммы***
4. **plt.grid(linestyle = '--')**
5. **g.plot\_marginals (sns.distplot, kde = True, color = 'g') *#Draw x, y гистограмма***



### Пример 3: график плотности kde

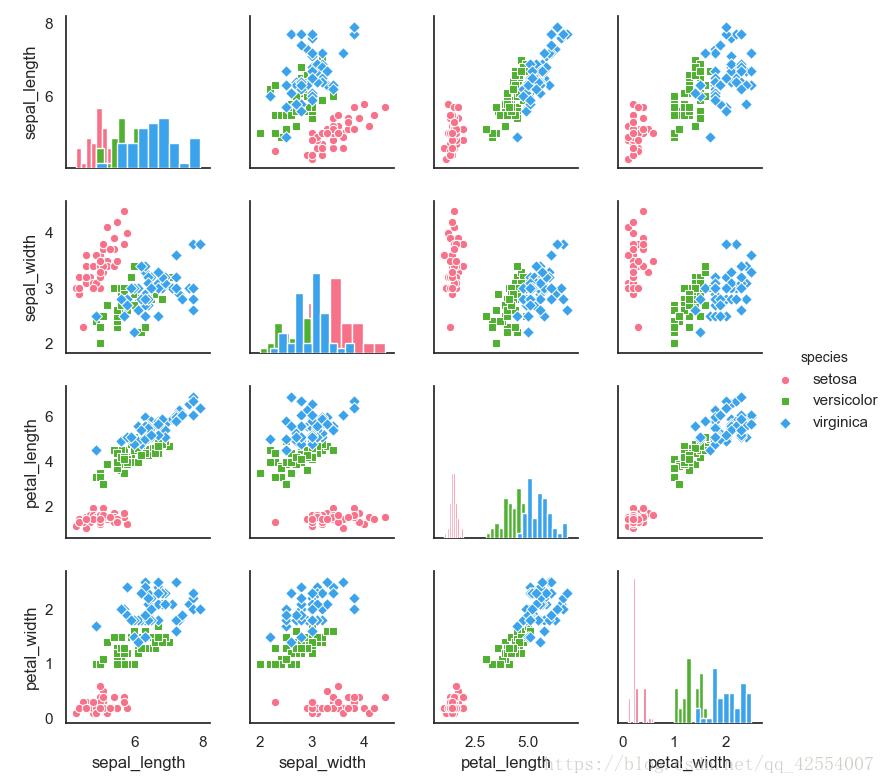
1. ***# Создать область таблицы чертежа, установить соответствующие данные по x, y***
2. **g = sns.JointGrid(x = 'total\_bill', y = 'tip', data = tips)**
3. **g = g.plot\_joint (sns.kdeplot, cmap = 'Reds\_r') *# Постройте график плотности***
4. **plt.grid(linestyle = '--')**
5. **g.plot\_marginals (sns.kdeplot, shade = True, color = 'r') *# Построить карту плотности по осям x, y***



### 2. Матрица рассеяния plot-pairplot ()

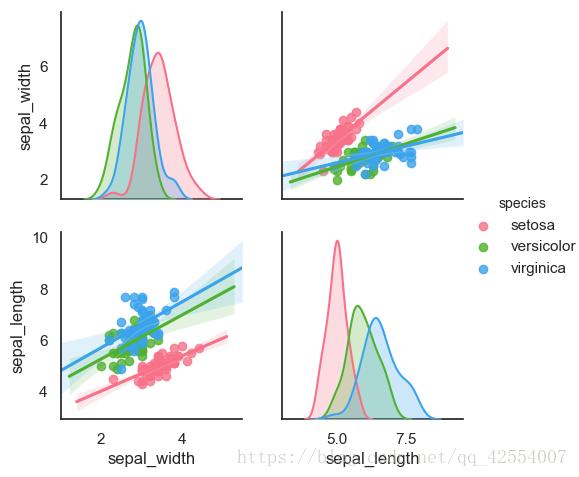
### Пример 1: Общая матричная диаграмма

1. ***# Установить стиль***
2. **sns.set\_style('white')**
3. ***# Читать данные***
4. **iris = sns.load\_dataset('iris')**
5. **print(iris.head())**
6. **sns.pairplot(iris,**
7. **kind = 'scatter', *# Scatter plot / map of regression {'scatter', 'reg'})***
8. **diag\_kind = 'hist', *# Гистограмма / Карта плотности {'hist', 'kde'}***
9. **hue = 'разновидности', *# классифицировать по полю***
10. **palette = 'husl', *# Установить палитру***
11. **markers = ['o', 's', 'D'], *# Установить стиль точки для разных серий (количество категорий в соответствии со ссылкой здесь)***
12. **размер = 2 *# размер значка***
13. **)**



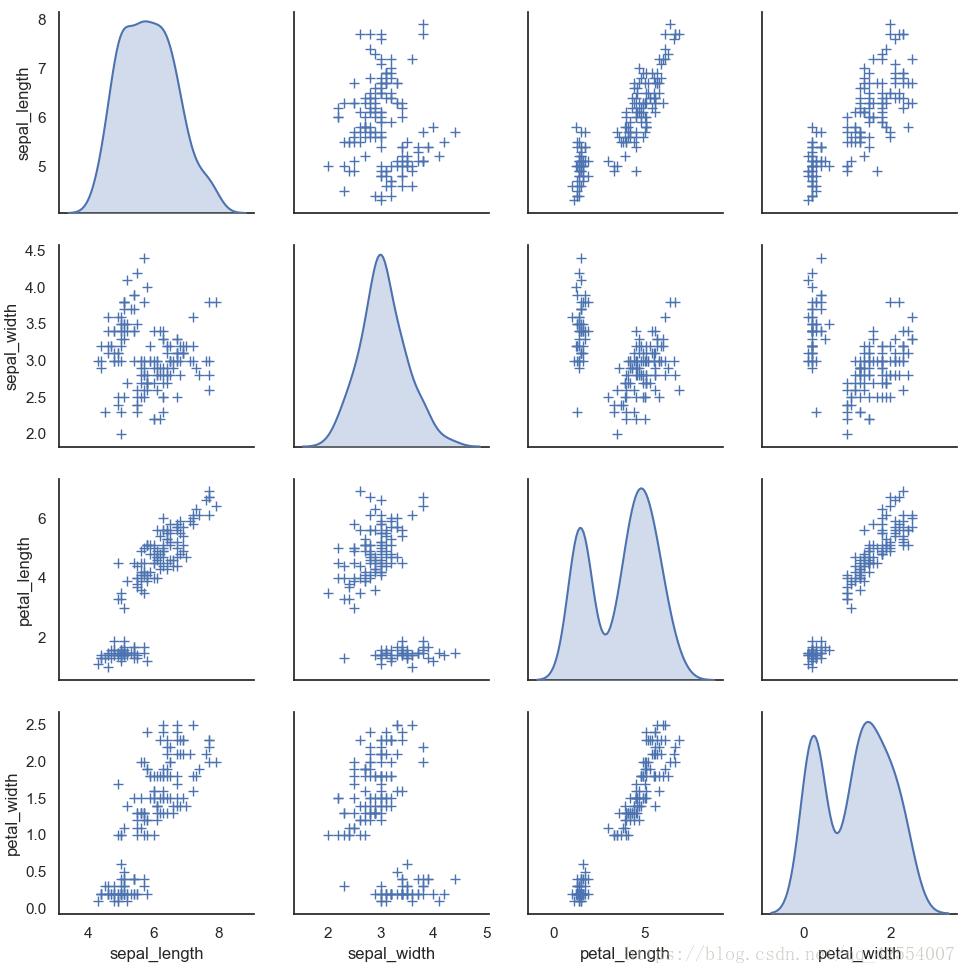
### Пример 2. Извлечение только локальных переменных для сравнения

1. **g = sns.pairplot(iris, vars = ['sepal\_width', 'sepal\_length'],**
2. **kind = 'reg', diag\_kind = 'kde',**
3. **hue = 'species', palette = 'husl')**



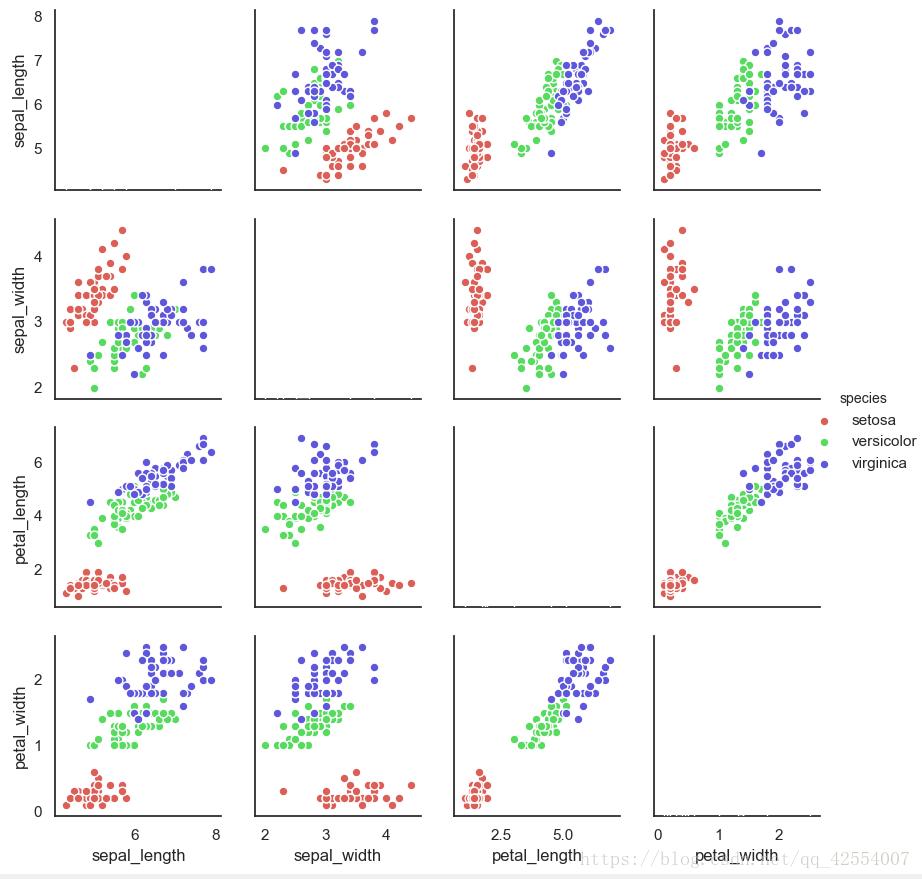
### Пример 3: Другие настройки параметров

1. ***# Другие настройки параметров***
2. **sns.pairplot(iris, diag\_kind = 'kde', markers = '+',**
3. **plot\_kws = dict(s = 50, edgecolor = 'b', linewidth = 1),**
4. ***# Установить стиль точки***
5. **diag\_kws = dict(shade = True)**
6. **) *# Установить стиль карты плотности***



### Пример 4: диаграмма рассеяния

1. ***# map\_diag() + map\_offdiag()***
2. **g = sns.PairGrid(iris, hue= 'species', palette = 'hls',**
3. **vars = ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width'])**
4. ***# Может фильтровать для создания области чертежной таблицы, устанавливать данные, соответствующие x, y, классифицированным по видам***
6. **Диагональная диаграмма, plt.hist / sns.kdeplot**
7. **g.map\_diag(plt.hist,**
8. **histtype = 'step', *#Optional: 'bar', 'barstacked', 'step', 'stepfilled'***
9. **linewidth = 1, edgecolor = 'w')**
11. ***# Другие графики: plt.scatter / plt.bar ...***
12. **g.map\_offdiag(plt.scatter, edgecolor = 'w', s = 40, linewidth = 1)**
13. ***# Задать цвет точки, размер, ширину обводки***
14. **g.add\_legend () *# Добавить легенду ()***



### Пример 5: верхний и нижний треугольники #map\_diag () + map\_lower () + map\_upper ()

1. **g = sns.PairGrid(iris)**
2. **g.map\_diag (sns.kdeplot, lw = 3) *# Установить диагональную диаграмму***
3. **g.map\_upper (plt.scatter, color = 'r') *# Установить диаграмму по диагонали***
4. **g.map\_lower (sns.kdeplot, cmap = 'Blues\_d') *# Установить нижний конец графика***

