Курс “Python для DataScience”

Практическое задание

Тема “Визуализация данных в Matplotlib”

Задание 1

- Загрузите модуль pyplot библиотеки matplotlib с псевдонимом plt, а также библиотеку numpy с псевдонимом np.
- Примените магическую функцию %matplotlib inline для отображения графиков в Jupyter Notebooki настройки конфигурации ноутбука со значением 'svg' для более четкого отображения графиков.

B [9]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
%config InlineBackend.figure_format = 'svg'
```

- Создайте список под названием x с числами 1, 2, 3, 4, 5, 6, 7 и список y с числами 3.5, 3.8, 4.2, 4.5, 5, 5.5, 7.

B [10]:
```
x = np.arange(1, 8)
print(x)
```

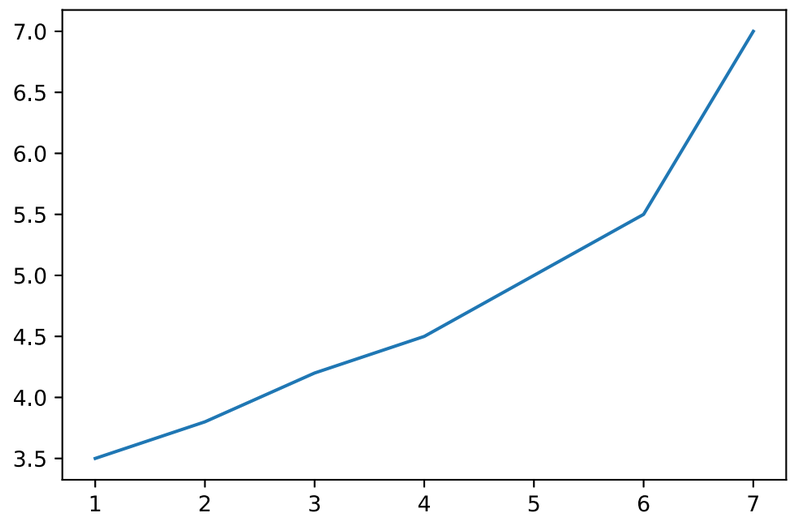
```
[1 2 3 4 5 6 7]
```

B [11]:
```
y = [3.5, 3.8, 4.2, 4.5, 5, 5.5, 7.]
print(y)
```

```
[3.5, 3.8, 4.2, 4.5, 5, 5.5, 7.0]
```

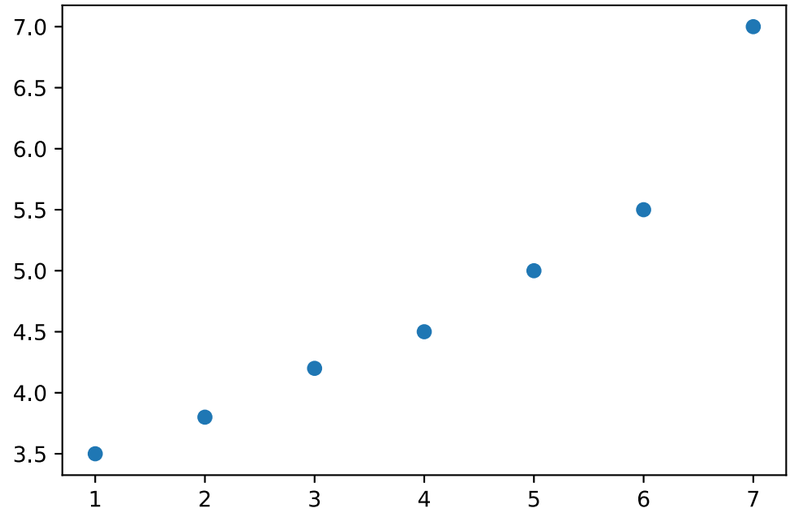
- С помощью функции plot постройте график, соединяющий линиями точки с горизонтальными координатами из списка x и вертикальными - из списка y.

In [12]:
```
plt.plot(x, y)
plt.show()
```



- Затем в следующей ячейке постройте диаграмму рассеяния (другие названия - диаграмма разброса, scatter plot).

In [13]:
```
plt.scatter(x, y)
plt.show()
```



Задание 2

- С помощью функции linspace из библиотеки Numpy создайте массив t из 51 числа от 0 до 10 включительно.

B [14]:
```
t = np.linspace(0, 10, 51)
print(t)
```

```
[ 0.   0.2  0.4  0.6  0.8  1.   1.2  1.4  1.6  1.8  2.   2.2  2.4  2.6
  2.8  3.   3.2  3.4  3.6  3.8  4.   4.2  4.4  4.6  4.8  5.   5.2  5.4
  5.6  5.8  6.   6.2  6.4  6.6  6.8  7.   7.2  7.4  7.6  7.8  8.   8.2
  8.4  8.6  8.8  9.   9.2  9.4  9.6  9.8 10. ]
```

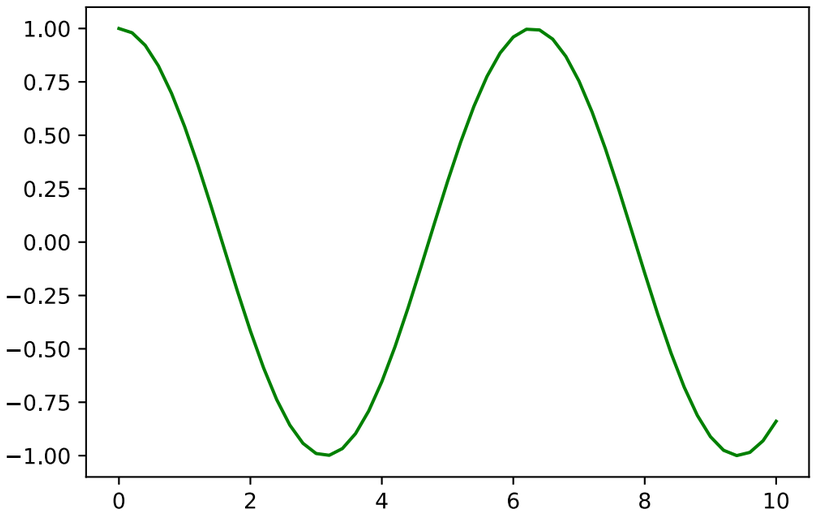
- Создайте массив Numpy под названием f, содержащий косинусы элементов массива t.

B [15]:
```
f = np.cos(t)
print(f)
```

```
[ 1.          0.98006658  0.92106099  0.82533561  0.69670671  0.54030231
  0.36235775  0.16996714 -0.02919952 -0.22720209 -0.41614684 -0.58850112
 -0.73739372 -0.85688875 -0.94222234 -0.9899925  -0.99829478 -0.96679819
 -0.89675842 -0.79096771 -0.65364362 -0.49026082 -0.30733287 -0.11215253
  0.08749898  0.28366219  0.46851667  0.63469288  0.77556588  0.88551952
  0.96017029  0.9965421   0.99318492  0.95023259  0.86939749  0.75390225
  0.60835131  0.43854733  0.25125984  0.05395542 -0.14550003 -0.33915486
 -0.51928865 -0.67872005 -0.81109301 -0.91113026 -0.97484362 -0.99969304
 -0.98468786 -0.93042627 -0.83907153]
```

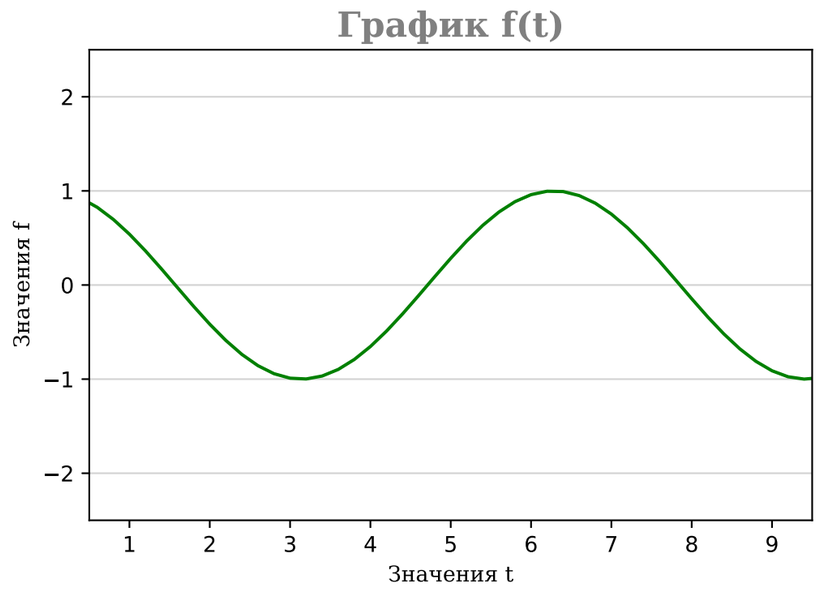
- Постройте линейную диаграмму, используя массив t для координат по горизонтали,а массив f - для координат по вертикали. Линия графика должна быть зеленого цвета.

B [16]:
```
plt.plot(t, f, color='green')
plt.show()
```



- Выведите название диаграммы - 'График f(t)'. Также добавьте названия для горизонтальной оси - 'Значения t' и для вертикальной - 'Значения f'.
- Ограничьте график по оси x значениями 0.5 и 9.5, а по оси y - значениями -2.5 и 2.5.

B [17]:
```
title_dict = {'fontsize': 16, 'fontweight':'bold', 'family': 'serif', 'color': '#808080', 'family': 'serif'}
label_font = {'fontsize': 10, 'family': 'serif'}

plt.plot(t, f, color='green')
plt.title('График f(t)', fontdict=title_dict)
plt.xlabel('Значения t', fontdict=label_font)
plt.ylabel('Значения f', fontdict=label_font)
plt.grid(axis='y', color='lightgrey')
#plt.grid(color='lightgrey')
plt.axis([0.5, 9.5, -2.5, 2.5])
plt.show()
```



*Задание 3

B [18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
%config InlineBackend.figure_format = 'svg'
```

С помощью функции linspace библиотеки Numpy создайте массив x из 51 числа от -3 до 3 включительно.

B [19]:
```
x = np.linspace(-3, 3, 51)
# print(x)
```

Создайте массивы y1, y2, y3, y4 по следующим формулам:

y1 = x**2

y2 = 2 * x + 0.5

y3 = -3 * x - 1.5

y4 = sin(x)

B [20]:
```
y1 = x**2
y2 = 2*x+0.5
y3 =-3*x-1.5
y4 = np.sin(x)
# print(y1)
```

Используя функцию subplots модуля matplotlib.pyplot, создайте объект matplotlib.figure.Figure с названием fig и массив объектов Axes под названием ax, причем так, чтобы у вас было 4 отдельных графика в сетке, состоящей из двух строк и двух столбцов.

В каждом графике массив x используется для координат по горизонтали.

В левом верхнем графике для координат по вертикали используйте y1, в правом верхнем - y2, в левом нижнем - y3, в правом нижнем - y4.

Дайте название графикам: 'График y1', 'График y2' и т.д.

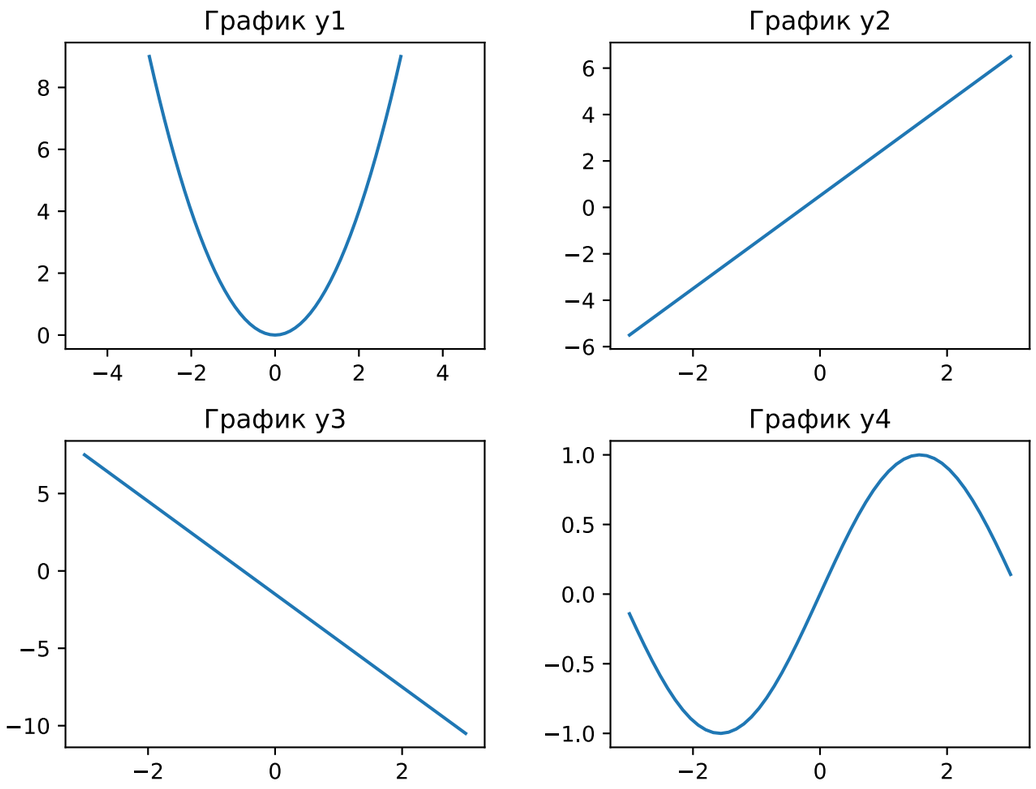
Для графика в левом верхнем углу установите границы по оси x от -5 до 5.

Установите размеры фигуры 8 дюймов по горизонтали и 6 дюймов по вертикали.

Вертикальные и горизонтальные зазоры между графиками должны составлять 0.3.

В [21]:
```
fig, ax = plt.subplots(nrows=2, ncols=2)  # создаём объект Figure с названием fig и массив объектов Axes под названием ax
ax1, ax2, ax3, ax4 = ax.flatten()  # Записываем ax в кортеж содержащий переменные ax1, ax2, ax3, ax4

fig.set_size_inches(8, 6)  # Установливаем размеры фигуры 8 дюймов по горизонтали и 6 дюймов по вертикали.
fig.subplots_adjust(wspace=0.3, hspace=0.3)  # Вертикальные и горизонтальные зазоры между графиками устанавливаем 0.3.

ax1.plot(x, y1)  # Строим 1-й график
ax1.set_title('График y1')  # Даём название графику: 'График y1'
ax1.set_xlim([-5, 5])  # Для графика в левом верхнем углу установливаем границы по оси x от -5 до 5.

ax2.plot(x, y2)  # Строим 2-й график
ax2.set_title('График y2')

ax3.plot(x, y3)  # Строим 3-й график
ax3.set_title('График y3')

ax4.plot(x, y4)  # Строим 4-й график
ax4.set_title('График y4')
```

Out[21]:
```
Text(0.5, 1.0, 'График y4')
```



***Задание 4**

В этом задании мы будем работать с датасетом, в котором приведены данные по мошенничеству с кредитными данными:

Credit Card Fraud Detection (информация об авторах: Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015).

Ознакомьтесь с описанием и скачайте датасет creditcard.csv с сайта Kaggle.com по ссылке:

Credit Card Fraud Detection (https://www.kaggle.com/mlg-ulb/creditcardfraud)

Данный датасет является примером несбалансированных данных, так как мошеннические операции с картами встречаются реже обычных.

Импортруйте библиотеку Pandas, а также используйте для графиков стиль ***"fivethirtyeight"***.

Подключение библиотек и скриптов

B [22]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
%config InlineBackend.figure_format = 'svg'
plt.style.use('fivethirtyeight')
```

Пути к директориям и файлам

B [23]:
```
DATASET_PATH = './creditcard.csv' # Путь до датасета, можно взять на kaggle.com (California Housing Prices)
PREPARED_DATASET_PATH = './creditcard.csv_prepared.csv' # Путь до предобработанного датасета
DATASET_PATH
```

Out[23]:
```
'./creditcard.csv'
```

B [24]:
```
#!dir
```

B [25]:
```
import os

os.listdir('/Users/sil/Desktop/Python_for_DataSience/Lesson4/HW')
```

Out[25]:
```
['.ipynb_checkpoints',
 'creditcard.csv',
 'lesson_4_hw_1.ipynb',
 'lesson_4_hw_1.pdf']
```

1. Загрузка данных

B [26]:
```
df = pd.read_csv(DATASET_PATH, sep=',')  # Читаем DataSet
#df.head(10)
df
```

Out[26]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651	0.823731	0.77	0
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472	-0.053527	24.79	0
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455	-0.026561	67.88	0
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821	0.104533	10.00	0
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267	-0.002415	0.013649	217.00	0

284807 rows × 31 columns

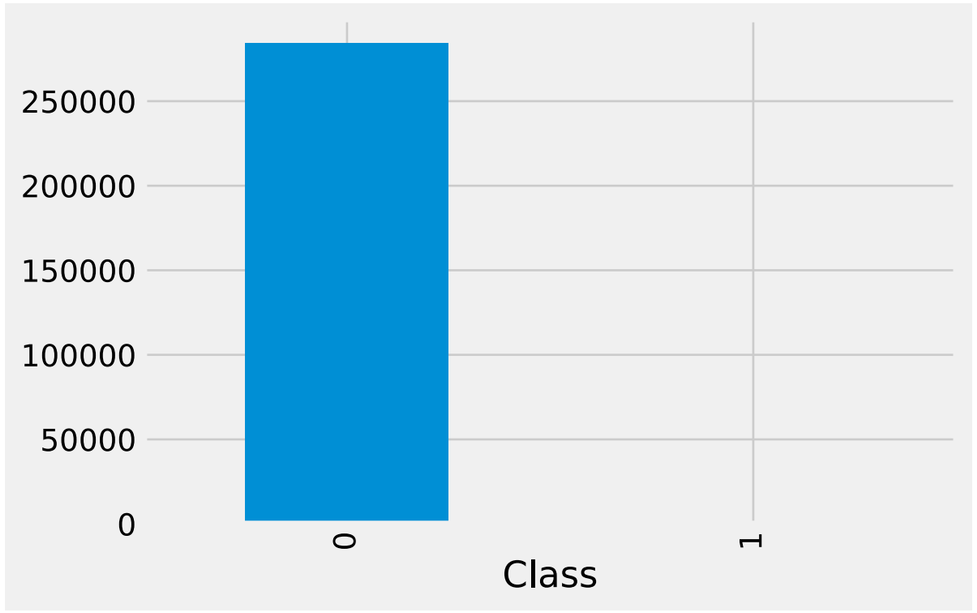
Посчитайте с помощью метода ***value_counts*** количество наблюдений для каждого значения целевой переменной ***Class*** и примените к полученным данным метод ***plot***, чтобы построить столбчатую диаграмму.

B [27]:
```
vc_class = df.value_counts('Class')
#print(vc_class[0], vc_class[1])
print(vc_class, '\n\n', type(vc_class))
```

```
Class
0    284315
1       492
dtype: int64

 <class 'pandas.core.series.Series'>
```

B [28]:
```
vc_class.plot(kind="bar")
```

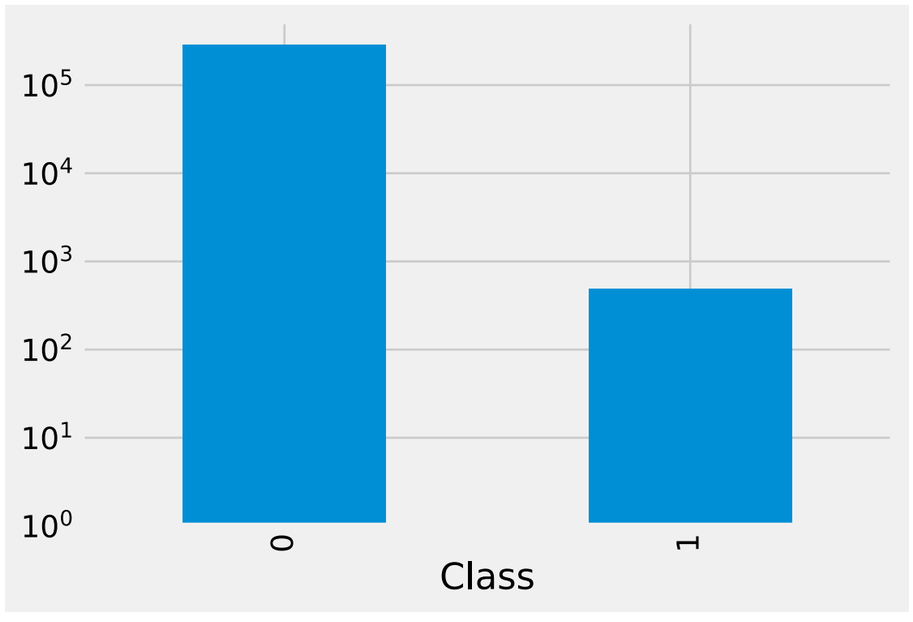
Out[28]:
```
<AxesSubplot:xlabel='Class'>
```



Затем постройте такую же диаграмму, используя логарифмический масштаб.

B [29]:
```
vc_class.plot(kind="bar", log=True)
```

Out[29]:
```
<AxesSubplot:xlabel='Class'>
```



На следующем графике постройте две гистограммы по значениям признака V1 - одну для мошеннических транзакций (Class равен 1) и другую - для обычных (Class равен 0).

B [30]:
```
df['V1'].describe()
```

Out[30]:
```
count    2.848070e+05
mean     3.919560e-15
std      1.958696e+00
min     -5.640751e+01
25%     -9.203734e-01
50%      1.810880e-02
75%      1.315642e+00
max      2.454930e+00
Name: V1, dtype: float64
```

Подберите значение аргумента density так, чтобы по вертикали графика было расположено не число наблюдений, а плотность распределения.

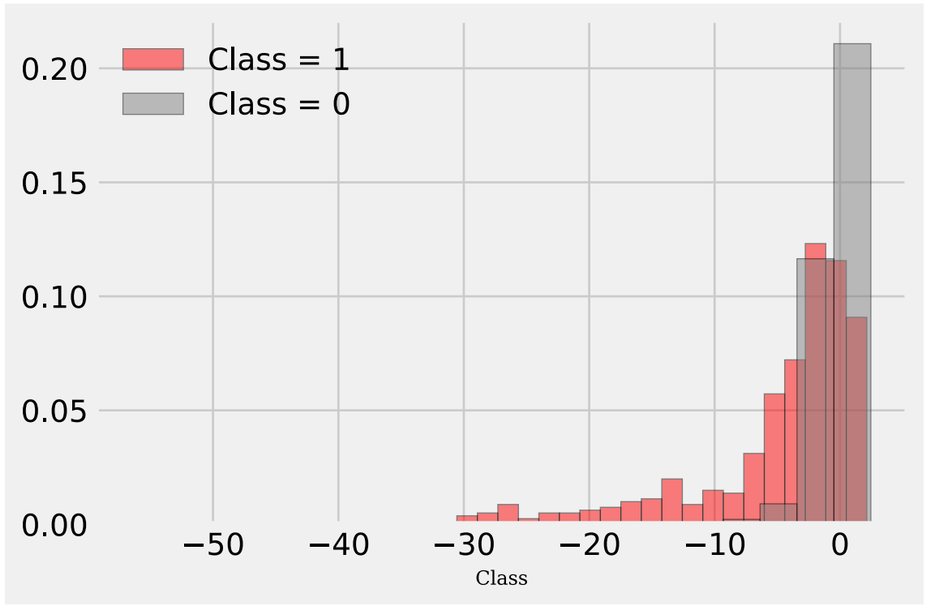
Число бинов должно равняться 20 для обеих гистограмм, а коэффициент alpha сделайте равным 0.5, чтобы гистограммы были полупрозрачными и не загораживали друг друга.

Создайте легенду с двумя значениями: “Class 0” и “Class 1”.

Гистограмма обычных транзакций должна быть серого цвета, а мошеннических - красного.

Горизонтальной оси дайте название “Class”.

B [31]:
```
label_font = {"fontsize": 9, "family": "serif"}

v1 = df.loc[df['Class'] == 1, "V1"]
v0 = df.loc[df['Class'] == 0, 'V1']
dns0=True
dns1=True

# Строим гистограмму по значениям признака V1 - для мошеннических транзакций (Class равен 1)
hist_v1_class_1 = plt.hist(v1,bins=20, density=dns0, ec='black', color='red', alpha=0.5, label="Class = 1")

# Строим гистограмму по значениям признака V1 - для обычных транзакций (Class равен 0)
hist_v1_class_0 = plt.hist(v0, bins=20, density=dns1, ec='black', color='gray', alpha=0.5, label="Class = 0")

plt.xlabel('Class', fontdict=label_font)
plt.legend(loc='upper left', frameon=False)
plt.show()
```



B [32]:
```
plt.hist?
```

PS: Стиль – это предопределенный кластер пользовательских настроек.

Чтобы увидеть доступные стили, используйте:

```
import matplotlib.pyplot as plt
print(plt.style.available)
```

Чтобы настроить стиль, вызовите:

```
plt.style.use('fivethirtyeight')
```

В Matplotlib содержится несколько таблиц стилей для справки. (https://matplotlib.org/gallery.html#style_sheets)

Построение графиков в Python при помощи Matplotlib (https://python-scripts.com/matplotlib)

Оригинал статьи: (https://realpython.com/python-matplotlib-guide/)

**Задание на повторение материала

B [201]:
```
import numpy as np
```

1. Создать одномерный массив Numpy под названием a из 12 последовательных целых чисел чисел от 12 до 24 невключительно

B [209]:
```
#a = np.array(12, 24, 12)
a = np.arange(12, 24)
a
```

Out[209]:
```
array([12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23])
```

2. Создать 5 двумерных массивов разной формы из массива a. Не использовать в аргументах метода reshape число -1.

B [207]:
```
a1 = a.reshape(3, 4)
print('a1 = \n',a1)
a2 = a.reshape(4, 3)
print('a2 = \n',a2)
a3 = a.reshape(2, 6)
print('a3 = \n',a3)
a4 = a.reshape(6, 2)
print('a4 = \n',a4)
a5 = a.reshape(12, 1)
print('a5 = \n',a5)
```

```
a1 =
 [[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]
a2 =
 [[12 13 14]
 [15 16 17]
 [18 19 20]
 [21 22 23]]
a3 =
 [[12 13 14 15 16 17]
 [18 19 20 21 22 23]]
a4 =
 [[12 13]
 [14 15]
 [16 17]
 [18 19]
 [20 21]
 [22 23]]
a5 =
 [[12]
 [13]
 [14]
 [15]
 [16]
 [17]
 [18]
 [19]
 [20]
 [21]
 [22]
 [23]]
```

3. Создать 5 двумерных массивов разной формы из массива a. Использовать в аргументах метода reshape число -1 (в трех примерах - для обозначения числа столбцов, в двух - для строк).

B [213]:
```
a1 = a.reshape(3, -1)
print('a1 = \n',a1)
a2 = a.reshape(4, -1)
print('a2 = \n',a2)
a3 = a.reshape(2, -1)
print('a3 = \n',a3)
a4 = a.reshape(-1, 2)
print('a4 = \n',a4)
a5 = a.reshape(-1, 1)
print('a5 = \n',a5)
```

```
a1 =
 [[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]
a2 =
 [[12 13 14]
 [15 16 17]
 [18 19 20]
 [21 22 23]]
a3 =
 [[12 13 14 15 16 17]
 [18 19 20 21 22 23]]
a4 =
 [[12 13]
 [14 15]
 [16 17]
 [18 19]
 [20 21]
 [22 23]]
a5 =
 [[12]
 [13]
 [14]
 [15]
 [16]
 [17]
 [18]
 [19]
 [20]
 [21]
 [22]
 [23]]
```

4. Можно ли массив Numpy, состоящий из одного столбца и 12 строк, назвать одномерным?

B [214]:
```
# Нельзя т. к. такой массив (a5 из предыдущего роумера) является массивом 12 одномерных массивов, содержащих по одному элементу.
```

B [215]:
```
#print('type(a) = ', type(a))
#print('type(a5) = ', type(a5))

#print('a =\n', a)
#print('a.transpose=\n', np.transpose(a))

#print('\na5 =\n', a5)
#print('a5.transpose=\n', np.transpose(a5))
#print('transpose(a5.transpose)=\n', np.transpose(np.transpose(a5)))
```

5. Создать массив из 3 строк и 4 столбцов, состоящий из случайных чисел с плавающей запятой из нормального распределения со средним, равным 0 и среднеквадратичным отклонением, равным 1.0. Получить из этого массива одномерный массив с таким же атрибутом size, как и исходный массив.

B [216]:
```
# Создать массив из 3 строк и 4 столбцов, состоящий из случайных чисел с плавающей запятой
# из нормального распределения со средним, равным 0 и среднеквадратичным отклонением, равным 1.0.
b = np.random.randn(3, 4)
print('b = \n',b,'\n\nb.size = ',b.size, '\n')
```

```
b =
 [[ 1.57921282  0.76743473 -0.46947439  0.54256004]
 [-0.46341769 -0.46572975  0.24196227 -1.91328024]
 [-1.72491783 -0.56228753 -1.01283112  0.31424733]]

b.size =  12
```

B [218]:
```
# Получить из этого массива одномерный массив с таким же атрибутом size, как и исходный массив.
c = b.flatten()
print('c =\n', c,'\n\nc.size = ',c.size, '\n')
```

```
c =
 [ 1.57921282  0.76743473 -0.46947439  0.54256004 -0.46341769 -0.46572975
  0.24196227 -1.91328024 -1.72491783 -0.56228753 -1.01283112  0.31424733]

c.size =  12
```

6. Создать массив a, состоящий из целых чисел, убывающих от 20 до 0 невключительно с интервалом 2.

B [58]:
```
a = np.arange(20, 0, -2)
print(a)
```

```
[20 18 16 14 12 10  8  6  4  2]
```

7. Создать массив b, состоящий из 1 строки и 10 столбцов: целых чисел, убывающих от 20 до 1 невключительно с интервалом 2. В чем разница между массивами a и b?

B [223]:
```
#b = np.arange(20, 0, -2)
# b = a.reshape(10, 1)
b = np.arange(20, 0, -2).reshape(10,1)
print(b)
# Массив a - одномерный, массив b - многомерный
```

```
[[20]
 [18]
 [16]
 [14]
 [12]
 [10]
 [ 8]
 [ 6]
 [ 4]
 [ 2]]
```

8. Вертикально соединить массивы a и b. a - двумерный массив из нулей, число строк которого больше 1 и на 1 меньше, чем число строк двумерного массива b, состоящего из единиц. Итоговый массив v должен иметь атрибут size, равный 10.

In [224]:
```
# Создаём массив a
a = np.zeros((2, 2))
print('a.size = ', a.size)
print(a)

# Создаём массив b
b = np.ones((3, 2))
print('\nb.size = ', b.size)
print(b)

# Вертикально соединить массивы a и b.
v = np.vstack((a, b))
print('\nv.size = ', v.size)
v
```

```
a.size =  4
[[0. 0.]
 [0. 0.]]

b.size =  6
[[1. 1.]
 [1. 1.]
 [1. 1.]]

v.size =  10
```

Out[224]:
```
array([[0., 0.],
       [0., 0.],
       [1., 1.],
       [1., 1.],
       [1., 1.]])
```

9. Создать одномерный массив a, состоящий из последовательности целых чисел от 0 до 12. Поменять форму этого массива, чтобы получилась матрица A (двумерный массив Numpy), состоящая из 4 строк и 3 столбцов. Получить матрицу At путем транспонирования матрицы A. Получить матрицу B, умножив матрицу A на матрицу At с помощью матричного умножения. Какой размер имеет матрица B? Получится ли вычислить обратную матрицу для матрицы B и почему?

In [225]:
```
# Создать одномерный массив a, состоящий из последовательности целых чисел от 0 до 12.
a = np.arange(0, 12)
print('a = ', a)
```

```
a =  [ 0  1  2  3  4  5  6  7  8  9 10 11]
```

In [231]:
```
# Поменять форму этого массива, чтобы получилась матрица A (двумерный массив Numpy), состоящая из 4 строк и 3 столбцов.
A = a.reshape(4, 3)
A
```

Out[231]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [229]:
```
# Получить матрицу At путем транспонирования матрицы A.
At = np.transpose(A)
At
```

Out[229]:
```
array([[ 0,  3,  6,  9],
       [ 1,  4,  7, 10],
       [ 2,  5,  8, 11]])
```

In [230]:
```
# Получить матрицу B, умножив матрицу A на матрицу At с помощью матричного умножения.
B = A.dot(At)
B
```

Out[230]:
```
array([[  5,  14,  23,  32],
       [ 14,  50,  86, 122],
       [ 23,  86, 149, 212],
       [ 32, 122, 212, 302]])
```

In [232]:
```
# Какой размер имеет матрица B?
print('Количество элементов матрицы B = ', B.size)
print('Размерность B: {}'.format(B.ndim))
r = np.linalg.matrix_rank(B)
print('Ранг матрицы B: {}'.format(r))
print('Форма B: {}'.format(B.shape))
```

```
Количество элементов матрицы B =  16
Размерность B: 2
Ранг матрицы B: 2
Форма B: (4, 4)
```

In [233]:
```
# Получится ли вычислить обратную матрицу для матрицы B и почему?
print('Определитель матрицы B, det(B) = ', np.linalg.det(B))

# Определитель матрицы B, det(B)=0. Поэтому вычислить обратную матрицу нельзя.

#B_inv = np.linalg.inv(B)
#print(B_invinv)
```

```
Определитель матрицы B, det(B) =  0.0
```

10. Инициализируйте генератор случайных числе с помощью объекта seed, равного 42.

In [160]:
```
np.random.seed(42)
```

11. Создайте одномерный массив c, составленный из последовательности 16-ти случайных равномерно распределенных целых чисел от 0 до 16 невключительно.

In [161]:
```
c = np.random.randint(0, 16,16)
print(c)
```

```
[ 6  3 12 14 10  7 12  4  6  9  2  6 10 10  7  4]
```

12. Поменяйте его форму так, чтобы получилась квадратная матрица C. Получите матрицу D, поэлементно прибавив матрицу B из предыдущего вопроса к матрице C, умноженной на 10. Вычислите определитель, ранг и обратную матрицу D_inv для D.

B [188]:
```
C = c.reshape(4, 4)  # Поменяйте его форму так, чтобы получилась квадратная матрица C.
print('c = \n',C)
```

```
c =
 [[ 6  3 12 14]
 [10  7 12  4]
 [ 6  9  2  6]
 [10 10  7  4]]
```

B [189]:
```
D = C * 10 + B  # Получите матрицу D, поэлементно прибавив матрицу B из предыдущего вопроса к матрице C, умноженной на 10.
print('D = \n', D)
```

```
D =
 [[ 65  44 143 172]
 [114 120 206 162]
 [ 83 176 169 272]
 [132 222 282 342]]
```

Вычислите определитель, ранг и обратную матрицу D_inv для D.

B [236]:
```
# Вычислите ранг матрицы D
print('Ранг матрицы D: {}'.format(np.linalg.matrix_rank(D)))
```

```
Ранг матрицы D: 4
```

B [235]:
```
# Вычислите определитель матрицы D
print('Определитель матрицы D, det(D) = ', np.linalg.det(D))
```

```
Определитель матрицы D, det(D) =  -28511999.999999944
```

B [237]:
```
# Вычислите обратную матрицу D_inv для D.
D_inv = np.linalg.inv(D)
print('Обратная матрица D_inv:\n', D_inv)

print('\nПроверка (D*D_inv=I) :\n', D.dot(D_inv))
```

```
Обратная матрица D_inv:
 [[ 0.00935396  0.04486532  0.05897517 -0.07286055]
 [-0.01503577 -0.00122896 -0.00192971  0.00967873]
 [-0.00356692 -0.01782828 -0.04152146  0.04326178]
 [ 0.00909091 -0.00181818  0.01272727 -0.01090909]]

Проверка (D*D_inv=I) :
 [[ 1.00000000e+00  0.00000000e+00 -4.44089210e-16 -8.88178420e-16]
 [ 1.66533454e-16  1.00000000e+00 -4.44089210e-16  0.00000000e+00]
 [ 2.22044605e-16  4.44089210e-16  1.00000000e+00 -2.22044605e-16]
 [ 2.77555756e-16  5.55111512e-16 -8.88178420e-16  1.00000000e+00]]
```

13. Приравняйте к нулю отрицательные числа в матрице D_inv, а положительные - к единице. Убедитесь, что в матрице D_inv остались только нули и единицы. С помощью функции numpy.where, используя матрицу D_inv в качестве маски, а матрицы B и C - в качестве источников данных, получите матрицу E размером 4x4. Элементы матрицы E, для которых соответствующий элемент матрицы D_inv равен 1, должны быть равны соответствующему элементу матрицы B, а элементы матрицы E, для которых соответствующий элемент матрицы D_inv равен 0, должны быть равны соответствующему элементу матрицы C.

B [238]:
```
# Приравняйте к нулю отрицательные числа в матрице D_inv, а положительные - к единице.
# Убедитесь, что в матрице D_inv остались только нули и единицы.
D_inv=np.where(D_inv<0, 0,1)
D_inv
```

Out[238]:
```
array([[1, 1, 1, 0],
       [0, 0, 0, 1],
       [0, 0, 0, 1],
       [1, 0, 1, 0]])
```

B [239]:
```
# С помощью функции numpy.where, используя матрицу D_inv в качестве маски,
# а матрицы B и C - в качестве источников данных, получите матрицу E размером 4x4.
print(B)
print()
print(C)
E=np.where(D_inv==1, B, C)
E
# Элементы матрицы E, для которых соответствующий элемент матрицы D_inv равен 1,
# должны быть равны соответствующему элементу матрицы B,
# а элементы матрицы E, для которых соответствующий элемент матрицы D_inv равен 0,
# должны быть равны соответствующему элементу матрицы C.
```

```
[[  5  14  23  32]
 [ 14  50  86 122]
 [ 23  86 149 212]
 [ 32 122 212 302]]

[[ 6  3 12 14]
 [10  7 12  4]
 [ 6  9  2  6]
 [10 10  7  4]]
```

Out[239]:
```
array([[  5,  14,  23,  14],
       [ 10,   7,  12, 122],
       [  6,   9,   2, 212],
       [ 32,  10, 212,   4]])
```

B [ ]:

B [ ]:

B [ ]: