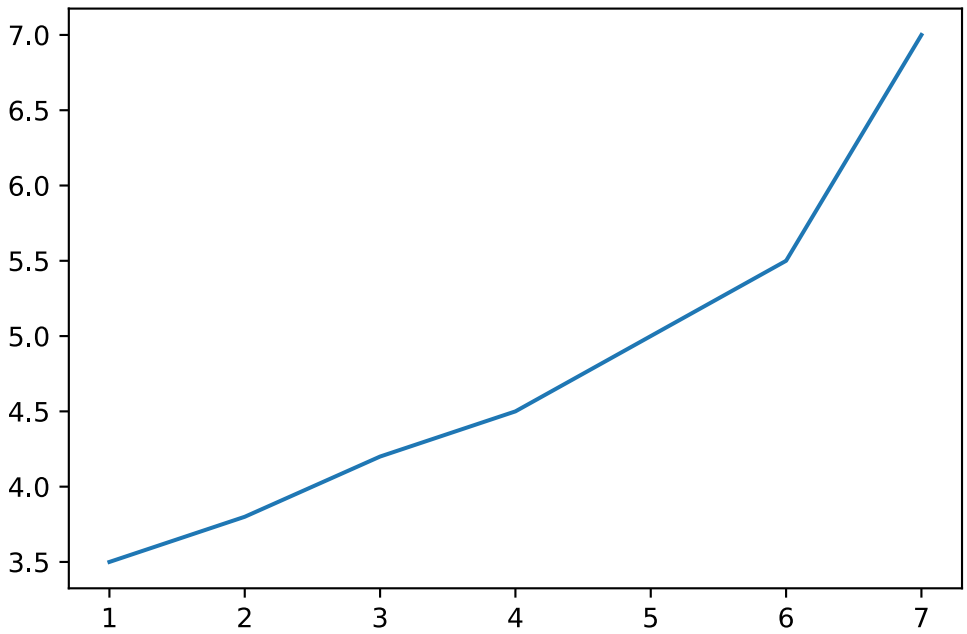
Тема “Визуализация данных в Matplotlib”

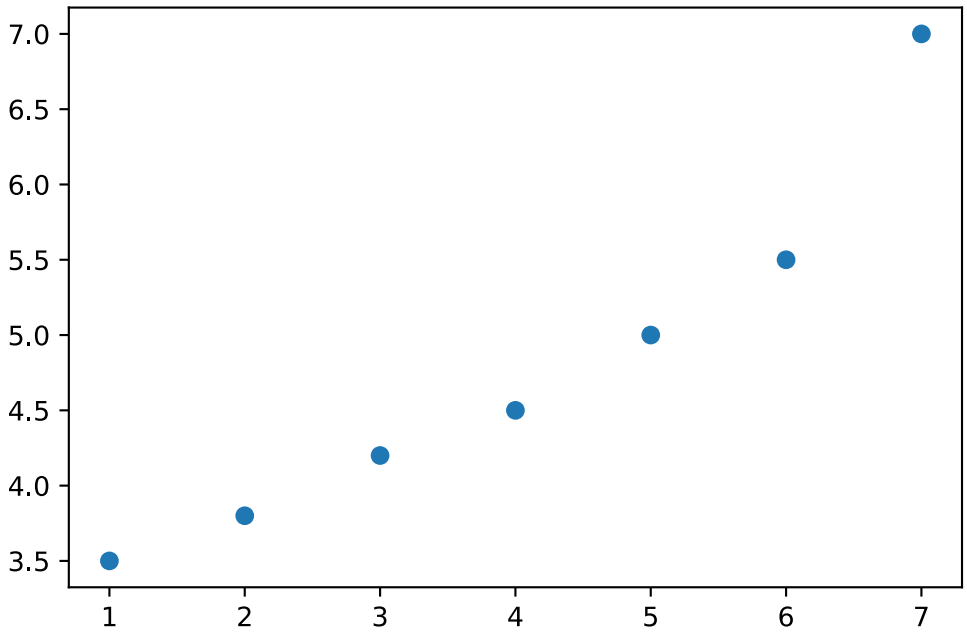
Задание 1

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 Notebook и настройки конфигурации ноутбука со значением 'svg' для более четкого отображения графиков. 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. С помощью функции plot постройте график, соединяющий линиями точки с горизонтальными координатами из списка x и вертикальными - из списка y. Затем в следующей ячейке постройте диаграмму рассеяния (другие названия - диаграмма разброса, scatter plot).

In [1]:
```
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd

%matplotlib inline
%config InlineBackend.figure_format = 'svg'
x = np.array([1, 2, 3, 4, 5, 6, 7])
y = np.array([3.5, 3.8, 4.2, 4.5, 5, 5.5, 7])
# print (x,y)

plt.plot(x,y)
plt.show()
```



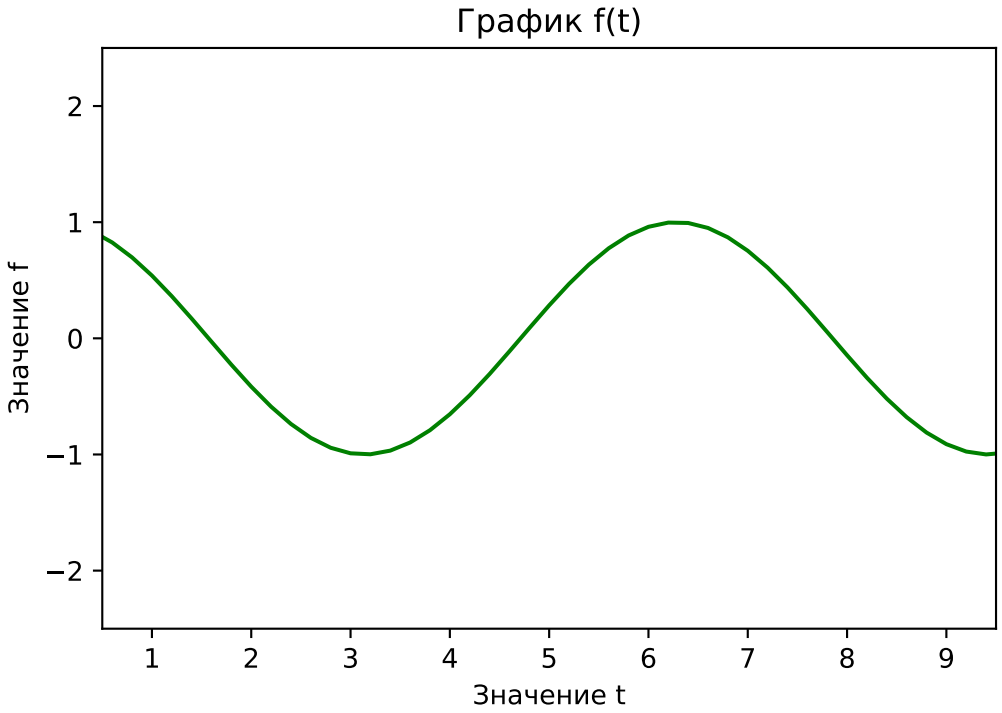
In [2]:
```
plt.scatter(x,y)
plt.show()
```



Задание 2

С помощью функции linspace из библиотеки Numpy создайте массив t из 51 числа от 0 до 10 включительно. Создайте массив Numpy под названием f, содержащий косинусы элементов массива t. Постройте линейную диаграмму, используя массив t для координат по горизонтали,а массив f - для координат по вертикали. Линия графика должна быть зеленого цвета. 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.

In [3]:
```
t = np.linspace(0, 10, 51)
f = np.cos(t)

plt.plot(t, f, color ='green')
plt.title('График f(t)')
plt.xlabel('Значение t')
plt.ylabel('Значение f')
plt.axis([0.5,9.5,-2.5,2.5])

plt.show()
```



*Задание 3

С помощью функции linspace библиотеки Numpy создайте массив x из 51 числа от -3 до 3 включительно. Создайте массивы y1, y2, y3, y4 по следующим формулам:

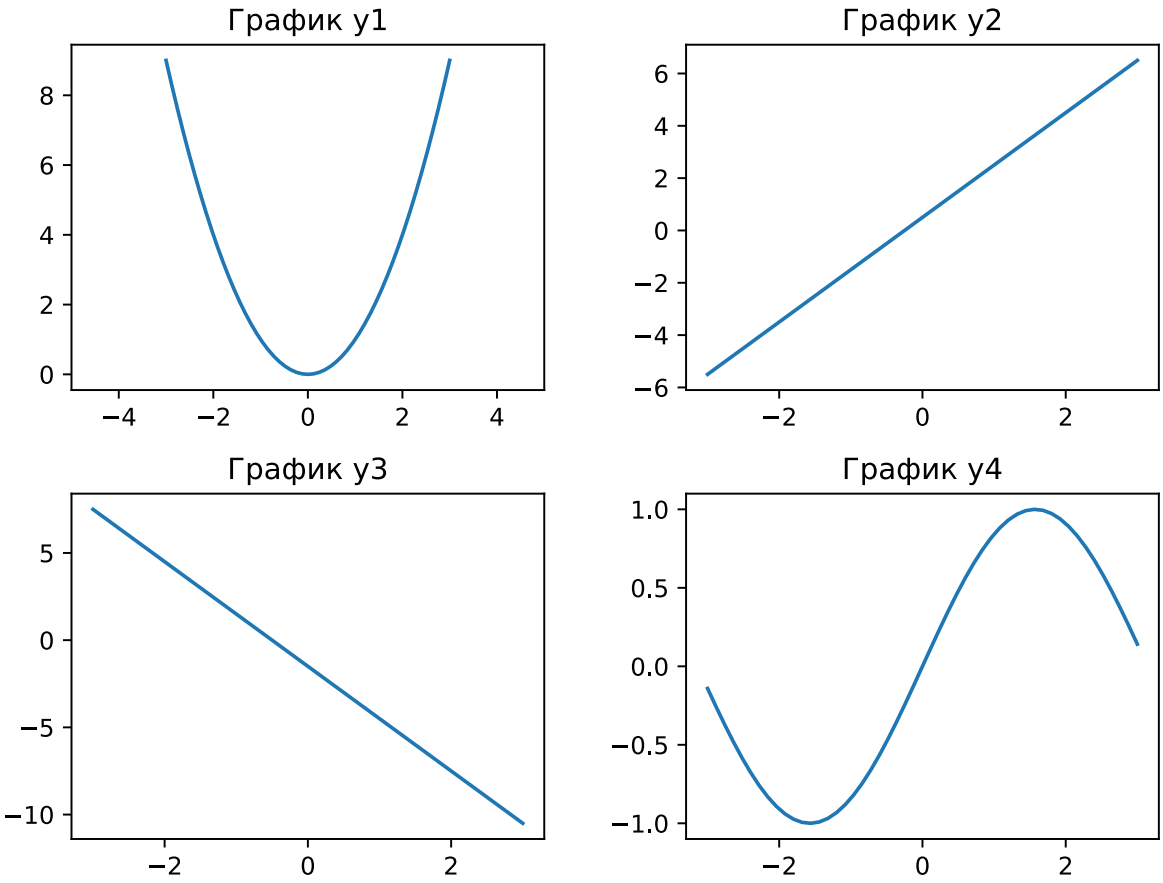
y1 = x*2 *y2 = 2* x + 0.5 y3 = -3 * x - 1.5 y4 = sin(x)

In [4]:
```
x = np.linspace(-3, 3, 51)
print(x)
y1 = x**2
y2 = 2*x + 0.5
y3 = -3*x - 1.5
y4 = np.sin(x)
print(y1,y2,y3,y4)
```

```
[-3.   -2.88 -2.76 -2.64 -2.52 -2.4  -2.28 -2.16 -2.04 -1.92 -1.8  -1.68
 -1.56 -1.44 -1.32 -1.2  -1.08 -0.96 -0.84 -0.72 -0.6  -0.48 -0.36 -0.24
 -0.12  0.    0.12  0.24  0.36  0.48  0.6   0.72  0.84  0.96  1.08  1.2
  1.32  1.44  1.56  1.68  1.8   1.92  2.04  2.16  2.28  2.4   2.52  2.64
  2.76  2.88  3.  ]
[9.     8.2944 7.6176 6.9696 6.3504 5.76   5.1984 4.6656 4.1616 3.6864
 3.24   2.8224 2.4336 2.0736 1.7424 1.44   1.1664 0.9216 0.7056 0.5184
 0.36   0.2304 0.1296 0.0576 0.0144 0.     0.0144 0.0576 0.1296 0.2304
 0.36   0.5184 0.7056 0.9216 1.1664 1.44   1.7424 2.0736 2.4336 2.8224
 3.24   3.6864 4.1616 4.6656 5.1984 5.76   6.3504 6.9696 7.6176 8.2944
 9.    ] [-5.5  -5.26 -5.02 -4.78 -4.54 -4.3  -4.06 -3.82 -3.58 -3.34 -3.1  -2.86
 -2.62 -2.38 -2.14 -1.9  -1.66 -1.42 -1.18 -0.94 -0.7  -0.46 -0.22  0.02
  0.26  0.5   0.74  0.98  1.22  1.46  1.7   1.94  2.18  2.42  2.66  2.9
  3.14  3.38  3.62  3.86  4.1   4.34  4.58  4.82  5.06  5.3   5.54  5.78
  6.02  6.26  6.5 ] [  7.5    7.14   6.78   6.42   6.06   5.7    5.34   4.98   4.6
2   4.26
   3.9    3.54   3.18   2.82   2.46   2.1    1.74   1.38   1.02   0.66
   0.3   -0.06  -0.42  -0.78  -1.14  -1.5   -1.86  -2.22  -2.58  -2.94
  -3.3   -3.66  -4.02  -4.38  -4.74  -5.1   -5.46  -5.82  -6.18  -6.54
  -6.9   -7.26  -7.62  -7.98  -8.34  -8.7   -9.06  -9.42  -9.78 -10.14
 -10.5 ] [-0.14112001 -0.25861935 -0.37239904 -0.48082261 -0.58233065 -0.67546318
 -0.75888071 -0.83138346 -0.89192865 -0.93964547 -0.97384763 -0.9940432
 -0.99994172 -0.99145835 -0.9687151  -0.93203909 -0.88195781 -0.81919157
 -0.74464312 -0.65938467 -0.56464247 -0.46177918 -0.35227423 -0.23770263
 -0.11971221  0.          0.11971221  0.23770263  0.35227423  0.46177918
  0.56464247  0.65938467  0.74464312  0.81919157  0.88195781  0.93203909
  0.9687151   0.99145835  0.99994172  0.9940432   0.97384763  0.93964547
  0.89192865  0.83138346  0.75888071  0.67546318  0.58233065  0.48082261
  0.37239904  0.25861935  0.14112001]
```

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.

In [5]:
```
fig, ax = plt.subplots(nrows=2, ncols=2)
ax1, ax2, ax3, ax4 = ax.flatten()
fig.set_size_inches(8,6)
fig.subplots_adjust(wspace = 0.3, hspace = 0.3)

ax1.plot(x,y1)
ax1.set_xlim([-5,5])
ax1.set_title('График y1')

ax2.plot(x, y2)
ax2.set_title('График y2')

ax3.plot(x, y3)
ax3.set_title('График y3')

ax4.plot(x, y4)
ax4.set_title('График y4')

plt.show()
```



*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

In [6]:
```
#url = 'https://www.kaggle.com/mlg-ulb/creditcardfraud'
#fraud_df = pd.read_csv(url)
#fraud_df.head - не удалось загрузить на прямую

fraud_df = './creditcard.csv'
df = pd.read_csv(fraud_df, sep=',')
df.head(4)
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.3
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.2
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.5
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.3

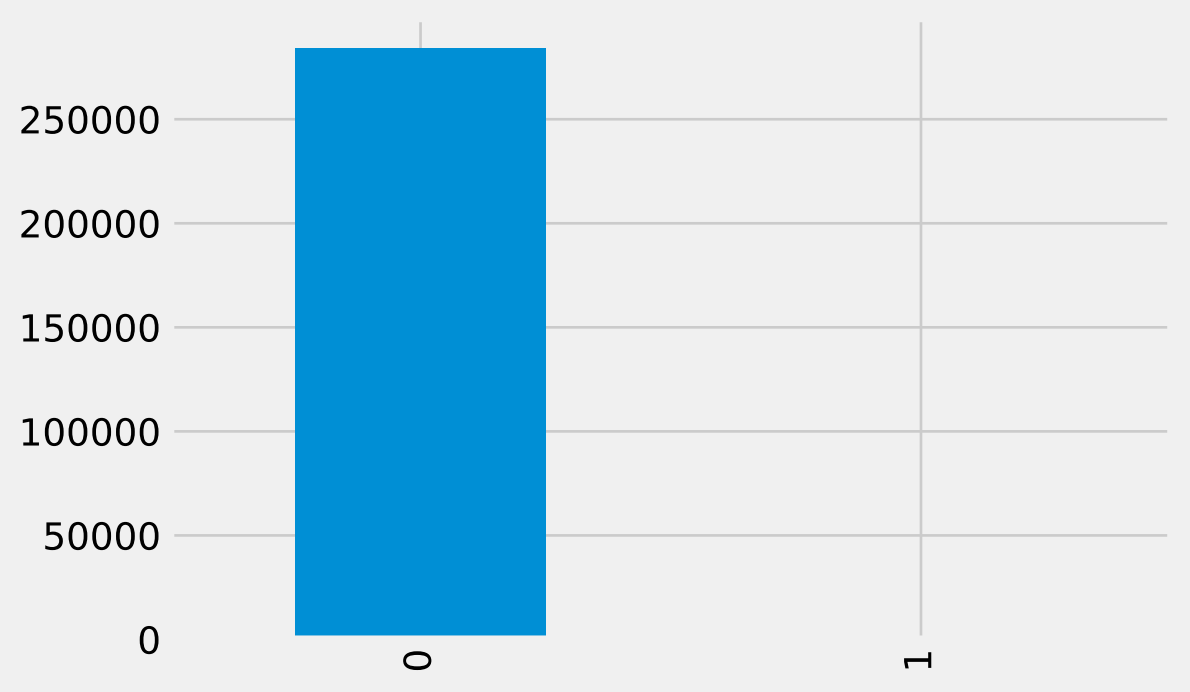
4 rows × 31 columns

Данный датасет является примером несбалансированных данных, так как мошеннические операции с картами встречаются реже обычных. Импортируйте библиотеку Pandas, а также используйте для графиков стиль “fivethirtyeight”. 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. Затем постройте такую же диаграмму, используя логарифмический масштаб.

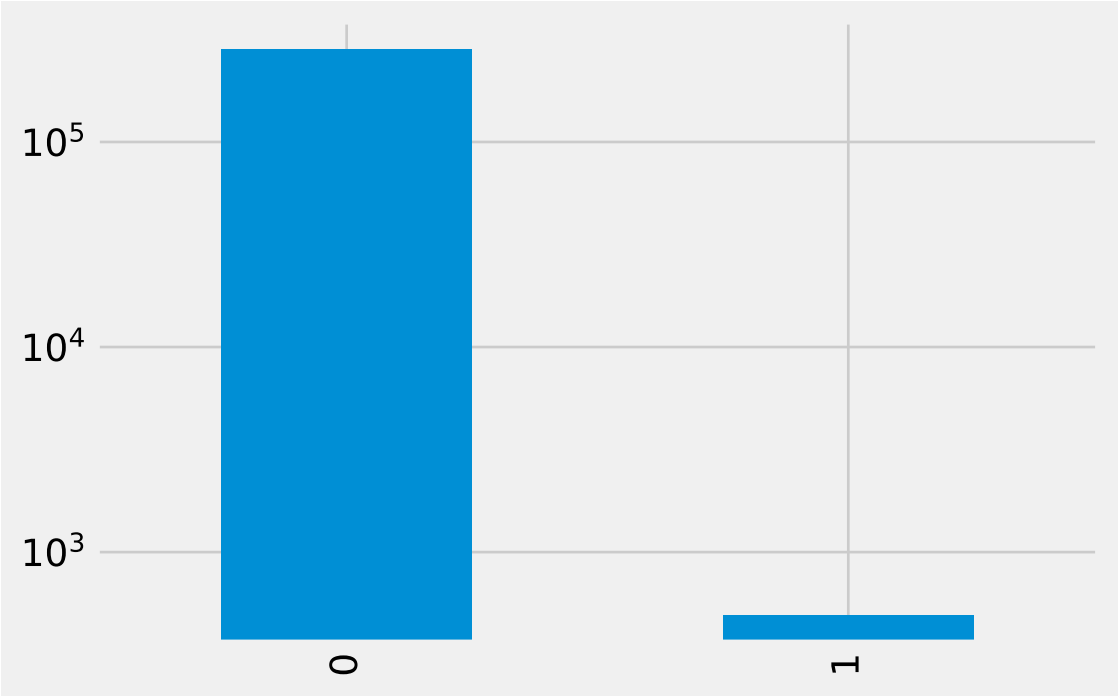
In [7]:
```
plt.style.use('fivethirtyeight')

#'Class' - это переменная ответа, и она принимает значение 1 в случае мошенничеств
t=df['Class'].value_counts()
t
```

Out[7]:
```
0    284315
1       492
Name: Class, dtype: int64
```

In [8]:
```
class_info = pd.Series(t)
class_info.plot(kind = 'bar')
plt.show()
```



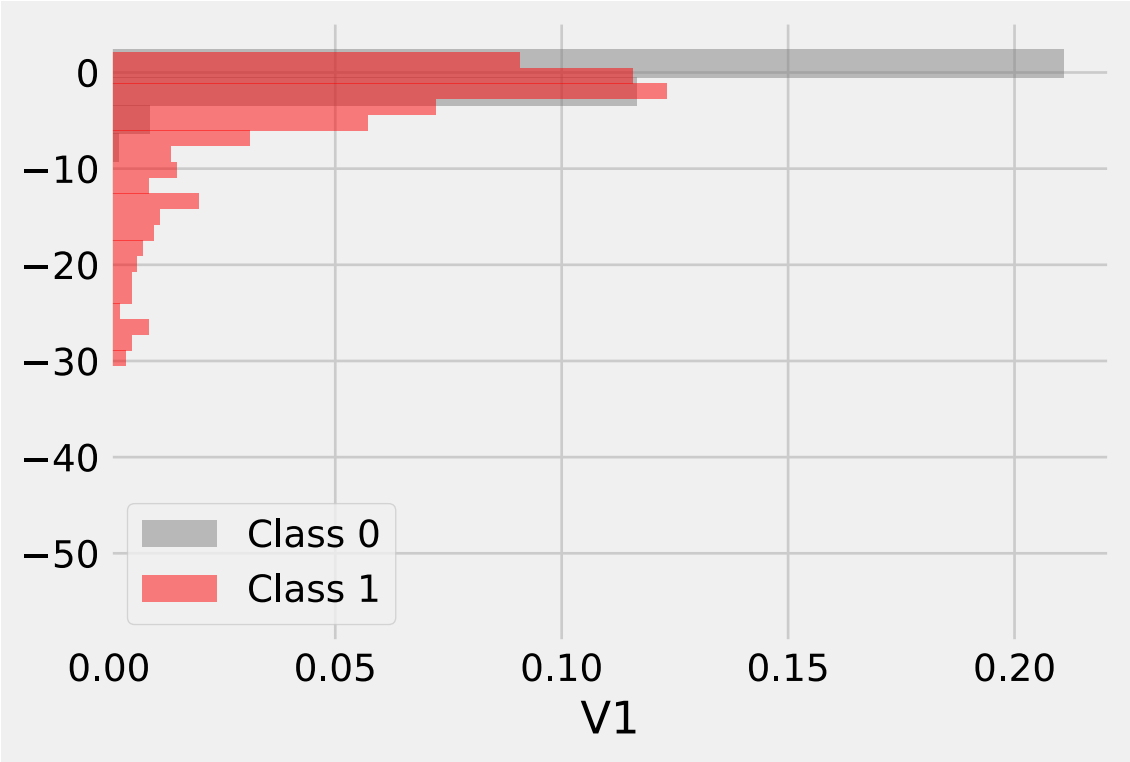
In [9]:
```
class_info = pd.Series(t)
class_info.plot(kind = 'bar', logy = True)
plt.show()
```



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

In [10]:
```
v1_class1=df.set_index('Class')['V1'].filter(like='1', axis=0)
v1_class1=v1_class1.reset_index()
v1_class1=v1_class1.drop('Class', axis=1)
v1_class1.head(), v1_class1.count()

v1_class0=df.set_index('Class')['V1'].filter(like='0', axis=0)
v1_class0=v1_class0.reset_index()
v1_class0=v1_class0.drop('Class', axis=1)
v1_class0.head(), v1_class0.count()

plt.hist(v1_class0['V1'], bins=20, color='grey', density = True, orientation = 'ho
plt.hist(v1_class1['V1'], bins=20, color='red', density = True, orientation = 'hor
plt.plot()
plt.xlabel('V1')
plt.legend(labels=['Class 0', 'Class 1'])
```

Out[10]:
```
<matplotlib.legend.Legend at 0x140aed39250>
```



In [ ]: