Task1

November 11, 2018

```
In [1]: import time
        import pandas as pd
        import numpy as np
```

Прочитаем файл с данными о выходе из строя серверов:

```
In [2]: data = pd.read_csv("6.csv.xls", header=None, names=["time"])
```

Посмотрим на первые несколько ячеек:

```
In [3]: data.head()

Out[3]:             time
        0  lambda = 88
        1    t_0 = 300
        2    t = 90000
        3      58.3458
        4     117.1273
```

Как можно увидеть в первых трех заданы значения величин λ (или $\frac{1}{\lambda}$), t_0 и t. Запишем их в соответствующие переменные и оставим в нашей таблице только времена выхода из строя серверов:

```
In [4]: lambd = 88
        t_0 = 300
        t = 90000
        data = data[3:]

In [5]: data.head()

Out[5]:        time
        3   58.3458
        4  117.1273
        5  303.7976
        6  481.9694
        7  496.6469
```

Для удобства сделаем из этих данных массив:

```
In [6]: values = np.array(data.values, dtype=float)
```

Определим какой же нам на вход дали параметр: λ или $\frac{1}{\lambda}$. Мы знаем, что времена между i-ым и i+1-ым моментами выхода из строя серверов являются независимыми в совокупности и имеют Экспоненциальное распределение с параметром λ. Таким образом мы можем считать, что данные случайные величины - это выборка из Экспоненциального распределения. Так как $E\overline{X} = EX_1 = \frac{1}{\lambda}$, то посчитав выборочное среднее можно определить параметр λ. Для этого сдвинем наши данные на 1 вправо и от исходных данных отнимем данные сдвинутые на один, таким образом как раз получим время между выходами серверов из строя. После этого возьмем их среднее.

```
In [7]: np.mean(values[1:] - np.roll(values, 1)[1:])
```

```
Out[7]: 86.6530113113113
```

Как можно заметить среднее значение приблизительно равно значению, которое нам дали в таблице, таким образом нам на вход дали не параметр λ, а $\frac{1}{\lambda}$. Заменим это в нашей переменной, а также для наглядности результатов уменьшим t_0 в 100 раз

```
In [8]: lambd = 1 / lambd
        t_0 = t_0 // 100
```

Теперь создадим функцию, которая будет каждые t_0 секунд выводить значение величины $E(N_t|N_{kt_0})$, где $k \in N$. Для начала посчитаем эту величину в явном виде:

$$E(N_t|N_{kt_0}) = E(N_t - N_{kt_0} + N_{kt_0}|N_{kt_0}) = E(N_t - N_{kt_0}|N_{kt_0}) +$$

$$+ E(N_{kt_0}|N_{kt_0}) = E(N_t - N_{kt_0}) + N_{kt_0} = \lambda(t - kt_0) + N_{kt_0}.$$

Здесь мы воспользовались линейностью условного математического ожидания, а также тем, что $N_t - N_{kt_0}$ не зависима с N_{kt_0} и $N_t - N_{kt_0}$ имеет Пуассоновское распределение с параметром $\lambda(t - kt_0)$.

В нашем случае, ограничим время работы 70 секундами. Можно также настроить этот параметр, передав в переменную stop время окончания программы.

```
In [24]: def count(stop=None):
             current_not_working = 0
             k = 0
             while (stop is None or k * t_0 <= stop) and current_not_working < len(values):
                 while current_not_working < len(values) \
                     and k * t_0 >= values[current_not_working]:
                     current_not_working += 1
                 print("E(N_t|N_{}) = ".format(k * t_0), lambd * (t - k * t_0)
                       + current_not_working)
                 time.sleep(t_0)
                 k += 1

         count(stop=70)
```

```
E(N_t|N_0) =  1022.7272727272727
E(N_t|N_3) =  1022.6931818181819
E(N_t|N_6) =  1022.659090909091
```

```
E(N_t|N_9) =  1022.625
E(N_t|N_12) =  1022.5909090909091
E(N_t|N_15) =  1022.5568181818182
E(N_t|N_18) =  1022.5227272727273
E(N_t|N_21) =  1022.4886363636364
E(N_t|N_24) =  1022.4545454545455
E(N_t|N_27) =  1022.4204545454546
E(N_t|N_30) =  1022.3863636363636
E(N_t|N_33) =  1022.3522727272727
E(N_t|N_36) =  1022.3181818181819
E(N_t|N_39) =  1022.284090909091
E(N_t|N_42) =  1022.25
E(N_t|N_45) =  1022.2159090909091
E(N_t|N_48) =  1022.1818181818182
E(N_t|N_51) =  1022.1477272727273
E(N_t|N_54) =  1022.1136363636364
E(N_t|N_57) =  1022.0795454545455
E(N_t|N_60) =  1023.0454545454546
E(N_t|N_63) =  1023.0113636363636
E(N_t|N_66) =  1022.9772727272727
E(N_t|N_69) =  1022.9431818181819
```

Посмотрим на несколько последних ячеек в нашей таблице:

```
In [136]: data.tail()
```

Out[136]:

	time
998	85857.0263
999	85964.9352
1000	86227.3704
1001	86424.2154
1002	86624.7041

Так как в нашем случае $t = 90000$, то действительно количество упавших серверов должно быть приблизительно немного больше 1000, что и предсказывает величина $E(N_t|N_s)$.

Вывод: с помощью условных матожиданий можно довольно точно предсказывать будущие события на основе текущих событий, в случае, если знать их распределение. Это следует из того, что по формуле полной вероятности: $E(E(N_t|N_s)) = EN_t$, поэтому наши предсказываемые данные не должны сильно отличаться от истинных.