Problem5.3

April 24, 2017

1 5. Условные математические ожидания и условные распределения II

```
In [2]: import warnings
        warnings.simplefilter('ignore')

        import numpy as np
        import pandas as pd
        from matplotlib import pyplot as plt
        import seaborn as sns

        %matplotlib inline
        from pylab import rcParams
        rcParams['figure.figsize'] = (15, 6)
        rcParams['image.cmap'] = 'viridis'
```

1.0.1 Решение

```
In [53]: data = pd.read_csv("../6.xls", header=None)
         data = np.array(data[3:], dtype=np.float32)
         data = data.reshape(len(data),)

In [38]: ## в файле содержаться времена
         ## выхода серверов из строя
         data[:5]

Out[38]: array([  6.8046999 ,  25.0515995 ,  29.82970047,  30.93790054,  32.83089828], dtype=flo
```

Вывод формулы $E(N_t|N_s)$: Воспользуемся линейностью условного матожидания:
$E(N_t|N_s) = E(N_t - N_s|N_s) + E(N_s|N_s) = E(N_t - N_s|N_s) + N_s$
Известно что * $(N_t - N_s) \sim Pois(\lambda(t-s))$; * $(N_t - N_s)$ независимо с N_s
Следовательно, $E(N_t|N_s) = E(N_t - N_s) + N_s$ и $E(N_t|N_s) = \lambda(t-s) + N_s$

```
In [52]: num_over(1000)

Out[52]: 63
```

1. Имеются серверы, которые периодически выходят из строя. Обозначим ξ_i время между i-м и $(i+1)$-м моментами выхода сервера из строя. Предполагается, что величины ξ_i независимы в совокупности и имеют экспоненциальное распределение с параметром λ.

   Обозначим N_t — количество серверов, которые вышли из строя к моменту времени t (в начальный момент времени $N_0 = 0$). В курсе случайных процессов будет доказано, что для любых $s < t$ величина $N_t - N_s \sim Pois(\lambda(t-s))$ и независима с N_s. При этом N_t как функция от t будет называться пуассоновским процессом интенсивности λ.

   Необходимо узнать, сколько серверов нужно докупить к моменту времени t взамен вышедших из строя. В момент времени s предсказанием количества серверов, вышедших из строя к моменту времени t, будем считать величину $\mathsf{E}(N_t|N_s)$. Напишите программу, которая с момента запуска через каждые t_0 секунд (можно брать $t_0/100$, чтобы не спать перед компьютером) будет выводить уточненное значение предсказания, т.е. $\mathsf{E}(N_t|N_{kt_0})$ для $k \in \mathbb{N}$. В текстовых полях jupyter-ноутбука напишите явно вывод формулы для $\mathsf{E}(N_t|N_s)$.

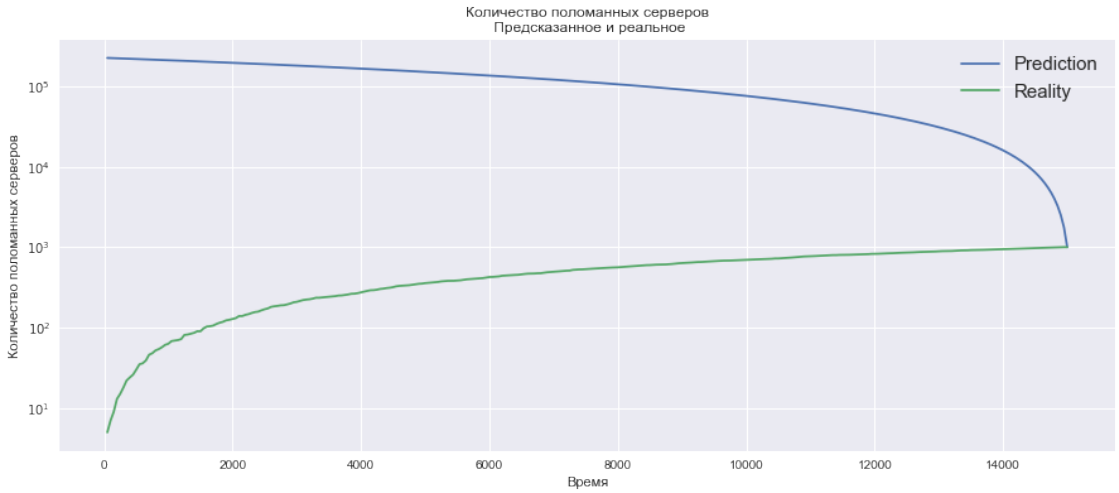
   В файле 6.csv содержатся сообщения о выходе из строя серверов. По этим данным напишите программу, которая каждые t_0 секунд выдает значение предсказания. Значения параметров t_0, t и λ также находятся в приложенном файле.

Задание

In [66]:
```
## взято из файла
lambda_ = 15
t_0 = 50
t = 15000
```

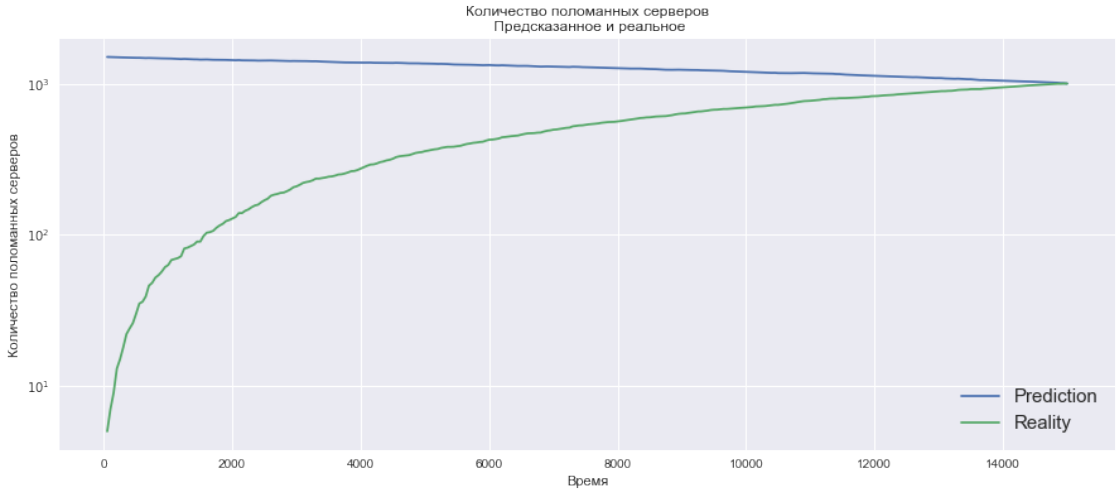
In [70]:
```
def plot_predictions(lambda_, t_0, t):
    def num_over(t_):
        return np.sum(data<=t_)

    prediction = []
    time = []
    game_over = []  # Вышло из строя на данный момент согласно файлику

    for i in range(t//t_0):
        time.append(t_0*(i+1))
        game_over.append(num_over(time[-1]))
        prediction.append(((t-time[-1])*lambda_) + game_over[-1])

    plt.plot(time, prediction, label='Prediction')
    plt.plot(time, game_over, label='Reality')
    plt.semilogy()
    plt.title("Количество поломанных серверов\n Предсказанное и реальное")
    plt.ylabel("Количество поломанных серверов")
    plt.xlabel("Время")
    plt.legend(fontsize=15)
    plt.show()
    return prediction
```

In [72]:
```
pred_15 = plot_predictions(lambda_, t_0, t)
```



Если взять lambda_ поменьше, то сходится получше и побыстрее. И значения ближе к истине.

```
In [74]: pred_1 = plot_predictions(0.1, t_0, t)
```




```
In [76]: # Предсказания:
         pred_15

Out[76]: [224255,
          223507,
          222759,
          222013,
          221265,
          220518,
```

219772,
219024,
218276,
217530,
216785,
216036,
215289,
214546,
213798,
213052,
212304,
211557,
210811,
210063,
209318,
208569,
207820,
207072,
206331,
205582,
204834,
204086,
203340,
202590,
201848,
201103,
200354,
199606,
198861,
198115,
197368,
196623,
195875,
195128,
194381,
193639,
192889,
192144,
191397,
190652,
189906,
189158,
188414,
187669,
186923,
186181,
185434,
184686,

183939,
183190,
182444,
181699,
180956,
180209,
179464,
178720,
177973,
177225,
176479,
175735,
174985,
174237,
173489,
172742,
171993,
171246,
170500,
169751,
169004,
168258,
167513,
166764,
166018,
165274,
164530,
163786,
163041,
162292,
161546,
160802,
160055,
159310,
158563,
157818,
157076,
156330,
155582,
154834,
154086,
153341,
152597,
151850,
151102,
150357,
149610,
148864,

148117,
147369,
146625,
145878,
145131,
144381,
143632,
142885,
142137,
141393,
140648,
139901,
139155,
138407,
137660,
136912,
136169,
135425,
134676,
133929,
133183,
132441,
131693,
130946,
130199,
129451,
128703,
127959,
127214,
126468,
125719,
124970,
124223,
123474,
122731,
121987,
121240,
120495,
119747,
119001,
118255,
117509,
116761,
116021,
115275,
114528,
113779,
113033,

112287,
111539,
110792,
110045,
109300,
108553,
107806,
107057,
106308,
105561,
104815,
104069,
103323,
102576,
101832,
101086,
100341,
99593,
98847,
98098,
97352,
96605,
95856,
95107,
94359,
93613,
92868,
92125,
91380,
90633,
89884,
89139,
88394,
87649,
86903,
86154,
85408,
84662,
83918,
83171,
82422,
81676,
80929,
80179,
79431,
78684,
77937,
77189,

76441,
75694,
74947,
74200,
73455,
72706,
71958,
71210,
70464,
69717,
68973,
68223,
67477,
66731,
65986,
65241,
64497,
63754,
63009,
62265,
61517,
60769,
60023,
59277,
58530,
57787,
57039,
56294,
55546,
54796,
54050,
53300,
52551,
51803,
51055,
50308,
49559,
48813,
48066,
47318,
46574,
45825,
45077,
44331,
43585,
42837,
42089,
41342,

```
40597,
39850,
39102,
38355,
37609,
36861,
36117,
35370,
34622,
33875,
33127,
32379,
31632,
30888,
30141,
29391,
28644,
27895,
27150,
26407,
25660,
24911,
24163,
23418,
22669,
21919,
21169,
20425,
19678,
18933,
18184,
17437,
16691,
15945,
15198,
14450,
13702,
12955,
12210,
11464,
10718,
9971,
9224,
8477,
7730,
6983,
6234,
5490,
```

```
4744,
3997,
3247,
2497,
1750,
1000]
```