

Mój kod

```
In [1]: 1 import csv
        2 import numpy as np
        3 import pandas as pd
        4 import matplotlib.pyplot as plt
        5 from sklearn import linear_model, preprocessing
        6 from sklearn.pipeline import Pipeline
        7 from sklearn.preprocessing import PolynomialFeatures
        8 from sklearn.linear_model import LinearRegression
        9 import time
```

```
In [2]: 1 data = pd.read_csv("outfile.csv")
        2 data.head(10)
```

	X_Val	Y_Val
0	180	99.074074
1	184	99.074074
2	188	99.074074
3	192	99.074074
4	196	99.074074
5	200	99.074074
6	204	99.074074
7	208	99.074074
8	212	99.074074
9	216	99.074074

```
In [3]: 1 X_Val = data.iloc[:, 0]
        2 Y_Val = data.iloc[:, 1]
        3 X_Val = X_Val.as_matrix()
        4 Y_Val = Y_Val.as_matrix()
```

E:\Users\sticz\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: FutureWarning: Method .as_matrix will be version. Use .values instead.

This is separate from the ipykernel package so we can avoid doing imports until

E:\Users\sticz\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: Method .as_matrix will be version. Use .values instead.

after removing the cwd from sys.path.

```
In [4]: 1 start_time = time.time()
2
3 X_Val = data.iloc[:, 0]
4 X_Val = X_Val.as_matrix()
5 model = Pipeline([('poly', PolynomialFeatures(degree=18)),
6                  ('linear', LinearRegression(fit_intercept=True, normalize=True))])
7 model = model.fit(X_Val.reshape(-1, 1), Y_Val.reshape(-1, 1))
8
9 elapsed_time_sk = time.time() - start_time
10
11 elapsed_time_sk

E:\Users\sticz\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: Method .as_matrix will be
version. Use .values instead.
    after removing the cwd from sys.path.

0.013889551162719727
```

```
In [5]: 1 model.named_steps['linear'].coef_ #Współczynniki wielomianu

array([[ 0.00000000e+00, -2.07438708e+01,  1.41300383e-01,
        -5.39264394e-04,  1.31448642e-06, -2.20692276e-09,
         2.67830261e-12, -2.42283563e-15,  1.66529922e-18,
        -8.79638579e-22,  3.59110429e-25, -1.13392794e-28,
         2.75710114e-32, -5.10696434e-36,  7.06724590e-40,
        -7.06828626e-44,  4.82254212e-48, -2.00728286e-52,
         3.84342654e-57]])
```

```
In [6]: 1 Xz = np.arange(1, 5000)
2 Predykcje = model.predict(Xz.reshape(-1, 1))
3 Predykcje

array([[1348.24411015],
       [1327.92038527],
       [1307.87285531],
       ...,
       [ 126.5464373 ],
       [ 126.70464042],
       [ 126.86382011]])
```

```
In [7]: 1 Predykcje_df = pd.DataFrame(Predykcje)
        2 Predykcje_df.head(50)
```

	0
0	1348.244110
1	1327.920385
2	1307.872855
3	1288.098363
4	1268.593781
5	1249.356013
6	1230.381993
7	1211.668686
8	1193.213084
9	1175.012211
10	1157.063120
11	1139.362891
12	1121.908635
13	1104.697491
14	1087.726625
15	1070.993233
16	1054.494537
17	1038.227787
18	1022.190261
19	1006.379264
20	990.792127
21	975.426209
22	960.278893
23	945.347591
24	930.629740
25	916.122801
26	901.824262
27	887.731637
28	873.842464
29	860.154306
30	846.664751
31	833.371410
32	820.271920
33	807.363942
34	794.645158
35	782.113279
36	769.766033
37	757.601175
38	745.616484
39	733.800757

```
In [8]: 1 Xz = pd.DataFrame(Xz)
        2 Xz
```

	0
0	1
1	2
2	3
3	4
4	5
...	...
4994	4995
4995	4996
4996	4997
4997	4998
4998	4999

4999 rows × 1 columns

```
In [9]: 1 #model.score(Predykcje, Y_Val, sample_weight=None)
```

```
In [10]: 1 result = pd.concat([Xz, Predykcje_df], axis=1, join='inner')
        2
        3 result.to_csv('porazka.csv', index=False, header=False)
```

```
In [11]: 1 result
```

	0	0
0	1	1348.244110
1	2	1327.920385
2	3	1307.872855
3	4	1288.098363
4	5	1268.593781
...
4994	4995	126.230031
4995	4996	126.386281
4996	4997	126.546437
4997	4998	126.704640
4998	4999	126.863820

4999 rows × 2 columns

Własna implementacja

```
In [12]: 1 X = np.arange(1, 5000)
```

```
In [13]: 1 Wielomianki = np.polyfit(X_Val,Y_Val,18)
          2 Wielomianki #wspolczynniki wielomianow

E:\Users\sticz\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: RankWarning: Polyfit may be poorly condi
      """Entry point for launching an IPython kernel.

array([ 4.40560841e-58, -1.93714767e-53,  3.72364500e-49, -4.01317313e-45,
        2.49337414e-41, -6.58703646e-38, -2.76996220e-34,  3.74065277e-30,
       -1.97467120e-26,  6.55903341e-23, -1.49134491e-19,  2.35315966e-16,
       -2.52395554e-13,  1.73937158e-10, -6.76156722e-08,  8.81668753e-06,
        2.83949183e-03, -1.04141348e+00,  1.88806722e+02])
```

```
In [14]: 1 def Mein(x):
          2     power = 0 #potegi
          3     suma = 0
          4     for w in reversed(Wielomianki):
          5         if(power ==0):
          6             suma += w
          7         else:
          8             res = w*x**power
          9             suma +=res
         10     power +=1
         11     return suma
```

```
In [15]: 1 Mein(4000)

168.22003739327192
```

```
In [16]: 1 def do_listy(X):
          2     lista = []
          3     for a in range(X.size):
          4         wyniczek = Mein(a)
          5         lista.append(wyniczek)
          6     return lista
```

```
In [17]: 1 start_time = time.time()
          2 lista = do_listy(X)
          3 elapsed_time_moje = time.time() - start_time
          4
          5 elapsed_time_moje

0.4627692699432373
```

```
In [18]: 1 moje_wyniki_df = pd.DataFrame(lista)
```

```
In [19]: 1 | moje_wyniki_df
```

	0
0	188.806722
1	187.768157
2	186.735323
3	185.708270
4	184.687047
...	...
4994	124.418402
4995	124.570158
4996	124.721839
4997	124.873432
4998	125.024772

4999 rows × 1 columns

```
In [20]: 1 | moj_result = pd.concat([Xz, moje_wyniki_df], axis=1, join='inner')
2 | moj_result
3 | moj_result.to_csv('mojaporazka.csv', index=False, header=False)
```

```
In [21]: 1 | moj_result
```

	0	0
0	1	188.806722
1	2	187.768157
2	3	186.735323
3	4	185.708270
4	5	184.687047
...
4994	4995	124.418402
4995	4996	124.570158
4996	4997	124.721839
4997	4998	124.873432
4998	4999	125.024772

4999 rows × 2 columns

Viva la fiesta i espania

```
In [22]: 1 import csv
          2 import numpy as np
          3 import pandas as pd
          4 import matplotlib.pyplot as plt
          5 from sklearn import linear_model, preprocessing
          6 from sklearn.pipeline import Pipeline
          7 from sklearn.preprocessing import PolynomialFeatures
          8 from sklearn.linear_model import LinearRegression
```

```
In [23]: 1 data = pd.read_csv("ElectionData.csv")
          2 data.head(10)
```

	TimeElapsed	time	territoryName	totalMandates	availableMandates	numParishes	numParishesAp
0	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081
1	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081
2	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081
3	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081
4	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081
5	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081
6	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081
7	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081
8	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081
9	0	2019-10-06 20:10:02	Território Nacional	0	226	3092	1081

10 rows × 28 columns

```
In [24]: 1 data_teritorio = data.query('territoryName == "Território Nacional"')
          2 data_teritorio = data_teritorio.loc[:, ["TimeElapsed", "Party", "Mandates", "FinalM
          3 data_teritorio
```

		TimeElapsed	Party	Mandates	FinalMandates
0	0		PS	0	106
1	0		PPD/PSD	0	77
2	0		B.E.	0	19
3	0		CDS-PP	0	5
4	0		PCP-PEV	0	12
...
21256	265		PURP	0	0
21257	265		PDR	0	0
21258	265		PPM	0	0
21259	265		PTP	0	0
21260	265		MAS	0	0

1134 rows × 4 columns

```
In [25]: 1 data_teritorio.Party.unique()

array(['PS', 'PPD/PSD', 'B.E.', 'CDS-PP', 'PCP-PEV', 'PAN', 'CH',
       'R.I.R.', 'PCTP/MRPP', 'A', 'L', 'IL', 'JPP', 'NC', 'PDR', 'PNR',
       'PURP', 'PPM', 'MPT', 'PTP', 'MAS'], dtype=object)
```



```
In [26]: 1 PS = data_teritorio.query('Party == "PS"')
          2 PPD_PSD = data_teritorio.query('Party == "PPD/PSD"')
          3 BE = data_teritorio.query('Party == "B.E."')
          4 CDS_PP = data_teritorio.query('Party == "CDS-PP"')
          5 PCP_PEV = data_teritorio.query('Party == "PCP-PEV"')
          6 PAN = data_teritorio.query('Party == "PAN"')
          7 CH = data_teritorio.query('Party == "CH"')
          8 RIR = data_teritorio.query('Party == "R.I.R."')
          9 PCTP_MRPP = data_teritorio.query('Party == "PCTP/MRPP"')
         10 A = data_teritorio.query('Party == "A"')
         11 L = data_teritorio.query('Party == "L"')
         12 IL = data_teritorio.query('Party == "IL"')
         13 JPP = data_teritorio.query('Party == "JPP"')
         14 NC = data_teritorio.query('Party == "NC"')
         15 PDR = data_teritorio.query('Party == "PDR"')
         16 PNR = data_teritorio.query('Party == "PNR"')
         17 PURP = data_teritorio.query('Party == "PURP"')
         18 PPM = data_teritorio.query('Party == "PPM"')
         19 MPT = data_teritorio.query('Party == "MPT"')
         20 PTP = data_teritorio.query('Party == "PTP"')
         21 MAS = data_teritorio.query('Party == "MAS"')

In [27]: 1 lista_partii = [PS, PPD_PSD, BE, CDS_PP, PCP_PEV, PAN, CH, RIR,
          2                  PCTP_MRPP, A, L, IL, JPP, NC, PDR, PNR, PURP, PPM, MPT, PTP, MAS]
```

```
In [28]: 1 #lista_partii[19]
        2 PS
```

	TimeElapsed	Party	Mandates	FinalMandates
0	0	PS	0	106
386	5	PS	0	106
772	10	PS	0	106
1158	15	PS	0	106
1544	20	PS	0	106
1930	25	PS	0	106
2316	30	PS	0	106
2702	35	PS	0	106
3105	40	PS	6	106
3508	45	PS	8	106
3911	50	PS	9	106
4314	55	PS	14	106
4717	60	PS	16	106
5120	65	PS	16	106
5523	70	PS	21	106
5926	75	PS	21	106
6329	80	PS	22	106
6732	85	PS	29	106
7135	90	PS	34	106
7538	95	PS	36	106
7941	100	PS	39	106
8344	105	PS	39	106
8747	110	PS	47	106
9150	115	PS	49	106
9553	120	PS	53	106
9956	125	PS	58	106
10359	130	PS	65	106
10762	135	PS	67	106
11165	140	PS	68	106
11568	145	PS	73	106
11971	150	PS	75	106
12374	155	PS	80	106
12777	160	PS	84	106
13180	165	PS	86	106
13583	170	PS	87	106
13986	175	PS	87	106
14389	180	PS	90	106
14792	185	PS	91	106
15195	190	PS	92	106
15598	195	PS	92	106

In []:

1

```
In [29]: 1 def partie_funkcja():
2     lista_partii = [PS, PPD_PSD, BE, CDS_PP, PCP_PEV, PAN, CH, RIR,
3                     PCTP_MRPP, A, L, IL, JPP, NC, PDR, PNR, PURP, PPM, MPT, PTP, MAS]
4     lista_wspolczynn timer = []
5     lista_predykcji = []
6     lista_X_Val = []
7     lista_Y_Val = []
8     lista_results = []
9
10    for a in range (0,21):
11        print(a)
12        time = lista_partii[a].iloc[:,0]
13        X_Val = time.as_matrix()
14
15        mandates = lista_partii[a].iloc[:, 2]
16        Y_Val = mandates.as_matrix()
17
18        model = Pipeline([('poly', PolynomialFeatures(degree=18)),
19                          ('linear', LinearRegression(fit_intercept=True, normalize=
20 model = model.fit(X_Val.reshape(-1, 1), Y_Val.reshape(-1, 1))
21
22
23        wspolczynn timer = model.named_steps['linear'].coef_
24        predykcje = model.predict(X_Val.reshape(-1, 1))
25
26        lista_wspolczynn timer.append(wspolczynn timer)
27        lista_Y_Val.append(Y_Val)
28
29
30        predykcje_df = pd.DataFrame(predykcje).astype(int)
31        lista_predykcji.append(predykcje)
32
33        X_Val_pd = pd.DataFrame(X_Val)
34        lista_X_Val.append(X_Val_pd)
35
36
37        lista_Y_Val.append(Y_Val)
38        Y_Val_df = pd.DataFrame(Y_Val)
39
40        result = pd.concat([X_Val_pd, predykcje_df, Y_Val_df], axis=1, join='inne
41        lista_results.append(result)
42
43        a+=1
44    ..
```

```
In [30]: 1 lista_wspolczynnikow, lista_predykcji, lista_X_Val, lista_Y_Val, lista_results = par  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20
```

E:\Users\sticz\Anaconda3\lib\site-packages\ipykernel_launcher.py:13: FutureWarning: Method .as_matrix will b

```
In [31]: 1 lista_results[2] #time, mandaty predykcja, mandaty rzeczywistosc
```

```
0 0 0
```

```
0 0 0 0
```

```
1 5 0 0
```

```
2 10 0 0
```

```
3 15 0 0
```

```
4 20 0 0
```

```
5 25 0 0
```

```
6 30 0 0
```

```
7 35 0 0
```

```
8 40 0 0
```

```
9 45 0 0
```

```
10 50 0 0
```

```
11 55 0 0
```

```
12 60 0 0
```

```
13 65 0 0
```

```
14 70 0 0
```

```
15 75 0 0
```

```
16 80 0 0
```

```
17 85 1 1
```

```
18 90 1 2
```

```
19 95 1 2
```

```
20 100 1 2
```

```
21 105 2 2
```

```
22 110 2 2
```

```
23 115 2 2
```

```
24 120 3 3
```

```
25 125 4 6
```

```
26 130 5 6
```

```
27 135 6 7
```

```
28 140 7 7
```

```
29 145 9 8
```

```
30 150 10 10
```

```
31 155 11 12
```

```
32 160 12 14
```

```
33 165 13 14
```

```
34 170 14 14
```

```
35 175 14 14
```

```
36 180 14 15
```

```
37 185 14 15
```

```
38 190 14 15
```

```
39 195 14 15
```

```
40 200 15 15
```

```
In [32]: 1 #najwieksze partie - predykcje mandatow
          2 predykcja_partia_PS = lista_results[0].iloc[:, 1].values.tolist()
          3 predykcja_partia_PPD = lista_results[1].iloc[:, 1].values.tolist()
          4 predykcja_partia_BE = lista_results[2].iloc[:, 1].values.tolist()
          5 predykcja_partia_PS

          [0,
           0,
           0,
           0,
           0,
           0,
           0,
           1,
           4,
           7,
           10,
           13,
           15,
           17,
           19,
           21,
           24,
           28,
           31,
           35,
           38,
           42,
           45,
           49,
           53,
           57,
           62,
           66,
           70,
           73,
           76,
           79,
           82,
           84,
           86,
           88,
           90,
           90,
           91,
           92,
           92,
           93,
           95,
           97,
           98,
           99,
           99,
           99,
           99,
           100,
           100,
           101,
           103,
           105]
```

```
In [33]: 1 def binary_search(item_list,mid):
2         first = item_list[0]
3         last  = item_list[-1]
4         szukany = last
5         print(item_list[mid])
6         if szukany - item_list[mid] <= 30 :
7             print('got it')
8             return item_list[mid],mid
9         else:
10            if szukany < item_list[mid]:
11                mid = mid - 1
12                return binary_search(item_list,mid)
13
14            else:
15                mid = mid + 1
16                return binary_search(item_list,mid)
17
```

```
In [34]: 1 mid = int(len(predykcja_partia_PS)/2)
2         a, b = binary_search(predykcja_partia_PS,mid)
3         wynik_PS = lista_results[0].iloc[b, :]
4
5         mid = int(len(predykcja_partia_PPD)/2)
6         c,d  = binary_search(predykcja_partia_PPD,mid)
7         wynik_PPD = lista_results[1].iloc[d, :]
8
9         mid = int(len(predykcja_partia_BE)/2)
10        d,e  = binary_search(predykcja_partia_BE,mid)
11        wynik_BE = lista_results[2].iloc[e, :]
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```
In [35]: 1 wynik_PS #timestamp - 150
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```



```
In [36]: 1 wynik_PPD #timestamp - 135

0 135
0 49
0 49
Name: 27, dtype: int64
```

```
In [37]: 1 wynik_BE #timestamp - 135, bo binary search szuka od srodka, a mandatow jest <

0 135
0 6
0 7
Name: 27, dtype: int64
```

Implementacja Sk-learn jest wielokrotnie szybsza, oraz wykazuje większą dokładność na tym zł

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 2
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

