Optimize the log-periodic power law (LPPL) parameters using Evolutionary Programming (EP) for predicting stock price crisis.

Dataset used : Daily data collected from National Stock exchange India from 2019-08-01 to 2020-05-14. Close price of N50 is used for evaluation.

Python programing language is used to Optimize the log-periodic power law (LPPL) parameters using Evolutionary programming for Predicting the stock price crisis. Libraries used: scipy’s optimize library,numpy,pandas,matplotlib

Methodology

EVOLUTIONARY PROGRAMMING(EP) introduced by in 1960 by Lawrence J. Fogel is a kind of strategy for stochastic optimization similar to Genetic Algorithm. Like any other evolutionary algorithms when other techniques such as gradient descent or direct, analytical discovery are not possible then it is useful. It has an underlying assumption that FITNESS space can be defined in terms of variable which can also characterize an optimum solution.

The basic 3 steps of EP are as follows

1. Randomly choose initial population of solutions. Although the number of solutions in the population is relevant in determining performance of algorithm there are no definite answers to it.

2. Using different mutation types ranging from minor to extreme the offsprings are mutated to produce new population solutions.

3. Inorder to assess the OFFSPRING solution we compute fitness. Top N performers are retained for the population of solutions. There is no need to maintain constant population size. Also no requirement that only a single offspring be generated from each Parent.

It should be noted that EP does not use any CROSSOVER operator.

Implementation works in multiple steps with different initial populaion set.

Class chromosomes

Though in evolutionary programming there is no constraint on the representation a class of chromosome is defined with fitness and parameters. Once the parameters are given we need to evaluate it.

evaluate

This function evaluate the fitness of each solution. To compute fitness use LPPL fitness function **a + b*tm + c*tm*np.cos(w*np.log(tc-t)-phi)** for the parameters of each individual solution. Then calulate the Mean sqaured errors of estimated values.

generate_ep_parameters

It starts with generating parameter set of a,b,tc,m,c,w,phi,start .

tc:=critical time (date of termination of the bubble and transition in a new regime a:=expected log price at the peak when the end of the bubble is reached at tc

b:=amplitude of the power law acceleration

c:=amplitude of the log-periodic oscillations

m:=degree of the super exponential growth

ω:=scaling ratio of the temporal hierarchy of oscillations

phi:=time scale of the oscillations

and start indicating the start time of each step. The lower bound and upper bound is defined for all parameters.

execute

This is the function to execute evolution. Initially when there is no population generate new chromosomes and evaluate it. Otherwise revaluate the chromsomes in the existing population. With the generated/revaluated population execute evolve function containing breed,mutation,elimination

breed

It generate fixed number of chromosomes

mutate

Individuals can mutate, i.e. alter their properties a bit. This is a good way to explore more possible solutions in different places and don’t go the wrong way deterministically.Based on mutate_ratio decide the number of individuls to undergo mutation. Decide a mutation point and then replace that point by a new value. Return the new evaluated individual.

eliminate

This function is used to eliminate useless individuals by checking if each parameter is within defined range or having invalid fitness. Remove duplicate individuals and then eliminate least performers based on eliminate_ratio

Based on the number of generations defined we repeat the process of breeding,mutation and elimination.

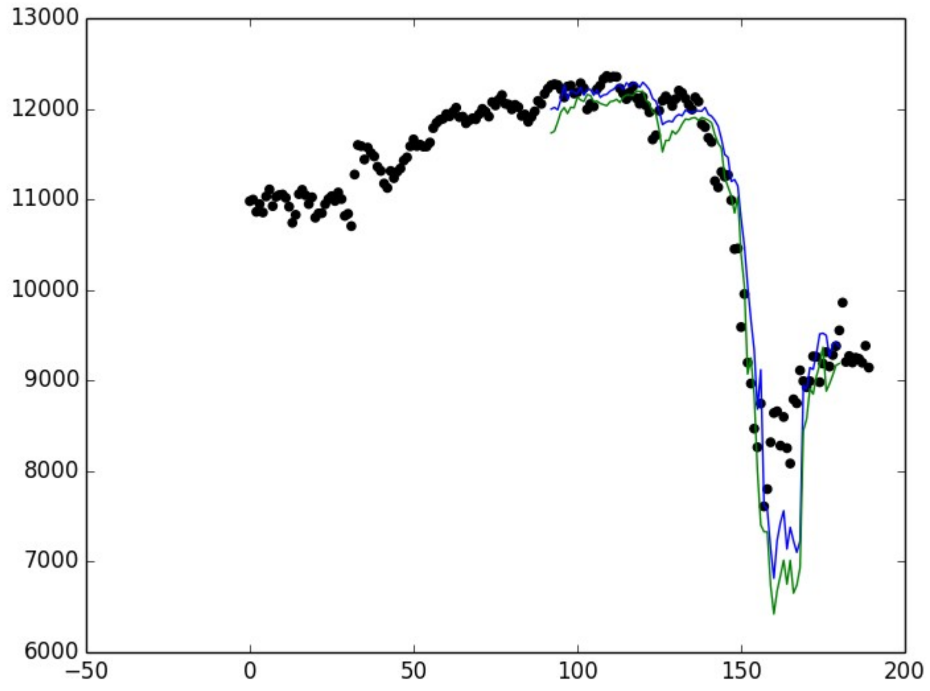
All the above given iterations constitute an iteration. After every iteration top 10 best solutions are added to result. With this final parameter set estimate the values by LPPL fitness function for a defined range of time. This is the prediction part. Record the critical time obtained in each solution set and their count to draw critical time distribution. For every point of time we considered for prediction plot the best guess upper most and lower most values.

The algorithm is executed for 90 generations with a pool size of 250

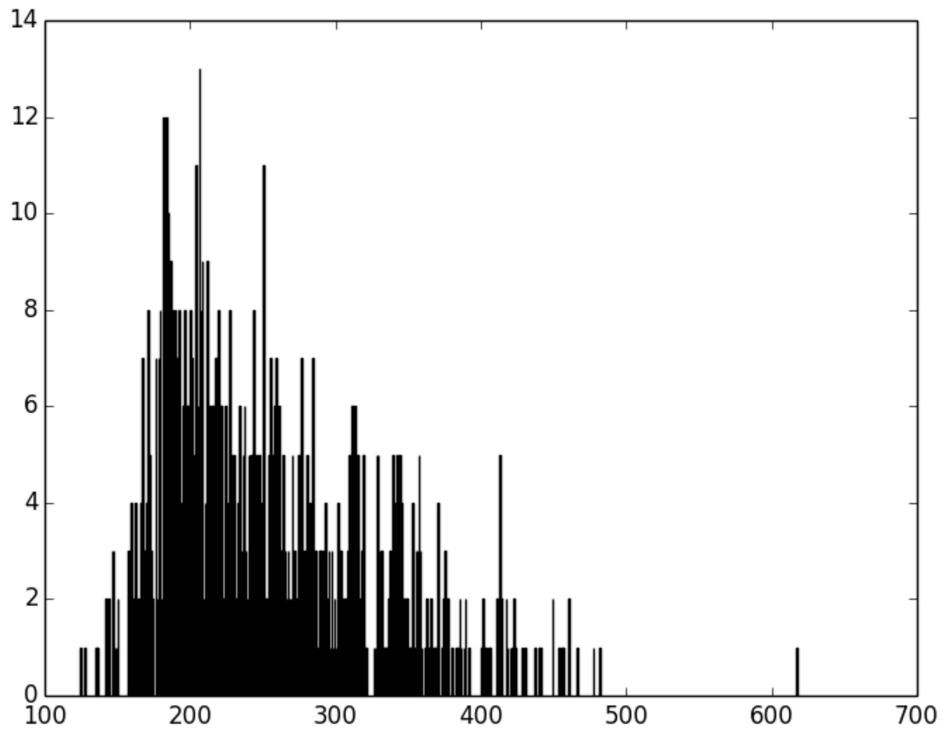
Results

LPPL fit to stock price

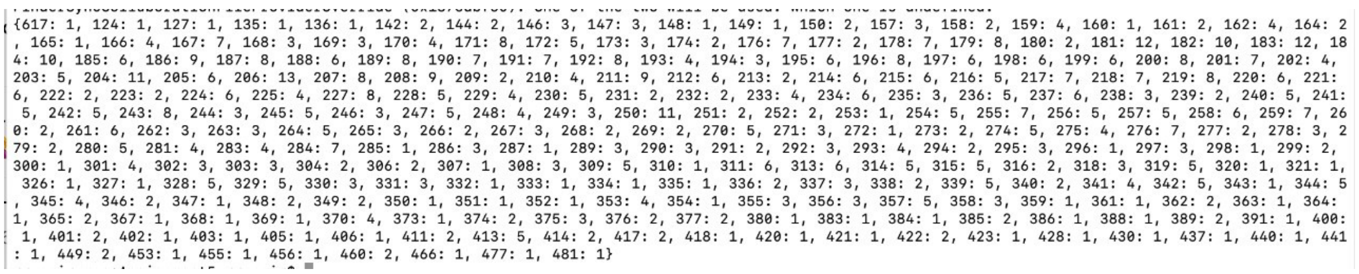
Plotting the best guess and worst guess for each point of time



critical time distribution



Critical time points



```
[illegible] One of the two will be used. Which one is undefined.
{617: 1, 124: 1, 127: 1, 135: 1, 136: 1, 142: 2, 144: 2, 146: 3, 147: 3, 148: 1, 149: 1, 150: 2, 157: 3, 158: 2, 159: 4, 160: 1, 161: 2, 162: 4, 164: 2
, 165: 1, 166: 4, 167: 7, 168: 3, 169: 3, 170: 4, 171: 8, 172: 5, 173: 3, 174: 2, 176: 7, 177: 2, 178: 7, 179: 8, 180: 2, 181: 12, 182: 10, 183: 12, 18
4: 10, 185: 6, 186: 9, 187: 8, 188: 6, 189: 8, 190: 7, 191: 7, 192: 8, 193: 4, 194: 3, 195: 6, 196: 8, 197: 6, 198: 6, 199: 6, 200: 8, 201: 7, 202: 4,
203: 5, 204: 11, 205: 6, 206: 13, 207: 8, 208: 9, 209: 2, 210: 4, 211: 9, 212: 6, 213: 2, 214: 6, 215: 6, 216: 5, 217: 7, 218: 7, 219: 8, 220: 6, 221:
6, 222: 2, 223: 2, 224: 6, 225: 4, 227: 8, 228: 5, 229: 4, 230: 5, 231: 2, 232: 2, 233: 4, 234: 6, 235: 3, 236: 5, 237: 6, 238: 3, 239: 2, 240: 5, 241:
 5, 242: 5, 243: 8, 244: 3, 245: 5, 246: 3, 247: 5, 248: 4, 249: 3, 250: 11, 251: 2, 252: 2, 253: 1, 254: 5, 255: 7, 256: 5, 257: 5, 258: 6, 259: 7, 26
0: 2, 261: 6, 262: 3, 263: 3, 264: 5, 265: 3, 266: 2, 267: 3, 268: 2, 269: 2, 270: 5, 271: 3, 272: 1, 273: 2, 274: 5, 275: 4, 276: 7, 277: 2, 278: 3, 2
79: 2, 280: 5, 281: 4, 283: 4, 284: 7, 285: 1, 286: 3, 287: 1, 289: 3, 290: 3, 291: 2, 292: 3, 293: 4, 294: 2, 295: 3, 296: 1, 297: 3, 298: 1, 299: 2,
300: 1, 301: 4, 302: 3, 303: 3, 304: 2, 306: 2, 307: 1, 308: 3, 309: 5, 310: 1, 311: 6, 313: 6, 314: 5, 315: 5, 316: 2, 318: 3, 319: 5, 320: 1, 321: 1,
 326: 1, 327: 1, 328: 5, 329: 5, 330: 3, 331: 3, 332: 1, 333: 1, 334: 1, 335: 1, 336: 2, 337: 3, 338: 2, 339: 5, 340: 2, 341: 4, 342: 5, 343: 1, 344: 5
, 345: 4, 346: 2, 347: 1, 348: 2, 349: 2, 350: 1, 351: 1, 352: 1, 353: 4, 354: 1, 355: 3, 356: 3, 357: 5, 358: 3, 359: 1, 361: 1, 362: 2, 363: 1, 364:
1, 365: 2, 367: 1, 368: 1, 369: 1, 370: 4, 373: 1, 374: 2, 375: 3, 376: 2, 377: 2, 380: 1, 383: 1, 384: 1, 385: 2, 386: 1, 388: 1, 389: 2, 391: 1, 400:
 1, 401: 2, 402: 1, 403: 1, 405: 1, 406: 1, 411: 2, 413: 5, 414: 2, 417: 2, 418: 1, 420: 1, 421: 1, 422: 2, 423: 1, 428: 1, 430: 1, 437: 1, 440: 1, 441
: 1, 449: 2, 453: 1, 455: 1, 456: 1, 460: 2, 466: 1, 477: 1, 481: 1}
[illegible]
```