Genetic Algorithms Project Report : Team 94

Jayant Panwar

2019114013

jayant.panwar@research.iiit.ac.in

Mayank Jain

2019101023

mayank.j@students.iiit.ac.in

GENETIC ALGORITHM SUMMARY

A **genetic algorithm** is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

Initial population was generated by mutation on an overfit vector and then the following steps were repeated.

Step 1 -Calculate fitness of the *population* and store in *popfit*

Step 2 -Find parent vectors on the basis of best fit score.

Step 3 -Using Crossover to generate children from selected parents and storing them in *offsprings*

Step 4 -Using mutation on *offsprings*

Step 5 -Computing fitness of the *offsprings*

Step 6 -Generating New *population* from selection of *offsprings* and initial *population* on the basis of best fit score.

Step 7 -Using new *population* as initial *population* for next generation and repeating step1 to step 6

After stopping the process, finding the best fit vector from the new population and submitting it .

1. Diagrams

Diagram 1

```
--- Initial Population ----
[0.0, -1.1972414993746475e-12, -2.406881067777487e-13, 7.649196433583433e-11,
-2.3918982867837016e-10, -1.7246909729539205e-15, 7.911359737375438e-16,
1.8102734393643636e-05, -1.1574908230378795e-06, -6.57609392635641e-09,
4.756957975705682e-10]

[0.0, -1.035253796035754e-12, -2.631661639506733e-13, 6.246202803747581e-11,
-2.1087343071763726e-10, -1.6415897057292425e-15, 6.749951533371433e-16,
1.7180667863700074e-05, -1.0782104898850587e-06, -7.698121826771682e-09,
4.984275020343377e-10]

[0.0, -1.079535153452445e-12, -2.104688073163699e-13, 6.43872047842357e-11,
-2.0661451579880152e-10, -1.4872224263480214e-15, 8.364617410025721e-16,
2.064932775606508e-05, -1.263889516424591e-06, -6.331237359994088e-09,
4.663834520481827e-10]

[0.0, -1.1494662430988356e-12, -2.460366380668869e-13, 6.732305055269349e-11,
-2.5340265233474384e-10, -1.5113997957441673e-15, 7.495296953394484e-16,
1.9799812091681453e-05, -1.240341359109723e-06, -7.589158883705078e-09,
4.770217759616848e-10]

[0.0, -1.1185891737397886e-12, -1.9023767006816107e-13, 7.45297686040849e-11,
-2.2900325370036764e-10, -1.4963768835060176e-15, 8.558336676986174e-16,
1.675053589486192e-05, -1.3147007334887472e-06, -6.757993918753393e-09,
5.382902440900408e-10]

[0.0, -1.1025519195689977e-12, -2.4504408222061947e-13, 6.78827473380817e-11,
-2.016001331445184e-10, -1.6507094145442349e-15, 8.000661913087781e-16,
1.9566517264507644e-05, -1.2277010983301869e-06, -7.0061282304269715e-09,
4.644033394764 24e-10]

[0.0, -9.72298200226506e-13, -2.297432902874333e-13, 6.968909389192914e-11,
-2.452119177094115e-10, -1.5595656807018448e-15, 8.40885234824761e-16,
2.134849145907601e-05, -1.2316591907706678e-06, -8.039463733982572e-09,
5.337934756779415e-10]

[0.0, -1.1930427372516256e-12, -2.5024812639536645e-13, 6.611922529986758e-11,
-2.618504123025813e-10, -1.5085320303821504e-15, 7.798944938218362e-16,
1.8092440021107 88e-05, -1.0860456312622421e-06, -6.40849185707947e-09,
```

```
4.916443006359673e-10]

[0.0, -1.1259171483361592e-12, -2.670026764079384e-13, 6.59708452048971e-11,
-2.352745319478894e-10, -1.5551800193584466e-15, 8.062920898794651e-16,
1.8452902380952305e-05, -1.4155797196716555e-06, -7.171211295197036e-09,
4.734957470231667e-10]

[0.0, -1.1351385833461303e-12, -2.5030128630506676e-13, 6.750529375825805e-11,
-2.4942114125512086e-10, -1.757404440098518e-15, 9.373632298181026e-16,
1.820986020230218e-05, -1.2844689608127254e-06, -7.172722877898097e-09,
4.527373956123947e-10]
```

```
After Selection (Parents Selected)
[0.0, -1.079535153452445e-12, -2.104688073163699e-13,
6.43872047842357e-11, -2.0661451579880152e-10,
-1.4872224263480214e-15, 8.364617410025721e-16,
2.064932775606508e-05, -1.263889516424591e-06,
-6.331237359994088e-09, 4.663834520481827e-10]

[0.0, -1.1185891737397886e-12, -1.9023767006816107e-13,
7.45297686040849e-11, -2.2900325370036764e-10,
-1.4963768835060176e-15, 8.558336676986174e-16,
1.675053589486192e-05, -1.3147007334887472e-06,
-6.757993918753393e-09, 5.382902440900408e-10]
```

```
---After Crossover----
0 [0.0, -1.079535153452445e-12, -2.104688073163699e-13,
6.43872047842357e-11, -2.0661451579880152e-10,
-1.4963768835060176e-15, 8.558336676986174e-16,
1.675053589486192e-05, -1.3147007334887472e-06,
-6.757993918753393e-09, 5.382902440900408e-10]
1 [0.0, -1.1185891737397886e-12, -1.9023767006816107e-13,
7.45297686040849e-11, -2.2900325370036764e-10,
-1.4872224263480214e-15, 8.364617410025721e-16,
2.064932775606508e-05, -1.263889516424591e-06,
-6.331237359994088e-09, 4.663834520481827e-10]
2 [0.0, -1.079535153452445e-12, -2.104688073163699e-13,
6.43872047842357e-11, -2.0661451579880152e-10,
-1.4963768835060176e-15, 8.558336676986174e-16,
1.675053589486192e-05, -1.3147007334887472e-06,
```

```
-6.757993918753393e-09, 5.382902440900408e-10]
3 [0.0, -1.1185891737397886e-12, -1.9023767006816107e-13,
7.45297686040849e-11, -2.2900325370036764e-10,
-1.4872224263480214e-15, 8.364617410025721e-16,
2.064932775606508e-05, -1.263889516424591e-06,
-6.331237359994088e-09, 4.663834520481827e-10]
4 [0.0, -1.079535153452445e-12, -2.104688073163699e-13,
6.43872047842357e-11, -2.0661451579880152e-10,
-1.4963768835060176e-15, 8.558336676986174e-16,
1.675053589486192e-05, -1.3147007334887472e-06,
-6.757993918753393e-09, 5.382902440900408e-10]
5 [0.0, -1.1185891737397886e-12, -1.9023767006816107e-13,
7.45297686040849e-11, -2.2900325370036764e-10,
-1.4872224263480214e-15, 8.364617410025721e-16,
2.064932775606508e-05, -1.263889516424591e-06,
-6.331237359994088e-09, 4.663834520481827e-10]
6 [0.0, -1.079535153452445e-12, -2.104688073163699e-13,
6.43872047842357e-11, -2.0661451579880152e-10,
-1.4963768835060176e-15, 8.558336676986174e-16,
1.675053589486192e-05, -1.3147007334887472e-06,
-6.757993918753393e-09, 5.382902440900408e-10]
7 [0.0, -1.1185891737397886e-12, -1.9023767006816107e-13,
7.45297686040849e-11, -2.2900325370036764e-10,
-1.4872224263480214e-15, 8.364617410025721e-16,
2.064932775606508e-05, -1.263889516424591e-06,
-6.331237359994088e-09, 4.663834520481827e-10]
8 [0.0, -1.079535153452445e-12, -2.104688073163699e-13,
6.43872047842357e-11, -2.0661451579880152e-10,
-1.4963768835060176e-15, 8.558336676986174e-16,
1.675053589486192e-05, -1.3147007334887472e-06,
-6.757993918753393e-09, 5.382902440900408e-10]
9 [0.0, -1.1185891737397886e-12, -1.9023767006816107e-13,
7.45297686040849e-11, -2.2900325370036764e-10,
-1.4872224263480214e-15, 8.364617410025721e-16,
2.064932775606508e-05, -1.263889516424591e-06,
-6.331237359994088e-09, 4.663834520481827e-10]
```

```
----After Mutation---
0 [0.0, -1.079535153452445e-12, -2.2130485941154686e-13,
6.797565083842135e-11, -2.2002842143839376e-10,
```

```
-1.4324039744627818e-15, 8.694377716202353e-16,
1.675053589486192e-05, -1.1922864671150518e-06,
-6.3559390051013274e-09, 5.356091083244155e-10]
1 [0.0, -1.1051196680035503e-12, -1.8268398500226053e-13,
7.45297686040849e-11, -2.0728401622370272e-10,
-1.4872224263480214e-15, 8.383223134936998e-16,
2.2613580933531015e-05, -1.1616043048228584e-06,
-5.963647186150295e-09, 4.240046501902943e-10]
2 [0.0, -1.1547889474539964e-12, -2.104688073163699e-13,
6.336935874322554e-11, -2.1008577336316 58e-10,
-1.4813419250551563e-15, 7.824393259623942e-16,
1.834303160248049e-05, -1.3147007334887472e-06,
-6.198872165965791e-09, 5.017195197519232e-10]
3 [0.0, -1.2288331476316485e-12, -1.9023767006816107e-13,
7.140336174388965e-11, -2.3824562043065945e-10,
-1.4165602818133668e-15, 8.268912672703639e-16,
2.064932775606508e-05, -1.263889516424591e-06,
-6.0665396489946285e-09, 4.3619633266426437e-10]
4 [0.0, -9.732631915278377e-13, -1.9268523458297063e-13,
6.43872047842357e-11, -1.9770478848078852e-10,
-1.3628622155180071e-15, 8.914497572530837e-16,
1.7249587891467777e-05, -1.2996202208266248e-06,
-6.394992200687335e-09, 5.382902440900408e-10]
5 [0.0, -1.1904130902486916e-12, -1.9023767006816107e-13,
8.054412655382535e-11, -2.2330682190800138e-10,
-1.3991652979218298e-15, 8.364617410025721e-16,
2.064932775606508e-05, -1.2040790538057312e-06,
-6.2683733178558865e-09, 4.901407034966781e-10]
6 [0.0, -1.079535153452445e-12, -2.022338853802752e-13,
6.202002501045991e-11, -1.9359938672722494e-10,
-1.4054083158024557e-15, 8.558336676986174e-16,
1.5207901198331794e-05, -1.2822165965716564e-06,
-6.757993918753393e-09, 5.696383985645926e-10]
7 [0.0, -1.1185891737397886e-12, -1.890653708032284e-13,
7.45297686040849e-11, -2.0676982648647442e-10,
-1.4784176947401568e-15, 8.364617410025721e-16,
1.9010462255803076e-05, -1.263889516424591e-06,
-5.890886098883041e-09, 5.080321414886827e-10]
8 [0.0, -1.0797722014377074e-12, -1.9463547796035373e-13,
6.278541066085177e-11, -1.8721715835885646e-10,
-1.4878170642546431e-15, 8.690507508641206e-16,
```

```
1.6094616751788894e-05, -1.3767083939267616e-06,
-6.220332125633963e-09, 5.174947693639254e-10]
9 [0.0, -1.1730789065124348e-12, -2.0274698475522815e-13,
7.55046924750878e-11, -2.442551828005791e-10,
-1.4785285634079156e-15, 8.364617410025721e-16,
2.0634252946937673e-05, -1.2470437646482502e-06,
-6.853554344589177e-09, 4.3772473516571867e-10]
```

P	Errors	After Selection	After Crossover	After Mutation
P0	7.66338e+11	P2	C0	m0
P1	3.97316e+12	P4	c1	m1
P2	6.83571e+11		c2	m2
P3	9.13661e+11		c3	m3
P4	4.82494e+11		c4	m4
P5	1.04253e+12		c5	m5
P6	4.0324e+12		c6	m6
P7	8.44193e+12		c7	m7
P8	1.27442e+13		c8	m8
P9	6.70617e+12		c9	m9

Diagram 2

```
--- Initial Population ----
[0.0, -1.1469285701828069e-12, -1.7215210710532604e-13,
7.45297686040849e-11, -2.2745359599912043e-10,
-1.4873915825082245e-15, 8.581813706869903e-16,
1.5202877073104481e-05, -1.1988088572875665e-06,
-6.757993918753393e-09, 5.505717178252742e-10]

[0.0, -1.079535153452445e-12, -2.078743660823062e-13,
5.919596719910297e-11, -2.163856637859474e-10,
-1.5711640113683867e-15, 8.792694982889512e-16,
2.064932775606508e-05, -1.263889516424591e-06,
-6.331237359994088e-09, 4.439310071461522e-10]

[0.0, -1.1494662430988356e-12, -2.505978772268713e-13,
6.360077459179161e-11, -2.439755314152263e-10,
-1.5113997957441673e-15, 7.495296953394484e-16,
1.8043722415625692e-05, -1.240341359109723e-06,
-7.401192136005241e-09, 5.149901482956164e-10]

[0.0, -1.2312539487401535e-12, -1.9347018328121458e-13,
6.112610583797424e-11, -2.100857733631658e-10,
-1.4872355501644878e-15, 7.824393259623942e-16,
1.8226635686066958e-05, -1.2761107110615816e-06,
-5.646029278939431e-09, 5.06603036852065e-10]

[0.0, -1.086686859160664e-12, -2.3537456382141234e-13,
6.975974757622723e-11, -2.107151357159645e-10,
-1.6347808921161718e-15, 7.613199254652827e-16,
2.1347358797044756e-05, -1.3220039529089216e-06,
-7.649421026901291e-09, 4.927792862821991e-10]

[0.0, -1.1337087008066112e-12, -2.422699621836107e-13,
7.29771284521478e-11, -2.424054586432543e-10,
-1.7373919692672812e-15, 8.01766462974277e-16, 1.8440481732159e-05,
-1.1241692930739975e-06, -6.006201618838176e-09,
4.679392040097709e-10]

[0.0, -1.2415425790123424e-12, -1.8155241789767618e-13,
```

```
6.761732849105636e-11, -2.518341099211178e-10,
-1.4165602818133668e-15, 8.931742514182361e-16,
2.104807674356895e-05, -1.263889516424591e-06,
-6.648118030056081e-09, 4.1234985852983957e-10]

[0.0, -9.941503212615745e-13, -1.9263287025145797e-13,
6.43872047842357e-11, -2.0867494044699705e-10,
-1.3883372835487258e-15, 8.746669749581988e-16,
1.7249587891467777e-05, -1.428578426607234e-06,
-6.976791830346058e-09, 4.980510385900694e-10]

[0.0, -1.0797722014377074e-12, -1.9463547796035373e-13,
6.776489658969167e-11, -1.8196133772708 92e-10,
-1.4971434184613162e-15, 8.449178719327519e-16,
1.6094616751788894e-05, -1.3767083939267616e-06,
-5.9084529721394936e-09, 5.1397532597549 97e-10]

[0.0, -1.2794998216447778e-12, -1.9399785781365432e-13,
6.810356431382087e-11, -2.442551828005791e-10,
-1.6166062639071623e-15, 8.804097742037109e-16,
1.9083153826189373e-05, -1.1455278078528839e-06,
-6.370931882541944e-09, 4.522990070171836e-10]
```

```
After Selection (Parents Selected)
[0.0, -1.086686859160664e-12, -2.3537456382141234e-13,
6.975974757622723e-11, -2.107151357159645e-10,
-1.6347808921161718e-15, 7.613199254652827e-16,
2.1347358797044756e-05, -1.3220039529089216e-06,
-7.649421026901291e-09, 4.927792862821991e-10]
[0.0, -1.1494662430988356e-12, -2.505978772268713e-13,
6.360077459179161e-11, -2.439755314152263e-10,
-1.5113997957441673e-15, 7.495296953394484e-16,
1.8043722415625692e-05, -1.240341359109723e-06,
-7.401192136005241e-09, 5.149901482956164e-10]
```

```
---After Crossover----
0 [0.0, -1.086686859160664e-12, -2.3537456382141234e-13,
6.975974757622723e-11, -2.107151357159645e-10,
```

```
-1.5113997957441673e-15, 7.495296953394484e-16,
1.8043722415625692e-05, -1.240341359109723e-06,
-7.401192136005241e-09, 5.149901482956164e-10]
1 [0.0, -1.1494662430988356e-12, -2.505978772268713e-13,
6.360077459179161e-11, -2.439755314152263e-10,
-1.6347808921161718e-15, 7.613199254652827e-16,
2.1347358797044756e-05, -1.3220039529089216e-06,
-7.649421026901291e-09, 4.927792862821991e-10]
2 [0.0, -1.086686859160664e-12, -2.3537456382141234e-13,
6.975974757622723e-11, -2.107151357159645e-10,
-1.5113997957441673e-15, 7.495296953394484e-16,
1.8043722415625692e-05, -1.240341359109723e-06,
-7.401192136005241e-09, 5.149901482956164e-10]
3 [0.0, -1.1494662430988356e-12, -2.505978772268713e-13,
6.360077459179161e-11, -2.439755314152263e-10,
-1.6347808921161718e-15, 7.613199254652827e-16,
2.1347358797044756e-05, -1.3220039529089216e-06,
-7.649421026901291e-09, 4.927792862821991e-10]
4 [0.0, -1.086686859160664e-12, -2.3537456382141234e-13,
6.975974757622723e-11, -2.107151357159645e-10,
-1.5113997957441673e-15, 7.495296953394484e-16,
1.8043722415625692e-05, -1.240341359109723e-06,
-7.401192136005241e-09, 5.149901482956164e-10]
5 [0.0, -1.1494662430988356e-12, -2.505978772268713e-13,
6.360077459179161e-11, -2.439755314152263e-10,
-1.6347808921161718e-15, 7.613199254652827e-16,
2.1347358797044756e-05, -1.3220039529089216e-06,
-7.649421026901291e-09, 4.927792862821991e-10]
6 [0.0, -1.086686859160664e-12, -2.3537456382141234e-13,
6.975974757622723e-11, -2.107151357159645e-10,
-1.5113997957441673e-15, 7.495296953394484e-16,
1.8043722415625692e-05, -1.240341359109723e-06,
-7.401192136005241e-09, 5.149901482956164e-10]
7 [0.0, -1.1494662430988356e-12, -2.505978772268713e-13,
6.360077459179161e-11, -2.439755314152263e-10,
-1.6347808921161718e-15, 7.613199254652827e-16,
2.1347358797044756e-05, -1.3220039529089216e-06,
-7.649421026901291e-09, 4.927792862821991e-10]
8 [0.0, -1.086686859160664e-12, -2.3537456382141234e-13,
6.975974757622723e-11, -2.107151357159645e-10,
-1.5113997957441673e-15, 7.495296953394484e-16,
```

```
1.8043722415625692e-05, -1.240341359109723e-06,
-7.401192136005241e-09, 5.149901482956164e-10]
9 [0.0, -1.1494662430988356e-12, -2.505978772268713e-13,
6.360077459179161e-11, -2.439755314152263e-10,
-1.6347808921161718e-15, 7.613199254652827e-16,
2.1347358797044756e-05, -1.3220039529089216e-06,
-7.649421026901291e-09, 4.927792862821991e-10]
---------------------------
```

```
----After Mutation---
0 [0.0, -1.122786960888209e-12, -2.486604108699062e-13,
7.240532675756741e-11, -2.086433661311917e-10,
-1.3789109426449851e-15, 7.495296953394484e-16,
1.927491099307306e-05, -1.2673836470386736e-06,
-7.925333075621078e-09, 5.643097109892329e-10]
1 [0.0, -1.0592162339024734e-12, -2.549326619459891e-13,
6.064966712030296e-11, -2.634886339090457e-10,
-1.6347808921161718e-15, 7.326820952319715e-16,
2.2691572023635446e-05, -1.3220039529089216e-06,
-7.174047626070847e-09, 4.927792862821991e-10]
2 [0.0, -1.086686859160664e-12, -2.241919959829822e-13,
6.524605540634632e-11, -2.2759192314440598e-10,
-1.6352534294833473e-15, 7.594763622873252e-16,
1.7554132924238075e-05, -1.2004692971480998e-06,
-6.961217194379991e-09, 5.215711314510807e-10]
3 [0.0, -1.1634925091352697e-12, -2.4556979993571535e-13,
6.360077459179161e-11, -2.452823409963677e-10,
-1.6584632261982193e-15, 7.406797006464158e-16,
1.9896280417485077e-05, -1.213899571507717e-06,
-7.649421026901291e-09, 4.635521650470679e-10]
4 [0.0, -1.122851393794114e-12, -2.2328940270486428e-13,
6.975974757622723e-11, -2.1813915314051674e-10,
-1.527850639501381e-15, 8.024085849465966e-16,
1.8874472301760855e-05, -1.240341359109723e-06,
-7.81560928979047e-09, 5.213336402107297e-10]
5 [0.0, -1.1140347731712305e-12, -2.6890969728267497e-13,
6.031250861459204e-11, -2.2230843429707067e-10,
-1.784425851856747e-15, 7.860008987442164e-16,
2.1347358797044756e-05, -1.3220039529089216e-06,
-7.3017381324454166e-09, 5.005214540189527e-10]
```

```
6 [0.0, -1.086686859160664e-12, -2.1739838435382605e-13,
6.362503372094426e-11, -2.1783281648764384e-10,
-1.5113997957441673e-15, 8.097343263462739e-16,
1.7658918563164984e-05, -1.240341359109723e-06,
-7.965826618503262e-09, 4.640109283332568e-10]
7 [0.0, -1.1746792934092583e-12, -2.4253614400645233e-13,
5.791053293618332e-11, -2.439755314152263e-10,
-1.6035666798026697e-15, 7.647002057913965e-16,
1.9766098779338523e-05, -1.2995349467159223e-06,
-7.649421026901291e-09, 5.279040947978224e-10]
8 [0.0, -1.0406278099761568e-12, -2.5543708908103114e-13,
7.451319729238309e-11, -2.122358080964194e-10,
-1.6324233443940322e-15, 7.495296953394484e-16,
1.8208675408957474e-05, -1.1849256048693095e-06,
-7.401192136005241e-09, 4.963010947672338e-10]
9 [0.0, -1.0859243366893135e-12, -2.3690775351859854e-13,
6.415951877065274e-11, -2.515964701996149e-10,
-1.6347808921161718e-15, 7.329801090549721e-16,
2.0325219975659615e-05, -1.3614175441851454e-06,
-7.1059921171247935e-09, 5.145410778827663e-10]
```

P	Errors
P0	4.39369e+12
P1	1.64151e+12
P2	7.64584e+11
P3	1.56964e+12
P4	1.44532e+12
P5	3.35862e+12
P6	4.90937e+12
P7	8.90986e+12
P8	1.84413e+12
P9	1.33764e+12

After Selection
P2
P4

After Crossover	After Mutation
C0	m0
c1	m1
c2	m2
c3	m3
c4	m4
c5	m5
c6	m6
c7	m7
c8	m8
c9	m9

Diagram 3

```
--- Initial Population ----
[0.0, -1.1494662430988356e-12, -2.263729437622432e-13,
6.360077459179161e-11, -2.5803247009994175e-10,
-1.6596351409861733e-15, 7.003382947209 94e-16,
1.788221781602218e-05, -1.3185443830340822e-06,
-6.672856371345904e-09, 4.965628859488993e-10]

[0.0, -1.1746792934092583e-12, -2.4769625497662684e-13,
5.943524103805658e-11, -2.4557163443427397e-10,
-1.6035666798026697e-15, 7.2240398141729 53e-16,
2.0387857453283018e-05, -1.3146351589380523e-06,
-7.964942031260605e-09, 5.279040947978224e-10]

[0.0, -1.0806023244330812e-12, -2.434060080708609e-13,
7.396712709918929e-11, -2.1071513571596 45e-10,
-1.7133931007134869e-15, 6.905709086967505e-16,
1.9740055825059504e-05, -1.210982940204502e-06,
-8.31211311051046e-09, 5.328128099357357e-10]

[0.0, -1.0592162339024734e-12, -2.3785003932760514e-13,
5.61761987604961e-11, -2.8891764566730374e-10,
-1.6347808921161718e-15, 6.778973690063764e-16,
2.4892574293961868e-05, -1.3387079834252114e-06,
-7.49063699627434e-09, 5.120756471527596e-10]

[0.0, -1.0859243366893135e-12, -2.4693398747027993e-13,
6.415951877065274e-11, -2.4860908401059995e-10,
-1.7003625265520764e-15, 7.329801090549721e-16,
2.0325219975659615e-05, -1.3614175441851454e-06,
-7.267575381233706e-09, 5.594489497632056e-10]

[0.0, -9.887904692637233e-13, -1.9545859195350447e-13,
6.436244116190 23e-11, -2.032802269179343e-10,
-1.4745087711422343e-15, 8.247550529582374e-16,
2.064932775606508e-05, -1.2053190548312718e-06,
-6.060758993686622e-09, 4.255384003868494e-10]
```

```
[0.0, -1.0410220138476847e-12, -2.6890969728267497e-13,
5.7779246909371996e-11, -2.2240957418394014e-10,
-1.908338441283763e-15, 7.105490742795518e-16,
1.9465180414061854e-05, -1.324534512025845e-06,
-7.3017381324454166e-09, 4.669829671583271e-10]

[0.0, -1.2318173532512166e-12, -1.9399785781365432e-13,
7.020662688973996e-11, -2.572550891473128e-10,
-1.7046786793170107e-15, 8.388227629196302e-16,
1.9083153826189373e-05, -1.1404717808098031e-06,
-6.792243544733407e-09, 4.522990070171836e-10]

[0.0, -1.0504343905840445e-12, -1.9463547796035373e-13,
6.238095085502305e-11, -1.8656789277961608e-10,
-1.6003359609270392e-15, 8.578971463332912e-16,
1.6249990466967808e-05, -1.4408933772626146e-06,
-5.9084529721394936e-09, 5.203709442551892e-10]

[0.0, -1.245491393061885e-12, -2.066213594419536e-13,
6.143779291779051e-11, -2.1388131371905996e-10,
-1.4318969744665095e-15, 7.824393259623942e-16,
1.8288421087103142e-05, -1.2676526372685014e-06,
-5.78023274778072e-09, 5.473631139605111e-10]

----------------------------
```

```
After Selection (Parents Selected)
[0.0, -9.887904692637233e-13, -1.9545859195350447e-13,
6.436244116190 23e-11, -2.032802269179343e-10,
-1.4745087711422343e-15, 8.247550529582374e-16,
2.064932775606508e-05, -1.2053190548312718e-06,
-6.060758993686622e-09, 4.255384003868494e-10]
[0.0, -1.1746792934092583e-12, -2.4769625497662684e-13,
5.943524103805658e-11, -2.4557163443427397e-10,
-1.6035666798026697e-15, 7.224039814172953e-16,
2.0387857453283018e-05, -1.3146351589380523e-06,
-7.964942031260605e-09, 5.279040947978224e-10]
```

```
---After Crossover----
0 [0.0, -9.887904692637233e-13, -1.9545859195350447e-13,
6.436244116190 23e-11, -2.032802269179343e-10,
-1.6035666798026697e-15, 7.2240398141729 53e-16,
2.0387857453283018e-05, -1.3146351589380523e-06,
-7.964942031260605e-09, 5.279040947978224e-10]
1 [0.0, -1.1746792934092583e-12, -2.4769625497662684e-13,
5.943524103805658e-11, -2.4557163443427397e-10,
-1.4745087711422343e-15, 8.247550529582374e-16,
2.064932775606508e-05, -1.2053190548312718e-06,
-6.060758993686622e-09, 4.255384003868494e-10]
2 [0.0, -9.887904692637233e-13, -1.9545859195350447e-13,
6.436244116190 23e-11, -2.032802269179343e-10,
-1.6035666798026697e-15, 7.2240398141729 53e-16,
2.0387857453283018e-05, -1.3146351589380523e-06,
-7.964942031260605e-09, 5.279040947978224e-10]
3 [0.0, -1.1746792934092583e-12, -2.4769625497662684e-13,
5.943524103805658e-11, -2.4557163443427397e-10,
-1.4745087711422343e-15, 8.247550529582374e-16,
2.064932775606508e-05, -1.2053190548312718e-06,
-6.060758993686622e-09, 4.255384003868494e-10]
4 [0.0, -9.887904692637233e-13, -1.9545859195350447e-13,
6.436244116190 23e-11, -2.032802269179343e-10,
-1.6035666798026697e-15, 7.2240398141729 53e-16,
2.0387857453283018e-05, -1.3146351589380523e-06,
-7.964942031260605e-09, 5.279040947978224e-10]
5 [0.0, -1.1746792934092583e-12, -2.4769625497662684e-13,
5.943524103805658e-11, -2.4557163443427397e-10,
-1.4745087711422343e-15, 8.247550529582374e-16,
2.064932775606508e-05, -1.2053190548312718e-06,
-6.060758993686622e-09, 4.255384003868494e-10]
6 [0.0, -9.887904692637233e-13, -1.9545859195350447e-13,
6.436244116190 23e-11, -2.032802269179343e-10,
-1.6035666798026697e-15, 7.2240398141729 53e-16,
2.0387857453283018e-05, -1.3146351589380523e-06,
-7.964942031260605e-09, 5.279040947978224e-10]
7 [0.0, -1.1746792934092583e-12, -2.4769625497662684e-13,
5.943524103805658e-11, -2.4557163443427397e-10,
-1.4745087711422343e-15, 8.247550529582374e-16,
2.064932775606508e-05, -1.2053190548312718e-06,
-6.060758993686622e-09, 4.255384003868494e-10]
```

```
8 [0.0, -9.887904692637233e-13, -1.9545859195350447e-13,
6.436244116190 23e-11, -2.032802269179343e-10,
-1.6035666798026697e-15, 7.2240398141729 53e-16,
2.0387857453283018e-05, -1.3146351589380523e-06,
-7.964942031260605e-09, 5.279040947978224e-10]
9 [0.0, -1.1746792934092583e-12, -2.4769625497662684e-13,
5.943524103805658e-11, -2.4557163443427397e-10,
-1.4745087711422343e-15, 8.247550529582374e-16,
2.064932775606508e-05, -1.2053190548312718e-06,
-6.060758993686622e-09, 4.255384003868494e-10]
----------------------------
```

```
----After Mutation---
0 [0.0, -1.0055027924753208e-12, -1.9586905247266753e-13,
6.020476190745503e-11, -2.0015648675430945e-10,
-1.6035666798026697e-15, 7.2240398141729 53e-16,
2.088339075019829e-05, -1.3429803348750737e-06,
-7.596984505008925e-09, 5.279040947978224e-10]
1 [0.0, -1.1746792934092583e-12, -2.4544240277217115e-13,
6.41517135588246e-11, -2.5172863691031333e-10,
-1.5566257728645448e-15, 8.247550529582374e-16,
2.00696287467287e-05, -1.2053190548312718e-06,
-6.060758993686622e-09, 4.1141096258680607e-10]
2 [0.0, -9.23106999378637e-13, -1.8532041797919692e-13,
5.835792894623448e-11, -1.987046096798244e-10,
-1.6035666798026697e-15, 7.282498499724051e-16,
1.8500135321689132e-05, -1.2569797317037775e-06,
-7.964942031260605e-09, 5.515876085573879e-10]
3 [0.0, -1.1746792934092583e-12, -2.5142530919920735e-13,
5.602466772630128e-11, -2.5953385694479963e-10,
-1.5766700449844e-15, 8.412305760016819e-16, 2.029999716364467e-05,
-1.1452412277373295e-06, -6.595158239259646e-09,
4.1129544462204313e-10]
4 [0.0, -1.006730862004299e-12, -2.1025983553875117e-13,
6.436244116190 23e-11, -2.0557190937149915e-10,
-1.7538682179567057e-15, 7.2240398141729 53e-16,
2.0054179409071747e-05, -1.2192727720981923e-06,
-8.660045333991128e-09, 5.729532798748851e-10]
5 [0.0, -1.1971667528409716e-12, -2.3119001258181556e-13,
5.943524103805658e-11, -2.3659357146592973e-10,
```

```
-1.3581899163894447e-15, 8.729259985944184e-16,
2.0313486829982953e-05, -1.1069310700615147e-06,
-5.655319351570651e-09, 4.2150124725583636e-10]
6 [0.0, -9.552786291762394e-13, -1.9545859195350447e-13,
6.618745594479859e-11, -2.050493597798407e-10,
-1.726018265135294e-15, 6.990370807065524e-16,
2.1704204320041904e-05, -1.3390120340920367e-06,
-7.681406236719803e-09, 5.279040947978224e-10]
7 [0.0, -1.1746792934092583e-12, -2.567742745047943e-13,
5.943524103805658e-11, -2.2363862216640078e-10,
-1.5000676260385129e-15, 8.660056203093905e-16,
2.0753830352227883e-05, -1.1739912884974962e-06,
-5.475922426685233e-09, 4.022161163923783e-10]
8 [0.0, -9.887904692637233e-13, -1.9545859195350447e-13,
6.288176016292492e-11, -2.217901580934635e-10,
-1.4483372366839704e-15, 7.69568771553268e-16,
2.022341228389472e-05, -1.3608573080066413e-06,
-8.317372516351351e-09, 5.750686453725023e-10]
9 [0.0, -1.153882511048267e-12, -2.684333605924368e-13,
5.462449900999165e-11, -2.6905988912481425e-10,
-1.5694410249367222e-15, 8.247550529582374e-16,
2.046671146424829e-05, -1.1913404700311248e-06,
-6.060758993686622e-09, 4.255384003868494e-10]
---------------------------
```

P	Errors	After Selection	After Crossover	After Mutation
P0	1.18577e+12	P5	C0	m0
P1	4.72118e+11	P1	c1	m1
P2	2.11275e+12		c2	m2
P3	1.47358e+12		c3	m3
P4	1.24557e+12		c4	m4
P5	7.0546e+11		c5	m5
P6	4.82967e+12		c6	m6
P7	8.91544e+11		c7	m7
P8	3.29938e+12		c8	m8
P9	6.18135e+12		c9	m9

2. Fitness Function

The fitness function is one of the most important parts of the genetic algorithm we have implemented in our project. The fitness function will determine how our new generation will look like. This is because the parent selection is based heavily on how fit the parent is. The function calculates the fitness of a vector using a linear combination of *validation* and *train errors*. The idea behind limiting *train error* using a *train_factor* is that since the initial population is overfit, the train errors will be too good and using them with the same weightage as *validation error* would be misleading and make the algorithm corrupt. This would lead to our model again being overfit and thus not providing any improvement on the model. Currently, **the smaller the linear combination of errors (*popfit* value), the better the fitness.**

```
def fitness():
   print("Train and Validation error ")
   for i in range(10):
       train, validation = server.get_errors(
           'QXoVw3Dy9NvP3wdrvWwcPgFBYMSDDJTxsL1npdylcxC0j90Ff3',
population[i])
       fit = (validation + (train)*train_factor)
       popfit.append(fit)

       print(train, validation)
```

A list of vectors is called fit when the linear combination of errors is less for that as compared to other lists of vectors. Therefore, we have implemented another function that normalizes the fitness, *normFit()*, for all lists of vectors. Currently, the *popfit* list consists of just a linear combination of errors. We normalize it using the sum of linear combination of errors and then taking the reciprocal of the fraction. This gives a normalized score. Here, **the larger the *popfit* value, better the fitness.**

```
def normFit():
   ss = np.sum(popfit)
   for i in range(10):
       popfit[i] = popfit[i]/ss
       popfit[i] = 1/popfit[i]
```

Similarly, there are other functions in our code, namely, *fitness2()* and *normFit2()* which have the exact same logic and the only difference being that they calculate fitness values for the offsprings created while the original fitness functions calculated fitness values for population being used. Finally, the fitness values of

both offsprings and population are compared and only the best 10 of them are included in the next generation.

3. Crossover Function

As stated earlier parents are selected according to their fitness. This algorithm makes use of only 2 parents. The best fit vector lists are selected as the parents. After Selection of parents to generate new generations (offsprings) crossover function is used .

```
def crossover():
    mx = -1e17
    ti =0
    for i in range(len(popfit)):
            if(popfit[i] > mx):
                di = i
                mx = popfit[i]
    maxx = -1e17
    for i in range(len(popfit)):
            if(popfit[i] > maxx and i!=di):
                mi = i
                maxx = popfit[i]
    maxxx = -1e17
    for i in range(len(popfit)):
            if(popfit[i] > maxxx and i!=di and i!=mi):
                ti = i
                maxxx = popfit[i]
    print("After Selection (Parents Selected) ")
    print(population[mi])
    print(population[di])
    print("----------------------------")
    for f in range(5):

        mom = population[mi]
        dad = population[di]

        part = 5
        # print(part)
        child = []
        child2 = []
        child = np.copy(dad)
        child2 = np.copy(mom)
        child2[0:part] = dad[0:part]
```

```
        child[0:part] = mom[0:part]
        c1 = child.tolist()
        c2 = child2.tolist()
        offsprings.append(c1)
        offsprings.append(c2)
```

Once the list of offsprings is generated, they are then mutated and their fitness values are checked using *fitness2()* and *normFit2()* as stated earlier. The offsprings and their fitness values are then appended to a list called *finallist* along with the initial population containing the parents and the other members and their corresponding fitness values. This *finallist* is then sorted according to the fitness values. The best 10 vectors of this list are then treated as the new generation or the initial population for the next iteration of the genetic algorithm.

```
crossover()
    print("---After Crossover----")
    for i in range(10):
        print(i,offsprings[i])
    print("----------------------------")
    #print offspring
    # popfit = []
    # population = []
    # population = offsprings
    for i in range(10):
        offsprings[i] = mutate(offsprings[i])
    print("----After Mutation--- ")
    for i in range(10):
        print(i,offsprings[i])
    print("----------------------------")
        #print
    fitness2()
    normFit2()
    finallist = []
    for i in range(10):
        tup =  (population[i],popfit[i])
        finallist.append(tup)

    for i in range(10):
        tup = (offsprings[i],popfit2[i])
        finallist.append(tup)
```

```
    finallist.sort(reverse=True , key=lambda x:x[1])

    for i in range(10):
        vec = finallist[i][0]
        population[i] = vec
```

4. Mutations

Mutating is something that is very important in genetic algorithms. It helps to ensure that there is some diversity when a new generation is created from an existing one. It protects us from encountering redundant offsprings. For the same reason, we have implemented mutation in our code as well. We try to generate a random value from within the mutation range, i.e., -0.1 to 0.1 and multiply it with our existing vector according to the probability. In order to ensure that our mutated vector remains within the range of -10 to 10, we intentionally change its values to the limits if it goes out of the range.

```
def mutate(vector):
    for i in range(len(vector)):
        fact = random.uniform(-mutate_range, mutate_range)
        vector[i] = np.random.choice(
            [vector[i]*(fact+1), vector[i]], p=[prob, 1-prob])
        if(vector[i] < -10):
            vector[i] = -10
        elif(vector[i] > 10):
            vector[i] = 10

    return vector
```

We utilize mutation at two major points in our algorithm. **First** is mutating our initial population at the start of every iteration and **second** is when new offsprings are generated after crossover. This ensures that our new generation will contain the best of mutated population and mutated offsprings.

5. Hyperparameters

There were many different parameters that we used according to the need of our algorithm. Some of them are listed below.

```
data = [0.0, -1.3447599686132163e-12, -2.5276829761244794e-13,
```

```
6.238662437014422e-11, -3.194875673670004e-10, -1.4487325427995935e-15,
6.748580769679454e-16, 1.568853941472079e-05, -1.0233670896755657e-06,
-4.51449312296727e-09, 3.5944935180850177e-10]
# data = [0.0, -1.45799022e-12, -2.28980078e-13,  4.62010753e-11,
-1.75214813e-10, -1.83669770e-15,  8.52944060e-16,  2.29423303e-05,
-2.04721003e-06, -1.59792834e-08,  9.98214034e-10]
mutate_range = 0.1
prob = 0.80
population = [list(data) for i in range(10)]
offsprings = []
popfit = []
popfit2 = []
mom = []
dad = []
train_factor = 0.5
```

Since we implemented our algorithm part by part instead of a pool, our population generally consisted of one our own vectors generated from previous iterations. This same vector was stored in *data* and then later duplicated into the population. This would later be mutated to ensure diversity.

Then we have the *mutate_range* to ensure that our mutated vectors remained within a certain range. The *prob* parameter here refers to the probability that a vector will be mutated or not. This ensures that although there is diversity in the mutated generation, some of the vectors also carry their original data with them. The specific value of 0.80 was obtained after many hit and trial iterations.

The *train_factor* parameter ensured that we only considered a part of the train error when building our fitness model. This is due to the fact that the given model is overfit and although it does get better iteration after iteration, it still remains overfit to a certain extent. Therefore considering a part of train error and the whole of validation error ensures that our fitness value model is built on strong foundation.

Apart from these parameters, the algorithm makes use of lists like *mom, dad, popfit, popfit2, population,* and *offsprings*. As the name suggests *mom* stores one of the parents, *dad* stores one of the parameters, *population* stores the current generation, and *offsprings* store the new generation created after crossover. *popfit* stores the fitness values of the mutated *population* while *popfit2* stores the fitness values of the mutated *offsprings*.

```
part = 5
# print(part)
```

```
child = []
child2 = []
child = np.copy(dad)
child2 = np.copy(mom)
child2[0:part] = dad[0:part]
child[0:part] = mom[0:part]
c1 = child.tolist()
c2 = child2.tolist()
offsprings.append(c1)
offsprings.append(c2)
```

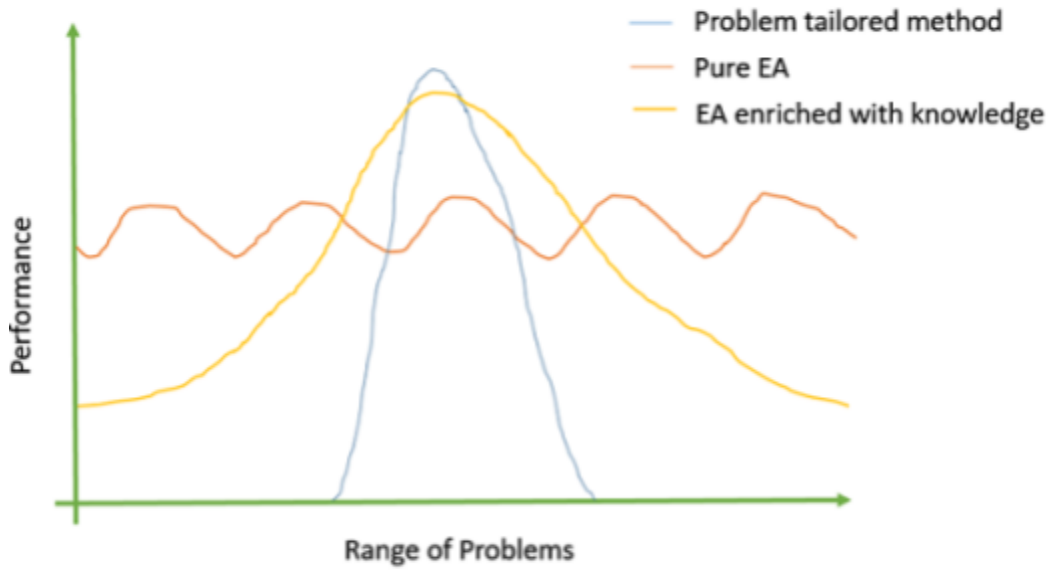
The splitting point for our genes was exactly half, i.e., 5 of the parents. This ensured that the children produced were original and their values did not lean too much towards a certain parent. Doing mutation later on these children resulted in a large amount of diversity which turned out to be good for our model in the end.

6. Statistical Information

Our model generally required within 5 to 8 iterations to generate “good” vectors, i,e, vectors that performed good on unseen dataset as their MSE (Mean-square error) ,calculated from validation and train error, was less.

```
for x in range(5):
```

The range is limited as like all genetic/evolutionary models there are peaks when models perform best. That peak generally existed within this range as we noted from our trials. Lesser iterations meant the model was still overfit and too many meant that the peak performance of the model had passed and the model was again proceeding towards being overfit.



Michalewicz's (1990) view of the EA

As one can see from Michalewicz's function genetic models (EA) when they are enriched with knowledge, there is a certain peak when the model performs best. Before and after that, the performance drops. Similarly, in our case running the model for 5 to 8 iterations ensured that our model remained close to the peak.

7. Heuristics Applied

There was a wide range of heuristics that we have applied. **Firstly,** was the fitness function that we had stated earlier as well. We made use of *train_factor* to build a linear combination of the validation and train errors. Then, we normalized the fitness values for the sake of legibility and better readability.

Secondly, as stated earlier, our mutation depended on a probability as well. This probability, i.e., 0.80 ensured that our vectors were being mutated most of the time but not everytime in order to preserve originality and at the same time introduce diversity.

Thirdly, we also tried to implement the Russian Roulette model when selecting our parents. We thought this would ensure that there remains some variation in our offsprings generated as it considered the fitness values along with a pinch of probability when selecting the parents. However, it did not perform as good as

selecting parents according to their fitness values purely. Therefore, it was not used at the end.

```
def russianRoule():
   s = np.sum(popfit)
   print("SUM Norm")
   print(s)
   print()
   r = random.randint(0, int(s))
   p = r
   for v in range(10):
       p += popfit[v]
       if(p > s):
           return v
   return v-1
```

8. Train and Validation Error

Since we ran our algorithm countless times, it is given that we got to see a wide range of train errors and validation errors when submitting our generations for fitness checks. However, a general trend was noticed that is also logically sound. The trend was that when **the difference between the train error and validation error reduced, the model performed better on the unseen dataset.** On the other hand when there are large differences between the errors, i.e., when the value of validation error would be a lot larger than the value of train error, the performance of the model in general would be a lot worse.

For instance, we can take the best values of train error and validation error our model was able to achieve.

```
Train and Validation error
207518088523.88824 248276902630.60898
139447980995.9074 328538213175.55
168182798563.93872 239479009147.58755
549196335657.23126 304073263583.3705
193982168749.3305 484026901621.1126
193003196944.06003 737202192196.5548
778041120670.1606 469410072450.6398
227839065183.96368 863896480825.5333
```

```
878592591234.2351 579621361826.6809
1032975202615.3147 699013096638.8289
```

The above values scored the least MSE Score our model was able to attain. The values of Train error are mostly in the range of 1e12 and the values of the validation error are mostly in the range of 1e12 too. Therefore, there is little difference between the two error values. If only our train error was low and our validation error was a lot higher, say in the powers of 1e13 or 1e14, then our MSE score would have been higher. This would signify that our model performed bad. This is because since only the train error was low, the model still remained overfit. Having error values in about the same range means that our model is no longer overfit as it's values are not biased towards the training dataset.

Hence, our model was able to perform nicely on the unseen data set as it is no longer overfit and delivers values in similar ranges for both validation and train error.