The source code, input and readme files can be found in:

https://github.com/mnarasim/ML7641/tree/master/HW2

In this project in addition to using NumPy, Pandas, Scikit learn libraries I have also used optimization algorithm libraries built by Genevieve Hayes (https://github.com/gkhayes/mlrose).

The optimization problems used in this project are as follows: 1) Knap Sack 2) Four Peaks 3) Max K – Colors. The idea is to analyze the 4 optimization algorithms on the chosen problems based on the following criteria: how well do these perform on the overall fitness function, how do these perform in terms of run time, does varying the number of iterations and the number of attempts at each step make any difference.

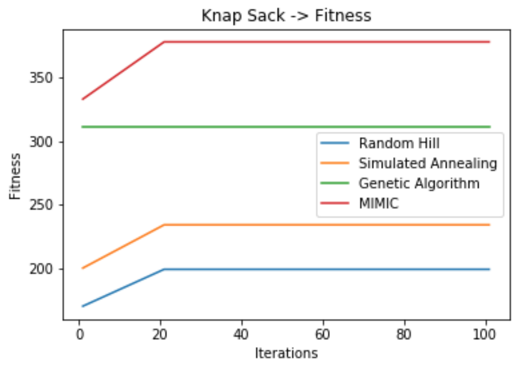
Knap Sack:

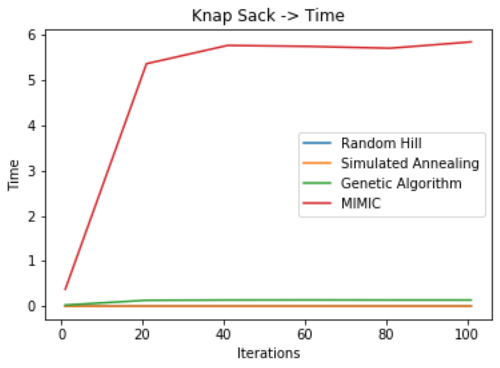
Given a set of values, corresponding weights and a maximum weight, this problem determines a fitness value that makes sure that the sum of product of the values and the corresponding weights is less than the maximum weight.

Based on the fitness values plotted for different number of iterations, I can clearly see that the MIMIC performs the best and Random Hill performs the worst. Simulated Annealing and Genetic Algorithm stand in between these two, with Genetic Algorithm performing better than Simulated Annealing.

It is interesting to note that except Genetic algorithm (GA) all the others do indeed take about 20 iterations to arrive at their best fitness value. GA on the other hand achieves its best fitness value in 1 iteration. Just because MIMIC returns the best fitness value, it doesn't mean that it should be used all the time for Knap Sack problems. The reason is the time taken by MIMIC for an iteration is way more than the time taken by all the other algorithms. So, there is a tradeoff between what it would cost to run one iteration of MIMIC and the corresponding benefit achieved in terms of a better fitness value.

The following two graphs were plotted for different number of iterations with the number of attempts set to a minimum value of 5.

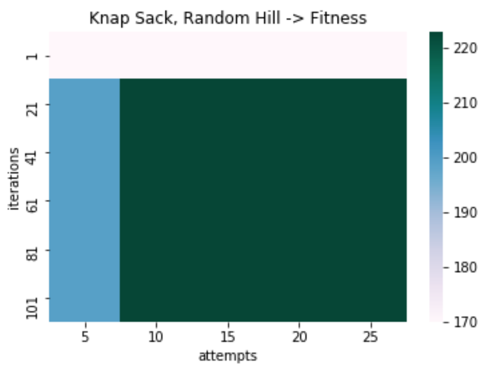


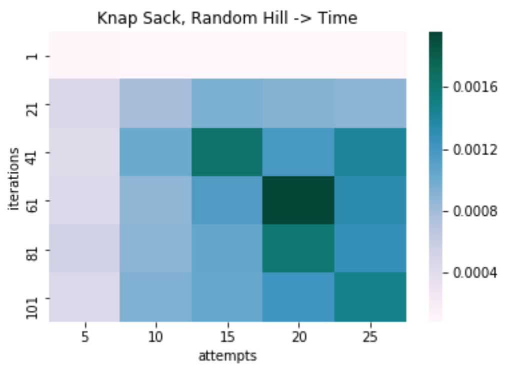


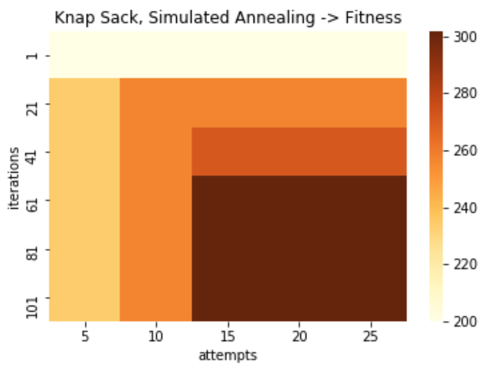
Next, I wanted to find out how varying the number of iterations and the number of attempts the algorithm would take to find a better state at each step affect the overall fitness.

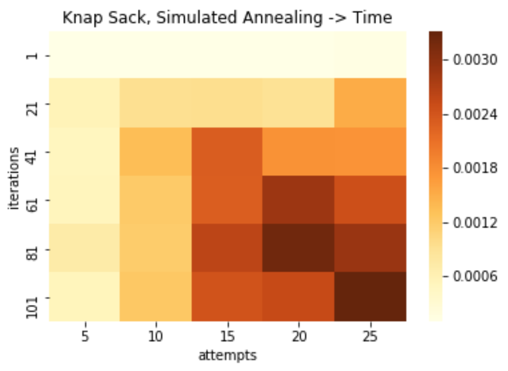
- **Random Hill (RH):** The number of iterations and the attempts don't affect the fitness significantly. The best fitness value is achieved even for low values of these parameters (no. of iterations at least 21 and no. of attempts at least 10). In terms of time, it is interesting to note that the highest time is observed for 61 iterations and 20 attempts. A possible explanation is that algorithm wanders away from the optimum at these parameters, hence takes longer to run though the fitness value is same.

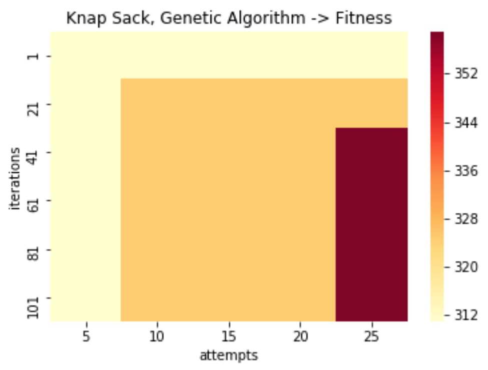
- **Simulated Annealing (SA):** The higher the number of iterations and the attempts the better the fitness. The same is applied for time also, the higher the number of iterations and the attempts the greater the run time.
- **Genetic Algorithm (GA):** The higher the number of iterations and attempts the better the fitness. However, this algorithm needs at least 25 attempts to yield the best fitness. It does look like the number of attempts matter more than the number of iterations because even for low number of iterations the algorithm gives the best fitness if the number of attempts is high.
- **MIMIC:** The number of iterations and attempts don't matter a whole lot as the best fitness is achieved even for a low number of iterations and attempts. However, each iteration of MIMIC takes more time than the other algorithms.

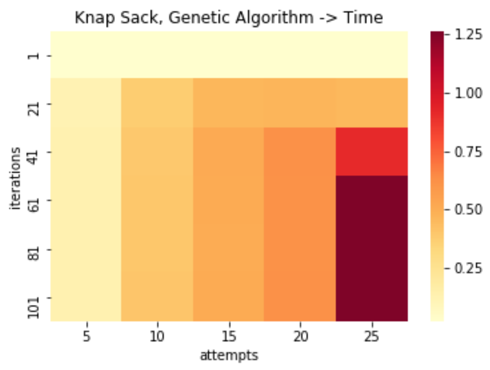


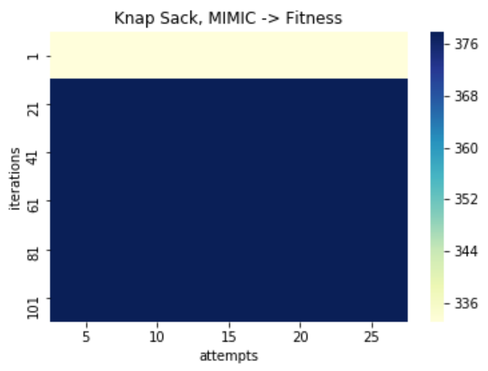


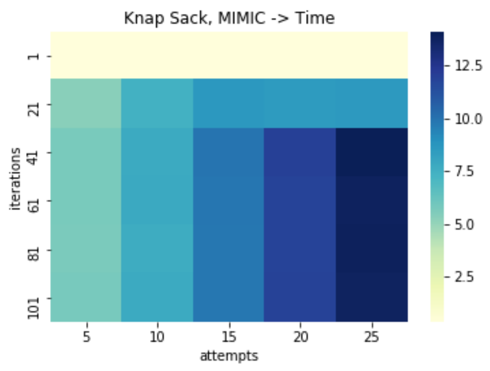












Four Peaks:

Given an n-dimensional state vector this algorithm evaluates the fitness function as

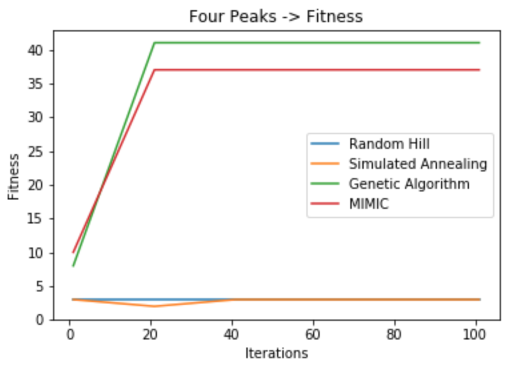
$F\ itne(x, T)$ = max($tail(0, x)$, $head(1, x)$) + $R(x, T)$ ----- source mlrose documentation (release 1.2.0)

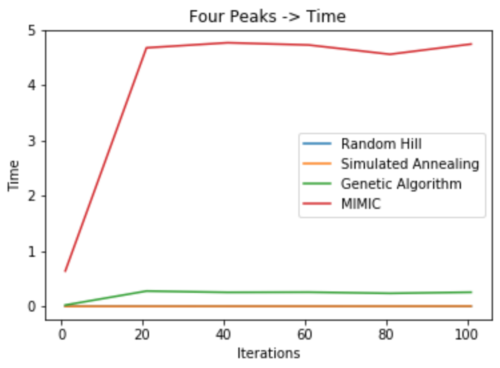
Where tail(0,x) indicates number of trailing 0's, head(1,x) indicates number of leading 1's.

If I look at the fitness values below, I can see that GA performs the best, MIMIC comes next, SA and RH come last. Also, the difference in fitness values obtained show that all the algorithms except GA get struck in a local optimum hence yield a low fitness value. At about 21 iterations I can see that SA is slightly worse than RH.

In terms of time, as expected MIMIC takes the longest whereas SA and RH take the least amount of time. GA only takes slightly longer than SA and RH but gives the best fitness value. Also, increasing the number of iterations after 20, doesn't just proportionally increase the run time.

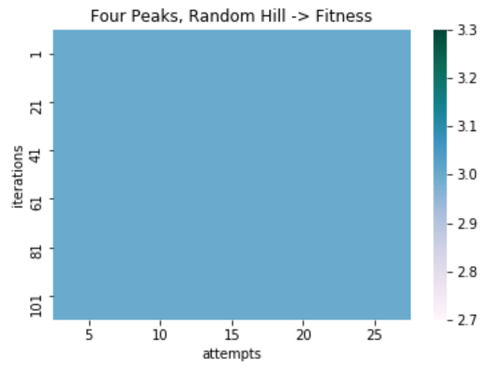
The following two graphs were plotted for different number of iterations with the number of attempts set to a minimum value of 5.

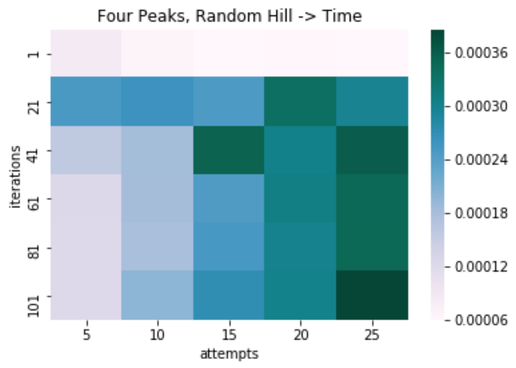


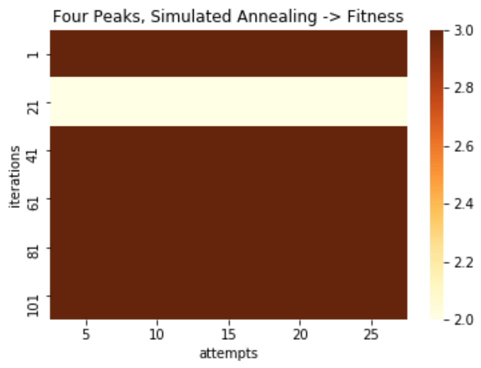


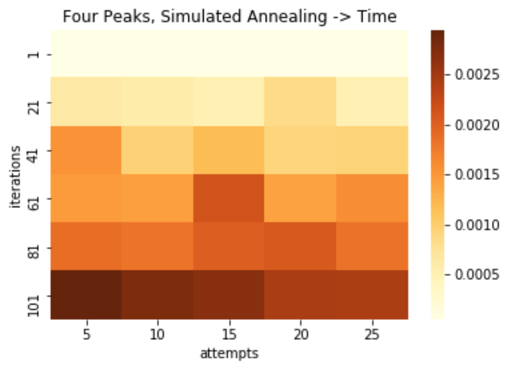
Iterations vs. Attempts:

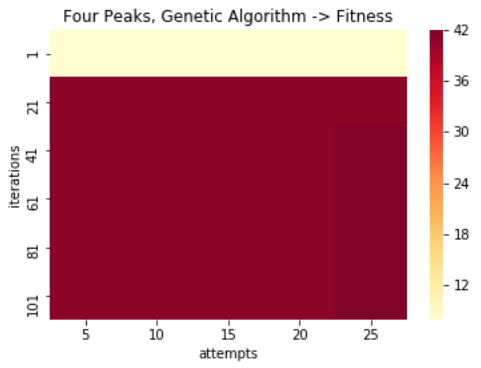
- **Random Hill (RH)**: No effect of varying the number of iterations and the attempts on the fitness. Model finds the local optimum easily. It is interesting to see how the optimization time varies for different number of iterations and attempts as even a fewer number of iterations and attempts take longer than greater number of iterations and attempts.
- **Simulated Annealing (SA):** I can see a drop in the fitness value at 21 iterations. Also, the time taken to run the optimization at 101 iterations and 5 attempts is more than the time at 101 iterations and 25 attempts, though the differences are small.
- **Genetic Algorithm (GA):** A good fitness value is achieved for even low number of iterations and attempts but the best is obtained for 25 attempts and iterations greater than 41. For iterations greater than 21, increasing the number of attempts for fixed number of iterations increases the run time.
- **MIMIC:** At least 10 attempts and 21 iterations are required to achieve the best fitness and the time graph also shows how the run time increases with increasing number of attempts and iterations.

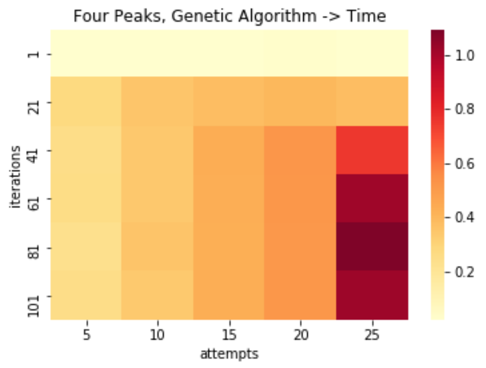


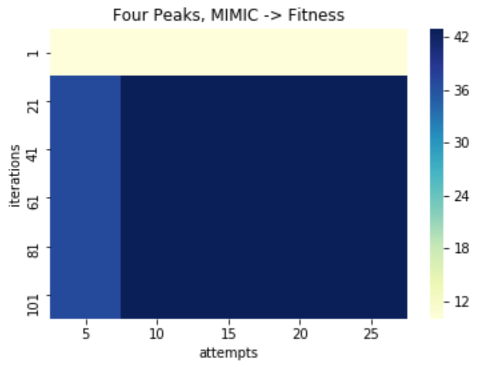


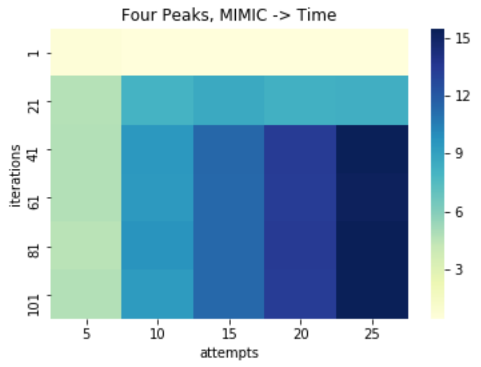












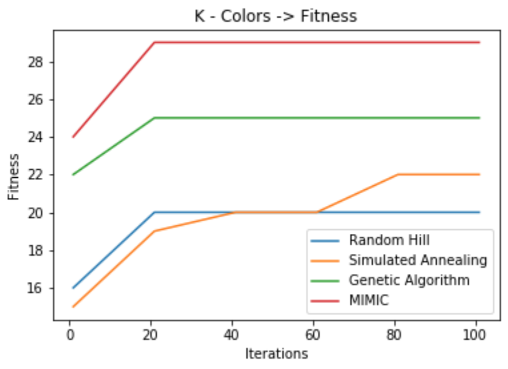
<u>**Max K – Colors:**</u>

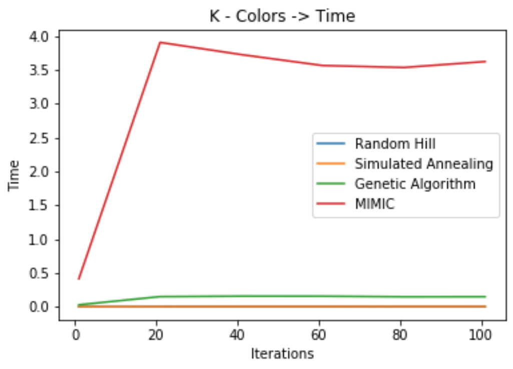
Given a list of pair of nodes that are connected this algorithm determines the maximum number of pairs of adjacent nodes of the same color.

Based on the fitness values, MIMIC performs the best, GA comes next, SA comes third and RH comes last. For low values of number of iterations RH does better than SA, however for high values SA does better than RH. In terms of time MIMIC is the slowest while GA is the next lowest and SA and RH take the least amount of time.

Also, looking at the fitness graph I can see that each of these algorithms except Simulated Annealing take at least 20 iterations to achieve the best fitness value. Simulated Annealing takes 80 iterations to achieve the best fitness value. It can be inferred that SA finds local optimum for low number of iterations and able to find the true optimum as the number of iterations increase to 80.

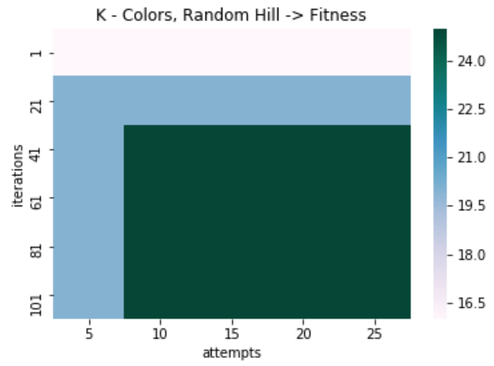
The following two graphs were plotted for different number of iterations with the number of attempts set to a minimum value of 5.

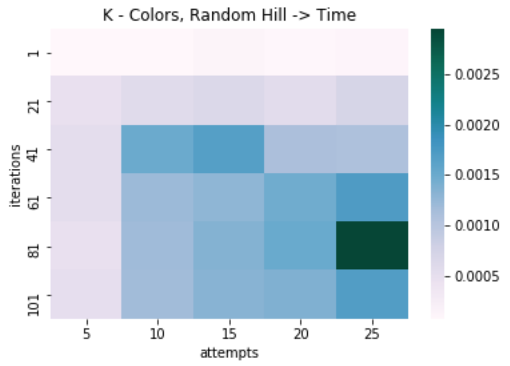


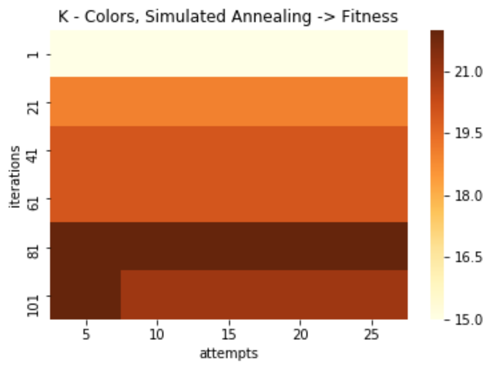


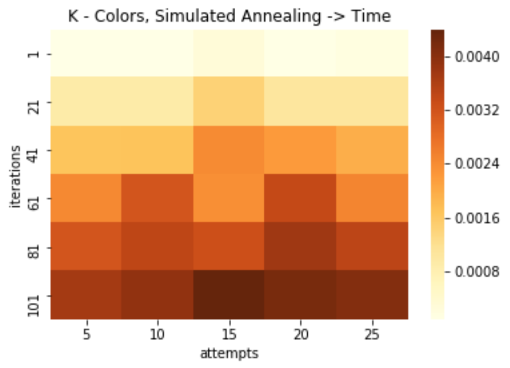
No. of Iterations vs. Attempts:

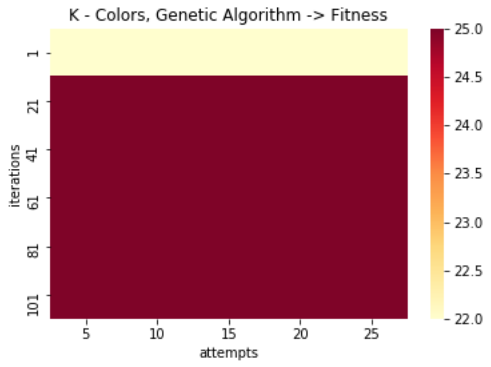
- **Random Hill (RH):** After surpassing the low number of iterations and attempts these parameters don't seem to affect the fitness value.
- **Simulated Annealing (SA):** Increasing the number of iterations seem to affect the fitness value more than the number of attempts. In other words, for a fixed number of attempts varying the number of iterations yields different fitness values whereas for a fixed number of iterations varying the number of attempts yields the same fitness value.
- **Genetic Algorithm (GA):** Both the number of attempts and iterations don't seem to affect the fitness except when the number of iterations is 1. At this instance increasing the number of attempts doesn't help the fitness value.
- **MIMIC:** same as GA above.

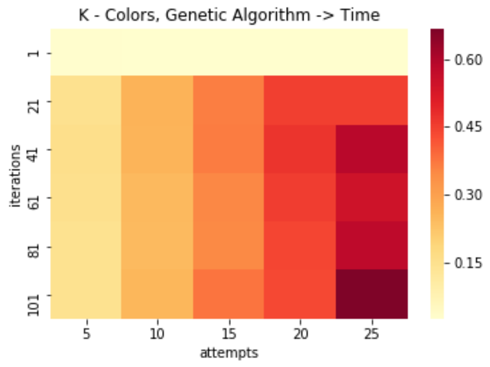


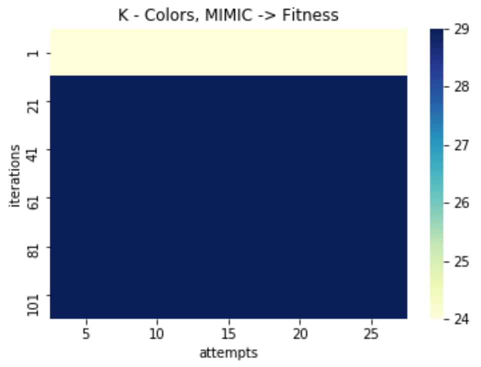


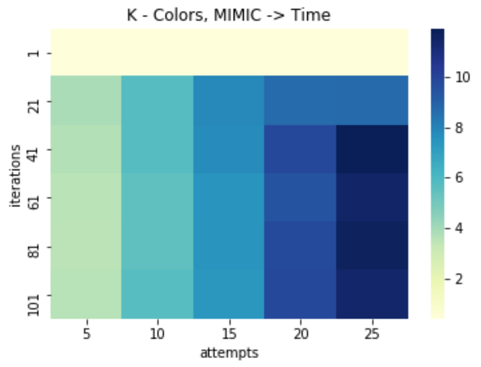












<u>Neural Networks:</u>

I used the heart.csv from project 1. This dataset is about predicting if someone has a heart disease and has 1,025 samples where each sample has 13 attributes. I understand that in Neural Networks there is a general risk of overfitting if the dataset is complex. I think, this dataset is simple without much noise, hence feel it is well suited for this part of this project.

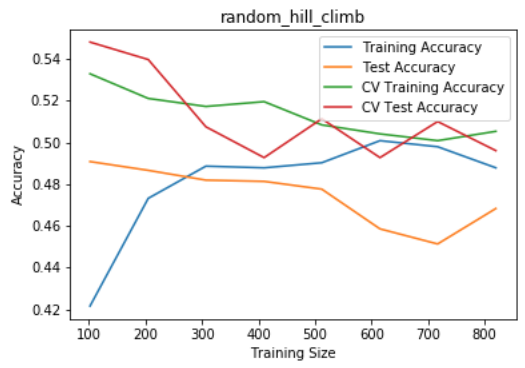
First, I optimized the weights of the Neural Network using the three optimization algorithms: Random Hill (RH), Simulated Annealing (SA) and Genetic Algorithm (GA). For each of these scenarios I ran the model for varying training data set sizes and obtained their training and test accuracies. In addition, for each of those training sizes, I performed cross validation (CV) to see if it helps with the accuracies. It should be noted that I fixed the number of iterations, attempts, learning rate and other parameters and varied only the training data size for this part. Later on, we can see how I fixed the training data set size and varied the number of iterations and attempts to determine the test accuracies.

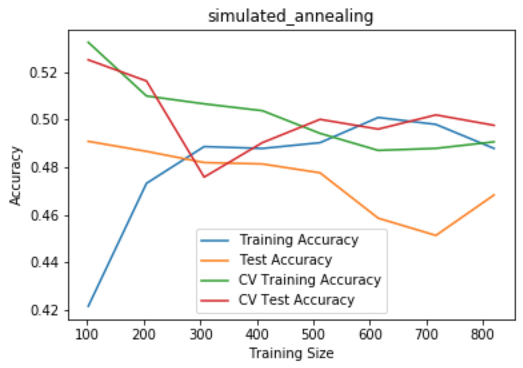
Random Hill (RH): I can see that the training accuracies are much better than the test accuracies after the training data size of 300. This tells me that overfitting is occurring after this data size because the training accuracy keeps increasing while the test accuracy keeps decreasing. The model does well on the training data but not able to apply its learning to the test data.

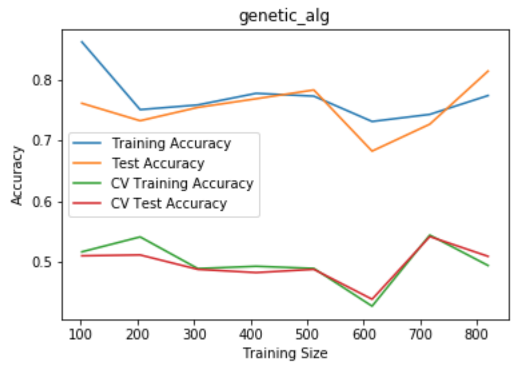
Even when cross validation is performed the test accuracy with CV really goes up and down more or less like a saw tooth. When CV is performed even the training error keeps decreasing with increasing data size. I think this could be because the model has generalized the problem too much and not able to learn effectively.

Simulated Annealing (SA): I see a similar pattern for the training and test accuracies with varying training data set sizes much like RH above. When cross validation is performed the behavior is also like RH except the test accuracy drops significantly at data size 300 and climbs back after that.

Genetic Algorithm (GA): As shown below, I can see that GA performs the best in terms of both training and test accuracies. The above-mentioned accuracies are the best of all the algorithms both with and without CV. However, it should be noted that the accuracies with CV are lower than the case when CV is not performed.





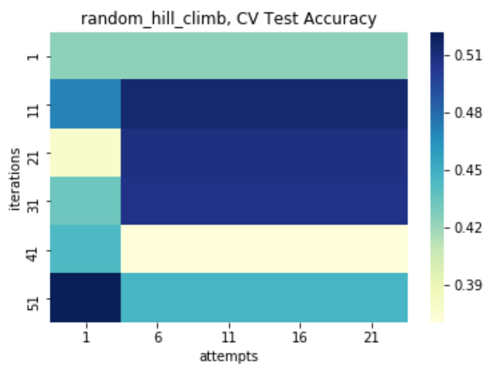


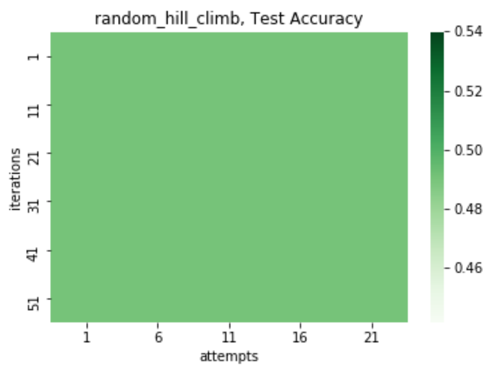
Next, I obtained the training data set size that gave the best cross validation test accuracy and ran the model for varying number of iterations and attempts by fixing the training data size. So, the following graphs denote the test accuracies for varying number of iterations and attempts while keeping the training data size fixed. It has to be noted I determined these test accuracies with and without CV.

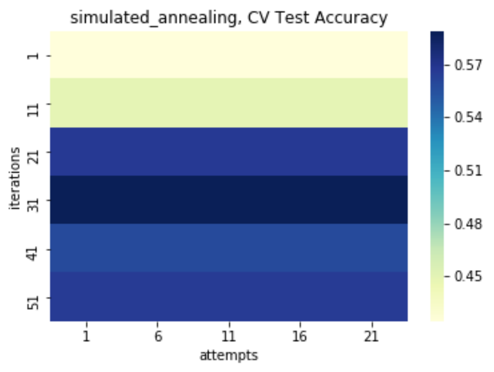
Random Hill (RH): Both the number of iterations and attempts don't seem to affect the test accuracies without CV. However, these parameters do make a difference when CV is performed.

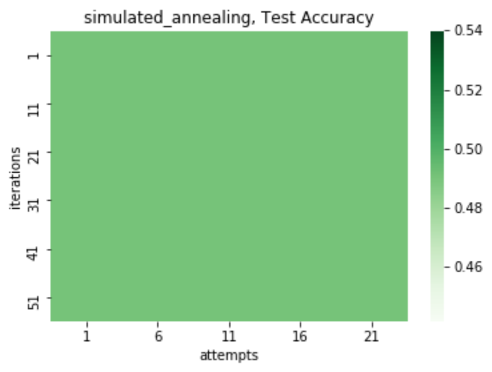
Simulated Annealing (SA): The number of iterations make a difference in this case. I can see a best CV test accuracy of 0.57 when the number of iterations is 31. This accuracy is better than the initial case where the number of iterations and attempts were fixed but the training data set size was varied.

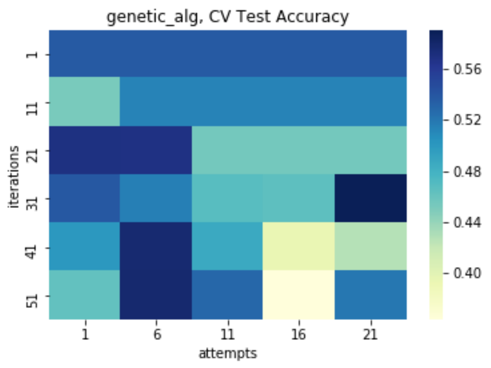
Genetic Algorithm (GA): Both the number of iterations and attempts do a make a difference when CV is performed. Without CV, the number of iterations and attempts don't affect the test accuracy after a certain minimum value of these parameters is surpassed.

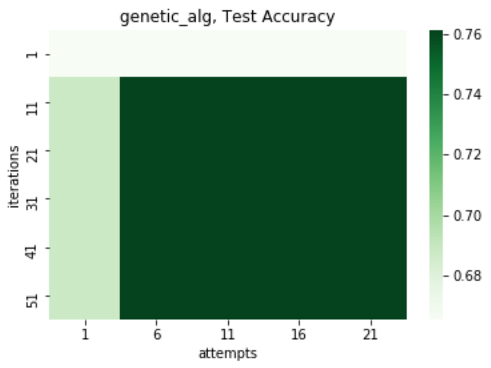









genetic_alg, CV Test Accuracy
iterations
1
11
21
31
41
51
attempts
1
6
11
16
21
0.56
0.52
0.48
0.44
0.40


genetic_alg, Test Accuracy
iterations
1
11
21
31
41
51
attempts
1
6
11
16
21
0.76
0.74
0.72
0.70
0.68