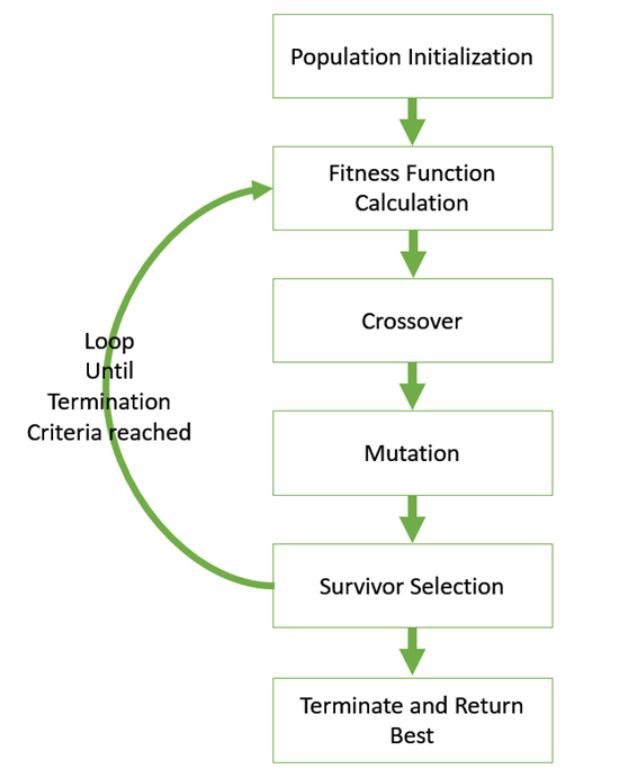
### Feature Selection using Genetic Algorithm



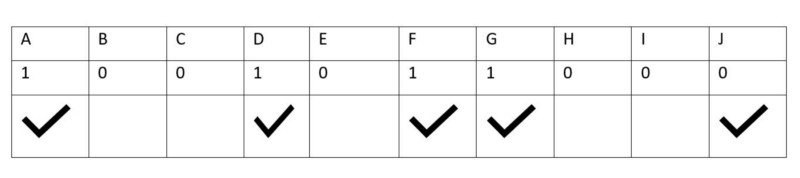
When we see this picture, the first thing that comes to our eyes is the selected man and second thought is what characters make this man stand out. Maybe his physical attractions like happiness, stylish, height, etc. or non-physical attractions like experience, education, income have been greatly played pivotal roles in his selection process. When it comes to the Supervised Machine Learning world, finding the best model and the strongest predictors of the response variable among hundreds of possibilities is an Art. **Feature reduction** or input selection, the process of finding the most relevant inputs for a model, helps us do this artistic job. By narrowing down the possible models, it enables us to build the best model with less time and cheaper computation. In this article, I will touch the **Genetic Algorithm** as a feature reduction way as it is getting a lot of attention in Machine Learning these days. It is an algorithm inspired by **Charles Darwin’s**’ theory of natural evolution. The evolution theory enables us to find the best model as we build a better model while going forward in the process. In this process, better models have higher chances to become parents and pass down their genetic information and worse models stand on the other side. i.e. they have lower chances to survive. To apply the Genetic Algorithm, we need to follow this flow chart.

Image credit: [Genetic Algorithm-Introduction](https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_quick_guide.htm)

The question is that how the Genetic Algorithm helps us detect the best model. Hypothetically speaking, Let’s suppose our dataset consists of 10 predictors, and its response variable is numerical. Hence, to build our model, we need a supervised Machine Learning algorithm and a regressor. Let’s dig into it to explore how we should find the best model using the Genetic Algorithm.

### 1-Population initialization:

In the first step, we create a set of n chromosomes. The n is representative of the initial population size. Each chromosome contains a set of m genes representing zero or one. The ones show the features are getting selected and zeros not. In this project, n and m correspondences initial population and number of features respectively. To better understand this part, let’s suppose the predictors are alphabetic letters from A to J in the order. Therefore, we need to generate a chromosome including 10 genes. Imagine our created chromosome is as follows.

Created by author

Regarding the table above, the first row represents the name of predictors, the second one shows our chromosome and the third one says which predictors consider as a part of the input dataset for training the model. Based on the mentioned chromosome, we will select predictors A, D, F, G, and J as features.

Let’s assume our initial population size is 20. Therefore, by end of this process, we have an initial population including 20 chromosomes that each one consists of 10 genes.

### 2- Fitness function:

This function helps us to explore which chromosomes get higher chances to become parents. The chance of reproduction is based on the fitness score. The fitness **score** can be any regression metrics such as MSE, RMSE, SSE, RSquard, and so on. Therefore, the lower the fitness score an individual chromosome catches, the higher probability it has to be selected. The ultimate goal of this function is to detect parents for the **cross over** or **mutation** steps. How likely cross over or mutation contributes to reproduction is an assumption that we need to make. Having said that, based on the genetic transformation nature in the real world, the former and latter usually gets higher and lower rate respectively.

In our case, by end of this process, as the initial population, we have 20 chromosomes with different chances to get selected. Also, Adjusted RSquared is the fitness metric.

### 3- Making a Cross Over:

Cross over is one of the two reproduction ways. In this way, we select two parents, mom, and dad, to have them passing down their genetic information to their child. How likely the child accepts mom or dad’s genes is the same. As a result, we have a child chromosome with the same number of genes as the parents do. In our case, the child chromosome must have 10 genes.

### 4- Making a Mutation:

As I mentioned before, the probability of occurrence of this reproduction way is usually pretty low. In this part, we need only one parent. To pass down the genetic information of the parent, two options are representing a gene either gets mutated or not. The opportunity of occurrence for the former option is occasionally much lower than the latter option.

### 5- Survivor Selection:

After creating children’s generation, we give them different chances to become parents for the next generation. Hence, the fitness function comes to play again. The parent generation will give birth to children generation by conducting **cross over** and **mutation**. For instance, in our case, after passing down the genetic information of parent(s) by using reproduction ways, by end of this step, we have 20 children chromosomes with different probabilities to get selected.

### When does this process stop?

This iterative process continues until the **termination criteria** reach. They are various case by case. In this project, the **number of consecutive best models** and the **maximum number of iterations** can be considered as stopping metrics. To meet the first stopping metric, we pick the best model after creating each generation and compare it with the previous one. if the best model doesn’t change for a defined number of consecutive, we reach out to our stopping criterion. For the maximum number of iterations, we allow the algorithm to run with limitations. Either stopping metric happens first, we will terminate the algorithm. To shed the light on the mentioned process, the Pseudocode is as follows.

#### START

Generate the initial population

Consider it as the current generation

Compute the fitness score

#### Repeat

Selection

Crossover/Mutation

Compute fitness score

Update the current generation

#### Until termination criteria reach

#### STOP

Finally, we can find the best model with the lowest SSE.

### Conclusion

Input selection using the genetic algorithm is becoming a heated arguable topic in Machine Learning among scholars. When a dataset contains a large number of features, many data Scientists use it as a facilitator of finding the most useful ones. Especially, when the neural network comes to play, the genetic algorithm shows its importance more. Although its advantages carry more weight, there are some disadvantages as well. The pros and cons are the following.

#### Pros

1. It usually outperforms over traditional raw feature selections such as step-wise backward/forward
2. To train the model, it does not need specific knowledge about the problem under study
3. It can handle datasets with many features

#### Cons

1. It might be computationally expensive since it needs to evaluate each individual requires the training of a model
2. In terms of converging, it takes time due to its stochastic nature

In short, the genetic algorithm is useful for finding the best variables, but to find the perfect model, more computation is necessary.

Thank you for reading this tutorial, and hope it was helpful for you. I am Sina Shariati, a data-savvy business analyst from San Francisco. You can find all the codes mentioned here on my Github page. If you have any questions/comments/suggestions, please feel free to reach out to me on [my LinkedIn account](https://www.linkedin.com/in/sina-shariati-5a26227a/).

https://github.com/sinashs/Tune-MCA-PCA.git