CLONALG for Credit Card Fraud Detection and Dataset Optimisation

CSE3013 - Artificial Intelligence

Final Project Report

J Component

Submitted By:

Sanjit C K S - 18BCE0715

Under the guidance of

Dr. Vasantha W B

In partial fulfilment of the degree of B.tech

In

Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vellore - 632014, Tamil Nadu, India

School of Computer Science and Engineering

June 2020

Abstract

Credit card fraud is an ever prevalent issue that has been addressed and researched quite well in the recent past. Incorporating artificial intelligence to a system to detect and handle fraudulent credit card information has great importance. This is an issue of great weight since any false negatives bring with it serious implications. In an attempt to provide a solution to this problem, in this project it has been attempted to used CLONALG which is an Artificial Immune System (AIS) algorithm emulates the immune response of the body to address pattern matching and optimisation. In this attempt, a sample dataset has been created and simulated with a modified version of the original CLONALG to provide adaptability to the dataset so that it is kept up conformed to the latest data.

Introduction

CLONALG is an Artificial Immune System Algorithm (AIS) that is slightly different from the usual genetic algorithms. That is, normal genetic algorithms use cross-over and mutation to create successful generation and bring in variance. In the case of AIS, the algorithms simulate asexual reproduction of cells and create a population of clones that are mutated for variance. CLONALG is one such AIS algorithm that has great patten matching and optimisation capabilities.

The credit industry - banking, general finance, credit card, retail etc. is huge. One really key application is in the detection of credit card fraud transactions. To provide citizens with the feature of online transactions, all major credit firms do a credit risk analysis. This is an extremely crucial and slightly complex step as it has to take into account personal and public credit history. When transactions are initiated there are a number of key feature data that are available to the credit card company, that enables it to check and affirm that the transaction is valid and has no malicious intent behind it.

In this project, 14 of such parameters have been decided on and used to create a simulated dataset and enable the proposed model to check and validate the transaction. It is essentially a classification algorithm that not only classifies unlabelled credit card data but more importantly optimises the dataset such that it stays relevant to new data. It is very interesting to note that as time progresses, fraudsters who attempt fraudulent credit card transaction get innovative. When this happens if, the dataset is static over time then, traditional classification models will fail inevitably.

CLONALG on the other hand, dynamically updates the dataset according to incoming data that is also labelled. This ensures that the model for credit card fraud detection using CLONALG will stay relevant to transaction data over long periods of time. This project model was tested with Python 3.7.3.

Keywords

Artificial Intelligence, Machine Learning, Artificial Immune System, CLONALG, Clonal Selection, Transaction Fraud Detection, Hybrid Algorithmic Approach.

Literature Review

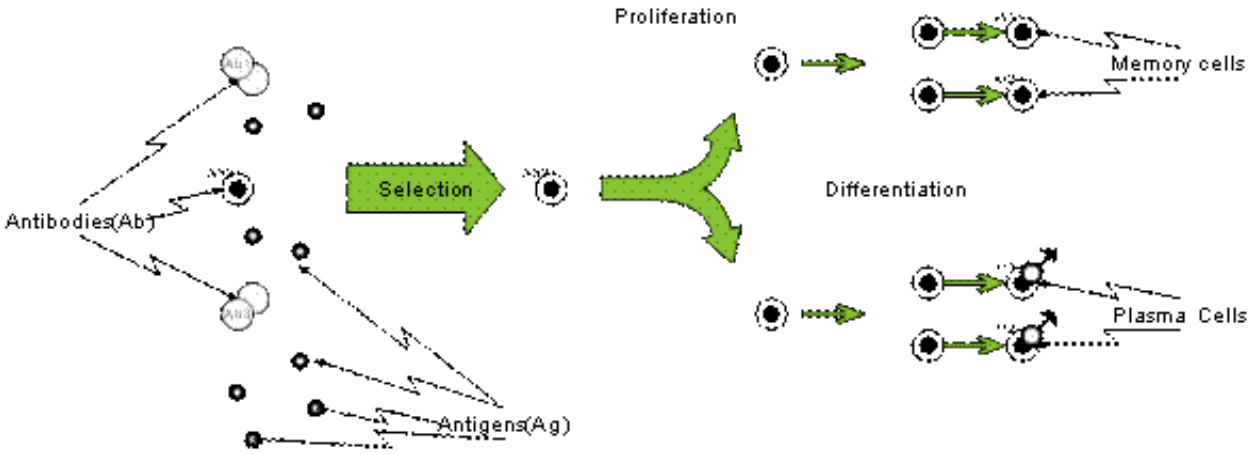
Author and Year	Title	Concept/ Model/ Framework	Method/ Implementation	Dataset	Finding	Limitation
Antonio Nascimento et al. 2012	An Experimental Investigation of Artificial Investigation Algorithms	Artificial Immune System (AIS) and CLONALG.	Mimicking human biological b-cells in immune response with CLONALG with kNN (replacing Euclidean Dist)	UCI Repository Dataset - Australian and German Dataset. Partitioned into training and independent test sets.	Improved CLONALG performance via efficiency at Peak Region	Loss of performance due to kNN algorithm after peak.
Pawan Kumar and Fahad Iqbal 2019	Credit Card Fraud Identification using machine learning approaches	Comparative Analysis 10 different ML techniques with the same dataset.	Implemented different ML techniques for the same data set with some visualisation.	2 days of credit card transactions (mastercard) - anonymised data.	Isolation Forest Model outperforms Local Outlier Factor.	Only basic ML algorithms are compared without sufficient implementational evidence.
Sahil Dhankhad and Emad A. Mohammed 2018	Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction: A Comparative Study	Sampling method is used to balance data and application of supervised learning.	Use of kNN, Random Forest, Gradient boosted tree classification accuracy with regard to classifiers for unbalanced data	Balanced dataset from unbalanced dataset. European credit card holders data.	Comparison of accuracy, F1 score, TPR factor.	Increased experiments with larger datasets might be required.

Author and Year	Title	Concept/ Model/ Framework	Method/ Implementation	Dataset	Finding	Limitation
Roge´rio de Lemos et al. 2007	Immune-Inspired Adaptable Error Detection for Automated Teller Machines	Detection and prevention of ATM failure on local and remote/ network levels	Framework revolves around: 1. Application Domain 2. Representation 3. Affinity Measure 4. Immune Algorithm Thus, taking a layered approach.	*ATM-data-sets* were derived from the concatenation of preprocessed ATM log files generated by different ATMs located in a common geographical area	1. Detection of failure occurrences: with classification accuracy of the AISEC algorithm 2. The enhancement of availability: By assessing the local AED	Does not cater to rare events that maybe associated with elusive faults because they are primarily non recurring.
Yashwant Prasad Singh and Amir Babiker 2019	Modified Clonal Selection Algorithm Based Classifiers	True random memory pool generation to create unique antibodies during initialisation	Initialisation is done by iterating through all the attributes and assigning random values to them.	Iris and Ionosphere Dataset (from MATLAB)	Increases the diversity of the algorithm	Would suffer minor space-time trade off.

Proposed Work and Implementation

Methodology

Here we shall discuss about how the algorithm imparts *adaptability* while classifying the credit card data. The input to this algorithm is a simulated dataset with multiple input vectors or *antigen* vectors as we call them. Clonal Selection is an immunological process in the body that, occurs in B-Lypmphocytes to counter incoming Pathogens (‘antigens’ or ‘non-self cells’ or ‘Ag’). They do so by creating *Antibodies or Ab* that can counter specific antigen. Simply, when Ag’s attacks the body, immune cells (B lymphocytes) are responding by producing a specific Ab’s for the attacking Ag’s. Ab’s are molecules attached primarily to the surface of B cells whose aim is to recognise and bind to Ag’s. A proliferation process will occur to the cells that recognised the attacking Ag’s producing two new types: attacking and memory cells. The attacking cells secrete a lot of effective Ab’s to eliminate the attacking Ag’s immediately. The memory cells have a long-life span in case of future exposures of the same or similar Ag’s, they can act faster and more effectively.



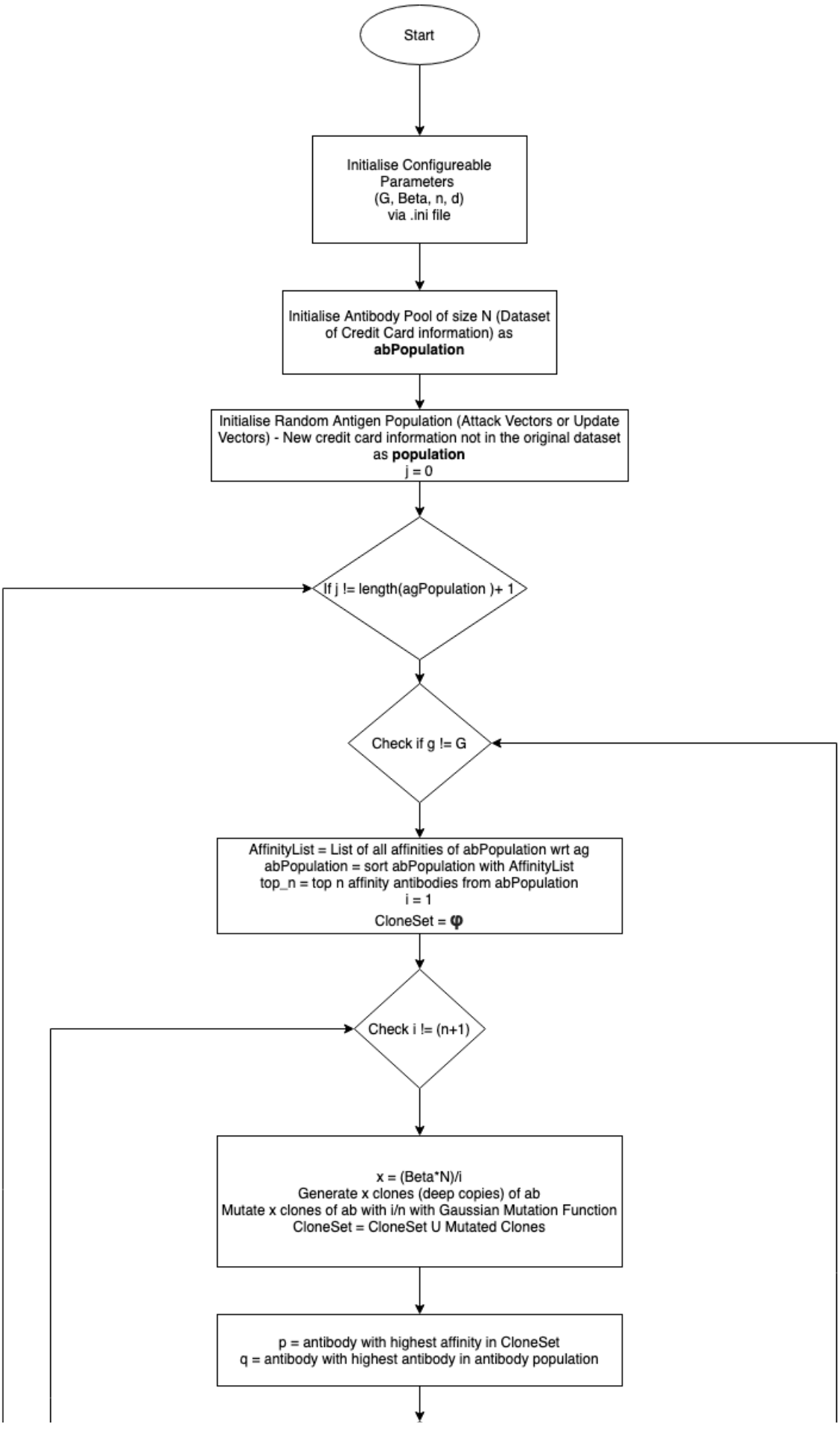
This immune response can be simulated for patten matching and optimisation. The input to this algorithm is a simulated dataset with multiple, unlabelled - input vectors or *antigen* vectors as we call them. As incoming Ag vectors approach the system, the system has a dataset of labelled vectors called Antibodies (Abs) that are used to classify the Ag Vector. While doing the the antibodies are cloned (‘copied’) and mutated (‘to generate variance’) that have better affinity to the current Ag. Therefore, the dataset would stay more relevant to the latest Ag vectors. This in the case of credit card transactions is of immense improvement because, the classification process would rely on a dataset that adapts whenever fraudsters adapt to newer technology.

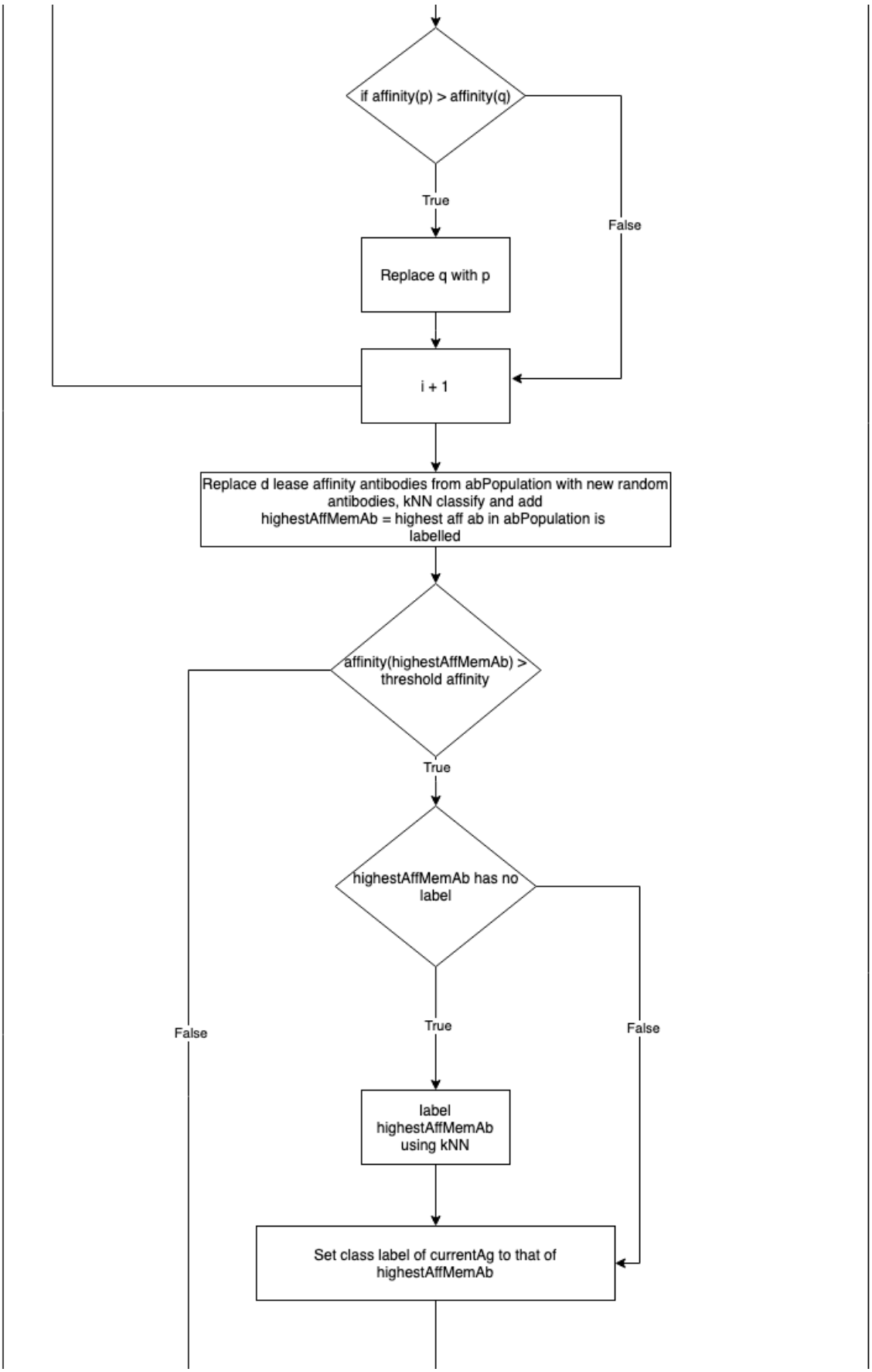
This proposed system is a hybrid model, that uses kNN to label data incoming data based on the updated dataset, according to affinity. The cloning and maturation of antibodies (the dataset) is done with the help of finding the affinity of Ag and Ab. Once the Antibodies are matured via mutation to have an affinity better than the threshold affinity, then the KNN algorithm classifies the Ag vector with the help of the updated dataset. The complete working of the algorithm is explained with the help of a flow chart following.

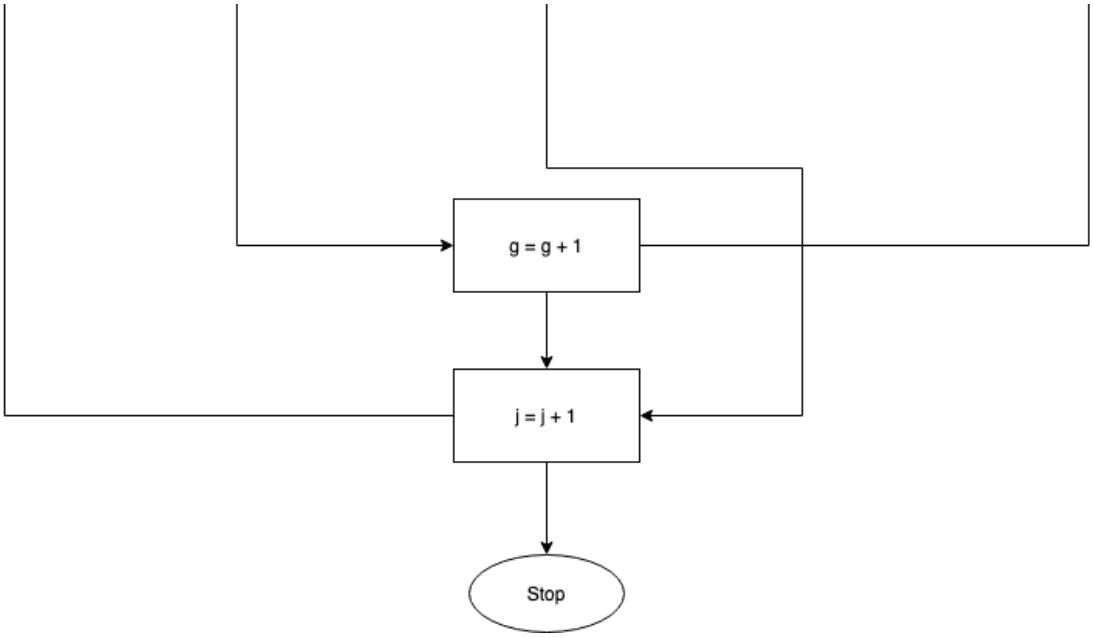
Algorithmic Working as Steps:

1. Input Parameter - G, beta, n, d, threshold_affinity are obtained (here from config parser)
2. The Input Vectors ('Antigen' or 'Ag') is sent one by one into hybrid CLONALG Model for Credit Card Detection. For each Ag:
    1. Next For a G number of generations the affinity maturation is done. For each generation
        1. For each generation, we calculate affinity of all the antibodies with respect to the current antigen (unlabelled data) using 'cosine' distance
        2. The antibody Population (dataset) is sorted in the order of affinity from greatest to least
        3. The top 'n' affinity antibodies are taken and for each antibody the following is done:
            1. Initialise or Reinitialise a Clone set as null set
            2. Get the number of Antibodies from the sorted pool
            3. $x = \frac{\beta * (antibodyPopulationSize)}{iterationNumber}$
            4. Generate x number of clones
            5. Mutate the x clones with gaussian Mutation Rate i/n
            6. CS = CS Union Mutated Clones
            7. Take 'P' as highest affinity Ab from CS
            8. Take 'Q' as highest affinity Ab from Antibody Population (in Memory - Dataset)
            9. If affinity(P) > affinity(Q), then replace Q with P
        4. Check if highest affinity Ab in Antibody Population has affinity greater than threshold affinity. If so then check if the highest affinity Ab has label (from original dataset) or or has no label (updated with clone).
        5. If labeled then, label the current antigen with highest affinity Ab label. Or if no label, label the Ab with KNN with the existing dataset and use that label the current antigen.
        6. If more antigen vectors exist then, go to next antigen.

Flowchart of Working




if affinity(p) > affinity(q)
True
False
Replace q with p
i + 1
Replace d lease affinity antibodies from abPopulation with new random antibodies, kNN classify and add highestAffMemAb = highest aff ab in abPopulation is labelled
affinity(highestAffMemAb) > threshold affinity
True
False
highestAffMemAb has no label
True
False
label highestAffMemAb using kNN
Set class label of currentAg to that of highestAffMemAb



Design

Key features of the design of this Project are

1. Object Oriented Programming Code
2. Hybrid Artificial Immune System for Classification and Optimisation
3. Scalable and Configuration adjustable

A hybrid model classification and optimisation model that identifies fraud transaction data and optimises the dataset according to recent classifications. A few **advantages** of the design are,

1. Relevancy of the model to present transactions scenario will always be high.
2. The model is theoretically scalable and will work in every scenario with enough hardware resources
3. Artificial Immune System approach has been used.
4. Cosine Distance is used to calculate affinity (here lesser the value better the affinity)

Although there are a lot of ups to this implementation, there is a limitation too. It hasn't slipped my notice that this CLONALG is not very optimal in the aspect that all clones but the best one are discarded even though that generation might have very high affinity clones which makes it less than optimal. It has scope for improvement in the area where, instead of replacing the best clone, we can replace multiple best clones while also finding a way to address the issue of not getting stuck on local maxima.

Implementation

The complete Implementation of the Project can be found in GitHub - https://github.com/sanjitk7/ClonalAlgorithmAI

Key features of implementation of this Project are

1. The project has been programmed in complete Object Oriented Programming in Python. All antigen and antibodies are classes with getter and setter method for

attribute initialisation and updation. Moreover, unique uuids (universal unique identifiers) have been used to track each and every antigen or antibody or antibody clone that has been created and deleted.
2. The implementation for CLONALG is done from scratch without using any underlying libraries or modules. The KNN classifier uses sci kit learn library.
3. All parameters are configureable using Config Parser. The config.ini file is responsible for this feature.
4. All functions are tested using unnittest module for functional working.
5. The algorithm has been implemented with utmost modularity that has enabled proper testing.
6. All stages of the algorithm have been properly tested and logged.

Software Requirement

The complete algorithm implementation has been tested on python 3.7.3 on MacOS Catalina Version 10.15.4.

Further Dependancies for the algorithm implementation include:
1. pandas - For data vectors handling
2. DEAP (Distributed Evolutionary Algorithms in Python) - Is used ONLY for gaussian mutation and generation of randomness.
3. sci-kit learn - For KNN Classifier
4. uuid - for unique identification of all cells (antigen and antibodies)
5. unittest - test working of functions
6. configparser - to test algorithm with different configurations

Innovation

The issue of credit card fraud detection has been addressed before by a lot of simpler one-tiered and non-hybrid approaches. They use normal supervised machine learning algorithms to only classify the fraud transaction with the help of a static dataset. None of the earlier work has attempted to use CLONALG for this application. This approach is simplistic and cannot be relied on for a long period of time where the characteristics of fraudulent transaction can change over time because fraudsters change their tactics over time.

The key innovation in this project is the development of a ‘hybrid’ algorithm, that uses CLONALG and KNN together to not just classify the fraud transactions but also update the dataset accordingly. The adaptability and flexibility that the CLONALG provides the system is a major development. The relevancy that the dataset would always posses due to the dynamic optimisation of Antibodies is another key innovation.

Dataset

For the purpose of this project can I have generated a dummy dataset in the format of CSV with each row as antigen vectors. The credit card transaction attributes selected and initialised include scores and bools for:

1. Transaction ID
2. Indiscriminate Purchase Score - Purchase total is substantially greater than your average customer
3. Relative Purchase Total Score - Sudden Extremely High Purchase Value
4. Purchase Total Relative Score - Sudden unusual number of expensive purchases
5. Card Present Status - Boolean for Card being physically present
6. Is Swipe or Chip - Boolean for if if the purchase was done by Swipe
7. Is Sign or Pin - Boolean for whether the bill was signed
8. Frequent Recent Purchase - Abnormally Frequent recent purchases
9. Easy Resale Items Score - Items that can be easily resold
10. Geographic distance Deviation - Geo Distance between place of purchase and Sale
11. Known IP - Boolean for if the IP is known or has history
12. Known MAC - Boolean for if the Device Fingerprint is authentic
13. Time of abnormality Score - Abnormality of purchase time
14. Geographic shipping deviation - Geo Distance between place of purchase and shipping
15. Known Browser - Boolean for verified browser
16. Label - Fraud(1) an NotFraud(0)

Results and Discussion:

For the Input Vectors (attackVectors.csv):

T_id	Indiscriminat	Purchase_tot	Expensive_ite	Card_presen	Swipe_or_chi	Sign_or_pin	Freq_recent_	Easy_resale_i	Geodist_devi	Known_ip	Known_mac	Time_abnorm	Geodist_ship	Known_brow	Fraud_label
12	12	29	2	0	0	0	43	70	80	1	1	10	40	0	-1
13	50	5	4	0	1	1	1	10	10	1	0	1	30	0	-1
14	90	60	30	0	0	1	40	80	50	1	0	50	30	-1	

With part of the the simulated dataset (kNNcardDataset.csv):

T_id	Indiscriminat	Purchase_tot	Expensive_ite	Card_presen	Swipe_or_chi	Sign_or_pin	Freq_recent_	Easy_resale_i	Geodist_devi	Known_ip	Known_mac	Time_abnorm	Geodist_ship	Known_brow	Fraud_label
1	10	1	2	0	0	0	1	5	30	1	0	1	10	1	0
2	22	5	4	0	0	0	1	10	20	0	0	1	20	0	0
3	33	12	1	1	0	1	90	12	1	1	1	23	1	1	1
4	42	33	5	0	0	0	45	2	40	1	1	53	20	1	0
5	54	70	16	1	1	1	22	2	1	1	0	75	1	1	1
6	69	11	30	1	0	0	82	50	1	0	1	90	1	1	1
7	74	41	1	0	0	0	10	20	1	1	0	12	83	1	0
8	12	29	2	0	0	0	43	70	80	1	1	10	40	0	1
9	45	47	2	1	1	1	20	5	1	1	0	31	1	1	0
10	21	8	3	0	0	0	15	67	1	0	1	70	51	0	0

For the General Configuration -> G = 20, n=2, beta=0.5, d=0, threshold_affinity=0.2
The algorithm works in the following manner for ONE generation. The following output shows logged states of each generation.

```
Generation # 4

Ag Population Size:
 3

Ag #:
 1

Affinity List:  [0.6256283002328499, 0.975268568605484, 0.7740643489680225, 0.9792254681763572, 0.9883717156796216, 0.7491677117990234, 0.6572817365811853, 0.9670192685799589, 0.8339267758948815, 0.5111793920785141]

yx before sort: [(0.6256283002328499, <antibody.Antibody object at 0x11c0cef60>), (0.975268568605484, <antibody.Antibody object at 0x11c6a3128>), (0.7740643489680225, <antibody.Antibody object at 0x11c6a3240>), (0.9792254681763572, <antibody.Antibody object at 0x11c6a3278>), (0.9883717156796216, <antibody.Antibody object at 0x11c6a32e8>), (0.7491677117990234, <antibody.Antibody object at 0x11c6a3358>), (0.6572817365811853, <antibody.Antibody object at 0x11c6a33c8>), (0.9670192685799589, <antibody.Antibody object at 0x11c6a3438>), (0.8339267758948815, <antibody.Antibody object at 0x11c6a34a8>), (0.5111793920785141, <antibody.Antibody object at 0x11c6a37b8>)]

yx after sort: [(0.5111793920785141, <antibody.Antibody object at 0x11c6a37b8>), (0.6256283002328499, <antibody.Antibody object at 0x11c0cef60>), (0.6572817365811853, <antibody.Antibody object at 0x11c6a33c8>), (0.7491677117990234, <antibody.Antibody object at 0x11c6a3358>), (0.7740643489680225, <antibody.Antibody object at 0x11c6a3240>), (0.8339267758948815, <antibody.Antibody object at 0x11c6a34a8>), (0.9670192685799589, <antibody.Antibody object at 0x11c6a3438>), (0.975268568605484, <antibody.Antibody object at 0x11c6a3128>), (0.9792254681763572, <antibody.Antibody object at 0x11c6a3278>), (0.9883717156796216, <antibody.Antibody object at 0x11c6a32e8>)]

Sorted AntiBodies:  [<antibody.Antibody object at 0x11c6a37b8>, <antibody.Antibody object at 0x11c0cef60>, <antibody.Antibody object at 0x11c6a33c8>, <antibody.Antibody object at 0x11c6a3358>, <antibody.Antibody object at 0x11c6a3240>, <antibody.Antibody object at 0x11c6a34a8>, <antibody.Antibody object at 0x11c6a3438>, <antibody.Antibody object at 0x11c6a3128>, <antibody.Antibody object at 0x11c6a3278>, <antibody.Antibody object at 0x11c6a32e8>]

Current clones after cloning:
 [<antibody.Antibody object at 0x11c284eb8>, <antibody.Antibody object at 0x11c293390>, <antibody.Antibody object at 0x11c6a3860>, <antibody.Antibody object at 0x11c6a39b0>, <antibody.Antibody object at 0x11c6a3940>]

Mutated Clones (current clone set):
 [<antibody.Antibody object at 0x11c284eb8>, <antibody.Antibody object at 0x11c293390>, <antibody.Antibody object at 0x11c6a3860>, <antibody.Antibody object at 0x11c6a39b0>, <antibody.Antibody object at 0x11c6a3940>]

Highest Affinity Clone (Affinity =  0.5110324539081772 ) in current clone set:
 71b8d7e1-e6a2-43c4-950b-1ccc3647c813:tid:1:[ INDP:5.2416057044347735,PTRS:4.004964302197548,EIS:-1.4853293555206224,CPS:1,ISOC:1,ISOP:1,FRPS:4.092073804144377,ERIS:7.367924765811882,GDF:28.829052481211974,KIP:1,KMAC:0,TAS:-3.3564328841221713,GSD:12.42404051976249,KBR:1]: FL:-1

Highest Affinity Antibody (Affinity =  0.5111793920785141 ) from Memorty Pool:
 3935c997-2093-48e1-945f-ac67fc1972c3:tid:1:[ INDP:5.443396580168872,PTRS:4.194636789761 36,EIS:-0.9249385601182394,CPS:0,ISOC:1,ISOP:1,FRPS:4.0824603607562775,ERIS:7.009004353213759,GDF:28.980317285099915,KIP:1,KMAC:1,TAS:-3.325960552985914,GSD:12.325437805689944,KBR:1]: FL:-1

****REPLACEMENT*****

Replaced Antibody from Memory Pool:
 3935c997-2093-48e1-945f-ac67fc1972c3:tid:1:[ INDP:5.443396580168872,PTRS:4.194636789761 36,EIS:-0.9249385601182394,CPS:0,ISOC:1,ISOP:1,FRPS:4.0824603607562775,ERIS:7.009004353213759,GDF:28.980317285099915,KIP:1,KMAC:1,TAS:-3.325960552985914,GSD:12.325437805689944,KBR:1]: FL:-1

Replaced with clone:
 71b8d7e1-e6a2-43c4-950b-1ccc3647c813:tid:1:[ INDP:5.2416057044347735,PTRS:4.004964302197548,EIS:-1.4853293555206224,CPS:1,ISOC:1,ISOP:1,FRPS:4.092073804144377,ERIS:7.367924765811882,GDF:28.829052481211974,KIP:1,KMAC:0,TAS:-3.3564328841221713,GSD:12.42404051976249,KBR:1]: FL:-1

Current clones after cloning:
 [<antibody.Antibody object at 0x11c284eb8>, <antibody.Antibody object at 0x11c293390>, <antibody.Antibody object at 0x11c6a3860>, <antibody.Antibody object at 0x11c6a39b0>, <antibody.Antibody object at 0x11c6a3940>, <antibody.Antibody object at 0x11c6a3ac8>, <antibody.Antibody object at 0x11c6a37f0>]

Mutated Clones (current clone set):
 [<antibody.Antibody object at 0x11c284eb8>, <antibody.Antibody object at 0x11c293390>, <antibody.Antibody object at 0x11c6a3860>, <antibody.Antibody object at 0x11c6a39b0>, <antibody.Antibody object at 0x11c6a3940>, <antibody.Antibody object at 0x11c284eb8>, <antibody.Antibody object at 0x11c293390>, <antibody.Antibody object at 0x11c6a3860>, <antibody.Antibody object at 0x11c6a39b0>, <antibody.Antibody object at 0x11c6a3940>, <antibody.Antibody object at 0x11c6a3ac8>, <antibody.Antibody object at 0x11c6a37f0>]

Highest Affinity Clone (Affinity =  0.5115667867948985 ) in current clone set:
 1a842ded-2717-4817-a2d8-96a00b2a426c:tid:1:[ INDP:4.995579899448583,PTRS:3.4359912065537244,EIS:-0.4373684917859878,CPS:1,ISOC:1,ISOP:1,FRPS:4.86585296652332 66,ERIS:6.299589833386234,GDF:28.757655682971993,KIP:1,KMAC:1,TAS:-4.478356727859667,GSD:11.971668129997324,KBR:0]: FL:-1

Highest Affinity Antibody (Affinity =  0.5629762827768647 ) from Memorty Pool:
 71b8d7e1-e6a2-43c4-950b-1ccc3647c813:tid:1:[ INDP:5.420911282228565,PTRS:3.9099895053360 52,EIS:-1.9528912919485322,CPS:0,ISOC:0,ISOP:0,FRPS:2.986542409401736,ERIS:7.4924100958210795,GDF:28.86071046052198,KIP:0,KMAC:0,TAS:-3.780946119986249,GSD:12.410737535139724,KBR:0]: FL:-1
```

Here we can see the replacement of new better affinity mutated clones replacing the top affinity Memory Antibody from the dataset. This replacement happens iteratively for top ’n’ number of Antibodies as a process in ***affinity maturation.*** (In both the image above and below)

In the output image below, we notice that the affinity has matured to reach threshold affinity value and is therefore labelled with the best antibody.

Finally, in the list of labelled vectors we have, the **FL** variable shows the value of fraud_label. Which here is shows whether the incoming data is fraudulent or not.

```
%%%%%%%OUTPUT%%%%%%%%%%
The Labelled Antigen Are:

Antigen Item #  0  : 53f4ba3f-623b-4ae3-8648-e88dd186f878:tid:12.0:[ INDP:12.0,PTRS:29.0,EIS:2.0,CPS:0.0,ISOC:0.
0,ISOP:0.0,FRPS:43.0,ERIS:70.0,GDF:80.0,KIP:1.0,KMAC:1.0,TAS:10.0,GSD:40.0,KBR:0.0]: FL:1
Antigen Item #  1  : 7a29da43-e49d-4fe6-9e3a-813b55e91114:tid:13.0:[ INDP:50.0,PTRS:5.0,EIS:4.0,CPS:0.0,ISOC:1.0
,ISOP:1.0,FRPS:1.0,ERIS:10.0,GDF:10.0,KIP:1.0,KMAC:0.0,TAS:1.0,GSD:30.0,KBR:0.0]: FL:0
Antigen Item #  2  : 688c1a37-2a01-4e31-9f9e-b4a5a535f8b1:tid:14.0:[ INDP:90.0,PTRS:60.0,EIS:30.0,CPS:0.0,ISOC:0
.0,ISOP:1.0,FRPS:40.0,ERIS:80.0,GDF:50.0,KIP:1.0,KMAC:0.0,TAS:50.0,GSD:30.0,KBR:-1.0]: FL:0
(ClonalAlgorithmAI) sanjitkumar@Sanjits-Air ClonalAlgorithmAI %
```

Also after the algorithm was run the Antibody Pool, that is our dataset was update and optimised dynamically as follows

As we are able to notice the last to antibodies have been optimised to recognise vectors

```
%%%%%%%The Antibody Pool after the Algorithm%%%%%%%

0e2936f3-d4aa-4e5e-97e7-556fc02b7d84:tid:1:[ INDP:10,PTRS:1,EIS:2,CPS:0,ISOC:0,ISOP:0,FRPS:1,ERIS:5,GDF:30,KIP:
1,KMAC:0,TAS:1,GSD:10,KBR:1]: FL:0
62a51957-17fc-45f2-a313-a98e616698ee:tid:2:[ INDP:22,PTRS:5,EIS:4,CPS:0,ISOC:0,ISOP:0,FRPS:1,ERIS:10,GDF:20,KIP
:0,KMAC:0,TAS:1,GSD:20,KBR:0]: FL:0
c9e57318-c89b-4bd1-b9ba-d7b72b3386cc:tid:3:[ INDP:33,PTRS:12,EIS:1,CPS:1,ISOC:0,ISOP:1,FRPS:90,ERIS:12,GDF:1,KI
P:1,KMAC:1,TAS:23,GSD:1,KBR:1]: FL:1
3e6e9dd7-fb46-4c9c-96f8-b581fb2c5877:tid:5:[ INDP:54,PTRS:70,EIS:16,CPS:1,ISOC:1,ISOP:1,FRPS:22,ERIS:2,GDF:1,KI
P:1,KMAC:0,TAS:75,GSD:1,KBR:1]: FL:1
800fab68-9069-4696-aeba-9353933f8916:tid:6:[ INDP:69,PTRS:11,EIS:30,CPS:1,ISOC:0,ISOP:0,FRPS:82,ERIS:50,GDF:1,K
IP:0,KMAC:1,TAS:90,GSD:1,KBR:1]: FL:1
850bdd3b-7831-4d2d-a4f5-56ac8886a7b7:tid:8:[ INDP:12,PTRS:29,EIS:2,CPS:0,ISOC:0,ISOP:0,FRPS:43,ERIS:70,GDF:80,K
IP:1,KMAC:1,TAS:10,GSD:40,KBR:0]: FL:1
cccc90e9-3354-462a-9c44-f4fd76327612:tid:9:[ INDP:45,PTRS:47,EIS:2,CPS:1,ISOC:1,ISOP:1,FRPS:20,ERIS:5,GDF:1,KIP
:1,KMAC:0,TAS:31,GSD:1,KBR:1]: FL:0
6ea0150b-7909-4248-be11-203a9f53c318:tid:10:[ INDP:21,PTRS:8,EIS:3,CPS:0,ISOC:0,ISOP:0,FRPS:15,ERIS:67,GDF:1,KI
P:0,KMAC:1,TAS:70,GSD:51,KBR:0]: FL:0
f539b73e-981b-46d0-b3cf-758dd6c79df7:tid:2:[ INDP:24.326005109057185,PTRS:5.778308367077161,EIS:3.6890955397823
566,CPS:0,ISOC:0,ISOP:1,FRPS:1.8063658056379897,ERIS:9.2730835598143,GDF:19.722543167820508,KIP:0,KMAC:0,TAS:1.
951842761381497,GSD:20.416479711434018,KBR:1]: FL:-1
a9b92933-120f-4f0a-9dcb-20f29fe41cc9:tid:4:[ INDP:49.44511249612718,PTRS:32.98188508322603,EIS:12.3701786087123
93,CPS:0,ISOC:1,ISOP:0,FRPS:45.388302084544094,ERIS:18.309542363947966,GDF:36.79648092690222,KIP:0,KMAC:1,TAS:5
3.65885973060107,GSD:18.40367113017232,KBR:0]: FL:-1
(ClonalAlgorithmAI) sanjitkumar@Sanjits-Air ClonalAlgorithmAI %
```

similar to our 3 Antigen Vectors in the future. So fraud transactions that have happened in the recent past serve as great reference and help classify future similar antigen quickly and efficiently. This enables the model to stay dynamically relevant in real time.

Conclusion and Future Scope

In in this attempt to address credit card fraud detection using CLONALG we have created a completely authentic hybrid algorithm that has the capability to classify fraud data vectors and also optimise the dataset to make it stay relevant to newer fraud tactics as time progresses. The algorithm has potential in the sense that it has attempted to increase the complexity of the solution offered for classical credit transactions. The implementation was done in a highly optimal Object Oriented Programming with elegant code that takes advantage of modular programming using unit tests for Test Driven Development.

The project is very interesting and has a lot of scope because of the flexibility it offers the system. But there is room for improvement, in the area of antibody replacement, there can be replacement of more than one antibody in one iteration as the discarding of antibodies created means wasted resources. This has to be done with ensuring that the algorithm does not get stuck on a local maxima for replacing high affinity antibodies. Over the course of the project I have learnt a lot and explored a new area of AI that is of great interest.

References

[1] A. I. S. Nascimento and G. C. Vasconcelos, "An experimental investigation of artificial immune system algorithms for credit risk assessment applications," 2012 IEEE Congress on Evolutionary Computation, Brisbane, QLD, 2012, pp. 1-8, doi: 10.1109/CEC.2012.6252947.

[2] P. Kumar and F. Iqbal, "Credit Card Fraud Identification Using Machine Learning Approaches," 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), CHENNAI, India, 2019, pp. 1-4, doi: 10.1109/ICIICT1.2019.8741490.

[3] S. Dhankhad, E. Mohammed and B. Far, "Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study," 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, 2018, pp. 122-125, doi: 10.1109/IRI.2018.00025.

[4] R. de Lemos, J. Timmis, M. Ayara and S. Forrest, "Immune-Inspired Adaptable Error Detection for Automated Teller Machines," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 5, pp. 873-886, Sept. 2007, doi: 10.1109/TSMCC.2007.900662.

[5] Y. P. Singh and A. S. H. Babiker, "Modified Clonal Selection Algorithm Based Classifiers," 2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications, Penang, 2011, pp. 108-113, doi: 10.1109/BIC-TA.2011.13.