Master's thesis

Applications of Evolutionary Algorithms in climate projections

Kanan Jafarli

kanan.jafarli@ufaz.az

Data Science and Artificial Intelligence

Supervisors **Prof. Rodrigo Abarca Del Rio**, **Prof. Pierre Collet**

and

Dr. Ulviyya Abdulkarimova

25 February - 25 June

BAKU 2022

Abstract

The global climate is a chaotic and complex system. The climate system constantly changes due to internal variability and natural external perturbation. The earth's axial rotation causes daily changes in temperature, precipitation, cloud cover, humidity and other climate variables. Over short time scales (ranging from minutes to several days) movement of clouds, air masses, and fronts contribute to rapid, often unpredictable, changes in weather. The earth's annual rotation about the sun causes large seasonal changes in climate. Multi-annual to decade mode transitions such as the El Nino and La Nina cycles in the Pacific cause additional variability in the climate system. Therefore, determining the global effects of ENSO occurrences is considered one of the important issues.

In this thesis, applications of Machine Learning and Evolutionary Algorithms on the global climate are presented. El Nino and La Nina events are classified according to Oceanic Nino Index values using Machine Learning classification methods and are also classified according to their shapes by eye and automatically by ML clustering methods. The equations are found for events using EAs where Genetic Programming is used to find average equations for groups and Real Valued Genetic Algorithm is used to find equation of each event with changing parameters of average equation. These algorithms are worked on island model in GPU programming to reduce time computation. Using equations, geophysicists and meteorologists can better understand climatic phenomena and make new analysis on these phenomena.

Keywords : Machine Learning, Classification algorithms, Clustering algorithms, Evolutionary Algorithms, Genetic Programming, Real Valued Genetic Algorithm, GPU programming.

Referat

Qlobal iqlim xaotik və mürəkkəb sistemdir. İqlim sistemi daxili dəyişkənlik və təbii xarici təlaşa görə daim dəyişir. Yerin eksenel fırlanması temperatur, yağıntı, bulud örtüyü, rütubət və digər iqlim dəyişənlərində gündəlik dəyişikliklərə səbəb olur. Qısa müddət ərzində (dəqiqələrdən bir neçə günə qədər) buludların, hava kütlələrinin və cəbhələrin hərəkəti havanın sürətli, tez-tez gözlənilməz dəyişikliklərinə səbəb olur. Yerin günəş ətrafında illik fırlanması iqlimdə böyük mövsümi dəyişikliklərə səbəb olur. Sakit okeanda El Nino və La Nina dövrləri kimi çoxillikdən onillik rejimə keçidlər iqlim sistemində əlavə dəyişkənliyə səbəb olur. Buna görə də ENSO hadisələrinin qlobal təsirlərinin müəyyən edilməsi mühüm məsələlərdən biri hesab olunur.

Bu tezisdə Maşın Öyrənməsi və Təkamül Alqoritmlərinin qlobal iqlimdə tətbiqləri təqdim olunur. El Nino və La Nina hadisələri Maşın Öyrənmə təsnifat metodlarından istifadə edərək Okean Nino İndeksi dəyərlərinə görə təsnif edilir və həmçinin formalarına görə göz və avtomatik olaraq ML klasterləşdirmə metodları ilə təsnif edilir. Tənliklər qruplar üçün orta tənlikləri tapmaq üçün Genetik Proqramlaşdırmadan və orta tənliyin dəyişən parametrləri ilə hər bir hadisənin tənliyini tapmaq üçün Real Qiymətləndirilmiş Genetik Alqoritmdən istifadə edilən EA-dan istifadə edilən hadisələr üçün tapılır. Bu alqoritmlər vaxt hesablamasını azaltmaq üçün GPU proqramlaşdırmasında ada rejimində işləyir. Tənliklərdən istifadə edərək, fiziklər və meteoroloqlar iqlim hadisələrini daha yaxşı başa düşə və bu hadisələrə dair yeni təhlillər apara bilərlər.

Açar sözlər : Maşın Öyrənməsi, Təsnifat alqoritmləri, Klasterləşdirmə alqoritmləri, Təkamül Alqoritmləri, Genetik Proqramlaşdırma, Real Qiymətli Genetik Alqoritm, GPU proqramlaşdırması.

Реферат

Глобальный климат представляет собой хаотичную и сложную систему. Климатическая система постоянно меняется из-за внутренней изменчивости и естественных внешних возмущений. Осевое вращение Земли вызывает ежедневные изменения температуры, осадков, облачности, влажности и других климатических переменных. В короткие промежутки времени (от минут до нескольких суток) движение облаков, воздушных масс и фронтов способствует быстрым, часто непредсказуемым изменениям погоды. Годовое вращение Земли вокруг Солнца вызывает большие сезонные изменения климата. Переходы от многолетних к десятилетним режимам, такие как циклы Эль-Ниньо и Ла-Нинья в Тихом океане, вызывают дополнительную изменчивость климатической системы. Поэтому определение глобальных последствий явлений ЭНЮК считается одним из важных вопросов.

В этой диссертации представлены приложения машинного обучения и эволюционных алгоритмов к глобальному климату. События Эль-Ниньо и Ла-Нинья классифицируются в соответствии со значениями индекса Oceanic Nino с использованием методов классификации машинного обучения, а также классифицируются в соответствии с их формой на глаз и автоматически с помощью методов кластеризации ML. Уравнения для событий находятся с использованием советников, где генетическое программирование используется для нахождения средних уравнений для групп, а генетический алгоритм с действительными значениями используется для нахождения уравнения каждого события с изменяющимися параметрами среднего уравнения. Эти алгоритмы работают в островном режиме при программировании графического процессора, чтобы сократить время вычислений. Используя уравнения, физики и метеорологи могут лучше понять климатические явления и провести новый анализ этих явлений.

Ключевые слова : Машинное обучение, алгоритмы классификации, алгоритмы кластеризации, эволюционные алгоритмы, генетическое программирование, генетический алгоритм с действительными значениями, программирование графического процессора.

Abbreviations

- **EA** — Evolutionary Algorithm
- **GP** — Genetic Programming
- **RVGA** — Real Valued Genetic Algorithm
- **ENSO** — El Nino–Southern Oscillation
- **ONI** — Oceanic Nino Index
- **ML** — Machine Learning
- **EASEA** — EAsy Specification of Evolutionary Algorithm
- **CUDA** — Compute Unified Device Architecture
- **GPU** — Graphics Processing Unit
- **CPU** — Central Processing Unit
- **SIMD** — Single Instruction Multiple Data
- **GPGPU** — General Purpose Graphic Processing Unit
- **Arithmetic and Logic Unit** — ALU
- **NOAA** — National Oceanic and Atmospheric Administration
- **SST** — Sea Surface Temperature
- **WE** — Weak El Nino
- **ME** — Moderate El Nino
- **SE** — Strong El Nino
- **VSE** — Very Strong El Nino

Contents

1 Introduction

The global climate is a chaotic and complex system that is influenced by many factors such as the oceans, the rotation of the earth, the land relief etc. To study this system, meteorology was created. In the 17^{th} century, meteorology therefore became a science, following the development of many tools allowing the measurement of temperature, humidity, pressure and the direction and speed of the wind. However, meteorology remained a minor branch of physics for a long time because compared to astronomy, which explained the world through precise equations and calculations, it highlighted a certain disorder, meteorologists having at their disposal rare and scattered.

Understanding the impact of climate systems on local and global economies [39], health and social repercussions [40] [41], and disaster risk reduction and vulnerability [42] on society is critical. The building of climate models, which involve data from several sources such as atmosphere measurements, land use, and carbon emission, among others, is a conventional strategy for researching climatic systems. However, for some places, obtaining and compiling these data is a challenge because it is time consuming and requires specific knowledge.

The aim of the study is to classify climate events El Nino, La Nina by shapes and to apply Evolutionary Algorithms (EAS) to find simple and good equation for each event. These equations can allow us to determine the strength of El Nino and La Nina events. These events are defined by the interaction of the atmosphere and the ocean that can affect weather around the world. El Nino and La Nina are the warm and cold phases of the ENSO cycle. The occurrence and duration of these events is recorded with Oceanic Nino Index (ONI), based on the monitoring of sea surface temperatures (SST) in the central Pacific Ocean. The ONI is used to identify the onset of an above average SST threshold that persists for several months, encompassing both the beginning and end of an ENSO episode.

Here, approaches are presented for climate projections. First approach is to classify El Nino and La Nina events using different machine learning classifier algorithms to check that these events were correctly categorized as weak, moderate, strong and very strong in the literature. This method is tested in different binary and multi-class experiments and very good results are obtained. Second approach is to group climate events according to their shapes semi-automatically by eye and to get good and simple equations using EAs for geophysicists to better understand these events. Third

approach is to group these events according to their shapes automatically by ML algorithms.

This thesis is organized as follows. In Section 2, some of the works related to this research experiment was presented. In Section 3, some basic concepts about Walker Circulation, El Nino-Southern Oscillation, El Nino and La Nina events were given. In Section 4, ML classification methods, Evolutionary algorithms, Genetic Programming, Real Valued Genetic Algorithm and GPU programming were described. In Section 5, ONI dataset was described. In Section 6, flow diagram of my research work was illustrated and the experimental results about the classification of the ENSO episodes and equations of shapes of events were shown.

2 Literature review

2.1 Walker Circulation

The Walker circulation, often known as the Walker cell, is a theoretical model of lower-atmosphere air flow in the tropics. Gilbert Walker was the one who discovered it. Jacob Bjerknes, a Norwegian-American meteorologist, introduced the term ”Walker circulation” in 1969 [23].

The Walker Circulation is the name given to the atmospheric loop located above the Pacific Ocean. Normally the trade winds, that is to say the regular winds in the intertropical region which move from east to west, blow over the Pacific and carry the underlying currents towards the west. The currents that are carried are composed mainly of water warmed by the sun in this region near the equator, which induces an accumulation of warm water in the western basin of the Pacific and consequently a rise in sea level (from 50 cm to 1 meter more compared to the eastern basin). Then this hot water evaporates to the west creating an updraft of moist and warm air which is blocked by the stratosphere, forcing it to move towards the east where it will descend following the meeting of an air colder. From this moment, the loop repeats.

By carrying large quantities of warm water towards the west, the breath of the trade winds creates a vacuum in the eastern basin. Any system tending towards equilibrium, this void is filled by an upwelling of cold, deep and nutrient-rich water. So in normal times, not only is the warm water carried west, there is also a rise in cold water in the east, creating a temperature difference between the two basins that can go up to 8°C.

2.2 ENSO

The El Nino Southern Oscillation (ENSO) is a cyclical change in below-normal and above-normal sea surface temperatures, as well as dry and wet conditions, in the tropical Pacific caused by the interaction of the atmosphere and ocean over a few years. The tropical ocean has an impact on the atmosphere above it, and the atmosphere has an impact on the ocean below it [24].

The thermocline is one of the layers of the Pacific Ocean affected by ENSO. The thermocline is a layer of water that separates surface and warm waters from deep and frigid seas. It can be thought of as the layer that divides warm and cold waters. This transition zone has the strongest

upward currents (or upwelling) along the equator. The depth of the thermocline is proportional to the temperature of the sea surface. Upwelling of cold, nutrient-rich deep-water is brought up from the bottom layers when the thermocline approaches the water top, resulting in colder temperatures at the water surface.

El Nino and La Nina events are defined by the interaction of the atmosphere and the ocean. El Nino and La Nina are Pacific Ocean climate phenomena that can affect weather around the world. El Nino and La Nina are the warm and cold phases of the ENSO cycle, with La Nina being referred to as the cold phase and El Nino as the warm phase. During an El Nino, sea level pressure is lower in the eastern Pacific and greater in the western Pacific, whereas during a La Nina, the opposite occurs. Weather, wildfires, ecosystems, and economies can all be affected by El Nino and La Nina. El Nino and La Nina episodes usually last nine to twelve months, but they can often last years. On average, El Nino and La Nina episodes occur every two to seven years, however they don't happen on a regular basis. El Nino tends to happen more frequently than La Nina.

Surface trade winds blow westward across the equatorial Pacific Ocean during ENSO neutral conditions. These winds, which blow against the ocean's surface, cause a westward current. On South America's northern Pacific coast, neutral conditions encourage the upwelling of cool, nutrient-rich sea water. This technique encourages a strong aquaculture industry while also supporting a healthy marine ecosystem. The equatorial region from South America to the central Pacific will have comparatively low sea temperatures compared to other equatorial Pacific regions, contributing to unusually dry weather in that area.

2.3 El Nino

El Nino is a weakening of the Walker Circulation that is characterized by extremely warm ocean temperatures in the Equatorial Pacific. El Nino is a tropical Pacific ocean-atmosphere system oscillation that has significant weather implications around the world. Fishermen off the coast of South America first noticed El Nino as the presence of unusually warm water in the Pacific Ocean around the beginning of the year. El Nino is a Spanish word that means "Little Boy" or "Christ Child." This term was given to the phenomena because of its tendency to appear around Christmas. El Nino is also sometimes referred to as "an ENSO warm event" [22].

Trade winds weaken during El Nino. Warm water is pushed back east, toward the west coast of the Americas. The normally occurring east to west winds weaken during El Nino conditions, resulting in an anomalous west to east flow. Warm equatorial waters are moved from the western Pacific to the eastern Pacific and northern South America by the west to east flow. El Nino causes an increase in sea level heights, a decrease in cool water upwelling, and an increase in sea surface temperature in the eastern equatorial Pacific. Increased convection and the danger of heavy precipitation in northwestern South America are caused by rising sea surface temperatures.

El Nino has the potential to have a huge impact on our weather. The Pacific jet stream is moving south of its neutral location due to the warmer oceans. The northern United States and Canada are drier and warmer than typical as a result of this shift. However, in the Southeast, these seasons have been wetter than typical, resulting in more floods.

El Nino has a big impact on the marine life off the coast of the Pacific. Upwelling delivers chilly, nutrient-rich water to the top during typical conditions. Upwelling weakens or ceases entirely during an El Nino. There are fewer phytoplankton off the coast without the deep-water nutrients. This has an impact on fish that eat phytoplankton, and everything that eats fish as a result. Warmer waters may also allow tropical species such as yellowtail and albacore tuna to migrate to locations where they would ordinarily be too cold.

2.4 La Nina

La Nina is characterized by unusually cold ocean temperatures in the Equatorial Pacific and is a strengthening of the Walker Circulation. La Nina means The Little Girl in Spanish. La Nina is also sometimes called El Viejo, anti-El Nino, or simply ”an ENSO cold event” [22].

La Nina has the opposite effect of El Nino. Trade winds are stronger than normal during La Nina occurrences, driving more warm water toward Asia. Upwelling occurs more frequently off the west coast of the Americas, sending cold, nutrient-rich water to the surface. The jet stream is being pushed northward by cool Pacific waters. Drought in the southern United States, as well as strong rainfall and flooding in the Pacific Northwest and Canada, are likely outcomes. Winter temperatures in the South are warmer than average during a La Nina year, whereas winter temperatures in the North are cooler than typical. A more severe hurricane season may result from La

Nina. The seas along the Pacific coast are cooler and more nutrient-rich than usual during La Nina. This environment supports more marine life and attracts more cold-water species, like squid and salmon, to regions like the California coast.

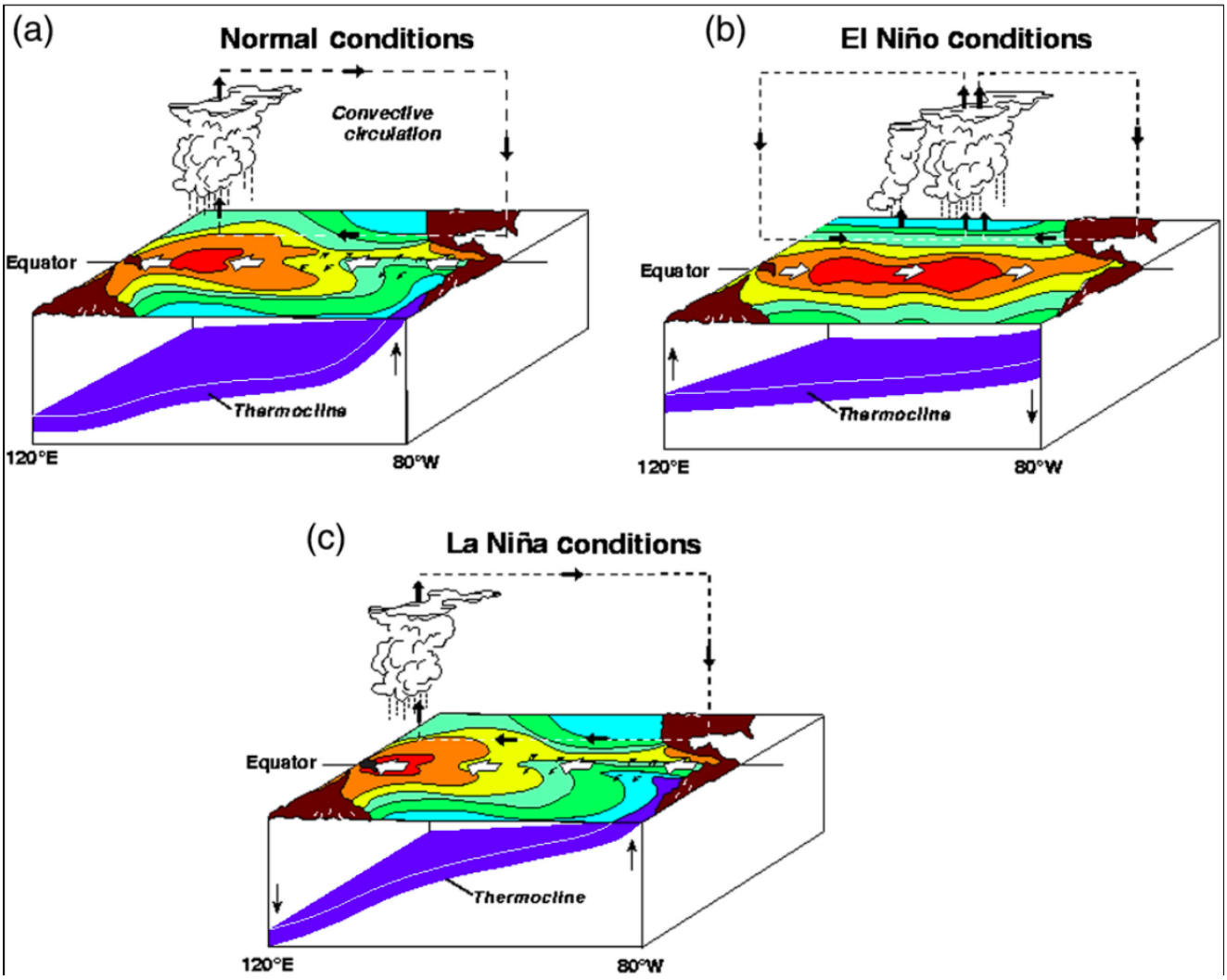


Figure 1: Schematic diagrams of (a) normal, (b) El Nino and (c) La Nina conditions. On the left side of each diagram is the western Pacific Ocean, near Asia, and on the right is the eastern Pacific Ocean, near South America. Warm ocean surface temperatures are indicated by red/orange colors, whereas cooler water is indicated by yellow/greenish colors. In the tropics, the thermocline is the depth at which the water temperature is around 20°C. Deep convective clouds rise over the pool of warm water. The warm pool of water flows eastward during El Nino (b), and the thermocline's slope flattens. The warm pool flows westward during La Nina (c), and the thermocline's slope steepens. (Source: NASA; see www.pmel.noaa.gov/elnino/schematic-diagrams)

2.5 Related work

In the case of the El Nino phenomenon, some studies concentrated on determining the global effects of ENSO occurrences [27]-[29] as well as possible connections to other climate or natural phenomena [29]-[33]. In terms of machine learning applications, various studies have attempted to predict ENSO events [25],[34]-[37]. They choose a set of spatiotemporal datasets and data to feed the forecasting model and perform some analysis. The purpose is to reduce predicting errors when given a specific problem indicator, such as the Oceanic Nino Index. Before to classify climate model, the important thing is the choice of training and test sets. Because both the training and test sets can contain the same ENSO occurrences. To avoid this, they've employed larger datasets that split the data so that each event appears only in the training or test sets.

Other some works have used deep learning for ENSO forecasting, such as [38] that utilized a deep learning approach that produces ENSO predictions for long-time periods. They used a transfer learning to train a convolutional neural network on historical and reanalysis data. El Nino events were predicted 12 months ahead of time using the proposed method. In [26], the researchers have used neural networks to predict ENSO episodes. The problem of ENSO forecasting is limited by a very low amount of data. There have only been 3–4 major El Nino (and a similar number of major La Nina) episodes since 1980. Because there is so little data, neural networks are prone to overfitting. To avoid this, neural networks must be regularized using techniques like Gaussian Noise layers, Dropout, Early Stopping, and L1 or L2 penalty terms. Another issue that can arise due to a lack of data is that a signal in a variable may appear in both the training and test data sets, ensuring that the model is a good generalization of the system for the researcher.

3 Research methods - Methodology

3.1 Machine Learning Classification

Machine learning (ML) is a branch of Artificial Intelligence whose goal is to develop computational learning techniques and build systems that can learn on their own [14]. Classification is one of the most essential components of the supervised learning strategy in machine learning, in which the computer program learns from the data it is given and makes new observations or classifications. Classification is the process of categorizing a set of data into categories. It can be done on both structured and unstructured data. Predicting the class of provided data points is the first step in the procedure. To describe the classes the terms target, label and categories are used.

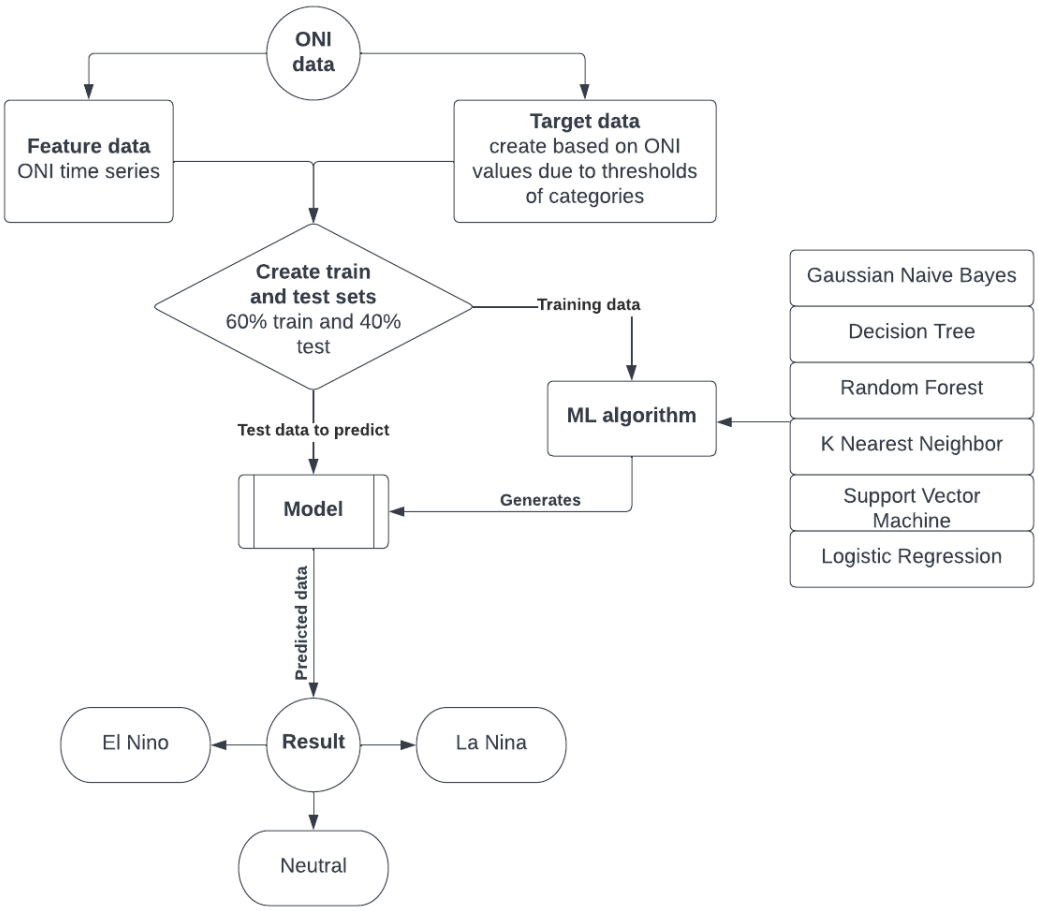


Figure 2: Flow diagram of ML classification

Here (Figure 2), an approach is presented to classify ENSO episodes applying ML algorithms. A method is to classify 3 month average ONI values to check that El Nino and La Nina events were correctly categorized as weak, moderate, strong and very strong. The idea is to take these ONI time series as feature and build categorical target data based on ONI values due to thresholds of categories (Table 4). Then, create train and test sets, randomly chosen 60% and 40% of all data respectively and applied 6 ML classifiers with default values. These ML classification algorithms train training data and generate a model which take test data to predict. As a result, algorithms learn a model to recognize ONI values correspond to which category. The code of this classification is shown in Annex C .

In the following, six classification algorithms (classifiers) is presented which used in my work.

1. Gaussian Naive Bayes [43],[44] - is a probabilistic classification technique based on Bayes' theorem, which assumes that predictors are independent.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

   where P(A|B) – the probability of event A occurring, given event B has occurred. P(B|A) – the probability of event B occurring, given event A has occurred. P(A) – the probability of event A. P(B) – the probability of event B. Note that events A and B are independent events.

   This classifier requires only a little quantity of training data in order to estimate the required parameters. In comparison to other classifiers, it is fast but it's only disadvantage is known to be a bad estimator.

2. Decision Tree [45], [46] - builds the classification model in the form of a tree structure in which each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents the target variable. A decision tree has the benefit of being simple to understand and visualize, and it also requires very little data preparation. The issue of the decision tree is that it can produce complex trees that are difficult to categorize. They can be highly unstable because even a minor change in the data might cause the decision tree's entire structure to fail.

3. Random Forest [47], [48] - is an ensemble learning method for classification and contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. The number of trees in the forest was taken as default value 100. The random forest has the advantage of being more accurate than decision trees because of the reduction in over-fitting. The only disadvantage of random forest classifiers is that they are difficult to develop and perform poorly in real-time prediction.

4. K Nearest Neighbor [49], [50] - is a lazy learning algorithm that stores all instances corresponding to training data in n-dimensional space. A new data point is classified based on similarity in the specific group of neighboring data points. The ”K” is the number of neighbors it checks. In this experiment K number was taken as default value 5. This classifier is simple to develop and is robust to noisy training data. It is quite efficient, even if the training data is enormous. The only disadvantage of the KNN algorithm is that it does not require determining the value of K, and it has a relatively high computation cost when compared to other algorithms.

5. Support Vector Machine [51] - is a classification algorithm that represents training data as points in space split into categories by a gap as a distance as possible. After then, new points are added to space by predicting which category they will fall into and which space they will belong to. Its decision function uses a subset of training points, making it memory efficient and very effective in high-dimensional spaces. The support vector machine's only issue is that the approach does not immediately provide probability estimations. Regularization parameter was taken as default value 1.0.

6. Logistic Regression [52]- is a classifier for predicting a categorical dependent variable using a given set of independent variables. The goal of logistic regression is to determine the best line between the two classes. Regularization was taken as default L2 which is called Ridge Regression and regularization parameter as default 1.0. It is commonly utilized in linear classification problems. The function that used in this method is called logistic function or sigmoid function. It is S-shaped curve that takes values between 0 and 1, but never takes these limits. The equation for the sigmoid function is following:

$$y = 1/(1 + e^{-x}) \tag{2}$$

where, e - represents Euler's number or exponential function (exponential constant) and it is equal to a value of approximately 2.71828.

This method was tested in different experimental setups, a binary problem (El Nino and non-El Nino occurrences) and a multi-class problem (strong El Nino, weak El Nino, strong La Nina, weak La Nina and regular year). In all cases, the results indicate that Random Forest and Decision Tree correctly predicted the ENSO episodes with 100% accuracy.

3.2 Evolutionary Algorithms

Evolutionary Algorithms (EA), inspired by Charles Darwin's theory of natural evolution and initially presented by Hans-Paul Schwefel amd Ingo Rechenberg in 1965 (Evolutionary Strategies) [[15]-[16]] and then John Holland in the 1960's and early 1970's (Genetic Algorithms) [2], are stochastic search and optimization algorithms based on the principles of natural evolution.

EA is a metaheuristic that be used to try to find a good maximum/minimum of an arbitrary dimensional function. This algorithm does the recombination is exchanging some genes between the fittest pair of chromosomes and mutation is to insert or change the recombined chromosome to make sure that the candidate solutions are not get stuck in the local interval. Although it is not guaranteed that satisfactory solution will be obtained after n generations, the algorithm tries to do better than Random Search in the hope that eventually, it will find a good enough solution to the problem[3].

The main parameters used in EA are the following:

- Population size and number of generations - The product of these two parameters define the number of evaluations. A small population can be used to evolve for a large number of generations or a large population can be used to evolve for a small number of generations. Using large population size helps to keep diversity and avoid premature convergence.

- Crossover and mutation probabilities - These probabilities depend on the problem. The crossover is called with probability of 80-90%. Mutation occurs after crossover. Mutation operator is usually called with probability of 100%. However, this does not imply that all children will be mutated. Every child has the potential to be mutated, but only a small percentage of them do. High mutation probability may cause non-convergence.

- Number of children per generation - is usually the same as the population size. Choosing the number of children per generation generally depends on the desired speed of the convergence of the algorithm. Allowing parents to compete with children is a powerful form of elitism that can be countered by creating many children per generation.

- Selection pressure - is one of the most important parameters in artificial evolution for parents selection and population reduction. Parents are selected with replacement and population reduction is done without replacement, the same selection operator has a different selection pressure in each case. Selection pressure increase nearly linearly for parents selection, but not linearly for population reduction.

EA follows an evolutionary loop as shown in Figure 3.

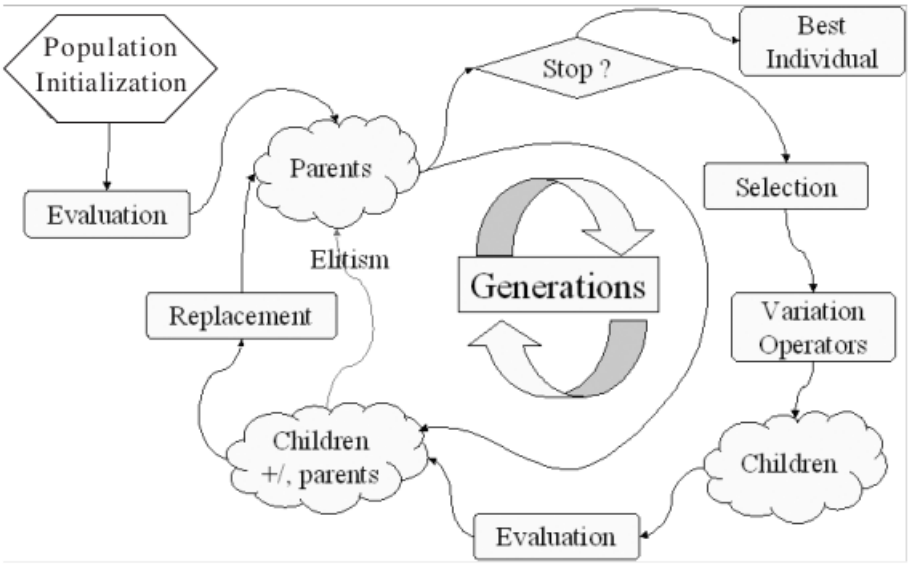


Figure 3: Flowchart of evolutionary algorithm (from [17])

3.2.1 Genetic Programming

In 1985 and 1992, Cramer and Koza developed Genetic Programming (GP) [[12]-[13]] whose aim was not to find optimal values to minimize fitness functions (optimization) but to create functions that would solve some observed data (laws), transferring artificial evolution into the Machine Learning domain. GP uses tree based representation. In case of GP, potential solution is equation. The major preparation steps for GP are the following:

- Specifying the set of terminals including independent variables of the problem, random constants.
- Specifying the set of primitive functions.
- Choosing the fitness measure.
- Setting parameters to control the run such as population size, probabilities of performing the crossover and mutation.
- Specifying the termination criterion.

The evolutionary steps are shown in Figure 4.

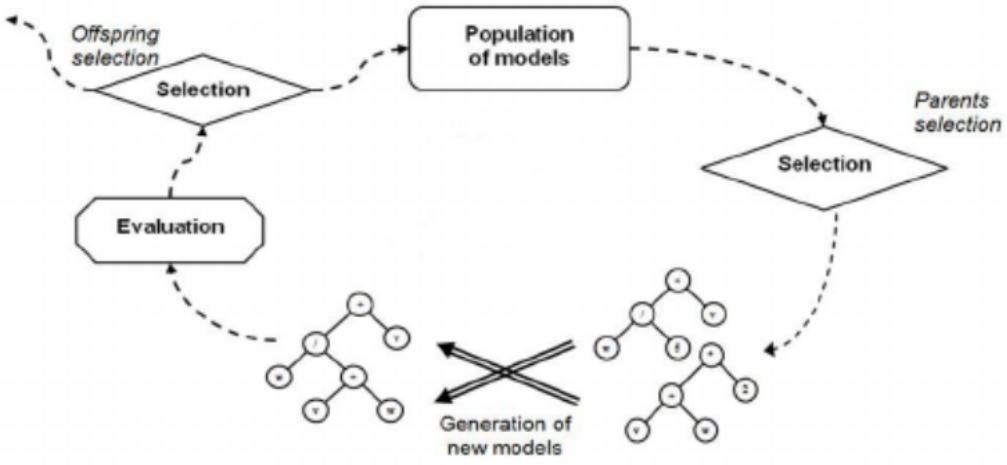


Figure 4: Flowchart of GP (from [18])

3.2.2 Real valued Genetic Algorithm

Real valued Genetic Algorithms (RVGA) involve three types of operators: selection, crossover and mutation. Real valued Genetic Algorithms work with a randomly generated population that is made of a set of individuals that contain a chromosome. Initial population consists of potential solutions to the problem. In the case of RVGA, potential solution is parameter values.

Each individual in the population is evaluated using the fitness function, given by the following formula :

$$fitness = \sqrt{\sum_{i=1}^{n} \frac{|f(x_i) - s_i|^2}{n}} \tag{3}$$

where $f(x_i)$ is the predicted value, x_i is the time, s_i is the expected value and n is the number of samples.

It measures how fit each individual or potential solution is to the given problem.

Tournament selection method is used that it is not replacing all parents with children. Once children are created, we create 1 child, now the population size is twice as big, but we must reduce it down to original size. So it is done by selecting k number of individuals and out of them choose the best one, and this is repeated until you get the desired population size.

Algorithm 1: RVGA

Result: Good minimum of $f(X)$ has been approximated

parsing;

initialization;

while *current number of runs* $< N$ **do**

  while *the criterion is not met* **do**

    calculate the fitness of all *chromosomes*;

    for *each fittest pair* (P_1, P_2) *in sorted chromosomes* **do**

      Offsprings = recombine(P_1, P_2);

      Mutants = mutate(*Offsprings*);

      replace the pair (P_1, P_2) with *Mutants*;

    end

  end

end

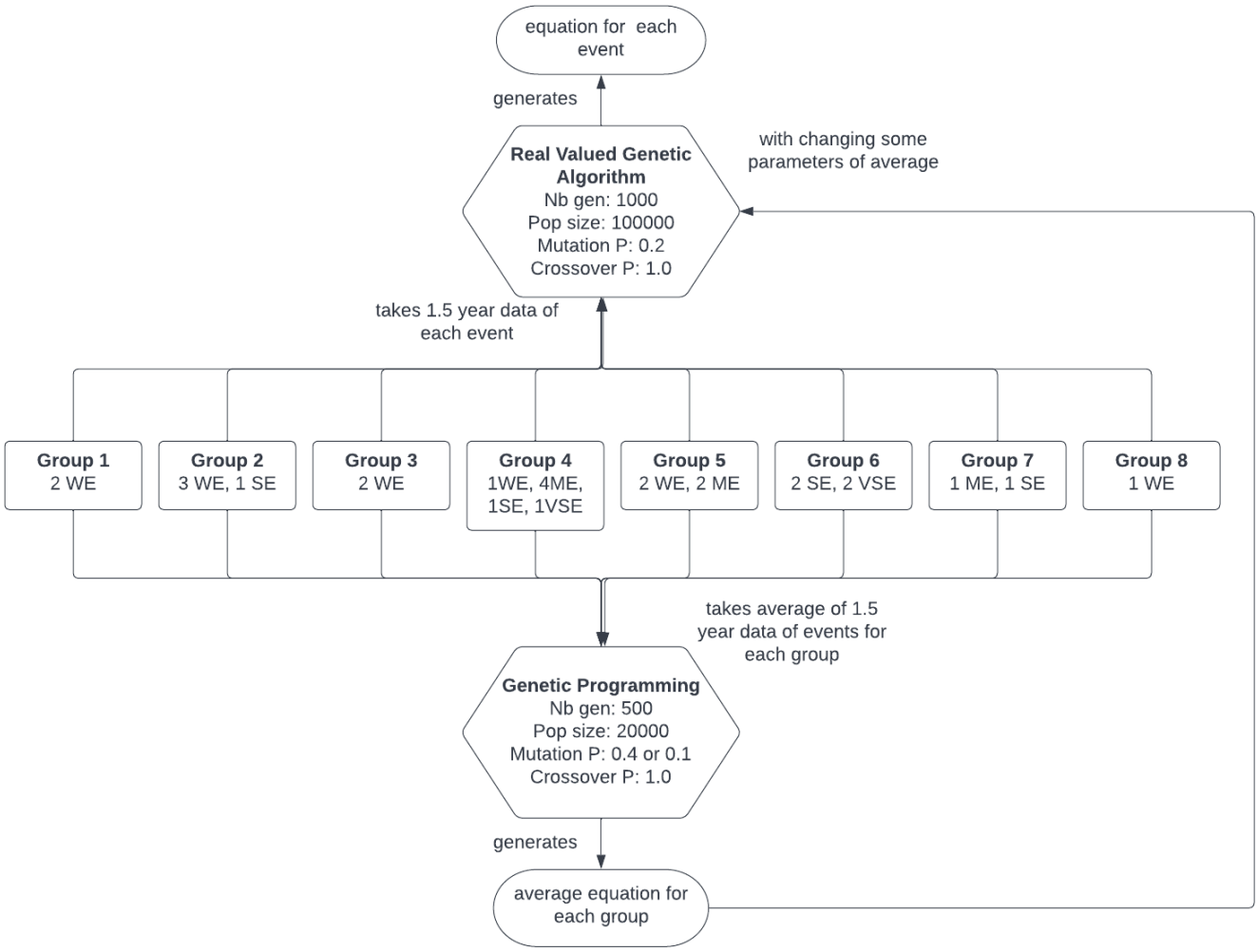


Figure 5: Flow diagram of EA

Here (Figure 5), an approach is presented to find equation for each event. Firstly, GP is applied to events to find equations, then RVGA is applied to change parameters of this equations.

3.3 GPU programming

In recent years, the Graphics Processing Unit (GPU) [6] has emerged as a powerful computation device. The GPU is much faster than the Central Processing Unit (CPU) for computation because of their architectures are different. While multi-core CPUs are designed to execute arbitrary functions such as input/output access, processing, and so on, the GPU is designed to optimize performance for specific applications [7]. The main issue is that not all algorithms can be implemented properly

on the GPU. This technology may benefit only numerical problems that are inherently parallel.

In contrast to the CPU, which consists of one or more sophisticated cores operating at high clock speeds, the GPU consists of thousands of simple cores operating at lower clock speeds and with simplified architectures. Because of its high clock speed, operation rescheduling capabilities, and enormous cache memory, the CPU can quickly handle many general-purpose activities, although it is less efficient in massively data-parallel applications. In contrast, GPUs that use the Single Instruction Multiple Data (SIMD) model can perform the same operations on many data items at the same time, resulting in great computational speed in data-parallel applications due to massive parallelism [8].

Nvidia [9] created a software architecture called Compute Unified Device Architecture, or CUDA, that allows C code to be practically directly translated to the GPU. In this case, the syntax is made up of only the most basic C extensions [10]. This feature provides an appropriate environment for converting existing C code to CUDA. Before building the first application in CUDA, programmers who are already familiar with the C language must integrate the concept of thread. The GPU is embodied by the thread notion, which is based on the SIMD approach from vector computers. A thread is a set of instructions that can be performed in parallel on several data units.

CUDA is an intuitive and scalable programming model based on the CUDA-C [11] programming language, which is an expanded version of the C programming language. This model combines the CPU and GPU, also known as host and device, into a heterogeneous computing system to maximize the benefits of both. CUDA contains three types of functions: The CPU is the only one who can call and execute the host functions. These functions are the same as those written in C. The kernel functions, which can only be called by the CPU and are only executed by the device. The qualifier, __*global*__, must be added before the type of return of the function, which is always void, for this type of function. Finally, device functions are those that are only called and executed by the device, and the qualifier, __*device*__, must be specified before the function's type of return. It is permissible to return any type of value in this case.

3.3.1 Parallelization of Evolutionary Algorithms

Evolutionary algorithms are inherently parallel. EAs are ideal for usage on a parallel computing platform due to a following variety of reasons [19]:

- EAs use a population of solutions in each iteration. Every member must be evaluated before any EA operator is applied to the population in order to determine the objective and constraint values in an optimization problem solving task. Since the evaluation of solutions is an independent event, the evaluation task can be performed parallely.
- The majority of EA operators involve one or two solutions, making them excellent for parallel implementation.
- EA is a Markov chain that only uses information from the previous population, reducing the number of parameters that must be stored iteration by iteration.

EAs are often parallelized and implemented in practical applications for these reasons. EAs can be parallelized using a single machine with multiple processor cores or a system of machines sharing the population over the network. The common idea is to divide the population into chunks and solve them simultaneously using multiple processors. There are two ways of doing that [20]:

- GPGPU parallelization - this model executes all the evolutionary operators on a single machine, but the fitness evaluation is distributed across several processors.
- Island parallelization - this model maintains a list of populations that are evolving independently. Each of these populations may run on a separate processor in parallel. The populations are commonly referred as islands since they appear to be evolutions on different continents from an evolutionary standpoint. The islands do not collaborate with each other, except a specific operator called migration operator. This operator periodically, in a predefined way, exchanges a small portion of the population among the islands. We intend to bring new genetic material by borrowing a few individuals from another island. The island model has been proved to increase the algorithm's solution quality and execution time on its own.

3.3.2 GPGPU parallelization

General Purpose Graphic Processing Units (GPGPUs) are processors that are initially designed to process images that are made of millions of pixels. The designers of Graphic Processing Units designed specific processors where most of the transistors are used to implement Arithmetic and Logic Units (ALUs), resulting in a strange architecture, with the aim of implementing as much computing power as possible on the limited space of a silicon chip.

Where on a multi-core CPU, all cores are independent (meaning that they can run their own programs independently), GPGPU designers chose to maximise computing power at the cost of reducing versatility, which was even more possible that when processing a million pixel / vertices image, all the algorithms run on all pixels or vertices are identical to the instruction.

So one very radical simplification that saves a lot of space on the chip is the following : rather than surrounding each ALU with everything it needs to be independent the designers decided to group a number of cores into what they called “multi-processors” and having them share the functional units that tend the processors. Typically, an ALU needs a fetch and dispatch unit, to fetch from memory the next operator and operands to be loaded in the ALU and dispatch them in the correct registers.

This means that in a 32 core multi-processor, all cores will share the same fetch and dispatch functional unit, that will fetch the next instruction to be executed and will dispatch it in all the 32 cores of the same multi-processor, meaning that in a multi-processor, all cores must execute the same instruction at the same time. This very restrictive form of parallelism is called Single Instruction Multiple Data (SIMD).

Now, there is not only one multiprocessor in a GPGPU chip, but many, all of them having their own fetch and dispatch unit meaning that if all the cores within one multiprocessor must do the exact same instruction at the same time , several multiprocessors can run different parts of the same program at the same time (they all have their own Program Counter). So the very restrictive SIMD parallelism is relaxed into what is called Single Program Multiple Data (SPMD), where different multiprocessors can simultaneously run different functions of the same program.

3.3.3 Island paralellization

When parallelizing on n different isolated islands may yield n different results, it may happen that an island gets stuck into a local optimum. When this happens, receiving an individual from another island may allow the population to diversify again in order to find a better result. In this case, after an island is stuck in a local optimum, it receives an individual from another island. Then two cases are possible:

- The incoming individual has a lower fitness value than the local individuals. In this case, the individual will not be used for recombination and it will very probably be removed from the population during the reduction phase, when creating the new population through selection and the island will remain stuck in its local optimum.
- The incoming individual has a better fitness value than the local individuals. In this case:
  - it will be very often selected among parents to create new children.
  - it will be selected to survive to the next generation and hang around until the average genotype of the island's population has migrated towards the genotype of the good immigrant.

Migrating towards the genotype of the good immigrant means that the island that was stuck into a local minimum will increase its diversity, therefore increasing the exploration potential of the algorithm. So an island algorithm optimises the exploration and exploitation ratio, islands exploit local optima and get stuck until they receive a potentially good individual from some other island, that will get them out of their local optimum, giving them the opportunity to find a better solution while going towards the good immigrant. The island model reduces variability of results, even if several islands are used on the same machine [53].

4 Data

I used Oceanic Nino Index (ONI) dataset from Climate Prediction Center of the National Oceanic and Atmospheric Administration (NOAA) [1]. It is the running 3-month mean Sea Surface Temperature (SST) anomaly for the Nino 3.4 region which located in the intervals [5° South; 5° North] and [170° West; 120° West] in the Pacific ocean.

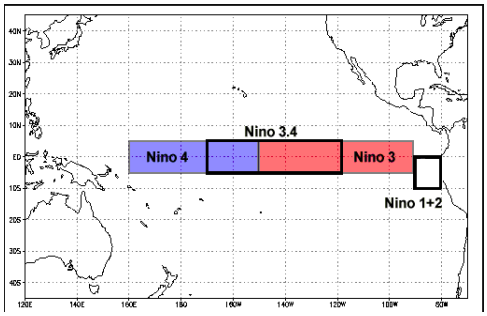


Figure 6: Nino 3.4 region, source

This dataset covers ONI time series from 1950 to present. During the period covered by this dataset, 26 El nino and 24 La nina events occurred (Table 4). The index is defined as the average of monthly SST anomalies. The monthly anomalies are calculated by subtracting the average temperature in the same month over the past 30 years. Then the running three-months average is calculated.

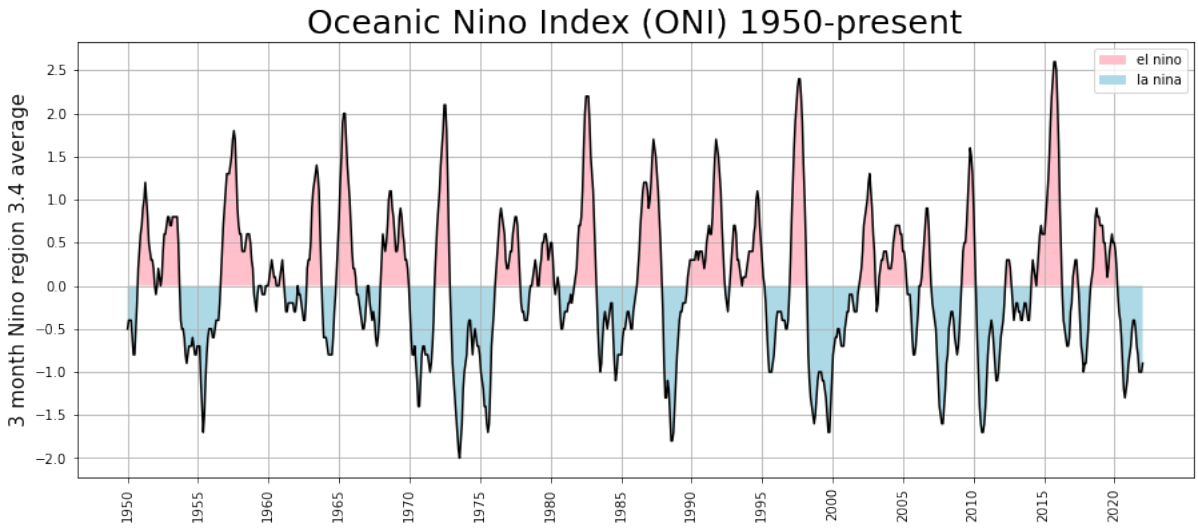


Figure 7: ONI time series

When El nino events are identified in at least five consecutive 3-month periods at or above $+0.5°$ anomaly, La nina events are identified at or below $-0.5°$ anomaly. These events can be categorized as weak, moderate, strong and very strong when their anomalies equaled or exceeded the threshold for at least three consecutive 3-month periods. The threshold for weak is from 0.5 to 0.9, for moderate is from 1.0 to 1.4, for strong is from 1.5 to 1.9 and for very strong is equal or greater than 2.0 (Table 4).

Table 1: Intervals of thresholds for each category

Category	Weak	Moderate	Strong	Very Strong
Threshold interval	[0.5 - 0.9]	[1.0 - 1.4]	[1.5 - 1.9]	[2.0 - ∞]

Table 2: El Nino and La Nina events

El Nino				**La Nina**		
Weak	Moderate	Strong	Very Strong	Weak	Moderate	Strong
1952-53	1951-52	1957-58	1982-83	1954-55	1955-56	1973-1974
1953-54	1963-64	1965-66	1997-98	1964-65	1970-71	1975-76
1958-59	1968-69	1972-73	2015-16	1971-72	1995-96	1988-89
1969-70	1986-87	1987-88		1974-75	2011-12	1998-99
1976-77	1994-95	1991-92		1983-84	2020-21	1999-00
1977-78	2002-03			1984-85	2021-22	2007-08
1979-80	2009-10			2000-01		2010-11
2004-05				2005-06		
2006-07				2008-09		
2014-15				2016-17		
2018-19				2017-18		

5 Experimental research

5.1 Flow diagram of research

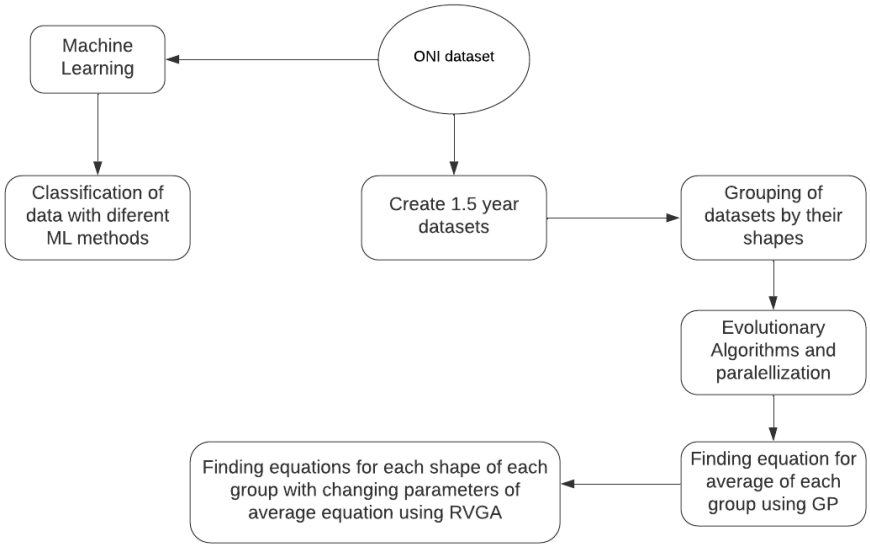


Figure 8: Flow diagram of research

In Figure 8, the flow diagram of the research that done during the experiment is shown:

- First, ONI dataset is classified with different machine learning methods to check that El Nino and La Nina events were correctly categorized as weak, moderate, strong and very strong.
- Then using this data, selected a peak point for each event and generated 1.5 year data that 11 months before and 6 months after that peak.
- Grouped these datasets according to their shapes.
- Using Genetic Programming that has been paralleled, found a sinusit equation that corresponds to the average of the shapes in each group.
- Finally, found equations that changing the parameters of average equation using parallel Real valued Genetic Algorithm for each shape in each group.

5.2 Setup

Here the link of my research work is shown where have the plots, codes and files : research source

5.2.1 Machine Learning part

I worked on Google Colab equipped with an Intel(R) Xeon(R) CPU @ 2.30GHz with one physical core, NVIDIA® Tesla® K80, memory around 12.6GB, disk around 33 GB available to use, and Linux based operating system.

Python programming language was used to classification. A list of main libraries used is the following :

- pandas - is easy to use data structures and operations for manipulating numerical tables and time series in data analysis.
- sklearn (scikit-learn) - is perhaps the most useful machine learning library. The sklearn library contains a lot of efficient, useful and simple tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
- matplotlib - is a comprehensive plotting library for creating static, animated, and interactive visualizations easily.

5.2.2 Evolutionary Algorithms part

I connected to UFAZ server remotely using CISCO (CISCO Anyconnect Secure Mobility Client) vpn app and worked on 3 supercomputer machines on island mode simultaneously and Linux based operating system. Connected with Cluster SSH (cssh) command that allows to manage multiple servers over SSH from a single administration console.

Evolutionary Algorithm Specification Language (EASEA) was used to find sinusit equations. EASEA [21] is an Artificial Evolution software platform that was originally designed to help non expert programmers to try out evolutionary algorithms to optimize their applied real-world problems without the need to implement a complete algorithm. It was revived to integrate the parallelization of children evaluation on NVidia GPU cards.

EASEA is a specification language with its own built-in compiler. Its compiler generates C++ source files that implement the entire evolutionary algorithm. The -cuda option can be added to the compiler in order to automatically generate a source code for parallelizing the evaluation of the children population on a CUDA compatible NVidia card out of the same . ez specification code, therefore allowing anyone who wants to use GPU cards to do so without knowing anything about GPU programming.

In Figure 9, run parameters of GP in EASEA are shown :

```
\Default run parameters :
  Number of generations : 500                   // NB_GEN
  Time limit: 0                                 // In seconds, 0 to deactivate
  Population size : 20000                       // POP_SIZE
  Offspring size : 100%                 // can be a percentage such as 40%
  Mutation probability : 0.4            // Probability to call the mutation function
  Crossover probability : 1.0             // Probability to call the crossover function
  Evaluator goal : minimise             // or Maximise
  Selection operator: Tournament 7      // to select parents
  Surviving parents: 100%               // to select breeders
  Surviving offspring: 100%             // to select among offspring for next generation
  Reduce parents operator: Tournament 2 // how to select the breeders
  Reduce offspring operator: Tournament 2 // how to select the offspring that will compete to access the next generation
  Final reduce operator: Tournament 7   // to select the individuals composing the next generation

  Elitism: Strong                               // Strong = from parents pop, Weak = from parents + children
  Elite: 1
  Print stats: true
  Generate csv stats file:false
  Generate gnuplot script:false
  Generate R script:false
  Plot stats:true

  Remote island model: true
  IP file: ip.txt                     //File containing all the remote island's IP
  Server port : 2929
  Migration probability: 0.33

  Save population: false
  Start from file: false

  // nouveaux paramètres
  max init tree depth : 4
  min init tree depth : 2
  max tree depth : 6

  size of prog buffer : 200000000
```

Figure 9: Parameters of Genetic Programming

How to run the program of GP:

- easena filename.ez -cuda_gp
- make
- ./filename

In Figure 10, run parameters of RVGA in EASEA are shown :

```
\Default run parameters :                // Please let the parameters appear in this order
  Number of generations : 1000           // NB_GEN
  Time limit: 0                          // In seconds, 0 to deactivate
  Population size : 100000               //131072 //16384 //262144 // 131072 // 16384
  Offspring size : 100%                  // 40%
  Mutation probability : 0.2             // MUT_PROB
  Crossover probability : 1              // XOVER_PROB
  Evaluator goal : minimise              // Maximise
  Selection operator: Tournament 10
  Surviving parents: 100                 //percentage or absolute
  Surviving offspring: 100%
  Reduce parents operator: Tournament 2
  Reduce offspring operator: Tournament 2
  Final reduce operator: Tournament 7

  Elitism: Weak                //Weak or Strong
  Elite: 1
  Print stats: true                          //Default: 1
  Generate csv stats file:false
  Generate gnuplot script:false
  Generate R script:false
  Plot stats:false                           //Default: 0

  Remote island model: true
  IP file: ip.txt                    //File containing all the remote island's IP
  Server port : 2929
  Migration probability: 0.1 // 0.15 or 0.5

  Save population: true
  Start from file: false
```

Figure 10: Parameters of Real valued Genetic Algorithm

How to run the program of RVGA:

- easena filename.ez -cuda
- make
- ./filename

5.3 Results

5.3.1 Classification of different datasets with different models

I set 4 experimental datasets and classified them using ML classification algorithms described in Section 3.1. Firstly, for each dataset 3 months average ONI values were taken as feature data and target data was built based on feature data due to thresholds (Table 4). There are 859 samples in all datasets. Then, split these datasets to create train (60% of all data) and test sets (40% of all data) which were chosen randomly. Train set takes 515 samples and test set takes 344 samples in all cases. And applied 6 ML classification models. These ML classification algorithms train training data and generate a model which take test data to predict

1. Dataset 1 - 2 classes with El Nino (EN) and anything else (AE). If ONI is greater or equal than 0.5, it will be EN, otherwise it will be AE. There are 242 values in EN and 617 values in AE. This dataset represents a binary classification because only have 2 classes. Train set takes 141 EN and 374 AE and test set takes 101 EN and 243 AE.

Table 3: Prediction for dataset 1

Classifier	accuracy	mean	std
Random Forest	1.0	1.0	0.0
Logistic Regression	1.0	1.0	0.0
Gaussian NB	0.962209	0.972032	0.017098
Decision Tree	1.0	1.0	0.0
Support Vector Machine	1.0	1.0	0.0
K Nearest Neighbor	1.0	1.0	0.0

Only Gaussian Naive Bayes classified value 0.5 as regular year, others classified all intervals correctly.

2. Dataset 2 - 3 classes with El Nino (EN), La Nina (LN) and regular year (RY). If ONI is greater or equal than 0.5, it will be EN, if ONI is less or equal than -0.5, it will be LN, otherwise it

will be RY. There are 242 values in EN, 248 values in LN and 369 values in RY. This dataset represents a multiclass classification because have 3 classes. Train set takes 141 EN, 152 LN and 222 RY and test set takes 101 EN, 96 LN and 147 RY.

Table 4: Prediction for dataset 2

Classifier	accuracy	mean	std
Random Forest	1.0	1.0	0.0
Logistic Regression	1.0	1.0	0.0
Gaussian NB	1.0	1.0	0.0
Decision Tree	1.0	1.0	0.0
Support Vector Machine	1.0	1.0	0.0
K Nearest Neighbor	1.0	1.0	0.0

All classifiers classified all intervals correctly.

3. Dataset 3 - 5 classes with strong El Nino (SEN), strong La Nina (SLN), regular year (RY), weak El Nino (WEN) and weak La Nina (WLN). If ONI is less or equal than -1, it will be SLN, if ONI is greater than -1 and less or equal than -0.5, it will be WLN, if ONI is greater than -0.5 and less than 0.5, it will be RY, if ONI is greater or equal than 0.5 and less than 1, it will be WEN, if ONI is greater or equal than 1, it will be SEN. There are 103 values in SEN, 91 values in SLN, 369 values in RY, 139 values in WEN and 157 values in WLN. This dataset represents a multiclass classification because have 5 classes. Train set takes 50 SLN, 102 WLN, 222 RY, 85 WEN and 56 SEN and test set takes 41 SLN, 55 WLN, 147 RY, 54 WEN and 47 SEN.

Table 5: Prediction for dataset 3

Classifier	accuracy	mean	std
Random Forest	1.0	1.0	0.0
Logistic Regression	0.872093	0.884721	0.022366
Gaussian NB	1.0	1.0	0.0
Decision Tree	1.0	1.0	0.0
Support Vector Machine	0.979651	1.0	0.0
K Nearest Neighbor	1.0	1.0	0.0

Logistic Regression classified values -0.5 and 0.5 as regular year, -1 as weak la nina, 1 as weak el nino and Support Vector Machine classified value 1 as weak el nino, others classified all intervals correctly.

4. Dataset 4 - 5 classes with weak El Nino (WE), moderate El Nino (ME), strong El Nino (SE), very strong El Nino (VSE) and anything else (AE). If ONI is greater or equal than 0.5 and less than 1, it will be WE, if ONI is greater or equal than 1 and less than 1.5, it will be ME, if ONI is greater or equal than 1.5 and less than 2, it will be SE, if ONI is greater or equal than 2, it will be VSE, otherwise it will be AE. There are 139 values in WE, 55 values in ME, 29 values in SE, 19 values in VSE and 617 values in AE. This dataset represents a multiclass classification because have 5 classes. Train set takes 374 AE, 85 WE, 26 ME, 20 SE and 10 VSE and test set takes 243 AE, 54 WE, 29 ME, 9 SE and 9 VSE.

Table 6: Prediction for dataset 4

Classifier	accuracy	mean	std
Random Forest	1.0	1.0	0.0
Logistic Regression	0.857558	0.921977	0.021429
Gaussian NB	1.0	1.0	0.0
Decision Tree	1.0	1.0	0.0
Support Vector Machine	0.973837	1.0	0.0
K Nearest Neighbor	0.994186	0.997674	0.004651

Logistic Regression classified values 2-2.5 (very strong el nino) as strong, values 1-1.3 (moderate el nino) as weak and value 0.5 (weak el nino) as regular climate event. Support Vector Machine classified value 2 (very strong el nino) as strong and value 1 (moderate el nino) as weak el nino. K Nearest Neighbor classified value 2 (very strong el nino) as strong el nino. Random Forest, Gaussian Naive Bayes and Decision Tree classified all intervals correctly.

In conclusion, Random Forest and Decision Tree classifiers gave the best result for all 4 datasets that classified all ONI values correctly due to their intervals.

5.3.2 Grouped events according to shapes semi-automatically by eye

After ML classification, events were classified according to their shapes. Firstly, 1.5 year data for El Nino events were generated that 11 months before and 6 months after selected peak. And grouped time series of these events according to their shapes semi-automatically by eye which are similar. These 8 groups are as follows:

1. Group 1 - contains 2 weak events (1952-53, 2014-15).

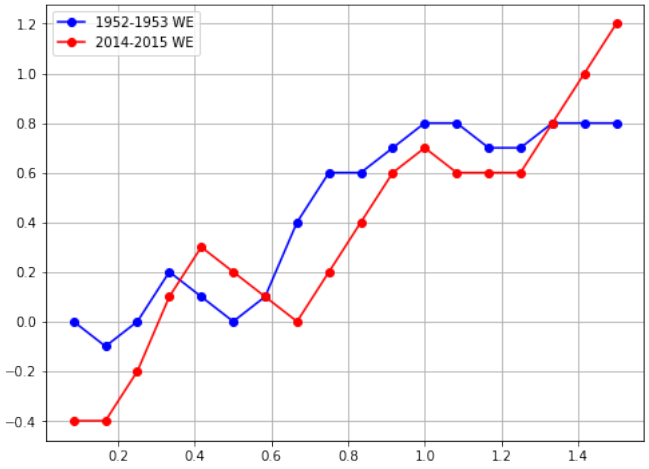


Figure 11: Shapes of group 1

2. Group 2 - contains 4 events that three of them are weak (1969-70, 1979-80, 2004-05) and one is strong event (1987-88).

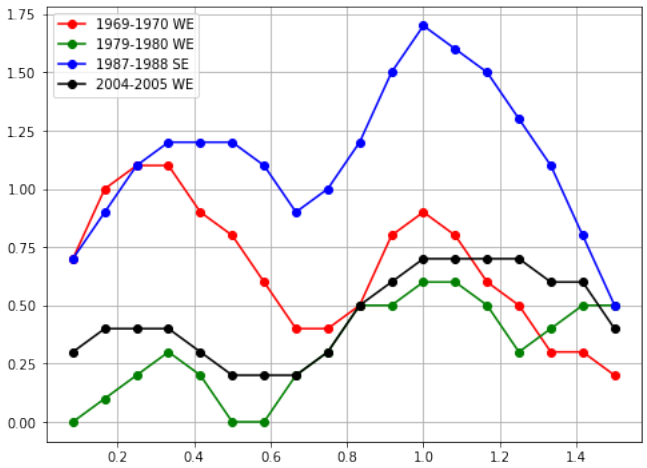


Figure 12: Shapes of group 2

3. Group 3 - contains 2 weak events (1958-59, 1977-78).

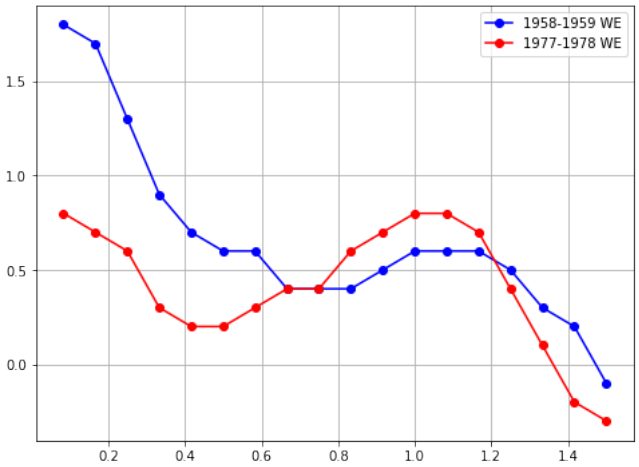


Figure 13: Shapes of group 3

4. Group 4 - contains 7 events that one of them is weak (2006-07), four of them are moderate (2002-03, 2009-10, 1963-64, 1994-95), one is strong (1957-58) and one is very strong event (1982-83).

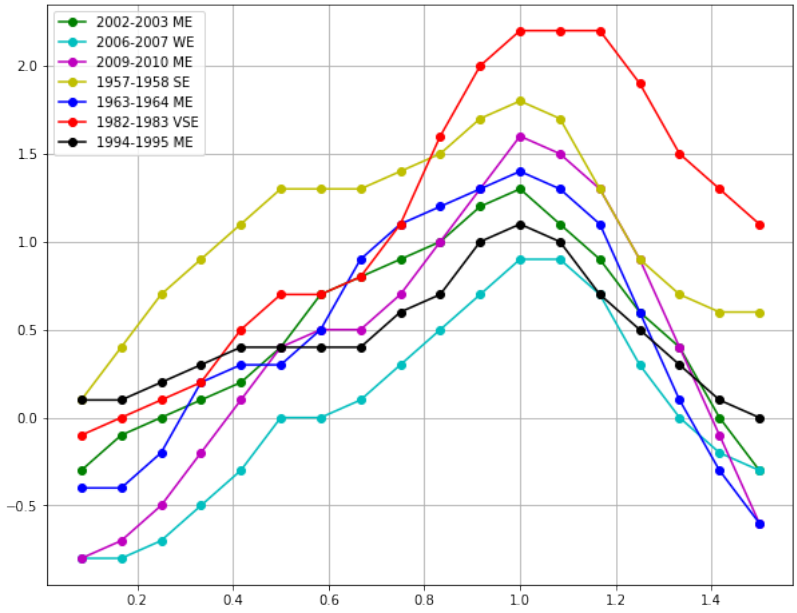


Figure 14: Shapes of group 4

5. Group 5 - contains 4 events that two of them are weak (1976-77, 2018-19) and two of them are moderate events (1951-52, 1986-87).

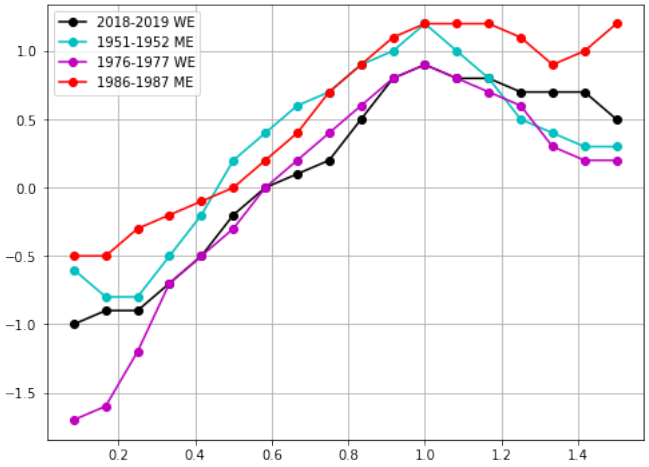


Figure 15: Shapes of group 5

6. Group 6 - contains 4 events that two of them are strong (1965-66, 1972-73) and two of them are very strong events (1997-98, 2015-16).

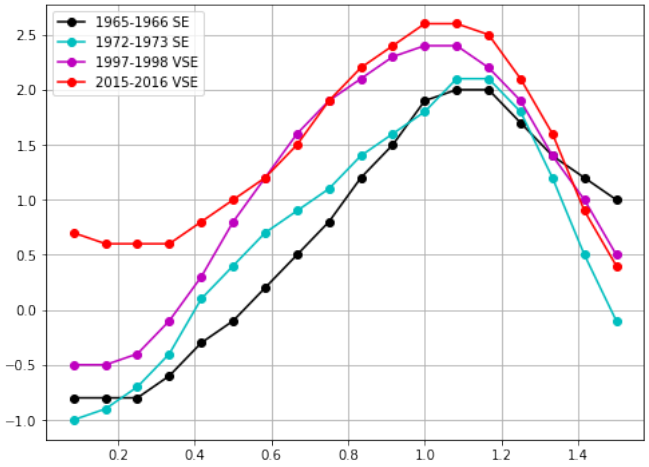


Figure 16: Shapes of group 6

7. Group 7 - contains 2 events that one of them is moderate (1968-69) and one is strong event (1991-92).

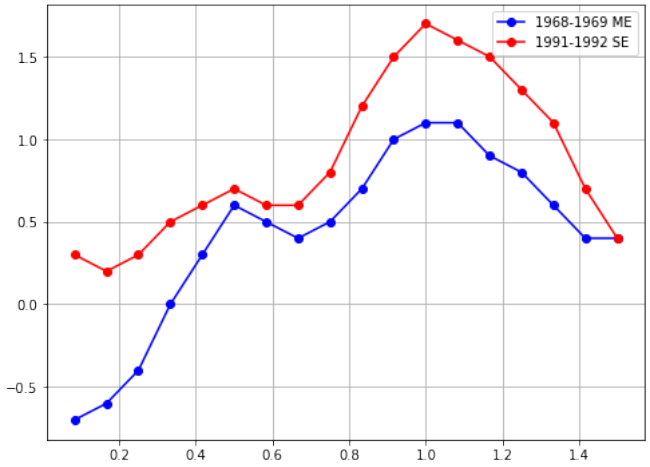


Figure 17: Shapes of group 7

8. Group 8 - contains 1 weak event (1953-54).

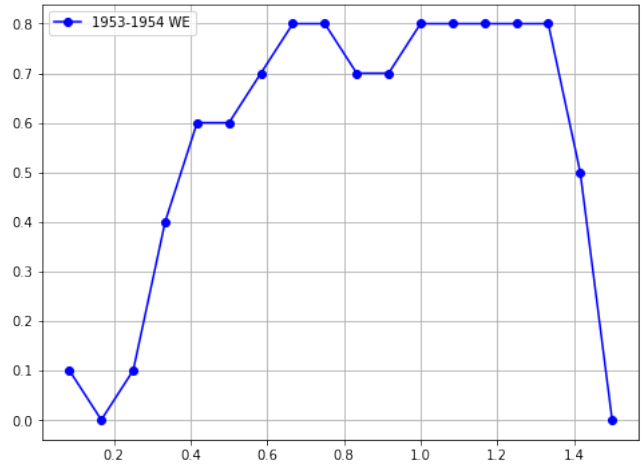


Figure 18: Shapes of group 8

5.3.3 Finding equation of average in the each group using GP

In this section good and simple equations were tried to find using EAs that corresponds to the average of shapes in each group. To make it easier for geophysicists to understand, these equations do not have multiplication of sines, sine inside sine and any division operator. Parallel Genetic Programming on island mode was used to do this work. The code of GP is shown in Annex E. GP takes average of 1.5 year data of events for each group and generates equation.

In the following, equation of average shape with its plotting graph for each group is shown :

1. Equation of average in group 1 :

$$eq_{avg} = 0.128129 * x * sin(8.640880 * x) + 0.870872 * x - 0.2798906 \quad (4)$$

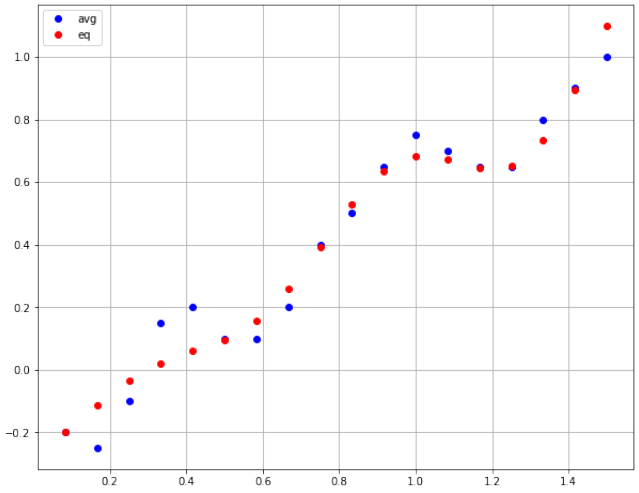


Figure 19: average shape with its equation shape in group 1

2. Equation of average in group 2 :

$$eq_{avg} = 1.189035 * sin(0.699276 * x) + (-0.920625 * x + 0.755611)$$

$$* sin(5.417646 * x) + 0.014571 \quad (5)$$

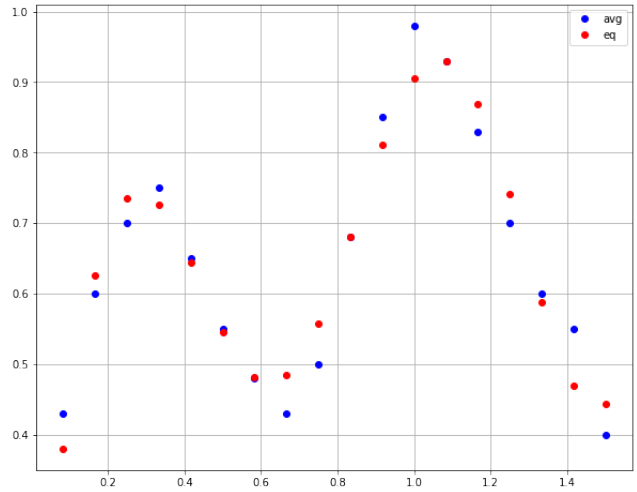


Figure 20: average shape with its equation shape in group 2

3. Equation of average in group 3 :

$$eq_{avg} = (1.4221796 - x) * (2 * x * (x - 0.995554) + (2 * x - 0.64533426) \quad (6)$$

$$* sin(x - 0.694731) + 1.0)$$

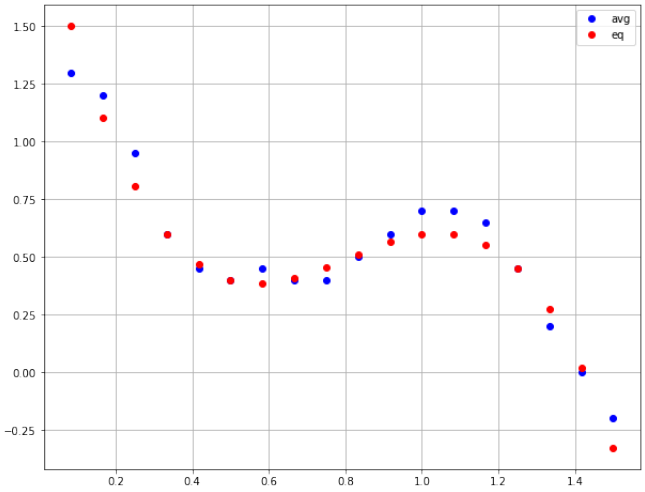


Figure 21: average shape with its equation shape in group 3

4. Equation of average in group 4 :

$$eq_{avg} = 0.660694 * x + x * sin(3.211015 * x - 1.065241) - 0.192678 \quad (7)$$

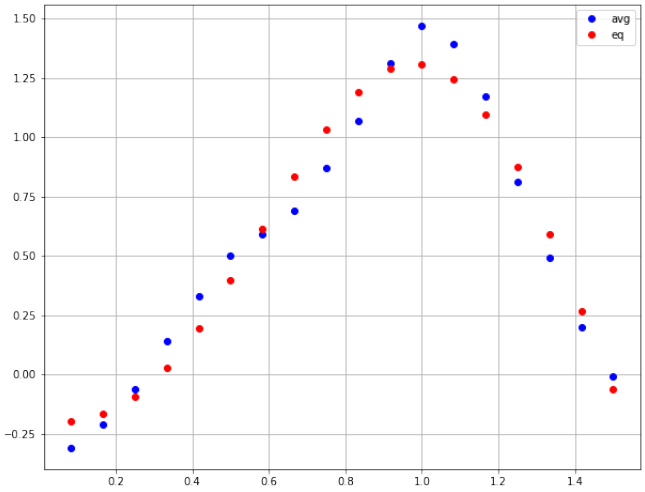


Figure 22: average shape with its equation shape in group 4

5. Equation of average in group 5 :

$$eq_{avg} = sin(2.892047 * x + 4.724232) \quad (8)$$

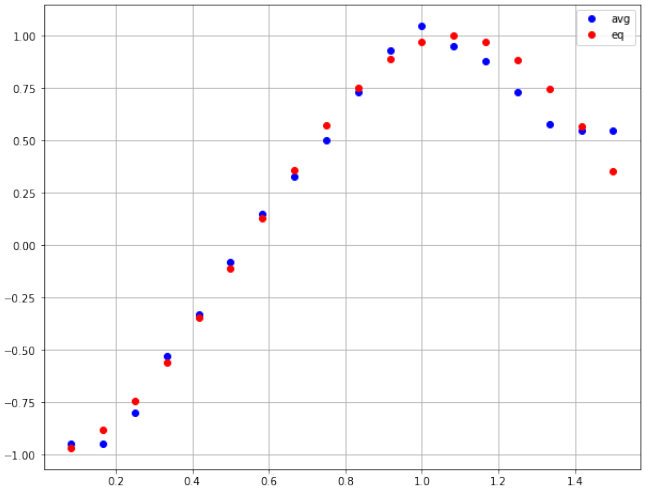


Figure 23: average shape with its equation shape in group 5

6. Equation of average in group 6 :

$$eq_{avg} = x + (-0.497150 - 3.264897 * x) * sin(2.908535 * x) \quad (9)$$

$$+5.213310 * x * sin(2.648657 * x) - 0.441066$$

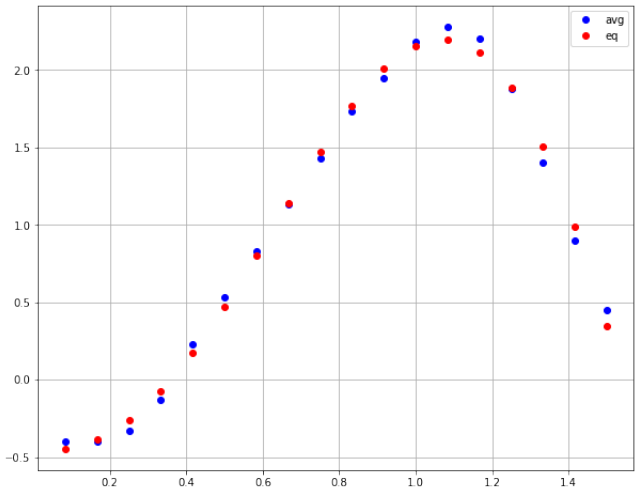


Figure 24: average shape with its equation shape in group 6

7. Equation of average in group 7:

$$eq_{avg} = 1.270450 * sin(0.869869 * x) + (0.873001 * x - 0.504249) * sin(7.622553 * x) \quad (10)$$

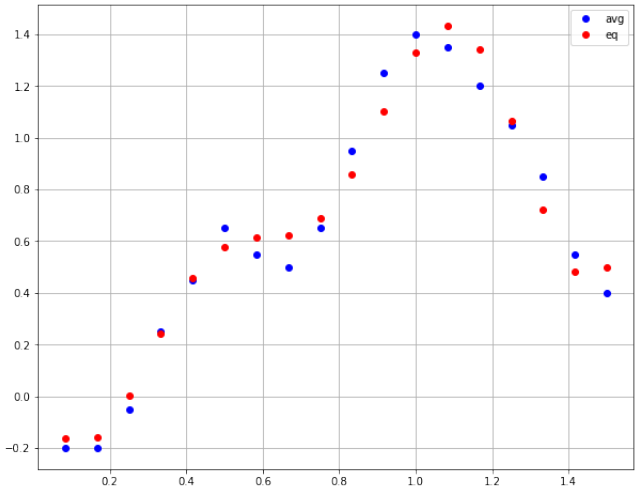


Figure 25: average shape with its equation shape in group 7

8. Equation of average in group 8:

$$eq_{avg} = -0.513958 * x - (4.298025 * x - 0.378910) * sin(x - 1.612762) \quad (11)$$
$$+(x - 2.151642) * x * sin(2.249024 * x)$$

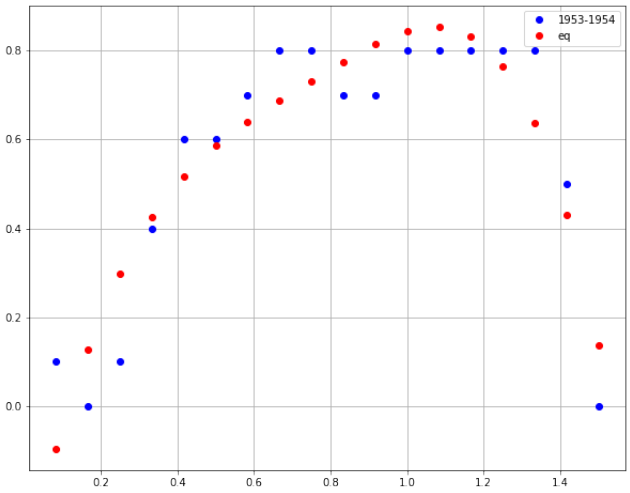


Figure 26: average shape with its equation shape in group 8

Because artificial evolution is stochastic, the results are never the same, so for statistical significance, the parallel GP run was repeated 20 times on 3 machines simultaneously on island model for each group and found 20 equations for each group with their best fitness values. The best fitness values in each group are displayed in violin plots (Figure 27). The wider sections of the violin plots show that most of the values are around that given value whereas the skinnier sections show that there are very few results around those values.

The lowest value of best fitness belongs to group 2, but best fitness interval of this group is the longest. In contrast, in group 7 best fitness interval is the smallest, but the lowest of its best fitness values is higher than other groups.

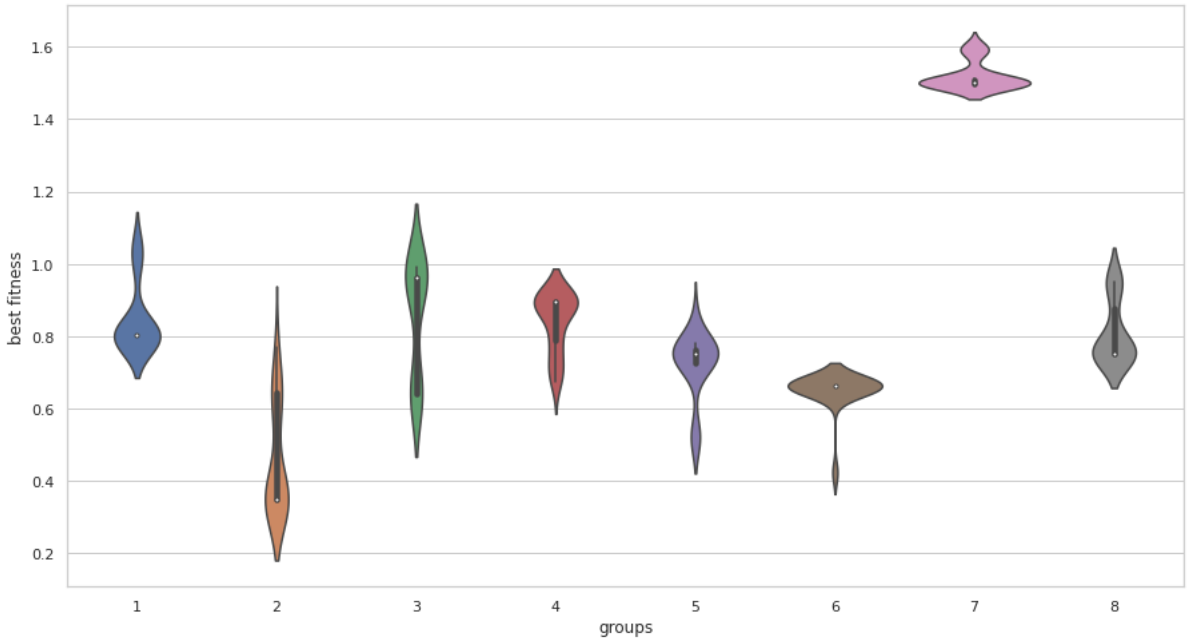


Figure 27: Violin plot of each group

5.3.4 Finding equation of each shape in the each group using RVGA

After to find average equation for each group, the next step is to find equation of each event's shape in the each group separately with changing some parameters of its average equation using Real Valued Genetic Algorithm. RVGA takes 1.5 year data of each event and generates equation. As Section 5.3.3 the parallel RVGA run was repeated 20 times on 3 machines simultaneously on island mode for each event and found 20 equations for each event with their best fitness values. The best fitness values of equations of events are displayed in violin plots in Annex B, but some events always get the same best fitness when repeating, so their best fitness plots are constant line. Then one equation with the lowest best fitness is chosen out of 20 equations for each event. The graph of each event for each group is shown Annex A .

In the following Tables [5.3.4 - 5.3.4], equations of El Nino events with their best fitness values are shown. There is one weak event in group 8, so equation of this event is the same with average equation of group.

Table 7: Equations of events in group 1

event	**function**	**best fitness**
avg	0.128129*x*sin(8.640880*x)+0.870872*x-0.2798906	0.80292
1952-53 WE	0.128129*x*sin(**8.046256***x)+0.870872*x-**0.228342**	10.079773
2014-15 WE	**0.090792***x*sin(8.640880*x)+0.870872*x-**0.326393**	13.7775

Table 8: Equations of events in group 2

event	**function**	**best fitness**
avg	1.189035*sin(0.699276*x)+(-0.920625*x+0.755611) *sin(5.417646*x) +0.014571	0.645371
1969-70 WE	**0.754583***sin(**1.650590***x)+(-0.920625*x+**0.991321**) *sin(5.417650*x)+0.014571	10.5425
1979-80 WE	**8.447183***sin(**0.115486***x)+(-0.920625*x+**0.731570**) *sin(5.417650*x)+**(-0.483086)**	10.750963
1987-88 SE	**1.440270***sin(**1.468600***x)+(-0.920625*x+**0.845974**) *sin(5.417650*x)+0.014571	10.6685
2004-05 WE	**7.800660***sin(**0.115838***x)+(-0.920625*x+**0.782986**) *sin(5.417650*x)+**(-0.296717)**	7.96481

Table 9: Equations of events in group 3

event	**function**	**best fitness**
avg	(1.4221796-x)*(2*x*(x-0.995554)+(2*x-0.64533426) *sin(x-0.694731)+1.0)	0.642469
1958-59 WE	(1.422180-x)*(2*x*(x-0.995554)+(2*x-**1.339720**) *sin(x-0.694731)+1.0)	11.8577
1977-78 WE	(1.422180-x)*(2*x*(x-0.995554)+(2*x-**(-0.114362)**) *sin(x-0.694731)+1.0)	10.795051

Table 10: Equations of events in group 4

event	**function**	**best fitness**
avg	0.660694*x+x*sin(3.211015*x-1.065241)-0.192678	0.888309
1957-58 SE	**1.225919***x+x*sin(3.211010*x-**0.332064**)-**(-0.114485)**	16.024298
1963-64 ME	**0.590984***x+x*sin(3.211015*x-**0.908812**)-0.192678	18.105898
1982-83 VSE	**1.028580***x+x*sin(3.211010*x-**1.417940**)-**0.039754**	15.245098
1994-95 ME	**-0.370771***x+x*sin(3.211010*x-**1.542475**)-**(-0.396837)**	15.751081
2002-03 ME	**0.594481***x+x*sin(3.211010*x-**0.952424**)-0.192678	6.638339
2006-07 WE	0.660694*x+x*sin(3.211010*x-**1.164224**)-**0.756267**	11.641994
2009-10 ME	**1.109158***x+x*sin(**3.366371***x-1.065241)-**0.645433**	23.579988

Table 11: Equations of events in group 5

event	**function**	**best fitness**
avg	sin(2.892047*x+4.724232)	0.732427
1951-52 ME	sin(**2.980902***x+**4.894319**)	13.713657
1976-77 WE	sin(**3.486610***x+**4.227690**)	26.215099
1986-87 ME	sin(**1.958630***x+**(-0.820861)**)	16.749201
2018-19 WE	sin(**2.955150***x+**4.485220**)	11.6058

Table 12: Equations of events in group 6

event	**function**	**best fitness**
avg	x+(-0.497150-3.264897*x)*sin(2.908535*x) +5.213310*x*sin(2.648657*x)-0.441066	0.638266
1965-66 SE	x+(**-1.145090**-3.264897*x)*sin(**2.849602***x) +5.213310*x*sin(2.648657*x)-0.441066	11.954185
1972-73 SE	x+(**-0.290019**-3.264900*x)*sin(**2.981270***x) +5.213310*x*sin(2.648660*x)-**0.924186**	14.265
1997-98 VSE	x+(**-0.220951**-3.264897*x)*sin(**2.973420***x) +5.213310*x*sin(2.648657*x)-**0.488703**	15.091798
2015-16 VSE	x+(**-11.884934**-3.264900*x)*sin(**0.300947***x) +5.213310*x*sin(**1.588590***x)-**(-0.750941)**	10.043792

Table 13: Equations of events in group 7

event	function	best fitness
avg	1.270450*sin(0.869869*x)+(0.873001*x-0.504249) *sin(7.622553*x)	1.49723
1968-69 ME	1.270450*sin(**0.607653***x)+(0.873001*x-**0.671472**) *sin(7.622553*x)	15.224091
1991-92 SE	1.270450*sin(**1.187490***x)+(0.873001*x-**0.368266**) *sin(7.622550*x)	15.917866

The idea behind this method is to find similar equation for all events within the same group and look for the pattern of different parameters changing. If there is a pattern, this would allow us to classify the strength of the events for the given shape category. However, the results don't show an obvious pattern and this needs to be investigated further. In the next section, ML clustering techniques are used for El Nino events to form new groups by shape.

5.3.5 Grouped El Nino events according to shapes automatically by ML

After classified El Nino events according to their shapes by eye, in this section they were grouped automatically using ML. Two clustering algorithms (KMeans and Agglomerative) are used to detect these events by shapes. The code of these clusters is shown in Annex D. The first thing is to find the better K number. Therefore, silhouette coefficient is computed for each K number to choose a better k number from the interval of k numbers [2,10]. K number is selected as 7 that its silhouette coefficient has higher score. Then, El Nino events are clustered. When groups are done by 2 methods, more restrictive groups which contain lesser events are keeped and new clustering is done on remain shapes again. These groups are as follows:

1. Group 1 - contains 3 strong events (1957-58, 1987-88, 1991-92).

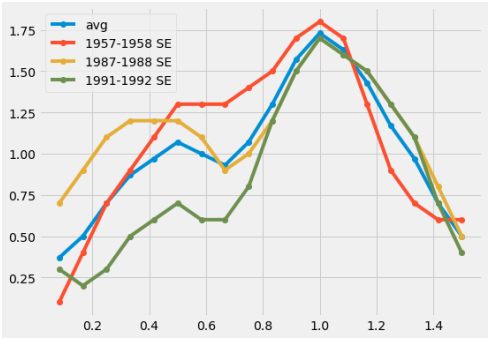


Figure 28: Shapes of group 1

2. Group 2 - contains 2 strong events (1965-66, 1972-73).

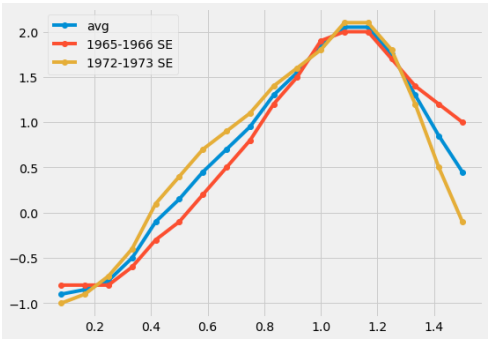


Figure 29: Shapes of group 2

3. Group 3 - contains 3 weak events (1969-70, 1958-59, 1977-78).

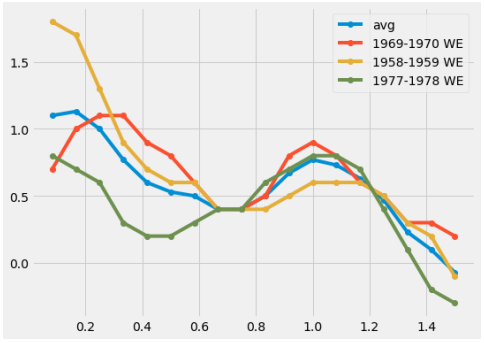


Figure 30: Shapes of group 3

4. Group 4 - contains 3 very strong events (1982-83, 1997-98, 2015-16).

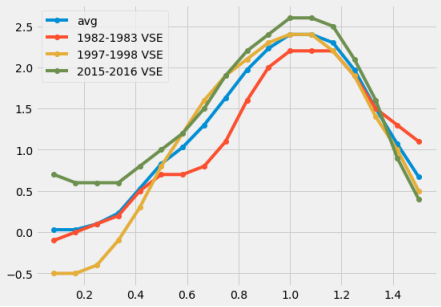


Figure 31: Shapes of group 4

5. Group 5 - contains 4 events that 3 of them are weak (1979-80, 2004-05, 2014-15) and one is moderate event (1952-53).

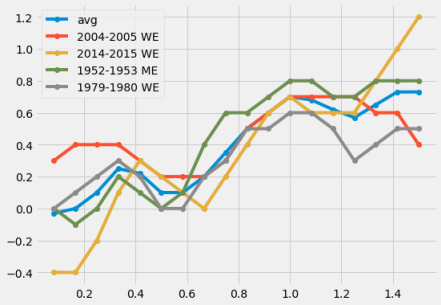


Figure 32: Shapes of group 5

6. Group 6 - contains 4 events that 3 of them are weak (1976-77, 2006-07, 2018-19) and one is moderate event (1951-52).

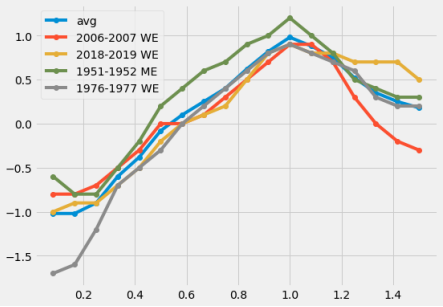


Figure 33: Shapes of group 6

7. Group 7 - contains 3 moderate events (1963-64, 2002-03, 2009-10).

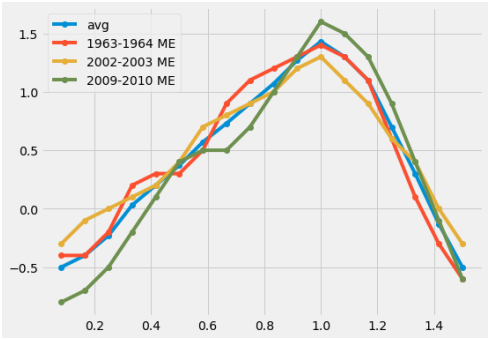


Figure 34: Shapes of group 7

8. Group 8 - contains 2 moderate events (1968-69,1986-87).

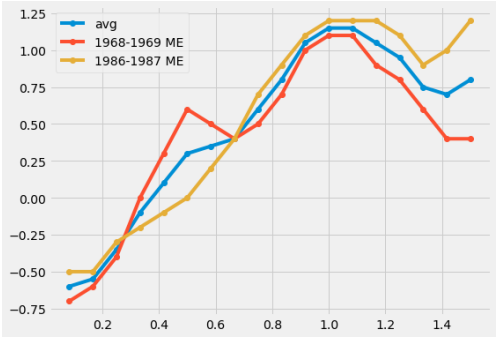


Figure 35: Shapes of group 8

9. Group 9 - contains 1 weak (1953-54) and 1 moderate event (1994-95).

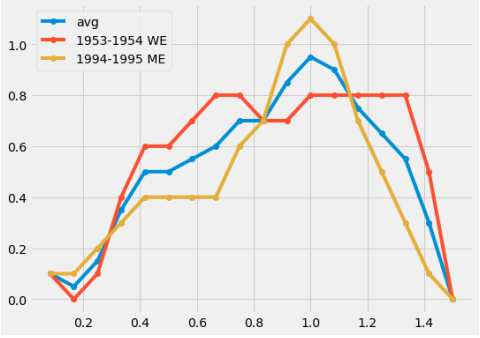


Figure 36: Shapes of group 9

When some of strong and very strong El Nino events, also some of moderate, strong and some of weak events are in the same groups in the previous method done by eye, here all of strong, all of very strong, most of weak and most of moderate events are in the different groups respectively like in the literature. Only a few of weak and a few of moderate events are together.

Table 14: Comparing classification of El Nino events by 2 methods

Grouped by eye		**Grouped by ML**	
Groups	Events	Groups	Events
1	52-53 WE, 14-15 WE	1	57-58 SE, 87-88 SE, 91-92 SE
2	69-70 WE, 79-80 WE, 87-88 SE 04-05 WE	2	65-66 SE, 72-73 SE
3	58-59 WE, 77-78 WE	3	69-70 WE, 58-59 WE, 77-78 WE
4	57-58 SE, 63-64 ME, 82-83 VSE 94-95 ME, 02-03 ME, 06-07 WE 09-10 ME	4	82-83 VSE, 97-98 VSE, 15-16 VSE
5	51-52 ME, 76-77 WE, 86-87 ME 18-19 WE	5	52-53 ME, 79-80 WE, 04-05 WE 14-15 WE
6	65-66 SE, 72-73 SE, 97-98 VSE 15-16 VSE	6	51-52 ME, 76-77 WE, 06-07 WE 18-19 WE
7	68-69 ME, 91-92 SE	7	63-64 ME, 02-03 ME, 09-10 ME
8	53-54 WE	8	68-69 ME, 86-87 ME
		9	53-54 WE, 94-95 ME

In Table 14 it is seen that, group 5 by ML is similar to combination of group 1 and 2 by eye. Group 3 is similar in both. Group 5 by eye is similar to group 6 by ML. Group 6 by eye is similar to combination of group 2 and 4 by ML. Group 7 by ML is subset of group 4 by eye.

Grouped by ML is seen better than grouped by eye, because groups by ML are similar to groups in the literature that classification according to shapes by ML is almost the same with classification by strength. Groups 1 and 2 by ML are only strong events, group 4 is very strong events, group 7 is moderate events, group 3 is weak events and others are mixed weak and moderate events.

5.3.6 Grouped La Nina events according to shapes automatically by ML

In this section La Nina events are grouped automatically using ML. These events are clustered with KMeans and Agglomerative methods like Section 5.3.5 where K number is selected as 7. When groups are done by 2 methods, more restrictive groups which contain lesser events are keeped and new analysis is done on remain shapes. These groups are as follows:

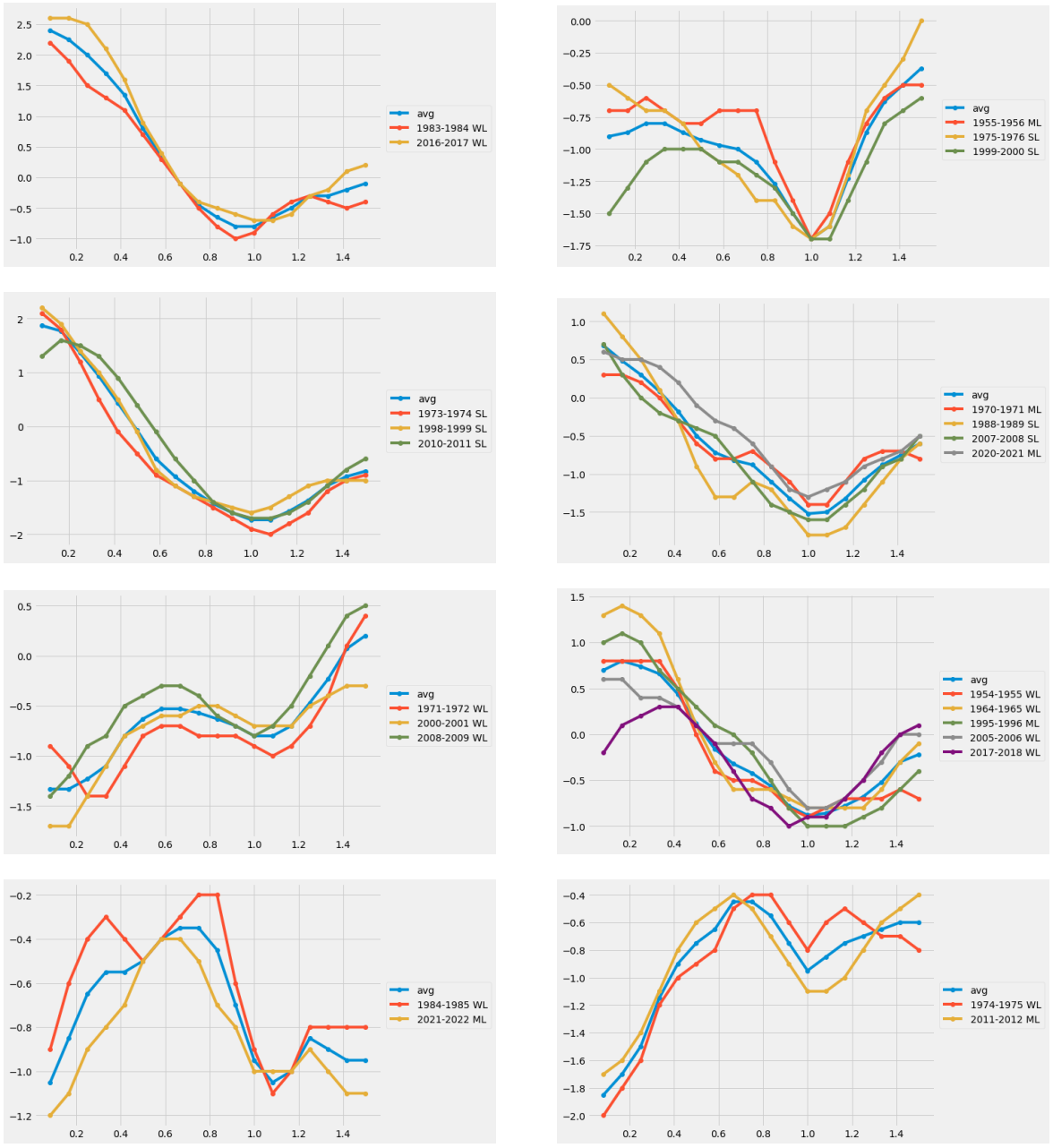


Figure 37: Groups

In comparing to the definition by literature, here some of strong and some of moderate, also some of moderate and some of weak events are in the same groups. There are also some groups that have only strong events or weak events.

6 Discussion

The strength of El Nino and La Nina events were classified according to their ONI values with six ML algorithms. When Logistic Regression gives a minimum accuracy of more than 80%, the maximum accuracy is achieved by the Random Forest and Decision Tree algorithms for all data sets.

Then El Nino events were classified according to their shapes by eye. 8 groups were gotten where some of strong, very strong events are together and some of moderate, weak events are in the same group. The similar equations were found for all events in the same group with changing different parameters using EAs. These equations can be used to determine the strength of events. However, a pattern is not found for the parameters of the equation. Thus, this work needs further investigation. So, to find the new easier equations with better fitness values which can define the strength of events can be future perspectives. Also, predicting future events was not in the scope of this research project and will be studied in the future.

Therefore, El Nino and La Nina events were classified according to their shapes automatically by ML clustering methods (KMeans and Agglomerative Clustering). El Nino events were distributed to 9 groups where strong, very strong, most of weak and moderate events are in the different groups respectively. La Nina events were distributed to 8 groups where some of strong and some of weak events are in the different groups when there are some moderate, some weak events in one group. This classification looks better than classification done by eye. Because the same categorical events is almost in the same group like in the literature. The equations for this automatic classification have not yet been found and is in the future perspectives.

7 Conclusion

This work presents an approach classifying El Nino and La Nina events and applying EAs to get equations for each event. These equations allow us to know the strength of the event and to which group it belongs. Classification was done by two methods according to shapes. First method was done semi-automatically. Only El Nino events were classified in this method. Events with similar shapes were grouped together by eye. Second method was done automatically by ML clustering methods. In this method, both El Nino and La Nina events were classified. General equations were found for each group using GP and the same form equations were found for all events in the same group by changing different parameters of general equations using RVGA. These algorithms are reproducible using seed values we can reproduce the same results. Each parameter in the equation has its own purpose to distinguish the events from each other.

In this work it was shown that El Nino and La Nina events can be model using evolutionary algorithms. The advantage of using Genetic Programming to model these events is that GP also returns an equation to describe the event which can be very useful in the prediction of future events.

In this thesis, only the equations of El Nino events which grouped by eye are shown. So, to find the equations of El Nino and La Nina events using EAs which grouped automatically by ML methods can be my future work.

As conclusion, global climate is one of the important complex systems. No matter how much climate science is studied, the exploration of its depths will continue for a long time.

References

[1] NOAA/ National Weather Service, National Centers for Environmental Prediction, Climate Prediction Center, 5830 University Research Court, College Park, Maryland 20740

[2] J. Holland, “Adaptation in natural and artificial systems”, University of Michigan press, Ann Arbor, 1975.

[3] Back, T. 1996, Evolutionary Algorithms in Theory and Practice (Oxford: Oxford University).

[4] Andre, David and Astro Teller. ”Evolving team Darwin United.” In RoboCup-98: Robot Soccer World Cup II, Minoru Asada and Hiroaki Kitano (eds). Lecture Notes in Computer Science, vol.1604, p.346-352. Springer-Verlag, 1999.

[5] Coello, Carlos. ”An updated survey of GA-based multiobjective optimization techniques.” ACM Computing Surveys, vol.32, no.2, p.109-143 (June 2000).

[6] GPGPU homepage.

[7] D. Castano-Díez, D. Moser, A. Schoenegger, S. Pruggnaller, A. S. Frangakis, “Performance evaluation of image processing algorithms on the GPU,” Journal of Structural Biology, vol. 164, no.1, pp. 153-160, 2008.

[8] Qin, Kai & Raimondo, Federico & Forbes, Florence & Ong, Yew. (2012). An Improved CUDA-Based Implementation of Differential Evolution on GPU. GECCO’12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation.

[9] NVIDIA homepage

[10] NVIDIA, CUDA 2.0 Programming Guide, NVIDIA_CUDA_Programming_Guide_2.0.pdf.

[11] NVIDIA CUDA C Programming Guide Version 4.0. CUDA_C_Programming_Guide.pdf.

[12] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In Proceedings Of an International Conference on Genetic Algorithms and their Applications, pages 183–187, 1985.

[13] J. R. Koza. Genetic Programming: On the Programming of Computers by means of Natural Evolution. MIT Press, Massachusetts, 1992.

[14] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.

[15] I. Rechenberg. Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution. Fromman-Hozlboog Verlag, Stuttgart, 1973.

[16] H. P. Schwefel. Numerical Optimization of Computer Models. John Wiley & Sons, New-York, 1981. 1995 – 2^{nd} edition.

[17] P. Collet and J. P. Rennard. Stochastic optimization algorithms. Handbook of Research on Nature Inspired Computing for Economics and Management, 04 2007.

[18] M. Kommenda, G. Kronberger, St. Winkler, M. Affenzeller, S. Wagner, L. Schickmair, and B. Lindner. Application of genetic programming on temper mill datasets. pages 58–62, 09 2009. doi: 10.1109/LINDI.2009.5258766.

[19] Ramnik Arora, Rupesh Tulshyan, Kalyanmoy Deb, Parallelization of Binary and Real-Coded Genetic Algorithms on CUDA, 2010.

[20] Cheng, J. and Gen, M. (2020). Parallel Genetic Algorithms with GPU Computing. IntechOpen.

[21] Collet, P., Lutton, E., Schoenauer, M., Louchet, J.: Take it easea. In: Informatics: 10 Years Back, 10 Years Ahead. LNCS, pp. 891–901. Springer, Heidelberg (2000)

[22] Philander, S.G.H., 1990: El Nino, La Nina and the Southern Oscillation. Academic Press, San Diego, CA, 289 pp.

[23] Bjerknes, J. (March 1969) ”Atmospheric teleconnections from the equatorial Pacific,” Monthly Weather Review, 97 (3) : 163–172.

[24] Columbia Climate School, International Research Institute for climate and society, ENSO_Info.

[25] Nicola Maher, Thibault P. Tabarin, and Sebastian Milinski, Combining machine learning and SMILEs to classify, better understand, and project changes in ENSO events, https://doi.org/10.5194/esd-2021-105 , 01 2022

[26] Henk A. Dijkstra, Paul Petersik, Emilio Hernández-García and Cristóbal López, The Application of Machine Learning Techniques to Improve El Niño Prediction Skill, doi:10.3389/fphy.2019.00153, 10 2019

[27] J. Fan, J. Meng, Y. Ashkenazy, S. Havlin, and H. J. Schellnhuber, Network analysis reveals strongly localized impacts of EL Niño, Proc. Nat. Acad. Sci. USA, vol. 114, no. 29, pp. 7543–7548, Jul. 2017.

[28] K. Yamasaki, A. Gozolchiani, and S. Havlin, Climate networks around the globe are significantly affected by EL Niño, Phys. Rev. Lett., vol. 100, no. 22, Jun. 2008, Art. no. 228501.

[29] J. Rao and R. Ren, Parallel comparison of the 1982/83, 1997/98 and 2015/16 super el Niños and their effects on the extratropical stratosphere, Adv. Atmos. Sci., vol. 34, no. 9, pp. 1121–1133, Sep. 2017.

[30] N. Boers, B. Goswami, A. Rheinwalt, B. Bookhagen, B. Hoskins, and J. Kurths, Complex networks reveal global pattern of extreme-rainfall teleconnections, Nature, vol. 566, no. 7744, pp. 373–377, Feb. 2019.

[31] P. Cashin, K. Mohaddes, and M. Raissi, Fair weather or foul? The macroeconomic effects of El Niño, J. Int. Econ., vol. 106, pp. 37–54, Nov. 2017.

[32] V. B. Rao, K. Maneesha, P. Sravya, S. H. Franchito, H. Dasari, and M. A. Gan, Future increase in extreme EL Niño events under greenhouse warming increases Zika virus incidence in south America, NPJ Climate Atmos. Sci., vol. 2, no. 1, p. 4, Dec. 2019.

[33] S. M. Hsiang, K. C. Meng, and M. A. Cane, Civil conflicts are associated with the global climate, Nature, vol. 476, no. 7361, pp. 438–441, Aug. 2011.

[34] A. G. Barnston, M. K. Tippett, M. Ranganathan, and M. L. L'Heureux, Deterministic skill of ENSO predictions from the north american multimodel ensemble, Climate Dyn., vol. 53, no. 12, pp. 7215–7234, Dec. 2019.

[35] A. G. Barnston, M. K. Tippett, M. L. L'Heureux, S. Li, and D. G. DeWitt, Skill of real-time seasonal enso model predictions during 2002-11 is our capability increasing? Bull. Amer. Meteorol. Soc., vol. 93, no. 5, pp. 631–651, 2012.

[36] J. Ludescher, A. Gozolchiani, M. I. Bogachev, A. Bunde, S. Havlin, and H. J. Schellnhuber, Improved EL Niño forecasting by cooperativity detection, Proc. Nat. Acad. Sci. USA, vol. 110, no. 29, pp. 11742–11745, Jul. 2013.

[37] J. Meng, J. Fan, Y. Ashkenazy, A. Bunde, and S. Havlin, Forecasting the magnitude and onset of EL Niño based on climate network, New J. Phys., vol. 20, no. 4, 2018, Art. no. 043036.

[38] Y.-G. Ham, J.-H. Kim, and J.-J. Luo, Deep learning for multi-year ENSO forecasts, Nature, vol. 573, no. 7775, pp. 568–572, Sep. 2019.

[39] P. Cashin, K. Mohaddes, and M. Raissi, ''Fair weather or foul? The acroeconomic effects of El Niño,'' J. Int. Econ., vol. 106, pp. 37–54, Nov. 2017.

[40] V. B. Rao, K. Maneesha, P. Sravya, S. H. Franchito, H. Dasari, and M. A. Gan, ''Future increase in extreme EL Niño events under greenhouse warming increases Zika virus incidence in south America,'' NPJ Climate Atmos. Sci., vol. 2, no. 1, p. 4, Dec. 2019.

[41] S. M. Hsiang, K. C. Meng, and M. A. Cane, ''Civil conflicts are associated with the global climate,'' Nature, vol. 476, no. 7361, pp. 438–441, Aug. 2011.

[42] L. Santos, L. Londe, T. Carvalho, D. Menasché, and D. Vega-Oliveros, About Interfaces Between Machine Learning, Complex Networks, Survivability Analysis, and Disaster Risk Reduction. Cham, Switzerland: Springer, 2019, pp. 185–215.

[43] Domingos, Pedro; Pazzani, Michael (1997). "On the optimality of the simple Bayesian classifier under zero-one loss". Machine Learning. 29 (2/3): 103–137. doi:10.1023/A:1007413511361.

[44] Webb, G. I.; Boughton, J.; Wang, Z. (2005). "Not So Naive Bayes: Aggregating One-Dependence Estimators". Machine Learning. 58 (1): 5–24. doi:10.1007/s10994-005-4258-6.

[45] Kaminski, B.; Jakubczyk, M.; Szufel, P. (2017). ”A framework for sensitivity analysis of decision trees”. Central European Journal of Operations Research. 26 (1): 135–159. doi:10.1007/s10100-017-0479-6.

[46] R. Quinlan, ”Learning efficient classification procedures”, Machine Learning: an artificial intelligence approach, Michalski, Carbonell and Mitchell (eds.), Morgan Kaufmann, 1983, p. 463–482. doi:10.1007/978-3-662-12405-5_15.

[47] Prinzie A, Poel D (2007). ”Random Multiclass Classification: Generalizing Random Forests to Random MNL and Random NB”. Database and Expert Systems Applications. Lecture Notes in Computer Science. Vol. 4653. p. 349. doi:10.1007/978-3-540-74469-6_35.

[48] Prinzie, A., Van den Poel, D. (2008). ”Random Forests for multiclass classification: Random MultiNomial Logit”. Expert Systems with Applications. 34 (3): 1721–1732. doi:10.1016/j.eswa.2007.01.029.

[49] Coomans, Danny; Massart, Desire L. (1982). ”Alternative k-nearest neighbour rules in supervised pattern recognition : Part 1. k-Nearest neighbour classification by using alternative voting rules”. Analytica Chimica Acta. 136: 15–27. doi:10.1016/S0003-2670(01)95359-0.

[50] Altman, Naomi S. (1992). ”An introduction to kernel and nearest-neighbor nonparametric regression”. The American Statistician. 46 (3): 175–185.

[51] Cristianini, Nello; Shawe-Taylor, John (2000). An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press. ISBN 0-521-78019-5.

[52] Cramer, J. S. (2002). The origins of logistic regression (Technical report). Vol. 119. Tinbergen Institute. pp. 167–178. doi:10.2139/ssrn.360300

[53] U. Abdulkarimova, A. Leonteva, C. Rolando, A. Jeannin-Girardon, P. Collet, ”The PARSEC Machine: a Non-Newtonian Supra-linear Supercomputer”, Azerbaijan Journal of High Performance Computing, Vol. 2, Issue 2, 2019, pp. 122-140

A Annex A

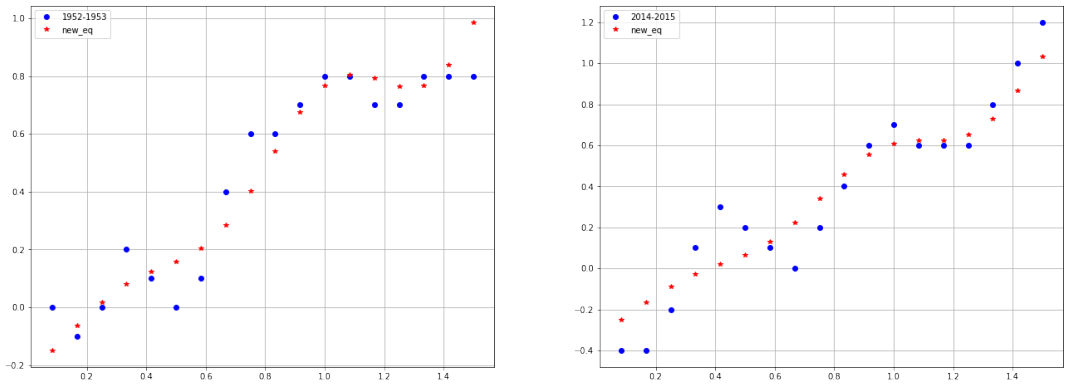


Figure 38: Shapes of events in group 1

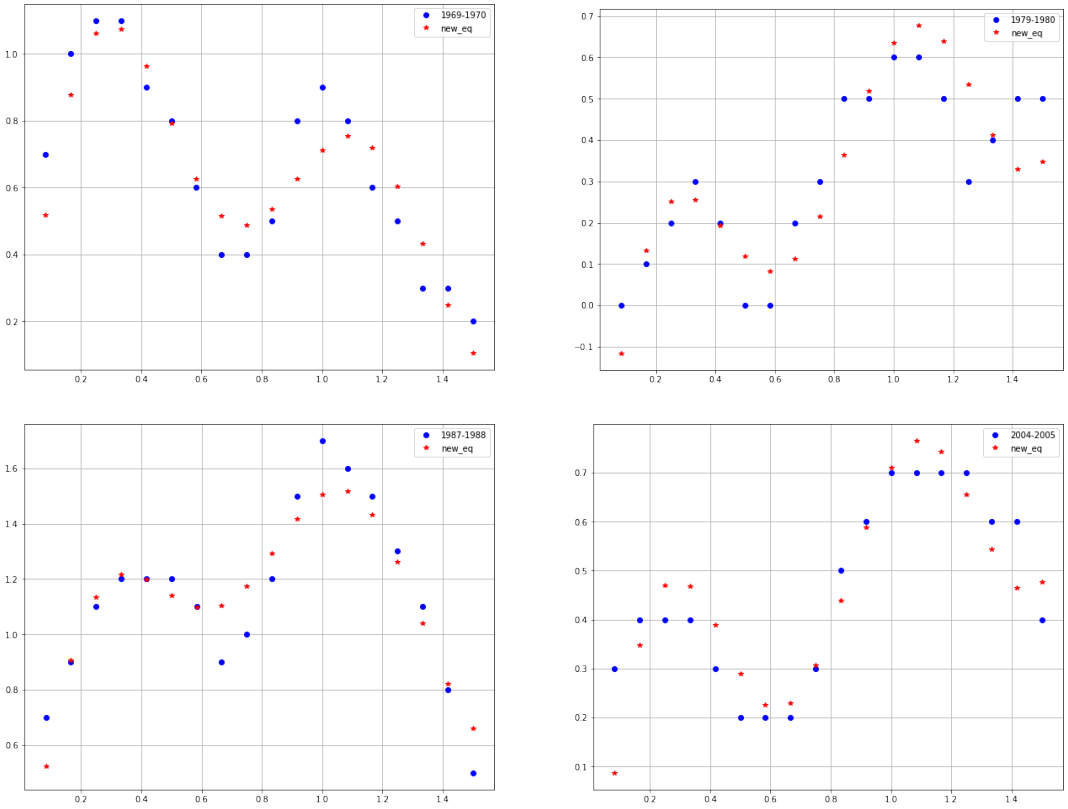


Figure 39: Shapes of events in group 2

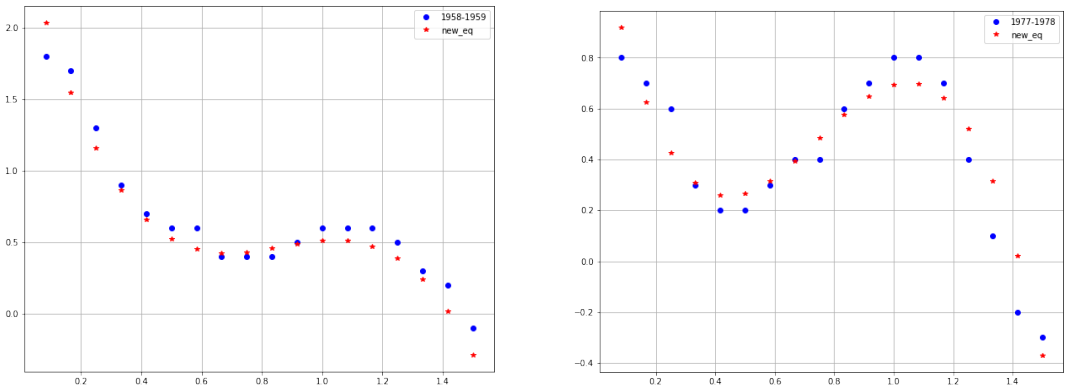


Figure 40: Shapes of events in group 3

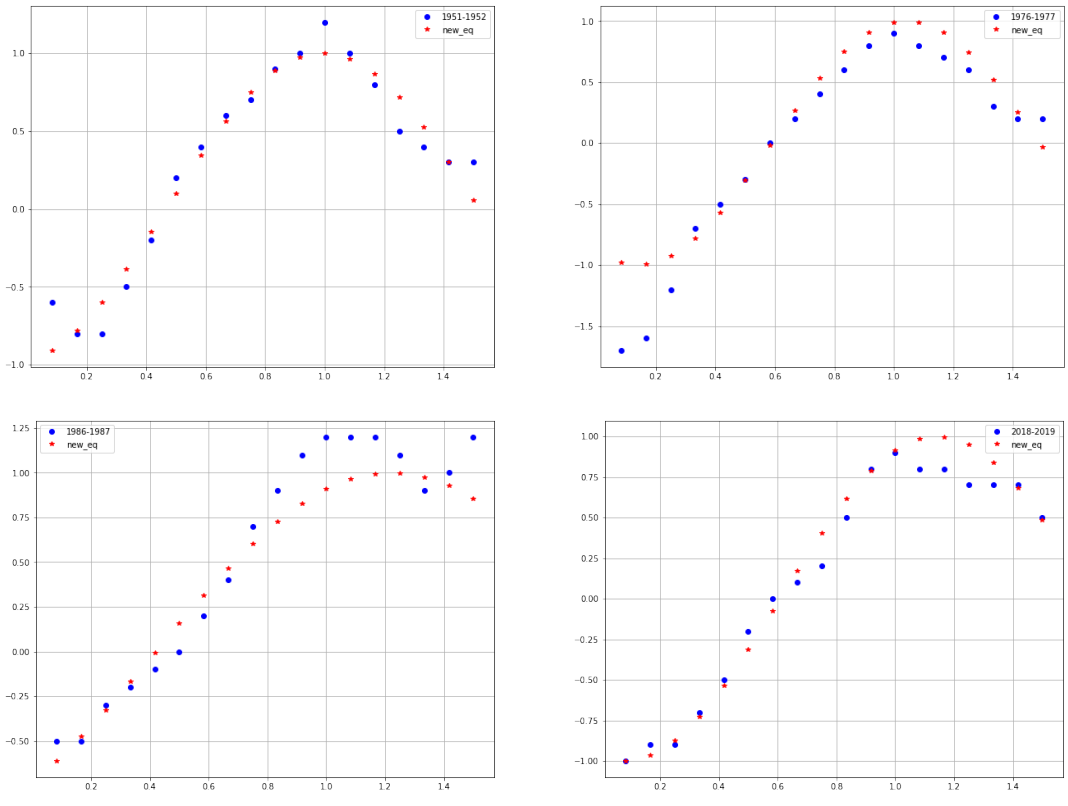


Figure 41: Shapes of events in group 5

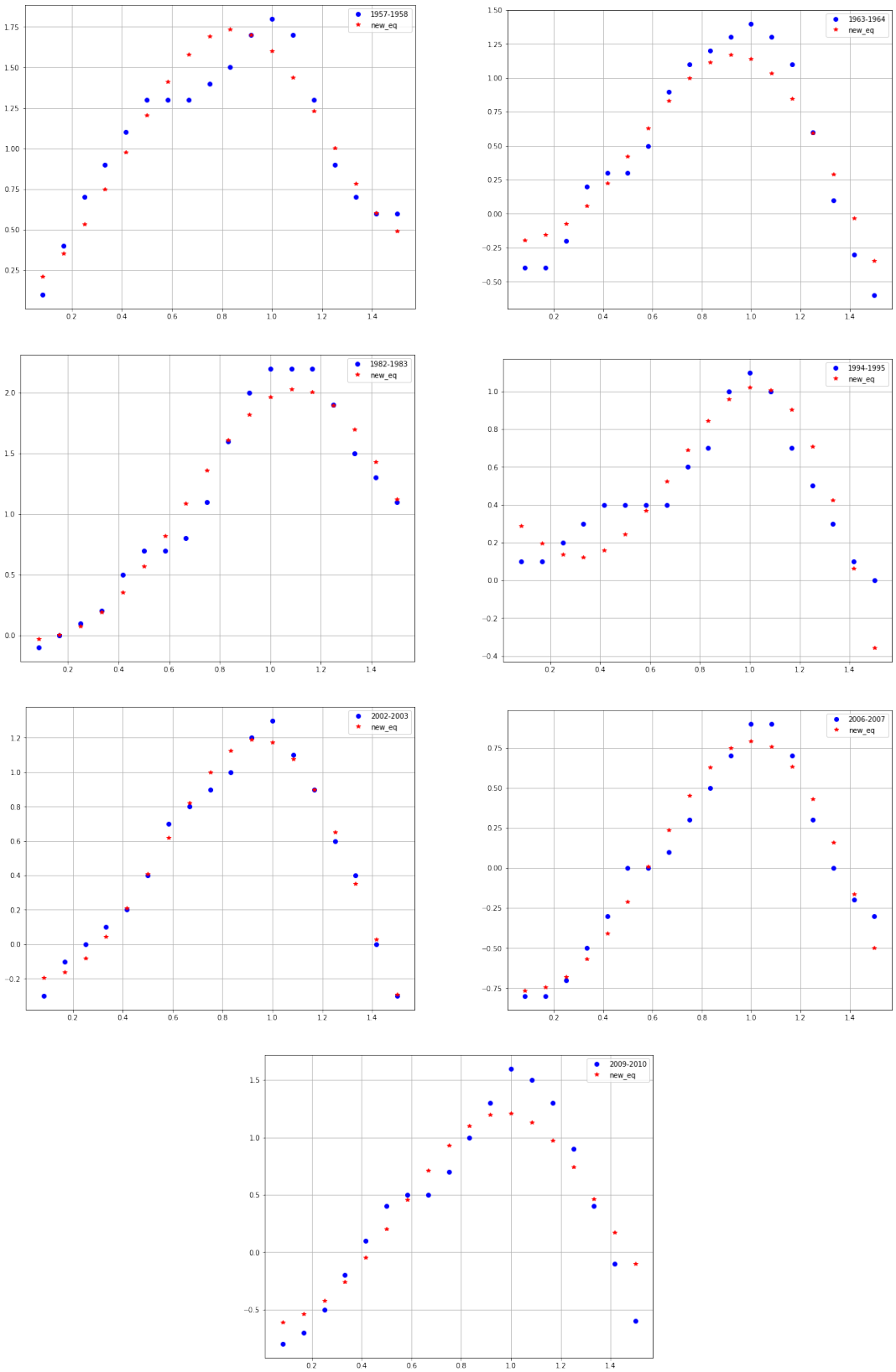


Figure 42: Shapes of events in group 4

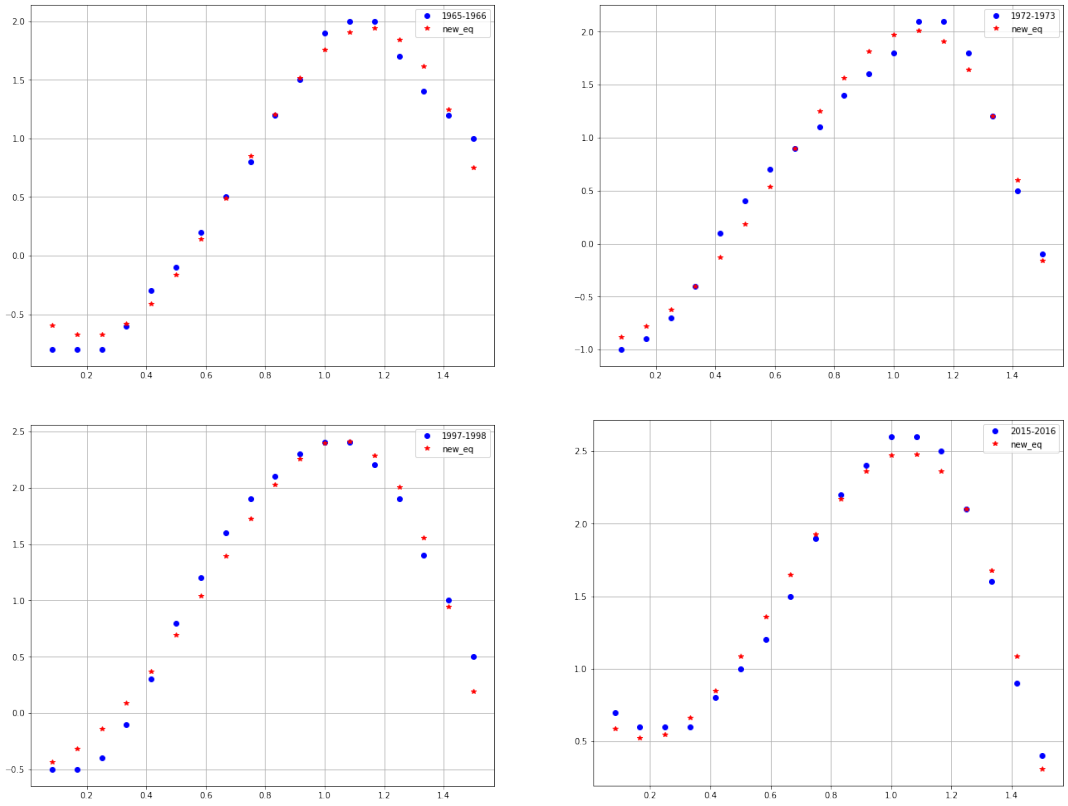


Figure 43: Shapes of events in group 6

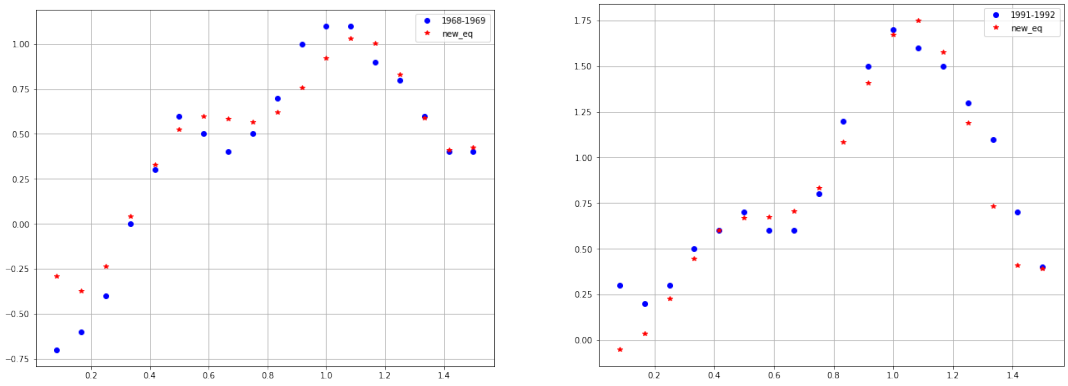
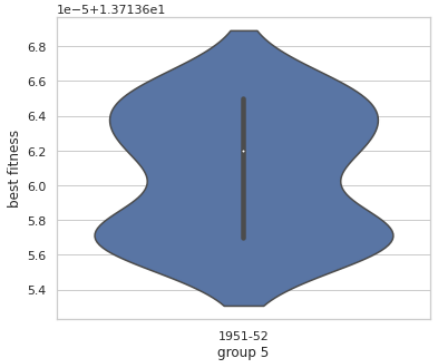
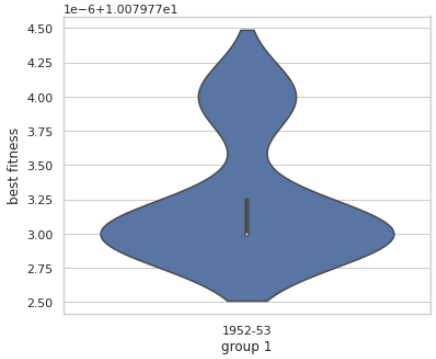
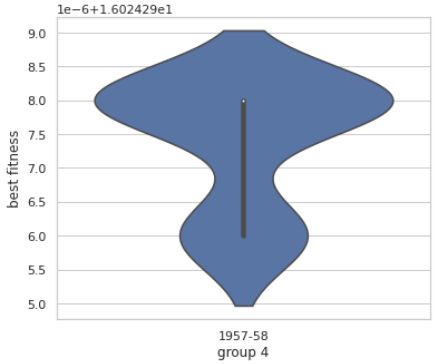


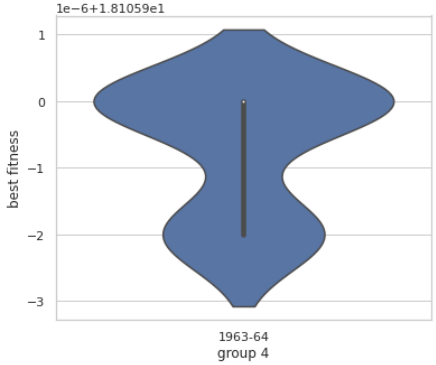
Figure 44: Shapes of events in group 7

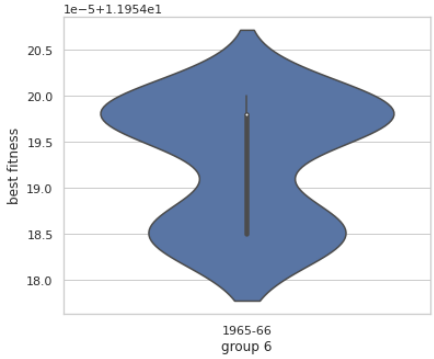
B Annex B

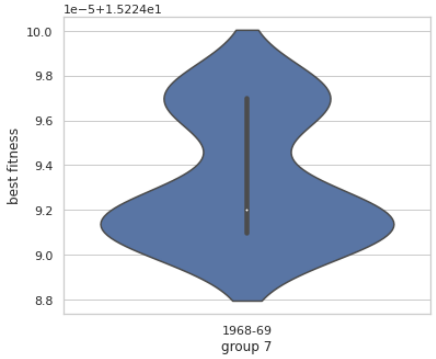

1e−5+1.37136e1
best fitness
6.8
6.6
6.4
6.2
6.0
5.8
5.6
5.4
1951-52
group 5

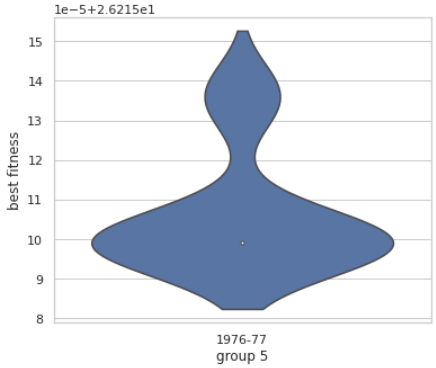

1e−6+1.007977e1
best fitness
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2.50
1952-53
group 1

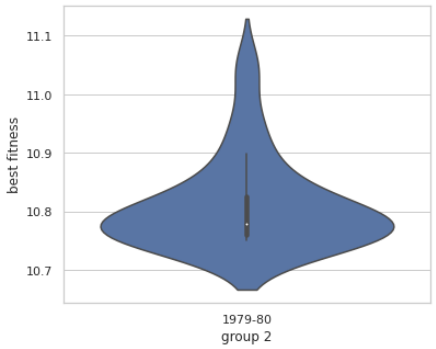

1e−6+1.602429e1
best fitness
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
1957-58
group 4

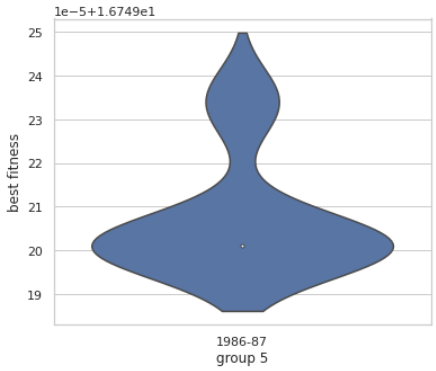

1e−6+1.81059e1
best fitness
1
0
−1
−2
−3
1963-64
group 4

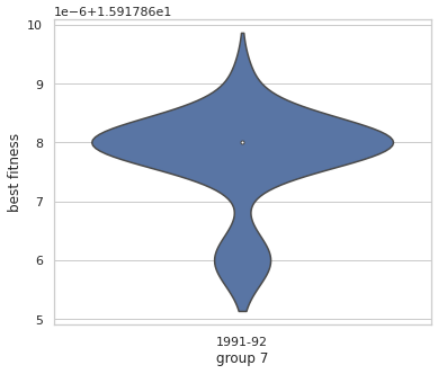

1e−5+1.1954e1
best fitness
20.5
20.0
19.5
19.0
18.5
18.0
1965-66
group 6

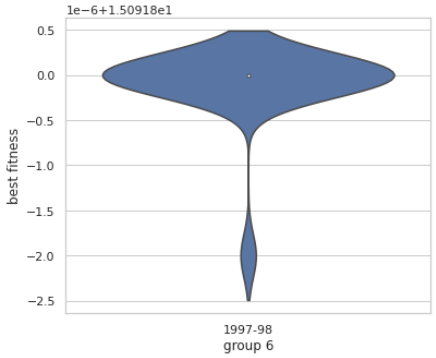

1e−5+1.5224e1
best fitness
10.0
9.8
9.6
9.4
9.2
9.0
8.8
1968-69
group 7

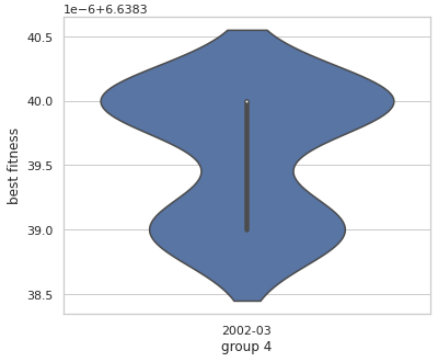

1e−5+2.6215e1
best fitness
15
14
13
12
11
10
9
8
1976-77
group 5

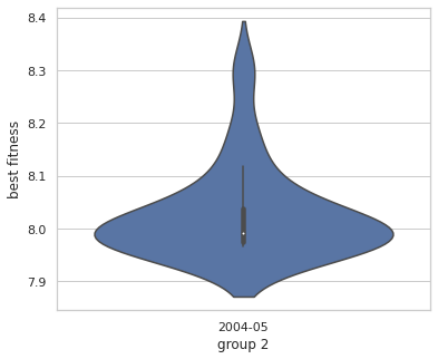

best fitness
11.1
11.0
10.9
10.8
10.7
1979-80
group 2

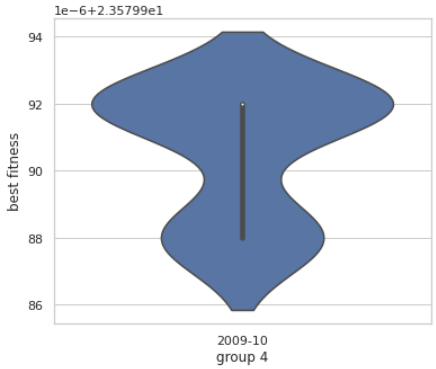

1e−5+1.6749e1
best fitness
25
24
23
22
21
20
19
1986-87
group 5

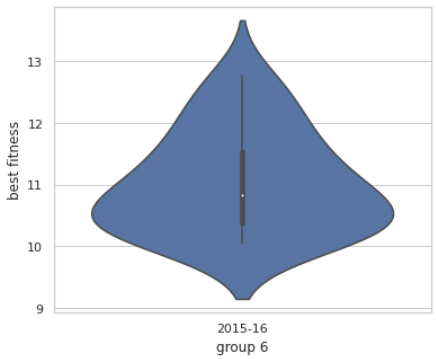

1e−6+1.591786e1
best fitness
10
9
8
7
6
5
1991-92
group 7

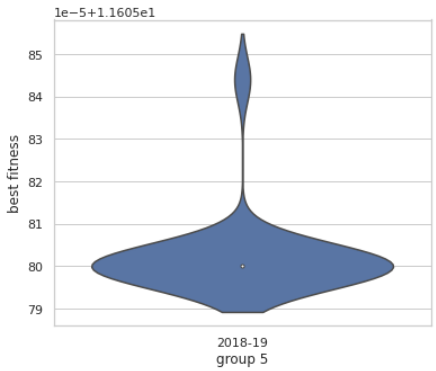

1e−6+1.50918e1
best fitness
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
1997-98
group 6


1e−6+6.6383
best fitness
40.5
40.0
39.5
39.0
38.5
2002-03
group 4


best fitness
8.4
8.3
8.2
8.1
8.0
7.9
2004-05
group 2


1e−6+2.35799e1
best fitness
94
92
90
88
86
2009-10
group 4


best fitness
13
12
11
10
9
2015-16
group 6


1e−5+1.1605e1
best fitness
85
84
83
82
81
80
79
2018-19
group 5

C Annex C

```
# LIBRARIES
import numpy as np
import pandas as pd
import time
# for ploting
import matplotlib.pyplot as plt
import seaborn as sns
# importing the required ML libraries
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

# LOADING DATA and PREPROCESSING
fname = 'oni.csv'
df = pd.read_csv(fname)
X = df.values.reshape(-1,1)
# FOR DATASET 1
def target_data_func(X):
  y = []
  for x in X:
    # EN
```

```
    if x>=0.5:
      y.append(1)
    # AE
    else : y.append(0)
  Y = np.array(y)
  return Y
# FOR DATASET 2
def target_data_func(X):
  y = []
  for x in X:
    # EN
    if x>=0.5:
      y.append(1)
    # LN
    elif x<=-0.5:
      y.append(2)
    # RY
    else : y.append(0)
  Y = np.array(y)
  return Y
# FOR DATSET 3
def target_data_func(X):
  y = []
  for x in X:
    # SLN
    if x<=-1:
      y.append(0)
    # WLN
    elif x<=-0.5 and x>-1:
      y.append(1)
```

```
    # WEN
    elif x>=0.5 and x<1:
      y.append(3)
    # SEN
    elif x>=1:
      y.append(4)
    # RY
    else : y.append(2)
  Y = np.array(y)
  return Y
# FOR DATASET 4
def target_data_func(X):
  y = []
  for x in X:
    # WE
    if x>=0.5 and x<1:
      y.append(1)
    # ME
    elif x>=1 and x<1.5:
      y.append(2)
    # SE
    elif x>=1.5 and x<2:
      y.append(3)
    # VSE
    elif x>=2:
      y.append(4)
    # Otherwise
    else : y.append(0)
  Y = np.array(y)
  return Y
```

```
Y = target_data_func(X)
# CREATING TRAIN and TEST SETS
X_train, X_test, Y_train, Y_test =
                train_test_split(X, Y, test_size = 0.4, random_state = 25)
# The variable random_state is used to set the seed
      # for the Random number generation (so the experience is reproducible).

# MACHINE LEARNING MODELS
# Random Forest Classifier
start = time.time()
rf_clf = RandomForestClassifier()
rf_clf.fit(X_train, Y_train)
rf_prediction = rf_clf.predict(X_test)
scores = cross_val_score(rf_clf, X, Y)
end = time.time()
rf_acc = accuracy_score(rf_prediction,Y_test)
rf_mean = scores.mean()
rf_std = scores.std()
print("Random Forest Classifier Accuracy: {0:.2%}".format(rf_acc))
print("K-Fold Cross Validation score: Mean : {0:.2}, Std : {1:.2}"
                                          .format(rf_mean, rf_std))
print("Execution time: {0:.5} seconds \n".format(end-start))
confusion_mtx  = confusion_matrix(Y_test, rf_prediction)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx,annot=True, fmt='g')
plt.xlabel('Prediction')
plt.ylabel('Label')
plt.show()
# Logistic Regression
start = time.time()
```

```
logreg_clf = LogisticRegression()
logreg_clf.fit(X_train, Y_train)
logreg_prediction = logreg_clf.predict(X_test)
scores = cross_val_score(logreg_clf, X, Y)
end = time.time()
logreg_acc = accuracy_score(logreg_prediction,Y_test)
logreg_mean = scores.mean()
logreg_std = scores.std()
print("Logistic Regression Classifier Accuracy: {0:.2%}".format(logreg_acc))
print("K-Fold Cross Validation score: Mean : {0:.2}, Std : {1:.2}"
                                                .format(logreg_mean, logreg_std))
print("Execution time: {0:.5} seconds \n".format(end-start))
confusion_mtx  = confusion_matrix(Y_test, logreg_prediction)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx,annot=True, fmt='g')
plt.xlabel('Prediction')
plt.ylabel('Label')
plt.show()
# Naive Bayes (Gaussian)
start = time.time()
nb_clf = GaussianNB()
nb_clf.fit(X_train, Y_train)
nb_prediction = nb_clf.predict(X_test)
scores = cross_val_score(nb_clf, X, Y)
end = time.time()
nb_acc = accuracy_score(nb_prediction,Y_test)
nb_mean = scores.mean()
nb_std = scores.std()
print("Gaussian Naive Bayes Classifier Accuracy: {0:.2%}".format(nb_acc))
print("K-Fold Cross Validation score: Mean : {0:.2}, Std : {1:.2}"
```

```
                                                .format(nb_mean, nb_std))
print("Execution time: {0:.5} seconds \n".format(end-start))
confusion_mtx  = confusion_matrix(Y_test, nb_prediction)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx,annot=True, fmt='g')
plt.xlabel('Prediction')
plt.ylabel('Label')
plt.show()
# Decision Tree
start = time.time()
dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train, Y_train)
dt_prediction = dt_clf.predict(X_test)
scores = cross_val_score(dt_clf, X, Y)
end = time.time()
dt_acc = accuracy_score(dt_prediction,Y_test)
dt_mean = scores.mean()
dt_std = scores.std()
print("Decision Tree Classifier Accuracy: {0:.2%}".format(dt_acc))
print("K-Fold Cross Validation score: Mean : {0:.2}, Std : {1:.2}"
                                                .format(dt_mean, dt_std))
print("Execution time: {0:.5} seconds \n".format(end-start))
confusion_mtx  = confusion_matrix(Y_test, dt_prediction)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx,annot=True, fmt='g')
plt.xlabel('Prediction')
plt.ylabel('Label')
plt.show()
# Support Vector Machine
start = time.time()
```

```
svc_clf = SVC()
svc_clf.fit(X_train, Y_train)
svc_prediction = svc_clf.predict(X_test)
scores = cross_val_score(svc_clf, X, Y)
end = time.time()
svc_acc = accuracy_score(svc_prediction,Y_test)
svc_mean = scores.mean()
svc_std = scores.std()
print("Support Vector Machine Classifier Accuracy: {0:.2%}".format(svc_acc))
print("K-Fold Cross Validation score: Mean : {0:.2}, Std : {1:.2}"
                                              .format(svc_mean, svc_std))
print("Execution time: {0:.5} seconds \n".format(end-start))
confusion_mtx  = confusion_matrix(Y_test, svc_prediction)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx,annot=True, fmt='g')
plt.xlabel('Prediction')
plt.ylabel('Label')
plt.show()
# K Nearest Neighbor
start = time.time()
knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, Y_train)
knn_prediction = knn_clf.predict(X_test)
scores = cross_val_score(knn_clf, X, Y)
end = time.time()
knn_acc = accuracy_score(knn_prediction,Y_test)
knn_mean = scores.mean()
knn_std = scores.std()
print("K Nearest Neighbor Classifier Accuracy: {0:.2%}".format(knn_acc))
print("K-Fold Cross Validation score: Mean : {0:.2}, Std : {1:.2}"
```

```
                                                .format(knn_mean, knn_std))
print("Execution time: {0:.5} seconds \n".format(end-start))
confusion_mtx  = confusion_matrix(Y_test, knn_prediction)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx,annot=True, fmt='g')
plt.xlabel('Prediction')
plt.ylabel('Label')
plt.show()
```

D Annex D

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans, AgglomerativeClustering
# KMeans
kmeans = KMeans(n_clusters=7, random_state=42)
kmeans_pred = kmeans.fit_predict(X)
kmeans_clusters = np.unique(kmeans_pred)
for cluster in kmeans_clusters:
  row_ix = np.where(kmeans_pred==cluster)
  print(cluster, X.iloc[row_ix])
# Agglomerative
agg_model = AgglomerativeClustering(n_clusters=7)
agg_pred = agg_model.fit_predict(X)
agg_clusters = np.unique(agg_pred)
for cluster in agg_clusters:
  row_ix = np.where(agg_pred==cluster)
  print(cluster, X.iloc[row_ix])
```

E Annex E

```
\User declarations :
// these 3 defines are mandatory here. Adjust as you like.
#define NO_FITNESS_CASES 18
#define VAR_LEN 1                // number of input variables
#define GROW_FULL_RATIO 0.5      // cf. Koza's book on GP
#define NUMTHREAD 1024 // nb of threads in parallel on GPU cards.
// 1024 is the maximum on Fermi architectures. It is 512 on older cards
#define MAX_STACK 15
#define PI (3.141592653589793)

float x[NO_FITNESS_CASES] = {0.0833, 0.1667, 0.25, 0.3333, 0.4167, 0.5,
0.5833, 0.6667, 0.75, 0.8333, 0.9167, 1, 1.0833, 1.1667, 1.25, 1.3333,
1.4167, 1.5};
float y[NO_FITNESS_CASES] = {-0.4, -0.4, -0.33, -0.13, 0.23, 0.53, 0.83,
1.13, 1.43, 1.73, 1.95, 2.18, 2.28, 2.2, 1.88, 1.4, 0.9, 0.45};
\end

\User functions:
int initData(float*** inputs, float** outputs) {
  int i=0;
  (*inputs) = new float*[NO_FITNESS_CASES];
  (*outputs) = new float[NO_FITNESS_CASES];
  for( i=0 ; i<NO_FITNESS_CASES ; i++ ){
    (*inputs)[i]=new float[VAR_LEN];
    (*inputs)[i][0] = x[i];
    (*outputs)[i] = y[i];
  }
  return NO_FITNESS_CASES;
```

```
}
void free_data(){
  for( int i=0 ; i<NO_FITNESS_CASES ;i++ ) delete[] inputs[i] ;
  delete[] outputs;
  delete[] inputs;
}
\end
\Before everything else function:
  initData(&inputs,&outputs); // inputs, outputs requis par EASEA
\end
\After everything else function:
  std::cout << "y=" << toString(((IndividualImpl*)EA->
  population->Best)->root) << std::endl;
  free_data();
\end

\User classes :
GenomeClass {
  GPNode* root; // GPNode is a reserved name that must be used for GP
}
\end
\GenomeClass::initialiser :
  Genome.root = ramped_hh(); // Initializer, cf. Koza GP
\end
\GenomeClass::crossover :
  simpleCrossOver(parent1,parent2,child); // cf. Koza GP
  child.valid = false;        // to force the evaluation of the child
\end
\GenomeClass::mutator :
  simple_mutator(&Genome); // cf. Koza GP
```

```
\end
\begin operator description :
OP_X, "x", 0, {RESULT=INPUT[0];};
OP_ERC, "ERC", 0, {RESULT=ERC;};
OP_ADD, "+", 2, {RESULT=OP1+OP2;};
OP_SUB, "-", 2, {RESULT=OP1-OP2;};
OP_MUL, "*", 2, {
    RESULT=OP1*OP2;
};
OP_SIN, "sin", 1, {
  RESULT=sin(OP1);
};
\GenomeClass::evaluator for each fc :
float expected_value = OUTPUT;
ERROR = fabs(expected_value-EVOLVED_VALUE);
\end
\GenomeClass::evaluator accumulator :
return ERROR;
\end

\User Makefile options:
CXXFLAGS+=-I/usr/local/cuda/common/inc/ -I/usr/local/cuda/include/
LDFLAGS+=
\end

\Default run parameters :
  Number of generations : 500
  Time limit: 0
  Population size : 20000
  Offspring size : 100%
```

```
  Mutation probability : 0.4
  Crossover probability : 1.0
  Evaluator goal : minimise
  Selection operator: Tournament 7
  Surviving parents: 100%
  Surviving offspring: 100%
  Reduce parents operator: Tournament 2
  Reduce offspring operator: Tournament 2
  Final reduce operator: Tournament 7

  Elitism: Strong
  Elite: 1
  Print stats: true
  Generate csv stats file:false
  Generate gnuplot script:false
  Generate R script:false
  Plot stats:true

  Remote island model: true
  IP file: ip.txt
  Server port : 2929
  Migration probability: 0.33
  Save population: false
  Start from file: false
  // nouveaux paramètres
  max init tree depth : 4
  min init tree depth : 2
  max tree depth : 6
  size of prog buffer : 200000000
\end
```