

Capilla, Javier; Alcaráz, Alba; Valarezo, Angel; Garcia-Hiernaux, Alfredo; Pérez-Amaral, Teodosio

Conference Paper

Eco-RETINA: a green flexible algorithm for model building

ITS 33rd European Conference 2025: "Digital innovation and transformation in uncertain times", Edinburgh, UK, 29th June – 1st July 2025

Provided in Cooperation with:

International Telecommunications Society (ITS)

Suggested Citation: Capilla, Javier; Alcaráz, Alba; Valarezo, Angel; Garcia-Hiernaux, Alfredo; Pérez-Amaral, Teodosio (2025) : Eco-RETINA: a green flexible algorithm for model building, ITS 33rd European Conference 2025: "Digital innovation and transformation in uncertain times", Edinburgh, UK, 29th June – 1st July 2025, International Telecommunications Society (ITS), Calgary

This Version is available at:

<https://hdl.handle.net/10419/331255>

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.



ICAE

Instituto Complutense
de
Análisis Económico

Eco-RETINA: a green flexible algorithm for model building

Javier Capilla

Universidad Complutense de Madrid

Alba Alcaraz

Universidad Complutense de Madrid

Ángel Valarezo

Universidad Complutense de Madrid, ICAE

Alfredo García-Hiernaux

Universidad Complutense de Madrid, ICAE

Teodosio Pérez Amaral

Universidad Complutense de Madrid, ICAE

Abstract

Eco-RETINA is an innovative and eco-friendly algorithm explicitly designed for out-of-sample prediction. Functioning as a regression-based flexible approximator, it is linear in parameters but nonlinear in inputs, employing a selective model search to optimize performance. The algorithm adeptly manages multicollinearity, emphasizing speed, accuracy, and environmental sustainability. Its modular and transparent structure facilitates easy interpretation and modification, making it an invaluable tool for researchers in developing explicit models for out-of-sample forecasting. The algorithm generates outputs such as a list of relevant transformed inputs, coefficients, standard deviations, and confidence intervals, enhancing its interpretability.

JEL: C14, C45, C51, C63

ICAE Working Paper # 0125

February 3, 2025

ISSN: 2341-2356

WEB DE LA COLECCIÓN:

<http://www.ucm.es/fundamentos-analisis-economico2/documentos-de-trabajo-del-icae>

Copyright © 2013, 2023 by ICAE.

Working papers are in draft form and are distributed for discussion. It may not be reproduced without permission of the author/s.

Eco-RETINA: a green flexible algorithm for model building.

Abstract continued:

Now implemented in Python and available on GitHub, Eco-RETINA includes several new features, such as the ability to measure CO2 emissions and energy consumption. These enhancements, along with improved data transformations, bottleneck removal, and a user-friendly interface, have significantly boosted its performance. The algorithm's efficiency in reducing carbon footprint and power consumption achieving significant reductions of computational time, underscores its sustainability and effectiveness. Empirical results suggest that Eco-RETINA is not only a sustainable alternative to conventional neural networks but also surpasses them in certain aspects, offering a competitive edge in process traceability, accuracy, interpretability, and environmental impact.

Introduction.

The power consumption and carbon footprint of AI-related algorithms is a matter of preoccupation for researchers, industry, and policymakers. See Strubell et al. (2019), Schwartz et al. (2020), Kaak et al. (2021), OECD (2022), Verdecchia et al. (2023), and Bouza et al. (2023), among others. The issues are so relevant that some researchers predict a complete depletion of available energy resources by 2030. Large technological firms are devoting significant resources to tackling these problems, and the European Commission has launched several projects related to this issue, such as Enfield¹.

The energy cost of developing, training, and predicting deep learning models has significantly increased in the last few years. As neural network models become larger and more complex to undertake increasingly sophisticated tasks, the demand for computational resources, particularly graphic processing units (GPU) and tensor processing units (TPU), has increased rapidly. This increase in the demand for computational resources has led to a corresponding increase in energy consumption, which leads to preoccupations about AI's sustainability and environmental impact. As a result, it has become crucial for the research and industry communities to explore the development of new algorithms and techniques for training neural network models that are more efficient. Several of these modifications require less performance compared with the original algorithms.

¹ The authors acknowledge the support of the HORIZON Research and Innovation Program of the European Union, under grant agreement No 101120657, project ENFIELD (European Lighthouse to Manifest Trustworthy and Green AI).

This paper introduces Eco-Retina, a new green algorithm designed for out-of-sample prediction. A preliminary release is in Capilla (2024a). The new algorithm is based on Pérez-Amaral et al. (2003, 2004) and Marinucci (2005, 2007).

Eco-RETINA is green by design and incorporates substantial advancements over previous approaches. It functions as a regression-based flexible approximator, offering a linear relationship in its parameters to enhance convergence while enabling nonlinearity in the inputs to provide greater flexibility for approximation purposes.

Eco-RETINA employs a selective search instead of an exhaustive one, allowing faster execution without compromising accuracy. It adeptly handles multicollinearity using a multicollinearity threshold and detects and deals with outliers. It emphasizes speed, accuracy, and environmental sustainability, making it a powerful and approachable tool.

The algorithm is modular, allowing parts to be replaced with other modules that may be better suited to the problem. It is transparent, as its functioning is open and modifiable; it does not operate like a black box.

Eco-RETINA's output is like that of many econometric packages. It includes an explicit forecasting model, point estimates, parameter standard deviations, and confidence intervals. The forecasting model is amenable to interpretation, providing researchers with an explicit model for out-of-sample forecasting that can be understood and modified.

Eco-RETINA is now written in Python and available on GitHub². It incorporates new functionalities, such as measuring CO2 emissions and energy spent. The algorithm allows data transformations, such as ratios, cross-products, inverses, logarithms, and other functions of the original inputs.

Bottleneck removal has been crucial when programming the algorithm, as it circumvents situations in which repetitive operations were performed at a high cost of time and energy. A user interface was developed to enhance user-friendliness.

The rest of the paper is organized as follows: Section 2 shows selective summaries of recent documents on green AI and RETINA. Section 3 describes the RETINA algorithm. Section 4 shows Eco-RETINA. Section 5 discusses hyperparameters and architecture, while section 6 summarizes the experiments. The conclusions are included in Section 7.

1. Related work.

This literature review is selective. It includes two parts: the first on green AI and the second on RETINA.

² <https://github.com/jcapilla780?tab=repositories>

1.1 Green AI.

Emma Strubell, Ananya Ganesh, and Andrew McCallum (2019) analyze the energy and policy implications of deep learning in Natural Language Processing (NLP). They deal with large models that require considerable computational resources and energy consumption.

These models are costly to train and develop, financially and environmentally, due to the carbon footprint required to fuel processing hardware. The paper quantifies different costs and consumptions, relating them to magnitudes familiar to consumers.

They propose an efficiency measure defined as accuracy per total dollar cost, which includes fixed and variable expenses (primarily the total cost). This encompasses the full cost of compute infrastructure, with a particular focus on cooling requirements.

The authors quantify the approximate financial and environmental costs of training various successful neural network models for NLP. Based on these findings, they propose that the researchers report the training time and computational resources required, including human time and specialization needed, and the sensitivity to hyperparameters.

They also recommend that academic researchers should have equitable access to computation resources and that researchers should prioritize computationally efficient hardware and algorithms.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni (2020) note that since 2012, AI has made significant advancements across various tasks, driven by increasingly large and computationally intensive deep learning models. Training costs for state-of-the-art models, such as BERT and GPT-3 (relative to their development at the time), have risen exponentially, with a 300,000x increase since 2012. This trend, known as “Red AI,” has been criticized for its high environmental and financial costs, creating barriers for participation in AI research. To address this, researchers suggest improving efficiency by reporting computational work for hyperparameter tuning, using more efficient software, and minimizing energy consumption and carbon emissions.

Kaack et al. (2021) examine the effects of AI and machine learning (ML) on climate change, calling for collaboration across research, policy, and industry. They highlight uncertainties in how ML impacts greenhouse gas (GHG) emissions due to a lack of clear mechanisms for measurement. The authors propose a systematic framework with three categories: (A) compute-related impacts, (B) immediate impacts of ML applications, and (C) system-level impacts. Using this framework, they assess GHG emissions from developing large models, which can be substantial, and emphasize the need for policy, innovation, and better impact assessments to mitigate ML’s environmental impact.

The OECD (2022) analyzes the costs and benefits of AI deployment, focusing on the “twin transformations” of green and digital development. It emphasizes the need for standards to measure and report AI’s environmental impact, including GHG emissions, energy, water, and natural resource usage. The report highlights that many studies overlook impacts on

developing economies and call for broader data collection on AI's environmental effects. Policy recommendations include developing measurement standards for sustainable AI, expanding environmental data collection, and improving transparency and equity, particularly in less developed countries.

Verdecchia, Sallou, and Cruz (2023) systematically review Green AI literature, analyzing 98 primary studies. Green AI research has grown significantly since 2020, focusing mainly on monitoring model footprints, tuning hyperparameters, and benchmarking models. Most studies are academic, with laboratory experiments being the dominant research strategy, while industrial involvement remains limited. Approximately half of the studies focus on the training phase, and more than half report at least 50% energy savings. The review concludes that Green AI has matured, suggesting a shift toward broader research strategies and transferring academic findings to industrial applications.

Bouza, Lannelongue, and Dahdaleh (2023) review several online and software tools designed to track energy consumption during machine learning model training. The paper provides an introduction to these tools, compares their features, and discusses technical requirements. They evaluate the energy consumption of two deep neural networks across different server types, comparing seven tools, with a recommendation for using the Green Algorithms tool. The study measures the consumption of various components (CPU, GPU, memory, etc.) and finds that eco-friendly measurement minimally impacts overall consumption. Their findings offer practical advice on selecting appropriate tools and infrastructure.

1.2 RETINA.

Pérez-Amaral, Gallo, and White (2003) propose a new method called the Relevant Transformation of the Inputs Network Approach, RETINA, which serves as a tool for model building. This method is designed with flexibility in mind, allowing for nonlinear transformations of the predictors of interest. It incorporates selective searches within the range of possible models, provides out-of-sample forecasting capabilities, and maintains computational simplicity. In tests conducted on simulated data, the method demonstrates a high success rate in retrieving the data-generating process, which improves as the sample size increases and performs well compared to other alternative procedures.

Pérez-Amaral, Gallo, and White (2005) compare the characteristics of RETINA with PcGets, a well-known automatic modelling method proposed by David Hendry. They point out similarities, differences, and complementarities between the two methods. Using US telecommunications demand data, they find that RETINA can improve both in- and out-of-sample over the usual linear regression model and over some models suggested by PcGets. Thus, both methods are useful components of the modern applied econometrician's automated modelling tool chest.

Marinucci (2005, 2007) improves on the programming of RETINA by using Matlab, adding a number of new features, such as dealing with dummy variables, a user-friendly interface, and allowing variables not to be transformed by the procedure. He also deals with applications using real data.

2. The algorithm.

Let us first deal with the term “Green AI”. Schwartz et al. (2020) state that the concept of Green AI refers to research in artificial intelligence (AI) that produces new outcomes without increasing computational costs and ideally reduces them. Barbierato and Gatti (2024) explain that Green AI achieves its objectives through the use of smaller datasets, less computationally intensive model training techniques, or renewable energy resources. They further emphasize that the Green AI approach prioritizes efficiency and sustainability over precision and performance. They highlight the importance of efficiency metrics, in addition to predictive metrics, for evaluating models, enabling researchers to focus on the efficiency of their models and their positive environmental impact.

The concept that opposes Green AI is known as Red AI. Schwartz et al. (2020) explain that this term refers to AI research aimed at achieving novel results in terms of precision or similar metrics through the use of massive computational power. Although significant achievements have been made through Red AI, this approach has substantial limitations. Generally, the relationship between a model’s performance and its complexity (measured as the number of parameters or training time) is, at best, logarithmic, meaning that a linear gain in performance necessitates an exponentially larger model. Similar results occur with increases in the size of the training dataset and the number of experiments, where diminishing returns on performance require increasing computational costs.

A point that is not always clearly articulated is the distinction between Green AI and artificial intelligence applied to environmental problems. As noted by Barbierato and Gatti (2024), while both share the goal of addressing environmental challenges and issues, their approaches, implementations, and outcomes differ. Green AI focuses on achieving a balance between performance and efficiency, aiming to minimize computational costs and resource usage. The Green AI framework is a broad concept that encompasses numerous research and development strategies and is not confined to specific industries or applications.

3.1 RETINA.

This section describes the original RETINA algorithm and its relationship with the Green AI approach. RETINA (Relevant Transformation of the Inputs Network Approach) is a methodology for model construction that focuses on flexibility (enabling nonlinear transformations of relevant predictors), selective model search, out-of-sample predictive capability, and computational simplicity. RETINA integrates the flexibility of neural network

models by accommodating nonlinearity and interaction effects (through nonlinear transformations of potentially useful variables in the conditioning set). It addresses the concavity issues in the likelihood function of weights in standard linear models, thereby avoiding the numerical complexity of estimation. It also identifies a subset of genuinely valuable features for predicting outcomes, adhering to the principle of parsimony. RETINA also employs a cross-validation estimation scheme during the model selection process to minimize the likelihood of performance being attributed to chance.

According to Pérez-Amaral et al. (2003), the RETINA algorithm proceeds as follows:

Stage 0 – Preliminar

1. Data Construction and Organization
 - a) Generate the set of transformed variables $\zeta(X) = \{W_1, \dots, W_m\}$.
 - b) Divide the sample into three random subsamples.

Stage I – Isolation of a ‘Candidate’ Model

1. Using data from the first subsample: a) Rank the variables in $\zeta(X)$ according to their sample correlation (absolute value) with the dependent variable, only within the first subsample, such that $W(1)$ is the variable with the highest absolute correlation with Y , $W(2)$ is the second most correlated, and so on.
2. Consider several sets of regressors that include a constant and $W(1)$. Each set of regressors $\zeta_\lambda(X)$ is indexed by a ‘collinearity threshold’ $0 \leq \lambda \leq 1$ and is constructed by including $W(j)$ ($j = 2, \dots, m$) in $\zeta_\lambda(X)$ if the R^2 of the regression of $W(j)$ on the variables already included is $\leq \lambda$. The number of regressor sets is controlled by the number of values of λ chosen between 0 and 1, say v .
3. Using data from both the first and second subsamples: a) Estimate each model by regressing Y on each set of regressors $\zeta_\lambda(X)$ using data only from the first subsample and compute an out-of-sample prediction criterion (mean squared error) using data only from the second subsample. This requires the estimation of v models.
b) Select a ‘candidate’ model corresponding to the one with the best out-of-sample performance, denoted $\zeta_{*\lambda}(X)$.

Stage II – Search Strategy

1. Using data from both the second and third subsamples: a) Search for a more parsimonious model: estimate all models that include a constant and all the regressors in $\zeta_\lambda(X)$, adding one at a time, in the order they were originally included, but also in the order produced by step 2a, this time based on the correlations in the second subsample. b) Perform an out-of-sample evaluation of the models (using data from the third subsample) by calculating a performance measure (mean squared error, possibly augmented by a penalty term for the number of parameters in the model).

Stage III – Model Selection

2. Repeat stages I and II, changing the order of the subsamples, and produce a candidate for each ordering of subsamples.
3. Select the model with the best performance across the entire sample.

3.2 RETINA and Green AI.

Although the Green AI approach is relatively recent and postdates RETINA, the RETINA algorithm inherently reflects the Green AI philosophy. RETINA employs a selective search rather than an exhaustive search. For a given number of m candidate regressors, a comprehensive search would require evaluating $2^m - 1$ models, which can become computationally expensive when m is relatively large. In contrast, RETINA evaluates only a subset of models, of the order of $2m$, a task that demands significantly less execution time and computational resources. By assessing a much smaller number of models, RETINA can identify a model with performance comparable to that achieved by evaluating all possible models.

Both selective search and Green AI aim to optimize resource usage and enhance efficiency, thereby reducing energy consumption and carbon footprint. This positions selective search and Green AI as complementary strategies in the broader context of sustainable AI development.

4. Eco-RETINA.

A later version of RETINA, by Marinucci (2007), featured new valuable functionalities, such as eliminating outliers and extending the original algorithm by including the ability to handle dummy variables. He also developed a graphical user interface for RETINA and implemented a new version in MATLAB. These extensions have been incorporated in the development of the estimator and interface of Eco-RETINA, and additional new functionalities have been added. Below is a list of all the new features included in Eco-RETINA:

- **Measure energy consumption and CO₂ emissions** associated with model training and forecasting. This functionality is relevant for this version's name. Measurements are performed using the Python package Codecarbon, whose information can be found in Schmidt et al. (2021). The availability of information about the energy consumption of different models' training facilitates the application of the Green AI approach.
- **Elimination of bottlenecks in step 2b.** If we review the description of step 2b of RETINA and consider its implementation, it becomes clear that this step can be computationally expensive. When programming it, selecting one parameter to determine the maximum allowed collinearity and another to form a grid is logical. Each element of the grid, or λ (following the notation of Perez-Amaral et al., 2003),

is used to indicate different levels of collinearity and index each set of regressors. This is problematic because not only do multiple regressions need to be performed for each grid value, but if a very fine grid is chosen, the process may need to be repeated a very large number of times. For example, with a maximum multicollinearity level of 0.5 and a grid with a separation of 0.001, the step would need to be repeated for 500 different λ values. Furthermore, it is generally unnecessary to perform this step so many times because after obtaining the first set of regressors for the first grid value, it often happens that the same set of regressors is obtained for the subsequent grid values, with only a few being added for higher λ values. Taking all this into account, step 2b has been modified as follows. First, to avoid numerous regressions for each lambda value, vectorized operations are used so that only one regression is needed for each grid value.³ Second, instead of performing a regression for each grid value, Eco-RETINA performs the step only for the first value of the original grid and then adapts the grid to be coarser, limiting the number of regressions to 50. This allows the first set of regressors to be obtained with a very fine grid, but prevents wasting time on hundreds of regressions that do not expand the set of regressors.

- **Elimination of bottlenecks in step 4b.** If the candidate model obtained in step 3b is very large, step 4b can consume a lot of compute time due to the need to perform a large number of regressions. If a candidate model has n regressors, step 4b requires performing 2^n regressions. This is problematic because, in some cases, the candidate model may have hundreds of regressors. Eco-RETINA solves this by limiting the number of regressions to 2^k , where k is the maximum between 50 and the number of original regressors. Thus, the number of regressions no longer depends on the size of the candidate model. Instead of considering all submodels of the candidate model, Eco-RETINA only considers the 10 simplest submodels, the 10 most complex submodels, and $k - 20$ submodels where the number of regressors varies uniformly.
- **Cross-product transformations for non-continuous variables.** This feature is useful in cases where some continuous variables contain many zeros. In these cases, quadratic transformations are not particularly helpful, and inverse transformations can be counterproductive because many rows in the data table would contain null values and would be ignored in a linear regression.
- **Use of dummy variables.** This feature was already included in the 2004 version, but in the set of transformed variables, both the transformed continuous variables and their products with the dummy variables were included. Thus, in the case of k continuous variables and d dummy variables, the set of transformed variables would consist of $d \times (2k^2 + 2k + 1)$ regressors. This is a very large number of variables, and in general, the products of dummy variables with transformations of continuous

³ Numpy Python library is used in vectorized operations.

variables are not widely used in econometrics because they introduce excessive complexity into the models and promote overfitting. Eco-RETINA corrects this by only considering the products of dummy variables with the original continuous variables. In this way, with k continuous variables and d dummy variables, the set of regressors consists of $2k^2 + (2 + d)k + d + 1$ variables.

- **Automatic search for the collinearity threshold based on the final model's maximum size.** When selecting a candidate model, Pérez-Amaral et al. (2003) mention the existence of a collinearity threshold $0 \leq \lambda \leq 1$ that indexes different sets of regressors, but they do not explain how to determine this threshold. Eco-RETINA addresses this issue by proposing a threshold based on the number of variables allowed in the final model. The value of this feature lies not only in resolving an open question from the original article but also in controlling the size of the models proposed by RETINA, ensuring they do not exceed a certain amount. This is crucial because RETINA is designed to propose parsimonious and relatively small models.
- **Outlier removal.** This feature is very similar to the one included in the 2007 version. However, Eco-RETINA allows for the selection of the interquartile range multiplier used to eliminate outliers.
- **Multiple polynomial transformations.** In RETINA, the set $\zeta(X)$ consisted of transformed variables W_{ij} , each obtained as $X_{il}^a \times X_{il}^b$, where $a, b = -1, 0, 1$; $j, l = 1, \dots, k^2$. Eco-RETINA goes a step further and allows a and b to take any value, including fractional values. Despite this feature's possibilities, choosing a large number of transformations is not recommended. The gains from expanding the transformations are limited, and adding even a few transformations significantly increases Eco-RETINA's computational cost.
- **More model selection metrics.** The original version of RETINA used the mean squared error as the model performance measure. Eco-RETINA adds the option to use the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC).
- **New types of regression.** RETINA could only be used in cases where the target variable was continuous. Eco-RETINA expands this capability, allowing it to be used for binary classification problems, incorporating models such as Logit, Probit, and the Linear Probability Model (LPM).
- **Option to use heteroscedasticity-consistent covariance.** Users can choose between different versions of heteroskedasticity-consistent covariance matrices.
- **Logarithmic transformations.** Eco-RETINA incorporates regression of the types $Y \sim \log X$, $\log Y \sim X$, and $\log Y \sim \log X$.
- **Parallelization of RETINA's six iterations.** Eco-RETINA can execute RETINA's six iterations in parallel, significantly reducing execution time.
- **Validation of results.** Eco-RETINA offers the option to divide the sample into four subsamples instead of three, using the fourth subsample as a validation set. This option is especially useful when evaluating a model without a separate test dataset.

or when determining which model hyperparameters to select. For example, one might be unsure whether to use RETINA’s standard transformations or add a cubic transformation. Both options can be evaluated on the validation set, and the option with the lowest prediction error can be selected. This way, since the test data do not participate in hyperparameter selection, the process remains unbiased.

- **Standardization.** Eco-RETINA allows for the standardization of continuous variables, which reduces the effect of scale differences between variables.

5. Hyperparameters and Architecture.

The algorithm is not tractable analytically, so we will perform experiments to learn about its behavior using real data. In our experiments, we will use data on diamond prices from Hrokr, (2023). These experiments aim to find a model which fulfills several requirements. No hyperparameter tuning will be performed, as the goal is to get an idea of Eco-RETINA’s predictive capacity and how “green” it is. For this, we examine its Mean Absolute Percentage Error (MAPE) and CO2 emissions during training and compare them with those of RETINA and a neural network.

For both RETINA and Eco-RETINA, the same hyperparameters will be used, which will be the most basic ones. No polynomial or logarithmic transformations will be applied, the data will not be standardized, nor will outliers be removed. The model size will not be restricted. The maximum possible collinearity will be allowed (1), and a grid with a separation of 0.005, which is the default for Eco-RETINA, will be used. Regarding the neural network, programmed with the TensorFlow2 library, a multilayer perceptron with a standard architecture will be used. We apply the general principles of Ranjan (2020) to design it. It will have two hidden layers, and each layer will contain a number of nodes following a geometric series of 2.

The first layer will have a number of nodes close to half the number of regressors, and the next layer will have half as many nodes as the first. Thus, for 10 variables, the first layer will have 4 nodes, and the second will have 2. For 21 variables, the first layer will have 8 nodes, and the second 4. In experiments with 31 and 41 variables, the first layer will have 16 nodes, and the second 8. Finally, for 50 regressors, the first layer will have 32 nodes, and the second 16. The ReLU activation function will be used in both hidden layers. In all experiments, the batch size will be 16, the mean squared error will be used as the loss function, and the Adam optimizer will be employed with a learning rate of 0.001. The data will be scaled using MinMax scaling before training the neural network.

Although the architecture to be used in each experiment has already been specified, one crucial hyperparameter remains to be determined for conducting experiments with the multilayer perceptron: the number of epochs for each experiment. This parameter

is critical, as it is directly related to the training time of the neural network and, therefore, to the energy consumption of the training. If a low number of epochs is chosen, energy consumption will be lower, but the network's predictive capacity may not be optimal. On the other hand, if the network is trained for more epochs, as long as overfitting is avoided, the probability of converging to an optimal solution will be higher, but so will the energy consumption during training.

One strategy often used to avoid training a neural network for an excessive number of epochs is to use a validation dataset, and if the loss function on the validation set does not decrease for a certain number of epochs, known as "patience," training is stopped. The only drawback of this method is that, since a validation set must be used, the neural network is trained with less data.

To train our neural network for an appropriate number of epochs without reducing the training set, the following procedure will be followed.

First, the training data will be split into a training set and a validation set, and the neural network will be trained for a certain number of iterations. Training will stop when no improvement in the selected evaluation metric on the validation set is observed for a specific patience. Then, using the number of iterations obtained, the neural network will be retrained, but this time on the entire training set. Only the emissions from this final training will be measured. This procedure combines the advantages of all the mentioned alternatives and also overcomes all their drawbacks.

The only remaining question is which patience value to choose. Usually, a value between 1 and 10 is chosen, although sometimes higher values are used. To avoid selecting a value that is too large or too small, we will use patience values of 5 and 10. If we observe an improvement in the mean squared error of predictions on the validation set when increasing from 5 to 10, we will use the number of epochs determined with a patience of 10. If the mean squared error remains the same, we will choose the number of epochs obtained with a patience of 5. Logically, to follow this procedure, the global seed of the environment must be fixed so that the network weights are initialized in the same way.

The following tables show the number of epochs and the validation loss function value for different numbers of variables and training set sizes. The rows in bold are those that will be used in the experiments.

Table 1: Number of Epochs with 100% Training Data

Number of Variables	Patience	Validation Loss	Number of Epochs
10	5	0.00572904	34
10	10	0.00572904	39

Number of Variables	Patience	Validation Loss	Number of Epochs
21	5	0.004340689	23
21	10	0.00434069	28
31	5	0.001081986	41
31	10	0.001080723	53
41	5	0.000984115	23
41	10	0.00095931	51
50	5	0.000659944	31
50	10	0.000659944	36

Source: Author's own elaboration.

Table 2: Number of Epochs with 10% Training Data

Number of Variables	Patience	Validation Loss	Number of Epochs
10	5	0.005682512	24
10	10	0.005453233	70
21	5	0.004315536	31
21	10	0.004282544	56
31	5	0.001637506	30
31	10	0.001536177	72
41	5	0.001450007	20
41	10	0.001450007	25
50	5	0.001265879	21
50	10	0.001265879	26

Source: Author's own elaboration. Table 3: Number of Epochs with 1% Training Data

Number of Variables	Patience	Validation Loss	Number of Epochs
10	5	0.007167303	39
10	10	0.007122321	58
21	5	0.005078375	36
21	10	0.005078375	41
31	5	0.003554217	29
31	10	0.003524646	44
41	5	0.002637084	37
41	10	0.002637084	42
50	5	0.004101982	21
50	10	0.004099518	32

Source: Authors' elaboration.

Note that both the data and methodology favor the neural network over Eco-RETINA. RETINA is an algorithm optimized for economic data, where there are usually few parameters. However, in our experiments, we will use data on diamond prices with up to 50 variables. Although we did not use a validation set to find the optimal parameters for Eco-RETINA, in the case of the neural network, to find the appropriate number of epochs, we conducted two training sessions: one with a patience of 5 and another with 10. Indeed, we are not tuning hyperparameters that are key to the architecture of a neural network, such as the number of layers or nodes, but for each experiment, we have chosen standard architectures that tend to perform well in most cases.

6. Experiments.

The data used for the experiments comes from Hrokr (2023) and will be uploaded to Capilla (2024b) along with the code used.

6.1 Data.

The dataset used contains sale prices and various characteristics of diamonds. The data was obtained after cleaning the original database, removing outliers, and converting categorical variables into binary form. In addition to the target variable, the diamond sale price, the dataset consists of 50 explanatory variables, of which 6 are continuous and 44 are binary, taking the value 1 if a particular observation meets a characteristic and 0 otherwise. The 6 continuous variables refer to aspects related to the size and weight of the diamonds. As for the binary variables, 2 relate to polishing, 2 to symmetry, 11 to clarity, 10 to color, 10 to cut, 6 to fluorescence, and 3 to the laboratory that evaluated and certified the diamond.

The dataset contains a total of 153,874 observations, divided into a training set of 115,405 (75%) observations and a test set of 38,469 (25%) observations.

6.2 Experimental Methodology

The goal of the experiments is to measure the performance of Eco-RETINA and compare it with that of RETINA and a neural network. The models' performance will be measured in terms of prediction error and the CO2 emissions generated during the models' training. The mean absolute percentage error (MAPE) has been chosen as the measure of prediction error, as the quantities being worked with, in this case, diamond prices can be pretty large, and measures like the mean squared error can be less intuitive.

A comprehensive comparison of the different algorithms will be made. The impact of the number of variables, the number of observations, and the execution environment will be assessed. In the comparison between RETINA and Eco-RETINA, the last aspect (execution

environment) will be omitted, as the most exciting aspect of this comparison lies in whether there is a significant difference between the predictive capabilities of both algorithms. Since Eco-RETINA is an improved version of RETINA, it will always have shorter execution times and produce lower CO2 emissions during training.

The variation in the number of variables will proceed as follows: for all three algorithms, the first set of experiments used the 6 continuous variables, the 2 related to polishing, and the 2 concerning symmetry. Subsequently, the 11 variables related to clarity were added, followed by 10 variables related to color. Next, the 10 regressors related to cut were added, and finally, the 6 variables related to fluorescence and the 3 related to the certifying laboratory were included. This way, for each algorithm, experiments were conducted using 10, 21, 31, 41, and 50 variables.

It was mentioned earlier that the training dataset contains 115,405 observations. This is true for one set of experiments, but not for all. To observe how the size of the training dataset affects the performance of the algorithms, different experiments were conducted by training the models with the original training dataset, with a subsample containing 10% of the training data and with another subsample containing 1% of the observations. Here, we explore Varian’s (2014) suggestion, which may have important implications.

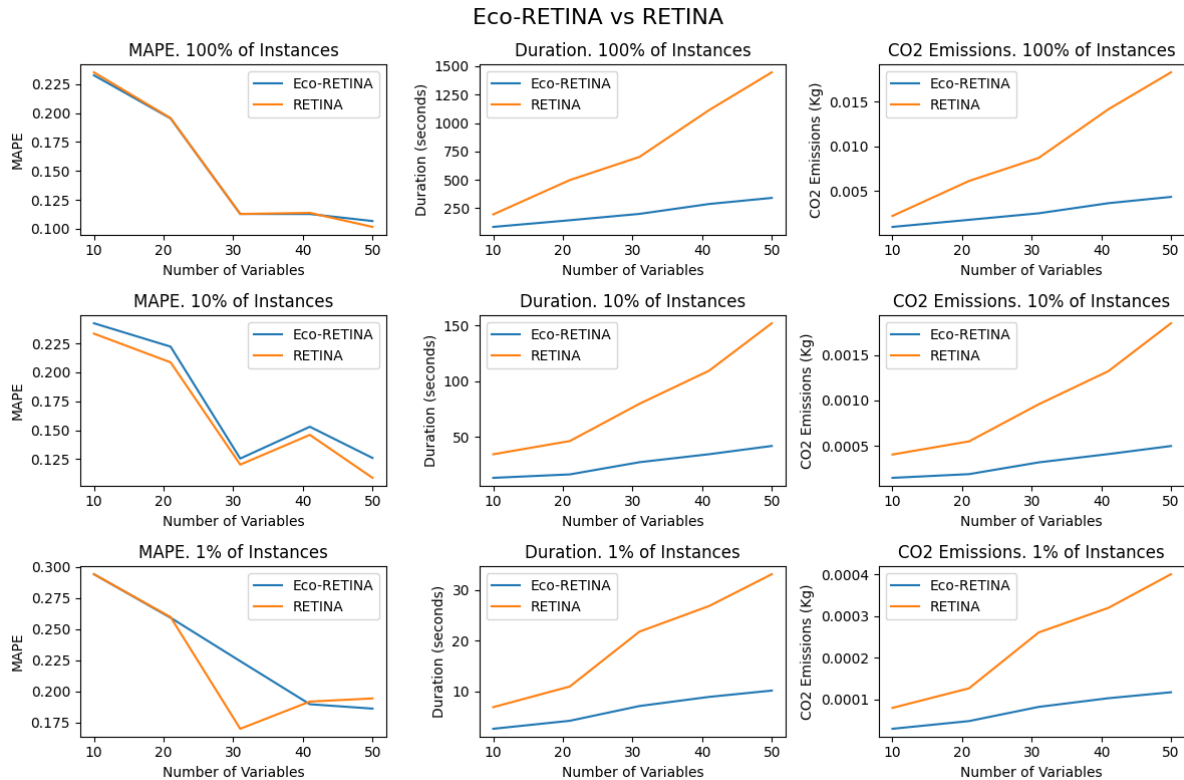
Experiments were also conducted in different execution environments. To do this, the experiments were repeated across several environments available in Google Colab. We used the CPU environment, the V100 (with high RAM capacity), and the A100. The CPU environment has 2 logical processors, an Intel® Xeon® CPU @ 2.20GHz, and 12.67 GB of RAM. The V100 environment has 8 logical processors, an Intel® Xeon® CPU @ 2.00GHz, 50.99 GB of RAM, and a Tesla V100-SXM2-16GB GPU. Finally, the A100 environment consists of 12 logical processors, an Intel® Xeon® CPU @ 2.20GHz, 83.47 GB of RAM, and an NVIDIA A100-SXM4-40GB GPU. All three environments use the Linux-6.1.58+-x86_64-with-glibc2.35 operating system. It should be noted that due to the functioning of Google Colab, direct comparisons of emissions between different execution environments cannot be made. This is because Google Colab assigns machines randomly, and these machines are usually located in different geographic regions. In our experiments, the CPU environment was located in South Carolina, the V100 in Groningen (PAIS), and the A100 in Iowa. To draw conclusions about which environment has higher energy consumption, both environments would need to be in the same location, so our results are intra-environmental rather than inter-environmental.

6.3 Comparison of Results between Eco-RETINA and RETINA.

Figure 1 presents a comparison of the performance of Eco-RETINA and RETINA. Both algorithms were executed in the A100 environment of Google Colab using the same hyperparameters. The only difference between the two is that Eco-RETINA does not have the bottlenecks identified in Section 4. In the first column, the models are compared in terms of MAPE (Mean Absolute Percentage Error); in the second column, execution time; and in the third column, training emissions. The first row uses 100% of the training data, the second row

uses 10%, and the third row uses only 1% of the observations from the training dataset. The subsamples are selected randomly. The aim is to assess how much is gained or lost by using subsamples. Little should be lost, according to Varian (2014).

Figure 1: Comparison between Eco-RETINA and RETINA



Source: Own elaboration

Regarding prediction error, for the dataset with 100% of observations, Eco-RETINA and RETINA show similar MAPE values. With 10 variables, Eco-RETINA has a MAPE of 0.2326, with 21 variables 0.1954, with 31 variables 0.1128, with 41 variables 0.1127, and with 50 variables 0.1066. On the other hand, RETINA presents MAPEs of 0.2352, 0.1956, 0.1128, 0.1137, and 0.1016 for 10, 21, 31, 41, and 50 variables, respectively. Interestingly, Eco-RETINA has a lower MAPE for 10, 21, and 41 variables, which could indicate some overfitting in RETINA.

For the dataset with 10% of the observations, RETINA consistently shows a lower prediction error in all experiments. However, the largest difference between the mean absolute percentage errors of Eco-RETINA and RETINA is only 0.017, which is not a significant discrepancy. In the subsample with 1% of the training dataset, Eco-RETINA achieves MAPEs of 0.2943, 0.2592, 0.2242, 0.1895, and 0.186 for 10, 21, 31, 41, and 50 variables, respectively, while RETINA's MAPEs are 0.2943, 0.2597, 0.1698, 0.1917, and 0.1942. Thus, Eco-RETINA achieves a lower MAPE for 21, 41, and 50 variables, while RETINA performs better with 31 variables, and both models have the same MAPE for 10 variables.

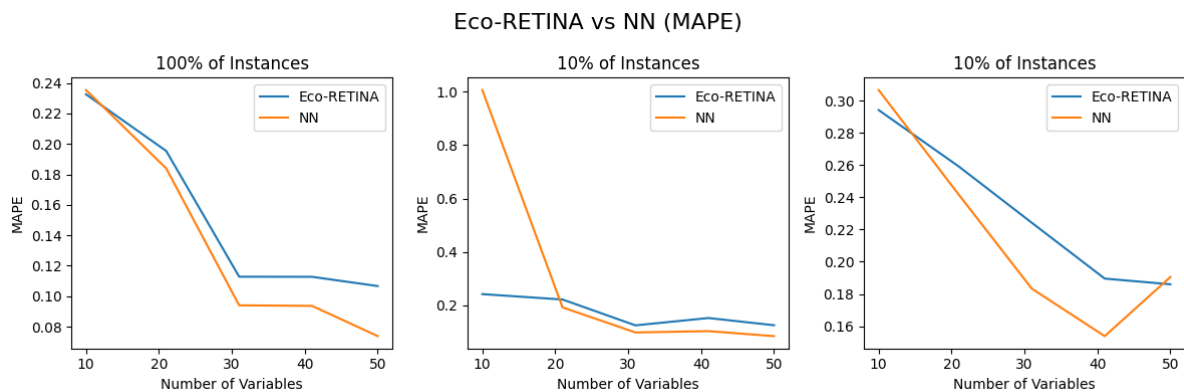
Regarding execution time and CO2 emissions from training, as shown in Figure 1, both are directly related. Eco-RETINA has shorter execution times and lower CO2 emissions. It is observed that both magnitudes increase at a higher rate for RETINA than for Eco-RETINA as the number of variables grows. For instance, with 100% of the training data and 10 variables, RETINA has an execution time of 195.41 seconds, while Eco-RETINA takes 85.32 seconds, which is 2.29 times faster. However, with 50 variables, RETINA's training time is 1146.21 seconds, while Eco-RETINA's is 340.64 seconds, approximately 4.25 times faster. This result highlights the advantages of Eco-RETINA, which arise from the increased computational cost of each regression in step I.2 as the number of variables grows, and from the greater number of regressions required as the candidate model size increases.

Additionally, the reduction in execution time and CO2 emissions when decreasing the number of observations is evident. In experiments with 50 variables, using 100% of the training dataset, Eco-RETINA's training time is 340.64 seconds with CO2 emissions of 0.004319 kg. With 10% of the observations, the time is reduced to 41.69 seconds and the emissions to 0.000500 kg. Finally, with 1% of the observations, Eco-RETINA's execution time is 10.12 seconds, and its CO2 emissions are 0.000117 kg.

The comparison between Eco-RETINA and RETINA reveals that although RETINA generally achieves a slightly lower prediction error, the difference is minimal. On the other hand, Eco-RETINA's training times and emissions are lower, and the gap between Eco-RETINA and RETINA widens as the number of variables increases.

Figure 2 compares the MAPE (Mean Absolute Percentage Error) of Eco-RETINA with that of a neural network, which has the architecture described in Section 5. The first column represents the case where both models were trained with 100% of the training data observations, the second column with 10%, and the third with 1%.

Figure 2: Comparison between Eco-RETINA and a Neural Network. MAPE



Source: Own elaboration

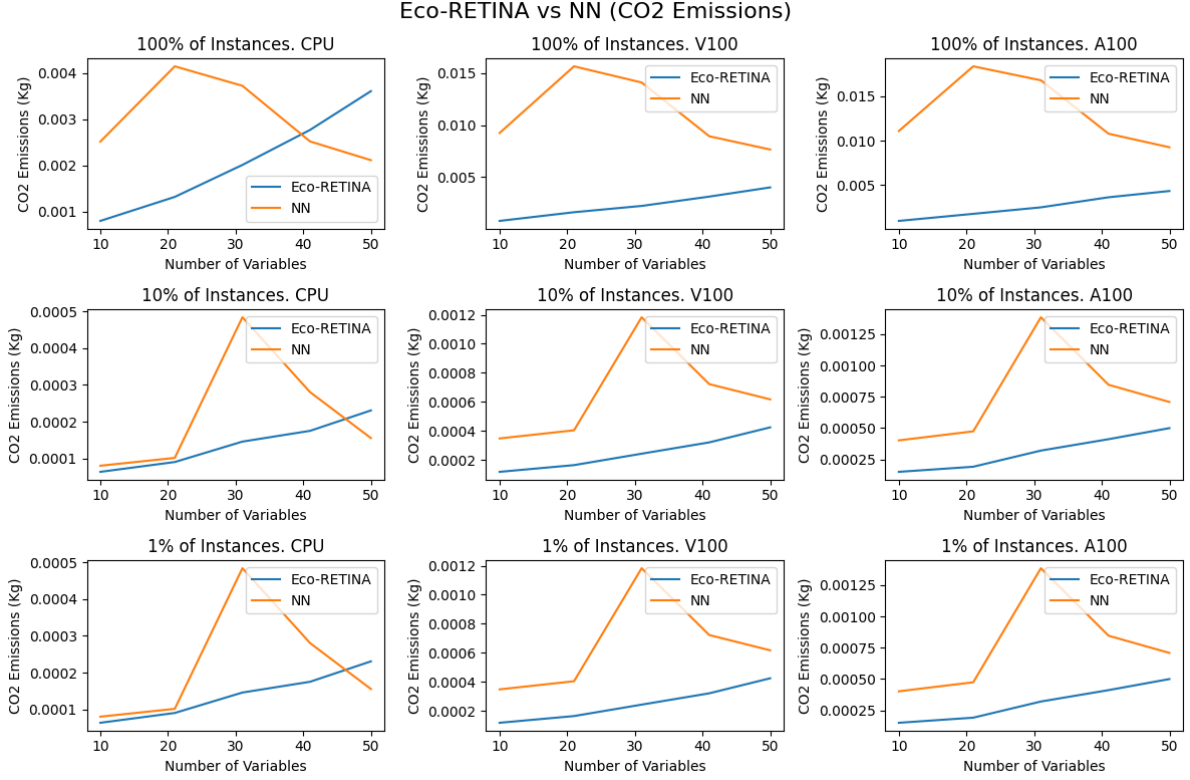
The MAPEs for Eco-RETINA have already been mentioned in the previous section. As for the neural network, for the 100% observation case, the MAPEs are 0.2354, 0.1839, 0.0940, 0.0937, and 0.0738 points for 10, 21, 31, 41, and 50 variables, respectively. With 10% of the training dataset observations, the neural network achieves MAPEs of 1.0066, 0.1928, 0.099, 0.1039, and 0.0852. Finally, with 1% of the training data, the MAPEs for the neural network are 0.3067, 0.2417, 0.1833, 0.1538, and 0.1905.

The most striking result is the MAPE of the neural network in the experiment with 10 variables and the 10% data subsample, which equals 1.0066. Although this unusually high MAPE is likely an isolated case and could be resolved by adjusting the neural network architecture, it reflects the higher sensitivity of the neural network to hyperparameters compared to Eco-RETINA, which maintained a moderate MAPE across all experiments. Another noteworthy result is that Eco-RETINA consistently produced lower prediction errors than the neural network for 10 variables across all training sets. This outcome aligns with the RETINA procedure's design.

Notice that out of sample prediction MAPEs are similar across comparable settings. This suggests that when it comes to decision-making, both models will lead to similar conclusions. In terms of policy decisions, these differences may be irrelevant in practice.

Figure 3 presents the CO2 emissions of Eco-RETINA and the neural network across various experiments. The first column shows the experiments performed in Google Colab's CPU environment, the second in the V100 environment, and the third in the A100 environment. As for the rows, the first row contains experiments using 100% of the training data, the second row 10%, and the third row 1%.

Figure 3: Comparison between Eco-RETINA and a Neural Network. CO2 Emissions



Source: Own elaboration

It can be observed that for Eco-RETINA, emissions increase as the number of variables increases, while this is not necessarily the case for the neural network. This is because emissions in a given environment are directly related to execution time, which depends on the number of epochs used to train the network. As seen in Tables 1, 2, and 3 above, a higher number of variables does not always correspond to more epochs. In the CPU environment, for the 100% training dataset, Eco-RETINA's CO2 emissions are 0.0008 kg, 0.0013 kg, 0.002 kg, 0.0028 kg, and 0.0036 kg, while the neural network's emissions are 0.0025 kg, 0.0042 kg, 0.0037 kg, 0.0025 kg, and 0.0021 kg.

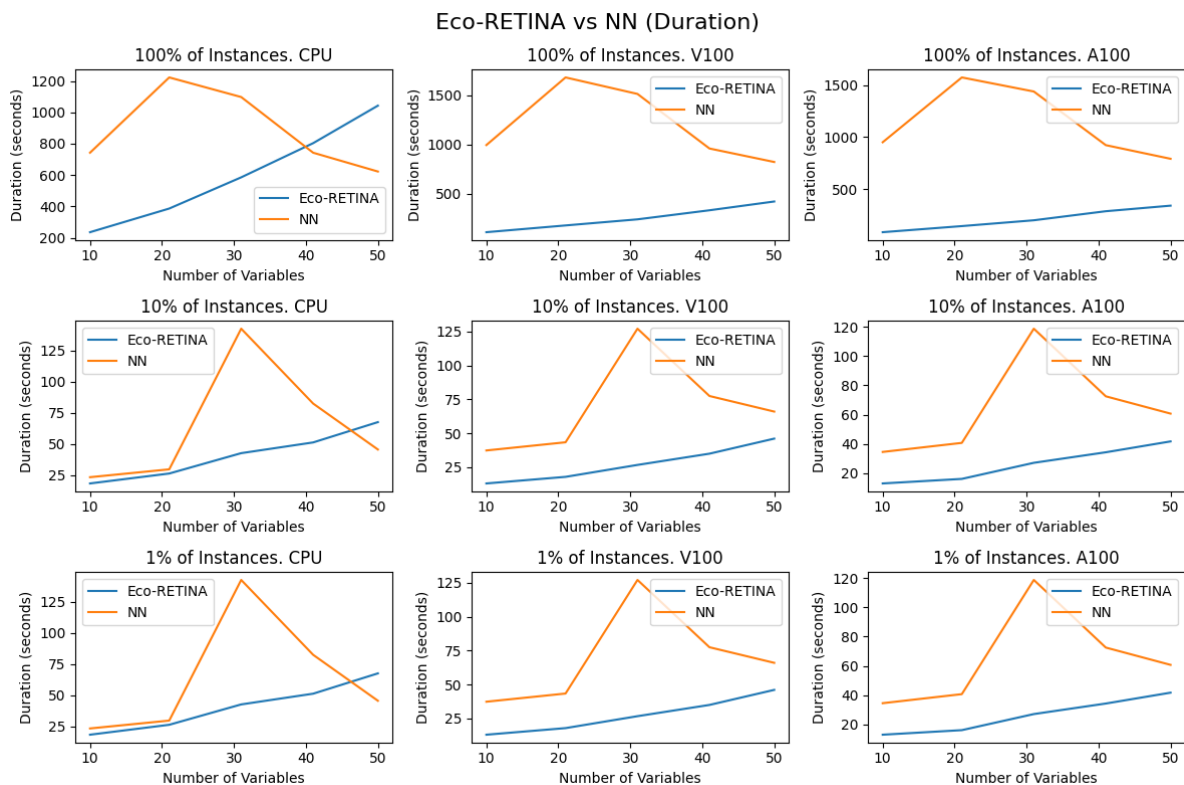
With 10% of the training dataset, Eco-RETINA's CO2 emissions are 6.33×10^{-5} kg, 9.01×10^{-5} kg, 1.46×10^{-4} kg, 1.75×10^{-4} kg, and 2.31×10^{-4} kg. In contrast, the neural network's emissions are 8×10^{-5} kg, 1.01×10^{-4} kg, 4.84×10^{-4} kg, 2.8×10^{-4} kg, and 1.55×10^{-4} kg. For the 1% subsample, Eco-RETINA shows emissions of 1.33×10^{-5} kg, 1.48×10^{-5} kg, 2.54×10^{-5} kg, 3.24×10^{-5} kg, and 3.46×10^{-5} kg, while the neural network's emissions are 7.18×10^{-5} kg, 9.15×10^{-5} kg, 3.53×10^{-5} kg, 5.04×10^{-5} kg, and 7.19×10^{-5} kg.

In the V100 and A100 environments, Eco-RETINA consistently achieves lower emissions than the neural network for any training set size and number of variables. For 100% of the training data, the cases of 10 and 21 variables stand out, where Eco-RETINA's emissions are approximately 10 times lower than those of the neural network in both execution environments.

Figures 1 and 3 indicate CO2 emissions are significantly different when training algorithms with fewer data. Given the moderate difference in MAPEs between models trained with 100%, 10%, and 1% of the data, it is worth considering whether to use only a representative subsample when handling large datasets, as suggested by Varian (2014), the chief economist at Google.

Figure 4 below presents the execution times for the various experiments. The charts are laid out in the same way as in Figure 3, with the only difference being that the y-axis now represents training duration instead of CO2 emissions.

Figure 4: Comparison between Eco-RETINA and a Neural Network. Training Duration



Source: Own elaboration

Figures 3 and 4 seem to indicate a direct relationship between training time and CO2 emissions, as both curves follow the same pattern. The results show that for a given execution environment, longer training times result in higher emissions. Moreover, the time-emissions relationship is influenced by the execution environment, as there are cases where, despite shorter training times, emissions are higher in specific environments. While studying the influence of execution environments on emissions is beyond the scope of this work, it would be interesting to investigate how much of this influence is due to the execution environment,

the geographic location of the machine, or other factors such as the time of year when the experiment is conducted.

When analyzing the results as a whole, comparing the average MAPE and emissions of Eco-RETINA and the neural network, Eco-RETINA has an average MAPE of 14.23% lower than the neural network, and emissions are 73.06% lower. However, the lower MAPE is due to the neural network's MAPE exceeding 1 in the experiment with 10 variables and 10% of the training data. If this experiment is excluded, Eco-RETINA's average MAPE is 13.51% higher than the neural network's, and its emissions are 73.13% lower. This outcome reflects that although Eco-RETINA generally has higher prediction errors than the neural network, it consumes significantly less energy and time.

6.4 Interpretability of Eco-RETINA

A key feature of Eco-RETINA is its straightforward interpretability. The algorithm generates a clear forecasting model in a standard regression output format, as shown in Figure 5.

Figure 5: Eco-RETINA output Summary

eco_retina.model.summary()

✓ 0.0s

OLS Regression Results							
Dep. Variable:	obj_variable	R-squared:	0.839				
Model:	OLS	Adj. R-squared:	0.839				
Method:	Least Squares	F-statistic:	5.631e+04				
Date:	Sun, 14 Apr 2024	Prob (F-statistic):	0.00				
Time:	18:39:16	Log-Likelihood:	-7.0650e+05				
No. Observations:	86553	AIC:	1.413e+06				
Df Residuals:	86544	BIC:	1.413e+06				
Df Model:	8						
Covariance Type:	nonrobust						
		coef	std err	t	P> t	[0.025	0.975]
	constant	2406.9761	1093.325	2.202	0.028	264.068	4549.884
	carat_weight^1*meas_width^1	1012.5979	1.530	662.004	0.000	1009.600	1015.596
	meas_width^1*meas_length^1	265.2951	18.546	14.304	0.000	228.944	301.646
	polish_Excellent*depth_percent	110.3873	25.818	4.276	0.000	59.784	160.991
	polish_Excellent*table_percent	-75.9719	27.756	-2.737	0.006	-130.373	-21.571
	polish_Very Good	2149.2133	603.394	3.562	0.000	966.566	3331.860
	polish_Excellent	257.7628	584.608	0.441	0.659	-888.064	1403.590
	polish_Very Good*depth_percent	91.2128	26.122	3.492	0.000	40.013	142.412
	polish_Very Good*table_percent	-91.0710	27.773	-3.279	0.001	-145.506	-36.636
	table_percent^1*depth_percent^1	-5306.3593	1510.614	-3.513	0.000	-8267.151	-2345.568
Omnibus:	13326.421	Durbin-Watson:	2.006				
Prob(Omnibus):	0.000	Jarque-Bera (JB):	166250.786				
Skew:	0.336	Prob(JB):	0.00				
Kurtosis:	9.756	Cond. No.	2.34e+16				

Source: Self-compiled

In this example the model selected by Eco-RETINA relates the dependent variable with the relevant list of (transformed) inputs in a straightforward expression, similar to the interpretation of interactions in regression models.

- constant
- carat_weight*meas_width: carat weight times measured width
- meas_length/meas_width: measured length over measured width
- polish_Excellent*depth_percent: Excellent polish times percent depth
- polish_Excellent*table_percent: Excellent polish times percent table
- polish_very good: very good polish
- polish-Excellent: Excellent polish
- polish_Very Good*table_percent: very good polish times percent table
- polish_very good*depth_percent: very good polish times percent depth
- depth_percent/table_percent: percent depth over percent table

The algorithm gives us a parsimonious model using 9 (possibly transformed) inputs plus the constant. Eco-RETINA also provides with

- Point estimates of the coefficients,
- Standard errors,
- Hypothesis testing
- F test of joint significance of coefficients
- t-statistics,
- p-values,
- Confidence intervals.

And a variety of tests and diagnostics, such as

- R^2 , coefficient of determination,
- Log-likelihood,
- AIC,
- BIC,
- Jarque-Bera, etc.

The interpretation of the variables and coefficients is straightforward (the two negative coefficients are challenging), and the appendix suggests that a simple interpretable model can be used for out-of-sample forecasting and for creating alternative scenarios. Eco-RETINA's results are directly replicable using standard regression software. Out-of-sample forecasting using this model is trivial and should consume a minimal amount of resources.

7. Conclusions.

Eco-RETINA has been shown to achieve prediction errors comparable to those of RETINA, while demonstrating significantly lower power consumption across numerous experimental settings and achieving execution speeds up to 4.25 times faster. Furthermore, Eco-RETINA has been benchmarked against neural networks. Although the prediction error of neural networks is slightly lower, the carbon emissions associated with training Eco-RETINA are significantly reduced in most cases.

The experimental results further indicate that Eco-RETINA surpasses neural networks in predictive accuracy when applied to problems involving a moderate number of inputs. Its reduced carbon footprint, combined with its open and transparent design, establishes Eco-RETINA as a strong contender among Green AI algorithms. Notably, out-of-sample prediction mean absolute percentage errors (MAPEs) remain similar across comparable settings, implying that both Eco-RETINA and neural networks are likely to lead to comparable decision-making outcomes, as these differences may lack practical significance.

Eco-RETINA's modular architecture, transparency, and interpretability enhance its accessibility and usability. Its focus on speed, accuracy, and environmental sustainability positions it as a powerful tool for both research and practical applications. A key contribution of Eco-RETINA is its ability to provide an explicit, interpretable forecasting model based on the transformed inputs, offering estimated coefficients, t- and F-statistics, coefficients of determination, and various diagnostic measures. It is particularly useful for scenario formulation and policy analysis.

While Eco-RETINA and neural networks can act as substitutes in certain applications, they also demonstrate strong complementarity. Specifically, Eco-RETINA serves as an effective exploratory tool for identifying promising subsets of transformed inputs at a lower computational and environmental cost compared to neural networks.

To further advance its capabilities, additional experiments should be conducted across diverse datasets and problem domains. As the development of Eco-RETINA progresses, integrating new libraries and optimization techniques could further reduce execution times and emissions, enhancing its overall efficiency and impact.

With its speed, accuracy, openness, interpretability, and environmental sustainability, Eco-RETINA represents a key development in the realm of Green AI. As the demand for sustainable computing continues to grow, algorithms like Eco-RETINA are poised to play an essential role in addressing the challenges of future AI development. The programming is currently undergoing substantial revisions, with further improvements expected in the near future.

References

- Barbierato, E., & Gatti, A.** (2024). Towards Green AI: A methodological survey of the scientific literature. *IEEE Access*.
- Bouza, L., Lannelongue, L., & Dahdaleh, V. P.** (2023). How to estimate carbon footprint when training deep learning models? A guide and review. Published September 25.
- Capilla, J.** (2024a). *Modelado automático con RETINA desde un enfoque de Green AI* [Undergraduate thesis]. Facultad de Económicas y Empresariales, Universidad Complutense de Madrid.
- Capilla, J.** (2024b). GitHub. Retrieved from <https://github.com/jcapilla780?tab=repositories>
- Hrokr.** (2023). *The largest diamond dataset currently on Kaggle*. Retrieved from <https://www.kaggle.com/datasets/hrokrin/the-largest-diamond-dataset-currently-on-kaggle>
- Kaack, L. H., Donti, P. L., Strubell, E., Kamiya, G., Creutzig, F., et al.** (2021). Aligning artificial intelligence with climate change mitigation. *HAL Archives*, ffhal-03368037f.
- OECD.** (2022). *Measuring the environmental impacts of AI compute and applications: The AI footprint* (No. 341). *Digital Economy Papers*, November.
- Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O.** (2020). Green AI. *Communications of the ACM*, 63(12), 54-63.
- Strubell, E., Ganesh, A., & McCallum, A.** (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3645–3650). Florence, Italy: Association for Computational Linguistics.
- Verdecchia, R., Sallou, J., & Cruz, L.** (2023). A systematic review of Green AI. *WIREs Data Mining and Knowledge Discovery*, 13, e1507. <https://doi.org/10.1002/widm.1507>
- Marinucci, M.** (2005). *RETINA Winpack for real data: A quick guide for automatic model selection*. Report, Universidad Complutense de Madrid.
- Marinucci, M.** (2007). *RETINA: Relevant transformations of the inputs network approach* [PhD dissertation]. Universidad Complutense de Madrid.
- Pérez-Amaral, T., Gallo, G., & White, H.** (2005). A comparison of complementary automatic modelling methods: RETINA and PCGETS. *Econometric Theory*, 21(2), 262-277.
- Pérez-Amaral, T., Gallo, G., & White, H.** (2003). A flexible tool for model building: The relevant transformations of the inputs network approach, RETINA. *Oxford Bulletin of Economics and Statistics*, 65(Supplement), 305-323.

Schmidt, V., Goyal, K., Joshi, A., Feld, B., Conell, L., Laskaris, N., Blank, D., Wilson, J., Friedler, S., & Luccioni, S. (2021). CodeCarbon: Estimate and track carbon emissions from machine learning computing. *Zenodo*. <https://doi.org/10.5281/zenodo.4658424>

Varian, H. R. (2014). Big data: New tricks for econometrics. *Journal of Economic Perspectives*, 28(2), 3-28.

White, H. (1998). *Artificial neural networks and alternative methods for assessing naval readiness* [Technical report]. NRDA, San Diego.

9. APPENDIX - THE ECO-RETINA ESTIMATOR AND INTERFACE

This appendix explains how to use the tools developed in this work. Eco-RETINA has been implemented as both a Python estimator and a user interface. Although some Eco-RETINA hyperparameters have already been introduced in discussing the new features offered, this section delves into further detail. Besides describing the hyperparameters, it will also show how to use the estimator and the user interface.

All necessary files to use Eco-RETINA or reproduce the experiments in this work will be available in Capilla (2024b).

9.1. Requirements for Use

Both the Eco-RETINA estimator and the user interface require an internet connection and an Intel processor. Additionally, using the estimator requires either Python and a code editor, like VSCode, or a Python distribution, such as Anaconda.

To use the Python estimator, you must download the files `eco_retina.py` and `my_functions_eco.py`. Additionally, all packages listed in `requirements.txt` must be installed in the virtual environment. For the user interface, you only need to download the `eco_retina.exe` file available in the GitHub repository. Once downloaded and installed, it can be used without requiring any additional installations.

9.2. The Estimator

As a typical Python estimator, Eco-RETINA includes fit and predict methods to train the model and make predictions, typically for out-of-sample data once the model is trained. The fit method accepts the following inputs:

- **y DataFrame:** Contains the target variable. An error message appears if no y value is provided.
- **x_c_t DataFrame:** Contains continuous variables for all possible transformations. The default is None.
- **x_c_nt DataFrame:** Contains continuous variables for only cross-product transformations. The default is None.
- **x_d DataFrame:** Contains dummy variables. The default is None.

While values for `x_c`, `x_c_nt`, and `x_d` are optional, at least one must be provided; otherwise, an error message appears. Additionally, the fit method includes the following hyperparameters:

- **seed:** This function sets the seed for Eco-RETINA's subsampling. It takes positive integer values, with a default of 8.
- **max_collinearity:** Specifies the maximum collinearity allowed by Eco-RETINA, with values ranging from 0 to 1 and a default of 0.5.
- **col_params_search and max_num_variables:** These two hyperparameters facilitate an automated search for the maximum collinearity (less than `max_collinearity`) and the distance between grid levels of collinearity. `col_params_search` takes boolean values (True or False) depending on whether or not automatic search is desired, while `max_num_variables` accepts positive integers, defaulting to 50.

- **collinearity_grid**: Forms the grid levels for collinearity. Takes values between 0 and 1. For example, if `max_collinearity` equals 0.5 and `collinearity_grid` equals 0.01, the grid for collinearity levels would be 0, 0.01, 0.02, ..., 0.49, 0.5. These levels index regressor sets in phase 2b of RETINA. The default for `collinearity_grid` is 0.005; note that if `col_params_search` is True, Eco-RETINA will ignore the assigned value and select automatically based on variable count.
- **iqr_multiplier and remove**: Used for outlier removal. `iqr_multiplier` sets the interquartile range multiplier for outlier elimination, with a default of 1.5. `remove` specifies whether or not to eliminate outliers (using `iqr_multiplier`), taking True if outliers should be removed and False otherwise. The default is False.
- **power_list**: A list of values for alpha and beta used to obtain transformed variables, as described by Perez-Amaral et al. (2003). The default is [-1,0,1], which generates original variables, squares, cross-products, inverses, and cross-ratios. Additional alpha and beta values must be ordered from lowest to highest; for example, to include powers of 3 and 4, `power_list` should be [-2,-1,0,1,2].
- **metric**: Sets the metric used in phases 3a and 4b for model selection, with options including 'MSE', 'AIC', or 'BIC'. The default is 'MSE'.
- **regression_type**: Indicates the regression type. Possible values include 'Linear' for linear regression, 'Binary (LPM)' for a linear probability model, 'Binary (logit)' for logistic regression, and 'Binary (probit)' for probit regression. The default is 'Linear'.
- **cov_type**: Specifies the covariance type in the regression, either 'nonrobust' (for non-robust covariance) or 'HC0', 'HC1', 'HC2', or 'HC3' (for heteroskedasticity-consistent covariances). The default is 'nonrobust'.
- **log_type**: Determines logarithmic transformations, with options: 'no' (no transformation), 'x' (applied to continuous regressors), 'y' (applied to the target variable), or 'x & y' (applied to both continuous regressors and the target variable). The default is 'no'.
- **standardize**: Standardizes continuous original variables if set to True, otherwise False. The default is False.
- **multiprocessing**: Determines if each iteration of Eco-RETINA runs in parallel (True) or sequentially (False). The default is True.
- **validation and random_state**: The validation hyperparameter divides the dataset into four subsamples instead of three, using the fourth for validation. Takes True for validation or False otherwise, with a default of False. If True, attributes `score1` and `score2` provide metrics to assess model performance on the validation set. For `regression_type` as 'Linear', `score1` returns MSE and `score2` returns MAE. For `regression_type` as 'Binary (LPM)', 'Binary (logit)', or 'Binary (probit)', `score1` returns the accuracy score and `score2` the ROC AUC score. The `random_state` hyperparameter sets a seed for validation set separation and accepts positive integers, with a default of 42.
- **eco**: Specifies whether to use RETINA or Eco-RETINA. If `eco` is set to True, Eco-RETINA will be used; if False, RETINA will be used. The default value is True. The primary difference between the two approaches is that Eco-RETINA avoids the bottlenecks present in RETINA.

9.3. Example of Estimator Use

To use Eco-RETINA, input data must be formatted as required. This includes a pandas DataFrame containing the target variable (in our example, `y_train`), a DataFrame with continuous variables for all transformations (in our example, `x_c_t_train`), a DataFrame with continuous variables for only cross-product transformations, and another DataFrame for dummy variables (in our example, `x_d_train`). In

this example, we allow all continuous variables to undergo all possible transformations, so `x_c_nt` will be set to `None`.

Once we have these inputs, simply import the `Eco_RETINA` class from the file `eco_retina.py`, create an instance of that class (named `eco_retina` in this example), and use the `fit` method with the desired inputs and hyperparameters. Figure 5 shows these steps in action. In this example, we have set the `max_num_variables` hyperparameter to 100, `max_collinearity` to 0.7, and `validation` to `True`. The remaining hyperparameters use their default values.

Figure 6: Training Eco-RETINA

```
from eco_retina import Eco_RETINA
eco_retina=Eco_RETINA()
eco_retina.fit(y=y_train,x_c_t=x_c_t_train,x_c_nt=None,x_d=x_d_train, max_num_variables=100, max_collinearity=0.7, validation=True)
```

✓ 55.8s

Source: Self-compiled

Once the model is trained, a summary of the results can be viewed using the `model.summary` method, as shown in Figure 5, in section 6.4 of the text.

Since we set `validation` to `True`, the attributes `score1` and `score2` are accessible, as shown in Figure 7.

Figure 7: Eco-RETINA Score1 and Score2

```
eco_retina.score1
```

✓ 0.0s

718295.9939484355

```
eco_retina.score2
```

✓ 0.0s

527.2415300297804

Source: Self-compiled

Another important attribute is `emissions`, which provides information on emissions and energy consumption during algorithm training. In this work we use Codecarbon, which proved to be superior to other software in terms of reliability and possibility of parallelization. Figure 8 provides an example.

Figure 8: Eco-RETINA Emissions

```
eco_retina.emissions
```

✓ 0.0s

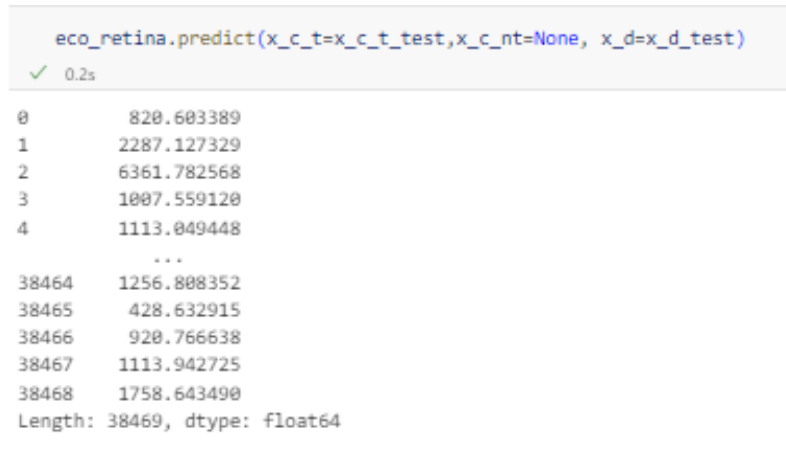
	timestamp	project_name	run_id	duration	emissions	emissions_rate	cpu_power	gpu_power	ram_power	cpu_energy	...
0	2024-04-14T18:39:12	Eco-RETINA	cfe0c450-6421-4877-bdec-a75edb4e6be5	50.759699	0.000022	4.271986e-07	7.5	0.0	0.13373	0.000106	...

1 rows x 31 columns

Source. self-constructed

Finally, to use the model for predictions, you can use the predict method. The inputs for this method include a DataFrame `x_c_t`, another `x_c_nt`, and another `x_d`, which are equivalent to those in the fit method. An example is shown in Figure 9.

Figure 9: Eco-RETINA Predictions



Source: Self-constructed

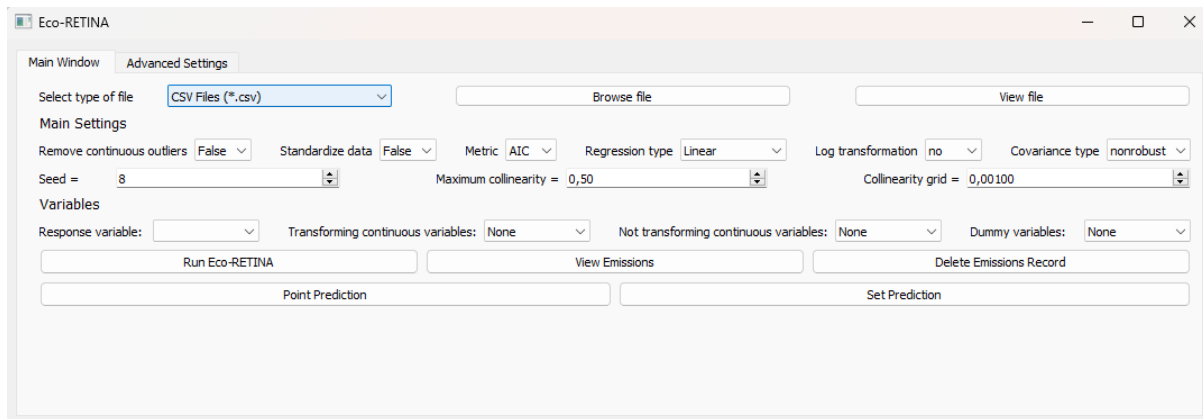
9.4. User Interface

The user interface is designed to be intuitive. To begin:

1. **Data Selection:** Choose your data's file type and locate it using the file browser.
2. **Hyperparameter Configuration:** After selecting the desired hyperparameters, initiate the training process by clicking the "Run Eco-RETINA" button.
3. **Energy Consumption and Emissions:**
 - To view the energy consumption and emissions during training, click the "View Emissions" button.
 - To delete the emissions record, click the "Delete Emissions Record" button.
4. **Predictions:**
 - For a single data point prediction, click the "Point Prediction" button.
 - To predict a dataset, click the "Set Prediction" button.

Figures 10 and 11 illustrate the appearance of each tab in the Eco-RETINA interface.

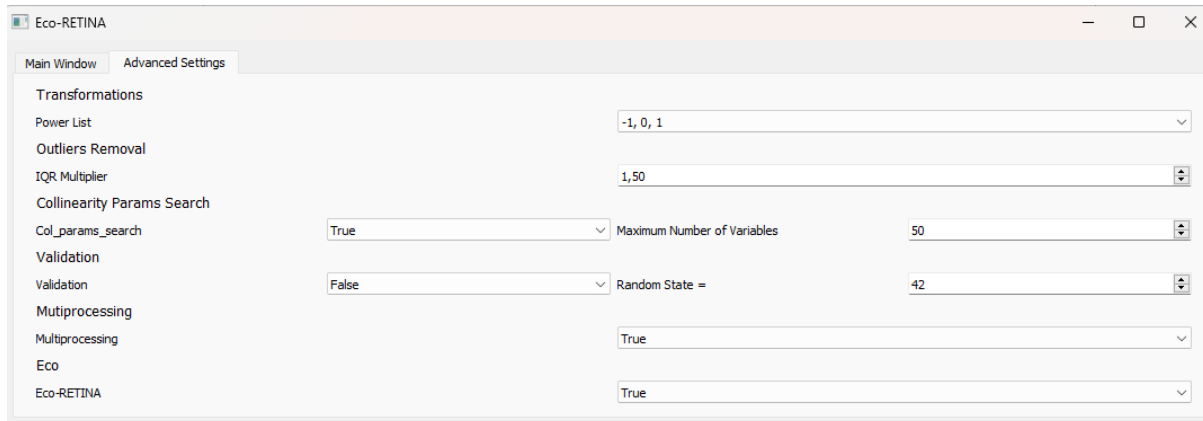
Figure 10: Main Tab of the Eco-RETINA GUI



The screenshot shows the 'Main Window' tab of the Eco-RETINA GUI. It features a 'Main Settings' section with various configuration options. At the top, there's a 'Select type of file' dropdown set to 'CSV Files (*.csv)', with 'Browse file' and 'View file' buttons. Below this, the 'Main Settings' section includes: 'Remove continuous outliers' (False), 'Standardize data' (False), 'Metric' (AIC), 'Regression type' (Linear), 'Log transformation' (no), and 'Covariance type' (nonrobust). A 'Seed =' field is set to 8, and 'Maximum collinearity =' is set to 0,50. A 'Collinearity grid =' field is set to 0,00100. Under 'Variables', there are dropdowns for 'Response variable:', 'Transforming continuous variables:' (None), 'Not transforming continuous variables:' (None), and 'Dummy variables:' (None). At the bottom, there are four buttons: 'Run Eco-RETINA', 'View Emissions', 'Delete Emissions Record', and 'Point Prediction'.

Source: Self-compiled

Figure 11: Advanced Settings Tab of the Eco-RETINA GUI



The screenshot shows the 'Advanced Settings' tab of the Eco-RETINA GUI. It contains a list of settings on the left and their corresponding values on the right. The settings and their values are: 'Power List' (-1, 0, 1), 'Outliers Removal' (1,50), 'Collinearity Params Search' (True), 'Col_params_search' (True), 'Maximum Number of Variables' (50), 'Validation' (False), 'Random State =' (42), 'Mutiprocessing' (True), 'Eco' (True), and 'Eco-RETINA' (True).

Source: Self-compiled