Antbear 1.00

Introduction

Antbear allows its users to solve various real-world machine learning (ML) problems using genetic programming (GP). This can be an extremely useful technique when the problem domain is poorly understood - particularly when obtaining a gradient associated with the Error Metric is impossible. In addition, unlike other techniques the model produced is in a human-readable format whether it be in C, R or Python code. This can lend a business intuition into what parameters are important and how these parameters interact with each other.

It should be noted that GP is a computationally expensive process and so one should be aware that although it can solve linear regression problems one should really use a mathematical technique which will be much more efficient.

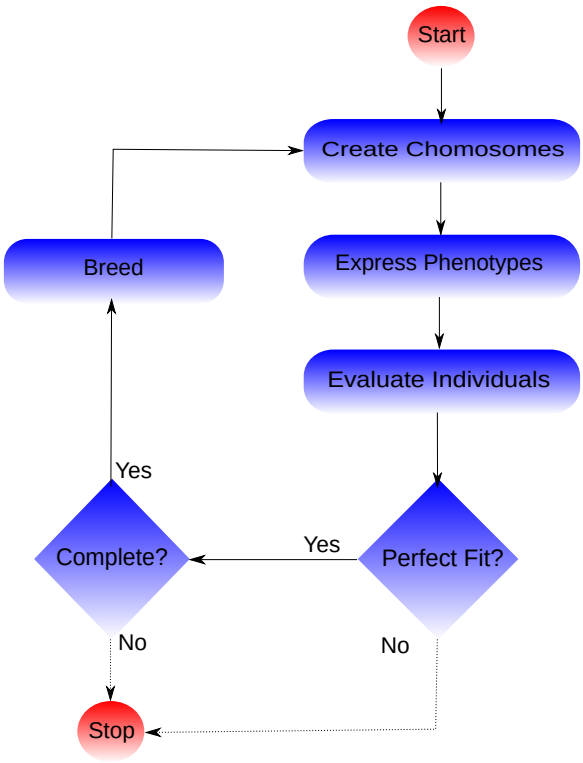

Figure 1: Genetic Programming Flowchart

Genetic Programming

In artificial intelligence, genetic programming (GP) is a technique of evolving programs. Initially a population with completely random genetic chomosomes is created. Each individual then expresses these chromosomes as code which is evaluated on a GPU. (In our case we use CUDA on NVidia cards)

After all the evaluations have been completed the best individuals in terms of the Error Metric are selected for reproduction. The children have a combination of their parents. The crossover operation involves swapping random parts of selected pairs (parents) to produce new and different offspring that become part of the new generation of programs. Mutation involves substitution of some random part of a program with some other random part of a program. Some programs not selected for reproduction are copied from the current generation to the new generation. Typically, members of each new generation are on average more fit than the members of the previous generation, and the best-of-generation program is often better than the best-of-generation programs from previous generations. Termination of the evolution usually occurs when some individual program reaches a predefined proficiency or fitness level.

User Interface

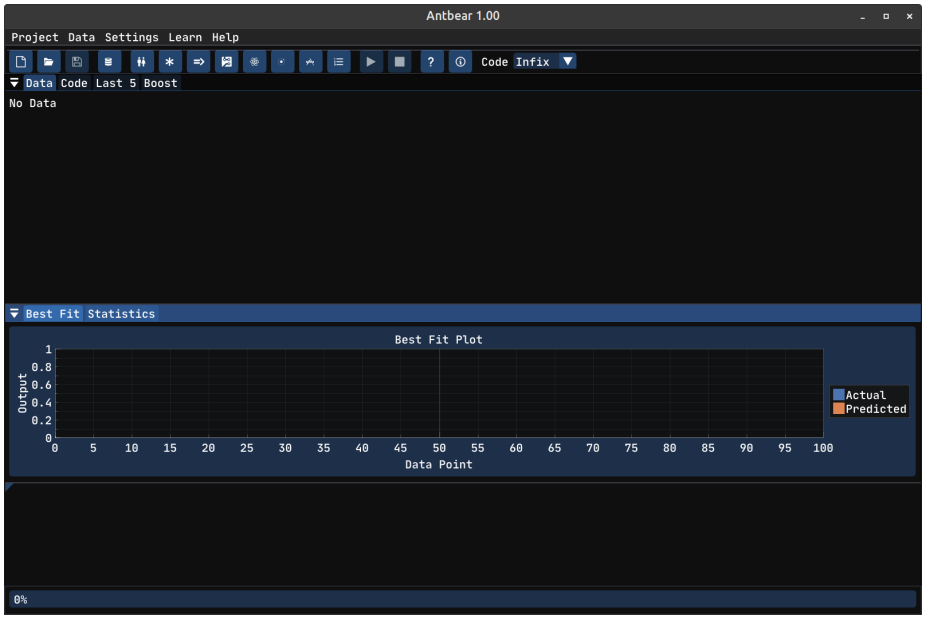

Figure 2: Main Window

Main Window

The main UI is split into three sections; the data and code section, the best fit and statistics section and the status section.

Data and Code

The data window displays the data used for the current project. Please note that it is not a substitute for a spreadsheet. It does not allow sorting or editing. It functions as a way of sanity checking that the data is sensible and valid for the purposes of training.

As training progresses the code windows will show the actual code for the best program in the current population. The desired coding language can be selected from the toolbar. At present the languages are Infix, C, R, Python and Latex.

Best Fit and Statistics

When the training is progressing the Best Fit chart will show how the predicted outputs match the actual target. This can be usefully not only for getting a feel of how training its progressing but can also catch whether the wrong metric is being used for the problem. The statistics view shows how the error is

progressing with respect to generations. This is useful in determing whether the model is likely to over or under fit the data especially when one select a holdout portion of the data.

Status

All important messages are displayed in this window. Any major errors will be shown hear along with important milestones in terms of progression. The progress bar showns the amount of training performed together with an estimated finish time for training.

Importing Data

Antbear expects data to be in CSV format only and all values must be in a valid float32 format. No infinities or non-numeric characters. If one has invalid entries please label them by using a large negative or positive number outside the current normal range of valid entries and specify this as a label using the NaN setting.

Population

Please note that it is critical that one starts with the default parameters for an initial run on any new project. This will give a solid baseline that can be optimized in an iterative fashion.

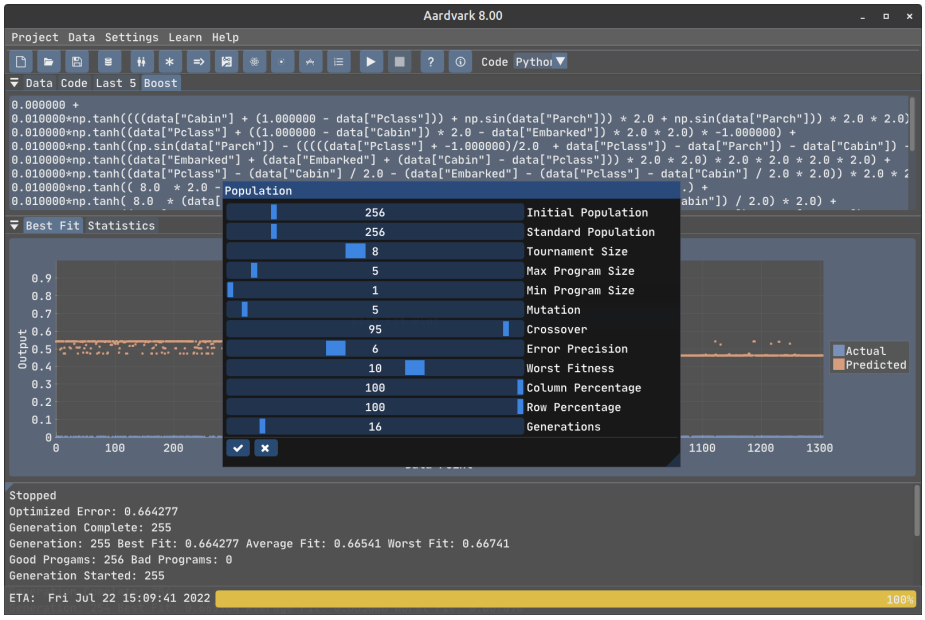

Figure 3: Population Dialog

Initial Population

The genetic information of an individual for the initial population is created wholly at random. As such it might be wise to increase the size of the initial population with respect to every other generation. This is especially true for projects with a number of false minima.

Standard Population

The standard population relates to every generation other than the initial one.

Tournament Size

This determined the number of individuals that are in a mini tournament to see which one gets to go forward to the main tournament proper. For instance a value of 8 means that 8 individuals are created from the best individuals from the previous generation. The best one in terms of average parental fit gets to go into the main tournament whilst the others are discarded.

Minimum Program Size

The minimum size of an individual program. For example a a zeroth function would count as 1 (An actual value such as 0,1,2 or x,y,z). A unary would count as 2 (sin(x) one for the function and one for the parameter). A binary would count as 3 (x<y one for the function and two for both parameters)

Maximum Program Size

The maximum size of each individual program. Note most programs will have a size typically between the mimimum and the maximum values.

Mutation

The percentage chance that an indivudals genetic code will mutate. As with genetics mutations in the real world this is usually a bad thing and so should be a low value.

Crossover

The percentage chance that two parents will breed. A value of zero will ensure that the initial population will live forever and any change will be due to mutations. A value of 100 will ensure no parents last more than one generation.

Error Precision

A value of three indicates that all data values are rounded off to three deciaml places.

Worst Fitness

A value of 6 indicates that the worst fitness of an error metric is truncated to one million. Note that least squares with huge data values and targets will require huge values in terms of this parameter to have any hope of converging. It is recommended that if this still doesn't work then it might be worth prescaling your data prior to presenting it to Antbear.

Column Percentage

The percentage of columns used per generation

Row Percentage

The percentage of rows used per generation. *Note that both Column Percentage and Row Percentage can have a marked effect on reducing overfitting*

Generations (Per Boost)

The total generations per Boost epoch.

Error Metrics

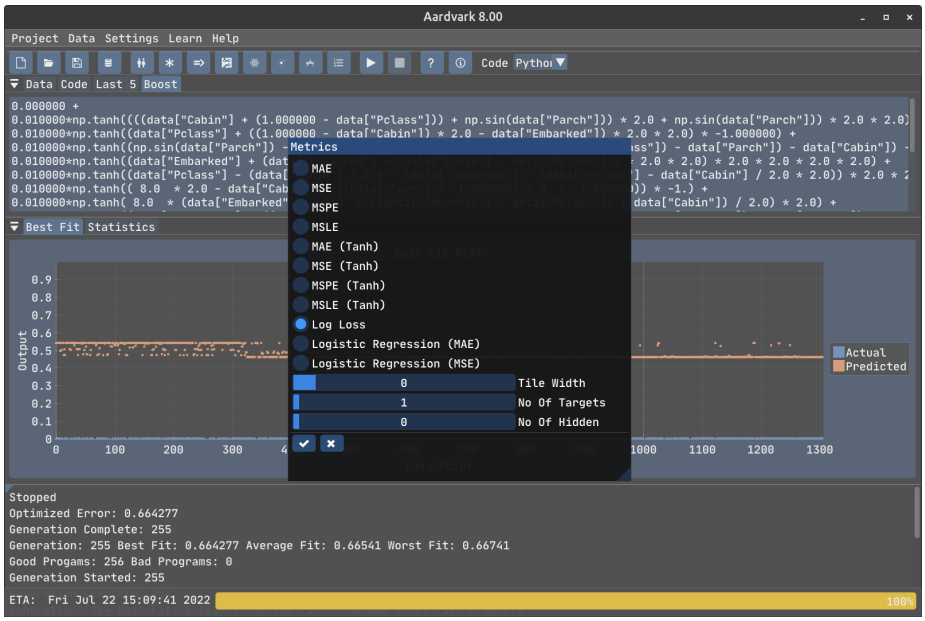

Figure 4: Metrics Dialog

Mean Absolute Error

$$Error = \frac{1}{n}\sum_{i=1}^{n}|p_i - y_i|$$

Mean Square Error

$$Error = \frac{1}{n}\sum_{i=1}^{n}p_i^2 - y_i^2$$

Mean Square Percentage Error

$$Error = \frac{1}{n}\sum_{i=1}^{n}\frac{(p_i - y_i)^2}{y_i^2}$$

Mean Square Log Error

$$Error = \frac{1}{n}\sum_{i=1}^{n}(log(1 + p_i) - log(1 + y_i))^2 \quad \text{Note: } y \subset \mathbb{R}^+$$

Mean Absolute Error (Tanh)

Individual boost outputs are wrapped inside a tanh function

$$Error = \frac{1}{n}\sum_{i=1}^{n}|p_i - y_i|$$

Mean Square Error (Tanh)

Individual boost outputs are wrapped inside a tanh function

$$Error = \frac{1}{n}\sum_{i=1}^{n}p_i^2 - y_i^2$$

Mean Square Percentage Error (Tanh)

Individual boost outputs are wrapped inside a tanh function

$$Error = \frac{1}{n}\sum_{i=1}^{n}\frac{(p_i - y_i)^2}{y_i^2} \quad \text{Note: } y \subset \mathbb{R}^+$$

Mean Square Log Error (Tanh)

Individual boost outputs are wrapped inside a tanh function

$$Error = \frac{1}{n}\sum_{i=1}^{n}(log(1+p_i) - log(1+y_i))^2 \quad \text{Note: } y \subset \mathbb{R}^+$$

Log Loss Error

$$Error = -\frac{1}{n}\sum_{i=1}^{n} -y_i log(x_i) + (1-y_i) log(1-x_i) \quad \text{Note: } \forall\, y \in (0,1)$$

Raw output is wrapped inside a sigmoid function which is omitted for clarity

Logistic Regression Error (MAE)

$$Error = \frac{1}{n}\sum_{i=1}^{n} |p_i - y_i| \quad \text{Note: } \forall\, y \in (0,1)$$

Raw output is wrapped inside a sigmoid function which is omitted for clarity

Logistic Regression Error (MSE)

$$Error = \frac{1}{n}\sum_{i=1}^{n} p_i^2 - y_i^2 \quad \text{Note: } \forall\, y \in (0,1)$$

Raw output is wrapped inside a sigmoid function which is omitted for clarity

Tile Width

This is an experimental feature for Images and assumes that the data represent a monochome square image. A tile value of 4 represents a 4 by 4 tile in a similar fashion to thos used for convolutions. A tile with of zero disable this feature.

Number of Targets

The number of target classes for training. Typically this would mainly be used with logloss and the softmax option for multiclass classification.

Number of Hidden

It can be worth using an abstraction layer and using the output parameters from this layer to create the predictions. It can also be be used to produced an N-dimension cluster associated with the targets. An example of this clustering technique is given in the tutorials section.

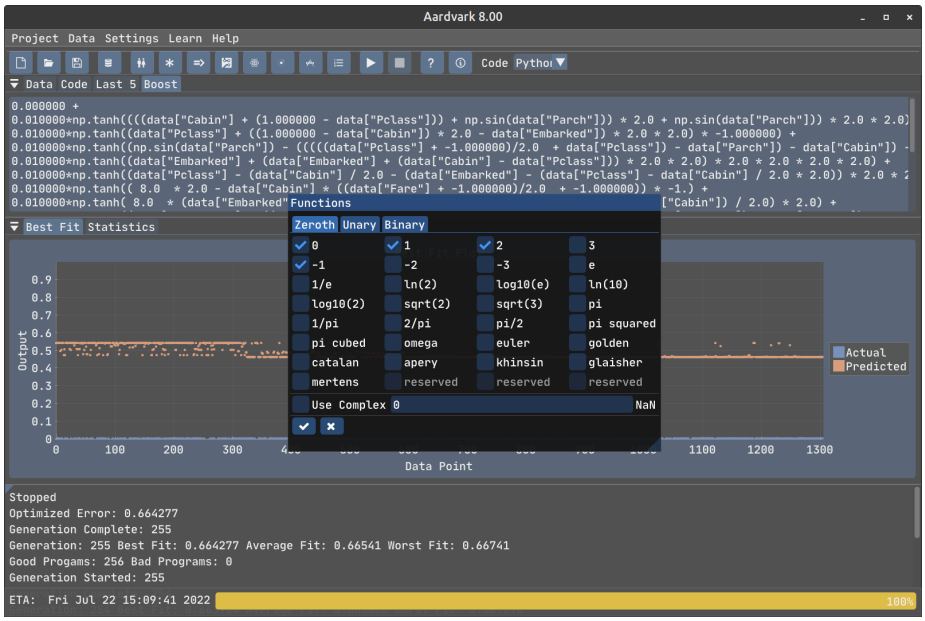

Figure 5: Functions Dialog

Functions

The functions are split into 3 sections depending on their contribution to overall program size. As with other settings it is advised to start a new project with the defaults as complexity can cause both overfitting and make it harder to interpret the best models.

Zeroth Function

The standard predefined constants that can be used by Antbear.

Unary Function

The unary functions such as sin, cos, tanh that are available for use by individual programs.

Binary Function

The binary function such as less or greater than available from Antbear.

Use Complex

Complex math is used if this option is selected.

NaN

The value that represents any unknown or undefined value. A value of zero means that this feature is turned off.

Hardware

The displays of all available GPUs. One can select which GPU one wishes.

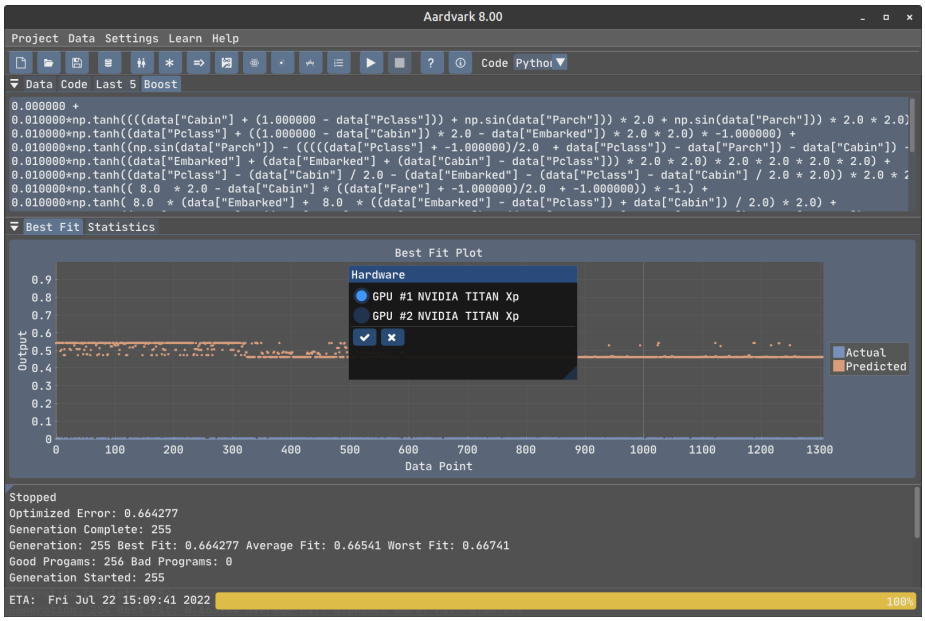

Figure 6: Hardware Dialog

Dummy Values

Although Antbear provided a significant number of constants in the functions settings one can add more simply by using random values created here or by editing these dummy values if one wishes to have specific numbers.

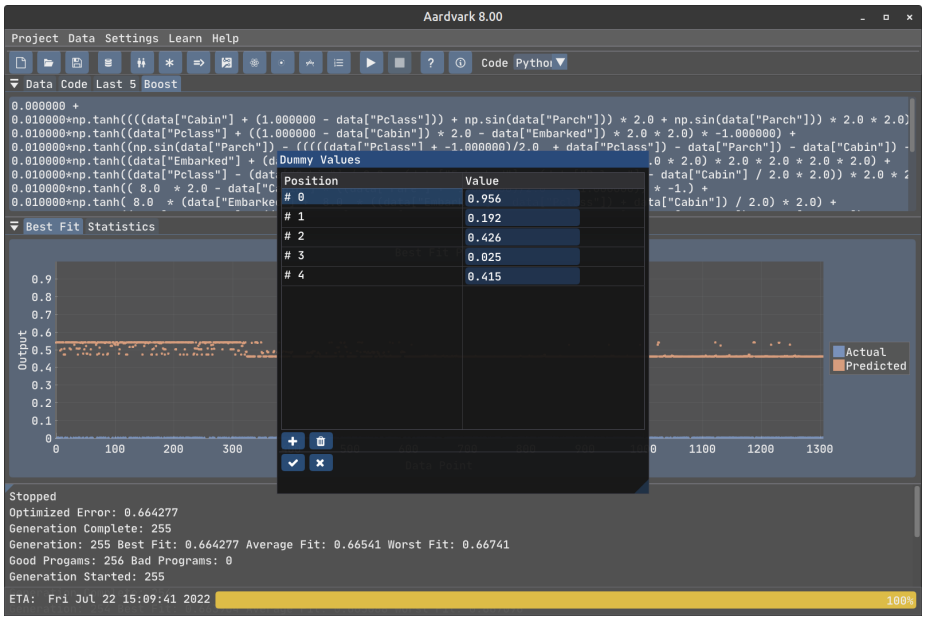

Figure 7: Dummies Dialog

Features

For wide datasets in terms of rows $\ll$ columns it may be worth turning on features which will train on a subset of parameters based on their Pearson correlation or mutual information (MI) with the target.

Pearson

All parameters have their pearson correlations calculated and the best parameters are selected.

MI

All parameters have their mutual information (MI) calculated and the best parameters are selected.

Bins

In order to speed up performance MI will put the data for each parameter in the specified number of bins prior to performing the calculation.
Note for massive data sets this can be both slow and unnecessary

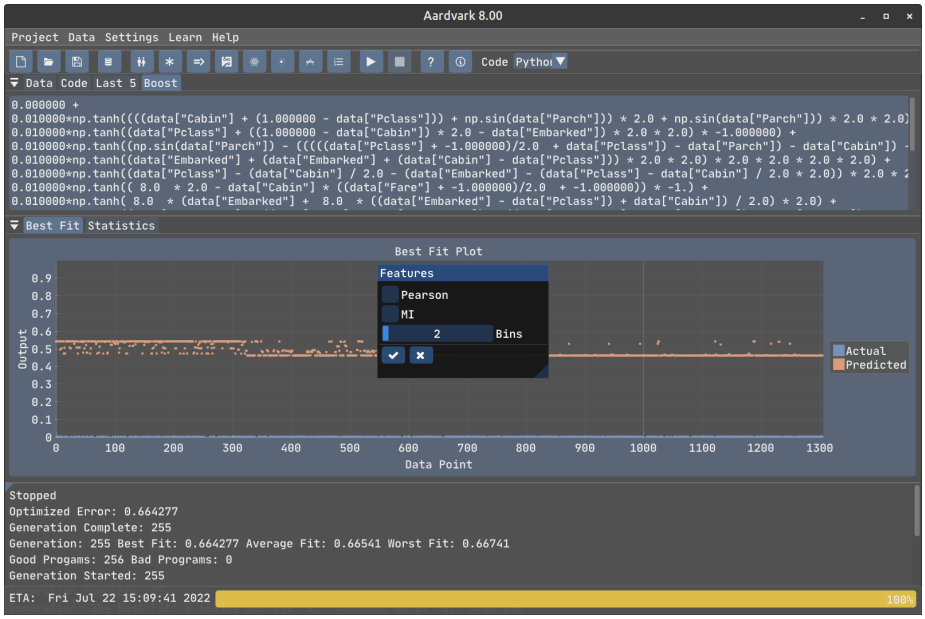

Figure 8: Features Dialog

Tutorials

It is highly recommended that one studies and plays with these examples in order to get a feel for Antbear's capabilities and performance. *Note all tutorial data and scripts can be found in /usr/share/antbear/tutorials*
Originals and many more besides can be found at the IC Irving ML Repository

Titanic

The Titanic data shows various parameters associated with passengers. The idea here is to predict who survived and who dies and then view the code to see what parameters are useful and which ones are irrelevant.

1. Create a new project
2. Import the Titanic Data
   - The Data View should be populated with PClass, Sex etc.
3. Change the Metric to LogLoss
4. Press Start
5. Compare the boost python code with that found in the tutorials directory
6. Reduce the learning rate to 0.1 and repeat. Does it improve performance?
7. Experiment!

Iris Clustering

The Iris flower data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. The idea here is to produce a 2 dimensional clustering program to display iris information.

1. Create a new project
2. Import the Iris Data
   - The Data View should be populated with Sepal_Length, Petal_width etc.
3. Leave the Metric to Least Squares but change Hidden to 2.
4. Press Start
5. Compare the boost python code with that found in the tutorials directory
6. Reduce the learning rate to 0.1 and repeat. Does it improve performance?
7. Experiment!

Wine

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

1. Create a new project
2. Import the Wine Data
    - The Data View should be populated with fixed acidity,volatile acidity etc. etc.
3. Leave the Metric to Least Squares.
4. Press Start
5. Compare the boost python code with that found in the tutorials directory
6. Reduce the learning rate to 0.1 and repeat. Does it improve performance?
7. Experiment!

End User License Agreement (EULA)

Non-commercial use end-user license agreement for Antbear 1.00

By installing, copying or otherwise using Antbear 1.00, you agree to be bound by the terms of the Non-Commercial-Use End User License Agreement (EULA). If you do not agree with the terms of this EULA, you may not use or copy the software product.

NON-COMMERCIAL USE

This EULA intends to make it easy for developers and end users to use Antbear 1.00 for Non-Commercial Purposes, where "Non-Commercial Purposes" means: To evaluate Antbear 1.00 and to do exploratory or educational development and "proof of concept" prototyping of software applications, whether at home for personal use or at work as a prototyping tool, and where "Non-Commercial Purposes" specifically excludes development of a system to be used for commercial gain, whether to be sold or to be used within a company, partnership, organization or entity that transacts commercial business.

GRANT OF LICENSE

This EULA grants you the following non-exclusive rights: Software: You may use the Antbear 1.00 on any number of computers, either standalone, or on a network, so long as every use of the Antbear 1.00 is for NON-COMMERCIAL USE. Unless expressly permitted under this EULA, you will not: Alter, remove, hide or cover proprietary notices in or on Antbear 1.00. Decompile, disassemble or otherwise attempt or assist others to reverse engineer Antbear 1.00. Use the Antbear 1.00 in any application that is intended to create or could, in the event of malfunction or failure, cause serious personal injury or property damage. Make use of the Antbear 1.00 for commercial gain, whether directly, indirectly or incidentally.

LIABILITY

This software is provided "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. in no event shall karl yoxall or kryoxall Ltd. be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

3RD PARTY LICENCES

Links to all 3rd party licenses are shown below:
Dear IMGUI
ImPlot
ImGuiFileDialog
NerdFonts