Gyroid
DATA ANALYTICS FOR SOCIAL GOOD

MACHINE LEARNING MODELER
SOFTWARE MANUAL

CONTENTS

1 LOAD DATA TAB

Load SansEC experiment files, label the data, select features and frequencies for analysis, and create the training data set for input into machine learning models.

1.1 Training Files Section

This section occupies the left half of the screen. When beginning a new project or adding new data, this is where the user will start.

1.1.1 Name Field

Enter the experiment name in this field. The directory containing all summary information, models, and configuration file for the analysis will be saved within a directory with the same name as the experiment name.

1.1.2 Load SansEC Files

Use this button to select a set of SansEC data files from disk. Files will appear in the field below. Use the checkboxes to the left of the filenames to select from which files to extract data.

1.1.3 Load SansEC Directory

Use this button to select a directory of SansEC data files. Similar to **Load SansEC Files**, the files will appear in the field below. Use the checkboxes to the left of the filenames to select from which files to extract data.

1.1.4 Label Data

There are two methods for labeling data: (1) Label by filename, and (2) Label by .csv file. For the former, the labels are parsed from the base filename such that any number after an underscore is converted to a label. For example, if the base filename is ‘experiment1_3.xlsx’ then the label would be 3. For the latter, the files are labeled by ingesting a .csv file

1.1.5 Ingest Files

Click this button to display the data contained in selected files in the Features/Columns field. From here, the user can select which features will be added to the dataset for analysis.

1.1.6 Create Dataset

This button isolates the data for which machine learning will be performed in the **Train Model** Tab.

1.2 Plot Preview Section

Selecting data in the Features/Columns field under will display selected features in the plot window. To change default scaling, adjust the slider bars below the plot and click **Refresh Plot**. The plot window is interactive, and allows the user to pan, zoom, and save images.

1.2.1 Refresh Plot

After adjusting the frequency sliders, click this button to update the plot window.

1.3 Save Configuration File

Once the dataset has been created, click this button to save the project configuration. The configuration file also contains information about the learning models and parameters used in the **Train Model** Tab. The configuration file can also be saved by selecting File -> Save Configuration File along the top menu bar.

1.3.1 Loading Configuration Files

To load a configuration file, select File -> Load Configuration File along the top menu bar.

2 TRAIN MODEL TAB

Select machine learning models for training, set hyperparameters, and customize pipeline specifications.

2.1 Machine Learning Models

Currently, the software supports training for eight base machine learning models, with varying degrees of hyperparameter customization.

2.1.1 Extra Trees

The Extra-trees method is an extremely randomized decision tree classifier. Decision trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualized.
- Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. Other techniques are usually specialized in analyzing datasets that have only one type of variable. See algorithms for more information.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by Boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

2.1.1.1 Number of Trees

A very small number will usually cause the tree to overfit, whereas a large number will prevent the tree from learning the data.

2.1.1.2 Maximum Features (p)

Select All, the square root of p, or the log of p, where p is the number of data points.

2.1.1.3 Criterion

The Gini index and entropy are metrics used to evaluate the classification utility of the algorithm.

2.1.2 Gaussian Process

Gaussian Processes (GPs) are a generic supervised learning method designed to solve regression and probabilistic classification problems.

The advantages of Gaussian processes are:

- The prediction interpolates the observations (at least for regular kernels).
- The prediction is probabilistic (Gaussian) so that one can compute empirical confidence intervals and decide based on those if one should refit (online fitting, adaptive fitting) the prediction in some region of interest.
- Versatile: different kernels can be specified. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of Gaussian processes include:

- They are not sparse, i.e., they use the whole samples/features information to perform the prediction.
- They lose efficiency in high dimensional spaces – namely when the number of features exceeds a few dozens.

2.1.2.1 Kernel

Kernels (also called “covariance functions” in the context of GPs) are a crucial ingredient of GPs which determine the shape of prior and posterior of the GP. They encode the assumptions on the function being learned by defining the “similarity” of two datapoints combined with the assumption that similar datapoints should have similar target values. Two categories of kernels can be distinguished: stationary kernels depend only on the distance of two datapoints and not on their absolute values and are thus invariant to translations in the input space, while non-stationary kernels depend also on the specific values of the datapoints. Stationary kernels can further be subdivided into isotropic and anisotropic kernels, where isotropic kernels are also invariant to rotations in the input space.

2.1.2.1.1 Radial Basis Function

The Radial-Bias Function (RBF) kernel is a stationary kernel. It is also known as the “squared exponential” kernel. This kernel is infinitely differentiable, which implies that GPs with this kernel as covariance function have mean square derivatives of all orders, and are thus very smooth.

2.1.2.1.2 Dot Product

The Dot Product kernel is non-stationary and invariant to a rotation of coordinates about the origin, but not translations.

2.1.2.1.3 Matérn

The Matérn kernel is a stationary kernel and a generalization of the RBF kernel. It has an additional parameter v which controls the smoothness of the resulting function.

2.1.2.1.4 Rational Quadratic

The Rational Quadratic kernel can be seen as a scale mixture (an infinite sum) of RBF kernels with different characteristic length-scales.

2.1.2.1.5 Exp-Sine Squared

The Exp-Sine Squared kernel allows modeling of periodic functions.

2.1.2.1.6 Constant

The Constant kernel is fixed and invariant.

2.1.3 Gradient-Boosted Trees

When used with Decision Trees, Gradient Boosting (GB) builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function. Gradient boosting is fairly robust to over-fitting.

2.1.3.1 Number of Trees

A very small number will usually cause the tree to overfit, whereas a large number will prevent the tree from learning the data.

2.1.3.2 Learning Rate

Learning rate shrinks the contribution of each tree by a given value. There is a trade-off between the learning rate and the number of boosting stages performed.

2.1.3.3 Row Subsampling

The fraction of samples to be used for fitting the individual base learners. If smaller than 1.0 this results in Stochastic Gradient Boosting. Choosing subsample < 1.0 leads to a reduction of variance and an increase in bias.

2.1.3.4 Maximum Tree Depth

Maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. Tune this parameter for best performance; the best value depends on the interaction of the input variables.

2.1.3.5 Loss Function

Loss function to be optimized. Deviation is a highly robust loss function solely based on order information of the input variables.

2.1.4 K-Nearest Neighbors

The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice. Neighbors-based methods are known as non-generalizing machine learning methods.
Despite its simplicity, nearest neighbors has been successful in a large number of classification and regression problems, including handwritten digits or satellite image scenes. Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular.

2.1.4.1 Number of Neighbors

The number of neighbors within a fixed radius r of each training point, where r is specified by the user.

2.1.4.2 Power Parameter

The power parameter specifies the Minkowski metric exponent, used to define the distance between two points.

2.1.5 Linear Model

The linear model will perform straightforward linear fitting to the data.

2.1.6 Neural Network

Neural Networks are different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. The advantages of Neural Networks are:

- The capability to learn non-linear models.

The disadvantages of Multi-layer Perceptron (MLP) include:

- MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy.
- MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
- MLP is sensitive to feature scaling.
-

2.1.6.1 Number of Neurons

Specifies the total number of neurons, including those in hidden layers.

2.1.6.2 Learning Rate

The learning rate controls the step size in parameter space while the algorithm is converging toward a solution.

2.1.6.2.1 Constant

Learning rate does not adjust.

2.1.6.2.2 Adaptive

The learning rate is decreased on-the-fly whenever the training loss fails to decrease.

2.1.6.2.3 Inverse Scaling

The learning rate decays with increasing iterations.

2.1.6.3 Weight Optimizer

The weight optimizer is the solver for the weight value of each neuron.

2.1.6.3.1 Adam

"Adam" refers to a particular stochastic gradient-based optimizer. Adam works well on relatively large datasets (with thousands of training samples or more) for both training time and validation score.

2.1.6.3.2 LBFGS

Limited-memory BFGS (LBFGS) is an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm using a limited amount of computer memory. For small datasets, LBFGS can converge faster and perform better than Adam.

2.1.6.3.3 SGD

Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention recently in the context of large-scale learning.

SGD has been can be applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.

The advantages of Stochastic Gradient Descent are:

- Efficiency.
- Ease of implementation (lots of opportunities for code tuning).

The disadvantages of Stochastic Gradient Descent include:

- SGD requires a number of hyperparameters such as the regularization parameter and the number of iterations.
- SGD is sensitive to feature scaling.

2.1.7 Random Forest

In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

2.1.7.1 Number of Features

The model will perform better with more features, but will take longer to compute. In addition, results will stop getting significantly better beyond a critical number of trees.

2.1.7.2 Maximum Features (p)

The size of the random subsets of features to consider when splitting a node. The lower the greater the reduction of variance, but also the greater the increase in bias.

2.1.7.3 Criterion

The Gini index and entropy are metrics used to evaluate the classification utility of the algorithm.

2.1.8 Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates.

2.1.8.1 Kernel

The Kernel of an SVM determines how the boundaries will be drawn between classification sets. Radial Basis Functions (RBFs) can be localized to finite sections of the parameter space, while polynomial boundary functions extend into infinity.

2.1.8.2 Polynomial Degree

Specifies the degree of the polynomial used for a polynomial kernel.

2.1.8.3 Regularization

Sets the level of regularization compared to the penalty function for the model parameters. A lower value may cause overfitting, while a higher value may cause underfitting.

2.1.8.4 Gamma

Gamma is an exponentiation factor used with an RBF kernel.

2.2 Pipeline Specifications

Standardize features, perform feature reduction, select training method, and automatically tune hyperparameters.

2.2.1 Standardize Features

Standardizes each column in the training datset to have mean 0 and variance of 1. The scaler transformers are saved from the training set so that the testing dataset can be placed on the same scale as the training dataset. The majority of machine learning models are sensitive to the scaling of data, so by default it is recommended to keep this set to “Yes”.

2.2.2 Feature Reduction

Methods for reducing size of the training and testing feature sets prior to model training. Two methods are currently implemented: (1) Tree-based, and (2) PCA (i.e., principal components analysis). For the former, tree-based feature selection calculates feature importances for each feature based on a random forest model and keeps only features that are above the average feature importance. For the latter, PCA uses an orthogonal transformation to convert the training dataset into a dataset consisting of linearly uncorrelated variables called principle components. The number of principle components is automatically selected such that 80% of the original solution is retained. In the case where the number of selected components equals the original number of features, if p defines the number of features, the number of principle components is set to p – 1 (i.e., to ensure the dimensionality is at least decreased by 1). The PCA transformation is saved so that a similar transformation can be performed on a testing dataset.

2.2.3 Training Method

Methods for training selected machine learning models. Two methods are currently implemented: (1) Holdout, and (2) CV (i.e., cross-validation). For the former, holdout consists of splitting the training dataset into two disjoint sets such that ~ 66% is used for model training and ~33% is used for model validation. After the validation metric is calculated, the model is refit on all the training data and saved for later deployment. For the latter, CV consists of splitting the training dataset into three equal disjoint samples (i.e., 3-fold CV training). Then, 2/3 of the folds are used for model training and the holdout fold is used for model validation and a metric is calculated. The procedure iterates over all permutations and the model metrics are averaged across folds. To help prevent overfitting, it is recommended to use CV, however, this method takes significantly more time to finish than the holdout training method.

2.2.4 Automatically Tune

Automatically identifies best combination of hyperparameters from pre-specified grid (i.e., combinations of hyperparameters to test) based on the selected training method. Although this method automatically tunes hyperparameters, it is significantly more computationally expensive than using only the holdout or CV training methods without automatically tuning.

2.3 Begin Training

Click this button to begin training after selecting the **Machine Learning Models** and **Pipeline Specifications**. The progress and results of the training will be shown in the Analysis Summary field occupying the majority of the screen.

2.4 Clear Log

This button clears the Analysis Summary log.

3 DEPLOY MODEL TAB

Select trained machine learning models, files to create testing data, test trained models, and generate analysis summary report.

3.1 Trained Models

Load directory of trained models and select which ones to deploy on testing dataset.

3.2 Testing Files

Similar to Load Data Tab, load single/multiple files or load directory of SansEC files to create testing dataset. By default, files will attempt to be labeled by filename.

3.3 Generate Report

Generates summary report of analysis for current experiment.

4 ADDITIONAL RESOURCES

For further assistance, please refer to http://scikit-learn.org or contact Gyroid Inc. at contact@gyroid.io.