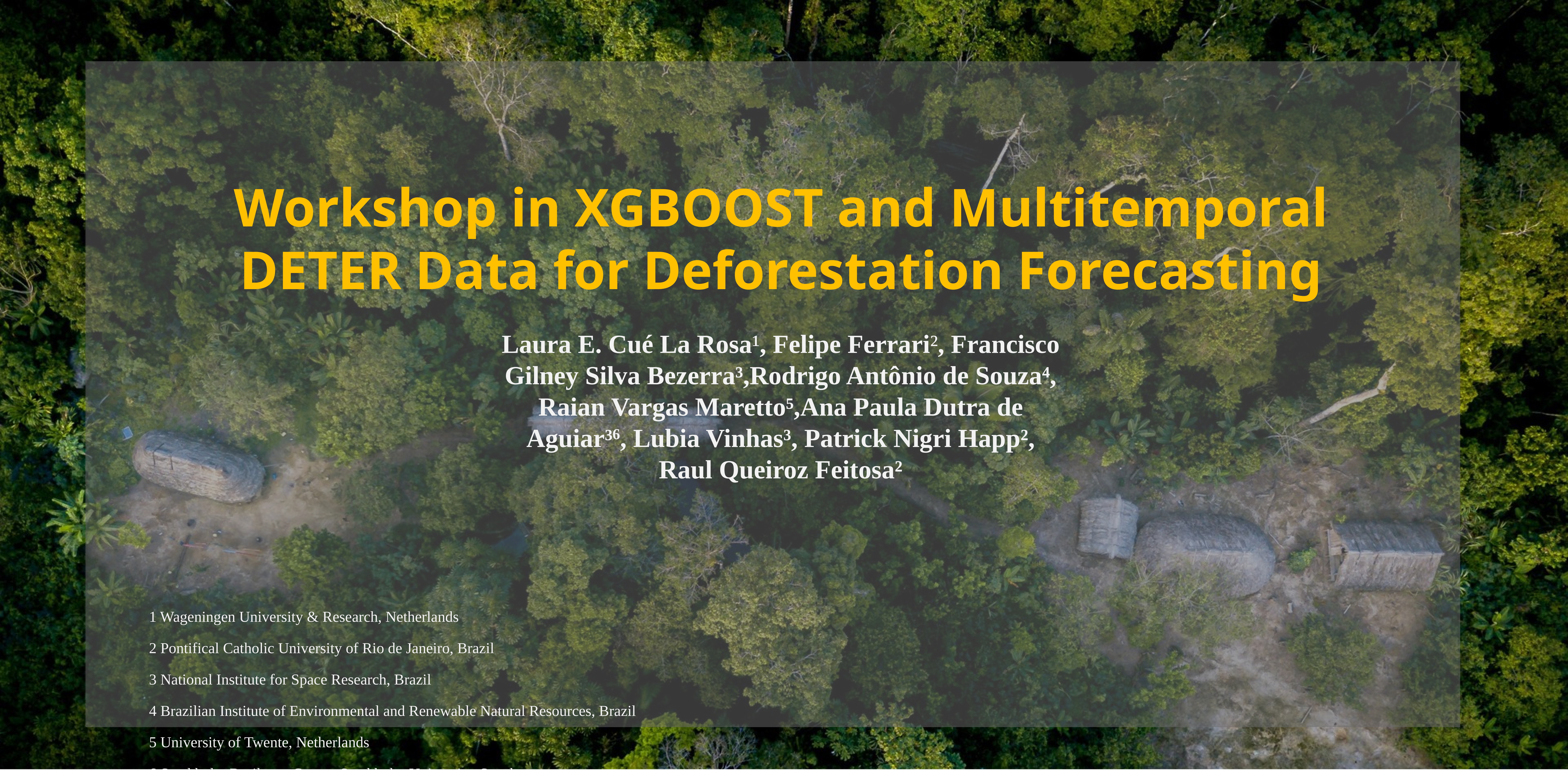


Workshop in XGBOOST and Multitemporal DETER Data for Deforestation Forecasting

Laura E. Cué La Rosa[1], Felipe Ferrari[2], Francisco Gilney Silva Bezerra[3],Rodrigo Antônio de Souza[4], Raian Vargas Maretto[5],Ana Paula Dutra de Aguiar[36], Lubia Vinhas[3], Patrick Nigri Happ[2], Raul Queiroz Feitosa[2]

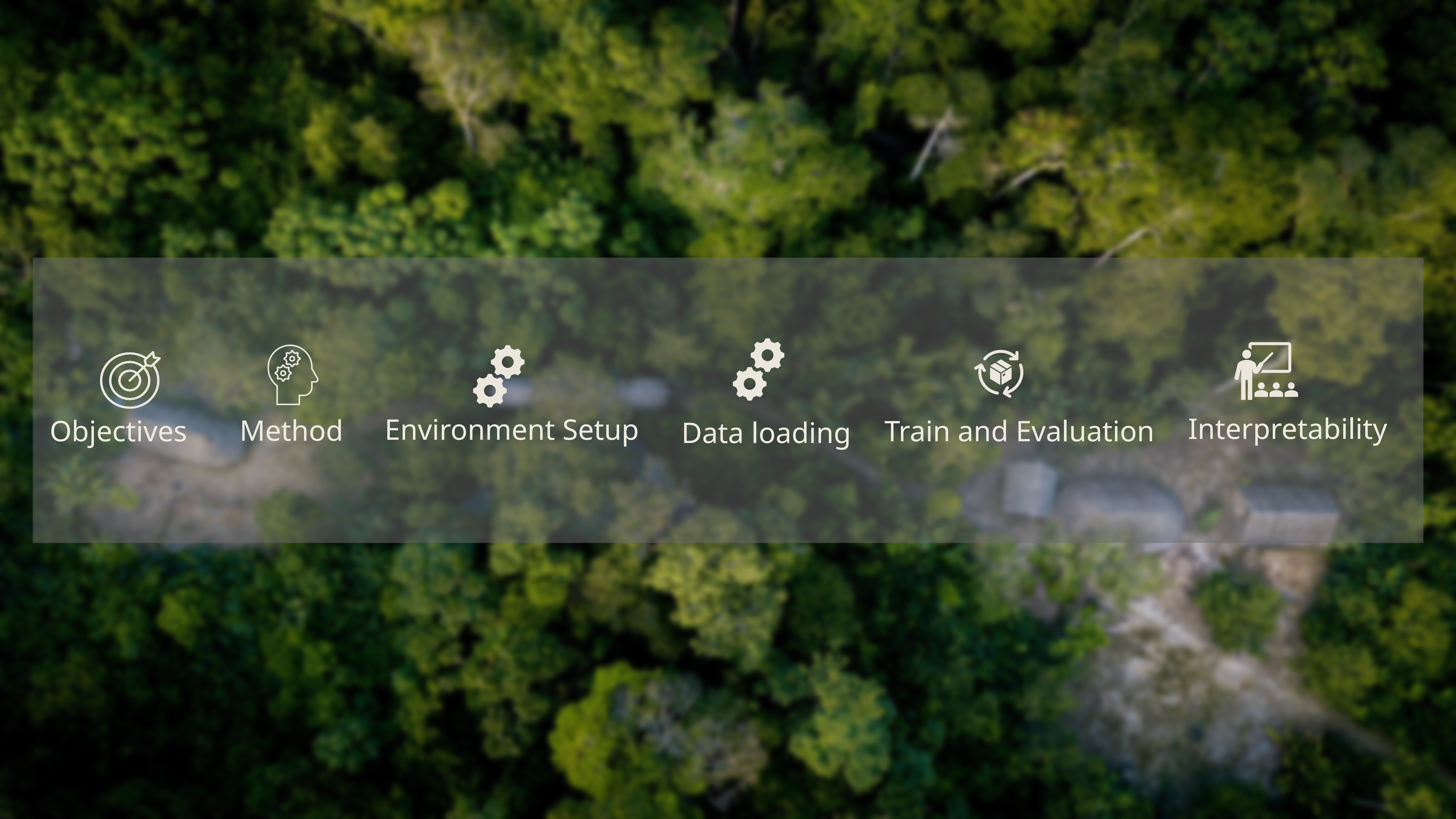
1 Wageningen University & Research, Netherlands

2 Pontifical Catholic University of Rio de Janeiro, Brazil

3 National Institute for Space Research, Brazil

4 Brazilian Institute of Environmental and Renewable Natural Resources, Brazil

5 University of Twente, Netherlands



Objectives

Method

Environment Setup

Data loading

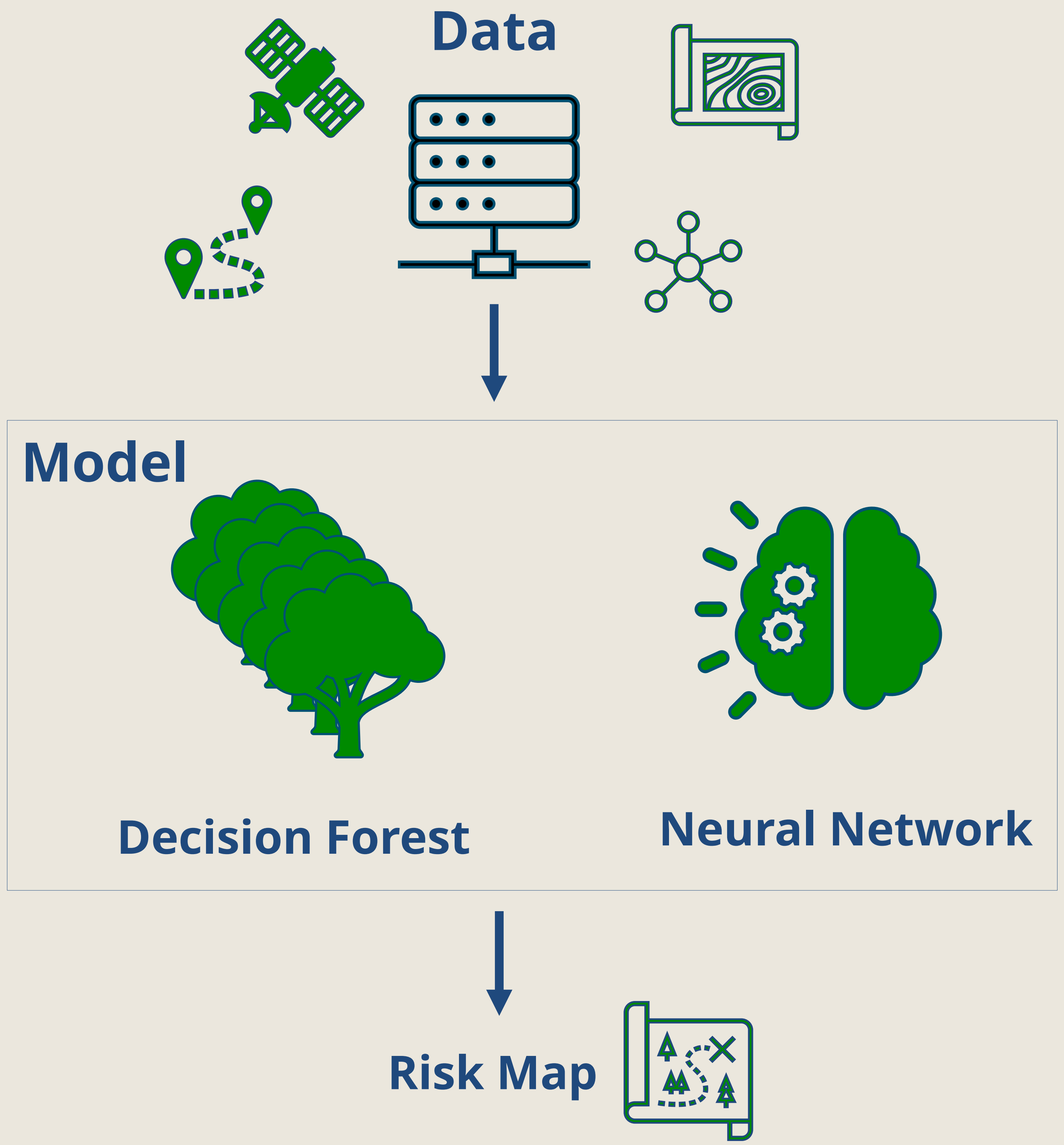
Train and Evaluation

Interpretability

Project scope: Produce biweekly deforestation risk maps incorporating data on forest degradation and deforestation from the DETER System and a wide range of socio-political and economic factors.

Main Objectives of the Workshop

- Understand and Set Up the Technical Environment
- Explore the Code Pipeline for Spatiotemporal Modeling
- Model train and evaluation
- Inference and Generate Predictive Maps
- Enable Operationalization and Reproducibility


Data
Model
Decision Forest
Neural Network
Risk Map

Objectives Method Environment Setup Data loading Train and Evaluation Interpretability

Data

Model

Decision Forest

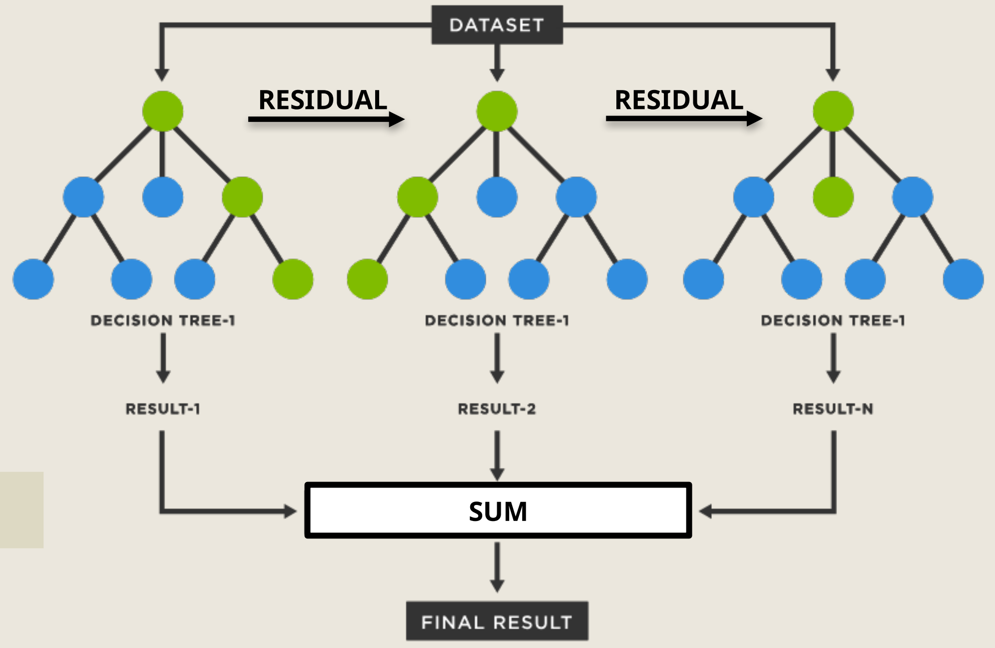
Neural Network

Risk Map

XGBoost

high performance on sparse data

higher interpretability



requires less data for accurate results

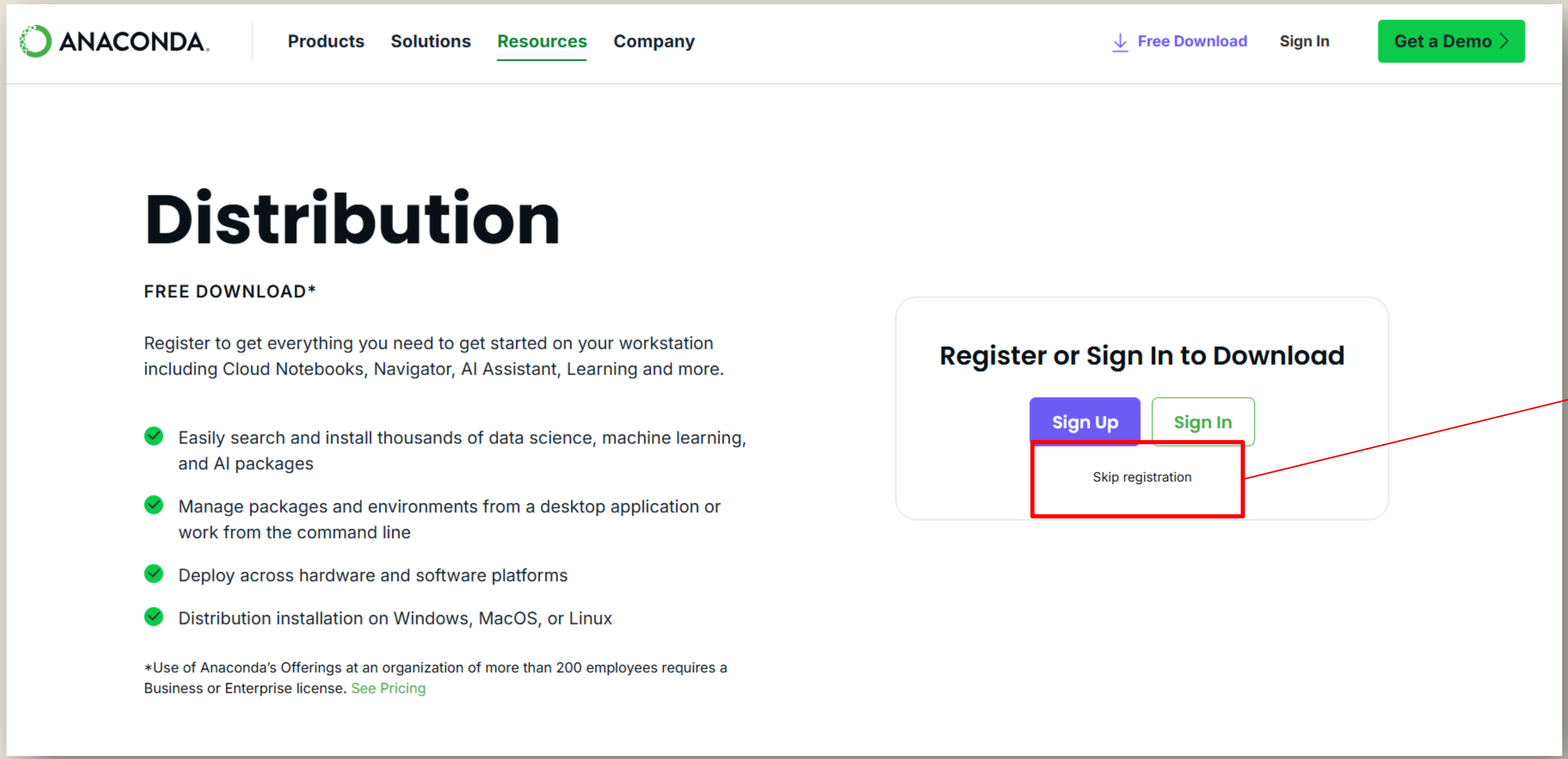
requires more modest computational resources

XGBoost builds an ensemble of decision trees rooted in the gradient-boosting paradigm

Step-by-Step: Conda + Jupyter + Packages

🧱 1. **Install Required Tools (if not already installed)**

- Download Conda from: https://www.anaconda.com/download



ANACONDA. Products Solutions Resources Company Free Download Sign In Get a Demo

Distribution

FREE DOWNLOAD*

Register to get everything you need to get started on your workstation including Cloud Notebooks, Navigator, AI Assistant, Learning and more.

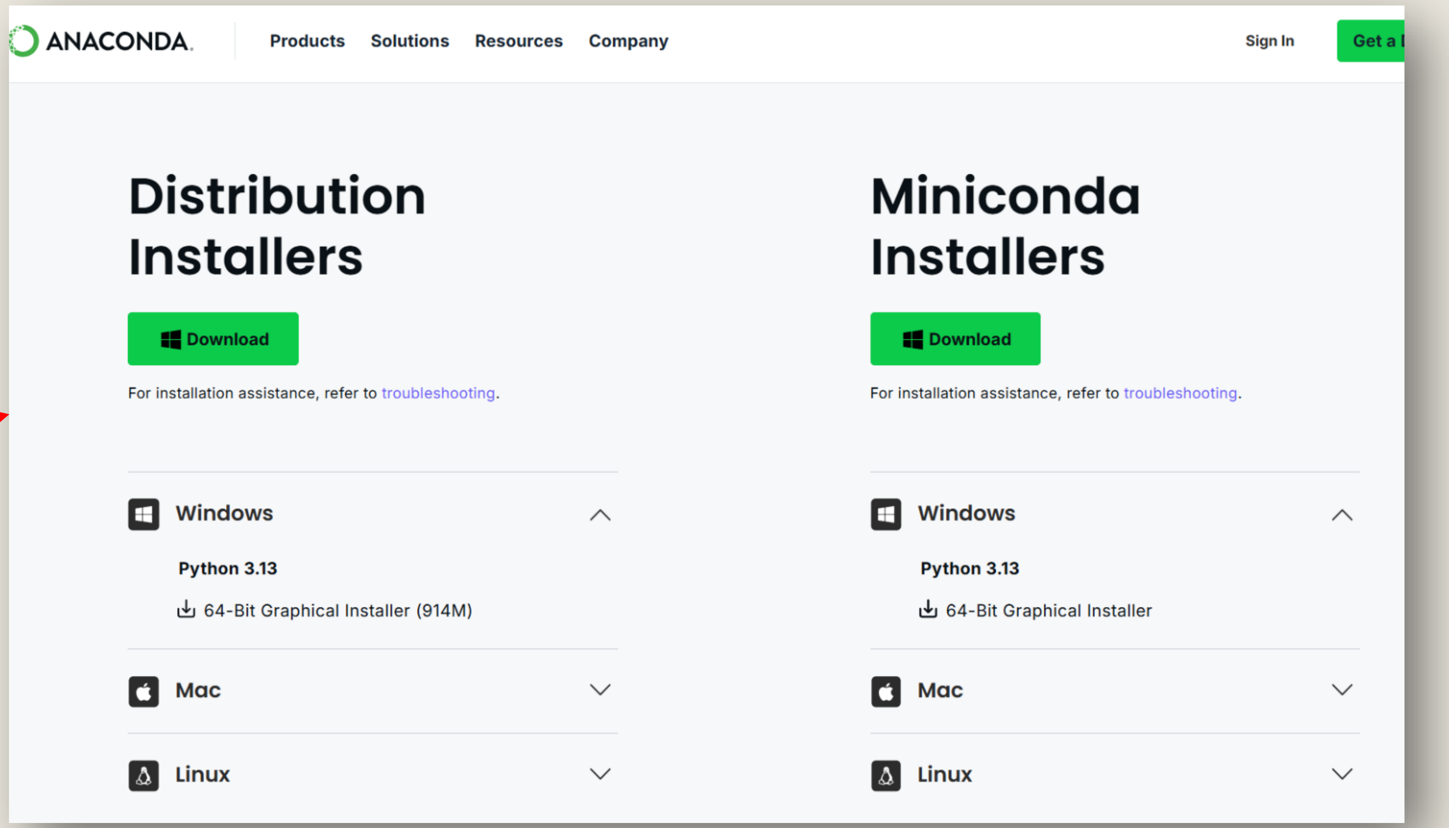
- Easily search and install thousands of data science, machine learning, and AI packages
- Manage packages and environments from a desktop application or work from the command line
- Deploy across hardware and software platforms
- Distribution installation on Windows, MacOS, or Linux

*Use of Anaconda's Offerings at an organization of more than 200 employees requires a Business or Enterprise license. See Pricing

Register or Sign In to Download

Sign Up Sign In

Skip registration



ANACONDA. Products Solutions Resources Company Sign In Get a

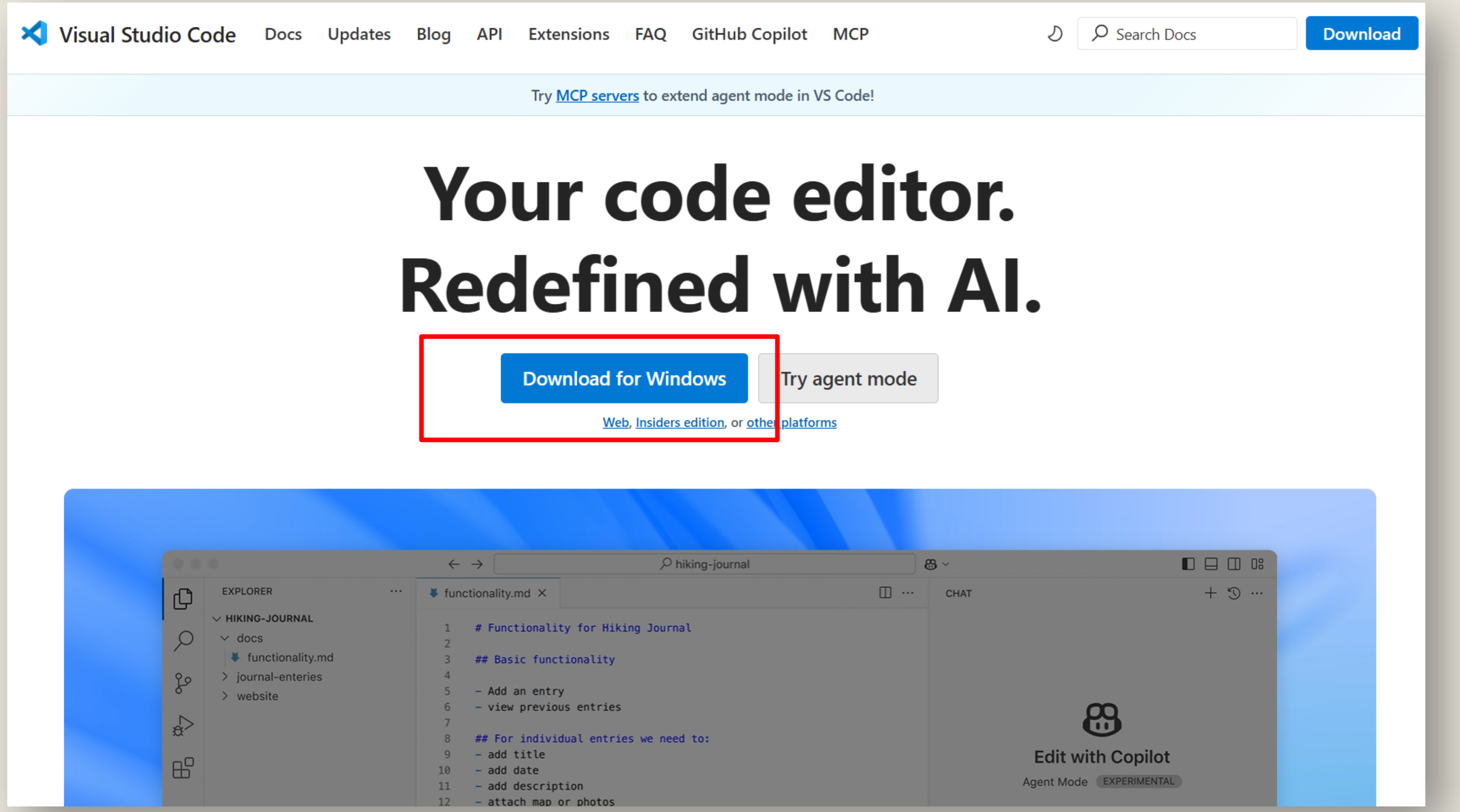
Distribution Installers	Miniconda Installers
Download	Download
For installation assistance, refer to troubleshooting.	For installation assistance, refer to troubleshooting.
Windows	Windows
Python 3.13	Python 3.13
64-Bit Graphical Installer (914M)	64-Bit Graphical Installer
Mac	Mac
Linux	Linux

Step-by-Step: Conda + Jupyter + Packages

🧱 1. **Install Required Tools (if not already installed)**

- Download Conda from: https://www.anaconda.com/download
- Download VS Code from: https://code.visualstudio.com/

Run the installer and follow the default installation steps



Try MCP servers to extend agent mode in VS Code!

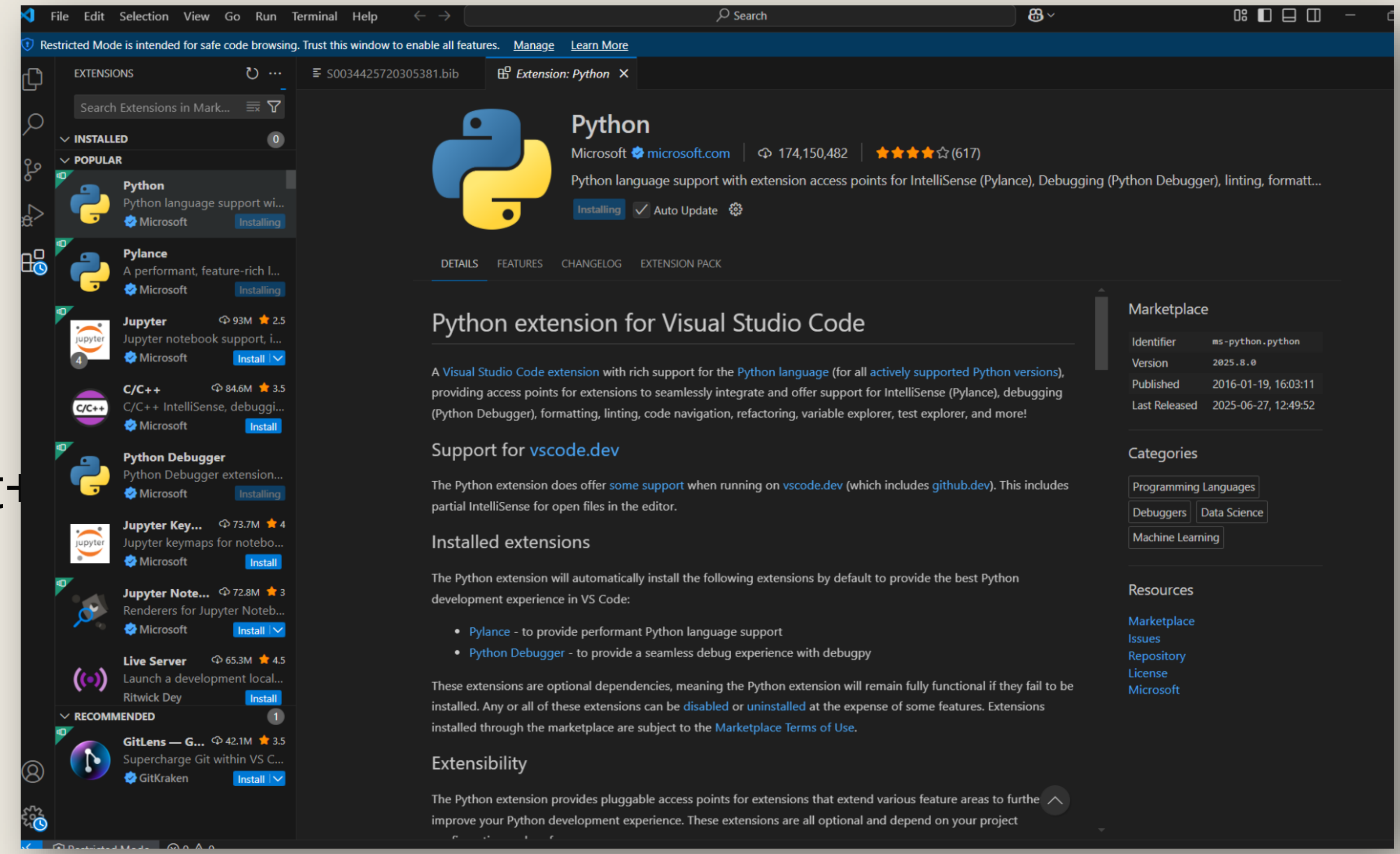
Your code editor. Redefined with AI.

```
# Functionality for Hiking Journal

## Basic functionality

- Add an entry
- view previous entries

## For individual entries we need to:
- add title
- add date
- add description
- attach map or photos
```

Step-by-Step: Conda + Jupyter + Packages

🧱 1. **Install Required Tools (if not already installed)**

- Download Conda from: https://www.anaconda.com/download
- Download VS Code from: https://code.visualstudio.com/
- Install Python Extension
    1. Click the Extensions icon in the Activity Bar (or press Ctrl+Shift+
    2. Search for "Python" (by Microsoft)
    3. Click "Install“
- Install Jupyter Notebook Extension
    1. Click the Extensions icon in the Activity Bar (or press Ctrl+Shift+X)
    2. Search for “Jupyter" (by Microsoft)
    3. Click "Install"


File Edit Selection View Go Run Terminal Help
Search
Manage Learn More
EXTENSIONS
Search Extensions in Mark...
S0034425720305381.bib
Extension: Python
DETAILS FEATURES CHANGELOG EXTENSION PACK
Installing
Auto Update
Install

Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

INSTALLED 0

POPULAR

Python	Python language support wi...	Microsoft	Installing
Pylance	A performant, feature-rich l...	Microsoft	Installing
Jupyter 93M ★ 2.5	Jupyter notebook support, i...	Microsoft	Install
C/C++ 84.6M ★ 3.5	C/C++ IntelliSense, debuggi...	Microsoft	Install
Python Debugger	Python Debugger extension...	Microsoft	Installing
Jupyter Key... 73.7M ★ 4	Jupyter keymaps for notebo...	Microsoft	Install
Jupyter Note... 72.8M ★ 3	Renderers for Jupyter Noteb...	Microsoft	Install
Live Server 65.3M ★ 4.5	Launch a development local...	Ritwick Dey	Install

RECOMMENDED 1

GitLens — G... 42.1M ★ 3.5	Supercharge Git within VS C...	GitKraken	Install

Python

Microsoft microsoft.com | 174,150,482 | ★★★★☆ (617)

Python language support with extension access points for IntelliSense (Pylance), Debugging (Python Debugger), linting, formatt...

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported Python versions), providing access points for extensions to seamlessly integrate and offer support for IntelliSense (Pylance), debugging (Python Debugger), formatting, linting, code navigation, refactoring, variable explorer, test explorer, and more!

Support for vscode.dev

The Python extension does offer some support when running on vscode.dev (which includes github.dev). This includes partial IntelliSense for open files in the editor.

Installed extensions

The Python extension will automatically install the following extensions by default to provide the best Python development experience in VS Code:

- Pylance - to provide performant Python language support
- Python Debugger - to provide a seamless debug experience with debugpy

These extensions are optional dependencies, meaning the Python extension will remain fully functional if they fail to be installed. Any or all of these extensions can be disabled or uninstalled at the expense of some features. Extensions installed through the marketplace are subject to the Marketplace Terms of Use.

Extensibility

The Python extension provides pluggable access points for extensions that extend various feature areas to furthe improve your Python development experience. These extensions are all optional and depend on your project

Marketplace

Identifier	ms-python.python
Version	2025.8.0
Published	2016-01-19, 16:03:11
Last Released	2025-06-27, 12:49:52

Categories

Programming Languages
Debuggers
Data Science
Machine Learning

Resources

Marketplace
Issues
Repository
License
Microsoft

Step-by-Step: Conda + Jupyter + Packages

🧱 1. **Install Required Tools (if not already installed)**

🔧 2. Create and Configure a Conda Environment

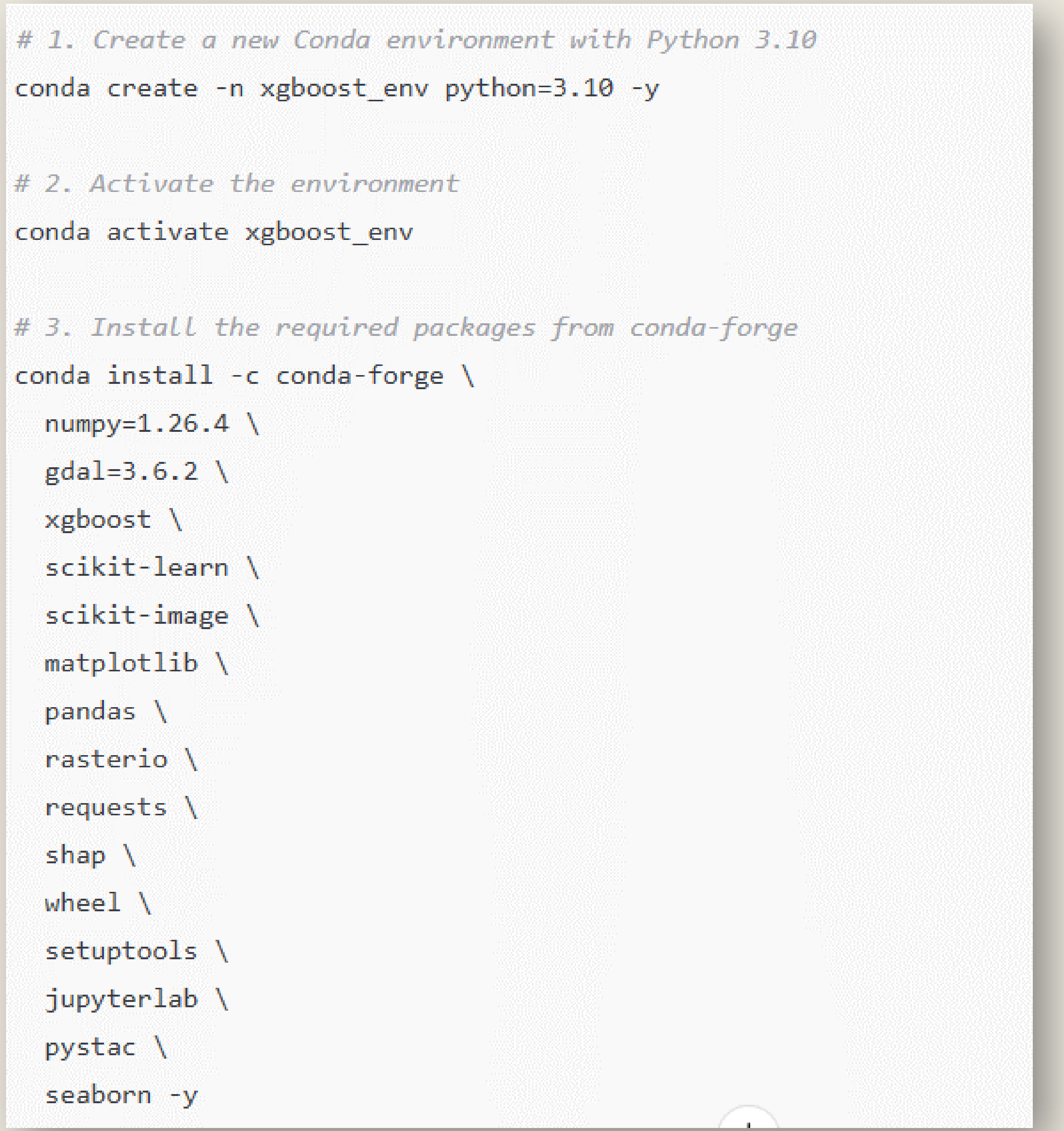
- Open **Anaconda Prompt** or a terminal and type

Step-by-Step: Conda + Jupyter + Packages

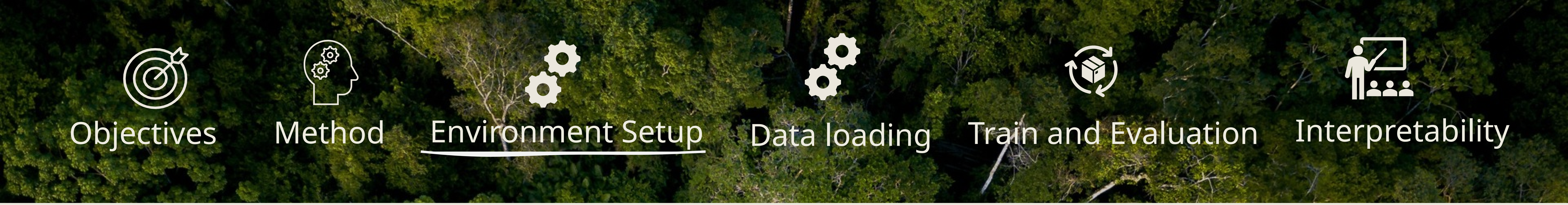
🧱 1. **Install Required Tools (if not already installed)**

🔧 2. Create and Configure a Conda Environment

- Open **Anaconda Prompt** or a terminal and type



```
# 1. Create a new Conda environment with Python 3.10
conda create -n xgboost_env python=3.10 -y

# 2. Activate the environment
conda activate xgboost_env

# 3. Install the required packages from conda-forge
conda install -c conda-forge \
  numpy=1.26.4 \
  gdal=3.6.2 \
  xgboost \
  scikit-learn \
  scikit-image \
  matplotlib \
  pandas \
  rasterio \
  requests \
  shap \
  wheel \
  setuptools \
  jupyterlab \
  pystac \
  seaborn -y
```



Step-by-Step: Conda + Jupyter + Packages

🧱 1. **Install Required Tools (if not already installed)**

🔧 2. Create and Configure a Conda Environment

🧭 3. Open the Project in VS Code

- Launch VS Code
- Open your project folder: File → Open Folder → Select the project directory

Step-by-Step: Conda + Jupyter + Packages

🧱 1. Install Required Tools (if not already installed)

🔧 2. Create and Configure a Conda Environment

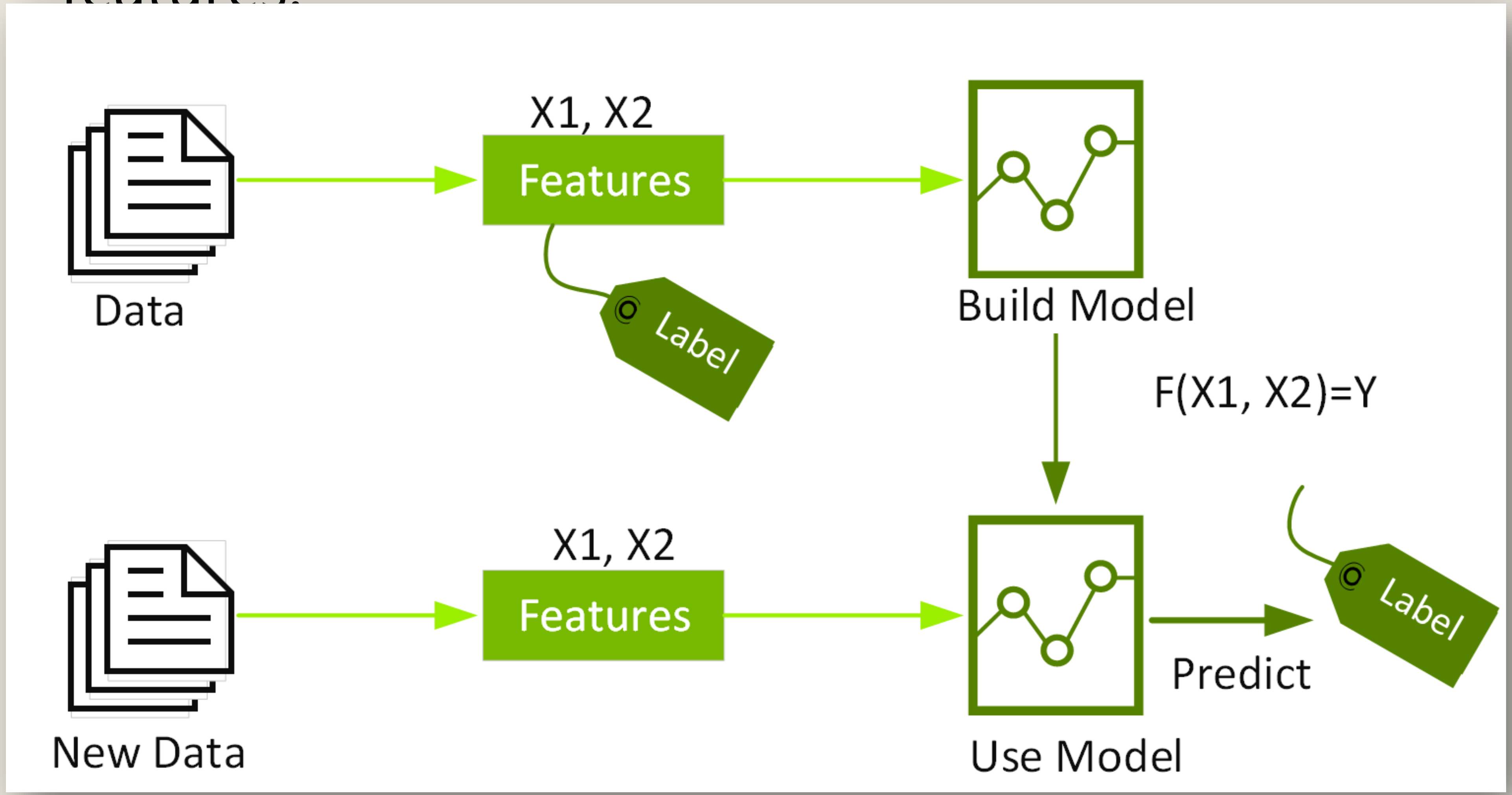
🧭 3. Open the Project in VS Code

⚙️ 4. Set VS Code Python Interpreter

- Press Ctrl+Shift+P
- Type: Python: Select Interpreter
- Select: xgboost_env (conda)

What is XGBoost?

- Supervised machine learning uses algorithms to train a model to find **patterns** in a dataset with **labels** and **features** and then uses the trained model to **predict** the labels on a **new** dataset's features.

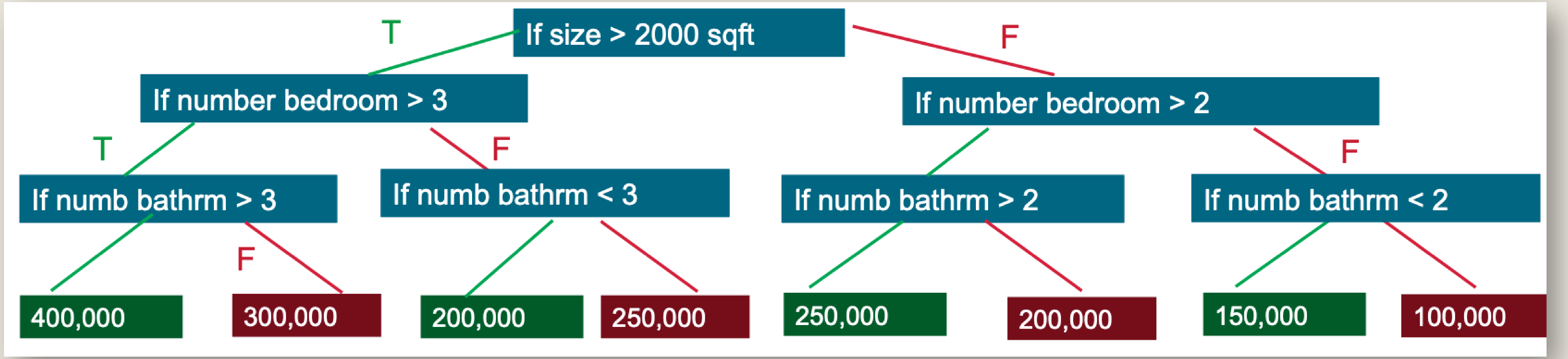


$$\mathrm{obj}(\theta) = L(\theta) + \Omega(\theta)$$

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2$$

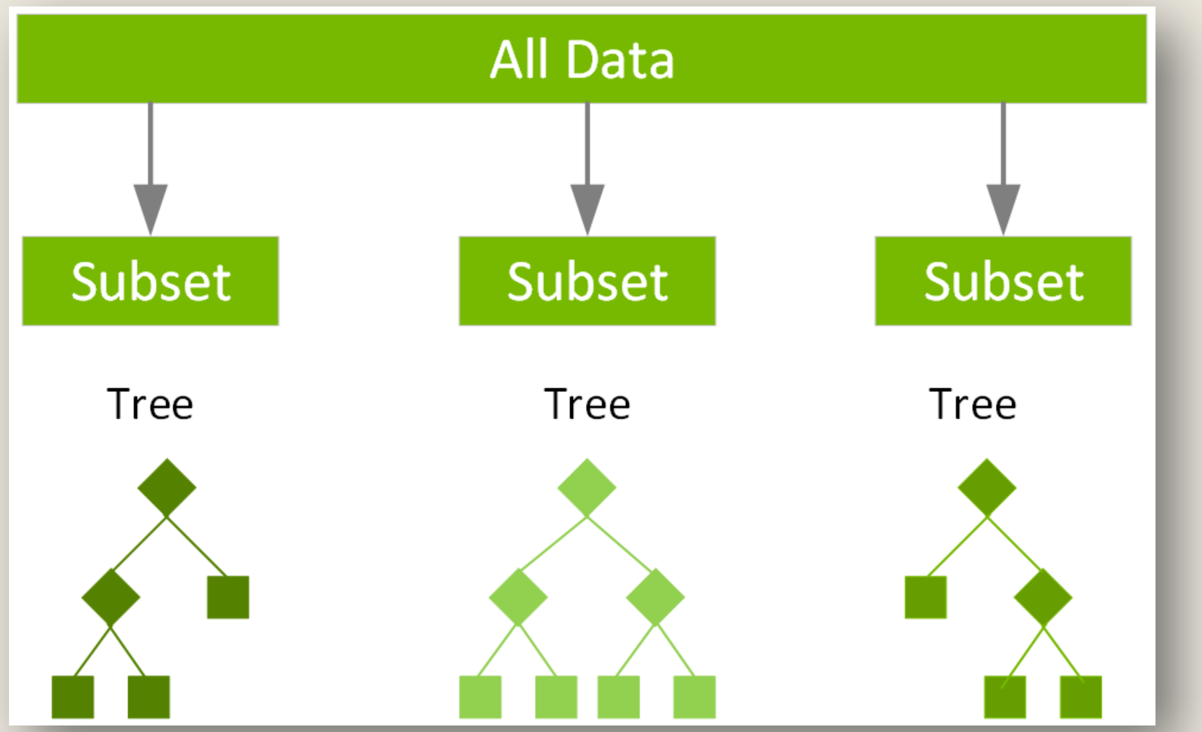
What is XGBoost?

- Predicts the label by evaluating a tree of **if-then-else true/false** feature questions
- Estimates the **minimum** number of questions needed to assess the probability of making a correct decision
- Can be used for **classification** to predict a category, or **regression** to predict a continuous numeric value



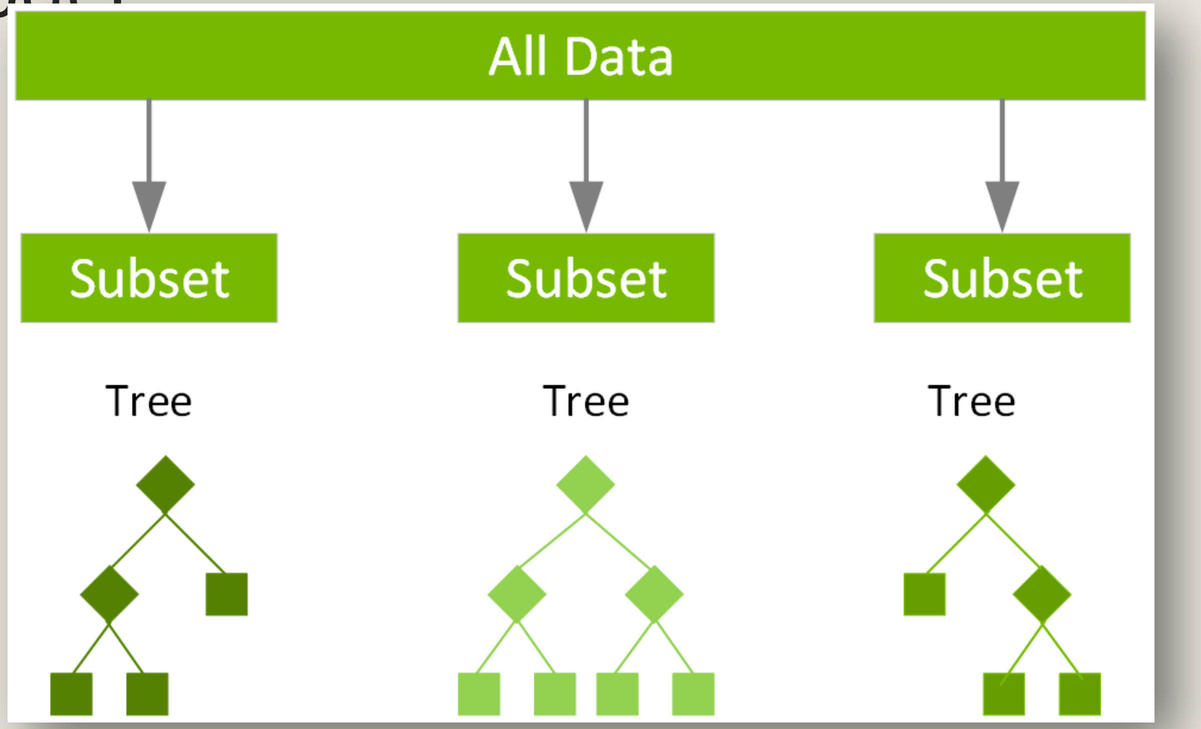
What is XGBoost?

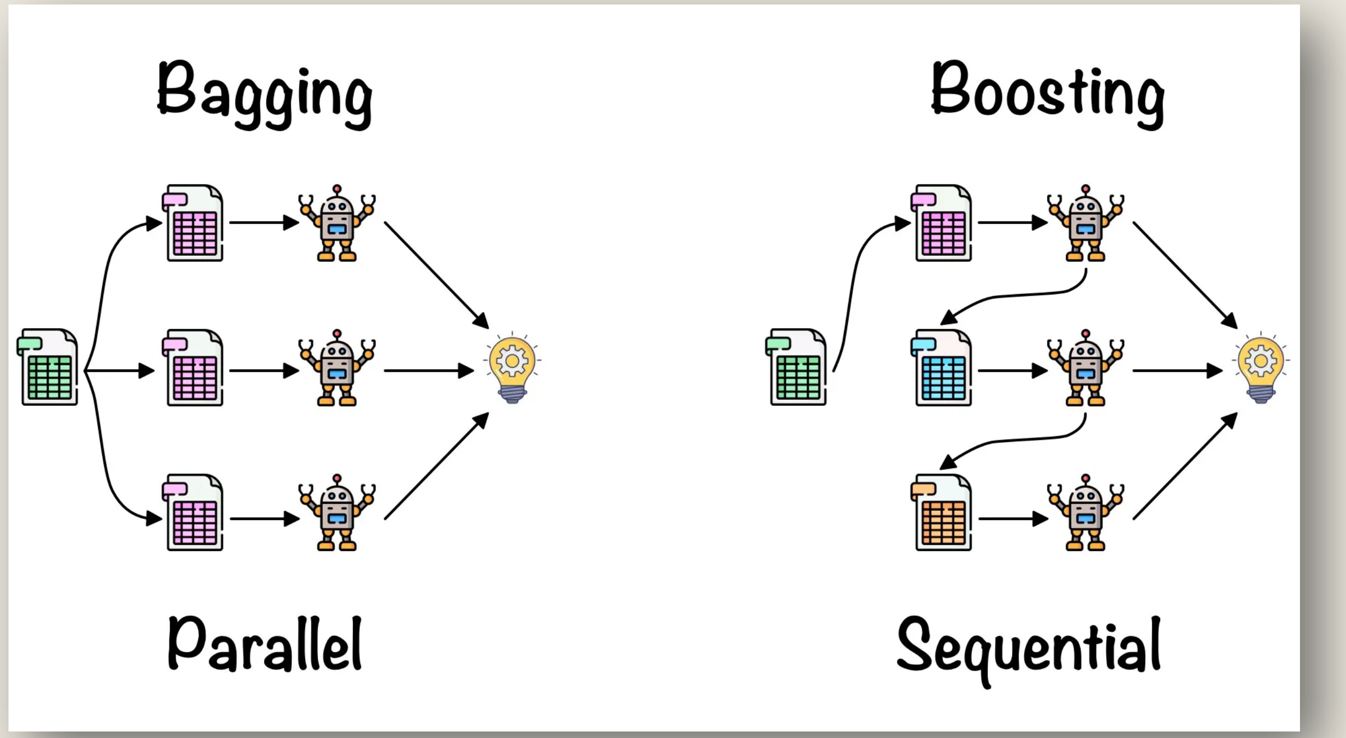
- **Ensembles** -> combine the decisions from multiple models to improve the overall performance.



What is XGBoost?

- **Ensembles** -> combine the decisions from multiple models to improve the overall performance.
- **BAGGING ->** subsetting the data and aggregating the results (e.g, Random Forest)
- **BOOSTING** -> sequential, each subsequent model attempts to correct the errors of the previous model





XGBoost: A BOOSTING Ensemble. Does it really work as the name… | by Rishabh Kesarwani | AlmaBetter | Medium

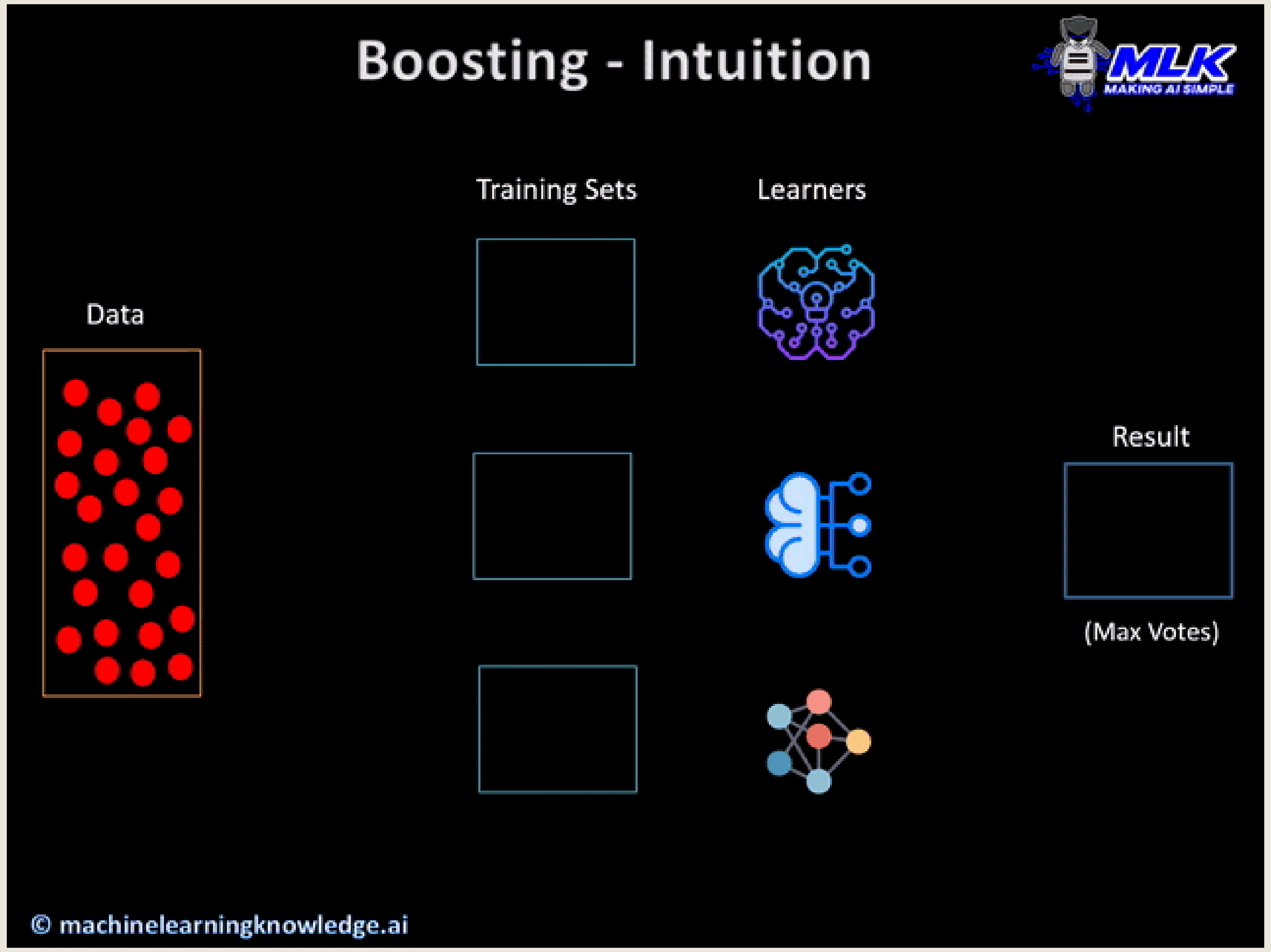
What Is XGBoost and Why Does It Matter? | NVIDIA Glossary

What is XGBoost?

GRADIENT BOOSTING

- Improving a single weak model by combining it with a number of other weak models -> Collectively strong model

- Each decision tree within the committee is specifically trained to address the residual errors propagated by the preceding trees



What is XGBoost?

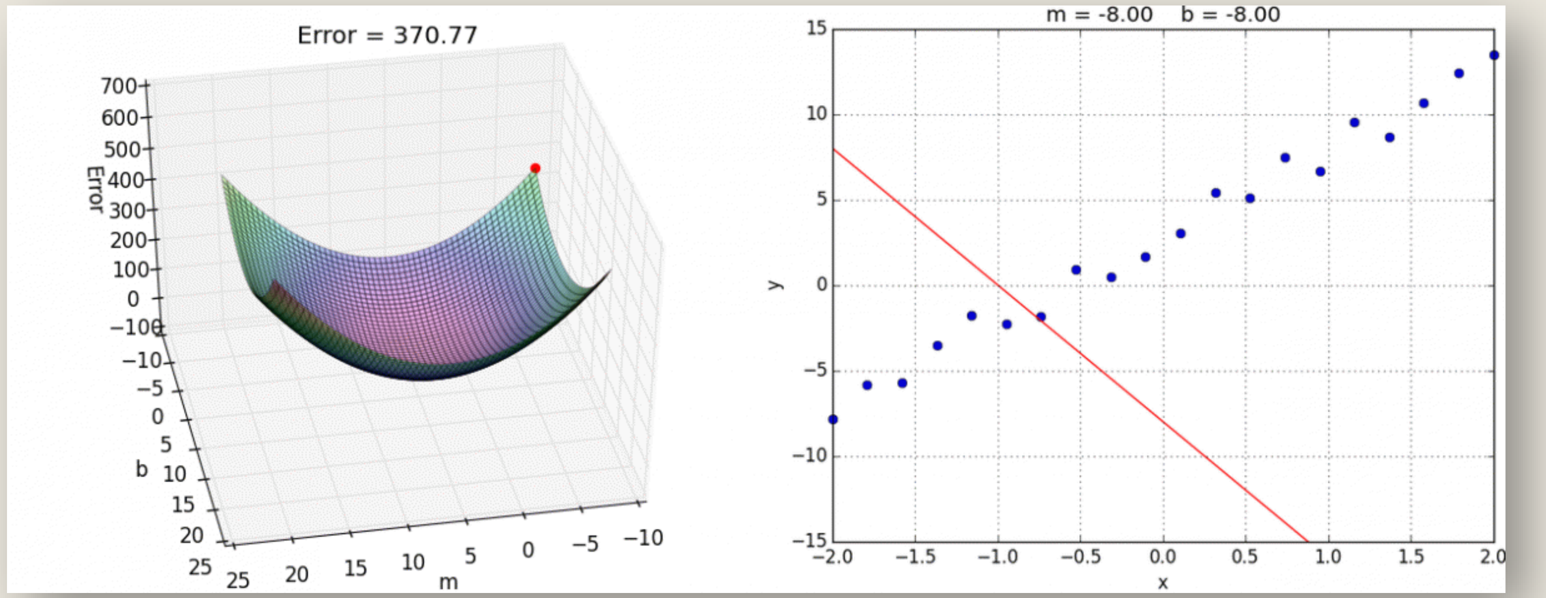
GRADIENT BOOSTING

- Improving a single weak model by combining it with a number of other weak models -> Collectively strong model

- Each decision tree within the committee is specifically trained to address the residual errors propagated by the preceding trees

- The gradient of the loss function guides this correction

- The training involves assigning weights to each tree based on their contribution

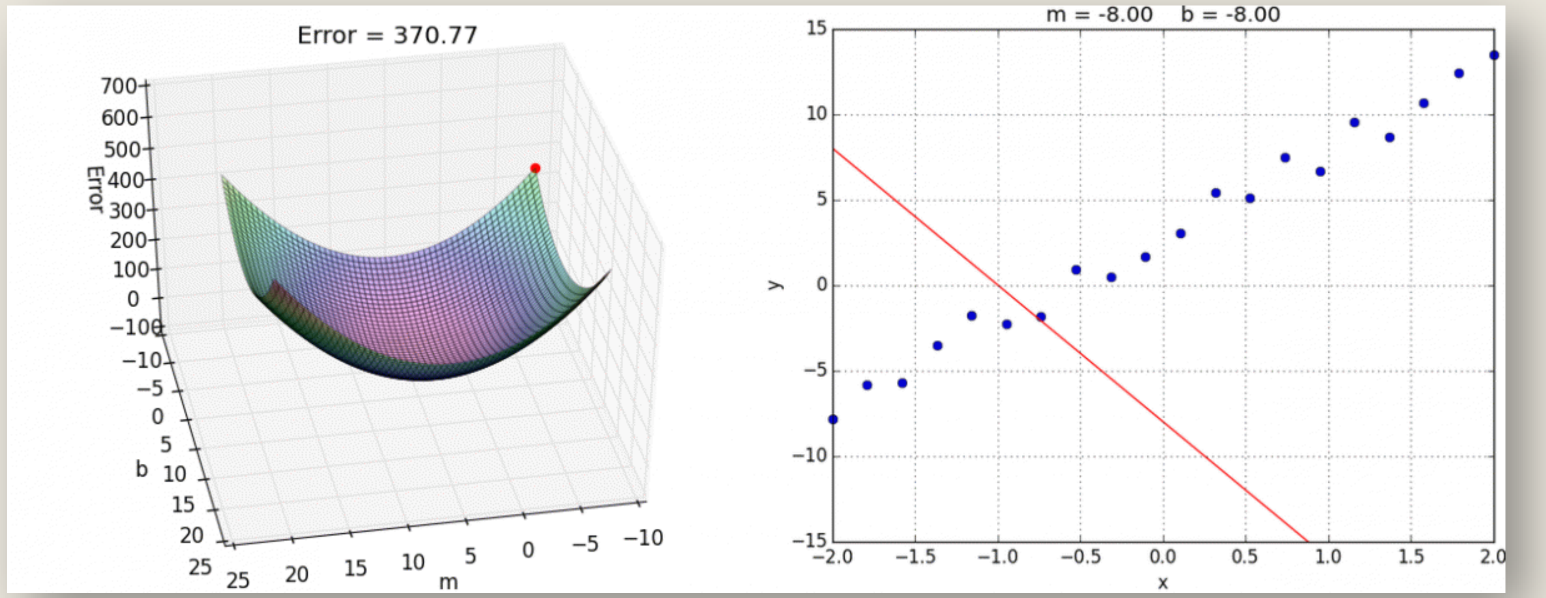


What Is XGBoost and Why Does It Matter? | NVIDIA Glossary

What is XGBoost?

GRADIENT BOOSTING

- In order to minimize the loss function we take the derivative of the loss function and add the negative of this value to the weight scaled by some learning rate to achieve a more accurate model this is process is repeated until convergence.

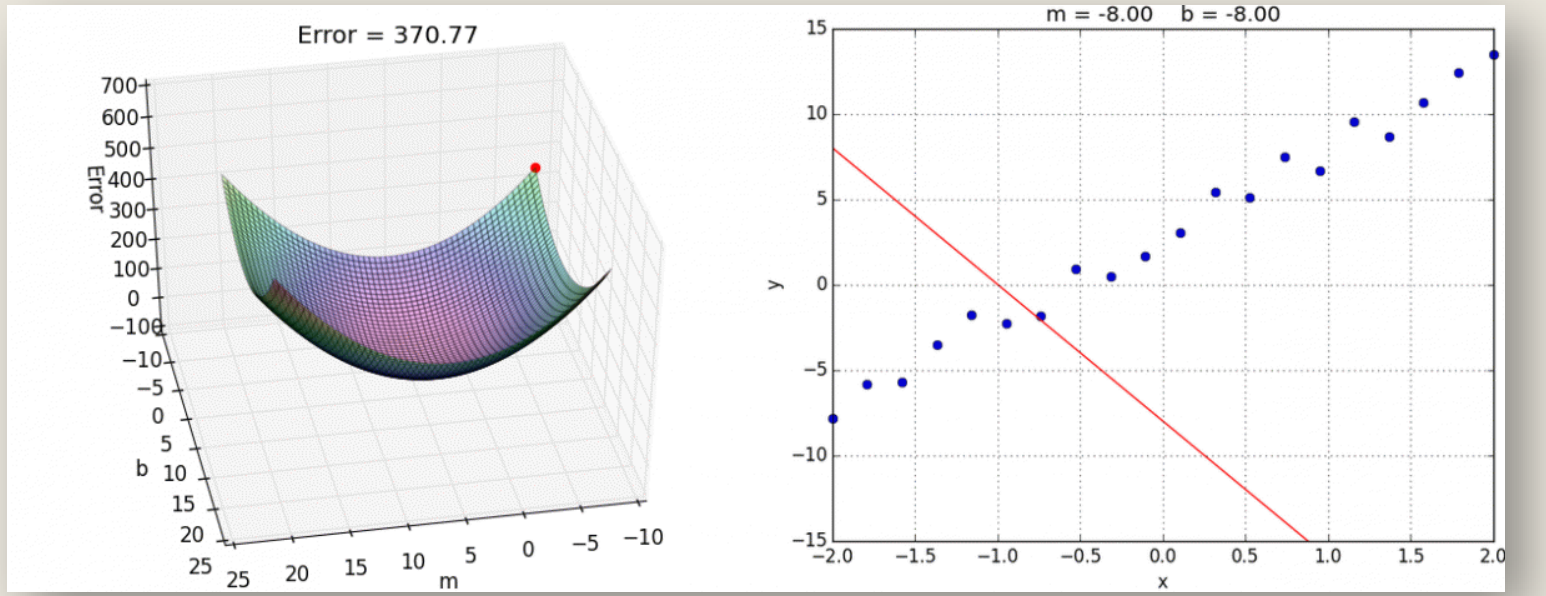


$$\theta^{(m+1)} = \theta^{(m)} - learningRate * \frac{\partial L}{\partial \theta^{(m)}}$$

XGBoost is a scalable and highly accurate implementation of gradient boosting

GRADIENT BOOSTING

- In order to minimize the loss function we take the derivative of the loss function and add the negative of this value to the weight scaled by some learning rate to achieve a more accurate model this is process is repeated until convergence.



$$\theta^{(m+1)} = \theta^{(m)} - learningRate * \frac{\partial L}{\partial \theta^{(m)}}$$

Why XGBoost?

- **Regularization:** penalize complex models through both L1 and L2 regularization. Regularization helps in preventing overfitting.

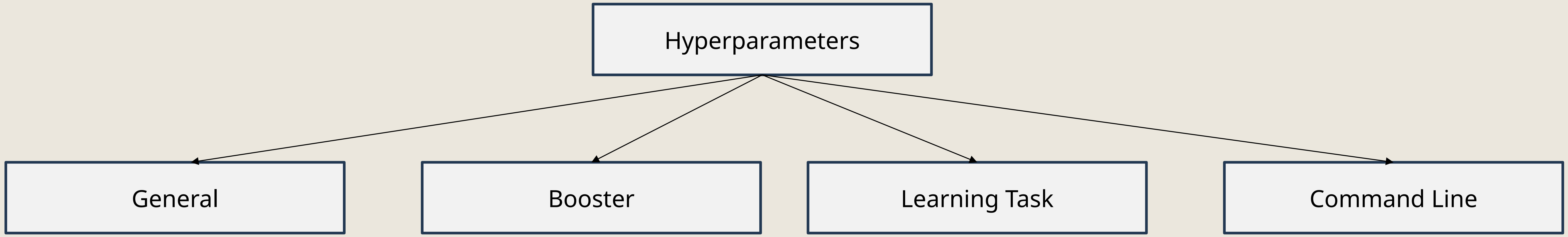
- **Handling sparse data:** sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data.

- **Weighted quantile sketch:** Most existing tree-based algorithms can find the split points when the data points are of equal weights (using a quantile sketch algorithm). However, they are not equipped to handle weighted data. XGBoost has a distributed weighted quantile sketch algorithm to effectively handle weighted data

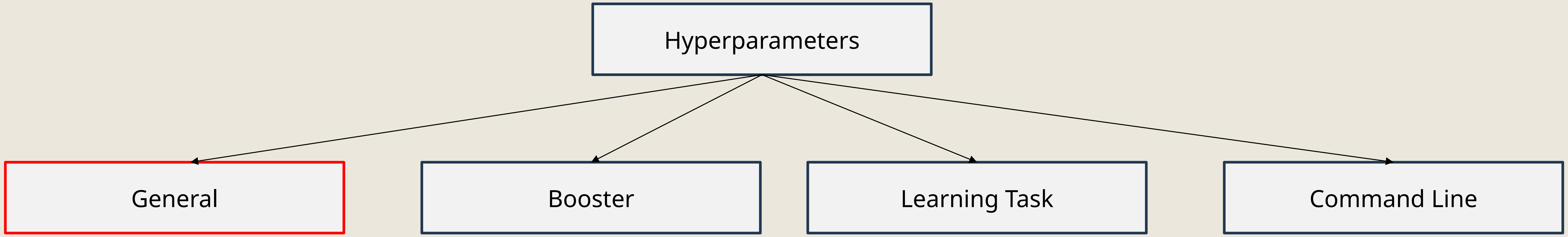
- **Block structure for parallel learning:** make use of multiple cores on the CPU.

XGBoost: A BOOSTING Ensemble. Does it really work as the name… | by Rishabh Kesarwani | AlmaBetter | Medium

Hyperparameters in XGBoost

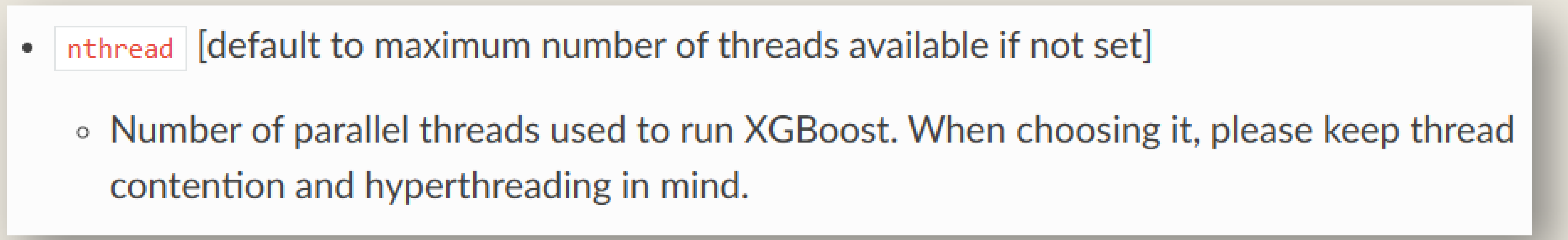


Hyperparameters in XGBoost

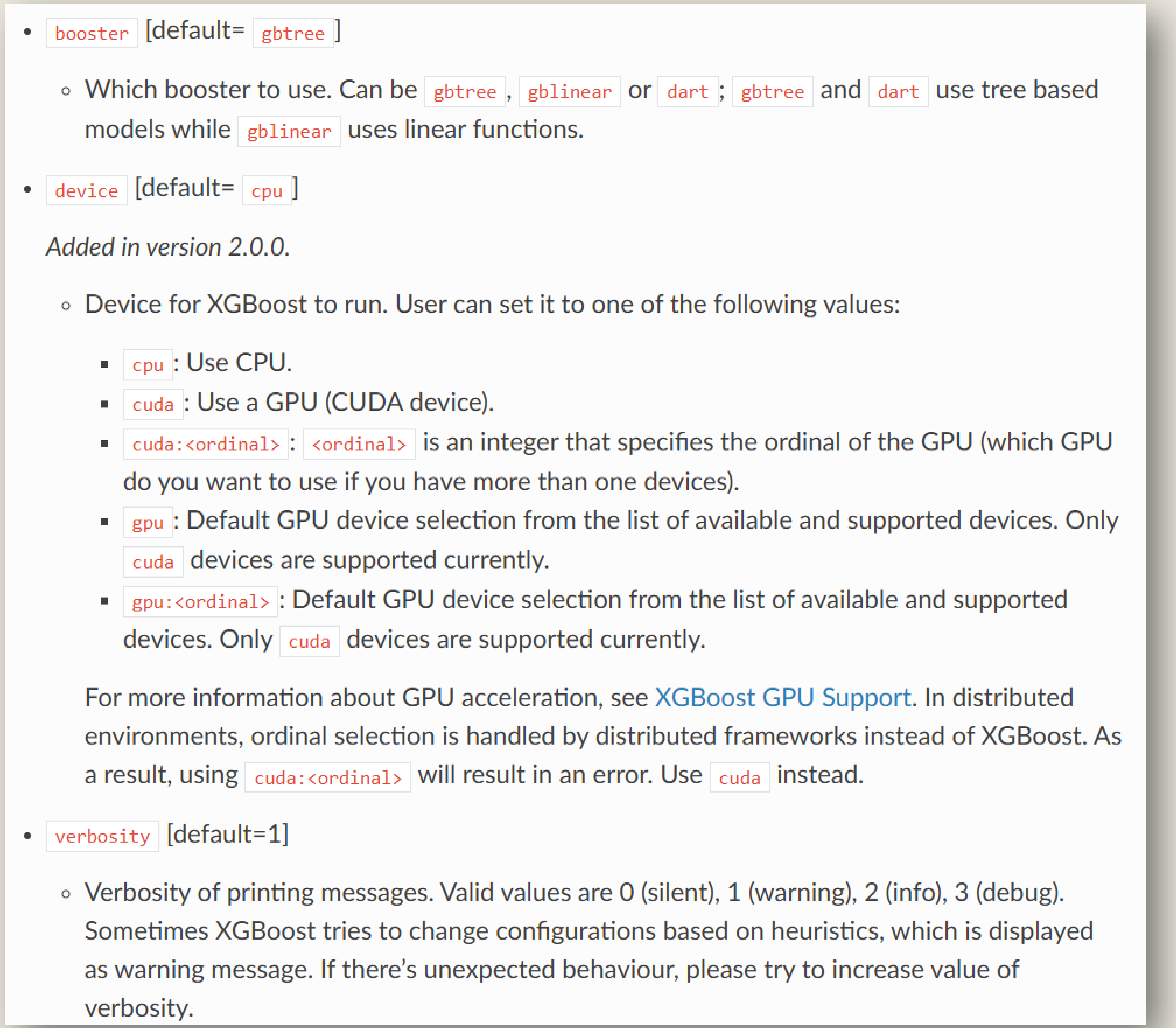


Hyperparameters in XGBoost

General



- `nthread` [default to maximum number of threads available if not set]
  - Number of parallel threads used to run XGBoost. When choosing it, please keep thread contention and hyperthreading in mind.



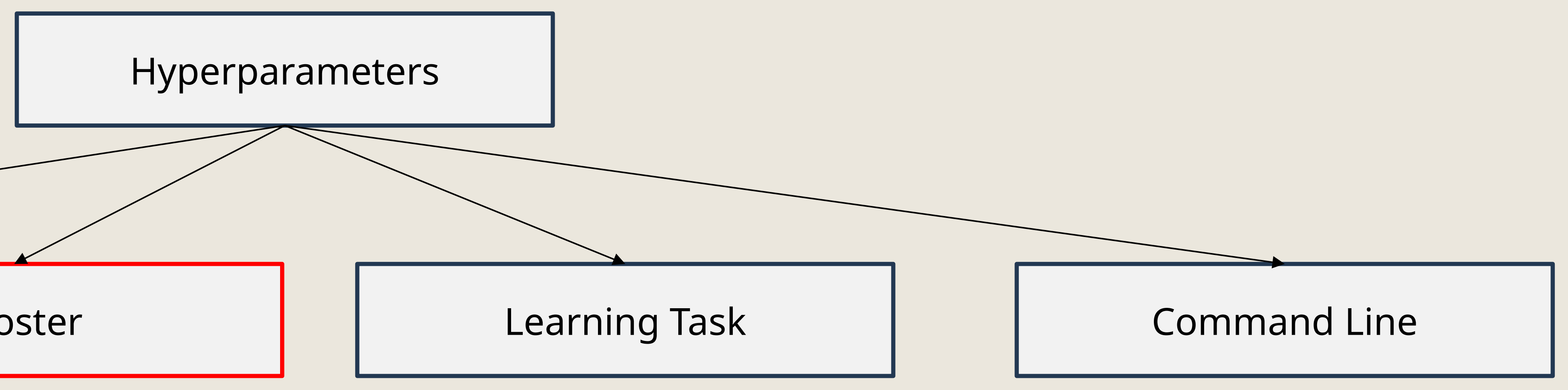
- `booster` [default= `gbtree`]
  - Which booster to use. Can be `gbtree`, `gblinear` or `dart`; `gbtree` and `dart` use tree based models while `gblinear` uses linear functions.
- `device` [default= `cpu`]

  Added in version 2.0.0.

  - Device for XGBoost to run. User can set it to one of the following values:
    - `cpu`: Use CPU.
    - `cuda`: Use a GPU (CUDA device).
    - `cuda:<ordinal>`: `<ordinal>` is an integer that specifies the ordinal of the GPU (which GPU do you want to use if you have more than one devices).
    - `gpu`: Default GPU device selection from the list of available and supported devices. Only `cuda` devices are supported currently.
    - `gpu:<ordinal>`: Default GPU device selection from the list of available and supported devices. Only `cuda` devices are supported currently.

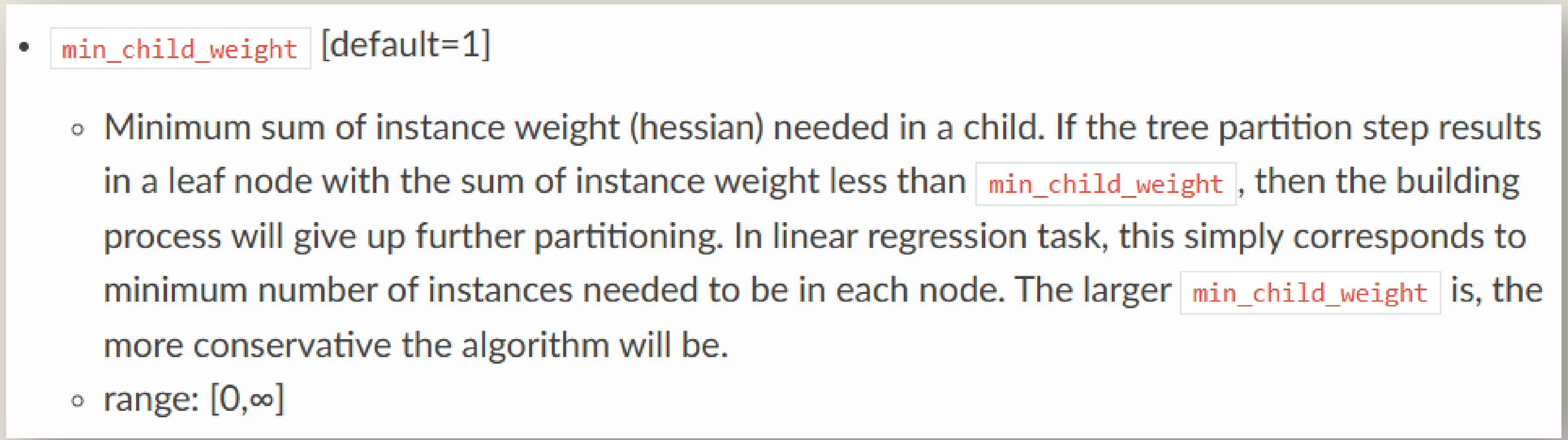
  For more information about GPU acceleration, see XGBoost GPU Support. In distributed environments, ordinal selection is handled by distributed frameworks instead of XGBoost. As a result, using `cuda:<ordinal>` will result in an error. Use `cuda` instead.
- `verbosity` [default=1]
  - Verbosity of printing messages. Valid values are 0 (silent), 1 (warning), 2 (info), 3 (debug). Sometimes XGBoost tries to change configurations based on heuristics, which is displayed as warning message. If there's unexpected behaviour, please try to increase value of verbosity.

Hyperparameters in XGBoost

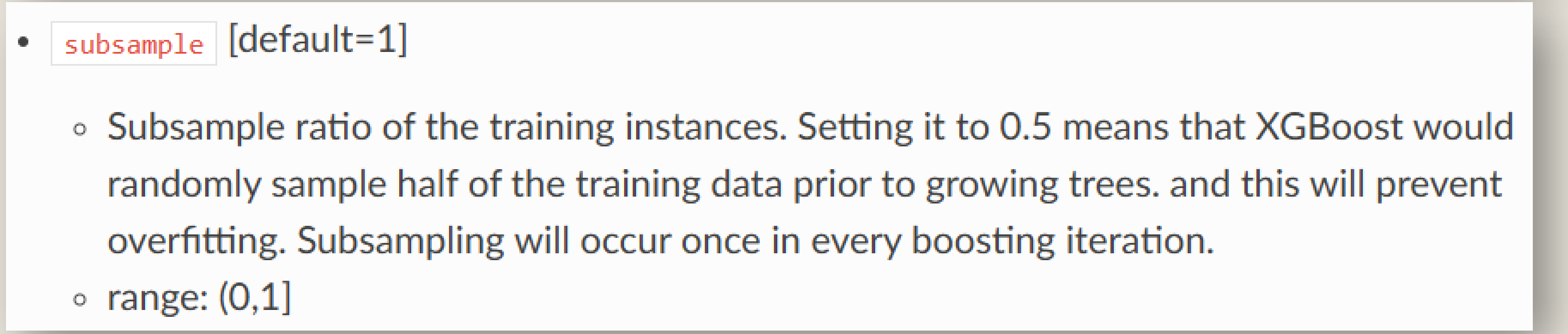


Hyperparameters in XGBoost

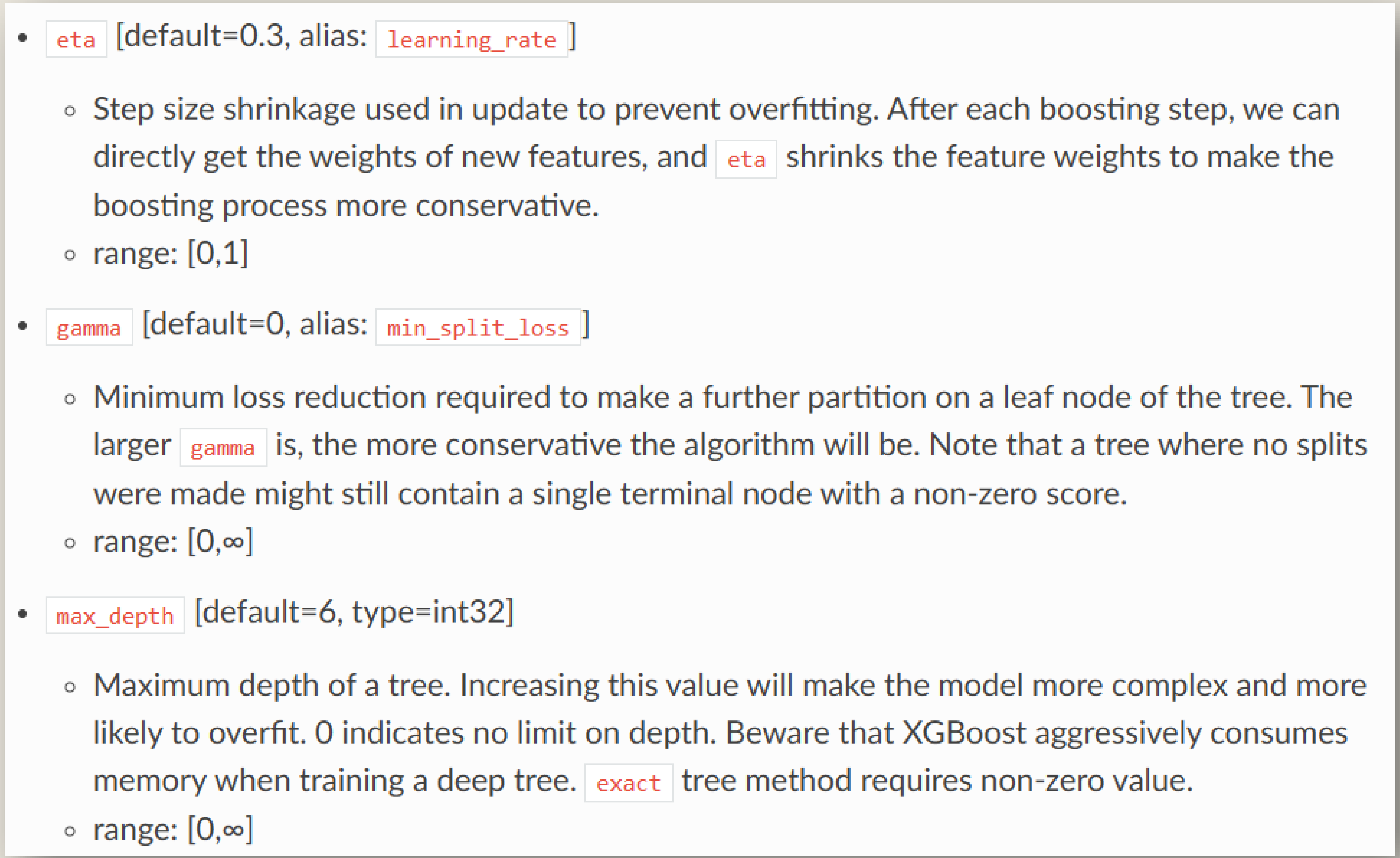
Booster



- `min_child_weight` [default=1]
  - Minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than `min_child_weight`, then the building process will give up further partitioning. In linear regression task, this simply corresponds to minimum number of instances needed to be in each node. The larger `min_child_weight` is, the more conservative the algorithm will be.
  - range: [0,∞]



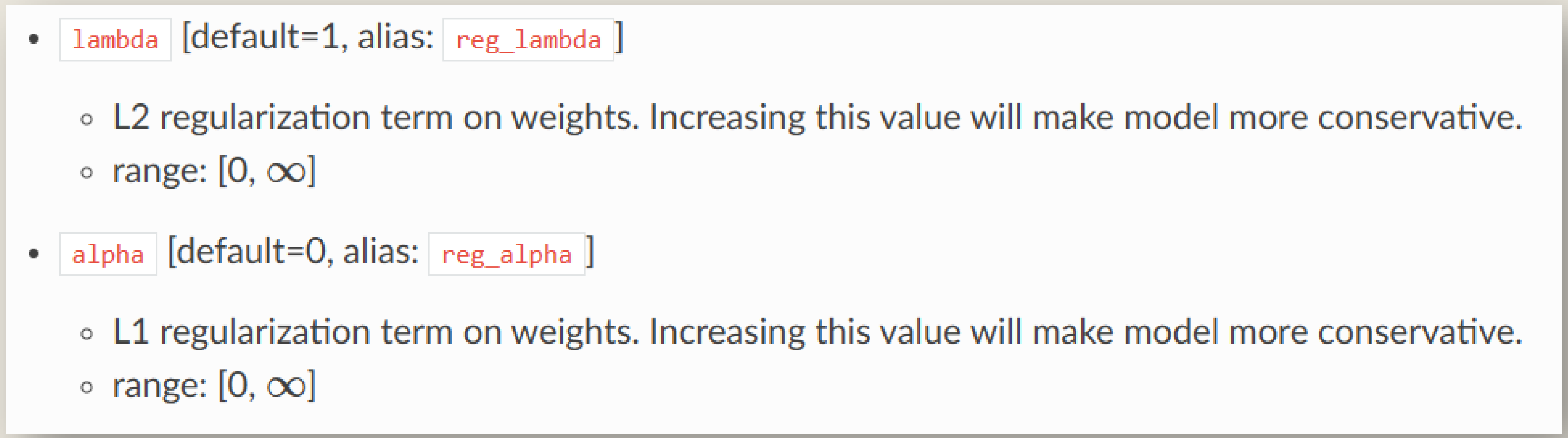
- `subsample` [default=1]
  - Subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. and this will prevent overfitting. Subsampling will occur once in every boosting iteration.
  - range: (0,1]



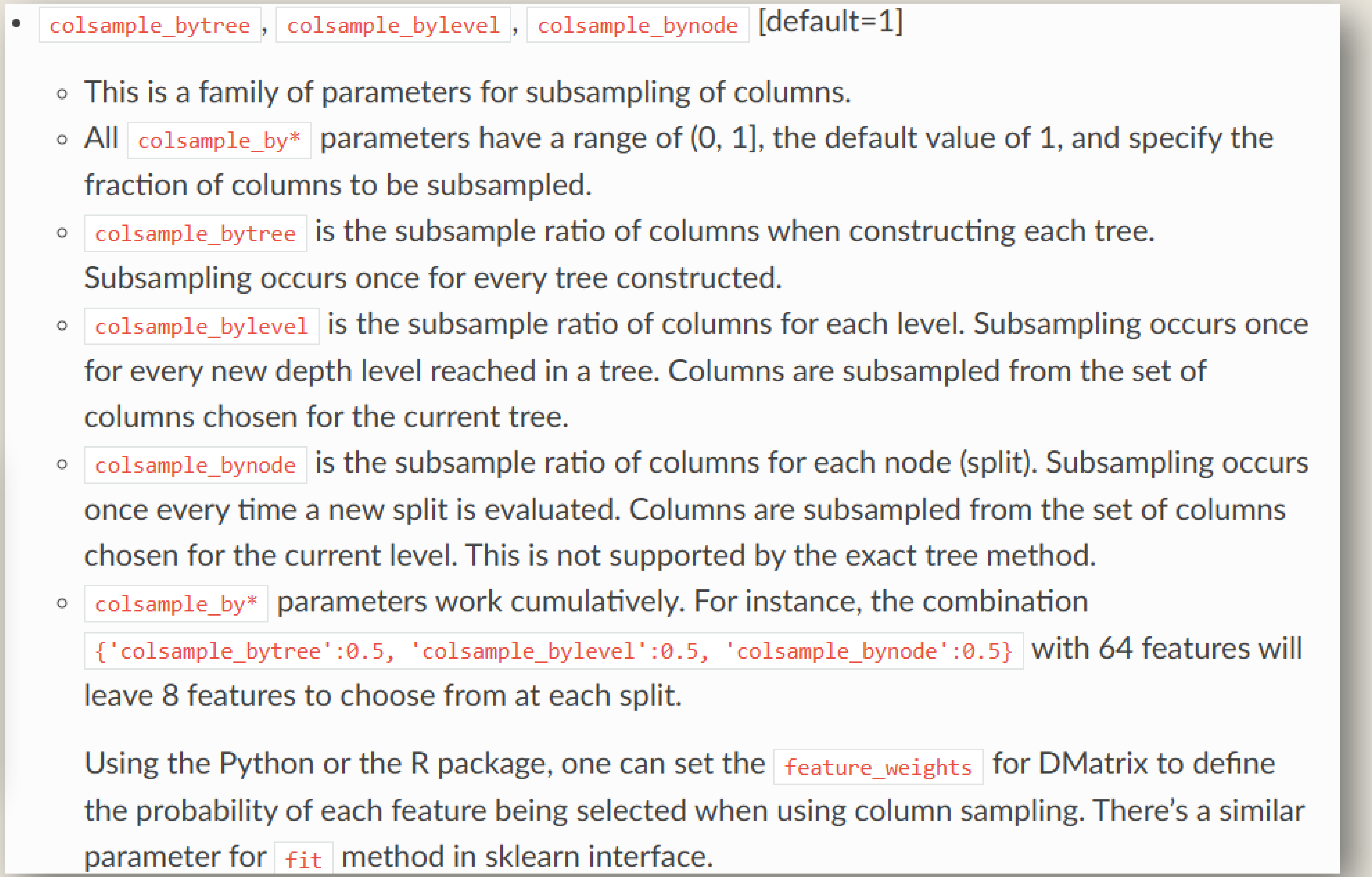
- `eta` [default=0.3, alias: `learning_rate`]
  - Step size shrinkage used in update to prevent overfitting. After each boosting step, we can directly get the weights of new features, and `eta` shrinks the feature weights to make the boosting process more conservative.
  - range: [0,1]
- `gamma` [default=0, alias: `min_split_loss`]
  - Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger `gamma` is, the more conservative the algorithm will be. Note that a tree where no splits were made might still contain a single terminal node with a non-zero score.
  - range: [0,∞]
- `max_depth` [default=6, type=int32]
  - Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 indicates no limit on depth. Beware that XGBoost aggressively consumes memory when training a deep tree. `exact` tree method requires non-zero value.
  - range: [0,∞]

Hyperparameters in XGBoost

Booster



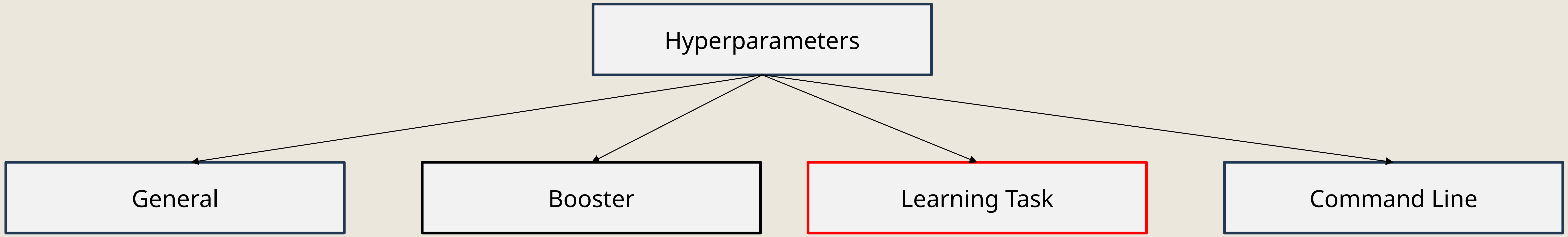
- `lambda` [default=1, alias: `reg_lambda`]
  - L2 regularization term on weights. Increasing this value will make model more conservative.
  - range: [0, ∞]
- `alpha` [default=0, alias: `reg_alpha`]
  - L1 regularization term on weights. Increasing this value will make model more conservative.
  - range: [0, ∞]



- `colsample_bytree`, `colsample_bylevel`, `colsample_bynode` [default=1]
  - This is a family of parameters for subsampling of columns.
  - All `colsample_by*` parameters have a range of (0, 1], the default value of 1, and specify the fraction of columns to be subsampled.
  - `colsample_bytree` is the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.
  - `colsample_bylevel` is the subsample ratio of columns for each level. Subsampling occurs once for every new depth level reached in a tree. Columns are subsampled from the set of columns chosen for the current tree.
  - `colsample_bynode` is the subsample ratio of columns for each node (split). Subsampling occurs once every time a new split is evaluated. Columns are subsampled from the set of columns chosen for the current level. This is not supported by the exact tree method.
  - `colsample_by*` parameters work cumulatively. For instance, the combination `{'colsample_bytree':0.5, 'colsample_bylevel':0.5, 'colsample_bynode':0.5}` with 64 features will leave 8 features to choose from at each split.

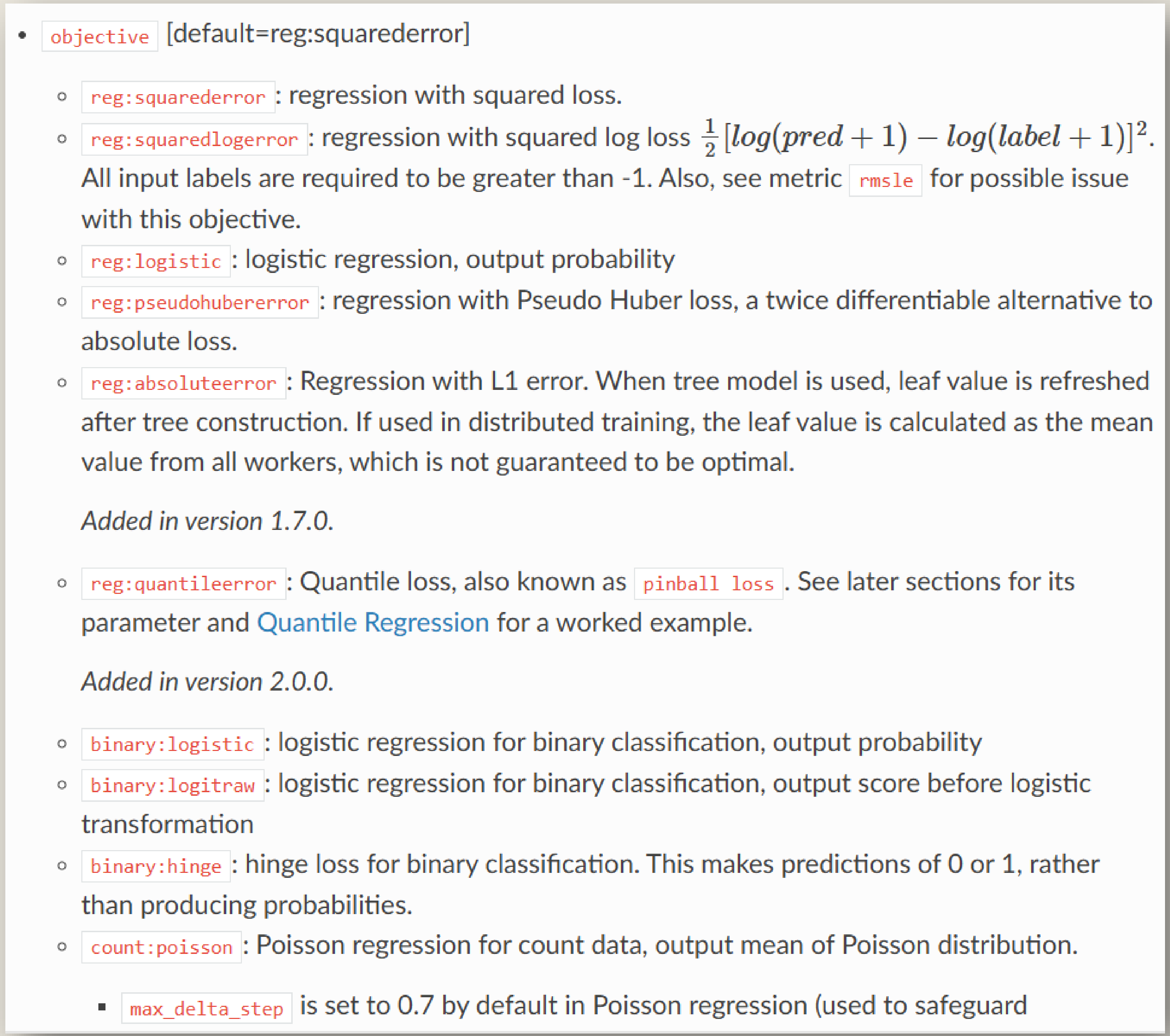
  Using the Python or the R package, one can set the `feature_weights` for DMatrix to define the probability of each feature being selected when using column sampling. There's a similar parameter for `fit` method in sklearn interface.

Hyperparameters in XGBoost



Hyperparameters in XGBoost

Learning Task



- `objective` [default=reg:squarederror]
  - `reg:squarederror`: regression with squared loss.
  - `reg:squaredlogerror`: regression with squared log loss $\frac{1}{2}[log(pred+1)-log(label+1)]^2$. All input labels are required to be greater than -1. Also, see metric `rmsle` for possible issue with this objective.
  - `reg:logistic`: logistic regression, output probability
  - `reg:pseudohubererror`: regression with Pseudo Huber loss, a twice differentiable alternative to absolute loss.
  - `reg:absoluteerror`: Regression with L1 error. When tree model is used, leaf value is refreshed after tree construction. If used in distributed training, the leaf value is calculated as the mean value from all workers, which is not guaranteed to be optimal.

    Added in version 1.7.0.

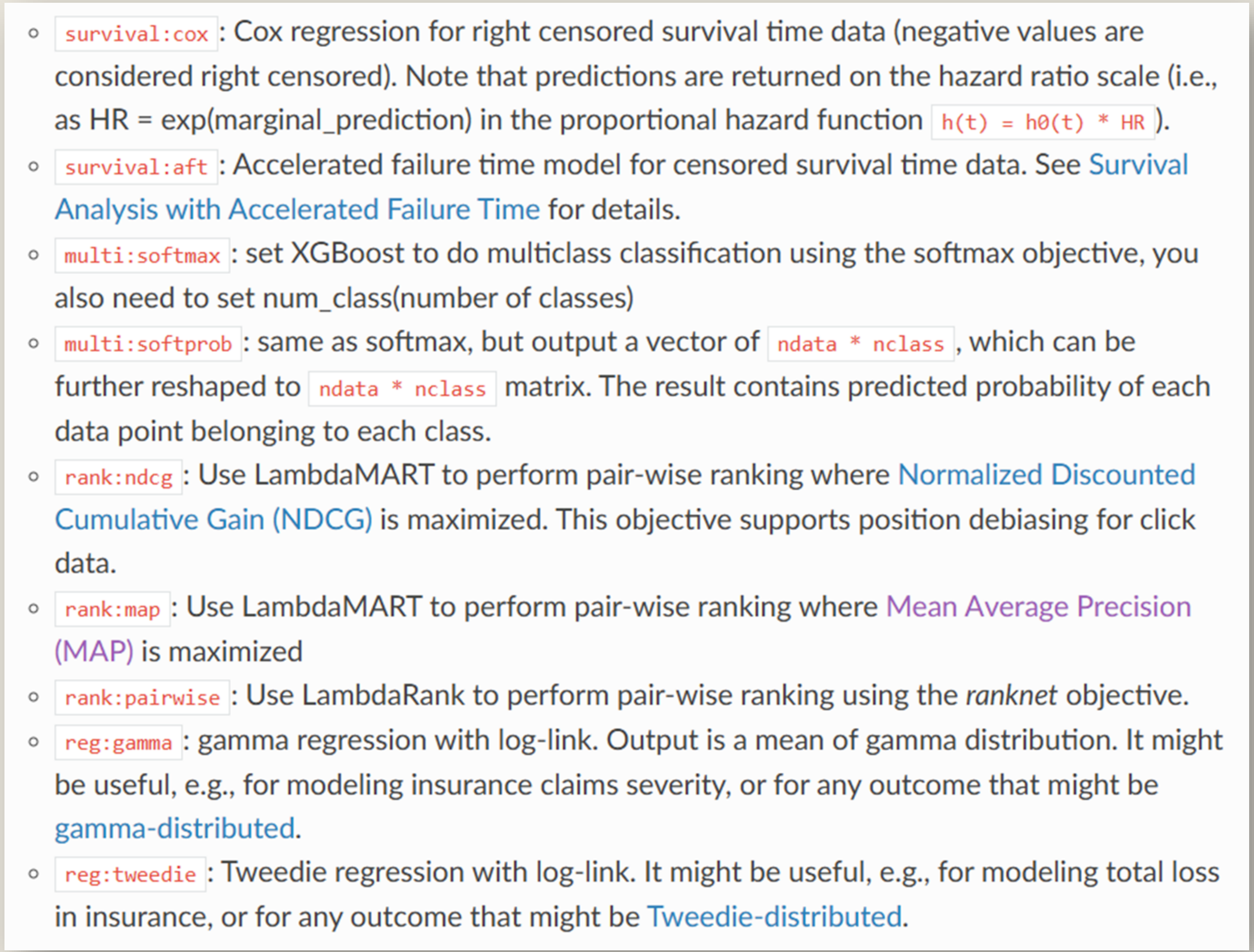
  - `reg:quantileerror`: Quantile loss, also known as `pinball loss`. See later sections for its parameter and Quantile Regression for a worked example.

    Added in version 2.0.0.

  - `binary:logistic`: logistic regression for binary classification, output probability
  - `binary:logitraw`: logistic regression for binary classification, output score before logistic transformation
  - `binary:hinge`: hinge loss for binary classification. This makes predictions of 0 or 1, rather than producing probabilities.
  - `count:poisson`: Poisson regression for count data, output mean of Poisson distribution.
    - `max_delta_step` is set to 0.7 by default in Poisson regression (used to safeguard

Hyperparameters in XGBoost

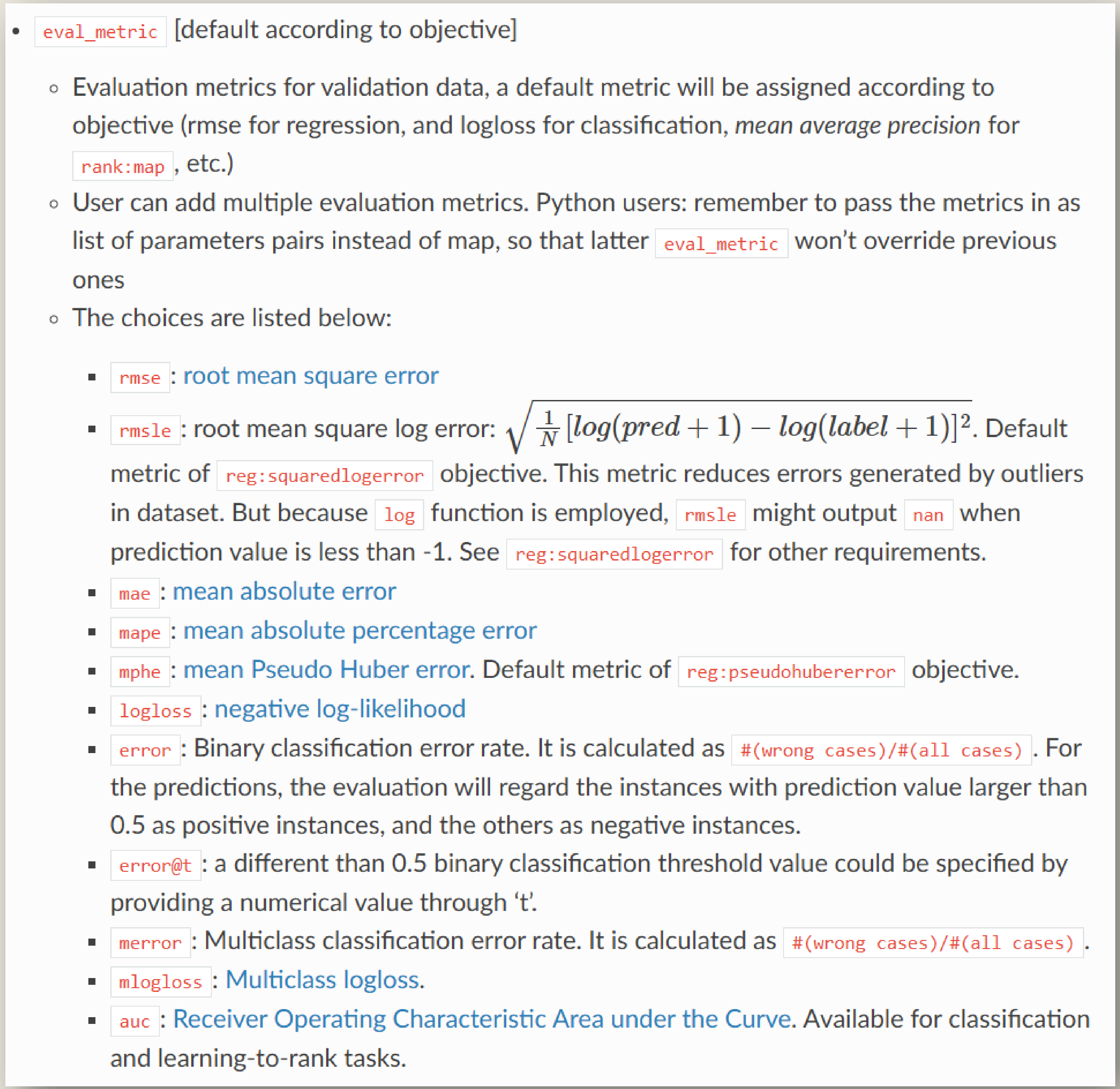
Learning Task



- `survival:cox`: Cox regression for right censored survival time data (negative values are considered right censored). Note that predictions are returned on the hazard ratio scale (i.e., as HR = exp(marginal_prediction) in the proportional hazard function `h(t) = h0(t) * HR`).
- `survival:aft`: Accelerated failure time model for censored survival time data. See Survival Analysis with Accelerated Failure Time for details.
- `multi:softmax`: set XGBoost to do multiclass classification using the softmax objective, you also need to set num_class(number of classes)
- `multi:softprob`: same as softmax, but output a vector of `ndata * nclass`, which can be further reshaped to `ndata * nclass` matrix. The result contains predicted probability of each data point belonging to each class.
- `rank:ndcg`: Use LambdaMART to perform pair-wise ranking where Normalized Discounted Cumulative Gain (NDCG) is maximized. This objective supports position debiasing for click data.
- `rank:map`: Use LambdaMART to perform pair-wise ranking where Mean Average Precision (MAP) is maximized
- `rank:pairwise`: Use LambdaRank to perform pair-wise ranking using the *ranknet* objective.
- `reg:gamma`: gamma regression with log-link. Output is a mean of gamma distribution. It might be useful, e.g., for modeling insurance claims severity, or for any outcome that might be gamma-distributed.
- `reg:tweedie`: Tweedie regression with log-link. It might be useful, e.g., for modeling total loss in insurance, or for any outcome that might be Tweedie-distributed.

Hyperparameters in XGBoost

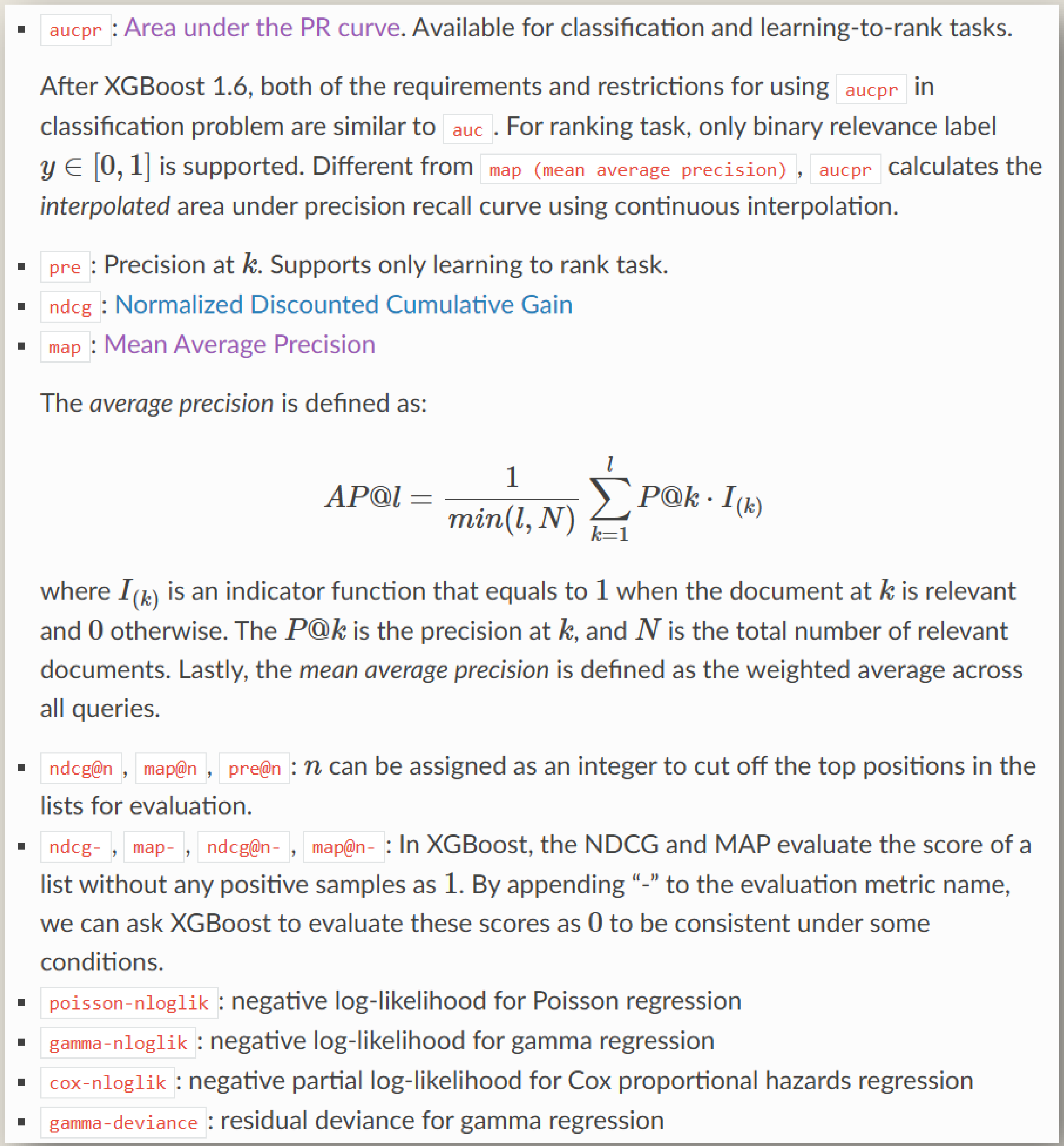
Learning Task



- `eval_metric` [default according to objective]
  - Evaluation metrics for validation data, a default metric will be assigned according to objective (rmse for regression, and logloss for classification, *mean average precision* for `rank:map`, etc.)
  - User can add multiple evaluation metrics. Python users: remember to pass the metrics in as list of parameters pairs instead of map, so that latter `eval_metric` won't override previous ones
  - The choices are listed below:
    - `rmse`: root mean square error
    - `rmsle`: root mean square log error: $\sqrt{\frac{1}{N}[log(pred+1) - log(label+1)]^2}$. Default metric of `reg:squaredlogerror` objective. This metric reduces errors generated by outliers in dataset. But because `log` function is employed, `rmsle` might output `nan` when prediction value is less than -1. See `reg:squaredlogerror` for other requirements.
    - `mae`: mean absolute error
    - `mape`: mean absolute percentage error
    - `mphe`: mean Pseudo Huber error. Default metric of `reg:pseudohubererror` objective.
    - `logloss`: negative log-likelihood
    - `error`: Binary classification error rate. It is calculated as `#(wrong cases)/#(all cases)`. For the predictions, the evaluation will regard the instances with prediction value larger than 0.5 as positive instances, and the others as negative instances.
    - `error@t`: a different than 0.5 binary classification threshold value could be specified by providing a numerical value through 't'.
    - `merror`: Multiclass classification error rate. It is calculated as `#(wrong cases)/#(all cases)`.
    - `mlogloss`: Multiclass logloss.
    - `auc`: Receiver Operating Characteristic Area under the Curve. Available for classification and learning-to-rank tasks.

Hyperparameters in XGBoost

Learning Task



- `aucpr`: Area under the PR curve. Available for classification and learning-to-rank tasks.

  After XGBoost 1.6, both of the requirements and restrictions for using `aucpr` in classification problem are similar to `auc`. For ranking task, only binary relevance label $y \in [0, 1]$ is supported. Different from `map (mean average precision)`, `aucpr` calculates the *interpolated* area under precision recall curve using continuous interpolation.

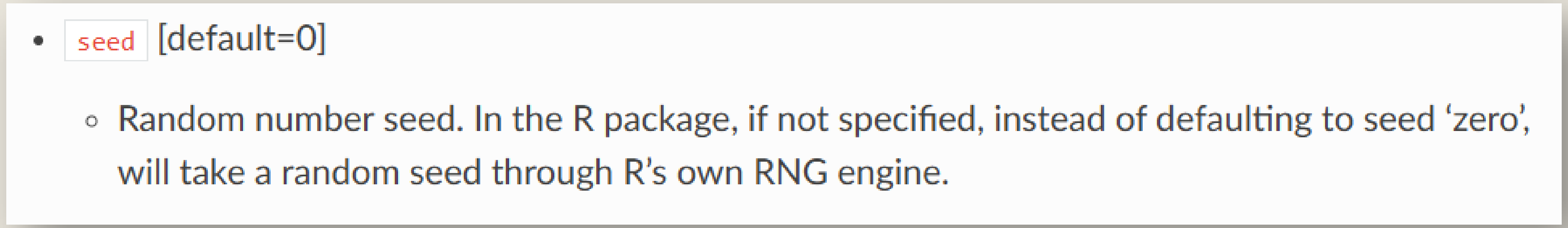
- `pre`: Precision at k. Supports only learning to rank task.
- `ndcg`: Normalized Discounted Cumulative Gain
- `map`: Mean Average Precision

  The *average precision* is defined as:

  $$AP@l = \frac{1}{min(l, N)} \sum_{k=1}^{l} P@k \cdot I_{(k)}$$

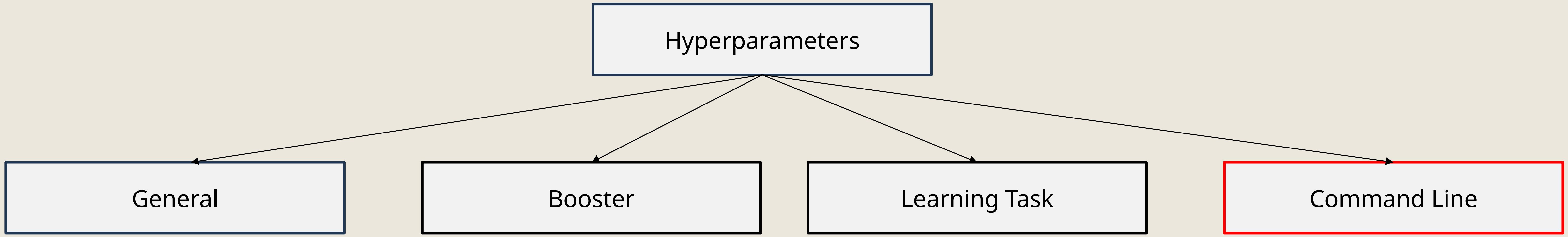
  where $I_{(k)}$ is an indicator function that equals to 1 when the document at k is relevant and 0 otherwise. The $P@k$ is the precision at k, and N is the total number of relevant documents. Lastly, the *mean average precision* is defined as the weighted average across all queries.

- `ndcg@n`, `map@n`, `pre@n`: n can be assigned as an integer to cut off the top positions in the lists for evaluation.
- `ndcg-`, `map-`, `ndcg@n-`, `map@n-`: In XGBoost, the NDCG and MAP evaluate the score of a list without any positive samples as 1. By appending "-" to the evaluation metric name, we can ask XGBoost to evaluate these scores as 0 to be consistent under some conditions.
- `poisson-nloglik`: negative log-likelihood for Poisson regression
- `gamma-nloglik`: negative log-likelihood for gamma regression
- `cox-nloglik`: negative partial log-likelihood for Cox proportional hazards regression
- `gamma-deviance`: residual deviance for gamma regression



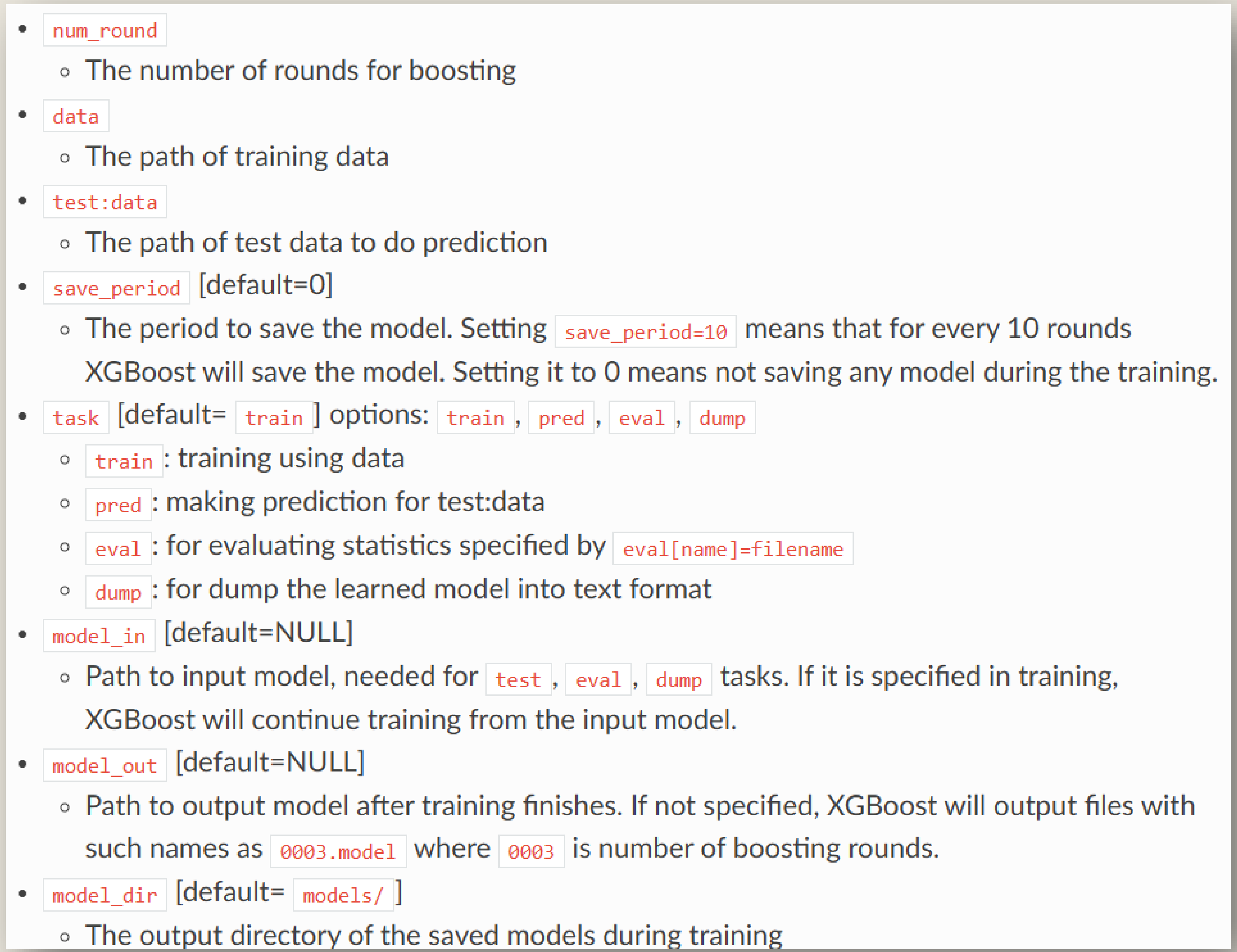
- `seed` [default=0]
  - Random number seed. In the R package, if not specified, instead of defaulting to seed 'zero', will take a random seed through R's own RNG engine.

Hyperparameters in XGBoost



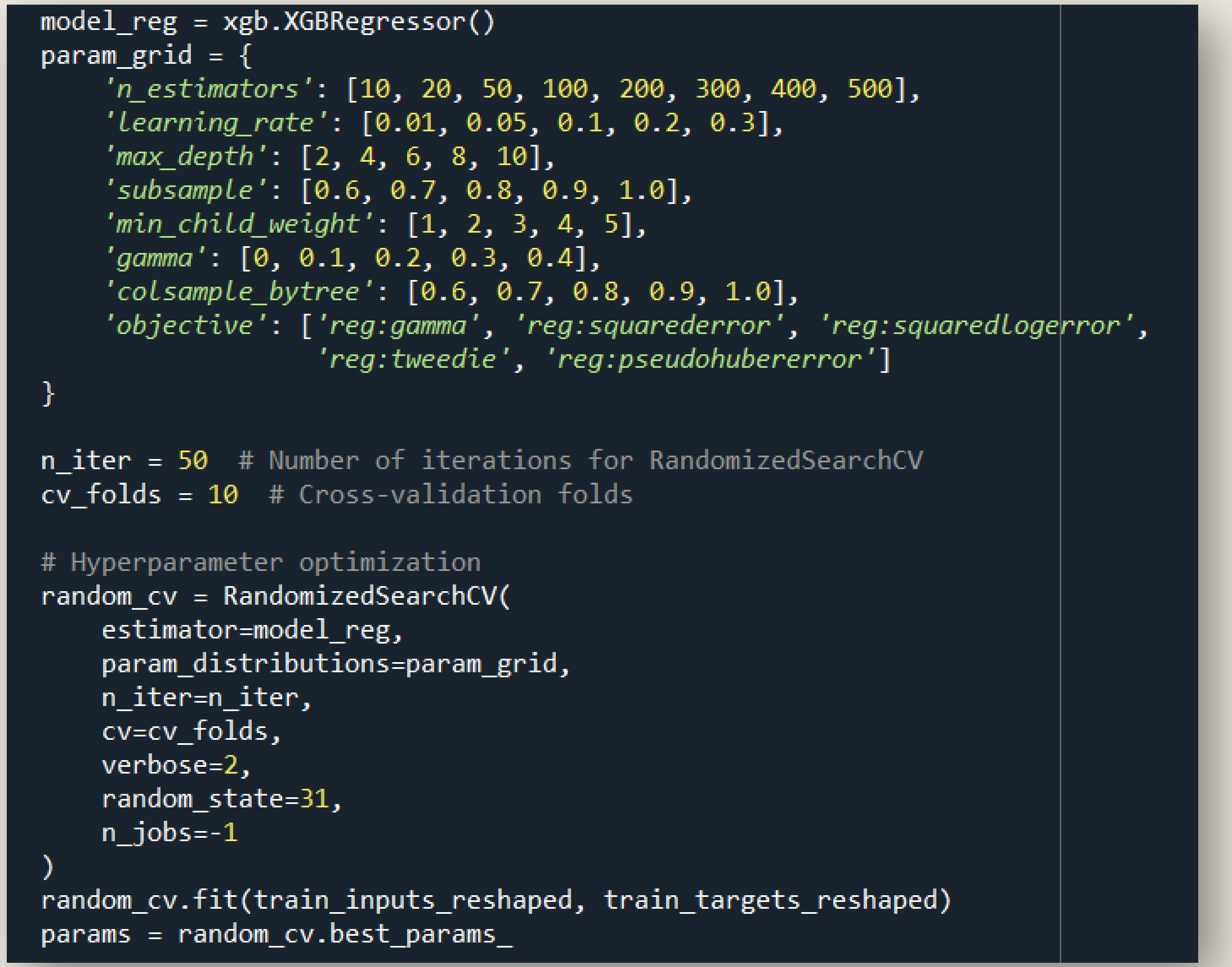
Hyperparameters in XGBoost

Command Line

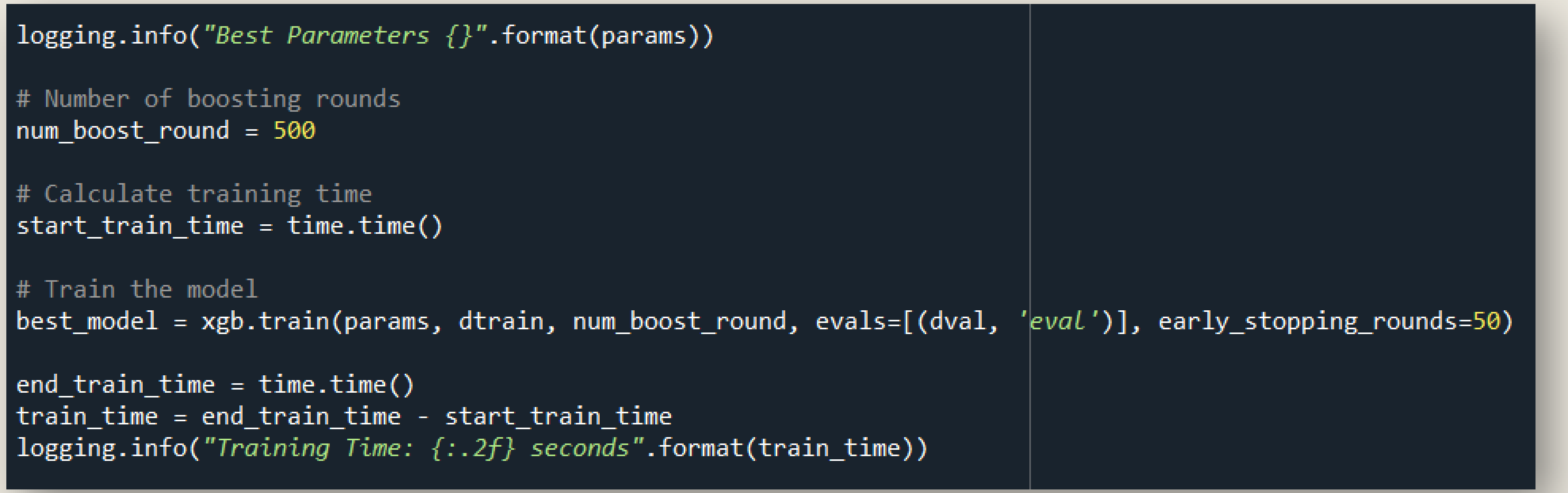


- `num_round`
  - The number of rounds for boosting
- `data`
  - The path of training data
- `test:data`
  - The path of test data to do prediction
- `save_period` [default=0]
  - The period to save the model. Setting `save_period=10` means that for every 10 rounds XGBoost will save the model. Setting it to 0 means not saving any model during the training.
- `task` [default= `train`] options: `train` , `pred` , `eval` , `dump`
  - `train` : training using data
  - `pred` : making prediction for test:data
  - `eval` : for evaluating statistics specified by `eval[name]=filename`
  - `dump` : for dump the learned model into text format
- `model_in` [default=NULL]
  - Path to input model, needed for `test` , `eval` , `dump` tasks. If it is specified in training, XGBoost will continue training from the input model.
- `model_out` [default=NULL]
  - Path to output model after training finishes. If not specified, XGBoost will output files with such names as `0003.model` where `0003` is number of boosting rounds.
- `model_dir` [default= `models/`]
  - The output directory of the saved models during training

Hyperparameters in XGBoost



```
model_reg = xgb.XGBRegressor()
param_grid = {
    'n_estimators': [10, 20, 50, 100, 200, 300, 400, 500],
    'learning_rate': [0.01, 0.05, 0.1, 0.2, 0.3],
    'max_depth': [2, 4, 6, 8, 10],
    'subsample': [0.6, 0.7, 0.8, 0.9, 1.0],
    'min_child_weight': [1, 2, 3, 4, 5],
    'gamma': [0, 0.1, 0.2, 0.3, 0.4],
    'colsample_bytree': [0.6, 0.7, 0.8, 0.9, 1.0],
    'objective': ['reg:gamma', 'reg:squarederror', 'reg:squaredlogerror',
                  'reg:tweedie', 'reg:pseudohubererror']
}

n_iter = 50  # Number of iterations for RandomizedSearchCV
cv_folds = 10  # Cross-validation folds

# Hyperparameter optimization
random_cv = RandomizedSearchCV(
    estimator=model_reg,
    param_distributions=param_grid,
    n_iter=n_iter,
    cv=cv_folds,
    verbose=2,
    random_state=31,
    n_jobs=-1
)
random_cv.fit(train_inputs_reshaped, train_targets_reshaped)
params = random_cv.best_params_
```

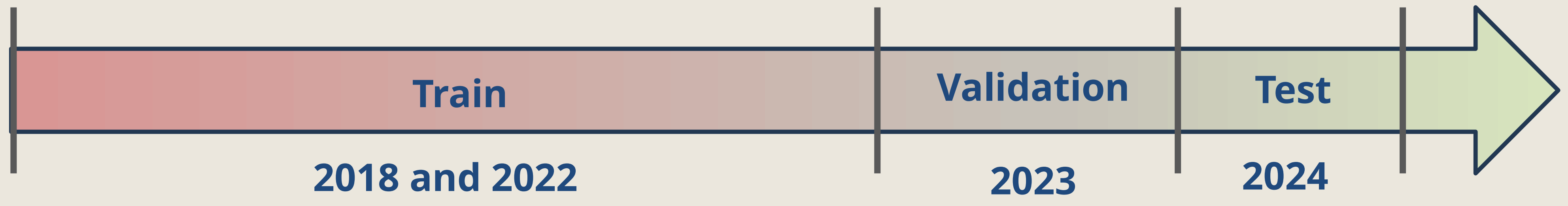


```
logging.info("Best Parameters {}".format(params))

# Number of boosting rounds
num_boost_round = 500

# Calculate training time
start_train_time = time.time()

# Train the model
best_model = xgb.train(params, dtrain, num_boost_round, evals=[(dval, 'eval')], early_stopping_rounds=50)

end_train_time = time.time()
train_time = end_train_time - start_train_time
logging.info("Training Time: {:.2f} seconds".format(train_time))
```

Downloading data from repository

📓 6. Use Jupyter Notebook

- In the conda environment go to the project directory using: cd “project directory”
- Type “jupyter lab”
- Open the 01_Data_Acquisition_and_Loading.ipynb notebook


Train
Validation
Test
2018 and 2022
2023
2024