# Getting Started with MOMENT:

# An R Package for Multi-Omics and Multi-Class Ensemble Learning in Predictive Clinical Outcome Modelling

## Introduction

MOMENT provides a set of ensemble machine learning algorithms that can successfully model multi-omics multi-class data. The algorithms are based on a technique known as late integration, training a machine learning model on each modality independently and aggregating the results in different ways to give a final prediction.

In addition, MOMENT allows users to model individual modalities, a concatenation of all modalities or a selected subset of modalities, for comparison with the ensemble models, employing either classification or survival analysis. Furthermore, an incremental model can determine a minimal set of modalities for accurate modelling of the data and an exploration model allows the user to plot 2-dimensional representations of the data using principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE) or Uniform Manifold Approximation and Projection (UMAP).

MOMENT is fully customisable and highly flexible, using an object-oriented design. It allows the user to model multi-omics multi-class data, without the need for a detailed understanding of the specifics of machine learning, nor of the mlr (Machine Learning in R) package, on which MOMENT relies.

## Preparing to Use MOMENT

### The Configuration File

MOMENT’s configuration file controls the modelling process, allowing the user to specify all of the parameters of that modelling. The configuration file is an Excel spreadsheet and its format is described in Table 1.

The *Main* tab specifies parameters that control the overall modelling process. Some of the parameters on this tab are specific to certain types of modelling, such as survival analysis, and can be omitted if that type of modelling is not being performed. Default values are used if a parameter is not specified. The *Datasets* tab gives details of the files containing the data to be modelled. The *BaseModels* tab describes details of the machine learning pipeline to be applied to each dataset. The *Parameters* tab sets up valid values for use by the other tabs in drop-down lists.

The data to be modelled should be provided in comma-separated (.csv) files, one modality per file. MOMENT expects the data to be tabular, with one sample per row and one variable per column. However, because Excel imposes a limit on the number of columns per file, very large datasets, such as omics data, are sometimes transposed, storing one sample per column and one variable per row. In this case a ‘T’ can be included in the Transpose column of the Datasets tab for that data modality to indicate that the data should be transposed when it is imported.

All of the modelling in MOMENT is performed using cross-validation. The number of folds and repeats for this cross validation are specified on the *Main* tab of the configuration file.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tab Name** | **Column Name** | **Content** | **Valid/Default Values** |
| **Main** | DataDir | The name of the directory in which he data files can be found. |  |
| CacheDir | The name of the directory in which the mlr cache files will be stored. If not supplied then caching will not be used. |  |
| ResultFile | The base name of the files to which the results will be saved. |  |
| TargetVar | The name of the target variable – the value to be predicted in the modelling. This variable MUST be present in every data file. |  |
| IDVar | The name of the variable containing a unique identifier for each sample. This variable MUST be present in every data file. |  |
| TimeVar | For survival analysis only – the name of the variable containing the time to the event. |  |
| StatusVar | For survival analysis only – the name of the variable containing the survival status. |  |
| BoostIters | For the boosted models only, (ADABoost and PB-MVBoost), the number of boosting steps on each iteration. |  |
| ItersOuter  FoldsOuter | The number of folds (FoldsOuter) and number of repeats (ItersOuter) for cross-validation. |  |
| ItersInner  FoldsInner | For the MetaLearner model only, the number of folds (FoldsInner) and number of repeats (ItersInner) for the inner cross-validation. |  |
| MetaLearner | For the MetaLearner model only, the type of model to be used as the top level meta learner. |  |
| MetaParams | For the MetaLearner model only, the hyperparameters to be passed to the MetaLearner. |  |
| **Datasets** | Modality | The name of the modality or data source. |  |
| Dataset | The name of the file in which the data for this modality is stored. |  |
| Transpose | Should the data be transposed after reading it in? | T or blank |
| Exclusions | The names of any features that should be excluded from modelling. |  |
| Inclusions | Limit the modelling to only these features. This is best used without feature selection. |  |
| Categoricals | The names of any categorical features. These columns will be explicitly converted to categorical variables. |  |
| **BaseModels** | Modality | The name of the modality or data source. |  |
| Normalisation | The normalisation method to be applied to the data. | NONE, STAND, LOGT, MINMAX, CPM, CPM\_LOGT |
| Imputation | The imputation method to be applied to the data. | NONE, MICE, KNN |
| Feature Selector | The feature selector to be used in the pipeline. | NONE, BORUTA, RANGER, RF |
| FSParams | The hyperparameters to be used with this feature selector. | Any valid values |
| Learner | The classifier to be used in the pipeline. | GBM, RANGER, RF, SVM, XGBTREE, XGBLIN, RPART |
| Params | The hyperparameters to be used with this classifier. |  |

### Machine Learning Algorithms for Classification

The following learners and feature selectors, and their default parameters, are available for classification within MOMENT:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Identifying Code** | **R package** | **Default Hyperparameters** |
| Linear regression – Lasso | LASSO | *glmnet* | alpha = 1, nfolds = 5 |
| Linear regression – Ridge | RIDGE | *glmnet* | alpha = 0, nfolds = 5 |
| Linear regression - ElasticNet | ELASTICNET | *glmnet* | alpha = 0.5, nfolds = 5 |
| Random Forest | RF | *random-Forest* | ntree = 1000, importance = TRUE, nodesize = 4 |
| Ranger random forest | RANGER | *ranger* | splitrule = "gini", importance = "permutation", num.trees = 1000, min.node.size = 15 |
| Gradient boosting machine | GBM | *gbm* | n.trees = 1000, interaction.depth = 6, shrinkage = 0.01, n.minobsinnode = 5, keep.data = TRUE |
| Extreme gradient boosting with tree-based models | XGBTREE | *xgboost* | booster = BOOSTER\_TREE, objective = "multi:softprob", eval\_metric = "mlogloss", num\_class = 4, max\_depth = 5, eta = 0.001, gamma = 3, subsample = 0.75 |
| Extreme gradient boosting with linear models | XGBLIN | *xgboost* | booster = BOOSTER\_LINEAR, alpha = 0, lambda = 19 |
| Naïve Bayes | NB | *e1071* |  |

### Machine Learning Algorithms for Survival Analysis

The following learners and feature selectors, and their default parameters, are available for survival analysis within MOMENT:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model and R package** | **Identifying Code** | **R package** | **Default Hyperparameters** |
| Cox proportional hazards | COXPH | *survival* |  |
| Linear regression - Lasso | LASSO | *glmnet* | alpha = 1, nfolds = 5 |
| Linear regression - Ridge | RIDGE | *glmnet* | alpha = 0, nfolds = 5 |
| Linear regression - ElasticNet | ELASTICNET | *glmnet* | alpha = 0.5, nfolds = 5 |
| Boosted Cox | COXBOOST | *coxBoost* |  |
| Gradient Boosting with Componentwise Linear Models | GLMBOOST | *survival*  *mboost* |  |
| Extreme gradient boosting with tree-based models | XGBTREE | *xgboost* | booster = BOOSTER\_TREE, alpha = 0 |
| Extreme gradient boosting with linear models | XGBLIN | *xgboost* | booster = BOOSTER\_LINEAR, alpha = 0 |
| Random Survival Forest | RFSRC | *survival*  *random-ForestSRC* | ntree = 1000, importance = TRUE |
| Ranger random forest | RANGER | *ranger* | splitrule = "maxstat", importance = "permutation", num.trees = 1000 |

## Multi-Modal Methods

MOMENT provides a set of ensemble machine learning algorithms that can successfully model multi-omics multi-class data. The algorithms are all based on late integration; i.e. they train a machine learning model on each modality independently and aggregate the results in different ways to give a final prediction.

Each of the methods has a similar interface, making them easy to use.

### Multi-Modal Voting Ensemble

**Description:**

The multi-modal voting ensembletrains a classifier on each modality and aggregates the results using a hard or soft vote to give a final classification.

**Class:** MM\_Voting

**Usage:**

MM\_Voting$new(config, decision = "prob", subset = NULL, balance = FALSE, filter\_zeroes = 90.0, filter\_missings = 50.0, filter\_corr = FALSE, filter\_var = FALSE)

MM\_Voting$learn(validate = FALSE)

**Arguments**

|  |  |
| --- | --- |
| config | Configuration object, specifying details of the modelling process. |
| decision | Type of decision to be used in aggregating the results– “vote” or “hard”, “prob” or “soft” |
| subset | Subset of modalities to be modelled. The format is a comma separated list of integers or NULL if no sub-setting is to be performed. |
| balance | Should the data in each modality be balanced? Useful for data containing unbalanced classes. |
| filter\_zeroes | The maximum percentage of zero values allowed in each feature. E.g. filter\_zeroes = 90 means that features containing more than 90% zero values will be excluded from the analysis. |
| filter\_missings | The maximum percentage of missing values allowed in each feature. E.g. filter\_missings = 50 means that features containing more than 50% missing values will be excluded from the analysis. |
| filter\_corr | Should the data be filtered by removing correlated values? T/F |
| filter\_var | Should the data be filtered by removing values with low variance? T/F |
| validate | Should validation be performed? T/F |

**Value**

An object of class MM\_Results containing all of the results from the model.

**Examples**

cc = config("config\_gbm\_boruta\_4C.xlsx")

vote\_ens\_h = MM\_Voting$new(cc, decision = 'hard', subset = NULL)

res\_ens\_h = vote\_ens\_h$learn(validate = FALSE)

res\_ens\_h$write("voting\_hard")

vote\_ens\_s = MM\_Voting$new(cc, decision = 'soft', subset = NULL, concat = FALSE)

res\_ens\_s = vote\_ens\_s$learn(validate = FALSE)

res\_ens\_s$write("voting\_soft")

### Multi-Modal Meta Learner

**Description:**

A meta learner is a model that learns from the outputs of other models. A classifier is first trained on each modality independently, giving a set of base models. The meta-learner is then trained on the out-of-fold predictions from the base-models i.e predictions made on data not seen during training. Finally, the meta-learner makes a prediction on new out-of-fold data.

**Class:** MM\_Meta\_Learner

**Usage:**

MM\_Meta\_Learner $new(config, decision = "prob", subset = NULL, balance = FALSE, filter\_zeroes = 90.0, filter\_missings = 50.0, filter\_corr = FALSE, filter\_var = FALSE)

MM\_Meta\_Learner $learn(validate = FALSE)

**Arguments**

|  |  |
| --- | --- |
| config | Configuration object, specifying details of the modelling process. |
| decision | Type of decision to be used in aggregating the results– “vote” or “hard”, “prob” or “soft” |
| subset | Subset of modalities to be modelled. The format is a comma separated list of integers or NULL if no sub-setting is to be performed. |
| balance | Should the data in each modality be balanced? Useful for data containing unbalanced classes. |
| filter\_zeroes | The maximum percentage of zero values allowed in each feature. E.g. filter\_zeroes = 90 means that features containing more than 90% zero values will be excluded from the analysis. |
| filter\_missings | The maximum percentage of missing values allowed in each feature. E.g. filter\_missings = 50 means that features containing more than 50% missing values will be excluded from the analysis. |
| filter\_corr | Should the data be filtered by removing correlated values? T/F |
| filter\_var | Should the data be filtered by removing values with low variance? T/F |
| validate | Should validation be performed? T/F |

**Value**

An object of class MM\_Results containing all of the results from the model.

**Examples**

cc = config("config\_gbm\_boruta\_4C.xlsx")

meta = MM\_Meta\_Learner$new(cc, subset = NULL)

res\_meta = meta$learn(validate = FALSE)

res\_meta$write("meta\_22")

### Multi-Modal Adaboost

**Description:**

AdaBoost (or Adaptive Boosting) is an ensemble machine learning technique that trains multiple weak classifiers in an iterative process. On each iteration, the data points that were misclassified in the previous iteration are given increased weight, encouraging the model to focus on those instances. In this way each classifier learns from the mistakes of its predecessors and ultimately these classifiers are combined using a weighted linear sum to form a stronger classifier.

Here, Adaboost has been adapted to handle multi-modal data, by training a classifier on each data modality at each iteration and aggregating the results of these classifiers using a hard vote a soft vote or a meta learner. In addition, an instance is considered misclassified unless it is correctly classified by a majority of modalities or with a probability that is significantly higher than the next highest class probability.

**Class:** MM\_Adaboost

**Usage:**

MM\_Adaboost$new(config, nrounds = 10, meta\_lrn = "RF", decision = "prob", subset = NULL, balance = FALSE, filter\_zeroes = 90.0, filter\_missings = 50.0, filter\_corr = FALSE, filter\_var = FALSE)

MM\_Adaboost$learn(validate = FALSE)

**Arguments**

|  |  |
| --- | --- |
| config | Configuration object, specifying details of the modelling process. |
| nrounds | The number of rounds of boosting to be performed. |
| meta\_lrn | The name of the meta learner. Options can be found in class Learner. |
| decision | Type of decision to be used in aggregating the results– “vote” or “hard”, “prob” or “soft” |
| subset | Subset of modalities to be modelled. The format is a comma separated list of integers or NULL if no sub-setting is to be performed. |
| balance | Should the data in each modality be balanced? Useful for data containing unbalanced classes. |
| filter\_zeroes | The maximum percentage of zero values allowed in each feature. E.g. filter\_zeroes = 90 means that features containing more than 90% zero values will be excluded from the analysis. |
| filter\_missings | The maximum percentage of missing values allowed in each feature. E.g. filter\_missings = 50 means that features containing more than 50% missing values will be excluded from the analysis. |
| filter\_corr | Should the data be filtered by removing correlated values? T/F |
| filter\_var | Should the data be filtered by removing values with low variance? T/F |
| validate | Should validation be performed? T/F |

**Value**

An object of class MM\_Results containing all of the results from the model.

**Examples**

cc = config("config\_gbm\_boruta\_4C.xlsx")

ada\_h = MM\_Adaboost$new(cc, decision = 'hard', subset = NULL)

res\_ada\_h = ada\_h$learn(validate)

res\_ada\_h$write("ada\_hard")

ada\_s = MM\_Adaboost$new(cc, decision = 'soft', subset = NULL)

res\_ada\_s = ada\_s$learn(validate)

res\_ada\_s$write("ada\_soft")

ada\_m = MM\_Adaboost$new(cc, decision = 'meta', subset = NULL)

res\_ada\_m = ada\_m$learn(validate = FALSE)

res\_ada\_m$write("ada\_meta")

### PB-MVBoost

**Description:**

PB-MVBoost is similar to Adaboost, but adds an extra dimension. It tries to balance the accuracy of the classifiers trained on each modality with the diversity of their outputs by learning two sets of weights – weights over the classifiers and weights over the modalities. It combines these results using a weighted vote, learning the weights by minimising an upper-bound on the error of the majority vote.

**Class:** PB\_MVBoost

**Usage:**

PB\_MVBoost$new(config, nrounds = 10, decision\_tree\_depth = 1, decision = "prob", subset = NULL, balance = FALSE, filter\_zeroes = 90.0, filter\_missings = 50.0, filter\_corr = FALSE, filter\_var = FALSE)

PB\_MVBoost$learn(validate = FALSE)

**Arguments**

|  |  |
| --- | --- |
| config | Configuration object, specifying details of the modelling process. |
| nrounds | The number of rounds of boosting to be performed. |
| decision\_tree\_depth | Depth of the decision trees used in the algorithm. |
| decision | Type of decision to be used in aggregating the results– “vote” or “hard”, “prob” or “soft” |
| subset | Subset of modalities to be modelled. The format is a comma separated list of integers or NULL if no sub-setting is to be performed. |
| balance | Should the data in each modality be balanced? Useful for data containing unbalanced classes. |
| filter\_zeroes | The maximum percentage of zero values allowed in each feature. E.g. filter\_zeroes = 90 means that features containing more than 90% zero values will be excluded from the analysis. |
| filter\_missings | The maximum percentage of missing values allowed in each feature. E.g. filter\_missings = 50 means that features containing more than 50% missing values will be excluded from the analysis. |
| filter\_corr | Should the data be filtered by removing correlated values? T/F |
| filter\_var | Should the data be filtered by removing values with low variance? T/F |
| validate | Should validation be performed? T/F |

**Value**

An object of class MM\_Results containing all of the results from the model.

**Examples**

cc = config("config\_gbm\_boruta\_4C.xlsx")

pbmv = PB\_MVBoost$new(cc, nrounds = 10, decision\_tree\_depth = 2, subset = NULL)

res\_pbmv = pbmv$learn(validate = FALSE)

res\_pbmv$write("pbmv\_21")

### Multi-Modal Mixture of Experts

**Description:**

A mixture of experts model divides a complex machine learning task into multiple sub-tasks and trains a model on each sub-task. In this adaptation of the method, a model (or expert) is trained for each class in the multi-class data, using a one-vs-rest approach i.e. each expert is a binary classifier, differentiating its own class from all other classes. A gating function then learns which expert to trust and selects the best expert to predict each sample following a set of predetermined rules.

**Class:** MM\_MoE

**Usage:**

MM\_MoE$new(config, nrounds = 10, decision\_tree\_depth = 2, decision = "prob", subset = NULL, balance = FALSE, filter\_zeroes = 90.0, filter\_missings = 50.0, filter\_corr = FALSE, filter\_var = FALSE)

MM\_MoE$learn(validate = FALSE)

**Arguments**

|  |  |
| --- | --- |
| config | Configuration object, specifying details of the modelling process. |
|  |  |
|  |  |
| decision | Type of decision to be used in aggregating the results– “vote” or “hard”, “prob” or “soft” |
| subset | Subset of modalities to be modelled. The format is a comma separated list of integers or NULL if no sub-setting is to be performed. |
| balance | Should the data in each modality be balanced? Useful for data containing unbalanced classes. |
| filter\_zeroes | The maximum percentage of zero values allowed in each feature. E.g. filter\_zeroes = 90 means that features containing more than 90% zero values will be excluded from the analysis. |
| filter\_missings | The maximum percentage of missing values allowed in each feature. E.g. filter\_missings = 50 means that features containing more than 50% missing values will be excluded from the analysis. |
| filter\_corr | Should the data be filtered by removing correlated values? T/F |
| filter\_var | Should the data be filtered by removing values with low variance? T/F |
| validate | Should validation be performed? T/F |

**Value**

An object of class MM\_Results containing all of the results from the model.

**Examples**

cc = config("config\_gbm\_boruta\_4C.xlsx")

moe = MM\_MoE$new(cc, subset = NULL)

res\_moe = moe$learn(cc, validate = TRUE)

res\_moe$write("moe")

## Modelling Single Modalities

**Description:**

The multi-modal voting ensembletrains a classifier on each modality and aggregates the results using a hard or soft vote to give a final classification.

**Class:** MM\_Single

**Usage:**

MM\_Single$new(config, model\_type = "CLASSIF, predict\_type = "response", concat = FALSE, balance = FALSE, filter\_zeroes = 90.0, filter\_missings = 50.0, filter\_corr = FALSE, filter\_var = FALSE)

MM\_Single$learn(active\_learners, learner\_type, validate = FALSE)

**Arguments**

|  |  |
| --- | --- |
| config | Configuration object, specifying details of the modelling process. |
| model\_type | Type of modelling to be performed: “CLASSFN” for classification or “SURV” for survival analysis. |
| predict\_type | Type of prediction:  Classification: “response” (= labels) or “prob” (= probabilities and labels by selecting the ones with maximal probability).  Survival: “response” (= some sort of orderable risk) or “prob” (= time dependent probabilities). |
| concat | Should all the modalities be concatenated together into a single dataset? |
| subset | Subset of modalities to be modelled. The format is a comma separated list of integers or NULL if no sub-setting is to be performed. |
| balance | Should the data in each modality be balanced? Useful for data containing unbalanced classes. |
| filter\_zeroes | The maximum percentage of zero values allowed in each feature. E.g. filter\_zeroes = 90 means that features containing more than 90% zero values will be excluded from the analysis. |
| filter\_missings | The maximum percentage of missing values allowed in each feature. E.g. filter\_missings = 50 means that features containing more than 50% missing values will be excluded from the analysis. |
| filter\_corr | Should the data be filtered by removing correlated values? T/F |
| filter\_var | Should the data be filtered by removing values with low variance? T/F |
| active\_learners | A code signifying the learners and feature selectors to be applied to each modality. |
| validate | Should validation be performed? T/F |

**Value**

An object of class MM\_Results containing all of the results from the model.

**Examples**

cc = config("config\_gbm\_boruta\_4C.xlsx")

vote\_ens\_h = MM\_Voting$new(cc, decision = 'hard', subset = NULL)

res\_ens\_h = vote\_ens\_h$learn(validate = FALSE)

res\_ens\_h$write("voting\_hard")

vote\_ens\_s = MM\_Voting$new(cc, decision = 'soft', subset = NULL, concat = FALSE)

res\_ens\_s = vote\_ens\_s$learn(validate = FALSE)

res\_ens\_s$write("voting\_soft")

## Incremental Model

**Description:**

It can be the case that not all modalities of data are useful in building a model. Therefore, MOMENT provides an incremental model that determines an optimal subset of modalities for accurate model building. This model operates in one of two modes. In forward mode, one modality is added to the model at a time until the performance fails to increase significantly. In reverse mode, the model starts with all modalities and removes one at a time until the performance drops off.

**Class:** MM\_Incremental

**Usage:**

MM\_ Incremental $new(config, metric = "F1", decision = "prob", subset = NULL, balance = FALSE, filter\_zeroes = 90.0, filter\_missings = 50.0, filter\_corr = FALSE, filter\_var = FALSE)

MM\_ Incremental $ train\_incremental\_forward()

MM\_ Incremental $ train\_incremental\_reverse ()

**Arguments**

|  |  |
| --- | --- |
| config | Configuration object, specifying details of the modelling process. |
| metric | The metric to be used in comparing the performance of the modalities – “F1”, “Acc” or “AUC” |
| decision | Type of decision to be used in aggregating the results– “vote” or “hard”, “prob” or “soft”. |
| subset | Subset of modalities to be modelled. The format is a comma separated list of integers or NULL if no sub-setting is to be performed. |
| balance | Should the data in each modality be balanced? Useful for data containing unbalanced classes. |
| filter\_zeroes | The maximum percentage of zero values allowed in each feature. E.g. filter\_zeroes = 90 means that features containing more than 90% zero values will be excluded from the analysis. |
| filter\_missings | The maximum percentage of missing values allowed in each feature. E.g. filter\_missings = 50 means that features containing more than 50% missing values will be excluded from the analysis. |
| filter\_corr | Should the data be filtered by removing correlated values? T/F |
| filter\_var | Should the data be filtered by removing values with low variance? T/F |

**Value**

A list containing .

**Examples**

cc = config("config\_gbm\_boruta\_4C.xlsx")

inc = MM\_Incremental$new(cc, “F1”)

inc\_res = inc$train\_incremental\_reverse()

print(inc\_res)

## Results Returned by MOMENT

**Description:**

All of the multi-modal methods in MOMENT return an object of class MM\_Results, which contains the results of the modelling. These results include the performance measures, ROC measures and plots, features selected and stability of features selected.

**Class:** MM\_Results

**Usage:**

MM\_Results $write(result\_file\_prefix, suffix = NULL)

**Arguments**

|  |  |
| --- | --- |
| result\_file\_prefix | The prefix of the names of the result files generated. |
| suffix | An optional suffix that can be appended to the file prefix. |

**Value**

An object of class MM\_Results containing all of the results from the model.

The write function creates the following files:

|  |  |
| --- | --- |
| **File name suffix** | **Contents** |
| \_auroc.txt | The overall area under the receiver operating curve for this model. |
| \_roc.csv | Sensitivity, Specificity, F1 Score and Accuracy for each class, for each fold of the cross validation. |
| \_featsel.csv | Feature selections from each fold of the cross validation. |
| \_featsel\_aggr.csv | Feature selections aggregated across all folds. |
| \_perf.csv | Performance metrics for each fold of cross validation. |
| \_predns.csv | Predictions made by the model for each sample on each iteration of the cross validation. |
| \_plot.jpg | A boxplot showing the range of values for sensitivity, specificity, f1 score and accuracy |
| \_stab.csv | Stability metrics. Five metrics are recorded – the Jaccard Index, the Dice score, Kuncheva’s stability index, Lustgarten’s stability measure and the relative weighted consistency index. |

**Examples**

cc = config("config\_gbm\_boruta\_4C.xlsx")

vote\_ens\_h = MM\_Voting$new(cc, decision = 'hard', subset = NULL)

res\_ens\_h = vote\_ens\_h$learn(validate = FALSE)

res\_ens\_h$write("voting\_hard")