Empowering Genomic Prediction: Two Essential Tools for Democratizing the Access to Statistical Machine Learning Methods

Talk given to the Center for Quantitative Genetics and Genomics (CQGG), Aarhus University, Denmark.

Dr. Osval Antonio Montesinos López
oamontes1@ucol.mx
Facultad de Telemática
Universidad de Colima
México

April, 20, 2023.

Outline

- Introduction
- SKM library
- Book: Multivariate Statistical Machine Learning Methods for genomic prediction

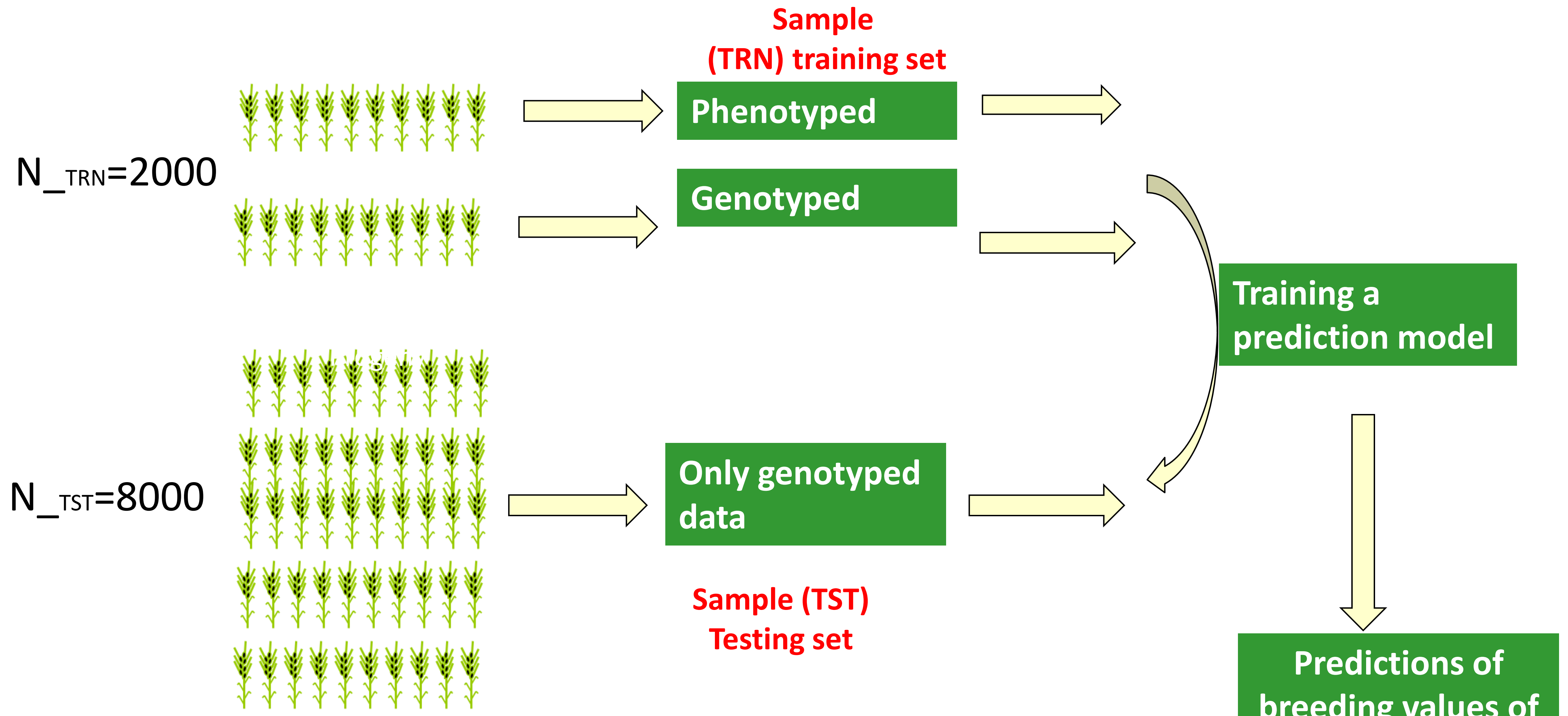
Introduction

- Genomic selection (GS) is revolutionizing plant breeding (Meuwissen, et al., 2001). But, because it is a predictive methodology, a basic understanding of statistical machine learning methods is necessary for its successful implementation.

- However, due to lack of time and appropriate training, it is difficult for breeders and scientists of related fields to learn all the fundamentals of prediction algorithms.

- Fortunately, with smart or highly automated software, it is possible for these professionals to appropriately implement any state-of-the-art statistical machine learning method.

Figure 1. Diagram about the logic of genomic selection



Fuente: Crossa et al. (2017)

SKM library

Sparse kernel methods (SKM) library

- SKM is an R library that allows us to implement seven of the most powerful state-of -the-art algorithms.

  - Bayesian methods
  - Random Forest (RF)
  - Support vector Machine (SVM)
  - Generalized linear models (GLM)
  - Grandient boosting machine (GBM)
  - Partial Least Squares (PLS)
  - Feed-forward artificial neural networks

Sparse kernel methods (SKM) library

- SKM not implement the algorithms itself but internally uses some already popular libraries for this purpose, which are known to have efficient and complete implementations of the algorithms.

- SKM use the same format for implementing 6 out of the 7 state of the art statistical machine learning algorithms.

- For algorithms with hyperparameters, SKM has an easy way to specify the conditions for hyperparameter tuning.

Sparse kernel methods (SKM) library

- Also, has many easy-to-use functions for computing metrics for evaluation of the prediction accuracy.

- Additionally provides many function for implementing many cross-validation strategies and for making summaries of prediction accuracy.

Hyperparameter tuning in statistical machine learning algorithms

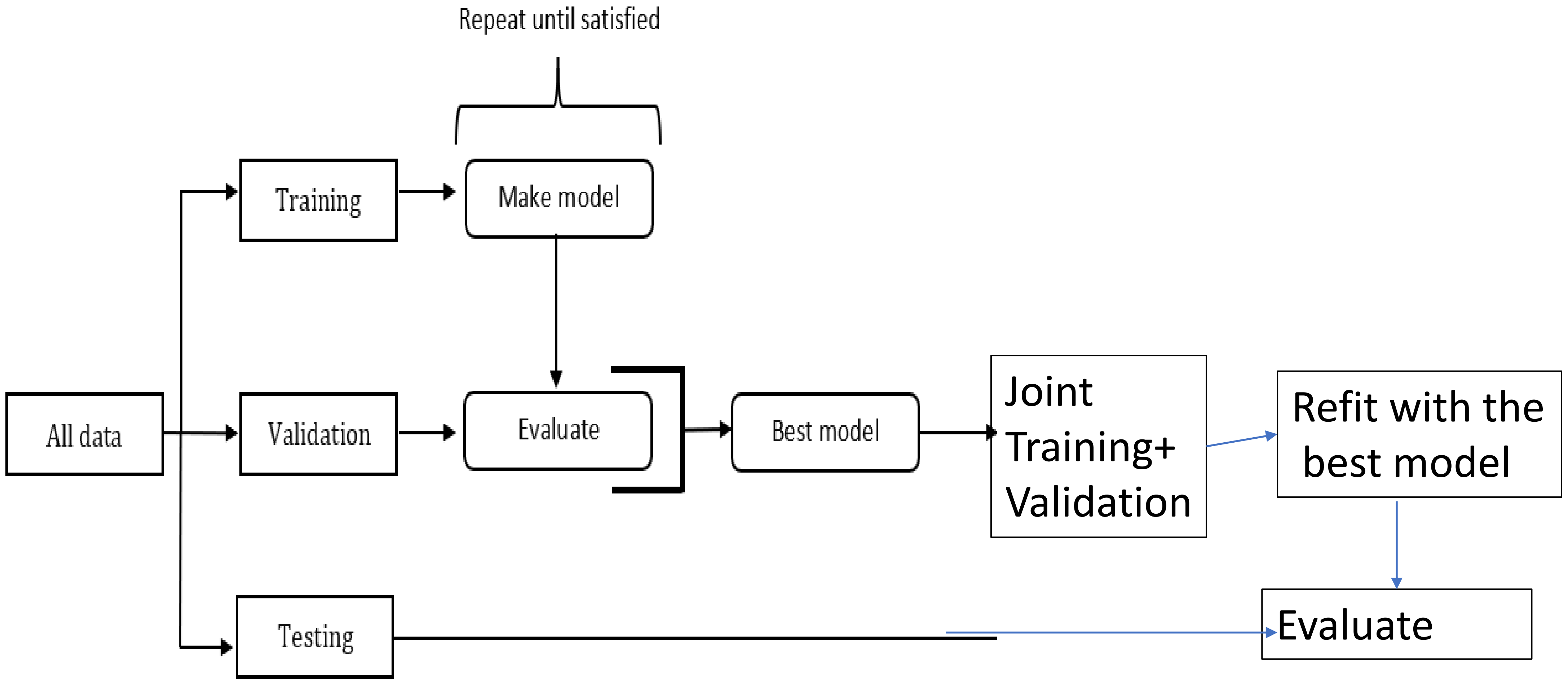


Figure 5.9. Schematic representation of the training, validation and testing sets adapted by Cook (2017).

Hyperparameter tuning in statistical machine learning algorithms

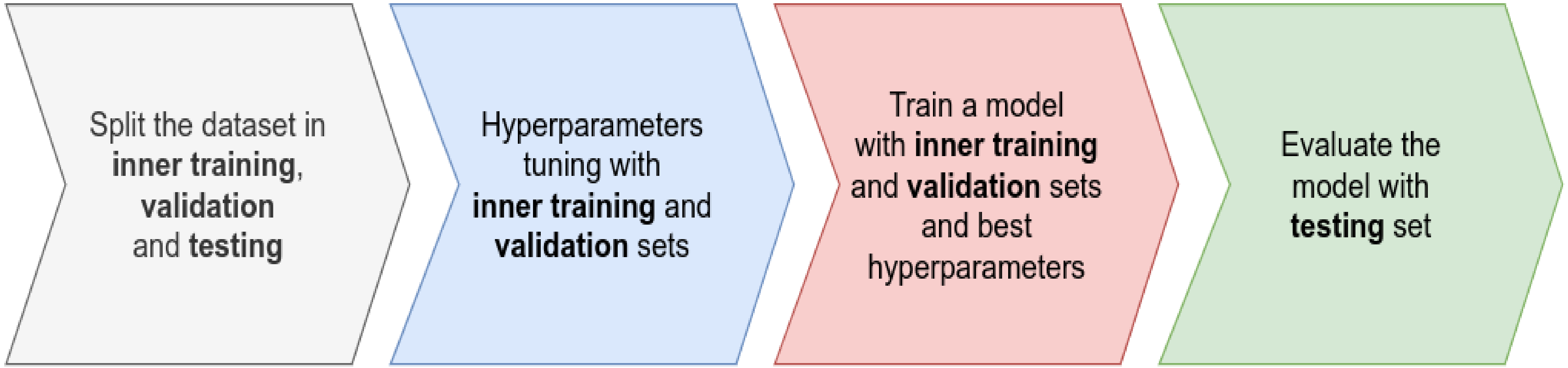

Split the dataset in **inner training**, **validation** and **testing**
Hyperparameters tuning with **inner training** and **validation** sets
Train a model with **inner training** and **validation** sets and best hyperparameters
Evaluate the model with **testing** set

Figure 1: Machine learning algorithm evaluation in one iteration of cross validation.

Installation of the SKM library

```
# Script to install all the required packages for SKM library

is_installed <- function(package) {

  return(all(package %in% rownames(installed.packages())))

}

is_windows_os <- function() {

  return(.Platform$OS.type == "windows")

}

ok <- TRUE

if (is_windows_os()) {

  if (!is_installed("installr")) {

    install.packages("installr")

  }

   ok <- install.Rtools(check = TRUE, check_r_update = FALSE)

}
```

Installation of the SKM library

```
if (ok) {
  if (!is_installed("devtools")) {
    install.packages("devtools")
  }

  devtools::install_github("gdlc/BGLR-R", upgrade = "default")
  devtools::install_github("rstudio/tensorflow", upgrade = "default")
  devtools::install_github("rstudio/keras", upgrade = "default")
  devtools::install_github("brandon-mosqueda/SKM", upgrade = "default")
}
```

Installation of the SKM library

If devtools can not be installed try:

https://cran.r-project.org/bin/windows/Rtools/rtools42/files/rtools42-5355-5357.exe

Kernel implementation on the SKM library

- One important aspect of the SKM library is the availability of kernels to capture non-linear patterns in data, giving the library its name.

- In the SKM library, you can find seven kernels that can be implemented: Linear, Polynomial, Sigmoid, Gaussian, Exponential, Arc-Cosine 1 and Arc-Cosine L (with L = 2, 3, …).

Evaluation of Prediction accuracy

Metrics for regression problems available in SKM

Mean Squared Error (`mse`),

Root Mean Squared Error (`rmse`),

Normalized Root Mean Squared Error (`nrmse`),

Mean Absolute Error (`mae`) and

Mean Arctangent Absolute Percentage Error (`maape`)

`numeric_summary()` provides all the above numeric metrics

Evaluation of Prediction accuracy

Metrics for classification problems available in SKM

Accuracy (accuracy),

precision (precision),

recall (recall),

sensitivity (sensitivity),

specificity (specificity),

Brier’s score (brier_score).

Details of most of these metrics can be found in Montesinos-López et al. (2022a)

Evaluation of Prediction accuracy

Metrics for classification problems available in SKM

F1 score (f1_score),

Kappa coefficient (kappa_coeff),

Matthews coefficient (matthews_coeff),

ROC area under the curve (roc_auc) and

precision-recall area under curve (pr_auc)

categorical_summary() provides all the above categorical metrics

Evaluation of Prediction accuracy

Specific functions for genomic prediction

gs_summaries is a function specific for the context of genomic prediction

gs_summaries expects a data frame with 5 columns: *Fold*, the fold number; *Line*, the line; *Env*, the environment; *Observed* and *Predicted*.

The type of response variable is automatically inferred so numeric or categorical metrics are computed and reported depending on that.

This function returns three different summaries computed by line, by environment and by fold.

Evaluation of Prediction accuracy

The following block of code shows how we can use the function `gs_summaries` with the predictions in the data frame generated.

```
# Errors metrics for prediction performance
Summaries <- gs_summaries(Predictions)
# Print summaries by line, environment and fold
Summaries$line
Summaries$env
Summaries$fold
```

Types of cross-validation

K-fold: cv_kfold() is the function in SKM

Stratified k-fold: cv_kfold_strata() is the name of the function in SKM

Random partition: `cv_random()` is the function to implement this cross-validation in SKM.

Random stratified partitions: `cv_random_strata()`

Types of cross-validation

Leave-one-group-out: `cv_leve_one_group_out()`

Random line: `cv_random_line()`

Missing: `cv_na()`

Hyperparameters tuning

Five algorithms in SKM are of models with hyperparameters (with the exception of bayesian_model and partial_least_squares that require hyperparameters but they are computed internally efficiently).

- `tune_type`: It can be "Grid_search" or "Bayesian_optimization".

- `tune_cv_type`: Cross validation for tuning, it can be "K_fold" or "Random" that call a random partition.

Hyperparameters tuning

```r
# Grid search
model <- SKM::random_forest(
  X_training,
  y_training,

  # Tunable hyperparameters
  trees_number = c(100, 200, 500),
  node_size = c(5, 2),
  sampled_x_vars_number = c(0.1, 0.5, 0.8),

  tune_type = "Grid_search",
  tune_cv_type = "k_fold",
  tune_folds_number = 5,
  tune_grid_proportion = 0.5,
  tune_loss_function = "mae"
)
```

Hyperparameters tuning

```r
# Bayesian optimization
model <- SKM::random_forest(
  X_training,
  y_training,

  # Tunable hyperparameters
  trees_number = list(min = 100, max = 500),
  node_size = list(min = 2, max = 5),
  sampled_x_vars_number = list(min = 0.2, max = 0.8),

  tune_type = "Bayesian_optimization",
  tune_cv_type = "k_fold",
  tune_folds_number = 5,
  tune_bayes_samples_number = 10,
  tune_bayes_iterations_number = 20,

  tune_loss_function = "mae"
)
```

Hyperparameters tuning

```r
# Show all the evaluated combinations with its loss value
model$hyperparams_grid


# Show the best combination used for fitting the final model
model$best_hyperparams
```

Bayesian models

- In SKM we refer to Bayesian models to all those models of the BGLR library (Perez & de los Campos, 2014).

- To implement these models can be used the function bayesian_model() of SKM, but, in contrast with the other six algorithms in SKM, the expected format of the predictor consist of a list with the different components of the model.

- Furthermore, instead of partitioning the data in training and testing sets, all the data is provided and the positions used for testing are set to NA or specified in a parameter included for this purpose.

Bayesian models

$$Y_{ij} = \mu + L_i + g_j + gL_{ij} + \epsilon_{ij}$$

where $L = (L_1, \dots, L_I)^T \sim \mathcal{N}_{\mathcal{J}}(\mathbf{0}, \sigma_L^2 \boldsymbol{H})$, where $\boldsymbol{H}$ denotes the environmental relationship matrix, that is, $\boldsymbol{H} = \frac{X_E X_E^T}{r}$, $\boldsymbol{g} = (g_1, \dots, g_J)^T \sim \mathcal{N}_J(\mathbf{0}, \sigma_g^2 \boldsymbol{G})$, $\boldsymbol{gL} = (gL_{11}, \dots, gL_{1J}, \dots, gL_{IJ})^T \sim \mathcal{N}_{\mathcal{J}\mathcal{J}}(\mathbf{0}, \sigma_{gL}^2 \boldsymbol{H} \odot Z_g \boldsymbol{G} Z_g^T)$, where $\boldsymbol{G}$ is the genomic relationship matrix as computed by (VanRaden, 2008), $\odot$ denotes the Hadamard product and $\boldsymbol{H}$ is the environmental relationship matrix of size I.

Bayesian models

```
X <- list(
  list(x =K_e, model = “BGLUP"),
  list(x =K_g, model = "BGBLUP"),
  list(x =K_ge, model = "BGBLUP")
)
# Sample of 20% of the observations used as testing set
testing_indices <- sample(nrow(X), nrow(X) * 0.2)

SKM::bayesian_model(
  X,
  y,
```

Bayesian models

```
 iterations_number = 100,
 burn_in = 50,
 thinning = 5,

testing_indices = testing_indices)
```

Bayesian models

Example:

Ex_S0. Univariate_Continous_GBLUP

All code are available at:

https://github.com/osval78/SKM_Genomic_Selection_Example

Random forest

Random Forest

```
# Bayesian optimization
model <- SKM::random_forest(
  X_training,
  y_training,

  # Tunable hyperparameters
  trees_number = list(min = 100, max = 500),
  node_size = list(min = 2, max = 5),
  sampled_x_vars_number = list(min = 0.2, max = 0.8),

  tune_type = "Bayesian_optimization",
  tune_cv_type = "k_fold",
  tune_folds_number = 5,
  tune_bayes_samples_number = 10,
  tune_bayes_iterations_number = 20,

  tune_loss_function = "mae"
)
```

Random Forest

Example:

Ex_S1.Univariate_Continous

Support Vector Machine

- The `support_vector_machine()` in the SKM library can be used to implement SVM models.
- For example, a SVM with radial kernel $\left(K(x_i, x_{i'}) = exp\left(-\gamma \sum_{j=1}^{p}(x_{ij} - x_{i'j})\right)\right)$ with γ a positive constant can be implemented using the following code:

Support Vector Machine

```
SKM::support_vector_machine(
  X_training, y_training,

  # Tunable hyperparameters
  kernel = "radial",
  gamma = 1/NCOL(x),
  coef0 = 0,
  cost = 1)
```

Support Vector Machine (SVM)

Example:

Ex_S2. Univariate_Binary

Generalized linear models (GLM)

Under generalized linear models, we use a set of p predictor variables and a

response variable to fit a model of the form:

Distribution: y_i is distributed among those which are considered exponential families of probability distributions (like normal, binomial, Poisson, etc.)

Linear predictor: $\eta_i = \beta_0 + X_{i1}\beta_1 + \cdots + X_{ip}\beta_p$

Link function: $\mathrm{g}(\mu_i) = \eta_i$

where y_i is the response variable for the i-th individual (sample), x_i is the predictor variable, with i=1, 2,...3, β_0 denotes and intercept term, β_i is the beta coefficient corresponding to the predictor x_i.

Generalized linear models (GLM)

The loss function that is minimized is

$$minimize\{-\ell(\boldsymbol{\beta};\boldsymbol{y}) + 0.5\lambda \sum_{j=1}^{P} \{(1-\alpha)\beta_j^2 + \alpha|\beta_j|\}$$

Where λ is the penalization parameter, when α=0 a Ridge penalization is implemented, while when α=1 a Lasso penalization is implemented and when 0<α <1 the Elastic Net penalization is implemented. Can be implementd prediction models for continuous response variables (identity link function) for binary (logit link function) for categorical (generalized logit link function) and for counts (exponential logit link function) (Friedman, Hastie, & Tibshirani, 2010).

Generalized linear models (GLM)

- Under the SKM library for continuous response variables, it is possible to train multi-trait models.

- The function to implement all these models is the generalized_linear_model() function.

- The required parameters for this function are shown in the following block of code:

Generalized linear models (GLM)

```
SKM::generalized_linear_model(
   X_training, y_training,

  # Tunable hyperparameters
  alpha = 1, ###Lasso Regression

  # Tune configuration parameters
  tune_type = "Grid_search",
  tune_folds_number = 5,
  tune_grid_proportion = 1,
  tune_bayes_samples_number = 10,
  tune_bayes_iterations_number = 10,
```

Generalized linear models (GLM)

```
# Other algorithm's parameters
  lambdas_number = 100,

  # Seed for reproducible results
  seed = NULL,
  verbose = TRUE
)
```

Generalized linear models (GLM)

Example:

Ex_S3. Univariate_Categorical

Gradiente boosting machine

The following block of code provides the specification of `generalized_boosted_machine()` function:

```
SKM::generalized_boosted_machine(
  X_training,  y_training,
  # Tunable hyperparameters
  trees_number = 500,
  max_depth = 1,
  node_size = 10,
  shrinkage = 0.1,
  sampled_records_proportion = 0.5)
```

Gradiente boosting machine

Example

Ex_S7.Univariate_Continous_GBM

Partial Least Squares (PLS)

- Partial Least Square (PLS) regression is one of the most popular in biological sciences, because it can model complex biological events, it is flexible for considering different factors, and it is unaffected by data collinearity.

- For this reason, authors suggest that the PLS is a potentially valuable method for modeling high-dimensional biological data (as derived from genomics, proteomics and peptidomics) (Palermo et al., 2009).

- PLS can model multiple responses, while efficiently dealing with multicollinearity.

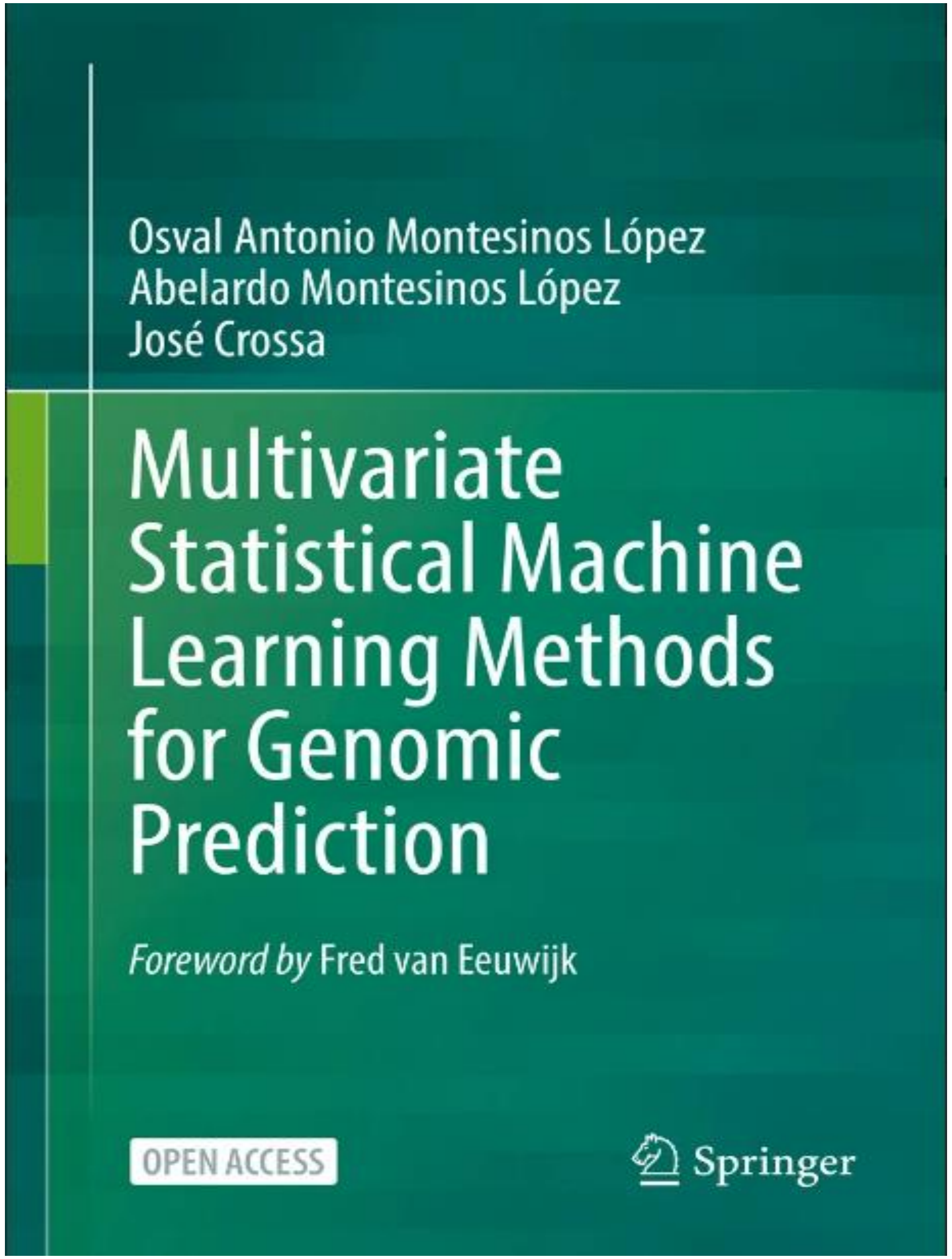
Partial Least Squares (PLS)

Example:

Ex_S8.Univariate_Continous_PLS

Book

State-of-the-art material for genomic prediction



Osval Antonio Montesinos López
Abelardo Montesinos López
José Crossa

Multivariate Statistical Machine Learning Methods for Genomic Prediction

Foreword by Fred van Eeuwijk

OPEN ACCESS

Springer

Open Source: https://link.springer.com/book/10.1007/978-3-030-89010-0#toc

Conclusions

Due to ghe fact that the GS is a predictive methodology still is challenging its implementation in many breeding programs since many factors affect its accuracy.

For this reason, a lot of research is being conducted to improve some of the factors that affects its accuracy.

However, as pointed out before statistical machine learning methods are key for the sucessful implementation of the GS methodology.

Conclusions

The SKM library provides an opportunity to utilize various statistical machine learning techniques for Genomic prediction without the need for extensive expertise in statistics and programming.

Additionally, SKM offers the advantage of incorporating other types of omics data, such as genomic, phenotypic and other omics information, which has been shown to enhance predictive accuracy based on empirical evidence.

Conclusions

In order to democratize the implementation of genomic prediction, there is a need for software that is easier to use, more automatic, and open source.

The book provides fundamental concepts, illustrative examples, and open-source code for various statistical machine learning methods that are useful for genomic selection.

Furthermore, this document contains most of the state-of-the-art algorithms utilized in genomic prediction, presenting a comprehensive resource for researchers in the field.

Questions ?

Thanks!

REFERENCES

Montesinos-López A, Montesinos-López OA, Montesinos-López JC, Flores-Cortes CA, de la Rosa R, Crossa J. (2021). A guide for kernel generalized regression methods for genomic-enabled prediction. Heredity (Edinb). 126(4):577-596. doi: 10.1038/s41437-021-00412-1.

Montesinos-López O. A., Montesinos-López A., Crossa, J. (2022a). Multivariate Statistical Machine Learning methods for genomic prediction. Springer Nature, Switzerland, ISBN: 978-3-030-89012-4.

Montesinos-López, O.A., Mosqueda-González, B.A., Palafox-González, A., Montesinos-López, A. and Crossa, J. (2022). A General-Purpose Machine Learning R Library for Sparse Kernels Methods With an Application for Genome-Based Prediction. Front. Genet. 13:887643. doi: 10.3389/fgene.2022.887643