Grupo de Usuarios de R de Sevilla

# Selección de variables en Machine Learning

Miguel Pavón Barrera

2 Abril 2019, 19 a 20:30 h

Sala TIC4, CRAI Reina Mercedes

http://bit.ly/SevillaRmeetup

R consortium

DSC::Energy Analytics

# INTRODUCCIÓN

- OBJETIVO
- PRINCIPALES MÉTODOS
- IMPLEMENTACIÓN EN R

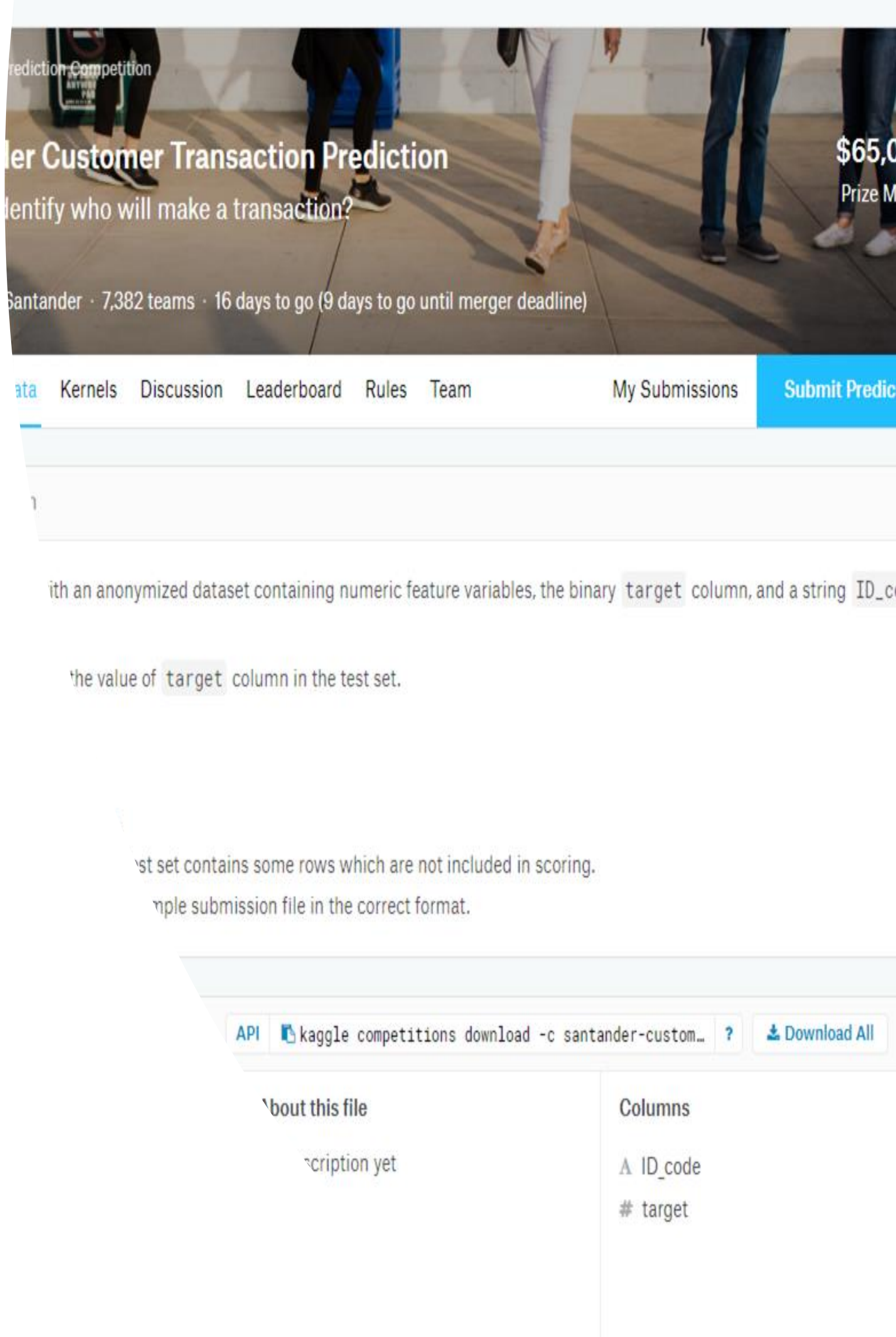
# CONJUNTO DE PRUEBAS

- DATASET TRAIN.CSV
- 200 000 OBSERVACIONES
- 200 VARIABLES PREDICTORAS
- PROBLEMA DE CLASIFICACIÓN
- AÑADIMOS 60 VARIABLES RANDOM
- PARTICIONAMOS EN TRAIN Y TEST

**SCORE TEST:** 0,9630

- DATASET TRAIN.CSV
- 1460 OBSERVACIONES
- 80 VARIABLES PREDICTORAS
- PROBLEMA DE REGRESIÓN
- AÑADIMOS 10 VARIABLES RANDOM

**SCORE VAL:** 0.1314
**SCORE TRAIN:** 0.0743



# TIPOS DE VARIABLES

- RELEVANTES
- REDUNDANTES
- IRRELEVANTES

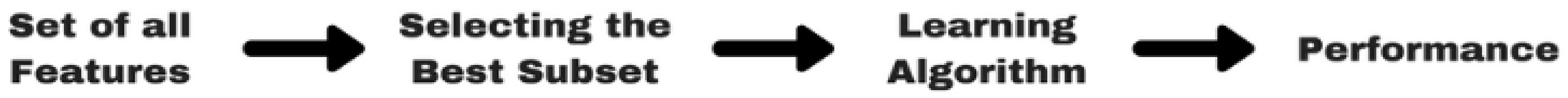
# CLASIFICACIÓN MÉTODOS

- **FILTER METHOD** – METODOS FILTRO
- **WRAPPER METHOD** – METODOS ENVOLTURA
- **EMBEDDED METHOD** – MÉTODOS EMPOTRADOS

# FILTER METHOD - CARACTERÍSTICAS

- **NO** SE BASAN EN ALGORITMOS DE **MACHINE LEARNING**
- MIDEN LA **RELACIÓN** ENTRE **ATRIBUTOS** Y **VARIABLE RESPUESTA**

- **SIMPLES Y RÁPIDOS COMPUTACIONALMENTE**
- **INDEPENDIENTES DEL CLASIFICADOR**




- **NO DETECTAN ATRIBUTOS REDUNDANTES**
- **IGNORAN LA INTERACCIÓN ENTRE ATRIBUTOS**

# FILTER METHOD – PRINCIPALES TÉCNICAS

- CORRELACIÓN
- ANOVA
- CHI.CUADRADO
- ENTROPY METHODS

| Feature\Response | Continuous | Categorical |
| --- | --- | --- |
| Continuous | Pearson's Correlation | LDA |
| Categorical | Anova | Chi-Square |

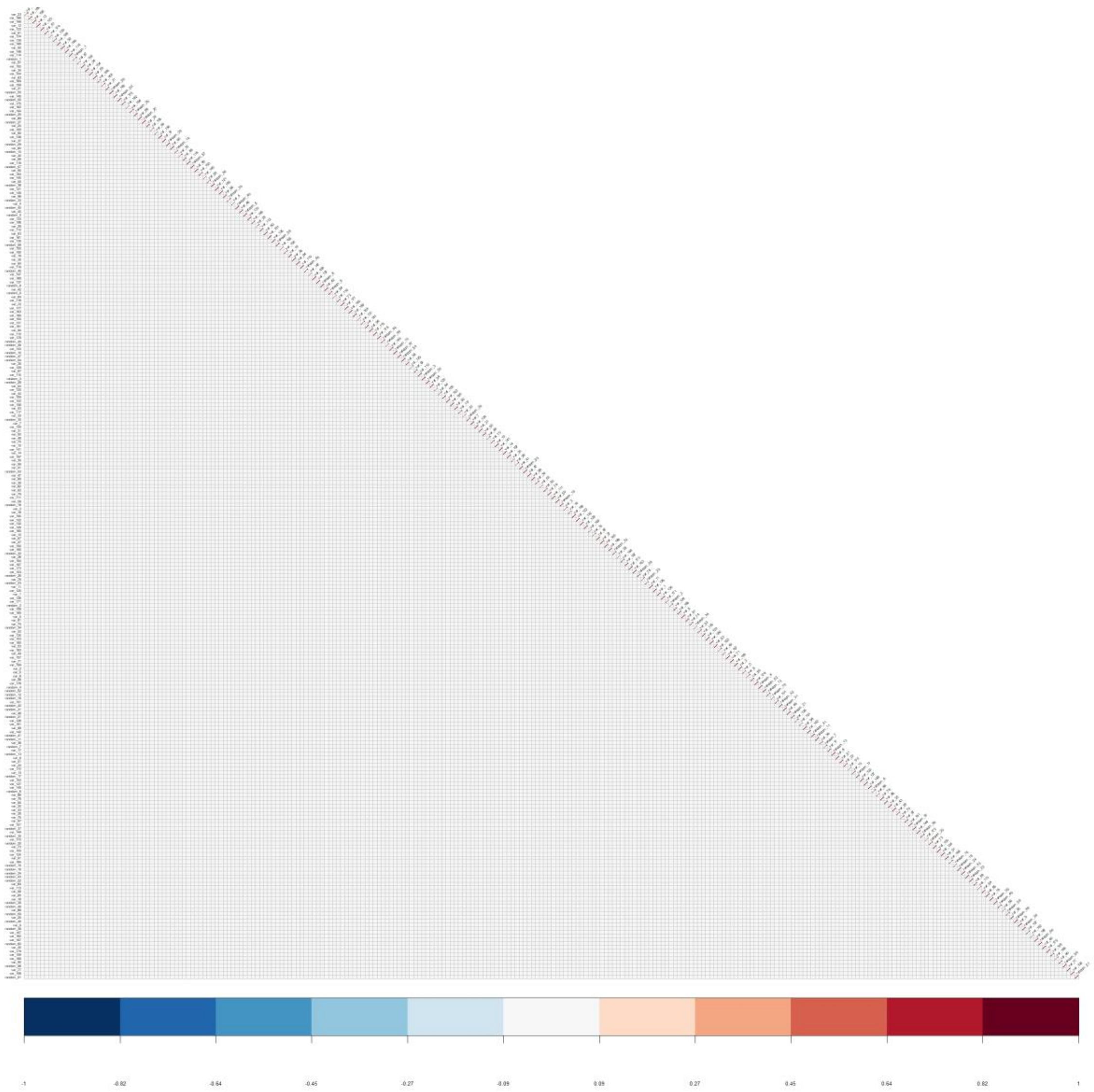
# FILTER METHOD – CORRELACIÓN LINEAL

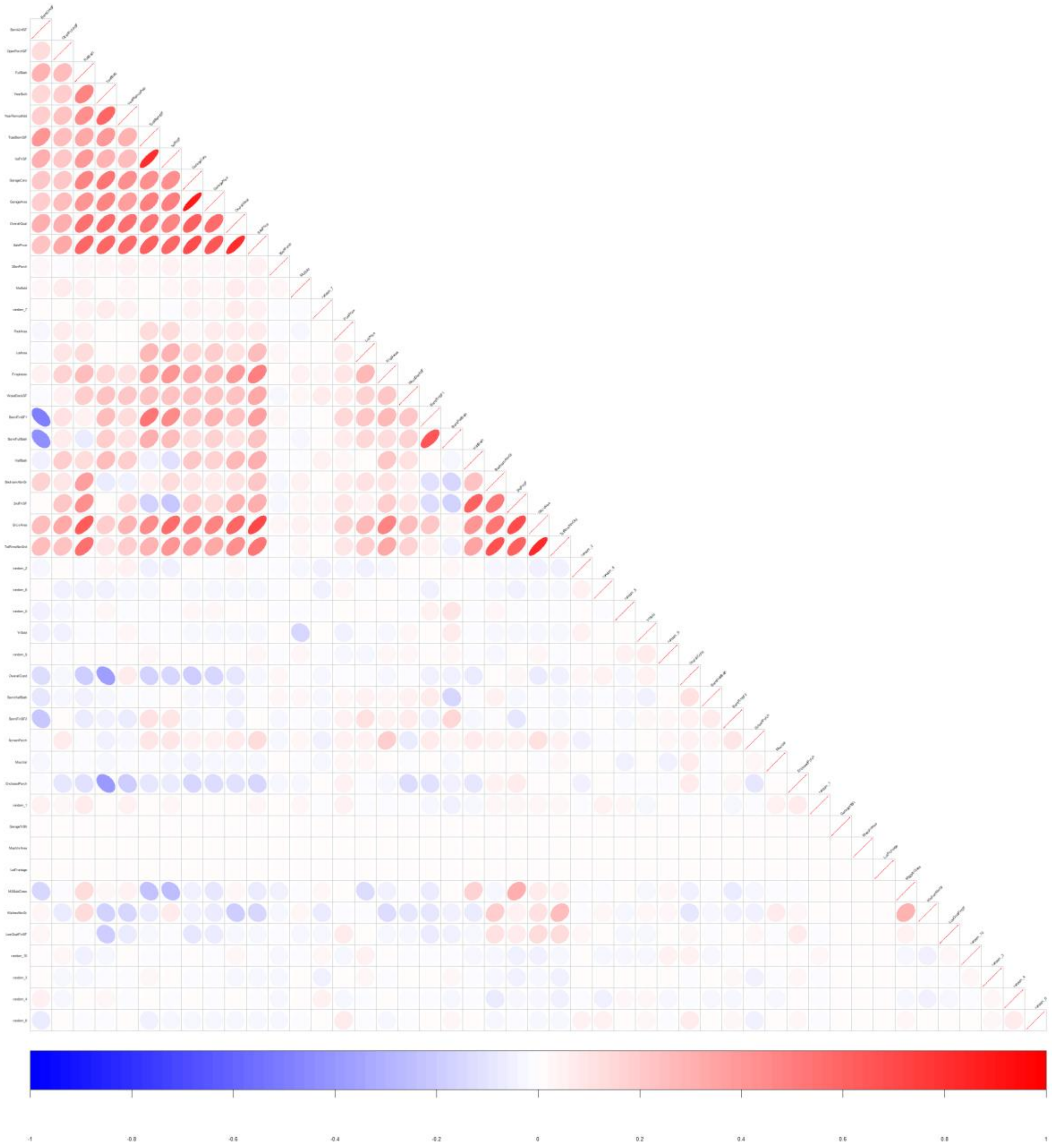
- Mide **dependencia lineal** entre predictores continuos.

Métodos Multivariantes:

- **Método CFS** (CorrelationbasedFeatureSelection)
- **FCBF** (FastCorrelation-BasedFilter)

# Correlación

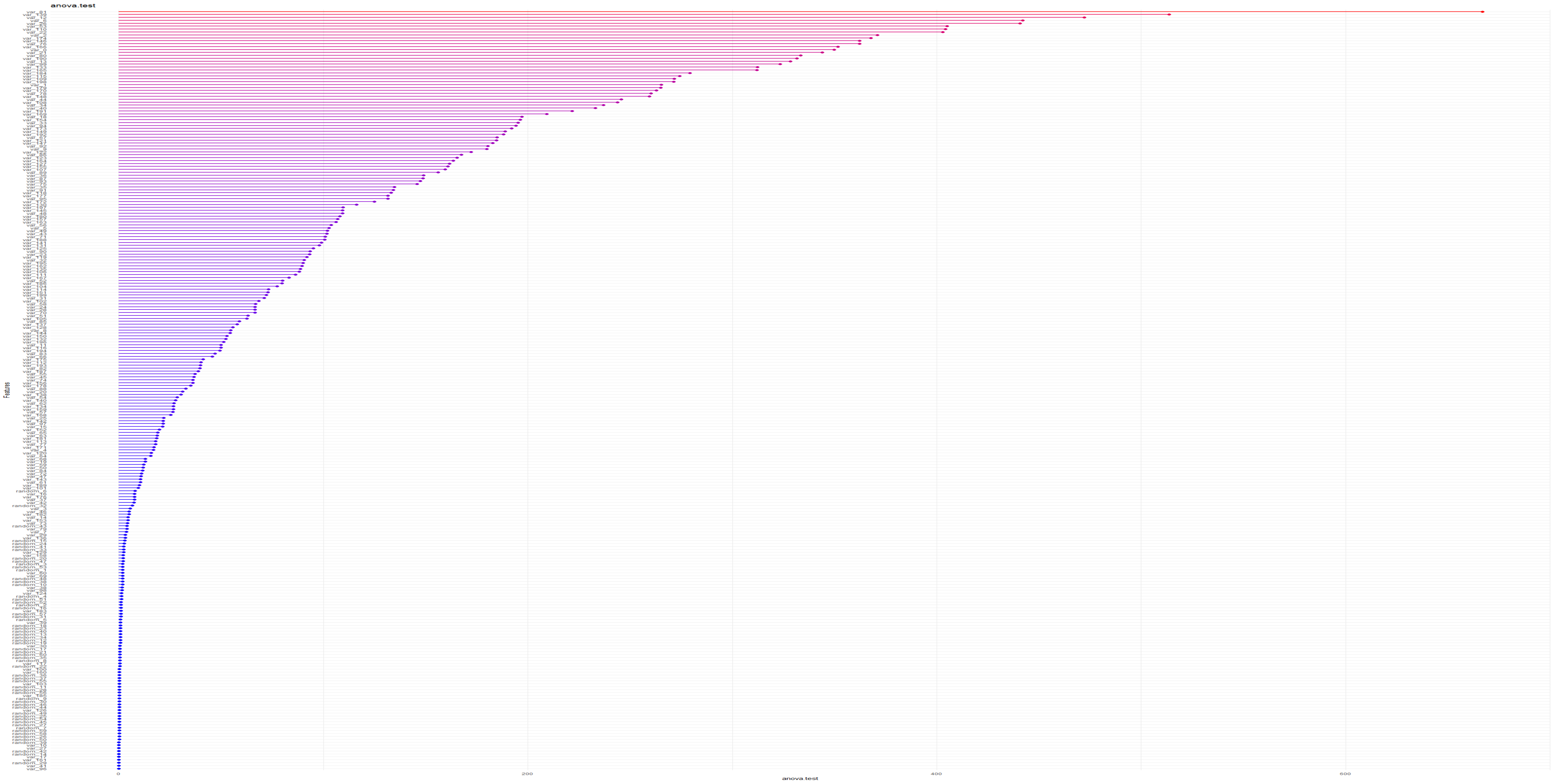




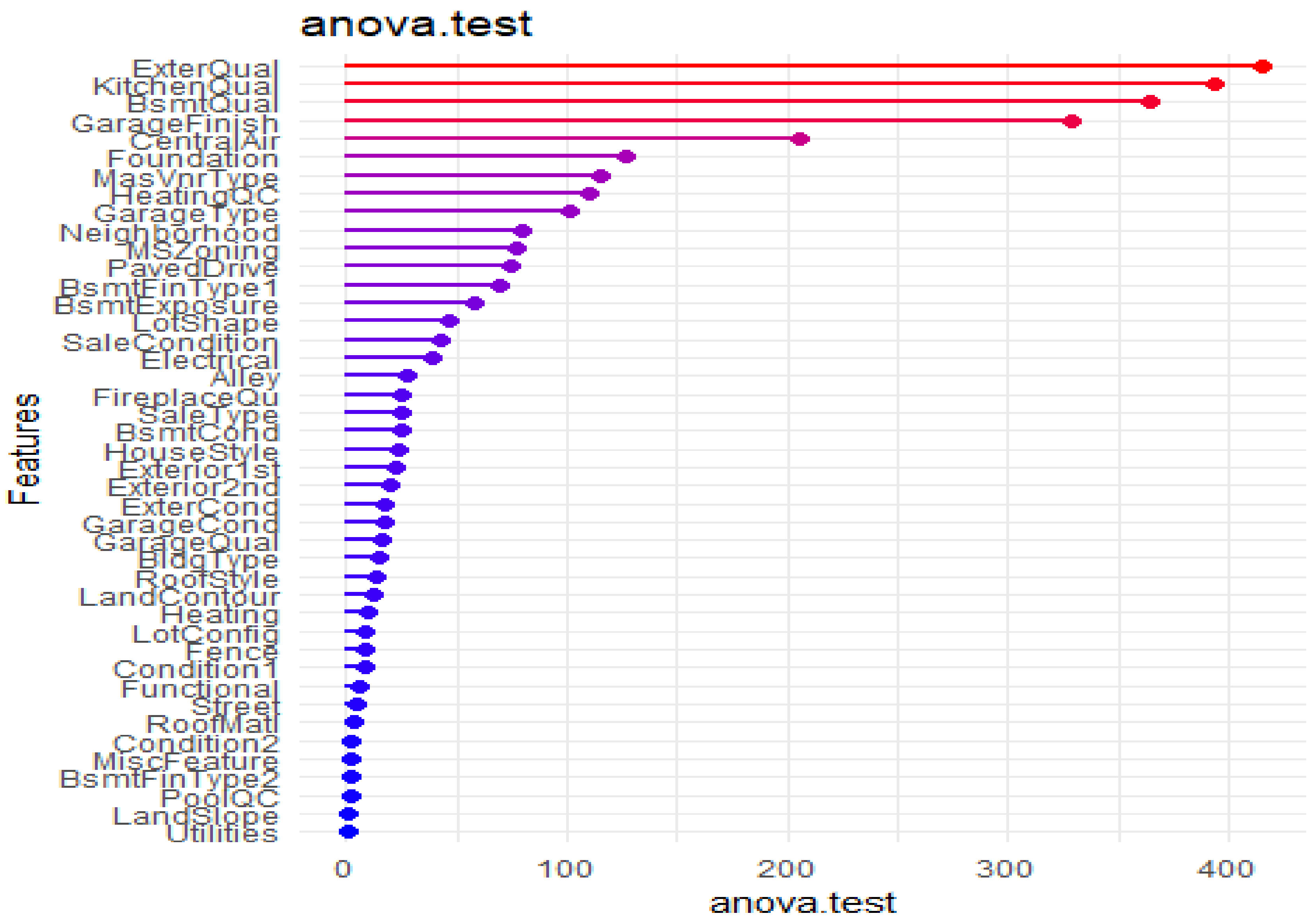
# FILTER METHOD - ANOVA

- ESTADÍSTICO F
- REFLEJA EL **GRADO DE PARECIDO** EXISTENTE ENTRE LAS MEDIAS DE UNA **VARIABLE CATEGÓRICA SOBRE UNA CONTINUA**
- SI F>>1 ENTONCES HAY MAYOR EVIDENCIA
- **SE NECESITA HIPÓTESIS PARA REALIZAR CONTRASTE**

# ESTADISTICO F - SANTANDER



# ESTADISTICO F - HP



# FILTER METHOD - ENTROPÍA

- **MEDIDA DE INCERTIDUMBRE** DE UNA VARIABLE ALEATORIA

$$H(X) = -\sum_i P(x_i) \log P(x_i)$$

**Entropía de $X$ después de observar otra variable Y:**

$$H(X \mid Y) = -\sum_j P(y_j) \sum_i P(x_i | y_j) \log P(x_i | y_j)$$

# FILTER METHOD - ENTROPÍA

### Details

`information.gain` is

$$H(Class) + H(Attribute) - H(Class, Attribute)$$

.

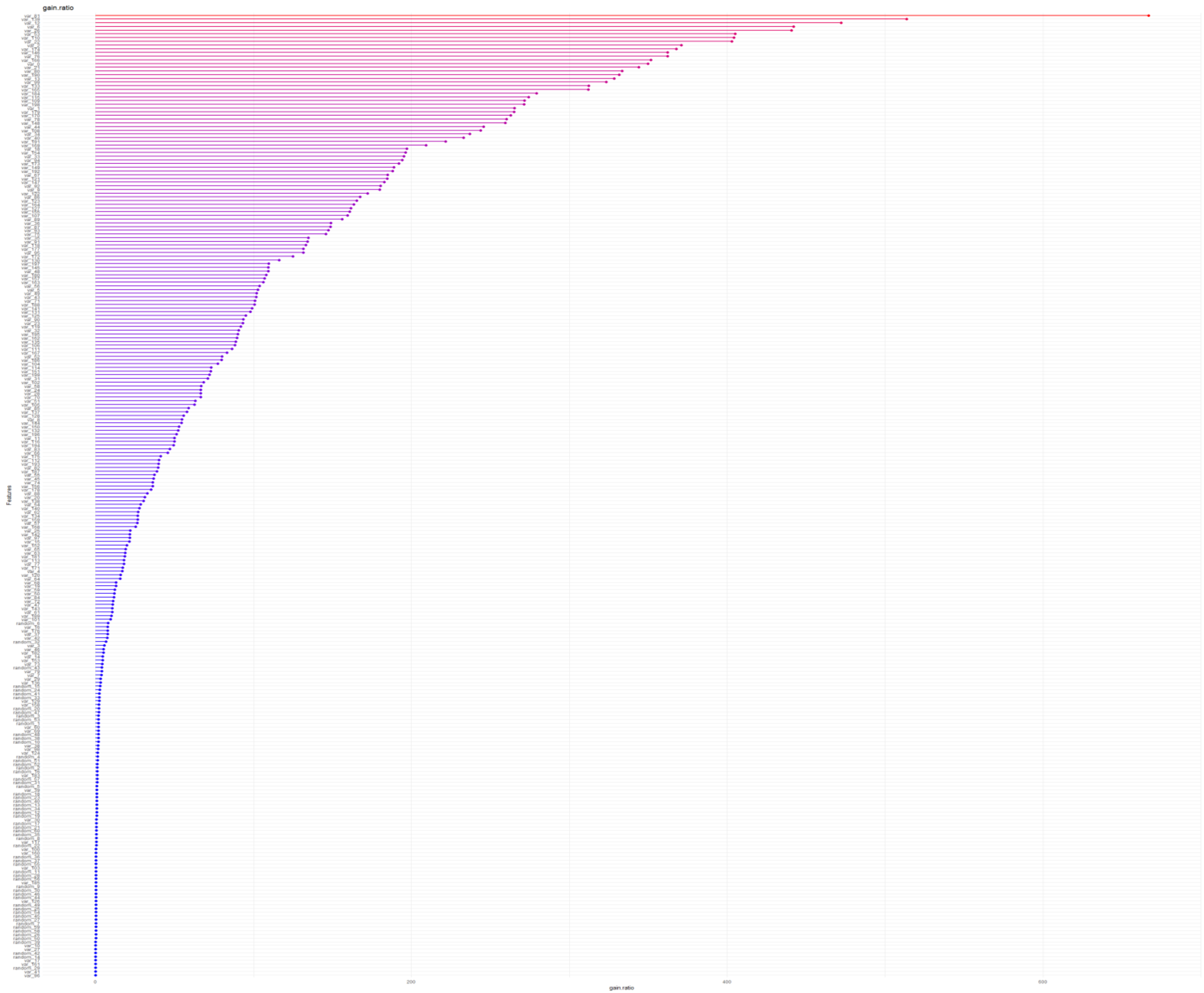
`gain.ratio` is

$$\frac{H(Class) + H(Attribute) - H(Class, Attribute)}{H(Attribute)}$$

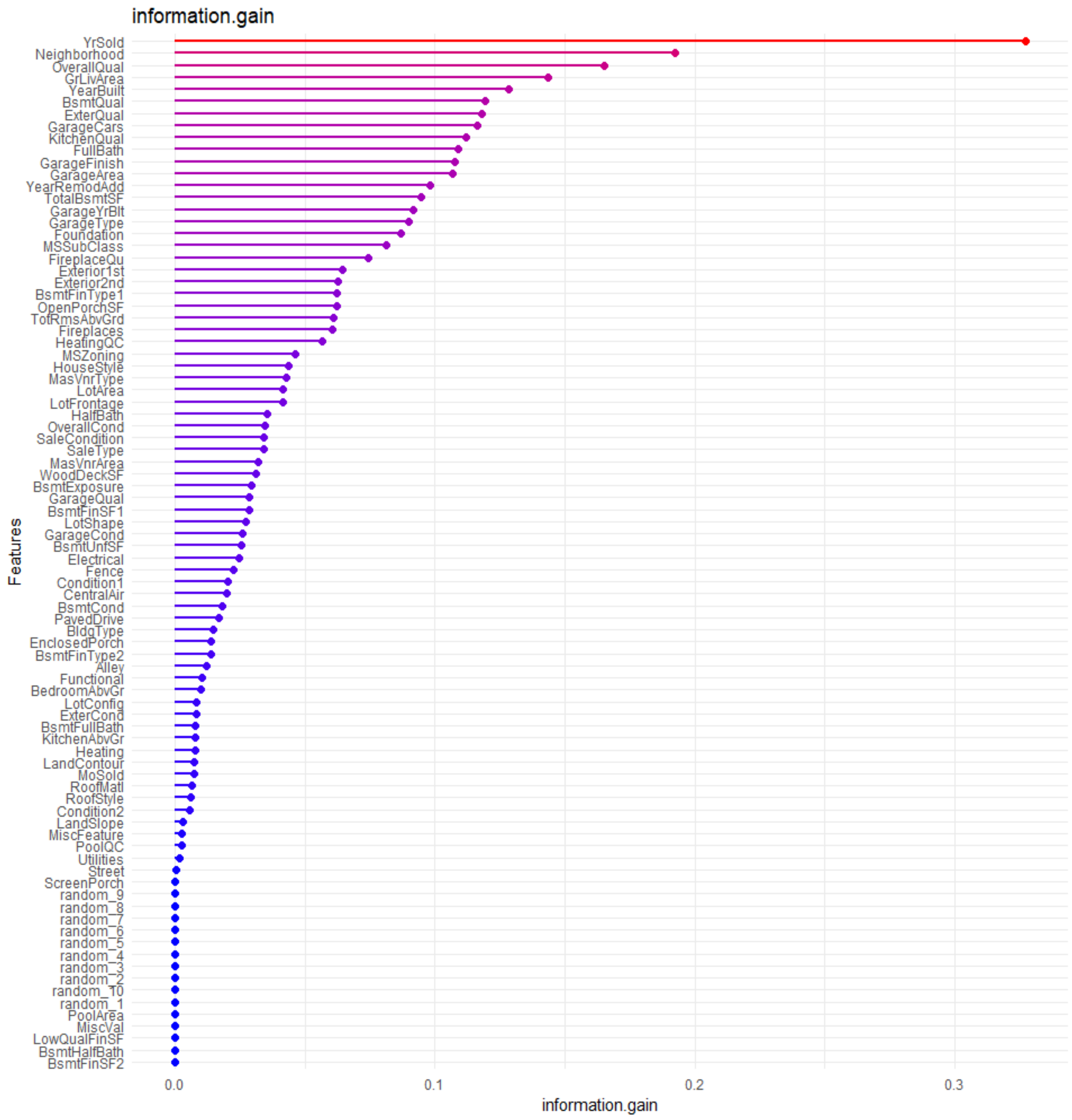
`symmetrical.uncertainty` is

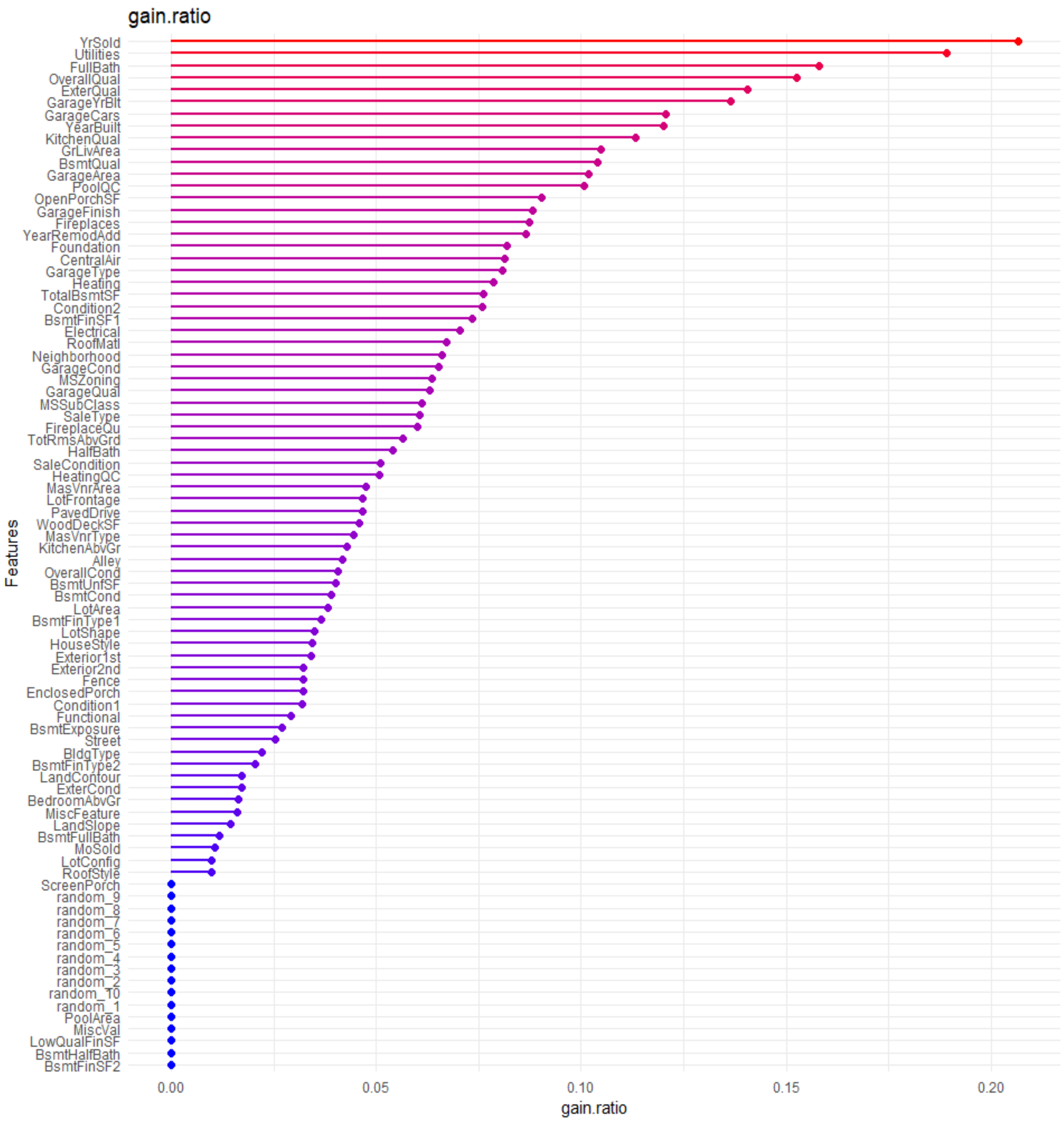
$$2\frac{H(Class) + H(Attribute) - H(Class, Attribute)}{H(Attribute) + H(Class)}$$

# GAIN RATIO - SANTNADER



# GAIN - HP


information.gain
Features
information.gain
0.0
0.1
0.2
0.3


gain.ratio
Features
gain.ratio
0.00
0.05
0.10
0.15
0.20

# FILTER METHOD – IMPLEMENTACIÓN R

- **LIBRERÍA MRL**

```
generateFilterValuesData(
            task   = ________ ,
            method = ________ )
```

**makeClassifTask** `(data, target)`

**makeMultilabelTask** `(data, target)`

**makeRegrTask** `(data, target)`

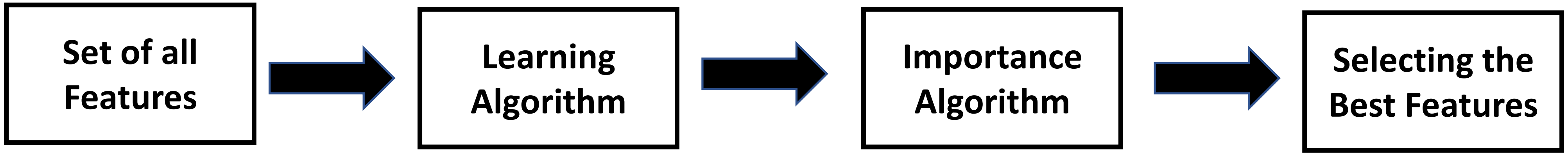
**listFilterMethods( )**

| | id | package | desc |
|---|---|---|---|
| 1 | anova.test | | ANOVA Test for binary and multiclass classification tasks |
| 2 | auc | | AUC filter for binary classification tasks |
| 3 | carscore | care | CAR scores |
| 4 | cforest.importance | party | Permutation importance of random forest fitted in package '... |
| 5 | chi.squared | FSelector | Chi-squared statistic of independence between feature and t... |
| 6 | gain.ratio | FSelector | Entropy-based gain ratio between feature and target |
| 7 | information.gain | FSelector | Entropy-based information gain between feature and target |
| 8 | kruskal.test | | Kruskal Test for binary and multiclass classification tasks |
| 9 | linear.correlation | | Pearson correlation between feature and target |
| 10 | mrmr | mRMRe | Minimum redundancy, maximum relevance filter |
| 11 | oneR | FSelector | oneR association rule |
| 12 | permutation.importance | | Aggregated difference between feature permuted and unpe... |
| 13 | randomForest.importance | randomForest | Importance based on OOB-accuracy or node inpurity of ran... |
| 14 | randomForestSRC.rfsrc | randomForestS... | Importance of random forests fitted in package 'randomFor... |
| 15 | randomForestSRC.var.select | randomForestS... | Minimal depth of / variable hunting via method var.select on... |
| 16 | ranger.impurity | ranger | Variable importance based on ranger impurity importance |
| 17 | ranger.permutation | ranger | Variable importance based on ranger permutation importan... |
| 18 | rank.correlation | | Spearman's correlation between feature and target |
| 19 | relief | FSelector | RELIEF algorithm |
| 20 | symmetrical.uncertainty | FSelector | Entropy-based symmetrical uncertainty between feature and... |
| 21 | univariate.model.score | | Resamples an mlr learner for each input feature individually.... |
| 22 | variance | | A simple variance filter |

| | id | package | desc |
|---|---|---|---|
| 1 | anova.test | | ANOVA Test for binary and multiclass classification tasks |
| 2 | auc | | AUC filter for binary classification tasks |
| 3 | carscore | care | CAR scores |
| 4 | cforest.importance | party | Permutation importance of random forest fitted in package '... |
| 5 | chi.squared | FSelector | Chi-squared statistic of independence between feature and t... |
| 6 | gain.ratio | FSelector | Entropy-based gain ratio between feature and target |
| 7 | information.gain | FSelector | Entropy-based information gain between feature and target |
| 8 | kruskal.test | | Kruskal Test for binary and multiclass classification tasks |
| 9 | linear.correlation | | Pearson correlation between feature and target |
| 10 | mrmr | mRMRe | Minimum redundancy, maximum relevance filter |
| 11 | oneR | FSelector | oneR association rule |
| 12 | permutation.importance | | Aggregated difference between feature permuted and unpe... |
| 13 | randomForest.importance | randomForest | Importance based on OOB-accuracy or node inpurity of ran... |
| 14 | randomForestSRC.rfsrc | randomForestS... | Importance of random forests fitted in package 'randomFor... |
| 15 | randomForestSRC.var.select | randomForestS... | Minimal depth of / variable hunting via method var.select on... |
| 16 | ranger.impurity | ranger | Variable importance based on ranger impurity importance |
| 17 | ranger.permutation | ranger | Variable importance based on ranger permutation importan... |
| 18 | rank.correlation | | Spearman's correlation between feature and target |
| 19 | relief | FSelector | RELIEF algorithm |
| 20 | symmetrical.uncertainty | FSelector | Entropy-based symmetrical uncertainty between feature and... |
| 21 | univariate.model.score | | Resamples an mlr learner for each input feature individually.... |
| 22 | variance | | A simple variance filter |

# EMBEDDED METHOD – CARACTERÍSTICAS

- SE BASAN EN ALGORITMOS DE **MACHINE LEARNING**
- ASIGNAN UN GRADO DE IMPORTANCIA A CADA VARIABLE
- RELACIONADO CON LA INTERPRETABILIDAD DE LOS MODELOS



# EMBEDDED – PRINCIPALES TÉCNICAS

- REGRESIÓN RIDGE
- REGRESIÓN LASSO
- SVM IMPORTANCE
- RANDOMFOREST IMPORTANCE
- XGBOOST IMPORTANCE
- LIGTHGBM IMPORTANCE
- ETC

# EMBEDDED METHOD – REGRESIÓN LASSO

- **REGRESIÓN LINEAL CON RESTRICCIONES**

$$\min_{\beta_0,\beta} \left\{ \frac{1}{N} \sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \beta)^2 \right\} \boxed{\text{subject to } \sum_{j=1}^{p} |\beta_j| \leq t.} \Rightarrow \min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} ||y - X\beta||_2^2 \boxed{+ \lambda ||\beta||_1} \right\}$$

- **λ** PARÁMETRO **REGULARIZACIÓN/PENALIZACIÓN**

- **IDEA:** variar **λ** y comparar la **DEVIANCE**

```
coef(cvfit, s = "lambda.min")
```

# EMBEDDED METHOD – REGRESIÓN LASSO

**glmnet(** x, y,

family = {'binomial', 'Gaussian' ,etc}

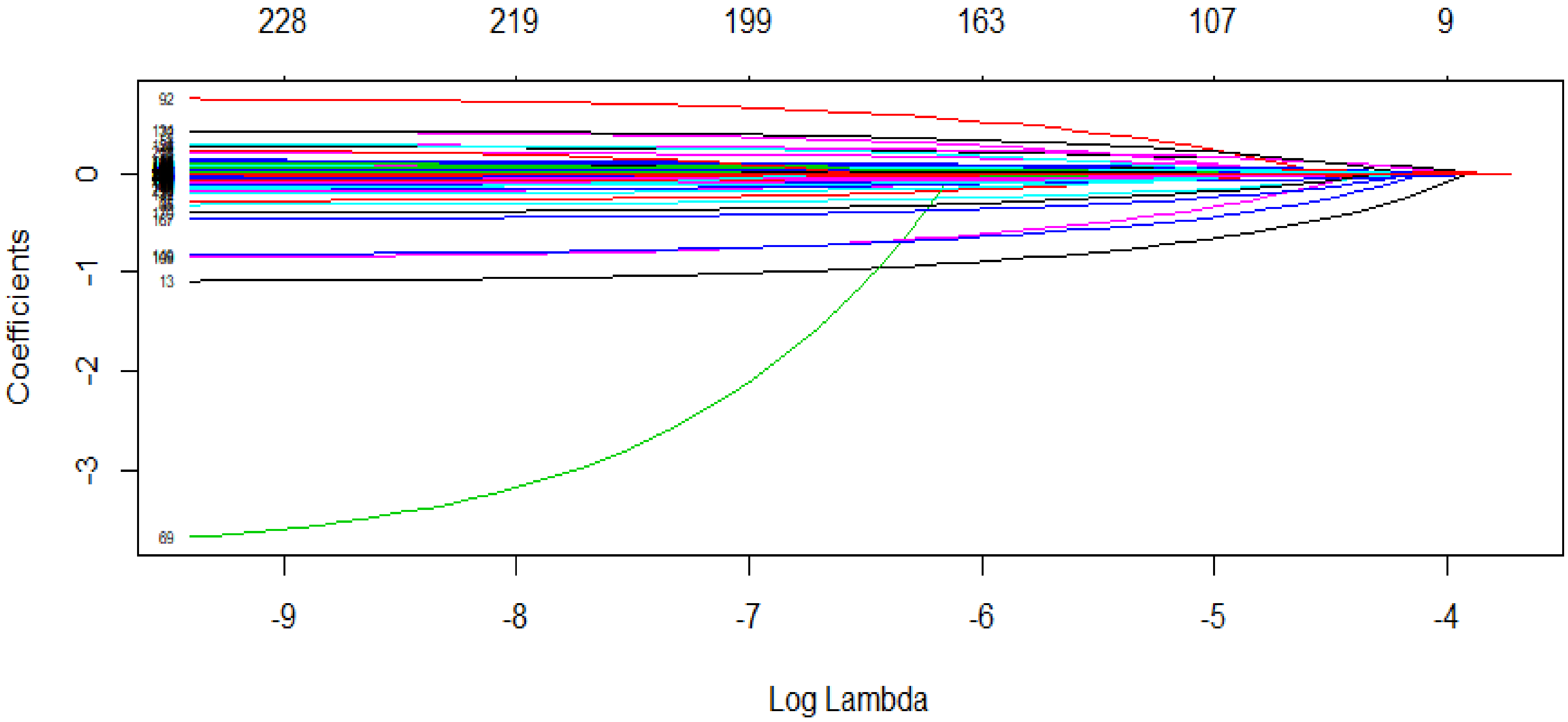
alpha = 1

standardize = {T,F}

nlambda = { 100 , 200 … } **)**

**glmnet.cv( )**

# LASSO PLOT



228
219
199
163
107
9
92
13
69
0
-1
-2
-3
Coefficients
-9
-8
-7
-6
-5
-4
Log Lambda

# EMBEDDED – RANDOMFOREST IMP.

- Random Forest construye CART de forma paralela
- Cada árbol **MINIMIZA** función de **IMPUREZA**

| Impurity | Task | Formula | Description |
|---|---|---|---|
| Gini impurity | Classification | $\sum_{i=1}^{C} -f_i(1-f_i)$ | $f_i$ is the frequency of label $i$ at a node and $C$ is the number of unique labels. |
| Entropy | Classification | $\sum_{i=1}^{C} -f_i \log(f_i)$ | $f_i$ is the frequency of label $i$ at a node and $C$ is the number of unique labels. |
| Variance / Mean Square Error (MSE) | Regression | $\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$ | $y_i$ is label for an instance, $N$ is the number of instances and $\mu$ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$ |

- **OOB** – Conjunto de Validación

# EMBEDDED – RANDOMFOREST IMP.

- **MEDIA DECREASE IMPURITY**

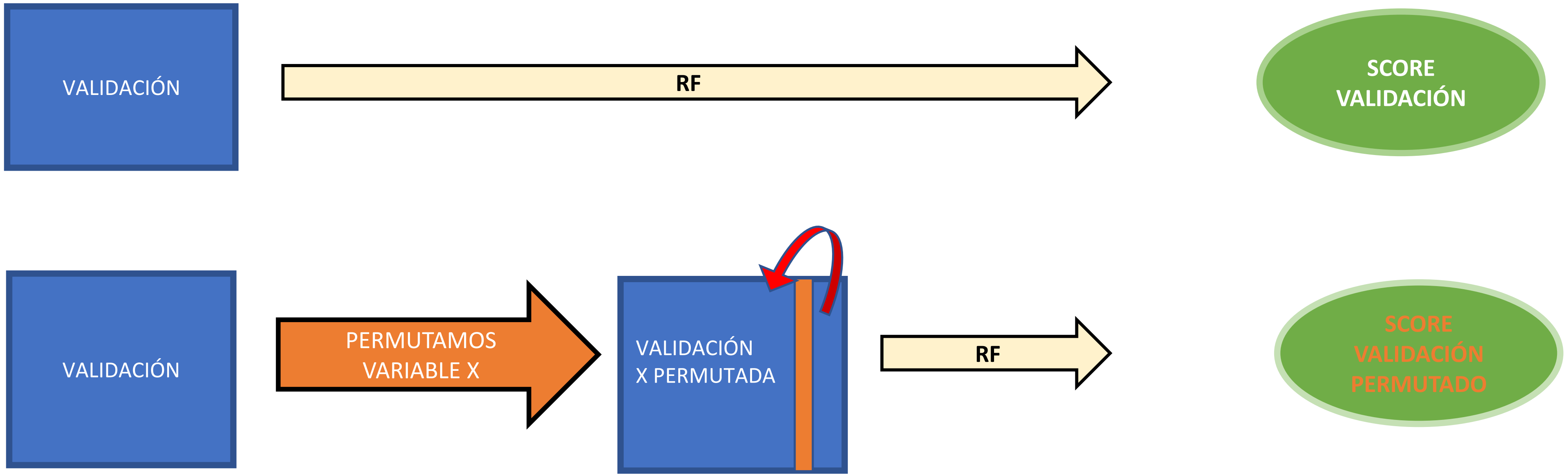
  - Mide cuánta **“información”** aporta una variable

  - Gini Importance, Information Importance…

  - **SESGADO para VARIABLES con ALTA CARDINALIDAD**

# EMBEDDED – RANDOMFOREST IMP.

- **PERMUTATION IMPORTANCE RF**




**IMPORTANCIA VARIABLE X** = SCORE VALIDACION – SCORE VALIDACION PERMUTADO

# EMBEDDED – RANDOMFOREST IMP.

- **PERMUTATION IMPORTANCE**

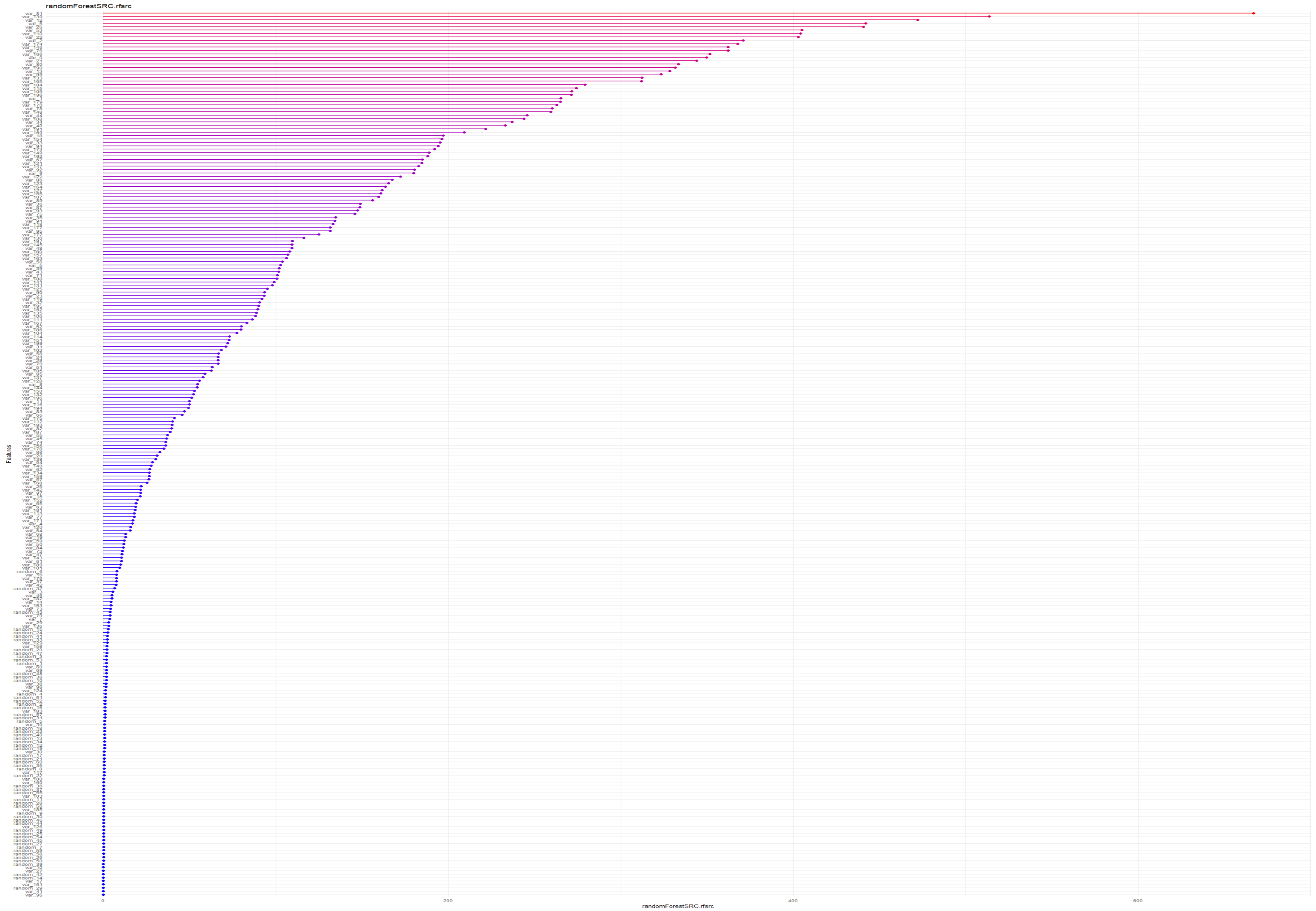
  **-** Descenso de validación al permutar variable

  \- RF validación OOB

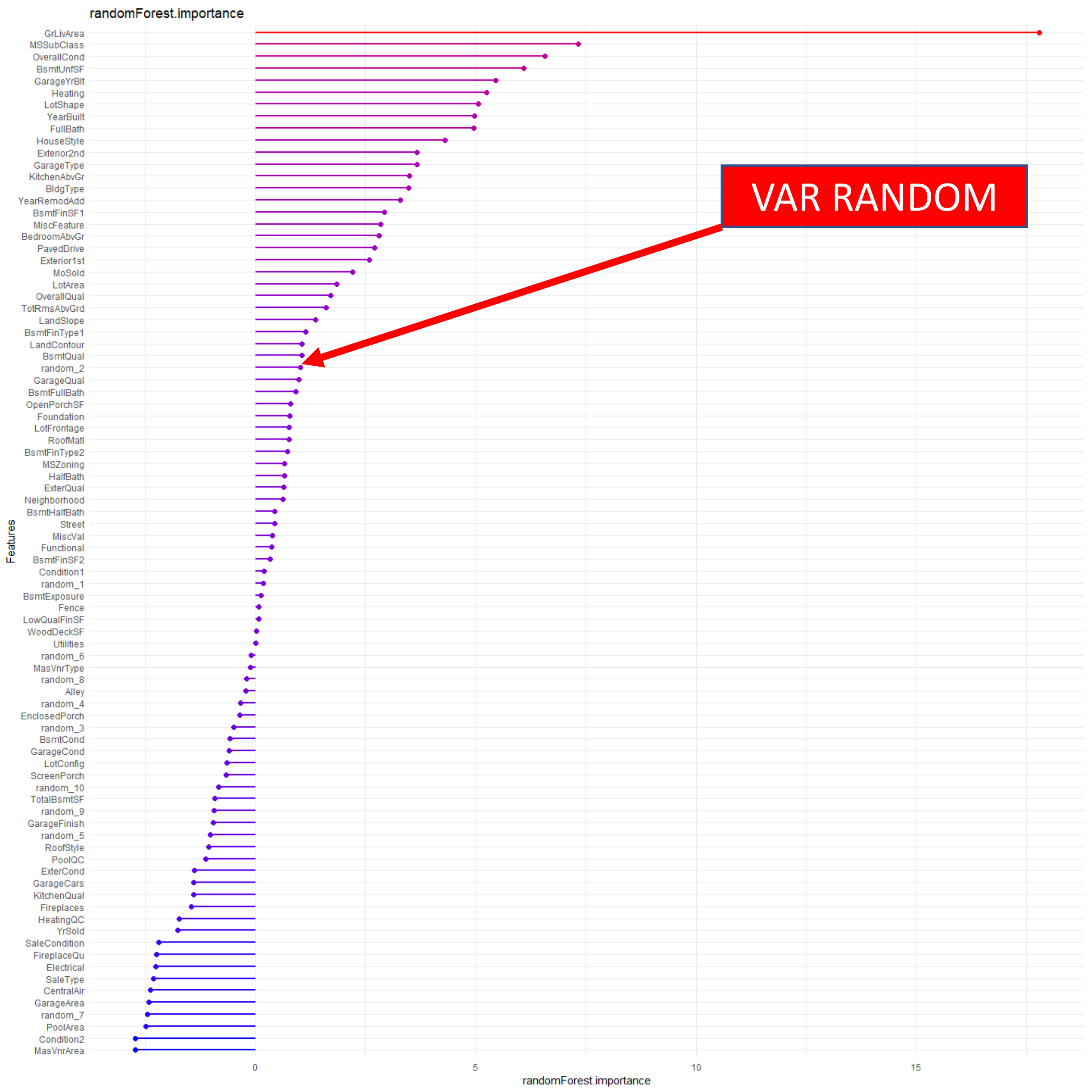
  **- Mayor COSTE COMPUTACIONAL**

  **- SESGADO CORRELACIONADAS**

# RF PERMUTE IMPORTANCE - SANTANDER



# RF PERMUTE IMPORTANCE - HP



# RF IMPORTANCIA - IMPLEMENTACIÓN

<u>randomForest library</u>

1) **randomForest(**formula,
data ,
**importance = TRUE** , __ )
2) **importance(** RF , type = {1,2} **)**

<u>randomForestRSC library</u>

1) **rfsrc(**formula,
data ,
**importance = TRUE** , __ )
2) **vimp(** RF , xvar.names **)**

**generateFilterValuesData(** task.train , **method = {…}** )

| | id | package | desc |
|---|---|---|---|
| 1 | cforest.importance | party | Permutation im |
| 2 | randomForest.importance | randomForest | Importance ba |
| 3 | randomForestSRC.rfsrc | randomForestSRC | Importance of |
| 4 | randomForestSRC.var.select | randomForestSRC | Minimal depth |

# EMBEDDED – XGBOOST IMP.

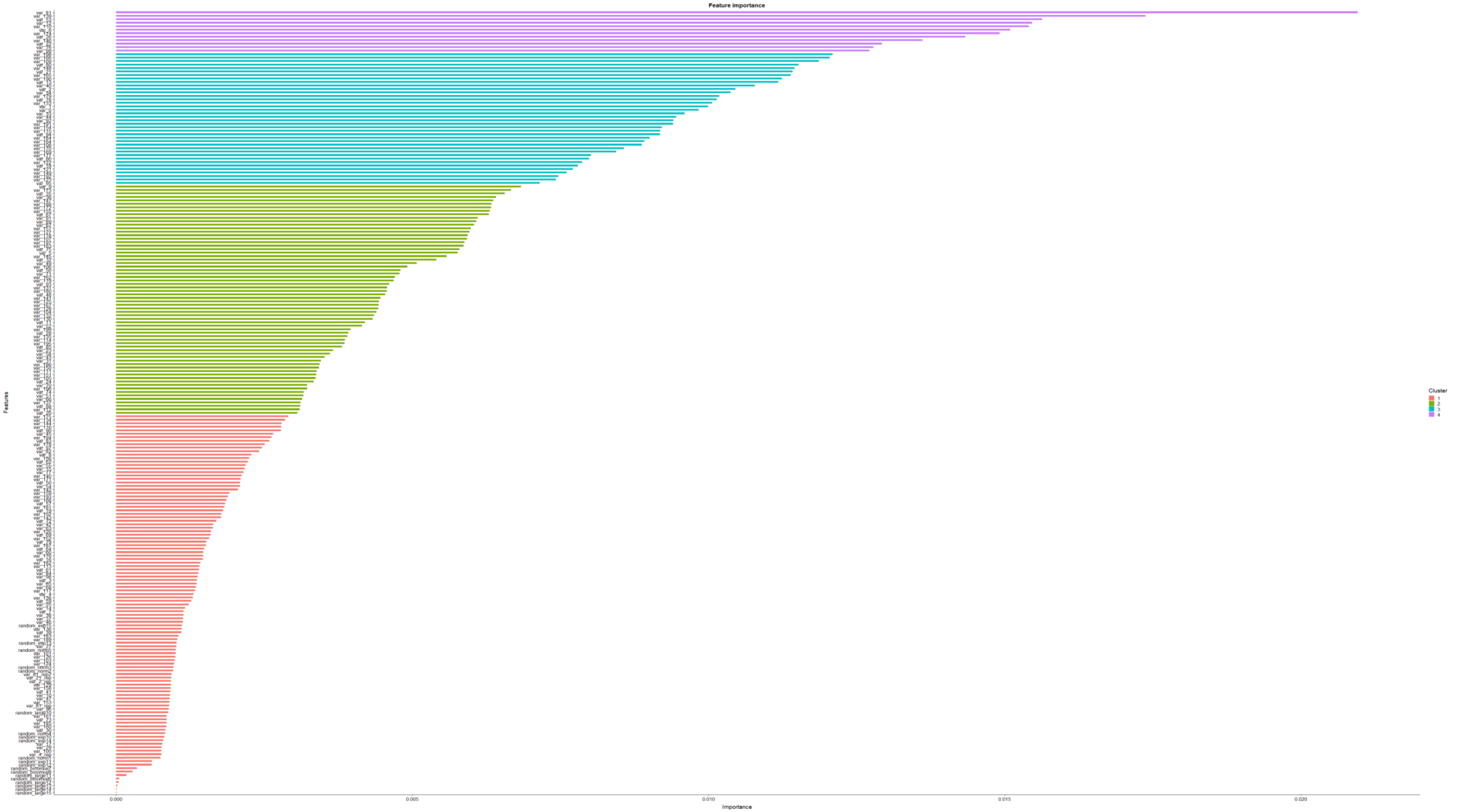
- XGB construye CART de forma secuencial
- **GAIN IMPORTANCE**
- **FRECUENCY IMPORTANCE**
- **COVER IMPORTANCE**

# XGB IMPORTANCIA - IMPLEMENTACIÓN

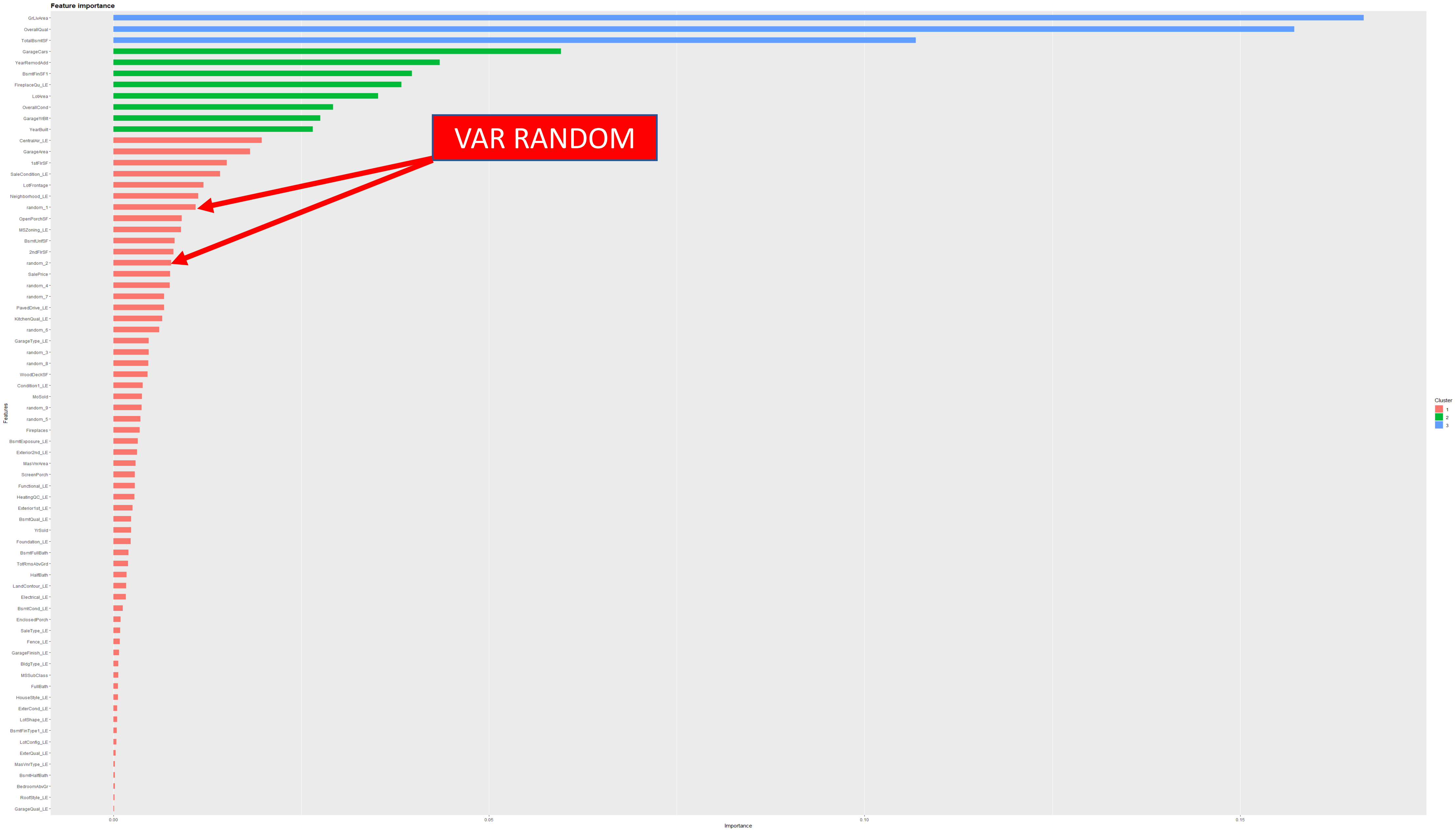
- **xgb.importance(** feature_names, xgb.model **)**

- **xgb.ggplot.importance(** xgb.importance , measure = {**‘Gain’** ,
  **‘Cover’** ,
  **‘Frequency’** **)**

# XGB GAIN - SANTANDER



# XGB GAIN - HP



Feature importance
GrLivArea
OverallQual
TotalBsmtSF
GarageCars
YearRemodAdd
BsmtFinSF1
FireplaceQu_LE
LotArea
OverallCond
GarageYrBlt
YearBuilt
CentralAir_LE
GarageArea
1stFlrSF
SaleCondition_LE
LotFrontage
Neighborhood_LE
random_1
OpenPorchSF
MSZoning_LE
BsmtUnfSF
2ndFlrSF
random_2
SalePrice
random_4
random_7
PavedDrive_LE
KitchenQual_LE
random_6
GarageType_LE
random_3
random_8
WoodDeckSF
Condition1_LE
MoSold
random_9
random_5
Fireplaces
BsmtExposure_LE
Exterior2nd_LE
MasVnrArea
ScreenPorch
Functional_LE
HeatingQC_LE
Exterior1st_LE
BsmtQual_LE
YrSold
Foundation_LE
BsmtFullBath
TotRmsAbvGrd
HalfBath
LandContour_LE
Electrical_LE
BsmtCond_LE
EnclosedPorch
SaleType_LE
Fence_LE
GarageFinish_LE
BldgType_LE
MSSubClass
FullBath
HouseStyle_LE
ExterCond_LE
LotShape_LE
BsmtFinType1_LE
LotConfig_LE
ExterQual_LE
MasVnrType_LE
BsmtHalfBath
BedroomAbvGr
RoofStyle_LE
GarageQual_LE
Features
0.00
0.05
0.10
0.15
Importance
Cluster
1
2
3
VAR RANDOM

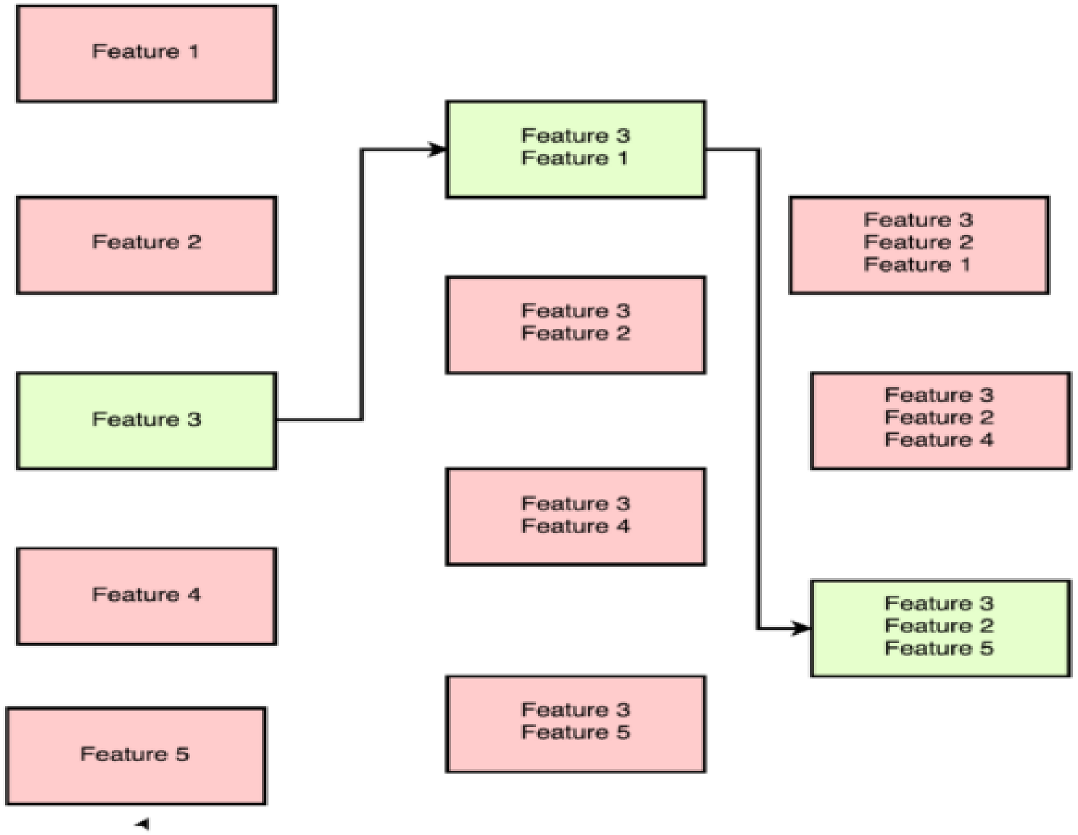
# WRAPPER METHOD - CARACTERÍSTICAS

- SE BASAN EN ALGORITMOS DE **MACHINE LEARNING**
- **BUSCAN MEJOR SUBCONJUNTO OPTIMICE MODELO**
- **COMPUTACIONALMENTE COSTOSOS**
- **OVERFITTING**

# WRAPPER METHOD - TÉCNICAS

- **FORDWARD**
- **BACKWARD**
- **GENETIC**
- **BORUTA**

# WRAPPER METHOD - FORDWARD



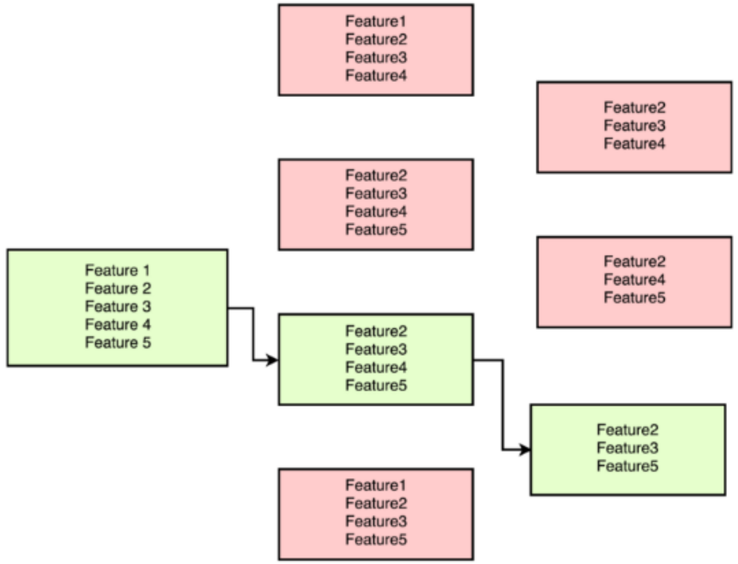
# WRAPPER METHOD - BACKWARD

- **RECURSIVE ELIMINATION**

- CARET ⟶ **rfe**( x , y , control , sises , metric , params )

**Algoritmo 1:** Recursive Feature Elimination

Entrenar el modelo con todos los atributos;
Evaluar el rendimiento del modelo;
Ordenar los atributos de acuerdo a una medida de importancia;
**foreach** $S_i$ $(i = 1, 2, \ldots)$ **do**
    Retener los $S_i$ atributos más importantes y eliminar los restantes;
    Entrenar el modelo usando los $S_i$ atributos;
    Evaluar el rendimiento del modelo;
    Recalcular la medida de importancia de los atributos [Opcional];
**end**
Ordenar los valores $S_i$ de acuerdo a su rendimiento;
A partir de la lista anterior, elegir un valor $S_i$ apropiado;
Considerar el modelo correspondiente a los $S_i$ atributos elegidos;



# WRAPPER METHOD - GENETIC

**Algoritmo 3:** Algoritmo genético para selección de atributos

Elegir el criterio de parada, el tamaño de la población ($m$), el número de hijos de cada generación ($GenSize$) y la probabilidad de mutación ($p_m$);

Hacer $p$=Número de atributos;

Generar aleatoriamente un conjunto inicial de $m$ cromosomas binarios, cada uno de longitud $p$;

**repeat**

    **foreach** *cromosoma* **do**

        Entrenar un modelo y calcular el rendimiento del cromosoma;

    **end**

    **foreach** $k = 1, \ldots, GenSize/2$ **do**

        Seleccionar dos cromosomas basándose en su rendimiento;

        **Cruzamiento:** seleccionar una posición aleatoria en el cromosoma e intercambiar los genes a partir de ese punto;

        **Mutación:** Alterar aleatoriamente, con probabilidad $p_m$, los valores binarios de cada gen en cada cromosoma de la descendencia;

    **end**

**until** *se cumpla el criterio de parada*;

# WRAPPER METHOD – IMPLEMENTACIÓN R

**selectFeatures(** control, learner , task ,resampling , measures = list( ) **)**

**SelControlExhautive(** max.features )
**FeatSelControlSequential(** method = { “sfs”, “sbs” } , alpha = 0.02)
**FeatSelControlRandom( )**
**FeatSelControlGA( )**

**makeLearner(** cl ="classif.xgboost", parametros **)** | **listLearners(**obj = NA**)**

**makeResampleDesc(** {‘CV’ , `LOO`} , iters = 5 , stratify = T**)**

**makeClassifTask(** data , task **)**

*measures:* https://mlr.mlr-org.com/articles/tutorial/measures.html

# WRAPPER METHOD - BORUTA

TRAIN + TRAIN PERMUTADO → RF IMPORTANCE

- Zi IMPORTANCIA VARIABLE i
- Z* MÁXIMA IMPORTANCIA VARIABLE SOMBRA

CONTRASTE HIPÓTESIS

$Zi < Z^*$

- **Zi << Z*** Atributo i Eliminado
- **Z >> Z*** Atributo Importante

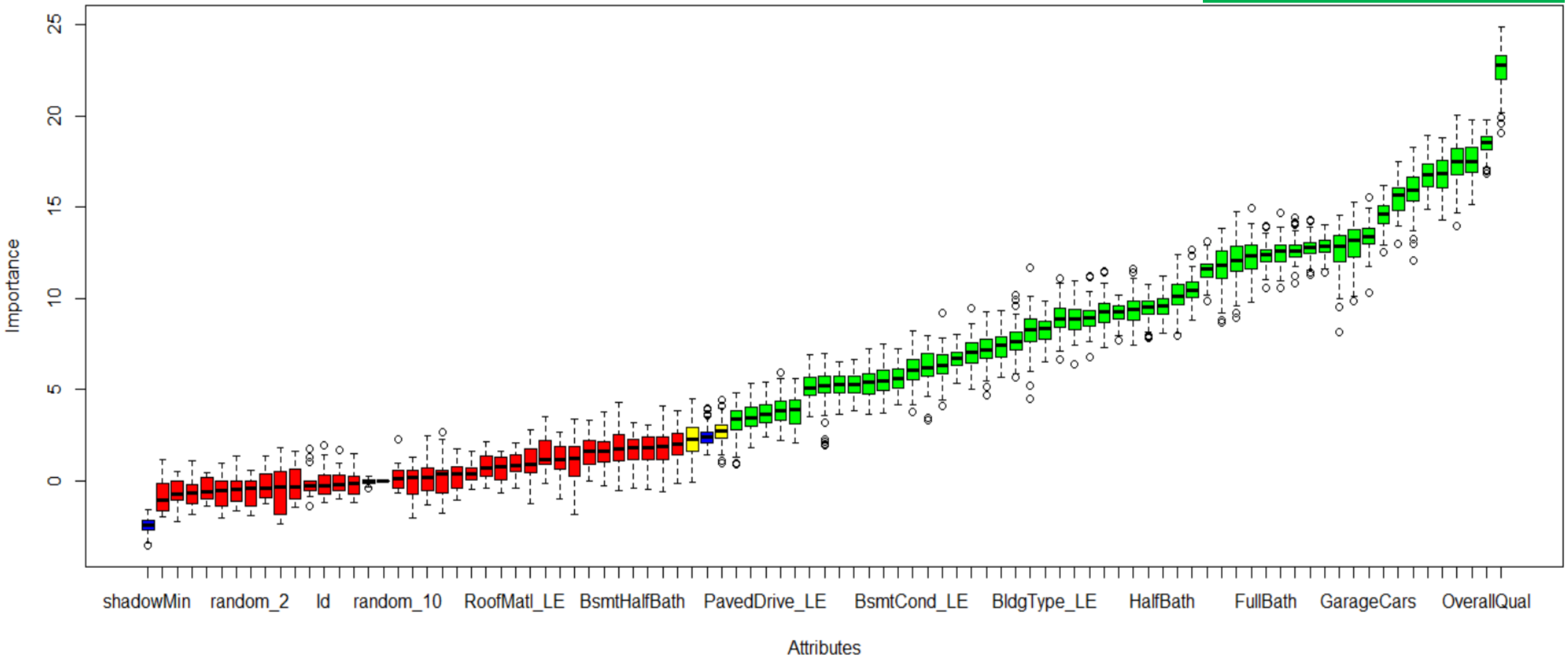
# WRAPPER METHOD – BORUTA IMPLEMETACIÓN

```
Boruta( formula ,
        data ,
        pValue = 0.01,
        mcAdj = TRUE,
        maxRuns = 100, __ )
```

```
> boruta.trn
Boruta performed 14 iterations in 17.50179 hours.
 No attributes deemed important.
 No attributes deemed unimportant.
 260 tentative attributes left: random_1, random_10, random_11, random_12, random_13 and 255
more;
> |
```

# BORUTA - HOUSEPRICE

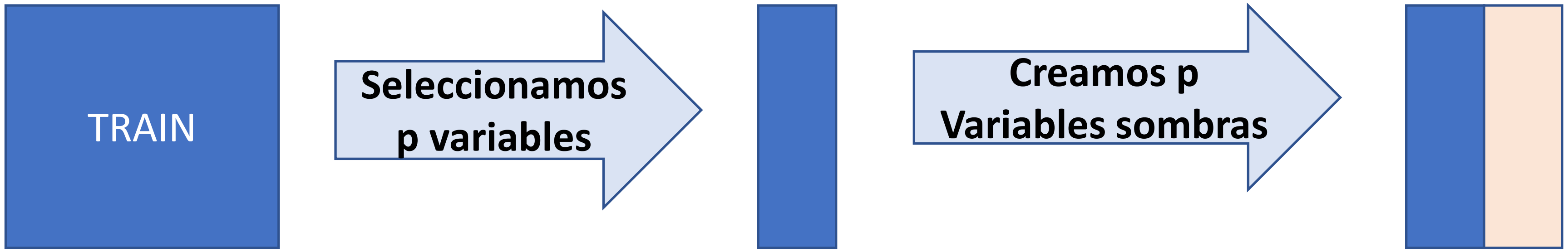
**55 variables selec.**
**SCORE VAL:** 0.1296
**SCORE TRAIN:** 0.0785



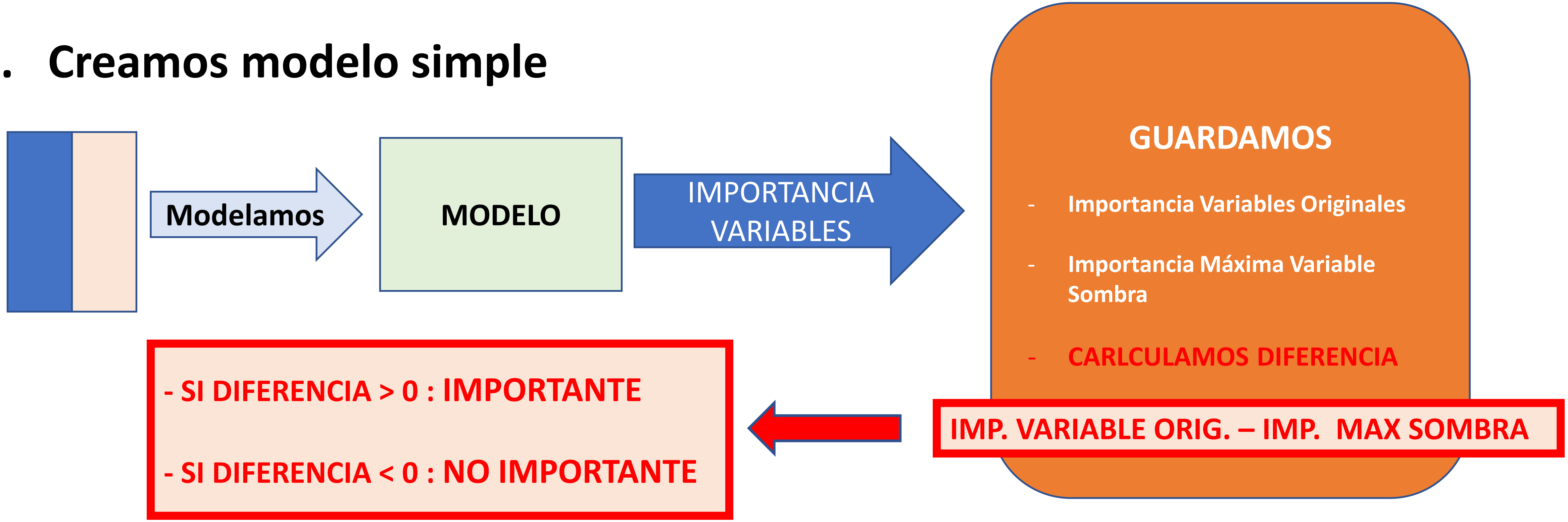
# ¿Y AHORA…?

# ITERACIÓN i:

1. **Seleccionamos p variables y creamos su sombra**




2. **Creamos modelo simple**



**FIN:**

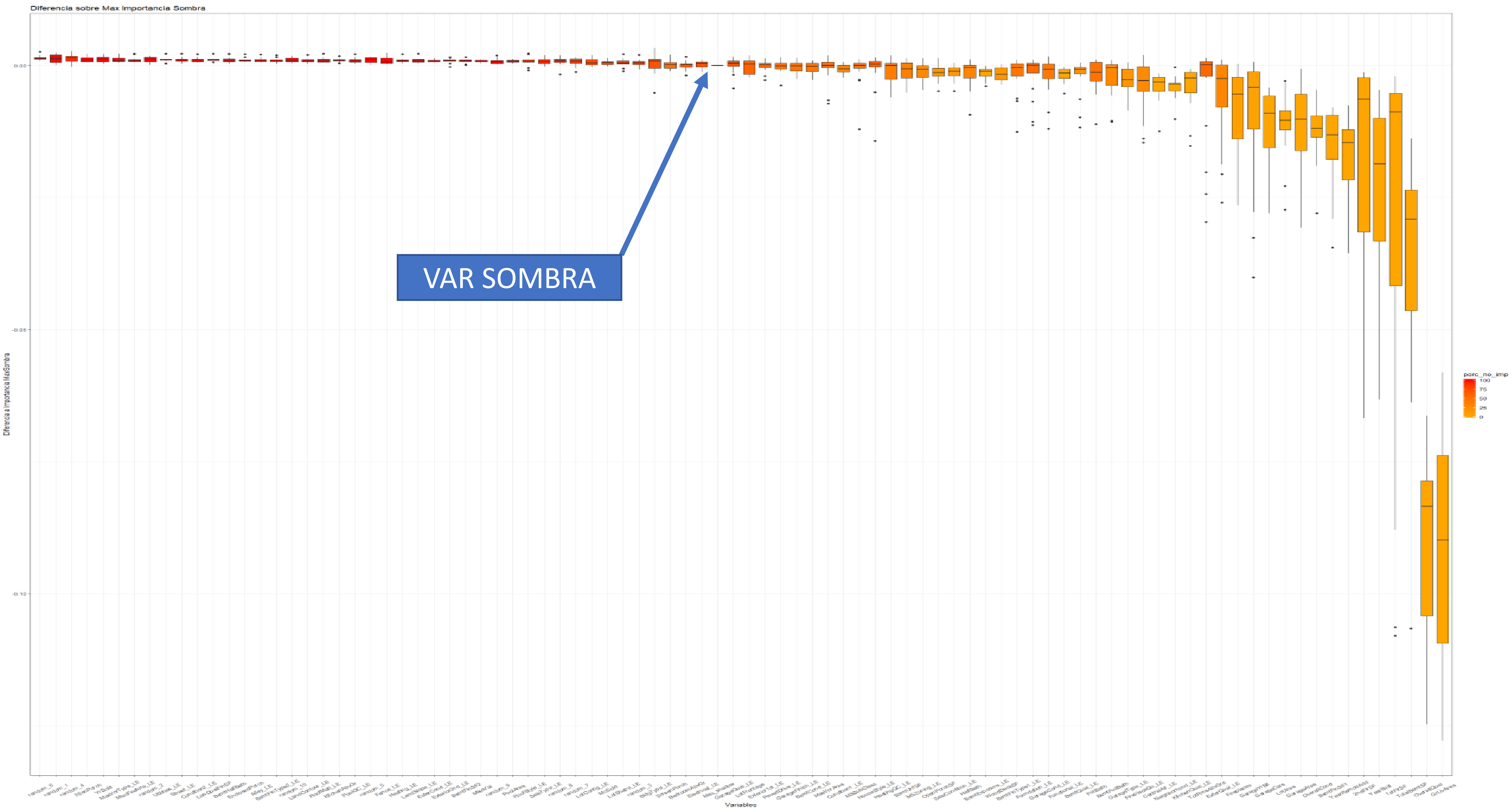
**Disponemos de un histórico**

| FEATURE | NUM ITERACIONES | Num Iteraciones no Importante | % No Importante | Media Diff MAX_SHADOW |
|---|---|---|---|---|
| random | 10 | 10 | 100% | -0,04852 |
| Var_81 | 30 | 0 | 0% | 3,2455 |
| Ramdom_2 | 12 | 10 | 83% | -0,0254 |
| Noise | 30 | 9 | 30% | 0,05458 |

**TOMAMOS DECISIÓN:**

- **Elegimos % no importante >> 50%**
- **Elegimos Media_DIFERENCIA > 0**
- **… etc**

# IMPORTANCIA HP



# EJEMPLO NUEVO – SANTANDER DATA

- **CREAMOS VARIABLES RUIDO**

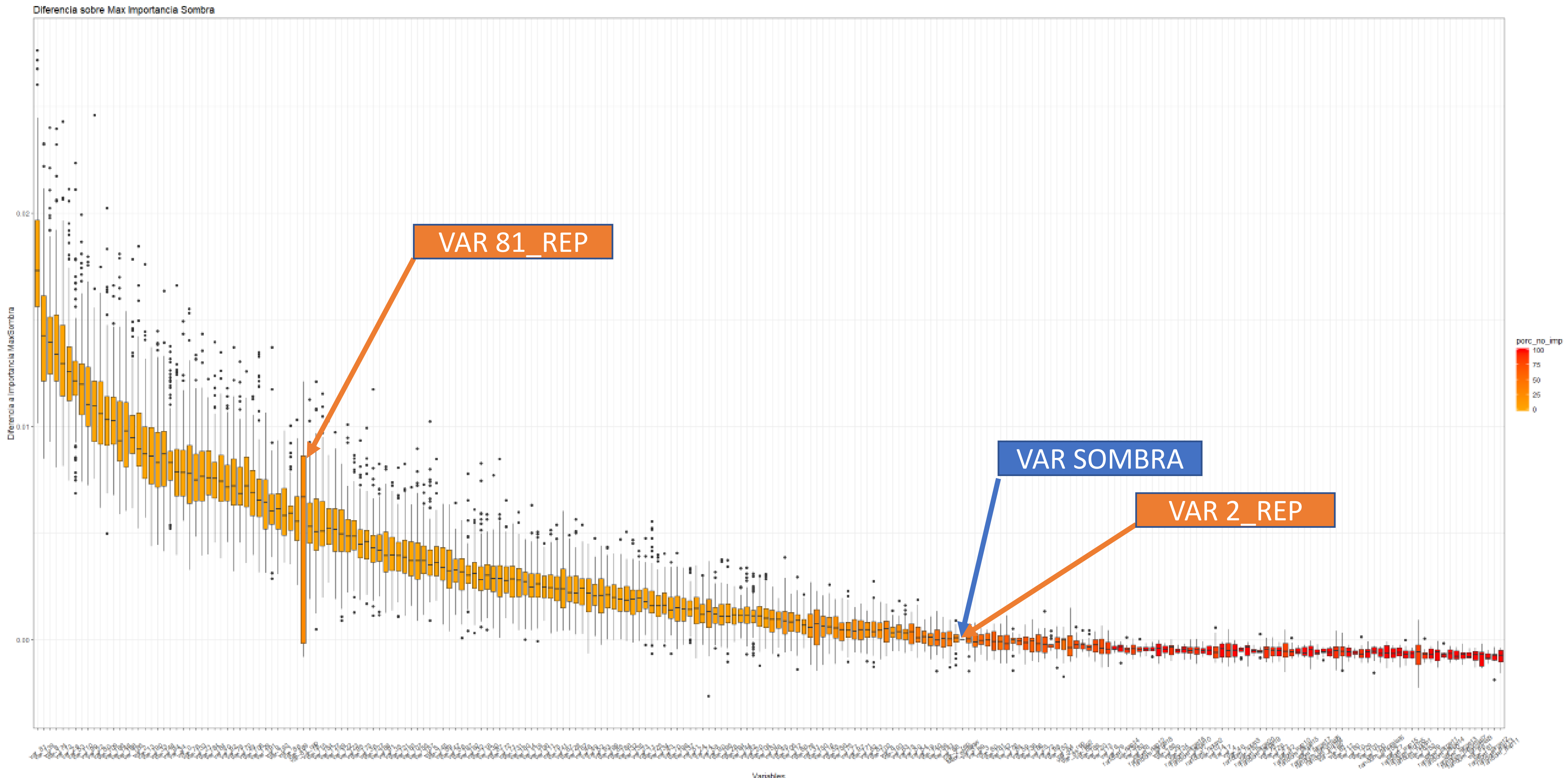
  - VAR_81_REP
  - VAR_2_REP

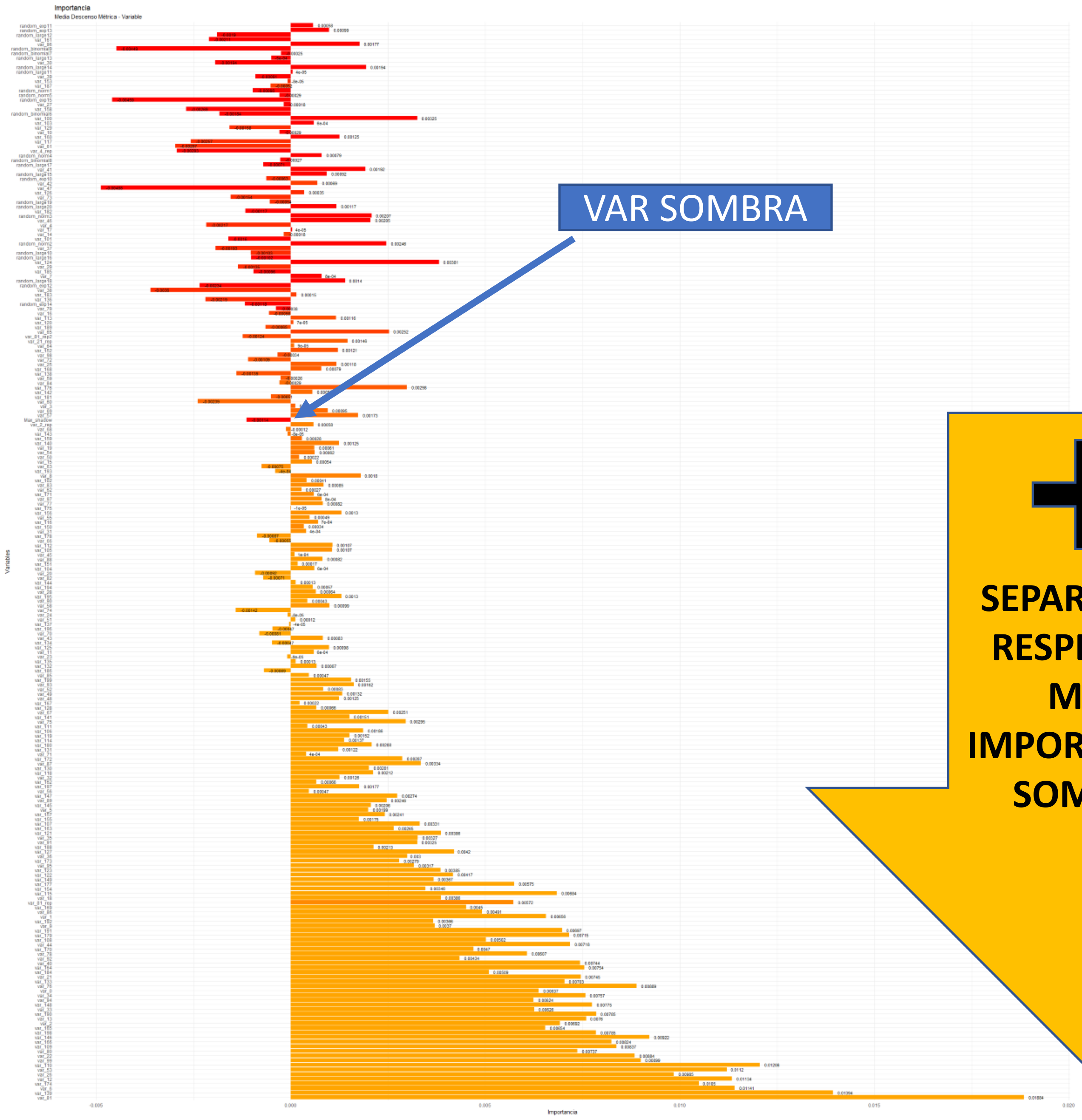
- **CREAMOS VARIABLES RANDOM**

  - RANDOMs

GAIN XGBOOST ASIGNA **BAJA IMPORTANCIA**

# Importancia






VAR SOMBRA

**DESCENSO MÉTRICA**

**SEPARACIÓN RESPECTO A MAX IMPORTANCIA SOMBRA**