Random Forest Regression y Clasificación

Humberto Vaquera Huerta

Verano 2022

// Bosques Aleatorios(Random Forest)

El concepto de bosques aleatorios seaparece por primera vez en

> Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

Un bosque aleatorio es una técnica de *aprendizaje automático* que se utiliza para resolver problemas de **regresión** y **clasificación**.

Un algoritmo de bosque aleatorio consiste de **árboles de decisión**. En Random Forests, la idea es relacionar los árboles que se generan en las diferentes muestras de **bootstrap** de los datos de entrenamiento, y luego simplemente reducimos la varianza en los árboles promediando.

El bosque aleatorio establece el resultado en función de las predicciones de los árboles de decisión. Predice tomando el promedio o la media de la salida de varios árboles. Aumentar el número de árboles aumenta la precisión del resultado.

Promediar los árboles nos ayuda a reducir la variación y también a mejorar el rendimiento de los árboles de decisión en el conjunto de pruebas y, finalmente, evitar el sobreajuste.

// ¿Qué es un bosque aleatorio?

Un bosque aleatorio consta de múltiples árboles de decisión aleatorios. Dos tipos de aleatoriedad están integrados en los árboles.

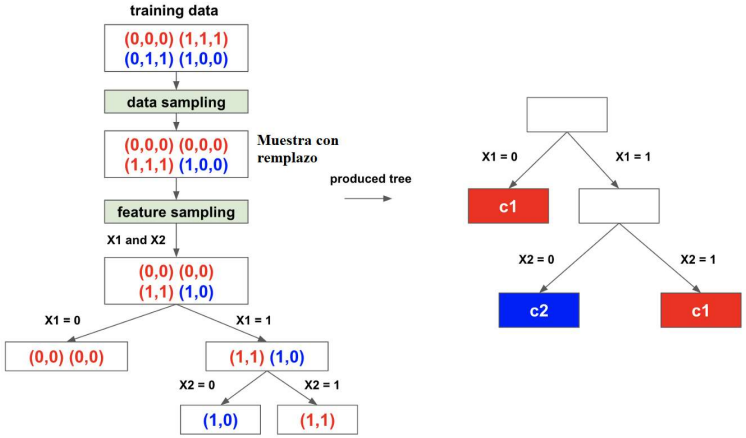
- Primero, cada árbol se construye a partir de una muestra aleatoria de los datos originales.
- En segundo lugar, en cada nodo del árbol, se selecciona aleatoriamente un subconjunto de características para generar la mejor división.

// Ejemplo:

Usamos el conjunto de datos a continuación para ilustrar cómo construir un árbol forestal aleatorio.

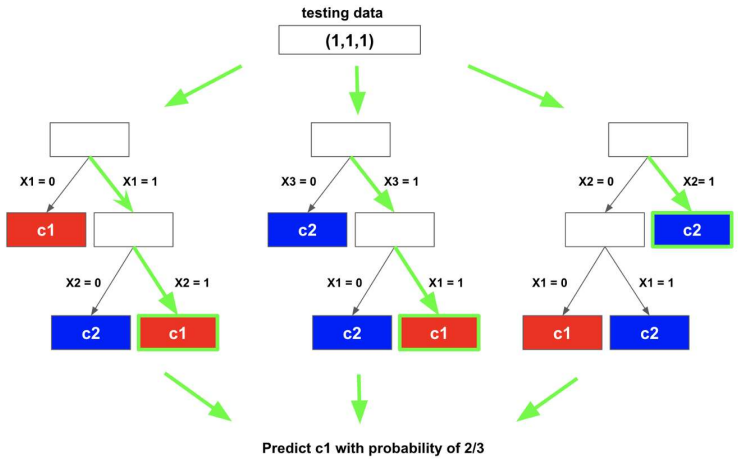
X1	X2	X3	Class
0	0	0	c1
1	1	1	c1
0	1	1	c2
1	0	0	c2

arbol de decision



arbol de decision

El mismo proceso se aplica para construir varios árboles. La siguiente figura ilustra el flujo de aplicar un bosque aleatorio con tres árboles a una instancia de datos de prueba.



arbol de decision

// Bosques Aleatorios(Random Forest)

- Random Forest se considera como la “panacea” en todos los problemas de ciencia de datos.
- Util para regresión y clasificación.
- Sirve como una técnica para reducción de la dimensionalidad.
- Se generan múltiples árboles (a diferencia de CART).
- Cada árbol da una clasificación (vota por una clase). Y el resultado es la clase con mayor número de votos en todo el bosque (forest).
- Para regresión, se toma el promedio de las salidas (predicciones) de todos los árboles.

// Ventajas de Random Forest

- Existen muy pocos supuestos y por lo tanto la preparación de los datos es mínima.
- Puede manejar hasta miles de variables de predictoras e identificar las más significativas. Método de reducción de dimensionalidad.
- Proporciona la importancia de variables. Incorpora métodos efectivos para estimar valores faltantes.
- Es posible usarlo como método no supervisado (clustering) y detección de outliers.

// Desventajas de Random Forest

- Pérdida de interpretación
- Bueno para clasificación, no tanto para regresión. Las predicciones no son de naturaleza continua.
- En regresión, no puede predecir más allá del rango de valores del conjunto de entrenamiento.
- Poco control en lo que hace el modelo (modelo caja negra para modeladores estadísticos)

// ¿Cuáles son las fortalezas y limitaciones de los bosques aleatorios?

- Las fortalezas de los bosques aleatorios incluyen:

Pueden manejar todo tipo de datos (por ejemplo, categóricos, continuos). A menudo funcionan de manera similar o mejor que los métodos de regresión. Funcionan bien con conjuntos de datos que contienen una gran cantidad de variables predictoras.

- Las limitaciones de los bosques aleatorios incluyen:

A menudo se los considera un método de "caja negra" porque los modelos no se interpretan fácilmente. Requieren un ajuste de modelo extenso para obtener los mejores resultados.

// caracteristicas:

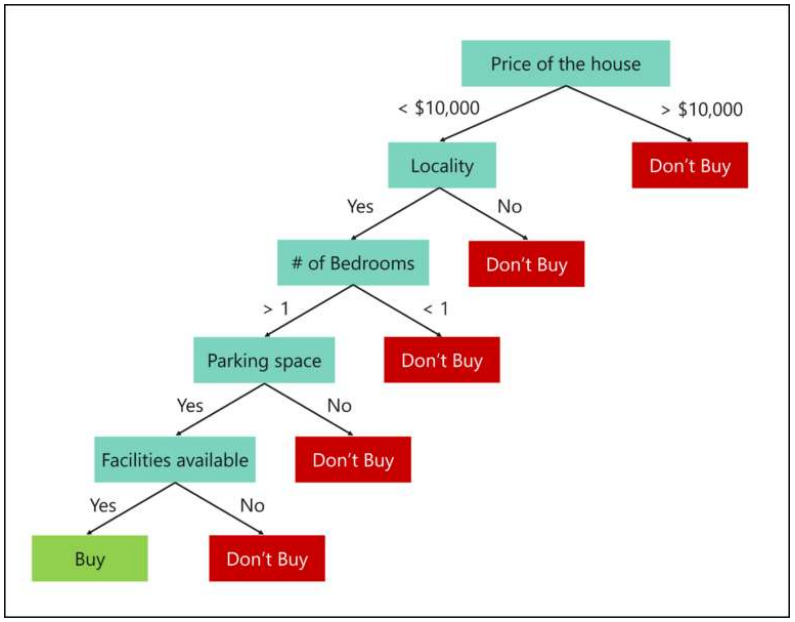
- Aplicable tanto a problemas de regresión como de clasificación.
- Maneja predictores categóricos de forma natural.
- Computacionalmente simple y rápido de instalar, incluso para problemas grandes.
- Sin supuestos de distribución formales (**no paramétrico**).
- Puede manejar interacciones altamente no lineales y límites de clasificación.
- Selección automática de variables. sí. Pero también necesita una importancia variable.
- Maneja valores perdidos

// Contruccion de un modelo random forest

Cada árbol se construye así:

- Dado que el número de casos en el conjunto de entrenamiento es N. Una muestra de esos N casos se toma aleatoriamente pero CON REEMPLAZO. Esta muestra será el conjunto de entrenamiento para construir el árbol i.
- Si existen M varibles de entrada, un número m<M se especifica tal que para cada nodo, m variables se seleccionan aleatoriamente de M. La mejor división de estos m atributos es usado para ramificar el árbol. El valor m se mantiene constante durante la generación de todo el bosque.
- Cada árbol crece hasta su máxima extensión posible y NO hay proceso de poda.
- Nuevas instancias se predicen a partir de la agregación de las predicciones de los x árboles (i.e., mayoría de votos para clasificación, promedio para regresión)

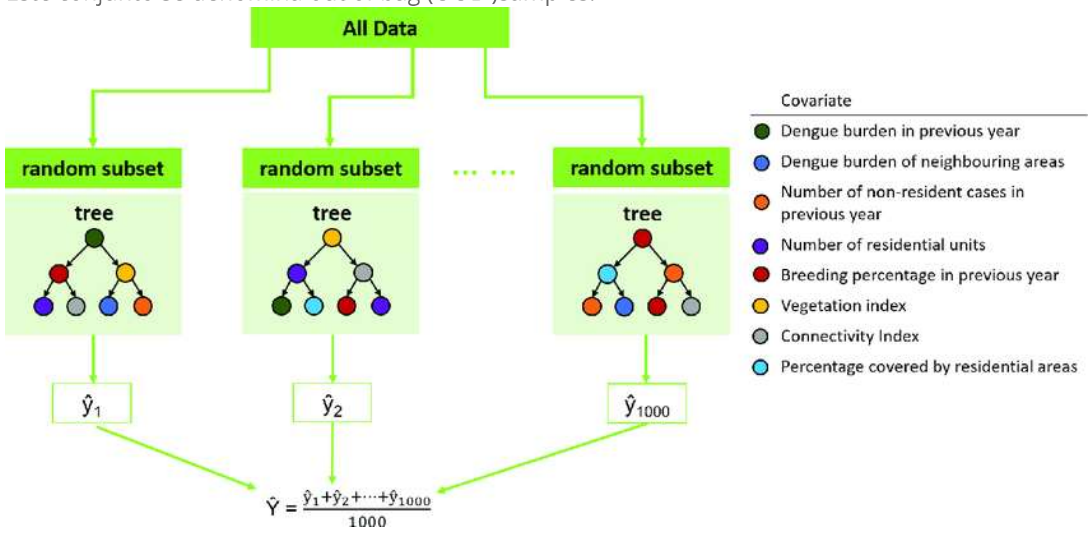
// Arbol de decision



arbol de decision

// Random Forest Regression y Bootstrap

- El proceso de muestreo de los datos con reemplazo se denomina bootstrap.
- Un tercio de los datos no se usan para el entrenamiento y pueden ser usados para test.
- Este conjunto se denomina out of bag (OOB)samples.



/ Random Forest Software en R

La implementacion mas antigua es:

- [randomForest]

lista:

- randomForest::randomForest
- h2o::h2o.randomForest
- DistributedR::hpdRF_parallelForest
- party::cForest: A random forest variant for response variables measured at arbitrary scales based on conditional inference trees.
- randomForestSRC implements a unified treatment of Breiman's random forests for survival, regression and classification problems.
- quantregForest can regress quantiles of a numeric response on exploratory variables via a random forest approach.
- ranger
- Rborist
- The caret contine paquetes para RF here](https://topepo.github.io/caret/Random_Forest.html)):
  - Conditional Inference Random Forest (`party::cForest`)
  - Oblique Random Forest (`obliqueRF`)
  - Parallel Random Forest (`randomForest` + `foreach`)
  - Random Ferns (`rFerns`)
  - Random Forest (`randomForest`)
  - Random Forest (`ranger`)
  - Quantile Random Forest (`quantregForest`)
  - Random Forest by Randomization (`extraTrees`)
  - Random Forest Rule-Based Model (`inTrees`)
  - Random Forest with Additional Feature Selection (`Boruta`)
  - Regularized Random Forest (`RRF`)
  - Rotation Forest (`rotationForest`)
  - Weighted Subspace Random Forest (`wsrf`)
- The mlr package wraps a number of different Random Forest packages in R:
  - Conditional Inference Random Forest (`party::cForest`)
  - Rotation Forest (`rotationForest`)
  - Parallel Forest (`ParallelForest`)
  - Survival Forest (`randomForestSRC`)
  - Random Ferns (`rFerns`)
  - Random Forest (`randomForest`)
  - Random Forest (`ranger`)

- Synthetic Random Forest (`randomForestSRC`)
- Random Uniform Forest (`randomUniformForest`)

// Ejemplo datos de Biomasa

Conjunto de datos de 40 pinos cortados en Luisiana, EE. UU. presentados en el artículo de Parresol de 2001.

/// Datos del articulo:

> Bernard R Parresol. Additivity of nonlinear biomass equations. Canadian Journal of Forest Research. 31(5): 865-878. https://doi.org/10.1139/x00-202

Los datos contienen las siguientes variables:

- TreeID: Tree observation record,
- DBH: Tree diameter at breast height, cm,
- HT: Tree height, m,
- LCL: Tree live crown length, m,
- Age: Age of the tree, years,
- Mass_wood: Green mass of the wood in the tree, kg,
- Mass_bark: Green mass of the bark in the tree, kg,
- Mass_crown: Green mass of the crown of the tree, kg, and
- Mass_tree: Green mass of all tree components, kg.

//// Nos interésa predecir la biomasa de todos los componentes de los árboles utilizando medidas comunes de árboles como el diámetro, la altura, la longitud de la copa viva y la edad del árbol.

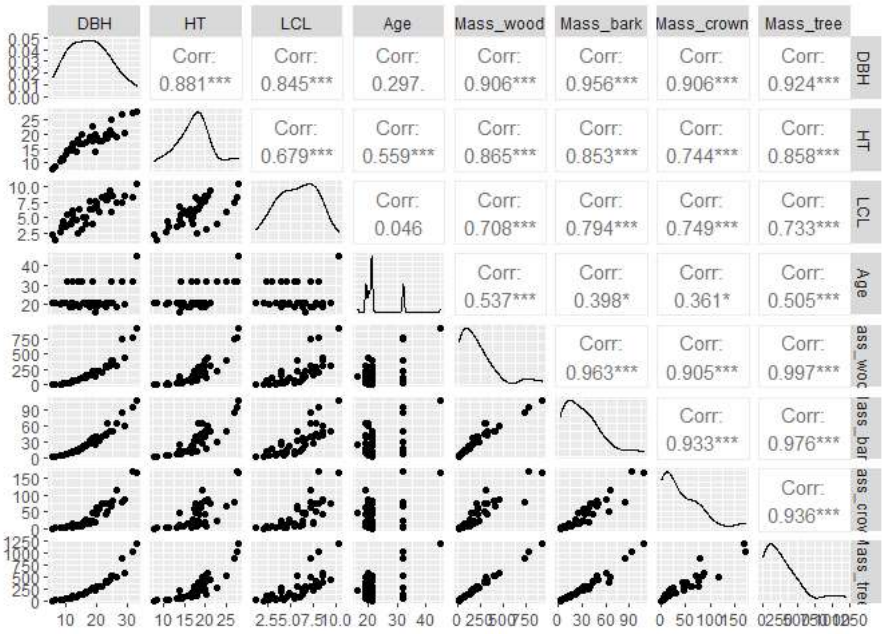
```
biomasa <- read.csv("biomasa.csv", row.names=1)
knitr::kable(biomasa)
```

DBH	HT	LCL	Age	Mass_wood	Mass_bark	Mass_crown	Mass_tree
5.6	7.9	2.1	21	6.5	2.3	1.0	9.8
6.4	8.5	1.2	21	7.4	2.6	2.1	12.1
8.1	10.7	2.7	20	17.6	4.5	2.3	24.4
8.4	11.3	3.4	21	18.5	4.3	4.2	27.0

DBH	HT	LCL	Age	Mass_wood	Mass_bark	Mass_crown	Mass_tree
9.1	11.0	4.3	21	22.6	5.4	5.6	33.6
9.9	13.1	3.4	21	30.6	7.4	5.5	43.5
10.4	14.3	5.5	32	32.9	6.7	6.4	46.0
11.2	14.6	4.0	19	40.6	9.3	6.2	56.1
11.7	14.3	4.0	21	46.0	10.7	7.7	64.4
12.2	14.9	6.1	21	51.6	13.1	6.1	70.8
11.9	16.8	4.6	32	60.4	10.1	5.4	75.9
13.2	13.7	4.6	20	62.8	15.2	10.7	88.7
12.2	15.8	5.5	19	67.5	12.9	15.3	95.7
13.7	18.0	2.4	32	81.2	12.5	8.7	102.4
14.2	16.5	6.4	19	94.3	18.2	11.2	123.7
15.0	20.1	3.0	32	123.4	16.5	7.7	147.6
15.7	16.8	4.9	21	107.3	21.5	19.7	148.5
16.5	17.1	4.9	20	123.8	22.1	28.9	174.8
16.5	17.1	4.0	21	151.6	24.6	16.8	193.0
19.6	13.7	6.7	16	140.4	25.1	46.2	211.7
17.5	19.2	7.6	19	170.4	27.4	16.8	214.6
17.8	18.3	6.4	21	169.6	31.7	24.0	225.3
18.5	17.7	6.4	21	160.3	36.9	47.5	244.7
19.6	19.8	7.6	19	199.8	38.7	19.7	258.2
18.5	22.9	4.0	32	231.6	29.6	24.6	285.8
19.8	18.6	6.7	21	217.9	33.9	45.8	297.6

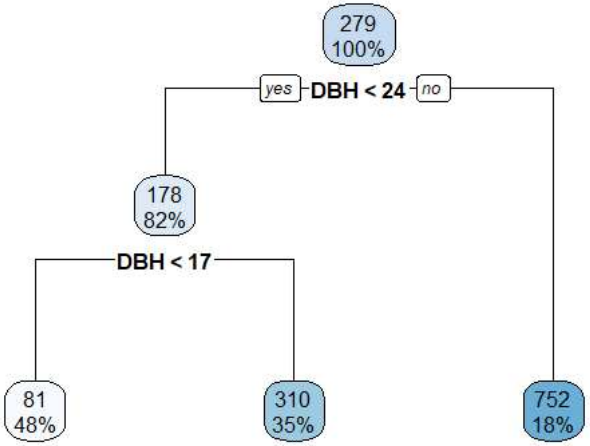
DBH	HT	LCL	Age	Mass_wood	Mass_bark	Mass_crown	Mass_tree
20.6	17.4	5.8	21	216.0	32.6	61.2	309.8
21.6	17.7	8.2	20	200.6	40.2	75.4	316.2
19.8	18.9	7.3	19	217.5	38.5	62.0	318.0
22.9	19.8	8.5	19	314.8	43.1	43.2	401.1
23.6	18.3	7.9	20	287.1	63.4	51.7	402.2
23.1	18.9	7.9	21	290.9	44.3	76.7	411.9
24.1	21.3	9.4	21	320.1	50.6	75.6	446.3
26.4	19.2	7.3	19	308.6	65.7	116.0	490.3
24.6	25.0	5.8	32	403.0	49.8	69.8	522.6
25.1	19.8	8.5	21	390.4	48.8	83.5	522.7
29.0	20.4	8.5	20	445.2	60.4	88.0	593.6
28.4	26.8	7.3	32	736.4	84.0	79.9	900.3
31.8	27.4	8.2	32	770.9	93.8	170.2	1034.9
33.0	27.7	10.4	45	921.3	108.0	169.2	1198.5

```
library(GGally)

ggpairs(biomasa)
```



// Uso de arbol de regresión

```
library(rpart)
arbol1 = rpart(Mass_tree ~ DBH + HT + LCL + Age,data = biomasa)
library(rpart.plot)
rpart.plot(arbol1)
```

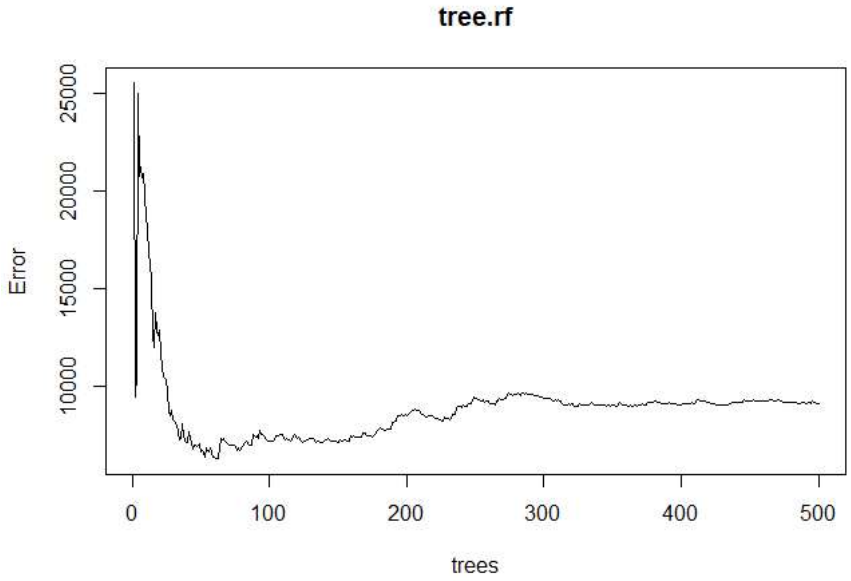


// Regresion de bosques aleatorios

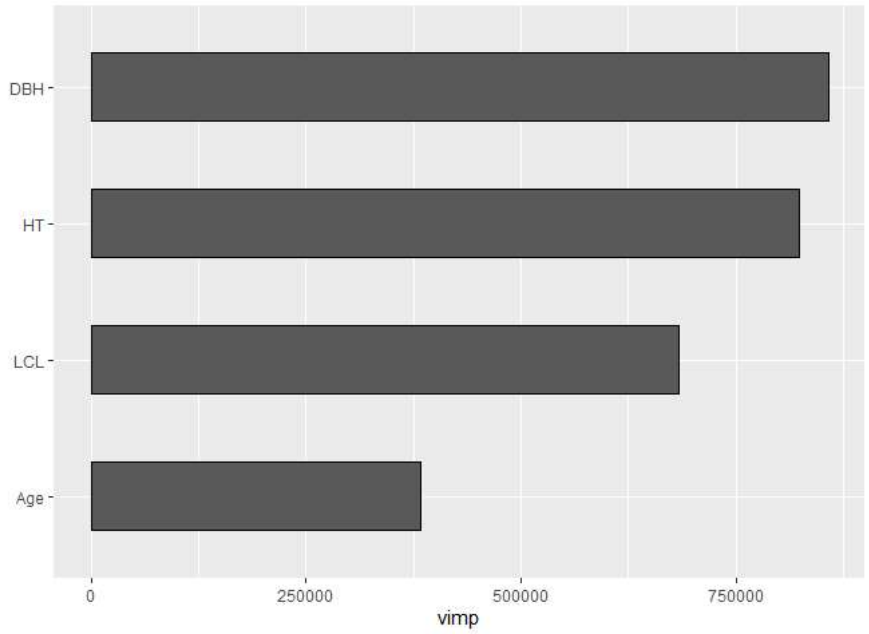
```
library(randomForest)
tree.rf <- randomForest(Mass_tree ~ DBH + HT + LCL + Age,
                        data = biomasa,
                        keep.forest = T,
                        importance = TRUE,
                        mtry = 1,
                        ntree = 500)
tree.rf
```

```
##
## Call:
##  randomForest(formula = Mass_tree ~ DBH + HT + LCL + Age, data = biomasa,      keep.forest
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 9113.917
##                     % Var explained: 87.68
```

```
plot(tree.rf)
```



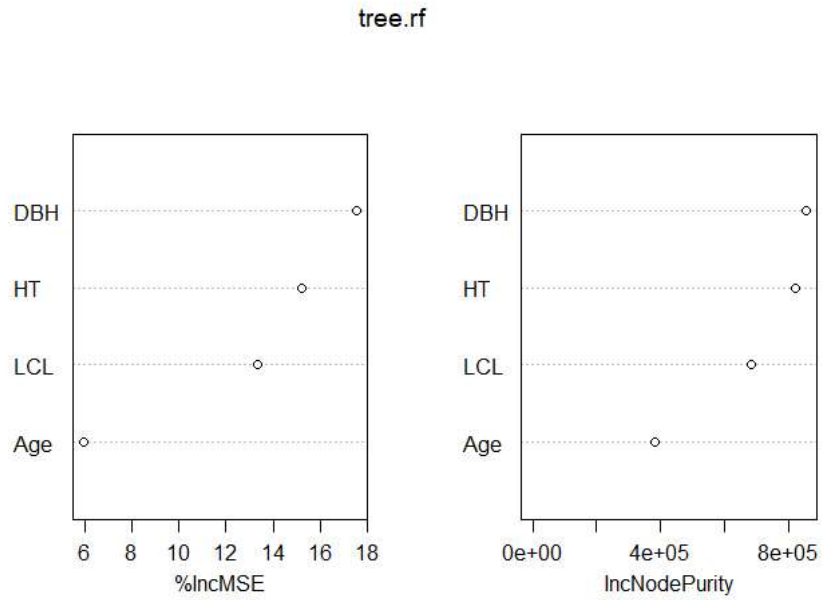

```
library(ggRandomForests)

plot(gg_vimp(tree.rf))
```



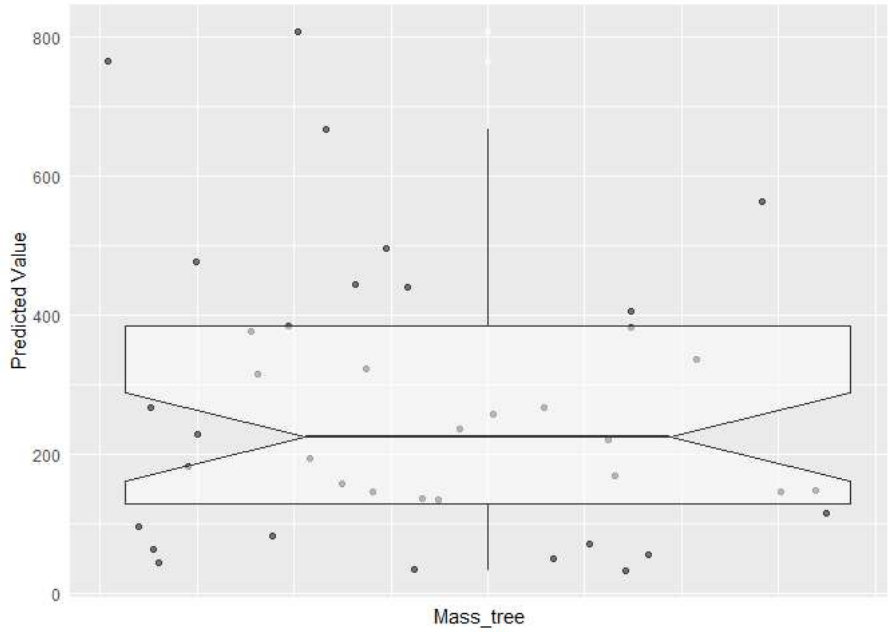

```
importance(tree.rf)
```

```
##      %IncMSE IncNodePurity
## DBH 17.55283      856948.6
## HT  15.23458      823629.3
## LCL 13.37017      682891.7
## Age  5.94164      382848.4
```

```
varImpPlot(tree.rf)
```

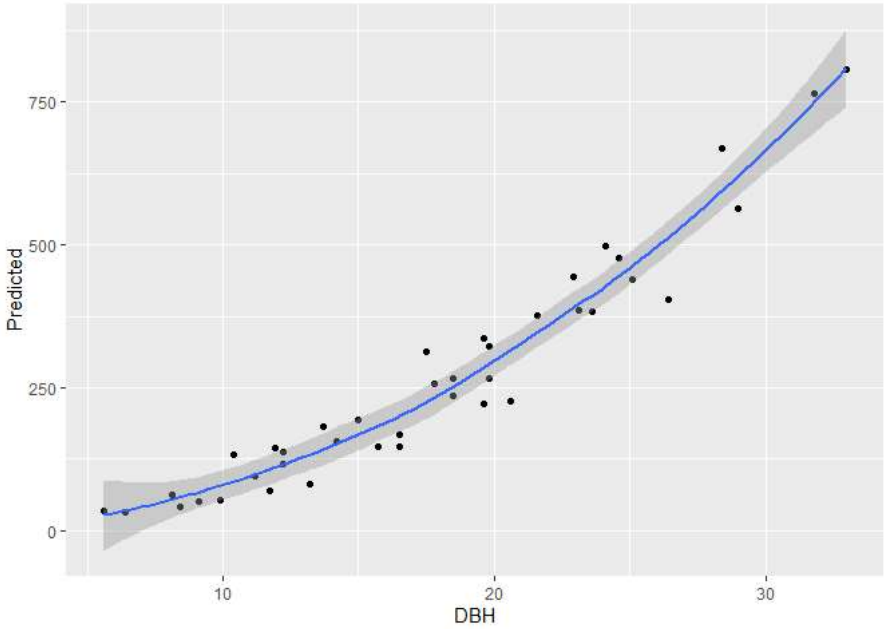



```
plot(gg_rfsrc(tree.rf), alpha=.5)
```

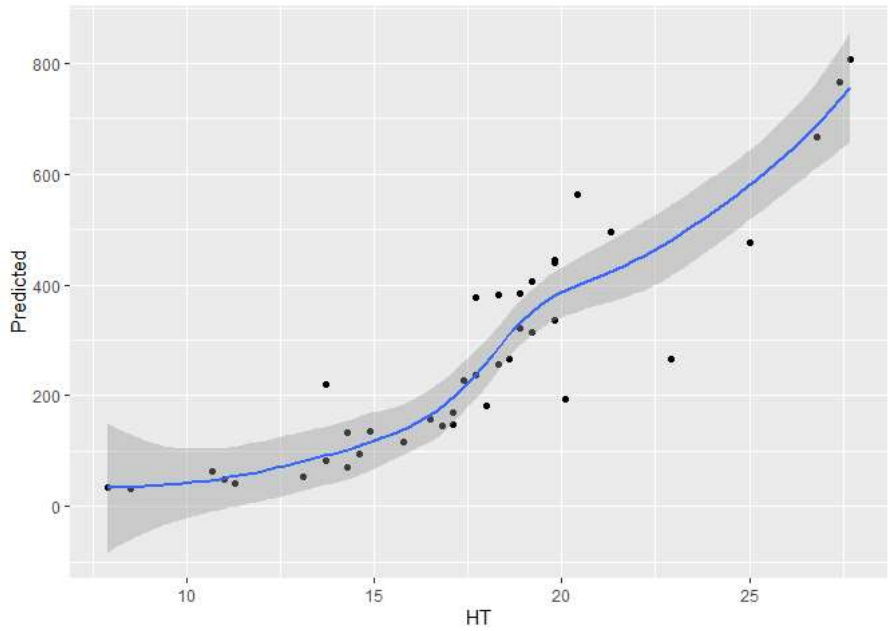


```
plot(gg_variable(tree.rf))
```

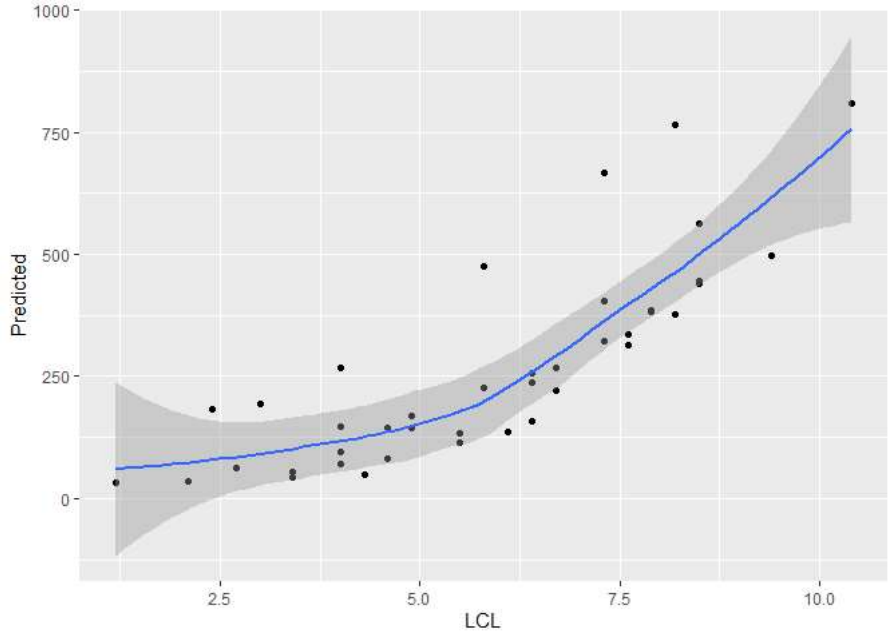
```
## [[1]]
```



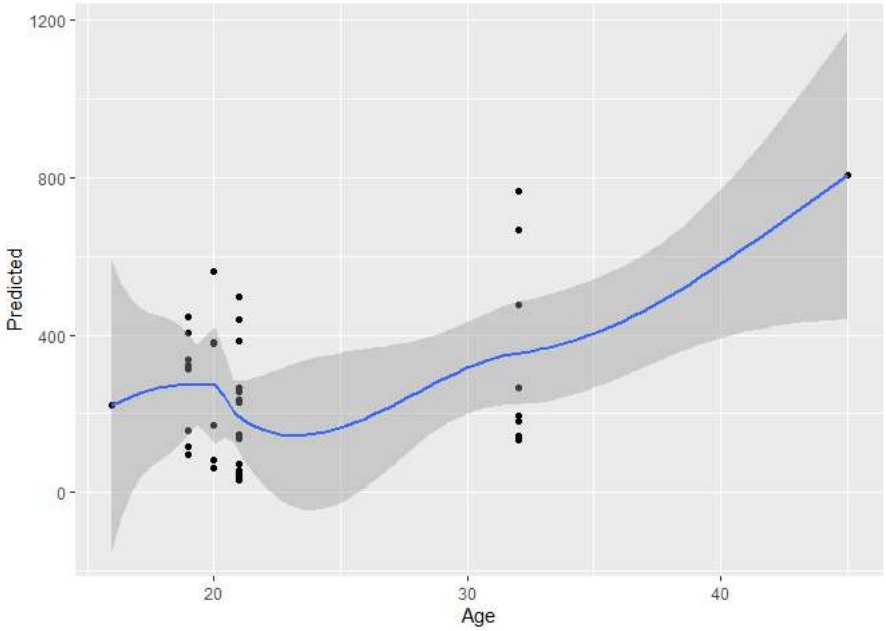

```
##
## [[2]]
```


800
600
400
200
0
Predicted
10
15
20
25
HT

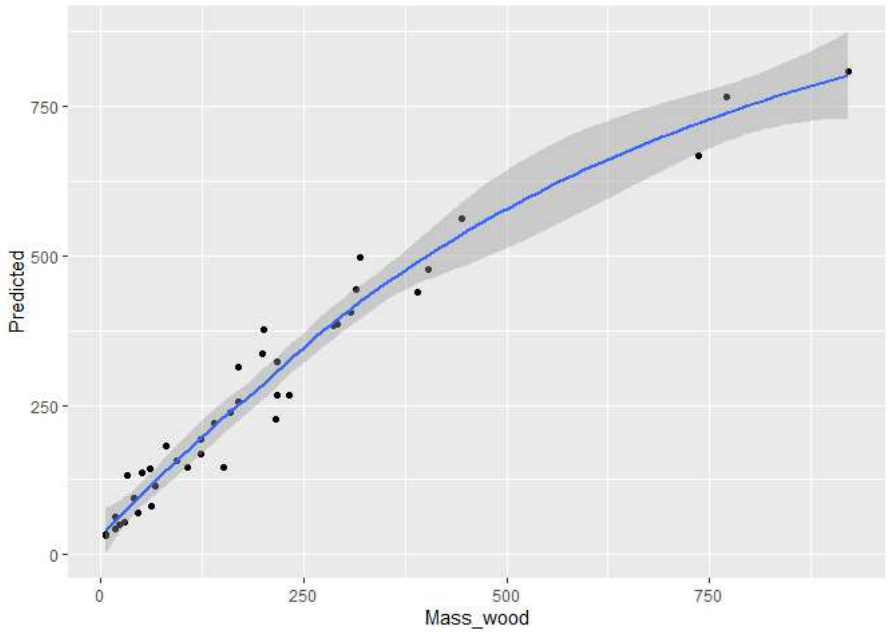
```
##
## [[3]]
```


1000
750
500
250
0
Predicted
2.5
5.0
7.5
10.0
LCL

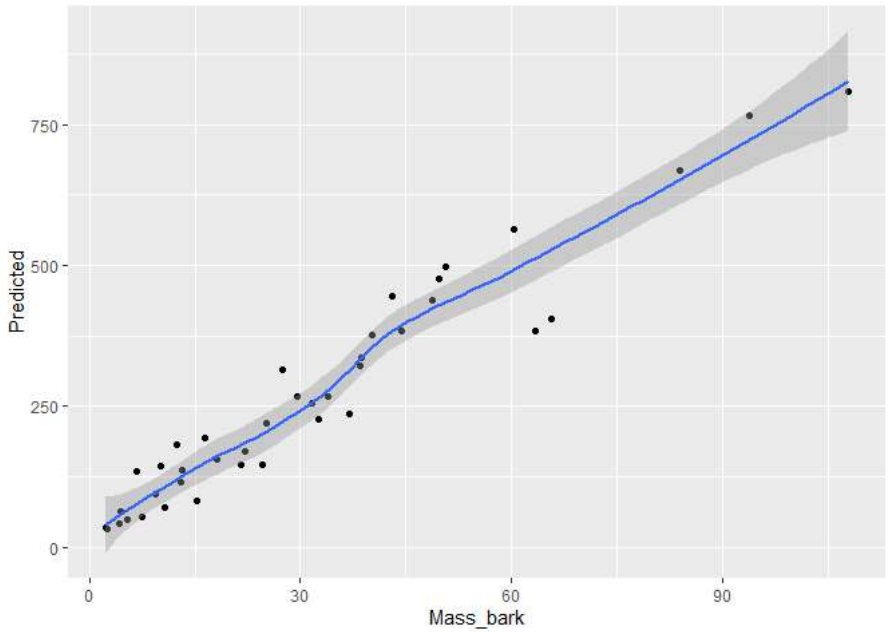
```
##
## [[4]]
```



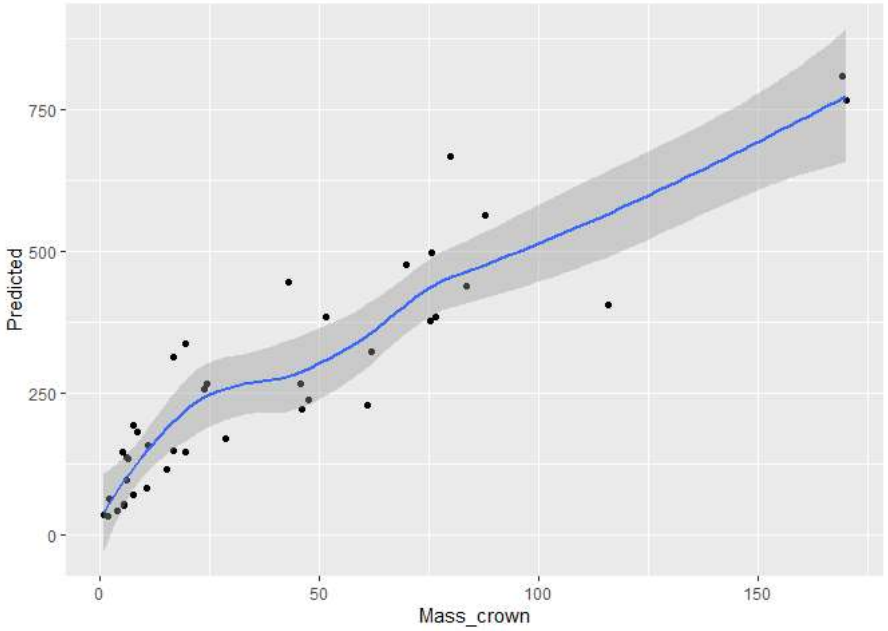

```
##
## [[5]]
```


750
500
250
0
Predicted
0
250
500
750
Mass_wood

```
##
## [[6]]
```


750
500
250
0
Predicted
0
30
60
90
Mass_bark

```
##
## [[7]]
```



// Usando Regresion Lineal

```
tree.reg <- lm(Mass_tree ~ DBH + HT + LCL + Age, data = biomasa)
summary(tree.reg)
```

```
##
## Call:
## lm(formula = Mass_tree ~ DBH + HT + LCL + Age, data = biomasa)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -136.285  -57.177   -9.399   43.822  189.758
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -545.374     67.916  -8.030 1.89e-09 ***
## DBH           40.523      5.778   7.013 3.68e-08 ***
## HT           -15.048      8.079  -1.862   0.0709 .
```

```
## LCL            2.490     12.259   0.203   0.8402
## Age           15.431      3.198   4.825 2.72e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 82.33 on 35 degrees of freedom
## Multiple R-squared:  0.9198, Adjusted R-squared:  0.9106
## F-statistic: 100.4 on 4 and 35 DF,  p-value: < 2.2e-16
```

// Predicciones usando RF y MCO

```
library(tidyverse)
Mass_pred_rf <- predict(tree.rf, biomasa, predict.all = F)
Mass_pred_reg <- predict(tree.reg, biomasa, predict.all = F)
tree2 <- as.data.frame(cbind(biomasa, Mass_pred_rf, Mass_pred_reg))
tree2 %>%
  summarize(Mass_tree, Mass_pred_rf, Mass_pred_reg)
```

```
##    Mass_tree Mass_pred_rf Mass_pred_reg
## 1        9.8     25.56669   -108.051811
## 2       12.1     25.02732    -86.903051
## 3       24.4     43.95141    -62.814807
## 4       27.0     36.87208    -42.513067
## 5       33.6     43.44165     -7.391764
## 6       43.5     47.84379     -8.814627
## 7       46.0    100.84321    168.354603
## 8       56.1     80.31464     -8.073626
## 9       64.4     66.75992     47.563293
## 10      70.8    103.47756     64.024945
## 11      75.9    115.71993    189.278688
## 12      88.7     86.44960    103.439719
## 13      95.7    104.80953     18.126885
## 14     102.4    145.98302    238.684823
## 15     123.7    140.69856     90.880242
## 16     147.6    178.42202    261.258307
## 17     148.5    146.55356    174.276553
## 18     174.8    169.85826    186.750002
## 19     193.0    170.52493    199.939638
## 20     211.7    211.64181    306.293014
## 21     214.6    276.58769    186.964881
## 22     225.3    243.34148    240.537957
## 23     244.7    244.80668    277.932654
## 24     258.2    294.40606    263.034375
```

```
## 25     285.8    273.12179    363.444771
## 26     297.6    282.08244    317.816460
## 27     309.8    266.76878    366.051168
## 28     316.2    345.96732    392.605018
## 29     318.0    322.32777    283.934957
## 30     401.1    427.57230    399.000959
## 31     402.2    390.31693    463.875242
## 32     411.9    400.36919    450.015697
## 33     446.3    474.27605    458.158909
## 34     490.3    434.95693    546.871939
## 35     522.6    492.40713    583.516204
## 36     522.7    471.53658    519.012527
## 37     593.6    574.09135    652.592720
## 38     900.3    820.32826    714.152257
## 39    1034.9    898.30507    845.142484
## 40    1198.5   1011.26338   1095.330861
```

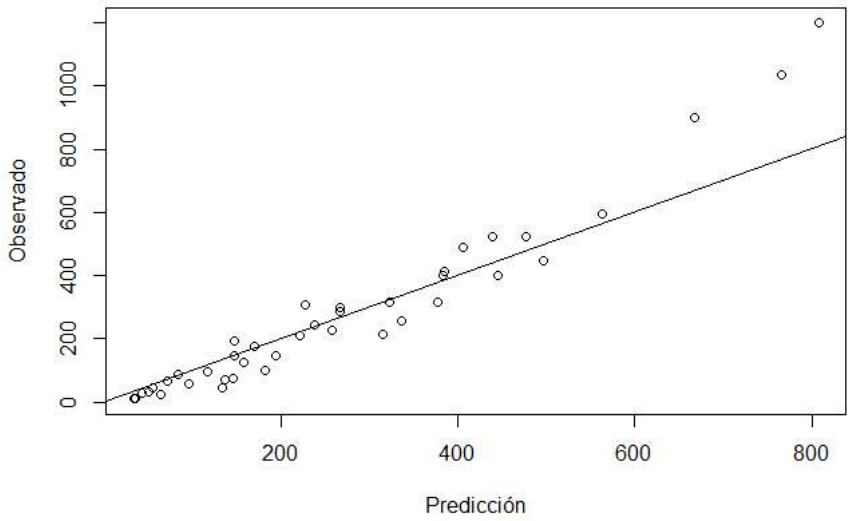
```
sd(Mass_pred_rf)
```

```
## [1] 236.8603
```

```
sd(Mass_pred_reg)
```

```
## [1] 264.1445
```

```
obs <- biomasa$Mass_tree
pred <- predict(tree.rf)

# plot(pred, obs, main = "Observado frente a predicciones (quality)",
#      xlab = "Predicción", ylab = "Observado")
plot(jitter(pred), jitter(obs), xlab = "Predicción", ylab = "Observado")
abline(a = 0, b = 1)
```

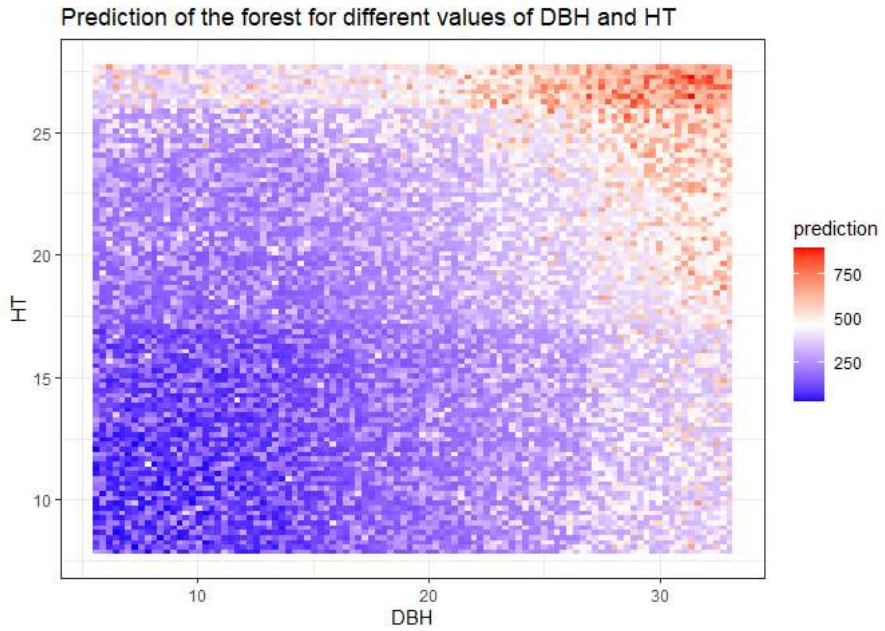



```
R2=cor(obs,pred)^2
R2
```

```
## [1] 0.9379276
```

Tenga en cuenta que algunas predicciones del modelo de regresión lineal en los 40 árboles proporcionan valores negativos para la masa total del árbol predicha, una característica indeseable que puede ser necesario abordar antes de implementar el modelo

```
library(randomForestExplainer)

plot_predict_interaction(tree.rf, biomasa, "DBH", "HT")
```



// Ejemplo de Fertilidad de Suelos

Datos de África occidental sobre la fertilidad del suelo y la respuesta de los cultivos a los fertilizantes. Estos datos son parte de un estudio más amplio que se describe en un artículo de próxima publicación (Bonilla et al.). Hijmans, R.J., 2019. Statistical modeling. In: Hijmans, R.J. and J. Chamberlin. Regional Agronomy: a pratical handbook. CIMMYT. https:/reagro.org/tools/statistical/

Estas son las variables que tenemos.

emp	Average temperature
precip	Annual precipitation
ExchP	Soil exchangeble P
TotK	Soil total K
ExchAl	Soil exchangeble Al
TotN	Soil total N
sand	Soil franction sand (%)
clay	Soil fraction clay (%)

emp	Average temperature
SOC	Soil organic carbon (g/kg)
pH	Soil pH
AWC	Soil water holding capacity
fert	fertilizer (index) kg/ha

```
#remotes::install_github("reagro/agrodata")
#remotes::install_github("reagro/agro")
library(agrodata)
library(agro)
library(randomForest)
library(rpart)
datos_fert=reagro_data("soilfert")
pander::pander(head(datos_fert))
```

Table continues below

	temp	precip	ExchP	TotK	ExchAl	TotN	sand	clay	SOC
1463	27	1260	933	120	610	1157	65	17	13.5
1464	27	1260	933	120	610	1157	65	17	13.5
1465	27	1260	933	120	610	1157	65	17	13.5
1466	27	1260	933	120	610	1157	65	17	13.5
1467	27	1260	933	120	610	1157	65	17	13.5
1468	27	1260	933	120	610	1157	65	17	13.5

	pH	AWC	fert
1463	6.3	26	120
1464	6.3	26	120
1465	6.3	26	120
1466	6.3	26	120

	pH	AWC	fert
1467	6.3	26	120
1468	6.3	26	120

```
model <- SOC~pH+precip+temp+sand+clay
lrm <- lm(model, data=datos_fert)
summary(lrm)
```

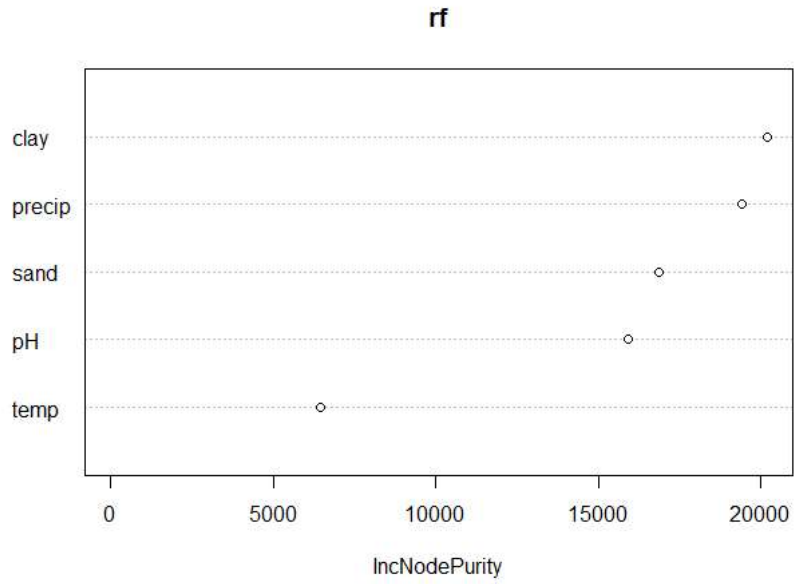
```
##
## Call:
## lm(formula = model, data = datos_fert)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.281  -2.713  -1.125   2.054  27.067
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.6781312  4.0759621  -1.884   0.0598 .
## pH          -1.8263969  0.3761195  -4.856 1.31e-06 ***
## precip       0.0052787  0.0003756  14.053  < 2e-16 ***
## temp        -0.0684204  0.1128496  -0.606   0.5444
## sand         0.1741979  0.0236394   7.369 2.69e-13 ***
## clay         0.7917492  0.0333736  23.724  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.733 on 1678 degrees of freedom
## Multiple R-squared:  0.5379, Adjusted R-squared:  0.5365
## F-statistic: 390.7 on 5 and 1678 DF,  p-value: < 2.2e-16
```

```
library(randomForest)
rf <- randomForest(model, data=datos_fert)
rf
```

```
##
## Call:
##  randomForest(formula = model, data = datos_fert)
```

```
##               Type of random forest: regression
##                     Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 2.494279
##                     % Var explained: 94.84
```

```
varImpPlot(rf)
```




```
library(randomForestExplainer)
forest1 <- randomForest(model,data=datos_fert, localImp=TRUE)
measure_importance(forest1)
frame <- measure_importance(forest1, measures=c("mean_min_depth", "times_a_root"))
plot_importance_ggpairs(frame, measures = c("mean_min_depth", "times_a_root"))

plot_multi_way_importance(measure_importance(forest1))
plot_min_depth_distribution(min_depth_distribution(forest1))
```

// Seleccion de variables en random forest

```
library(randomForestExplainer)
forest1 <- randomForest(model, data = datos_fert, localImp = TRUE)
set.seed(1234)
#library(VSURF)
#graf1 <- VSURF(model, data = datos_fert)
#summary(graf1)
```

/ Random Forest para Clasificar

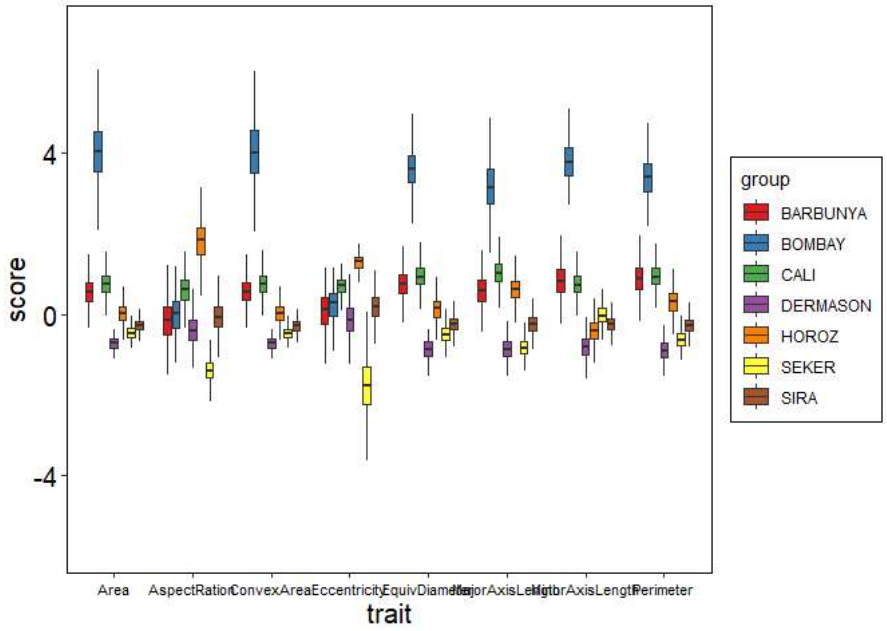
// Clasificacion semillas de frijol

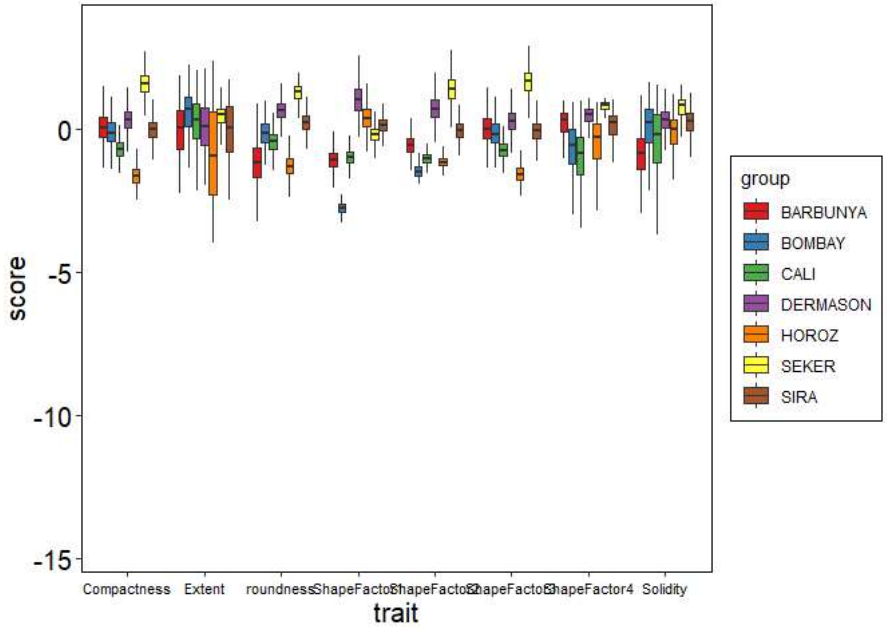
Seven different types of dry beans were used in this research, taking into account the features such as form, shape, type, and structure by the market situation. A computer vision system was developed to distinguish seven different registered varieties of dry beans with similar features in order to obtain uniform seed classification. For the classification model, images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by computer vision system were subjected to segmentation and feature extraction stages, and a total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.

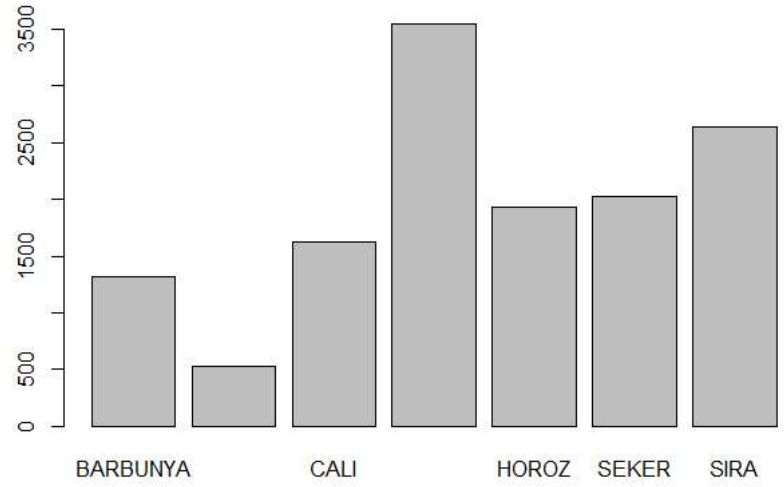
```
##    Area Perimeter MajorAxisLength MinorAxisLength AspectRation Eccentricity
## 1 28395   610.291        208.1781        173.8887     1.197191    0.5498122
## 2 28734   638.018        200.5248        182.7344     1.097356    0.4117853
## 3 29380   624.110        212.8261        175.9311     1.209713    0.5627273
## 4 30008   645.884        210.5580        182.5165     1.153638    0.4986160
## 5 30140   620.134        201.8479        190.2793     1.060798    0.3336797
## 6 30279   634.927        212.5606        181.5102     1.171067    0.5204007
##   ConvexArea EquivDiameter    Extent  Solidity roundness Compactness
## 1      28715      190.1411 0.7639225 0.9888560 0.9580271   0.9133578
## 2      29172      191.2728 0.7839681 0.9849856 0.8870336   0.9538608
## 3      29690      193.4109 0.7781132 0.9895588 0.9478495   0.9087742
## 4      30724      195.4671 0.7826813 0.9766957 0.9039364   0.9283288
## 5      30417      195.8965 0.7730980 0.9908932 0.9848771   0.9705155
## 6      30600      196.3477 0.7756885 0.9895098 0.9438518   0.9237260
##   ShapeFactor1 ShapeFactor2 ShapeFactor3 ShapeFactor4 Variety
## 1  0.007331506  0.003147289    0.8342224    0.9987239   SEKER
## 2  0.006978659  0.003563624    0.9098505    0.9984303   SEKER
## 3  0.007243912  0.003047733    0.8258706    0.9990661   SEKER
## 4  0.007016729  0.003214562    0.8617944    0.9941988   SEKER
## 5  0.006697010  0.003664972    0.9419004    0.9991661   SEKER
## 6  0.007020065  0.003152779    0.8532696    0.9992358   SEKER
```

Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent
28395	610.291	208.1781	173.8887	1.197191	0.5498122	28715	190.1411	0.7639225
28734	638.018	200.5248	182.7344	1.097357	0.4117853	29172	191.2728	0.7839681
29380	624.110	212.8261	175.9311	1.209713	0.5627273	29690	193.4109	0.7781132
30008	645.884	210.5580	182.5165	1.153638	0.4986160	30724	195.4671	0.7826813
30140	620.134	201.8479	190.2793	1.060798	0.3336797	30417	195.8965	0.7730980
30279	634.927	212.5606	181.5102	1.171067	0.5204007	30600	196.3477	0.7756885
30477	670.033	211.0502	184.0391	1.146768	0.4894779	30970	196.9886	0.7624015
30519	629.727	212.9968	182.7372	1.165590	0.5137596	30847	197.1243	0.7706818
30685	635.681	213.5341	183.1571	1.165852	0.5140809	31044	197.6597	0.7715615
30834	631.934	217.2278	180.8975	1.200834	0.5536422	31120	198.1390	0.7836828
30917	640.765	213.5601	184.4399	1.157885	0.5041024	31280	198.4055	0.7708053
31091	638.558	210.4863	188.3268	1.117665	0.4466219	31458	198.9630	0.7863773
31107	640.594	214.6485	184.9693	1.160455	0.5073659	31423	199.0142	0.7610461
31158	642.626	216.4848	183.6443	1.178827	0.5295143	31492	199.1773	0.7987592
31158	641.105	212.0670	187.1930	1.132879	0.4699242	31474	199.1773	0.7813135
31178	636.888	212.9759	186.5621	1.141582	0.4823522	31520	199.2412	0.7641105
31202	644.454	215.6407	184.4717	1.168964	0.5178712	31573	199.3179	0.7791929
31203	639.782	215.0677	184.8749	1.163315	0.5109468	31558	199.3211	0.7629842
31272	638.666	212.4503	187.5359	1.132851	0.4698835	31593	199.5413	0.7703222
31335	635.011	216.7901	184.1634	1.177161	0.5275867	31599	199.7422	0.7742772

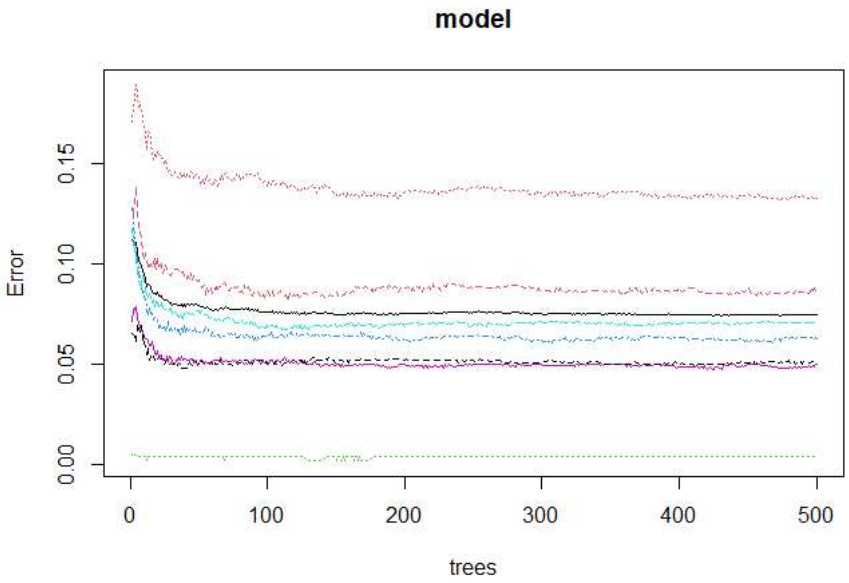
```
## Warning: package 'multiDimBio' was built under R version 4.2.1
```

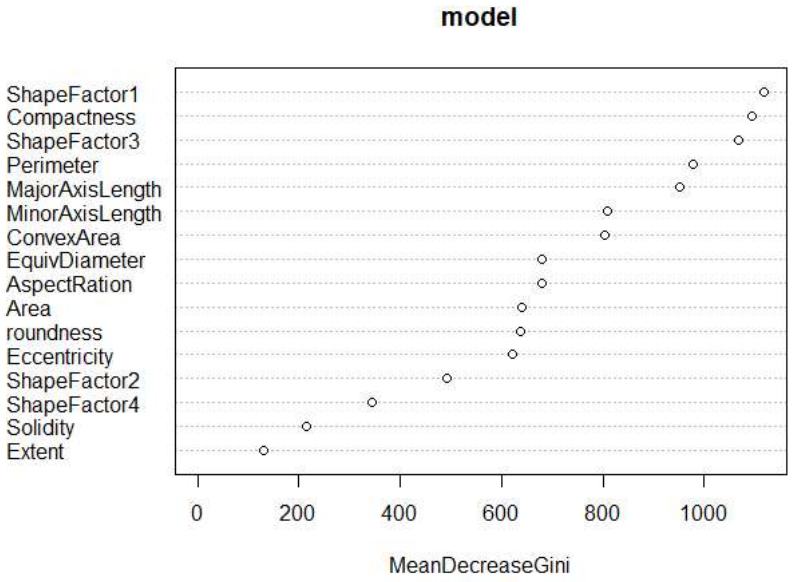

score
4
0
-4
Area
AspectRation
ConvexArea
Eccentricity
EquivDiameter
MajorAxisLength
MinorAxisLength
Perimeter
trait
group
BARBUNYA
BOMBAY
CALI
DERMASON
HOROZ
SEKER
SIRA


score
0
-5
-10
-15
Compactness
Extent
roundness
ShapeFactor3
ShapeFactor2
ShapeFactor1
ShapeFactor4
Solidity
trait
group
BARBUNYA
BOMBAY
CALI
DERMASON
HOROZ
SEKER
SIRA

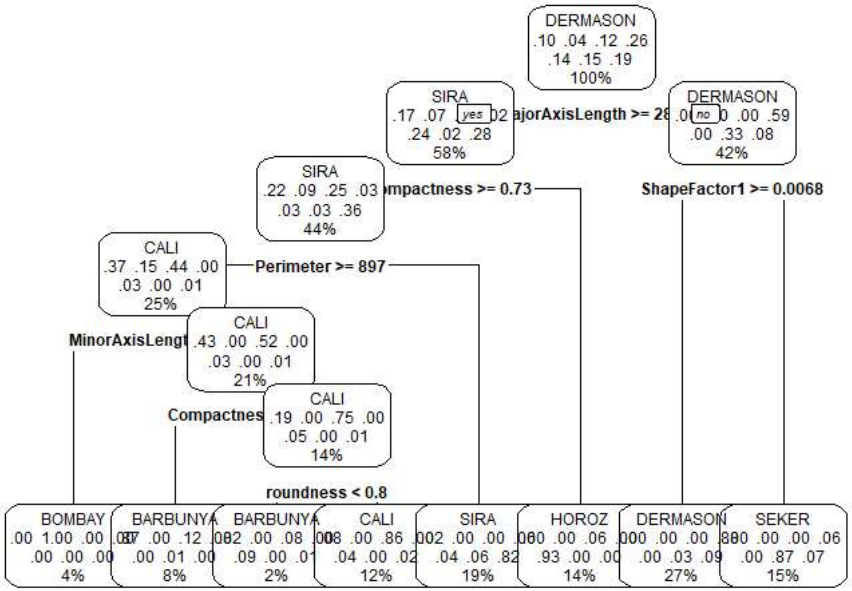



```
##
## Call:
##  randomForest(formula = as.factor(Variety) ~ ., data = frijol)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 7.5%
## Confusion matrix:
##          BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA class.error
## BARBUNYA     1207      1   69        0     8    11   26 0.086989410
## BOMBAY          2    520    0        0     0     0    0 0.003831418
## CALI           55      0 1528        0    32     3   12 0.062576687
## DERMASON        0      0    0     3294     5    54  193 0.071065990
## HOROZ           7      0   29       13  1832     0   47 0.049792531
## SEKER           7      0    0       51     0  1923   46 0.051307351
## SIRA            8      0    7      257    37    41 2286 0.132776935
```






```
## Warning: All boxes will be white (the box.palette argument will be ignored) because
## the number of classes in the response 7 is greater than length(box.palette) 6.
## To silence this warning use box.palette=0 or trace=-1.
```



El paquete Ranger es una forma rápida de implementar el modelo de bosque aleatorio en big data. "Ranger" es la abreviatura de RANdom Forest GEneRator.

// Prediccion y validacion cruzada

```
## Warning: package 'caret' was built under R version 4.2.1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
##   BARBUNYA      228      1   21        0     1     0    2
##   BOMBAY          0    100    0        0     0     0    0
##   CALI           18      0  284        0     7     0    4
##   DERMASON        0      0    0      653     3     9   60
##   HOROZ           3      0    8        1   374     0    4
##   SEKER           1      0    0       10     0   395   10
##   SIRA            8      0    5       47    12     9  442
##
## Overall Statistics
##
##                Accuracy : 0.9103
```

```
##                 95% CI : (0.8989, 0.9208)
##     No Information Rate : 0.2614
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8913
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: BARBUNYA Class: BOMBAY Class: CALI Class: DERMASON
## Sensitivity                  0.88372       0.99010      0.8931          0.9184
## Specificity                  0.98985       1.00000      0.9879          0.9642
## Pos Pred Value               0.90119       1.00000      0.9073          0.9007
## Neg Pred Value               0.98784       0.99962      0.9859          0.9709
## Prevalence                   0.09485       0.03713      0.1169          0.2614
## Detection Rate               0.08382       0.03676      0.1044          0.2401
## Detection Prevalence         0.09301       0.03676      0.1151          0.2665
## Balanced Accuracy            0.93678       0.99505      0.9405          0.9413
##                      Class: HOROZ Class: SEKER Class: SIRA
## Sensitivity                0.9421       0.9564      0.8467
## Specificity                0.9931       0.9909      0.9631
## Pos Pred Value             0.9590       0.9495      0.8451
## Neg Pred Value             0.9901       0.9922      0.9636
## Prevalence                 0.1460       0.1518      0.1919
## Detection Rate             0.1375       0.1452      0.1625
## Detection Prevalence       0.1434       0.1529      0.1923
## Balanced Accuracy          0.9676       0.9737      0.9049
```

// Boosting Xgboost

El impulso es una técnica de conjunto secuencial en la que el modelo se mejora utilizando la información de modelos más débiles desarrollados anteriormente. Este proceso continúa durante múltiples iteraciones hasta que se construye un modelo final que predirá un resultado más preciso. Hay 3 tipos de técnicas de impulso: 1. Adaboost 2. Descenso de gradiente. 3. Xgboost (aumento de gradiente extremo) es una versión avanzada de la técnica de aumento de gradiente descendente, que se utiliza para aumentar la velocidad y la eficiencia del cálculo del algoritmo.