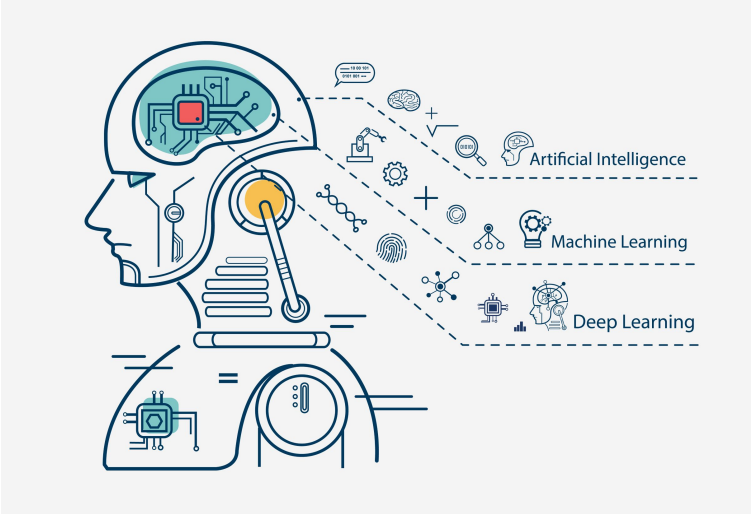
Proyecto final

COA-501 Herramientas de cómputo para investigadores (R y Python)

Iván F. Quiroz Ibáñez*

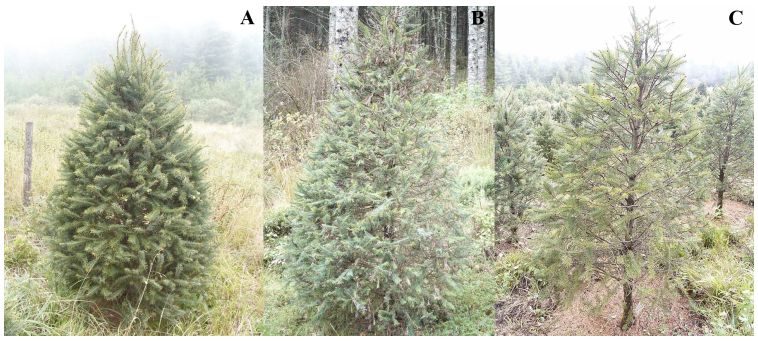
17 de noviembre de 2022



*IFIT-CP, quiroz.ivan@colpos.mx

0) Estudio de caso

Se cuenta con la base de datos de un experimento realizado en Puebla, dónde se evaluaron los efectos de fertilizantes y plaguicidas contra una enfermedad foliar conocida como tizón suizo en árboles de navidad de la especie Douglas-fir. Se tomaron variables como transparencia de copa, severidad, incidencia, longitud de brotes, área de ácículas, etc. El objetivo del análisis es determinar la mejor técnica de machine learning o aprendizaje automático (Naive Bayes, K-NN y Random Forest) para clasificar la transparencia de copa en Douglas-fir.



1) Base de datos y AED

```
#base de datos
summary(base)
```

```
##      Arbol       Bloque       Longitud         Latitud          X_UTM
##  1      : 12   Alta :216   Min.   :-97.99   Min.   :19.72   Min.   :605708
##  2      : 12   Baja :216   1st Qu.:-97.99   1st Qu.:19.72   1st Qu.:605724
##  3      : 12   Media:216   Median :-97.99   Median :19.72   Median :605732
##  4      : 12               Mean   :-97.99   Mean   :19.72   Mean   :605732
##  5      : 12               3rd Qu.:-97.99   3rd Qu.:19.72   3rd Qu.:605741
##  6      : 12               Max.   :-97.99   Max.   :19.72   Max.   :605759
##  (Other):576
##      Y_UTM            Altitud      AltRan    Fungicida Nutrimento      Rep
##  Min.   :2180985   Min.   :2973   Alta:312   1:216     1:216      Min.   :1.0
##  1st Qu.:2181009   1st Qu.:2979   Baja:336   2:216     2:216      1st Qu.:1.0
##  Median :2181025   Median :2982              3:216     3:216      Median :1.5
##  Mean   :2181027   Mean   :2985                                   Mean   :1.5
##  3rd Qu.:2181039   3rd Qu.:2991                                   3rd Qu.:2.0
##  Max.   :2181087   Max.   :2994                                   Max.   :2.0
##
##  Muestreo  DDA        Anio         AcicR             Abs
##  1:108    0  :108   2016:324   Min.   : 30.00   Min.   :  0.00
##  2:108    63 :108   2017:324   1st Qu.: 83.75   1st Qu.:  0.00
##  3:108    91 :108              Median :101.00   Median :  2.00
##  4:108    119:108              Mean   :103.14   Mean   :  6.91
##  5:108    147:108              3rd Qu.:121.00   3rd Qu.:  8.00
##  6:108    175:108              Max.   :225.00   Max.   :104.00
##                                NA's   :24       NA's   :24
##     TotalAc           Inc              Sevmed            Sevmin
##  Min.   : 36.0   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
```

```
##  1st Qu.: 91.0   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :108.0   Median :0.2667   Median :0.01483   Median :0.00000
##  Mean   :110.1   Mean   :0.4584   Mean   :0.12498   Mean   :0.06102
##  3rd Qu.:128.0   3rd Qu.:0.9667   3rd Qu.:0.24258   3rd Qu.:0.10333
##  Max.   :225.0   Max.   :1.0000   Max.   :0.61900   Max.   :0.50667
##  NA's   :24      NA's   :24       NA's   :24        NA's   :24
##      Sevmax           LonBrot          indcol          Afmed
##  Min.   :0.00000   Min.   : 3.300   Min.   : 0.00   Min.   :0.1322
##  1st Qu.:0.00000   1st Qu.: 7.000   1st Qu.: 0.00   1st Qu.:0.2610
##  Median :0.07333   Median : 8.450   Median : 0.46   Median :0.3096
##  Mean   :0.20214   Mean   : 8.629   Mean   :11.72   Mean   :0.3117
##  3rd Qu.:0.40083   3rd Qu.: 9.963   3rd Qu.:22.85   3rd Qu.:0.3577
##  Max.   :0.90667   Max.   :23.400   Max.   :61.90   Max.   :0.6133
##  NA's   :24        NA's   :24       NA's   :24      NA's   :24
##      Afmax           Afmin           Aftotal            CA
##  Min.   :0.152   Min.   :0.0910   Min.   : 3.966   Min.   : 0.000
##  1st Qu.:0.300   1st Qu.:0.2087   1st Qu.: 7.831   1st Qu.: 0.000
##  Median :0.352   Median :0.2470   Median : 9.287   Median : 1.646
##  Mean   :0.358   Mean   :0.2486   Mean   : 9.350   Mean   : 6.178
##  3rd Qu.:0.411   3rd Qu.:0.2870   3rd Qu.:10.730   3rd Qu.: 7.692
##  Max.   :0.849   Max.   :0.4440   Max.   :18.398   Max.   :64.662
##  NA's   :24      NA's   :24       NA's   :24       NA's   :24
##        RA              Color            trat        ABCPEIC
##  Min.   : 35.34   Min.   :1.000   11     : 72   Min.   :   0.0000
##  1st Qu.: 92.31   1st Qu.:1.000   12     : 72   1st Qu.:   0.0000
##  Median : 98.35   Median :2.000   13     : 72   Median :   0.1925
##  Mean   : 93.82   Mean   :1.694   21     : 72   Mean   : 326.8269
##  3rd Qu.:100.00   3rd Qu.:2.000   22     : 72   3rd Qu.: 683.7950
##  Max.   :100.00   Max.   :3.000   23     : 72   Max.   :2850.9250
##  NA's   :24       NA's   :24      (Other):216
```

```
class(base)
```

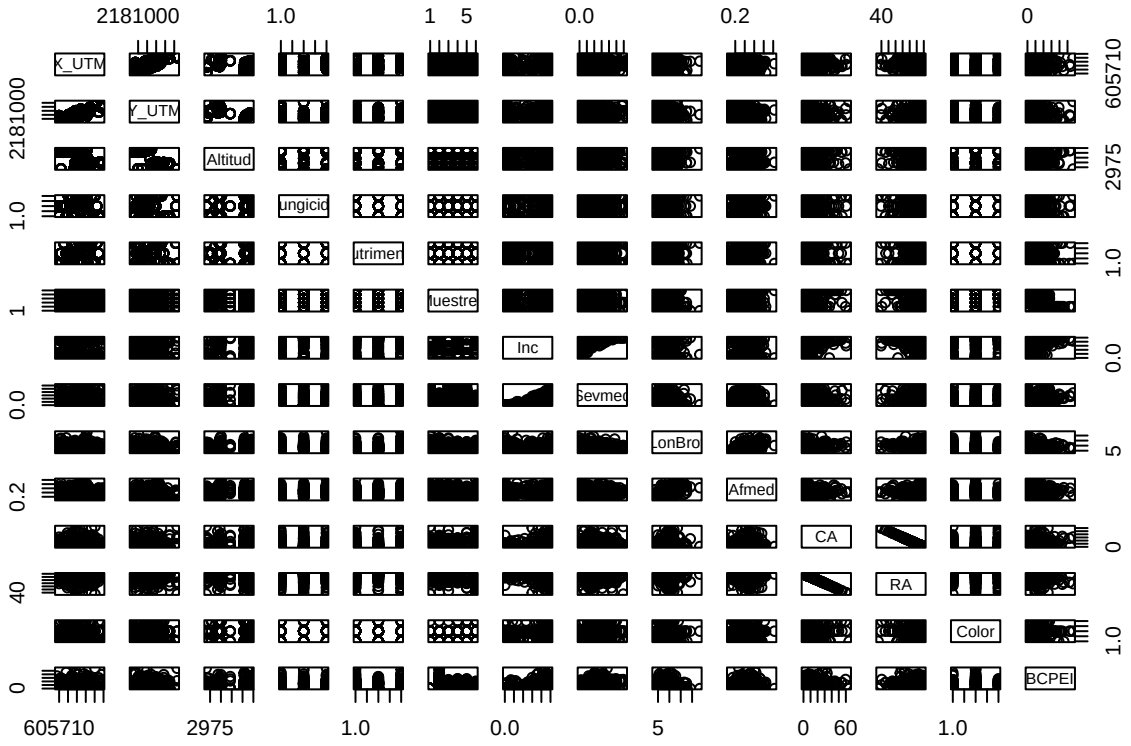
```
## [1] "data.frame"
```

```
str(base)
```

```
## 'data.frame':    648 obs. of  32 variables:
##  $ Arbol     : Factor w/ 54 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Bloque    : Factor w/ 3 levels "Alta","Baja",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Longitud  : num  -98 -98 -98 -98 -98 ...
##  $ Latitud   : num  19.7 19.7 19.7 19.7 19.7 ...
##  $ X_UTM     : num  605748 605744 605740 605754 605742 ...
##  $ Y_UTM     : num  2181070 2181057 2181039 2181034 2181026 ...
##  $ Altitud   : num  2979 2979 2980 2980 2984 ...
##  $ AltRan    : Factor w/ 2 levels "Alta","Baja": 2 2 2 2 2 1 1 1 1 1 ...
##  $ Fungicida : Factor w/ 3 levels "1","2","3": 1 1 2 1 2 1 1 2 2 3 ...
##  $ Nutrimento: Factor w/ 3 levels "1","2","3": 1 2 2 2 1 3 3 3 2 1 ...
##  $ Rep       : int  1 1 1 2 1 1 2 1 2 1 ...
##  $ Muestreo  : Factor w/ 6 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ DDA       : Factor w/ 6 levels "0","63","91",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Anio      : Factor w/ 2 levels "2016","2017": 1 1 1 1 1 1 1 1 1 1 ...
##  $ AcicR     : int  74 129 69 33 98 71 68 44 92 92 ...
```

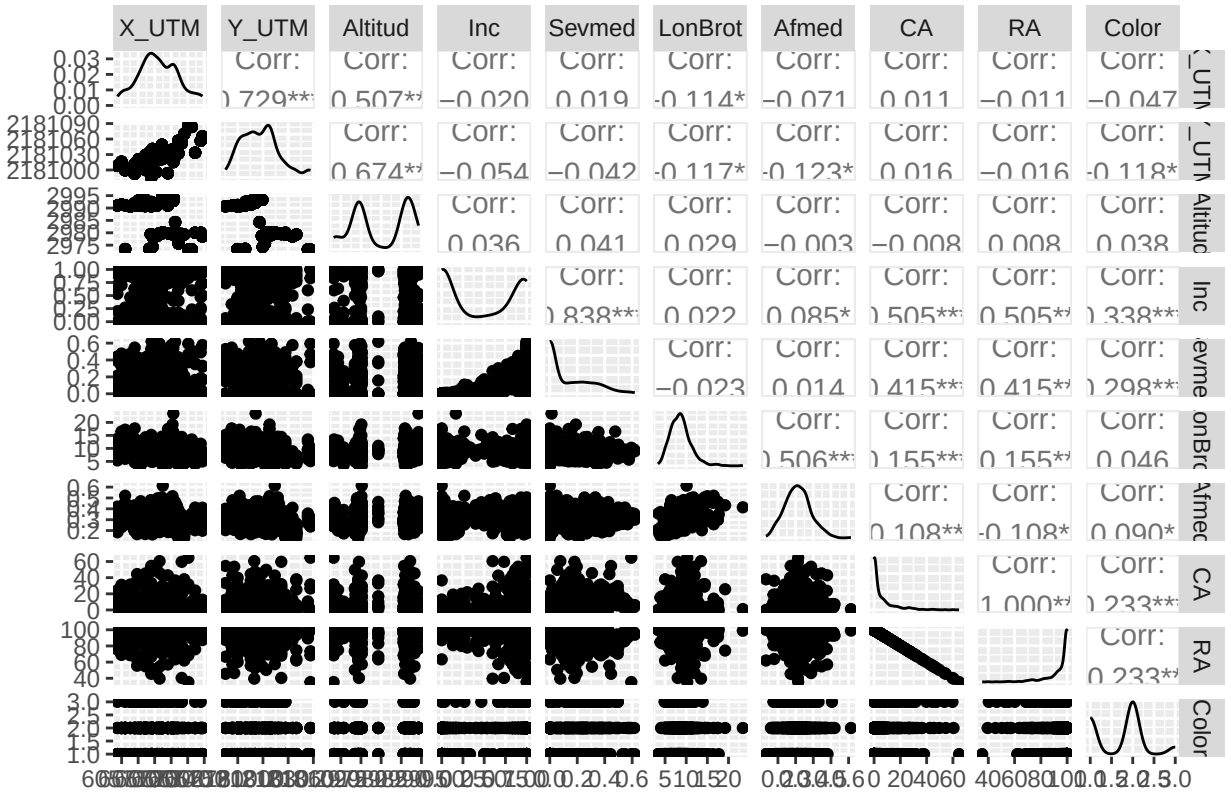
```
##  $ Abs       : int  25 2 2 6 20 15 3 5 4 15 ...
##  $ TotalAc   : int  99 131 71 39 118 86 71 49 96 107 ...
##  $ Inc       : num  0.9 1 0.767 1 1 ...
##  $ Sevmed    : num  0.15 0.331 0.347 0.351 0.384 ...
##  $ Sevmin    : num  0 0.137 0 0.21 0.197 ...
##  $ Sevmax    : num  0.4 0.667 0.583 0.573 0.56 ...
##  $ LonBrot   : num  7.25 12.9 5.1 3.9 9.25 5 6.8 4.5 7.7 9.55 ...
##  $ indcol    : num  13.5 33.1 26.6 35.1 38.4 ...
##  $ Afmed     : num  0.239 0.374 0.196 0.37 0.362 ...
##  $ Afmax     : num  0.275 0.436 0.217 0.411 0.482 ...
##  $ Afmin     : num  0.194 0.308 0.163 0.286 0.29 ...
##  $ Aftotal   : num  7.18 11.23 5.88 11.1 10.87 ...
##  $ CA        : num  25.25 1.53 2.82 15.38 16.95 ...
##  $ RA        : num  74.7 98.5 97.2 84.6 83.1 ...
##  $ Color     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ trat      : Factor w/ 9 levels "11","12","13",..: 1 2 5 2 4 3 3 6 5 7 ...
##  $ ABCPEIC   : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
#matriz de diagramas de dispersion
pairs(base[,c(5,6,7,9,10,12,18,19,22,24,28:30,32)])

library(GGally)
```

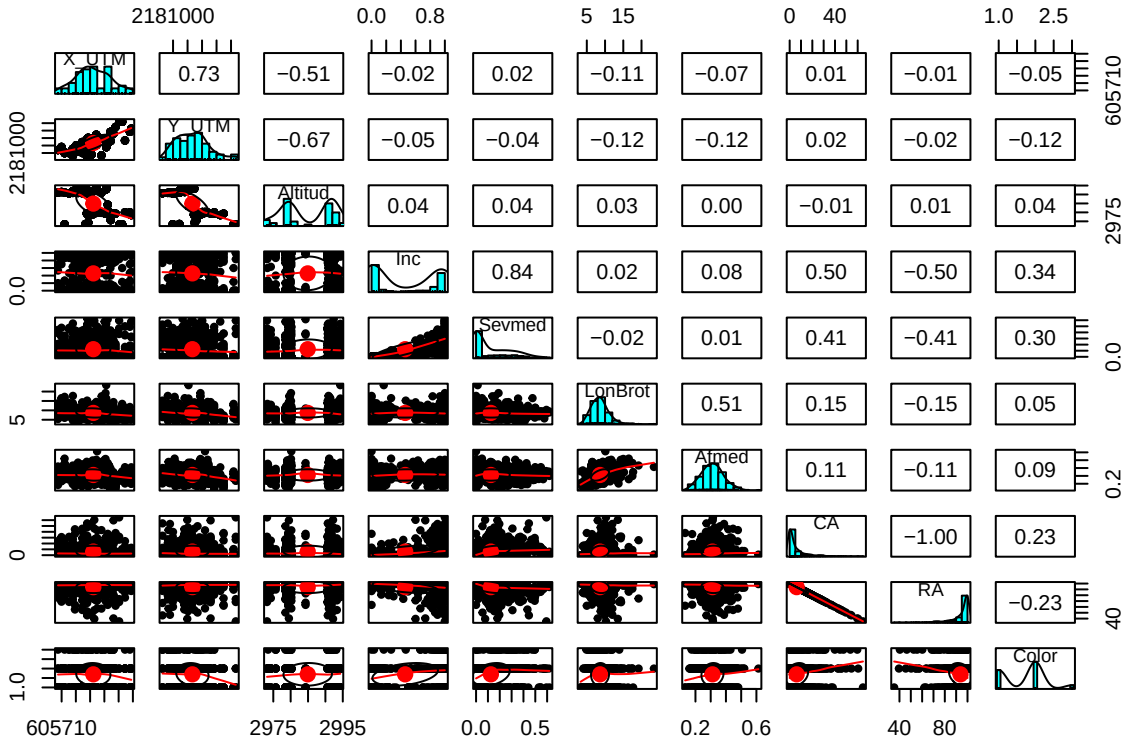



```
ggpairs(base[,c(5,6,7,18,19,22,24,28:30)],
        title="correlograma con ggpairs")
```

correlograma con ggpairs



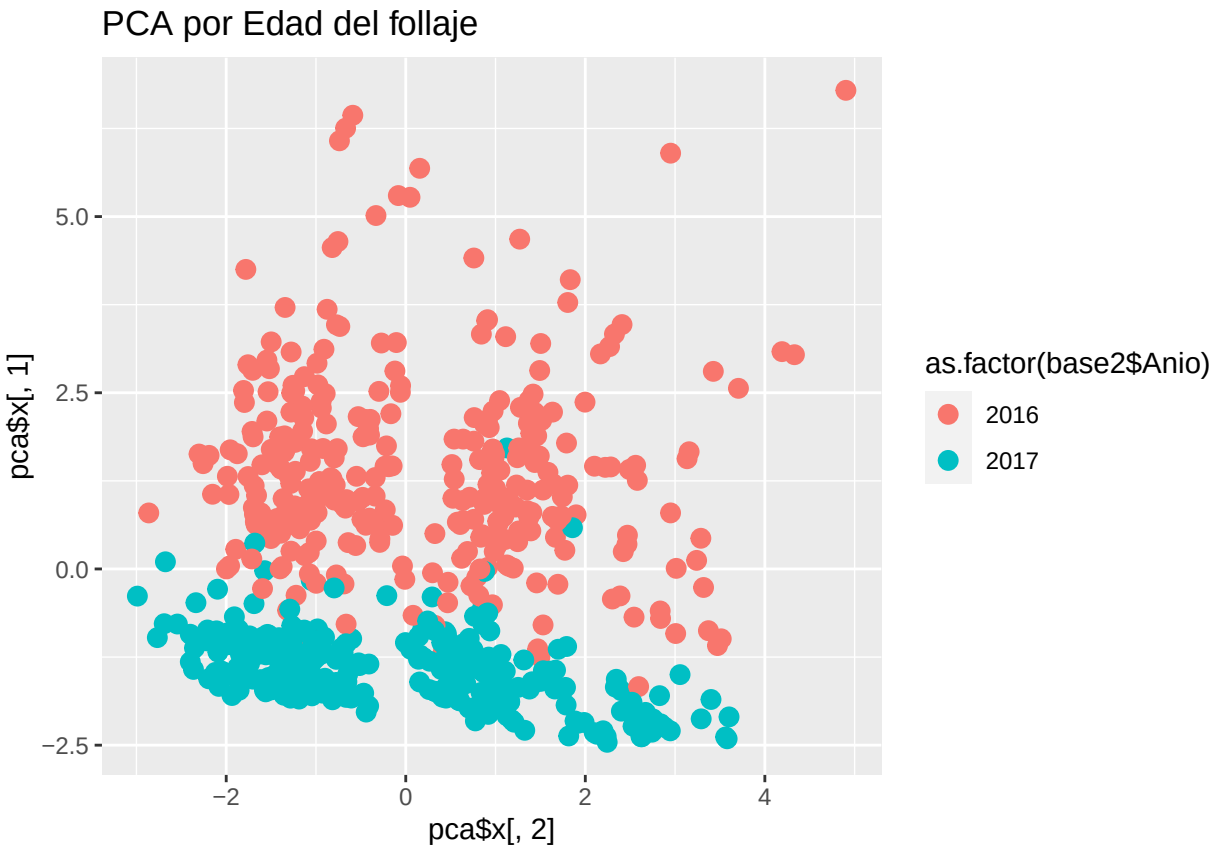

```
psych::pairs.panels(base[,c(5,6,7,18,19,22,24,28:30)])
```



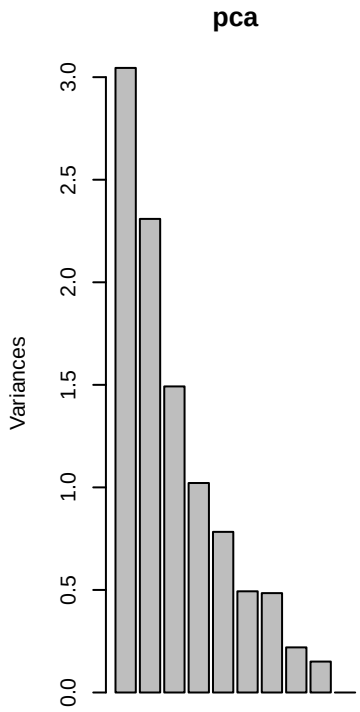

```
base2<- na.omit(base[,c(14,2,5,6,7,18,19,22,24,28:30)])

#Análisis de componentes principales
pca <- prcomp(base2[,-c(1,2)], scale = T, center=T)
summary(pca)
```

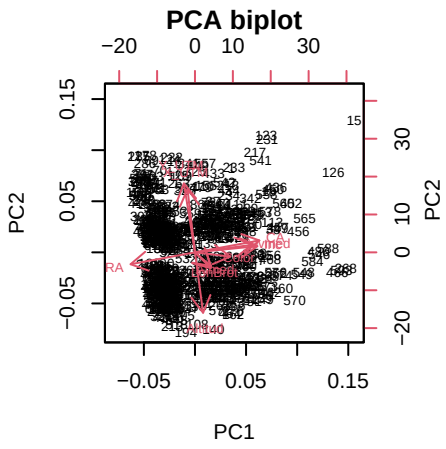
```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.7451 1.5196 1.2216 1.0107 0.88496 0.70241 0.69614
## Proportion of Variance 0.3045 0.2309 0.1492 0.1021 0.07832 0.04934 0.04846
## Cumulative Proportion  0.3045 0.5354 0.6847 0.7868 0.86512 0.91446 0.96292
##                           PC8     PC9      PC10
## Standard deviation     0.46896 0.38840 1.265e-10
## Proportion of Variance 0.02199 0.01509 0.000e+00
## Cumulative Proportion  0.98491 1.00000 1.000e+00
```

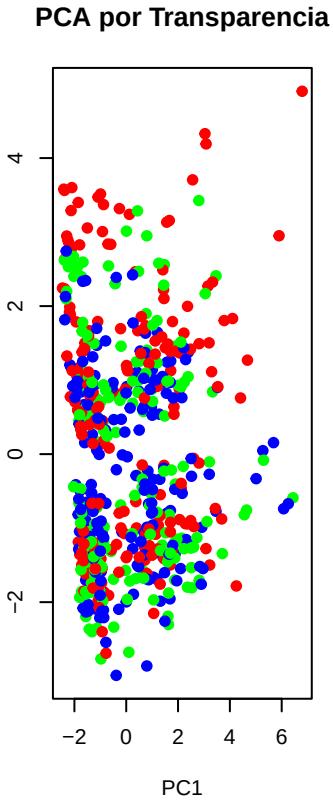
```
#biplot
library(ggplot2)
  ggplot(base2, aes(x = pca$x[,2], y = pca$x[,1],
  colour = as.factor(base2$Anio)))+
  geom_point(size=3) +
  ggtitle("PCA por Edad del follaje")
```




```
par(mfrow = c(1,3))
plot(pca)
biplot(pca, main="PCA biplot")

colores <- function(vec){
  # la funci?n rainbow() devuelve un vector que contiene el n?mero de colores distintos
  col <- rainbow(length(unique(vec)))
  return(col[as.numeric(as.factor(vec))])
}

plot(pca$x[,c(1, 2)], col = colores(base$Bloque),
     pch = 19,
     xlab = "PC1",
     ylab = "PC2", main="PCA por Transparencia")
```








```
dev.off()
```

```
## null device
##           1
```

Interpretación: De acuerdo con el AED y ACP se seleccionaron las variables transparencia de copa, coordenadas UTM (X y Y), altitud, incidencia, severidad, longitud de brote, área de acícula, acículas caidas, acículas retenidas y color de acícula, para implementar las técnicas de ML.

2) Random Forest

```
library(randomForest)
library(caret)

#Getting Data
base_rf <- na.omit(base[,c(2,5,6,7,18,19,22,24,28:30)])

base_rf$Bloque <- as.factor(base_rf$Bloque)
table(base_rf$Bloque)
```

```
##
##  Alta  Baja Media
##   200   216   208
```

```
#Data Partition
set.seed(123)
ind <- sample(2, nrow(base_rf), replace = TRUE, prob = c(0.7, 0.3))
train <- base_rf[ind==1,]
test <- base_rf[ind==2,]

#Random Forest in R
rf <- randomForest(Bloque~., data=train, proximity=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = Bloque ~ ., data = train, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 2.28%
## Confusion matrix:
##       Alta Baja Media class.error
## Alta   132    4     1  0.03649635
## Baja     0  160     0  0.00000000
## Media    4    1   136  0.03546099
```

```
#Confusion Matrix and Statistics
p1 <- predict(rf, train)
confusionMatrix(p1, train$Bloque)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Alta Baja Media
##      Alta   137    0     0
##      Baja     0  160     0
##      Media    0    0   141
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9916, 1)
##     No Information Rate : 0.3653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Alta Class: Baja Class: Media
## Sensitivity               1.0000      1.0000       1.0000
## Specificity               1.0000      1.0000       1.0000
## Pos Pred Value            1.0000      1.0000       1.0000
```

```
## Neg Pred Value            1.0000      1.0000       1.0000
## Prevalence                0.3128      0.3653       0.3219
## Detection Rate            0.3128      0.3653       0.3219
## Detection Prevalence      0.3128      0.3653       0.3219
## Balanced Accuracy         1.0000      1.0000       1.0000
```

```
(tab1 <- table(p1, train$Bloque))
```

```
##
## p1      Alta Baja Media
##   Alta   137    0     0
##   Baja     0  160     0
##   Media    0    0   141
```

```
1 - sum(diag(tab1)) / sum(tab1)
```

```
## [1] 0
```

```
#error del 0%


p2 <- predict(rf, test)
confusionMatrix(p2, test$Bloque)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Alta Baja Media
##      Alta    60    0     0
##      Baja     2   56     0
##      Media    1    0    67
##
## Overall Statistics
##
##                Accuracy : 0.9839
##                  95% CI : (0.9536, 0.9967)
##     No Information Rate : 0.3602
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9758
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Alta Class: Baja Class: Media
## Sensitivity               0.9524      1.0000       1.0000
## Specificity               1.0000      0.9846       0.9916
## Pos Pred Value            1.0000      0.9655       0.9853
## Neg Pred Value            0.9762      1.0000       1.0000
## Prevalence                0.3387      0.3011       0.3602
## Detection Rate            0.3226      0.3011       0.3602
```

```
## Detection Prevalence      0.3226      0.3118      0.3656
## Balanced Accuracy         0.9762      0.9923      0.9958
```

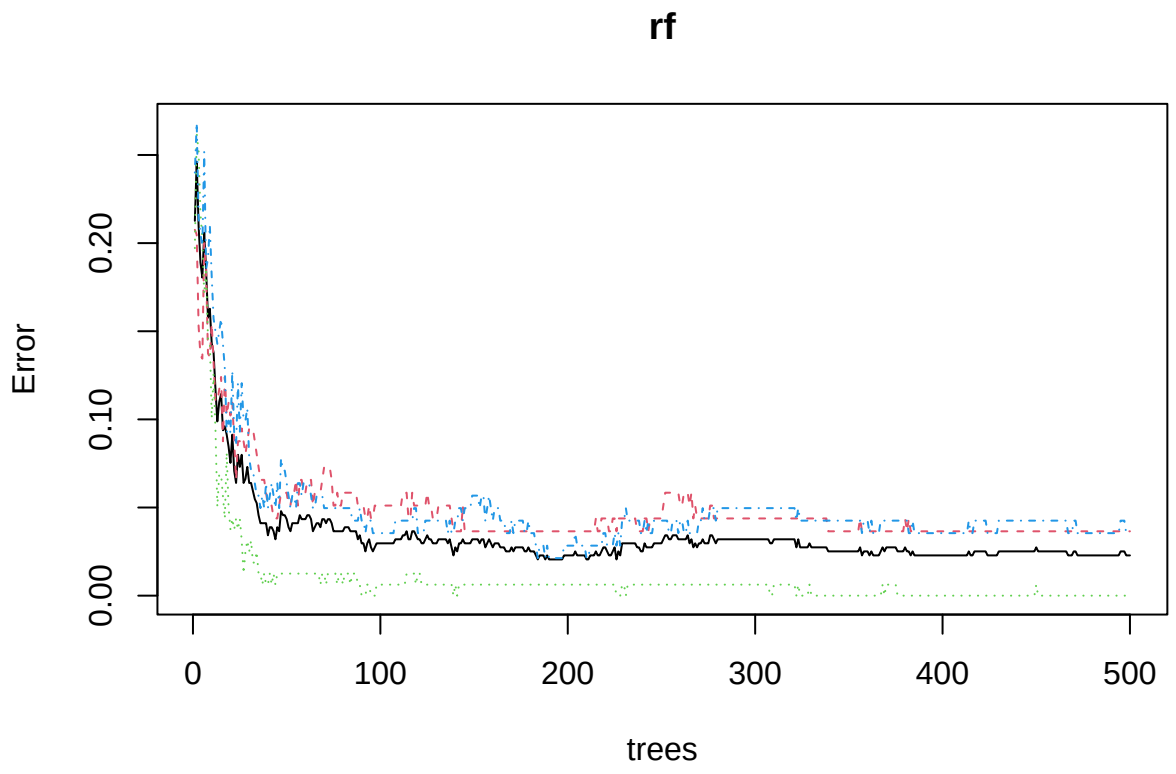
```
(tab2 <- table(p2, test$Bloque))
```

```
##
## p2      Alta Baja Media
##   Alta    60    0     0
##   Baja     2   56     0
##   Media    1    0    67
```

```
1 - sum(diag(tab2)) / sum(tab2)
```

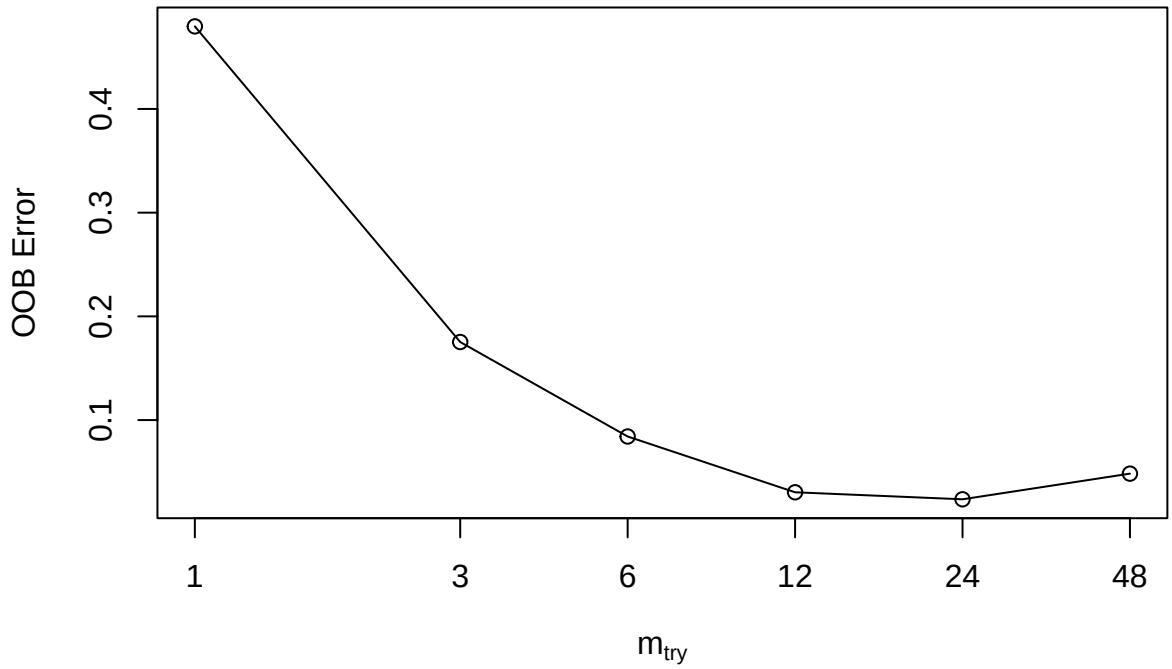
```
## [1] 0.01612903
```

```
#error del 0%


#Error rate of Random Forest
plot(rf)
```



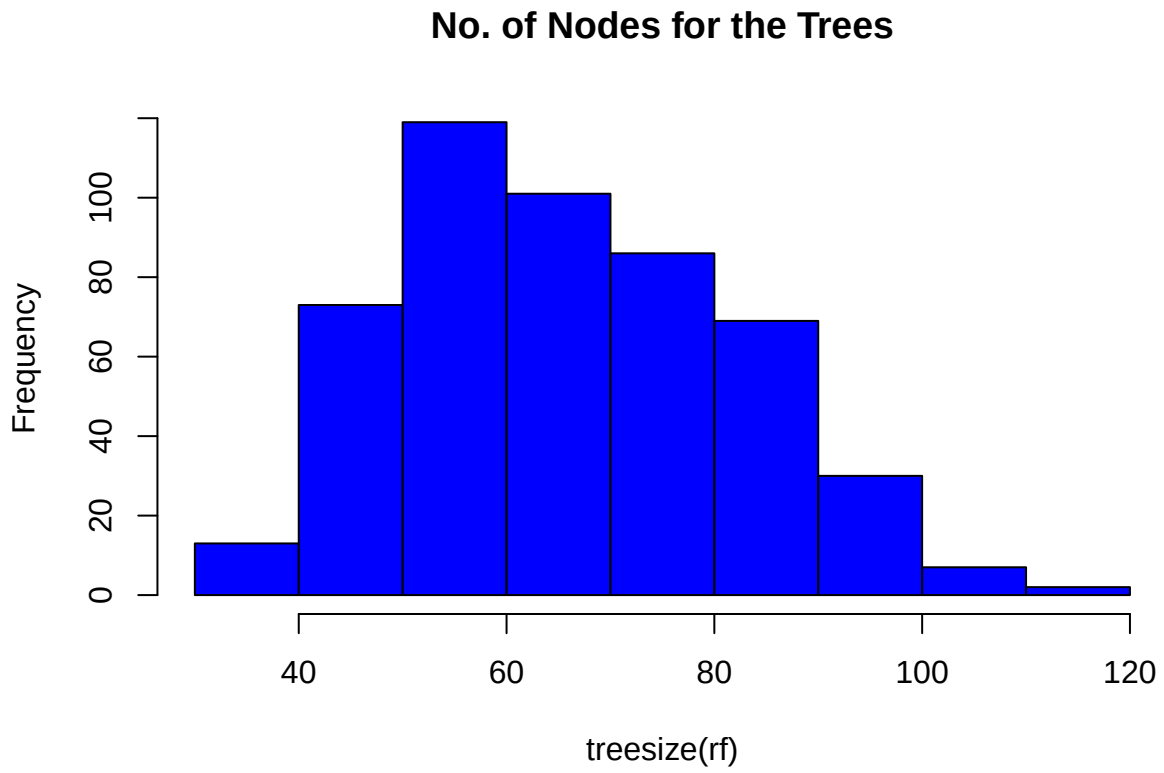
```
#Tune mtry (Número de variables aleatorias utilizadas en cada árbol)
t <- tuneRF(train[,-1], train[,1],
            stepFactor = 0.5,
            plot = TRUE,
            ntreeTry = 5,
            trace = TRUE,
            improve = 0.05)
```

```
## mtry = 3  OOB error = 17.53%
## Searching left ...
## mtry = 6     OOB error = 8.42%
## 0.519802 0.05
## mtry = 12    OOB error = 3.04%
## 0.6390171 0.05
## mtry = 24    OOB error = 2.36%
## 0.2224409 0.05
## mtry = 48    OOB error = 4.83%
## -1.04665 0.05
## Searching right ...
## mtry = 1     OOB error = 47.96%
## -19.30272 0.05
```

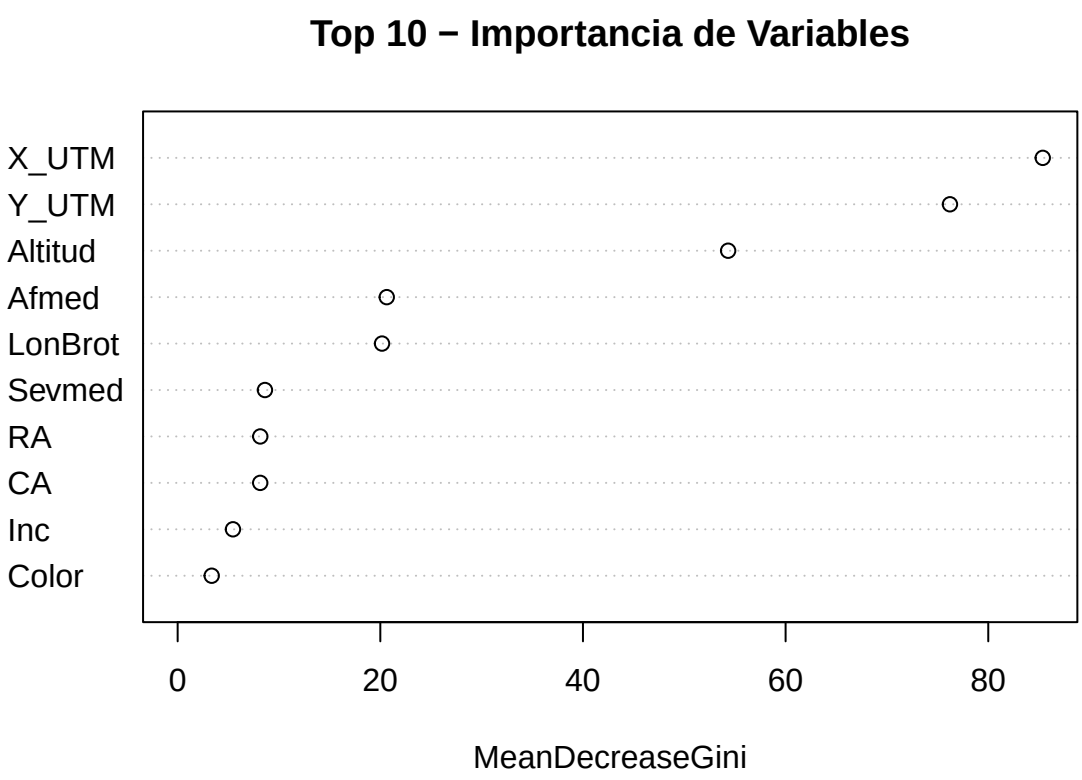



```
#mtry=6

#No. of nodes for the trees
```

```
hist(treesize(rf),
     main = "No. of Nodes for the Trees",
     col = "blue")
```



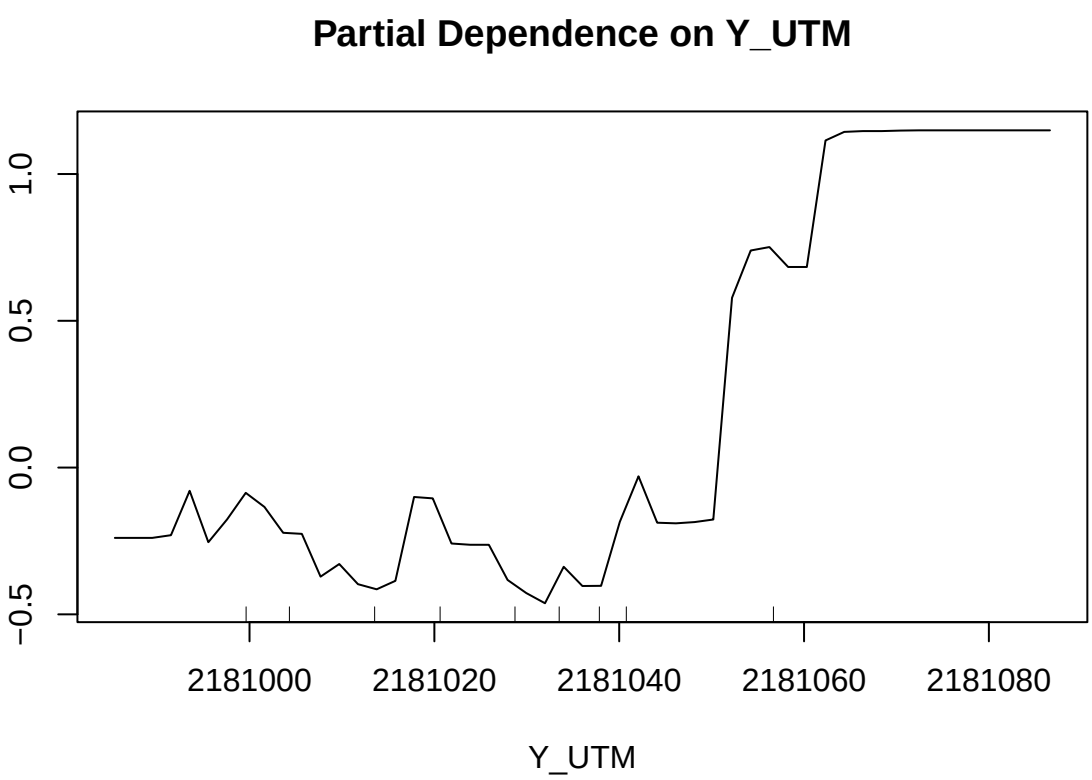

```
#media de 60 árboles

#Variable Importance
varImpPlot(rf,
           sort = T,
           n.var = 10,
           main = "Top 10 - Importancia de Variables")
```




```
importance(rf)
```

```
##          MeanDecreaseGini
## X_UTM           85.378578
## Y_UTM           76.219320
## Altitud         54.328330
## Inc              5.457896
## Sevmed           8.613401
## LonBrot         20.168919
## Afmed           20.634459
## CA               8.147649
## RA               8.153187
## Color            3.360461
```

```
#Partial Dependence Plot
partialPlot(rf, train, Y_UTM, "Alta")
```



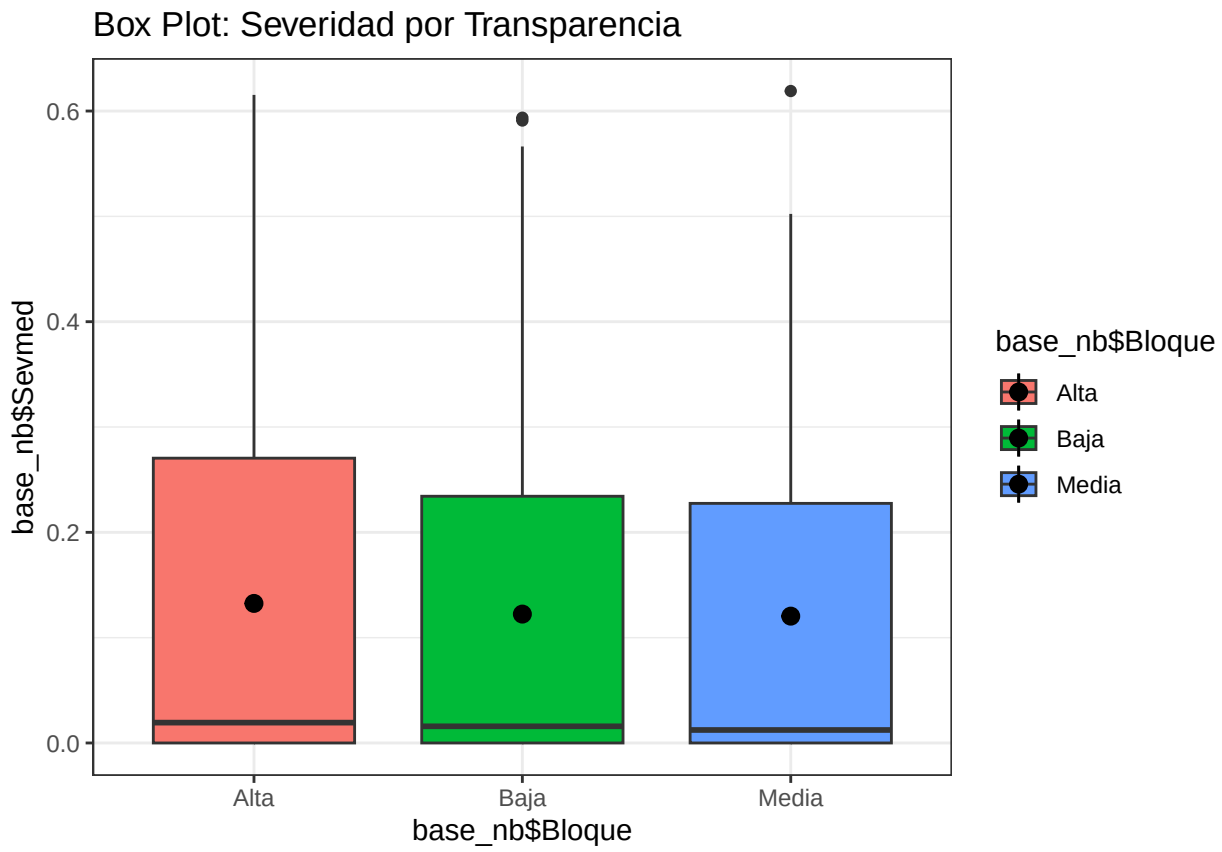
Interpretación: se obtuvo una precisión del 98% y un valor κ= 0.98, una mtry (Número de variables aleatorias utilizadas en cada árbol) de 6, 60 nodos promedio por árbol, las variables de mayor peso son las coordenadas UTM (X y Y).

3) Naive Bayes

```
#clasificacion en datos poco correlacionados
#https://www.r-bloggers.com/2021/04/naive-bayes-classification-in-r/
#Para segmentar imagenes (severidad)
#https://plantcv.readthedocs.io/en/latest/tutorials/machine_learning_tutorial/
library(naivebayes)
library(dplyr)
library(ggplot2)


base_nb <- na.omit(base[,c(2,5,6,7,18,19,22,24,28:30)])

#Dplyr
base_nb$Bloque <- as.factor(base_nb$Bloque)
table(base_nb$Bloque)
```

```
##
##  Alta  Baja Media
##   200   216   208
```

```
base_nb %>%
  ggplot(aes(x=base_nb$Bloque
             , y=base_nb$Sevmed, fill = base_nb$Bloque)) +
  geom_boxplot() +theme_bw()+stat_summary(fun="mean")+
  ggtitle("Box Plot: Severidad por Transparencia")
```




```
#particion de datos
set.seed(1234)
ind <- sample(2, nrow(base_nb), replace = T, prob = c(0.7, 0.3))
train_nb <- base_nb[ind == 1,]
test_nb <- base_nb[ind == 2,]



nb <- naive_bayes(Bloque ~ ., data = train_nb, usekernel = T)
nb
```

```
##
## ================================== Naive Bayes ==================================
##
##  Call:
## naive_bayes.formula(formula = Bloque ~ ., data = train_nb, usekernel = T)
##
## ---------------------------------------------------------------------------------
##
```

```
## Laplace smoothing: 0
##
## -------------------------------------------------------------------------------
##
##  A priori probabilities:
##
##      Alta      Baja     Media
## 0.3294931 0.3525346 0.3179724
##
## -------------------------------------------------------------------------------
##
##  Tables:
##
## -------------------------------------------------------------------------------
##  ::: X_UTM::Alta (KDE)
## -------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (143 obs.);  Bandwidth 'bw' = 4.234
##
##        x                y
##  Min.   :605708   Min.   :6.845e-05
##  1st Qu.:605724   1st Qu.:5.646e-03
##  Median :605740   Median :1.531e-02
##  Mean   :605740   Mean   :1.563e-02
##  3rd Qu.:605756   3rd Qu.:2.251e-02
##  Max.   :605772   Max.   :3.999e-02
##
## -------------------------------------------------------------------------------
##  ::: X_UTM::Baja (KDE)
## -------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (153 obs.);  Bandwidth 'bw' = 3.012
##
##        x                y
##  Min.   :605698   Min.   :0.0000586
##  1st Qu.:605716   1st Qu.:0.0035397
##  Median :605733   Median :0.0080609
##  Mean   :605733   Mean   :0.0145803
##  3rd Qu.:605750   3rd Qu.:0.0230454
##  Max.   :605767   Max.   :0.0516069
##
## -------------------------------------------------------------------------------
##  ::: X_UTM::Media (KDE)
## -------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
```

```
## Data: x (138 obs.);  Bandwidth 'bw' = 3.566
##
##        x                y
##  Min.   :605699   Min.   :7.663e-05
##  1st Qu.:605713   1st Qu.:7.099e-03
##  Median :605727   Median :2.061e-02
##  Mean   :605727   Mean   :1.823e-02
##  3rd Qu.:605740   3rd Qu.:2.765e-02
##  Max.   :605754   Max.   :3.286e-02
##
## ------------------------------------------------------------------------------
##  ::: Y_UTM::Alta (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (143 obs.);  Bandwidth 'bw' = 8.552
##
##        x                 y
##  Min.   :2180968   Min.   :1.452e-05
##  1st Qu.:2181004   1st Qu.:1.747e-03
##  Median :2181040   Median :7.805e-03
##  Mean   :2181040   Mean   :6.930e-03
##  3rd Qu.:2181076   3rd Qu.:1.125e-02
##  Max.   :2181112   Max.   :1.466e-02
##
## ------------------------------------------------------------------------------
##  ::: Y_UTM::Baja (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (153 obs.);  Bandwidth 'bw' = 6.059
##
##        x                 y
##  Min.   :2180967   Min.   :2.968e-05
##  1st Qu.:2180994   1st Qu.:2.931e-03
##  Median :2181021   Median :7.321e-03
##  Mean   :2181021   Mean   :9.275e-03
##  3rd Qu.:2181048   3rd Qu.:1.296e-02
##  Max.   :2181075   Max.   :2.738e-02
##
## ------------------------------------------------------------------------------
##  ::: Y_UTM::Media (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (138 obs.);  Bandwidth 'bw' = 5.663
##
##        x                 y
```

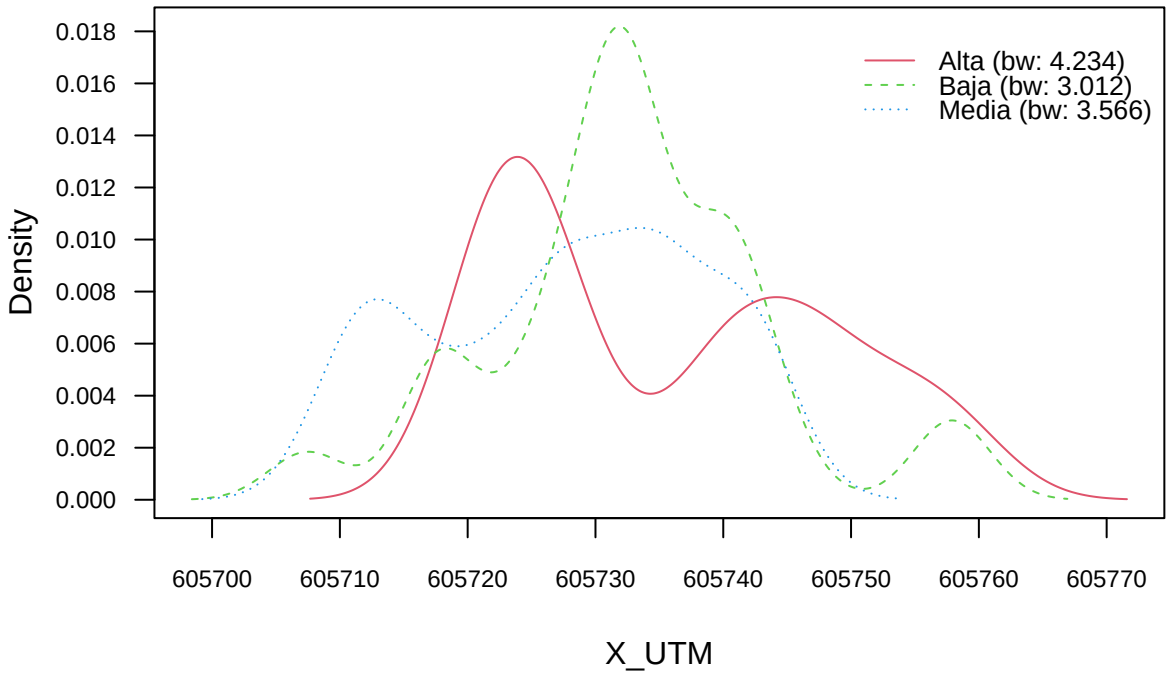
```
##  Min.   :2180983   Min.   :5.218e-05
##  1st Qu.:2181003   1st Qu.:4.901e-03
##  Median :2181024   Median :1.382e-02
##  Mean   :2181024   Mean   :1.195e-02
##  3rd Qu.:2181045   3rd Qu.:1.862e-02
##  Max.   :2181066   Max.   :2.012e-02
##
## ------------------------------------------------------------------------------
##  ::: Altitud::Alta (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (143 obs.);  Bandwidth 'bw' = 2.165
##
##        x               y
##  Min.   :2967   Min.   :8.123e-05
##  1st Qu.:2975   1st Qu.:7.206e-03
##  Median :2983   Median :1.904e-02
##  Mean   :2983   Mean   :3.037e-02
##  3rd Qu.:2991   3rd Qu.:5.454e-02
##  Max.   :3000   Max.   :8.754e-02
##
## ------------------------------------------------------------------------------
##  ::: Altitud::Baja (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (153 obs.);  Bandwidth 'bw' = 2.301
##
##        x               y
##  Min.   :2967   Min.   :0.0001377
##  1st Qu.:2976   1st Qu.:0.0068304
##  Median :2984   Median :0.0229024
##  Mean   :2984   Mean   :0.0295464
##  3rd Qu.:2993   3rd Qu.:0.0524126
##  Max.   :3001   Max.   :0.0725437
##
## ------------------------------------------------------------------------------
##  ::: Altitud::Media (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (138 obs.);  Bandwidth 'bw' = 2.332
##
##        x               y
##  Min.   :2966   Min.   :0.0001916
##  1st Qu.:2975   1st Qu.:0.0078593
##  Median :2983   Median :0.0223586
```

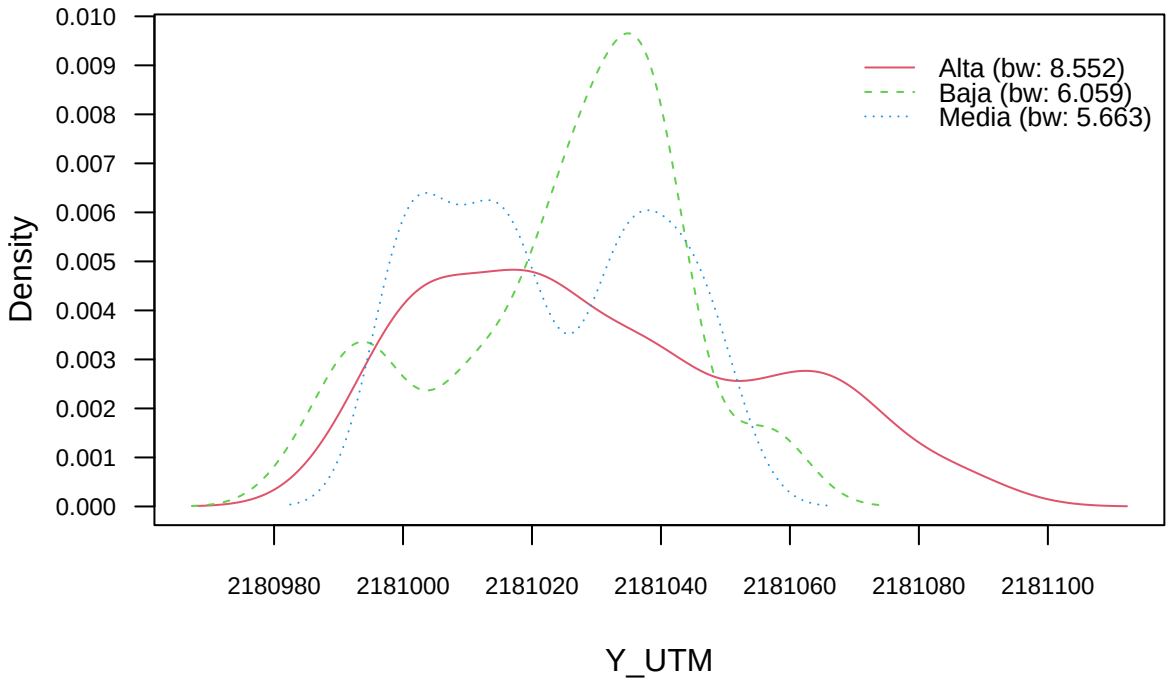
```
##  Mean   :2983   Mean   :0.0298194
##  3rd Qu.:2992   3rd Qu.:0.0515802
##  Max.   :3000   Max.   :0.0763652
##
## ------------------------------------------------------------------------------
##  ::: Inc::Alta (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (143 obs.);  Bandwidth 'bw' = 0.1516
##
##        x                  y
##  Min.   :-0.45489   Min.   :0.007724
##  1st Qu.: 0.02255   1st Qu.:0.184333
##  Median : 0.50000   Median :0.430753
##  Mean   : 0.50000   Mean   :0.522644
##  3rd Qu.: 0.97745   3rd Qu.:0.877189
##  Max.   : 1.45489   Max.   :1.250835
##
## ------------------------------------------------------------------------------
##  ::: Inc::Baja (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (153 obs.);  Bandwidth 'bw' = 0.147
##
##        x                  y
##  Min.   :-0.44114   Min.   :0.008662
##  1st Qu.: 0.02943   1st Qu.:0.234846
##  Median : 0.50000   Median :0.439232
##  Mean   : 0.50000   Mean   :0.530306
##  3rd Qu.: 0.97057   3rd Qu.:0.877922
##  Max.   : 1.44114   Max.   :1.208382
##
## ------------------------------------------------------------------------------
##  ::: Inc::Media (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (138 obs.);  Bandwidth 'bw' = 0.1492
##
##        x                  y
##  Min.   :-0.44755   Min.   :0.00673
##  1st Qu.: 0.02622   1st Qu.:0.17381
##  Median : 0.50000   Median :0.42338
##  Mean   : 0.50000   Mean   :0.52670
##  3rd Qu.: 0.97378   3rd Qu.:0.84646
##  Max.   : 1.44755   Max.   :1.41405
```

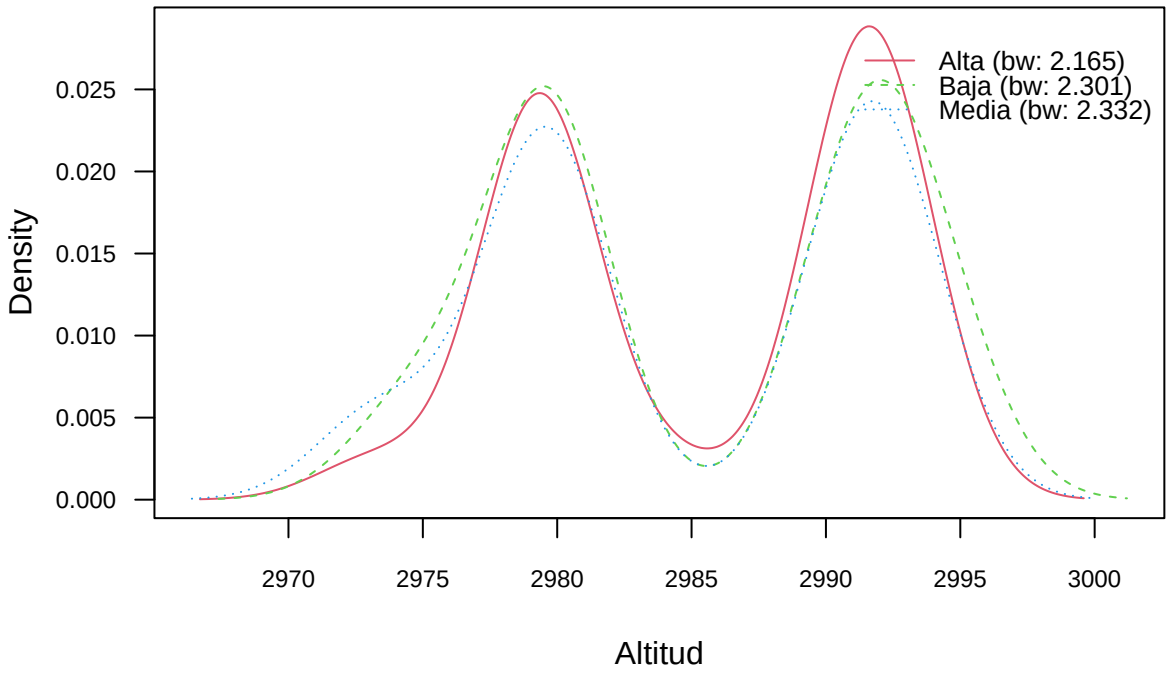
```
##
## ------------------------------------------------------------------------------
##  ::: Sevmed::Alta (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (143 obs.);  Bandwidth 'bw' = 0.05544
##
##        x                  y
##  Min.   :-0.16632   Min.   :0.00084
##  1st Qu.: 0.07067   1st Qu.:0.21204
##  Median : 0.30767   Median :1.00329
##  Mean   : 0.30767   Mean   :1.05316
##  3rd Qu.: 0.54466   3rd Qu.:1.22723
##  Max.   : 0.78166   Max.   :3.88998
##
## ------------------------------------------------------------------------------
##  ::: Sevmed::Baja (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (153 obs.);  Bandwidth 'bw' = 0.05547
##
##        x                  y
##  Min.   :-0.16640   Min.   :0.001247
##  1st Qu.: 0.06522   1st Qu.:0.321627
##  Median : 0.29683   Median :0.890326
##  Mean   : 0.29683   Mean   :1.077629
##  3rd Qu.: 0.52845   3rd Qu.:1.361567
##  Max.   : 0.76007   Max.   :3.893333
##
## ------------------------------------------------------------------------------
##  ::: Sevmed::Media (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (138 obs.);  Bandwidth 'bw' = 0.0524
##
##        x                  y
##  Min.   :-0.15720   Min.   :0.00182
##  1st Qu.: 0.04698   1st Qu.:0.42085
##  Median : 0.25117   Median :0.97273
##  Mean   : 0.25117   Mean   :1.22230
##  3rd Qu.: 0.45535   3rd Qu.:1.23741
##  Max.   : 0.65953   Max.   :4.53500
##
## ------------------------------------------------------------------------------
##
```

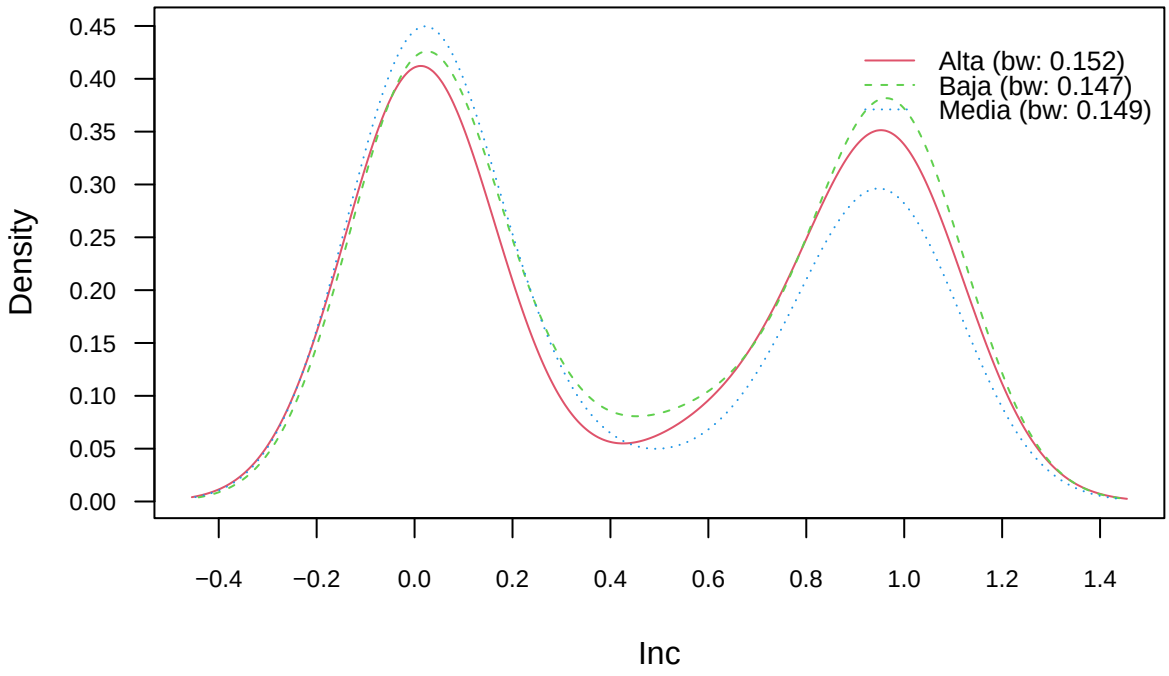
```
## # ... and 5 more tables
##
## ----------------------------------------------------------------------------
```

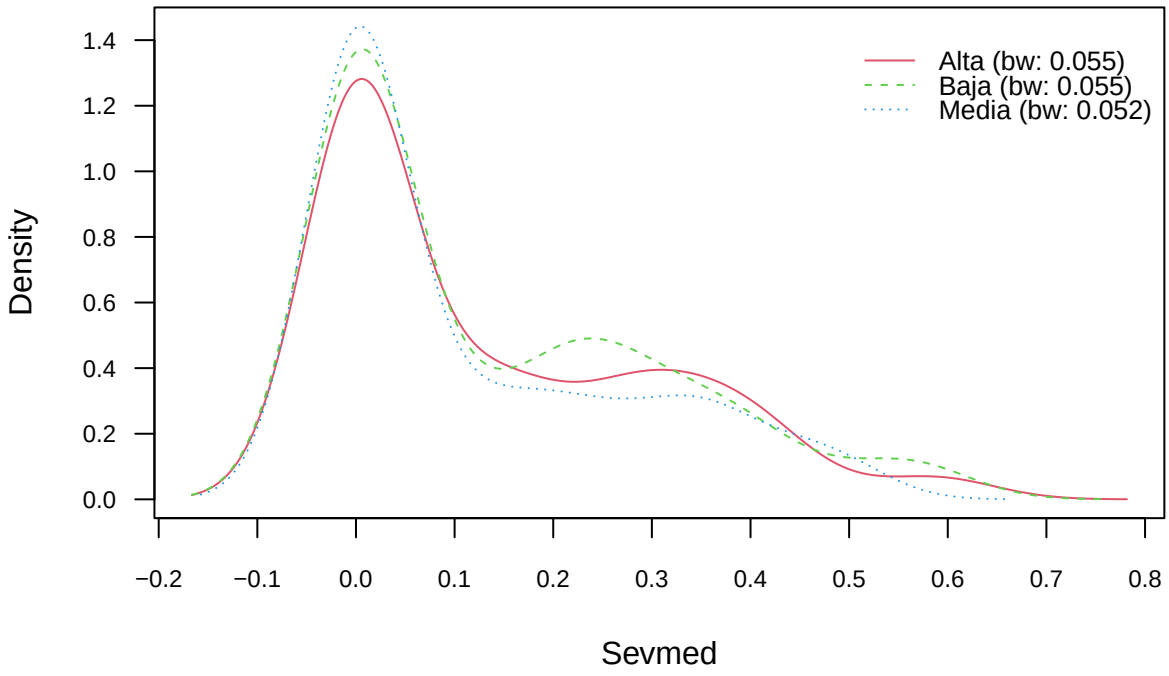
```
plot(nb)
```

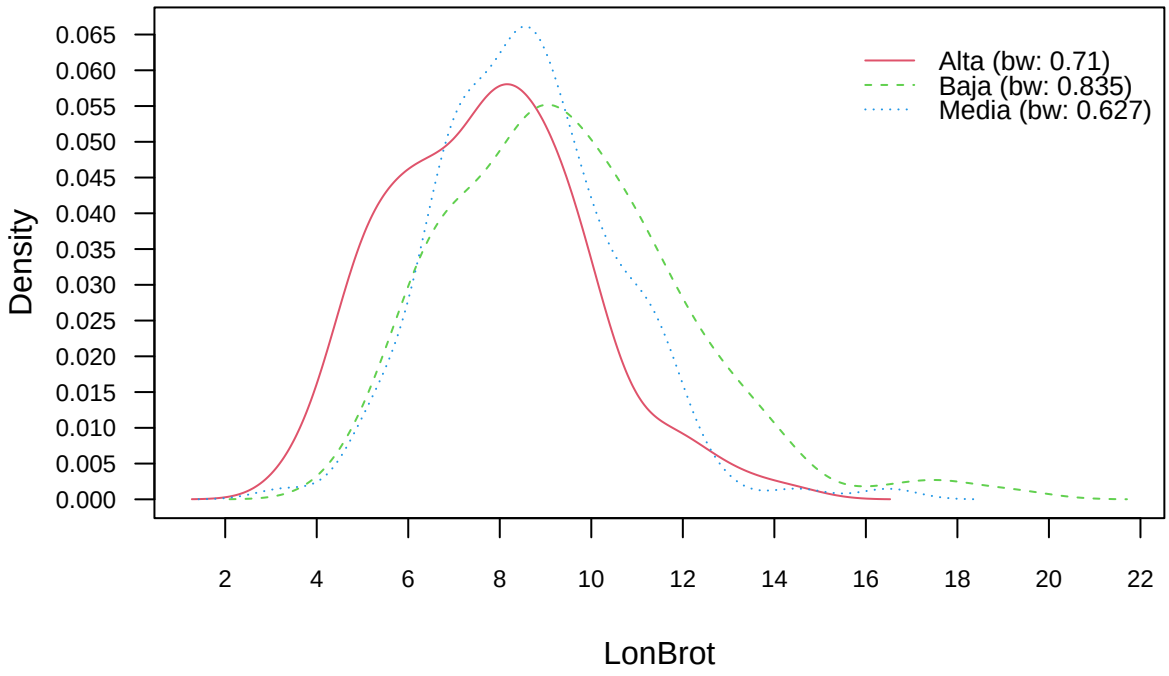


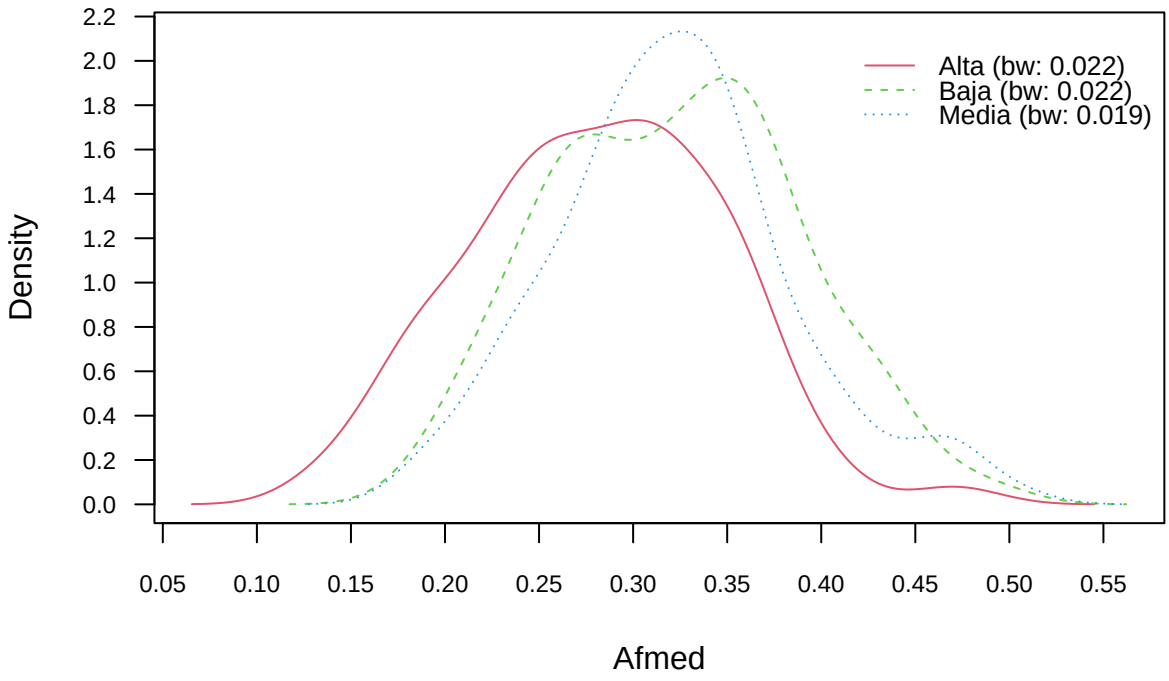

0.010
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
Alta (bw: 8.552)
Baja (bw: 6.059)
Media (bw: 5.663)
2180980
2181000
2181020
2181040
2181060
2181080
2181100
Y_UTM

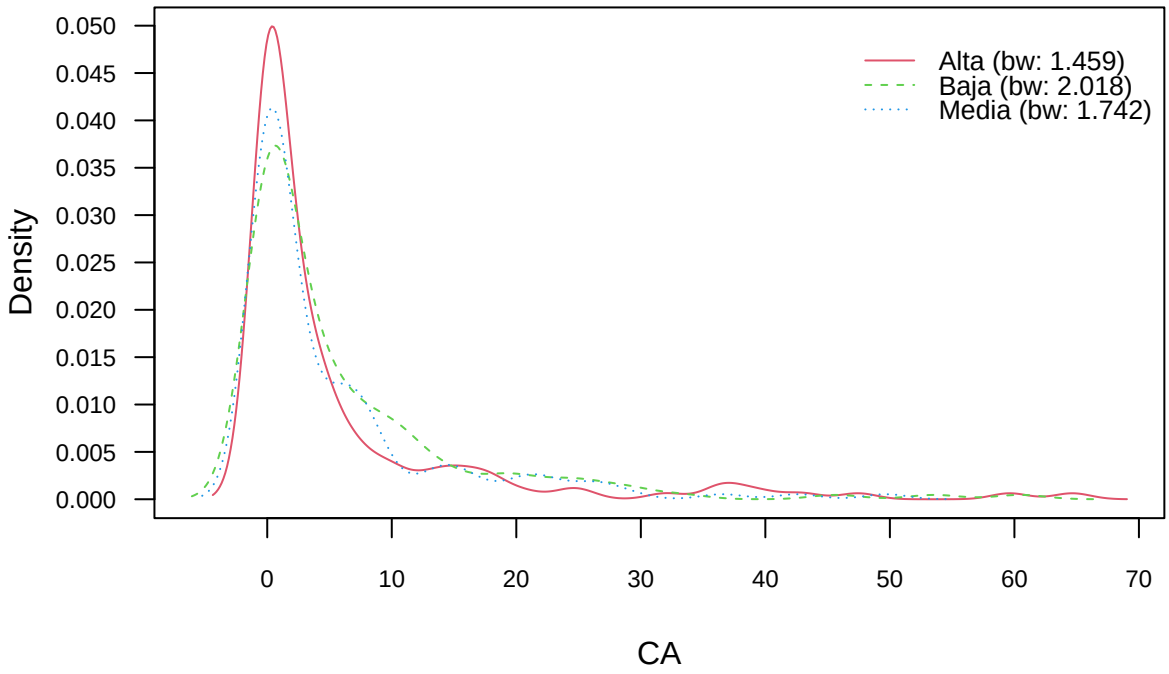

Alta (bw: 2.165)
Baja (bw: 2.301)
Media (bw: 2.332)
Density
0.025
0.020
0.015
0.010
0.005
0.000
2970
2975
2980
2985
2990
2995
3000
Altitud

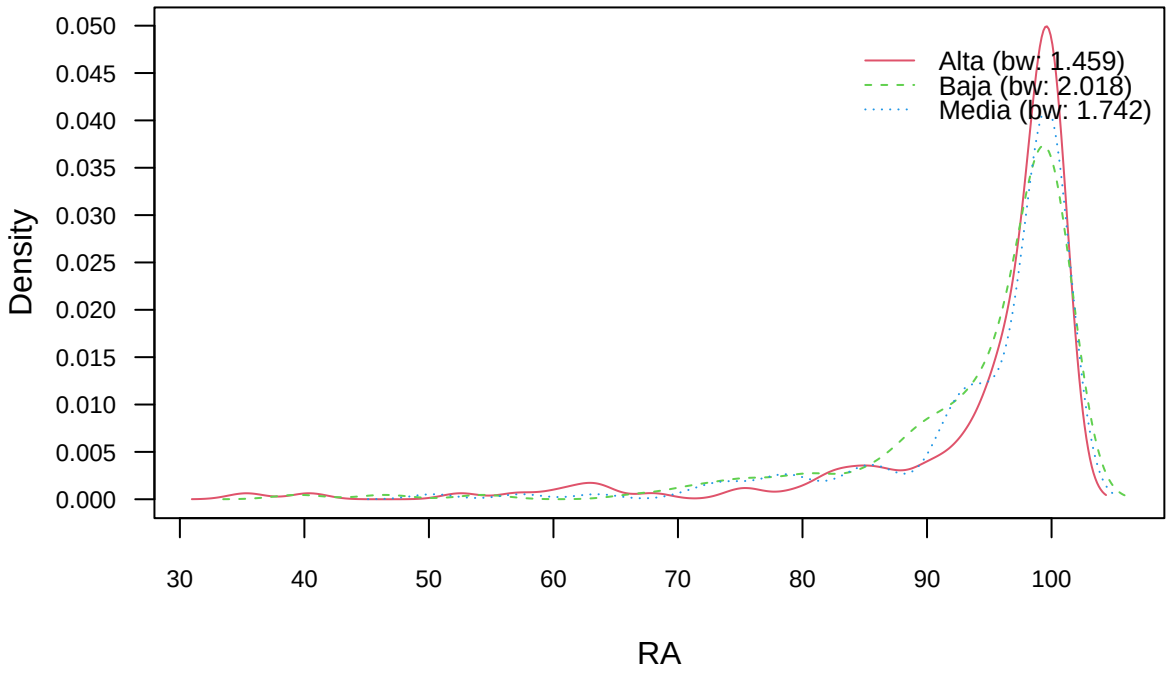

Alta (bw: 0.152)
Baja (bw: 0.147)
Media (bw: 0.149)
Density
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Inc

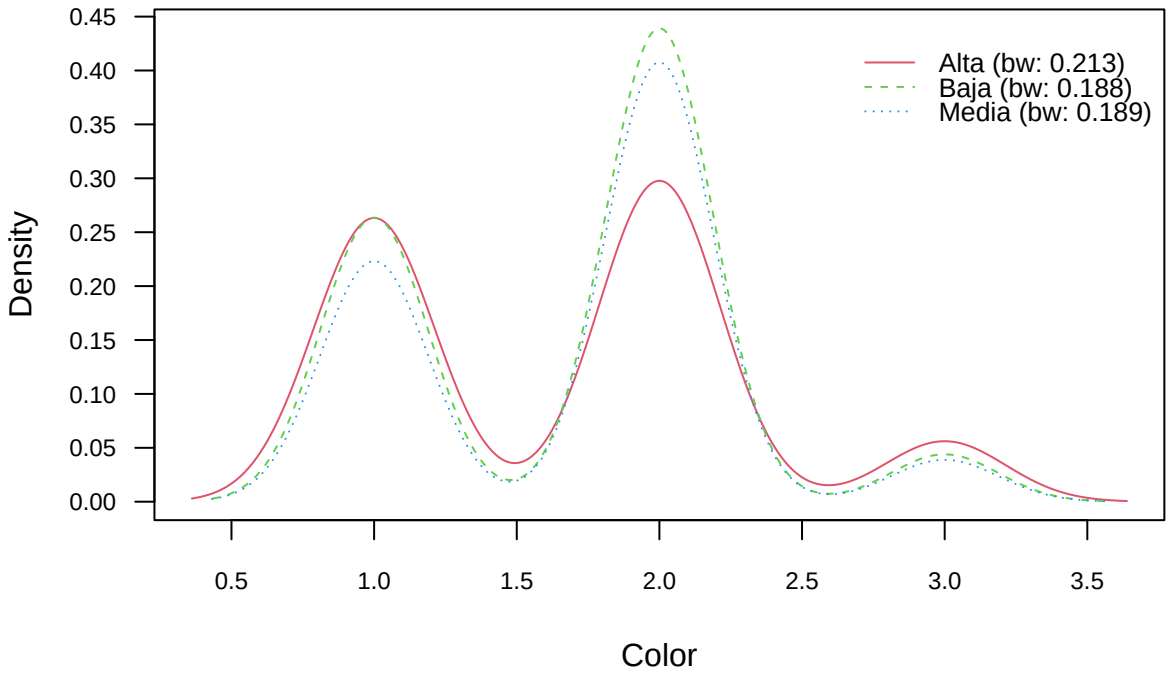

Alta (bw: 0.055)
Baja (bw: 0.055)
Media (bw: 0.052)
Density
Sevmed
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


Alta (bw: 0.71)
Baja (bw: 0.835)
Media (bw: 0.627)
Density
0.065
0.060
0.055
0.050
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
2
4
6
8
10
12
14
16
18
20
22
LonBrot


Density
Afmed
Alta (bw: 0.022)
Baja (bw: 0.022)
Media (bw: 0.019)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55


Alta (bw: 1.459)
Baja (bw: 2.018)
Media (bw: 1.742)
Density
0.050
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50
60
70
CA


Alta (bw: 1.459)
Baja (bw: 2.018)
Media (bw: 1.742)
Density
0.050
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
30
40
50
60
70
80
90
100
RA




```
#Prediccion
p <- predict(nb, train_nb, type = 'prob')
head(cbind(p, train_nb))
```

```
##        Alta       Baja       Media Bloque    X_UTM   Y_UTM Altitud       Inc
## 1 0.9787447 0.01850964 0.002745704   Alta 605747.9 2181070  2979.3 0.9000000
## 2 0.4113505 0.53406359 0.054585864   Alta 605744.3 2181057  2979.3 1.0000000
## 3 0.6491587 0.25629479 0.094546554   Alta 605740.2 2181039  2979.5 0.7666667
## 4 0.8842636 0.11442525 0.001311104   Alta 605754.5 2181034  2980.0 1.0000000
## 6 0.8065772 0.14937314 0.044049685   Alta 605727.7 2181021  2992.3 1.0000000
## 7 0.7160262 0.07648249 0.207491317   Alta 605724.3 2181003  2990.8 1.0000000
##      Sevmed LonBrot     Afmed        CA       RA Color
## 1 0.1503333    7.25 0.2393667 25.252525 74.74747     1
## 2 0.3310000   12.90 0.3744867  1.526718 98.47328     1
## 3 0.3466667    5.10 0.1960667  2.816901 97.18310     1
## 4 0.3510000    3.90 0.3699000 15.384615 84.61538     1
## 6 0.2486667    5.00 0.2253667 17.441860 82.55814     1
## 7 0.4033333    6.80 0.2549333  4.225352 95.77465     1
```

```
#Confusion Matrix - train data
p1 <- predict(nb, train_nb)
(tab1 <- table(p1, train_nb$Bloque))
```

```
##
## p1      Alta Baja Media
```

```
##   Alta    91   19    20
##   Baja    19   97    26
##   Media   33   37    92
```

```
1 - sum(diag(tab1)) / sum(tab1) #error del 35%
```

```
## [1] 0.3548387
```

```
#Confusion Matrix - test data
p2 <- predict(nb, test_nb)
(tab2 <- table(p2, test_nb$Bloque))
```

```
##
## p2      Alta Baja Media
##   Alta    28   14    13
##   Baja    12   30    22
##   Media   17   19    35
```

```
p2
```

```
##   [1] Baja  Baja  Alta  Baja  Media Media Baja  Media Alta  Alta  Baja  Alta
##  [13] Media Alta  Media Baja  Media Media Media Media Baja  Media Baja  Baja
##  [25] Baja  Baja  Alta  Media Baja  Alta  Alta  Baja  Alta  Alta  Media Baja
##  [37] Baja  Baja  Baja  Baja  Media Alta  Media Alta  Media Alta  Media Media
##  [49] Media Media Media Alta  Media Alta  Baja  Baja  Media Alta  Baja  Media
##  [61] Alta  Baja  Media Alta  Media Media Baja  Media Media Alta  Alta  Baja
##  [73] Baja  Alta  Media Baja  Media Baja  Alta  Alta  Media Media Media Alta
##  [85] Media Media Baja  Media Alta  Baja  Baja  Media Media Alta  Baja  Baja
##  [97] Baja  Media Media Media Alta  Alta  Baja  Baja  Media Alta  Media Alta
## [109] Baja  Media Alta  Alta  Alta  Media Media Alta  Media Alta  Alta  Baja
## [121] Media Baja  Media Media Alta  Alta  Media Alta  Alta  Media Baja  Baja
## [133] Baja  Baja  Baja  Media Media Baja  Media Baja  Baja  Alta  Media Media
## [145] Alta  Alta  Baja  Media Media Media Alta  Alta  Baja  Alta  Baja  Media
## [157] Alta  Alta  Media Alta  Alta  Alta  Media Baja  Baja  Baja  Baja  Media
## [169] Baja  Media Baja  Baja  Baja  Baja  Baja  Baja  Alta  Baja  Alta  Media
## [181] Alta  Alta  Media Media Media Baja  Baja  Media Baja  Media
## Levels: Alta Baja Media
```

```
1 - sum(diag(tab2)) / sum(tab2) #error del 51%
```

```
## [1] 0.5105263
```

```
library(caret)
confusionMatrix(p2,test_nb$Bloque)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Alta Baja Media
##      Alta    28   14    13
```

```
##      Baja    12   30    22
##      Media   17   19    35
##
## Overall Statistics
##
##                Accuracy : 0.4895
##                  95% CI : (0.4164, 0.5629)
##     No Information Rate : 0.3684
##     P-Value [Acc > NIR] : 0.0004302
##
##                   Kappa : 0.2309
##
##  Mcnemar's Test P-Value : 0.8238127
##
## Statistics by Class:
##
##                      Class: Alta Class: Baja Class: Media
## Sensitivity               0.4912      0.4762       0.5000
## Specificity               0.7970      0.7323       0.7000
## Pos Pred Value            0.5091      0.4688       0.4930
## Neg Pred Value            0.7852      0.7381       0.7059
## Prevalence                0.3000      0.3316       0.3684
## Detection Rate            0.1474      0.1579       0.1842
## Detection Prevalence      0.2895      0.3368       0.3737
## Balanced Accuracy         0.6441      0.6042       0.6000
```

Interpretación: se obtuvo una precisión de 49% y un valor de κ=0.23. En general esta técnica no fue tan buena para clasificar la transparencia de copa.

4) K-NN

```
#https://rpubs.com/JairoAyala/601703

library(kknn)

base_knn <- na.omit(base[,c(2,5,6,7,9,10,12,18,19,22,24,28:30,32)])

base_knn$Bloque <- as.factor(base_knn$Bloque)
table(base_knn$Bloque)
```

```
##
##  Alta  Baja Media
##   200   216   208
```

```
set.seed(2020)
muestra <- sample(1:624, 437)
train_knn <- base_knn[muestra,]#70%
test_knn<- base_knn[-muestra,]#30%
dim(train_knn)[1]
```

```
## [1] 437
```

```
dim(test_knn)[1]
```

```
## [1] 187
```

```
knn <- train.kknn(Bloque~ ., data = train_knn, kmax = 9)
knn
```

```
##
## Call:
## train.kknn(formula = Bloque ~ ., data = train_knn, kmax = 9)
##
## Type of response variable: nominal
## Minimal misclassification: 0.5102975
## Best kernel: optimal
## Best k: 9
```

```
entre <- predict(knn, train_knn[,-1])
tt  <- table(train_knn[,1],entre)
tt
```

```
##        entre
##         Alta Baja Media
##   Alta   129    5     5
##   Baja    16  122    15
##   Media    7    7   131
```

```
precision <- (sum(diag(tt)))/sum(tt)
precision
```

```
## [1] 0.8741419
```

```
#precisión del 100 % en datos de entrenamiento


#Precisión test de prueba
pred    <- predict(knn, test_knn[,-1])
table   <- table(test_knn[,1],pred)
table
```

```
##        pred
##         Alta Baja Media
##   Alta    35   13    13
##   Baja    13   29    21
##   Media   12   16    35
```

```
clas    <- (sum(diag(table)))/sum(table)
clas
```

```
## [1] 0.5294118
```

```
#Precisión del 53% de datos de prueba

#matriz de confusion con la prueba

library(caret)
confusionMatrix(pred,test_knn$Bloque)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Alta Baja Media
##      Alta    35   13    12
##      Baja    13   29    16
##      Media   13   21    35
##
## Overall Statistics
##
##                Accuracy : 0.5294
##                  95% CI : (0.4552, 0.6026)
##     No Information Rate : 0.3369
##     P-Value [Acc > NIR] : 4.914e-08
##
##                   Kappa : 0.294
##
##  Mcnemar's Test P-Value : 0.8695
##
## Statistics by Class:
##
##                      Class: Alta Class: Baja Class: Media
## Sensitivity               0.5738      0.4603       0.5556
## Specificity               0.8016      0.7661       0.7258
## Pos Pred Value            0.5833      0.5000       0.5072
## Neg Pred Value            0.7953      0.7364       0.7627
## Prevalence                0.3262      0.3369       0.3369
## Detection Rate            0.1872      0.1551       0.1872
## Detection Prevalence      0.3209      0.3102       0.3690
## Balanced Accuracy         0.6877      0.6132       0.6407
```

Interpretación: se obtuvo una precisión de 53% y un valor de κ=0.30. En general esta técnica no fue tan buena para clasificar la transparencia de copa.

5) Curvas Receiver Operating Characteristic (ROC)

```
par(mfrow = c(1,3))

# Validation set assessment #2: ROC curves and AUC
# Needs to import ROCR package for ROC curve plotting:
library(ROCR)
# Calculate the probability of new observations belonging to each class
# prediction_for_roc_curve will be a matrix with dimensions data_set_size x number_of_classes
prediction_for_roc_curve <- predict(rf,test[,-1],type="prob")
```

```
# Use pretty colours:
pretty_colours <- c("#F8766D","#00BA38","#619CFF")
# Specify the different classes
test$Bloque <- as.factor(test$Bloque)
classes <- levels(test$Bloque)
# For each class
for (i in 1:3)
{
  # Define which observations belong to class[i]
  true_values <- ifelse(test[,1]==classes[i],1,0)
  # Assess the performance of classifier for class[i]
  pred <- prediction(prediction_for_roc_curve[,i],true_values)
  perf <- performance(pred, "tpr", "fpr")
  if (i==1)
  {
    plot(perf,main="ROC Curve-Random Forest",col=pretty_colours[i])
    abline (a = 0, b = 1, lty="dotted", lwd=2)
  }
  else
  {
    plot(perf,main="ROC Curve- Random Forest",col=pretty_colours[i],add=TRUE)
  }
  # Calculate the AUC and print it to screen
  auc.perf <- performance(pred, measure = "auc")
  print(auc.perf@y.values)
}
```

```
## [[1]]
## [1] 0.9993548
##
## [[1]]
## [1] 1
##
## [[1]]
## [1] 1
```

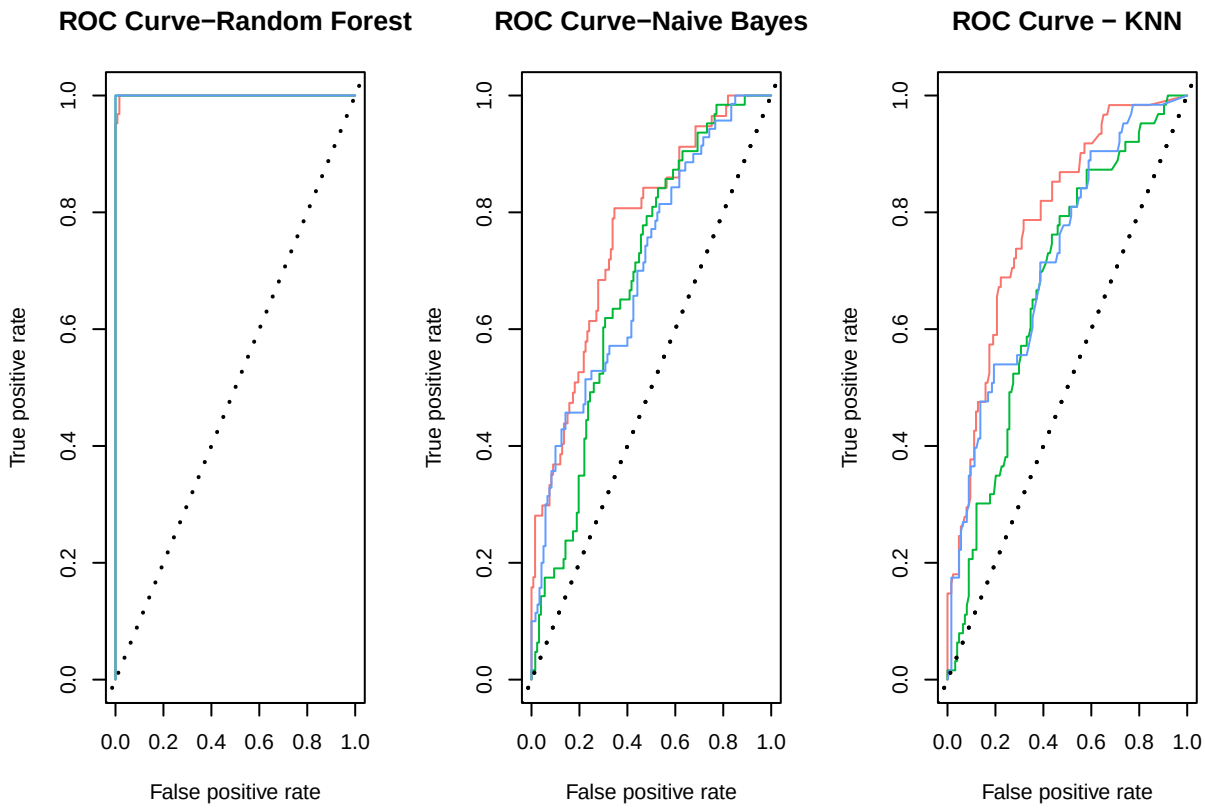
```
#Naive Bayes (ROC)

# Validation set assessment #2: ROC curves and AUC
# Needs to import ROCR package for ROC curve plotting:
library(ROCR)
# Calculate the probability of new observations belonging to each class
# prediction_for_roc_curve will be a matrix with dimensions data_set_size x number_of_classes
prediction_for_roc_curve <- predict(nb,test_nb[,-1],type="prob")
# Use pretty colours:
pretty_colours <- c("#F8766D","#00BA38","#619CFF")
# Specify the different classes
test_nb$Bloque <- as.factor(test_nb$Bloque)
classes <- levels(test_nb$Bloque)
# For each class
for (i in 1:3)
{
  # Define which observations belong to class[i]
```

```
  true_values <- ifelse(test_nb[,1]==classes[i],1,0)
  # Assess the performance of classifier for class[i]
  pred <- prediction(prediction_for_roc_curve[,i],true_values)
  perf <- performance(pred, "tpr", "fpr")
  if (i==1)
  {
    plot(perf,main="ROC Curve-Naive Bayes",col=pretty_colours[i])
  }
  else
  {
    plot(perf,main="ROC Curve-Naive Bayes",col=pretty_colours[i],add=TRUE)
    abline (a = 0, b = 1, lty="dotted", lwd=2)
  }
  # Calculate the AUC and print it to screen
  auc.perf <- performance(pred, measure = "auc")
  print(auc.perf@y.values)
}
```

```
## [[1]]
## [1] 0.7632238
##
## [[1]]
## [1] 0.6860392
##
## [[1]]
## [1] 0.702381
```

```
#KNN (ROC)

# Validation set assessment #2: ROC curves and AUC
# Needs to import ROCR package for ROC curve plotting:
library(ROCR)
# Calculate the probability of new observations belonging to each class
# prediction_for_roc_curve will be a matrix with dimensions data_set_size x number_of_classes
prediction_for_roc_curve <- predict(knn,test_knn[,-1],type="prob")
# Use pretty colours:
pretty_colours <- c("#F8766D","#00BA38","#619CFF")
# Specify the different classes
test_knn$Bloque <- as.factor(test_knn$Bloque)
classes <- levels(test_knn$Bloque)
# For each class
for (i in 1:3)
{
  # Define which observations belong to class[i]
  true_values <- ifelse(test_knn[,1]==classes[i],1,0)
  # Assess the performance of classifier for class[i]
  pred <- prediction(prediction_for_roc_curve[,i],true_values)
  perf <- performance(pred, "tpr", "fpr")
  if (i==1)
  {
    plot(perf,main="ROC Curve - KNN",col=pretty_colours[i])
  }
  else
```

```
  {
    plot(perf,main="ROC Curve -KNN",col=pretty_colours[i],add=TRUE)
    abline (a = 0, b = 1, lty="dotted", lwd=2)
  }
  # Calculate the AUC and print it to screen
  auc.perf <- performance(pred, measure = "auc")
  print(auc.perf@y.values)
}
```




```
## [[1]]
## [1] 0.7848686
##
## [[1]]
## [1] 0.6751792
##
## [[1]]
## [1] 0.7210061
```

```
dev.off()
```

```
## null device
##           1
```

```
#cuadro con valores de precision
p <-rbind(c(0.98,0.49,0.53))
k <- rbind(c(0.98,0.23,0.30))
r <- rbind(c(1.00, 0.70, 0.72))
pres <- rbind(p,k,r)
pres <- as.data.frame(pres)
colnames(pres) <- c("Random Forest","Naive Bayes", "K-NN")
pres <- cbind(Métrica=c("Precisión(%)","Kappa", "AUC-ROC"),pres)

library(gt)
pres %>% gt()
```

Métrica	Random Forest	Naive Bayes	K-NN
Precisión(%)	0.98	0.49	0.53
Kappa	0.98	0.23	0.30
AUC-ROC	1.00	0.70	0.72

Interpretación: De acuerdo la evaluación de las curvas ROC, es evidente qu el mejor modelo para clasificar a la transparencia de copa de Douglas-fir fue el Random Forest, ya que tuvo mayor cantidad de positivos verdaderos en comparación de Naive Bayes y K-NN, donde se pueden considera que son modelos de regulares a malos.