Modelos de Regresión Lineal

Javier Carpio & Paul Belches

25/3/2020

Naive Bayes

Como primer paso para realizar el método de Naive Bayes se procederá con realizar el proceso de clustering que nos permite agrupar las casas en Bajo, Intermedio y Alto, y así obtener la variable respuesta:

```
datos<-read.csv("train.csv")

#data <-select(datos, GarageYrBlt, GrLivArea, X1stFlrSF, X2ndFlrSF,
GarageCars, SalePrice)
#data <- na.omit(data)
data <-select(datos, LotFrontage, LotArea, YearBuilt, YearRemodAdd,
MasVnrArea,
BsmtFinSF1,BsmtFinSF2,BsmtUnfSF,TotalBsmtSF,X1stFlrSF,X2ndFlrSF,LowQualFinSF,
GrLivArea,TotRmsAbvGrd,Fireplaces,GarageYrBlt,GarageCars,GarageArea,WoodDeckS
F,OpenPorchSF,EnclosedPorch,ScreenPorch,PoolArea,MoSold,YrSold,SalePrice)
#Data cleanup
data <- na.omit(data)
```

Además se realizar una matriz de correlación, para ver el peso de las variables en el modelo.

```
matriz_cor <- cor(data)
matriz_cor

##               LotFrontage      LotArea    YearBuilt YearRemodAdd
MasVnrArea
## LotFrontage    1.00000000  0.421184102  0.109725571   0.08641397
0.18996859
## LotArea        0.42118410  1.000000000  0.029205413   0.02684785
0.10611543
## YearBuilt      0.10972557  0.029205413  1.000000000   0.62317127
0.33218984
## YearRemodAdd   0.08641397  0.026847846  0.623171270   1.00000000
0.19337560
## MasVnrArea     0.18996859  0.106115431  0.332189842   0.19337560
1.00000000
## BsmtFinSF1     0.24135223  0.230441380  0.236940941   0.12077442
0.28533133
## BsmtFinSF2     0.04930532  0.138233605 -0.054413993  -0.05702407 -
0.07526068
## BsmtUnfSF      0.11530588  0.011288124  0.177545400   0.19989263
```

```
0.11006742
## TotalBsmtSF    0.38761951  0.302553906  0.409133562   0.30869623
0.38443408
## X1stFlrSF      0.45108503  0.329678689  0.308874836   0.28143596
0.36320926
## X2ndFlrSF      0.07500380  0.074611842 -0.011621305   0.10362739
0.18056732
## LowQualFinSF   0.01114809  0.020039426 -0.164358630  -0.05347869 -
0.06293045
## GrLivArea      0.39630602  0.307163514  0.204967302   0.29004951
0.41402420
## TotRmsAbvGrd   0.34842111  0.237917977  0.121416862   0.18199519
0.31560393
## Fireplaces     0.26032083  0.255754683  0.133076661   0.12589807
0.25252540
## GarageYrBlt    0.06987812  0.013730760  0.823519546   0.64580847
0.27709541
## GarageCars     0.28658681  0.172428230  0.532562838   0.46266302
0.37526882
## GarageArea     0.35685094  0.211362399  0.471285901   0.40747074
0.38216230
## WoodDeckSF     0.08216563  0.133576037  0.238548109   0.24460217
0.17464860
## OpenPorchSF    0.16181512  0.099170000  0.235432138   0.26052120
0.12953180
## EnclosedPorch  0.01426101 -0.023630663 -0.392693146  -0.21411482 -
0.11683237
## ScreenPorch    0.03590598  0.072517046 -0.063694409  -0.03428804
0.05264566
## PoolArea       0.21174612  0.109147070  0.006716815   0.01930744
0.02164782
## MoSold         0.01881453  0.008998481  0.013784446   0.02688387
0.01585016
## YrSold         0.01326707 -0.006903891 -0.004585485   0.04130151 -
0.01756923
## SalePrice      0.34426977  0.299962206  0.525393598   0.52125327
0.48865815
##                BsmtFinSF1  BsmtFinSF2     BsmtUnfSF  TotalBsmtSF
X1stFlrSF
## LotFrontage    0.24135223  0.04930532  1.153059e-01  0.387619513
0.4510850287
## LotArea        0.23044138  0.13823360  1.128812e-02  0.302553906
0.3296786887
## YearBuilt      0.23694094 -0.05441399  1.775454e-01  0.409133562
0.3088748362
## YearRemodAdd   0.12077442 -0.05702407  1.998926e-01  0.308696227
0.2814359592
## MasVnrArea     0.28533133 -0.07526068  1.100674e-01  0.384434076
0.3632092600
## BsmtFinSF1     1.00000000 -0.03577983 -5.022248e-01  0.530916507
```

```
0.4680197587
## BsmtFinSF2    -0.03577983  1.00000000 -2.201905e-01  0.094079397
0.0730896330
## BsmtUnfSF     -0.50222479 -0.22019049  1.000000e+00  0.404510415
0.3149725896
## TotalBsmtSF    0.53091651  0.09407940  4.045104e-01  1.000000000
0.8359993534
## X1stFlrSF      0.46801976  0.07308963  3.149726e-01  0.835999353
1.0000000000
## X2ndFlrSF     -0.12082282 -0.11185026 -1.002185e-02 -0.176721795 -
0.2089292412
## LowQualFinSF  -0.05082356  0.01545875  3.899073e-05 -0.047901479 -
0.0130255395
## GrLivArea      0.23988762 -0.03854111  2.238598e-01  0.464644664
0.5613722585
## TotRmsAbvGrd   0.08020688 -0.05490018  2.165844e-01  0.283676127
0.4053140299
## Fireplaces     0.27030558  0.02234751  5.515445e-02  0.347729684
0.4101442139
## GarageYrBlt    0.16035595 -0.07547715  2.089150e-01  0.352876850
0.2790531180
## GarageCars     0.19644275 -0.07547708  2.770639e-01  0.459656896
0.4687573955
## GarageArea     0.28665692 -0.04795896  2.353287e-01  0.522051222
0.5211829925
## WoodDeckSF     0.20624572  0.03233756  5.391473e-03  0.233663743
0.2376282834
## OpenPorchSF    0.12790025  0.01051764  1.515723e-01  0.291285868
0.2448455634
## EnclosedPorch -0.10541028  0.04722069 -3.579056e-02 -0.130223306 -
0.1135952632
## ScreenPorch    0.05963521  0.06789878 -6.398081e-03  0.080258724
0.0875796921
## PoolArea       0.19434944  0.06121181 -5.389385e-02  0.171488860
0.1517613301
## MoSold        -0.01528148 -0.03610120  2.706835e-02 -0.001498092
0.0277310418
## YrSold         0.01022418  0.03639527 -2.673628e-02 -0.003377490
0.0004204947
## SalePrice      0.39030052 -0.02802137  2.131287e-01  0.615612237
0.6079691062
##                 X2ndFlrSF  LowQualFinSF   GrLivArea TotRmsAbvGrd
Fireplaces
## LotFrontage    0.07500380  1.114809e-02  0.39630602   0.34842111
0.26032083
## LotArea        0.07461184  2.003943e-02  0.30716351   0.23791798
0.25575468
## YearBuilt     -0.01162130 -1.643586e-01  0.20496730   0.12141686
0.13307666
## YearRemodAdd   0.10362739 -5.347869e-02  0.29004951   0.18199519
```

```
0.12589807
## MasVnrArea     0.18056732 -6.293045e-02  0.41402420   0.31560393
0.25252540
## BsmtFinSF1    -0.12082282 -5.082356e-02  0.23988762   0.08020688
0.27030558
## BsmtFinSF2    -0.11185026  1.545875e-02 -0.03854111  -0.05490018
0.02234751
## BsmtUnfSF     -0.01002185  3.899073e-05  0.22385980   0.21658444
0.05515445
## TotalBsmtSF   -0.17672179 -4.790148e-02  0.46464466   0.28367613
0.34772968
## X1stFlrSF     -0.20892924 -1.302554e-02  0.56137226   0.40531403
0.41014421
## X2ndFlrSF      1.00000000  6.241162e-02  0.68829155   0.61777593
0.19934310
## LowQualFinSF   0.06241162  1.000000e+00  0.12208092   0.10234764
0.02149048
## GrLivArea      0.68829155  1.220809e-01  1.00000000   0.82431212
0.47105987
## TotRmsAbvGrd   0.61777593  1.023476e-01  0.82431212   1.00000000
0.35204779
## Fireplaces     0.19934310  2.149048e-02  0.47105987   0.35204779
1.00000000
## GarageYrBlt    0.04973733 -4.632193e-02  0.24373384   0.16720675
0.06457898
## GarageCars     0.18013604 -2.338131e-02  0.49463136   0.42396283
0.25268507
## GarageArea     0.12275686  5.708720e-03  0.48754960   0.38192956
0.21655099
## WoodDeckSF     0.11447978 -1.737401e-02  0.26970261   0.19052652
0.17776256
## OpenPorchSF    0.20346028  3.296761e-02  0.35353411   0.24671363
0.18527413
## EnclosedPorch  0.07647940  6.098785e-02 -0.01487387  -0.03165122 -
0.03447839
## ScreenPorch    0.04703892  5.647180e-02  0.10845307   0.07089430
0.19212865
## PoolArea       0.09407574  9.908857e-02  0.19855114   0.09338651
0.11710776
## MoSold         0.04148506 -2.664535e-02  0.05307080   0.04309712
0.04878812
## YrSold        -0.02800994 -1.625314e-02 -0.02443609  -0.02481218 -
0.03140227
## SalePrice      0.30687900 -1.481983e-03  0.70515357   0.54706736
0.46187269
##                GarageYrBlt  GarageCars  GarageArea   WoodDeckSF
OpenPorchSF
## LotFrontage    0.069878118  0.28658681  0.35685094  0.082165627
0.16181512
## LotArea        0.013730760  0.17242823  0.21136240  0.133576037
```

```
0.09917000
## YearBuilt      0.823519546  0.53256284  0.47128590  0.238548109
0.23543214
## YearRemodAdd   0.645808468  0.46266302  0.40747074  0.244602168
0.26052120
## MasVnrArea     0.277095408  0.37526882  0.38216230  0.174648597
0.12953180
## BsmtFinSF1     0.160355947  0.19644275  0.28665692  0.206245716
0.12790025
## BsmtFinSF2    -0.075477153 -0.07547708 -0.04795896  0.032337560
0.01051764
## BsmtUnfSF      0.208915005  0.27706388  0.23532868  0.005391473
0.15157230
## TotalBsmtSF    0.352876850  0.45965690  0.52205122  0.233663743
0.29128587
## X1stFlrSF      0.279053118  0.46875740  0.52118299  0.237628283
0.24484556
## X2ndFlrSF      0.049737325  0.18013604  0.12275686  0.114479784
0.20346028
## LowQualFinSF  -0.046321925 -0.02338131  0.00570872 -0.017374011
0.03296761
## GrLivArea      0.243733841  0.49463136  0.48754960  0.269702612
0.35353411
## TotRmsAbvGrd   0.167206751  0.42396283  0.38192956  0.190526524
0.24671363
## Fireplaces     0.064578978  0.25268507  0.21655099  0.177762561
0.18527413
## GarageYrBlt    1.000000000  0.60090342  0.59263525  0.255915956
0.25714101
## GarageCars     0.600903418  1.00000000  0.83941492  0.234276205
0.25813718
## GarageArea     0.592635246  0.83941492  1.00000000  0.223954993
0.30255823
## WoodDeckSF     0.255915956  0.23427620  0.22395499  1.000000000
0.07552504
## OpenPorchSF    0.257141006  0.25813718  0.30255823  0.075525042
1.00000000
## EnclosedPorch -0.308277701 -0.15188609 -0.11574897 -0.121060644 -
0.13056551
## ScreenPorch   -0.067595752  0.02513541  0.02644616 -0.087574843
0.11244284
## PoolArea      -0.009295071  0.01282888  0.08087138  0.033075524
0.03378559
## MoSold         0.009232878  0.05748115  0.03759656  0.041547155
0.08976692
## YrSold         0.009596052 -0.03350744 -0.01620605  0.014809970 -
0.05303516
## SalePrice      0.504753018  0.64703361  0.61932962  0.336855121
0.34335381
##               EnclosedPorch  ScreenPorch     PoolArea       MoSold
```

```
## LotFrontage     0.014261014  0.035905984  0.211746117  0.018814535
## LotArea        -0.023630663  0.072517046  0.109147070  0.008998481
## YearBuilt      -0.392693146 -0.063694409  0.006716815  0.013784446
## YearRemodAdd   -0.214114825 -0.034288042  0.019307439  0.026883872
## MasVnrArea     -0.116832373  0.052645658  0.021647815  0.015850157
## BsmtFinSF1     -0.105410284  0.059635214  0.194349437 -0.015281479
## BsmtFinSF2      0.047220690  0.067898778  0.061211811 -0.036101200
## BsmtUnfSF      -0.035790558 -0.006398081 -0.053893848  0.027068355
## TotalBsmtSF    -0.130223306  0.080258724  0.171488860 -0.001498092
## X1stFlrSF      -0.113595263  0.087579692  0.151761330  0.027731042
## X2ndFlrSF       0.076479404  0.047038918  0.094075738  0.041485056
## LowQualFinSF    0.060987845  0.056471796  0.099088571 -0.026645350
## GrLivArea      -0.014873875  0.108453071  0.198551141  0.053070805
## TotRmsAbvGrd   -0.031651218  0.070894298  0.093386510  0.043097116
## Fireplaces     -0.034478393  0.192128653  0.117107761  0.048788120
## GarageYrBlt    -0.308277701 -0.067595752 -0.009295071  0.009232878
## GarageCars     -0.151886088  0.025135406  0.012828877  0.057481155
## GarageArea     -0.115748972  0.026446162  0.080871376  0.037596558
## WoodDeckSF     -0.121060644 -0.087574843  0.033075524  0.041547155
## OpenPorchSF    -0.130565513  0.112442842  0.033785590  0.089766922
## EnclosedPorch   1.000000000 -0.081550145  0.076341565 -0.061083343
## ScreenPorch    -0.081550145  1.000000000  0.067356042  0.012859261
## PoolArea        0.076341565  0.067356042  1.000000000 -0.054872361
## MoSold         -0.061083343  0.012859261 -0.054872361  1.000000000
## YrSold         -0.001184577 -0.004118063 -0.053887689 -0.150576612
## SalePrice      -0.154843204  0.110426815  0.092488120  0.051568064
##                      YrSold    SalePrice
## LotFrontage    0.0132670710  0.344269772
## LotArea       -0.0069038907  0.299962206
## YearBuilt     -0.0045854854  0.525393598
## YearRemodAdd   0.0413015131  0.521253270
## MasVnrArea    -0.0175692330  0.488658155
## BsmtFinSF1     0.0102241754  0.390300523
## BsmtFinSF2     0.0363952692 -0.028021366
## BsmtUnfSF     -0.0267362844  0.213128680
## TotalBsmtSF   -0.0033774903  0.615612237
## X1stFlrSF      0.0004204947  0.607969106
## X2ndFlrSF     -0.0280099423  0.306879002
## LowQualFinSF  -0.0162531362 -0.001481983
## GrLivArea     -0.0244360874  0.705153567
## TotRmsAbvGrd  -0.0248121829  0.547067360
## Fireplaces    -0.0314022732  0.461872689
## GarageYrBlt    0.0095960523  0.504753018
## GarageCars    -0.0335074412  0.647033611
## GarageArea    -0.0162060545  0.619329622
## WoodDeckSF     0.0148099704  0.336855121
## OpenPorchSF   -0.0530351628  0.343353812
## EnclosedPorch -0.0011845771 -0.154843204
## ScreenPorch   -0.0041180627  0.110426815
## PoolArea      -0.0538876893  0.092488120
```

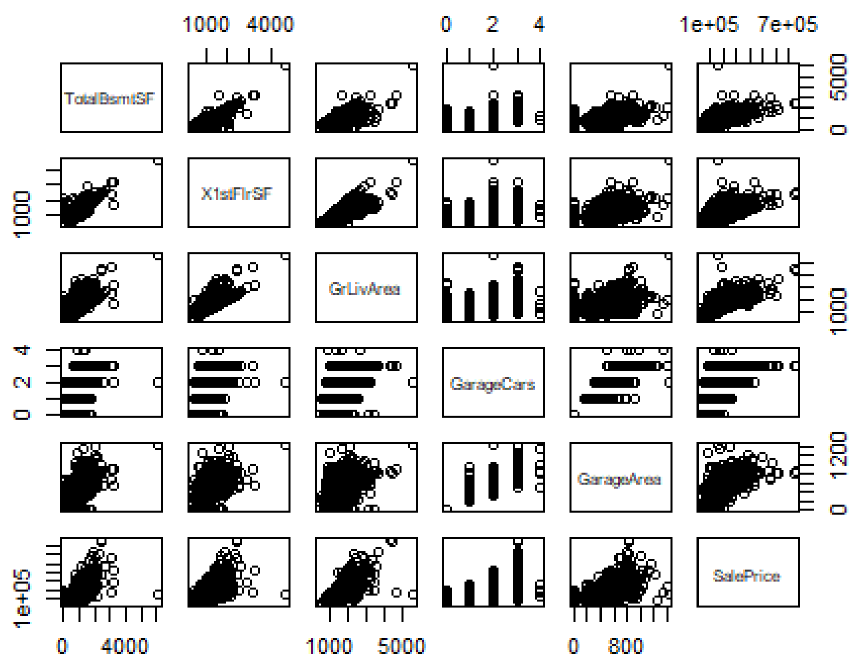
```
## MoSold        -0.1505766122  0.051568064
## YrSold         1.0000000000 -0.011868823
## SalePrice     -0.0118688234  1.000000000
```

```
#corrplot(matriz_cor)
```

Tomando en cuenta el bajo valor de correlaciónse eliminan todas las variables con un valor menor a 0.6, del modelo.

Se grafican las variables en pares para observar la relación que puede existir entre las mismas, además de realizar un grafico calor.

```
data <- select(datos, TotalBsmtSF,X1stFlrSF,GrLivArea,GarageCars,GarageArea,
SalePrice)
data <- na.omit(data)
plot(data)
```

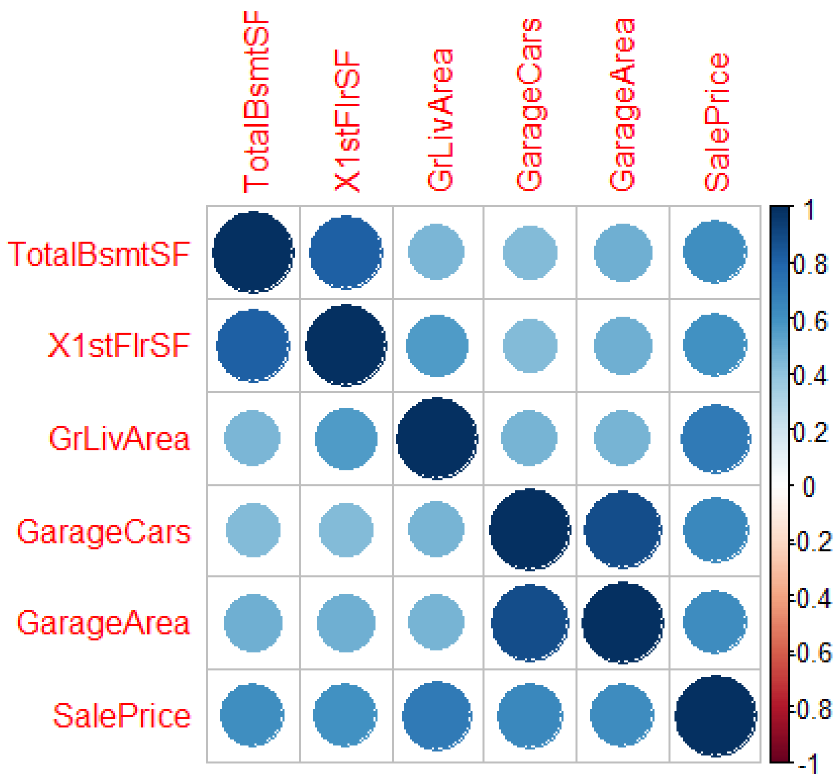



```
matriz_cor <- cor(data)
matriz_cor

##             TotalBsmtSF X1stFlrSF GrLivArea GarageCars GarageArea
SalePrice
## TotalBsmtSF   1.0000000 0.8195300 0.4548682  0.4345848  0.4866655
0.6135806
## X1stFlrSF     0.8195300 1.0000000 0.5660240  0.4393168  0.4897817
0.6058522
## GrLivArea     0.4548682 0.5660240 1.0000000  0.4672474  0.4689975
0.7086245
```

```
## GarageCars    0.4345848 0.4393168 0.4672474  1.0000000  0.8824754
0.6404092
## GarageArea    0.4866655 0.4897817 0.4689975  0.8824754  1.0000000
0.6234314
## SalePrice     0.6135806 0.6058522 0.7086245  0.6404092  0.6234314
1.0000000
```

```r
corrplot(matriz_cor)
```

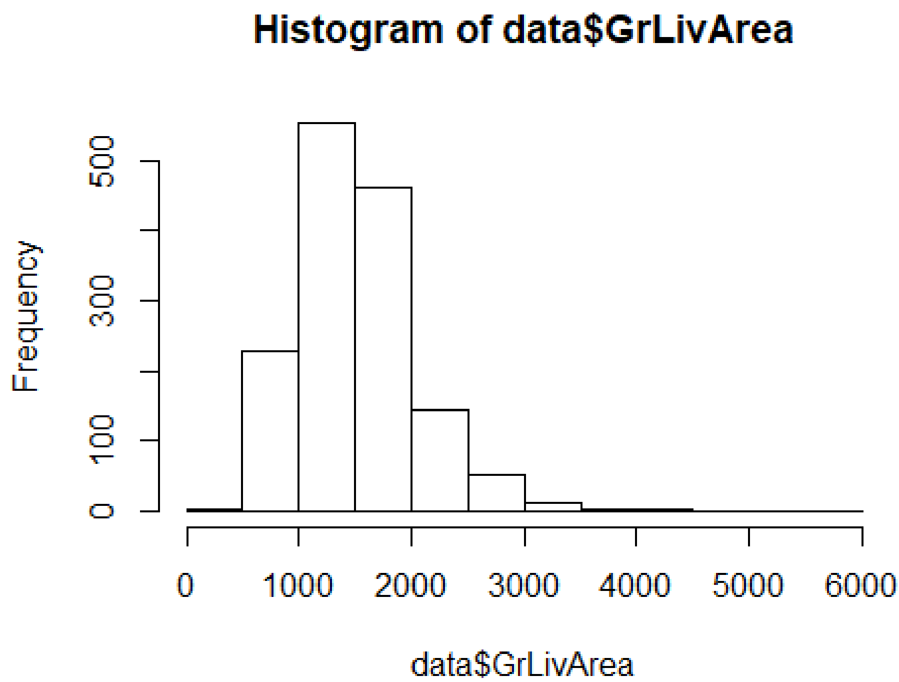


Por el alto valor de correlación entre TotalBsmtSF y X1stFlrSF. Se elimina X1stFlrSF, por tener menor valor de correlación contra SalePrice. De igual manera se elimina GarageArea.

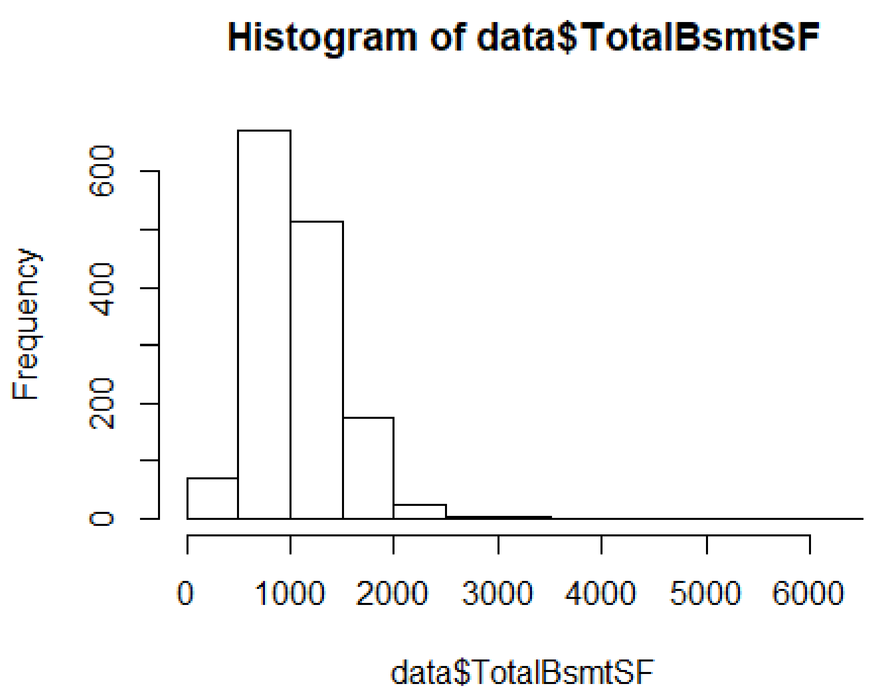
```r
data <- select(datos, TotalBsmtSF,GrLivArea,GarageCars, SalePrice)
data <- na.omit(data)
```

Se procede a realizar un test de normalidad para las variables no categoricas.

```r
hist(data$GrLivArea)
```




```
hist(data$TotalBsmtSF)
```



```r
library(normtest)
library(nortest)
sf.test(data$TotalBsmtSF)
```

```
##
##  Shapiro-Francia normality test
##
## data:  data$TotalBsmtSF
## W = 0.91517, p-value < 2.2e-16
```

```r
sf.test(data$GrLivArea)
```

```
##
##  Shapiro-Francia normality test
##
## data:  data$GrLivArea
## W = 0.92687, p-value < 2.2e-16
```

Utilizando el test de normalidad de Shapiro-Francia, podemos afirmar que las varibales mostradas anteriormente cuentan con una distribución normal.

A continuación se procede a la generación de los clústeres mediante K-means.

```r
data <- select(datos, TotalBsmtSF,GrLivArea,GarageCars, SalePrice)
data <- na.omit(data)
cluster <- data
km<-kmeans(data, 3)
data$grupo<-km$cluster

g1 <- data[data$grupo==1, ]
g2 <- data[data$grupo==2, ]
g3 <- data[data$grupo==3, ]

summary(g1$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  112500  133000  129105  148500  172000
```

```r
summary(g2$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  295000  318000  342643  373287  394809  755000
```

```r
summary(g3$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  172400  185888  208400  215390  239547  294000
```

```r
data$grupo <- mapvalues(data$grupo, c(1,2,3), c("Bajo", "Alto",
"Intermedio")) #El orden del identificador depende de los resumentes de la
celda anterior
```

Como se puede observar se genera una clasificación adecuada de grupos,Intermedio,Alto y Bajo.

Particionamos el dataset en test y train, pero poder realizar un proceso de entrenamiento (con train) y verificar el modelo (con test)

```
porcentaje<-0.7
set.seed(123)

corte <- sample(nrow(data), nrow(data) * porcentaje)
train <- data[corte, ]
test <- data[-corte, ]
```

Ahora sí… creamos el modelo de Naive Bayes y observemos cómo rinde el modelo con la matriz de confusión:

```
head(test)

##    TotalBsmtSF GrLivArea GarageCars SalePrice      grupo
## 1          856      1710          2    208500 Intermedio
## 3          920      1786          2    223500 Intermedio
## 7         1686      1694          2    307000       Alto
## 14        1494      1494          3    279500 Intermedio
## 15        1253      1253          1    157000       Bajo
## 21        1158      2376          3    325300       Alto

modelo<-naiveBayes(as.factor(train$grupo)~., data=train)
predBayes<-predict(modelo, newdata = test[, 1:4])
confusionMatrix(table(predBayes, test$grupo))

## Confusion Matrix and Statistics
##
##
## predBayes    Alto Bajo Intermedio
##   Alto         33    0          7
##   Bajo          0  219          7
##   Intermedio    3   12        158
##
## Overall Statistics
##
##                Accuracy : 0.9339
##                  95% CI : (0.9065, 0.9553)
##     No Information Rate : 0.5262
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8835
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: Alto Class: Bajo Class: Intermedio
## Sensitivity              0.91667      0.9481            0.9186
## Specificity              0.98263      0.9663            0.9438
## Pos Pred Value           0.82500      0.9690            0.9133
## Neg Pred Value           0.99248      0.9437            0.9474
## Prevalence               0.08200      0.5262            0.3918
## Detection Rate           0.07517      0.4989            0.3599
## Detection Prevalence     0.09112      0.5148            0.3941
## Balanced Accuracy        0.94965      0.9572            0.9312
```

La sensitividad y la especificidad son bastantes altas, pues, están por encima de 0.94 en promedio, por lo tanto, podemos decir que el Naive Bayes está acertando. Además, hay un accuracy del 93.85%, así que podemos asumir que no hay overfitting porque se ajustó bastante bien con el dataset de test. También, se confirma que el modelo está bastante bien pues en la matriz de confusión:

```
  * Las casas de alto precio se acertaron 33 de 40 = 83%.
  * Las casas de precio bajo se acertaron 219 de 226 = 98%.
  * Las casas de precio intermedio se certaron 158 de 173 = 91%.
```

Ahora, lo haremos con Cross Validation (con Caret):

```
## + Fold01: usekernel= TRUE, fL=0, adjust=1
## - Fold01: usekernel= TRUE, fL=0, adjust=1
## + Fold01: usekernel=FALSE, fL=0, adjust=1
## - Fold01: usekernel=FALSE, fL=0, adjust=1
## + Fold02: usekernel= TRUE, fL=0, adjust=1
## - Fold02: usekernel= TRUE, fL=0, adjust=1
## + Fold02: usekernel=FALSE, fL=0, adjust=1
## - Fold02: usekernel=FALSE, fL=0, adjust=1
## + Fold03: usekernel= TRUE, fL=0, adjust=1
## - Fold03: usekernel= TRUE, fL=0, adjust=1
## + Fold03: usekernel=FALSE, fL=0, adjust=1
## - Fold03: usekernel=FALSE, fL=0, adjust=1
## + Fold04: usekernel= TRUE, fL=0, adjust=1
## - Fold04: usekernel= TRUE, fL=0, adjust=1
## + Fold04: usekernel=FALSE, fL=0, adjust=1
## - Fold04: usekernel=FALSE, fL=0, adjust=1
## + Fold05: usekernel= TRUE, fL=0, adjust=1
## - Fold05: usekernel= TRUE, fL=0, adjust=1
## + Fold05: usekernel=FALSE, fL=0, adjust=1
## - Fold05: usekernel=FALSE, fL=0, adjust=1
## + Fold06: usekernel= TRUE, fL=0, adjust=1
## - Fold06: usekernel= TRUE, fL=0, adjust=1
## + Fold06: usekernel=FALSE, fL=0, adjust=1
## - Fold06: usekernel=FALSE, fL=0, adjust=1
## + Fold07: usekernel= TRUE, fL=0, adjust=1
## - Fold07: usekernel= TRUE, fL=0, adjust=1
## + Fold07: usekernel=FALSE, fL=0, adjust=1
```

```
## - Fold07: usekernel=FALSE, fL=0, adjust=1
## + Fold08: usekernel= TRUE, fL=0, adjust=1
## - Fold08: usekernel= TRUE, fL=0, adjust=1
## + Fold08: usekernel=FALSE, fL=0, adjust=1
## - Fold08: usekernel=FALSE, fL=0, adjust=1
## + Fold09: usekernel= TRUE, fL=0, adjust=1
## - Fold09: usekernel= TRUE, fL=0, adjust=1
## + Fold09: usekernel=FALSE, fL=0, adjust=1
## - Fold09: usekernel=FALSE, fL=0, adjust=1
## + Fold10: usekernel= TRUE, fL=0, adjust=1
## - Fold10: usekernel= TRUE, fL=0, adjust=1
## + Fold10: usekernel=FALSE, fL=0, adjust=1
## - Fold10: usekernel=FALSE, fL=0, adjust=1
## Aggregating results
## Selecting tuning parameters
## Fitting fL = 0, usekernel = TRUE, adjust = 1 on full training set
```

```
confusionMatrix(table(prediccionCaret, test$grupo))
```

```
## Confusion Matrix and Statistics
##
##
## prediccionCaret Alto Bajo Intermedio
##      Alto         35    0          2
##      Bajo          0  222          3
##      Intermedio    1    9        167
##
## Overall Statistics
##
##                Accuracy : 0.9658
##                  95% CI : (0.9443, 0.9808)
##     No Information Rate : 0.5262
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9396
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Alto Class: Bajo Class: Intermedio
## Sensitivity              0.97222      0.9610            0.9709
## Specificity              0.99504      0.9856            0.9625
## Pos Pred Value           0.94595      0.9867            0.9435
## Neg Pred Value           0.99751      0.9579            0.9809
## Prevalence               0.08200      0.5262            0.3918
## Detection Rate           0.07973      0.5057            0.3804
## Detection Prevalence     0.08428      0.5125            0.4032
## Balanced Accuracy        0.98363      0.9733            0.9667
```

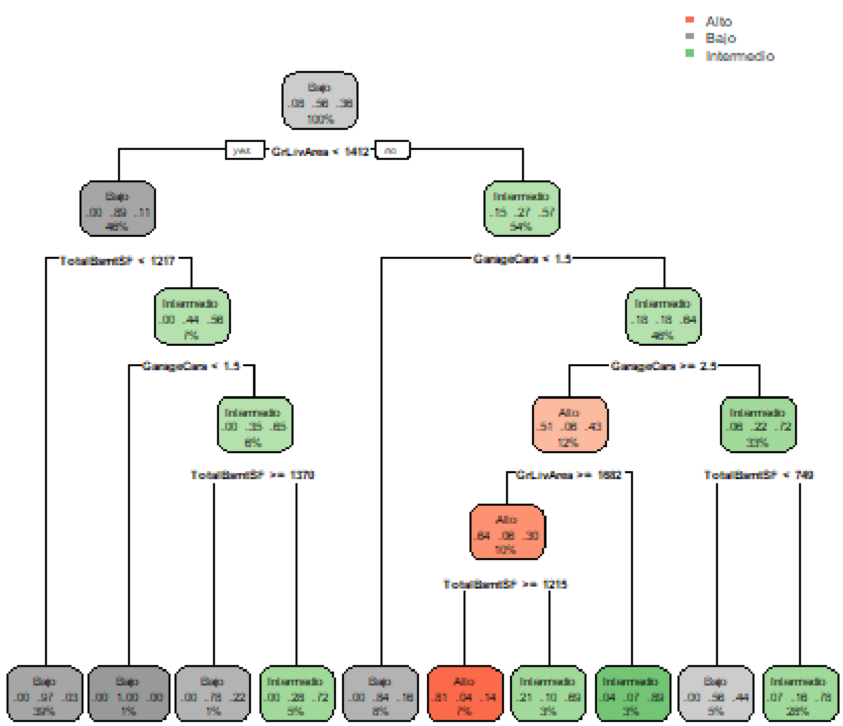
En el proceso de Cross Validation, notamos que es ligeramente más certero que Naive Bayes, debido a:

```
* Las casas de alto precio se acertaron 35 de 37 = 95%.
* Las casas de precio bajo se acertaron 222 de 225 = 99%.
* Las casas de precio intermedio se certaron 167 de 177 = 94%.
```

Todos están arriba del proceso anterior, además, se obtuvo un 0.95 de accuracy y el promedio de la sensitividad y la especificidad es del 0.96.

Árbol de clasificación

```r
#Clasiffication Tree
dt_model<-rpart(train$grupo~.,train[1:3],method = "class") #Genrar modelo
rpart.plot(dt_model)
```




```r
prediccionCT <- predict(dt_model, newdata = test[, 1:3]) #Predecir
#prediccionCT
columnaMasAlta<-apply(prediccionCT, 1, function(x)
colnames(prediccionCT)[which.max(x)])
test$prediccionCT<-columnaMasAlta
confusionMatrix(table(test$prediccionCT,test$grupo))
```

```
## Confusion Matrix and Statistics
##
##
##               Alto Bajo Intermedio
```

```
##   Alto         21    0          9
##   Bajo          0  200         38
##   Intermedio   15   31        125
##
## Overall Statistics
##
##                Accuracy : 0.7882
##                  95% CI : (0.7469, 0.8255)
##     No Information Rate : 0.5262
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6193
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Alto Class: Bajo Class: Intermedio
## Sensitivity              0.58333      0.8658            0.7267
## Specificity              0.97767      0.8173            0.8277
## Pos Pred Value           0.70000      0.8403            0.7310
## Neg Pred Value           0.96333      0.8458            0.8246
## Prevalence               0.08200      0.5262            0.3918
## Detection Rate           0.04784      0.4556            0.2847
## Detection Prevalence     0.06834      0.5421            0.3895
## Balanced Accuracy        0.78050      0.8416            0.7772
```

Como se puede observar el árbol de clasificación tiene un accuracy del 81.55%, así que podemos asumir que no hay overfitting porque se ajustó bastante bien con el dataset de test. A partir de la matriz de confusión podemos observar los siguintes valores.

```
  * Las casas de alto precio se acertaron 47 de 28 = 60%.
  * Las casas de precio bajo se acertaron 228 de 209 = 92%.
  * Las casas de precio intermedio se certaron 164 de 121 = 74%.
```

Conclusión

Podemos afirmar que el algoritmo de Naive Bayes, es superior a el árbol de clasificación. Esto gracias a que al comparar factores como el accuracy y la sensitividad y especificidad son mejores. Además los porcentajes de predicción para cada uno de los grupos también es mejor. En el caso de el árbol de clasificación, el hecho de que tenga un porcentaje del 60% para las casas de alto precio, puede llegar a ser una desventaja. Gracias a que puede significar una perdida monetaria. Por lo que se recomienda la utlización de naive bayes en su lugar.