Lab6

Javier Mombiela, Jose Hernandez, Pablo Gonzalez

2023-03-10

Lab 6 Regresión Logística

```
datos <-read.csv("train.csv")
datos_numericos <- datos %>%
  select_if(is.numeric)
cualitativas <- datos %>%
  select_if(.predicate = function(x) !is.numeric(x))
datos <- datos %>% mutate_at(colnames(cualitativas), function(x) as.factor(x))
datos_numericos <-datos_numericos[complete.cases(datos_numericos),]
```

```
datos_numericos <-scale(na.omit(datos_numericos))
```

Creacion de la variable de clasificacion de precios

```
datos_numericos <-data.frame(datos_numericos)
q1 <- quantile(datos_numericos$SalePrice,0.33)
q2 <- quantile(datos_numericos$SalePrice,0.5)
q3 <-quantile(datos_numericos$SalePrice,0.7)
datos_numericos$clasificacion <- sapply(datos_numericos$SalePrice, function(x) ifelse(x <= q1, "Economi
datos_numericos$clasificacion <-factor(datos_numericos$clasificacion)
```

1.1 Crear variables dicotomicas

```
datos_con_dummy <- dummy_cols(datos_numericos, select_columns = c("clasificacion"))
datos_con_dummy <- select(datos_con_dummy, -clasificacion, -clasificacion_Economicas, -clasificacion_In
datos_con_dummy$clasificacion_Caras <- datos_con_dummy$clasificacion_Caras
datos_con_dummy<-datos_con_dummy %>% mutate_at(c("clasificacion_Caras"),as.factor)
```

1.2 Datos de test y datos de entrenamiento

```
porcentaje <- 0.7
set.seed(123)
datos_con_dummy <-select(datos_con_dummy, -Id)
```

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

1.3 Modelo con todas las variables numericas

```
modelo_logistico<-glm(clasificacion_Caras~., data = train,family = binomial(), maxit=100)
```

1.4 Summary del modelo

```
summary(modelo_logistico)
```

```
##
## Call:
## glm(formula = clasificacion_Caras ~ ., family = binomial(), data = train,
##     maxit = 100)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5292  -0.6429  -0.1555   0.3976   2.7200
##
## Coefficients: (2 not defined because of singularities)
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.481119   0.148289   3.244 0.001177 **
## MSSubClass     0.250077   0.131724   1.898 0.057631 .
## LotFrontage   -0.021859   0.156324  -0.140 0.888792
## LotArea       -0.008925   0.279412  -0.032 0.974519
## OverallQual   -0.105849   0.228465  -0.463 0.643148
## OverallCond    0.321843   0.153238   2.100 0.035704 *
## YearBuilt      0.208071   0.287282   0.724 0.468896
## YearRemodAdd   0.041232   0.184905   0.223 0.823542
## MasVnrArea    -0.030529   0.150329  -0.203 0.839073
## BsmtFinSF1     0.498407   0.276847   1.800 0.071814 .
## BsmtFinSF2     0.225911   0.133654   1.690 0.090976 .
## BsmtUnfSF     -0.080507   0.250915  -0.321 0.748322
## TotalBsmtSF          NA         NA      NA       NA
## X1stFlrSF     -0.043316   0.296707  -0.146 0.883929
## X2ndFlrSF     -0.177215   0.291815  -0.607 0.543660
## LowQualFinSF  -0.200798   0.146102  -1.374 0.169329
## GrLivArea            NA         NA      NA       NA
## BsmtFullBath  -0.299089   0.156599  -1.910 0.056146 .
## BsmtHalfBath  -0.238155   0.103013  -2.312 0.020783 *
## FullBath      -0.692821   0.196387  -3.528 0.000419 ***
## HalfBath      -0.063232   0.157973  -0.400 0.688957
## BedroomAbvGr   0.355298   0.173124   2.052 0.040143 *
## KitchenAbvGr   0.241149   0.142519   1.692 0.090637 .
## TotRmsAbvGrd  -0.111033   0.261812  -0.424 0.671498
## Fireplaces    -0.043019   0.127677  -0.337 0.736165
## GarageYrBlt   -0.299592   0.239700  -1.250 0.211350
```

```
## GarageCars    -0.164900   0.227305  -0.725 0.468172
## GarageArea     0.591465   0.223689   2.644 0.008190 **
## WoodDeckSF    -0.007878   0.111001  -0.071 0.943423
## OpenPorchSF   -0.014229   0.125931  -0.113 0.910035
## EnclosedPorch  0.152448   0.126980   1.201 0.229920
## X3SsnPorch     0.158793   0.112554   1.411 0.158299
## ScreenPorch   -0.005026   0.115722  -0.043 0.965360
## PoolArea      -0.132070   0.088467  -1.493 0.135472
## MiscVal        0.004535   0.104879   0.043 0.965508
## MoSold         0.051329   0.112646   0.456 0.648629
## YrSold        -0.095860   0.109926  -0.872 0.383187
## SalePrice      4.442238   0.506333   8.773  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1080.2  on 783  degrees of freedom
## Residual deviance:  577.5  on 748  degrees of freedom
## AIC: 649.5
##
## Number of Fisher Scoring iterations: 7
```

Como se puede visualizar en el summary existen variables que tienen un coeficiente significativo estas son: -SalePrice -GarageArea -FullBath -BedroomAbvGr -BsmtHalfBath -OverallCond

1.5 correlacion de datos

```
set.seed(123)
datos_numericos <-select(datos_numericos, -clasificacion)
correlac <- cor(datos_numericos)
correlac <- cor(datos_numericos)
indices <- which(correlac > 0.6 & upper.tri(correlac), arr.ind = TRUE)
nombres <- colnames(correlac)
for (i in 1:nrow(indices)) {
  fila <- indices[i, 1]
  col <- indices[i, 2]
  cat(nombres[fila], "y", nombres[col], "tienen una correlación de", correlac[fila, col], "\n")
}
```

```
## YearBuilt y YearRemodAdd tienen una correlación de 0.6231713
## TotalBsmtSF y X1stFlrSF tienen una correlación de 0.8359994
## OverallQual y GrLivArea tienen una correlación de 0.6074661
## X2ndFlrSF y GrLivArea tienen una correlación de 0.6882916
## BsmtFinSF1 y BsmtFullBath tienen una correlación de 0.6517267
## GrLivArea y FullBath tienen una correlación de 0.6148873
## X2ndFlrSF y HalfBath tienen una correlación de 0.6063367
## X2ndFlrSF y TotRmsAbvGrd tienen una correlación de 0.6177759
## GrLivArea y TotRmsAbvGrd tienen una correlación de 0.8243121
## BedroomAbvGr y TotRmsAbvGrd tienen una correlación de 0.6502846
## YearBuilt y GarageYrBlt tienen una correlación de 0.8235195
## YearRemodAdd y GarageYrBlt tienen una correlación de 0.6458085
```

```
## GarageYrBlt y GarageCars tienen una correlación de 0.6009034
## GarageCars y GarageArea tienen una correlación de 0.8394149
## OverallQual y SalePrice tienen una correlación de 0.7978807
## TotalBsmtSF y SalePrice tienen una correlación de 0.6156122
## X1stFlrSF y SalePrice tienen una correlación de 0.6079691
## GrLivArea y SalePrice tienen una correlación de 0.7051536
## GarageCars y SalePrice tienen una correlación de 0.6470336
## GarageArea y SalePrice tienen una correlación de 0.6193296
```

1.6 Prediccion con el modelo

```
pred<-predict(modelo_logistico,newdata = test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

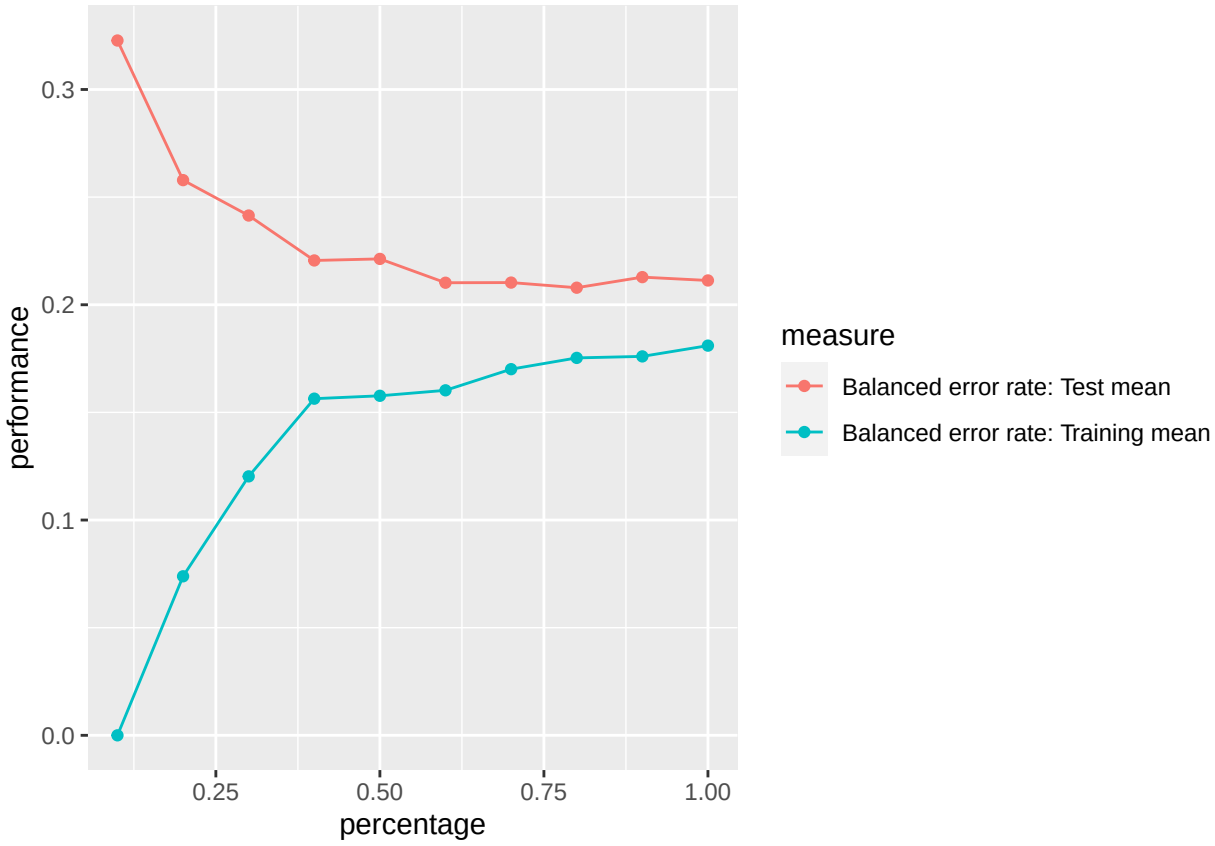
```
prediccion<-ifelse(pred>=0.5,1,0)
confusionMatrix(as.factor(test$clasificacion_Caras),as.factor(prediccion))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 151  31
##          1  48 107
##
##                Accuracy : 0.7656
##                  95% CI : (0.7166, 0.8098)
##     No Information Rate : 0.5905
##     P-Value [Acc > NIR] : 1.003e-11
##
##                   Kappa : 0.5243
##
##  Mcnemar's Test P-Value : 0.07184
##
##             Sensitivity : 0.7588
##             Specificity : 0.7754
##          Pos Pred Value : 0.8297
##          Neg Pred Value : 0.6903
##              Prevalence : 0.5905
##          Detection Rate : 0.4481
##    Detection Prevalence : 0.5401
##       Balanced Accuracy : 0.7671
##
##        'Positive' Class : 0
##
```

Como se puede visualizar con las variables numericas se cuenta con un buen modelo esto debido a que se tiene un modelo con 0.76 de accuracy lo cual es muy bueno, este tambien cuenta con una sensibilidad de 0.76 y una specifity de 0.77.

1.7 curva de aprendizage

```
datos.task = makeClassifTask(data = train, target = "clasificacion_Caras")
rin2 = makeResampleDesc(method = "CV", iters = 10, predict = "both")
lrn = makeLearner("classif.multinom", predict.type = "prob", trace = FALSE)
lc2 = generateLearningCurveData(learners = lrn, task = datos.task,
                                percs = seq(0.1, 1, by = 0.1),
                                measures = list(ber, setAggregation(ber, train.mean)), resampling = rin
                                show.info = FALSE)
plotLearningCurve(lc2, facet = "learner")
```



Al momento de observar las cuarvas de aprendizage de nuestro modelo se puede conlcluir que el modelo no cuenta con Overfitting esto debido a que las dos curvas corvengen a un mismo punto lo cual nos da un indicador de que noe xiste overfitting de nuestro modelo.

Seleccion del segundo modelo

Para el segundo modelo se usaran las variables que tengan una relacion entre 0.6 y 0.70 respectivamente YearBuilt YearRemodAdd OverallQual GrLivArea X2ndFlrSF BsmtFinSF1 BsmtFullBath FullBath HalfBath X2ndFlrSF TotRmsAbvGrd BedroomAbvGr GarageYrBlt GarageCars TotalBsmtSF
SalePrice
X1stFlrSF SalePrice GarageArea

1.8 Datos de test2 y datos de entrenamiento2

```
porcentaje <- 0.7
set.seed(123)
# Seleccionar las columnas deseadas
datos_seleccionados <- datos_con_dummy[, c("YearBuilt", "YearRemodAdd", "OverallQual", "GrLivArea", "X2
corte <- sample(nrow(datos_seleccionados), nrow(datos_seleccionados) * porcentaje)
train <- datos_seleccionados[corte, ]
test <- datos_seleccionados[-corte, ]
```

1.9 Modelo con todas las variables numericas

```
modelo_logistico2<-glm(clasificacion_Caras~., data = train,family = binomial(), maxit=100)
```

1.10 Summary del modelo

```
summary(modelo_logistico2)
```

```
##
## Call:
## glm(formula = clasificacion_Caras ~ ., family = binomial(), data = train,
##     maxit = 100)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4904  -0.6544  -0.2011   0.4125   2.5141
##
## Coefficients: (2 not defined because of singularities)
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.43520    0.13420   3.243  0.00118 **
## YearBuilt    -0.16658    0.23287  -0.715  0.47439
## YearRemodAdd  0.13837    0.15104   0.916  0.35959
## OverallQual  -0.10064    0.21158  -0.476  0.63433
## GrLivArea    -3.49704    2.29765  -1.522  0.12801
## X2ndFlrSF     2.76844    1.94391   1.424  0.15440
## BsmtFinSF1    0.36677    0.16360   2.242  0.02497 *
## BsmtFullBath -0.06137    0.12937  -0.474  0.63524
## FullBath     -0.51308    0.18320  -2.801  0.00510 **
## HalfBath     -0.05900    0.15003  -0.393  0.69413
## X2ndFlrSF.1        NA         NA      NA       NA
## TotRmsAbvGrd -0.03613    0.24138  -0.150  0.88100
## BedroomAbvGr  0.31497    0.15890   1.982  0.04745 *
## GarageYrBlt  -0.29176    0.22158  -1.317  0.18793
## GarageCars   -0.11807    0.21262  -0.555  0.57868
## TotalBsmtSF  -0.05192    0.24303  -0.214  0.83082
## SalePrice     3.92282    0.44562   8.803  < 2e-16 ***
## X1stFlrSF     2.47821    1.71567   1.444  0.14861
## SalePrice.1        NA         NA      NA       NA
```

```
## GarageArea    0.60287    0.20772   2.902  0.00370 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1080.23  on 783  degrees of freedom
## Residual deviance:  603.15  on 766  degrees of freedom
## AIC: 639.15
##
## Number of Fisher Scoring iterations: 6
```

Como se puede visualizar en el summary existen variables que tienen un coeficiente significativo estas son: -SalePrice -GarageArea -FullBath -BedroomAbvGr -BsmtFinSF1

1.11 Prediccion con el modelo

```
pred<-predict(modelo_logistico2,newdata = test, type = "response")
```

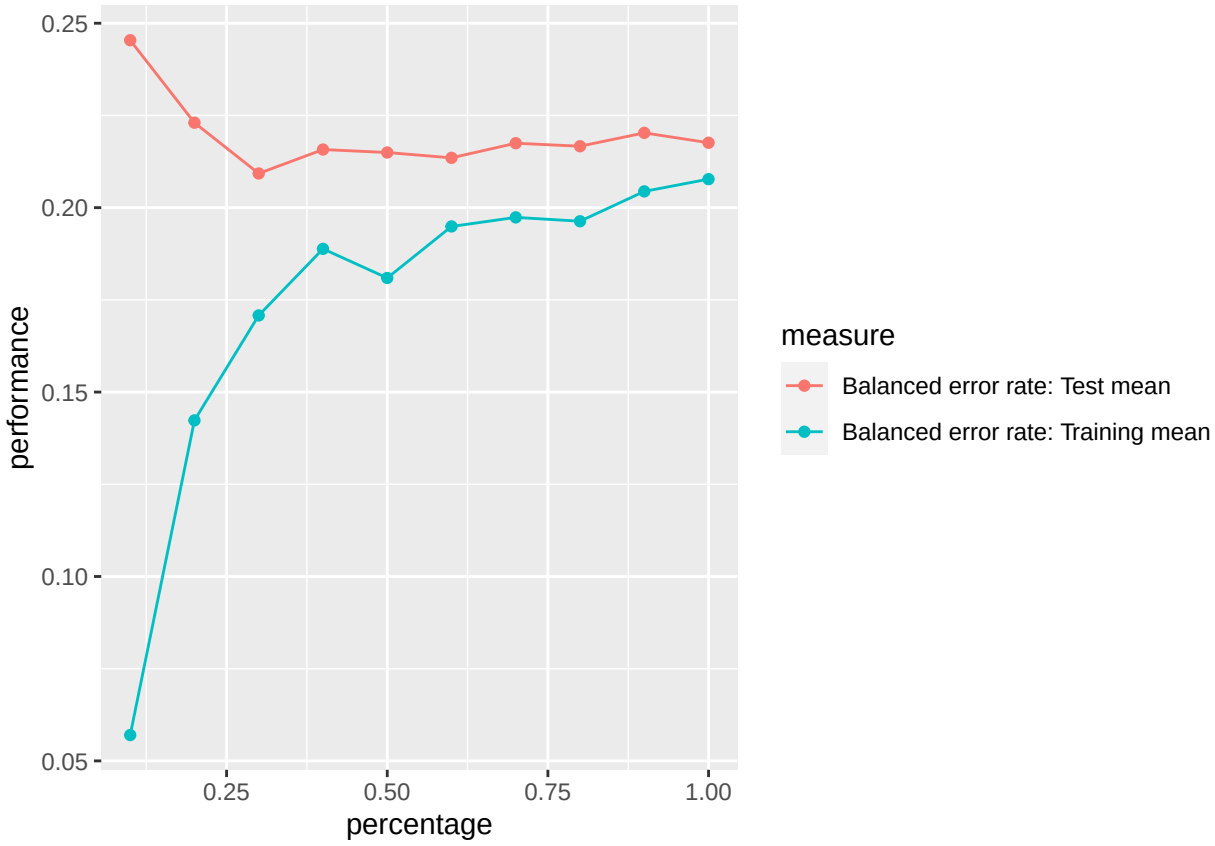
```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
prediccion<-ifelse(pred>=0.5,1,0)
confusionMatrix(as.factor(test$clasificacion_Caras),as.factor(prediccion))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 149  33
##          1  47 108
##
##                Accuracy : 0.7626
##                  95% CI : (0.7135, 0.807)
##     No Information Rate : 0.5816
##     P-Value [Acc > NIR] : 2.489e-12
##
##                   Kappa : 0.5189
##
##  Mcnemar's Test P-Value : 0.1461
##
##             Sensitivity : 0.7602
##             Specificity : 0.7660
##          Pos Pred Value : 0.8187
##          Neg Pred Value : 0.6968
##              Prevalence : 0.5816
##          Detection Rate : 0.4421
##    Detection Prevalence : 0.5401
##       Balanced Accuracy : 0.7631
##
##        'Positive' Class : 0
##
```

Se puede mencionar que este modelo obtuvo una accuracy de 0.7626 por lo que es unn poco menos peor que el anterior ya antes planteado con las otras variables se puede mencionar que este tambien cuenta con un specifity de 0.7660.

```
datos.task = makeClassifTask(data = train, target = "clasificacion_Caras")
rin2 = makeResampleDesc(method = "CV", iters = 10, predict = "both")
lrn = makeLearner("classif.multinom", predict.type = "prob", trace = FALSE)
lc2 = generateLearningCurveData(learners = lrn, task = datos.task,
                                percs = seq(0.1, 1, by = 0.1),
                                measures = list(ber, setAggregation(ber, train.mean)), resampling = rin
                                show.info = FALSE)
plotLearningCurve(lc2, facet = "learner")
```



Al momento de observar las cuarvas de aprendizage de nuestro modelo se puede conlcluir que el modelo no cuenta con Overfitting esto debido a que las dos curvas corvengen a un mismo punto lo cual nos da un indicador de que noe xiste overfitting de nuestro modelo.

Seleccion del tercer modelo

Para el segundo modelo se usaran las variables que tengan una relacion mayor a 0.70 respectivamente

1.12 Datos de test3 y datos de entrenamiento3

TotalBsmtSF X1stFlrSF
GrLivArea TotRmsAbvGrd YearBuilt GarageYrBlt GarageCars GarageArea SalePrice

```r
porcentaje <- 0.7
set.seed(123)
# Seleccionar las columnas deseadas
datos_seleccionados <- datos_con_dummy[,c("TotalBsmtSF", "X1stFlrSF", "GrLivArea", "TotRmsAbvGrd", "Yea
corte <- sample(nrow(datos_seleccionados), nrow(datos_seleccionados) * porcentaje)
train <- datos_seleccionados[corte, ]
test <- datos_seleccionados[-corte, ]
```

1.13 Modelo con todas las variables numericas

```r
modelo_logistico3<-glm(clasificacion_Caras~., data = train,family = binomial(), maxit=100)
```

1.14 Summary del modelo

```r
summary(modelo_logistico3)
```

```
##
## Call:
## glm(formula = clasificacion_Caras ~ ., family = binomial(), data = train,
##     maxit = 100)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1820  -0.6365  -0.2461   0.4657   2.0548
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.36930    0.12166   3.035 0.002401 **
## TotalBsmtSF   0.16798    0.22019   0.763 0.445541
## X1stFlrSF    -0.01862    0.21845  -0.085 0.932061
## GrLivArea    -0.39722    0.23269  -1.707 0.087799 .
## TotRmsAbvGrd  0.05355    0.19972   0.268 0.788598
## YearBuilt    -0.19479    0.21374  -0.911 0.362109
## GarageYrBlt  -0.42525    0.20609  -2.063 0.039069 *
## GarageCars   -0.26843    0.19996  -1.342 0.179460
## GarageArea    0.74927    0.19756   3.793 0.000149 ***
## SalePrice     3.54281    0.35050  10.108  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1080.23  on 783  degrees of freedom
## Residual deviance:  627.24  on 774  degrees of freedom
## AIC: 647.24
##
## Number of Fisher Scoring iterations: 6
```

Como se puede visualizar en el summary existen variables que tienen un coeficiente significativo estas son: -SalePrice -GarageArea -GarageYrBlt

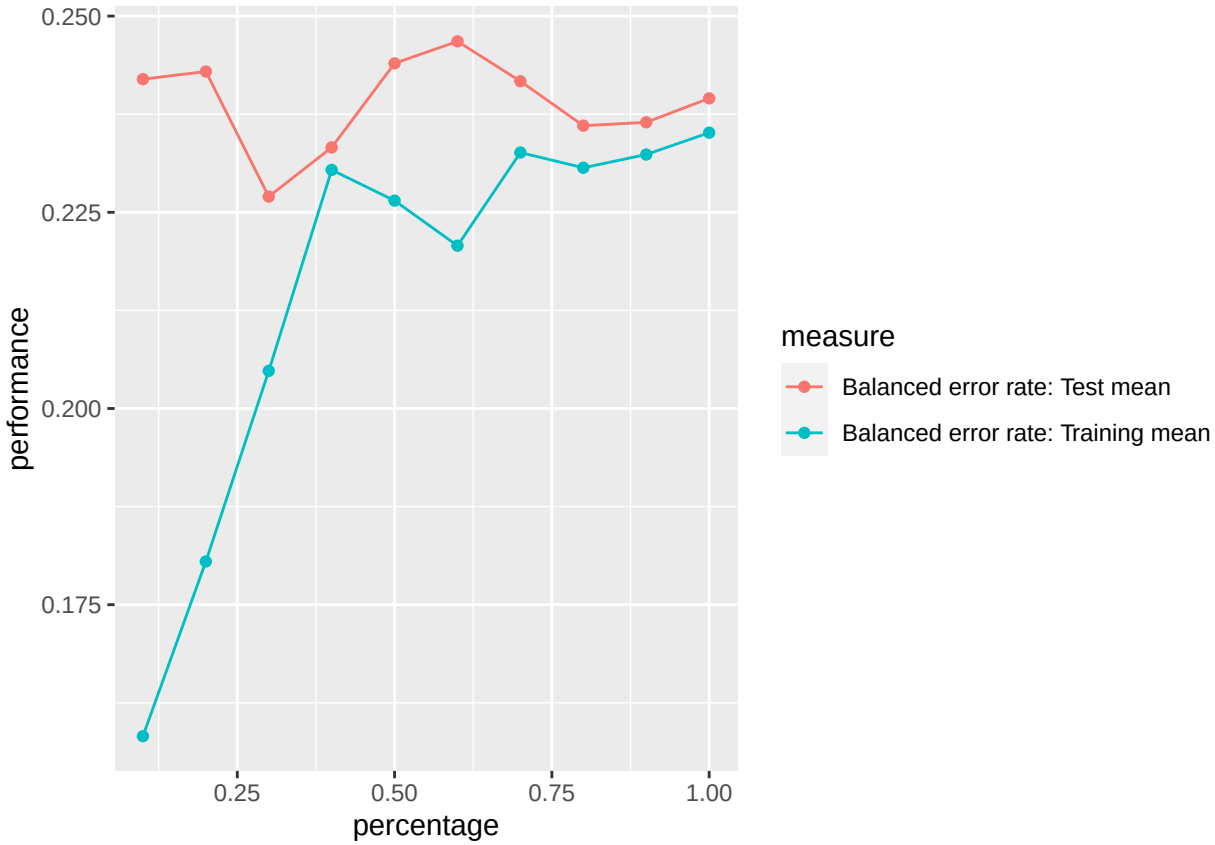
1.15 Prediccion con el modelo

```
pred<-predict(modelo_logistico3,newdata = test, type = "response")
prediccion<-ifelse(pred>=0.5,1,0)
confusionMatrix(as.factor(test$clasificacion_Caras),as.factor(prediccion))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 149  33
##          1  51 104
##
##                Accuracy : 0.7507
##                  95% CI : (0.701, 0.796)
##     No Information Rate : 0.5935
##     P-Value [Acc > NIR] : 9.942e-10
##
##                   Kappa : 0.4939
##
##  Mcnemar's Test P-Value : 0.06362
##
##             Sensitivity : 0.7450
##             Specificity : 0.7591
##          Pos Pred Value : 0.8187
##          Neg Pred Value : 0.6710
##              Prevalence : 0.5935
##          Detection Rate : 0.4421
##    Detection Prevalence : 0.5401
##       Balanced Accuracy : 0.7521
##
##        'Positive' Class : 0
##
```

Como se puede visaulzizar el modelo obtuvo un accuracy de 0.75 lo cual no es mayor al primer modelo por lo que el primer modelo es el mehor se puede mencionar que este obtuvo un specifity de 0.75.

```
datos.task = makeClassifTask(data = train, target = "clasificacion_Caras")
rin2 = makeResampleDesc(method = "CV", iters = 10, predict = "both")
lrn = makeLearner("classif.multinom", predict.type = "prob", trace = FALSE)
lc2 = generateLearningCurveData(learners = lrn, task = datos.task,
                                percs = seq(0.1, 1, by = 0.1),
                                measures = list(ber, setAggregation(ber, train.mean)), resampling = rin
                                show.info = FALSE)
plotLearningCurve(lc2, facet = "learner")
```



Al momento de observar las cuarvas de aprendizage de nuestro modelo se puede conlcluir que el modelo no cuenta con Overfitting esto debido a que las dos curvas corvengen a un mismo punto lo cual nos da un indicador de que noe xiste overfitting de nuestro modelo.

Determinar cual es el mejor modelo

Se pudo determinar que el mejor modelo fu el primer modelo esto debido a que cuenta con una cantidad mayor de accuracy lo cual nos dice que este es un mejor modelo el modelo obtuvo un accuracy de 0.77 mientras que los otros dos modelos de 0.76 y 0.75 lo cual no es una gran diferencia pero ya es un poco mejor que los otros.

Metodo arboles de decision

```
porcentaje <- 0.7
set.seed(123)
corte <- sample(nrow(datos_con_dummy), nrow(datos_con_dummy) * porcentaje)
train <- datos_con_dummy[corte, ]
test <- datos_con_dummy[-corte, ]
```

1.16 arboles de decision

```
regression_tree <-rpart(formula = clasificacion_Caras ~.,data = train)
```

1.17 Summary arboles de decision

```
summary(regression_tree)
```

```
## Call:
## rpart(formula = clasificacion_Caras ~ ., data = train)
##   n= 784
##
##          CP nsplit rel error      xerror        xstd
## 1 0.6544944      0 1.0000000 1.000000000 0.039159655
## 2 0.1727528      1 0.3455056 0.351123596 0.028793183
## 3 0.0100000      3 0.0000000 0.008426966 0.004855994
##
## Variable importance
##   SalePrice OverallQual   GrLivArea  GarageCars  GarageArea    FullBath
##          34          14          10           9           8           7
##   YearBuilt TotalBsmtSF GarageYrBlt
##           7           7           4
##
## Node number 1: 784 observations,    complexity param=0.6544944
##   predicted class=0  expected loss=0.4540816  P(node) =1
##     class counts:   428   356
##    probabilities: 0.546 0.454
##   left son=2 (551 obs) right son=3 (233 obs)
##   Primary splits:
##       SalePrice   < 0.210169     to the left,  improve=197.60860, (0 missing)
##       OverallQual < 0.2083466    to the left,  improve= 78.43926, (0 missing)
##       GarageArea  < -0.1357194   to the left,  improve= 75.93322, (0 missing)
##       TotalBsmtSF < -0.001380157 to the left,  improve= 68.14740, (0 missing)
##       GrLivArea   < -0.01415104  to the left,  improve= 60.72876, (0 missing)
##   Surrogate splits:
##       OverallQual < 0.9325528    to the left,  agree=0.842, adj=0.468, (0 split)
##       TotalBsmtSF < 0.7020764    to the left,  agree=0.821, adj=0.399, (0 split)
##       GarageCars  < 0.9495766    to the left,  agree=0.816, adj=0.382, (0 split)
##       GrLivArea   < 0.5567605    to the left,  agree=0.814, adj=0.373, (0 split)
##       GarageArea  < 0.3845609    to the left,  agree=0.809, adj=0.356, (0 split)
##
## Node number 2: 551 observations,    complexity param=0.1727528
##   predicted class=0  expected loss=0.2232305  P(node) =0.7028061
##     class counts:   428   123
##    probabilities: 0.777 0.223
##   left son=4 (277 obs) right son=5 (274 obs)
##   Primary splits:
##       SalePrice   < -0.5422483   to the left,  improve=55.515960, (0 missing)
##       GarageArea  < -0.3370339   to the left,  improve= 8.248107, (0 missing)
##       X1stFlrSF   < -0.3174326   to the left,  improve= 7.468676, (0 missing)
##       TotalBsmtSF < -0.2078172   to the left,  improve= 7.366203, (0 missing)
##       LotFrontage < 0.1168066    to the left,  improve= 7.037066, (0 missing)
```

```
##   Surrogate splits:
##       GarageCars  < -0.5787441   to the left,  agree=0.751, adj=0.500, (0 split)
##       YearBuilt   < 0.0398821    to the left,  agree=0.740, adj=0.478, (0 split)
##       OverallQual < -0.5158597   to the left,  agree=0.733, adj=0.464, (0 split)
##       GarageArea  < -0.6063248   to the left,  agree=0.731, adj=0.460, (0 split)
##       FullBath    < -0.1467534   to the left,  agree=0.726, adj=0.449, (0 split)
##
## Node number 3: 233 observations
##   predicted class=1  expected loss=0  P(node) =0.2971939
##     class counts:     0   233
##    probabilities: 0.000 1.000
##
## Node number 4: 277 observations
##   predicted class=0  expected loss=0  P(node) =0.3533163
##     class counts:   277     0
##    probabilities: 1.000 0.000
##
## Node number 5: 274 observations,    complexity param=0.1727528
##   predicted class=0  expected loss=0.4489051  P(node) =0.3494898
##     class counts:   151   123
##    probabilities: 0.551 0.449
##   left son=10 (151 obs) right son=11 (123 obs)
##   Primary splits:
##       SalePrice   < -0.2494742   to the right, improve=135.56930, (0 missing)
##       YearBuilt   < 0.3950906    to the right, improve= 29.82042, (0 missing)
##       FullBath    < -0.1467534   to the right, improve= 29.44409, (0 missing)
##       GrLivArea   < -0.5850625   to the right, improve= 26.40385, (0 missing)
##       GarageYrBlt < 0.429866     to the right, improve= 25.18373, (0 missing)
##   Surrogate splits:
##       FullBath    < -0.1467534   to the right, agree=0.737, adj=0.415, (0 split)
##       YearBuilt   < 0.3950906    to the right, agree=0.726, adj=0.390, (0 split)
##       GrLivArea   < -0.5850625   to the right, agree=0.708, adj=0.350, (0 split)
##       GarageYrBlt < 0.429866     to the right, agree=0.708, adj=0.350, (0 split)
##       OverallQual < -0.5158597   to the right, agree=0.697, adj=0.325, (0 split)
##
## Node number 10: 151 observations
##   predicted class=0  expected loss=0  P(node) =0.192602
##     class counts:   151     0
##    probabilities: 1.000 0.000
##
## Node number 11: 123 observations
##   predicted class=1  expected loss=0  P(node) =0.1568878
##     class counts:     0   123
##    probabilities: 0.000 1.000
```

1.18 Prediccion regresion tree

```
prediccionrandom <- predict(regression_tree,newdata = test)
```

1. 19 Metricas de evaluacion

```
precision <- sum(prediccionrandom == test$clasificacion_Caras) / nrow(test)
precision
```

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

Modelo random forest

1.20 Random forest

```
rf_model <- randomForest(clasificacion_Caras ~.,data = train)
```

1.21 Metricas Random Forest

```
predictions <- predict(rf_model, newdata = test)
accuracy <- mean(predictions == test$clasificacion_Caras)
accuracy
```

```
## [1] 0.9673591
```

Modelo naibe bayes

1.21 Naibe bayes

```
modelobayes <- naiveBayes(clasificacion_Caras ~ ., data = train)
```

1.22 Prediccion Naive bayes

```
prediccionBayes <- predict(modelobayes, newdata = test)
```

1.23 Metricas de Naive Bayes

```
confusionMatrix(prediccionBayes, test$clasificacion_Caras)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   0   1
##          0 153  52
```

```
##          1  29 103
##
##                Accuracy : 0.7596
##                  95% CI : (0.7104, 0.8043)
##     No Information Rate : 0.5401
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.5108
##
##  Mcnemar's Test P-Value : 0.01451
##
##             Sensitivity : 0.8407
##             Specificity : 0.6645
##          Pos Pred Value : 0.7463
##          Neg Pred Value : 0.7803
##              Prevalence : 0.5401
##          Detection Rate : 0.4540
##    Detection Prevalence : 0.6083
##       Balanced Accuracy : 0.7526
##
##        'Positive' Class : 0
##
```

Se puede mencionar que el accuracy del metodo de naibe bayes es de 0.76 aproximadamente muy similar al dato obtenido con la regresion logistica.

Comparación de los algoritmos

Comos se puede visualizar se realizaron 4 diferentes algoritmos, regresión logística, naive bayes, Random forest y árboles de decisión, como comparación se puede mencionar que los 4 algoritmos tomaron relativamente el mismo tiempo en ser procesados ninguno se sacó diferencia con el otro mientras que respecto con el tema de accuracy el que mejor lo hizo fue el modelo de árboles de decisión ya que este obtuvo una precisión perfecta de 1 mientras que el que peor lo hizo fue el modelo de naive bayes teniendo un accuracy de 0.7596 pero cabe la pena mencionar que este es muy similar al modelo de regresión logística ya que este obtuvo un accuracy de 0.7656 por lo que los do lo hicieron similarmente, se puede concluir que el mejor modelo es el modelo de árboles de decisión para este set de datos.