Informe Proyecto 2 entrega 5

Juan Luis Solórzano (carnet: 201598) Micaela Yataz (carnet: 18960)

2025-01-20

https://github.com/JusSolo/Mineria_Proyecto2.git

git: https://github.com/JusSolo/Mineria_Proyecto2.git

1. Cree una variable dicotómica por cada una de las categorías de la variable respuesta categórica que creó en hojas anteriores. Debería tener 3 variables dicotómicas (valores 0 y 1) una que diga si la vivienda es cara o no, media o no, económica o no.

```
y <- datos$SalePrice
set.seed(123)
trainI<- createDataPartition(y, p=0.7, list=FALSE)

train <- datos[trainI, ]
test <- datos[-trainI, ]
```

```
train$precio_categoria<-cut(train$SalePrice,
                            breaks = c(0, 129975, 214000,Inf),
                            labels = c("Economica", "Intermedia", "Cara" ),
                            include.lowest = TRUE)

test$precio_categoria<-cut(test$SalePrice,
                            breaks = c(0, 129975, 214000, Inf),
                            labels = c("Economica", "Intermedia", "Cara"),
                            include.lowest = TRUE)
```

```
train$casa_cara<-ifelse(train$precio_categoria=="Cara",1,0)
train$casa_intermedia<-ifelse(train$precio_categoria=="Intermedio",1,0)
train$casa_economica<-ifelse(train$precio_categoria=="Economica",1,0)

test$casa_cara<-ifelse(test$precio_categoria=="Cara",1,0)
test$casa_intermedia<-ifelse(test$precio_categoria=="Intermedio",1,0)
test$casa_economica<-ifelse(test$precio_categoria=="Economica",1,0)

head(train[, c("precio_categoria", "casa_cara", "casa_intermedia", "casa_economica")])
```

```
##   precio_categoria casa_cara casa_intermedia casa_economica
## 2       Intermedia         0               0              0
## 3             Cara         1               0              0
## 5             Cara         1               0              0
## 6       Intermedia         0               0              0
## 7             Cara         1               0              0
```

```
## 9        Economica         0               0              1
```

```
head(test[,c("precio_categoria", "casa_cara", "casa_intermedia", "casa_economica")])
```

```
##    precio_categoria casa_cara casa_intermedia casa_economica
## 1        Intermedia         0               0              0
## 4        Intermedia         0               0              0
## 8        Intermedia         0               0              0
## 11        Economica         0               0              1
## 13       Intermedia         0               0              0
## 15       Intermedia         0               0              0
```

2. Use los mismos conjuntos de entrenamiento y prueba que utilizó en las hojas anteriores.

```
x_train<-train[, !(names(train) %in% c("SalePrice", "precio_categoria"))]
y_train<-train$precio_categoria

x_test<- test[, !(names(test) %in% c("SalePrice", "precio_categoria"))]
y_test<-test$precio_categoria
```

3. Elabore un modelo de regresión logística para conocer si una vivienda es cara o no, utilizando el conjunto de entrenamiento y explique los resultados a los que llega. El experimento debe ser reproducible por lo que debe fijar que los conjuntos de entrenamiento y prueba sean los mismos siempre que se ejecute el código. Use validación cruzada.

```
ybin<-train$casa_cara
x_vars<-train[,vars_cuantitativas]

modelo<-cbind(casa_cara=ybin,x_vars)

modelo_logistico<-glm(casa_cara~.,data = modelo, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(modelo_logistico)
```

```
##
## Call:
## glm(formula = casa_cara ~ ., family = binomial, data = modelo)
##
## Coefficients: (2 not defined because of singularities)
##                 Estimate Std. Error    z value Pr(>|z|)
## (Intercept)   -1.503e+17  3.276e+09  -45862727   <2e-16 ***
## SalePrice      3.525e+10  6.116e+01  576354260   <2e-16 ***
## LotFrontage   -3.227e+11  6.447e+04   -5005476   <2e-16 ***
## LotArea        1.037e+10  2.492e+02   41608080   <2e-16 ***
## OverallQual    1.013e+14  2.893e+06   35000358   <2e-16 ***
## OverallCond    2.454e+14  2.347e+06  104559479   <2e-16 ***
## YearBuilt      1.062e+13  1.573e+05   67496953   <2e-16 ***
## YearRemodAdd  -8.657e+12  1.585e+05  -54612080   <2e-16 ***
## MasVnrArea    -6.234e+11  1.425e+04  -43744953   <2e-16 ***
## BsmtFinSF1    -1.948e+10  1.056e+04   -1845949   <2e-16 ***
```

```
## BsmtFinSF2     1.264e+11  1.529e+04    8264633   <2e-16 ***
## BsmtUnfSF     -2.351e+11  9.464e+03  -24844468   <2e-16 ***
## TotalBsmtSF           NA         NA         NA       NA
## X1stFlrSF      2.397e+11  1.356e+04   17681004   <2e-16 ***
## X2ndFlrSF     -4.174e+11  1.168e+04  -35736648   <2e-16 ***
## LowQualFinSF  -1.719e+12  4.317e+04  -39812975   <2e-16 ***
## GrLivArea             NA         NA         NA       NA
## BsmtFullBath  -1.859e+14  6.140e+06  -30277642   <2e-16 ***
## BsmtHalfBath  -1.169e+15  9.769e+06 -119708645   <2e-16 ***
## FullBath       4.813e+13  6.661e+06    7225961   <2e-16 ***
## HalfBath       3.736e+14  6.120e+06   61044777   <2e-16 ***
## BedroomAbvGr   2.669e+14  3.845e+06   69420939   <2e-16 ***
## KitchenAbvGr   7.046e+14  1.099e+07   64090519   <2e-16 ***
## TotRmsAbvGrd   3.851e+13  2.814e+06   13681916   <2e-16 ***
## Fireplaces    -5.985e+13  4.080e+06  -14670834   <2e-16 ***
## GarageYrBlt   -6.589e+12  1.620e+05  -40660702   <2e-16 ***
## GarageCars    -2.975e+14  6.531e+06  -45547994   <2e-16 ***
## GarageArea     8.984e+11  2.223e+04   40407755   <2e-16 ***
## WoodDeckSF     6.194e+11  1.803e+04   34353299   <2e-16 ***
## OpenPorchSF    7.420e+11  3.537e+04   20978756   <2e-16 ***
## EnclosedPorch  6.129e+10  4.058e+04    1510340   <2e-16 ***
## X3SsnPorch    -1.851e+11  8.273e+04   -2236985   <2e-16 ***
## ScreenPorch   -1.225e+11  4.057e+04   -3018847   <2e-16 ***
## PoolArea      -1.296e+12  5.525e+04  -23463211   <2e-16 ***
## MiscVal       -1.144e+11  3.662e+03  -31239829   <2e-16 ***
## MoSold         4.108e+13  7.880e+05   52130580   <2e-16 ***
## YrSold         7.356e+13  1.631e+06   45104099   <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: 1147.2  on 1023  degrees of freedom
## Residual deviance: 1081.3  on  989  degrees of freedom
## AIC: 1151.3
##
## Number of Fisher Scoring iterations: 16
```

Los coeficientes de las varibales son muy grandes lo que puede haber multicolinealidad. El AIC de este modelo es 1151.3

```r
control <- trainControl(method = "cv", number = 10)

modelo_clean <- modelo[complete.cases(modelo), ]

modelo_cv <- train(as.factor(casa_cara) ~ .,
                   data = modelo_clean,
                   method = "glm",
                   family = binomial(),
                   trControl = control,
                   metric = "Accuracy")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

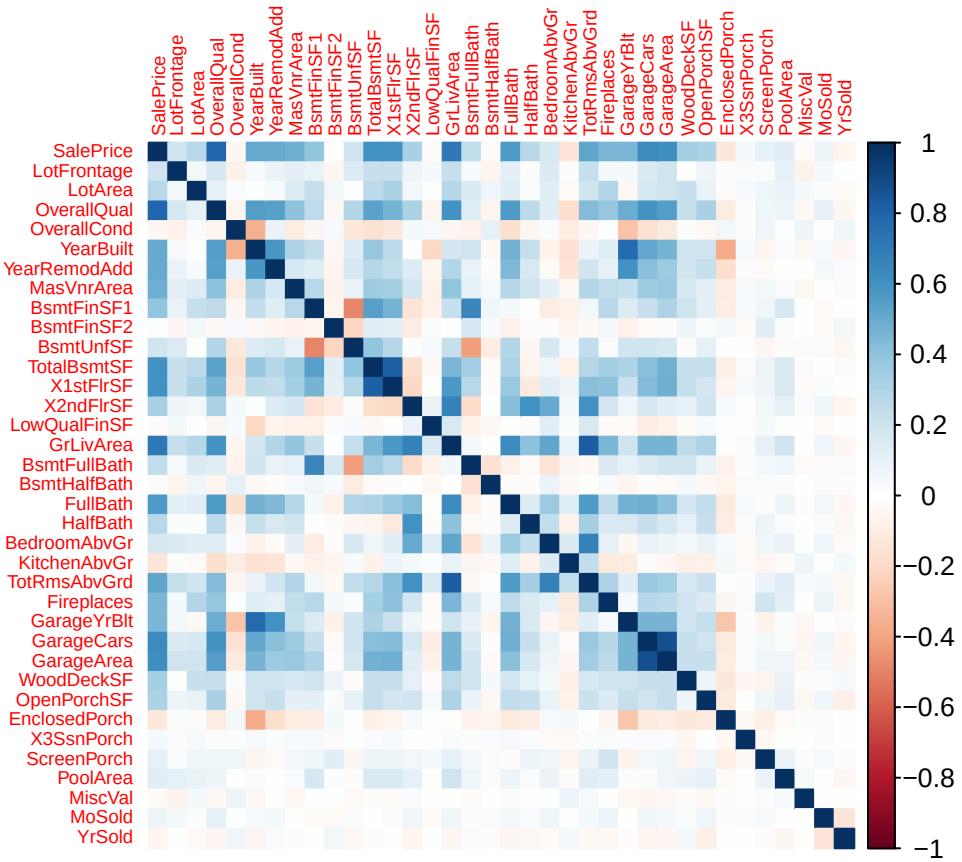
```
print(modelo_cv)
```

```
## Generalized Linear Model
##
## 1024 samples
##   36 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 922, 922, 922, 922, 921, 921, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9765277  0.9366528
```

Hay problema de multicolinealidad entre las variables predictorias.

4. Analice el modelo. Determine si hay multicolinealidad en las variables, y cuáles son las que aportan al modelo, por su valor de significación. Haga un análisis de correlación de las variables del modelo y especifique si el modelo se adapta bien a los datos.

```
correl<-cor(x_vars, use = "complete.obs")
corrplot(correl, method = "color", tl.cex = 0.6)
```



Las variables con correlacion fuerte son SalePrice, OveralQual y GrLivArea son posiblemente los predictores mas importantes del precio de las casas. Pero por la multicolinealidad, sera necesario seleccionar una variable.

5. Utilice el modelo con el conjunto de prueba y determine la eficiencia del algoritmo para clasificar.

```
x_test<-test[,vars_cuantitativas]
pred<-predict(modelo_logistico, newdata = x_test, type = "response")
prediccion<-ifelse(pred>=0.5,1,0)
confusionMatrix(factor(prediccion), factor(test$casa_cara), positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 325  16
##          1   3  92
##
##                Accuracy : 0.9564
##                  95% CI : (0.9328, 0.9736)
##     No Information Rate : 0.7523
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8782
##
##  Mcnemar's Test P-Value : 0.005905
##
```

```
##             Sensitivity : 0.8519
##             Specificity : 0.9909
##          Pos Pred Value : 0.9684
##          Neg Pred Value : 0.9531
##              Prevalence : 0.2477
##          Detection Rate : 0.2110
##    Detection Prevalence : 0.2179
##       Balanced Accuracy : 0.9214
##
##        'Positive' Class : 1
##
```

El modelo aunque parece que tiene buen rendimiento predictivo, es importante ver la multicolinealidad para obtener otro modelo mas estable.

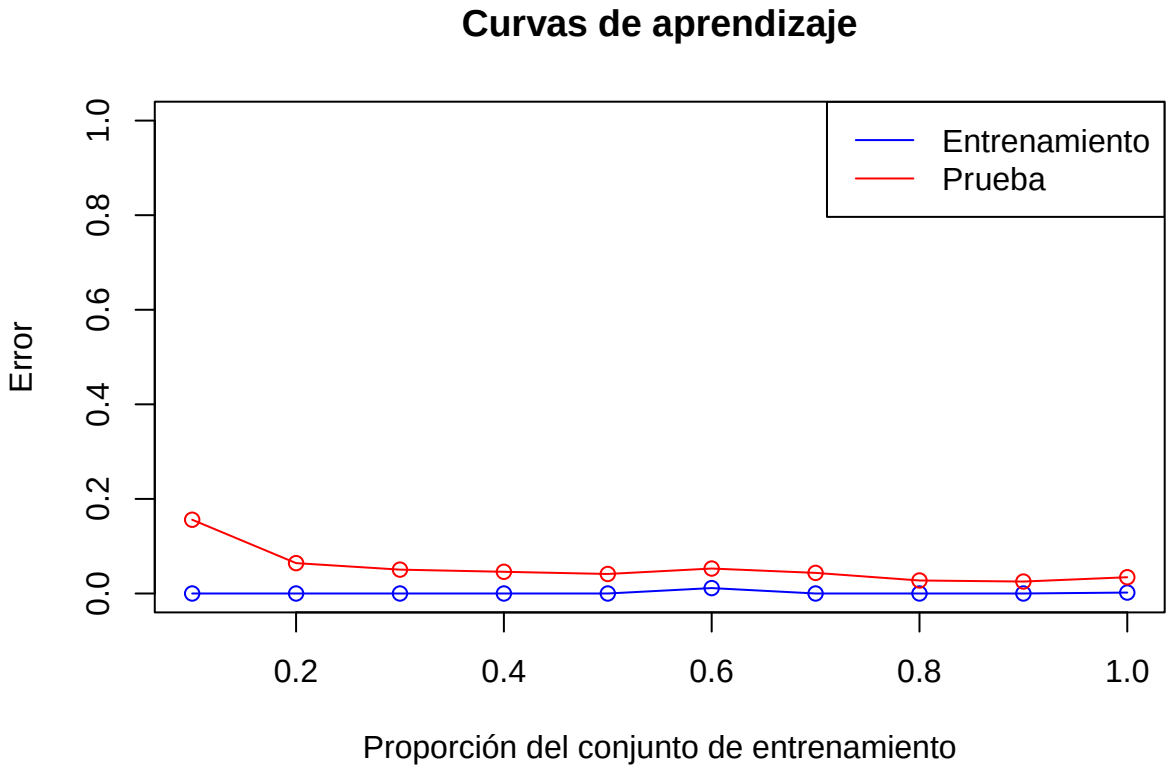
6. Explique si hay sobreajuste (overfitting) o no (recuerde usar para esto los errores del conjunto de prueba y de entrenamiento). Muestre las curvas de aprendizaje usando los errores de los conjuntos de entrenamiento y prueba

```r
library(Metrics)
tamaños<-seq(0.1,1,by=0.1)
errores_train<-c()
errores_test<-c()

for (t in tamaños) {
  idx <- sample(1:nrow(modelo), size = floor(t * nrow(modelo)))
  sub_train <- modelo[idx, ]
  sub_model <- glm(casa_cara ~ ., data = sub_train, family = binomial)

  pred_train <- ifelse(predict(sub_model, newdata = sub_train, type = "response") > 0.5, 1, 0)
  pred_test <- ifelse(predict(sub_model, newdata = x_test, type = "response") > 0.5, 1, 0)

  errores_train <- c(errores_train, mean(pred_train != sub_train$casa_cara))
  errores_test <- c(errores_test, mean(pred_test != test$casa_cara))
}
```

```
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# Graficar errores
plot(tamaños, errores_train, type = "o", col = "blue", ylim = c(0,1),
     ylab = "Error", xlab = "Proporción del conjunto de entrenamiento",
     main = "Curvas de aprendizaje")
lines(tamaños, errores_test, type = "o", col = "red")
legend("topright", legend = c("Entrenamiento", "Prueba"),
       col = c("blue", "red"), lty = 1)
```



Segun la grafica el modelo tienen muy poco error en el conjunto de entrenamiento y prueba, por lo que se ajusta bien.

7. Haga un tuneo del modelo para determinar los mejores parámetros, recuerde que los modelos de regresión logística se pueden regularizar como los de regresión lineal.

```r
y_train <- train$casa_cara
x_train <- train[, vars_cuantitativas]



# Paso 2: grid de búsqueda
```

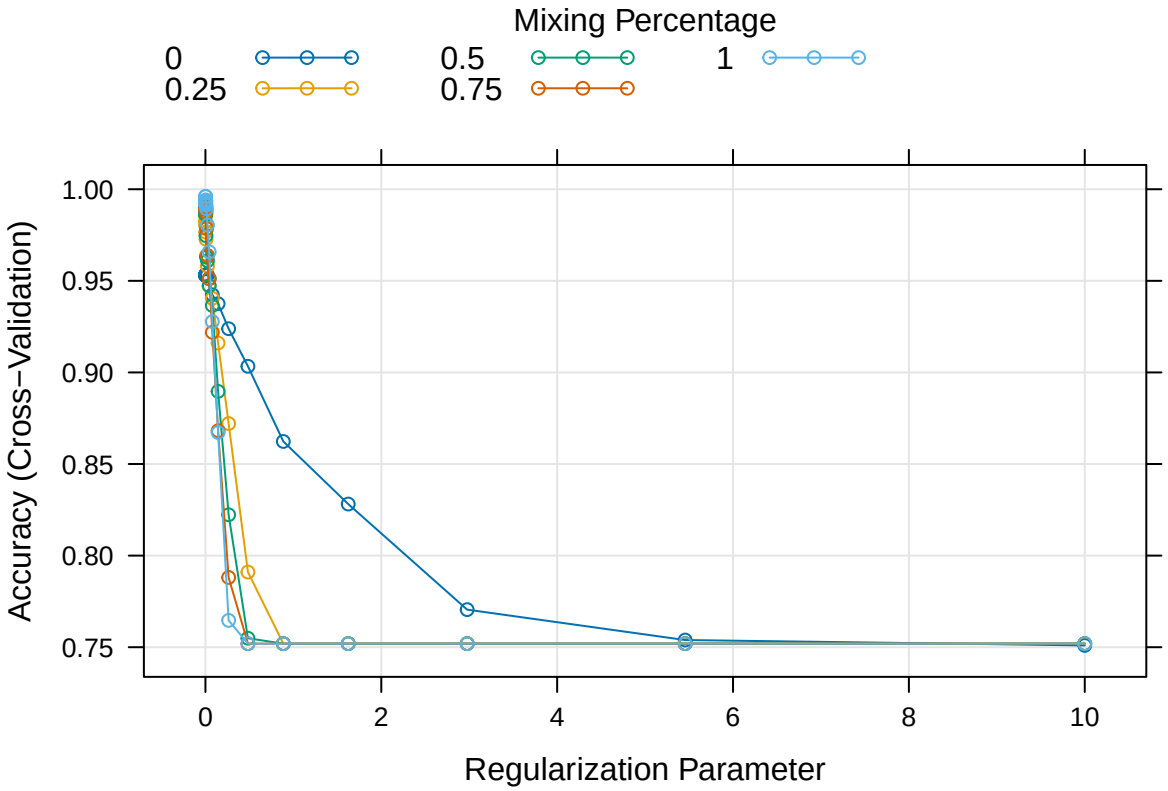
```r
grid <- expand.grid(
  alpha = seq(0, 1, length = 5),   # de Ridge (0) a Lasso (1)
  lambda = 10^seq(-4, 1, length = 20)  # penalización suave a fuerte
)

# Paso 3: control de entrenamiento
control <- trainControl(method = "cv", number = 5)

# Paso 4: entrenamiento con caret
set.seed(123)
modelo_tuneado <- train(
  x = as.matrix(x_train),
  y = as.factor(y_train),  # debe ser factor
  method = "glmnet",
  trControl = control,
  tuneGrid = grid,
  family = "binomial"
)

# Resultados
print(modelo_tuneado)
```

```
## glmnet
##
## 1024 samples
##   36 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 820, 819, 819, 819, 819
## Resampling results across tuning parameters:
##
##   alpha  lambda        Accuracy   Kappa
##   0.00   1.000000e-04  0.9531325   0.867563142
##   0.00   1.832981e-04  0.9531325   0.867563142
##   0.00   3.359818e-04  0.9531325   0.867563142
##   0.00   6.158482e-04  0.9531325   0.867563142
##   0.00   1.128838e-03  0.9531325   0.867563142
##   0.00   2.069138e-03  0.9531325   0.867563142
##   0.00   3.792690e-03  0.9531325   0.867563142
##   0.00   6.951928e-03  0.9531325   0.867563142
##   0.00   1.274275e-02  0.9531325   0.867563142
##   0.00   2.335721e-02  0.9531325   0.867563142
##   0.00   4.281332e-02  0.9511813   0.861498071
##   0.00   7.847600e-02  0.9423960   0.834913252
##   0.00   1.438450e-01  0.9375179   0.818102663
##   0.00   2.636651e-01  0.9238546   0.773202267
##   0.00   4.832930e-01  0.9033477   0.701645639
##   0.00   8.858668e-01  0.8623242   0.546003163
##   0.00   1.623777e+00  0.8281492   0.400665297
##   0.00   2.976351e+00  0.7705165   0.110028862
##   0.00   5.455595e+00  0.7539120   0.015602216
##   0.00   1.000000e+01  0.7509804  -0.001932184
```

```
##   0.25   1.000000e-04  0.9873123   0.965936742
##   0.25   1.832981e-04  0.9882879   0.968493110
##   0.25   3.359818e-04  0.9873075   0.965958211
##   0.25   6.158482e-04  0.9853515   0.961202109
##   0.25   1.128838e-03  0.9824247   0.953182663
##   0.25   2.069138e-03  0.9804782   0.947653047
##   0.25   3.792690e-03  0.9765710   0.936525520
##   0.25   6.951928e-03  0.9726686   0.924348673
##   0.25   1.274275e-02  0.9638785   0.899034269
##   0.25   2.335721e-02  0.9580297   0.880563199
##   0.25   4.281332e-02  0.9472836   0.848187366
##   0.25   7.847600e-02  0.9404448   0.827034886
##   0.25   1.438450e-01  0.9160450   0.746498697
##   0.25   2.636651e-01  0.8720947   0.583944716
##   0.25   4.832930e-01  0.7910282   0.221526136
##   0.25   8.858668e-01  0.7519560   0.000000000
##   0.25   1.623777e+00  0.7519560   0.000000000
##   0.25   2.976351e+00  0.7519560   0.000000000
##   0.25   5.455595e+00  0.7519560   0.000000000
##   0.25   1.000000e+01  0.7519560   0.000000000
##   0.50   1.000000e-04  0.9892635   0.971295986
##   0.50   1.832981e-04  0.9892635   0.971295986
##   0.50   3.359818e-04  0.9892635   0.971262167
##   0.50   6.158482e-04  0.9892587   0.971257407
##   0.50   1.128838e-03  0.9863319   0.963295934
##   0.50   2.069138e-03  0.9863319   0.963536218
##   0.50   3.792690e-03  0.9863319   0.962985848
##   0.50   6.951928e-03  0.9746150   0.929440489
##   0.50   1.274275e-02  0.9629029   0.895737540
##   0.50   2.335721e-02  0.9609469   0.889572766
##   0.50   4.281332e-02  0.9472836   0.848229672
##   0.50   7.847600e-02  0.9365375   0.813535921
##   0.50   1.438450e-01  0.8896557   0.651267241
##   0.50   2.636651e-01  0.8222956   0.374208828
##   0.50   4.832930e-01  0.7548876   0.017534400
##   0.50   8.858668e-01  0.7519560   0.000000000
##   0.50   1.623777e+00  0.7519560   0.000000000
##   0.50   2.976351e+00  0.7519560   0.000000000
##   0.50   5.455595e+00  0.7519560   0.000000000
##   0.50   1.000000e+01  0.7519560   0.000000000
##   0.75   1.000000e-04  0.9902391   0.973889117
##   0.75   1.832981e-04  0.9902439   0.974032486
##   0.75   3.359818e-04  0.9892635   0.971400908
##   0.75   6.158482e-04  0.9912147   0.976482437
##   0.75   1.128838e-03  0.9902391   0.973855074
##   0.75   2.069138e-03  0.9921903   0.979009082
##   0.75   3.792690e-03  0.9931707   0.981502334
##   0.75   6.951928e-03  0.9873123   0.964785415
##   0.75   1.274275e-02  0.9785127   0.940359749
##   0.75   2.335721e-02  0.9638737   0.897823131
##   0.75   4.281332e-02  0.9511860   0.859356537
##   0.75   7.847600e-02  0.9218938   0.765207726
##   0.75   1.438450e-01  0.8681923   0.568048333
##   0.75   2.636651e-01  0.7880870   0.202430852
```

```
##   0.75   4.832930e-01  0.7519560   0.000000000
##   0.75   8.858668e-01  0.7519560   0.000000000
##   0.75   1.623777e+00  0.7519560   0.000000000
##   0.75   2.976351e+00  0.7519560   0.000000000
##   0.75   5.455595e+00  0.7519560   0.000000000
##   0.75   1.000000e+01  0.7519560   0.000000000
##   1.00   1.000000e-04  0.9921903   0.979147822
##   1.00   1.832981e-04  0.9921903   0.979147822
##   1.00   3.359818e-04  0.9921903   0.979147822
##   1.00   6.158482e-04  0.9931659   0.981706273
##   1.00   1.128838e-03  0.9960976   0.989489866
##   1.00   2.069138e-03  0.9960976   0.989489866
##   1.00   3.792690e-03  0.9941463   0.984164611
##   1.00   6.951928e-03  0.9912195   0.975962578
##   1.00   1.274275e-02  0.9892635   0.970632937
##   1.00   2.335721e-02  0.9804639   0.945875689
##   1.00   4.281332e-02  0.9658202   0.903339055
##   1.00   7.847600e-02  0.9277475   0.784305023
##   1.00   1.438450e-01  0.8672119   0.564422466
##   1.00   2.636651e-01  0.7646437   0.072190063
##   1.00   4.832930e-01  0.7519560   0.000000000
##   1.00   8.858668e-01  0.7519560   0.000000000
##   1.00   1.623777e+00  0.7519560   0.000000000
##   1.00   2.976351e+00  0.7519560   0.000000000
##   1.00   5.455595e+00  0.7519560   0.000000000
##   1.00   1.000000e+01  0.7519560   0.000000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 1 and lambda = 0.002069138.
```

```
plot(modelo_tuneado)
```



8. Haga un análisis de la eficiencia del algoritmo usando una matriz de confusión. Tenga en cuenta la efectividad, donde el algoritmo se equivocó más, donde se equivocó menos y la importancia que tienen los errores, el tiempo y la memoria consumida. Para esto último puede usar "profvis" si trabaja con R y "cProfile" en Python.

```r
# 1. Preparar datos de test
y_test <- test$casa_cara
x_test <- test[, vars_cuantitativas]

# Asegurar mismos niveles que en entrenamiento (usando niveles del modelo)
y_test <- factor(y_test, levels = levels(modelo_tuneado$trainingData$.outcome))

# 2. Predecir en test
predicciones <- predict(modelo_tuneado,
                        newdata = as.matrix(x_test),
                        type = "raw")

# 3. Matriz de confusión (especificar clase positiva como "1")
confusionMatrix(data = predicciones,
                reference = y_test,
                positive = "1") # <- ¡Clase positiva es "1", no "TRUE"!
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 328   3
```

```
##          1   0 105
##
##                Accuracy : 0.9931
##                  95% CI : (0.98, 0.9986)
##     No Information Rate : 0.7523
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9814
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 0.9722
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9909
##              Prevalence : 0.2477
##          Detection Rate : 0.2408
##    Detection Prevalence : 0.2408
##       Balanced Accuracy : 0.9861
##
##        'Positive' Class : 1
##
```

Podemos notar que el modelo es muy bueno, no clasifica ninguna cara como no cara y dentro las caras solo clasifica 3 como no caras.

9. Determine cual de todos los modelos es mejor, puede usar AIC y BIC para esto, además de los parámetros de la matriz de confusión y los del profiler.

```
#
```

Podemos notar que comparado con en primer modelo en modelo tuneado tiene un mejor desempeño, con un acurency del 0.9931 vs uno del 0.9564

10. Haga un modelo de regresión logística para la variable categórica para el precio de las casas (categorías: barata, media y cara). Asegúrese de tunearlo para obtener el mejor modelo posible.

```
# Variables predictoras y respuesta
x_train <- train[, vars_cuantitativas]
y_train <- train$precio_categoria  # factor con 3 niveles

# Confirmar niveles
levels(y_train)
```

```
## [1] "Economica"  "Intermedia" "Cara"
```

```
# Esperado: "casa_economica" "casa_intermedia" "casa_cara"


# Grid para tuning
grid <- expand.grid(
  alpha = seq(0, 1, length.out = 5),        # Ridge (0) a Lasso (1)
  lambda = 10^seq(-4, 1, length.out = 20)   # Penalización suave a fuerte
```

```
)

set.seed(123)
modelo_multiclase <- train(
  x = as.matrix(x_train),
  y = y_train,  # Ya es factor con 3 clases
  method = "glmnet",
  family = "multinomial",  # <--- necesario para multiclase
  trControl = control,
  tuneGrid = grid
)
```

```
## Warning: from glmnet C++ code (error code -98); Convergence for 98th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
## Warning: from glmnet C++ code (error code -98); Convergence for 98th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
## Warning: from glmnet C++ code (error code -97); Convergence for 97th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
# Preparar datos de prueba
x_test <- test[, vars_cuantitativas]
y_test <- test$precio_categoria
y_test <- factor(y_test, levels = levels(modelo_multiclase$trainingData$.outcome))

# Predecir
pred_multiclase <- predict(modelo_multiclase, newdata = as.matrix(x_test))

# Evaluar con matriz de confusión
confusionMatrix(pred_multiclase, y_test)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara
##   Economica        105          2    0
##   Intermedia         4        217    5
##   Cara               0          0  103
##
## Overall Statistics
##
##                Accuracy : 0.9748
##                  95% CI : (0.9553, 0.9873)
##     No Information Rate : 0.5023
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9593
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: Economica Class: Intermedia Class: Cara
## Sensitivity                    0.9633            0.9909      0.9537
## Specificity                    0.9939            0.9585      1.0000
## Pos Pred Value                 0.9813            0.9602      1.0000
## Neg Pred Value                 0.9878            0.9905      0.9850
## Prevalence                     0.2500            0.5023      0.2477
## Detection Rate                 0.2408            0.4977      0.2362
## Detection Prevalence           0.2454            0.5183      0.2362
## Balanced Accuracy              0.9786            0.9747      0.9769
```

11. Compare la eficiencia del modelo anterior con los de clasificación de las entregas anteriores ¿Cuál se demoró más en procesar?¿Cuál se equivocó más?¿Cuál se equivocó menos?¿por qué?

Arbol

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara Lujo
##   Economica         49          9    0    0
##   Intermedia        10        103   15    0
##   Cara               0          4   40    0
##   Lujo               0          0    0    4
##
## Overall Statistics
##
##                Accuracy : 0.8376
##                  95% CI : (0.784, 0.8824)
##     No Information Rate : 0.4957
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7389
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara Class: Lujo
## Sensitivity                    0.8305            0.8879      0.7273     1.00000
## Specificity                    0.9486            0.7881      0.9777     1.00000
## Pos Pred Value                 0.8448            0.8047      0.9091     1.00000
## Neg Pred Value                 0.9432            0.8774      0.9211     1.00000
## Prevalence                     0.2521            0.4957      0.2350     0.01709
## Detection Rate                 0.2094            0.4402      0.1709     0.01709
## Detection Prevalence           0.2479            0.5470      0.1880     0.01709
## Balanced Accuracy              0.8895            0.8380      0.8525     1.00000
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara Lujo
##   Economica        109        115    3    0
##   Intermedia         0         18    5    0
##   Cara               0         85   72    0
##   Lujo               0          1   19    9
##
## Overall Statistics
##
##                Accuracy : 0.4771
##                  95% CI : (0.4293, 0.5251)
##     No Information Rate : 0.5023
##     P-Value [Acc > NIR] : 0.8647
##
##                   Kappa : 0.3121
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara
##   Economica         68         10    0
##   Intermedia        41        186   33
##   Cara               0         23   75
##
## Overall Statistics
##
##                Accuracy : 0.7546
##                  95% CI : (0.7114, 0.7943)
##     No Information Rate : 0.5023
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.591
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara
## Sensitivity                    0.6239            0.8493      0.6944
## Specificity                    0.9694            0.6590      0.9299
## Pos Pred Value                 0.8718            0.7154      0.7653
## Neg Pred Value                 0.8855            0.8125      0.9024
## Prevalence                     0.2500            0.5023      0.2477
## Detection Rate                 0.1560            0.4266      0.1720
## Detection Prevalence           0.1789            0.5963      0.2248
## Balanced Accuracy              0.7966            0.7542      0.8122
```

KNN

El mejor de los modelos es: La regresión lógistica.