Entrega 6

Juan Luis Solórzano

2025-04-24

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

1. Exploración de los datos

Observaciones: - La variable respuesta `precio_categoria` está balanceada en torno a 25–50% por clase. - Existen variables numéricas que requieren centrado y escalado, y posibles valores faltantes.

2. Preparación de los datos

Para SVM es crucial que las variables numéricas estén normalizadas y no existan NA. Además, convertiremos factores a dummies.

```
# 1) Eliminación de predictores de varianza casi cero
nzv <- nearZeroVar(train, saveMetrics = TRUE)
train <- train[, !nzv$zeroVar]
test  <- test[, colnames(test) %in% colnames(train)]

# 2) Imputación de valores faltantes + centrado y escalado
pp <- preProcess(train %>% select(-SalePrice, -precio_categoria),
                 method = c("center","scale","knnImpute"))
train_pp <- predict(pp, train)
test_pp  <- predict(pp, test)

# 3) Codificación de factores en dummies para predictores categóricos
dummies <- dummyVars(~ ., data = train_pp %>% select(-SalePrice, -precio_categoria))
train_x <- predict(dummies, newdata = train_pp)
test_x  <- predict(dummies, newdata = test_pp)

# Preparamos data para caret
x_train <- as.data.frame(train_x)
y_train <- train_pp$precio_categoria
x_test  <- as.data.frame(test_x)
y_test  <- test_pp$precio_categoria
```

3. Definición de control de entrenamiento

```
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 3,
                     classProbs = TRUE, summaryFunction = multiClassSummary)
```

4. Modelos SVM

4.1 SVM Lineal

```
grid_lin <- expand.grid(C = c(0.1, 1, 10))
svm_lin <- train(x_train, y_train, method = "svmLinear",
                 trControl = ctrl, tuneGrid = grid_lin)
svm_lin
```

```
## Support Vector Machines with Linear Kernel
##
## 1024 samples
##   35 predictor
##    3 classes: 'Economica', 'Intermedia', 'Cara'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 818, 821, 819, 819, 819, 819, ...
## Resampling results across tuning parameters:
##
##   C     logLoss    AUC        prAUC      Accuracy   Kappa      Mean_F1
##    0.1  0.4274141  0.9482423  0.8884803  0.8466445  0.7509514  0.8443582
##    1.0  0.4455822  0.9473231  0.8864311  0.8414285  0.7408239  0.8377079
##   10.0  0.4664172  0.9461297  0.8844024  0.8381939  0.7332534  0.8329701
##   Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value  Mean_Neg_Pred_Value
##   0.8362863         0.9114446         0.8561436            0.9164732
##   0.8262286         0.9069208         0.8555335            0.9149067
##   0.8163159         0.9027783         0.8614337            0.9150557
##   Mean_Precision  Mean_Recall  Mean_Detection_Rate  Mean_Balanced_Accuracy
##   0.8561436       0.8362863    0.2822148            0.8738654
##   0.8555335       0.8262286    0.2804762            0.8665747
##   0.8614337       0.8163159    0.2793980            0.8595471
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.1.
```

4.2 SVM Radial (RBF)

```
grid_rad <- expand.grid(sigma = c(0.001, 0.01, 0.1), C = c(0.1, 1, 10))
svm_rad <- train(x_train, y_train, method = "svmRadial",
                 trControl = ctrl, tuneGrid = grid_rad)
svm_rad
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1024 samples
##   35 predictor
##    3 classes: 'Economica', 'Intermedia', 'Cara'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 819, 819, 819, 820, 819, 819, ...
## Resampling results across tuning parameters:
##
##   sigma  C     logLoss    AUC        prAUC      Accuracy   Kappa      Mean_F1
```

```
##   0.001   0.1  0.5546005  0.9402993  0.8787211  0.7796430  0.6634994  0.7852081
##   0.001   1.0  0.4338778  0.9430833  0.8836443  0.8196625  0.7116493  0.8198889
##   0.001  10.0  0.4050926  0.9492930  0.8937655  0.8385261  0.7396542  0.8373561
##   0.010   0.1  0.4572172  0.9423134  0.8866822  0.8001279  0.6876875  0.8039683
##   0.010   1.0  0.3814616  0.9512624  0.9011152  0.8346268  0.7336363  0.8332279
##   0.010  10.0  0.3704936  0.9548218  0.9073267  0.8433835  0.7464197  0.8412003
##   0.100   0.1  0.5861882  0.9224454  0.8444093  0.7770254  0.6359824  0.7726574
##   0.100   1.0  0.4640849  0.9365448  0.8720024  0.8115020  0.6877312  0.8043564
##   0.100  10.0  0.4908162  0.9307737  0.8606447  0.7994583  0.6671215  0.7913346
##   Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value  Mean_Neg_Pred_Value
##   0.8189915         0.8910883         0.7752766            0.8845414
##   0.8211449         0.9003409         0.8217873            0.9001308
##   0.8333548         0.9086425         0.8445294            0.9113104
##   0.8206834         0.8958344         0.7973943            0.8908246
##   0.8305185         0.9068425         0.8391678            0.9090471
##   0.8349799         0.9102960         0.8507995            0.9144289
##   0.7615673         0.8707949         0.7898349            0.8774352
##   0.7848582         0.8863123         0.8370010            0.9004611
##   0.7707599         0.8789249         0.8268624            0.8939269
##   Mean_Precision  Mean_Recall  Mean_Detection_Rate  Mean_Balanced_Accuracy
##   0.7752766       0.8189915    0.2598810            0.8550399
##   0.8217873       0.8211449    0.2732208            0.8607429
##   0.8445294       0.8333548    0.2795087            0.8709986
##   0.7973943       0.8206834    0.2667093            0.8582589
##   0.8391678       0.8305185    0.2782089            0.8686805
##   0.8507995       0.8349799    0.2811278            0.8726379
##   0.7898349       0.7615673    0.2590085            0.8161811
##   0.8370010       0.7848582    0.2705007            0.8355852
##   0.8268624       0.7707599    0.2664861            0.8248424
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 10.
```

4.3 SVM Polinomial

```
grid_poly <- expand.grid(degree = c(2, 3, 4), scale = c(0.001, 0.01), C = c(0.1, 1, 10))
svm_poly <- train(x_train, y_train, method = "svmPoly",
                  trControl = ctrl, tuneGrid = grid_poly)
svm_poly
```

```
## Support Vector Machines with Polynomial Kernel
##
## 1024 samples
##   35 predictor
##    3 classes: 'Economica', 'Intermedia', 'Cara'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 818, 820, 819, 819, 820, 820, ...
## Resampling results across tuning parameters:
##
##   degree  scale  C     logLoss    AUC        prAUC      Accuracy   Kappa
##   2       0.001   0.1  0.5547119  0.9407783  0.8793010  0.7880650  0.6765371
##   2       0.001   1.0  0.4488754  0.9431750  0.8835717  0.8267821  0.7224443
```

```
##   2       0.001  10.0  0.4308580  0.9487165  0.8896987  0.8437230  0.7480515
##   2       0.010   0.1  0.4534784  0.9432156  0.8831891  0.8284177  0.7240475
##   2       0.010   1.0  0.4310074  0.9503482  0.8935556  0.8486059  0.7552878
##   2       0.010  10.0  0.4185081  0.9533077  0.8994385  0.8577180  0.7689501
##   3       0.001   0.1  0.5554992  0.9407919  0.8790649  0.7851318  0.6721808
##   3       0.001   1.0  0.4469091  0.9439705  0.8840543  0.8254972  0.7178635
##   3       0.001  10.0  0.4300125  0.9497345  0.8914850  0.8486059  0.7554221
##   3       0.010   0.1  0.4568495  0.9461173  0.8863397  0.8310272  0.7264445
##   3       0.010   1.0  0.4245633  0.9532113  0.8983660  0.8518595  0.7600920
##   3       0.010  10.0  0.4399155  0.9497044  0.8927726  0.8492547  0.7539326
##   4       0.001   0.1  0.5477613  0.9409291  0.8794674  0.7919627  0.6812866
##   4       0.001   1.0  0.4474038  0.9444859  0.8847576  0.8287381  0.7233971
##   4       0.001  10.0  0.4221668  0.9508839  0.8937518  0.8547801  0.7653234
##   4       0.010   0.1  0.4555103  0.9476547  0.8880317  0.8355770  0.7339521
##   4       0.010   1.0  0.4358625  0.9521212  0.8951291  0.8505571  0.7569469
##   4       0.010  10.0  0.4419496  0.9460407  0.8876642  0.8362179  0.7317523
##   Mean_F1    Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value
##   0.7942952  0.8282762         0.8959976         0.7851914
##   0.8270452  0.8267359         0.9037241         0.8301292
##   0.8426950  0.8384627         0.9115382         0.8491390
##   0.8280667  0.8254009         0.9036446         0.8335407
##   0.8472204  0.8410499         0.9134910         0.8555946
##   0.8556017  0.8471176         0.9174673         0.8675905
##   0.7914854  0.8258732         0.8944699         0.7823350
##   0.8240811  0.8180291         0.9005955         0.8332190
##   0.8472374  0.8418896         0.9137749         0.8550174
##   0.8292080  0.8223481         0.9033409         0.8394880
##   0.8503491  0.8431882         0.9149365         0.8602862
##   0.8463913  0.8344297         0.9112775         0.8633767
##   0.7982165  0.8290772         0.8969853         0.7891983
##   0.8271575  0.8221301         0.9027566         0.8350181
##   0.8533524  0.8477533         0.9170992         0.8612136
##   0.8337466  0.8275622         0.9060859         0.8428419
##   0.8481590  0.8384009         0.9130352         0.8624420
##   0.8327719  0.8193941         0.9032750         0.8530852
##   Mean_Neg_Pred_Value  Mean_Precision  Mean_Recall  Mean_Detection_Rate
##   0.8888253            0.7851914       0.8282762    0.2626883
##   0.9039080            0.8301292       0.8267359    0.2755940
##   0.9139269            0.8491390       0.8384627    0.2812410
##   0.9050101            0.8335407       0.8254009    0.2761392
##   0.9169902            0.8555946       0.8410499    0.2828686
##   0.9228799            0.8675905       0.8471176    0.2859060
##   0.8874352            0.7823350       0.8258732    0.2617106
##   0.9039257            0.8332190       0.8180291    0.2751657
##   0.9168844            0.8550174       0.8418896    0.2828686
##   0.9073925            0.8394880       0.8223481    0.2770091
##   0.9189511            0.8602862       0.8431882    0.2839532
##   0.9189884            0.8633767       0.8344297    0.2830849
##   0.8898935            0.7891983       0.8290772    0.2639876
##   0.9057492            0.8350181       0.8221301    0.2762460
##   0.9204674            0.8612136       0.8477533    0.2849267
##   0.9097172            0.8428419       0.8275622    0.2785257
##   0.9193086            0.8624420       0.8384009    0.2835190
##   0.9119312            0.8530852       0.8193941    0.2787393
```

```
##   Mean_Balanced_Accuracy
##   0.8621369
##   0.8652300
##   0.8750005
##   0.8645227
##   0.8772704
##   0.8822925
##   0.8601715
##   0.8593123
##   0.8778323
##   0.8628445
##   0.8790623
##   0.8728536
##   0.8630312
##   0.8624433
##   0.8824263
##   0.8668240
##   0.8757180
##   0.8613346
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 2, scale = 0.01 and C = 10.
```

5. Predicción y matrices de confusión

```
# Mejor modelo según Accuracy (ejemplo: svm_rad)
best <- svm_rad
pred_test <- predict(best, x_test)
confusionMatrix(pred_test, y_test)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara
##   Economica         86         16    0
##   Intermedia        23        190   18
##   Cara               0         13   90
##
## Overall Statistics
##
##                Accuracy : 0.8394
##                  95% CI : (0.8016, 0.8727)
##     No Information Rate : 0.5023
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7397
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara
## Sensitivity                    0.7890            0.8676      0.8333
## Specificity                    0.9511            0.8111      0.9604
```

```
## Pos Pred Value                 0.8431           0.8225      0.8738
## Neg Pred Value                 0.9311           0.8585      0.9459
## Prevalence                     0.2500           0.5023      0.2477
## Detection Rate                 0.1972           0.4358      0.2064
## Detection Prevalence           0.2339           0.5298      0.2362
## Balanced Accuracy              0.8700           0.8393      0.8968
```

```r
# Confusion matrices para cada modelo
a_list <- list(
  Linear = svm_lin,
  Radial = svm_rad,
  Poly   = svm_poly
)
for(name in names(a_list)){
  cat("\nModelo:", name, "\n")
  print(confusionMatrix(predict(a_list[[name]], x_test), y_test))
}
```

```
##
## Modelo: Linear
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara
##   Economica         88         17    0
##   Intermedia        21        191   24
##   Cara               0         11   84
##
## Overall Statistics
##
##                Accuracy : 0.8326
##                  95% CI : (0.7941, 0.8664)
##     No Information Rate : 0.5023
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7273
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara
## Sensitivity                    0.8073           0.8721      0.7778
## Specificity                    0.9480           0.7926      0.9665
## Pos Pred Value                 0.8381           0.8093      0.8842
## Neg Pred Value                 0.9366           0.8600      0.9296
## Prevalence                     0.2500           0.5023      0.2477
## Detection Rate                 0.2018           0.4381      0.1927
## Detection Prevalence           0.2408           0.5413      0.2179
## Balanced Accuracy              0.8777           0.8324      0.8721
##
## Modelo: Radial
## Confusion Matrix and Statistics
##
##             Reference
```

```
## Prediction   Economica Intermedia Cara
##   Economica         86         16    0
##   Intermedia        23        190   18
##   Cara               0         13   90
##
## Overall Statistics
##
##                Accuracy : 0.8394
##                  95% CI : (0.8016, 0.8727)
##     No Information Rate : 0.5023
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7397
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara
## Sensitivity                    0.7890            0.8676      0.8333
## Specificity                    0.9511            0.8111      0.9604
## Pos Pred Value                 0.8431            0.8225      0.8738
## Neg Pred Value                 0.9311            0.8585      0.9459
## Prevalence                     0.2500            0.5023      0.2477
## Detection Rate                 0.1972            0.4358      0.2064
## Detection Prevalence           0.2339            0.5298      0.2362
## Balanced Accuracy              0.8700            0.8393      0.8968
##
## Modelo: Poly
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara
##   Economica         86         17    0
##   Intermedia        23        191   22
##   Cara               0         11   86
##
## Overall Statistics
##
##                Accuracy : 0.8326
##                  95% CI : (0.7941, 0.8664)
##     No Information Rate : 0.5023
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7273
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara
## Sensitivity                    0.7890            0.8721      0.7963
## Specificity                    0.9480            0.7926      0.9665
## Pos Pred Value                 0.8350            0.8093      0.8866
```

```
## Neg Pred Value              0.9309          0.8600      0.9351
## Prevalence                  0.2500          0.5023      0.2477
## Detection Rate              0.1972          0.4381      0.1972
## Detection Prevalence        0.2362          0.5413      0.2225
## Balanced Accuracy           0.8685          0.8324      0.8814
```