Lab7

Javier Mombiela, Jose Hernandez, Pablo Gonzalez

2023-03-10

Lab 7 Modelo de maquinas Vectorial de Soporte

1 Division de variables en datos numericos

1.1 Transformacion de la data

Al momento de analizar los datos se pudieron encontrar que muchos de los datos estan en diferentes escalas y tambien que varias columnas cuentan con datos faltantes, ademas se puede concluir por hojas anteriores que ninguno de los datos estas normalziados.

```
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))
```

2. 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)
```

3. Cojuntis de train y test

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

4. Generar mas de dos modelos de SVM con diferentes kernels y distintos valores en los parametros c y en d

```
modeloSVM_1<-svm(clasificacion~., data=train, cost=2^-5, kernel="linear")
modeloSVM_2<-svm(clasificacion~., data=train, cost=0.5, kernel="linear")
modeloSVM_3<-svm(clasificacion~., data=train, gamma=2^-5, kernel="radial")
```

5. Generar las diferentes predicciones de los modelos.

5.1 Prediccion primer modelo

```
prediccion1<-predict(modeloSVM_1,newdata=test)
```

5.2 Prediccion segundo modelo

```
prediccion2<-predict(modeloSVM_2,newdata=test)
```

5.3 Prediccion tercer modelo

```
prediccion3<-predict(modeloSVM_3,newdata=test)
```

6. Matrices de confusion de cada modelo

6.1 Matriz de confusion primer modelo

```
confusionMatrix(test$clasificacion,prediccion1)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Caras Economicas Intermedias
##   Caras         119         23          13
##   Economicas      9         97           2
##   Intermedias    45          6          23
##
## Overall Statistics
##
##                Accuracy : 0.7092
##                  95% CI : (0.6575, 0.7571)
##     No Information Rate : 0.5134
##     P-Value [Acc > NIR] : 1.955e-13
##
##                   Kappa : 0.5304
```

```
##
##  Mcnemar's Test P-Value : 1.060e-05
##
## Statistics by Class:
##
##                      Class: Caras Class: Economicas Class: Intermedias
## Sensitivity                0.6879            0.7698            0.60526
## Specificity                0.7805            0.9479            0.82943
## Pos Pred Value             0.7677            0.8981            0.31081
## Neg Pred Value             0.7033            0.8734            0.94297
## Prevalence                 0.5134            0.3739            0.11276
## Detection Rate             0.3531            0.2878            0.06825
## Detection Prevalence       0.4599            0.3205            0.21958
## Balanced Accuracy          0.7342            0.8589            0.71735
```

6.2 Matriz de confusion segundo modelo

```
confusionMatrix(test$clasificacion,prediccion2)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Caras Economicas Intermedias
##   Caras         123         23           9
##   Economicas      5        103           0
##   Intermedias    37          0          37
##
## Overall Statistics
##
##                Accuracy : 0.7804
##                  95% CI : (0.7324, 0.8235)
##     No Information Rate : 0.4896
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6487
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Caras Class: Economicas Class: Intermedias
## Sensitivity                0.7455            0.8175            0.8043
## Specificity                0.8140            0.9763            0.8729
## Pos Pred Value             0.7935            0.9537            0.5000
## Neg Pred Value             0.7692            0.8996            0.9658
## Prevalence                 0.4896            0.3739            0.1365
## Detection Rate             0.3650            0.3056            0.1098
## Detection Prevalence       0.4599            0.3205            0.2196
## Balanced Accuracy          0.7797            0.8969            0.8386
```

6.3 Matriz de confusion tecer modelo

```
confusionMatrix(test$clasificacion,prediccion3)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Caras Economicas Intermedias
##   Caras         122         22          11
##   Economicas     12         94           2
##   Intermedias    35          6          33
##
## Overall Statistics
##
##                Accuracy : 0.7389
##                  95% CI : (0.6885, 0.785)
##     No Information Rate : 0.5015
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5811
##
##  Mcnemar's Test P-Value : 0.0005675
##
## Statistics by Class:
##
##                      Class: Caras Class: Economicas Class: Intermedias
## Sensitivity                0.7219            0.7705            0.71739
## Specificity                0.8036            0.9349            0.85911
## Pos Pred Value             0.7871            0.8704            0.44595
## Neg Pred Value             0.7418            0.8777            0.95057
## Prevalence                 0.5015            0.3620            0.13650
## Detection Rate             0.3620            0.2789            0.09792
## Detection Prevalence       0.4599            0.3205            0.21958
## Balanced Accuracy          0.7627            0.8527            0.78825
```

7. Analisis de sobreajustamiento

7.1 Analisis primer modelo

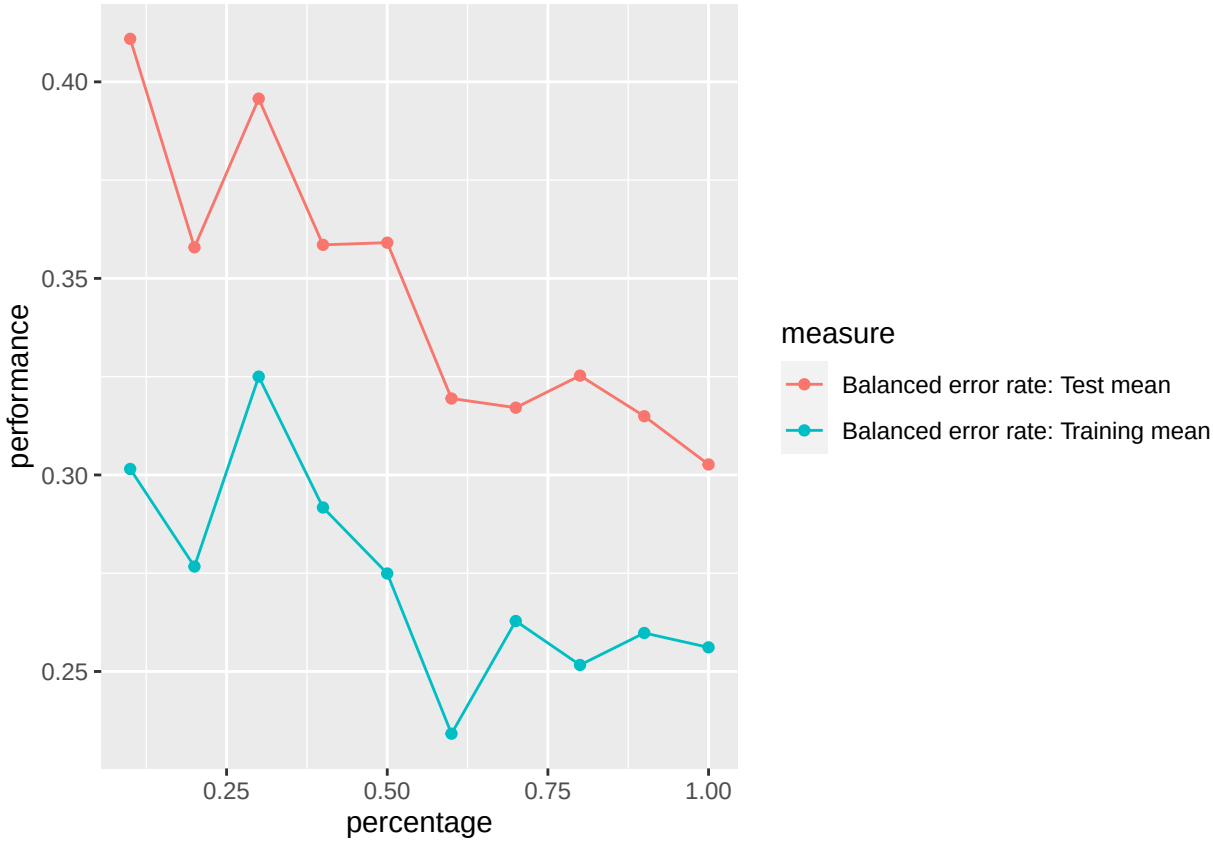
```
    datos.task = makeClassifTask(data = train, target = "clasificacion")
    rin2 = makeResampleDesc(method = "CV", iters = 10, predict = "both")
    lrn = makeLearner("classif.svm", kernel = "linear", cost = 2^-5, type = "C-classification")
    lc2 = generateLearningCurveData(learners = lrn, task = datos.task,
                                    percs = seq(0.1, 1, by = 0.1),
                                    measures = list(ber, setAggregation(ber, train.mean)), resampling =
                                    show.info = FALSE)
```

```
## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.
```

```
## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'BsmtHalfBath' and 'X3SsnPorch' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'KitchenAbvGr' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' and 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'X3SsnPorch' and 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' and 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' and 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' and 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.
```

```
## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.
```

```
plotLearningCurve(lc2, facet = "learner")
```



Al momento de realizar la grafica se puede visualizar que la curva de training presenta una tendencia al decenso a pesar de la cantidad de datos que existe, por lo que se puede concluir que este modelo sufre de overfitting. Una de las posibles soluciones que se puede dar es reducir la C para que el modelo acpete mayor cantidad de datos y pueda trabajar mejor.

7.2 Analisis segundo modelo

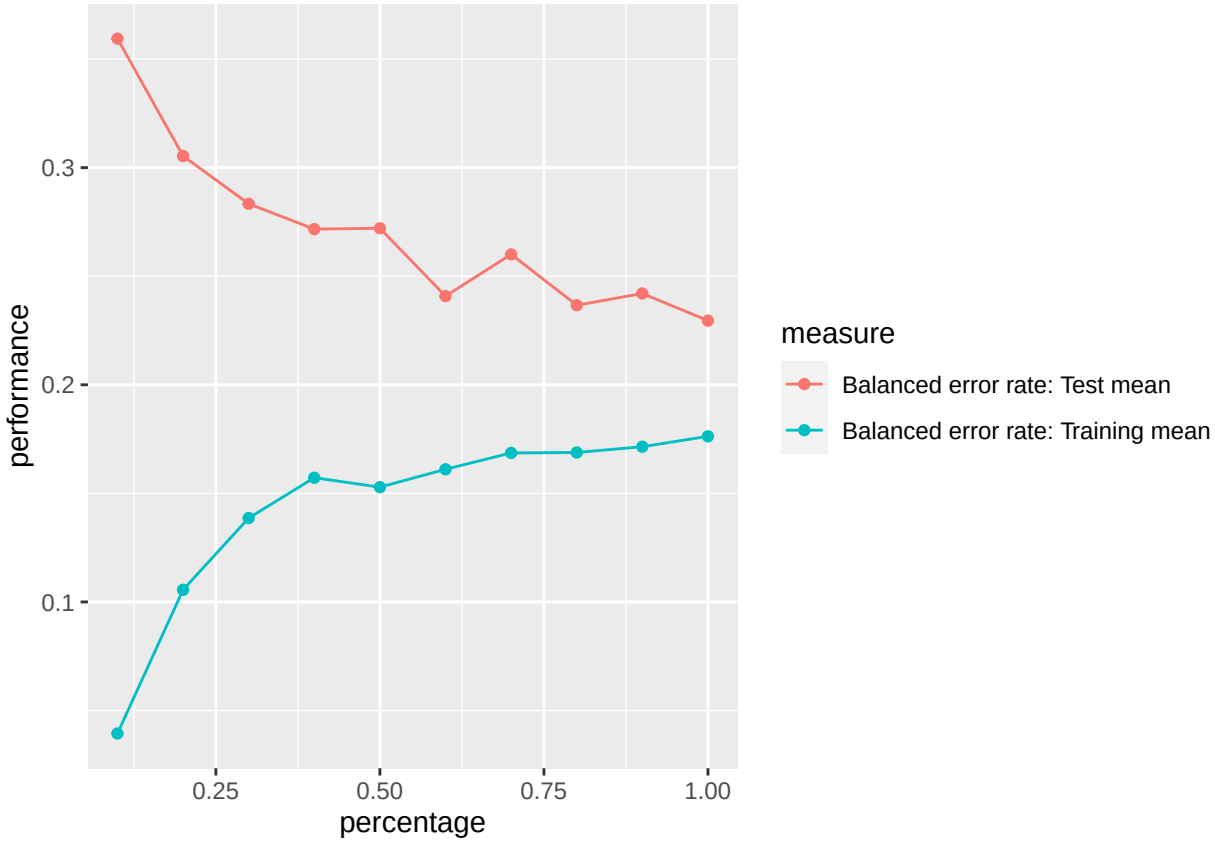
```
datos.task = makeClassifTask(data = train, target = "clasificacion")
rin2 = makeResampleDesc(method = "CV", iters = 10, predict = "both")
lrn = makeLearner("classif.svm", kernel = "linear", cost = 0.5, type = "C-classification")
lc2 = generateLearningCurveData(learners = lrn, task = datos.task,
                                percs = seq(0.1, 1, by = 0.1),
                                measures = list(ber, setAggregation(ber, train.mean)), resampling =
                                show.info = FALSE)
```

```
## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
```

```
## : Variable(s) 'LowQualFinSF' and 'KitchenAbvGr' and 'PoolArea' constant. Cannot
## scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'X3SsnPorch' and 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'KitchenAbvGr' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' and 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'X3SsnPorch' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' and 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.
```

```
## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'LowQualFinSF' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'X3SsnPorch' and 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.

## Warning in svm.default(d$data, d$target, probability = .learner$predict.type ==
## : Variable(s) 'PoolArea' constant. Cannot scale data.
```

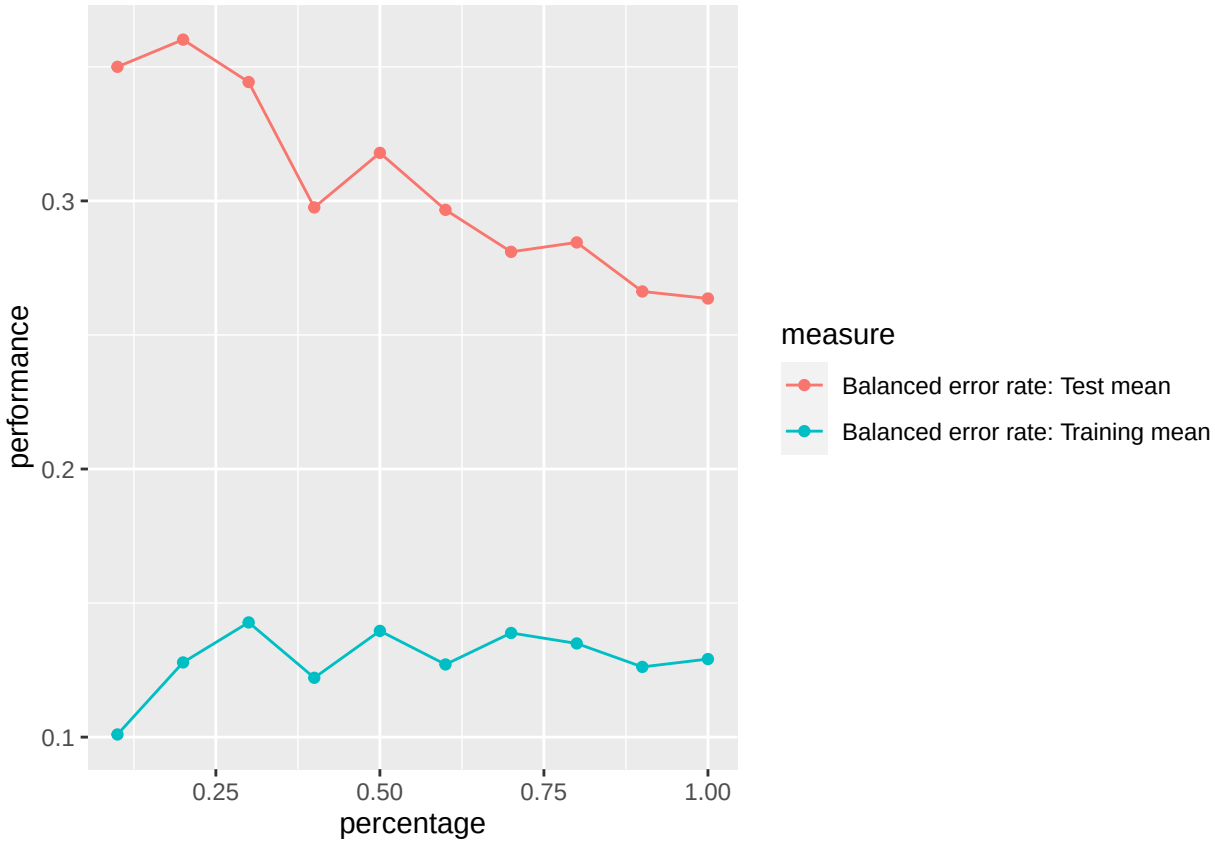
```
plotLearningCurve(lc2, facet = "learner")
```



Al momento de realizar la grafica se puede visualizar que la curva de training y la de test convergen hacia un mismo punto por lo que se puede conlcluir que este modelo no sufre de overfitting ni underfitting.

```
datos.task = makeClassifTask(data = train, target = "clasificacion")
rin2 = makeResampleDesc(method = "CV", iters = 10, predict = "both")
lrn = makeLearner("classif.svm", kernel = "radial", cost = 1, gamma = 2^-5, type = "C-classificatio
```

```
lc2 = generateLearningCurveData(learners = lrn, task = datos.task,
                                percs = seq(0.1, 1, by = 0.1),
                                measures = list(ber, setAggregation(ber, train.mean)), resampling =
                                show.info = FALSE)
plotLearningCurve(lc2, facet = "learner")
```



Al momento de realizar la grafica se puede mencionar que el modelo sufre de overtfitting esto debido a que primero la curva de training sigue una pequena tendencia a una linea recta y se puede mencionar que esta y la curva de test estan muy alejadas.

8. Comparaciones de algoritmos

Se puede mencionar que el segundo modelo es el único modelo que no presentó overfitting por o cual esto es un gran punto a tomar en cuenta para saber cual es el mejor modelo, en el caso de la precisión se puede mencionar que el mejor modelo fue el modelo 3 el cual obtuvo un accuracy del 0.73 mientras que el peor modelo fue el modelo 2 el cual obtuvo un accuracy de 0.64 lo cual no es tan malo porque acierta más de lo que erra, por lo cual se seleccionará el modelo 2 como el de preferencia.

9. Comparar la eficiencia del mejor modelo SVM con los resultados obtenidos en los algoritmos de hojas anteriores

Se puede mencionar que en comparación con los modelos de Hojas anteriores el que sigue resaltando es el modelo de árboles de decisión el cual obtuvo un accuracy de 1 por lo cual se puede concluir que no se equivocó en cambio el modelo de SVM tiene un accuracy de 0.64 por lo cual se va altamente superado y en

comparación de los otros modelos como el de random forest y el de naibe_bayes también se ve superado ya que el modelo de random forest obtuvo un 0.96 de accuracy y el modelo de naive bayes de 0.76.

10. Generar un modelo de regresion

10.1 Modelo de Regresion lineal Simple

```
Modelo_lineal_simple <-lm(SalePrice~OverallQual,data = train)
```

Resumen del modelo

```
summary(Modelo_lineal_simple)
```

```
##
## Call:
## lm(formula = SalePrice ~ OverallQual, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4139 -0.3619 -0.0109  0.2543  4.7549
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.02778    0.02112  -1.316    0.189
## OverallQual  0.77808    0.02166  35.916   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5913 on 782 degrees of freedom
## Multiple R-squared:  0.6226, Adjusted R-squared:  0.6221
## F-statistic:  1290 on 1 and 782 DF,  p-value: < 2.2e-16
```

RMSE

```
PrediccionSimple <-predict(Modelo_lineal_simple,newdata = test)
RMSE(PrediccionSimple,test$SalePrice)
```

```
## [1] 0.6326996
```

10.2 Modelo de Regresion Multivariable

```
Modelo_multiV <- lm(train$SalePrice~., data = train)
```

Resumen del modelo

```
summary(Modelo_multiV)
```

```
##
## Call:
## lm(formula = train$SalePrice ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5052 -0.2023 -0.0236  0.1860  3.4924
##
## Coefficients: (2 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.0760806  0.0274502   2.772 0.005717 **
## MSSubClass                 -0.1045898  0.0193461  -5.406 8.67e-08 ***
## LotFrontage                -0.0522736  0.0195559  -2.673 0.007681 **
## LotArea                     0.0556182  0.0163308   3.406 0.000695 ***
## OverallQual                 0.2918004  0.0291927   9.996  < 2e-16 ***
## OverallCond                 0.0509155  0.0205333   2.480 0.013370 *
## YearBuilt                   0.1181998  0.0377006   3.135 0.001784 **
## YearRemodAdd                0.0427009  0.0252613   1.690 0.091375 .
## MasVnrArea                  0.0616581  0.0188321   3.274 0.001109 **
## BsmtFinSF1                  0.0385827  0.0365452   1.056 0.291422
## BsmtFinSF2                  0.0051185  0.0176796   0.290 0.772268
## BsmtUnfSF                   0.0162193  0.0329115   0.493 0.622288
## TotalBsmtSF                        NA         NA      NA       NA
## X1stFlrSF                   0.1379169  0.0387162   3.562 0.000391 ***
## X2ndFlrSF                   0.1546144  0.0367891   4.203 2.96e-05 ***
## LowQualFinSF                0.0350299  0.0160366   2.184 0.029246 *
## GrLivArea                          NA         NA      NA       NA
## BsmtFullBath                0.0824783  0.0224989   3.666 0.000264 ***
## BsmtHalfBath                0.0311713  0.0159077   1.960 0.050424 .
## FullBath                    0.0800016  0.0283480   2.822 0.004897 **
## HalfBath                    0.0153159  0.0243992   0.628 0.530379
## BedroomAbvGr               -0.0790112  0.0235194  -3.359 0.000821 ***
## KitchenAbvGr               -0.0596872  0.0212447  -2.810 0.005091 **
## TotRmsAbvGrd                0.1359564  0.0332747   4.086 4.87e-05 ***
## Fireplaces                  0.0451997  0.0192653   2.346 0.019228 *
## GarageYrBlt                -0.0138982  0.0320949  -0.433 0.665116
## GarageCars                  0.1435264  0.0325407   4.411 1.18e-05 ***
## GarageArea                 -0.0034153  0.0321054  -0.106 0.915311
## WoodDeckSF                  0.0420047  0.0169448   2.479 0.013398 *
## OpenPorchSF                -0.0044741  0.0170937  -0.262 0.793594
## EnclosedPorch              -0.0008093  0.0181815  -0.045 0.964506
## X3SsnPorch                  0.0230325  0.0162540   1.417 0.156889
## ScreenPorch                 0.0667917  0.0160444   4.163 3.51e-05 ***
## PoolArea                   -0.0660013  0.0154510  -4.272 2.19e-05 ***
## MiscVal                    -0.0040578  0.0155268  -0.261 0.793899
## MoSold                     -0.0233380  0.0156002  -1.496 0.135075
## YrSold                     -0.0169784  0.0156392  -1.086 0.277994
## clasificacionEconomicas    -0.1050344  0.0528883  -1.986 0.047402 *
## clasificacionIntermedias -0.3317748  0.0453495  -7.316 6.60e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.426 on 747 degrees of freedom
## Multiple R-squared:  0.8129, Adjusted R-squared:  0.8039
## F-statistic: 90.14 on 36 and 747 DF,  p-value: < 2.2e-16
```

RMSE

```
PrediccionMulti <- predict(Modelo_multiV,newdata = test)
```

```
## Warning in predict.lm(Modelo_multiV, newdata = test): prediction from a
## rank-deficient fit may be misleading
```

```
RMSE(PrediccionMulti,test$SalePrice)
```

```
## [1] 0.4793374
```

11. Comparación de Resultados

Se puede mencionar que el modelo de regresión múltiple es el mejor modelo esto debido a que presentó un r^2 de 0.81 mientras que el modelo de regresión generado en hojas anteriores presentó un r^2 de 0.62 y también se puede mencionar que el modelo multivariable presenta un menor RMSE a comparación del primer modelo por lo que esto nos quiere decir que este modelo se ajusta mejor a los datos de entrada y se concluye que el modelo de regresión múltiple es mejor.