Informe Proyecto 2 entrega 4

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

2025-01-20

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

1. Elabore un modelo de regresión usando K nearest Neighbors (KNN), el conjunto de entrenamiento y la variable respuesta SalesPrice. Prediga con el modelo y explique los resultados a los que llega. Asegúrese que los conjuntos de entrenamiento y prueba sean los mismos de las entregas anteriores para que los modelos sean comparables.

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

train<-datosC[trainI, ]
test<-datosC[-trainI, ]

## Modelo:
train<-train[complete.cases(train),]
k_vecinos <- round(sqrt(nrow(datosC)),0)
parametros <- expand.grid(k = k_vecinos)

modelo_knn1 <- train(
  SalePrice~.,
  data=train,
  method = "knn",
  preProcess= c("center","scale","knnImpute"),
  tuneGrid = parametros
)
predModelo1 <- predict(modelo_knn1, newdata=test)
```

2. Analice los resultados del modelo de regresión usando KNN. ¿Qué tan bien le fue prediciendo? Utilice las métricas correctas.

```
pred_test <- predict(modelo_knn1, newdata = test)

metrics <- postResample(pred_test, test$SalePrice)
print(metrics)
```

```
##         RMSE     Rsquared          MAE
## 4.056178e+04 7.835675e-01 2.336609e+04
```

El $R^2 = 0.784$ es aceptable, El MAE es del orden de los \$20,000 lo que para el precio de una casa parece aceptable.

3. Compare los resultados con el modelo de regresión lineal, el mejor modelo de árbol de regresión y de naive bayes que hizo en las entregas pasadas. ¿Cuál funcionó mejor?

```
##              Modelo      RMSE       MAE        R2         r
## RMSE...1        KNN  40561.78  23366.09 0.7835675 0.8851935
## RMSE...2     Lineal  36792.87  21871.94 0.7772167 0.8815989
## RMSE...3      Árbol  46340.50  30161.08 0.6470656 0.8044039
## RMSE...4 NaiveBayes 196887.52 180897.46 0.6168137 0.7853749
```

Podemos notar que para todas las medidas de error excepto el MAE el mejor modelo es el KNN, el segundo mejor es el modelo lineal, seguido por el Arbol y el peor es el Naïve Bayes.

4. Haga un modelo de clasificación, use la variable categórica que hizo con el precio de las casas (barata, media y cara) como variable respuesta.

```
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)
#normalizar datos

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


#modelo de casificacion

modelo_knn_class<-train(
  x=x_train,
  y=y_train,
  method="knn"
  )
```

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

```
predic_knn_clas<-predict(modelo_knn_class, newdata = x_test)
```

6. Haga un análisis de la eficiencia del modelo de clasificación 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.

```
consfusion_m<- confusionMatrix(predic_knn_clas, y_test)
print(consfusion_m)
```

```
## 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
```

El modelo clasificó corectamente el 75.23% de las casas, segun el Accuaracy, es moderado. Hay desbalance de clases, en especifico para la clase Lujo, nótese por la sensibilidad, ya que no identifico a ninguna casa de lujo. La clasificacion erronea de viviendas de lujo como econommicas tienen impacto significativo ya que tienen una gran diferencia en el valor de las propiedades.

7. Analice el modelo. ¿Cree que pueda estar sobreajustado?

```
# analicemos el rendimiento del modelo con los datos de entrenamiento
predic_knn_clas_train<-predict(modelo_knn_class, newdata = x_train)

consfusion_m<- confusionMatrix(predic_knn_clas_train, y_train)
print(consfusion_m)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara
##   Economica        162         25    0
##   Intermedia        86        452   57
##   Cara               8         37  197
##
## Overall Statistics
##
##                Accuracy : 0.792
##                  95% CI : (0.7658, 0.8165)
##     No Information Rate : 0.502
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6557
##
##  Mcnemar's Test P-Value : 6.323e-10
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara
## Sensitivity                    0.6328            0.8794      0.7756
## Specificity                    0.9674            0.7196      0.9416
## Pos Pred Value                 0.8663            0.7597      0.8140
## Neg Pred Value                 0.8877            0.8555      0.9271
## Prevalence                     0.2500            0.5020      0.2480
## Detection Rate                 0.1582            0.4414      0.1924
## Detection Prevalence           0.1826            0.5811      0.2363
## Balanced Accuracy              0.8001            0.7995      0.8586
```

Al hacer una matriz de confucion con los datos de entrenamiento, podelmos notar que clasificó correctamente el 78.91% de las casa, muy similar al rendimiento con los datos de prueba (75.23%) . En general no parece sobre ajustado. Comparemos el rendimiento por clases:

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

En la tabla podemos notar que las columnas de test son similares a las de train, por lo que no pare que haya sobre ajuste.

Table 1: Métricas de Test y Train Comparadas

Metric	Economica..Train.	Economica..Test.	Intermedia..Train.	Intermedia..Test.	Cara..Train.	Cara..Test.
Sensitivity	0.6367	0.6239	0.8755	0.8539	0.7677	0.6759
Specificity	0.9674	0.9694	0.7176	0.6498	0.9390	0.9329
Pos Pred Value	0.8670	0.8718	0.7576	0.7110	0.8058	0.7684
Neg Pred Value	0.8888	0.8855	0.8512	0.8150	0.9246	0.8974
Prevalence	0.2500	0.2500	0.5020	0.5023	0.2480	0.2477
Detection Rate	0.1592	0.1560	0.4395	0.4289	0.1904	0.1674
Detection Prevalence	0.1836	0.1789	0.5801	0.6032	0.2363	0.2179
Balanced Accuracy	0.8021	0.7966	0.7966	0.7518	0.8533	0.8044

8. Haga un modelo usando validación cruzada, compare los resultados de este con los del modelo anterior. ¿Cuál funcionó mejor?

```
trctrl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 3)
modelo_knn_class_cruss <- train(
  x= x_train,
  y=y_train,
  method = "knn",
  trControl = trctrl,
  preProcess= c("center","scale")
)

pred_test_cruss<-predict(modelo_knn_class_cruss, newdata = x_test)

consfusion_m<- confusionMatrix(pred_test_cruss, y_test)
print(consfusion_m)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara
##   Economica         84         28    0
##   Intermedia        25        178   37
##   Cara               0         13   71
##
## Overall Statistics
##
##                Accuracy : 0.7638
##                  95% CI : (0.721, 0.8029)
##     No Information Rate : 0.5023
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6137
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara
## Sensitivity                    0.7706            0.8128      0.6574
```

```
## Specificity                 0.9144           0.7143      0.9604
## Pos Pred Value              0.7500           0.7417      0.8452
## Neg Pred Value              0.9228           0.7908      0.8949
## Prevalence                  0.2500           0.5023      0.2477
## Detection Rate              0.1927           0.4083      0.1628
## Detection Prevalence        0.2569           0.5505      0.1927
## Balanced Accuracy           0.8425           0.7635      0.8089
```

Parece que con validación cruzada el modelo es un poco mejor. porque clasifica muy bien las casas económicas pero un poco peor las demás classes. En general la diferencia no parece significativas.

#9. Tanto para los modelos de regresión como de clasificación, pruebe con varios valores de los hiperparámetros ¿Qué parámetros pueden tunearse en un KNN?, use el mejor modelo del tuneo, ¿Mejoraron los resultados usando el mejor modelo ahora? Explique

```
trctrl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 3)
# Ajuste de k con valores específicos: Clasificacion Por si te sirve
tune_grid <- expand.grid(k = c(3, 5, 7, 9, 11))
modelo_knn_class <- train(
  x = x_train,
  y = y_train,
  method = "knn",
  trControl = trctrl,
  tuneGrid = tune_grid
)
```

#10. Compare la eficiencia del algoritmo con el resultado obtenido con el árbol de decisión (el de clasificación), el modelo de random forest y el de naive bayes que hizo en las entregas pasadas. ¿Cuál es mejor para predecir? ¿Cuál se demoró más en procesar?