Informe Proyecto 2 entrega 3

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 bayes ingenuo (naive bayes), 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 hojas anteriores para que los modelos sean comparables.

En esta ocasión como naive Bayes así lo permite se tomaran todas las variables incluso las culitativas.

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

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

trainC<-datosC[trainI, ]
testC<-datosC[-trainI, ]
```

```
modelo <- naiveBayes(SalePrice ~ ., data = train)
```

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

```
# Predicciones para test
predTest <- predict(modelo, newdata = test)
predTest <- as.numeric(as.character(predTest))  # Convertir a numérico

# Calcular métricas para test
rmse_test <- rmse(test$SalePrice, predTest)
mae_test  <- mae(test$SalePrice, predTest)
rmsle_test <- sqrt(mean((log1p(test$SalePrice) - log1p(predTest))^2))

# Calcular R^2 para test
SST_test <- sum((test$SalePrice - mean(test$SalePrice))^2)  # Suma de cuadrados total
```

```r
SSE_test <- sum((test$SalePrice - predTest)^2)              # Suma de cuadrados del error
r2_test <- 1 - (SSE_test / SST_test)  # Fórmula de R^2

# Mostrar resultados para test
cat("RMSE test:", rmse_test, "\n")
```

```
## RMSE test: 45549.32
```

```r
cat("MAE test:", mae_test, "\n")
```

```
## MAE test: 30727.89
```

```r
cat("RMSLE test:", rmsle_test, "\n")
```

```
## RMSLE test: 0.2519688
```

```r
cat("R^2 test:", r2_test, "\n")
```

```
## R^2 test: 0.657578
```

```r
# ----------------------------------------------------------

# Predicciones para train
predTrain <- predict(modelo, newdata = train)
predTrain <- as.numeric(as.character(predTrain))  # Convertir a numérico

# Calcular métricas para train
rmse_train <- rmse(train$SalePrice, predTrain)
mae_train  <- mae(train$SalePrice, predTrain)
rmsle_train <- sqrt(mean((log1p(train$SalePrice) - log1p(predTrain))^2))

# Calcular R^2 para train
SST_train <- sum((train$SalePrice - mean(train$SalePrice))^2)  # Suma de cuadrados total
SSE_train <- sum((train$SalePrice - predTrain)^2)              # Suma de cuadrados del error
r2_train <- 1 - (SSE_train / SST_train)  # Fórmula de R^2

# Mostrar resultados para train
cat("RMSE train:", rmse_train, "\n")
```

```
## RMSE train: 43886.59
```

```r
cat("MAE train:", mae_train, "\n")
```

```
## MAE train: 20423.46
```

```r
cat("RMSLE train:", rmsle_train, "\n")
```

```
## RMSLE train: 0.2030676
```

```r
cat("R^2 train:", r2_train, "\n")
```

```
## R^2 train: 0.699633
```

El R^2 es de 0.657 no esta muy cercano a 1, pero tampoco es terrible. El RMSLE es de 0.2 lo que nos indica que el modelo tiene margen de mejora.

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

```r
# Entrenar modelo Naïve Bayes
library(e1071)
library(glmnet)

# Entrenar modelo Naïve Bayes
modelo_bayes <- naiveBayes(SalePrice ~ ., data = train)

# Predicciones para test
pred_bayes <- predict(modelo, newdata = test)
predBayes_numeric <- as.numeric(as.character(pred_bayes))  # Convertir factor a numérico

# Calcular métricas para Naïve Bayes
rmse_bayes <- rmse(test$SalePrice, predBayes_numeric)
mae_bayes  <- mae(test$SalePrice, predBayes_numeric)
r2_bayes   <- 1 - sum((test$SalePrice - predBayes_numeric)^2) / sum((test$SalePrice - mean(test$SalePri

# Modelo de Regresión Lineal
X_train <- model.matrix(SalePrice ~ ., data = trainC)[, -1]  # Matriz sin el intercepto
X_test <- model.matrix(SalePrice ~ ., data = testC)[, -1]
y_train <- train$SalePrice
y_test <- testC$SalePrice

set.seed(123)
modelo_lasso <- cv.glmnet(X_train, y_train, alpha = 1)  # Regularización Lasso

# Predicción con Lasso
pred_lasso <- predict(modelo_lasso, s = "lambda.min", newx = X_test)

# Calcular métricas para modelo de regresión lineal regularizada
rmse_lasso <- rmse(y_test, pred_lasso)
mae_lasso  <- mae(y_test, pred_lasso)
r2_lasso   <- 1 - sum((y_test - pred_lasso)^2) / sum((y_test - mean(y_test))^2)

# Modelo de árbol de decisión
arbol <- rpart(SalePrice ~ ., data = trainC)

# Predicción para test
pred_arbol <- predict(arbol, newdata = testC)

# Calcular métricas para árbol de decisión
rmse_arbol <- rmse(y_test, pred_arbol)
mae_arbol  <- mae(y_test, pred_arbol)
r2_arbol   <- 1 - sum((y_test - pred_arbol)^2) / sum((y_test - mean(y_test))^2)

# Comparación de modelos
resultados <- data.frame(
  Modelo = c("Naïve Bayes", "Regresión Lineal ", "Árbol de Regresión"),
  RMSE = c(rmse_bayes, rmse_lasso, rmse_arbol),
  MAE  = c(mae_bayes, mae_lasso, mae_arbol),
```

```
  R2   = c(r2_bayes, r2_lasso, r2_arbol)
)

# Mostrar resultados comparativos
print(resultados)
```

```
##                Modelo     RMSE      MAE        R2
## 1         Naïve Bayes 45549.32 30727.89 0.6575780
## 2  Regresión Lineal  37983.42 22724.28 0.7618854
## 3 Árbol de Regresión 46340.50 30161.08 0.6455791
```

En la tabla comparativa con cualquiera de las tres métricas se puede concluir que el mejor modelo es el lineal, el segundo mejor es el Naive Bayes y el peor es el árbol de regresión.

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.

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

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.