Informe Proyecto 2 entrega 2

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

2025-01-20

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

1 Se usaran los mismos conjuntos de entrenamiento y prueba que usó para los modelos de regresión lineal en la entrega anterior.

Pero antes se agregará la varible nueva CategoriaPrecios, que agrupe los precios de las casas en 3 categorías: Económicas, Intermedias o Caras.

```
y<- datosC$SalePrice
set.seed(123)
trainI<- createDataPartition(y, p=0.8, list=FALSE)
train<-datosC[trainI, ]
test<-datosC[-trainI, ]
```

```
## Conjunto de entrenamiento (cantidad de muestras: 1169 )
##
```

```
##   SalePrice LotFrontage LotArea OverallQual OverallCond YearBuilt YearRemodAdd
## 1    208500          65    8450           7           5      2003         2003
## 2    181500          80    9600           6           8      1976         1976
## 3    223500          68   11250           7           5      2001         2002
## 4    140000          60    9550           7           5      1915         1970
## 5    250000          84   14260           8           5      2000         2000
## 6    143000          85   14115           5           5      1993         1995
##   MasVnrArea BsmtFinSF1 BsmtFinSF2 BsmtUnfSF TotalBsmtSF X1stFlrSF X2ndFlrSF
## 1        196        706          0       150         856       856       854
## 2          0        978          0       284        1262      1262         0
## 3        162        486          0       434         920       920       866
## 4          0        216          0       540         756       961       756
## 5        350        655          0       490        1145      1145      1053
## 6          0        732          0        64         796       796       566
##   LowQualFinSF GrLivArea BsmtFullBath BsmtHalfBath FullBath HalfBath
## 1            0      1710            1            0        2        1
## 2            0      1262            0            1        2        0
## 3            0      1786            1            0        2        1
## 4            0      1717            1            0        1        0
## 5            0      2198            1            0        2        1
## 6            0      1362            1            0        1        1
##   BedroomAbvGr KitchenAbvGr TotRmsAbvGrd Fireplaces GarageYrBlt GarageCars
## 1            3            1            8          0        2003          2
## 2            3            1            6          1        1976          2
## 3            3            1            6          1        2001          2
## 4            3            1            7          1        1998          3
```

```
## 5            4            1            9          1        2000          3
## 6            1            1            5          0        1993          2
##   GarageArea WoodDeckSF OpenPorchSF EnclosedPorch X3SsnPorch ScreenPorch
## 1        548          0          61             0          0           0
## 2        460        298           0             0          0           0
## 3        608          0          42             0          0           0
## 4        642          0          35           272          0           0
## 5        836        192          84             0          0           0
## 6        480         40          30             0        320           0
##   PoolArea MiscVal MoSold YrSold
## 1        0       0      2   2008
## 2        0       0      5   2007
## 3        0       0      9   2008
## 4        0       0      2   2006
## 5        0       0     12   2008
## 6        0     700     10   2009
```

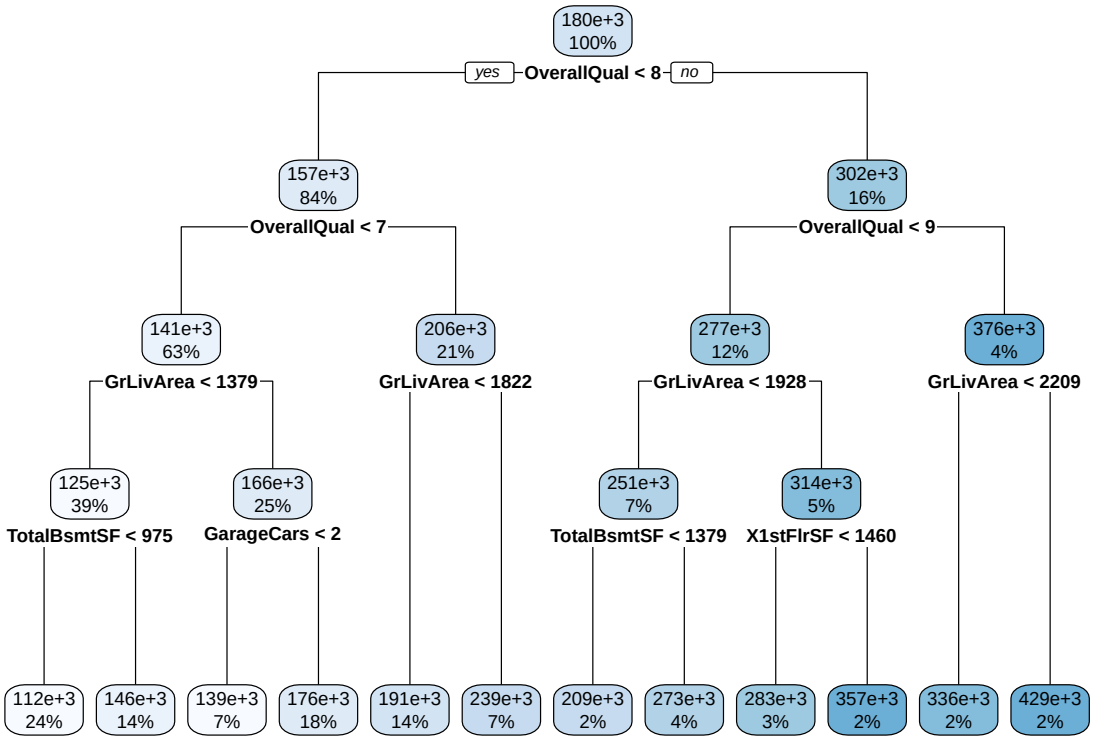
```
## Conjunto de prueba (cantidad de muestras: 291 )
##
```

```
##    SalePrice LotFrontage LotArea OverallQual OverallCond YearBuilt YearRemodAdd
## 8     200000           0   10382           7           6      1973         1973
## 11    129500          70   11200           5           5      1965         1965
## 13    144000           0   12968           5           6      1962         1962
## 30     68500          60    6324           4           6      1927         1950
## 37    145000         112   10859           5           5      1994         1995
## 39    109000          68    7922           5           7      1953         2007
##    MasVnrArea BsmtFinSF1 BsmtFinSF2 BsmtUnfSF TotalBsmtSF X1stFlrSF X2ndFlrSF
## 8         240        859         32       216        1107      1107       983
## 11          0        906          0       134        1040      1040         0
## 13          0        737          0       175         912       912         0
## 30          0          0          0       520         520       520         0
## 37          0          0          0      1097        1097      1097         0
## 39          0        731          0       326        1057      1057         0
##    LowQualFinSF GrLivArea BsmtFullBath BsmtHalfBath FullBath HalfBath
## 8             0      2090            1            0        2        1
## 11            0      1040            1            0        1        0
## 13            0       912            1            0        1        0
## 30            0       520            0            0        1        0
## 37            0      1097            0            0        1        1
## 39            0      1057            1            0        1        0
##    BedroomAbvGr KitchenAbvGr TotRmsAbvGrd Fireplaces GarageYrBlt GarageCars
## 8             3            1           7          2        1973          2
## 11            3            1           5          0        1965          1
## 13            2            1           4          0        1962          1
## 30            1            1           4          0        1920          1
## 37            3            1           6          0        1995          2
## 39            3            1           5          0        1953          1
##    GarageArea WoodDeckSF OpenPorchSF EnclosedPorch X3SsnPorch ScreenPorch
## 8         484        235         204           228          0           0
## 11        384          0           0             0          0           0
## 13        352        140           0             0          0         176
## 30        240         49           0            87          0           0
## 37        672        392          64             0          0           0
## 39        246          0          52             0          0           0
```

```
##    PoolArea MiscVal MoSold YrSold
## 8         0     350     11   2009
## 11        0       0      2   2008
## 13        0       0      9   2008
## 30        0       0      5   2008
## 37        0       0      6   2009
## 39        0       0      1   2010
```

2. Arbol de regresión para predecir el precio de las casas usando todas las variables.

```
arbol1 <- rpart(SalePrice~.,data = train)
rpart.plot(arbol1)
```



3. Úselo para predecir y analice el resultado. ¿Qué tal lo hizo?

```
# Calcular las predicciones
predicciones <- predict(arbol1, newdata = test)

# Calcular MSE (Error Cuadrático Medio)
mse <- mean((train$SalePrice - predicciones)^2)
```

```
## Warning in train$SalePrice - predicciones: longer object length is not a
## multiple of shorter object length
```

```
# Calcular MAE (Error Absoluto Medio)
mae <- mean(abs(train$SalePrice - predicciones))
```

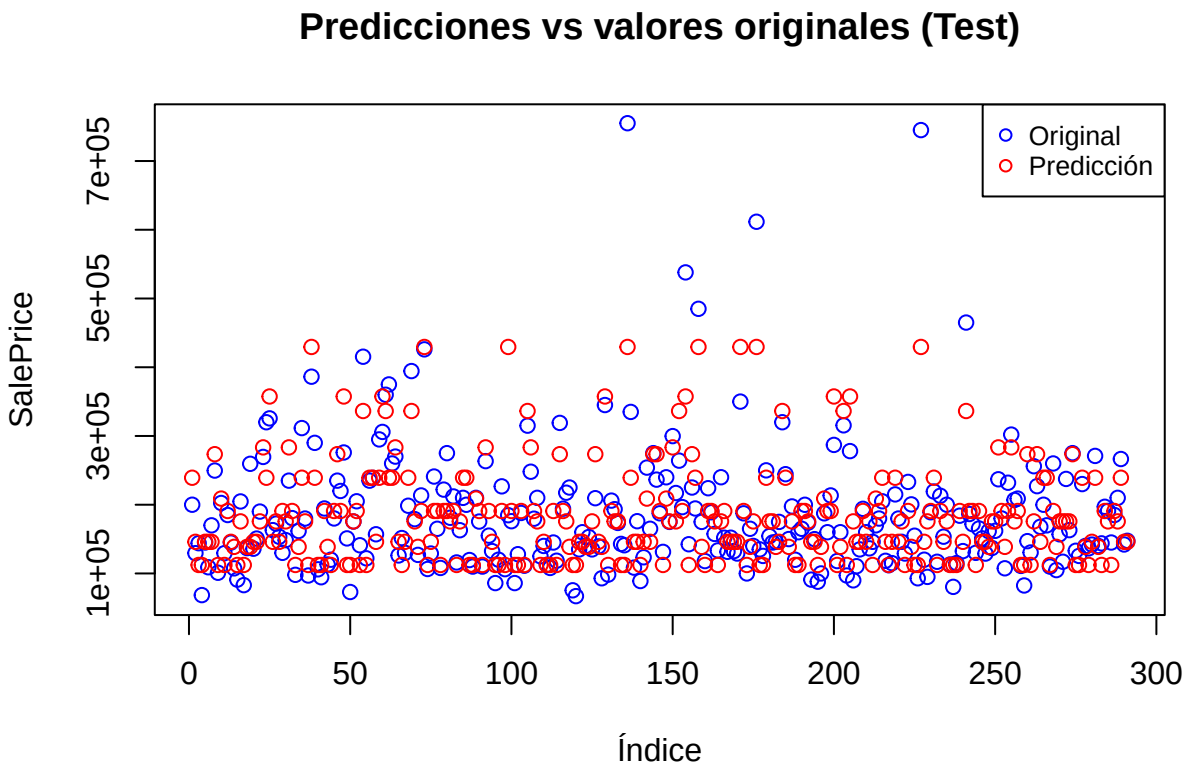
```
## Warning in train$SalePrice - predicciones: longer object length is not a
```

```
## multiple of shorter object length
```

```
# Mostrar los resultados
cat("MSE:", mse, "\nMAE:", mae)
```

```
## MSE: 11520168038
## MAE: 79367.99
```

```
# Graficar los valores originales del conjunto de prueba
plot(test$SalePrice, col = "blue", main = "Predicciones vs valores originales (Test)",
     xlab = "Índice", ylab = "SalePrice")

# Agregar las predicciones al gráfico
points(predicciones, col = "red")

# Agregar la leyenda
legend("topright", legend = c("Original", "Predicción"),
       col = c("blue", "red"), pch = 1, cex = 0.8)
```



El model tiene un MAE y un MSE altos, la predicción es muy burda.

4. Haga, al menos, 3 modelos más, cambiando el parámetro de la profundidad del árbol. ¿Cuál es el mejor modelo para predecir el precio de las casas?

```
# Modelo original (sin especificar maxdepth, usa el máximo por defecto)
arbol1 <- rpart(SalePrice ~ ., data = train)

# Modelos con diferentes profundidades
arbol2 <- rpart(SalePrice ~ ., data = train, control = rpart.control(maxdepth = 4))
```

```r
arbol3 <- rpart(SalePrice ~ ., data = train, control = rpart.control(maxdepth = 3))
arbol4 <- rpart(SalePrice ~ ., data = train, control = rpart.control(maxdepth = 2))

# Función para calcular MSE y MAE
calcular_errores <- function(modelo) {
  pred <- predict(modelo, newdata = train)
  mse <- mean((train$SalePrice - pred)^2)
  mae <- mean(abs(train$SalePrice - pred))
  return(c(MSE = mse, MAE = mae))
}

# Calcular errores para cada modelo
errores1 <- calcular_errores(arbol1)
errores2 <- calcular_errores(arbol2)
errores3 <- calcular_errores(arbol3)
errores4 <- calcular_errores(arbol4)

# Mostrar los resultados
resultados <- data.frame(
  Modelo = c("Original (sin maxdepth)", "maxdepth = 4", "maxdepth = 3", "maxdepth = 2"),
  MSE = c(errores1[1], errores2[1], errores3[1], errores4[1]),
  MAE = c(errores1[2], errores2[2], errores3[2], errores4[2])
)

print(resultados)
```

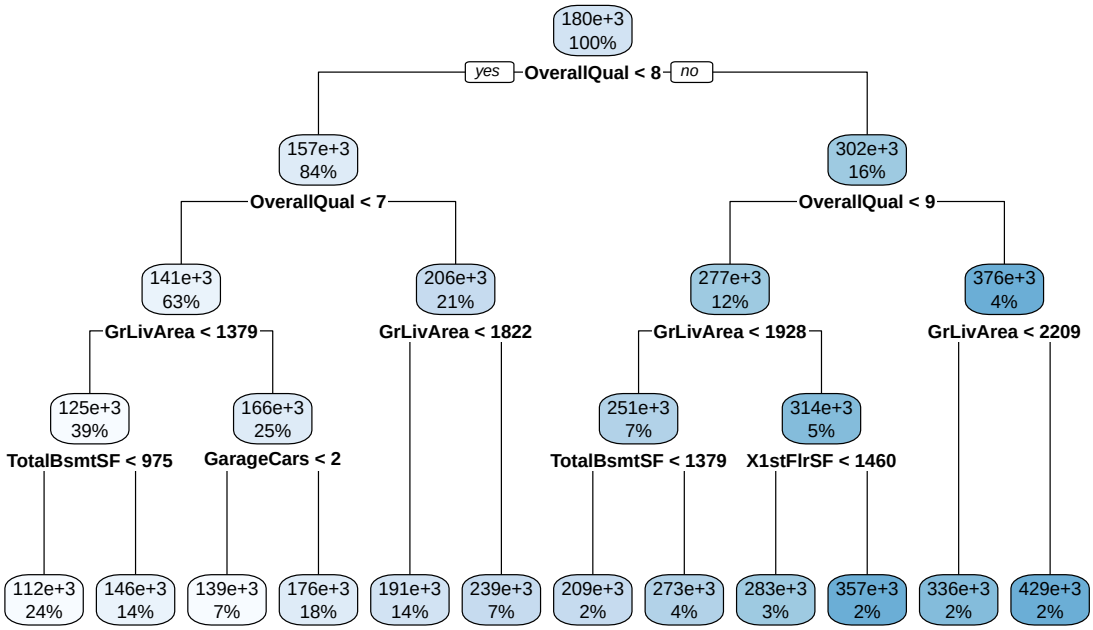
```
##                    Modelo        MSE      MAE
## 1 Original (sin maxdepth) 1236870940 25300.55
## 2            maxdepth = 4 1236870940 25300.55
## 3            maxdepth = 3 1536838560 29101.22
## 4            maxdepth = 2 2091057015 33936.92
```

```r
# Identificar el mejor modelo (menor MSE)
mejor_modelo <- resultados[which.min(resultados$MSE), "Modelo"]
cat("\nEl mejor modelo según el MSE es:", mejor_modelo, "\n")
```

```
##
## El mejor modelo según el MSE es: Original (sin maxdepth)
```

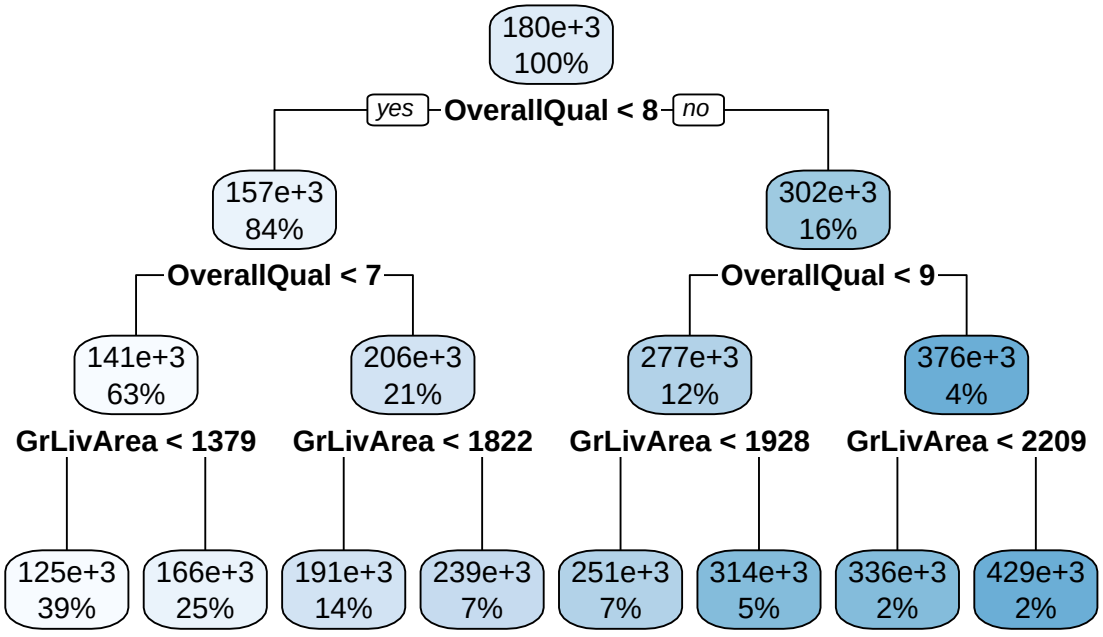
```r
rpart.plot(arbol2, main = "Árbol con maxdepth = 4")
```

Árbol con maxdepth = 4

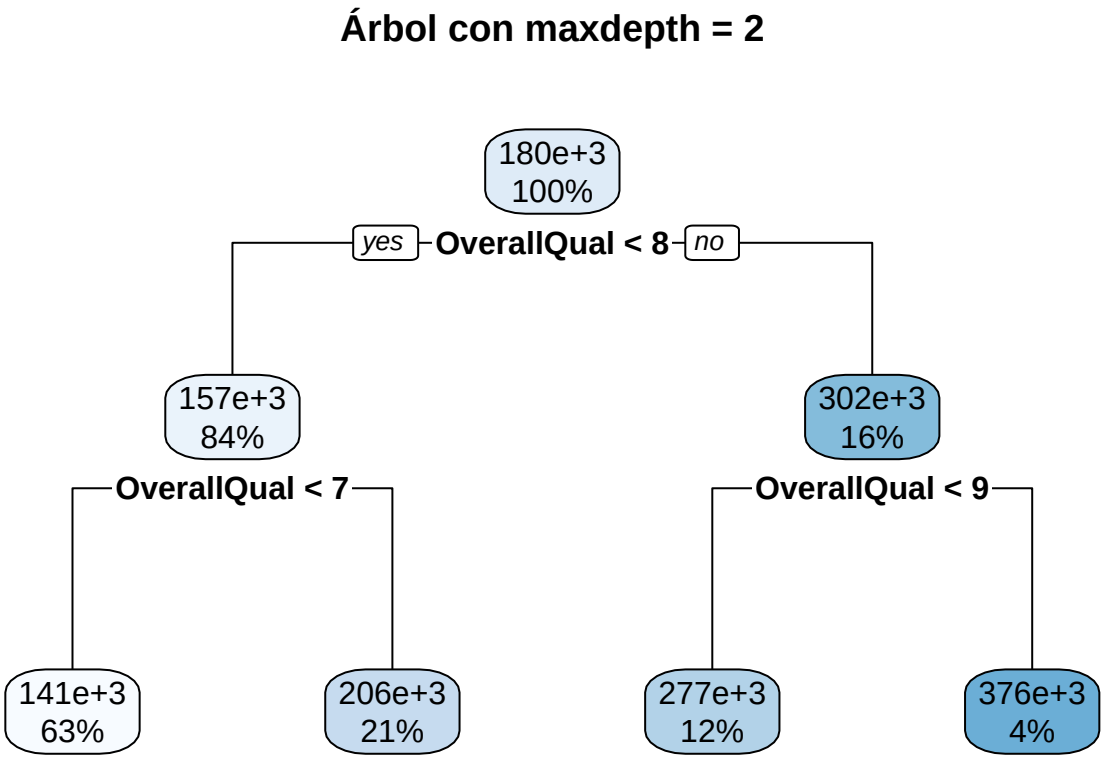



```
rpart.plot(arbol3, main = "Árbol con maxdepth = 3")
```

Árbol con maxdepth = 3




```
rpart.plot(arbol4, main = "Árbol con maxdepth = 2")
```




```
# Resetear la ventana gráfica
par(mfrow = c(1, 1))
```

Como es de esperar a mayor profundidad mayor error

5. Compare los resultados con el modelo de regresión lineal de la hoja anterior, ¿cuál lo hizo mejor?

```
## Variables seleccionadas: LotArea OverallQual OverallCond YearBuilt YearRemodAdd MasVnrArea BsmtFinSF
```

```
# Modelo original (sin especificar maxdepth, usa el máximo por defecto)
arbol1 <- rpart(SalePrice ~ ., data = train)


#modelo de hoja anterior
# Filtra las variables seleccionadas en el data frame
formula <- as.formula(paste("SalePrice ~", paste(variables_seleccionadas, collapse = "+")))
modelo2 <- lm(formula, data = train)
modelo2 <-step(modelo2, direction = "backward")
```

```
## Start:  AIC=24300.12
## SalePrice ~ LotArea + OverallQual + OverallCond + YearBuilt +
##     YearRemodAdd + MasVnrArea + BsmtFinSF1 + TotalBsmtSF + X1stFlrSF +
##     GrLivArea + BsmtFullBath + BedroomAbvGr + KitchenAbvGr +
##     TotRmsAbvGrd + Fireplaces + GarageYrBlt + GarageCars + GarageArea +
##     WoodDeckSF + OpenPorchSF + ScreenPorch + PoolArea + YrSold
##
##                Df  Sum of Sq        RSS   AIC
## - GarageArea    1 1.0796e+08 1.1960e+12 24298
## <none>                       1.1959e+12 24300
```

```
## - Fireplaces    1 2.2765e+09 1.1982e+12 24300
## - GarageYrBlt   1 2.9224e+09 1.1988e+12 24301
## - BsmtFinSF1    1 3.0920e+09 1.1990e+12 24301
## - TotalBsmtSF   1 3.7249e+09 1.1997e+12 24302
## - OpenPorchSF   1 3.7344e+09 1.1997e+12 24302
## - YrSold        1 4.0832e+09 1.2000e+12 24302
## - YearRemodAdd  1 4.8179e+09 1.2007e+12 24303
## - ScreenPorch   1 6.1345e+09 1.2021e+12 24304
## - X1stFlrSF     1 8.6312e+09 1.2046e+12 24306
## - WoodDeckSF    1 8.9420e+09 1.2049e+12 24307
## - BsmtFullBath  1 1.4400e+10 1.2103e+12 24312
## - GarageCars    1 1.5089e+10 1.2110e+12 24313
## - LotArea       1 1.5184e+10 1.2111e+12 24313
## - YearBuilt     1 2.2571e+10 1.2185e+12 24320
## - OverallCond   1 2.3097e+10 1.2190e+12 24320
## - MasVnrArea    1 2.3772e+10 1.2197e+12 24321
## - PoolArea      1 2.5138e+10 1.2211e+12 24322
## - KitchenAbvGr  1 2.8080e+10 1.2240e+12 24325
## - BedroomAbvGr  1 2.8131e+10 1.2241e+12 24325
## - TotRmsAbvGrd  1 2.8650e+10 1.2246e+12 24326
## - GrLivArea     1 7.3083e+10 1.2690e+12 24368
## - OverallQual   1 1.9648e+11 1.3924e+12 24476
##
## Step:  AIC=24298.22
## SalePrice ~ LotArea + OverallQual + OverallCond + YearBuilt +
##     YearRemodAdd + MasVnrArea + BsmtFinSF1 + TotalBsmtSF + X1stFlrSF +
##     GrLivArea + BsmtFullBath + BedroomAbvGr + KitchenAbvGr +
##     TotRmsAbvGrd + Fireplaces + GarageYrBlt + GarageCars + WoodDeckSF +
##     OpenPorchSF + ScreenPorch + PoolArea + YrSold
##
##                Df  Sum of Sq        RSS   AIC
## <none>                       1.1960e+12 24298
## - Fireplaces    1 2.2012e+09 1.1982e+12 24298
## - BsmtFinSF1    1 3.1927e+09 1.1992e+12 24299
## - GarageYrBlt   1 3.2826e+09 1.1993e+12 24299
## - TotalBsmtSF   1 3.7837e+09 1.1998e+12 24300
## - OpenPorchSF   1 3.8410e+09 1.1999e+12 24300
## - YrSold        1 4.0719e+09 1.2001e+12 24300
## - YearRemodAdd  1 4.7164e+09 1.2008e+12 24301
## - ScreenPorch   1 6.1798e+09 1.2022e+12 24302
## - X1stFlrSF     1 8.8763e+09 1.2049e+12 24305
## - WoodDeckSF    1 8.9140e+09 1.2049e+12 24305
## - BsmtFullBath  1 1.4476e+10 1.2105e+12 24310
## - LotArea       1 1.5370e+10 1.2114e+12 24311
## - YearBuilt     1 2.2465e+10 1.2185e+12 24318
## - OverallCond   1 2.3617e+10 1.2197e+12 24319
## - MasVnrArea    1 2.3980e+10 1.2200e+12 24319
## - PoolArea      1 2.5066e+10 1.2211e+12 24320
## - KitchenAbvGr  1 2.8367e+10 1.2244e+12 24324
## - BedroomAbvGr  1 2.8380e+10 1.2244e+12 24324
## - TotRmsAbvGrd  1 2.8589e+10 1.2246e+12 24324
## - GarageCars    1 4.7181e+10 1.2432e+12 24342
## - GrLivArea     1 7.3874e+10 1.2699e+12 24366
## - OverallQual   1 1.9638e+11 1.3924e+12 24474
```

```r
# Función para calcular MSE y MAE
calcular_errores <- function(modelo) {
  pred <- predict(modelo, newdata = train)
  mse <- mean((train$SalePrice - pred)^2)
  mae <- mean(abs(train$SalePrice - pred))
  return(c(MSE = mse, MAE = mae))
}

# Calcular errores para cada modelo
erroresarbol1 <- calcular_errores(arbol1)
erroresmodelo2 <- calcular_errores(modelo2)


# Mostrar los resultados
resultados <- data.frame(
  Modelo = c("Arbol 1", "Modelo 2" ),
  MSE = c(erroresarbol1[1], erroresmodelo2[1]),
  MAE = c(erroresarbol1[2], erroresmodelo2[2])
)

print(resultados)
```

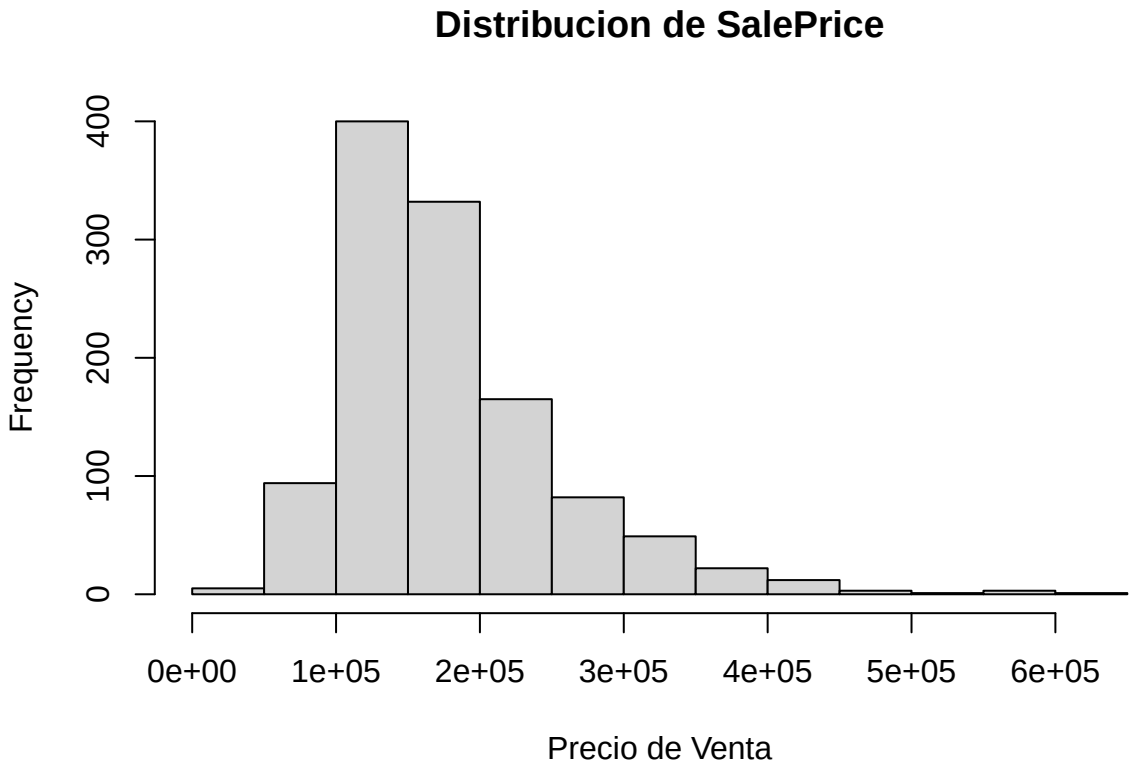
```
##     Modelo        MSE      MAE
## 1  Arbol 1 1236870940 25300.55
## 2 Modelo 2 1023126215 20940.05
```

```r
# Identificar el mejor modelo (menor MSE)
mejor_modelo <- resultados[which.min(resultados$MSE), "Modelo"]
cat("\nEl mejor modelo según el MSE es:", mejor_modelo, "\n")
```

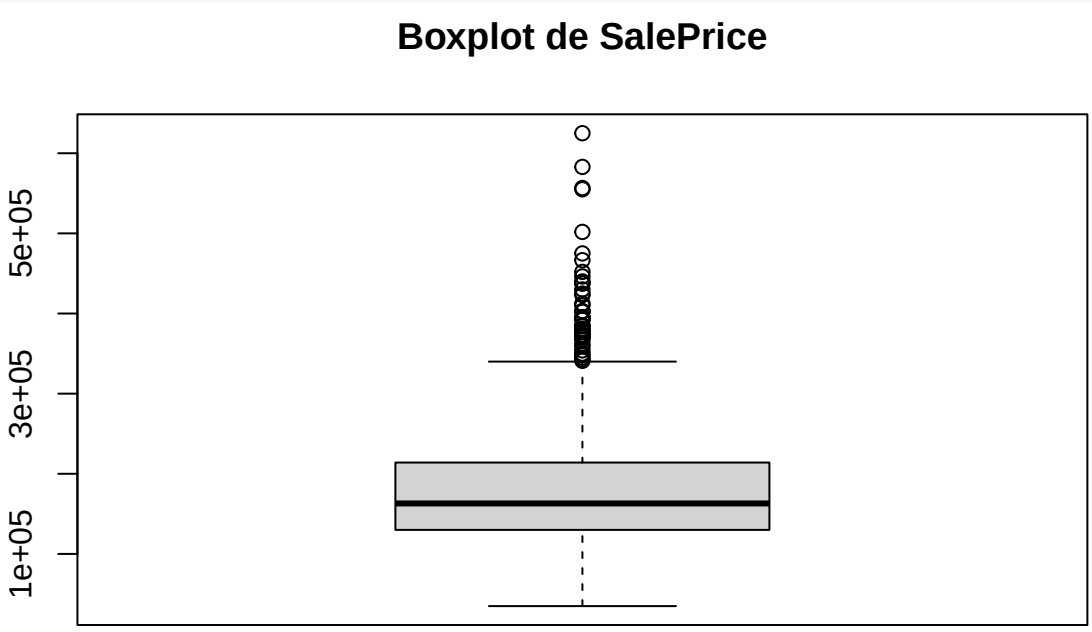
```
##
## El mejor modelo según el MSE es: Modelo 2
```

6. Dependiendo del análisis exploratorio elaborado cree una variable respuesta que le permita clasificar las casas en Económicas, Intermedias o Caras. Los límites de estas clases deben tener un fundamento en la distribución de los datos de precios, y estar bien explicados

```r
hist(train$SalePrice, main="Distribucion de SalePrice", xlab="Precio de Venta")
```




```
boxplot(train$SalePrice, main="Boxplot de SalePrice")
```




```
summary(train$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  130000  163000  179755  214000  625000
```

```
quantile(train$SalePrice, probs=c(0.25, 0.5, 0.75))
```

```
##    25%    50%    75%
## 130000 163000 214000
```

Notese que en el histograma hay asimetria positiva, por lo que hay cola larga a los precios altos.

En el Boxplot se nota valores atipicos, que son significativamente altos, que pueden conciderarse como las casas con precios muy altos, por lo que se concidera crear otra cateoria, que se llame lujo representado las casas que en precio son superiores a 400000.

La creacion de la variable categorica se basa en los cuartiles de SalePrice, con los cortes lógicos de la siguiente forma:

Económicas, hasta el primer cuartil (129975) Intermedias, entre el primer y tercer cuartil (129975, 214000) Cara, por encima del tercer cuartil hasta el comienzo de puntos atipicos (214000, 400000) Lujo, Por encima de los valores atipicos (400000)

La distribución queda de la siguente forma.

```
train$precio_categoria<-cut(train$SalePrice,
                            breaks = c(0, 129975, 214000, 400000, Inf),
                            labels = c("Economica", "Intermedia", "Cara", "Lujo"),
                            include.lowest = TRUE)
table(train$precio_categoria)
```

```
##
##  Economica Intermedia       Cara       Lujo
##        292        587        270         20
```

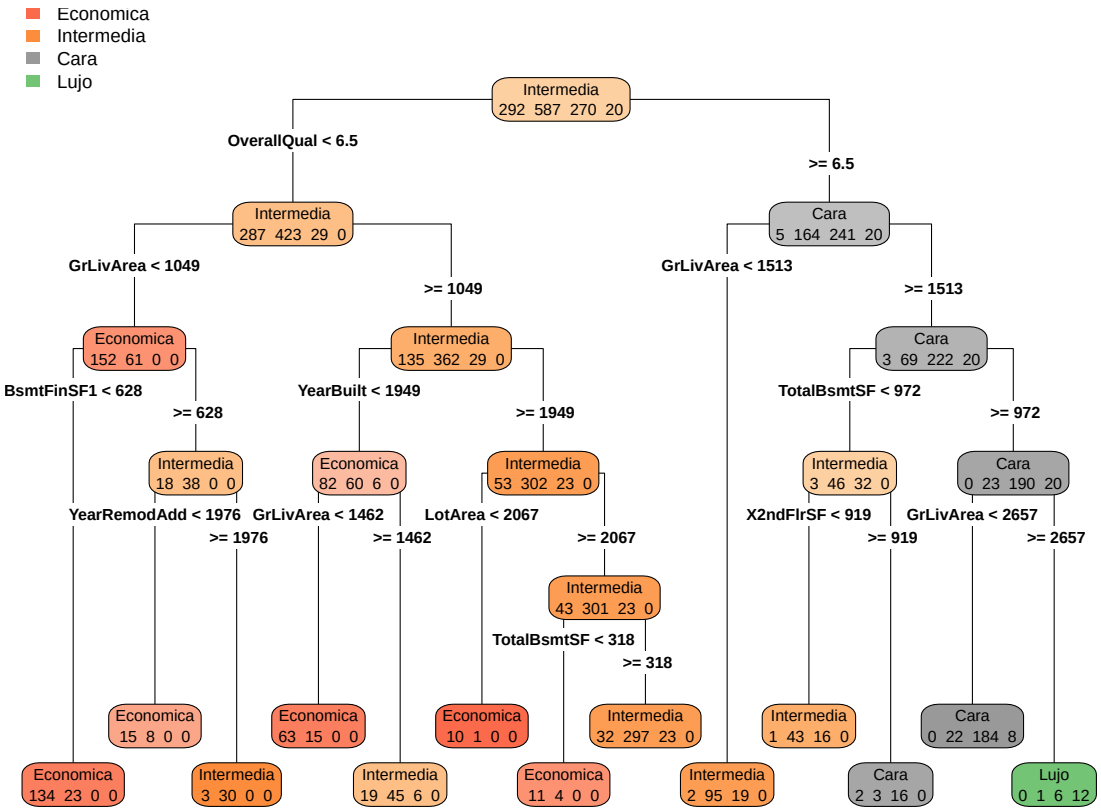
7. Elabore un árbol de clasificación utilizando la variable respuesta que creó en el punto anterior. Explique los resultados a los que llega. Muestre el modelo gráficamente. Recuerde que la nueva variable respuesta es categórica, pero se generó a partir de los precios de las casas, no incluya el precio de venta para entrenar el modelo.

```
arbol_clasificacion<- rpart(precio_categoria ~ .-SalePrice, data = train )
#summary(arbol_clasificacion)
```

```
set.seed(123)
corte<-sample(1:nrow(train), 0.8 * nrow(train))
train_set<-train[corte, ]
test_set<-train[-corte, ]
train$SalePrice<-NULL
test$SalePrice<-NULL
```

```
rpart.plot(arbol_clasificacion, type = 4, extra=1)
```

```
## Warning: Cannot retrieve the data used to build the model (model.frame: object 'SalePrice' not found
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



El árbol utiliza las varibales GrLivArea, OverallQual, BsmtFinSF1, YearBuilt, para clasificar las viviendas. Mostrando que el tamaño, calidad y antiguedad de la casa son los factores que mas influyen en el precio. Las viviendas de precio alto son las de alta calidad y gran área. Las de baja calidad y area mas pequeña las calasifica como en precio bajo. La clasificacion de Lujo tiene menor cantidad de de valores de acierto, que puede deberse a pocos datos en esta categoria a comparacion con las otras.

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

9. Haga un análisis de la eficiencia del algoritmo usando una matriz de confusión para el árbol de clasificación. Tenga en cuenta la efectividad, donde el algoritmo se equivocó más, donde se equivocó menos y la importancia que tienen los errores.

```
predicciones_clas<-predict(arbol_clasificacion, newdata = test_set)


predicciones_clas <-apply(predicciones_clas, 1, function(x) colnames(predicciones_clas)[which.max(x)])

predicciones_clas<- as.factor(predicciones_clas)

cfm <- confusionMatrix(predicciones_clas, test_set$precio_categoria)
```

```
## Warning in confusionMatrix.default(predicciones_clas,
## test_set$precio_categoria): Levels are not in the same order for reference and
```

```
## data. Refactoring data to match.
```

```
print(cfm)
```

```
## 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
```

Según la matriz de confusión el modelo clasifica de buena manera las categorias económica y Lujo con precision alta, aunque hay cierta confusión entre las categorias intermedia y cara, donde se clasifica algunas casas en intermedia cuando es cara y viceversa.

Las categorias muestran alta sencibilidad y y especificidad por lo que el modelo identifica correctamente las viviendas en cada categoría. POr otro lado la categoria Intermedia tiene el valor mas alto de 0.9892 a comparación con Lujo que tiene 0.8000, por que se concluye que el modelo tiende a clasificar de mejor manera la categoria Intermedia que Lujo a comparación con intermedia.

Nótese que como valor de Accuracy hay 88.89%, por lo que el modelo tiene buen rendimiento.

10. Entrene un modelo usando validación cruzada, prediga con él. ¿le fue mejor que al modelo anterior?

```
control<-trainControl(method = "cv", number = 10)
abrol_cv<-train(precio_categoria ~ .- SalePrice, data=train_set, method="rpart", trControl=control)
predicciones_cv<-predict(abrol_cv, newdata=test_set)
accuarcy_cv<-sum(diag(table(test_set$precio_categoria, predicciones_cv))) /nrow(test_set)
print(paste("Precision con validacion cruzada:", accuarcy_cv))
```

```
## [1] "Precision con validacion cruzada: 0.683760683760684"
```

```
predicciones_test<-predict(arbol_clasificacion, newdata=test_set, type = "class")
accuarcy_test<-sum(diag(table(test_set$precio_categoria, predicciones_cv))) /nrow(test_set)
print(paste("Precision del modelo original:", accuarcy_test))
```

```
## [1] "Precision del modelo original: 0.683760683760684"
```

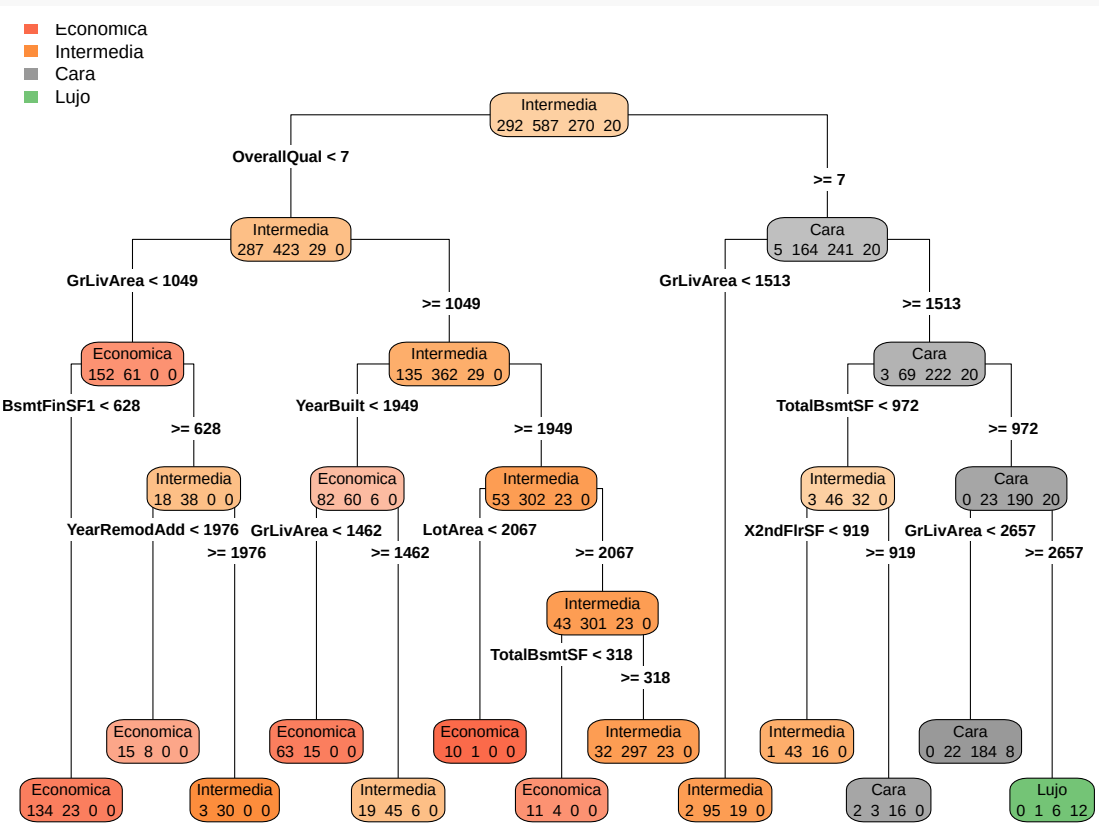
```
diferencia<-accuarcy_cv-accuarcy_test

result<-sign(diferencia)

print(paste("Diferencia de precision:", diferencia))
```

```
## [1] "Diferencia de precision: 0"
```

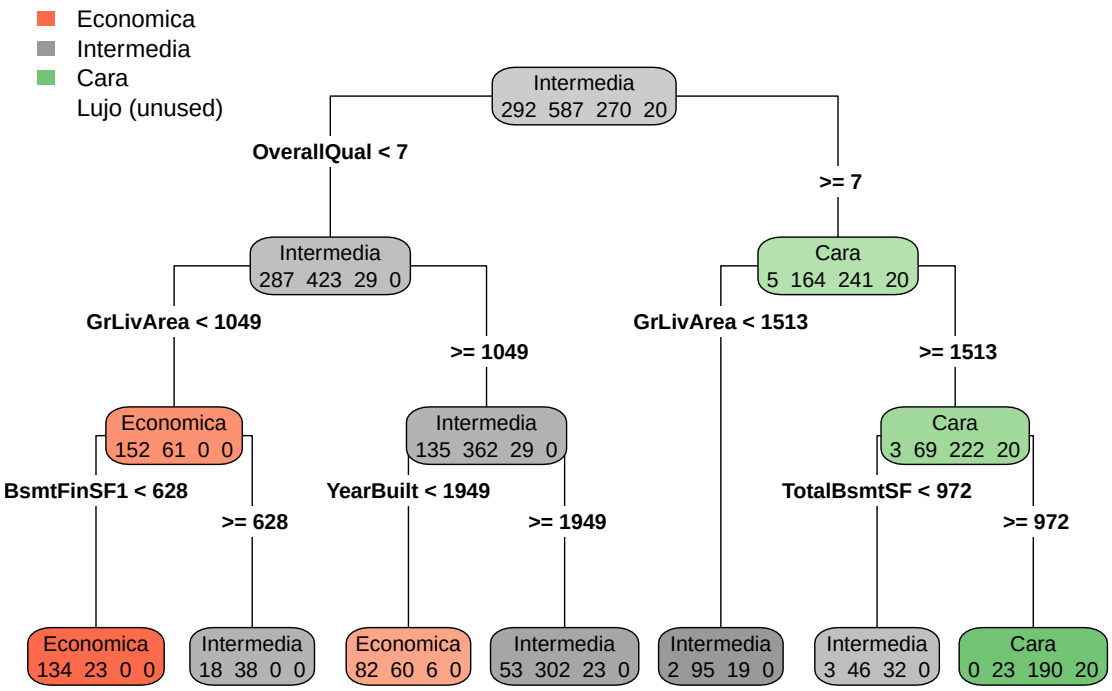
Los modelos tiene el mismmo rendimiento

11. Haga al menos, 3 modelos más, cambiando la profundidad del árbol. ¿Cuál funcionó mejor?

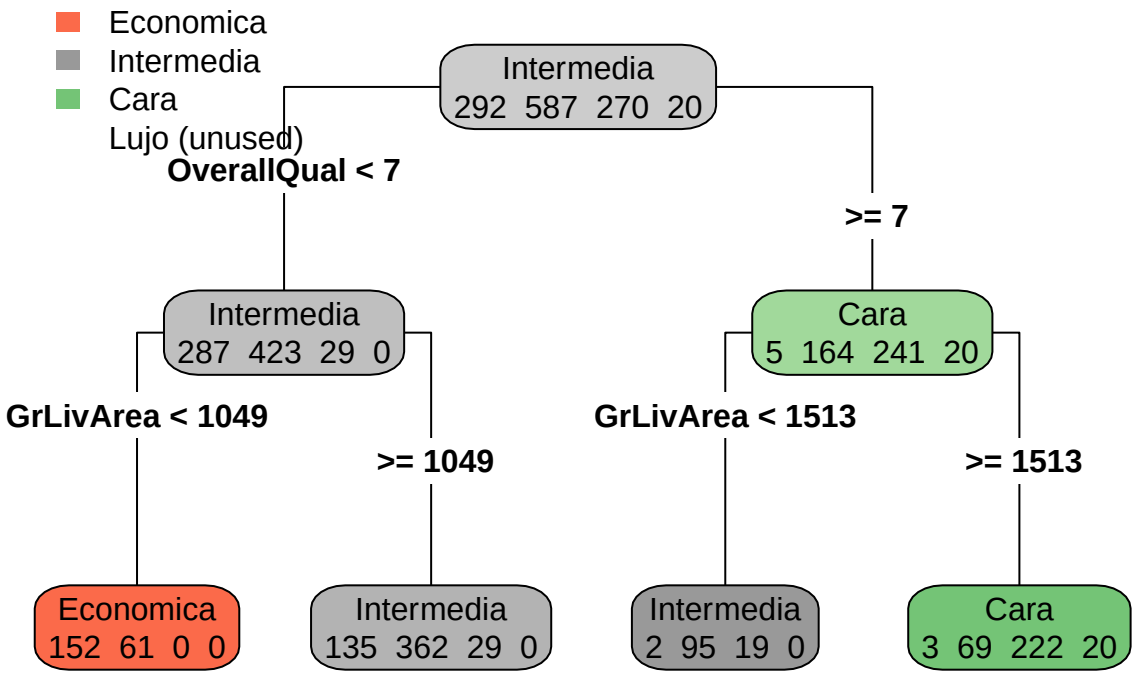
```
arbol_clasificacion1 <- rpart(precio_categoria ~ ., data = train,control = rpart.control(maxdepth = 6)
arbol_clasificacion2 <- rpart(precio_categoria ~ ., data = train,control = rpart.control(maxdepth = 3)
arbol_clasificacion3 <- rpart(precio_categoria ~ ., data = train,control = rpart.control(maxdepth = 2)

rpart.plot(arbol_clasificacion1, type = 4, extra=1)
```



```
rpart.plot(arbol_clasificacion2, type = 4, extra=1)
```




```
rpart.plot(arbol_clasificacion3, type = 4, extra=1)
```




```
errores2 <- calcular_errores(arbol_clasificacion1)
errores3 <- calcular_errores(arbol_clasificacion2)
errores4 <- calcular_errores(arbol_clasificacion3)


# Generar predicciones para el conjunto de prueba
```

```r
pred1 <- predict(arbol_clasificacion1, newdata = test_set, type = "class")
pred2 <- predict(arbol_clasificacion2, newdata = test_set, type = "class")
pred3 <- predict(arbol_clasificacion3, newdata = test_set, type = "class")

# Crear matrices de confusión
confusion1 <- confusionMatrix(pred1, test_set$precio_categoria)
confusion2 <- confusionMatrix(pred2, test_set$precio_categoria)
confusion3 <- confusionMatrix(pred3, test_set$precio_categoria)


# Mostrar las matrices de confusión
print("Matriz de Confusión para arbol_clasificacion1:")
```

```
## [1] "Matriz de Confusión para arbol_clasificacion1:"
```

```r
print(confusion1)
```

```
## 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
```

```r
print("Matriz de Confusión para arbol_clasificacion2:")
```

```
## [1] "Matriz de Confusión para arbol_clasificacion2:"
```

```r
print(confusion2)
```

```
## Confusion Matrix and Statistics
```

```
##
##             Reference
## Prediction   Economica Intermedia Cara Lujo
##   Economica         47         14    0    0
##   Intermedia        12         98   16    0
##   Cara               0          4   39    4
##   Lujo               0          0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.7863
##                  95% CI : (0.7282, 0.837)
##     No Information Rate : 0.4957
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6554
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara Class: Lujo
## Sensitivity                    0.7966            0.8448      0.7091     0.00000
## Specificity                    0.9200            0.7627      0.9553     1.00000
## Pos Pred Value                 0.7705            0.7778      0.8298         NaN
## Neg Pred Value                 0.9306            0.8333      0.9144     0.98291
## Prevalence                     0.2521            0.4957      0.2350     0.01709
## Detection Rate                 0.2009            0.4188      0.1667     0.00000
## Detection Prevalence           0.2607            0.5385      0.2009     0.00000
## Balanced Accuracy              0.8583            0.8038      0.8322     0.50000
```

```
print("Matriz de Confusión para arbol_clasificacion3:")
```

```
## [1] "Matriz de Confusión para arbol_clasificacion3:"
```

```
print(confusion3)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Economica Intermedia Cara Lujo
##   Economica         29         14    0    0
##   Intermedia        30         91   11    0
##   Cara               0         11   44    4
##   Lujo               0          0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.7009
##                  95% CI : (0.6378, 0.7588)
##     No Information Rate : 0.4957
##     P-Value [Acc > NIR] : 1.573e-10
##
##                   Kappa : 0.5134
##
```

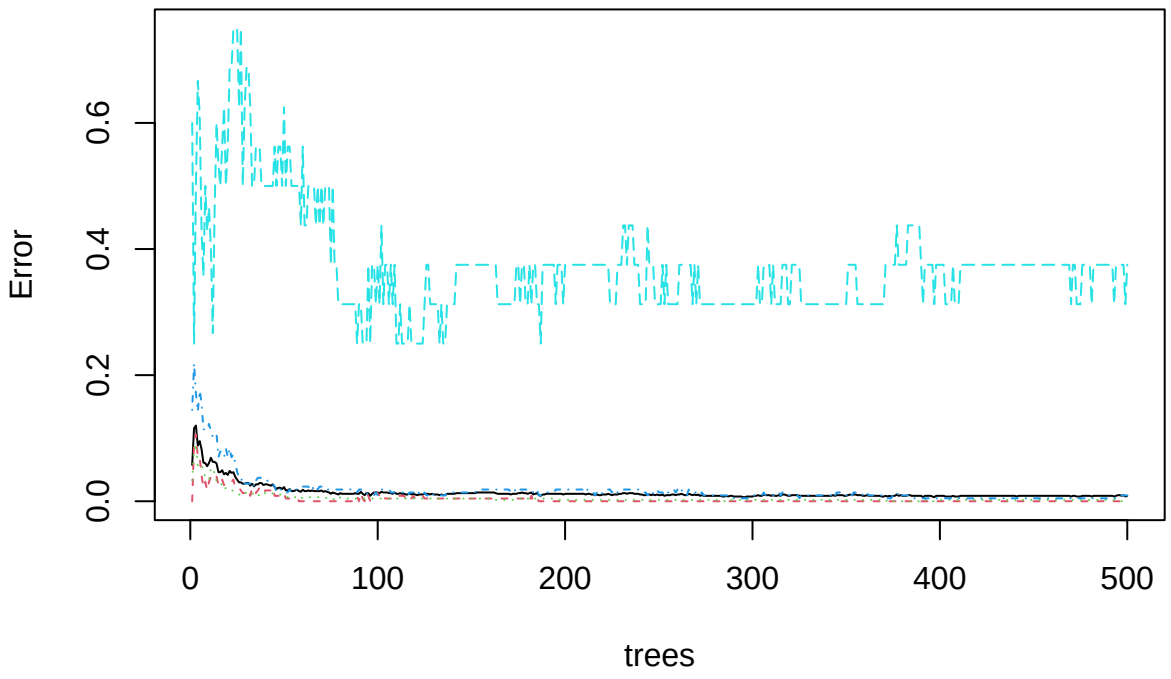
```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Economica Class: Intermedia Class: Cara Class: Lujo
## Sensitivity                    0.4915            0.7845      0.8000     0.00000
## Specificity                    0.9200            0.6525      0.9162     1.00000
## Pos Pred Value                 0.6744            0.6894      0.7458         NaN
## Neg Pred Value                 0.8429            0.7549      0.9371     0.98291
## Prevalence                     0.2521            0.4957      0.2350     0.01709
## Detection Rate                 0.1239            0.3889      0.1880     0.00000
## Detection Prevalence           0.1838            0.5641      0.2521     0.00000
## Balanced Accuracy              0.7058            0.7185      0.8581     0.50000
```

Como es de esperarce el Arbol que tienen mayor profundidad, de 6 de profundidad fue el que obtuvo mejores resultados. Sin embargo tiene una presicion de 83.76% lo que podría indicar que el modelo empieza a sobre ajustarce. Como el modelo de profundidad 3 tiene la midad de profundidad y una precicion de 0.7863% la cual es parecida con menos riesgo de sobre ajuste y con un modelo más simple. Por ello se considera al arbo de profundidad 3 como el mejor modelo para la clasificacion.

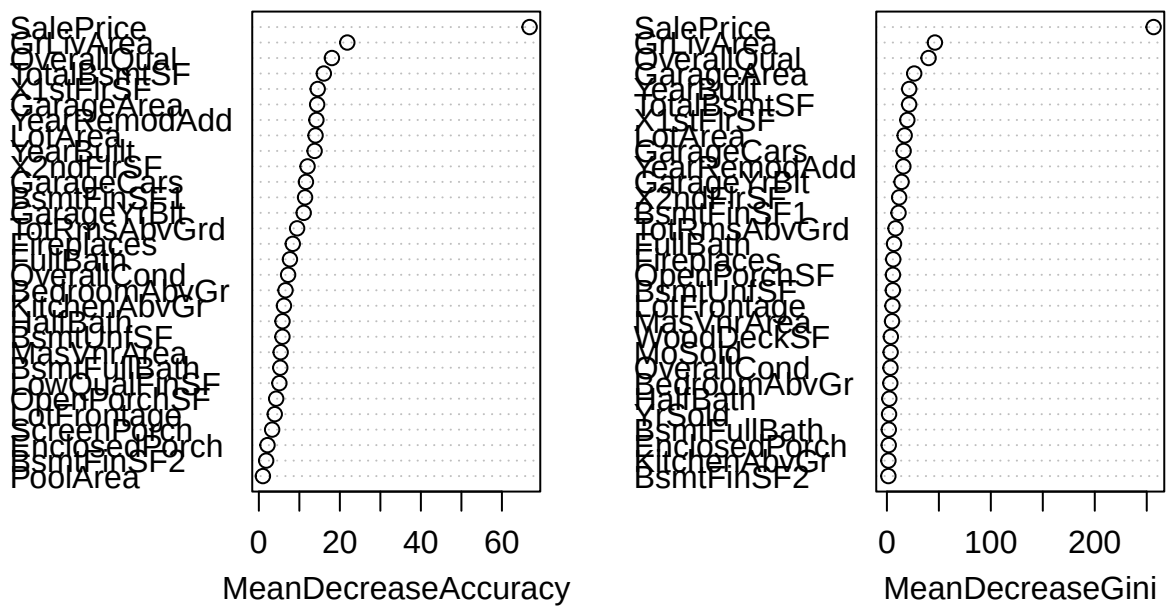
12. Repita los análisis usando random forest como algoritmo de predicción, explique sus

resultados comparando ambos algoritmos.

```r
rf_model <- randomForest(precio_categoria ~ ., data = train_set, ntree = 500, importance = TRUE)
# Suponiendo que ya has creado el modelo rf_model




# Suponiendo que ya has creado el modelo rf_model
plot(rf_model, main = "Error de OOB")
```




```r
# Suponiendo que ya has creado el modelo rf_model
varImpPlot(rf_model, main = "Importancia de Variables")
```




```r
# Realiza las predicciones sobre test_set
predictions <- predict(rf_model, newdata = test_set)
```

```r
# Genera la matriz de confusión comparando las predicciones con los valores reales
confusion_matrix <- table(Predicted = predictions, Actual = test_set$precio_categoria)
print(confusion_matrix)
```

```
##             Actual
## Predicted    Economica Intermedia Cara Lujo
##   Economica         59          0    0    0
##   Intermedia         0        116    1    0
##   Cara               0          0   54    1
##   Lujo               0          0    0    3
```

```r
# Opcional: Calcula la precisión del modelo
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Precisión:", round(accuracy * 100, 2), "%"))
```

```
## [1] "Precisión: 99.15 %"
```