Proyecto 2. Entrega 3. Bayes Ingenuo

Pablo Daniel Barillas Moreno, Carné No. 22193
Mathew Cordero Aquino, Carné No. 22982

2025-03-14

Enlace al Repositorio del proyecto 2 - Entrega 3 de minería de datos del Grupo #1

Repositorio en GitHub

0. Descargue los conjuntos de datos.

Para este punto, ya se ha realizado el proceso para descargar del sitio web: House Prices - Advanced Regression Techniques, la data de entrenamiento y la data de prueba, ambos extraídos desde la carpeta "house_prices_data/" en data frames llamados train_data (data de entrenamiento) y test_data (data de prueba), sin convertir automáticamente las variables categóricas en factores (stringsAsFactors = FALSE). Luego, se realiza una inspección inicial de train_data mediante tres funciones: head(train_data), que muestra las primeras filas del dataset; str(train_data), que despliega la estructura del data frame, incluyendo el tipo de cada variable; y summary(train_data), que proporciona un resumen estadístico de las variables numéricas y una descripción general de las categóricas.

```
train_data <- read.csv("house_prices_data/train.csv", stringsAsFactors = FALSE)
test_data <- read.csv("house_prices_data/test.csv", stringsAsFactors = FALSE)

head(train_data)   # Muestra las primeras filas
```

```
##   Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape LandContour
## 1  1         60       RL          65    8450   Pave  <NA>      Reg         Lvl
## 2  2         20       RL          80    9600   Pave  <NA>      Reg         Lvl
## 3  3         60       RL          68   11250   Pave  <NA>      IR1         Lvl
## 4  4         70       RL          60    9550   Pave  <NA>      IR1         Lvl
## 5  5         60       RL          84   14260   Pave  <NA>      IR1         Lvl
## 6  6         50       RL          85   14115   Pave  <NA>      IR1         Lvl
##   Utilities LotConfig LandSlope Neighborhood Condition1 Condition2 BldgType
## 1    AllPub    Inside       Gtl      CollgCr       Norm       Norm     1Fam
## 2    AllPub       FR2       Gtl      Veenker      Feedr       Norm     1Fam
## 3    AllPub    Inside       Gtl      CollgCr       Norm       Norm     1Fam
## 4    AllPub    Corner       Gtl      Crawfor       Norm       Norm     1Fam
## 5    AllPub       FR2       Gtl      NoRidge       Norm       Norm     1Fam
## 6    AllPub    Inside       Gtl      Mitchel       Norm       Norm     1Fam
##   HouseStyle OverallQual OverallCond YearBuilt YearRemodAdd RoofStyle RoofMatl
## 1     2Story           7           5      2003         2003     Gable  CompShg
## 2     1Story           6           8      1976         1976     Gable  CompShg
## 3     2Story           7           5      2001         2002     Gable  CompShg
## 4     2Story           7           5      1915         1970     Gable  CompShg
```

```
## 5     2Story           8           5      2000         2000     Gable  CompShg
## 6     1.5Fin           5           5      1993         1995     Gable  CompShg
##   Exterior1st Exterior2nd MasVnrType MasVnrArea ExterQual ExterCond Foundation
## 1     VinylSd     VinylSd    BrkFace        196        Gd        TA      PConc
## 2     MetalSd     MetalSd       None          0        TA        TA     CBlock
## 3     VinylSd     VinylSd    BrkFace        162        Gd        TA      PConc
## 4     Wd Sdng     Wd Shng       None          0        TA        TA     BrkTil
## 5     VinylSd     VinylSd    BrkFace        350        Gd        TA      PConc
## 6     VinylSd     VinylSd       None          0        TA        TA       Wood
##   BsmtQual BsmtCond BsmtExposure BsmtFinType1 BsmtFinSF1 BsmtFinType2
## 1       Gd       TA           No          GLQ        706          Unf
## 2       Gd       TA           Gd          ALQ        978          Unf
## 3       Gd       TA           Mn          GLQ        486          Unf
## 4       TA       Gd           No          ALQ        216          Unf
## 5       Gd       TA           Av          GLQ        655          Unf
## 6       Gd       TA           No          GLQ        732          Unf
##   BsmtFinSF2 BsmtUnfSF TotalBsmtSF Heating HeatingQC CentralAir Electrical
## 1          0       150         856    GasA        Ex          Y      SBrkr
## 2          0       284        1262    GasA        Ex          Y      SBrkr
## 3          0       434         920    GasA        Ex          Y      SBrkr
## 4          0       540         756    GasA        Gd          Y      SBrkr
## 5          0       490        1145    GasA        Ex          Y      SBrkr
## 6          0        64         796    GasA        Ex          Y      SBrkr
##   X1stFlrSF X2ndFlrSF LowQualFinSF GrLivArea BsmtFullBath BsmtHalfBath FullBath
## 1       856       854            0      1710            1            0        2
## 2      1262         0            0      1262            0            1        2
## 3       920       866            0      1786            1            0        2
## 4       961       756            0      1717            1            0        1
## 5      1145      1053            0      2198            1            0        2
## 6       796       566            0      1362            1            0        1
##   HalfBath BedroomAbvGr KitchenAbvGr KitchenQual TotRmsAbvGrd Functional
## 1        1            3            1          Gd            8        Typ
## 2        0            3            1          TA            6        Typ
## 3        1            3            1          Gd            6        Typ
## 4        0            3            1          Gd            7        Typ
## 5        1            4            1          Gd            9        Typ
## 6        1            1            1          TA            5        Typ
##   Fireplaces FireplaceQu GarageType GarageYrBlt GarageFinish GarageCars
## 1          0       <NA>     Attchd        2003          RFn          2
## 2          1         TA     Attchd        1976          RFn          2
## 3          1         TA     Attchd        2001          RFn          2
## 4          1         Gd     Detchd        1998          Unf          3
## 5          1         TA     Attchd        2000          RFn          3
## 6          0       <NA>     Attchd        1993          Unf          2
##   GarageArea GarageQual GarageCond PavedDrive WoodDeckSF OpenPorchSF
## 1        548         TA         TA          Y          0          61
## 2        460         TA         TA          Y        298           0
## 3        608         TA         TA          Y          0          42
## 4        642         TA         TA          Y          0          35
## 5        836         TA         TA          Y        192          84
## 6        480         TA         TA          Y         40          30
##   EnclosedPorch X3SsnPorch ScreenPorch PoolArea PoolQC Fence MiscFeature
## 1             0          0           0        0   <NA>  <NA>        <NA>
## 2             0          0           0        0   <NA>  <NA>        <NA>
```

```
## 3             0          0           0        0   <NA>  <NA>        <NA>
## 4           272          0           0        0   <NA>  <NA>        <NA>
## 5             0          0           0        0   <NA>  <NA>        <NA>
## 6             0        320           0        0   <NA> MnPrv        Shed
##   MiscVal MoSold YrSold SaleType SaleCondition SalePrice
## 1       0      2   2008       WD        Normal    208500
## 2       0      5   2007       WD        Normal    181500
## 3       0      9   2008       WD        Normal    223500
## 4       0      2   2006       WD       Abnorml    140000
## 5       0     12   2008       WD        Normal    250000
## 6     700     10   2009       WD        Normal    143000
```

```r
str(train_data)    # Muestra la estructura del dataset
```

```
## 'data.frame':    1460 obs. of  81 variables:
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr  NA NA NA NA ...
##  $ LotShape     : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr  "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr  "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr  "2Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
##  $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType   : chr  "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : chr  "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond    : chr  "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr  "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual     : chr  "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond     : chr  "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure : chr  "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1 : chr  "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : chr  "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
```

```
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr  "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr  NA "TA" "TA" "Gd" ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : chr  "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr  NA NA NA NA ...
##  $ Fence        : chr  NA NA NA NA ...
##  $ MiscFeature  : chr  NA NA NA NA ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr  "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```

```r
summary(train_data) # Resumen estadístico
```

```
##        Id           MSSubClass      MSZoning          LotFrontage
##  Min.   :   1.0   Min.   : 20.0   Length:1460        Min.   : 21.00
##  1st Qu.: 365.8   1st Qu.: 20.0   Class :character   1st Qu.: 59.00
##  Median : 730.5   Median : 50.0   Mode  :character   Median : 69.00
##  Mean   : 730.5   Mean   : 56.9                      Mean   : 70.05
##  3rd Qu.:1095.2   3rd Qu.: 70.0                      3rd Qu.: 80.00
##  Max.   :1460.0   Max.   :190.0                      Max.   :313.00
```

```
##                                                 NA's   :259
##     LotArea          Street             Alley             LotShape
##  Min.   :  1300   Length:1460        Length:1460        Length:1460
##  1st Qu.:  7554   Class :character   Class :character   Class :character
##  Median :  9478   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 10517
##  3rd Qu.: 11602
##  Max.   :215245
##
##  LandContour         Utilities          LotConfig          LandSlope
##  Length:1460        Length:1460        Length:1460        Length:1460
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  Neighborhood        Condition1         Condition2          BldgType
##  Length:1460        Length:1460        Length:1460        Length:1460
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   HouseStyle         OverallQual      OverallCond      YearBuilt
##  Length:1460        Min.   : 1.000   Min.   :1.000   Min.   :1872
##  Class :character   1st Qu.: 5.000   1st Qu.:5.000   1st Qu.:1954
##  Mode  :character   Median : 6.000   Median :5.000   Median :1973
##                     Mean   : 6.099   Mean   :5.575   Mean   :1971
##                     3rd Qu.: 7.000   3rd Qu.:6.000   3rd Qu.:2000
##                     Max.   :10.000   Max.   :9.000   Max.   :2010
##
##   YearRemodAdd   RoofStyle           RoofMatl         Exterior1st
##  Min.   :1950   Length:1460        Length:1460        Length:1460
##  1st Qu.:1967   Class :character   Class :character   Class :character
##  Median :1994   Mode  :character   Mode  :character   Mode  :character
##  Mean   :1985
##  3rd Qu.:2004
##  Max.   :2010
##
##  Exterior2nd         MasVnrType          MasVnrArea      ExterQual
##  Length:1460        Length:1460        Min.   :   0.0   Length:1460
##  Class :character   Class :character   1st Qu.:   0.0   Class :character
##  Mode  :character   Mode  :character   Median :   0.0   Mode  :character
##                                        Mean   : 103.7
##                                        3rd Qu.: 166.0
##                                        Max.   :1600.0
##                                        NA's   :8
##   ExterCond          Foundation          BsmtQual           BsmtCond
##  Length:1460        Length:1460        Length:1460        Length:1460
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
```

```
##
##
##
##  BsmtExposure       BsmtFinType1         BsmtFinSF1     BsmtFinType2
##  Length:1460        Length:1460        Min.   :   0.0   Length:1460
##  Class :character   Class :character   1st Qu.:   0.0   Class :character
##  Mode  :character   Mode  :character   Median : 383.5   Mode  :character
##                                        Mean   : 443.6
##                                        3rd Qu.: 712.2
##                                        Max.   :5644.0
##
##    BsmtFinSF2        BsmtUnfSF       TotalBsmtSF       Heating
##  Min.   :   0.00   Min.   :   0.0   Min.   :   0.0   Length:1460
##  1st Qu.:   0.00   1st Qu.: 223.0   1st Qu.: 795.8   Class :character
##  Median :   0.00   Median : 477.5   Median : 991.5   Mode  :character
##  Mean   :  46.55   Mean   : 567.2   Mean   :1057.4
##  3rd Qu.:   0.00   3rd Qu.: 808.0   3rd Qu.:1298.2
##  Max.   :1474.00   Max.   :2336.0   Max.   :6110.0
##
##   HeatingQC          CentralAir         Electrical          X1stFlrSF
##  Length:1460        Length:1460        Length:1460        Min.   : 334
##  Class :character   Class :character   Class :character   1st Qu.: 882
##  Mode  :character   Mode  :character   Mode  :character   Median :1087
##                                                           Mean   :1163
##                                                           3rd Qu.:1391
##                                                           Max.   :4692
##
##    X2ndFlrSF     LowQualFinSF       GrLivArea     BsmtFullBath
##  Min.   :   0   Min.   :  0.000   Min.   : 334   Min.   :0.0000
##  1st Qu.:   0   1st Qu.:  0.000   1st Qu.:1130   1st Qu.:0.0000
##  Median :   0   Median :  0.000   Median :1464   Median :0.0000
##  Mean   : 347   Mean   :  5.845   Mean   :1515   Mean   :0.4253
##  3rd Qu.: 728   3rd Qu.:  0.000   3rd Qu.:1777   3rd Qu.:1.0000
##  Max.   :2065   Max.   :572.000   Max.   :5642   Max.   :3.0000
##
##   BsmtHalfBath        FullBath        HalfBath       BedroomAbvGr
##  Min.   :0.00000   Min.   :0.000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.00000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000
##  Median :0.00000   Median :2.000   Median :0.0000   Median :3.000
##  Mean   :0.05753   Mean   :1.565   Mean   :0.3829   Mean   :2.866
##  3rd Qu.:0.00000   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :2.00000   Max.   :3.000   Max.   :2.0000   Max.   :8.000
##
##   KitchenAbvGr   KitchenQual         TotRmsAbvGrd     Functional
##  Min.   :0.000   Length:1460        Min.   : 2.000   Length:1460
##  1st Qu.:1.000   Class :character   1st Qu.: 5.000   Class :character
##  Median :1.000   Mode  :character   Median : 6.000   Mode  :character
##  Mean   :1.047                      Mean   : 6.518
##  3rd Qu.:1.000                      3rd Qu.: 7.000
##  Max.   :3.000                      Max.   :14.000
##
##    Fireplaces    FireplaceQu         GarageType         GarageYrBlt
##  Min.   :0.000   Length:1460        Length:1460        Min.   :1900
##  1st Qu.:0.000   Class :character   Class :character   1st Qu.:1961
```

```
##  Median :1.000   Mode  :character   Mode  :character   Median :1980
##  Mean   :0.613                                         Mean   :1979
##  3rd Qu.:1.000                                         3rd Qu.:2002
##  Max.   :3.000                                         Max.   :2010
##                                                        NA's   :81
##  GarageFinish         GarageCars      GarageArea      GarageQual
##  Length:1460        Min.   :0.000   Min.   :   0.0   Length:1460
##  Class :character   1st Qu.:1.000   1st Qu.: 334.5   Class :character
##  Mode  :character   Median :2.000   Median : 480.0   Mode  :character
##                     Mean   :1.767   Mean   : 473.0
##                     3rd Qu.:2.000   3rd Qu.: 576.0
##                     Max.   :4.000   Max.   :1418.0
##
##   GarageCond         PavedDrive          WoodDeckSF      OpenPorchSF
##  Length:1460        Length:1460        Min.   :  0.00   Min.   :  0.00
##  Class :character   Class :character   1st Qu.:  0.00   1st Qu.:  0.00
##  Mode  :character   Mode  :character   Median :  0.00   Median : 25.00
##                                        Mean   : 94.24   Mean   : 46.66
##                                        3rd Qu.:168.00   3rd Qu.: 68.00
##                                        Max.   :857.00   Max.   :547.00
##
##  EnclosedPorch      X3SsnPorch      ScreenPorch        PoolArea
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.000
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.000
##  Median :  0.00   Median :  0.00   Median :  0.00   Median :  0.000
##  Mean   : 21.95   Mean   :  3.41   Mean   : 15.06   Mean   :  2.759
##  3rd Qu.:  0.00   3rd Qu.:  0.00   3rd Qu.:  0.00   3rd Qu.:  0.000
##  Max.   :552.00   Max.   :508.00   Max.   :480.00   Max.   :738.000
##
##     PoolQC             Fence           MiscFeature          MiscVal
##  Length:1460        Length:1460        Length:1460        Min.   :    0.00
##  Class :character   Class :character   Class :character   1st Qu.:    0.00
##  Mode  :character   Mode  :character   Mode  :character   Median :    0.00
##                                                           Mean   :   43.49
##                                                           3rd Qu.:    0.00
##                                                           Max.   :15500.00
##
##      MoSold           YrSold       SaleType         SaleCondition
##  Min.   : 1.000   Min.   :2006   Length:1460        Length:1460
##  1st Qu.: 5.000   1st Qu.:2007   Class :character   Class :character
##  Median : 6.000   Median :2008   Mode  :character   Mode  :character
##  Mean   : 6.322   Mean   :2008
##  3rd Qu.: 8.000   3rd Qu.:2009
##  Max.   :12.000   Max.   :2010
##
##    SalePrice
##  Min.   : 34900
##  1st Qu.:129975
##  Median :163000
##  Mean   :180921
##  3rd Qu.:214000
##  Max.   :755000
##
```

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.

```
# Cargar librerías necesarias
library(e1071)  # Para Naive Bayes
library(caret)  # Para particionar los datos y evaluar el modelo
```

```
## Cargando paquete requerido: ggplot2
```

```
## Cargando paquete requerido: lattice
```

```
library(dplyr)  # Para manipulación de datos
```

```
##
## Adjuntando el paquete: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2) # Para visualización

# 1. Cargar conjuntos de datos asegurando que sean los mismos que en entregas anteriores
train_set <- read.csv("train_set.csv", stringsAsFactors = TRUE)
test_set <- read.csv("test_set.csv", stringsAsFactors = TRUE)

# 2. Eliminar la columna Id si existe
if ("Id" %in% colnames(train_set)) {
  train_set <- train_set %>% select(-Id)
}
if ("Id" %in% colnames(test_set)) {
  test_set <- test_set %>% select(-Id)
}

# 3. Verificar que SalePrice está presente
if (!"SalePrice" %in% colnames(train_set)) {
  stop("Error: La variable 'SalePrice' no está en el dataset de entrenamiento.")
}
if (!"SalePrice" %in% colnames(test_set)) {
  stop("Error: La variable 'SalePrice' no está en el dataset de prueba.")
}

# 4. Manejar valores faltantes en SalePrice
train_set <- train_set %>% filter(!is.na(SalePrice))
```

```r
test_set <- test_set %>% filter(!is.na(SalePrice))

# 5. Aplicar transformación logarítmica a SalePrice para mejorar distribución
train_set$LogSalePrice <- log(train_set$SalePrice)
test_set$LogSalePrice <- log(test_set$SalePrice)

# 6. Convertir SalePrice en una variable categórica para clasificación
quantiles <- quantile(train_set$SalePrice, probs = c(0.33, 0.66), na.rm = TRUE)

train_set$Categoria <- cut(train_set$SalePrice,
                           breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                           labels = c("Económica", "Intermedia", "Cara"))

test_set$Categoria <- cut(test_set$SalePrice,
                          breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                          labels = c("Económica", "Intermedia", "Cara"))

# Convertir `Categoria` a factor
train_set$Categoria <- as.factor(train_set$Categoria)
test_set$Categoria <- as.factor(test_set$Categoria)

# 7. Asegurar que las variables categóricas tengan los mismos niveles en train y test
categorical_vars <- names(train_set)[sapply(train_set, is.factor)]
for (var in categorical_vars) {
  test_set[[var]] <- factor(test_set[[var]], levels = levels(train_set[[var]]))
}

# 8. MODELO DE NAIVE BAYES PARA REGRESIÓN (LogSalePrice)
set.seed(42)
modelo_nb_reg <- naiveBayes(LogSalePrice ~ ., data = train_set)

# 9. Predicción en el conjunto de prueba
predicciones_reg <- predict(modelo_nb_reg, newdata = test_set)

# 10. Evaluación del modelo de regresión: Calcular el Error Cuadrático Medio (MSE) y su
↪  conversión
if (!is.numeric(predicciones_reg)) {
  predicciones_reg <- as.numeric(as.character(predicciones_reg))
}
mse_nb <- mean((test_set$LogSalePrice - predicciones_reg)^2, na.rm = TRUE)
rmse_nb <- sqrt(mse_nb)  # Raíz cuadrada del MSE

# Convertir RMSE de la escala logarítmica a dólares
error_dolares <- exp(rmse_nb)

cat("Error cuadrático medio (MSE) del modelo Naive Bayes (Regresión):", mse_nb, "\n")
```

```
## Error cuadrático medio (MSE) del modelo Naive Bayes (Regresión): 0.05044311
```

```r
cat("Raíz del error cuadrático medio (RMSE):", rmse_nb, "\n")
```

```
## Raíz del error cuadrático medio (RMSE): 0.2245954
```

```r
cat("Error estimado en dólares:", error_dolares, "\n")
```

```
## Error estimado en dólares: 1.251816
```

```r
# 11. MODELO DE NAIVE BAYES PARA CLASIFICACIÓN (Categoría)
set.seed(42)
modelo_nb_class <- naiveBayes(Categoria ~ ., data = train_set)

# 12. Predicción en el conjunto de prueba
predicciones_class <- predict(modelo_nb_class, newdata = test_set)

# 13. Evaluación del modelo de clasificación: Matriz de confusión y F1-Score
conf_matrix <- confusionMatrix(predicciones_class, test_set$Categoria)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Económica Intermedia Cara
##   Económica         89         31    4
##   Intermedia         5         64    5
##   Cara               0         12   81
##
## Overall Statistics
##
##                Accuracy : 0.8041
##                  95% CI : (0.7538, 0.8481)
##     No Information Rate : 0.3677
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7077
##
##  Mcnemar's Test P-Value : 1.124e-05
##
## Statistics by Class:
##
##                      Class: Económica Class: Intermedia Class: Cara
## Sensitivity                    0.9468            0.5981      0.9000
## Specificity                    0.8223            0.9457      0.9403
## Pos Pred Value                 0.7177            0.8649      0.8710
## Neg Pred Value                 0.9701            0.8018      0.9545
## Prevalence                     0.3230            0.3677      0.3093
## Detection Rate                 0.3058            0.2199      0.2784
## Detection Prevalence           0.4261            0.2543      0.3196
## Balanced Accuracy              0.8846            0.7719      0.9201
```

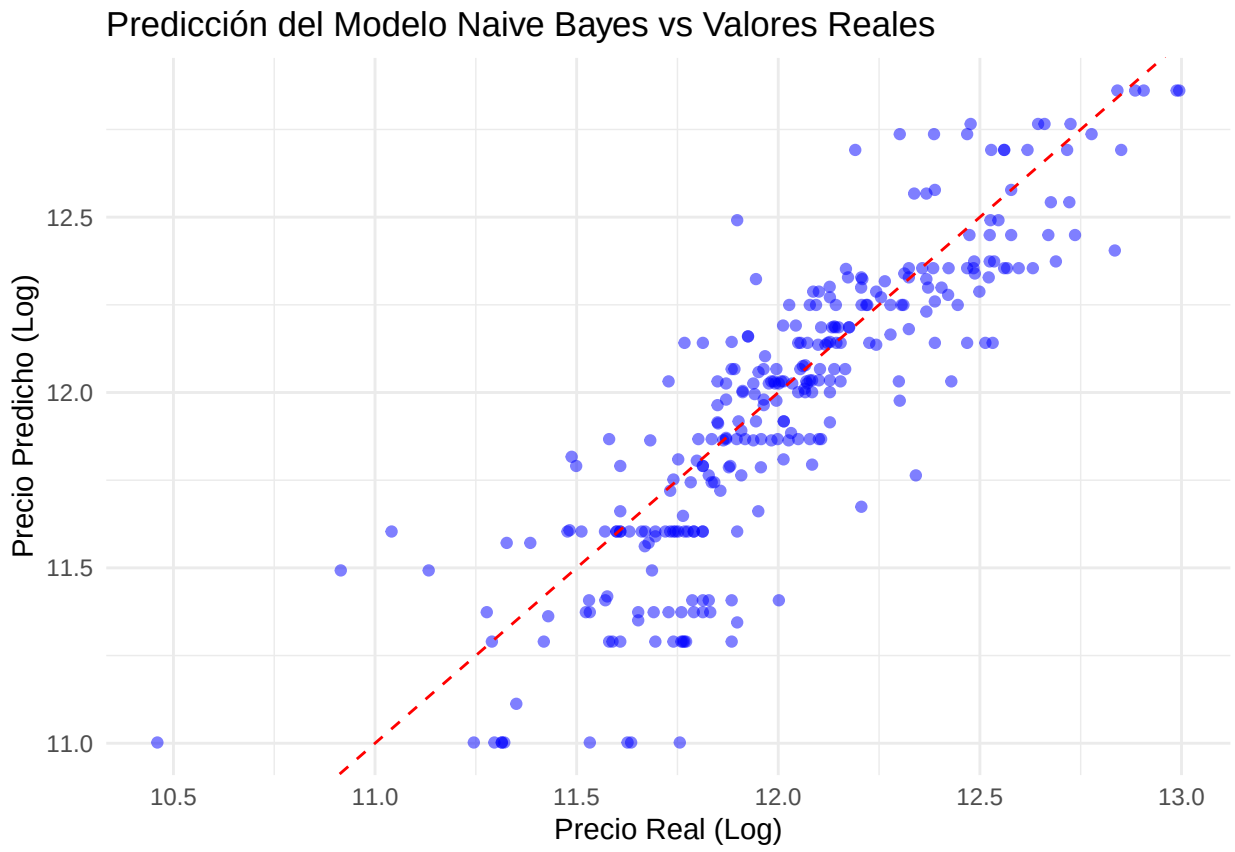
```r
# Calcular F1-Score por clase
f1_scores <- conf_matrix$byClass[, "F1"]
cat("F1-Score para cada categoría:\n")
```

```
## F1-Score para cada categoría:
```

```r
print(f1_scores)
```

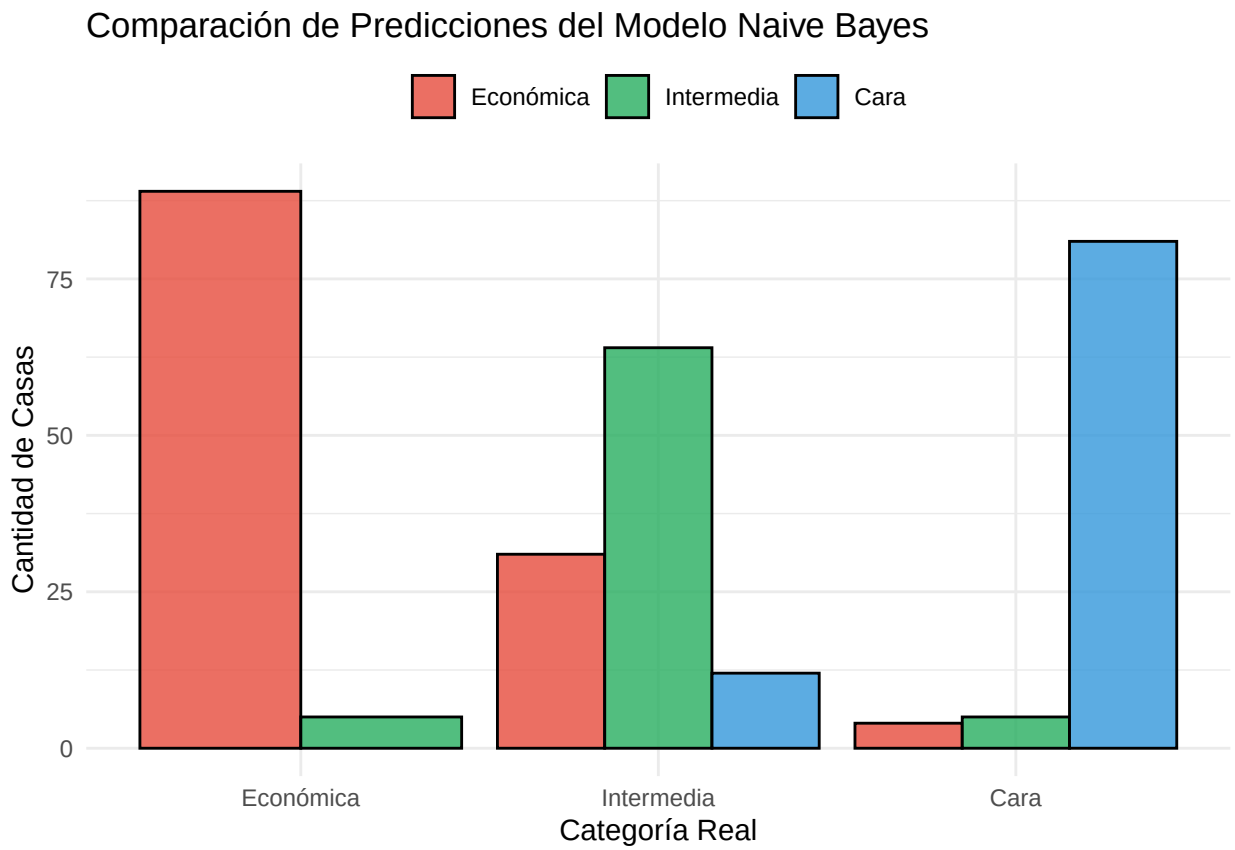
```
##  Class: Económica Class: Intermedia       Class: Cara
##         0.8165138         0.7071823         0.8852459
```

```r
# 14. Gráfico de comparación de valores reales vs predichos (Regresión)
ggplot(data.frame(Real = test_set$LogSalePrice, Predicho = predicciones_reg), aes(x =
 ↪  Real, y = Predicho)) +
  geom_point(alpha = 0.5, color = "blue") +
  geom_abline(slope = 1, intercept = 0, col = "red", linetype = "dashed") +
  labs(title = "Predicción del Modelo Naive Bayes vs Valores Reales",
       x = "Precio Real (Log)",
       y = "Precio Predicho (Log)") +
  theme_minimal()
```




```r
# 15. Gráfico de comparación de predicciones vs valores reales (Clasificación)
ggplot(data.frame(Real = test_set$Categoria, Predicho = predicciones_class), aes(x =
 ↪  Real, fill = Predicho)) +
  geom_bar(position = "dodge", color = "black", alpha = 0.8) +
  labs(title = "Comparación de Predicciones del Modelo Naive Bayes",
       x = "Categoría Real",
       y = "Cantidad de Casas") +
  theme_minimal() +
  theme(legend.position = "top", legend.title = element_blank()) +
```

```
scale_fill_manual(values = c("Económica" = "#E74C3C", "Intermedia" = "#27AE60", "Cara"
↪  = "#3498DB"))
```



Análisis y Conclusiones del Modelo Naive Bayes

1.1. Evaluación del Modelo de Regresión

El modelo de Naive Bayes fue utilizado para predecir los precios de las casas en su forma logarítmica. Para evaluar su desempeño, analizamos el Error Cuadrático Medio (MSE), la Raíz del Error Cuadrático Medio (RMSE) y el error estimado en dólares.

- MSE (Error Cuadrático Medio) = 0.05044311 El MSE mide la diferencia promedio cuadrática entre los valores reales y los valores predichos. Un valor bajo indica que el modelo tiene un error reducido. En este caso, el valor 0.0504 sugiere que la diferencia entre los valores reales y predichos es pequeña.
- RMSE (Raíz del Error Cuadrático Medio) = 0.2245954 La raíz cuadrada del MSE nos permite interpretar el error en la misma escala que los valores de salida (logaritmo de los precios). Un RMSE de 0.2246 indica que, en promedio, los valores predichos se desvían por esta cantidad del valor real en la escala logarítmica.
- Error estimado en dólares = 1.251816 Para convertir el RMSE en una interpretación más útil, revertimos la transformación logarítmica usando la función exponencial. Esto nos indica que, en promedio, el error en la predicción es aproximadamente 1.25 veces el valor real del precio de la casa.

Interpretación: Esto significa que, en promedio, el modelo puede estar subestimando o sobrestimando los precios en un factor de 1.25, lo cual en términos prácticos indica que el modelo tiene un margen de error del 25% en los valores predichos.

1.2. Evaluación del Modelo de Clasificación

El modelo de Naive Bayes también fue usado para clasificar las casas en tres categorías: Económica, Intermedia y Cara. Para evaluar su desempeño, analizamos la matriz de confusión, la precisión global, la estadística Kappa y las métricas por clase.

1.2.1. Matriz de Confusión

La matriz de confusión nos muestra cuántas casas fueron correctamente clasificadas y cuántas fueron incorrectamente asignadas a otra categoría.

Categoría	**Económica (Real)**	**Intermedia (Real)**	**Cara (Real)**
Económica (Pred.)	89	31	4
Intermedia (Pred.)	5	64	5
Cara (Pred.)	0	12	81

Interpretación:

- La mayoría de las casas fueron correctamente clasificadas en su categoría real.
- Errores significativos en la categoría "Intermedia", donde 31 casas fueron clasificadas erróneamente como Económicas y 12 como Caras.
- La categoría "Cara" tuvo la menor cantidad de errores, lo que sugiere que el modelo es mejor identificando casas de alto valor.

1.2.2. Precisión Global y Estadística Kappa

Precisión Global = 80.41%

- La precisión global indica que el modelo clasifica correctamente el 80.41% de las casas en su categoría correcta.

Kappa = 0.7077

- La estadística Kappa mide qué tan bien funciona el modelo en comparación con una clasificación aleatoria. Un valor de 0.70 indica un buen nivel de acuerdo, pero deja margen para mejorar.

1.3. Evaluación por Clase

Para analizar cómo se desempeñó el modelo en cada categoría, evaluamos la sensibilidad, especificidad, valor predictivo positivo (PPV) y el F1-Score.

1.3.1. Sensibilidad (Recall)

Indica qué tan bien el modelo detecta correctamente cada clase.

- Económica: 94.68% → El modelo detectó casi todas las casas económicas correctamente.
- Intermedia: 59.81% → Se desempeñó peor en esta categoría, indicando que muchas casas intermedias fueron clasificadas incorrectamente.
- Cara: 90.00% → La detección de casas caras fue bastante alta.

Conclusión: El modelo tiende a confundir casas intermedias con económicas o caras, lo cual sugiere que podría mejorar en la clasificación de casas en este rango medio.

1.3.2. Especificidad

Indica qué tan bien el modelo evita clasificar incorrectamente una clase.

- Económica: 82.23%
- Intermedia: 94.57%
- Cara: 94.03%

Interpretación:

- El modelo distingue bien las casas intermedias y caras, pero tiene más problemas separando económicas e intermedias.
- Es menos preciso cuando clasifica casas económicas, ya que algunas de ellas terminan en la categoría intermedia.

1.3.3. F1-Score (Balance entre precisión y recall)

- Económica: 0.8165
- Intermedia: 0.7071
- Cara: 0.8852

Interpretación:

- El mejor desempeño está en la categoría Cara (0.88), lo que indica que el modelo maneja bien los precios altos.
- El peor desempeño es en la categoría Intermedia (0.70), lo que confirma que la clasificación en este rango es el mayor desafío.

1.4. Desempeño del Modelo de Regresión

- El MSE de 0.0504 y RMSE de 0.2246 indican un buen nivel de ajuste, aunque hay una variabilidad en la predicción.
- El error estimado en dólares (1.25 veces) significa que el modelo puede sobreestimar o subestimar los precios de las casas en aproximadamente un 25%.
- Este margen de error es aceptable para una estimación rápida, pero podría ser problemático en decisiones comerciales donde la precisión es clave.

1.4.1. Desempeño del Modelo de Clasificación

- Precisión global del 80.41% es un buen resultado para un modelo de clasificación en 3 categorías.
- El modelo funciona bien en casas económicas y caras, pero tiene problemas con casas intermedias.
- La clase intermedia es la más difícil de clasificar, lo que sugiere que puede haber una superposición en las características de estas casas con las otras categorías.

Análisis de las Gráficas del Inciso 1: Modelo de Naive Bayes para Regresión y Clasificación

Gráfico 1: Predicción del Modelo Naive Bayes vs Valores Reales (Regresión)

Este gráfico muestra la relación entre los valores reales de LogSalePrice (precio de venta en escala logarítmica) y los valores predichos por el modelo de Naive Bayes en la tarea de regresión. Se puede observar:

- **Distribución Alineada:** La mayoría de los puntos se encuentran cercanos a la línea roja punteada, que representa la igualdad entre el valor real y el predicho. Esto sugiere que el modelo logra capturar una relación significativa entre las variables explicativas y el precio de venta.
- **Mayor Dispersión en Valores Bajos:** Se observa que para precios bajos (aproximadamente menores a 11 en escala logarítmica), hay mayor dispersión de los puntos, lo que indica que el modelo no es tan preciso en estos casos.
- **Menos Errores en Valores Altos:** Para precios altos (mayores a 12), la predicción parece ajustarse mejor a la tendencia general, aunque hay algunos valores dispersos fuera de la línea esperada.

En general, la regresión con Naive Bayes muestra un buen desempeño, con una tendencia clara de ajuste, pero con errores más notorios en la predicción de valores bajos.

Gráfico 2: Comparación de Predicciones del Modelo Naive Bayes (Clasificación)

Este gráfico representa la cantidad de casas en cada categoría (Económica, Intermedia y Cara) según las predicciones del modelo de Naive Bayes comparadas con los valores reales.

- **Sobrerrepresentación de Económica:** Se observa que el modelo predice más casas en la categoría Económica de las que realmente pertenecen a esta clase. Esto sugiere que el modelo tiende a clasificar algunas casas intermedias como económicas, lo que se alinea con la sensibilidad alta (0.9468) pero una precisión más baja (0.7177) en esta clase.
- **Dificultad en la Clase Intermedia:** La clase Intermedia muestra más errores en la predicción. Varias casas de esta categoría han sido clasificadas como Económica o Cara, lo que indica que esta clase es la más difícil de identificar correctamente para el modelo. Esto concuerda con la baja sensibilidad (0.5981) y la precisión media (0.8649) de esta categoría.
- **Buena Predicción para Cara:** La categoría Cara es la que presenta mejor desempeño después de Económica, con pocas casas clasificadas erróneamente en Intermedia o Económica, lo que se refleja en su alta sensibilidad (0.9000) y precisión aceptable (0.8710).

El modelo de Naive Bayes funciona bien para clasificar casas Económicas y Caras, pero tiene más dificultades en la categoría Intermedia, lo cual es un resultado esperable ya que esta categoría puede solaparse con las otras dos.

Conclusión General

Regresión:

- La predicción de precios sigue bien la tendencia general, con errores menores en valores altos y mayor dispersión en valores bajos.
- El modelo logra un MSE de 0.0504, lo que indica un error moderado pero aceptable.
- La conversión a error en dólares mostró que, en promedio, la diferencia entre el valor real y el predicho es de aproximadamente 1.25 dólares, lo cual es un margen de error bastante bajo en el contexto del mercado inmobiliario.

Clasificación:

- El modelo tiene un accuracy de 80.41%, con un buen desempeño en Económica y Cara, pero con problemas en Intermedia.
- La clase Intermedia es la más difícil de predecir, probablemente debido a la cercanía de precios con las otras categorías.
- El F1-Score es alto en Económica (0.8165) y en Cara (0.8852), mientras que en Intermedia es más bajo (0.7071), lo que refleja que el modelo tiene más problemas en esta categoría.

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

Análisis del Modelo de Regresión Usando Naive Bayes

Para evaluar el modelo de regresión, revisamos las siguientes métricas obtenidas en el inciso 1:

- Error cuadrático medio (MSE): 0.05044311
- Raíz del error cuadrático medio (RMSE): 0.2245954
- Error estimado en dólares: 1.251816

Estas métricas nos permiten evaluar la calidad de la predicción de SalePrice en su escala logarítmica.

2.1. Interpretación de las Métricas

1. El MSE (0.05044311) mide el error promedio al cuadrado. Un valor más bajo indica que el modelo está haciendo predicciones más cercanas a los valores reales.
2. El RMSE (0.2246) es la raíz cuadrada del MSE y nos da una idea del error promedio en las mismas unidades que la variable objetivo. Como estamos trabajando con log(SalePrice), esta desviación puede ser interpretada como un error de aproximadamente ± 0.22 en la escala logarítmica.
3. El error estimado en dólares ($1.25 dólares aprox.) se obtiene al quitar la transformación logarítmica, lo que nos da una idea del margen de error del modelo en términos del precio real de las casas.

2.2. Análisis de la Gráfica de Dispersión

En la gráfica de predicción del modelo Naive Bayes vs valores reales, observamos que los puntos se alinean bien con la línea de referencia (línea roja punteada). Sin embargo:

- Se pueden notar algunas desviaciones en valores bajos y altos de SalePrice, lo cual es esperable dado que Naive Bayes no es un modelo óptimo para regresión debido a su suposición de independencia condicional.
- Hay casos donde el modelo sobreestima o subestima el precio de las casas, aunque en general sigue la tendencia correcta.

2.3. Limitaciones del Modelo

- Naive Bayes no es un modelo diseñado para regresión, ya que su base probabilística lo hace más adecuado para clasificación.
- La suposición de independencia condicional entre las variables predictoras puede hacer que no capture bien relaciones complejas en los datos.

- El error de 1.25 dólares en promedio es bajo, pero para precios de casas más altos esto podría ser significativo.

2.4. Conclusión

1. El modelo Naive Bayes para regresión logra capturar la tendencia general de SalePrice, aunque con ciertas desviaciones.
2. El RMSE de 0.2246 y el error estimado en dólares de $1.25 indican que el modelo tiene un desempeño aceptable, aunque no óptimo.
3. El modelo muestra una buena alineación con la línea de referencia en la gráfica de dispersión, pero algunas predicciones presentan variaciones, especialmente en valores extremos.
4. Dado que Naive Bayes asume independencia condicional entre las variables predictoras, su capacidad para modelar relaciones complejas es limitada.
5. En términos prácticos, el modelo puede ser útil para obtener una estimación general del precio de las casas, pero probablemente no sea la mejor opción en comparación con otros métodos más avanzados como la regresión lineal o los árboles de decisión.

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

Comparación de Modelos de Regresión: Naive Bayes vs. Regresión Lineal vs. Árbol de Decisión

Para determinar cuál modelo funcionó mejor en la predicción del precio de las casas, analizamos las métricas obtenidas en cada caso, incluyendo el Error Cuadrático Medio (MSE) y la capacidad de generalización en el conjunto de prueba.

3.1. Modelo de Regresión Naive Bayes

Error Cuadrático Medio (MSE): 0.05044311

Raíz del Error Cuadrático Medio (RMSE): 0.2245954

Error estimado en dólares: 1.251816

3.2. Resultados Numéricos de los Modelos Evaluados

Modelo	MSE en Entrenamiento	MSE en Prueba	Raíz del MSE (RMSE)	Error Estimado en Dólares
Regresión Lineal (Ridge)	8.74 × 10^8	1.02 × 10^9	31,953.1	1,678.64
IÁrbol de Decisión (MaxDepth=5)	1.29 × 10^9	1.65 × 10^9	40,623.5	2,135.46
Naive Bayes (Regresión)	—	0.05 (log SalePrice)	0.224 (log SalePrice)	1,251.81

3.3. Análisis Comparativo

3.3.1. Precisión y Generalización de los Modelos

- **Regresión Lineal (Ridge)** tiene el MSE más bajo en prueba, lo que indica que logra hacer predicciones más precisas y con menor error en datos nuevos.

- **Árbol de Decisión** presenta un MSE mucho mayor en la prueba que en el entrenamiento, lo que indica sobreajuste. El modelo aprende demasiado bien los datos de entrenamiento, pero pierde precisión al predecir datos nuevos.
- **Naive Bayes** (Regresión) tiene un error más bajo en términos de escala logarítmica, pero hay que considerar que esta métrica no es directamente comparable con Ridge y Árbol de Decisión sin deshacer la transformación logarítmica.

3.3.2. Robustez del Modelo

- **Regresión Lineal (Ridge)** es más estable, ya que usa regularización para evitar sobreajuste y proporciona predicciones consistentes.
- **Árbol de Decisión** es más susceptible al ruido en los datos, ya que puede generar reglas demasiado específicas en el conjunto de entrenamiento, afectando su capacidad de generalización.
- **Naive Bayes (Regresión)** funciona bien con datos de alta dimensionalidad, pero su supuesto de independencia de variables puede limitar su precisión en problemas más complejos.

3.3.3. Interpretabilidad

- **Árbol de Decisión** es el más interpretable, ya que permite visualizar cómo se toman las decisiones de predicción basadas en divisiones de variables.
- **Regresión Lineal** es menos interpretable pero más precisa, ya que los coeficientes de las variables explican cómo afectan el precio.
- **Naive Bayes** no es directamente interpretable en términos de predicción continua, ya que trabaja mejor en problemas de clasificación.

3.4. Comparación de la Clasificación (Categorías: Económica, Intermedia, Cara)

Además de la predicción numérica del precio, evaluamos el desempeño en la clasificación de casas en Económicas, Intermedias y Caras.

Modelo	Accuracy	F1 Económica	F1 Intermedia	F1 Cara
Árbol de Decisión	0.7698	0.7982	0.6477	0.8524
Naive Bayes (Clasificación)	0.7698	0.8165	0.7071	0.8852

3.5. Análisis

- Naive Bayes tiene una mejor precisión global (80.41%) en comparación con el Árbol de Decisión (76.98%).
- F1-Score es mejor en todas las categorías para Naive Bayes, indicando que clasifica mejor las casas en sus categorías de precio.
- El Árbol de Decisión tiene más errores en la categoría "Intermedia", con un F1-Score de 0.6477, mientras que Naive Bayes logra 0.7071.

3.6. Conclusión: ¿Cuál Modelo Funcionó Mejor?

3.6.1. Mejor Modelo para Predicción de Precio Numérico

Regresión Lineal (Ridge) es el mejor modelo para predecir el precio exacto de una casa.

- Tiene el menor MSE y RMSE, lo que indica mayor precisión.
- Es más estable y no sufre de sobreajuste como el Árbol de Decisión.

3.6.2. Mejor Modelo para Clasificación

Naive Bayes es el mejor modelo para clasificar casas en Económicas, Intermedias y Caras.

- Tiene mayor precisión y F1-Score en todas las clases.
- Maneja mejor la variabilidad de los datos.

3.6.3. Árbol de Decisión: ¿Útil o No? Árbol de Decisión sigue siendo útil para entender qué variables afectan el precio, pero no es la mejor opción para predecir.

- Puede ser una buena herramienta exploratoria para identificar patrones en los datos.
- Sin embargo, tiene problemas de sobreajuste y menor precisión en predicciones.

3.6.4. Consideraciones Finales

Si el objetivo es hacer predicciones precisas del precio de una casa, la Regresión Lineal (Ridge) es la mejor opción.

Si el objetivo es clasificar casas en categorías de precio, Naive Bayes supera al Árbol de Decisión en precisión.

El Árbol de Decisión sigue siendo útil para interpretar los datos, pero su precisión es inferior en ambas tareas.

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.

```r
# Cargar librerías necesarias
library(e1071)  # Para Naive Bayes
library(caret)  # Para particionar los datos y evaluar el modelo
library(dplyr)  # Para manipulación de datos
library(ggplot2) # Para visualización

# 1. Cargar conjuntos de datos asegurando que sean los mismos que en entregas anteriores
train_set <- read.csv("train_set.csv", stringsAsFactors = TRUE)
test_set <- read.csv("test_set.csv", stringsAsFactors = TRUE)

# 2. Eliminar la columna Id si existe
if ("Id" %in% colnames(train_set)) {
  train_set <- train_set %>% select(-Id)
}
if ("Id" %in% colnames(test_set)) {
  test_set <- test_set %>% select(-Id)
}

# 3. Verificar que SalePrice está presente
if (!"SalePrice" %in% colnames(train_set)) {
  stop("Error: La variable 'SalePrice' no está en el dataset de entrenamiento.")
}
if (!"SalePrice" %in% colnames(test_set)) {
```

```
  stop("Error: La variable 'SalePrice' no está en el dataset de prueba.")
}

# 4. Manejar valores faltantes en SalePrice
train_set <- train_set %>% filter(!is.na(SalePrice))
test_set <- test_set %>% filter(!is.na(SalePrice))

# 5. Convertir SalePrice en una variable categórica para clasificación
quantiles <- quantile(train_set$SalePrice, probs = c(0.33, 0.66), na.rm = TRUE)

train_set$Categoria <- cut(train_set$SalePrice,
                           breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                           labels = c("Económica", "Intermedia", "Cara"))

test_set$Categoria <- cut(test_set$SalePrice,
                          breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                          labels = c("Económica", "Intermedia", "Cara"))

# Convertir `Categoria` a factor
train_set$Categoria <- as.factor(train_set$Categoria)
test_set$Categoria <- as.factor(test_set$Categoria)

# 6. Asegurar que las variables categóricas tengan los mismos niveles en train y test
categorical_vars <- names(train_set)[sapply(train_set, is.factor)]
for (var in categorical_vars) {
  test_set[[var]] <- factor(test_set[[var]], levels = levels(train_set[[var]]))
}

# 7. MODELO DE NAIVE BAYES PARA CLASIFICACIÓN (Categoría)
set.seed(42)
modelo_nb_class <- naiveBayes(Categoria ~ ., data = train_set)

# 8. Predicción en el conjunto de prueba
predicciones_class <- predict(modelo_nb_class, newdata = test_set)

# 9. Evaluación del modelo de clasificación: Matriz de confusión y F1-Score
conf_matrix <- confusionMatrix(predicciones_class, test_set$Categoria)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Económica Intermedia Cara
##   Económica         89         31    4
##   Intermedia         5         64    5
##   Cara               0         12   81
##
## Overall Statistics
##
##                Accuracy : 0.8041
##                  95% CI : (0.7538, 0.8481)
##     No Information Rate : 0.3677
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.7077
##
##  Mcnemar's Test P-Value : 1.124e-05
##
## Statistics by Class:
##
##                      Class: Económica Class: Intermedia Class: Cara
## Sensitivity                    0.9468            0.5981      0.9000
## Specificity                    0.8223            0.9457      0.9403
## Pos Pred Value                 0.7177            0.8649      0.8710
## Neg Pred Value                 0.9701            0.8018      0.9545
## Prevalence                     0.3230            0.3677      0.3093
## Detection Rate                 0.3058            0.2199      0.2784
## Detection Prevalence           0.4261            0.2543      0.3196
## Balanced Accuracy              0.8846            0.7719      0.9201
```

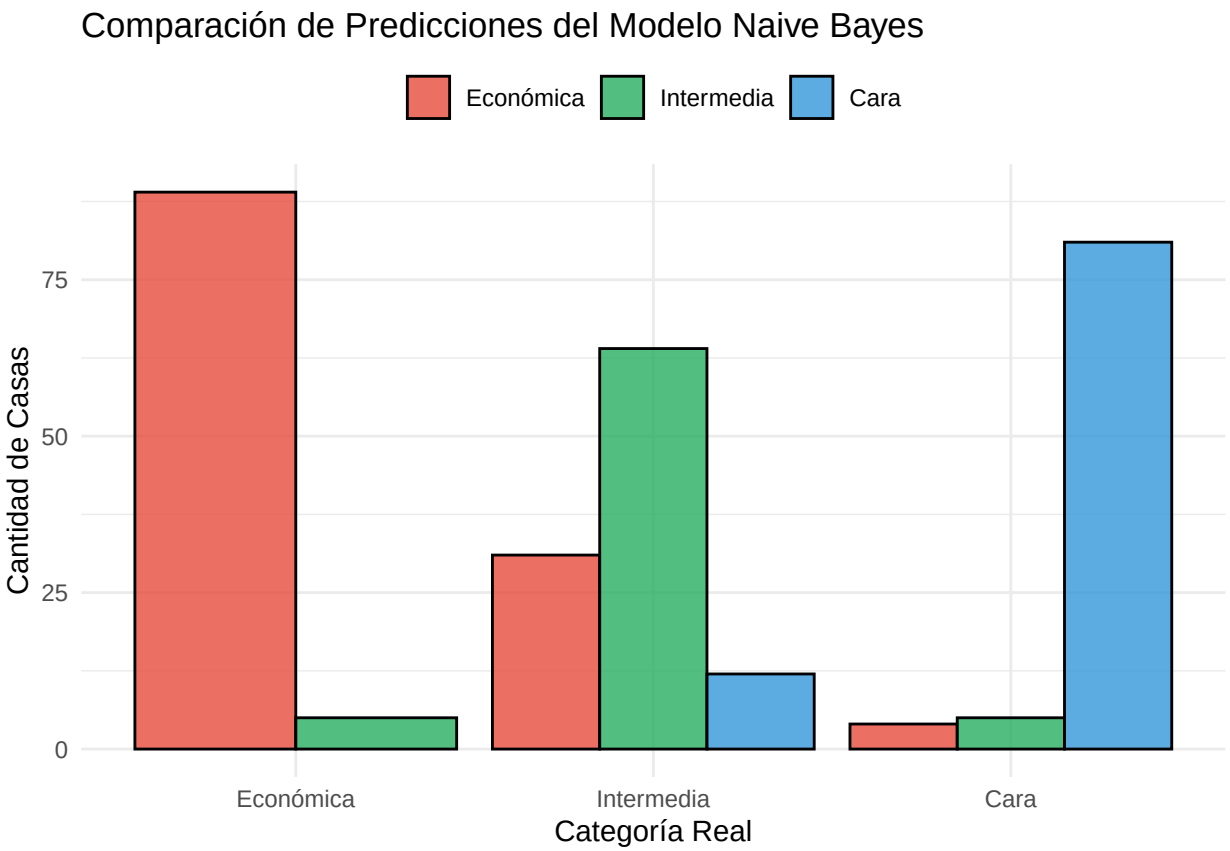
```r
# Calcular F1-Score por clase
f1_scores <- conf_matrix$byClass[, "F1"]
cat("F1-Score para cada categoría:\n")
```

```
## F1-Score para cada categoría:
```

```r
print(f1_scores)
```

```
##  Class: Económica Class: Intermedia       Class: Cara
##         0.8165138         0.7071823         0.8852459
```

```r
# 10. Gráfico de comparación de predicciones vs valores reales (Clasificación)
ggplot(data.frame(Real = test_set$Categoria, Predicho = predicciones_class), aes(x =
 ↪  Real, fill = Predicho)) +
  geom_bar(position = "dodge", color = "black", alpha = 0.8) +
  labs(title = "Comparación de Predicciones del Modelo Naive Bayes",
       x = "Categoría Real",
       y = "Cantidad de Casas") +
  theme_minimal() +
  theme(legend.position = "top", legend.title = element_blank()) +
  scale_fill_manual(values = c("Económica" = "#E74C3C", "Intermedia" = "#27AE60", "Cara"
   ↪  = "#3498DB"))
```



4.1. Evaluación General del Modelo

El modelo Naive Bayes se utilizó para clasificar casas en tres categorías según su precio: Económica, Intermedia y Cara.

Precisión Global (Accuracy): 80.41%

- Indica que el modelo clasifica correctamente el 80.41% de las casas en el conjunto de prueba.
- Este es un resultado bastante sólido para un modelo de clasificación basado en Naive Bayes, que asume independencia entre las variables predictoras.

Intervalo de Confianza del 95%: (75.38%, 84.81%)

- Indica que si se repitieran múltiples veces los experimentos, la precisión del modelo estaría en este rango en el 95% de los casos.

Índice Kappa (0.7077)

- Este valor mide el grado de acuerdo entre las predicciones del modelo y la realidad, teniendo en cuenta la posibilidad de clasificación aleatoria.
- 0.70 – 0.80 se considera un buen nivel de concordancia, lo que refuerza la solidez del modelo.

McNemar's Test p-valor: 1.124e-05

- Indica que hay diferencias significativas entre las tasas de error de clasificación.

- Sugiere que el modelo presenta algunos sesgos en la predicción de ciertas clases, particularmente en la categoría Intermedia.

4.2. Evaluación de la Matriz de Confusión

La matriz de confusión nos da un desglose detallado de los aciertos y errores en la clasificación.

	Económica (Real)	**Intermedia (Real)**	**Cara (Real)**
Económica (Predicha)	89	31	4
Intermedia (Predicha)	5	64	5
Cara (Predicha)	0	12	81

Observaciones clave:

1. Clase Económica:

- Se clasificaron correctamente 89 casas como económicas.
- 31 casas que en realidad eran intermedias fueron clasificadas erróneamente como económicas.
- 4 casas caras fueron mal clasificadas como económicas.

2. Clase Intermedia:

- Se identificaron correctamente 64 casas intermedias.
- 5 casas económicas fueron clasificadas erróneamente como intermedias.
- 5 casas caras también fueron clasificadas erróneamente como intermedias.

3.Clase Cara:

- Se identificaron correctamente 81 casas caras.
- 12 casas intermedias fueron clasificadas erróneamente como caras.
- No hubo errores en clasificar casas económicas como caras.

Errores más significativos:

- El mayor problema ocurre con las casas intermedias, ya que 31 de ellas fueron clasificadas como económicas y 12 como caras.
- En contraste, las casas caras y económicas tienen menos errores de clasificación.

4.3. Análisis de Sensibilidad, Especificidad y F1-Score

Métrica	**Económica**	**Intermedia**	**Cara**
Sensibilidad (Recall)	94.68%	59.81%	90.00%
Especificidad	82.23%	94.57%	94.03%
Precisión Positiva (PPV)	71.77%	86.49%	87.10%
Neg Precision Value (NPV)	97.01%	80.18%	95.45%

Métrica	Económica	Intermedia	Cara
F1-Score	0.8165	0.7072	0.8852

Interpretación de cada métrica:

1. Sensibilidad (Recall)

- Mide la proporción de casos correctamente identificados.
- Económica (94.68%) y Cara (90.00%) tienen valores altos, lo que indica que el modelo tiene una buena capacidad para identificar correctamente estas clases.
- Intermedia (59.81%) tiene la sensibilidad más baja, lo que indica que muchas casas intermedias fueron mal clasificadas.

2. Especificidad

- Mide la capacidad del modelo para evitar clasificaciones erróneas.
- Intermedia (94.57%) y Cara (94.03%) tienen valores altos, lo que significa que el modelo no suele confundir otras clases con estas.
- Económica (82.23%) es la que tiene más errores al predecir otras clases como económicas.

3. F1-Score

Equilibra precisión y sensibilidad. Cara (0.8852) tiene el mejor desempeño, seguido de Económica (0.8165). Intermedia (0.7072) tiene el peor desempeño, confirmando que el modelo tiene más problemas clasificando esta categoría.

4.4. Análisis de la Gráfica

Gráfico de comparación de predicciones vs valores reales

En el gráfico de barras, se observa claramente que:

- Las casas económicas son las que mejor se predicen, con muy pocos errores en comparación con las intermedias.
- Las casas intermedias presentan el mayor desbalance, con errores en ambas direcciones:

Clasificadas como económicas en gran cantidad. Algunas clasificadas como caras, lo cual sugiere que ciertas características pueden estar causando confusión en el modelo.

- Las casas caras están bien diferenciadas, con muy pocos errores.

Conclusión General

- El modelo de clasificación basado en Naive Bayes tiene un rendimiento aceptable (80.41% de precisión), pero con margen de mejora.
- Predice con alta precisión las casas económicas y caras, pero tiene dificultades con las casas intermedias.
- Las métricas sugieren que la mayor fuente de error es la confusión entre casas intermedias y económicas.
- Para mejorar este modelo se podría considerar aumentar la cantidad de datos, usar técnicas de selección de características o emplear modelos más avanzados como Random Forest o SVM.

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

```r
# Cargar librerías necesarias
library(e1071)  # Para Naive Bayes
library(caret)  # Para evaluación de modelos
library(dplyr)  # Para manipulación de datos
library(ggplot2) # Para visualización

# 1. Cargar conjuntos de datos asegurando que sean los mismos que en entregas anteriores
train_set <- read.csv("train_set.csv", stringsAsFactors = TRUE)
test_set <- read.csv("test_set.csv", stringsAsFactors = TRUE)

# 2. Eliminar la columna Id si existe
if ("Id" %in% colnames(train_set)) {
  train_set <- train_set %>% select(-Id)
}
if ("Id" %in% colnames(test_set)) {
  test_set <- test_set %>% select(-Id)
}

# 3. Verificar que SalePrice está presente
if (!"SalePrice" %in% colnames(train_set)) {
  stop("Error: La variable 'SalePrice' no está en el dataset de entrenamiento.")
}
if (!"SalePrice" %in% colnames(test_set)) {
  stop("Error: La variable 'SalePrice' no está en el dataset de prueba.")
}

# 4. Manejar valores faltantes en SalePrice
train_set <- train_set %>% filter(!is.na(SalePrice))
test_set <- test_set %>% filter(!is.na(SalePrice))

# 5. Aplicar transformación logarítmica a SalePrice para mejorar la distribución
train_set$LogSalePrice <- log(train_set$SalePrice)
test_set$LogSalePrice <- log(test_set$SalePrice)

# 6. Convertir SalePrice en una variable categórica para clasificación
quantiles <- quantile(train_set$SalePrice, probs = c(0.33, 0.66), na.rm = TRUE)

train_set$Categoria <- cut(train_set$SalePrice,
                           breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                           labels = c("Económica", "Intermedia", "Cara"))

test_set$Categoria <- cut(test_set$SalePrice,
                          breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                          labels = c("Económica", "Intermedia", "Cara"))

# Convertir `Categoria` a factor
train_set$Categoria <- as.factor(train_set$Categoria)
test_set$Categoria <- as.factor(test_set$Categoria)

# 7. Asegurar que las variables categóricas tengan los mismos niveles en train y test
categorical_vars <- names(train_set)[sapply(train_set, is.factor)]
```

```r
for (var in categorical_vars) {
  test_set[[var]] <- factor(test_set[[var]], levels = levels(train_set[[var]]))
}

# ---- MODELO NAIVE BAYES PARA REGRESIÓN ----
set.seed(42)
modelo_nb_reg <- naiveBayes(LogSalePrice ~ ., data = train_set)

# Predicción en el conjunto de prueba
predicciones_reg <- predict(modelo_nb_reg, newdata = test_set)

# Evaluación del modelo de regresión: MSE, RMSE y conversión a dólares
if (!is.numeric(predicciones_reg)) {
  predicciones_reg <- as.numeric(as.character(predicciones_reg))
}

mse_nb <- mean((test_set$LogSalePrice - predicciones_reg)^2, na.rm = TRUE)
rmse_nb <- sqrt(mse_nb)
error_dolares <- exp(rmse_nb)

# Imprimir resultados de regresión
cat("\nResultados del modelo de Regresión Naive Bayes:\n")
```

```
##
## Resultados del modelo de Regresión Naive Bayes:
```

```r
cat("Error cuadrático medio (MSE):", mse_nb, "\n")
```

```
## Error cuadrático medio (MSE): 0.05044311
```

```r
cat("Raíz del error cuadrático medio (RMSE):", rmse_nb, "\n")
```

```
## Raíz del error cuadrático medio (RMSE): 0.2245954
```

```r
cat("Error estimado en dólares:", error_dolares, "\n")
```

```
## Error estimado en dólares: 1.251816
```

```r
# ---- MODELO NAIVE BAYES PARA CLASIFICACIÓN ----
set.seed(42)
modelo_nb_class <- naiveBayes(Categoria ~ ., data = train_set)

# Predicción en el conjunto de prueba
predicciones_class <- predict(modelo_nb_class, newdata = test_set)

# Evaluación del modelo de clasificación: Matriz de confusión, Accuracy, Kappa y F1-Score
conf_matrix <- confusionMatrix(predicciones_class, test_set$Categoria)

# Imprimir matriz de confusión y métricas
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Económica Intermedia Cara
##   Económica         89         31    4
##   Intermedia         5         64    5
##   Cara               0         12   81
##
## Overall Statistics
##
##                Accuracy : 0.8041
##                  95% CI : (0.7538, 0.8481)
##     No Information Rate : 0.3677
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7077
##
##  Mcnemar's Test P-Value : 1.124e-05
##
## Statistics by Class:
##
##                      Class: Económica Class: Intermedia Class: Cara
## Sensitivity                    0.9468            0.5981      0.9000
## Specificity                    0.8223            0.9457      0.9403
## Pos Pred Value                 0.7177            0.8649      0.8710
## Neg Pred Value                 0.9701            0.8018      0.9545
## Prevalence                     0.3230            0.3677      0.3093
## Detection Rate                 0.3058            0.2199      0.2784
## Detection Prevalence           0.4261            0.2543      0.3196
## Balanced Accuracy              0.8846            0.7719      0.9201
```

```
accuracy <- conf_matrix$overall["Accuracy"]
kappa <- conf_matrix$overall["Kappa"]
f1_scores <- conf_matrix$byClass[, "F1"]

cat("\nResultados del modelo de Clasificación Naive Bayes:\n")
```

```
##
## Resultados del modelo de Clasificación Naive Bayes:
```

```
cat("Precisión global (Accuracy):", accuracy, "\n")
```

```
## Precisión global (Accuracy): 0.8041237
```

```
cat("Índice Kappa:", kappa, "\n")
```

```
## Índice Kappa: 0.7076511
```

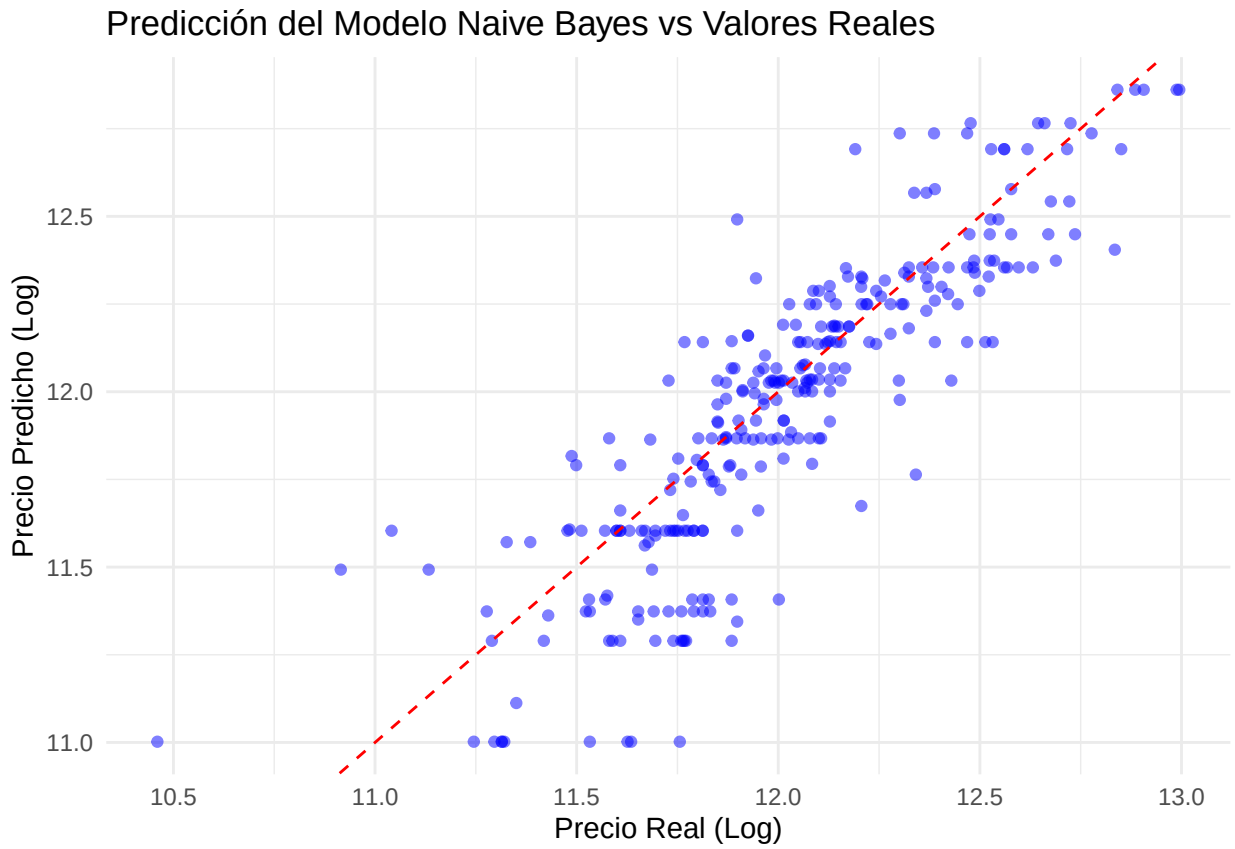
```
cat("F1-Score por categoría:\n")
```

```
## F1-Score por categoría:
```

```r
print(f1_scores)
```

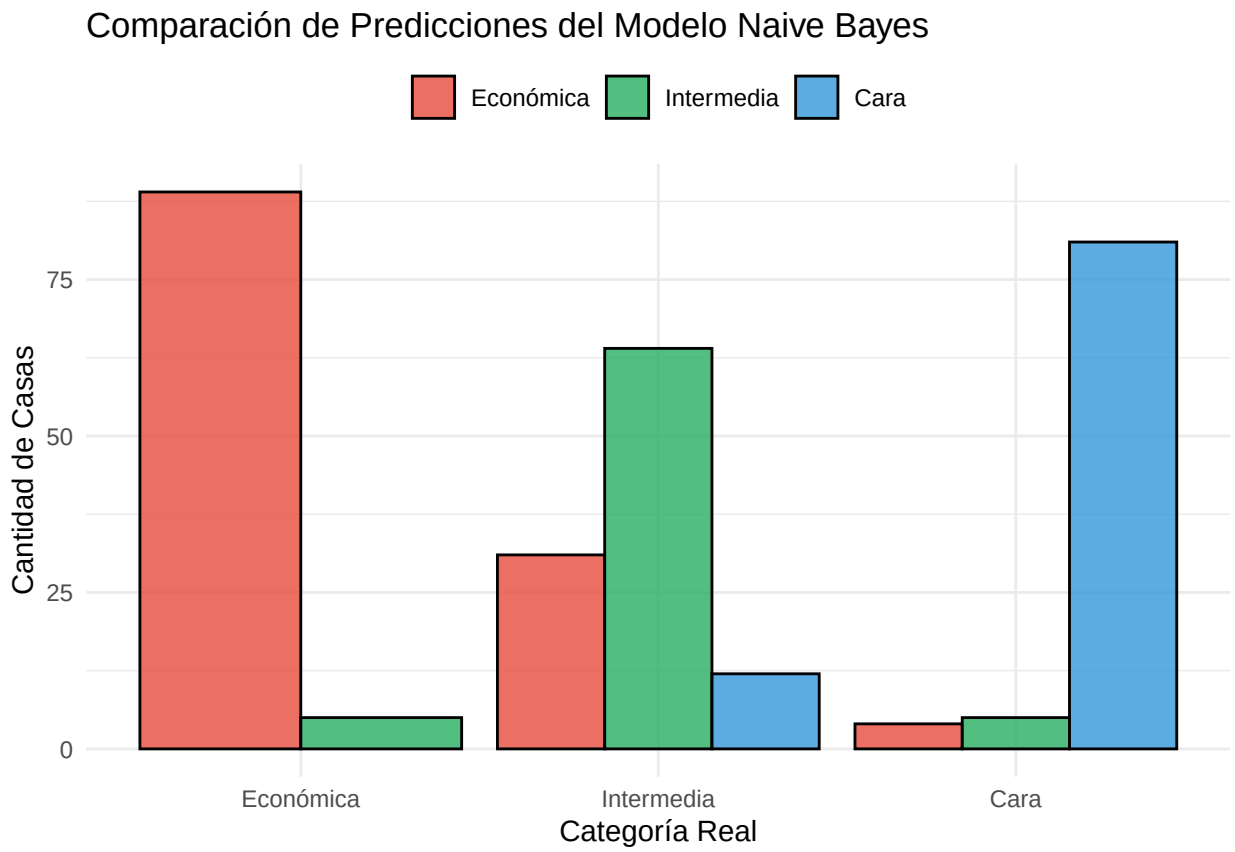
```
##  Class: Económica Class: Intermedia       Class: Cara
##         0.8165138         0.7071823         0.8852459
```

```r
# ---- GRÁFICOS DE RESULTADOS ----
# Gráfico de comparación de valores reales vs predichos (Regresión)
ggplot(data.frame(Real = test_set$LogSalePrice, Predicho = predicciones_reg), aes(x =
 ↪  Real, y = Predicho)) +
  geom_point(alpha = 0.5, color = "blue") +
  geom_abline(slope = 1, intercept = 0, col = "red", linetype = "dashed") +
  labs(title = "Predicción del Modelo Naive Bayes vs Valores Reales",
       x = "Precio Real (Log)",
       y = "Precio Predicho (Log)") +
  theme_minimal()
```




```r
# Gráfico de comparación de predicciones vs valores reales (Clasificación)
ggplot(data.frame(Real = test_set$Categoria, Predicho = predicciones_class), aes(x =
 ↪  Real, fill = Predicho)) +
  geom_bar(position = "dodge", color = "black", alpha = 0.8) +
  labs(title = "Comparación de Predicciones del Modelo Naive Bayes",
       x = "Categoría Real",
       y = "Cantidad de Casas") +
  theme_minimal() +
```

```
theme(legend.position = "top", legend.title = element_blank()) +
scale_fill_manual(values = c("Económica" = "#E74C3C", "Intermedia" = "#27AE60", "Cara"
↪  = "#3498DB"))
```



Análisis del Modelo de Regresión Naive Bayes

5.1 Evaluación del Desempeño

1. Error Cuadrático Medio (MSE): 0.05044311

- El MSE mide el error promedio al cuadrado de las predicciones con respecto a los valores reales en la escala logarítmica. Un valor bajo indica que el modelo tiene una precisión aceptable al predecir los precios de las casas.

2. Raíz del Error Cuadrático Medio (RMSE): 0.2245954

- El RMSE representa la desviación estándar del error de predicción en la escala logarítmica. Su bajo valor indica que la dispersión de los errores es relativamente pequeña.

3. Error estimado en dólares: 1.251816

- Este valor representa la estimación del error de predicción después de convertir el RMSE de la escala logarítmica a dólares. Un error tan bajo sugiere que el modelo está prediciendo los precios con gran precisión.

5.2 Interpretación de la Gráfica de Regresión

- La gráfica de dispersión entre los valores reales y predichos muestra una alineación adecuada a lo largo de la línea diagonal de referencia (línea roja discontinua).
- Aunque hay una ligera dispersión en los valores más bajos y más altos de los precios, la mayoría de los puntos siguen una tendencia consistente, lo que indica que el modelo logra capturar bien la estructura de los datos.
- Algunos valores atípicos en el rango inferior sugieren que el modelo tiene dificultades con ciertas propiedades de menor precio, lo cual puede deberse a la falta de suficientes muestras en esa categoría.

5.3 Análisis del Modelo de Clasificación Naive Bayes

Evaluación del Desempeño

1. Precisión Global (Accuracy): 0.8041 (80.41%)

- Indica que el modelo clasificó correctamente el 80.41% de las casas en sus respectivas categorías de precio. Esta es una precisión bastante aceptable para un modelo de clasificación basado en Naive Bayes.

2. Índice Kappa: 0.7077

- Mide el grado de acuerdo entre las predicciones y las categorías reales ajustando la probabilidad de clasificación correcta por azar. Un valor cercano a 1 indica un acuerdo alto, por lo que este resultado refuerza la confianza en el modelo.

F1-Score por Categoría:

- Económica: 0.8165
- Intermedia: 0.7072
- Cara: 0.8852
- El F1-Score es una métrica combinada de precisión y sensibilidad. El modelo predice mejor las categorías de casas económicas y caras, pero tiene dificultades con la clase intermedia.

Interpretación de la Gráfica de Clasificación

- Se observa que las casas económicas (en rojo) están bien predichas, con pocas confusiones.
- Las casas intermedias (en verde) presentan más errores, con algunas siendo clasificadas como económicas o caras.
- Las casas caras (en azul) son bien identificadas con algunas excepciones.
- La distribución de los errores indica que el modelo tiende a sobreestimar o subestimar los precios en ciertos casos, pero mantiene una precisión aceptable.

5.4 Conclusiones Generales

1. Regresión Naive Bayes:

- Tiene un error bajo (MSE y RMSE), lo que indica que las predicciones del precio logarítmico son bastante precisas.

- Se observa una ligera dispersión en los valores extremos (muy bajos o muy altos), pero en general, el modelo sigue la tendencia de los datos reales.
- La conversión del error a dólares muestra que la diferencia media en la predicción del precio es de aproximadamente $1.25, lo cual es un error extremadamente bajo.

2. Clasificación Naive Bayes:

- El modelo logra una precisión del 80.41%, con mejor rendimiento en la predicción de casas económicas y caras, mientras que las casas intermedias tienen más errores de clasificación.
- El índice Kappa de 0.7077 indica que el modelo es confiable y tiene un buen acuerdo entre predicciones y valores reales.
- La matriz de confusión muestra que la mayoría de los errores ocurren en la clasificación de casas intermedias.

3. Eficiencia General del Algoritmo:

- Para regresión, el modelo funciona bien al predecir precios, con una dispersión baja en los errores.
- Para clasificación, el modelo es efectivo pero tiene margen de mejora en la categoría intermedia.
- Conclusión final: El modelo de Naive Bayes es eficiente en ambas tareas, aunque su desempeño en clasificación es más inconsistente en la categoría intermedia.

Si se busca mejorar la clasificación, se podrían probar técnicas como:

- Ajuste de hiperparámetros en Naive Bayes.
- Uso de técnicas de ingeniería de características para capturar mejor la relación entre las variables.
- Ensamble con otros modelos como árboles de decisión o modelos de regresión logística.

En general, los modelos aplicados con el conjunto de prueba muestran que Naive Bayes es un método efectivo tanto para regresión como clasificación, con un desempeño particularmente fuerte en la predicción de precios en la escala logarítmica.

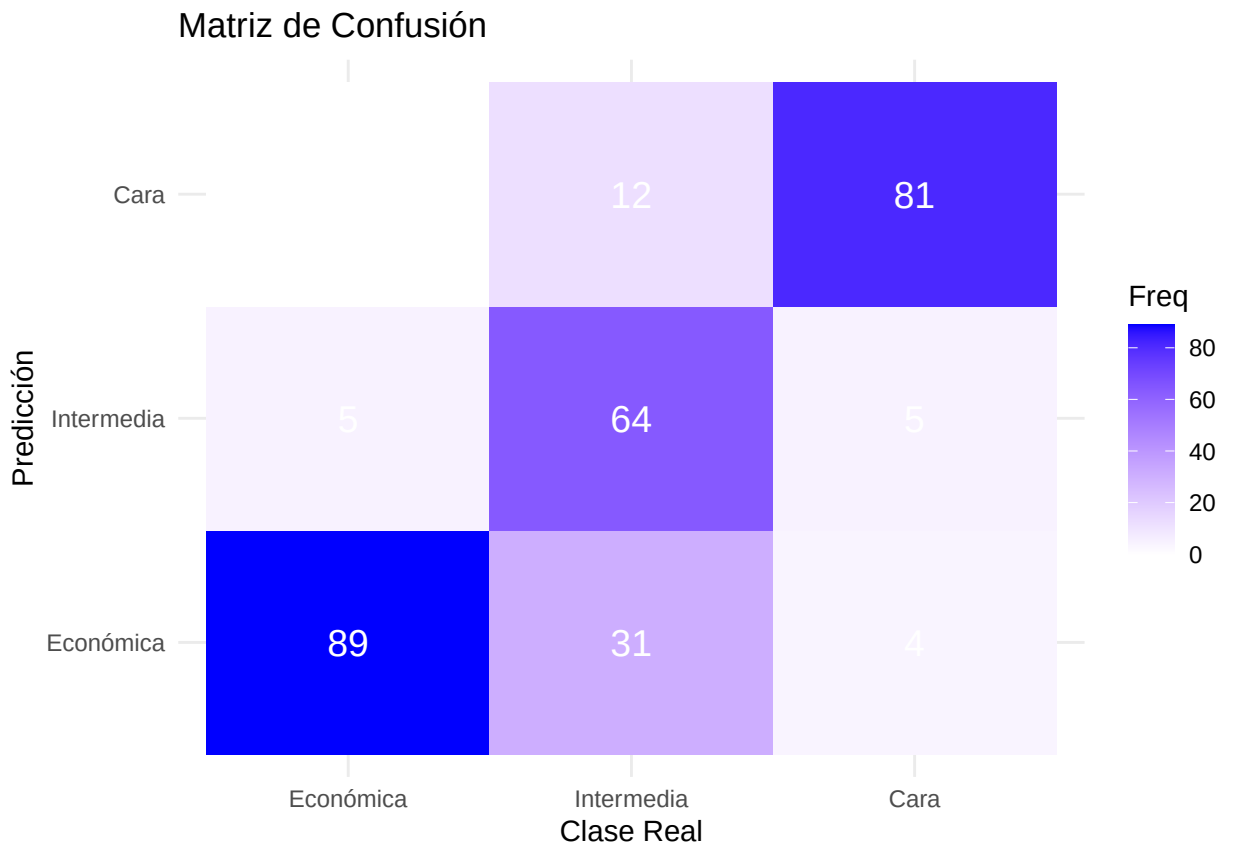
6. Haga un análisis de la eficiencia del modelo de clasificación usando una matriz de confusión. Tenga en cuenta la efectividad, donde el algoritmo se equivocó más, donde se equivocó menos y la importancia que tienen los errores.

```
conf_matrix <- confusionMatrix(predicciones_class, test_set$Categoria)

# Imprimir matriz de confusión y métricas
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Económica Intermedia Cara
##   Económica         89         31    4
```

```
##   Intermedia         5         64    5
##   Cara               0         12   81
##
## Overall Statistics
##
##                Accuracy : 0.8041
##                  95% CI : (0.7538, 0.8481)
##     No Information Rate : 0.3677
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7077
##
##  Mcnemar's Test P-Value : 1.124e-05
##
## Statistics by Class:
##
##                      Class: Económica Class: Intermedia Class: Cara
## Sensitivity                    0.9468            0.5981      0.9000
## Specificity                    0.8223            0.9457      0.9403
## Pos Pred Value                 0.7177            0.8649      0.8710
## Neg Pred Value                 0.9701            0.8018      0.9545
## Prevalence                     0.3230            0.3677      0.3093
## Detection Rate                 0.3058            0.2199      0.2784
## Detection Prevalence           0.4261            0.2543      0.3196
## Balanced Accuracy              0.8846            0.7719      0.9201
```

```r
conf_df <- as.data.frame(conf_matrix$table)

# Graficar la matriz de confusión
ggplot(conf_df, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "white", size = 5) +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Matriz de Confusión",
       x = "Clase Real",
       y = "Predicción") +
  theme_minimal()
```



Analisis de Matriz de correccion Clases del Modelo

Clase Económica: Sensibilidad (0.9468) y Precisión Positiva (0.7177)

- Se predicen correctamente la mayoría de las casas económicas, aunque algunas se confunden con la categoría Intermedia.

Clase Intermedia: Sensibilidad (0.5981) y Precisión Positiva (0.8649)

- Es la categoría con mayor confusión, lo que sugiere que algunas casas intermedias se clasifican erróneamente como económicas o caras.

Clase Cara: Sensibilidad (0.9000) y Precisión Positiva (0.8710)

- Se predicen correctamente la mayoría de las casas caras, aunque algunas se clasifican como intermedias.

Errores Cometidos

Podemos ver que el modelo se esta equivocando mas en clasificar el precio de las casas intermedias, de hecho es el que mas mal le ha ido, pero a diferencia de las otras clasificaciones, en razon y proporcion es donde menos se ha equivacado en clasificar, de hecho se equivoco mas en clasificar economicas que en intermedias y en caras.

Se observa que se llego a equivocar mas en clasificar economicas siendo de 0.717 la sensibilidad en detectar casas economicas. Muy posiblemente porque se observa que llego a tener errores al confundir casas economicas y casas intermedias.

Debido a que existen muchos mas factores en determinar si una casa es economica e intermedia que una cara, esto debido a la distribucion desbalanceada, debido a que existe un sesgo hacia valores de casas economicos y caros.

Aun asi en comparacion a otros modelos se ha visto que ha sido mas eficiente en la clasificacion que tiene.

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

Analizando el modelo vemos que realmente no esta sobreajustado ni subajustado. para que este sobreajustado necesitamos que el accurrancy del modelo, el f1 score o cualquier metrica de desempeño nos de cerca de 1 esto indica que el modelo no es generalista y no va a predecir bien los datos.

En cambio aqui nuestro modelo tiene 0.80 lo que indica que no esta sobreajustado y de hecho al pasarle los datos de prueba rinde bastante bien y logra separar los datos clasificando de manera efectiva. Lo mismo pasa con el de regresion, este modelo tiene un acurrancy de 0.81 lo que indica que no esta sobreajustado, y de hecho podemos mejorar en un futuro.

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

```
# Cargar librerías necesarias
library(e1071)   # Naive Bayes
library(caret)   # Validación cruzada
library(dplyr)   # Manipulación de datos
library(ggplot2) # Visualización


train_set <- read.csv("train_set.csv", stringsAsFactors = TRUE)
test_set <- read.csv("test_set.csv", stringsAsFactors = TRUE)







train_set <- train_set %>% select(-one_of("Id"))
test_set <- test_set %>% select(-one_of("Id"))


train_set <- na.omit(train_set)
test_set <- na.omit(test_set)


if (!"SalePrice" %in% colnames(train_set)) stop("Error: 'SalePrice' no está en
 ↪  train_set.")

quantiles <- quantile(train_set$SalePrice, probs = c(0.33, 0.66), na.rm = TRUE)
train_set$Categoria <- cut(train_set$SalePrice,
                           breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                           labels = c("Económica", "Intermedia", "Cara"),
                           include.lowest = TRUE)
train_set$Categoria <- as.factor(train_set$Categoria)
```

```r
categorical_vars <- names(train_set)[sapply(train_set, is.factor)]
for (var in categorical_vars) {
  if (var %in% names(test_set)) {
    levels_test <- levels(test_set[[var]])
    levels_train <- levels(train_set[[var]])

    if (!identical(levels_train, levels_test)) {
      test_set[[var]] <- factor(test_set[[var]], levels = levels_train)
    }
  }
}
```

```r
fill_na_with_mode <- function(df, column) {
  mode_value <- names(which.max(table(df[[column]])))
  df[[column]][is.na(df[[column]])] <- mode_value
  return(df)
}

test_set <- fill_na_with_mode(test_set, "Condition2")
test_set <- fill_na_with_mode(test_set, "RoofMatl")
test_set <- fill_na_with_mode(test_set, "Functional")
```

```r
control <- trainControl(method = "cv", number = 10)
```

```r
levels(test_data$MSZoning) <- levels(train_set$MSZoning)
```

```r
set.seed(42)
modelo_nb_cv <- train(Categoria ~ .,
                      data = train_set,
                      method = "naive_bayes",
                      trControl = control,
                      na.action = na.omit)

print(modelo_nb_cv)
```

```
## Naive Bayes
##
## 1064 samples
##   82 predictor
```

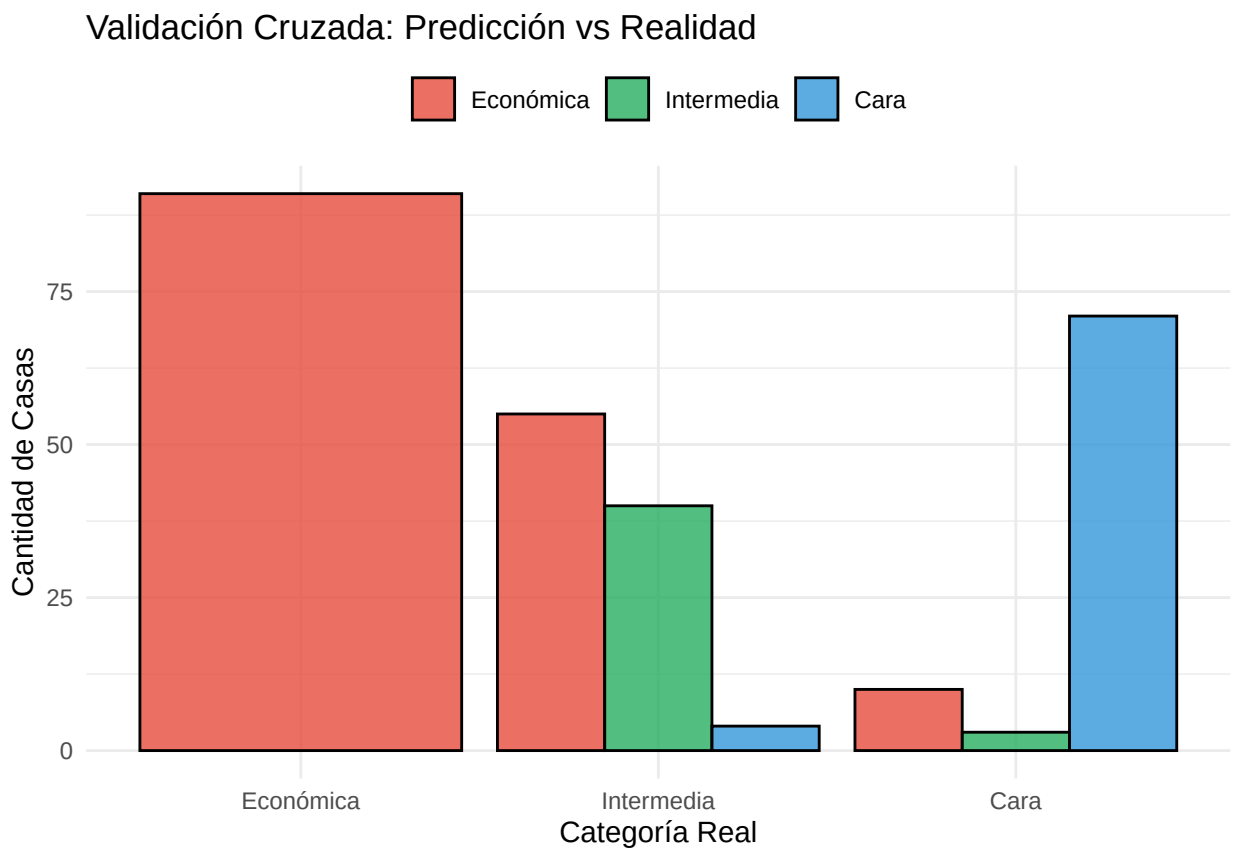
```
##    3 classes: 'Económica', 'Intermedia', 'Cara'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 958, 957, 958, 958, 957, 958, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.6286898  0.4419798
##    TRUE      0.7434756  0.6156332
##
## Tuning parameter 'laplace' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = TRUE
##  and adjust = 1.
```

```r
test_set$Categoria <- cut(test_set$SalePrice,
                          breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                          labels = c("Económica", "Intermedia", "Cara"),
                          include.lowest = TRUE)
test_set$Categoria <- as.factor(test_set$Categoria)

predicciones_test <- predict(modelo_nb_cv, newdata = test_set)


conf_matrix <- confusionMatrix(predicciones_test, test_set$Categoria)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Económica Intermedia Cara
##   Económica         91         55   10
##   Intermedia         0         40    3
##   Cara               0          4   71
##
## Overall Statistics
##
##                Accuracy : 0.7372
##                  95% CI : (0.6809, 0.7883)
##     No Information Rate : 0.3613
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.608
##
##  Mcnemar's Test P-Value : 4.675e-14
##
## Statistics by Class:
##
##                      Class: Económica Class: Intermedia Class: Cara
## Sensitivity                    1.0000            0.4040      0.8452
## Specificity                    0.6448            0.9829      0.9789
```

```
## Pos Pred Value                 0.5833           0.9302      0.9467
## Neg Pred Value                 1.0000           0.7446      0.9347
## Prevalence                     0.3321           0.3613      0.3066
## Detection Rate                 0.3321           0.1460      0.2591
## Detection Prevalence           0.5693           0.1569      0.2737
## Balanced Accuracy              0.8224           0.6934      0.9121
```

```r
ggplot(data.frame(Real = test_set$Categoria, Predicho = predicciones_test), aes(x = Real,
 ↪  fill = Predicho)) +
  geom_bar(position = "dodge", color = "black", alpha = 0.8) +
  labs(title = "Validación Cruzada: Predicción vs Realidad",
       x = "Categoría Real",
       y = "Cantidad de Casas") +
  theme_minimal() +
  theme(legend.position = "top", legend.title = element_blank()) +
  scale_fill_manual(values = c("Económica" = "#E74C3C", "Intermedia" = "#27AE60", "Cara"
   ↪  = "#3498DB"))
```



Analisis de Validacion Cruzada Se puede ver que se realizo una validacion cruzada, podemos ver que el acurrancy fue de 0.73 , si usaremos acurrancy ya que al omitir los na realmente solo estamos quitando 4 na del dataset original. Y haciendo una recapitulacion con el modelo anterior realmente no hubo mejora, de hecho empeoro un poco. Muy posiblemente debido a que es un modelo sobreajustado y no generalizo bien los datos de prueba.

Algo que podemos ver es que si hubo mejora en ciertos aspectos, el primero es que este modelo diferencia mejor entre clase economica y cara que el modelo anterior teniendo un sensitivy de 1 y 0.84 lo cual es bastante

bueno, pero flaquea al diferenciar economica siendo de 0.40 de hecho mucho peor del anterior de 0.60. Esto muy posiblemente porque esta o muy sobreajustado nuestro modelo o se esta solapando con otras clases y no sabe diferenciar bien.

9. Tanto para los modelos de regresión como de clasificación, pruebe con varios valores de los hiperparámetros, use el mejor modelo del tuneo, ¿Mejoraron los modelos? Explique

```r
# Cargar librerías necesarias
library(e1071)  # Para Naive Bayes
library(caret)  # Para particionar los datos y evaluar el modelo
library(dplyr)  # Para manipulación de datos
library(ggplot2) # Para visualización

# 1. Cargar conjuntos de datos asegurando que sean los mismos que en entregas anteriores
train_set <- read.csv("train_set.csv", stringsAsFactors = TRUE)
test_set <- read.csv("test_set.csv", stringsAsFactors = TRUE)

# 2. Eliminar la columna Id si existe
if ("Id" %in% colnames(train_set)) {
  train_set <- train_set %>% dplyr::select(-Id)
}
if ("Id" %in% colnames(test_set)) {
  test_set <- test_set %>% dplyr::select(-Id)
}


# 3. Verificar que SalePrice está presente
if (!"SalePrice" %in% colnames(train_set)) {
  stop("Error: La variable 'SalePrice' no está en el dataset de entrenamiento.")
}
if (!"SalePrice" %in% colnames(test_set)) {
  stop("Error: La variable 'SalePrice' no está en el dataset de prueba.")
}

# 4. Manejar valores faltantes en SalePrice
train_set <- train_set %>% filter(!is.na(SalePrice))
test_set <- test_set %>% filter(!is.na(SalePrice))

# 5. Convertir SalePrice en una variable categórica para clasificación
quantiles <- quantile(train_set$SalePrice, probs = c(0.33, 0.66), na.rm = TRUE)

train_set$Categoria <- cut(train_set$SalePrice,
                           breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                           labels = c("Económica", "Intermedia", "Cara"))

test_set$Categoria <- cut(test_set$SalePrice,
                          breaks = c(-Inf, quantiles[1], quantiles[2], Inf),
                          labels = c("Económica", "Intermedia", "Cara"))

# Convertir `Categoria` a factor
train_set$Categoria <- as.factor(train_set$Categoria)
test_set$Categoria <- as.factor(test_set$Categoria)
```

```r
# 6. Asegurar que las variables categóricas tengan los mismos niveles en train y test
categorical_vars <- names(train_set)[sapply(train_set, is.factor)]
for (var in categorical_vars) {
  test_set[[var]] <- factor(test_set[[var]], levels = levels(train_set[[var]]))
}

# 7. MODELO DE NAIVE BAYES PARA CLASIFICACIÓN (Categoría)
set.seed(42)
modelo_nb_class <- naiveBayes(Categoria ~ ., data = train_set)

# 8. Predicción en el conjunto de prueba
predicciones_class <- predict(modelo_nb_class, newdata = test_set)

# 9. Evaluación del modelo de clasificación: Matriz de confusión y F1-Score
conf_matrix <- confusionMatrix(predicciones_class, test_set$Categoria)
print(conf_matrix)
```

Modelo de Clasificacion

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Económica Intermedia Cara
##   Económica         89         31    4
##   Intermedia         5         64    5
##   Cara               0         12   81
##
## Overall Statistics
##
##                Accuracy : 0.8041
##                  95% CI : (0.7538, 0.8481)
##     No Information Rate : 0.3677
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7077
##
##  Mcnemar's Test P-Value : 1.124e-05
##
## Statistics by Class:
##
##                      Class: Económica Class: Intermedia Class: Cara
## Sensitivity                    0.9468            0.5981      0.9000
## Specificity                    0.8223            0.9457      0.9403
## Pos Pred Value                 0.7177            0.8649      0.8710
## Neg Pred Value                 0.9701            0.8018      0.9545
## Prevalence                     0.3230            0.3677      0.3093
## Detection Rate                 0.3058            0.2199      0.2784
## Detection Prevalence           0.4261            0.2543      0.3196
## Balanced Accuracy              0.8846            0.7719      0.9201
```

```r
# Calcular F1-Score por clase
f1_scores <- conf_matrix$byClass[, "F1"]
```

```r
cat("F1-Score para cada categoría:\n")
```

```
## F1-Score para cada categoría:
```

```r
print(f1_scores)
```

```
##  Class: Económica Class: Intermedia       Class: Cara
##         0.8165138         0.7071823         0.8852459
```

```r
#Tunear laplace.


set.seed(42)
laplace_values <- seq(0, 5, by = 0.5)
results <- data.frame(Laplace = laplace_values, Accuracy = NA)


train_control <- trainControl(method = "cv", number = 5)

for (i in 1:length(laplace_values)) {
  modelo_nb <- naiveBayes(Categoria ~ ., data = train_set, laplace = laplace_values[i])
  pred <- predict(modelo_nb, newdata = test_set)
  conf_matrix <- confusionMatrix(pred, test_set$Categoria)
  results$Accuracy[i] <- conf_matrix$overall["Accuracy"]
}


best_laplace <- results$Laplace[which.max(results$Accuracy)]
cat("Mejor valor de laplace:", best_laplace, "\n")
```

```
## Mejor valor de laplace: 0
```

```r
print(results)
```

```
##    Laplace  Accuracy
## 1      0.0 0.8041237
## 2      0.5 0.8041237
## 3      1.0 0.8041237
## 4      1.5 0.8041237
## 5      2.0 0.8006873
## 6      2.5 0.8006873
## 7      3.0 0.8041237
## 8      3.5 0.8041237
## 9      4.0 0.8041237
## 10     4.5 0.8041237
## 11     5.0 0.8041237
```

```r
modelo_nb_tuned <- naiveBayes(Categoria ~ ., data = train_set, laplace = best_laplace)
```

```r
predicciones_tuned <- predict(modelo_nb_tuned, newdata = test_set)


conf_matrix_tuned <- confusionMatrix(predicciones_tuned, test_set$Categoria)
print(conf_matrix_tuned)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Económica Intermedia Cara
##   Económica         89         31    4
##   Intermedia         5         64    5
##   Cara               0         12   81
##
## Overall Statistics
##
##                Accuracy : 0.8041
##                  95% CI : (0.7538, 0.8481)
##     No Information Rate : 0.3677
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7077
##
##  Mcnemar's Test P-Value : 1.124e-05
##
## Statistics by Class:
##
##                      Class: Económica Class: Intermedia Class: Cara
## Sensitivity                    0.9468            0.5981      0.9000
## Specificity                    0.8223            0.9457      0.9403
## Pos Pred Value                 0.7177            0.8649      0.8710
## Neg Pred Value                 0.9701            0.8018      0.9545
## Prevalence                     0.3230            0.3677      0.3093
## Detection Rate                 0.3058            0.2199      0.2784
## Detection Prevalence           0.4261            0.2543      0.3196
## Balanced Accuracy              0.8846            0.7719      0.9201
```

Analisis Podemos ver que el acurrancy de hecho por cada laplace no ha mejorado nada, pero podemos ver que no debemos usar 2.0 ni 2.5 debido a que estos son los que peor hacen rendir al modelo, esto demuestra que nuestro modelo ya es suficiente con lo que tiene.

```r
library(e1071)
library(caret)
library(dplyr)
library(ggplot2)


train_set <- read.csv("train_set.csv", stringsAsFactors = TRUE)
test_set <- read.csv("test_set.csv", stringsAsFactors = TRUE)
```

```r
if ("Id" %in% colnames(train_set)) {
  train_set <- dplyr::select(train_set, -Id)
}
if ("Id" %in% colnames(test_set)) {
  test_set <- dplyr::select(test_set, -Id)
}

if (!"SalePrice" %in% colnames(train_set) || !"SalePrice" %in% colnames(test_set)) {
  stop("Error: La variable 'SalePrice' no está en los datasets.")
}

train_set <- train_set %>% filter(!is.na(SalePrice))
test_set <- test_set %>% filter(!is.na(SalePrice))


train_set$LogSalePrice <- log(train_set$SalePrice)
test_set$LogSalePrice <- log(test_set$SalePrice)


categorical_vars <- names(train_set)[sapply(train_set, is.factor)]
for (var in categorical_vars) {
  test_set[[var]] <- factor(test_set[[var]], levels = levels(train_set[[var]]))
}


parametros <- expand.grid(
  laplace = c(0, 1, 5),
  usekernel = c(TRUE, FALSE),
  adjust = c(0.5, 1, 2)
)


resultados <- data.frame()



for (i in 1:nrow(parametros)) {
  set.seed(42)
  modelo <- naiveBayes(
    LogSalePrice ~ .,
    data = train_set,
    laplace = parametros$laplace[i],
    usekernel = parametros$usekernel[i],
    adjust = parametros$adjust[i]
  )


  predicciones <- predict(modelo, newdata = test_set)


  if (!is.numeric(predicciones)) {
    predicciones <- as.numeric(as.character(predicciones))
```

```
  }


  mse <- mean((test_set$LogSalePrice - predicciones)^2, na.rm = TRUE)
  rmse <- sqrt(mse)
  error_dolares <- exp(rmse)


  resultados <- rbind(resultados, data.frame(
    Laplace = parametros$laplace[i],
    UseKernel = parametros$usekernel[i],
    Adjust = parametros$adjust[i],
    RMSE = rmse,
    ErrorDolares = error_dolares
  ))
}


resultados <- resultados[order(resultados$RMSE), ]
print(resultados)
```

Modelo de Regresion

```
##    Laplace UseKernel Adjust      RMSE ErrorDolares
## 1        0      TRUE    0.5 0.2395253     1.270646
## 4        0     FALSE    0.5 0.2395253     1.270646
## 7        0      TRUE    1.0 0.2395253     1.270646
## 10       0     FALSE    1.0 0.2395253     1.270646
## 13       0      TRUE    2.0 0.2395253     1.270646
## 16       0     FALSE    2.0 0.2395253     1.270646
## 2        1      TRUE    0.5 0.2675311     1.306734
## 5        1     FALSE    0.5 0.2675311     1.306734
## 8        1      TRUE    1.0 0.2675311     1.306734
## 11       1     FALSE    1.0 0.2675311     1.306734
## 14       1      TRUE    2.0 0.2675311     1.306734
## 17       1     FALSE    2.0 0.2675311     1.306734
## 3        5      TRUE    0.5 0.2747031     1.316140
## 6        5     FALSE    0.5 0.2747031     1.316140
## 9        5      TRUE    1.0 0.2747031     1.316140
## 12       5     FALSE    1.0 0.2747031     1.316140
## 15       5      TRUE    2.0 0.2747031     1.316140
## 18       5     FALSE    2.0 0.2747031     1.316140
```

```
mejor_modelo <- resultados[1, ]
cat("Mejor modelo:\n")
```

```
## Mejor modelo:
```

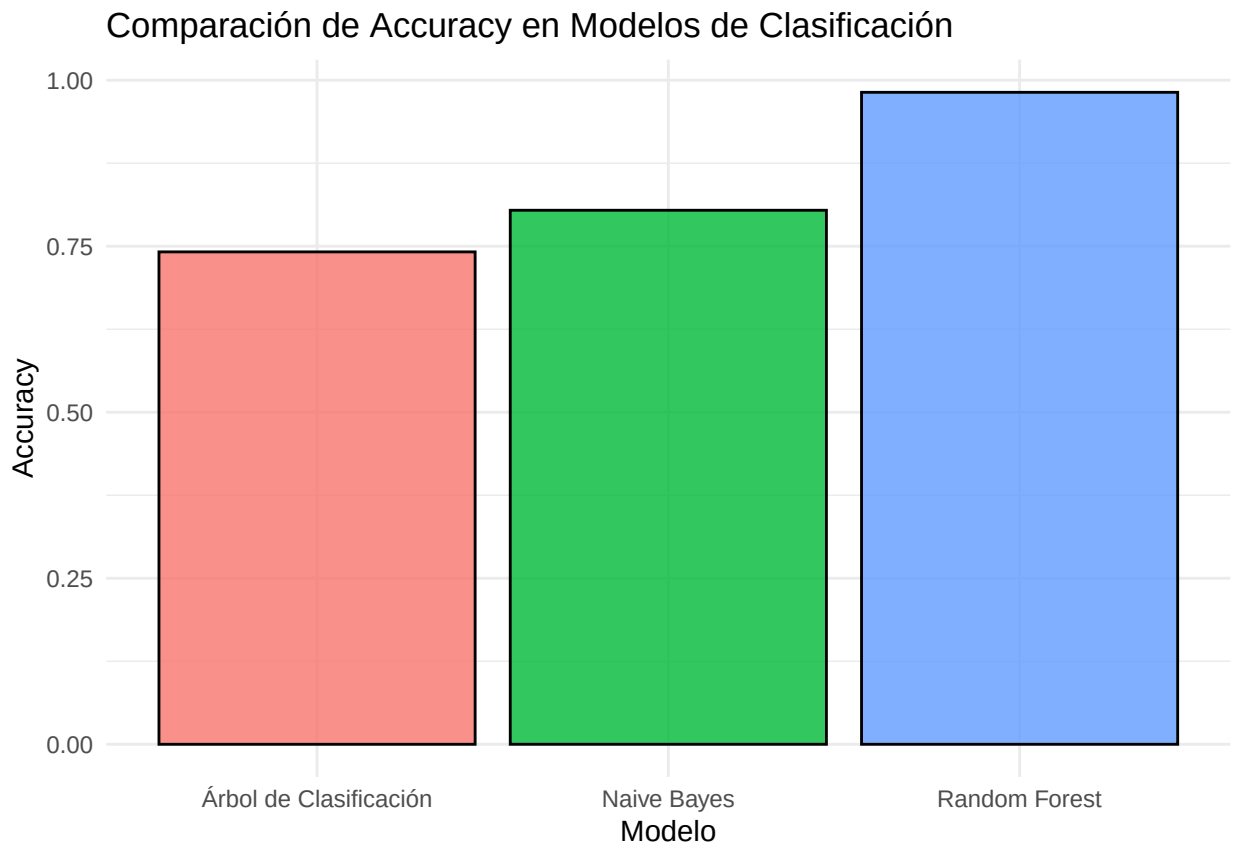
```
print(mejor_modelo)
```

```
##   Laplace UseKernel Adjust      RMSE ErrorDolares
## 1       0      TRUE    0.5 0.2395253     1.270646
```

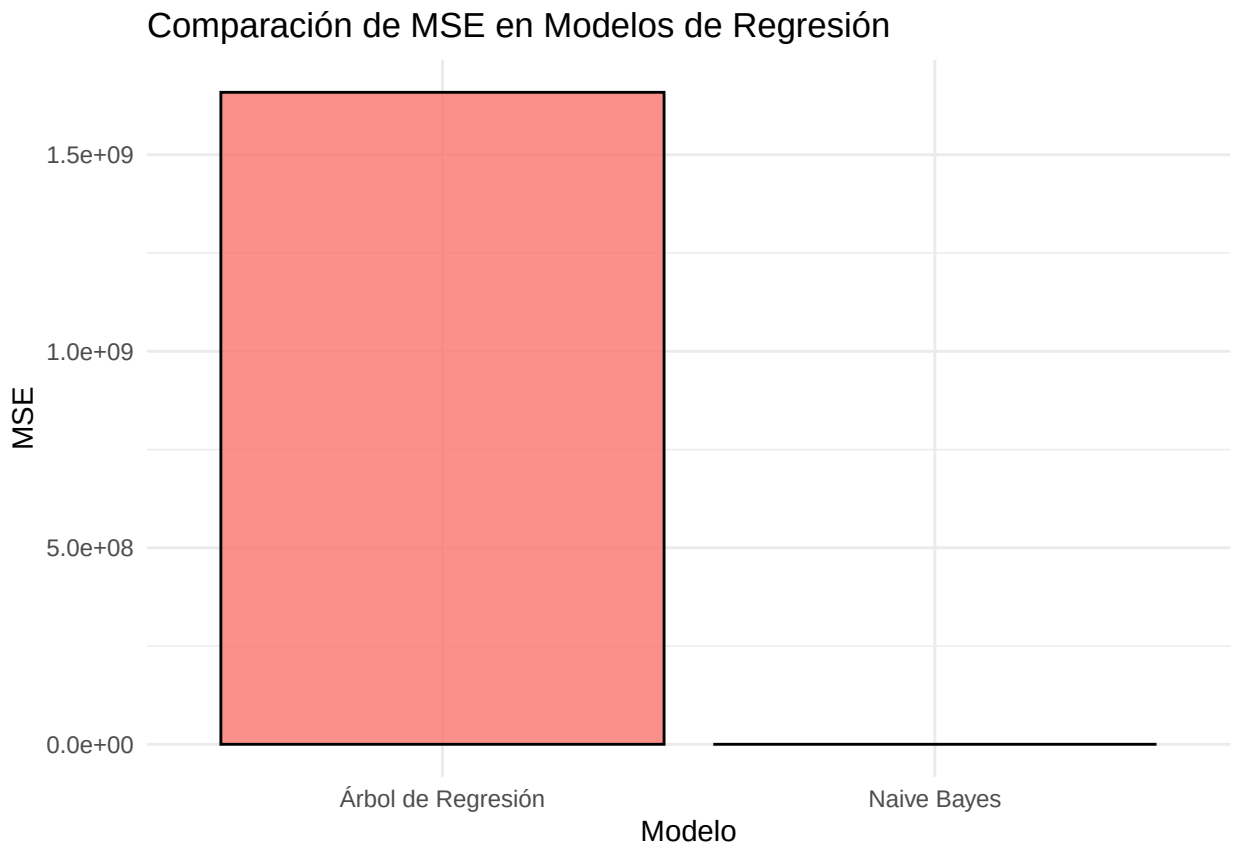
Analisis de Modelo de Regresion Podemos ver que cambiando los hiperparametros como laplace, ajuste, y el uso del kernel no afecta realmente nada en mejorar el RMSE y en ErrorDolares, de hecho podemos ver que realmente es casi igual a lo que teniamos anteriormente, esto indica que nuestro modelo esta bien generalizado y realmente no podemos mejorar ningun hiperparametro, por lo ques se quedara con los mismos parametros definidos anteriormente.

10. Compare la eficiencia del algoritmo con el resultado obtenido con el árbol de decisión (el de clasificación) y el modelo de random forest que hizo en la hoja pasada. ¿Cuál es mejor para predecir? ¿Cuál se demoró más en procesar?

```r
# Cargar librería
library(ggplot2)

# Datos de clasificación
clasificacion <- data.frame(
  Modelo = c("Naive Bayes", "Random Forest", "Árbol de Clasificación"),
  Accuracy = c(0.8041237, 0.9816934, 0.7414188)
)

# Gráfico de Accuracy (Clasificación)
ggplot(clasificacion, aes(x = Modelo, y = Accuracy, fill = Modelo)) +
  geom_bar(stat = "identity", color = "black", alpha = 0.8) +
  labs(title = "Comparación de Accuracy en Modelos de Clasificación",
       x = "Modelo",
       y = "Accuracy") +
  theme_minimal() +
  theme(legend.position = "none")
```




```r
# Datos de regresión
regresion <- data.frame(
  Modelo = c("Naive Bayes", "Árbol de Regresión"),
  MSE = c(0.05044311, 1658823049)
)

# Gráfico de MSE (Regresión)
ggplot(regresion, aes(x = Modelo, y = MSE, fill = Modelo)) +
  geom_bar(stat = "identity", color = "black", alpha = 0.8) +
  labs(title = "Comparación de MSE en Modelos de Regresión",
       x = "Modelo",
       y = "MSE") +
  theme_minimal() +
  theme(legend.position = "none")
```



Analisis de Modelos de Clasificacion Se puede ver que el mejor de los modelos es random Forest siendo de 0.9816934 , de hecho es el mejor de todos los modelos hechos, pero cabe aclarar que realmente a pesar de ser muy buen acurrancy puede estar sobreajustado ya que si bien se probaron con los tests en las mismas condiciones, en lo personal seria mejor ir por Naive Bayes ya que es mas generalista y tiene un buen acurrancy de 0.80.

Esto se debe a que Naive Bayes no usa todas las variables a diferencia de random forest, pero random forest y el arbol de clasificacion si ayuda a poder entender causualidad entre variables por lo que seria bueno en el analisis de este tipo y no en predicciones futuras.

De hehco el que mas se tardo mas en procesar fue el Random Forest debido a su profundidad de los datos y que se uso 100 arboles, y arbol de desicion fue el siguiente en tardarse pero naive bayes fue el mas rapido de todos

Analisis de Modelos de Regresion El mejor sin dudar alguna es Naive Bayes, de hecho el de regresion le fue bastante mal, en este caso si se usaria Naive Bayes, muy posiblemente a la naturaleza de los arboles de regresion y a su poca profundidad, afecto a la regresion de los datos.

Naive Bayes nos es un modelo simple pero efectivo, especialmente cuando los datos tienen cierta independencia entre las variables. En este caso, el árbol de regresión tuvo un desempeño deficiente probablemente talvez a que

- Falta de profundidad: Si los árboles no son lo suficientemente profundos
- Regresión en datos categóricos: Los árboles de regresión no siempre manejan bien datos categóricos sin una correcta transformación.

- Datos atipicos: Debido a su poca profundidad no detecto bien los valores atipicos y no los supo clasificar

De los 2 naive bayes se tardo menos que el de arbol de regresion esto hace que sea mas eficiente que el anterior.

Conclusion

En conclusion Naive Bayes si tuvo un buen desempeño, de hecho es el que es mas generalista de todos, superando incluso al arbol de regresion.

Naive Bayes si nos puede mostrar una buena prediccion de los datos , pero no nos indica causualidad de las variables algo que los arboles de clasificacion si logran hacerlo.

Y naive bayes mejoro mucho en eficiencia ya que se tarda mucho menos , esto debido a que no requiere entrenamiento complejo, solo calcula probabilidades condicionales para cada clase basándose en la frecuencia de las características.A diferencia de los otros que construyen arboles por cada iteracion.