PEC1 Análisis Estadístico Modelización Predictiva

Stwart Porras

2024-10-29

Contents

Carga de los datos

```
df <- read.csv("Astronomy.csv", header = TRUE, sep = ',')
```

```
head(df,5)
```

```
##   Temperatura Luminosidad   Radio Magnitud Tipo_re      Tipo_Cat      Color
## 1        7740     0.00049 0.01234    14.02       2   White Dwarf      White
## 2        8500     0.00050 0.01000    14.50       2   White Dwarf      White
## 3        8570     0.00081 0.00970    14.20       2   White Dwarf Blue white
## 4        8052     8.70000 1.80000     2.42       3 Main Sequence    Whitish
## 5        8930     0.00056 0.00950    13.78       2   White Dwarf      white
##   Clase_Espectral
## 1               A
## 2               A
## 3               A
## 4               A
## 5               A
```

```
summary(df)
```

```
##   Temperatura     Luminosidad           Radio              Magnitud
##  Min.   : 1939   Min.   :     0.0   Min.   :   0.0084   Min.   :-11.920
##  1st Qu.: 3344   1st Qu.:     0.0   1st Qu.:   0.1027   1st Qu.: -6.232
##  Median : 5776   Median :     0.1   Median :   0.7625   Median :  8.313
##  Mean   :10497   Mean   :107188.4   Mean   : 237.1578   Mean   :  4.382
##  3rd Qu.:15056   3rd Qu.:198050.0   3rd Qu.:  42.7500   3rd Qu.: 13.697
##  Max.   :40000   Max.   :849420.0   Max.   :1948.5000   Max.   : 20.060
##     Tipo_re      Tipo_Cat            Color           Clase_Espectral
##  Min.   :0.0   Length:240         Length:240         Length:240
##  1st Qu.:1.0   Class :character   Class :character   Class :character
##  Median :2.5   Mode  :character   Mode  :character   Mode  :character
##  Mean   :2.5
##  3rd Qu.:4.0
##  Max.   :5.0
```

```
sum(sapply(df, is.null))
```

```
## [1] 0
```

```
column_types <- sapply(df, class)
non_null_counts <- sapply(df, function(x) sum(!is.na(x)))

info_df <- data.frame(Column = names(df), Type = column_types,
                      NonNullCount = non_null_counts)
print(info_df)
```

```
##                          Column      Type NonNullCount
## Temperatura         Temperatura   integer          240
## Luminosidad         Luminosidad   numeric          240
## Radio                     Radio   numeric          240
## Magnitud               Magnitud   numeric          240
## Tipo_re                 Tipo_re   integer          240
## Tipo_Cat               Tipo_Cat character          240
## Color                     Color character          240
## Clase_Espectral Clase_Espectral character          240
```

1 Regresión Lineal

1.1 Estudio de correlación lineal y gráfico de dispersión

1.1.1 Correlación lineal

Para calcular la matriz de correlación lineal entre todas las variables cuantitativas de la base de datos se hará uso de:

```
library(ggplot2)
correlation_matrix <- cor(df[, sapply(df, is.numeric)])
```

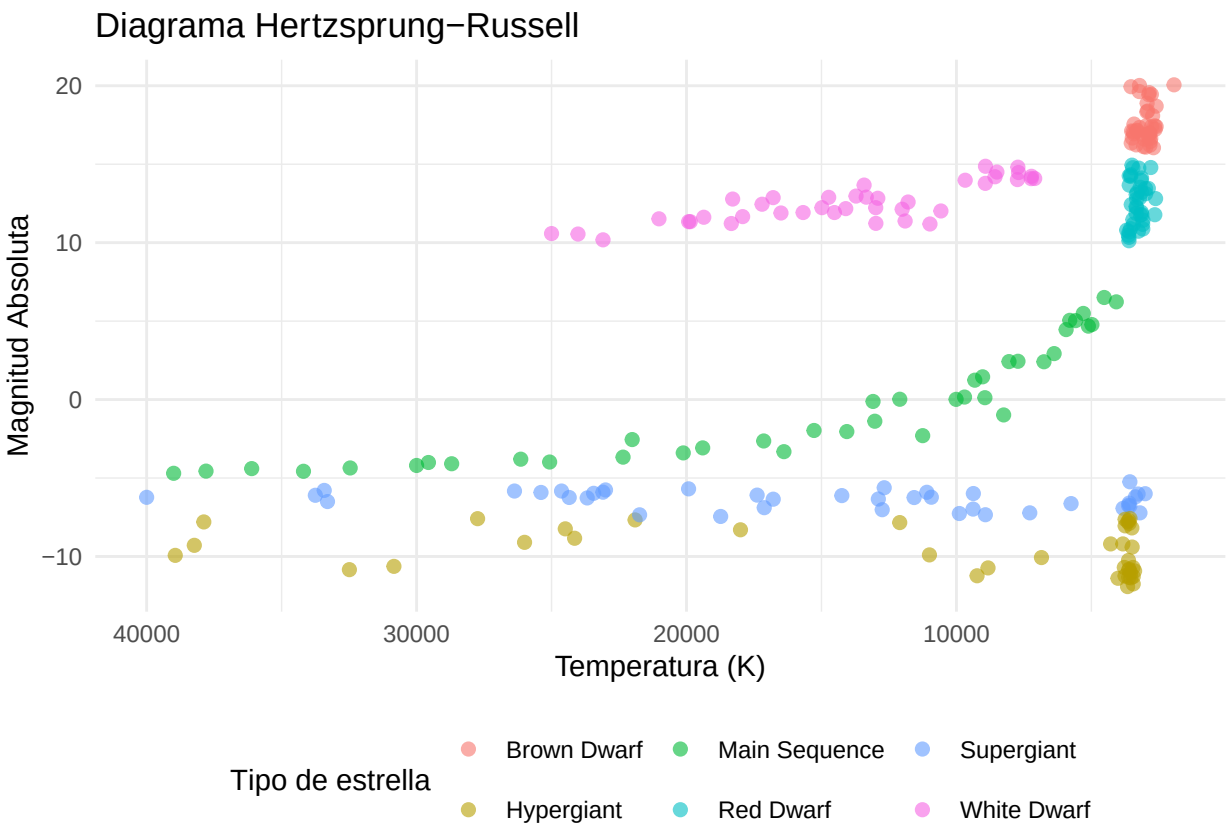
```
correlation_matrix
```

```
##             Temperatura Luminosidad       Radio   Magnitud    Tipo_re
## Temperatura  1.00000000   0.3934041  0.06421597 -0.4202605  0.4111288
## Luminosidad  0.39340408   1.0000000  0.52651572 -0.6926192  0.6768450
## Radio        0.06421597   0.5265157  1.00000000 -0.6087282  0.6609753
## Magnitud    -0.42026054  -0.6926192 -0.60872823  1.0000000 -0.9552756
## Tipo_re      0.41112879   0.6768450  0.66097527 -0.9552756  1.0000000
```

La matriz de correlación lineal nos da información sobre como se relacionan las variables cuantitativas entre sí, en este caso, linealmente. Una correlación grande en valor absoluto cercana a 1, nos indica una fuerte dependencia entre las variables. El signo indica si la relación es directa o inversamente proporcional. Obviamente cada variable tendrá un valor de correlación máxima con ella misma, de ahí que las diagonales sean (1).

Como se puede observar la Temperatura de una estrella es directamente proporcional a su Luminosidad en un 39.34%, y en tipo_re en un 41.11% e inversamente proporcional a la magnitud en un 42.02%. El mismo análisis se puede realizar para el resto de columnas. Los valores más altos de correlación lo tienen Magnitud con Tipo_re, con una correlación inversamente proporcional del 95.55%. ### 1.1.2 Diagrama HR (Hertzsprung-Russell) Para representar el diagrama HR, se hace uso de la libería ggplot, se usan los parámetros Temperatura y Magnitud, para los datos, y se agrupan por color según el tipo de estrella.

```
ggplot(df, aes(x = Temperatura, y = Magnitud, color = Tipo_Cat)) +
  geom_point(size = 2, alpha = 0.6) +
  scale_x_reverse() +
  labs(title = "Diagrama Hertzsprung-Russell",
       x = "Temperatura (K)",
       y = "Magnitud Absoluta",
       color = "Tipo de estrella") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



En un diagrama HR se indica la relación que tienen las estrellas con respecto a magnitudes absolutas. La magnitud absoluta, es la que tendría un objeto si estuviera a 10 pársecs (3x10^14 km) en un espacio con absorción interestelar. Cuanto más luminosa es una estrella menor es su valor de magnitud absoluta. Esto incluye valores negativos, lo que refiere a una gran luminosidad. Como se puede observar, las estrellas de menor luminosidad y temperatura son, las enanas blancas y marrones respectivamente. Es por ello que están más representadas más a la derecha (porque se invirtió el eje en el gráfico). Junto con ellas también están las enanas rojas. En cuanto a la secuencia principal, las supergigantes y las hyper gigantes se desarrollan a lo largo del gráfico, en el que las estrellas super gigantes son las que tiene valores de magnitud más negativos. Deberían ser también las de mayor temperatura pero se puede ver que las super gigantes no se quedan atrás.

1.2 Creación de una nueva variable y generación de los conjuntos de entrenamiento y de test

Se procede a juntar las categorías G y K de la variable Clase_Espectral.

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.2
```

```
##
## Attaching package: 'dplyr'
```

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

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

```r
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.4.2
```

```r
df <- df %>%
  mutate(Clase_Espectral = ifelse(Clase_Espectral == "G", "K", Clase_Espectral))
```

```r
unique(df$Clase_Espectral)
```

```
## [1] "A" "B" "F" "K" "M" "O"
```

Se comprueba que efectivamente la clase G no existe, es decir, el cambio de la clase G a K se ha realizado satisfactoriamente. Ahora se separan los datos en dos, el training y el de prueba a un 80% del original. Este paso es necesario para hacer un análisis predictivo, la idea es entrenear un subconjunto de datos (training) con todos los datos, para crear un modelo capaz de predecir alguno de los datos de manera efectiva.

```r
set.seed(42)
subsamples <- sample.split(df, SplitRatio = 0.8)
training <- subset(df, subsamples == TRUE)
testing <- subset(df, subsamples == FALSE)
```

Por último se comprueba el tamaño de ambas muestras.

```r
cat("Training: ",nrow(training), "\n")
```

```
## Training:  180
```

```r
cat("Testing: ", nrow(testing), "\n")
```

```
## Testing:  60
```

1.3 Estimación del modelo de regresión lineal con predictores cuantitativos

A continuación se realiza un modelo de estimación lineal por mínimos cuadrados la relación de la variable Luminosidad, con el resto de las variables cuantitativas. (Similar a la regresión lineal de Excel en diagramas de dispersión y el valor de R^2).

```r
RL2 <- lm(Luminosidad ~ ., data = training[, sapply(training, is.numeric)])
```

```r
summary(RL2)
```

```
##
## Call:
## lm(formula = Luminosidad ~ ., data = training[, sapply(training,
##     is.numeric)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -227315  -61207   -2537   35178  518394
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 119040.455  60489.029   1.968  0.05065 .
## Temperatura      4.584      1.093   4.195 4.32e-05 ***
## Radio           64.166     23.707   2.707  0.00747 **
## Magnitud     -9921.357   3013.669  -3.292  0.00120 **
## Tipo_re     -14164.408  19606.300  -0.722  0.47099
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 120500 on 175 degrees of freedom
## Multiple R-squared:  0.5368, Adjusted R-squared:  0.5263
## F-statistic: 50.71 on 4 and 175 DF,  p-value: < 2.2e-16
```

No hay que confundir los valores de multiple R-squared and adjusted R-squared con los valores de de R^2 individuales de sus respectivos ajustes vs Luminosidad. Ya que estos valores son los valores en general o en total y se calculan:

$$R^2 = \frac{\sum_i^n (y_i - \hat{y}_i)^2/(n-p-1)}{\sum_i^n (y_i - \overline{y})^2/(n-1)}$$

donde n es el número de datos y p el número de variables.

Y a continuación los valores de R^2 por individual.

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

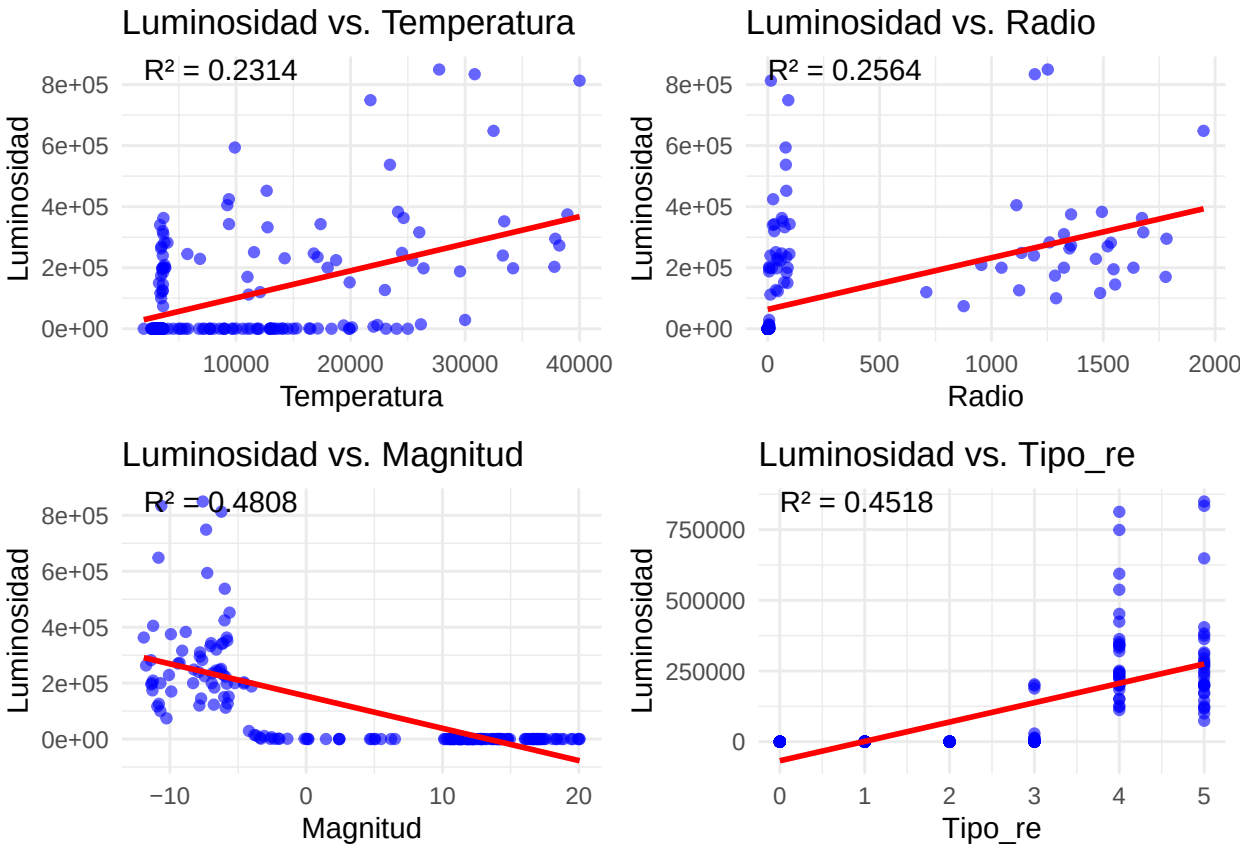
```
variables <- training[, sapply(training, is.numeric)]

relations <- variables[, !colnames(variables) %in% "Luminosidad"]
plots <- list()

for (var in colnames(relations)) {

  copy <- as.formula(paste("Luminosidad ~", var))
  model <- lm(copy, data = training)

  r_squared <- summary(model)$r.squared

  plot_grid <- ggplot(training, aes_string(x = var, y = "Luminosidad")) +
    geom_point(color = "blue", alpha = 0.6) +
    geom_smooth(method = "lm", color = "red", se = FALSE) +
    annotate("text", x = min(training[[var]], na.rm = TRUE),
                     y = max(training$Luminosidad, na.rm = TRUE),
                     label = paste("R² =", round(r_squared, 4)),
                     color = "black", size = 4, hjust = 0) +
    labs(title = paste("Luminosidad vs.", var),
         x = var,
         y = "Luminosidad") +
    theme_minimal()

  plots[[var]] <- plot_grid
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
do.call("grid.arrange", c(plots, ncol = 2))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



Volviendo al ejercicio,

```
summary(RL2)
```

```
##
## Call:
## lm(formula = Luminosidad ~ ., data = training[, sapply(training,
##     is.numeric)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -227315  -61207   -2537   35178  518394
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 119040.455  60489.029   1.968  0.05065 .
## Temperatura      4.584      1.093   4.195 4.32e-05 ***
## Radio           64.166     23.707   2.707  0.00747 **
## Magnitud     -9921.357   3013.669  -3.292  0.00120 **
```

```
## Tipo_re     -14164.408  19606.300  -0.722  0.47099
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 120500 on 175 degrees of freedom
## Multiple R-squared:  0.5368, Adjusted R-squared:  0.5263
## F-statistic: 50.71 on 4 and 175 DF,  p-value: < 2.2e-16
```

De los resultados del modelo se pueden destacar las relaciones de la Luminosidad con respecto a la Temperatura, Radio y Magnitud. Al fijarse en la columna (Pr>|t|), tienen valores de 4.32e-05, 0.00747 y 0.00120 respectivamente, los cuales son menores que 0.05 presentando una evidencia clara en contra de la hipótesis nula. Además el modelo explica aproximadamente el 53% de la variabilidad de la luminosidad se puede considerar como un buen modelo, a pesar de su valores residuales altos.

Sí, se puede afirmar que hay una relación entre la luminosidad y la magnitud, además es inversa, lo que encaja con el gráfico HR en la que las estrellas más luminosas tienen menores valores de magnitud.

1.3.1 Comprobación de la colinealidad.

Es necesario comprobar la colinealidad ya que ésta existe cuando las variables que se usan para predecir el modelo están relacionadas entre ellas. Lo que les impide predecir de manera efectiva el valor de la variable dependiente, reduciendo su significación estadística.

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.4.2
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.4.2
```

```
##
## Attaching package: 'car'
```

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

```
fiv <- vif(RL2)
```

```
print(fiv)
```

```
## Temperatura       Radio    Magnitud     Tipo_re
##    1.324138    1.902861   12.401198   13.993354
```

```
cat("\nUmbral: ", 1/(1-summary(RL2)$r.squared))
```

```
##
## Umbral:  2.15909
```

Como se puede observar las Variables Magnitud y Tipo_re son mayores que el umbral, por lo que la relación entre las variables predictoras es más fuerte que la relación entre ellas y la respuesta, hay una multicolinealidad clara.

La magnitud y Tipo_re son candidatos para la exclusión en este análisis, al menos con este método.

1.4 Estimación del modelo de regresión lineal con predictores cuantitativos y cualitativos

Se añadirá al modelo del apartado anterior la variable cualitativa Clase_Espectral.

```r
modelo_completo <- lm(Luminosidad ~ Temperatura + Radio +
                        Clase_Espectral, data = training)
```

```r
summary(modelo_completo)
```

```
##
## Call:
## lm(formula = Luminosidad ~ Temperatura + Radio + Clase_Espectral,
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -315914  -25860  -18729   -5358  599933
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -11940.342  34516.626  -0.346   0.7298
## Temperatura           3.013      1.648   1.828   0.0693 .
## Radio               146.051     16.383   8.915 6.95e-16 ***
## Clase_EspectralB  -4994.725  38314.205  -0.130   0.8964
## Clase_EspectralF -13797.860  43927.578  -0.314   0.7538
## Clase_EspectralK   9126.087  55401.182   0.165   0.8694
## Clase_EspectralM  21183.133  34873.965   0.607   0.5444
## Clase_EspectralO 248225.901  41361.776   6.001 1.13e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 109500 on 172 degrees of freedom
## Multiple R-squared:  0.6242, Adjusted R-squared:  0.6089
## F-statistic: 40.81 on 7 and 172 DF,  p-value: < 2.2e-16
```

Aquí hay que tener en cuenta varias cosas. Como se puede observar la variable Temperatura han pasado a tener un valor menos relevante. Por otro lado, la variable Clase_Espectral, que es cualitativa ha pasado a separarse en varias columnas dummy al modelo. Destacando notablemente la Clase_EspectalO con un valor p 1.13e-8. Ahora el R^2 es de 62.42%, y el ajustado de 61% por lo que explique una mayor variabilidad de la Luminosidad.

```r
fiv_completo <- vif(modelo_completo)
```

```r
fiv_completo
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## Temperatura     3.648113  1        1.910003
## Radio           1.100834  1        1.049206
## Clase_Espectral 3.694189  5        1.139599
```

```r
R2_modelo_completo <- summary(modelo_completo)$r.squared
```

```r
print(fiv_completo[, "GVIF^(1/(2*Df))"])
```

```
##     Temperatura           Radio Clase_Espectral
##        1.910003        1.049206        1.139599
```

```r
cat("\nUmbral: ", 1/(1-summary(modelo_completo)$r.squared))
```

```
##
## Umbral:  2.660874
```

En este caso hay que fijarse en la columna GVIFˆ(1/(2*Df)), ya que tiene en cuenta los grados de libertad, de nuevo como en el apartado anterior, las variables que superen el umbral como Magnitud, y Tipo_re, presentan una multicolinealidad. Éstas van a ser eliminadas, y se comprobará de nuevo la eficacia del modelo.

```
Clase_Espectral_dummies <- model.matrix(~ Clase_Espectral - 1, data = training)
training <- cbind(training, Clase_Espectral_dummies)
```

```
head(training,5)
```

```
##   Temperatura Luminosidad  Radio Magnitud Tipo_re      Tipo_Cat      Color
## 3        8570     8.1e-04 0.0097    14.20       2   White Dwarf Blue white
## 4        8052     8.7e+00 1.8000     2.42       3 Main Sequence    Whitish
## 5        8930     5.6e-04 0.0095    13.78       2   White Dwarf      white
## 6        9675     4.5e-04 0.0109    13.98       2   White Dwarf Blue White
## 7        9030     4.5e+01 2.6300     1.45       3 Main Sequence Blue-white
##   Clase_Espectral Clase_EspectralA Clase_EspectralB Clase_EspectralF
## 3               A                1                0                0
## 4               A                1                0                0
## 5               A                1                0                0
## 6               A                1                0                0
## 7               A                1                0                0
##   Clase_EspectralK Clase_EspectralM Clase_EspectralO
## 3                0                0                0
## 4                0                0                0
## 5                0                0                0
## 6                0                0                0
## 7                0                0                0
```

```
modelo_adaptado <- lm(Luminosidad ~ Temperatura + Radio + Clase_EspectralO +
                      Clase_EspectralM,
                      data = training)

summary(modelo_adaptado)
```

```
##
## Call:
## lm(formula = Luminosidad ~ Temperatura + Radio + Clase_EspectralO +
##     Clase_EspectralM, data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -316084  -25089  -18505   -5322  599343
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -15244.661  23088.457  -0.660   0.5099
## Temperatura           2.926      1.368   2.139   0.0338 *
## Radio               147.089     15.945   9.225   <2e-16 ***
## Clase_EspectralO 253125.597  26407.290   9.585   <2e-16 ***
## Clase_EspectralM  24501.540  23368.121   1.049   0.2959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 108600 on 175 degrees of freedom
## Multiple R-squared:  0.6238, Adjusted R-squared:  0.6152
## F-statistic: 72.53 on 4 and 175 DF,  p-value: < 2.2e-16
```

```r
R2_modelo_adaptado <- summary(modelo_adaptado)$r.squared
fiv_completo_adaptado <- vif(modelo_adaptado)
```

```r
print(fiv_completo_adaptado)
```

```
##      Temperatura            Radio Clase_EspectralO Clase_EspectralM
##         2.555178         1.059669         1.476946         2.072539
```

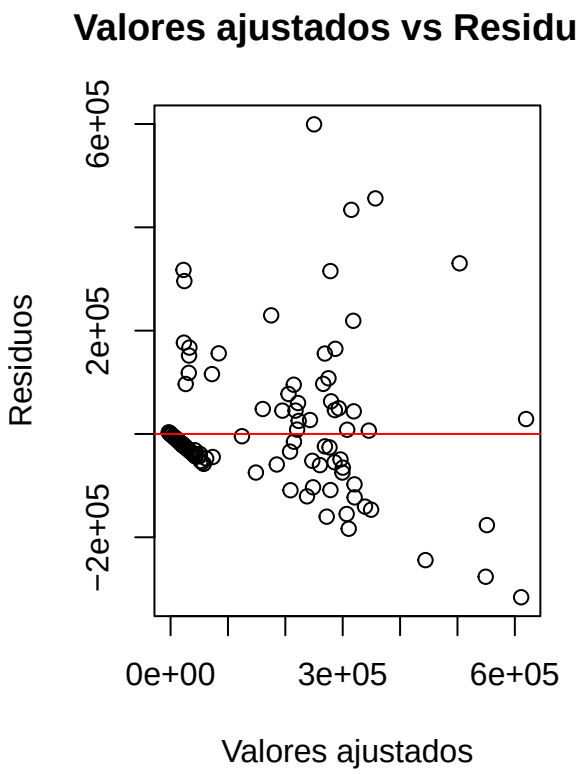
```r
cat("\nUmbral: ", 1/(1-summary(modelo_adaptado)$r.squared))
```

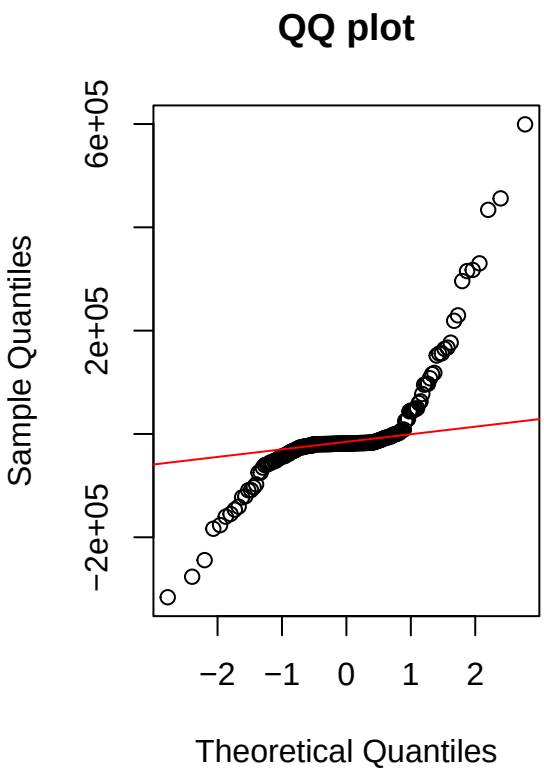
```
##
## Umbral:  2.657858
```

Como se puede observar ya no existe multicolinealidad, así que se seguirá adelante con este modelo. Aunque en este caso el R2 ajustado es ligeramente menor que en el anterior, sigue explicando el 62.38% de los datos, el anterior era 62.42%, pero lo que se pierde en %, se gana en performance. Pero al ser una diferencia ínfima, confirma que la importancia estadística de los valores eliminados no eran importantes.

1.5 Diagnosis del modelo

```r
par(mfrow = c(1,2))

residuos_modelo_adaptado <- residuals(modelo_adaptado)

valores_ajustados <- fitted(modelo_adaptado)

plot(valores_ajustados, residuos_modelo_adaptado,
     main = "Valores ajustados vs Residuos",
     xlab = "Valores ajustados",
     ylab = "Residuos",
     )
abline(h = 0, col = "red")
qqnorm(residuos_modelo_adaptado, main = "QQ plot")
qqline(residuos_modelo_adaptado, col = "red")
```






```r
par(mfrow = c(1,1))
```

```r
adaptado_df <- training[, c("Luminosidad", "Temperatura", "Radio",
                            "Clase_EspectralO", "Clase_EspectralM")]
```

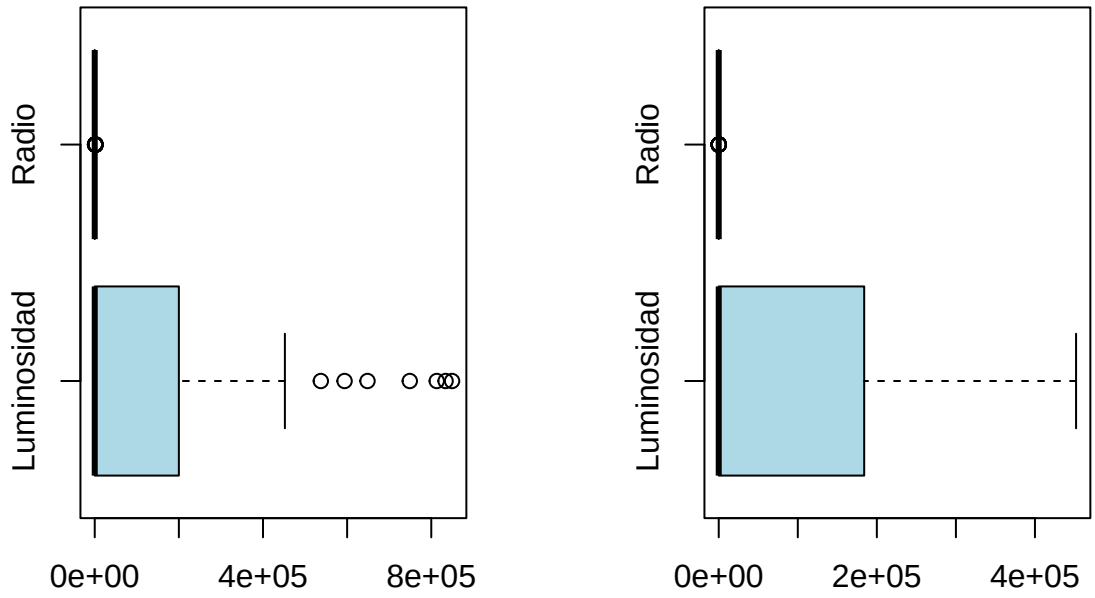
```r
head(adaptado_df,5)
```

```
##   Luminosidad Temperatura  Radio Clase_EspectralO Clase_EspectralM
## 3     8.1e-04        8570 0.0097                0                0
## 4     8.7e+00        8052 1.8000                0                0
## 5     5.6e-04        8930 0.0095                0                0
## 6     4.5e-04        9675 0.0109                0                0
## 7     4.5e+01        9030 2.6300                0                0
```

```r
par(mfrow = c(1, 2))

boxplot(adaptado_df$Luminosidad, adaptado_df$Radio,
        names = c("Luminosidad", "Radio"),
        main = "Boxplot of Luminosidad and Radio (With Outliers)",
        col = c("lightblue", "lightgreen"),
        horizontal = TRUE, outline = TRUE)

remove_outliers <- function(x) {
  Q1 <- quantile(x, 0.25)
  Q3 <- quantile(x, 0.75)
  IQR <- Q3 - Q1
  x[x < (Q1 - 1.5 * IQR) | x > (Q3 + 1.5 * IQR)] <- NA
```

```r
  return(x)
}

adaptado_df_no_outliers <- adaptado_df
adaptado_df_no_outliers$Luminosidad <- remove_outliers(adaptado_df$Luminosidad)
adaptado_df_no_outliers$Radio <- remove_outliers(adaptado_df$Radio)

boxplot(adaptado_df_no_outliers$Luminosidad, adaptado_df_no_outliers$Radio,
        names = c("Luminosidad", "Radio"),
        main = "Boxplot of Luminosidad and Radio (Without Outliers)",
        col = c("lightblue", "lightgreen"),
        horizontal = TRUE, outline = TRUE)
```

lot of Luminosidad and Radio (Witht of Luminosidad and Radio (Witho



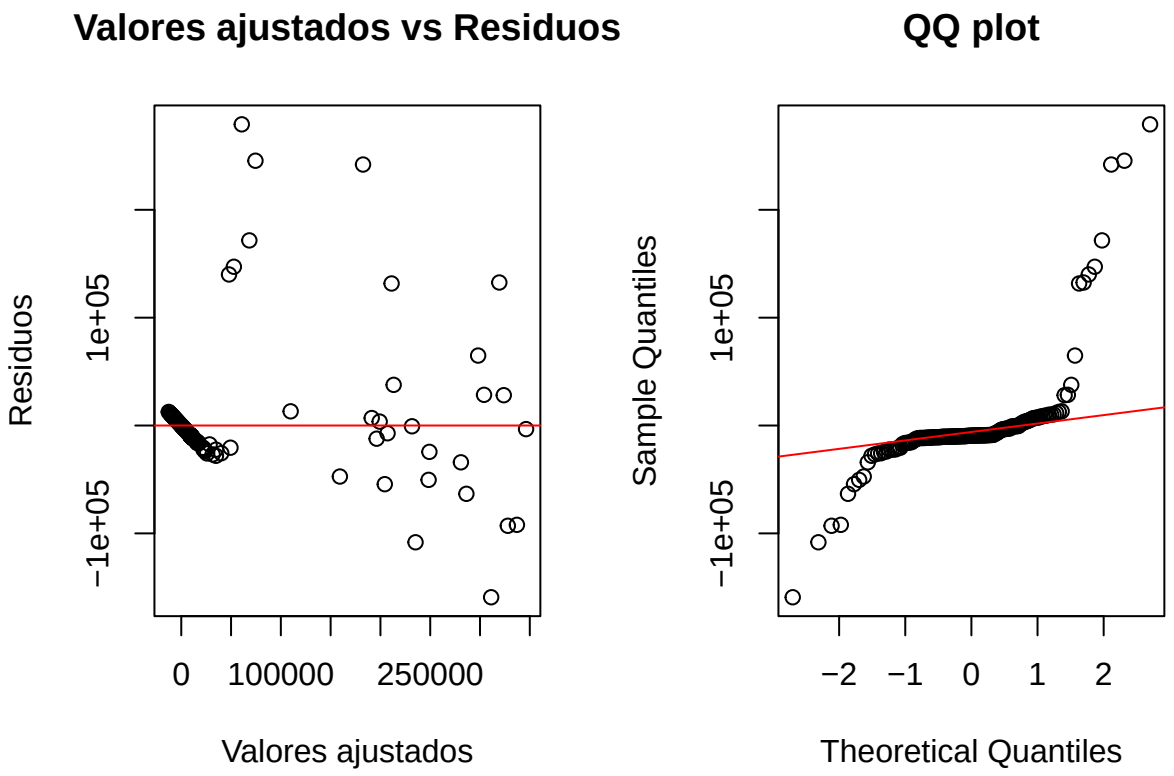

```r
par(mfrow = c(1, 1))
```

```r
modelo_adaptado_no_outliers <- lm(Luminosidad ~ Temperatura + Radio
                                  + Clase_Espectral0 +
                      Clase_EspectralM,
                      data = adaptado_df_no_outliers)

summary(modelo_adaptado_no_outliers)
```

```
##
## Call:
## lm(formula = Luminosidad ~ Temperatura + Radio + Clase_Espectral0 +
##     Clase_EspectralM, data = adaptado_df_no_outliers)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -159208  -11259   -9378    -705  279239
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -2.219e+04  1.333e+04  -1.665   0.0982 .
## Temperatura       1.922e+00  8.194e-01   2.346   0.0204 *
## Radio             2.212e+03  2.664e+02   8.305 7.69e-14 ***
## Clase_EspectralO  1.336e+05  1.991e+04   6.710 4.46e-10 ***
## Clase_EspectralM  2.560e+04  1.282e+04   1.997   0.0478 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 53030 on 140 degrees of freedom
##   (35 observations deleted due to missingness)
## Multiple R-squared:  0.761,  Adjusted R-squared:  0.7542
## F-statistic: 111.5 on 4 and 140 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(1,2))

residuos_modelo_adaptado_no_outliers <- residuals(modelo_adaptado_no_outliers)

valores_ajustados_no_outliers <- fitted(modelo_adaptado_no_outliers)

plot(valores_ajustados_no_outliers, residuos_modelo_adaptado_no_outliers,
     main = "Valores ajustados vs Residuos",
     xlab = "Valores ajustados",
     ylab = "Residuos",
     )
abline(h = 0, col = "red")
qqnorm(residuos_modelo_adaptado_no_outliers, main = "QQ plot")
qqline(residuos_modelo_adaptado_no_outliers, col = "red")
```




```
par(mfrow = c(1,1))
```

En el gráfico de dispersión los residuos se dispersan aleatoriamente alrededor del 0, sin formar patrones, lo que indica que la varianza es constante y el modelo es adecuado en términos de heterocdasticidad. Por otro lado en el QQ plot, la mayoría de valores están a lo largo de la línea diagonal roja, por lo que se va a hacer un box plot, para determinar outliers y mejorar el ajuste.

1.6 Predicción del modelo

```
testing <- testing[, c("Temperatura", "Luminosidad", "Radio",
                       "Clase_Espectral")]
dummy_vars <- model.matrix(~ Clase_Espectral - 1, data = testing)
testing <- testing[, -which(names(testing) == "Clase_Espectral")]
testing <- cbind(testing, dummy_vars)
head(testing)
```

```
##    Temperatura Luminosidad     Radio Clase_EspectralA Clase_EspectralB
## 1         7740 4.90000e-04 1.234e-02                1                0
## 2         8500 5.00000e-04 1.000e-02                1                0
## 9        12098 6.89000e+02 7.010e+00                1                0
## 10        8924 2.80000e-04 8.790e-03                1                0
## 17        9320 2.90000e+01 1.910e+00                1                0
## 18        8829 5.37493e+05 1.423e+03                1                0
##    Clase_EspectralF Clase_EspectralK Clase_EspectralM Clase_EspectralO
## 1                 0                0                0                0
## 2                 0                0                0                0
```

```
## 9                0                0                0                0
## 10               0                0                0                0
## 17               0                0                0                0
## 18               0                0                0                0
```

```r
predicted_values <- predict(modelo_adaptado, newdata = testing)
residuals <- testing$Luminosidad - predicted_values
squared_residuals <- residuals^2
mse <- mean(squared_residuals)
rmse <- sqrt(mse)

estrella <- data.frame(
  Temperatura = 15000,
  Radio = 8,
  Clase_EspectralM = 0,
  Clase_EspectralO = 0
)
prediccion_luminosidad <- predict(modelo_adaptado_no_outliers, estrella)
cat("Valor predecido de Luminosidad: ", prediccion_luminosidad)
```

```
## Valor predecido de Luminosidad:  24342.21
```

```r
cat("\nRMSE: ",rmse)
```

```
##
## RMSE:  143221.1
```

Como se puede observar el RMSE es bastante alto, por lo que la predicccción se aleja bastante del valor verdadero alrededor de (2907), por lo que habría que usar otras técnicas de ML para mejorar el modelo.

2. Regresión Logística

El objetivo en esta sección, será construir un modelo de regresión logística binaria que nos permita predecir si una estrella pertenece a la clasificación espectral M o no, es decir aquellas que se denominan estrellas rojas, por su color. Posteriormente se creará un modelo de regresión logística multinomial, para poder clasificar las estrellas según su tipo (Tipo_Cat).

2.1 Creación de nuevas variables y generación de los conjuntos de entrenamiento y de test

```r
df$Temp_re <- ifelse(df$Temperatura <= 4000, "Fría", "Caliente")
df$Clase_re <- ifelse(df$Clase_Espectral == "M", 1, 0)

set.seed(42)
split <- sample.split(df$Temp_re, SplitRatio = 0.8)
training <- subset(df, split == TRUE)
testing <- subset(df, split == FALSE)
```

2.2 Estimación del modelo de regresión logística con el conjunto de entrenamiento

```r
modelo_final <- glm(Clase_re ~ Radio + Magnitud + Luminosidad,
                        data = training,
                        family = binomial)
```

```r
summary(modelo_final)
```

```
##
## Call:
## glm(formula = Clase_re ~ Radio + Magnitud + Luminosidad, family = binomial,
##     data = training)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.003e+00  3.230e-01  -3.104  0.00191 **
## Radio        2.302e-03  4.707e-04   4.890 1.01e-06 ***
## Magnitud     1.082e-01  2.574e-02   4.205 2.62e-05 ***
## Luminosidad -2.453e-06  1.689e-06  -1.452  0.14642
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 265.15  on 191  degrees of freedom
## Residual deviance: 218.06  on 188  degrees of freedom
## AIC: 226.06
##
## Number of Fisher Scoring iterations: 4
```

```r
fiv_modelo_final <- vif(modelo_final)
print(fiv_modelo_final)
```

```
##       Radio    Magnitud Luminosidad
##    2.513324    2.657087    2.493373
```

Los valores presentan cierta colinealidad (valores entre 2 y 5) pero no son un problema al no ser valores muy altos, se van a dejar en el análisis.

2.3 Cálculo de las Odds_Ratio (OR) y matriz de confusión

2.3.1 Odds_Ratio

```r
OR <- exp(coef(modelo_final))
IC <- exp(confint(modelo_final))
```

```
## Waiting for profiling to be done...
```

```r
Tabla_resultados <- data.frame(
  Variable = names(coef(modelo_final)),
  OR = OR,
  Lower_IC = IC[,1],
  Upper_IC = IC[,2]
)
Tabla_resultados
```

```
##                Variable        OR  Lower_IC  Upper_IC
## (Intercept) (Intercept) 0.3668839 0.1890794 0.6766692
## Radio             Radio 1.0023044 1.0014339 1.0033003
## Magnitud       Magnitud 1.1142865 1.0610666 1.1743540
## Luminosidad Luminosidad 0.9999975 0.9999939 1.0000006
```

Magnitud es un factor de riesgo ya que la probabilidad de que la estrella pertenezca a la clase M aumenta en un 11.4% los valores de IC, están entre 1.061 y 1.174 lo que sugiere un factor de riesgo estadísticamente significativo, al no incluir el 1 en el rango. La lumininosidad es irrelevante estadísticamente por lo que se confirma en el análsis de los valores p su significancia y se procede a eliminarla del análisis.

```
modelo_final <- glm(Clase_re ~ Radio + Magnitud,
                        data = training,
                        family = binomial)
```

2.3.2 Matriz de confusión

```
predicted_values <- predict(modelo_final, newdata = testing, type = "response")
predictions <- ifelse(predicted_values >= 0.5, 1, 0)

confussion_matrix <- table(Predicted = predictions, Actual = testing$Clase_re)
print(confussion_matrix)
```

```
##           Actual
## Predicted  0  1
##         0 16  0
##         1 10 22
```

```
TP <- confussion_matrix[2, 2]  #Verdadero positivos
TN <- confussion_matrix[1, 1]  #Verdaderos negativos
FP <- confussion_matrix[2, 1]  #Falsos positivos
FN <- confussion_matrix[1, 2]  #Falsos negativos

sensibilidad <- TP / (TP + FN)
especificidad <- TN / (TN + FP)

cat("Sensibilidad:", sensibilidad, "\n")
```

```
## Sensibilidad: 1
```

```
cat("Especificidad:", especificidad, "\n")
```

```
## Especificidad: 0.6153846
```

La matriz de confusión es especialmente útil en los análisis predictivos, en especial en ML, muestra cuántos de los valores predichos fueron correcta e incorrectamente clasificados. La sensibilidad es la proporción de estrellas correctamente clasificadas en este caso, correctamente clasificadas como clase M, y la especificidad lo contrario, las estrellas que no son M, clasificadas efectivamente como no clase M.

2.4 Predicción

Identificad aquéllas estrellas cuya probabilidad de estar clasificadas con la letra "M", sea superior al 50%, según el modelo de predicciones contra el conjunto de prueba (testing) calculado anteriormente. Comparad este resultado con la clasificación espectral de las estrellas de la base de datos testing.

```
results <- data.frame(
  Probabilidad_M = predicted_values,
  Clase_real = testing$Clase_re
)
```

```
estrellas_clase_M <- results[results$Probabilidad_M > 0.5, ]
head(estrellas_clase_M,5)
```

```
##    Probabilidad_M Clase_real
```

```
## 1       0.6276581          0
## 2       0.6418503          0
## 21      0.5622991          0
## 28      0.5698161          0
## 30      0.5449636          0
```

```
comparacion <- table(
  Predicted = ifelse(results$Probabilidad_M > 0.5, 1, 0),
  Actual = results$Clase_real
)
head(comparacion,5)
```

```
##          Actual
## Predicted  0  1
##         0 16  0
##         1 10 22
```

Valores negativos reales hay 16. Falsos positivos hay 10, es decir, hubieron predicciones que decían que era de la clase M y realmente no lo eran. No hay falsos negativos. Por último valores positivos reales hay 22, es decir el modelo predijo que 22 valores eran M, y efectivamente eran M

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.4.2
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

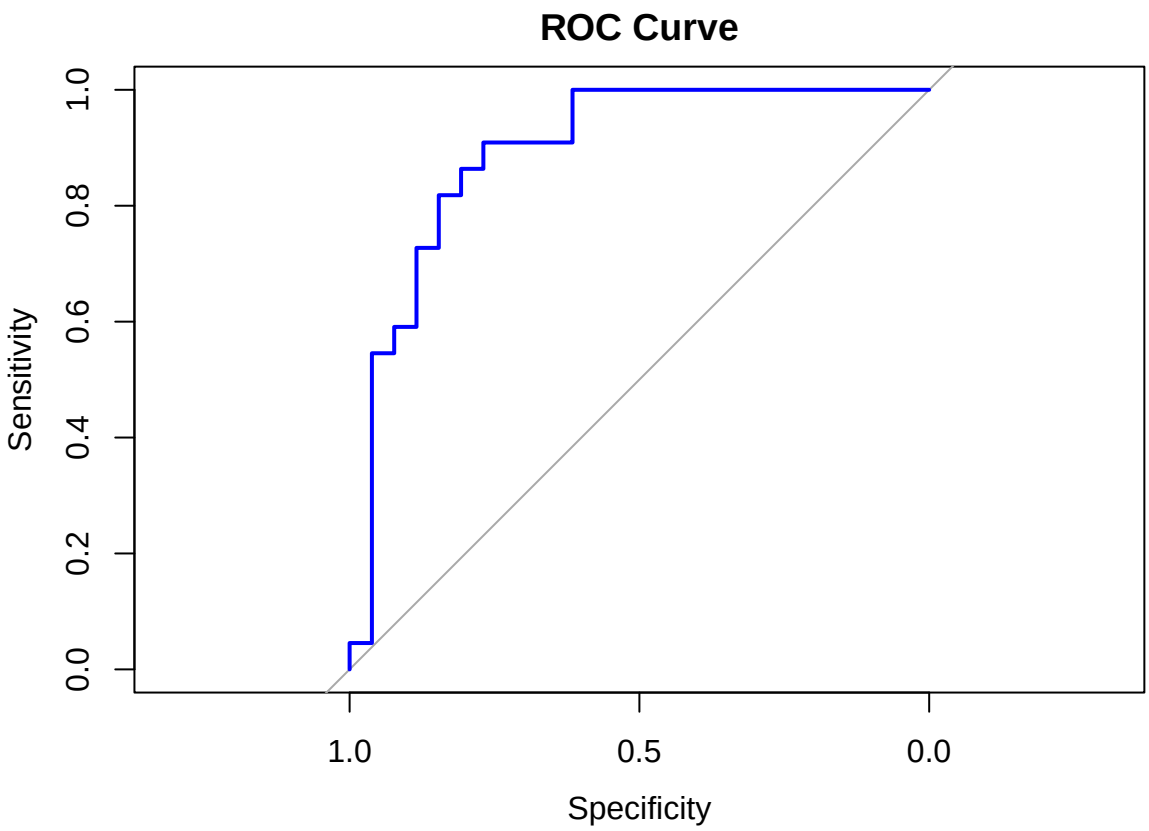
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
curva_roc <- roc(testing$Clase_re, predicted_values)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(curva_roc, col = "blue", lwd = 2, main = "ROC Curve")
```



La curva ROC mide la capacidad del modelo para distinguir las clases objetivo, el eje Y es la proporción de valores correctamente clasificadas como positivas y el eje X es la proporción de observaciones negativas clasfificadas como positivas de manera incorrecta. Al aproximarse la curva bastante al eje superior izquierdo indica que el modelo es bueno al diferenciar entre clases, estar cerca de la curva gris da lugar a que el modelo no sería mejor que una predicción aleatoria. El área debajo de la curca (AUC) está bastante cerca de 1 por lo que se puede confirmar que el modelo tiene una capacidad predictiva alta. La curva tiene un buen rendimiento.

```
auc_value <- auc(curva_roc)
cat("Area Under the Curve (AUC):", auc_value, "\n")
```

```
## Area Under the Curve (AUC): 0.8933566
```

2.6 Modelo de regresión logística multinomial

Se creará un modelo de regresión logística multinomial, para poder clasificar las estrellas según su tipo. Para ello se tomará como variable dependiente Tipo_Cat y variables explicativas Temp_re, Radio y Magnitud.

```
library(nnet)
```

```
## Warning: package 'nnet' was built under R version 4.4.2
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
training <- training[, c("Tipo_Cat","Temp_re","Radio","Magnitud")]
testing_ <- testing[, c("Tipo_Cat","Temp_re","Radio","Magnitud")]
```

De manera adicional se procede a escalar los datos numéricos.

```r
numerical_cols <- c("Radio", "Magnitud")
training[, numerical_cols] <- scale(training[, numerical_cols])
testing[, numerical_cols] <- scale(testing[, numerical_cols])
```

```r
Temp_re_dummies_train <- model.matrix(~ Temp_re - 1, data = training)
Temp_re_dummies_test <- model.matrix(~ Temp_re - 1, data = testing)

training <- cbind(training, Temp_re_dummies_train)
testing <- cbind(testing, Temp_re_dummies_test)
```

```r
training <- training[, c("Tipo_Cat","Temp_reCaliente","Temp_reFría","Radio",
                         "Magnitud")]
testing <- testing[, c("Tipo_Cat","Temp_reCaliente","Temp_reFría","Radio",
                       "Magnitud")]
```

```r
head(training,5)
```

```
##        Tipo_Cat Temp_reCaliente Temp_reFría      Radio   Magnitud
## 3   White Dwarf               1           0 -0.4619448  0.9628772
## 5   White Dwarf               1           0 -0.4619451  0.9231796
## 6   White Dwarf               1           0 -0.4619425  0.9420833
## 7 Main Sequence               1           0 -0.4569674 -0.2422282
## 8 Main Sequence               1           0 -0.4487043 -0.5966709
```

```r
head(testing,5)
```

```
##         Tipo_Cat Temp_reCaliente Temp_reFría      Radio   Magnitud
## 1    White Dwarf               1           0 -0.4412323  0.7908788
## 2    White Dwarf               1           0 -0.4412372  0.8374062
## 4  Main Sequence               1           0 -0.4375290 -0.3335331
## 13 Main Sequence               1           0 -0.4293567 -0.7658501
## 16 Main Sequence               1           0 -0.4288450 -0.5797406
```

```r
model <- multinom(Tipo_Cat ~ ., data = training)
```

```
## # weights:  36 (25 variable)
## initial  value 344.017818
## iter  10 value 57.488505
## iter  20 value 1.067000
## iter  30 value 0.024970
## iter  40 value 0.000561
## final  value 0.000073
## converged
```

```r
summary(model)
```

```
## Warning in sqrt(diag(vc)): NaNs produced
```

```
## Call:
## multinom(formula = Tipo_Cat ~ ., data = training)
##
## Coefficients:
##               (Intercept) Temp_reCaliente Temp_reFría      Radio   Magnitud
## Hypergiant    448.3582263        26.18523   422.17300 -334.35623 -1015.6614
## Main Sequence 313.5232497       298.35818    15.16507 -848.34397  -671.2458
## Red Dwarf     313.2038810      -284.75742   597.96130 -802.10854 -1196.6947
## Supergiant     73.2383481      -356.72124   429.95959 -879.95545 -1682.2496
```

```
## White Dwarf    -0.6600948       618.59688  -619.25698   62.81388   161.7716
##
## Std. Errors:
##                (Intercept) Temp_reCaliente  Temp_reFría        Radio
## Hypergiant    2.781368e+02    2.781368e+02 3.937674e-25 1.742585e+02
## Main Sequence 3.840736e-07    3.840735e-07          NaN 1.168788e-07
## Red Dwarf     2.178036e-09    7.316763e-59 2.178036e-09 6.379410e-10
## Supergiant    3.056600e+02    3.056600e+02 2.178109e-09 9.301244e+01
## White Dwarf   2.801530e+01    2.801530e+01 0.000000e+00 8.194304e+01
##                   Magnitud
## Hypergiant    2.475770e+02
## Main Sequence 3.496894e-07
## Red Dwarf     1.904948e-09
## Supergiant    2.782946e+02
## White Dwarf   3.127965e+01
##
## Residual Deviance: 0.0001450599
## AIC: 40.00015
```

```
predictions <- predict(model, newdata = testing)
```

```
conf_matrix <- table(Predicted = predictions, Actual = testing$Tipo_Cat)
print(conf_matrix)
```

```
##                Actual
## Predicted       Brown Dwarf Hypergiant Main Sequence Red Dwarf Supergiant
##   Brown Dwarf             8          0             0         0          0
##   Hypergiant              0          8             0         0          0
##   Main Sequence           0          0             7         0          0
##   Red Dwarf               1          0             0        11          0
##   Supergiant              0          0             2         0          4
##   White Dwarf             0          0             0         0          0
##                Actual
## Predicted       White Dwarf
##   Brown Dwarf             0
##   Hypergiant              0
##   Main Sequence           0
##   Red Dwarf               0
##   Supergiant              0
##   White Dwarf             7
```

```
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
print(paste("Accuracy: ", accuracy))
```

```
## [1] "Accuracy:  0.9375"
```

Confirmamos que el modelo es capaz de predecir los test con una precisión del 93%.

3 Conclusiones

Para finalizar concluimos con un análsis breve en el que en la primera parte de la prática se buscaba predecir la luminosidad de una estrella, en base a las variables Temperatura, Radio y las clases espectrales M y O, obtuvimos un resultado con mucho error, por lo que sabemos que el resultado predicho no está cercano del auténtico. Utilizando el QQ plot, podemos observar donde podría haber un problema, habría que escalar los datos, de manera para que todas las variables tengan el mismo rango.

Por otro lado en la segunda parte de la práctica hemos podido observar lo potente que son las técnicas de ML, la regresión logística binomial y multinomial, básicamente tratan de predecir si una variable es, o no es, en caso de la binomial, y la multinomial, pretender decir qué categoría entre todas las categorias que hayan es. Concluimos con que el resultado de la clasificación binomial y multinomial han sido satisfactorias.