Métodos Lineales de Regresión y Clasificación

ASIGNATURA: ALMACENES Y MINERÍA DE DATOS

PRÁCTICA 4

CURSO: 2025/26

Pascual Albericio, Irene (NIP 871627)

Vázquez Martín, Lucía (NIP 871886)

Índice

1. Introducción

En este informe se recoge el desarrollo de la práctica 4, centrada en la aplicación de **métodos lineales de regresión y clasificación** mediante lenguaje R, para resolver problemas complejos a partir de datos reales. El trabajo trata sobre 3 casos de estudio que necesitan estrategias analíticas como la identificación de compuestos químicos influyentes en la calidad de la cerveza mediante **selección automática de variables**; la predicción del destino de fardos mediante **regresión logística multinomial** combinada con optimización matemática; y la validación estadística de sesgos arbitrales en la Liga Nacional de Kárate mediante el **proceso ETL** y **modelos logísticos con interacción**.

2. Problema 1. Calidad de la cerveza

El objetivo de este primer problema consiste en determinar la influencia real que tienen 3 compuestos químicos específicos (*tcp, ma, tso2*), sobre la actividad antioxidante de la cerveza. Para ello, se dispone de un conjunto de datos (*lagerdata.csv*) que recoge las mediciones de 5 ensayos antioxidantes diferentes: *dsa, asa, orac, tp* y *mca.*

A través de técnicas de regresión lineal múltiple, se pretende identificar qué variables actúan como predictores significativos en cada ensayo y descartar aquellas que no aportan información relevante.

2.1. Interpretación de la figura 1

Como paso previo al desarrollo de los modelos, se ha procedido a interpretar los estadísticos del modelo inicial mostrado en el enunciado (regresión para la variable *dsa*).

```
> summary(lm(dsa~tpc+ma+tso2, data=beer.data))
 Coefficients:
             Estimate    Std. Error   t value  Pr(>|t|)
(Intercept) -0.0418869   0.0824789   -0.508   0.6147
tpc          0.0034765   0.0004144    8.390   5.42e-10 ***
ma           0.0032586   0.0065274    0.499   0.6207
tso2         0.0036574   0.0020140    1.816   0.0777 .

Residual standard error: 0.09466 on 36 degrees of freedom
Multiple R-squared: 0.7181,   Adjusted R-squared: 0.6946
F-statistic: 30.56 on 3 and 36 DF, p-value: 5.3e-10
```

Figura 1. Imagen a interpretar

El análisis de los valores marcados en rojo muestra las siguientes conclusiones:

- **Residual standard error (0.09466):** este valor representa la desviación estándar de los residuos, es decir, indica el error promedio que comete el modelo al intentar predecir la actividad *dsa*. Cuanto menor sea este valor, mayor precisión tendrán las predicciones.
- **Multiple R-squared (0.7181) y Adjusted R-squared (0.6946):** cuantifican la bondad de ajuste, indicando la precisión con la que el modelo matemático se adapta a los datos observados.
    - **Multiple R-squared (0.7181):** indica que el 71,81% de la variabilidad en la actividad antioxidante (dsa) se explica por las variables introducidas (tpc, ma, tso2). Sin embargo, este valor tiende a subir artificialmente siempre que se añaden más variables.
    - **Adjusted R-squared (0.6946):** esta es la métrica clave. Funciona igual que la anterior, pero aplica una penalización matemática si se incluyen variables innecesarias o redundantes.

Al obtener un 0.6946, se afirma que el modelo explica casi el 70% del comportamiento real de los antioxidantes (filtrando el ruido estadístico). Un ajuste cercano al 70% en este contexto se considera bastante robusto y valida la calidad del modelo para predecir.

- **F-statistic (30.56) y p-value ($5.3e^{-10}$):** estos indicadores evalúan si el modelo, en su conjunto, tiene capacidad real para predecir la actividad antioxidante o si, por el contrario, las variables elegidas (*tpc, ma, tso2*) no tienen ninguna relación real con el resultado (*dsa*).

    - **F-statistic (30.56):** compara la variabilidad que el modelo es capaz de explicar frente a la variabilidad que no explica. Un valor de 30.56 es considerablemente alto, y se determina que la influencia de los químicos es mayor que el ruido.

    - **P-valor ($5.3e^{-10}$):** se utiliza como estándar para aceptar un modelo. Si este valor es menor a 0.05/0.01 (depende de lo que se establezca), se puede rechazar con total seguridad la hipótesis de que ningún compuesto influye en la oxidación, como es el caso.

    También se puede afirmar con certeza que la relación detectada no es fruto del azar, sino que existe un efecto químico real y estadísticamente significativo.

- **Significatividad de los predictores (Pr(>|t|)):** al analizar los p-valores individuales, se observa bastante diferencia en la influencia de cada compuesto:

    - El **tpc ($p = 5.42e^{-10}$)** es altamente significativo (***), siendo el predictor principal.

    - La **ma ($p = 0.6207$)** no es significativa, lo que sugiere que no influye en este ensayo y que probablemente sólo aporta ruido.

    - El **tso2 ($p = 0.0777$)** presenta una significancia marginal (.). Aunque no alcanza el nivel de confianza exigente del 99% ($\alpha = 0.01$) o el estricto del 95% ($\alpha = 0.05$), sí sería significativo considerando un nivel de confianza del 90% ($\alpha = 0.10$). Esto sugiere que aporta información relevante al modelo y no debe descartarse automáticamente.

Con todos estos datos, se concluye que el modelo inicial es válido y robusto en términos globales. Sin embargo, el análisis individual de los coeficientes revela que no todas las variables contribuyen por igual. Esto justifica la necesidad de aplicar, en el siguiente apartado, un método de selección de variables para depurar el modelo y quedar solo con los componentes realmente influyentes.

2.2. Metodología de selección de predictores

Hay dos opciones para abordar el problema. La primera es probar manualmente todas las combinaciones posibles: con 3 predictores, esto significa crear y comparar 35 modelos diferentes (7 por cada ensayo). Aunque es factible, resulta tedioso y propenso a errores humanos al tener que comparar tantos estadísticos a ojo.

Por eso, se opta por automatizar el proceso mediante selección de predictores por pasos (stepwise selection). Este algoritmo hace exactamente lo mismo que se hace manualmente, pero de forma sistemática: prueba a añadir y quitar variables, calculando en cada paso el *Criterio de Información de Akaike* (AIC), que penaliza modelos innecesariamente complejos. Al final, se queda con el modelo más sencillo que mejor explica los datos.

2.3. Implementación y ejecución en R

Para llevar a cabo la selección de predictores, se ha desarrollado un script en R que automatiza el análisis para las cinco variables dependientes:

2.3.1. Configuración y carga de datos

El proceso inicia estableciendo el entorno de trabajo y cargando la información. Con *setwd(...)* se define el directorio de trabajo donde se encuentran los archivos, y con *read.csv(..., sep?";")* se carga el fichero *lagerdata.csv* en la variable *beer.data* (especificando que el archivo de origen usa el punto y coma como delimitador en vez de comas).

```
> setwd("C:/Users/irene/OneDrive/Escritorio/datasetsPractica4/datasets")
> getwd()
[1] "C:/Users/irene/OneDrive/Escritorio/datasetsPractica4/datasets"
> list.files()
[1] "datacala.csv"    "Karate2020.xlsx" "Karate2021.xlsx"
[4] "lagerdata.csv"
>
> beer.data <- read.csv("lagerdata.csv", sep=";")
> head(beer.data)
  beer    tpc    ma  tso2  dsa  asa orac   rp   mca
1    1 148.23 13.37  6.74 0.66 0.81 3.81 0.45 10.65
2    2 160.38 10.96 13.44 0.63 0.64 2.85 0.41 15.47
3    3 170.41  9.22  6.05 0.62 0.81 3.34 0.48 15.70
4    4 208.65  9.65 37.32 0.90 1.01 3.34 0.50 76.65
5    5 146.03 11.72  7.64 0.64 0.90 3.18 0.47  9.39
6    6 180.19  9.33  6.31 0.62 1.48 4.33 0.71 23.38
> str(beer.data)
'data.frame':   40 obs. of  9 variables:
 $ beer: int  1 2 3 4 5 6 7 8 9 10 ...
 $ tpc : num  148 160 170 209 146 ...
 $ ma  : num  13.37 10.96 9.22 9.65 11.72 ...
 $ tso2: num  6.74 13.44 6.05 37.32 7.64 ...
 $ dsa : num  0.66 0.63 0.62 0.9 0.64 0.62 0.58 0.47 0.59 0.72 ...
 $ asa : num  0.81 0.64 0.81 1.01 0.9 1.48 1.45 1.17 2.23 1.52 ...
 $ orac: num  3.81 2.85 3.34 3.34 3.18 4.33 5.82 1.5 2.5 2.55 ...
 $ rp  : num  0.45 0.41 0.48 0.5 0.47 0.71 0.68 0.57 0.65 0.63 ...
 $ mca : num  10.65 15.47 15.7 76.65 9.39 ...
```

Figura 2. Comprobación de la correcta carga de los datos *lagerdata.csv*

2.3.2. Definición del espacio de búsqueda

Antes de ejecutar el algoritmo, es necesario definir los límites entre los que el programa buscará el mejor modelo. Con la siguiente instrucción se define el límite superior de la búsqueda (donde indica que el modelo más complejo posible es aquel que incluye los 3 predictores: *tpc, ma, tso2.*

```
> # Definir la fórmula del modelo COMPLETO (superior)
> formula_completa <- as.formula("~ tpc + ma + tso2")
```

Figura 3. Definición de límites del mejor modelo en función de los predictores

2.3.3. Ejecución del algoritmo

Este bloque realiza la selección automática de variables. Se repite con la misma estructura para los cinco ensayos (*dsa, asa, orac, rp, mca*), por lo que se va a explicar en detalle la primera:

```
> modelo_dsa_completo <- lm(dsa ~ tpc + ma + tso2, data=beer.data)
> modelo_dsa_optimo <- step(modelo_dsa_completo,
+                           scope = list(lower = ~1, upper = formula_completa),
+                           direction = "both",
+                           trace = 0) # trace=0 para no mostrar todos los pasos
>
> summary(modelo_dsa_optimo)

Call:
lm(formula = dsa ~ tpc + tso2, data = beer.data)

Residuals:
      Min       1Q   Median       3Q      Max
-0.10807 -0.06153 -0.01832  0.03494  0.37108

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.0202398  0.0694448  -0.291   0.7723
tpc          0.0035661  0.0003696   9.648 1.21e-11 ***
tso2         0.0032960  0.0018602   1.772   0.0847 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.09369 on 37 degrees of freedom
Multiple R-squared:  0.7161,    Adjusted R-squared:  0.7008
F-statistic: 46.67 on 2 and 37 DF,  p-value: 7.644e-11
```

Figura 4. Resultados con el modelo formado por los predictores *tpc* y *tso2*

Se inicia el proceso ajustando el modelo completo con todas las variables. Con *step(...)* se ejecuta el algoritmo de selección paso a paso, donde se definen:

- *scope = list(...)*: para definir el rango de búsqueda con su límite inferior y superior.
- *direction = "both"*: permite la búsqueda bidireccional, es decir, evalúa en cada paso si debe eliminar una variable no significativa o si debe reintroducir una variable previamente descartada porque ha vuelto a ser relevante.
- *trace = 0*: no muestra los cálculos intermedios y muestra el resultado final de forma limpia.

Finalmente, mediante *summary(...)*, se muestran los estadísticos finales (coeficientes, p-valores, R^2…) del modelo ganador seleccionado para este ensayo. En la imagen se muestra cómo los predictores seleccionados finalmente han sido: *tpc* y *tso2*.

Una vez detallado el funcionamiento del algoritmo para el primer caso, se procede a replicar la metodología para las cuatro variables dependientes restantes. En todos los casos, se parte del modelo completo y se aplica el criterio de minimización de AIC para depurar los predictores no significativos.

A continuación, se muestra el código correspondiente a la ejecución secuencial de los modelos:

```
None
# Definir el modelo más complejo posible (con los 3 predictores)
formula_completa <- as.formula("~ tpc + ma + tso2")

# ------------------------------------------------------------------------
# PASO 1: Entrenar modelos para cada ensayo
# ------------------------------------------------------------------------
# El algoritmo step() prueba todas las combinaciones y se queda con la mejor
# según el criterio AIC (penaliza modelos innecesariamente complejos)
# DSA
modelo_dsa_completo <- lm(dsa ~ tpc + ma + tso2, data=beer.data)
modelo_dsa_optimo <- step(modelo_dsa_completo,
                          scope = list(lower = ~1, upper = formula_completa),
                          direction = "both", trace = 0)

# ASA
modelo_asa_completo <- lm(asa ~ tpc + ma + tso2, data=beer.data)
modelo_asa_optimo <- step(modelo_asa_completo,
                          scope = list(lower = ~1, upper = formula_completa),
                          direction = "both", trace = 0)

# ORAC
modelo_orac_completo <- lm(orac ~ tpc + ma + tso2, data=beer.data)
modelo_orac_optimo <- step(modelo_orac_completo,
                           scope = list(lower = ~1, upper = formula_completa),
                           direction = "both", trace = 0)

# RP
modelo_rp_completo <- lm(rp ~ tpc + ma + tso2, data=beer.data)
modelo_rp_optimo <- step(modelo_rp_completo,
                         scope = list(lower = ~1, upper = formula_completa),
                         direction = "both", trace = 0)
```

```
# MCA
modelo_mca_completo <- lm(mca ~ tpc + ma + tso2, data=beer.data)
modelo_mca_optimo <- step(modelo_mca_completo,
                          scope = list(lower = ~1, upper = formula_completa),
                          direction = "both", trace = 0)
```

2.3.4. Obtención de resultados y resumen

Para facilitar la interpretación final de los datos, se ha incluido un último bloque de código destinado a la síntesis de resultados:

```
None
# ------------------------------------------------------------------------------
# PASO 2: Crear tabla resumen con los resultados
# ------------------------------------------------------------------------------
# Función auxiliar: extrae qué predictores quedaron en el modelo final
extraer_predictores <- function(modelo) {
  coefs <- names(coef(modelo))
  predictores <- coefs[coefs != "(Intercept)"]
  if (length(predictores) == 0) return("Ninguno")
  return(paste(predictores, collapse = " + "))
}

# Construir la tabla comparativa
resultados <- data.frame(
  Ensayo = c("dsa", "asa", "orac", "rp", "mca"),
  Predictores = c(
    extraer_predictores(modelo_dsa_optimo),
    extraer_predictores(modelo_asa_optimo),
    extraer_predictores(modelo_orac_optimo),
    extraer_predictores(modelo_rp_optimo),
    extraer_predictores(modelo_mca_optimo)
  ),
  R2_Ajustado = round(c(
    summary(modelo_dsa_optimo)$adj.r.squared,
    summary(modelo_asa_optimo)$adj.r.squared,
    summary(modelo_orac_optimo)$adj.r.squared,
    summary(modelo_rp_optimo)$adj.r.squared,
    summary(modelo_mca_optimo)$adj.r.squared
  ), 4)
)

print(resultados)

# ------------------------------------------------------------------------------
# PASO 3: Ver los detalles estadísticos de cada modelo
# ------------------------------------------------------------------------------
cat("\n=== MODELO DSA ===\n")
summary(modelo_dsa_optimo)

cat("\n=== MODELO ASA ===\n")
summary(modelo_asa_optimo)
```

```
cat("\n=== MODELO ORAC ===\n")
summary(modelo_orac_optimo)

cat("\n=== MODELO RP ===\n")
summary(modelo_rp_optimo)

cat("\n=== MODELO MCA ===\n")
summary(modelo_mca_optimo)
```

En primer lugar, se ha definido una función auxiliar (*extraer_predictores*) que lee el objeto del modelo óptimo y extrae los nombres de las variables seleccionadas. Posteriormente, se crea un data frame (*resultados_seleccion*) que recopila el nombre del ensayo, la combinación final de predictores y el valor del R^2 ajustado.

	Ensayo	Predictores_Óptimos	R2_Ajustado
1	dsa	tpc + tso2	0.7007632
2	asa	tpc	0.2647691
3	orac	tpc + ma	0.1580159
4	rp	tpc + tso2	0.5758896
5	mca	tpc	0.3706069

Figura 5. Contenido de *resultados_seleccion*

Finalmente, mediante los comandos *summary(...)* se imprimen por consola los estadísticos detallados de cada modelo óptimo.

```
> summary(modelo_dsa_optimo)

Call:
lm(formula = dsa ~ tpc + tso2, data = beer.data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.10807 -0.06153 -0.01832  0.03494  0.37108

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.0202398  0.0694448  -0.291   0.7723
tpc          0.0035661  0.0003696   9.648 1.21e-11 ***
tso2         0.0032960  0.0018602   1.772   0.0847 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.09369 on 37 degrees of freedom
Multiple R-squared:  0.7161,    Adjusted R-squared:  0.7008
F-statistic: 46.67 on 2 and 37 DF,  p-value: 7.644e-11
```

```
> summary(modelo_asa_optimo)

Call:
lm(formula = asa ~ tpc, data = beer.data)

Residuals:
     Min       1Q   Median       3Q      Max
-1.09745 -0.13653  0.08454  0.17441  0.94192

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.480533   0.226812   2.119 0.040716 *
tpc         0.005079   0.001310   3.879 0.000404 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3414 on 38 degrees of freedom
Multiple R-squared:  0.2836,    Adjusted R-squared:  0.2648
F-statistic: 15.04 on 1 and 38 DF,  p-value: 0.000404
```

```
> summary(modelo_orac_optimo)

Call:
lm(formula = orac ~ tpc + ma, data = beer.data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1563 -1.4871  0.1516  1.3949  5.1867

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.329002   1.637808   0.811  0.42230
tpc          0.029523   0.009774   3.020  0.00456 **
ma          -0.264516   0.143800  -1.839  0.07388 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.235 on 37 degrees of freedom
Multiple R-squared:  0.2012,    Adjusted R-squared:  0.158
F-statistic:  4.66 on 2 and 37 DF,  p-value: 0.01567
```

```
> summary(modelo_rp_optimo)

Call:
lm(formula = rp ~ tpc + tso2, data = beer.data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.23257 -0.05460 -0.01241  0.06315  0.32396

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.2259917  0.0890036   2.539   0.0154 *
tpc          0.0030284  0.0004737   6.393 1.85e-07 ***
tso2        -0.0051425  0.0023841  -2.157   0.0376 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1201 on 37 degrees of freedom
Multiple R-squared:  0.5976,    Adjusted R-squared:  0.5759
F-statistic: 27.48 on 2 and 37 DF,  p-value: 4.846e-08
```

```
> summary(modelo_mca_optimo)

Call:
lm(formula = mca ~ tpc, data = beer.data)

Residuals:
    Min      1Q  Median      3Q     Max
-41.853 -11.846  -0.313   9.788  44.982

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -25.93648   11.70377  -2.216   0.0327 *
tpc           0.33079    0.06757   4.895 1.84e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.62 on 38 degrees of freedom
Multiple R-squared:  0.3867,    Adjusted R-squared:  0.3706
F-statistic: 23.96 on 1 and 38 DF,  p-value: 1.84e-05
```

Figura 6. Información más detallada del contenido de *resultados_seleccion*

2.4. Análisis detallado de los resultados

A continuación, se van a explicar las conclusiones sacadas a partir de los resultados obtenidos:

- **Modelos con alta capacidad explicativa**

En los ensayos de *dsa* y *rp*, el análisis de regresión ha sido capaz de explicar más de la mitad de la variabilidad de los datos, lo que indica que hay una relación química fuerte entre los predictores seleccionados y la respuesta.

  - **Ensayo dsa:** el modelo resultante (*dsa ~ tcp + tso2*) es el más robusto de todo el estudio y explica el 70.1% de la varianza. El predictor principal es *tcp*, ya que su p-valor es menor que el 0.01. Por otro lado, el predictor *tso2* tiene un p-valor individual marginal ($p \approx 0.08$), y el algoritmo lo ha retenido porque su inclusión mejora la calidad global del modelo.

  - **Ensayo rp:** el algoritmo seleccionó la misma combinación de variables, logrando explicar un 57.6% del comportamiento. A diferencia del caso anterior, el *tso2* muestra una significancia estadística mayor ($p < 0.05$), por eso está marcado con *. Esto confirma su importancia en las reacciones de reducción. Es interesante resaltar que según los coeficientes, el *tpc* contribuye positivamente, mientras que el *tso2* tiene un coeficiente negativo (tendrá una interacción inversa en este ensayo).

- **Modelos con capacidad explicativa moderada o baja**

Para los ensayos *mca*, *asa* y *orac*, los modelos obtenidos presentan un R^2 inferior a 0.4. Esto indica que, aunque existen relaciones significativas, gran parte de la variabilidad de estos ensayos depende de factores químicos que no están contemplados en este conjunto de datos.

  - **Ensayos mca y asa:** el modelo explica un 37.1% y un 26.5% de la variabilidad respectivamente. Esta baja capacidad predictiva sugiere que el *tpc* no es suficiente por sí solo para modelar con precisión.

  - **Ensayo orac:** es el peor ajuste, este ensayo tiene el resultado más deficiente del estudio con un 15.8% de ajuste. El predictor *ma* tiene una significancia marginal y tiene un R^2 tan bajo que muestra que la melanoidina tiene una influencia despreciable o nula en la capacidad antioxidante de la cerveza.

El análisis realizado permite concluir que el predictor *tpc* es el determinable indiscutible de la actividad antioxidante en la cerveza, influyendo significativamente en todos los ensayos. Además, el *tso2* también ayuda a mejorar la precisión en los modelos de mayor ajuste, mientras que la *ma* ha demostrado carecer de relevancia estadística. Por tanto, el control de la oxidación debe centrarse prioritariamente en el contenido fenólico, es decir, en el *tpc.*

3. Problema 2. Un problema de clasificación

El objetivo de este problema es desarrollar un modelo predictivo capaz de clasificar el destino de unos fardos lanzados al mar en función de sus coordenadas de lanzamiento. Los fardos pueden llegar a tres posibles destinos (Cala 0, Cala 1 y Cala 2), por lo que en este problema de clasificación, se tiene una variable respuesta categórica de más de dos niveles.

3.1. Metodología

Para abordar el problema, se ha optado por la regresión logística multinomial, ya que es ideal para predecir la probabilidad de pertenencia a una categoría nominal (las calas) a partir de variables predictoras continuas (latitud y longitud). Para la implementación en R, se ha utilizado la librería *nnet* y la función *multinom()*.

3.2. Implementación y ejecución en R

Para llevar a cabo la clasificación de los fardos, se ha desarrollado un script en R que automatiza todo el proceso de análisis, desde la carga de datos hasta la visualización de resultados.

3.2.1. Configuración y carga de datos

El proceso inicia estableciendo el entorno de trabajo y cargando la información necesaria. Con *setwd(...)* se define el directorio de trabajo donde se encuentran los archivos, y con *read.csv(...)* se carga el fichero *datacala.csv* en la variable *cala.data*.

Posteriormente, es necesario convertir la variable cala a tipo factor, para que R sea capaz de entender que se trata de una variable categórica (no numérica ordinal):

```
None
cala.data$cala <- as.factor(cala.data$cala)
```

```
> setwd("C:/Users/irene/OneDrive/Escritorio/datasetsPractica4/datasets")
> getwd()
[1] "C:/Users/irene/OneDrive/Escritorio/datasetsPractica4/datasets"
> list.files()
[1] "datacala.csv"    "Karate2020.xlsx" "Karate2021.xlsx" "lagerdata.csv"
>
> cala.data <- read.csv("datacala.csv")
>
> head(cala.data)
  X longitud.x latitud.y cala
1 1   9.562446  1.094132    1
2 2   9.570058  1.079876    0
3 3   9.560222  1.098586    2
4 4   9.566776  1.085960    0
5 5   9.571361  1.078007    0
6 6   9.562661  1.093751    1
> str(cala.data)
'data.frame':   600 obs. of  4 variables:
 $ X         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ longitud.x: num  9.56 9.57 9.56 9.57 9.57 ...
 $ latitud.y : num  1.09 1.08 1.1 1.09 1.08 ...
 $ cala      : int  1 0 2 0 0 1 2 0 2 1 ...
> summary(cala.data)
       X           longitud.x      latitud.y          cala
 Min.   :  1.0   Min.   :9.554   Min.   :1.076   Min.   :0
 1st Qu.:150.8   1st Qu.:9.558   1st Qu.:1.084   1st Qu.:0
 Median :300.5   Median :9.563   Median :1.094   Median :1
 Mean   :300.5   Mean   :9.563   Mean   :1.093   Mean   :1
 3rd Qu.:450.2   3rd Qu.:9.568   3rd Qu.:1.102   3rd Qu.:2
 Max.   :600.0   Max.   :9.572   Max.   :1.109   Max.   :2
>
```

Figura 7. Comprobación de la correcta carga de los datos *datacala.csv*

3.2.2. Identificación de puntos extremos de la playa

Antes de entrenar el modelo, se localizan los puntos con menor y mayor latitud, es decir, los límites geográficos que representan el punto más al sur y más al norte de la playa respectivamente.

```
None
idx_sur <- which.min(cala.data$latitud.y)
punto_sur <- cala.data[idx_sur, c("longitud.x", "latitud.y")]

idx_norte <- which.max(cala.data$latitud.y)
punto_norte <- cala.data[idx_norte, c("longitud.x", "latitud.y")]
```

```
> # Mostrar las coordenadas
> print("Coordenadas del punto más al Sur:")
[1] "Coordenadas del punto más al Sur:"
> print(punto_sur)
    longitud.x latitud.y
196   9.571293  1.076168
> print("Coordenadas del punto más al Norte:")
[1] "Coordenadas del punto más al Norte:"
> print(punto_norte)
    longitud.x latitud.y
115     9.5537  1.108584
```

Figura 8. Coordenadas punto más al sur y más al norte

3.2.3. Entrenamiento del modelo multinomial

Se procede a entrenar el modelo de regresión logística multinomial utilizando la función *multinom()* de la librería *nnet*, lo cual permite clasificar el destino de los fardos entre las calas 0, 1 y 2 usando como predictores las coordenadas geográficas de longitud y latitud.

```
> modelo_multinom <- multinom(cala ~ longitud.x + latitud.y, data = cala.data)
# weights:  12 (6 variable)
initial  value 659.167373
iter  10 value 330.179132
iter  20 value 281.310994
iter  30 value 209.582722
iter  40 value 208.228525
iter  40 value 208.228525
final  value 208.228525
converged
> summary(modelo_multinom)
Call:
multinom(formula = cala ~ longitud.x + latitud.y, data = cala.data)

Coefficients:
  (Intercept) longitud.x latitud.y
1    28.25784  -80.58581  683.1711
2    51.41181 -146.38143 1234.2198

Std. Errors:
  (Intercept) longitud.x latitud.y
1   1.2873455   3.633008  30.79643
2   0.9554057   2.798712  23.06487

Residual Deviance: 416.4571
AIC: 428.4571
```

Figura 9. Entrenamiento y coeficientes del modelo de regresión logística multinomial

El modelo toma la Cala 0 (Sur) como categoría de referencia, lo que se puede observar en el resultado devuelto por *summary(modelo_multinom),* donde solo se muestran coeficientes para las categorías 1 y 2, lo cual indica que la categoría 0 (ausente) es la base de comparación. En regresión logística multinomial, R utiliza por defecto el primer nivel del factor como referencia, que en este caso corresponde a *levels(cala.data$cala)[1] = "0"*.

Los coeficientes para las Calas 1 y 2 son considerablemente altos, ya que tienen valores superiores a 600 para *latitud.y*. Esto se debe a que la escala de las coordenadas geográficas es muy pequeña debido a que las latitudes varían entre 1.081 y 1.1. Esto significa que cambios pequeños en la latitud tienen un gran impacto en la probabilidad que un fardo llegue a una cala u otra, confirmando de este modo que el predictor dominante del modelo es la latitud, lo cual es coherente con la disposición geográfica Norte-Sur de las calas.

3.2.4. Definición de funciones auxiliares de predicción

Para facilitar el uso del modelo entrenado, se han creado dos funciones auxiliares: *predict_cala_clase* y *predict_cala_prob*, que devuelven respectivamente la cala predicha (0, 1 o 2) para unas coordenadas dadas y un vector con las probabilidades de que llegue a cada una de las calas (todas las probabilidades suman 1).

```
None
# ------------------------------------------------------------------------------
# PASO 3: Funciones auxiliares para predecir
# ------------------------------------------------------------------------------
# Devuelve la cala predicha (0, 1 o 2)
predict_cala_clase <- function(x_coord, y_coord, model) {
  new_data <- data.frame(longitud.x = x_coord, latitud.y = y_coord)
  return(predict(model, newdata = new_data, type = "class"))
}

# Devuelve las probabilidades de cada cala
predict_cala_prob <- function(x_coord, y_coord, model) {
  new_data <- data.frame(longitud.x = x_coord, latitud.y = y_coord)
  pred_probs <- predict(model, newdata = new_data, type = "probs")

  # Convertir a matriz si es necesario
  if (is.vector(pred_probs)) {
    pred_probs <- t(as.matrix(pred_probs))
  }

  colnames(pred_probs) <- c("Cala 0", "Cala 1", "Cala 2")
  return(pred_probs)
}
```

3.2.5. Modelado geométrico de la línea de costa

Para determinar los puntos de corte exactos que delimitan las zonas de influencia de calas a lo largo de la costa de forma precisa, se ha modelado la playa como un segmento rectilíneo que une el punto más al sur con el punto más al norte. Además, se crea una función *get_coords(t)* capaz de recorrer esta recta teórica, donde el parámetro t varía de 0 (Sur) a 1 (Norte).

```
None
# Modelar la playa como una línea recta de Sur a Norte
dx <- punto_norte$longitud.x - punto_sur$longitud.x
dy <- punto_norte$latitud.y - punto_sur$latitud.y
```

```
# Función para moverse por la playa: t=0 es Sur, t=1 es Norte
get_coords <- function(t) {
  x <- punto_sur$longitud.x + t * dx
  y <- punto_sur$latitud.y + t * dy
  return(data.frame(x=x, y=y))
}
```

3.2.6. Cálculo de las fronteras mediante optimización

Las fronteras entre zonas se definen como aquellos puntos de la línea de costa donde la incertidumbre del modelo es máxima, es decir, donde las probabilidades de pertenecer a dos calas adyacentes son idénticas. Matemáticamente, se buscan los puntos donde:

a. **Frontera Sur-Centro:** se utiliza la función *optimize()* para encontrar el valor de t que minimiza la diferencia absoluta: $|P(Cala_0) - P(Cala_1)|$.

b. **Frontera Centro-Norte:** se utiliza la función *optimize()* para encontrar el valor de t que minimiza la diferencia absoluta: $|P(Cala_1) - P(Cala_2)|$.

Este enfoque garantiza que los puntos de separación calculados (*punto_separacion_1* y *punto_separacion_2*) corresponden al límite natural de clasificación del modelo.

```
> print("Coordenadas del punto de separación Sur/Centro:")
[1] "Coordenadas del punto de separación Sur/Centro:"
> print(punto_separacion_1)
  longitud.x latitud.y
1   9.565435  1.086961
> print("Coordenadas del punto de separación Centro/Norte:")
[1] "Coordenadas del punto de separación Centro/Norte:"
> print(punto_separacion_2)
  longitud.x latitud.y
1   9.558737  1.099302
```

Figura 10. Coordenadas puntos de separación Sur/Centro y Centro/Norte

Estos puntos representan las **fronteras naturales de clasificación** según el modelo entrenado. Un fardo lanzado exactamente en estas coordenadas tendría aproximadamente un 50% de probabilidad de llegar a cada una de las dos calas adyacentes.

3.3. Evaluación del modelo mediante matriz de confusión

Para evaluar la capacidad predictiva del modelo, se ha generado la matriz de confusión utilizando los mismos datos con los que se entrenó:

```
None
predicciones_clase <- predict(modelo_multinom, newdata = cala.data, type = "class")
matriz_confusion <- table(Prediccion = predicciones_clase, Real = cala.data$cala)
```

```
> print("Matriz de Confusión (Pred vs Real):")
[1] "Matriz de Confusión (Pred vs Real):"
> print(matriz_confusion)
           Real
Prediccion sur centro norte
    sur    179     24     0
    centro  21    142    27
    norte    0     34   173
```

Figura 11. Resultado matriz de confusión

El modelo presenta una precisión del 82,33% en el conjunto de entrenamiento, lo que indica una alta capacidad para clasificar correctamente el destino de los fardos. Este valor se calcula sumando los aciertos totales en las predicciones, que corresponden a los valores de la diagonal, dividido entre el número total de observaciones (600). Los errores se concentran en las fronteras entre zonas, donde el modelo detecta correctamente que existe mayor incertidumbre, ya que como se puede observar los casos en los que se predice sur y es norte o se predice norte y es sur tienen como valor 0.

3.4. Predicciones para puntos específicos de prueba

Se ha evaluado el modelo con los tres puntos específicos solicitados en el enunciado:

```
[1] "--- Predicciones para Puntos de Prueba ---"
> for (i in 1:nrow(puntos_prueba)) {
+   x_p <- puntos_prueba[i, "x"]
+   y_p <- puntos_prueba[i, "y"]
+
+   cala_predicha <- predict_cala_clase(x_p, y_p, modelo_multinom)
+   probs_predichas <- predict_cala_prob(x_p, y_p, modelo_multinom)
+
+   cat(sprintf("\nCoordenadas: (%.6f, %.3f)\n", x_p, y_p))
+   cat(sprintf("  Cala Predicha: %s\n", cala_predicha))
+   cat("  Probabilidades:\n")
+   print(probs_predichas, digits = 4)
+ }

Coordenadas: (9.558359, 1.100)
  Cala Predicha: 2
  Probabilidades:
       Cala 0 Cala 1 Cala 2
[1,] 3.053e-05  0.399 0.6009

Coordenadas: (9.564329, 1.089)
  Cala Predicha: 1
  Probabilidades:
     Cala 0 Cala 1   Cala 2
[1,] 0.1848 0.8133 0.001927

Coordenadas: (9.568671, 1.081)
  Cala Predicha: 0
  Probabilidades:
     Cala 0  Cala 1    Cala 2
[1,]  0.987 0.01295 2.809e-07
```

Figura 12. Resultados predicciones para 3 puntos dados

Los tres puntos de prueba ilustran cómo varía el comportamiento del modelo a lo largo del eje Norte-Sur. El **primer punto (9.558359, 1.100)**, cuya latitud tiene casi el mismo valor que la del *punto_norte*, es clasificado como **Cala 2 (Norte)** con un 60.09% de probabilidad, situándose en una **zona de transición** con certeza moderada. El **segundo punto (9.564329, 1.089)** muestra una predicción más clara que la anterior, **Cala 1 (Centro)** con 81.33%, indicando que se encuentra dentro de la zona de influencia central y el inspector puede dirigirse con confianza a esta cala. El **tercer punto (9.568671, 1.081)**, al igual que el anterior, tiene un valor de latitud más cercano al del *punto_sur*, presenta la mayor certeza eligiendo **Cala 0 (Sur)** con 98.7%, estando claramente alejado de cualquier frontera.

Los resultados demuestran que el modelo identifica correctamente tanto zonas de alta certeza como zonas de transición. Cuanto más alejado está un punto de las fronteras calculadas, mayor es la confianza de la predicción.

3.5. Visualización gráfica de resultados

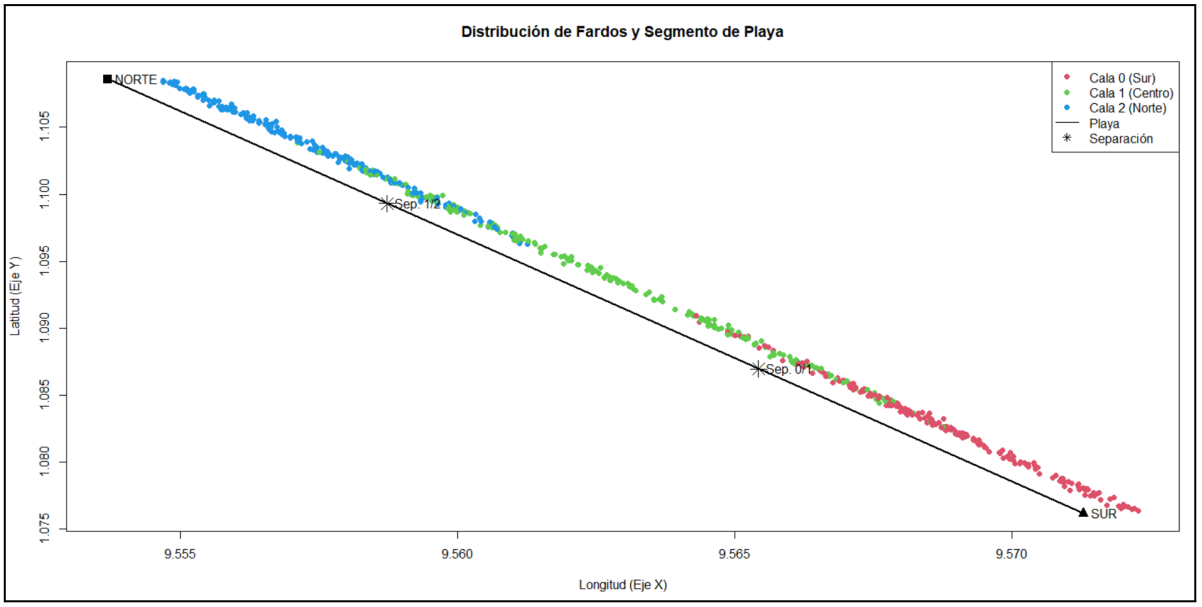

Figura 13. Distribución espacial de fardos y delimitación de zonas

En la distribución resultante se distingue con claridad la separación espacial entre las tres zonas geográficas. También se aprecian áreas de transición donde los colores se mezclan —por ejemplo, puntos rojos próximos a verdes, o verdes cercanos a azules—, lo que refleja una mayor incertidumbre en esas regiones. Los puntos que marcan la separación coinciden geométricamente con estas zonas de transición. Además, la representación confirma que la latitud es el factor que más influye en la disposición de los datos, dando lugar a una distribución mayoritariamente horizontal de cada grupo de color.

3.6. Localización geográfica

El enunciado plantea la siguiente pregunta: ¿En qué país es muy probable que vivan los protagonistas de esta historia? Para responderla, se utilizan las coordenadas GPS proporcionadas, que son aproximadamente (9.56° E, 1.09° N). Al analizar su posición en el mapa, estas coordenadas se sitúan en la zona continental de **Guinea Ecuatorial**, concretamente en **Abenelam**, muy próxima a la costa del Golfo de Guinea y al límite con Gabón. Por ello, el país más probable para que residan los protagonistas es Guinea Ecuatorial.

3.7. Análisis detallado de los resultados

- **Fortalezas del modelo**

El modelo presenta una alta precisión global, con un porcentaje de acierto muy elevado sobre los datos de entrenamiento. Esto permite clasificar el destino de los fardos de forma fiable. Además, identifica con claridad las fronteras entre zonas, gracias al algoritmo de optimización que determina los puntos exactos donde cambia la influencia de cada cala. También cuantifica la incertidumbre, proporcionando probabilidades que indican el nivel de confianza de cada predicción. Finalmente, el modelo es fácilmente interpretable, ya que solo utiliza las coordenadas de lanzamiento y genera resultados transparentes.

- **Limitaciones y consideraciones**

Existen zonas de transición donde las probabilidades entre dos calas son similares, lo que aumenta la incertidumbre y puede requerir decisiones complementarias por parte del inspector. También se asume que la costa es rectilínea, una simplificación que no capta completamente la complejidad real de las corrientes marinas. Además, el modelo se basa en datos de un experimento concreto, por lo que variaciones meteorológicas, estacionales o de mareas podrían afectar la precisión futura.

- **Recomendaciones operativas**

    - Probabilidades superiores al 75%: el inspector debería dirigirse directamente a la cala indicada, dado que el margen de acierto es muy alto.

    - Probabilidad se sitúa entre el 55% y el 75%, es recomendable priorizar la cala con mayor valor, pero manteniendo capacidad de reacción o incluso mandando a su subordinado a la otra cala destino.

    - Probabilidad menor del 55% o muy similares entre dos calas, el punto se encuentra en una zona de transición y conviene evaluar estrategias adicionales o esperar nuevos datos.

Cabe destacar que la latitud es el factor más determinante, ya que variaciones de apenas 0,01° pueden cambiar el destino del fardo, por lo que obtener coordenadas precisas es esencial.

El modelo de regresión logística multinomial se ha demostrado eficaz para apoyar la labor del inspector Augusto Puerrot. Permite clasificar los fardos con precisión, identificar fronteras entre zonas, ofrecer probabilidades útiles para la toma de decisiones y visualizar claramente las áreas de riesgo.

4. Problema 3. La Liga Nacional de Kárate

El objetivo de este estudio es determinar si el orden de salida de los participantes en la modalidad de Kata influye en sus posibilidades de clasificarse para la siguiente ronda. La hipótesis planteada por la empresa Talent on Web es que los competidores que salen en las primeras posiciones podrían estar siendo penalizados debido a un posible sesgo por parte de los jueces. Para comprobarlo, se analizarán los datos completos de las temporadas 2020 y 2021 de la LNK.

4.1. Manipulación y consolidación de datos

El primer desafío consiste en transformar los datos dispersos en múltiples hojas Excel en un único data frame estructurado de R que permita realizar análisis estadísticos robustos.

4.1.1. Configuración inicial y carga de librerías

El proceso comienza estableciendo el entorno de trabajo y cargando las librerías necesarias. En este caso, se utilizan *readxl*, que permite leer archivos Excel y convertirlos en data frames de R, y *dplyr*, que facilita la manipulación y transformación de datos mediante una sintaxis clara y eficiente.

```
setwd("D:/CUARTO/AMD/PRACTICA4/datasets")
# Instalar y cargar librerías (solo si no están)
#install.packages("readxl")
#install.packages("dplyr")
library(readxl)
library(dplyr)
```

Figura 14. Establecimiento de entorno y carga de librerías

4.1.2. Comprensión de la estructura de datos

Se observa que para cada uno de los archivos Excel existen múltiples hojas asociadas a la competición de ese año específico (2020 o 2021), pero que hacen referencia a diferentes categorías y jornadas.

Dentro de cada una de las hojas, se pueden visualizar los ejercicios individuales de cada competidor en las diferentes filas, mientras que las columnas hacen referencia a los 28 campos relevantes de cada uno de ellos, donde se incluyen sus identificadores, puntuaciones de jueces, resultados calculados, …

Al inspeccionar los archivos con mayor detenimiento, se observa que algunas hojas contienen columnas adicionales no documentadas, lo que genera errores al intentar asignar nombres de columna de forma automática. Por ello, se decidió implementar una estrategia de estandarización robusta.

4.1.3. Definición del esquema de columnas

Para garantizar la homogeneidad de los datos, se definió manualmente el vector *columnas_lectura* con los 28 campos que deben extraerse de cada hoja. Esta definición explícita permite controlar qué información se importa y facilita el diagnóstico de inconsistencias entre hojas.

```
None
columnas_karate <- c(
  "id_Competicion", "id_Competidor", "ronda", "grupo", "ordenSalida",
  paste0("puntosTecJuez", 1:7),
  paste0("puntosAtlJuez", 1:7),
```

```
  "puntosExtra",
  "totalPuntosTec", "totalPuntosAtl", "totalComputado", "totalDesempate",
  "totalDesempateLimite", "puesto", "estadoSigRonda", "columnaExtra"
)
```

4.1.4. Proceso interactivo de consolidación

Primero, se crea un data frame vacío llamado *datos_karate_final*, donde se almacenarán los datos de todas las hojas de los archivos Excel de 2020 y 2021.

A continuación, se define un vector con los nombres de los archivos Excel (*Karate2020.xlsx* y *Karate2021.xlsx*) y se recorre cada archivo mediante un bucle for. Dentro de cada archivo, se itera sobre todas sus hojas, ya que cada hoja contiene información de una jornada y categoría diferente.

Cada hoja se carga usando la función *read_xlsx*, saltando las primeras 4 filas y sin asignar nombres de columnas aún, permitiendo leer los datos tal como están y evitando errores derivados.

Se realiza luego una estandarización del número de columnas. Si la hoja tiene menos columnas de las esperadas (28), se añade una columna vacía (NA). En caso contrario, se descarta la hoja, asegurando así que todas las hojas sean compatibles al usar *rbind*.

Posteriormente, se asignan los nombres correctos a las columnas usando el vector *columnas_karate* y se extraen además metadatos importantes de cada hoja, como el año, la jornada y la modalidad, a partir del nombre de la hoja. Esta información se añade como columnas en el data frame, lo que permite realizar análisis por temporada, categoría o ronda de manera directa.

Finalmente, los datos procesados de cada hoja se agregan al data frame *datos_karate_final* mediante *rbind*. Al finalizar todos los bucles, este data frame contiene todas las jornadas, categorías y años, estandarizado, limpio y listo para el análisis estadístico y la modelización.

```
None
columnas_karate <- c(
  "id_Competicion", "id_Competidor", "ronda", "grupo", "ordenSalida",
  paste0("puntosTecJuez", 1:7),
  paste0("puntosAtlJuez", 1:7),
  "puntosExtra",
  "totalPuntosTec", "totalPuntosAtl", "totalComputado", "totalDesempate",
  "totalDesempateLimite", "puesto", "estadoSigRonda", "columnaExtra"
)
datos_karate <- data.frame()
ficheros <- c("Karate2020.xlsx", "Karate2021.xlsx")
# Silenciar los avisos de readxl durante la carga
suppressMessages({
  for (fichero in ficheros) {
    hojas <- excel_sheets(fichero)
    for (hoja in hojas) {
      datos_hoja <- as.data.frame(read_xlsx(fichero, hoja, skip = 4, col_names = FALSE))
```

```
      if (ncol(datos_hoja) < 28) {
        datos_hoja$columnaExtra <- NA
      } else if (ncol(datos_hoja) > 28) {
        next
      }

      names(datos_hoja) <- columnas_karate

      datos_hoja$anyo <- as.numeric(substr(hoja, 1, 4))
      datos_hoja$jornada <- as.numeric(substr(hoja, 6, 6))
      datos_hoja$modalidad <- as.factor(substr(hoja, 7, 7))

      datos_karate <- rbind(datos_karate, datos_hoja)
    }
  }
})
```

4.1.5. Conversión final de tipos de datos

Una vez consolidados todos los datos en *datos_karate_final*, es necesario crear la variable objetivo pasa que indica si un competidor clasifica a la siguiente ronda.

Dado que en la Ronda 1 la columna *estadoSigRonda* presenta valores NA para los competidores no clasificados (en lugar de un texto explícito como "no pasa"), se optó por una estrategia basada en el campo puesto. En el reglamento de la Liga Nacional de Kárate, se clasifican los 4 primeros puestos de cada grupo en la Ronda 1. Por tanto, se define la variable pasa de la siguiente manera:

- pasa = "SI" si puesto ≤ 4
- pasa = "NO" si puesto > 4

Esta definición garantiza una clasificación consistente y objetiva de la variable objetivo, evitando las inconsistencias derivadas de valores NA o cadenas vacías en *estadoSigRonda*.

Finalmente, se convierten las variables *pasa* y *modalidad* a factores con niveles explícitos (*c("SI", "NO")* y *c("F", "M")* respectivamente), garantizando una estructura correcta para los análisis estadísticos y modelos de clasificación posteriores.

Al finalizar este proceso, el data frame *datos_karate_final* contiene 3.870 observaciones correctamente estructuradas, con 32 variables listas para el análisis.

```
None
# Crear variable objetivo: ¿clasifica a la siguiente ronda?
datos_karate <- datos_karate %>%
  mutate(pasa = ifelse(puesto <= 4, "SI", "NO"))

datos_karate$pasa <- factor(datos_karate$pasa, levels = c("SI", "NO"))
datos_karate$modalidad <- factor(datos_karate$modalidad, levels = c("F", "M"))
```

4.2. Primera tarea: análisis estadístico

El enunciado plantea dos preguntas:

1. **¿La puntuación media de los primeros karatecas es igual a la puntuación media de los últimos?**
2. **¿La proporción de clasificados a la siguiente ronda entre los primeros y los últimos es igual?**

Para responder a estas preguntas, primero es necesario definir qué significa "salir entre los primeros" y "salir entre los últimos". Para ello se estableció un umbral de corte en k = 3.

- **Primeros**: competidores cuyo *ordenSalida* ≤ *3*.
- **Segundos**: competidores cuyo *ordenSalida* > *3*.

Dado que el orden de salida en rondas posteriores a la primera, está condicionado por el rendimiento en la ronda anterior, el análisis se restringe exclusivamente a la ronda 1, donde el orden es aleatorio o determinado únicamente por cabezas de serie.

4.2.1 Preprocesamiento y segmentación

Se filtraron las observaciones correspondientes a la Ronda 1 y se generó la variable categórica *grupo_orden*, segmentando la muestra bajo el criterios establecido *(ordenSalida ≤ 3 (Primeros) vs. ordenSalida > 3 (Últimos))*. De esta forma, se clasifica a los competidores en "Primeros" o "Últimos".

```
None
datos_ronda1 <- datos_karate %>%
  filter(ronda == 1, !is.na(totalComputado)) %>%
  mutate(grupo_orden = ifelse(ordenSalida <= 3, "Primeros", "Últimos"))

datos_ronda1$grupo_orden <- factor(datos_ronda1$grupo_orden,
                                   levels = c("Primeros", "Últimos"))
```

4.2.2 Pregunta 1: Comparación de medias (prueba t de Student)

Para determinar si existe diferencia significativa entre las puntuaciones de ambos grupos, se aplica una **prueba t de Student para muestras independientes.** Esta prueba es apropiada cuando la variable dependiente es continua (*totalComputado*), la variable independiente es categórica binaria (*grupo_orden*) y se asume distribución aproximadamente normal de los datos.

Hipótesis estadísticas:

- $\mathbf{H_0}$ (hipótesis nula): μ_primeros = μ_últimos (no hay diferencia en las medias).
- $\mathbf{H_1}$ (hipótesis alternativa): μ_primeros ≠ μ_últimos (existe diferencia significativa).

Nivel de significancia: $\alpha = 0.05$

Antes de realizar la prueba, se calcularon las estadísticas descriptivas por grupo:

```
None
estadisticas <- datos_ronda1 %>%
  group_by(grupo_orden) %>%
  summarise(
    n = n(),
    media = round(mean(totalComputado, na.rm = TRUE), 2),
    desv = round(sd(totalComputado, na.rm = TRUE), 2)
  )

print(estadisticas)
```

Se observa que el grupo de "Últimos" presenta una puntuación media superior (22.73) en comparación con los "Primeros" (21.75). Ambos grupos presentan desviaciones estándar similares, y sugiere variabilidad comparable en las puntuaciones.

	grupo_orden	n	media	desv_std	mediana	min	max
1	Primeros	679	21.74996	2.075808	22.02	0	27.12
2	Últimos	1284	22.72572	2.134929	22.92	0	28.14

Figura 15. Estadísticas descriptivas de competidores "Primeros" y "Últimos"

A continuación, se ejecuta la prueba t de Student:

```
None
resultado_t <- t.test(totalComputado ~ grupo_orden, data = datos_ronda1)
```

```
> print(resultado_t_test)

        Welch Two Sample t-test

data:  totalComputado by grupo_orden
t = -9.8088, df = 1414.7, p-value < 2.2e-16
alternative hypothesis: true difference in means between group Primeros and group Últimos is not equal to 0
95 percent confidence interval:
 -1.1709007 -0.7806207
sample estimates:
mean in group Primeros  mean in group Últimos
              21.74996               22.72572
```

Figura 16. Resultado de la prueba t Student

Se rechaza la hipótesis nula con un alto grado de significancia ($t(1414.7) = -9.81, p < 0.001$). Los datos confirman que salir entre los tres primeros perjudica la nota: estos competidores obtienen, de media, casi un punto menos (0.98) que los que salen después. El intervalo de confianza (entre 0.78 y 1.17) asegura que esto no es casualidad, sino que demuestra un sesgo claro de los jueces, que tienden a ser más exigentes al inicio de la competición.

4.2.3 Pregunta 2: Comparación de proporciones de clasificación

Para evaluar si el orden de salida influye en la probabilidad de clasificarse a la siguiente ronda, se comparan las proporciones de clasificación entre ambos grupos mediante una prueba de proporciones.

Hipótesis estadísticas:

- H_0: p_primeros = p_últimos (las proporciones de clasificación son iguales).
- H_1: p_primeros ≠ p_últimos (las proporciones de clasificación son diferentes).

Nivel de significancia: $\alpha = 0.05$

Primero, se construye la tabla de contingencia:

```
None
tabla_cont <- table(datos_ronda1$grupo_orden, datos_ronda1$pasa)
print(tabla_cont)
```

```
> print(tabla_contingencia)

           SI  NO
Primeros  271 408
Últimos   724 560
```

Figura 17. Proporciones de clasificados y no clasificados entre competidores "Primeros" y "Últimos"

Se observa una diferencia notable en las proporciones, mientras que el 39.9% (271 / 679) de los competidores que salen entre los primeros, logran clasificar; el 56.4% (724 / 1284) de los que salen entre los últimos lo consiguen. Esto representa una diferencia de 16.5 puntos porcentuales.

A continuación, se realiza la prueba de proporciones:

```
None
resultado_prop <- prop.test(tabla_cont)
```

```
> resultado_prop_test <- prop.test(tabla_contingencia)
> print(resultado_prop_test)

        2-sample test for equality of proportions with continuity
        correction

data:  tabla_contingencia
X-squared = 47.57, df = 1, p-value = 5.307e-12
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.2116167 -0.1178764
sample estimates:
   prop 1    prop 2
0.3991163 0.5638629
```

Figura 18. Resultado del test de proporciones con la diferencia de clasificación entre los dos grupos

El análisis es contundente ($p < 0.0001$) y lleva a **rechazar la hipótesis nula**: el orden de salida sí afecta a la clasificación. Los datos muestran que los competidores que salen al principio tienen muchas menos opciones de pasar de ronda. Concretamente, el intervalo de confianza dice que salir entre los tres primeros reduce la probabilidad de clasificar **entre un 11.8% y un 21.2%** respecto a los demás. Esto confirma que el sesgo de los jueces es grave: no solo bajan las puntuaciones, sino que este factor decide directamente quién sigue en la competición y quién no.

4.3. Segunda tarea: análisis avanzado mediante regresión logística

Para ir más allá de la simple comparación de grupos y medir con precisión cuánto influye cada posición en el orden de salida sobre el éxito del competidor, se aplican modelos de regresión logística. Esta técnica es muy útil cuando la variable de interés es binaria (Clasifica / No Clasifica).

4.3.1 Modelo 1: Modelo de regresión logística simple

Primero se genera un modelo simple para evaluar la relación directa entre el número de orden de salida (*ordenSalida*) y la probabilidad de clasificación (*pasa*), sin considerar otras variables.

```
None
datos_ronda1$pasa_binario <- ifelse(datos_ronda1$pasa == "SI", 1, 0)

modelo_simple <- glm(pasa_binario ~ ordenSalida,
                     data = datos_ronda1,
                     family = binomial)
```

```
> print(summary(modelo_simple))

Call:
glm(formula = pasa_binario ~ ordenSalida, family = binomial,
    data = datos_ronda1)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.68564    0.09693  -7.074 1.51e-12 ***
ordenSalida  0.14620    0.01760   8.307  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2720.9  on 1962  degrees of freedom
Residual deviance: 2648.7  on 1961  degrees of freedom
AIC: 2652.7

Number of Fisher Scoring iterations: 4
```

Figura 19. Resumen del modelo de regresión logística simple analizando el orden de salida

El análisis muestra un coeficiente de *ordenSalida* positivo: **β = 0.1462** ($p < 2\times10^{-16}$). Esto confirma que existe una relación directa: a mayor número de orden (salir más tarde), mayor probabilidad de éxito. Para interpretar este coeficiente en términos de probabilidad, se calcula el **odds ratios (OR):**

```
None
coef_orden <- coef(modelo_simple)["ordenSalida"]
odds_ratio <- exp(coef_orden)
cambio_pct <- (odds_ratio - 1) * 100
```

```
> cat("\n>>> INTERPRETACIÓN DEL COEFICIENTE:\n")

>>> INTERPRETACIÓN DEL COEFICIENTE:
> cat("    Coeficiente de ordenSalida:", round(coef_orden, 6), "\n")
    Coeficiente de ordenSalida: 0.1462
> cat("    Odds Ratio (exp(coef)):", round(odds_ratio, 4), "\n")
    Odds Ratio (exp(coef)): 1.1574
> cat("    Cambio porcentual:", round(porcentaje_cambio, 2), "%\n\n")
    Cambio porcentual: 15.74 %
```

Figura 20. Cálculo del Odds Ratio mostrando el incremento de probabilidad éxito vs. posición salida

Como el *coef_orden* muestra que es mayor que 0, por cada posición que se retrasa la salida, la ventaja (odds) de clasificar aumenta un 15.74%. Por tanto, los datos demuestran que el beneficio de salir tarde es acumulativo para condiciones iguales.

4.3.2 Modelo 2: Modelo con interacción, efecto diferencial por modalidad

Una duda crítica es si este sesgo favorece o perjudica de forma distinta según el género del competidor (modalidad: Femenina, Masculina). Para averiguarlo, se añade un término de interacción al modelo (*pasa ~ ordenSalida * modalidad)*. Este modelo permite que el efecto de *ordenSalida* sea diferente para cada modalidad.

```
None
modelo_interaccion <- glm(pasa_binario ~ ordenSalida * modalidad,
                          data = datos_ronda1,
                          family = binomial)

coef_int <- summary(modelo_interaccion)$coefficients["ordenSalida:modalidadM", ]
```

```
> print(summary(modelo_interaccion))

Call:
glm(formula = pasa_binario ~ ordenSalida * modalidad, family = binomial,
    data = datos_ronda1)

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)           -0.60867    0.13396  -4.544 5.53e-06 ***
ordenSalida            0.13104    0.02428   5.396 6.80e-08 ***
modalidadM            -0.16076    0.19412  -0.828    0.408
ordenSalida:modalidadM  0.03163    0.03525   0.897    0.369
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2720.9  on 1962  degrees of freedom
Residual deviance: 2647.9  on 1959  degrees of freedom
AIC: 2655.9

Number of Fisher Scoring iterations: 4
```

Figura 21. Modelo de regresión con interacción que verifica si el sesgo del orden varía según género

El p-valor obtenido es mucho mayor que el umbral de significancia de 0.05, por tanto no existe evidencia estadística de que el efecto del orden varíe entre modalidades, es decir, el sesgo afecta por igual a la categoría femenina y a la masculina.

4.4. Análisis detallado de los resultados

El análisis de las temporadas 2020 y 2021 confirma sin lugar a dudas la hipótesis de Talent on Web: existe un sesgo sistemático en el arbitraje. Tras consolidar los datos de casi 2.000 competidores en la primera ronda, se obtiene la conclusión clara de que quienes salen al final obtienen, de media, un punto más que los primeros (22.73 frente a 21.75). Esto no es solo una cuestión de nota, sino que decide quién sigue compitiendo, ya que los que salen tarde se clasifican el 56% de las veces, mientras que los que abren la pista solo lo logran el 40%. De hecho, el modelo matemático dice que por cada puesto que retrasas tu salida, tus opciones de pasar mejoran un 16% acumulativo.

Es importante destacar que este problema afecta exactamente igual a la categoría masculina y femenina, por lo que no es cuestión de género. Todo apunta a que los jueces tienden a ser conservadores al principio y van puntuando con notas más altas a medida que tienen más referencias.

5. Conclusiones

La realización de esta práctica ha supuesto un reto que ha permitido consolidar los conocimientos adquiridos durante el curso al enfrentarse a problemas reales de análisis de datos.

Uno de los mayores aprendizajes ha sido la comprobación de la ejecución de los algoritmos de selección automática (*stepwise*). Al inicio del Problema 1, la comparación manual de decenas de modelos se presentaba como una tarea inabarcable, y la implementación del algoritmo, así como la observación de cómo filtraba las variables irrelevantes, ha puesto de manifiesto la importancia de optimizar los flujos de trabajo en minería de datos sin perder rigor estadístico.

En el Problema 3 se encontraron datos "sucios" y dispersos, lo que resultó especialmente desafiante y obligó a investigar y emplear librerías como *readxl* y *dplyr*. Esta fase de preprocesamiento fue la más tediosa y puso de manifiesto que el modelado estadístico carece de utilidad si no se dispone de datos limpios que permitan obtener resultados fiables.

Durante el desarrollo de la práctica ha resultado especialmente interesante observar cómo herramientas como la regresión logística o los tests de hipótesis pueden emplearse para destapar problemas sociales o estructurales, como los identificados en los distintos problemas analizados.

6. Tabla de esfuerzos invertidos

A continuación, se detallan las tareas realizadas para el desarrollo de la práctica. De acuerdo con la metodología de trabajo establecida, todas las actividades se llevaron a cabo de forma conjunta, mediante programación en pareja y revisión simultánea de los resultados.

Apartados del trabajo realizados	Lucía	Irene
Investigación de nuevas librerías y configuraciones	2	2
Problema 1: implementación del algoritmo *stepwise*, análisis de modelos y justificación de variables significativas	3	3
Problema 2: desarrollo del modelo multinomial y del algoritmo de optimización matemática	5	5
Problema 3: desarrollo del script de carga iterativa para unificar los excels, limpieza, estructuración del dataset, realización de los tests estadísticos, modelos logísticos e interpretación de los coeficientes	7	7
Preguntas	2	2
Informe	3	3
	22	**22**

7. Webgrafía

Selección automática de variables (*step*)

- Algoritmos para selección de variables en lm (Bookdown)
- Stepwise variable selection in R with step() function (YouTube)
- Documenta el uso de la función nativa *step()* para automatizar la selección de predictores basada en el criterio AIC (Problema 1).

Optimización matemática (*optimize*)

- Optimización Matemática con R - Vol I (CRAN)
- Documenta el uso de *optimize()* para hallar mínimos exactos de funciones que es necesario para encontrar las fronteras entre calas (Problema 2).

Manipulación de datos moderna (*dplyr*)

- Manipulación de datos ordenados usando dplyr (Eliocamp)
- Explica la sintaxis "Tidy Verse" (*filter, mutate, %>%*) utilizada para limpiar y transformar los datos de forma más legible que con R base (Problema 3).

Lectura de archivos Excel (*readxl*)

- Leer Excel en R con readxl (R Coder)
- Describe el paquete estándar para importar archivos *.xlsx* con múltiples hojas, ya que *read.csv* (visto en clase) no soporta este formato (Problema 3).

Pruebas estadísticas clásicas (*t.test* y *prop.test*)

- Tutorial de pruebas T en R (DataCamp)
- Detalla la función específica *t.test()* para comparar medias entre dos grupos independientes (Problema 3 - Pregunta 1).
- La función prop.test en R (R Coder)
- Explica la función *prop.test()* para la comparación estadística de proporciones de éxito (Problema 3 - Pregunta 2).

Interacción entre variables en fórmulas (*)

- Fórmulas en R (PDF U-Cursos)
- Define formalmente que el operador * en una fórmula de R incluye tanto los efectos principales como el término de interacción (Problema 3 - Modelo 2).

8. Anexos

8.1 Script Problema1

None

```
# ============================================================================
# PROBLEMA 1: CALIDAD DE LA CERVEZA - Selección automática de predictores
# ============================================================================
# Configurar directorio y cargar datos
setwd("C:/Users/irene/OneDrive/Escritorio/datasetsPractica4/datasets")
beer.data <- read.csv("lagerdata.csv", sep=";")

# Definir el modelo más complejo posible (con los 3 predictores)
formula_completa <- as.formula("~ tpc + ma + tso2")

# ----------------------------------------------------------------------------
# PASO 1: Entrenar modelos para cada ensayo
# ----------------------------------------------------------------------------
# El algoritmo step() prueba todas las combinaciones y se queda con la mejor
# según el criterio AIC (penaliza modelos innecesariamente complejos)

# DSA
modelo_dsa_completo <- lm(dsa ~ tpc + ma + tso2, data=beer.data)
modelo_dsa_optimo <- step(modelo_dsa_completo,
                          scope = list(lower = ~1, upper = formula_completa),
                          direction = "both", trace = 0)

# ASA
modelo_asa_completo <- lm(asa ~ tpc + ma + tso2, data=beer.data)
modelo_asa_optimo <- step(modelo_asa_completo,
                          scope = list(lower = ~1, upper = formula_completa),
                          direction = "both", trace = 0)

# ORAC
modelo_orac_completo <- lm(orac ~ tpc + ma + tso2, data=beer.data)
modelo_orac_optimo <- step(modelo_orac_completo,
                           scope = list(lower = ~1, upper = formula_completa),
                           direction = "both", trace = 0)

# RP
modelo_rp_completo <- lm(rp ~ tpc + ma + tso2, data=beer.data)
modelo_rp_optimo <- step(modelo_rp_completo,
                         scope = list(lower = ~1, upper = formula_completa),
                         direction = "both", trace = 0)

# MCA
modelo_mca_completo <- lm(mca ~ tpc + ma + tso2, data=beer.data)
modelo_mca_optimo <- step(modelo_mca_completo,
                          scope = list(lower = ~1, upper = formula_completa),
                          direction = "both", trace = 0)

# ----------------------------------------------------------------------------
# PASO 2: Crear tabla resumen con los resultados
# ----------------------------------------------------------------------------
# Función auxiliar: extrae qué predictores quedaron en el modelo final
extraer_predictores <- function(modelo) {
  coefs <- names(coef(modelo))
```

```
  predictores <- coefs[coefs != "(Intercept)"]
  if (length(predictores) == 0) return("Ninguno")
  return(paste(predictores, collapse = " + "))
}

# Construir la tabla comparativa
resultados <- data.frame(
  Ensayo = c("dsa", "asa", "orac", "rp", "mca"),
  Predictores = c(
    extraer_predictores(modelo_dsa_optimo),
    extraer_predictores(modelo_asa_optimo),
    extraer_predictores(modelo_orac_optimo),
    extraer_predictores(modelo_rp_optimo),
    extraer_predictores(modelo_mca_optimo)
  ),
  R2_Ajustado = round(c(
    summary(modelo_dsa_optimo)$adj.r.squared,
    summary(modelo_asa_optimo)$adj.r.squared,
    summary(modelo_orac_optimo)$adj.r.squared,
    summary(modelo_rp_optimo)$adj.r.squared,
    summary(modelo_mca_optimo)$adj.r.squared
  ), 4)
)

print(resultados)

# ------------------------------------------------------------------------
# PASO 3: Ver los detalles estadísticos de cada modelo
# ------------------------------------------------------------------------

cat("\n=== MODELO DSA ===\n")
summary(modelo_dsa_optimo)

cat("\n=== MODELO ASA ===\n")
summary(modelo_asa_optimo)

cat("\n=== MODELO ORAC ===\n")
summary(modelo_orac_optimo)

cat("\n=== MODELO RP ===\n")
summary(modelo_rp_optimo)

cat("\n=== MODELO MCA ===\n")
summary(modelo_mca_optimo)
```

8.2 Script Problema2

None

```
# ============================================================================
# PROBLEMA 2: CLASIFICACIÓN DE FARDOS - Regresión Logística Multinomial
# ============================================================================

setwd("C:/Users/irene/OneDrive/Escritorio/datasetsPractica4/datasets")

# install.packages("nnet")  # Descomentar si no lo tienes instalado
library(nnet)

# Cargar datos y convertir la variable respuesta a factor
cala.data <- read.csv("datacala.csv")
cala.data$cala <- as.factor(cala.data$cala)

# ----------------------------------------------------------------------------
# PASO 1: Identificar los extremos de la playa
# ----------------------------------------------------------------------------
idx_sur <- which.min(cala.data$latitud.y)
punto_sur <- cala.data[idx_sur, c("longitud.x", "latitud.y")]

idx_norte <- which.max(cala.data$latitud.y)
punto_norte <- cala.data[idx_norte, c("longitud.x", "latitud.y")]

cat("\n=== PUNTOS EXTREMOS ===\n")
cat("Sur: ", punto_sur$longitud.x, ",", punto_sur$latitud.y, "\n")
cat("Norte:", punto_norte$longitud.x, ",", punto_norte$latitud.y, "\n\n")

# ----------------------------------------------------------------------------
# PASO 2: Entrenar el modelo
# ----------------------------------------------------------------------------
modelo_multinom <- multinom(cala ~ longitud.x + latitud.y, data = cala.data)

# ----------------------------------------------------------------------------
# PASO 3: Funciones auxiliares para predecir
# ----------------------------------------------------------------------------
# Devuelve la cala predicha (0, 1 o 2)
predict_cala_clase <- function(x_coord, y_coord, model) {
  new_data <- data.frame(longitud.x = x_coord, latitud.y = y_coord)
  return(predict(model, newdata = new_data, type = "class"))
}

# Devuelve las probabilidades de cada cala
predict_cala_prob <- function(x_coord, y_coord, model) {
  new_data <- data.frame(longitud.x = x_coord, latitud.y = y_coord)
  pred_probs <- predict(model, newdata = new_data, type = "probs")

  # Convertir a matriz si es necesario
  if (is.vector(pred_probs)) {
    pred_probs <- t(as.matrix(pred_probs))
  }

  colnames(pred_probs) <- c("Cala 0", "Cala 1", "Cala 2")
  return(pred_probs)
}
```

```
# ------------------------------------------------------------------------
# PASO 4: Calcular los puntos de separación entre calas
# ------------------------------------------------------------------------
# Modelar la playa como una línea recta de Sur a Norte
dx <- punto_norte$longitud.x - punto_sur$longitud.x
dy <- punto_norte$latitud.y - punto_sur$latitud.y

# Función para moverse por la playa: t=0 es Sur, t=1 es Norte
get_coords <- function(t) {
  x <- punto_sur$longitud.x + t * dx
  y <- punto_sur$latitud.y + t * dy
  return(data.frame(x=x, y=y))
}

# Buscar donde las probabilidades de Cala 0 y Cala 1 se cruzan
diferencia_01 <- function(t) {
  if (t < 0 | t > 1) return(NA)
  coords <- get_coords(t)
  probs <- predict_cala_prob(coords$x, coords$y, modelo_multinom)
  return(abs(probs[1, 1] - probs[1, 2]))
}

# Buscar donde las probabilidades de Cala 1 y Cala 2 se cruzan
diferencia_12 <- function(t) {
  if (t < 0 | t > 1) return(NA)
  coords <- get_coords(t)
  probs <- predict_cala_prob(coords$x, coords$y, modelo_multinom)
  return(abs(probs[1, 2] - probs[1, 3]))
}

# Optimizar para encontrar los puntos exactos
t1_opt <- optimize(diferencia_01, interval = c(0, 0.5))$minimum
punto_separacion_1 <- get_coords(t1_opt)
colnames(punto_separacion_1) <- c("longitud.x", "latitud.y")

t2_opt <- optimize(diferencia_12, interval = c(0.5, 1))$minimum
punto_separacion_2 <- get_coords(t2_opt)
colnames(punto_separacion_2) <- c("longitud.x", "latitud.y")

cat("\n=== FRONTERAS ENTRE CALAS ===\n")
cat("Sur/Centro:", punto_separacion_1$longitud.x, ",", punto_separacion_1$latitud.y,
"\n")
cat("Centro/Norte:", punto_separacion_2$longitud.x, ",", punto_separacion_2$latitud.y,
"\n\n")

# ------------------------------------------------------------------------
# PASO 5: Matriz de confusión
# ------------------------------------------------------------------------
predicciones_clase <- predict(modelo_multinom, newdata = cala.data, type = "class")
matriz_confusion <- table(Prediccion = predicciones_clase, Real = cala.data$cala)

colnames(matriz_confusion) <- c("sur", "centro", "norte")
rownames(matriz_confusion) <- c("sur", "centro", "norte")

cat("\n=== MATRIZ DE CONFUSIÓN ===\n")
print(matriz_confusion)
```

```
# Calcular precisión
precision <- sum(diag(matriz_confusion)) / sum(matriz_confusion)
cat("\nPrecisión global:", round(precision * 100, 2), "%\n\n")

# ------------------------------------------------------------------------
# PASO 6: Predicciones para puntos específicos
# ------------------------------------------------------------------------
puntos_prueba <- data.frame(
  x = c(9.558359, 9.564329, 9.568671),
  y = c(1.100, 1.089, 1.081)
)

cat("\n=== PREDICCIONES PARA PUNTOS DE PRUEBA ===\n")
for (i in 1:nrow(puntos_prueba)) {
  x_p <- puntos_prueba[i, "x"]
  y_p <- puntos_prueba[i, "y"]

  cala_predicha <- predict_cala_clase(x_p, y_p, modelo_multinom)
  probs_predichas <- predict_cala_prob(x_p, y_p, modelo_multinom)

  cat(sprintf("\nPunto %d: (%.6f, %.3f)\n", i, x_p, y_p))
  cat("  Destino predicho: Cala", cala_predicha, "\n")
  cat("  Probabilidades:", sprintf("%.1f%%", probs_predichas * 100), "\n")
}

# ------------------------------------------------------------------------
# PASO 7: Visualización
# ------------------------------------------------------------------------
plot(cala.data$longitud.x, cala.data$latitud.y,
     col = as.numeric(cala.data$cala) + 1,
     pch = 19,
     xlab = "Longitud", ylab = "Latitud",
     main = "Distribución de Fardos por Cala")

# Línea de costa
segments(punto_sur$longitud.x, punto_sur$latitud.y,
         punto_norte$longitud.x, punto_norte$latitud.y,
         col = "black", lwd = 2)

# Puntos extremos
points(punto_sur$longitud.x, punto_sur$latitud.y, col = "black", pch = 17, cex = 1.5)
text(punto_sur$longitud.x, punto_sur$latitud.y, "SUR", pos = 4)

points(punto_norte$longitud.x, punto_norte$latitud.y, col = "black", pch = 15, cex =
1.5)
text(punto_norte$longitud.x, punto_norte$latitud.y, "NORTE", pos = 4)

# Puntos de separación
points(punto_separacion_1$longitud.x, punto_separacion_1$latitud.y, col = "black", pch
= 8, cex = 2)
text(punto_separacion_1$longitud.x, punto_separacion_1$latitud.y, "Sur/Centro", pos =
4)

points(punto_separacion_2$longitud.x, punto_separacion_2$latitud.y, col = "black", pch
= 8, cex = 2)
text(punto_separacion_2$longitud.x, punto_separacion_2$latitud.y, "Centro/Norte", pos
= 4)
```

```
legend("topright",
       legend = c("Cala 0 (Sur)", "Cala 1 (Centro)", "Cala 2 (Norte)", "Costa",
"Frontera"),
       col = c(2, 3, 4, "black", "black"),
       pch = c(19, 19, 19, NA, 8),
       lty = c(NA, NA, NA, 1, NA))
```

8.3 Script Problema3

None

```
# ============================================================================
# PROBLEMA 3: LIGA NACIONAL DE KÁRATE - Análisis de sesgo por orden de salida
# ============================================================================

setwd("C:/Users/irene/OneDrive/Escritorio/datasetsPractica4/datasets")

library(readxl)
library(dplyr)

# ----------------------------------------------------------------------------
# PASO 1: Cargar y consolidar datos de 2020 y 2021
# ----------------------------------------------------------------------------
columnas_karate <- c(
  "id_Competicion", "id_Competidor", "ronda", "grupo", "ordenSalida",
  paste0("puntosTecJuez", 1:7),
  paste0("puntosAtlJuez", 1:7),
  "puntosExtra",
  "totalPuntosTec", "totalPuntosAtl", "totalComputado", "totalDesempate",
  "totalDesempateLimite", "puesto", "estadoSigRonda", "columnaExtra"
)

datos_karate <- data.frame()
ficheros <- c("Karate2020.xlsx", "Karate2021.xlsx")

# Silenciar los avisos de readxl durante la carga
suppressMessages({
  for (fichero in ficheros) {
    hojas <- excel_sheets(fichero)

    for (hoja in hojas) {
      datos_hoja <- as.data.frame(read_xlsx(fichero, hoja, skip = 4, col_names = FALSE))

      if (ncol(datos_hoja) < 28) {
        datos_hoja$columnaExtra <- NA
      } else if (ncol(datos_hoja) > 28) {
        next
      }

      names(datos_hoja) <- columnas_karate

      datos_hoja$anyo <- as.numeric(substr(hoja, 1, 4))
      datos_hoja$jornada <- as.numeric(substr(hoja, 6, 6))
      datos_hoja$modalidad <- as.factor(substr(hoja, 7, 7))

      datos_karate <- rbind(datos_karate, datos_hoja)
    }
  }
})

# Crear variable objetivo: ¿clasifica a la siguiente ronda?
datos_karate <- datos_karate %>%
  mutate(pasa = ifelse(puesto <= 4, "SI", "NO"))

datos_karate$pasa <- factor(datos_karate$pasa, levels = c("SI", "NO"))
datos_karate$modalidad <- factor(datos_karate$modalidad, levels = c("F", "M"))

cat("\n=== DATOS CARGADOS ===\n")
```

```
cat("Total de observaciones:", nrow(datos_karate), "\n")
cat("Años:", unique(datos_karate$anyo), "\n\n")

# ------------------------------------------------------------------------
# PASO 2: Filtrar y preparar datos de la Ronda 1
# ------------------------------------------------------------------------
datos_ronda1 <- datos_karate %>%
  filter(ronda == 1, !is.na(totalComputado)) %>%
  mutate(grupo_orden = ifelse(ordenSalida <= 3, "Primeros", "Últimos"))

datos_ronda1$grupo_orden <- factor(datos_ronda1$grupo_orden,
                                   levels = c("Primeros", "Últimos"))

cat("=== DISTRIBUCIÓN EN RONDA 1 ===\n")
print(table(datos_ronda1$grupo_orden))

# ------------------------------------------------------------------------
# PASO 3: Pregunta 1 - ¿Difieren las puntuaciones medias?
# ------------------------------------------------------------------------
cat("\n\n=== PREGUNTA 1: PUNTUACIONES MEDIAS ===\n\n")

estadisticas <- datos_ronda1 %>%
  group_by(grupo_orden) %>%
  summarise(
    n = n(),
    media = round(mean(totalComputado, na.rm = TRUE), 2),
    desv = round(sd(totalComputado, na.rm = TRUE), 2)
  )

print(estadisticas)

# Prueba t de Student
resultado_t <- t.test(totalComputado ~ grupo_orden, data = datos_ronda1)

cat("\nPrueba t de Student:\n")
cat("  Diferencia de medias:", round(diff(resultado_t$estimate), 2), "\n")
cat("  t =", round(resultado_t$statistic, 2), "\n")
cat("  p-valor =", format(resultado_t$p.value, scientific = TRUE, digits = 3), "\n")

if (resultado_t$p.value < 0.05) {
  cat("SÍ existe diferencia significativa (p < 0.05)\n")
} else {
  cat("NO existe diferencia significativa (p >= 0.05)\n")
}

# ------------------------------------------------------------------------
# PASO 4: Pregunta 2 - ¿Difieren las proporciones de clasificación?
# ------------------------------------------------------------------------
cat("\n\n=== PREGUNTA 2: PROPORCIONES DE CLASIFICACIÓN ===\n\n")

tabla_cont <- table(datos_ronda1$grupo_orden, datos_ronda1$pasa)
print(tabla_cont)

cat("\nProporciones:\n")
print(round(prop.table(tabla_cont, margin = 1), 3))

# Prueba de proporciones
resultado_prop <- prop.test(tabla_cont)
```

```
cat("\nPrueba de proporciones:\n")
cat(" X² =", round(resultado_prop$statistic, 2), "\n")
cat(" p-valor =", format(resultado_prop$p.value, scientific = TRUE, digits = 3), "\n")

if (resultado_prop$p.value < 0.05) {
  cat("  ✓ SÍ existe diferencia significativa (p < 0.05)\n")
} else {
  cat("  ✗ NO existe diferencia significativa (p >= 0.05)\n")
}

# ------------------------------------------------------------------------
# PASO 5: Regresión logística simple
# ------------------------------------------------------------------------
cat("\n\n=== MODELO 1: REGRESIÓN LOGÍSTICA SIMPLE ===\n")
cat("Fórmula: pasa ~ ordenSalida\n\n")

datos_ronda1$pasa_binario <- ifelse(datos_ronda1$pasa == "SI", 1, 0)

modelo_simple <- glm(pasa_binario ~ ordenSalida,
                     data = datos_ronda1,
                     family = binomial)

coef_orden <- coef(modelo_simple)["ordenSalida"]
odds_ratio <- exp(coef_orden)
cambio_pct <- (odds_ratio - 1) * 100

cat("Coeficiente ordenSalida:", round(coef_orden, 4), "\n")
cat("Odds Ratio:", round(odds_ratio, 4), "\n")
cat("Cambio porcentual:", round(cambio_pct, 2), "%\n")
cat("p-valor:", format(summary(modelo_simple)$coefficients["ordenSalida", "Pr(>|z|)"],
                       scientific = TRUE, digits = 3), "\n")

cat("\nInterpretación: Por cada posición que se retrasa la salida,\n")
cat("las probabilidades de clasificar aumentan un", round(cambio_pct, 1), "%\n")

# ------------------------------------------------------------------------
# PASO 6: Regresión logística con interacción (género)
# ------------------------------------------------------------------------

cat("\n\n=== MODELO 2: CON INTERACCIÓN (GÉNERO) ===\n")
cat("Fórmula: pasa ~ ordenSalida * modalidad\n\n")

modelo_interaccion <- glm(pasa_binario ~ ordenSalida * modalidad,
                          data = datos_ronda1,
                          family = binomial)

coef_int <- summary(modelo_interaccion)$coefficients["ordenSalida:modalidadM", ]

cat("Término de interacción (ordenSalida:modalidadM):\n")
cat("  Coeficiente:", round(coef_int[1], 4), "\n")
cat("  p-valor:", round(coef_int[4], 4), "\n")

if (coef_int[4] < 0.05) {
  cat("El efecto del orden SÍ varía según género (p < 0.05)\n")
  cat("Se requieren intervenciones diferenciadas\n")
} else {
  cat("El efecto del orden NO varía según género (p >= 0.05)\n")
  cat("El sesgo afecta igual a ambas categorías\n")
}
```

```
# ------------------------------------------------------------------------
# PASO 7: Resumen final
# ------------------------------------------------------------------------
cat("\n\n=== RESUMEN FINAL ===\n\n")

resumen <- data.frame(
  Pregunta = c(
    "¿Difieren las puntuaciones medias?",
    "¿Difieren las proporciones de clasificación?",
    "¿Varía el efecto según género?"
  ),
  P_valor = c(
    format(resultado_t$p.value, scientific = TRUE, digits = 3),
    format(resultado_prop$p.value, scientific = TRUE, digits = 3),
    round(coef_int[4], 4)
  ),
  Significativo = c(
    ifelse(resultado_t$p.value < 0.05, "SÍ", "NO"),
    ifelse(resultado_prop$p.value < 0.05, "SÍ", "NO"),
    ifelse(coef_int[4] < 0.05, "SÍ", "NO")
  )
)

print(resumen)

# Guardar resultados
write.csv(resumen, "resumen_karate.csv", row.names = FALSE)
write.csv(estadisticas, "estadisticas_karate.csv", row.names = FALSE)
```