# Modelo Lineal y Logístico (Código):

# Accidentes viales EE. UU.

Nombre: Maricela Flores Manqui.

Profesor Cátedra: Ricardo Crespo Vergara.

Profesor Laboratorio: Claudio Álvarez Soto.

Ayudante: Jimmy Fierro Herrera.

Curso: Econometría espacial.

Fecha: 14/11/2023.

##INTALACION LIBRERIAS - CARGAR DATOS----------

```{r}

# Cargar la librerias

install.packages("ggplot2")

install.packages("lubridate")

install.packages("dplyr")

install.packages("vcd")

install.packages("reshape")

install.packages("leaflet")

install.packages("caTools")

install.packages("ROCR")

install.packages("caret")

install.packages("pROC")

library(pROC)

library(ROCR)

library(caret)

library(caTools)

library(dplyr)

library(vcd)

library(ggplot2)

library(reshape)

library(leaflet)

library(lubridate)
```

```
##Cargar datos

datos = read.csv("C:/ECONOMETRIA/US_Accidents_March23.csv")

##filtro por estado

filtro1 = datos[datos$State == "OR",]

##filtro para columbia

#filtro2 = filtro1[filtro1$County == "Columbia",]

filtro2 = filtro1[filtro1$County %in% c("Columbia", "Clatsop", "Washington"), ]


##seleccion de variables a usar

data <- data.frame(ID = filtro2$ID,

                   severity = filtro2$Severity,

                   latitud = filtro2$Start_Lat,

                   longitud = filtro2$Start_Lng,

                   precipitacion = filtro2$Precipitation.in.,

                   distancia = filtro2$Distance.mi.,

                   hora_inicio = filtro2$Start_Time,

                   condado = filtro2$County,

                   visibilidad = filtro2$Visibility.mi.,

                   velocidad_viento = filtro2$Wind_Speed.mph.,

                   cond_meteo = filtro2$Weather_Condition,

                   crossing = filtro2$Crossing,

                   give_way = filtro2$Give_Way,

                   stop = filtro2$Stop)
```

##SUMARY-DUPLICADOS-NA----------

```{r}

##SUMARY

summary(data)

##Analisis de Duplicados

dupli = duplicated(data)

suma_dupli = sum(dupli)

suma_dupli

##NA

##NA de precipitacion se cambia por la media de la catgoria de condicion meteorologica

# Crear una tabla de resumen de la cantidad de NA en 'precipitación' por categoría de 'cond_meteo'

tabla_resumen <- data %>%

  group_by(cond_meteo) %>%

  summarise(NA_count = sum(is.na(precipitacion)))

# Mostrar la tabla de resumen

print(tabla_resumen)

##FUNCION PARA CAMBIAR NA POR LA MEDIA POR CATEGORIA DE CONDICION METEOROLOGICA

imputar_precipitacion <- function(datos, categoria) {

  # Calcula la media de 'precipitacion' para la categoría especificada

  media_categoria <- mean(datos$precipitacion[datos$cond_meteo == categoria], na.rm = TRUE)

  # Reemplaza los NA en 'precipitacion' con la media de la categoría
```

```
  datos$precipitacion <- ifelse(is.na(datos$precipitacion) & datos$cond_meteo == categoria, media_categoria, datos$precipitacion)

  return(datos)

}


# Uso de la función para imputar 'precipitacion' para las diferentes categorías

data <- imputar_precipitacion(data, "Light Drizzle")

data <- imputar_precipitacion(data, "Drizzle")

data <- imputar_precipitacion(data, "Clear")

data <- imputar_precipitacion(data, "Fog")

data <- imputar_precipitacion(data, "Haze")

data <- imputar_precipitacion(data, "Light Freezing Fog")

data <- imputar_precipitacion(data, "Light Freezing Rain")

data <- imputar_precipitacion(data, "Heavy Rain")

data <- imputar_precipitacion(data, "Light Rain")

data <- imputar_precipitacion(data, "Light Snow")

data <- imputar_precipitacion(data, "Mist")

data <- imputar_precipitacion(data, "Mostly Cloudy")

data <- imputar_precipitacion(data, "Overcast")

data <- imputar_precipitacion(data, "Partly Cloudy")

data <- imputar_precipitacion(data, "Patches of Fog")

data <- imputar_precipitacion(data, "Rain")

data <- imputar_precipitacion(data, "Scattered Clouds")

data <- imputar_precipitacion(data, "Shallow Fog")

data <- imputar_precipitacion(data, "Smoke")
```

```
data <- imputar_precipitacion(data, "Snow")

data <- imputar_precipitacion(data, "Thunderstorm")


##Revision NA

tabla_resumen <- data %>%

  group_by(cond_meteo) %>%

  summarise(NA_count = sum(is.na(precipitacion)))

# Mostrar la tabla de resumen

print(tabla_resumen)


####

# Calcular la moda de la variable cond_meteo

# Calcular la tabla de frecuencias

tabla_frecuencias <- table(data$cond_meteo)

# Encontrar la moda

moda <- names(tabla_frecuencias[tabla_frecuencias == max(tabla_frecuencias)])

moda

# Reemplazar NA por la moda

data$cond_meteo[data$cond_meteo == ""] <- moda

data <- imputar_precipitacion(data, moda)

####

#Obtener hora del dia del accidente

data$hour_of_day <- hour(data$hora_inicio)

categoricas <- data[,c("ID","severity","cond_meteo","crossing","give_way","stop","hour_of_day")]
```

```
##SEPARACION VAR CONTINUAS

continuas <- data[,c("ID","severity","latitud","longitud","precipitacion","distancia","visibilidad","velocidad_viento")]

##NA POR MEDIA

# Calcular la media de las columnas con NA y reemplazar los NA con la media

for (col in colnames(continuas)) {

  if (any(is.na(continuas[, col]))) {

    col_mean <- mean(continuas[, col], na.rm = TRUE)

    continuas[is.na(continuas[, col]), col] <- col_mean

  }

}
```

###OTL

```{r}
# Función para reemplazar outliers por estadísticos

replace_outliers_with_statistic <- function(x, method = "median", threshold = 2) {

  q1 <- quantile(x, 0.25, na.rm = TRUE)

  q3 <- quantile(x, 0.75, na.rm = TRUE)

  iqr <- q3 - q1

  lower_limit <- q1 - threshold * iqr

  upper_limit <- q3 + threshold * iqr


  if (method == "min") {

    x[x < lower_limit] <- min(x, na.rm = TRUE)

  } else if (method == "max") {
```

```
  x[x > upper_limit] <- max(x, na.rm = TRUE)
 } else if (method == "median") {
  x[x < lower_limit] <- median(x, na.rm = TRUE)
  x[x > upper_limit] <- median(x, na.rm = TRUE)
 } else if (method == "p25") {
  x[x < lower_limit] <- quantile(x, 0.25, na.rm = TRUE)
 } else if (method == "p75") {
  x[x > upper_limit] <- quantile(x, 0.75, na.rm = TRUE)
 }

 return(x)
}
###mejor correlacion con datos transformados
variables_a_procesar <- colnames(continuas)[6:ncol(continuas)]
for (var in variables_a_procesar) {
 continuas[[var]] <- replace_outliers_with_statistic(continuas[[var]], method = "max")
}
```

```
###ESCALAMIENTO
```{r}
library(dplyr)
# Obtener las columnas que deseas escalar (excepto las 4 primeras)
columnas_a_escalar <- colnames(continuas)[6:ncol(continuas)]
```
```

```
# Escalar las columnas seleccionadas con el método Min-Max

datos_escala_min_max <- continuas %>%

  mutate(across(all_of(columnas_a_escalar), ~ (.-min(., na.rm = TRUE)) / (max(., na.rm = TRUE) - min(., na.rm = TRUE))))

# Mantener las primeras 4 columnas y la última sin cambios

datos_escala_min_max <- datos_escala_min_max %>%

  select(1:5, all_of(columnas_a_escalar), ncol(continuas))
```

```
###TRANSFORMACION

```{r}


# Definir las columnas a transformar

columnas_a_transformar <- colnames(datos_escala_min_max)[5:ncol(datos_escala_min_max)]


# Aplicar la transformación log1p a las columnas seleccionadas

datos_transformados <- datos_escala_min_max

datos_transformados[, columnas_a_transformar] <- log1p(datos_escala_min_max[, columnas_a_transformar])


```

```
### HOT ENCONDIG

```{r}

# Función para realizar one-hot encoding

one_hot_encoding <- function(data, variable) {

  # Asegurarse de que la variable sea de tipo factor
```

```
  data[[variable]] <- as.factor(data[[variable]])
  # Realizar one-hot encoding utilizando la función 'model.matrix'
  one_hot_encoded <- model.matrix(~ data[[variable]] - 1, data = data)
  # Obtener los nombres de las categorías originales
  categories <- levels(data[[variable]])
  # Cambiar los nombres de las columnas generadas
  colnames(one_hot_encoded) <- paste(variable, categories, sep = "_")
  # Eliminar la variable categórica original
  data[[variable]] <- NULL
  # Combinar el conjunto de datos original con las nuevas columnas one-hot encoded
  data <- cbind(data, one_hot_encoded)

  return(data)
}

# Realizar one-hot encoding en la variable "crossing"
categ_1 <- one_hot_encoding(categoricas, "crossing")
# Almacenar unicamente varible con presencia de cruce y ID
categ_1 <- data.frame(ID = categ_1$ID, crossing_true = categ_1$crossing_True)
# Realizar one-hot encoding en la variable "give_way"
categ_2 <- one_hot_encoding(categoricas, "give_way")
# Almacenar unicamente varible con presencia de ceda el paso y ID
categ_2 <- data.frame(ID = categ_2$ID, give_way_true = categ_2$give_way_True)
# Realizar one-hot encoding en la variable "stop"
```

```
categ_3 <- one_hot_encoding(categoricas, "stop")

# Almacenar unicamente varible con presencia de pare y ID

categ_3 <- data_frame(ID = categ_3$ID, stop_true = categ_3$stop_True)
```

```
###MAPPING

```{r}

# Definir el mapeo "cond_meteo"

mapeo <- c("Blowing Dust"          =5,                "Blowing Dust / Windy"   =5,
           "Blowing Snow / Windy" =4 ,

        "Clear"=1,                 "Cloudy"=2,                    "Cloudy / Windy"=2,
           "Drizzle"          =3,              "Drizzle / Windy"          =3,

        "Fair"      =1,"Fair / Windy"          =1,                    "Fog"    =5,"Fog / Windy"          =5,
                   "Freezing Rain / Windy"  =3,

        "Funnel Cloud"     =5,              "Hail"   =3,              "Haze"  =5,              "Haze       /
Windy"  =5,              "Heavy Drizzle"  =3,

        "Heavy Rain"       =3,"Heavy Rain / Windy" =3,      "Heavy Snow"    =4,              "Heavy
Snow / Windy"    =4,

        "Heavy T-Storm"   =6,      "Heavy T-Storm / Windy"=6,      "Light Drizzle"    =3,
           "Light Drizzle / Windy"    =3,

        "Light Freezing Drizzle"     =3,      "Light Freezing Fog"       =5,      "Light      Freezing      Rain"
           =3,"Light Freezing Rain / Windy"  =3,

        "Light Rain"       =3,              "Light Rain / Windy"      =3,"Light     Rain     with     Thunder"
           =3,"Light Sleet / Windy"  =3,

        "Light Snow"       =4,      "Light Snow / Windy"     =4,"Light Snow and Sleet"          =4,"Light
Snow and Sleet / Windy"  =4,

        "Light Thunderstorms and Rain"      =6,"Mist"          =5,              "Mostly Cloudy" =2,
           "Mostly Cloudy / Windy" =2,

        "N/A Precipitation"          =2,              "Overcast"       =2,                    "Partly
Cloudy" =2,              "Partly Cloudy / Windy"   =2,

        "Patches of Fog"    =5,                    "Rain"   =3,      "Rain / Windy"   =3,"Scattered
Clouds" =1,      "Shallow Fog"    =5,
```
```

```
        "Small Hail"          =3,"Smoke"        =5,        "Smoke / Windy"           =5,"Snow"         =4,
          "Snow / Windy"  =4,"Squalls"       =5,

        "T-Storm" =6,"T-Storm / Windy"      =6,"Thunder"      =6,                "Thunder        /        Windy"
          =6,"Thunder in the Vicinity"        =6,

        "Thunderstorm"     =6,       "Widespread Dust / Windy"          =5,"Wintry Mix" =2,"Wintry     Mix     /
Windy"  =2)


# Realizar el mapeo y reemplazar los valores en el campo

map <- categoricas %>%

  mutate(cond_meteo = ifelse(cond_meteo %in% names(mapeo), mapeo[cond_meteo], cond_meteo))


map$cond_meteo <- as.numeric(map$cond_meteo)

# Crear las nuevas variables binarias

map <- map %>%

  mutate(despejado = as.numeric(map$cond_meteo == 1),

         nublado = as.numeric(map$cond_meteo == 2),

         lluvia = as.numeric(map$cond_meteo == 3),

         nieve = as.numeric(map$cond_meteo == 4),

         niebla_polvo = as.numeric(map$cond_meteo == 5),

         tormenta = as.numeric(map$cond_meteo == 6))


# Definir el mapeo variable "hour_of_day"

mapeo1 <- c("0" = "0",

            "1" = "0",

            "2" = "0",

            "3" = "0",
```

```
"4" = "0",
"5" = "0",
"6" = "1",
"7" = "1",
"8" = "1",
"9" = "1",
"10" = "1",
"11" = "1",
"12" = "2",
"13" = "2",
"14" = "2",
"15" = "2",
"16" = "2",
"17" = "2",
"18" = "2",
"19" = "2",
"20" = "2",
"21" = "3",
"22" = "3",
"23" = "3")

# Realizar el mapeo y reemplazar los valores en el campo
map <- map %>%
  mutate(hour_of_day = ifelse(hour_of_day %in% names(mapeo1), mapeo1[hour_of_day], hour_of_day))
```

```
map$hour_of_day <- as.numeric(map$hour_of_day)

# Crear las nuevas variables binarias

map <- map %>%

  mutate(madrugada = as.numeric(map$hour_of_day == 0),

         dia = as.numeric(map$hour_of_day == 1),

         tarde = as.numeric(map$hour_of_day == 2),

         noche = as.numeric(map$hour_of_day == 3))


#Union de data procedente de hot encoding y mapping

categ_uni <- merge(map, categ_1, by="ID", all=FALSE)

categ_uni <- merge(categ_uni, categ_2, by="ID", all=FALSE)

categ_uni <- merge(categ_uni, categ_3, by="ID", all=FALSE)

#Eliminar variables categoricas de la union anterior

categ_uni <- data_frame(ID = categ_uni$ID,

                        madrugada = categ_uni$madrugada,

                        dia = categ_uni$dia,

                        tarde = categ_uni$tarde,

                        noche = categ_uni$noche,

                        nublado = categ_uni$nublado,

                        despejado = categ_uni$despejado,

                        lluvia = categ_uni$lluvia,

                        nieve = categ_uni$nieve,

                        niebla_polvo = categ_uni$niebla_polvo,
```

```
                tormenta = categ_uni$tormenta,
                crossing_true = categ_uni$crossing_true,
                stop_true = categ_uni$stop_true,
                give_way_true = categ_uni$give_way_true)
```

```
##UNION MEJOR CORRELACION CONTINUAS Y CATEGORICAS TRATADAS
```

```{r}
##union de variables categoricas codificadas y continuas
datas <- merge(categ_uni, datos_transformados, by="ID", all=FALSE)
# Función para crear un heatmap de correlación
crear_heatmap_correlacion <- function(matriz_correlacion, titulo = "Heatmap de Correlación") {
  # Convertir la matriz de correlación en un formato adecuado para ggplot
  cor_matrix_melted <- melt(matriz_correlacion)
  # Crear el heatmap utilizando ggplot2
  ggplot(cor_matrix_melted, aes(X1, X2, fill = value)) +
    geom_tile() +
    geom_text(aes(label = round(value, 2)), color = "black", size = 4) +
    scale_fill_gradient2(low = "#11AAAA", mid = 'white', high = "red") +
    labs(title = titulo) +
    theme_minimal() +
    theme(
      axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
      axis.text.y = element_text(vjust = 0.5),
```

```
    plot.title = element_text(hjust = 0.5)
  )
}
# Correlacion de variables continuas y categoricas codificadas omite el campo ID
matriz_correlacion <- cor(datas[, -1])
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación dependiente v/s independientes")
```

##PARTICION DATOS DE ENTRENAMIENTO Y PRUEBA DEL MODELO

```{r}
##PARTICION DE DATOS
set.seed(123)  # Establecer una semilla
split <- sample.split(datas$severity, SplitRatio = 0.7)
entrenamiento <- subset(datas, split == TRUE)#datos de entrenamiento
prueba <- subset(datas, split == FALSE)#datos prueba

###GRAFICO DE PORCENTAJES POR CATEGORIA
frecuencias <- table(entrenamiento$severity)# Calcular las frecuencias
porcentajes <- prop.table(frecuencias) * 100# Calcular porcentajes
resultados <- data.frame(Categoria = names(frecuencias), Frecuencia = as.numeric(frecuencias), Porcentaje = porcentajes)
# Crear el gráfico de barras
ggplot(resultados, aes(x = order(resultados$Categoria), y = resultados$Porcentaje.Freq, fill = resultados$Categoria)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = sprintf("%.1f%%", resultados$Porcentaje.Freq)), vjust = -0.5) +
  labs(x = "Categorías", y = "Porcentaje (%)", title = "Gráfico de Barras Acumulado con Frecuencias") +
```

```
  theme_minimal() +
  theme(legend.position = "none")



##DUPLICAR
duplicar_categoria <- function(df, categoria) {
  datos <- df  # Copia el data frame original para no modificarlo
  max_obs <- max(table(datos$severity)) #maxima observacion de la data
  #duplicar hasta que llegue a la misma cantidad de datos de la variable de mayor observaciones
  while (table(datos$severity)[categoria] < max_obs) {
    indice_aleatorio <- sample(which(datos$severity == categoria), 1)
    nueva_fila <- datos[indice_aleatorio, ]
    datos <- rbind(datos, nueva_fila)
  }

  return(datos)
}
entrenamiento_1<- entrenamiento
# Llama a la función y la categoría a duplicar
entrenamiento_1 <- duplicar_categoria(entrenamiento_1, 1)#duplica datos categoria 1
entrenamiento_1 <- duplicar_categoria(entrenamiento_1, 3)#duplica datos categoria 3
entrenamiento_1 <- duplicar_categoria(entrenamiento_1, 4)#duplica datos categoria 4


##CONVERTIR VARIABLE DEPENDIENTE A DUMMY
```

```
entrenamiento_dummy <- entrenamiento

entrenamiento_dummy$severity <- ifelse(entrenamiento_dummy$severity %in% c(3,4), 1, 0)

prueba_dummy <- prueba

prueba_dummy$severity <- ifelse(prueba_dummy$severity %in% c(3,4), 1, 0)

###DUPLICAR DATOS

# Calcular el número de observaciones con dummy_var igual a 0

num_obs_0 <- sum(entrenamiento_dummy$severity == 0)

# Filtrar las filas con dummy_var igual a 1

filas_dummy_1 <- entrenamiento_dummy[entrenamiento_dummy$severity == 1, ]

# Duplicar las filas igual a 1 hasta que alcance la misma cantidad de observaciones que 0

while (sum(entrenamiento_dummy$severity == 1) < num_obs_0) {

  entrenamiento_dummy <- rbind(entrenamiento_dummy, filas_dummy_1)

}
```

```
##MODELO LINEAL MANUAL

```{r}

formula_1 = "severity ~ precipitacion + visibilidad + velocidad_viento + distancia + despejado + nublado + lluvia + nieve +  tormenta + crossing_true + stop_true + give_way_true + dia + tarde + noche"

modelo_lm_1 <- lm(formula_1, data= entrenamiento_1)

summary(modelo_lm_1)

plot(modelo_lm_1)



formula_2 = "severity ~ visibilidad + velocidad_viento + distancia + despejado + nublado + lluvia + nieve + niebla_polvo + crossing_true + give_way_true + dia + tarde + madrugada"

modelo_lm_2 <- lm(formula_2, data= entrenamiento_1)

summary(modelo_lm_2)
```
```

```
plot(modelo_lm_2)



formula_3 = "severity ~ precipitacion + velocidad_viento + distancia + despejado + nublado  + nieve + tormenta + crossing_true + stop_true + give_way_true + dia + tarde + noche"

modelo_lm_3 <- lm(formula_3, data= entrenamiento_1)

summary(modelo_lm_3)

plot(modelo_lm_3)



formula_4 = "severity ~ precipitacion + visibilidad + velocidad_viento + distancia + despejado + nublado + lluvia + nieve +  tormenta + crossing_true + stop_true + give_way_true + dia + tarde + noche -1"

modelo_lm_4 <- lm(formula_4, data= entrenamiento_1)

summary(modelo_lm_4)

plot(modelo_lm_4)



```

##GRAFICOS RESIDUOS VS PREDICHOS & PREDICCION VS VALOR REAL

```{r}

# Gráfico de residuos vs. valores predichos

residuals <- resid(modelo_lm_4)

plot(fitted(modelo_lm_4), residuals, main="Gráfico de Residuos vs. Predicciones", xlab="Predicciones", ylab="Residuos")



# Gráfico de dispersión de predicciones vs. valores reales

nuevas_predicciones <- predict(modelo_lm_4, newdata = prueba)

plot(prueba$severity, nuevas_predicciones, main="Gráfico de Predicciones vs. Valores Reales", xlab="Valores Reales", ylab="Predicciones")
```

```

##METODO STEP BOTH

```{r}

modelo_inicial <- lm(severity ~ precipitacion + visibilidad + velocidad_viento + distancia + despejado + nublado + lluvia + crossing_true + stop_true + give_way_true + dia + tarde + noche , data=entrenamiento_1)

modelo_step_both <- step(modelo_inicial, direction="both")

summary(modelo_step_both)

plot(modelo_step_both)

```

## MODELO LOGISTICO MANUAL 1-- EN CADA MODELO VA --> MODELO-MATRIZ-CURVA ROC-CURVA PREDICCIONES

```{r}

formula_1 <- "severity ~ precipitacion + visibilidad + velocidad_viento + distancia + despejado + nublado + lluvia + nieve +  tormenta + crossing_true + stop_true + give_way_true + dia + tarde + noche"

modelo_logistico_1 <- glm(formula_1, data=entrenamiento_dummy, family = "binomial")

summary(modelo_logistico_1)



###MATRIZ DE CONFUSION

# Realizar predicciones

predicciones <- predict(modelo_logistico_1, newdata = prueba_dummy, type = "response")

valores_reales <- prueba_dummy$severity

predicted_class <- ifelse(predicciones > 0.5, "grave", "leve")# definicion pto de corte

reales_class <- ifelse(valores_reales > 0.5, "grave", "leve")

# Convierte ambos en factores con los mismos niveles

predicted_class <- factor(predicted_class, levels = c("grave", "leve"))

reales_class <- factor(reales_class, levels = c("grave", "leve"))
```
```

```
# Crea la matriz de confusión con factores

matriz_confusion <- confusionMatrix(data = predicted_class, reference = reales_class)

matriz_confusion


###CURVA ROC

# Crear un objeto ROCR para la curva ROC

roc_obj <- prediction(predicciones, prueba_dummy$severity)

# Calcular las métricas de la curva ROC

roc_perf <- performance(roc_obj, "tpr", "fpr")

# Graficar la curva ROC

plot(roc_perf, main = "Curva ROC")

abline(a = 0, b = 1, col = "red")  # Línea de referencia diagonal


##CURVA DE PREDICCIONES

predicted.data <- data.frame(

  probability.of.h=modelo_logistico_1$fitted.values,

  hd=entrenamiento_dummy$severity)


predicted.data <- predicted.data[

  order(predicted.data$probability.of.h, decreasing=FALSE),]

predicted.data$rank <- 1:nrow(predicted.data)


ggplot(data=predicted.data, aes(x=rank, y=probability.of.h)) +

  geom_point(aes(color=hd), alpha=1, shape=4, stroke=2) +
```

```
  xlab("Index") +
  ylab("Predicted probability")
```

```

##MODELO LOGISTICO 2

```{r}

###MODELO FORMULA 2

formula_2 <- "severity ~ precipitacion + visibilidad + distancia + nublado + lluvia + nieve + crossing_true + tarde + noche"

modelo_logistico_2 <- glm(formula_2, data=entrenamiento_dummy, family = "binomial")

summary(modelo_logistico_2)


###MATRIZ DE CONFUSION

#realizar predicciones

predicciones <- predict(modelo_logistico_2, newdata = prueba_dummy, type = "response")

valores_reales <- prueba_dummy$severity


predicted_class <- ifelse(predicciones > 0.5, "grave", "leve")#definicion pto de corte

reales_class <- ifelse(valores_reales > 0.5, "grave", "leve")


# Convierte ambos en factores con los mismos niveles

predicted_class <- factor(predicted_class, levels = c("grave", "leve"))

reales_class <- factor(reales_class, levels = c("grave", "leve"))


# Crea la matriz de confusión con factores
```
```

```
matriz_confusion <- confusionMatrix(data = predicted_class, reference = reales_class)

matriz_confusion


###CURVA ROC

# Crear un objeto ROCR para la curva ROC

roc_obj <- prediction(predicciones, prueba_dummy$severity)

# Calcular las métricas de la curva ROC

roc_perf <- performance(roc_obj, "tpr", "fpr")

# Graficar la curva ROC

plot(roc_perf, main = "Curva ROC")

abline(a = 0, b = 1, col = "red")  # Línea de referencia diagonal


##CURVA DE PREDICCIONES

predicted.data <- data.frame(

  probability.of.h=modelo_logistico_2$fitted.values,

  hd=entrenamiento_dummy$severity)


predicted.data <- predicted.data[

  order(predicted.data$probability.of.h, decreasing=FALSE),]

predicted.data$rank <- 1:nrow(predicted.data)


ggplot(data=predicted.data, aes(x=rank, y=probability.of.h)) +

  geom_point(aes(color=hd), alpha=1, shape=4, stroke=2) +

  xlab("Index") +
```

ylab("Predicted probability")

```

##MODELO LOGISTICO 3

```{r}

###MODELO FORMULA 3

formula_3 <- "severity ~ precipitacion + velocidad_viento + distancia + despejado + nublado + nieve + tormenta + crossing_true + stop_true + give_way_true + dia + tarde + noche"

modelo_logistico_3 <- glm(formula_3, data=entrenamiento_dummy, family = "binomial")

summary(modelo_logistico_3)

###MATRIZ DE CONFUSION

#realizar predicciones

predicciones <- predict(modelo_logistico_3, newdata = prueba_dummy, type = "response")

valores_reales <- prueba_dummy$severity

predicted_class <- ifelse(predicciones > 0.5, "grave", "leve")#definicion pto de corte

reales_class <- ifelse(valores_reales > 0.5, "grave", "leve")

# Convierte ambos en factores con los mismos niveles

predicted_class <- factor(predicted_class, levels = c("grave", "leve"))

reales_class <- factor(reales_class, levels = c("grave", "leve"))

# Crea la matriz de confusión con factores

matriz_confusion <- confusionMatrix(data = predicted_class, reference = reales_class)

matriz_confusion
```
```

```
###CURVA ROC

# Crear un objeto ROCR para la curva ROC

roc_obj <- prediction(predicciones, prueba_dummy$severity)

# Calcular las métricas de la curva ROC

roc_perf <- performance(roc_obj, "tpr", "fpr")

# Graficar la curva ROC

plot(roc_perf, main = "Curva ROC")

abline(a = 0, b = 1, col = "red")  # Línea de referencia diagonal


##CURVA DE PREDICCIONES

predicted.data <- data.frame(

  probability.of.h=modelo_logistico_3$fitted.values,

  hd=entrenamiento_dummy$severity)


predicted.data <- predicted.data[

  order(predicted.data$probability.of.h, decreasing=FALSE),]

predicted.data$rank <- 1:nrow(predicted.data)


ggplot(data=predicted.data, aes(x=rank, y=probability.of.h)) +

  geom_point(aes(color=hd), alpha=1, shape=4, stroke=2) +

  xlab("Index") +

  ylab("Predicted probability")
```

###MODELO LOGISTICO MANUAL 4

```{r}

###MODELO FORMULA 4

formula_4 <- "severity ~ precipitacion + visibilidad + velocidad_viento + distancia + despejado + nublado + lluvia + crossing_true + stop_true + give_way_true + dia + tarde + noche -1"

modelo_logistico_4 <- glm(formula_4, data=entrenamiento_dummy, family = "binomial")

summary(modelo_logistico_4)


###MATRIZ DE CONFUSION

#Realizar predicciones

predicciones <- predict(modelo_logistico_4, newdata = prueba_dummy, type = "response")

valores_reales <- prueba_dummy$severity


predicted_class <- ifelse(predicciones > 0.5, "grave", "leve")#definicion pto de corte

reales_class <- ifelse(valores_reales > 0.5, "grave", "leve")


# Convierte ambos en factores con los mismos niveles

predicted_class <- factor(predicted_class, levels = c("grave", "leve"))

reales_class <- factor(reales_class, levels = c("grave", "leve"))


# Crea la matriz de confusión con factores

matriz_confusion <- confusionMatrix(data = predicted_class, reference = reales_class)

matriz_confusion


###CURVA ROC
```

```
# Crear un objeto ROCR para la curva ROC

roc_obj <- prediction(predicciones, prueba_dummy$severity)

# Calcular las métricas de la curva ROC

roc_perf <- performance(roc_obj, "tpr", "fpr")

# Graficar la curva ROC

plot(roc_perf, main = "Curva ROC")

abline(a = 0, b = 1, col = "red")  # Línea de referencia diagonal


##CURVA DE PREDICCIONES

predicted.data <- data.frame(

  probability.of.h=modelo_logistico_4$fitted.values,

  hd=entrenamiento_dummy$severity)


predicted.data <- predicted.data[

  order(predicted.data$probability.of.h, decreasing=FALSE),]

predicted.data$rank <- 1:nrow(predicted.data)


ggplot(data=predicted.data, aes(x=rank, y=probability.of.h)) +

  geom_point(aes(color=hd), alpha=1, shape=4, stroke=2) +

  xlab("Index") +

  ylab("Predicted probability")


```