UNIVERSIDAD DE GRANADA

Práctica 1:
Preparación de datos

Autor
Juan Manuel Castillo Nievas

Máster Profesional en Ingeniería Informática

—

Granada, 15 de diciembre de 2020

Índice

1. Introducción

En esta práctica se ha trabajado con el dataset **accidentes.xls**. La idea es preparar los datos poco a poco para ir mejorando el porcentaje de clasificación. Para trabajar con este dataset, primero se han realizado dos tratamientos utilizando **LibreOffice Calc**:

1. **Valores desconocidos:** se han dejado en blanco las celdas de aquellas variables que contienen datos perdidos. Por ejemplo, las celdas de la columna **HOUR** cuyo valor es **99**, se dejan en blanco.
2. **Construir la variable clase:** se ha creado una nueva columna cuyo nombre es **ACCIDENTE** y que contiene la descripción de la gravedad del accidente. Dependiendo de si es uno u otro, esta columna tiene 3 posibles valores: **FATALITIES**, **INJURY_CRASH** y **PRPTYDMG_CRASH**.

También se ha realizado una **única partición de datos de entrenamiento** (80 %) y **prueba** (20 %).

Por último, se han creado dos datasets: **accidentes_sin_imputar**, que contiene todo el dataset con las columnas sin imputar; y **accidentes_imputados**, que contiene todo el dataset con las columnas imputadas.

Una vez hecho esto, en las siguientes secciones se van a explicar los procesos que se han seguido para el tratamiento de datos.

2. Discretización

Se han creado dos data frames llamados **train_frame_i** y **test_frame_i** que contiene el conjunto de datos de entrenamiento y prueba con las **variables imputadas**, respectivamente.

2.1. Algoritmo de clasificación

Sin hacer previamente ningún tratamiento de datos, se ha ejecutado el algoritmo de clasificación **C5.0** con el conjunto con valores imputados para ver cómo se dividen las características (ver Listing 1).

```
train_frame_i$ACCIDENTE<-as.factor(train_frame_i$ACCIDENTE)
model <- C5.0(train_frame_i[-25],train_frame_i$ACCIDENTE)
summary(model)
predictions <- predict(model, test_frame_i, type="class")

table(predictions, test_frame_i$ACCIDENTE)
```

Listing 1: Algoritmo de clasificación C5.0

En la Figura 1 se puede ver la salida del modelo creado. En la Figura 2 se puede ver una tabla con los resultados. En las diagonales aparecen los aciertos de cada clase.

```
Evaluation on training data (44901 cases):

          Decision Tree
        ----------------
        Size      Errors

         355 15662(34.9%)   <<


         (a)   (b)   (c)    <-classified as
        ----  ----  ----
          23   285   121    (a): class FATALITIES
           3 12649  9022    (b): class INJURY_CRASH
           1  6230 16567    (c): class PRPTYDMG_CRASH
```

Figura 1: Salida de **summary(model)**

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	1	4
INJURY_CRASH	80	3375	2283
PRPTYDMG_CRASH	22	1960	3418

Figura 2: Tabla de predicciones con el conjunto test

El árbol de decisión tiene un **tamaño de 355** y hay un **34.9 % de errores**.

2.2. Algoritmo de discretización top-down CAIM

Se ha usado la librería **discretization**. Se ha aplicado el algoritmo top-down CAIM sobre todas las variables numéricas y se ha vuelto a aplicar el algoritmo de clasificación (ver Listing 2).

```
accidentes_imputados_data_frame <- as.data.frame.data.frame(accidentes_imputados)
caim=disc.Topdown(accidentes_imputados_data_frame, method=1)
# Intervalos
caim$cutp
# Datos discretizados
accidentes_imputados_disc = caim$Disc.data

ind=sample(2,nrow(accidentes_imputados_disc),replace=TRUE,prob=c(0.8,0.2))
train_imputados_disc=accidentes_imputados_disc[ind==1,]
test_imputados_disc=accidentes_imputados_disc[ind==2,]

# Se crea el modelo
train_imputados_disc$ACCIDENTE<-as.factor(train_imputados_disc$ACCIDENTE)
model <- C5.0(train_imputados_disc[-25],train_imputados_disc$ACCIDENTE)
summary(model)
predictions <- predict(model, test_frame_i, type="class")
```

Listing 2: Discretización top-down CAIM y clasificación

En la Figura 3 se puede ver la salida del modelo creado. En la Figura 4 se puede ver una tabla con los resultados. En las diagonales aparecen los aciertos de cada clase. Como se puede ver, al discretizar **todas las variables** se obtiene un peor resultado (**41.6 % de errores**). Es cierto que el tamaño del árbol de decisión baja a **70**.

En la próxima sección, se van a discretizar sólamente algunas variables de acuerdo a mi propio criterio.

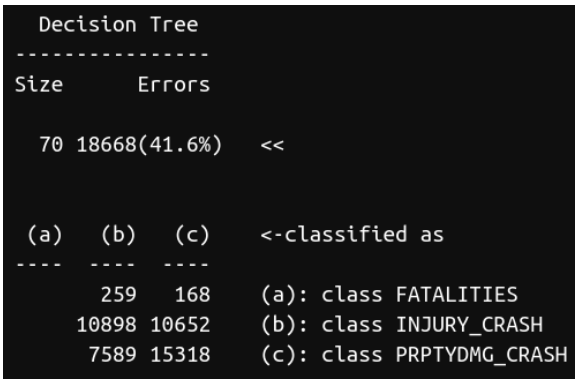

Figura 3: Salida de **summary(model)** una vez aplicada la discretización top-down CAIM

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	0	0
INJURY_CRASH	49	2513	1781
PRPTYDMG_CRASH	56	2805	3943

Figura 4: Tabla de predicciones con el conjunto de test una vez aplicada la discretización top-down CAIM

2.3. Mejora de la discretización

En este apartado se proponen discretizaciones de varias variables de acuerdo al significado de la característica, visualización de datos, etc.

2.3.1. WKDY_I

Esta variable contiene un número del 1 al 7 que representa el día de la semana, siendo 1 domingo y 7 sábado. En la Figura 5 se muestra un histograma con las frecuencias. Al ver este histograma, he decidido discretizar esta variable de acuerdo a lo siguiente (ver Listing 3):

- **semana:** se tomará el valor "semana" si el valor está en el intervalo [1, 5], que representa los días de semana.
- **fin de semana:** se tomará el valor "fin de semana" si el valor está en [6,7], correspondiente a los días **viernes** y **sábado**.

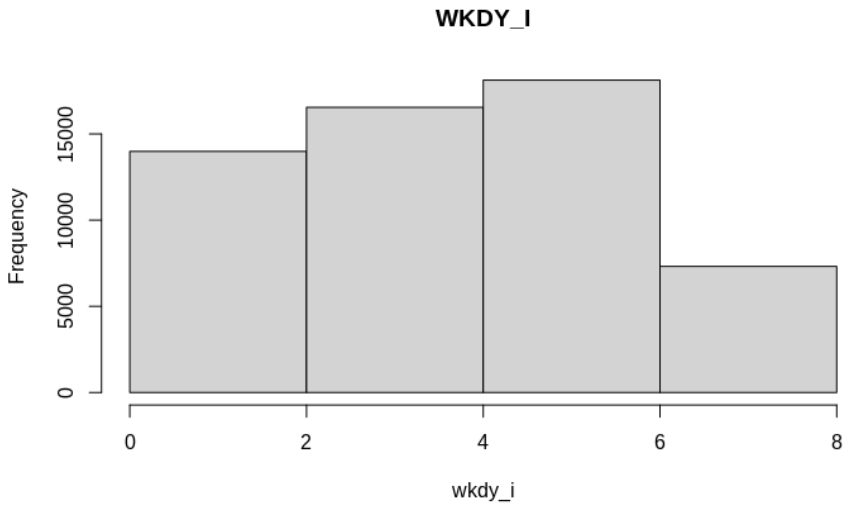

Figura 5: Histograma de **WKDY_I**

```r
wkdy_i <- accidentes_imputados_disc[,2]
hist(wkdy_i, breaks = 3, main = "WKDY_I")
accidentes_imputados_disc$WKDY_I <- cut(accidentes_imputados_disc$WKDY_I,
                                        breaks = c(-Inf, 6, +Inf),
                                        labels = c("semana", "fin de semana"),
                                        right = FALSE)
```

Listing 3: Discretización de **WKDY_I**

2.3.2. HOUR_I

Esta variable contiene un número del 0 al 24 que representa la hora del día en la que ha ocurrido el accidente en formato HH. En la Figura 6 se muestra un histograma con las frecuencias. Al ver este histograma, he decidido discretizar esta variable de acuerdo a lo siguiente (ver Listing 4):

- **hora punta:** se tomará el valor "hora punta" si el accidente ha ocurrido en el intervalo [5,20)
- **no:** se tomará el valor "no" si el accidente no ha ocurrido en hora punta

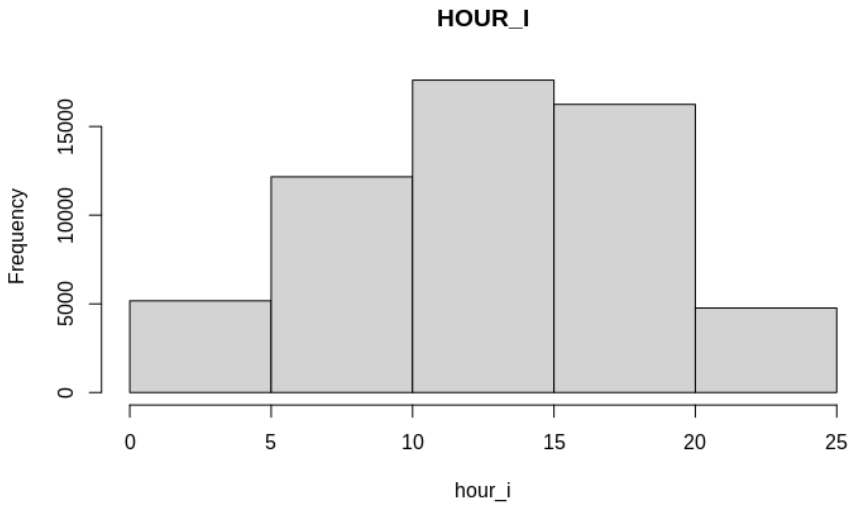

Figura 6: Histograma de **HOUR_I**

```r
hour_i <- accidentes_imputados_disc[,3]
hist(hour_i, breaks = 5, main = "HOUR_I")
accidentes_imputados_disc$HOUR_I <- cut(accidentes_imputados_disc$HOUR_I,
                                        breaks = c(-Inf, 5, 20, +Inf),
                                        labels = c("no", "hora punta", "no"),
                                        right = FALSE)
```

Listing 4: Discretización de **HOUR_I**

2.3.3. SPDLIM_H

Esta variable contiene el límite de velocidad permitido en el lugar donde ha ocurrido el accidente en millas por hora. He decidido discretizar esta variable igualando en frecuencia. Para ello, he usado la librería arules. La discretización ha sido dividida de acuerdo a 4 intervalos (ver Listing 5):

- **muy baja:** se tomará el valor "muy baja" si el valor está en el intervalo [0, 35)
- **baja:** se tomará el valor "baja" si el valor está en el intervalo [35, 40)
- **alta:** se tomará el valor "alta" si el valor está en el intervalo [40, 55)
- **alta:** se tomará el valor "muy baja" si el valor está en el intervalo [55, 75]

```r
accidentes_imputados_disc <- discretizeDF(accidentes_imputados_disc, methods = list(
          SPDLIM_H = list(method = "frequency", breaks = 4,
                          labels = c("muy baja", "baja", "normal", "alta"))
          ),
          default = list(method = "none")
```

Listing 5: Discretización de **SPDLIM_H**

2.3.4. VEH_INVL

Esta variable contiene el número de vehículos involucrados en el accidente. La discretización ha sido dividida de acuerdo a mi criterio (ver Listing 6):

- **multiple:** se tomará el valor "múltiple" si en el accidente hay 2 o más coches involucrados
- **individual:** se tomará el valor "individual" si en el accidente hay sólo un coche involucrado

```r
accidentes_imputados_disc <- accidentes_imputados_disc$VEH_INVL <- cut(accidentes_imputados_disc$V
                                    breaks = c(-Inf, 2, +Inf),
                                    labels = c("individual", "multiple"),
                                    right = FALSE)
```

Listing 6: Discretización de **VEH_INVL**

2.3.5. NON_INVL

Esta variable contiene el número de no motoristas involucrados en el accidente. La discretización ha sido dividida de acuerdo a mi criterio (ver Listing 8):

- **no:** se tomará el valor "no" si en el accidente no hay vehículos no motoristas involucrados, es decir, el valor es 0.
- **si:** se tomará el valor "si" si en el accidente hay vehículos no motoristas involucrados

```r
accidentes_imputados_disc$NON_INVL <- cut(accidentes_imputados_disc$NON_INVL,
                                    breaks = c(-Inf, 1, +Inf),
                                    labels = c("no", "si"),
                                    right = FALSE)
```

Listing 7: Discretización de **NON_INVL**

2.3.6. PED_ACC

Esta variable contiene un número que indica el tipo de accidente según los involucrados (peatonal, motoristas, ciclistas, etc). La discretización ha sido dividida de acuerdo a mi criterio (ver Listing 8):

- **no:** se tomará el valor "no" si en el accidente no ha habido peatones o ciclistas involucrados
- **si:** se tomará el valor "si" si en el accidente ha habido peatones o ciclistas involucrados

```r
accidentes_imputados_disc$PED_ACC <- cut(accidentes_imputados_disc$PED_ACC,
                                         breaks = c(-Inf, 1, +Inf),
                                         labels = c("no", "si"),
                                         right = FALSE)
```

Listing 8: Discretización de **PED_ACC**

2.3.7. Clasificación

Una vez se ha hecho la discretización propuesta, se ha vuelto a clasificar con el algoritmo C5.0 (ver Listing 9).

```r
# Se convierte a factor la clase variable y a variable numérica las variables
# que se han discretizado
accidentes_imputados_disc$ACCIDENTE<-as.factor(accidentes_imputados_disc$ACCIDENTE)
accidentes_imputados_disc$HOUR_I<-as.numeric(accidentes_imputados_disc$HOUR_I)
accidentes_imputados_disc$SPDLIM_H<-as.numeric(accidentes_imputados_disc$SPDLIM_H)
accidentes_imputados_disc$WKDY_I<-as.numeric(accidentes_imputados_disc$WKDY_I)
accidentes_imputados_disc$NON_INVL<-as.numeric(accidentes_imputados_disc$NON_INVL)
accidentes_imputados_disc$VEH_INVL<-as.numeric(accidentes_imputados_disc$VEH_INVL)
accidentes_imputados_disc$PED_ACC<-as.numeric(accidentes_imputados_disc$PED_ACC)

# Separo el dataset en conjunto de entrenamiento y test (80% y 20%)
ind=sample(2,nrow(accidentes_imputados_disc),replace=TRUE,prob=c(0.8,0.2))
train_acc_imp_disc=accidentes_imputados_disc[ind==1,]
test_acc_imp_disc=accidentes_imputados_disc[ind==2,]

# Se crea el modelo
model <- C5.0(train_acc_imp_disc[-25],train_acc_imp_disc$ACCIDENTE)
summary(model)
predictions <- predict(model, test_acc_imp_disc, type="class")

# Los resultados en la diagonal principal
# muestran los aciertos del modelo solamente
table(predictions, test_acc_imp_disc$ACCIDENTE)
```

Listing 9: Algoritmo de clasificación con la discretización propuesta

En la Figura 7 se puede ver la salida del modelo creado. En la Figura 8 se puede ver una tabla con los resultados. En las diagonales aparecen los aciertos de cada clase. Como se puede ver, al hacer una discretización con mi criterio se ha mejorado el resultado (**36.3 % de errores**) con respecto a la clasificación anterior (**41.6 % de errores**). El tamaño del árbol de decisión es de **299**.

```
Evaluation on training data (44834 cases):

           Decision Tree
         ----------------
         Size      Errors

          299 16294(36.3%)   <<


          (a)   (b)   (c)    <-classified as
         ----  ----  ----
                312   121    (a): class FATALITIES
               12999  8584    (b): class INJURY_CRASH
                7277 15541    (c): class PRPTYDMG_CRASH
```

Figura 7: Salida de **summary(model)** con mi discretización

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	0	0
INJURY_CRASH	79	2990	2064
PRPTYDMG_CRASH	19	2324	3654

Figura 8: Tabla de predicciones con el conjunto test con mi discretización

3. Valores perdidos

En este punto se va a trabajar con las variables que contienen valores perdidos, es decir, aquellas variables que no están imputadas.

3.1. Algoritmo de clasificación sobre los datos sin imputar

Sin hacer previamente ningún tratamiento de datos, se ha ejecutado el algoritmo de clasificación **C5.0** con el conjunto sin valores imputados (ver Listing 10).

```
train_frame$ACCIDENTE<-as.factor(train_frame$ACCIDENTE)
# Se crea el modelo
model <- C5.0(train_frame[-25],train_frame$ACCIDENTE)
summary(model)
predictions <- predict(model, test_frame, type="class")

# Los resultados en la diagonal principal
# muestran los aciertos del modelo solamente
table(predictions, test_frame$ACCIDENTE)
```

Listing 10: Algoritmo de clasificación C5.0 con los datos sin imputar

En la Figura 9 se puede ver la salida del modelo creado. En la Figura 10 se puede ver una tabla con los resultados. En las diagonales aparecen los aciertos de cada clase.

```
Evaluation on training data (44787 cases):

          Decision Tree
        ----------------
        Size      Errors

         286 15720(35.1%)   <<


         (a)   (b)   (c)    <-classified as
        ----  ----  ----
          12   328    79    (a): class FATALITIES
             14122  7457    (b): class INJURY_CRASH
           2  7854 14933    (c): class PRPTYDMG_CRASH
```

Figura 9: Salida de **summary(model)**

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	1	0
INJURY_CRASH	77	3071	2053
PRPTYDMG_CRASH	25	2264	3652

Figura 10: Tabla de predicciones con el conjunto test

El árbol de decisión tiene un **tamaño de 286** y hay un **35.1 % de errores**, similar a la clasificación que se hizo en la Sección 2.1 con los valores imputados.

3.2. Valores perdidos con la media o moda

En esta sección se van a reemplazar los valores perdidos con la media o moda y se va a comprobar el efecto que tiene. En cada subsección se va a mostrar una gráfica de los valores de cada variable para ver si es mejor reemplazar los valores perdidos con la media o la moda.

Nota: la variable **WEEKDAY**, a pesar de que tienen una columna **_I**, no contiene valores perdidos y por lo tanto no se va a hacer nada sobre ella.

3.2.1. HOUR

En la Figura 11 se muestra un gráfico de barras con las instancias de la variable **HOUR**. Se van a reemplazar los valores perdidos usando la media, como se muestra en el código de la Listing 11. Los valores perdidos se reemplazan por **13**, indicando que el accidente se ha producido a las 13:00.

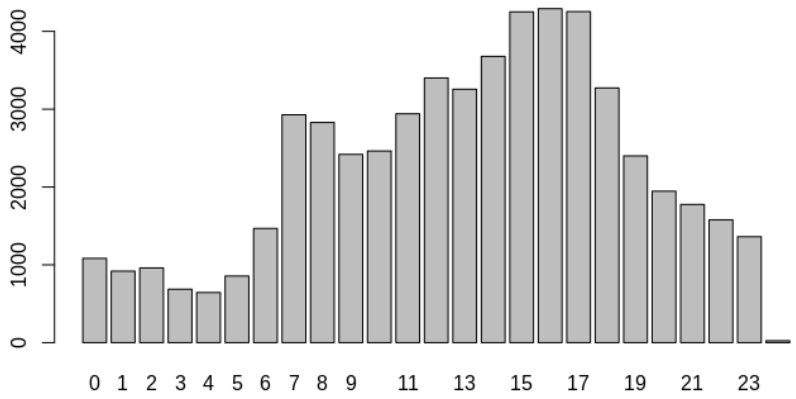

Figura 11: Gráfico de barras de la variable **HOUR**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$HOUR))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$HOUR))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$HOUR=ifelse(
  is.na(accidentes_sin_imputar_data_frame$HOUR),
  as.integer(mean(accidentes_sin_imputar_data_frame$HOUR,na.rm=T)),
  as.integer(accidentes_sin_imputar_data_frame$HOUR))
```

Listing 11: Valores perdidos **HOUR** con la media

3.2.2. MAN_COL

En la Figura 12 se muestra un gráfico de barras con las instancias de la variable **MAN_COL**, que indica la manera de colisión. No tiene sentido usar la media, así que se van a reemplazar los valores perdidos usando la moda, como se muestra en el código de la Listing 12. Los valores perdidos se reemplazan por **0**.

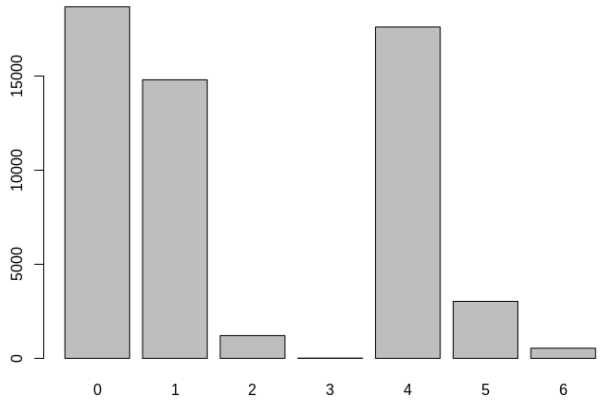

Figura 12: Gráfico de barras de la variable **MAN_COL**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$MAN_COL))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$MAN_COL))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$MAN_COL=ifelse(
  is.na(accidentes_sin_imputar_data_frame$MAN_COL),
  as.integer(names(which.max(table(accidentes_sin_imputar_data_frame$MAN_COL,useNA="no")))),
  as.integer(accidentes_sin_imputar_data_frame$MAN_COL))
```

Listing 12: Valores perdidos **MAN_COL** con la moda

3.2.3. PROFILE

En la Figura 13 se muestra un gráfico de barras con las instancias de la variable **PROFILE**, que indica el perfil de la carretera. En el gráfico se ve que no tiene sentido usar la media porque realmente toma un valor clasificando el perfil de la carretera y además provocaría un valor que no tiene sentido, así que se van a reemplazar los valores perdidos usando la moda, como se muestra en el código de la Listing 13. Los valores perdidos se reemplazan por **1**, que indica que el perfil de la carretera es **level**.

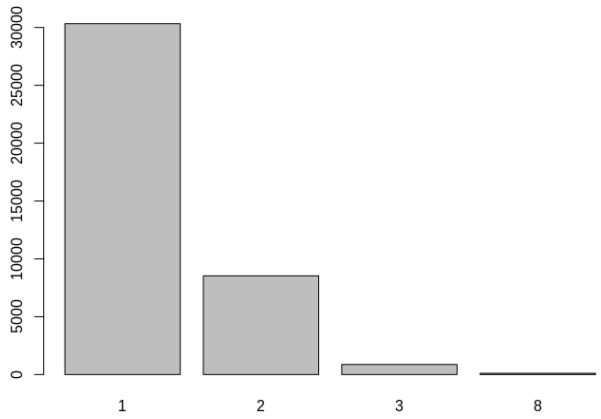

Figura 13: Gráfico de barras de la variable **PROFILE**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$PROFILE))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$PROFILE))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$PROFILE=ifelse(
  is.na(accidentes_sin_imputar_data_frame$PROFILE),
  as.integer(names(which.max(table(accidentes_sin_imputar_data_frame$PROFILE,useNA="no")))),
  as.integer(accidentes_sin_imputar_data_frame$PROFILE))
```

Listing 13: Valores perdidos **PROFILE** con la moda

3.2.4. REL_JCT

En la Figura 14 se muestra un gráfico de barras con las instancias de la variable **REL_JCT**, que indica la relación con un cruce de carretera. En el gráfico se ve que no tiene sentido usar la media porque realmente toma un valor que actúa como clasificador y además provocaría un valor que no tiene sentido, así que se van a reemplazar los valores perdidos usando la moda, como se muestra en el código de la Listing 14. Los valores perdidos se reemplazan por **0**, que indica que no hay intersección.

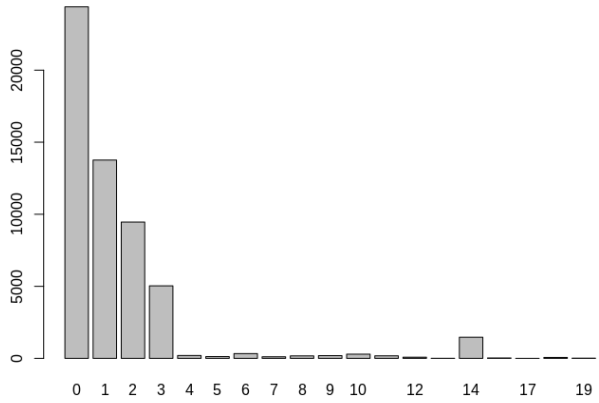

Figura 14: Gráfico de barras de la variable **REL_JCT**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$REL_JCT))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$REL_JCT))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$REL_JCT=ifelse(
  is.na(accidentes_sin_imputar_data_frame$REL_JCT),
  as.integer(names(which.max(table(accidentes_sin_imputar_data_frame$REL_JCT,useNA="no")))),
  as.integer(accidentes_sin_imputar_data_frame$REL_JCT))
```

Listing 14: Valores perdidos **REL_JCT** con la moda

3.2.5. ALIGN

En la Figura 15 se muestra un gráfico de barras con las instancias de la variable **ALIGN**. Sólo se toman dos posibles valores, así que se van a reemplazar los valores perdidos usando la moda, como se muestra en el código de la Listing 15. Los valores perdidos se reemplazan por **1**.

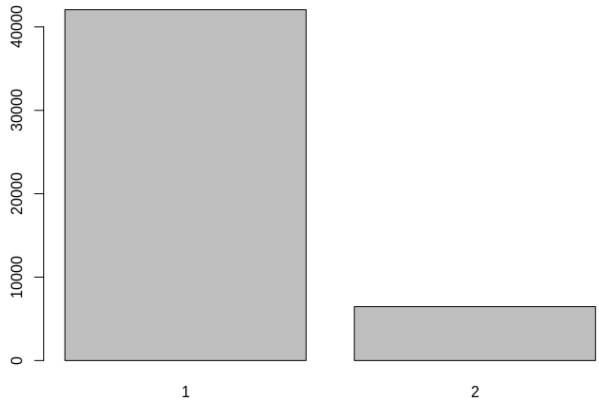

Figura 15: Gráfico de barras de la variable **ALIGN**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$ALIGN))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$ALIGN))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$ALIGN=ifelse(
  is.na(accidentes_sin_imputar_data_frame$ALIGN),
  as.integer(names(which.max(table(accidentes_sin_imputar_data_frame$ALIGN,useNA="no")))),
  as.integer(accidentes_sin_imputar_data_frame$ALIGN))
```

Listing 15: Valores perdidos **ALIGN** con la moda

3.2.6. SUR_COND

En la Figura 16 se muestra un gráfico de barras con las instancias de la variable **SUR_COND**, que indica las condiciones de la superficie en el accidente. Se van a reemplazar los valores perdidos usando la moda, como se muestra en el código de la Listing 16. Los valores perdidos se reemplazan por **1**, que indica que la superficie estaba seca.

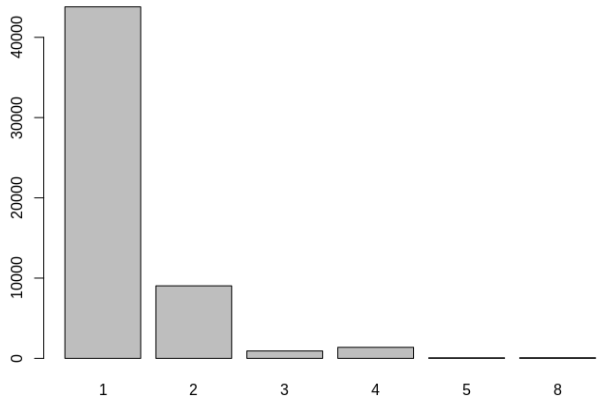

Figura 16: Gráfico de barras de la variable **SUR_COND**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$SUR_COND))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$SUR_COND))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$SUR_COND=ifelse(
  is.na(accidentes_sin_imputar_data_frame$SUR_COND),
  as.integer(names(which.max(table(accidentes_sin_imputar_data_frame$SUR_COND,useNA="no")))),
  as.integer(accidentes_sin_imputar_data_frame$SUR_COND))
```

Listing 16: Valores perdidos **SUR_COND** con la moda

3.2.7. TRAF_CON

En la Figura 17 se muestra un gráfico de barras con las instancias de la variable **TRAF_CON**, que indica la señal de tráfico. Se van a reemplazar los valores perdidos usando la moda, como se muestra en el código de la Listing 17. Los valores perdidos se reemplazan por **0**, que indica que no hay señales de tráfico.

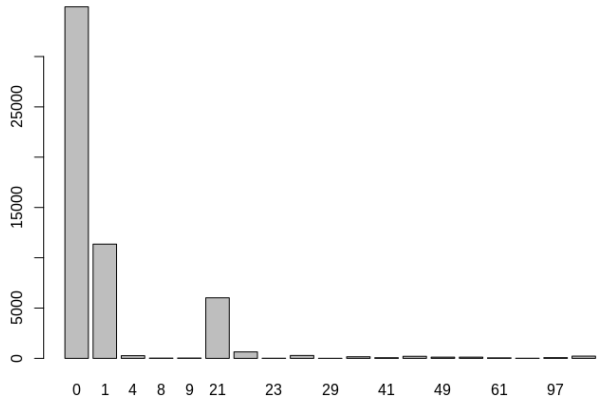

Figura 17: Gráfico de barras de la variable **TRAF_CON**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$TRAF_CON))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$TRAF_CON))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$TRAF_CON=ifelse(
  is.na(accidentes_sin_imputar_data_frame$TRAF_CON),
  as.integer(names(which.max(table(accidentes_sin_imputar_data_frame$TRAF_CON,useNA="no")))),
  as.integer(accidentes_sin_imputar_data_frame$TRAF_CON))
```

Listing 17: Valores perdidos **TRAF_CON** con la moda

3.2.8. SPD_LIM

En la Figura 18 se muestra un gráfico de barras con las instancias de la variable **SPD_LIM**, que indica la velocidad máxima permitida en la carretera. Se van a reemplazar los valores perdidos usando la media, como se muestra en el código de la Listing 18. En esta variable sí tiene sentido usar la media.

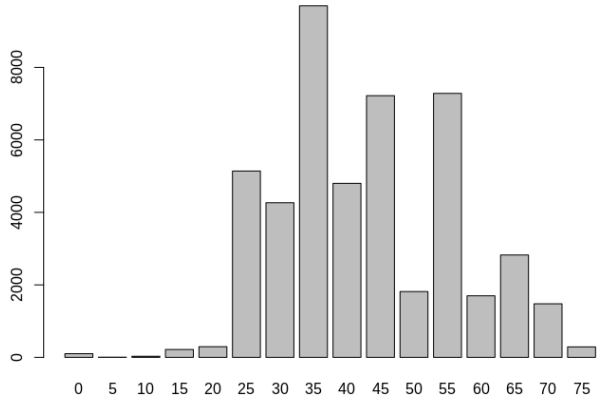

Figura 18: Gráfico de barras de la variable **SPD_LIM**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$SPD_LIM))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$SPD_LIM))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$SPD_LIM=ifelse(
  is.na(accidentes_sin_imputar_data_frame$SPD_LIM),
  as.integer(mean(accidentes_sin_imputar_data_frame$SPD_LIM,na.rm=T)),
  as.integer(accidentes_sin_imputar_data_frame$SPD_LIM))
```

Listing 18: Valores perdidos **SPD_LIM** con la media

3.2.9. LGHT_CON

En la Figura 19 se muestra un gráfico de barras con las instancias de la variable **LGHT_CON**, que indica la condición de luz. Se van a reemplazar los valores perdidos usando la moda, como se muestra en el código de la Listing 19. Los valores perdidos se reemplazan por **1**, que indica que había luz del día.

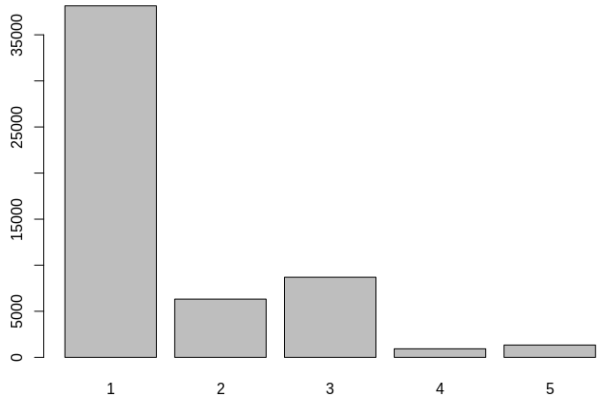

Figura 19: Gráfico de barras de la variable **LGHT_CON**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$LGHT_CON))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$LGHT_CON))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$LGHT_CON=ifelse(
  is.na(accidentes_sin_imputar_data_frame$LGHT_CON),
  as.integer(names(which.max(table(accidentes_sin_imputar_data_frame$LGHT_CON,useNA="no")))),
  as.integer(accidentes_sin_imputar_data_frame$LGHT_CON))
```

Listing 19: Valores perdidos **LGHT_CON** con la media

3.2.10. WEATHER

En la Figura 20 se muestra un gráfico de barras con las instancias de la variable **WEATHER**, que indica las condiciones meteorológicas. Se van a reemplazar los valores perdidos usando la moda, como se muestra en el código de la Listing 20. Los valores perdidos se reemplazan por **1**, que indica que las condiciones meteorológicas eran normales.

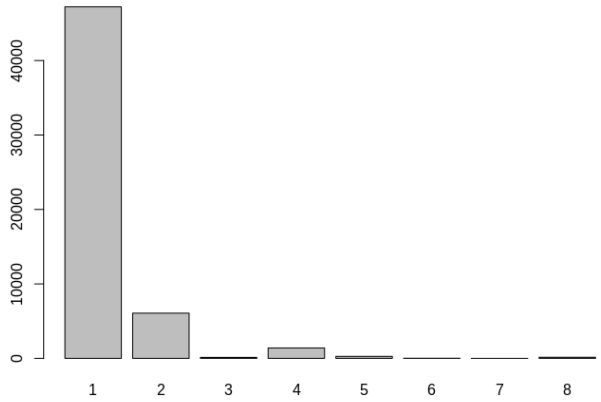

Figura 20: Gráfico de barras de la variable **WEATHER**

```
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$WEATHER))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$WEATHER))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$WEATHER=ifelse(
  is.na(accidentes_sin_imputar_data_frame$WEATHER),
  as.integer(names(which.max(table(accidentes_sin_imputar_data_frame$WEATHER,useNA="no")))),
  as.integer(accidentes_sin_imputar_data_frame$WEATHER))
```

Listing 20: Valores perdidos **WEATHER** con la moda

3.2.11. ALCOHOL

En la Figura 21 se muestra un gráfico de barras con las instancias de la variable **ALCOHOL**, que indica si la persona había bebido alcohol. Se van a reemplazar los valores perdidos usando la moda, como se muestra en el código de la Listing 21, ya que la media no tendría sentido. Los valores perdidos se reemplazan por **2**, que indica que la persona no había bebido alcohol.

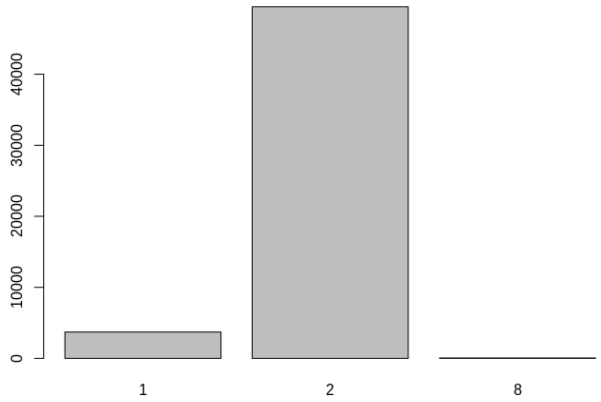

Figura 21: Gráfico de barras de la variable **ALCOHOL**

```
# ALCOHOL
# compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame$ALCOHOL))
# gráfico de barras
barplot(table(accidentes_sin_imputar_data_frame$ALCOHOL))
# se reemplazan los valores perdidos
accidentes_sin_imputar_data_frame$ALCOHOL=ifelse(
  is.na(accidentes_sin_imputar_data_frame$ALCOHOL),
  as.integer(names(which.max(table(accidentes_sin_imputar_data_frame$ALCOHOL,useNA="no")))),
  as.integer(accidentes_sin_imputar_data_frame$ALCOHOL))
```

Listing 21: Valores perdidos **ALCOHOL** con la moda

3.2.12. Clasificación

Una vez se han reemplazado los valores perdidos, se han discretizado los valores como se hizo en la Sección 2.3 y se ha vuelto a aplicar el algoritmo de clasificación C5.0.

En la Figura 22 se puede ver la salida del modelo creado. En la Figura 23 se puede ver una tabla con los resultados. En las diagonales aparecen los aciertos de cada clase.

```
Evaluation on training data (44604 cases):

          Decision Tree
        ----------------
        Size      Errors

         290 16196(36.3%)   <<


         (a)   (b)   (c)    <-classified as
        ----  ----  ----
           8   304   106    (a): class FATALITIES
           4 12672  8760    (b): class INJURY_CRASH
           2  7020 15728    (c): class PRPTYDMG_CRASH
```

Figura 22: Salida de **summary(model)**

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	4	4
INJURY_CRASH	82	2960	1920
PRPTYDMG_CRASH	30	2452	3733

Figura 23: Tabla de predicciones con el conjunto de test

3.3. Eliminar filas que tienen valores perdidos

En este apartado se van a eliminar aquellas filas que tengan algún campo con una variable perdida. En la Listing 22 se muestra el código creado para realizar esta operación.

Una vez se han eliminado las filas con valores perdidos, se han discretizado los valores como se hizo en la Sección 2.3 y se ha vuelto a aplicar el algoritmo de clasificación C5.0.

En la Figura 24 se puede ver la salida del modelo creado. En la Figura 25 se puede ver una tabla con los resultados. En las diagonales aparecen los aciertos de cada clase.

Como se puede ver, el número de instancias en el conjunto de entrenamiento baja a 22.592, y es que la variable **SPD_LIM** contiene muchos valores perdidos.

```r
# Compruebo que hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame))
# Elimino filas con valores perdidos
accidentes_sin_imputar_data_frame = na.omit(accidentes_sin_imputar_data_frame)
# Compruebo que ya no hay valores perdidos
any(is.na(accidentes_sin_imputar_data_frame))
```

Listing 22: Eliminar filas con valores perdidos

```
Evaluation on training data (22592 cases):

           Decision Tree
         ----------------
         Size      Errors

          194 8155(36.1%)   <<


          (a)   (b)   (c)    <-classified as
         ----  ----  ----
                189    58    (a): class FATALITIES
               7144  4008    (b): class INJURY_CRASH
               3900  7293    (c): class PRPTYDMG_CRASH
```

Figura 24: Salida de **summary(model)**

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	1	0
INJURY_CRASH	56	1695	1237
PRPTYDMG_CRASH	12	1030	1591

Figura 25: Tabla de predicciones con el conjunto test

3.4. Eliminar variables que tienen valores perdidos

En este apartado se van a eliminar aquellas columnas que tengan variables perdidas. En la Listing 23 se muestra el código creado para realizar esta operación.

```
accidentes_sin_imputar_data_frame =
accidentes_sin_imputar_data_frame[ , colSums(is.na(accidentes_sin_imputar_data_frame)) == 0]
```

Listing 23: Eliminar columnas con valores perdidos

Una vez se han eliminado las columnas con valores perdidos, se han discretizado los valores como se hizo en la Sección 2.3 y se ha vuelto a aplicar el algoritmo de clasificación C5.0.

En la Figura 26 se puede ver la salida del modelo creado. En la Figura 27 se puede ver una tabla con los resultados. En las diagonales aparecen los aciertos de cada clase.

Inicialmente se tenían 25 columnas y, tras haber eliminado las columnas con valores perdidos, se ha reducio a 14 columnas.

```
Evaluation on training data (44775 cases):

          Decision Tree
        ----------------
        Size      Errors

         172 17451(39.0%)   <<


         (a)   (b)   (c)    <-classified as
        ----  ----  ----
               236   184    (a): class FATALITIES
             10972 10564    (b): class INJURY_CRASH
              6467 16352    (c): class PRPTYDMG_CRASH
```

Figura 26: Salida de **summary(model)**

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	0	0
INJURY_CRASH	61	2587	1901
PRPTYDMG_CRASH	46	2709	3832

Figura 27: Tabla de predicciones con el conjunto test

4. Selección de características

En esta sección se van a prescindir de algunas características y se va a volver a ejecutar el algoritmo de clasificación C5.0.

4.1. Atributos usados en la clasificación

Se va a aplicar el algoritmo de clasificación sobre el conjunto de datos completo que contiene las variables imputadas (variables que terminan en _I). Este conjunto está discretizado tal y como se hizo en la Sección 2.3. Voy a comprobar qué variables se utilizan para clasificar y voy a descartar aquellas que no se utilicen.

En la Figura 28 se muestran los atributos que se han usado para la clasificación.

```
Attribute usage:

100.00% PED_ACC
 95.17% REL_RWY
 91.70% NON_INVL
 91.59% MANCOL_I
 87.65% TRAF_WAY
 82.38% ALCHL_I
 79.01% REGION
 68.70% SURCON_I
 52.98% RELJCT_I
 48.38% NUM_LAN
 47.03% VEH_INVL
 41.11% LAND_USE
 40.06% SPDLIM_H
 29.96% ALIGN_I
 27.87% TRFCON_I
 27.87% PROFIL_I
 22.80% HOUR_I
 19.91% WRK_ZONE
 18.24% INT_HWY
 10.54% WEATHR_I
  9.00% WKDY_I
  6.92% MONTH
  3.36% LGTCON_I
```

Figura 28: Atributos usados para la clasificación

4.2. Selección de características usando correlaciones

Para la selección de características se ha realizado una matriz de correlación para identificar variables que están estrechamente relacionadas entre sí. El código que he usado se muestra en la

Listing 24.

```r
# Se quita la variable ACCIDENTE que es la variable de clasificación
cor(accidentes_imputados_disc[-25])
# Visualizar la matriz de correlación
library(corrplot)
corrplot(cor(accidentes_imputados_disc[-25]), method="number", is.corr=FALSE
```

Listing 24: Matriz de correlación

En la Figura 29 se muestra la matriz de correlación del dataset con las variables imputadas (las que acaban en **_I**).

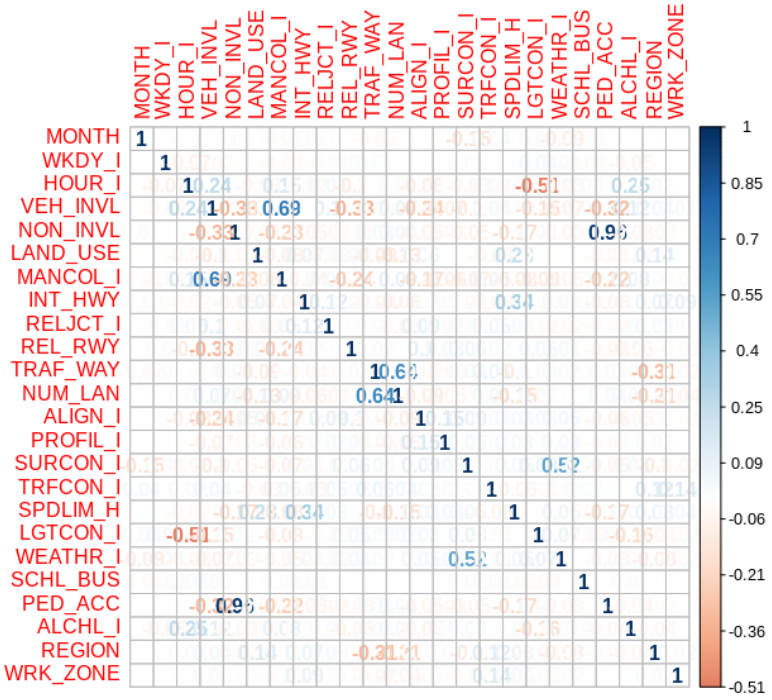

Figura 29: Matriz de correlación

Las variables más cercanas a 1 y -1 tienen una fuerte correlación, y eso significa que se puede prescindir de alguna de las dos variables. Variables con fuerte correlación:

- **PED_ACC** y **NON_INVL**

- **MANCOL_I** y **VEH_INLV**
- **LGTCOND_I** y **HOUR_I**

De este modo, voy a eliminar las variables **PED_ACC**, **MANCOL_I** y **LGTCOND_I**. En el código de la Listing 25 se muestra el código que realiza esto y su posterior clasificación.

```
# Se quita la variable ACCIDENTE que es la variable de clasificación
cor(accidentes_imputados_disc[-25])
# Visualizar la matriz de correlación
library(corrplot)
corrplot(cor(accidentes_imputados_disc[-25]), method="number", is.corr=FALSE)

# Se crea el modelo
# En el modelo
train_acc_imp_disc <- train_acc_imp_disc[c(-7,-18,-21)]
test_acc_imp_disc <- test_acc_imp_disc[c(-7,-18,-21)]
model <- C5.0(train_acc_imp_disc[-22],train_acc_imp_disc$ACCIDENTE)
summary(model)
predictions <- predict(model, test_acc_imp_disc, type="class")

# Los resultados en la diagonal principal
# muestran los aciertos del modelo solamente
table(predictions, test_acc_imp_disc$ACCIDENTE)
```

Listing 25: Eliminación de variables y clasificación

En la Figura 30 se puede ver la salida del modelo creado. En la Figura 31 se puede ver una tabla con los resultados. En las diagonales aparecen los aciertos de cada clase.

El porcentaje de error es de **36.8 %**, algo más bajo que en los anteriores preprocesamientos. En la Figura 32 se pueden ver el uso de las variables en esta nueva clasificación. En previas clasificaciones, el atributo **PED_ACC** tenía un 100 % de uso. Al haber eliminado esta variable por su correlación con **NON_INVL**, esta variable pasa a ser la variable con un 100 % junto con **REL_RWY**.

```
Evaluation on training data (44827 cases):

           Decision Tree
         ----------------
         Size      Errors

          275 16474(36.8%)   <<


          (a)   (b)   (c)    <-classified as
         ----  ----  ----
            4   288   132    (a): class FATALITIES
              12509  9011    (b): class INJURY_CRASH
            1  7042 15840    (c): class PRPTYDMG_CRASH
```

Figura 30: Salida de **summary(model)**

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	0	0
INJURY_CRASH	80	2919	1979
PRPTYDMG_CRASH	27	2458	3674

Figura 31: Tabla de predicciones con el conjunto test

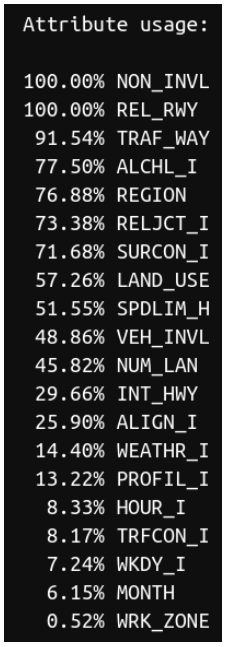

Figura 32: Porcentaje de atributos usados

5. Selección de instancias

En esta sección se van a aplicar técnicas de muestreo para el preprocesamiento de los datos.

5.1. Muestreo aleatorio sin reemplazo

Primero se ha hecho una clasificación seleccionando instancias mediante un muestreo aleatorio sin reemplazo. Exactamente se han escogido 5.000 instancias. El código se puede ver en la Listing 26.

```
accidentes_imputados_disc <- accidentes_imputados_data_frame
accidentes_imputados_disc$ACCIDENTE<-as.factor(accidentes_imputados_disc$ACCIDENTE)
# muestra sin reemplazo
ind_sin_reemplazo<- sample(1:nrow(accidentes_imputados_disc),size=5000,replace=FALSE)
muestra_sin_reemplazo=accidentes_imputados_disc[ind_sin_reemplazo,]

# CLASIFICACIÓN SIN REEMPLAZO
# Separo el dataset en conjunto de entrenamiento y test (80% y 20%)
ind=sample(2,nrow(muestra_sin_reemplazo),replace=TRUE,prob=c(0.8,0.2))
train_sin_reemplazo=muestra_sin_reemplazo[ind==1,]
test_sin_reemplazo=muestra_sin_reemplazo[ind==2,]

# Se crea el modelo
model <- C5.0(train_sin_reemplazo[-25],train_sin_reemplazo$ACCIDENTE)
summary(model)
predictions <- predict(model, test_sin_reemplazo, type="class")

# Los resultados en la diagonal principal
# muestran los aciertos del modelo solamente
table(predictions, test_sin_reemplazo$ACCIDENTE)
```

Listing 26: Muestreo simple sin reemplazo

En las Figuras 33 y 34 se muestra la salida del modelo y la matriz de confusión, respectivamente. El error que se produce es de **35.2 %**, que es el mejor error que hemos obtenido hasta ahora.

```
Evaluation on training data (4033 cases):

           Decision Tree
         ----------------
         Size      Errors

           39 1420(35.2%)   <<


          (a)   (b)   (c)    <-classified as
         ----  ----  ----
                 23    14    (a): class FATALITIES
               1110   836    (b): class INJURY_CRASH
                547  1503    (c): class PRPTYDMG_CRASH
```

Figura 33: Salida de **summary(model)**

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	0	0
INJURY_CRASH	4	253	143
PRPTYDMG_CRASH	4	233	330

Figura 34: Matriz de confusión

5.2. Muestreo aleatorio con reemplazo

En esta sección se ha hecho lo mismo que en la sección anterior pero con reemplazo. Se han escogido 5.000 instancias igualmente. El código es similar al anterior y se puede ver en la Listing 27.

```
accidentes_imputados_disc <- accidentes_imputados_data_frame
accidentes_imputados_disc$ACCIDENTE<-as.factor(accidentes_imputados_disc$ACCIDENTE)
# muestra con reemplazo
ind_con_reemplazo<- sample(1:nrow(accidentes_imputados_disc),size=5000,replace=FALSE)
muestra_con_reemplazo=accidentes_imputados_disc[ind_con_reemplazo,]

# CLASIFICACIÓN CON REEMPLAZO
# Separo el dataset en conjunto de entrenamiento y test (80% y 20%)
ind=sample(2,nrow(muestra_con_reemplazo),replace=TRUE,prob=c(0.8,0.2))
train_con_reemplazo=muestra_con_reemplazo[ind==1,]
test_con_reemplazo=muestra_con_reemplazo[ind==2,]

# Se crea el modelo
model <- C5.0(train_con_reemplazo[-25],train_con_reemplazo$ACCIDENTE)
summary(model)
predictions <- predict(model, test_con_reemplazo, type="class")

# Los resultados en la diagonal principal
# muestran los aciertos del modelo solamente
table(predictions, test_con_reemplazo$ACCIDENTE)
```

Listing 27: Muestreo simple con reemplazo

En las Figuras 35 y 36 se muestra la salida del modelo y la matriz de confusión, respectivamente. El error que se produce es de **39.1 %**, que es peor que el que se ha obtenido con la muestra aleatoria sin reemplazo.

```
Evaluation on training data (3998 cases):

           Decision Tree
         ----------------
         Size      Errors

           19 1565(39.1%)   <<


          (a)   (b)   (c)    <-classified as
         ----  ----  ----
                23    14    (a): class FATALITIES
              1063   881    (b): class INJURY_CRASH
               647  1370    (c): class PRPTYDMG_CRASH
```

Figura 35: Salida de **summary(model)**

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	0	0
INJURY_CRASH	10	247	161
PRPTYDMG_CRASH	4	228	352

Figura 36: Matriz de confusión

5.3. Muestreo aleatorio sin reemplazo y con discretización

Como se ha obtenido mejor resultado con la muestra aleatoria sin reemplazo, he decidido aplicarle también la discretización que hice en la Sección 2.3.

En las Figuras 37 y 38 se muestra la salida del modelo y la matriz de confusión, respectivamente. El error es un **0.5 % peor** que el que se obtuvo cuando los datos no estaban discretizados en la Sección 5.1. Esto puede deberse a que se ha aplicado sobre las variables que ya estaban imputadas (_I), y puede ser que estas variables estén erróneas. Además, la variable **SPDLIM_I** es la más afectada ya que originalmente es la que más datos valores perdidos posee. Aún así, el porcentaje de error es muy similar.

```
Evaluation on training data (3977 cases):

          Decision Tree
        ----------------
        Size      Errors

          49 1418(35.7%)   <<


         (a)   (b)   (c)    <-classified as
        ----  ----  ----
           2    25    12    (a): class FATALITIES
        1184   714          (b): class INJURY_CRASH
         667  1373          (c): class PRPTYDMG_CRASH
```

Figura 37: Salida de **summary(model)**

predictions	FATALITIES	INJURY_CRASH	PRPTYDMG_CRASH
FATALITIES	0	0	0
INJURY_CRASH	7	279	221
PRPTYDMG_CRASH	3	216	297

Figura 38: Matriz de confusión

6. Conclusiones

En la Tabla 1 se muestra una comparación de los resultados obtenidos en las distintas secciones de este documento. En la columna **conjunto usado** se indica si se ha trabajado con los datos imputados (todas las variables con _I) o con los datos sin imputar (las variables con valores perdidos). También se han coloreado aquellas celdas que tienen en común la técnica de preprocesamiento usada, correspondiente a las 4 secciones que he redactado. Estas son mis conclusiones:

- En cuanto a la **discretización** (en amarillo), la discretización que peor resultado da es el algoritmo top-down CAIM. Esto puede ser debido a que el algoritmo se aplica a todas las variables, y hay variables en las que no es necesario una discretización. Sin embargo, mi discretización propuesta en la Sección 2.3 obtiene mejores resultados. Aún así, se obtiene mejores resultados cuando no se ha hecho ninguna discretización, y esto puede deberse a que se necesitarán aplicar más técnicas o bien que se pueden proponer discretizaciones diferentes a la mía y que pueden ser mejores.
- En cuanto a los **valores perdidos** (en verde), el peor resultado se obtiene cuando se eliminan las variables con valores perdidos. En esta técnica se redujeron a 14 las variables, casi la mitad, y algunas de estas variables eran bastante importantes (como la hora).
- En cuanto a la **selección de características** (en morado), al eliminar las variables de acuerdo a las correlaciones se ha obtenido un mejor resultado que eliminando las variables con valores perdidos. Al usar una matriz de correlación, se eliminan variables que están estrechamente relacionaas y que, por tanto, eliminar algunas de ellas no influye demasiado.
- En cuanto a la **selección de instancias** (en naranja), el mejor resultado se ha obtenido con un muestreo aleatorio sin reemplazo. Realmente la diferencia entre ambos es que en el muestreo con reemplazo, cuando se escoge una variable puede ser seleccionado una vez más, pero estas técnicas son aleatorias. Aún así, al discretizar la muestra aleatoria sin reemplazo se ha obtenido un 0.5 % de error más que cuando no se ha discretizado, y esto puede deberse a que aún se deberían aplicar más técnicas de preprocesamiento además de la discretización o bien proponer otra forma de discretización.

Técnica de preprocesamiento	Conjunto usado	Porcentaje de error
Sin ningún tratamiento de datos	Con datos imputados	34.9
Discretización top-down CAIM en todas las variables	Con datos imputados	41.6
Discretización propia	Con datos imputados	36.3
Sin ningún tratamiento de datos	Sin datos imputados	35.1
Valores perdidos con la media o moda y discretización propia	Sin datos imputados	36.3
Eliminar filas con valores perdidos y discretización propia	Sin datos imputados	36.1
Eliminar variables con valores perdidos y discretización propia	Sin datos imputados	39.0
Selección de características usando correlaciones	Con datos imputados	36.8
Muestreo aleatorio sin reemplazo	Con datos imputados	35.2
Muestreo aleatorio con reemplazo	Con datos imputados	39.1
Muestreo aleatorio sin reemplazo y discretización propia	Con datos imputados	35.7

Tabla 1: Comparación de resultados