UNIVERSIDAD DE GRANADA

Preprocesamiento de datos

Carlos Morales Aguilera

20/11/2020

Lectura de datos

El primer paso en el problema es leer correctamente los datos, para ello se utiliza la función `read_excel` de la librería **readxl**, a continuación se transforma el conjunto de datos obtenido en una estructura del tipo **dataframe** propia de *R*.

```
f <- "C:\\Users\\power\\Desktop\\TID\\accidentes.xls"
my_description <- read_excel(f, sheet = "variables")

bd_accidentes <- as.data.frame(read_excel(f, sheet = "datos"))
```

Eliminar valores desconocidos a partir de la información del conjunto de datos

A continuación se procedería a eliminar los valores perdidos o en este caso, desconocidos, para ello hay que leer la información de las distintas variables del conjunto y eliminar las instancias que posean algún valor desconocido en alguna de ellas. Toda esta información se obtiene de la pestaña *variables* del excel de **accidentes.xls**.

```
bd_accidentes$MANCOL_I <- replace(bd_accidentes$MANCOL_I, bd_accidentes$MANCOL_I==0, NA)
bd_accidentes$RELJCT_I <- replace(bd_accidentes$RELJCT_I, bd_accidentes$RELJCT_I==99, NA)
bd_accidentes$PROFIL_I <- replace(bd_accidentes$PROFIL_I, bd_accidentes$PROFIL_I==9, NA)
bd_accidentes$SURCON_I <- replace(bd_accidentes$SURCON_I, bd_accidentes$SURCON_I==9, NA)
bd_accidentes$TRFCON_I <- replace(bd_accidentes$TRFCON_I, bd_accidentes$TRFCON_I==99, NA)
bd_accidentes$SPDLIM_H <- replace(bd_accidentes$SPDLIM_H, bd_accidentes$SPDLIM_H==99, NA)
bd_accidentes$LGTCON_I <- replace(bd_accidentes$LGTCON_I, bd_accidentes$LGTCON_I==9, NA)
bd_accidentes$WEATHR_I <- replace(bd_accidentes$WEATHR_I, bd_accidentes$WEATHR_I==9, NA)
bd_accidentes$ALCHL_I <- replace(bd_accidentes$ALCHL_I, bd_accidentes$ALCHL_I>=8, NA)

bd_accidentes <- na.omit(bd_accidentes)
```

Composición de variable de clasificación

En este caso se permitía la libre elección de la combinación de las variables *FATALITIES*, *INJURY_CRASH* (daños personales) y *PRPTYDMG_CRASH* (daños materiales), por lo que se ha decidido crear una variable **SEVERITY** (gravedad) que clasifica en dos tipos: Graves (1) y Leves (0).

- Graves: Todo aquel accidente que involucre algún tipo de daño físico o personal.
- Leves: Accidentes que únicamente conllevan daño material.

```
bd_accidentes <- bd_accidentes %>%
  mutate(SEVERITY = ifelse(FATALITIES == 1 | INJURY_CRASH == 1, 1, 0))

bd_accidentes[,"FATALITIES"] <- NULL
bd_accidentes[,"INJURY_CRASH"] <- NULL
bd_accidentes[,"PRPTYDMG_CRASH"] <- NULL
```

Creación de grupos de entrenamiento y test

Para comprobar que el modelo de árbol de decisión se realiza una validación cruzada, por lo que se necesitan dos conjuntos: *train* y *test*. El conjunto de entrenamiento comprenderá el 80% de los datos tomados para

entrenar el árbol de decisión que clasifique el problema, y el conjunto de testeo se empleará para validad el modelo de ajuste obtenido.

```
create_train_test <- function(data, size = 0.8, train = TRUE) {
  n_row = nrow(data)
  total_row = size * n_row
  train_sample <- 1: total_row
  if (train == TRUE) {
    return (data[train_sample, ])
  } else {
    return (data[-train_sample, ])
  }
}
```

Entrenamiento y validación

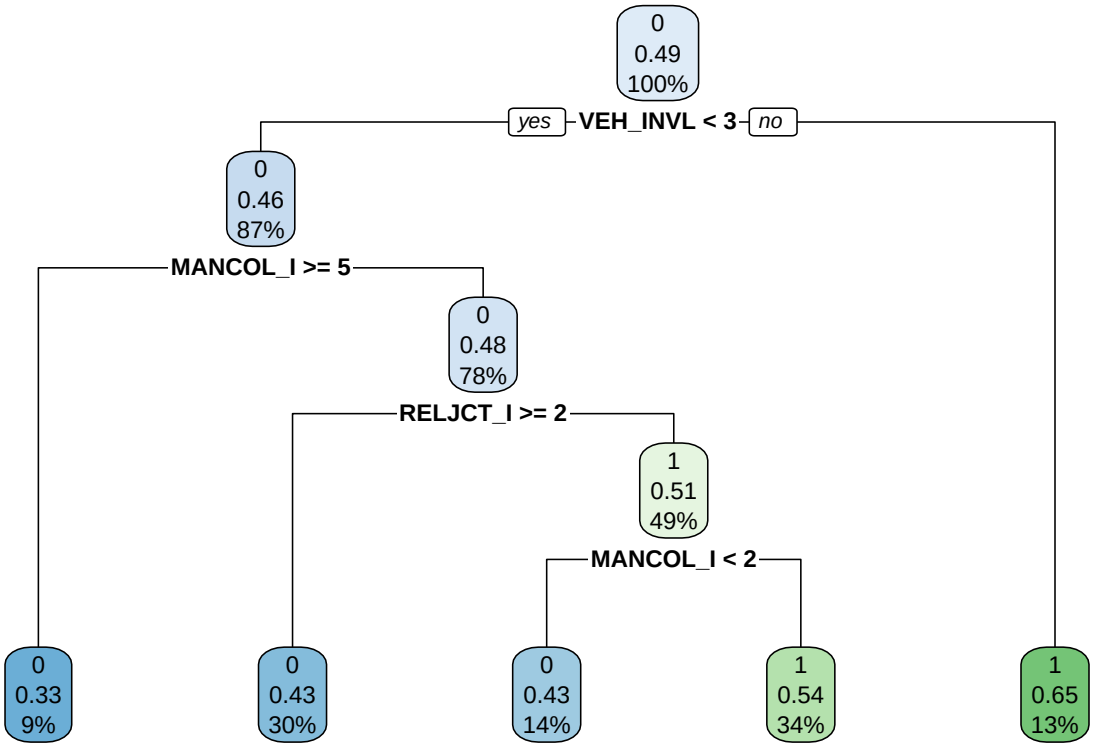
Para entrenar se ha utilizado un modelo de clasificación consistente en un árbol de decisión binario, el cual dado un conjunto de datos, divide este por características de forma que se tome una decisión o su opuesta, para clasificar. Se ha definido una función que solicita un conjunto de entrenamiento y realiza la clasificación en un árbol de decisión binario. Para ello se ha utilizado la librería rpart, concretamente haciendo uso tanto de la función rpart como de su representación mediante rpart.plot, perteneciente a la librería rpart.plot.

Por otro lado, se ha definido también una función de evaluación del modelo, la cual hace uso de la función predict, y realizando posteriormente una validación cruzada entre las soluciones estimadas en el conjunto de test y las soluciones reales de ese conjunto de test.

```
train_model <- function(data){
  data_train <- create_train_test(data, 0.8, train = TRUE)

  fit <- rpart(SEVERITY~., data = data_train, method = 'class')
  rpart.plot(fit)
  return (fit)
}

test_model <- function(data, model){
  data_test <- create_train_test(data, 0.8, train = FALSE)

  predict_unseen <-predict(model, data_test, type='class')
  table_mat <- table(data_test$SEVERITY, predict_unseen)
  accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)
  print(paste('Accuracy for test', accuracy_Test))
}
```

A continuación se muestra la primera clasificación del modelo, en el que únicamente se han eliminado los valores desconocidos.

```
model <- train_model(bd_accidentes)
```



Tras ajustar el árbol de decisión y su modelo pertinente, se evalúa el modelo de ajuste con el conjunto de test y obtenemos el *accuracy* (o porcentaje de acierto).

```
test_model(bd_accidentes, model)
```

```
## [1] "Accuracy for test 0.594474989942336"
```

Discretización de variables

Una de las técnicas principales de preprocesamiento de datos consisten en la discretización de valores continuos, o discretización de valores categóricos como agrupación de estas categorías en otras que abarquen más categoría en una misma. Para ello se ha utilizado la función cut, perteneciente al paquete *base* del propio *R*.

Para la realización de la discretización de las variables categóricas del problema, se han estudiado los distintos casos posibles en la información de *variables* del conjunto de datos, para ello se han tomado las siguientes decisiones:

- **WKDY_I**: Clasificación natural de días de semana
  - *Semana* (0): 1-5.
  - *Fin de semana* (1): 6-7.
- **HOUR_I**: Para esta variable se han tenido que observar el número de casos de accidentes obtenidos en cada hora, y se han agrupado según cantidad de accidentes en grupos de frecuencia de accidentes: Alta, Media, Baja.
  - *Low* (1): 24:00-6:00.

  - *Medium* (2): 6:00-15:00 y 18:00-24:00.
  - *High* (3): 15:00-18:00.

- **MANCOL_I**: Para esta variable se han visualizado todos los casos. **NO** se corresponden con la indicación en la descripción, por lo que la discretización se realiza teniendo en cuenta las variables **realmente** existentes.
  - *Damage* (1): Daños relacionados con el coche o el entorno, 1-4.
  - *Jackknife* (2): Asalto, 5.
  - *Non-collission* (3): No existe colisión, 6.

- **RELJCT_I**: Se divide en dos grupos principales.
  - *No intersection* (0): 0 y 8-10.
  - *Intersection* (1): 1-7 y 11-99.

- **PROFIL_I**: Perfil de la carretera, se reunen en dos grupos según su llanura.
  - *Level* (0): 1.
  - *Grade* (1): 2-9.

- **SURCON_I**: Se clasifica la condicicón de la superficie en dos tipos.
  - *Good* (0): 1.
  - *Bad* (1): 2-8.

- **TRFCON_I**: Control de tráfico en el que se clasifica según si existencia.
  - *No control* (0): 1.
  - *Control* (1): 2-98.

- **SPDLIM_H**: Se clasifica en alta y baja velocidad.
  - *Slow* (0): 0-55 km/h.
  - *Fast* (1): 60-98 km/h.

- **LGTCON_I**: Condición lumínica, considerando si se tiene buena visión o mala.
  - *Good* (0): 1.
  - *Bad* (1): 2-8.

- **WEATHR_I**: Se considera si existe una buena condición meteorológica o adversa.
  - *Good* (0): 1.
  - *Bad* (1): 2-8.

- **ALCHL_I**: Únicamente se considera si hay alcohol en el registro del accidente o no.
  - *Yes* (0): 1.
  - *No* (1): 2 y 8.

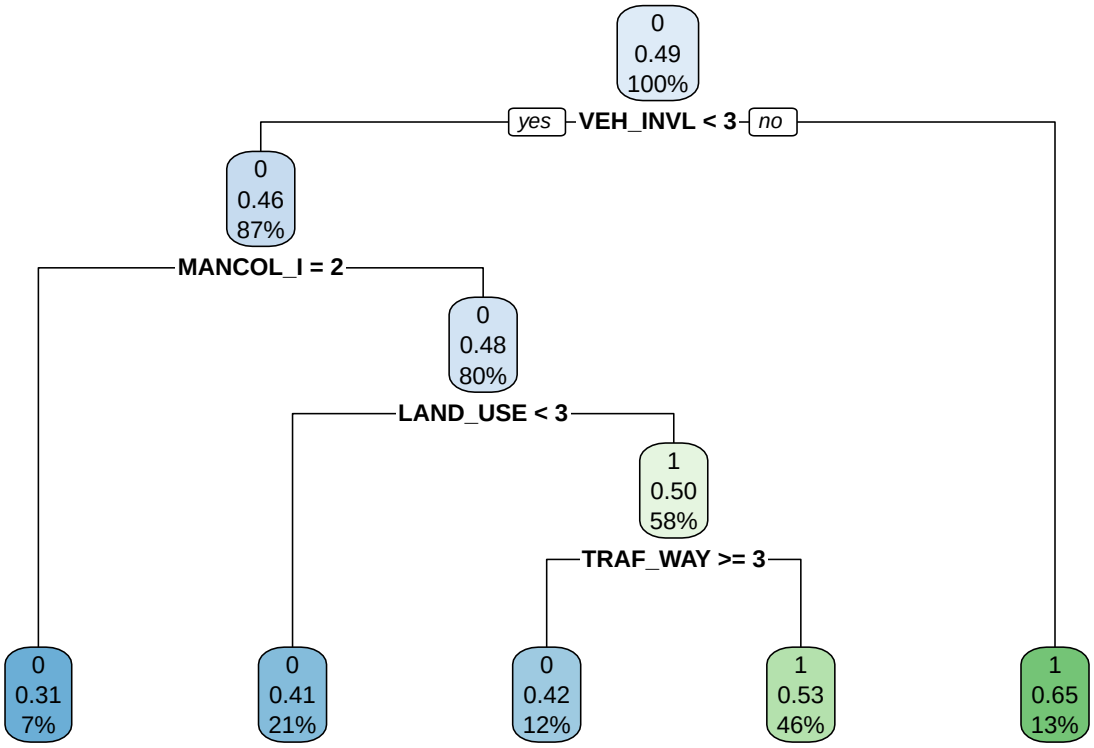
Tras realizar la discretización de estas variables, se eliminan las siguientes variables, ya que se han discretizado su versión imputada: *WEEKDAY*, *HOUR*, *REL_JCT*, *ALIGN*, *PROFILE*, *SUR_COND*, *TRAF_CON*, *SPD_LIM*, *LGHT_CON*, *WEATHER* y *ALCOHOL*.

```
# Discretize days by week-weekend (Week, Weekend)
bd_accidentes[,"WKDY_I"] <- cut(bd_accidentes[,"WKDY_I"], breaks = c(1,6,8),
                          labels = c(0, 1), right = FALSE)
# Discretize by level of hour of accident (Low, Medium, High)
bd_accidentes[,"HOUR_I"] <- cut(bd_accidentes[,"HOUR_I"], breaks = c(0,6,15,18, 24, 25),
                          labels = c(1, 2, 3, 2, 1), right = FALSE)
# Discretize manner of colision (Damage, Jackknife, Non-collision)
bd_accidentes[,"MANCOL_I"] <- cut(bd_accidentes[,"MANCOL_I"], breaks = c(1,5,6,99),
```

```
                            labels = c(1, 2, 3),
                            right = FALSE)
# Discretize relation to junction (No inters, Intersec, No inters, Intersec)
bd_accidentes[,"RELJCT_I"] <- cut(bd_accidentes[,"RELJCT_I"],
                                  breaks = c(0,1,8,11,99),
                                  labels = c(0, 1, 0, 1), right = FALSE)
# Discretize road profile (Level, Grade)
bd_accidentes[,"PROFIL_I"] <- cut(bd_accidentes[,"PROFIL_I"], breaks=c(1,2,9),
                            labels = c(0, 1), right = FALSE)
# Discretize surface condition (Good, Bad)
bd_accidentes[,"SURCON_I"] <- cut(bd_accidentes[,"SURCON_I"], breaks=c(1,2,9),
                            labels = c(0, 1), right = FALSE)
# Discretize traffic control (No control, Control)
bd_accidentes[,"TRFCON_I"] <- cut(bd_accidentes[,"TRFCON_I"], breaks=c(0,1,99),
                            labels = c(0, 1),
                            right = FALSE)
# Discretize speed limit (Slow, Fast)
bd_accidentes[,"SPDLIM_H"] <- cut(bd_accidentes[,"SPDLIM_H"], breaks=c(0,60,99),
                            labels = c(0, 1),
                            right = FALSE)
# Discretize light condition (Good, Bad)
bd_accidentes[,"LGTCON_I"] <- cut(bd_accidentes[,"LGTCON_I"], breaks=c(1,2,9),
                            labels = c(0, 1),
                            right = FALSE)
# Discretize weather condition (Good, Bad)
bd_accidentes[,"WEATHR_I"] <- cut(bd_accidentes[,"WEATHR_I"], breaks=c(1,2,9),
                            labels = c(0, 1),
                            right = FALSE)
# Discretize alcohol involved (Yes, No)
bd_accidentes[,"ALCHL_I"] <- cut(bd_accidentes[,"ALCHL_I"], breaks = c(1,2,9),
                           labels = c(0, 1),
                           right = FALSE)

bd_accidentes[,"WEEKDAY"] <- NULL
bd_accidentes[,"HOUR"] <- NULL
bd_accidentes[,"MAN_COL"] <- NULL
bd_accidentes[,"REL_JCT"] <- NULL
bd_accidentes[,"ALIGN"] <- NULL
bd_accidentes[,"PROFILE"] <- NULL
bd_accidentes[,"SUR_COND"] <- NULL
bd_accidentes[,"TRAF_CON"] <- NULL
bd_accidentes[,"SPD_LIM"] <- NULL
bd_accidentes[,"LGHT_CON"] <- NULL
bd_accidentes[,"WEATHER"] <- NULL
bd_accidentes[,"ALCOHOL"] <- NULL
```

Se entrena el modelo y se visualiza el árbol de decisión tras la discretización.

```
model <- train_model(bd_accidentes)
```



Tras ajustar el árbol de decisión y su modelo pertinente, se evalúa el modelo de ajuste con el conjunto de test y obtenemos el *accuracy* (o porcentaje de acierto).

```
test_model(bd_accidentes, model)
```

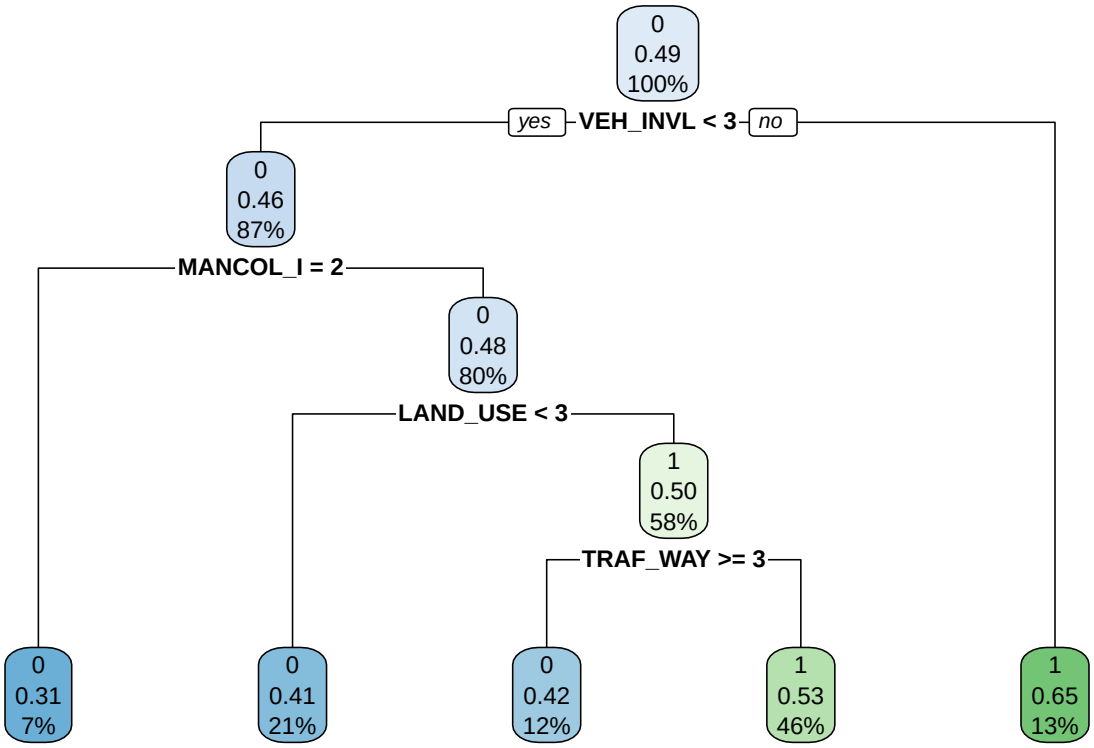
```
## [1] "Accuracy for test 0.531983371328953"
```

Tras realizar la discretización, es conveniente realizar una omisión de valores perdidos, por si algún valor no hubiera entrado en la discretización realizada y se hubiera almacenado como *NA*, por lo que eliminaremos todos los datos con información errónea.

```
bd_accidentes <- na.omit(bd_accidentes)
```

Se entrena el modelo y se visualiza el árbol de decisión tras la omisión de valores perdidos.

```
model <- train_model(bd_accidentes)
```



Tras ajustar el árbol de decisión y su modelo pertinente, se evalúa el modelo de ajuste con el conjunto de test y obtenemos el *accuracy* (o porcentaje de acierto).

```
test_model(bd_accidentes, model)
```

```
## [1] "Accuracy for test 0.531983371328953"
```

Selección de características

Otra de las técnicas que se emplean en el preprocesamiento de datos es la selección de características a emplear, en este caso, hablamos de lo que se denomina stepwise regression/backward elimination, la cual nos permite comprender que variables son realmente representativas de como se compone el conjunto de datos y cuales no deberíamos considerar de cara a su clasificación. Para ello se ha utilizado la función stepAIC, la cual nos permite hacer distintos tipos de regresiones de características, pero en nuestro caso indicamos que realice de tipo *backward*. Para ello primero entrenamos el modelo con un modelo lineal utilizando la función lm.

Se muestran a continuación de forma ordenada:

1. Modelo inicial
2. Modelo final con ajuste.
3. Características descartadas y su significancia en el conjunto de datos.

```r
model <- lm(SEVERITY~., data = bd_accidentes)
step.model <- stepAIC(model, direction = "backward", trace = FALSE)
anova <- step.model$anova
anova
```

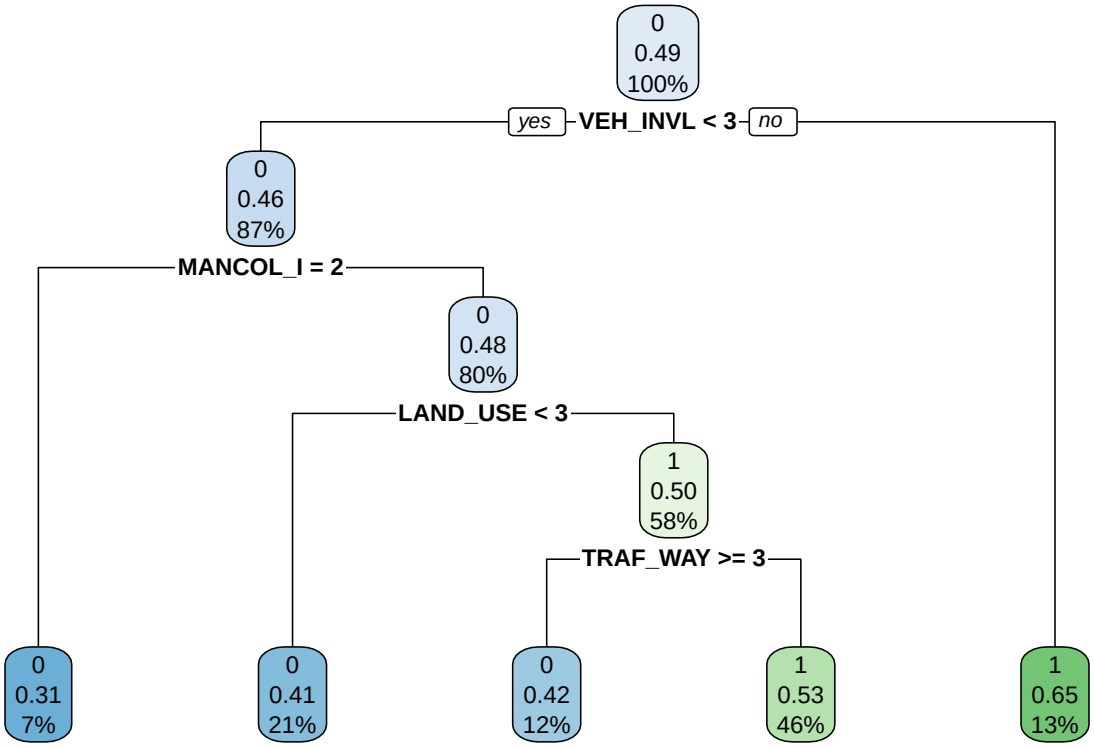
```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## SEVERITY ~ MONTH + WKDY_I + HOUR_I + VEH_INVL + NON_INVL + LAND_USE +
##     MANCOL_I + INT_HWY + RELJCT_I + REL_RWY + TRAF_WAY + NUM_LAN +
##     ALIGN_I + PROFIL_I + SURCON_I + TRFCON_I + SPDLIM_H + LGTCON_I +
##     WEATHR_I + SCHL_BUS + PED_ACC + ALCHL_I + REGION + WRK_ZONE
##
## Final Model:
## SEVERITY ~ MONTH + HOUR_I + VEH_INVL + NON_INVL + LAND_USE +
##     MANCOL_I + RELJCT_I + TRAF_WAY + SURCON_I + TRFCON_I + SPDLIM_H +
##     LGTCON_I + SCHL_BUS + PED_ACC + ALCHL_I + REGION
##
##
##         Step Df    Deviance Resid. Df Resid. Dev       AIC
## 1                                37257   8980.417 -53020.44
## 2   - WKDY_I  1 0.002894210     37258   8980.420 -53022.43
## 3  - INT_HWY  1 0.008817163     37259   8980.428 -53024.40
## 4 - WEATHR_I  1 0.010144338     37260   8980.439 -53026.35
## 5  - REL_RWY  1 0.165475127     37261   8980.604 -53027.67
## 6 - WRK_ZONE  1 0.222816649     37262   8980.827 -53028.74
## 7  - NUM_LAN  1 0.260683388     37263   8981.088 -53029.66
## 8  - ALIGN_I  1 0.297021157     37264   8981.385 -53030.43
## 9 - PROFIL_I  1 0.415150511     37265   8981.800 -53030.70
```

Se eliminan por lo tanto las variables que no son utilizadas para el problema.

```r
bd_accidentes$WKDY_I <- NULL
bd_accidentes$INT_HWY <- NULL
bd_accidentes$WEATHR_I <- NULL
bd_accidentes$REL_RWY <- NULL
bd_accidentes$WRK_ZONE <- NULL
bd_accidentes$NUM_LAN <- NULL
bd_accidentes$ALIGN_I <- NULL
bd_accidentes$PROFIL_I <- NULL
```

Se entrena el modelo y se visualiza el árbol de decisión tras la selección de características.

```
model <- train_model(bd_accidentes)
```



Tras ajustar el árbol de decisión y su modelo pertinente, se evalúa el modelo de ajuste con el conjunto de test y obtenemos el *accuracy* (o porcentaje de acierto).

```
test_model(bd_accidentes, model)
```

```
## [1] "Accuracy for test 0.531983371328953"
```

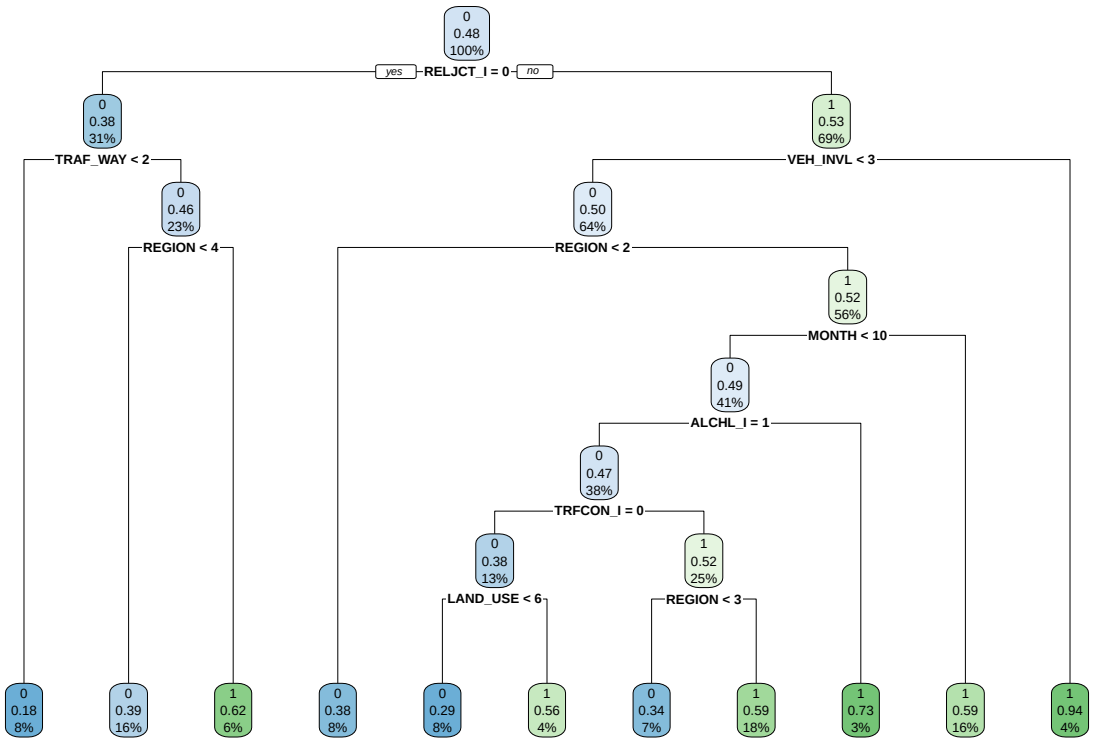
Selección de instancias

Uno de los mayores problemas que podemos encontrar a la hora de realizar una clasificación con un conjunto de datos, es que este se encuentre desbalanceado y los resultados obtenidos no sean lo suficientemente representativos, por lo que se aplican técnicas de selección o restricción de instancias, para ello se va a realizar una selección aleatoria de instancias hasta en 5 ocasiones con diferentes tamaños, con el objetivo de ver como afecta a los modelos obtenidos y por lo tanto, la clasificación.

Para realizar la selección de subconjuntos de datos se va a emplear la función sample_n del paquete tidyverse.

Primera ejecución aleatoria con un conjunto de 500 valores.

```
set.seed(1234)

bd_accidentes2 <- sample_n(bd_accidentes, size= 500)

model <- train_model(bd_accidentes2)
```

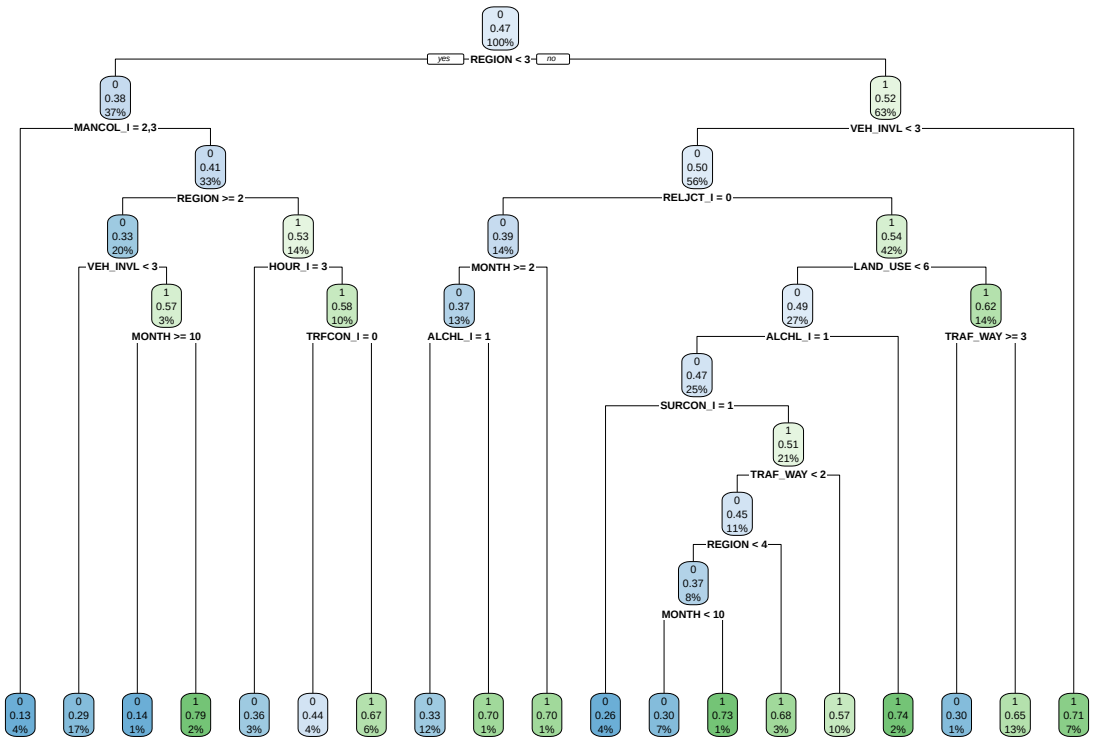



```
test_model(bd_accidentes2, model)
```

```
## [1] "Accuracy for test 0.49"
```

Segunda ejecución aleatoria con un conjunto de 1000 valores.

```
bd_accidentes2 <- sample_n(bd_accidentes, size= 1000)

model <- train_model(bd_accidentes2)
```

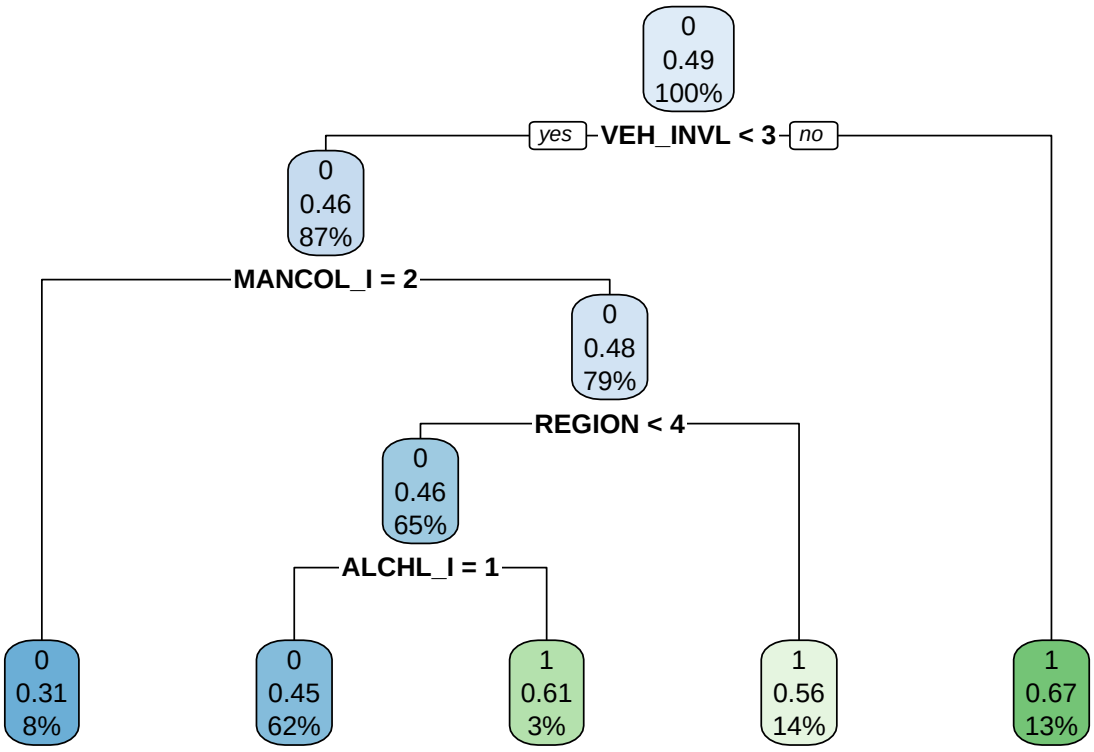



```
test_model(bd_accidentes2, model)
```

```
## [1] "Accuracy for test 0.605"
```

Tercera ejecución aleatoria con un conjunto de 5000 valores.

```
bd_accidentes2 <- sample_n(bd_accidentes, size= 5000)

model <- train_model(bd_accidentes2)
```

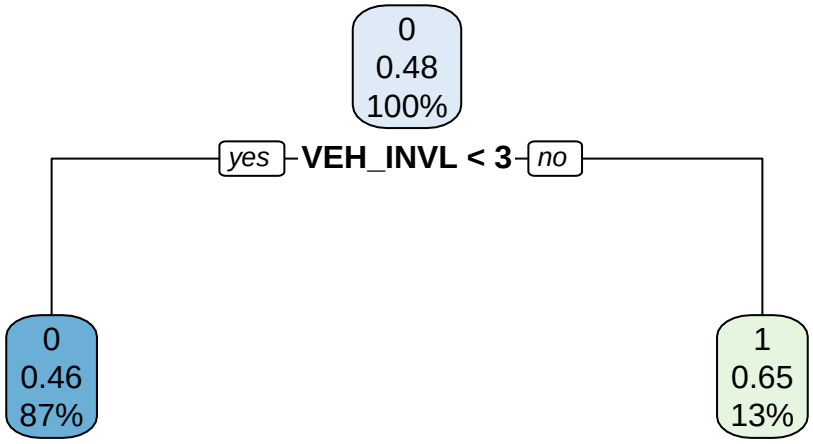



```
test_model(bd_accidentes2, model)
```

```
## [1] "Accuracy for test 0.565"
```

Cuarta ejecución aleatoria con un conjunto de 10000 valores.

```
bd_accidentes2 <- sample_n(bd_accidentes, size= 10000)

model <- train_model(bd_accidentes2)
```

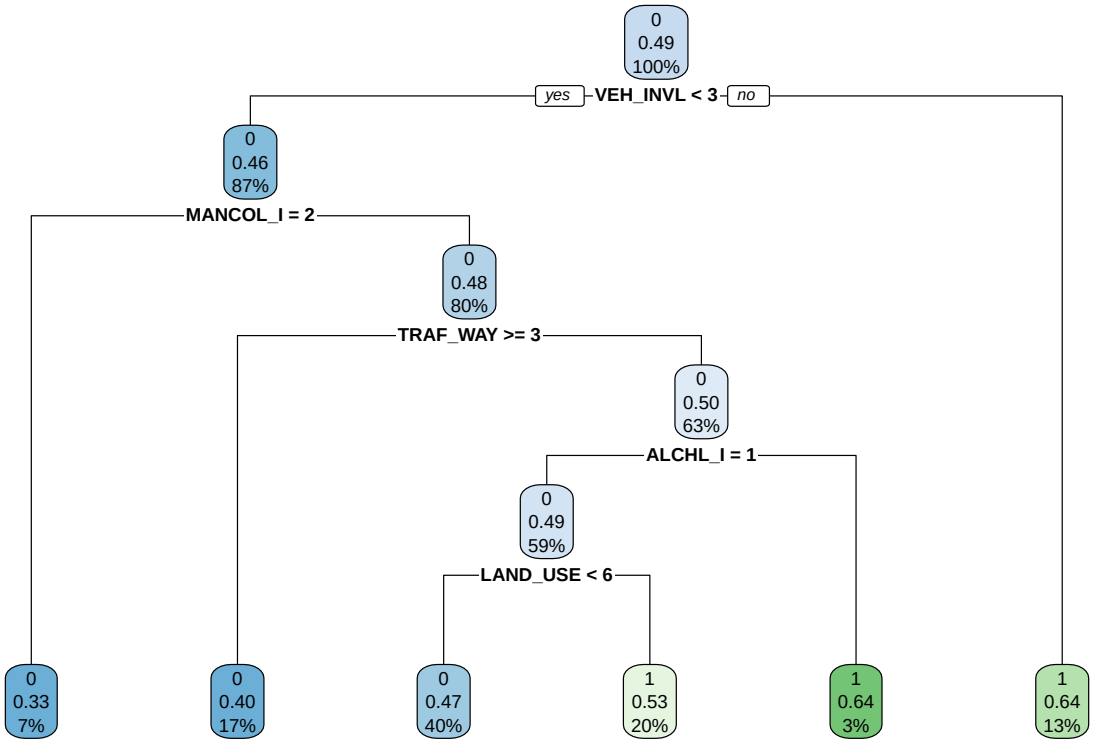



```
test_model(bd_accidentes2, model)
```

```
## [1] "Accuracy for test 0.541"
```

Quinta ejecución aleatoria con un conjunto de 20000 valores.

```
bd_accidentes2 <- sample_n(bd_accidentes, size= 20000)

model <- train_model(bd_accidentes2)
```




```
test_model(bd_accidentes2, model)
```

```
## [1] "Accuracy for test 0.5815"
```

Conclusiones

Tras realizar el preprocesamiento de datos se pueden analizar una serie de eventos que se han producido en al proceso, y una serie de conclusiones sobre el conjunto de datos ofrecido:

- El conjunto de datos está **desbalanceado**, por lo que resulta complicado seleccionar una elección de variable de clasificación, la más adecuada teniendo en cuenta el contexto y el desbalanceamiento de los datos, ha sido la realizada en la práctica, obteniendo una variable que clasifica el número de casos en dos casos con más o menos el mismo número de ocurrencias.
- El conjunto de datos tiene información muy **dispersa**, si bien se realiza una discretización inicial, la elección de las discretizaciones es un proceso complejo, que puede llegar a ser positivo o negativo. En este caso podemos ver que los datos se encuentran desbalanceados, por lo que una discretización que tenga en cuenta toda la información puede resultar negativa, quizás sería mejor discretizar entre los casos más comunes y agrupar los casos más particulares, pero es una elección que conlleva un mayor estudio del problema. En este caso, la discretización escogida, pese a ser lógica acorde al problema, no resulta efectiva, entre otros motivos por el desbalanceamiento mencionado previamente.
- La omisión de valores desconocidos resulta **claramente efectiva**, ya que se elimina información que resulta enormemente negativa de cara al aprendizaje, por no ser tampoco representativa. Además es importante la omisión de estos valores tras la discretización, ya que no solo limpia valores desconocidos

o incompletos, sino que también nos permite darnos cuenta de que por ejemplo la variable *MANCOL_I* posee una información que no se corresponde con la afirmada en la descripción del conjunto de datos. Además, valores como el 0 para esta variable, en la discretización se indicarán como valores perdidos, y por lo tanto se omitirán. Realmente la **combinación** de estas dos herramientas resulta útil para tratar con información representativa.

- La selección de características es una técnica muy importante de cara a tratar los datos, ya que existe información redundante o que realmente no aporta información alguna, complicando el problema innecesariamente. Existen numerosas herramientas que permiten realizar esta técnica, pero en este caso se ha escogido un modelo lineal que nos ha permitido ver que variables son realmente relevantes en el problema y descartarlas. Aunque evidentemente no afecta en una primera instancia en nuestro problema (ya que el árbol de decisión lógicamente va a seguir tomando la misma decisión si eliminamos variables irrelevantes) sí que es una técnica muy necesaria para otras tareas, y que llevada a cabo puede facilitar la interpretación de la información. Durante la realización de la práctica se realizaron modelos de *Random Forest* en los que se veía una mejora evidente al eliminar dichas características, ya que permitían al algoritmo acercarse más a un mejor modelo final, alejándose de modelos que utilizaran estas variables.

- La selección de instancias cobra un gran sentido en a la hora de realizar comprobaciones de clasificaciones como en nuestro problema, ya que demuestra la importancia de la significancia de los datos si se escogen pequeños grupos de ellos. Aunque existen numerosas técnicas como *CNN* o *K-nearest neighbours* entre otras, se ha utilizado una selección aleatoria de distintos conjuntos y con distintos tamaños, para ver como esto afectaba al problema de clasificación. Se puede comprobar con éxito, ya que por ejemplo en el conjunto aleatorio de 500 instancias se ha obtenido un *accuracy* del 40%, mientras que en el conjunto de 1000 instancias (completamente distinto) se ha obtenido un *accuracy* del 60% (el cual es un resultado bastante positivo para lo complejo que resulta el conjunto de datos).

Tras analizar estas técnicas y comprobar su efecto en un problema sencillo de clasificación se puede comprobar la importancia de las técnicas de preprocesamiento a la hora de realizar un problema con un conjunto de datos grande. Aunque evidentemente existen técnicas más avanzadas y que podrían llevar a mejores resultados, los resultados obtenidos son bastante positivos teniendo en cuenta lo mencionado previamente sobre el conjunto de datos.

Resultados obtenidos

Clasificación	Accuracy
Conjunto inicial	0.594474989942336
Conjunto discretizado	0.531983371328953
Conjunto sin valores perdidos	0.531983371328953
Conjunto tras selección de características	0.531983371328953
Subconjunto de 500 instancias aleatorias	0.49
Subconjunto de 1000 instancias aleatorias	0.605
Subconjunto de 5000 instancias aleatorias	0.565
Subconjunto de 10000 instancias aleatorias	0.541
Subconjunto de 20000 instancias aleatorias	0.5815