Trabajo Andrés Lara Romero De Ávila

Andrés Lara Romero De Ávila

2023-06-04

Índice

Descripción de los datos

La base de datos utilizada trata sobre diferentes accidentes en carretera. Los datos datan de 2016 y el lugar donde se produjeron los accidentes fue Reino Unido. La base de datos se encuentra en kaggle en el siguiente enlace: https://www.kaggle.com/datasets/bluehorseshoe/uk-2016-road-safety-data?select=dftRoadSafety_

Accidents_2016.csv. La base de datos como comenta el usuario que la subió pertenece al apartado de transporte del gobierno de Reino Unido. Esta página contiene gran cantidad de recursos para entrenamiento en machine learning e incluso tiene actualizados a partir de estos mismos estudios en los que explican las zonas y las condiciones en las que se producen mayor cantidad de accidentes para prevenir a los ciudadanos . La interpretación de los atributos de la tabla la he realizado gracias la explicación de los distintos valores de cada variable que vienen en kaggle, pues no he encontrado ningún glosario.

Entre las columnas encontramos:

- Fallecido: implica si hubo fallecidos

  0. No hubo fallecidos

  1. Hubo al menos un fallecido

- Numero_Vehiculos : El número de vehículos implicados en el accidente. Este número puede ir desde un solo vehículo implicado hasta un máximo de 16 vehículos.

- Numero_Afectados : Número de heridos (victimas o con lesiones) afectados por el accidente. Esta es una variable cuantitativa cuyo valor mínimo es cero afectados y el máximo es 60.

- Tipo_Via_Principal: Indica el tipo de vía donde se produjo el accidente. Sus valores pueden ser:

  1. Autovía
  2. Autovía de tipo A
  3. Carretera General del estado
  4. Carretera tipo B
  5. Carretera tipo C
  6. No clasificado

- Especificación_Vía: Indica algún detalle relevante sobre la vía principal.

  1. Rotonda
  2. Calle de un solo carril
  3. Carretera con dos carriles

  6. Carretera con un carril
  7. Carril de aceleración o deceleración

  9. Desconocido

- Límite de velocidad: Indica el límite máximo de velocidad permitido en la vía donde se produjo el accidente. Las unidades son millas por hora.

- Detalles de cruce: Indica información adicional en el caso de que en la zona del accidente había un cruce

  0. No hay cruce o lo hay a al menos 20 metros
  1. Rotonda
  2. Mini-rotonda
  3. Intersección en forma de codo

  5. Carril de deceleración o aceleración
  6. Cruce de carreteras
  7. Más de 4 vías sin rotonda
  8. Entrada privada
  9. Otro cruce

- Organización_Control_Cruce_Persona: Nos indica si existe alguna organización como policia o protección civil que se encargue del paso de peatones en la vía del accidente.

    0. No hay nada en al menos 50 metros
    1. Control escolar por una patrulla
    2. Control por otra organización autorizada

- Facilidades_Cruce_Personas: Indica si hay alguna infrastructura que facilite el paso de peatones en la zona.

    0. No hay en al menos 50 metros
    1. Paso de zebra

    4. "Paso de pelícano": paso de zebra controlado por semáforos.
    5. Semáforo para peatones

    7. Puente o tunel
    8. Refugio central: zona segura entre carriles de coches

- Condiciones lumínicas: sus posibles valores son:

    1. Luz diurna

    4. Oscuridad con luz artificial
    5. Oscuridad con dispositivos lumínicos apagados
    6. Oscuridad sin dispositivos lumínicos
    7. Oscuridad y desconocimiento sobre existencia de dispositivos lumínicos

- Condiciones climatológicas:

    1. Bueno, sin vientos fuertes
    2. Lloviendo sin fuertes vientos
    3. Nevando sin fuertes vientos
    4. Bueno con vientos fuertes
    5. Lloviendo con vientos fuertes
    6. Nevando con fuertes vientos
    7. Niebla
    8. Otro
    9. Desconocido

- Condiciones superficie vía: Indica las condiciones de la parte superficial de la vía en el momento del accidente.

    1. Seco
    2. Húmedo o mojado
    3. Nevado
    4. Helado
    5. Inundación por encima de los 3cm
    6. Aceite o combustible
    7. Barro

- Zona urbana o rural: Indica si la zona del acontecimiento pertenecía a una zona rural o urbana

    1. Urbana
    2. Rural
    3. Desconocida

- Tipo de vehículo:

    1. Vehículo a pedal
    2. Motocicleta inferior a 50cc
    3. Motocicleta inferior a 125cc
    4. Motocicleta superior a 125cc y por debajo de 500cc
    5. Motocicleta superior a 500cc

    8. Taxi o coche privado, alquilado
    9. Coche
    10. Minibus
    11. Autobús

    16. Caballo
    17. Vehículo agrícola
    18. Tranvía
    19. Caravana
    20. Camión entre 3.5 y 7.5 toneladas
    21. Camión mayor a 7.5 toneladas
    22. Scooter
    23. Motocicleta eléctrica

    90. Otro

    97. Motocicleta sin conocer sus cc
    98. Camión de mercancías sin conocer su peso

- Maniobra del vehículo: Indica la maniobra que realizaba el vehículo en el momento del accidente.

    1. Marcha atrás
    2. Aparcado
    3. Esperando para salir
    4. Decelerando o parando
    5. Apartándose
    6. Haciendo un cambio de sentido
    7. Girando izquierda
    8. Esperando para girar a la izquierda
    9. Girando a la derecha
    10. Esperando para girar a la derecha
    11. Cambiando al carril derecho
    12. Cambiando al carril izquierdo
    13. Apartando un vehículo en movimiento fuera de la carretera
    14. Apartando un vehículo parado fuera de la carretera

- Edad del vehículo: Nos índica la edad del vehículo principal afectado.

- Localización cruce : Indica donde se encontraba el cruce respecto a la zona del accidente

    0. No había en al menos 20 metros
    1. Intersección o esperando entrar al cruce
    2. Intersección depejada o esperando a salir de ella
    3. Saliendo de una rotonda
    4. Entrando a una rotonda
    5. Dejando la carretera principal
    6. Entrando a la carretera principal
    7. Entrando desde un carril de aceleración

8. En mitad de una intersección o de una rotonda

- Primer punto de impacto: Nos indica donde se produjo el primer impacto que recibió el vehículo implicado

0. No impactó
1. Choque frontal
2. Alcance
3. Choque al lateral del vehículo
4. Rozando alguna parte

- Objetivo del viaje : Objetivo que tenía el viaje para el conductor principal.

1. Viaje de trabajo
2. Traslado al puesto de trabajo
3. Llevando a los niños al colegio
4. Adolescentes conduciendo a casa
5. Otro
6. Desconocido

- Sexo del conductor:

1. Masculino
2. Femenino
3. Desconocido

- Edad del conductor
- Clase afectado: Indica el rol del implicado en el accidente.

1. Conductor
2. Pasajero
3. Peatón

- Sexo del afectado :

1. Masculino
2. Femenino
3. Desconocido

- Edad del afectado
- Fallecido

0. No hubo fallecidos
1. Hubo al menos un fallecido

Todas las demás columnas que inicialmente aparecen en la tabla serán borradas por temas de rapidez o por escasa información aportada.

Limpiar datos

En primer lugar voy a convertir todos los archivos excel a data frame y luego los voy a concatenar según el índice del accidente en la tabla data. Aunque a priori hay celdas vacias y valores a nulos pero luego los quitaremos.

```
data <- read.csv("dftRoadSafety_Accidents_2016.csv")
veh <- read.csv("veh.csv")
cas <- read.csv("Cas.csv")
data <- merge(merge(data, veh, by = "Accident_Index"), cas, by = "Accident_Index")
```

Vemos que la clase de nuestra tabla ya es un data frame

Ahora vamos a comenzar a limpiarla

```
#Eliminamos las variables que no podemos utilizar
data <- subset(data, select = -c(1,2,3,4,5,6,12,13,14,16,22,28,31,32,33,35,37,39,40,41,42,44,48,49,50,5

#Cambiar el nombre de las columnas
colnames(data) <- c("Gravedad_Accidente","Numero_Vehiculos,","Numero_Afectados","Fecha","Dia_Semana","T
```

Comenzamos viendo que aún quitando las variables que no nos sirven hay que modificar algunas columnas para poder utilizarlas en los algoritmos.

Vamos a convertir la variable límite de velocidad en entero para que detecte si hay valores vacios o nulos.

```
#Convertimos también la velocida máxima permitida en entero
data$Limite_Velocidad <- as.integer(data$Limite_Velocidad)

table(factor(data$Limite_Velocidad))
```

```
##
##     20     30     40     50     60     70
##  13531 198393  33348  18578  54798  35259
```

Vamos a ver en que columnas hay valores vacíos

```
#Vamos a comprobar en que columna están los datos vacios
colSums(is.na(data))
```

```
##                  Gravedad_Accidente                   Numero_Vehiculos,
##                                   0                                   0
##                    Numero_Afectados                               Fecha
##                                   0                                   0
##                         Dia_Semana                  Tipo_Via_principal
##                                   0                                   0
##                  Especificacion_Via                    Limite_Velocidad
##                                   0                                  86
##                      Detalles_cruce                       Control_Cruce
##                                   0                                   0
##                Tipo_Via_Secundaria Organizacion_Control_Cruce_Personas
##                                   0                                   0
##          Facilidades_Cruce_Personas               Condiciones_Luminicas
##                                   0                                   0
##          Condiciones_Climatologicas          Condiciones_Superficie_Via
##                                   0                                   0
##                          Imprevistos                   Zona_Urbana_Rural
##                                   0                                   0
```

```
##                      Tipo_Vehiculo                  Maniobra_Vehiculo
##                                  0                                  0
##                  Localizacion_Cruce              Primer_Punto_Impacto
##                                  0                                  0
##                     Objetivo_Viaje                   Sexo_Conductor
##                                  0                                  0
##                     Edad_Conductor                    Edad_Vehiculo
##                                  0                                  0
##                     Clase_Afectado                    Sexo_Afectado
##                                  0                                  0
##                      Edad_Afectado
##                                  0
```

Los datos vacíos se encuentran en la columna Límite de velocidad y al tener tantas filas lo más conveniente es eliminarlos.

```
data <- data[complete.cases(data), ]

#Volvemos a comporbar que valores son NA
anyNA(data)
```

```
## [1] FALSE
```

Como vemos ya hemos eliminado las filas que contenían valores vacios.

Vamos a comprobar también si hay valores nulos en alguna variable:

```
is.null(data)
```

```
## [1] FALSE
```

Al tener la variable fecha como carácter no la podemos utilizar para el estudio por lo que nos vamos a quedar con los meses para estudiar si un accidente en un mes determinado importa en su gravedad. El año es el mismo por lo que no va a explicar nada, y el día es complicado pues se vuelven a repetir cada mes y por tanto es difícil diferenciarlos con el mes

```
data$Fecha <- as.Date(data$Fecha, format = "%d/%m/%Y")

# Extraer los meses de la columna "fecha"
data$Mes <- format(data$Fecha, format = "%m")
#Convertimos los meses en entero
data$Mes <- as.integer(data$Mes)

#Convertida la variable fecha la eliminamos
data <- subset(data, select = -c(Fecha))

#Ahora voy a colocar la variable mes
Mes <- data$Mes
data <- subset(data, select = -c(Mes))
data <- cbind(data[, 1:4], Mes, data[, 5:ncol(data)])

#Comprobamos la variable
table(factor(data$Mes))
```

```
##
##     1     2     3     4     5     6     7     8     9    10    11    12
## 29238 27223 28082 28112 30005 29239 30977 31076 29544 30520 31582 28309
```

Convertir variable respuesta

Vamos a separar el archivo según la columna de gravedad del accidente. Vamos a dividir los accidentes entre los que tuvieron fallecidos y los que no pues estos primeros nos interesan más, y el resultado lo introducimos en una variable (Fallecido) en la que representamos con un 1 que sí que hay fallecidos y con un 0 que no hubo ningún fallecido. En este caso hay que comprobar la variable Gravedad del accidente pues es la que nos dice si algún individuo murió. Esta variable puede presentar 3 valores los cuales son:

- 1 : Hubo más de un fallecido.
- 2 : El accidente fue importante pues hubo heridos graves y un fallecido como mucho.
- 3 : El accidente ha sido leve y no hay fallecidos.

```
#Vamos a dividir las tuplas en dos con la nueva variable Gravedad_Accidente
data$Fallecido <- ifelse(data$Gravedad_Accidente > 2, "0", "1")

#Al dividir las tuplas ya no hace falta la variable gravedad_Accidente pues es explicativa por lo que l
data <- subset(data, select = -c(Gravedad_Accidente))

#Convertimos la variable a entero
data$Fallecido <- as.factor(data$Fallecido)

#Vamos a ver el resultado
table(data$Fallecido)
```

```
##
##      0      1
## 291166  62741
```

Como vemos la mayoría de los casos estudiados no implicaron ningún fallecido.

Corregir valores negativos

Voy a convertir todos los datos con un -1 a valores de tipo NA para que podamos limpiarlos y tratar con ellos. Al tener tantas filas lo más conveniente es eliminar todos esos datos para evitar contradicciones en los algoritmos, aunque bien es cierto que algunos algoritmos son capaces de trabajar con valores vacios(NA)

```
data[data == -1] <- NA
```

Ahora vamos a eliminar todas las filas que tienen valores vacios pues aún así tendremos muchas tuplas para trabajar.

```
condicion <- complete.cases(data[, -20])
data <- data[condicion, ]
```

Ahora ha corregido todos los valores 'NA' menos los de la columna de la localización del cruce por lo que la vamos a quitar.

```
data <- subset(data, select = -c(Localizacion_Cruce))
```

Quitamos filas para acelerar los algoritmos

Al tener una tabla tan grande, la ejecución de los algoritmos duraba demasiado tiempo y no era algo viable el tener que ejecutar cada vez que quería comprobar resultados nuevos. Por ello voy a quitar alrededor de 88mil filas para quedarme con unas 50mil y poder practicar de forma más ágil.

```
eliminados <- sample(nrow(data), 88000)

# Eliminar las filas del data frame
data <- data[-eliminados, ]
```

Finalmente las dimensiones de la tabla son:

```
dim(data)
```

```
## [1] 50992    28
```

Comprobar las variables y sus valores

Vamos a ver de forma rápida las variables que tenemos :

```
str(data)
```

```
## 'data.frame':    50992 obs. of  28 variables:
##  $ Numero_Vehiculos,                  : int  1 1 2 3 4 2 2 1 2 2 ...
##  $ Numero_Afectados                   : int  1 1 1 1 1 3 3 1 1 1 ...
##  $ Dia_Semana                         : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Mes                                : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ Tipo_Via_principal                 : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Especificacion_Via                 : int  6 6 6 6 2 6 6 6 6 6 ...
##  $ Limite_Velocidad                   : int  30 30 30 30 30 30 30 30 30 30 ...
##  $ Detalles_cruce                     : int  9 3 3 9 3 3 3 3 3 3 ...
##  $ Control_Cruce                      : int  4 2 4 4 4 4 4 2 4 4 ...
##  $ Tipo_Via_Secundaria                : int  6 3 6 6 6 6 6 3 5 4 ...
##  $ Organizacion_Control_Cruce_Personas: int  0 0 0 0 2 0 0 0 0 0 ...
##  $ Facilidades_Cruce_Personas         : int  0 0 8 0 4 0 0 4 0 0 ...
##  $ Condiciones_Luminicas              : int  4 1 1 1 1 1 1 4 1 4 ...
##  $ Condiciones_Climatologicas         : int  1 1 1 1 2 1 1 1 1 1 ...
##  $ Condiciones_Superficie_Via         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Imprevistos                        : int  0 0 0 0 2 0 0 0 0 0 ...
##  $ Zona_Urbana_Rural                  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Tipo_Vehiculo                      : int  9 11 9 19 9 9 9 11 9 8 ...
##  $ Maniobra_Vehiculo                  : int  18 18 18 3 3 18 18 4 9 7 ...
##  $ Primer_Punto_Impacto               : int  4 0 2 4 2 1 1 1 1 1 ...
```

```
##  $ Objetivo_Viaje                    : int  6 1 6 6 2 6 6 1 6 1 ...
##  $ Sexo_Conductor                    : int  2 1 2 1 2 1 1 1 1 1 ...
##  $ Edad_Conductor                    : int  36 55 30 29 52 37 37 37 38 48 ...
##  $ Edad_Vehiculo                     : int  1 7 10 3 13 14 14 5 16 2 ...
##  $ Clase_Afectado                    : int  1 2 1 1 1 1 1 3 2 1 ...
##  $ Sexo_Afectado                     : int  2 2 2 1 2 1 1 1 2 1 ...
##  $ Edad_Afectado                     : int  36 59 30 36 52 79 37 22 53 31 ...
##  $ Fallecido                         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Como vemos todas las variables son de tipo entero con valores todos positivos (pues ya borramos anteriormente las filas que presentaban valores negativos) menos la variable Fallecido que es nuestra variable respuesta y que es de tipo factor para poder utilizarla mejor en los algoritmos.

División de datos

Vamos a comenzar dividiendo nuestros datos en la parte de entrenamiento y la parte de test. La proporción será de 70% para el entrenamiento, 30% para el test, y en ningún momento se podrá filtrar información de los datos de test a los del entrenamiento. En primer lugar vamos a fijar una semilla para que siempre haga la misma división de los datos.

```
library(caret)

set.seed(100)
muestra <- createDataPartition(data$Fallecido, p = .70, list = FALSE)

train.accidentes=data[muestra,]
test.accidentes=data[-muestra,]

dim(train.accidentes)
```

```
## [1] 35695    28
```

```
dim(test.accidentes)
```

```
## [1] 15297    28
```

```
prop.table(table(train.accidentes$Fallecido))
```

```
##
##         0         1
## 0.8464771 0.1535229
```

```
prop.table(table(test.accidentes$Fallecido))
```

```
##
##         0         1
## 0.8465059 0.1534941
```

Entrenamiento del modelo

Para poder comparar correctamente el modelo entrenado por cada algoritmo hay que asegurar que se utilizan las mismas particiones de datos para todos los algoritmos a utilizar. Vamos a crear nuestras particiones con la función create folds de la librería caret. Vamos a utilizar en nuestro caso 5 folds:

```
#Primero fijamos la semillas para que siempre nos de el mismo resultado
set.seed(100)
modelos <- createFolds(train.accidentes$Fallecido, k = 5)

trainControl <- trainControl(method = "cv", number = 5, index = modelos)
```

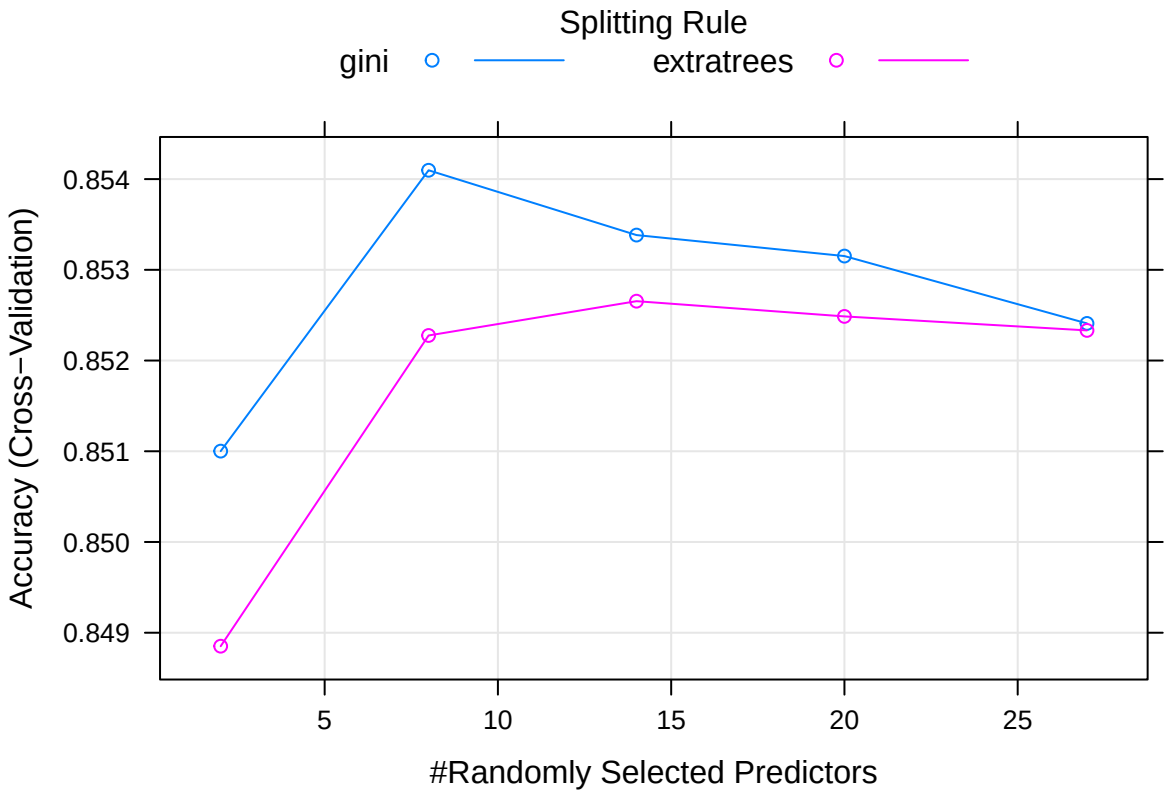
Random Forest

El primer algoritmo que voy a utilizar es random forest. En primer lugar voy a crear un objeto llamado caret y dentro de el contendrá instancias de los diferentes algoritmos:

```
RF<- train(Fallecido ~ ., data = train.accidentes,method = "ranger",trControl = trainControl, tuneLengt

RF
```

```
## Random Forest
##
## 35695 samples
##    27 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 7139, 7139, 7139, 7139, 7139
## Resampling results across tuning parameters:
##
##   mtry  splitrule   Accuracy   Kappa
##    2    gini        0.8510015  0.05303228
##    2    extratrees  0.8488514  0.02589346
##    8    gini        0.8540972  0.12764002
##    8    extratrees  0.8522762  0.08213138
##   14    gini        0.8533828  0.14084395
##   14    extratrees  0.8526544  0.10210042
##   20    gini        0.8531517  0.14950314
##   20    extratrees  0.8524863  0.11148258
##   27    gini        0.8524093  0.15359827
##   27    extratrees  0.8523323  0.12053838
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 8, splitrule = gini
##  and min.node.size = 1.
```

```
plot(RF)
```



El Random Forest, o en español Bosque Aleatorio, es una técnica de machine learning que utiliza muchos árboles de decisión para realizar predicciones. La ventaja principal frente al modelo de un solo árbol se encuentra en que con un solo árbol la decisión puede ser mucho más drástica en el sentido en el que puede haber una gran probabilidad debida al azar de los datos y que el algoritmo no la tiene en cuenta. Por tanto, en el modelo de Bosque Aleatorio, esa variabilidad debida al azar se reduce, ya que la predicción final se decide teniendo en cuenta todos los árboles del bosque. Además, se evitan fenómenos como el sobreajuste al tener en cuenta diferentes conjuntos de datos.

La técnica del Random Forest elige datos aleatorios para cada árbol, es decir, los datos pueden elegirse ninguna, una o muchas veces. Cada árbol del bosque estará creado mediante una técnica como Bagging o Bootstrap Aggregating. El Bootstrap consiste en que, en lugar de utilizar los mismos datos como sería el caso de un único árbol de decisión, cada vez que se genera un árbol de decisión se eligen datos aleatorios del dataset completo. Esto conlleva que haya datos que puedan no aparecer en ningún árbol o repetirse en varios árboles. Este efecto conlleva dos ventajas principales:

- La primera ventaja consiste en que las decisiones debidas a la aleatoriedad de los datos se reducen, ya que cada árbol se entrena con datos diferentes. Además, esto ayuda mucho a evitar el sobreajuste.
- Al utilizar datos diferentes, podemos predecir mucho mejor la importancia que tiene cada variable en la predicción del modelo, ya que se puede obtener una medida de la importancia relativa de las características en el conjunto de datos.

Este mecanismo de Bootstrap genera un Bootstrap dataset que es el que utiliza el árbol para su predicción, y este mecanismo se repite generalmente más de 100 veces (en algunos algoritmos, el número de árboles por defecto es de 500, lo cual es un tamaño más que considerable para eliminar la aleatoriedad). En resumen, la técnica del Random Forest mezcla las predicciones de cada árbol mediante una especie de media o una "votación" para obtener una predicción final. Esto proporciona un clasificador muy robusto

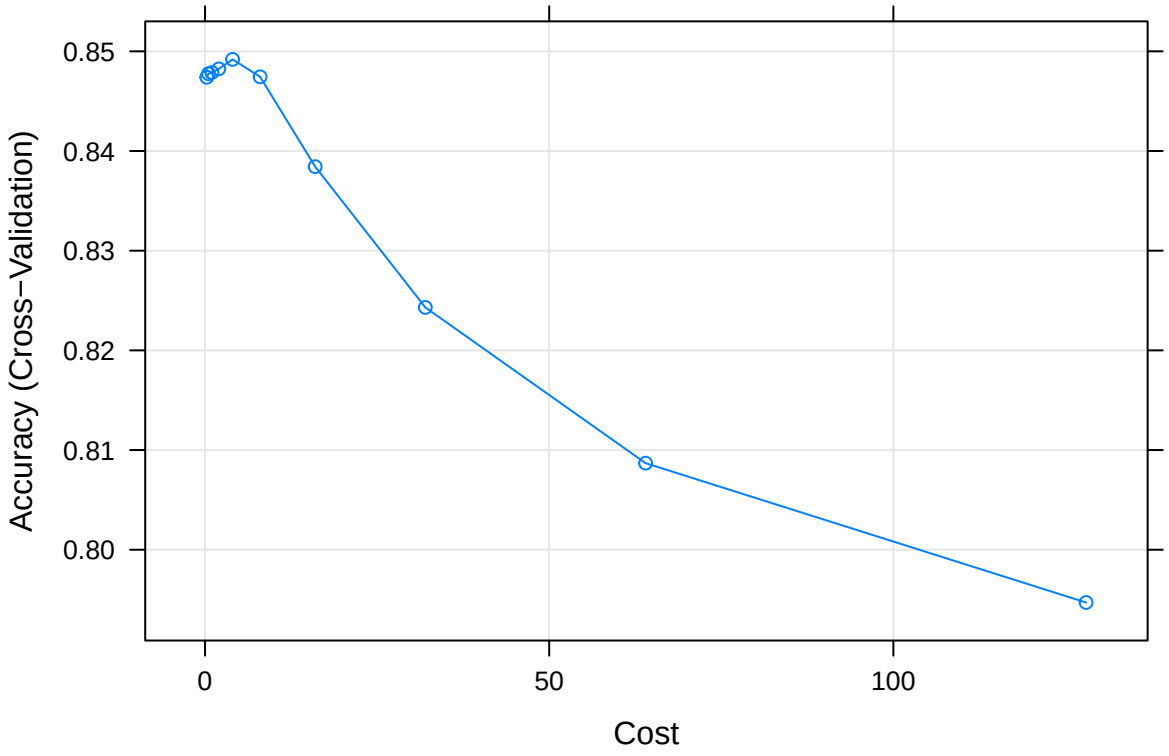
que está habilitado para manejar conjuntos enormes de datos y dar un valor de importancia a las variables utilizadas en el modelo.

Support Vector Machine

```
SVM <- train(Fallecido ~ .,data = train.accidentes,method = "svmRadial",preProcess = c("center", "scale

SVM
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 35695 samples
##    27 predictor
##     2 classes: '0', '1'
##
## Pre-processing: centered (27), scaled (27)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 7139, 7139, 7139, 7139, 7139
## Resampling results across tuning parameters:
##
##   C       Accuracy   Kappa
##     0.25  0.8474016  0.01015239
##     0.50  0.8477518  0.01397440
##     1.00  0.8478779  0.01565687
##     2.00  0.8482421  0.02200749
##     4.00  0.8491946  0.04707525
##     8.00  0.8474436  0.09367862
##    16.00  0.8384298  0.13454111
##    32.00  0.8243101  0.15248159
##    64.00  0.8086847  0.15628447
##   128.00  0.7947051  0.14963931
##
## Tuning parameter 'sigma' was held constant at a value of 0.02497268
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.02497268 and C = 4.
```

```
plot(SVM)
```



El método de support vector machine es un tipo de algoritmo para entrenamiento de un modelo en machine learning. Consiste en encontrar el mejor plano de separación entre diferentes categorías de datos en grandes cantidades de datos. Este algoritmo es particularmente útil cuando nuestros datos no pueden ser facilmente separados de forma lineal.

El objetivo principal del support vector machine es encontrar un hiperplano lo más perfecto posible que maximice la distancia entre los puntos de datos más cercanos entre las diferentes clases, que son los vectores de apoyo. Estos vectores de apoyo son esenciales para definir el plano de separación.

La ventaja de SVM se encuentra en su capacidad para manejar datos en espacios de alta dimensión y su capacidad para lidiar con situaciones donde los límites de decisión no son lineales. Esto se logra mediante el uso de funciones de núcleo, que mapean los datos en un espacio de mayor dimensión donde se pueden encontrar planos de separación lineal.

Una vez que el modelo SVM está entrenado, se evalúa su rendimiento utilizando métricas de evaluación, como la exactitud, el área bajo la curva ROC o la matriz de confusión. Estas métricas permiten medir la capacidad del modelo para clasificar correctamente los datos y evaluar su calidad.

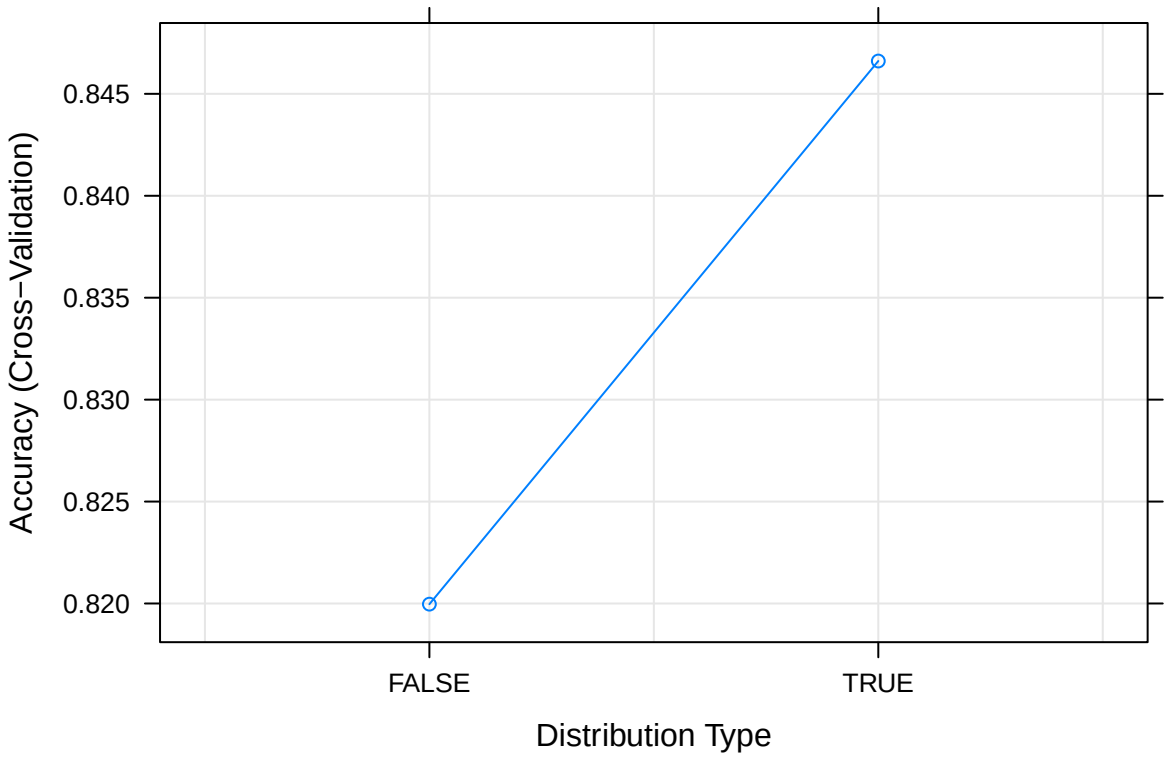
Naive Bayes

```
NaiveB <- train(Fallecido ~ .,data = train.accidentes,method = "naive_bayes",preProcess = c("center", "

NaiveB
```

```
## Naive Bayes
```

```
##
## 35695 samples
##    27 predictor
##     2 classes: '0', '1'
##
## Pre-processing: centered (27), scaled (27)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 7139, 7139, 7139, 7139, 7139
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.8199678  0.10614565
##    TRUE      0.8466102  0.00385424
##
## Tuning parameter 'laplace' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = TRUE
##  and adjust = 1.
```

```
plot(NaiveB)
```



El método de Naive Bayes es un algoritmo de aprendizaje utilizado para la clasificación de datos. Se basa en el principio de "Teorema de Bayes" para estimar la probabilidad de pertenencia a una clase dadas ciertas características observadas en los datos.

La suposición principal del método de "Naive Bayes" es que todas las características son independientes entre sí, lo cual puede ser una simplificación excesiva en algunos casos, pero permite un cálculo eficiente de las probabilidades condicionales.

Durante el proceso de entrenamiento, se estiman las probabilidades de las clases y las distribuciones de las características utilizando el conjunto de datos de entrenamiento. Para ello, se pueden utilizar diferentes variantes de Naive Bayes, como Naive Bayes Gaussiano, Naive Bayes Multinomial o Naive Bayes Bernoulli, dependiendo de la naturaleza de las características.

Naive Bayes es especialmente adecuado para conjuntos de datos con muchas características y cuando la suposición de independencia entre las características es razonable. Sin embargo, puede tener limitaciones en escenarios con dependencias complejas entre las características o cuando se requiere una alta precisión en la clasificación.

Comparando los modelos

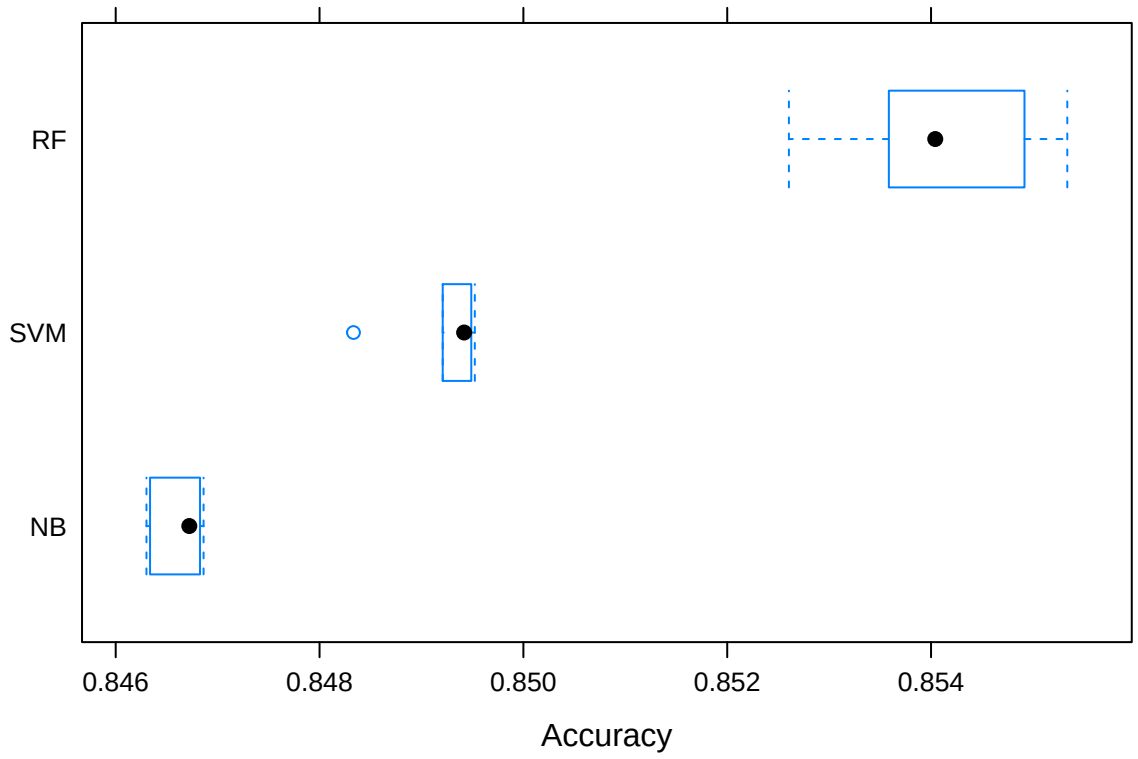
Vamos por último a comparar los modelos de entrenamientos realizados anteriormente. Para comparar estos modelos vamos a utilizar resamples que es una función de caret apta para los entrenamientos realizados con caret.

```
library(resample)
# Creamos la lista de modelos a comparar
lista_mod <- list(SVM = SVM, RF = RF,NB = NaiveB)

# Guardamos en objeto comparacion_modelos el resultado de la función resamples
comparacion <- resamples(lista_mod)

# Resultados
summary(comparacion)
```

```
##
## Call:
## summary.resamples(object = comparacion)
##
## Models: SVM, RF, NB
## Number of resamples: 5
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## SVM 0.8483331 0.8492086 0.8494187 0.8491946 0.8494887 0.8495237    0
## RF  0.8526054 0.8535859 0.8540412 0.8540972 0.8549167 0.8553369    0
## NB  0.8463020 0.8463370 0.8467222 0.8466102 0.8468273 0.8468623    0
##
## Kappa
##             Min.      1st Qu.      Median       Mean     3rd Qu.        Max.
## SVM 0.0357143915 0.0462950041 0.047324261 0.04707525 0.052825185 0.053217418
## RF  0.1103032157 0.1237159966 0.127531488 0.12764002 0.134960777 0.141688622
## NB  0.0002816677 0.0003516615 0.005495842 0.00385424 0.006364539 0.006777487
##     NA's
## SVM    0
## RF     0
## NB     0
```

Como vemos a simple vista el que mejor media de precisión presenta es el RF y seguramente sea el que mejor predice el modelo

```
bwplot(comparacion, metric = "Accuracy")
```



En el box plot podemos comprobar como por diferencia el mejor predictor es random forest

Obtención del resultado con el modelo de entrenamiento seleccionado y los datos de test

Ahora vamos a utilizar el modelo random forest para probar como predice en los datos de test:

```
#Predecimos el modelo
prediccionFinal <- predict(RF ,test.accidentes)

#Sacamos la matriz de confusión
matrizConfusion <- confusionMatrix(prediccionFinal,test.accidentes$Fallecido, positive = "1")

#Mostramos la matriz de confusión
matrizConfusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction     0     1
##          0 12885  1872
##          1    64   476
##
##                Accuracy : 0.8734
##                  95% CI : (0.8681, 0.8787)
##     No Information Rate : 0.8465
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2888
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.20273
##             Specificity : 0.99506
##          Pos Pred Value : 0.88148
##          Neg Pred Value : 0.87314
##              Prevalence : 0.15349
##          Detection Rate : 0.03112
##    Detection Prevalence : 0.03530
##       Balanced Accuracy : 0.59889
##
##        'Positive' Class : 1
##
```

```
#Vamos a meter todos los valores en un data frame para poder mostrarlos bien
sensitivity <- matrizConfusion$byClass["Sensitivity"]
specificity<- matrizConfusion$byClass["Specificity"]
recall <- matrizConfusion$byClass["Recall"]
precision <- matrizConfusion$byClass["Precision"]
accuracy <- matrizConfusion$overall["Accuracy"]
F1 <- matrizConfusion$byClass["F1"]


resultados <- data.frame(Modelo = "RandomForest(caret-ranger)",
                         Accuracy = round(accuracy,4),
                         Recall = round(sensitivity,4),
                         Specificity = round(specificity,4),
                         Precision = round(precision,4),
                         F1 = round(F1,4),
                         row.names = NULL)

resultados
```

```
##                       Modelo Accuracy Recall Specificity Precision     F1
## 1 RandomForest(caret-ranger)   0.8734 0.2027      0.9951    0.8815 0.3296
```

Los resultados explicados son:

- Matriz de confusión: La matriz de confusión muestra la proporción de aciertos y fallos realizadas por el modelo. En este caso, la matriz de confusión es de tamaño 2x2, donde las filas representan las clases que hemos intentado predecir y las columnas representan la referencia.

- Accuracy (Precisión): La precisión es la medida de la exactitud del modelo, es decir, la proporción de

predicciones correctas sobre el total de predicciones realizadas. En este caso, la precisión es de 84.9% de los casos clasificados correctamente.

- 95% CI: El intervalo de confianza del 95% proporciona un rango estimado donde se espera que esté el valor real de la precisión. En este caso, el intervalo de confianza está entre 0.8451 y 0.8528.

- No Information Rate: La tasa de no información es la precisión que se obtendría si se predijera siempre la clase más común. En este caso, la tasa de no información es de 0.8139, lo que indica que el modelo tiene un rendimiento superior a la predicción más básica.

- Kappa: El coeficiente de Kappa es una medida de la concordancia entre las clasificaciones del modelo y las clasificaciones esperadas al azar. En este caso, el valor de Kappa es 0.3452, lo que indica una concordancia moderada.

- Sensitivity (Sensibilidad): La sensibilidad es el porcentaje de casos reales que el modelo acertó al clasificarlos. En este caso, la sensibilidad es 0.28475, lo que indica que el modelo identificó correctamente el 28.475% de los casos positivos.

- Specificity (Especificidad): La especificidad es el porcentaje de casos negativos que de realmente son negativos que fueron clasificados correctamente como negativos por el modelo. En este caso, la especificidad es 0.97800, lo que indica que el modelo identificó correctamente el 97.8% de los casos negativos.

En general para casi cualquier base de datos random forest produce generalmente un modelo bastante bueno clasificando pues al utilizar tantos árboles es casi como si repitiera muchas veces el algoritmo de árbol de decisión simple. En nuestro caso la predicción del modelo es buena (84.9%) teniendo en cuenta que hemos probado ya los datos de test y la precisión es parecida a la de los datos de entreamiento, además de que no hemos sobreajustado el modelo.

Bibliografía

- https://topepo.github.io/caret/index.html
- https://www.youtube.com/watch?v=HJB6XFkmezM&t=601s
- https://www.youtube.com/watch?v=_JdK4FMzd28&t=546s
- https://www.youtube.com/watch?v=tmahDuw0s3g
- Data Mining Theories- Algorithms- and Examples, NONG YE, CRC Press
- Data Mining Practical Machine Learning Tools and Techniques, Ian H. Witten, Eibe Frank
- https://rpubs.com/joser/caret
- https://remolinator.com/svm/