



*ugr* | Universidad  
de **Granada**

MÁSTER EN INGENIERÍA INFORMÁTICA  
SISTEMAS INTELIGENTES PARA LA GESTIÓN EN LA EMPRESA

---

## Práctica 1. Preprocesamiento de datos y clasificación binaria. Predicción de diabetes.

---

Autor  
José Alberto Gómez García



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

—  
Granada, abril de 2023

## Tabla de Contenidos

<b>1</b>	<b>Introducción</b>	<b>2</b>
<b>2</b>	<b>Lectura de datos</b>	<b>2</b>
<b>3</b>	<b>Análisis exploratorio</b>	<b>2</b>
3.1	Resumen de las características . . . . .	2
3.2	Resumen de los datos . . . . .	3
3.3	Estratificación . . . . .	5
3.4	DataExplorer . . . . .	8
<b>4</b>	<b>Procesamiento de datos.</b>	<b>13</b>
4.1	Tratamiento de valores perdidos. . . . .	13
4.2	Selección de características . . . . .	14
4.3	Detección y tratamiento de outliers . . . . .	15
4.4	Discretización de variables . . . . .	16
4.5	Normalización de variables . . . . .	17
<b>5</b>	<b>Clasificación con varios modelos</b>	<b>18</b>
5.1	Árboles de decisión. . . . .	18
5.2	Random Forest . . . . .	22
5.3	Naive Bayes . . . . .	24
5.4	Redes neuronales artificiales . . . . .	25
<b>6</b>	<b>Comparativa de los resultados</b>	<b>28</b>
<b>7</b>	<b>Conclusiones</b>	<b>29</b>
<b>8</b>	<b>Bibliografía</b>	<b>30</b>

# 1 Introducción

El problema propuesto en esta práctica se deriva de una competición de Kaggle, llamada *Diabetes Health Indicators Dataset*, cuyo objetivo es la detección y prevención de la enfermedad de la diabetes a partir de una serie de indicadores de salud y factores de riesgo, tales como la presencia de colesterol, alcoholismo; así como datos demográficos, como edad y sexo.

En esta práctica se llevarán a cabo una serie de tareas, especialmente centradas con el análisis exploratorio y el pre-procesamiento de datos, que habilitarán la construcción de modelos con los que predecir la posibilidad de padecer diabetes o no. Cabe destacar que la diabetes es una enfermedad crónica que afecta a millones de personas en todo el mundo, y su prevención y tratamiento son de gran importancia para la salud pública.

## 2 Lectura de datos

En primer lugar, deberemos realizar la lectura de los datos desde el fichero .csv que se nos proporciona para el desarrollo de esta práctica. En este paso aprovechamos también para re-codificar los valores vacíos, que en este dataset aparecen codificados con el valor -999, y cambiar las etiquetas de la variable que utilizaremos para clasificar a valores más legibles, como *No* o *Yes*.

```
training_data_raw <- read_csv2("data/diabetes.csv", na = "-999", show_col_types = FALSE)

training_data <- training_data_raw %>%
  mutate(Label = ifelse(Diabetes_binary == 0, "No", "Yes")) %>%
  select(-Diabetes_binary)
```

## 3 Análisis exploratorio

Tras haber leído los datos realizaremos un análisis exploratorio de los mismos. Este paso es esencial, pues puede ayudarnos a comprender las características de los datos, identificar patrones, tendencias, valores atípicos, etc. Esto nos permitirá seleccionar las técnicas de procesamiento y análisis más adecuadas.

### 3.1 Resumen de las características

Antes de afrontar el procesamiento de los datos, tenemos que comprender qué características aparecen en el conjunto de datos proporcionado. Por tanto, comenzaremos listando el significado de cada una de las variables presentes.

- **Diabetes binary.** Presencia de diabetes (1) o no (0) en el paciente.
- **High BP.** Presencia de presión sanguínea elevada en el paciente.
- **High Col.** Presencia de altos niveles de colesterol en sangre.
- **Chol Check.** Indica si el paciente ha medido su nivel de colesterol en los últimos 5 años.
- **BMI.** Índice de masa corporal.
- **Smoker.** Indica si el paciente ha fumado al menos 100 cigarrillos en toda su vida.
- **Stroke.** Indica si se ha tenido alguna vez un ictus o derrame cerebral.
- **HeartDiseaseorAttack.** Representa si el paciente ha sufrido de una cardiopatía coronaria o infarto de miocardio.
- **PhysActivity.** Representa si el paciente ha realizado ejercicio físico, fuera del trabajo, en los últimos 30 días.
- **Fruits y Veggies.** Consumo de frutas o verduras al menos una vez al día.
- **HvyAlcoholConsump.** Indica si el paciente es un “gran bebedor”, entendiendo como tal a los hombres que toman mas de 14 bebidas a la semana, o a las mujeres que toman más de 7 bebidas semanales.
- **AnyHealthCare.** Informa de si el paciente tienen contratado algún tipo de cobertura sanitaria.
- **NoDocbcCst.** Indica si el paciente tendría que haber visitado al doctor en los últimos 12 meses pero no lo hizo debido al costo asociado.
- **GenHlth.** En una escala del 1 (excelente) al 5 (péssimo), indica el estado de salud del paciente según él mismo.

- **MenHlth.** Durante cuantos días durante el último mes considera el paciente que ha sufrido estrés, depresión, problemas emocionales, etc.
- **PhysHlth.** Durante cuantos días durante el último mes considera el paciente que su salud física no ha sido buena, ha sufrido heridas, etc.
- **DiffWalk.** Indica si se tienen serias dificultades subiendo escaleras o andando.
- **Sex.** Hombre o mujer.
- **Age.** Edad según la escala AGE5YR (que agrupa en rangos de 5 años y mayores de 80)
- **Income.** Ingresos anuales según la escala INCOME2.
- **Education.** Nivel educativo según la escala EDUCA.

### 3.2 Resumen de los datos

Leídos los datos y entendiendo el significado de cada uno de ellos, visualicemos un resumen de distintas métricas estadísticas para cada uno de ellos. Muchos de los valores provienen de encuestas en las que la respuesta es binaria, representándose *Yes* como 1 y *No* como 0.

```
summary(training_data)
```

```
##      HighBP      HighChol      CholCheck      BMI
##  Min.   :0.0000  Min.   :0.0000  Min.   :0.0000  Min.   :12.00
##  1st Qu.:0.0000  1st Qu.:0.0000  1st Qu.:1.0000  1st Qu.:25.00
##  Median :1.0000  Median :1.0000  Median :1.0000  Median :28.00
##  Mean   :0.5319  Mean   :0.5018  Mean   :0.9723  Mean   :29.52
##  3rd Qu.:1.0000  3rd Qu.:1.0000  3rd Qu.:1.0000  3rd Qu.:33.00
##  Max.   :1.0000  Max.   :1.0000  Max.   :1.0000  Max.   :98.00
##                NA's   :591     NA's   :1537
##      Smoker      Stroke      HeartDiseaseorAttack      PhysActivity
##  Min.   :0.0000  Min.   :0.00000  Min.   :0.0000  Min.   :0.0000
##  1st Qu.:0.0000  1st Qu.:0.00000  1st Qu.:0.0000  1st Qu.:0.0000
##  Median :0.0000  Median :0.00000  Median :0.0000  Median :1.0000
##  Mean   :0.4719  Mean   :0.05754  Mean   :0.1359  Mean   :0.7148
##  3rd Qu.:1.0000  3rd Qu.:0.00000  3rd Qu.:0.0000  3rd Qu.:1.0000
##  Max.   :1.0000  Max.   :1.00000  Max.   :1.0000  Max.   :1.0000
##                NA's   :601
##      Fruits      Veggies      HvyAlcoholConsump      AnyHealthcare
##  Min.   :0.0000  Min.   :0.0000  Min.   :0.00000  Min.   :0.0000
##  1st Qu.:0.0000  1st Qu.:1.0000  1st Qu.:0.00000  1st Qu.:1.0000
##  Median :1.0000  Median :1.0000  Median :0.00000  Median :1.0000
##  Mean   :0.6191  Mean   :0.7931  Mean   :0.04596  Mean   :0.9543
##  3rd Qu.:1.0000  3rd Qu.:1.0000  3rd Qu.:0.00000  3rd Qu.:1.0000
##  Max.   :1.0000  Max.   :1.0000  Max.   :1.00000  Max.   :1.0000
##                NA's   :5748
##      NoDocbcCost      GenHlth      MentHlth      PhysHlth
##  Min.   :0.00  Min.   :1.000  Min.   : 0.000  Min.   : 0.000
##  1st Qu.:0.00  1st Qu.:2.000  1st Qu.: 0.000  1st Qu.: 0.000
##  Median :0.00  Median :3.000  Median : 0.000  Median : 0.000
##  Mean   :0.09  Mean   :2.756  Mean   : 3.629  Mean   : 5.425
##  3rd Qu.:0.00  3rd Qu.:3.000  3rd Qu.: 2.000  3rd Qu.: 5.000
##  Max.   :1.00  Max.   :5.000  Max.   :30.000  Max.   :30.000
##                NA's   :5748
##      DiffWalk      Sex          Age          Education
##  Min.   :0.000  Min.   :0.0000  Min.   : 1.000  Min.   :1.000
##  1st Qu.:0.000  1st Qu.:0.0000  1st Qu.: 7.000  1st Qu.:4.000
##  Median :0.000  Median :0.0000  Median : 9.000  Median :5.000
##  Mean   :0.232  Mean   :0.4532  Mean   : 8.463  Mean   :4.959
##  3rd Qu.:0.000  3rd Qu.:1.0000  3rd Qu.:11.000 3rd Qu.:6.000
##  Max.   :1.000  Max.   :1.0000  Max.   :13.000  Max.   :6.000
##                NA's   :5748
##      Income      Label
##  Min.   :0.0000  Min.   :0.0000
##  1st Qu.:0.0000  1st Qu.:0.0000
##  Median :0.0000  Median :0.0000
##  Mean   :0.0000  Mean   :0.0000
##  3rd Qu.:0.0000  3rd Qu.:0.0000
##  Max.   :1.0000  Max.   :1.0000
```

```

## Min. :1.000 Length:60796
## 1st Qu.:4.000 Class :character
## Median :6.000 Mode :character
## Mean :5.801
## 3rd Qu.:8.000
## Max. :8.000
## NA's :668

```

También podemos consultar el estado de cada una de las variables del conjunto de datos.

```
status <- df_status(training_data)
```

	variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
## 1	HighBP	28457	46.81	0	0.00	0	0	numeric	2
## 2	HighChol	30288	49.82	0	0.00	0	0	numeric	2
## 3	CholCheck	1666	2.74	591	0.97	0	0	numeric	2
## 4	BMI	0	0.00	1537	2.53	0	0	numeric	76
## 5	Smoker	32107	52.81	0	0.00	0	0	numeric	2
## 6	Stroke	57298	94.25	0	0.00	0	0	numeric	2
## 7	HeartDiseaseorAttack	52017	85.56	601	0.99	0	0	numeric	2
## 8	PhysActivity	17339	28.52	0	0.00	0	0	numeric	2
## 9	Fruits	23156	38.09	0	0.00	0	0	numeric	2
## 10	Veggies	12577	20.69	0	0.00	0	0	numeric	2
## 11	HvyAlcoholConsump	58002	95.40	0	0.00	0	0	numeric	2
## 12	AnyHealthcare	2776	4.57	0	0.00	0	0	numeric	2
## 13	NoDocbcCost	50082	82.38	5748	9.45	0	0	numeric	2
## 14	GenHlth	0	0.00	0	0.00	0	0	numeric	5
## 15	MentHlth	41518	68.29	0	0.00	0	0	numeric	31
## 16	PhysHlth	35254	57.99	0	0.00	0	0	numeric	31
## 17	DiffWalk	46693	76.80	0	0.00	0	0	numeric	2
## 18	Sex	33242	54.68	0	0.00	0	0	numeric	2
## 19	Age	0	0.00	0	0.00	0	0	numeric	13
## 20	Education	0	0.00	0	0.00	0	0	numeric	6
## 21	Income	0	0.00	668	1.10	0	0	numeric	8
## 22	Label	0	0.00	0	0.00	0	0	character	2

De esta tabla podemos obtener información como la cantidad de valores 0 que hay para cada variable (así como la probabilidad). También podemos ver rápidamente en qué variables tenemos valores nulos, gracias a la columna “q\_na”. Estas filas son *CholCheck*, *BMI*, *HeartDiseaseOrAttack*, *NoDocbcCost* e *Income*. Gracias al número de valores únicos para cada variable podemos hacernos una idea de que variables son binarias y cuáles son realmente numéricas o resultado de un proceso de discretización (como *Age*).

También resulta conveniente consultar si el conjunto de datos se encuentra balanceado o no. En este caso, tenemos 35346 observaciones en las que no se padece la enfermedad de la diabetes y 25450 observaciones en las que sí se padece. Podemos observar que el conjunto de datos se encuentra desbalanceado en cierta medida, al tener un 58.14% de instancias de una clase y el 41.86% de otra, pero no parece ser demasiado preocupante.

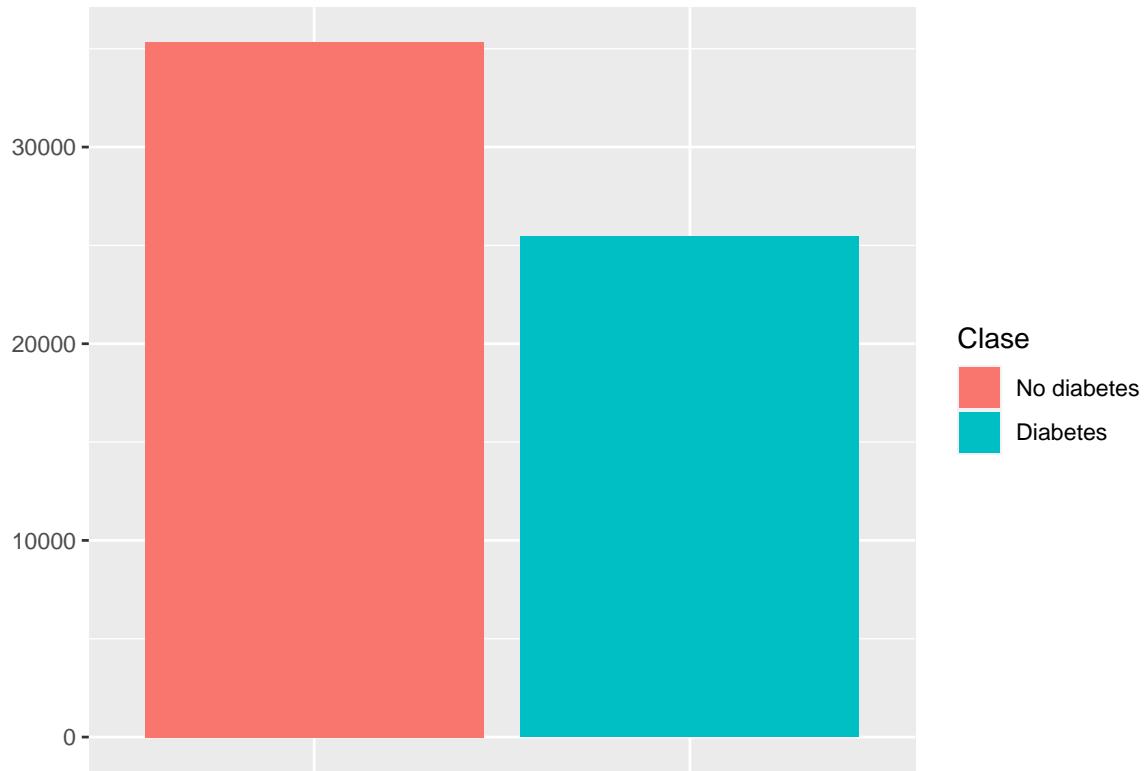
```
table(training_data$Label)
```

```

## 
##   No   Yes 
## 35346 25450 

ggplot(training_data) +
  geom_histogram(aes(x = Label, fill = Label), stat = "count") +
  scale_fill_discrete(name = "Clase", labels=c("No diabetes", "Diabetes")) +
  labs(x = "", y = "") +
  theme(axis.text.x=element_blank())

```



### 3.3 Estratificación

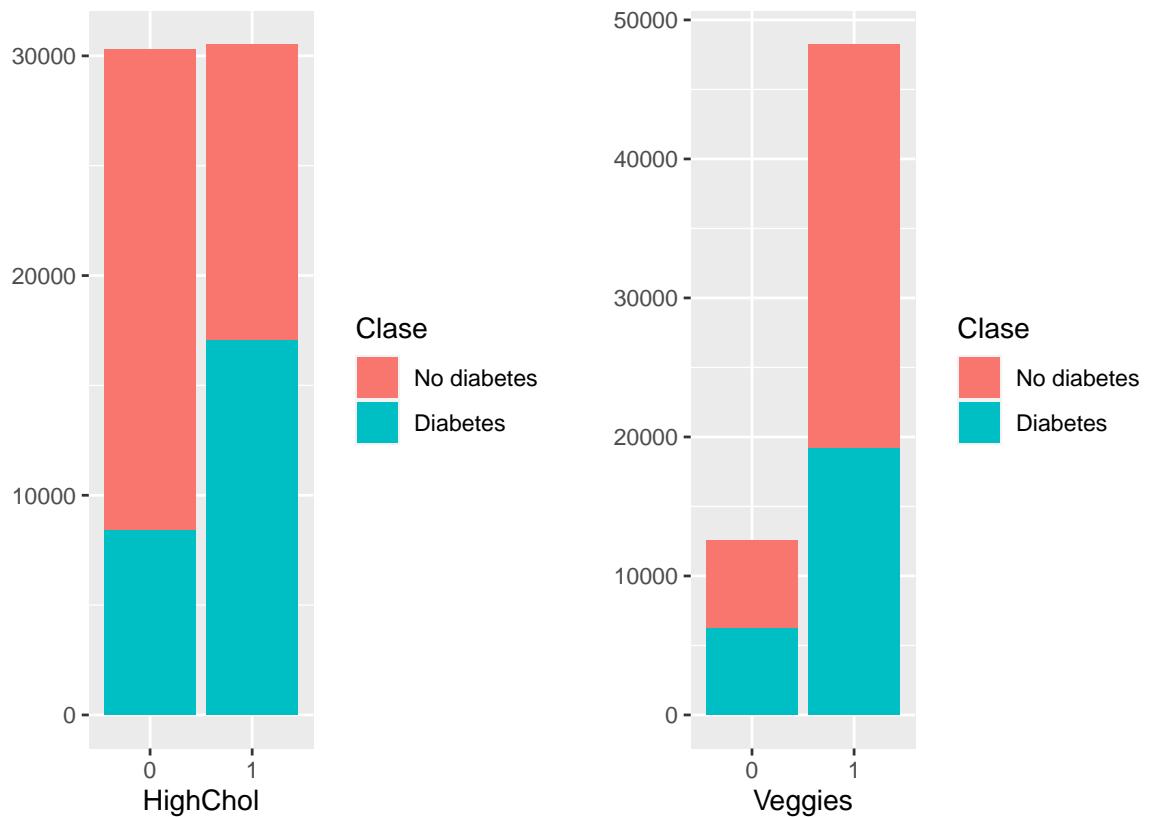
Como veíamos anteriormente, el conjunto de datos presenta un cierto desbalanceo, al existir aproximadamente un 16.3% más de instancias negativas que positivas (respecto de padecer la enfermedad). Estos resultados parecen tener una cierta lógica, dado que uno tiende a pensar que es más frecuente no padecer diabetes.

En este apartado, haremos uso del procedimiento conocido como *estratificación* para observar la distribución de la variable de clasificación respecto de otras variables del conjunto de datos.

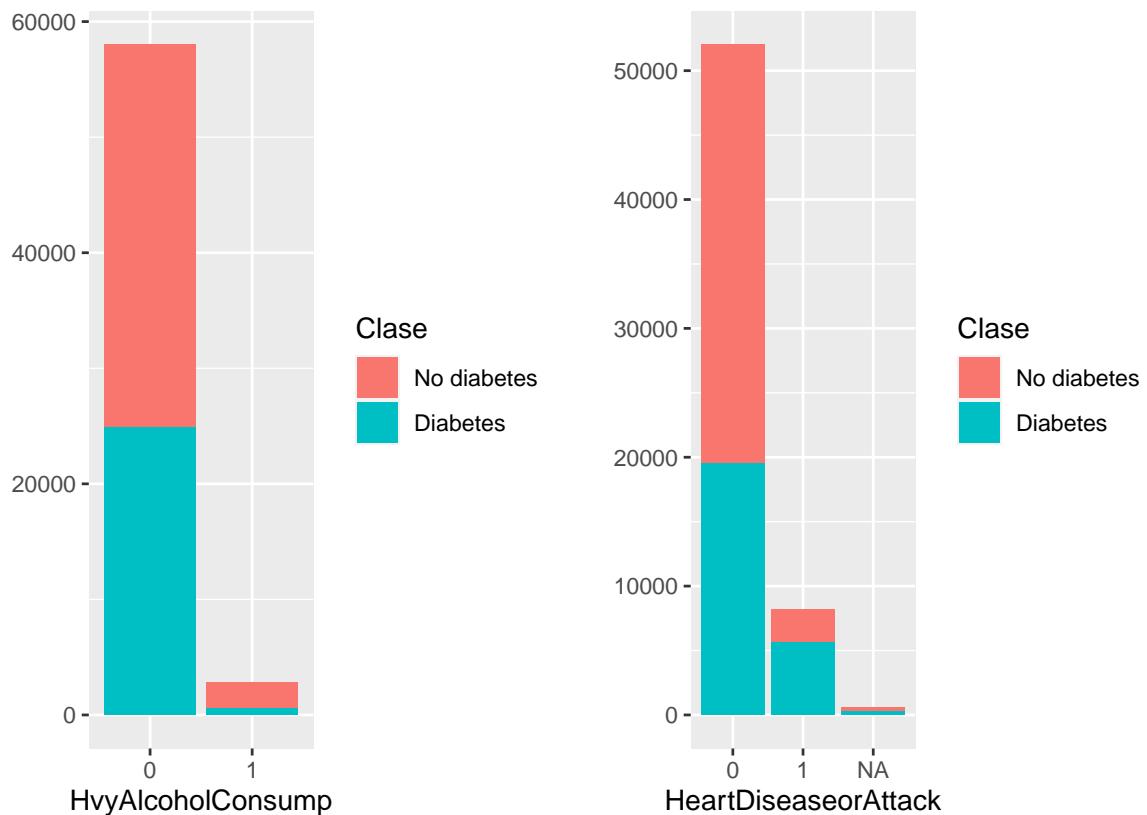
Aunque se han analizado varias variables, mostraremos gráficas relativas a la presencia o no de un alto nivel de colesterol (izquierda, *HighChol*) y el consumo diario de verduras u hortalizas (derecha, *Veggies*), al considerarlas de especial interés. Otras variables interesantes podrían ser *HvyAlcoholConsump* y/o *HeartDiseaseorAttack*, por lo que también se muestran.

En estas se pueden apreciar resultados que quizás pudieran parecer obvios. Existe una mayor proporción de población diabética si se tiene un alto nivel de colesterol, si no se comen verduras a diario, si se abusa del alcohol o si se ha tenido problemas cardíacos previamente. Nótese en la cuarta gráfica de barras que aún no hemos tratado los posibles valores nulos que puedan existir en los datos.

```
p1 <- ggplot(training_data) +
  geom_histogram(aes(x = as.factor(HighChol), fill = Label), stat = "count") +
  scale_fill_discrete(name ="Clase", labels=c("No diabetes", "Diabetes")) +
  labs(x = "HighChol", y = "")
p2 <- ggplot(training_data) +
  geom_histogram(aes(x = as.factor(Veggies), fill = Label), stat = "count") +
  scale_fill_discrete(name ="Clase", labels=c("No diabetes", "Diabetes")) +
  labs(x = "Veggies", y = "")
grid.arrange(p1, p2, ncol = 2)
```



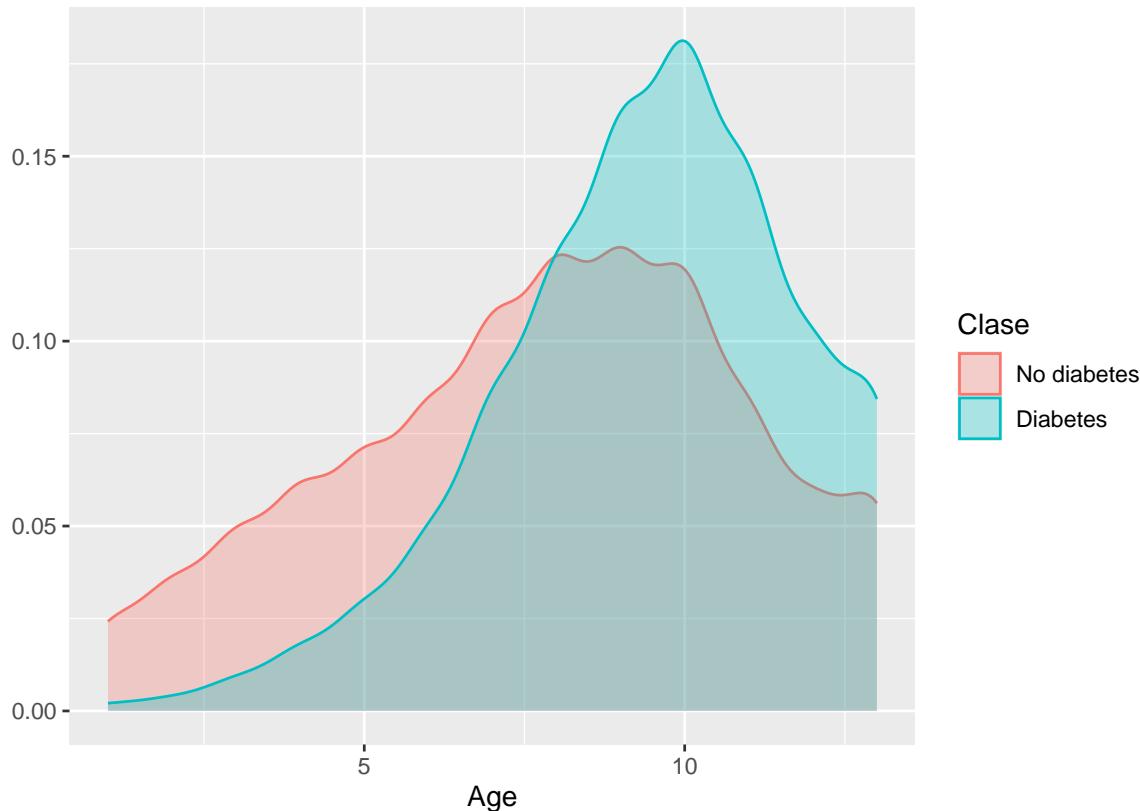
```
p3<- ggplot(training_data) +
  geom_histogram(aes(x = as.factor(HvyAlcoholConsump), fill = Label), stat = "count") +
  scale_fill_discrete(name ="Clase", labels=c("No diabetes", "Diabetes")) +
  labs(x = "HvyAlcoholConsump", y = "")
p4 <- ggplot(training_data) +
  geom_histogram(aes(x = as.factor(HeartDiseaseorAttack), fill = Label), stat = "count") +
  scale_fill_discrete(name ="Clase", labels=c("No diabetes", "Diabetes")) +
  labs(x = "HeartDiseaseorAttack", y = "")
grid.arrange(p3, p4, ncol = 2)
```



Por otra parte, se pueden generar las pseudo-distribuciones de probabilidades de las diferentes variables. Aunque durante el desarrollo de la práctica se han calculado varias de ellas, se mostrará únicamente la correspondiente a la variable *Age*, dada la facilidad de su interpretación.

Recordemos que las edades se encuentran representadas siguiendo la escala AGE5YR. Así pues, podemos ver como la cantidad de muestras que presentan diabetes crece enormemente a partir de los 45-59 años aproximadamente, alcanzando su valor más alto en la franja de los 65-69 años.

```
ggplot(training_data) +
  geom_density(bw = 0.5, aes(x = Age, fill = Label, color = Label),
              alpha = 0.3) + labs(x = "Age", y = "") +
  scale_fill_discrete(name = "Clase", labels=c("No diabetes", "Diabetes")) +
  scale_color_discrete(name = "Clase", labels=c("No diabetes", "Diabetes"))
```



### 3.4 DataExplorer

También podemos generar un informe del análisis exploratorio de los datos haciendo uso de la librería DataExplorer.

El fichero generado contiene bastante información útil, como estadísticas generales, histogramas y diagramas de barras de variables numéricas, gráficos Q-Q, análisis de correlación entre variables, análisis de componentes principales y muchos más.

También es posible crear las tablas y gráficos por separado, así que aprovecharemos esta funcionalidad y mostraremos los más relevantes en este documento.

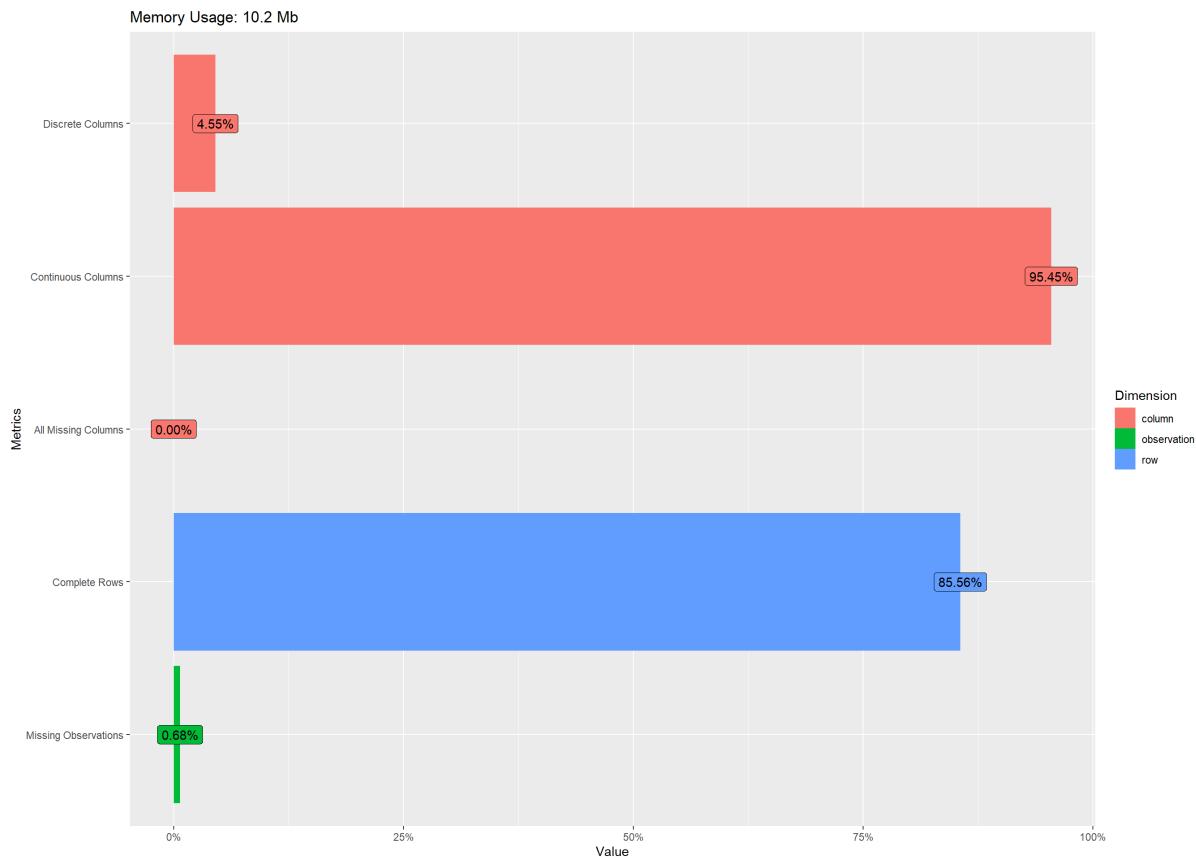
Comenzaremos por la información inicial del conjunto de datos y los valores perdidos que se poseen, la cual mostramos en forma de tabla y en formato gráfico.

```
introduce(training_data)
```

Nombre	Valor
Rows	60,796
Columns	22
Discrete columns	1
Continuous columns	21
All missing columns	0
Missing observations	9,145
Complete Rows	52,018
Total observations	1,337,512
Memory allocation	10.2 Mb

```
plot_intro(training_data)
```

Este paquete también permite generar un gráfico que nos muestra las variables con valores perdidos y su porcentaje. Este dato ya lo tratamos antes, y como podemos volver a ver, tenemos pocas variables con



valores perdidos, siendo la más destacable *NoDocbcCost* (con un nivel según el paquete OK). Podemos ver como la variable *HeartDiseaseOrAttack* posee valores nulos, como ya anticipaban las gráficas que mostrabamos anteriormente, aunque apenas tiene un 1% de valores perdidos.

```
plot_missing(training_data_raw)
```

A continuación, analizaremos más en profundidad las distintas variables que posee el conjunto de datos. Podemos ver como se distribuyen utilizando para ello histogramas y gráficos de barras, en función de si son variables numéricas o *booleanas*.

```
plot_histogram(training_data_raw)
plot_bar(training_data_raw)
```

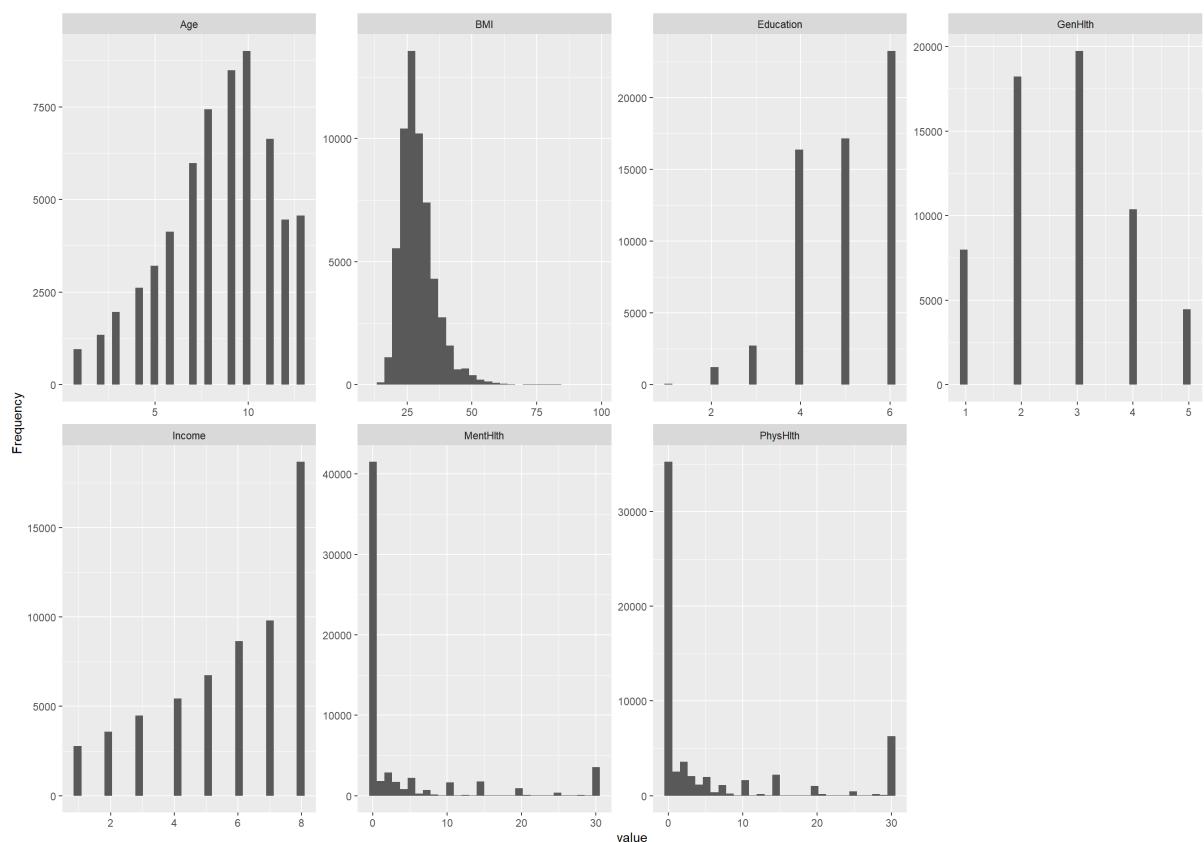
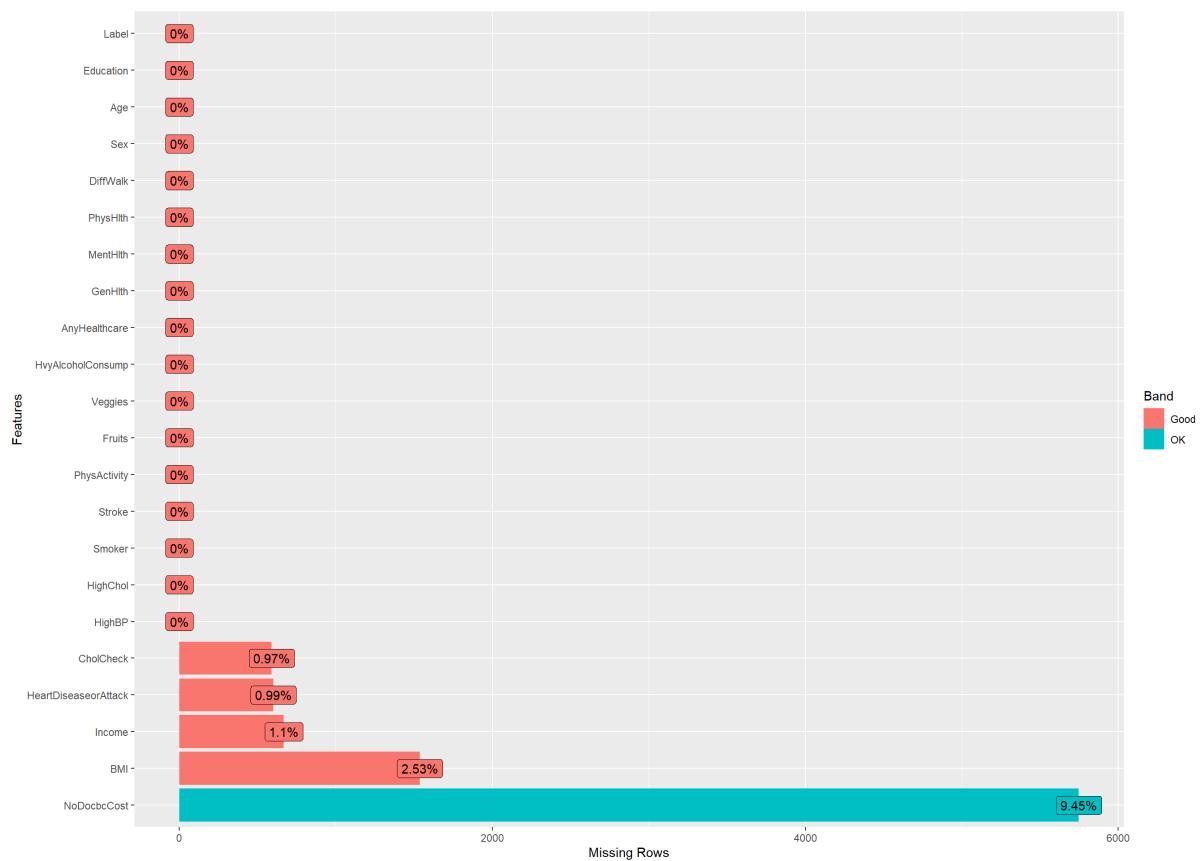
También podemos consultar gráficas Q-Q. Aunque se mostrarán las gráficas generadas para todas las variables, la interpretación de este tipo de gráficos sólo es útil en el caso de las variables numéricas (o al menos, aquellas no booleanas), como *Age*, *BMI*, *GenHealth*, *MentHlth*, *PhysHlth*, *Age*, *Education* e *Income*.

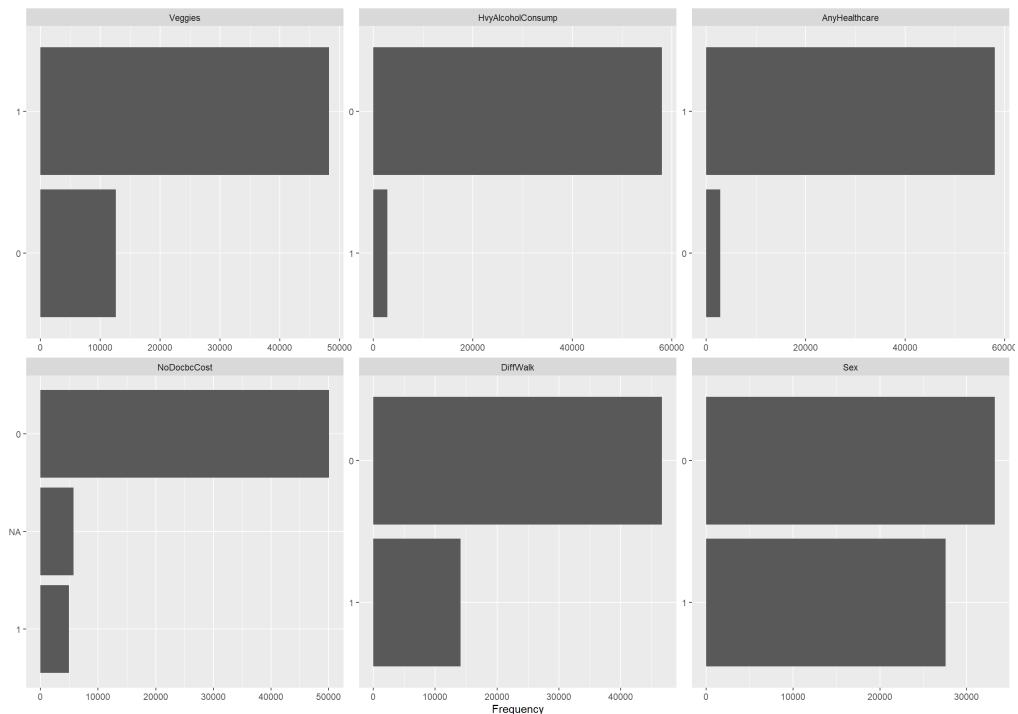
```
plot_qq(training_data_raw)
```

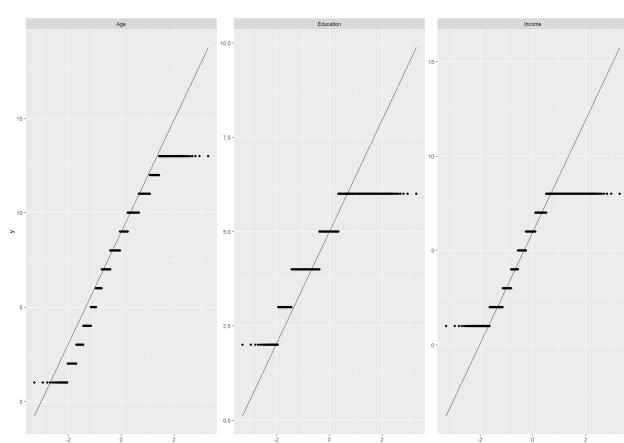
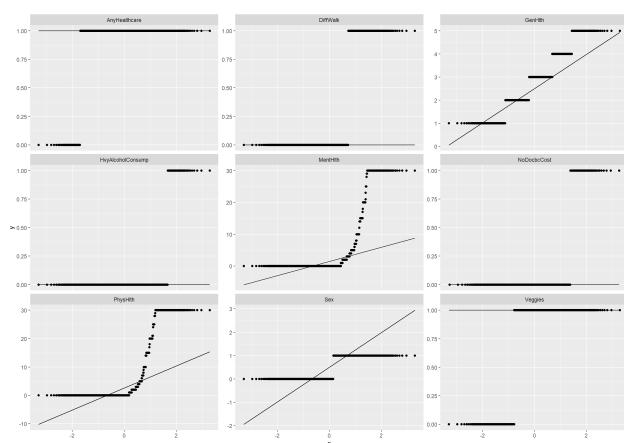
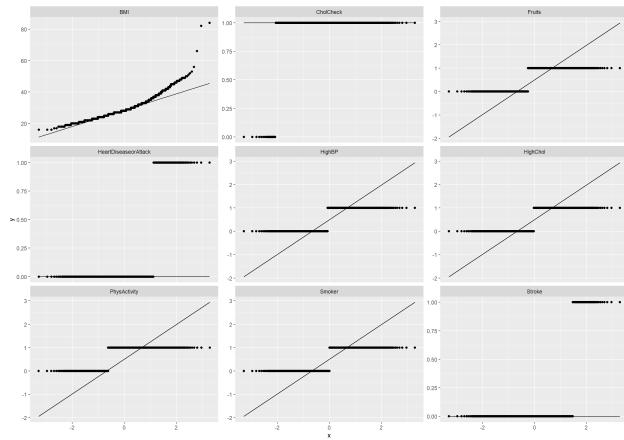
Por último, podemos observar la correlación de las diferentes variables utilizando una matriz de correlación coloreada convenientemente. No se adjunta la tabla con los valores numéricos dada su difícil lectura en un documento PDF, pero siempre se puede consultar en el código R adjunto en esta entrega. La figura mostrada ha sido rescatada del documento HTML generado anteriormente, dado que la figurada generada dentro de RStudio no tiene tanta calidad y su interpretación resulta algo más engorrosa.

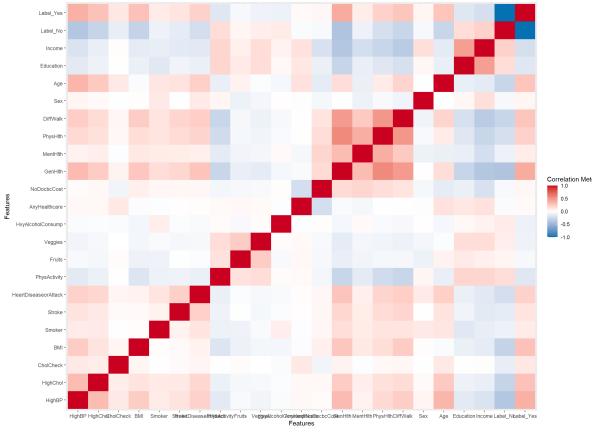
```
plot_correlation(training_data_raw)
correlation_matrix <- cor(training_data_raw, use = "complete.obs")
```

Podemos observar cierto grado de correlación entre variables primitivas (*GenHlth*) y variables aparentemente derivadas (*PhysHlth* y *MentHlth*); parejas de variables que representan conceptos relacionados (*PhysHlth* y *DiffWalk*, o *HighBP* y *HighChol*); o parejas de variables demográficas con valores habitualmente









relacionados (*Income* y *Education*). En cualquier caso, no parece detectarse correlación entre variables más allá de lo que uno podría esperar dado el sentido común, y aquellas correlladas no presentan un valor excesivamente alto (son menores a 0.6).

## 4 Procesamiento de datos.

El siguiente paso en este trabajo será realizar un pre-procesamiento de los datos a partir del conocimiento recabado durante el análisis exploratorio. En particular, trataremos los valores perdidos, intentaremos eliminar variables altamente correladas o que consideremos inútiles, buscaremos *outliers* y los eliminaremos. Por último, valoraremos la posibilidad de realizar discretización y normalización de las variables que no hayan sido ya objeto de estos tratamientos.

### 4.1 Tratamiento de valores perdidos.

En esta sección realizaremos el tratamiento de los valores perdidos.

Como pudimos observar durante el análisis exploratorio, tenemos cinco variables con valores nulos, *CholCheck*, *HearthDiseaseOrAttack*, *Income*, *BMI* y *NoDocbsCost*.

Uno de los pre-procesamientos más sencillos de realizar sería omitir los valores nulos, descartando aquellas observaciones en las que algunas de las variables presenta valores nulos. Podríamos fundamentar esta decisión en que, al no tener todos los datos necesarios para determinadas observaciones, estas no nos aportarían información fiable y completa con la que construir posteriormente el modelo de clasificación.

```
training_data_na.omit <- na.omit(training_data_raw)
```

De realizar este procesamiento, reduciríamos el número de observaciones a 52.018, de las 60.796 originales. Si analizamos en detalle cada una de las variables con valores nulos, podemos justificar la decisión de omitir observaciones con valores nulos.

Las variables *CholCheck* y *HearthDiseaseOrAttack* son binarias y su porcentaje de valores perdidos es muy pequeño, inferior al 1%, por lo que no perdemos gran cantidad de información. Realizar imputación por media o mediana no sería apropiado, pues llenar con valores decimales no tendría sentido al ser la variable binaria, y realizar imputación por moda podría desequilibrar aún más las clases. Algo similar sucede con la variable *NoDocbsCost*, en que el porcentaje de valores perdidos es cercano al 10%.

La variable *Income* es una variable ya discretizada bastante desbalanceada, pues existe una gran predominancia del valor 8 (ingresos superiores a 75.000\$ anuales). Realizar imputación con métodos simples podría acentuar más este desbalanceo de los datos.

La única variable puramente numérica con valores perdidos, y que por tanto podría ser sometida a un proceso de imputación por media o mediana, es *BMI* con cerca de 2.5% de valores perdidos.

Otra alternativa por la que podríamos optar es hacer uso de funciones que realicen imputaciones basándose en modelos matemáticos más complejos. En particular, podemos usar la función *complete* de la librería

*mice*. Esta función, por defecto, utiliza los coeficientes de regresión estimados en cada imputación para predecir los valores perdidos en el conjunto de datos completo.

```
imputed_data <- mice(training_data_raw)
imputed_data <- complete(imputed_data, action = 1)
```

Optaremos por esta versión del conjunto de datos con valores imputados de cara a las siguientes etapas del procesamiento de los datos, aunque en cierto momento recuperaremos el dataset en que se eliminaron los valores nulos.

## 4.2 Selección de características

Antes de realizar otras técnicas de procesado, realizaremos un proceso de selección de características. De esta manera, filtraremos características que se pueden deducir a partir de otras o características que, directamente, consideramos que no aportan información útil al futuro modelo de clasificación a construir.

En el apartado de análisis exploratorio observamos que no hay demasiadas características fuertemente correladas, siendo las más destacables *GenHealth* y sus derivadas *MentHlth* y *PhysHlth*. Una primera aproximación podría ser eliminar la característica padre y hacer uso únicamente de las derivadas, aunque de momento no tomaremos este enfoque.

Otro enfoque puede ser hacer uso de PCA (Principal Component Analysis), una técnica utilizada para reducir la dimensionalidad de un conjunto de datos mediante la identificación de patrones de covarianza entre las variables. En el informe generado por DataExplorer en el apartado anterior se incluían la variabilidad explicada si se usaban entre 1 y 14 variables, así como la importancia de cada característica original en las nuevas características generadas para el caso de PCA con 14 dimensiones. Estas tablas no se adjuntan en la memoria para facilitar su lectura, pero se pueden encontrar en la carpeta de imágenes.

Aunque esta técnica se ha probado bastante útil, no soy demasiado partidario de ella, pues las nuevas variables generadas son combinaciones lineales de las originales. De cara al futuro entrenamiento del modelo, siento que pierdo la capacidad de saber qué características del dataset original fueron utilizadas para tomar unas u otras decisiones en la clasificación. En casos como el de la medicina, resulta especialmente útil saber qué características e indicadores fundamentan una decisión, de manera que un doctor o personal experto pueda corroborar el “razonamiento” seguido por el algoritmo, matizarlo o descartar su conclusión.

Optaré por entrenar un modelo lineal y observar qué variables determina el algoritmo que son necesarias para la construcción del modelo, descartando el resto. Para este proceso haremos uso del dataset imputado. A modo de curiosidad y experimentación se probó a entrenar el modelo con el dataset en el que los valores nulos habían sido descartados, obteniendo el mismo subconjunto de características.

```
model_na <- lm(as.numeric(Diabetes_binary) ~ ., data = training_data_na.omit)
k_na <- ols_step_both_p(model_na)
summary(k_na$model)

model_imputed <- lm(as.numeric(Diabetes_binary) ~ ., data = imputed_data)
k_imputed <- ols_step_both_p(model_imputed)
summary(k_imputed$model)

##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.48752 -0.32805 -0.03633  0.35666  1.26693
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -0.5922304  0.0182316 -32.484 < 2e-16 ***
## HighBP                0.1476247  0.0039370  37.496 < 2e-16 ***
```

```

## HighChol          0.1045581  0.0036550  28.606 < 2e-16 ***
## BMI              0.0118114  0.0002586  45.680 < 2e-16 ***
## HeartDiseaseorAttack 0.0646434  0.0053561 12.069 < 2e-16 ***
## PhysActivity     -0.0094990  0.0040052 -2.372 0.017712 *
## GenHlth           0.0968873  0.0020763  46.664 < 2e-16 ***
## PhysHlth          -0.0013747  0.0002241 -6.135 8.54e-10 ***
## DiffWalk          0.0381297  0.0050197  7.596 3.10e-14 ***
## Age               0.0222449  0.0006617 33.618 < 2e-16 ***
## Education         -0.0043059  0.0019005 -2.266 0.023477 *
## Income            -0.0091591  0.0009627 -9.514 < 2e-16 ***
## HvyAlcoholConsump -0.1124443  0.0080761 -13.923 < 2e-16 ***
## CholCheck          0.1377717  0.0104204 13.221 < 2e-16 ***
## Sex                0.0372245  0.0034806 10.695 < 2e-16 ***
## Stroke             0.0326579  0.0075526  4.324 1.53e-05 ***
## Veggies            -0.0153077  0.0042703 -3.585 0.000338 ***
## MentHlth           -0.0007558  0.0002348 -3.220 0.001284 **
## AnyHealthcare      0.0151122  0.0082968  1.821 0.068545 .

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4145 on 60777 degrees of freedom
## Multiple R-squared:  0.2943, Adjusted R-squared:  0.2941
## F-statistic:  1408 on 18 and 60777 DF,  p-value: < 2.2e-16

```

Tras este proceso descartaremos tres características, *Smoker*, *Fruits* y *NoDocbcCost*.

```

imputed_data_reduced <- imputed_data
imputed_data_reduced[, "Smoker"] <- NULL
imputed_data_reduced[, "Fruits"] <- NULL
imputed_data_reduced[, "NoDocbcCost"] <- NULL

```

Para analizar que subconjunto de características es realmente relevante podríamos haber usado otras técnicas, como análisis del factor de inflación de la varianza (VIF-Analysis), tests ANOVA o chi-square, etc.

### 4.3 Detección y tratamiento de outliers

En nuestro conjunto de datos pueden existir valores anómalos que distorsionan la distribución y parámetros estadísticos de los datos, dificultando el futuro aprendizaje de un modelo de clasificación. Estos son los conocidos como *outliers*, los cuáles deberíamos eliminar de nuestro conjunto de datos.

En el dataset proporcionado la eliminación de *outliers* es compleja, dado que para su detección se suelen utilizar variables numéricas, y en este caso solo *BMI*, *MentHlth* y *PhysHlth* son puramente numéricas. Intentar detectar y eliminar *outliers* basándonos únicamente en estas variables (por separado o de manera conjunta) puede producir resultados poco realistas, así que seguiremos un enfoque multi-variable para todas las características del dataset.

El siguiente código genera un modelo de regresión y computa la influencia de cada observación en la salida a partir del cálculo de la llamada distancia de Cook. Aquellas observaciones con mayor influencia, o que se encuentren por encima de un determinado umbral (normalmente cuatro veces la media), se consideran posibles *outliers* y serían susceptibles de ser eliminados.

```

imputed_data_reduced$Diabetes_binary <- as.factor(imputed_data_reduced$Diabetes_binary)

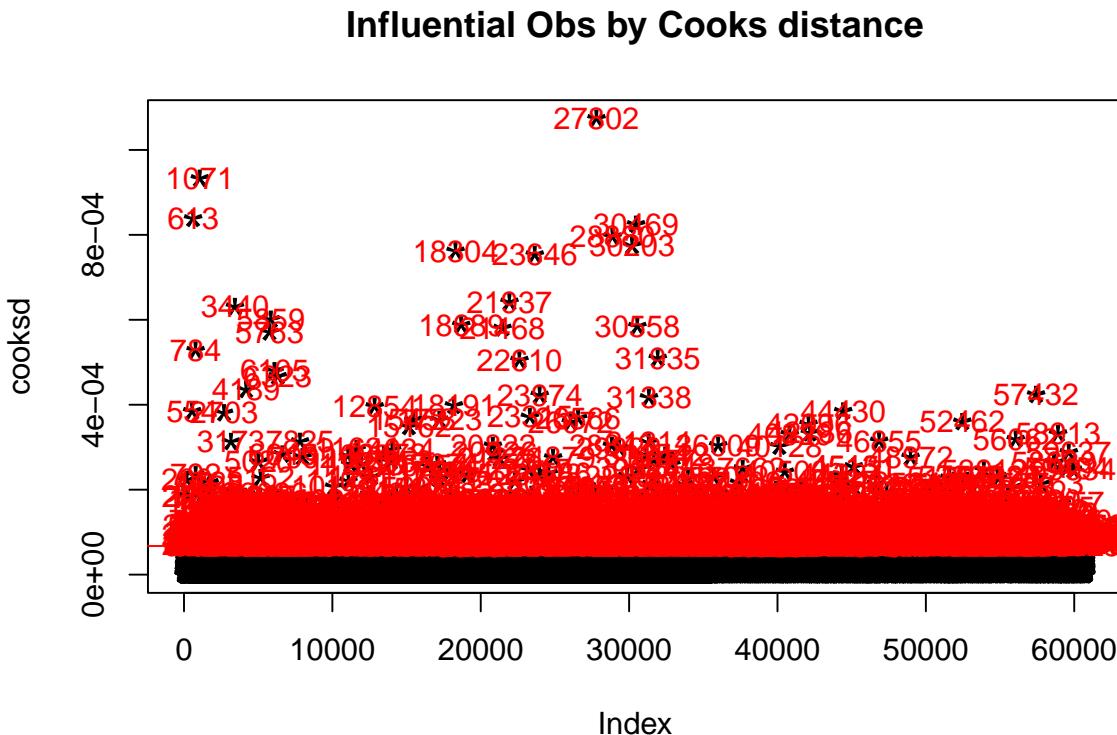
mod <- lm(as.numeric(Diabetes_binary)~., data=imputed_data_reduced)

cooksd <- cooks.distance(mod)

plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")
abline(h = 4*mean(cooksd, na.rm=T), col="red")
text(x=1:length(cooksd)+1, y=cooksd,

```

```
labels=ifelse(cooksd>4*mean(cooksd, na.rm=T),
             names(cooksd), ""),
             col="red")
```



```
influential <- as.numeric(names(cooksd)[(cooksd > 4*mean(cooksd, na.rm=T))])
imputed_data_red_nout <- imputed_data_reduced
imputed_data_red_nout[influential, ] # influential observations that we are gonna delete.
imputed_data_red_nout <- imputed_data_red_nout[-influential,]
```

Tras este procesamiento, mantenemos 58.423 observaciones de las 60.796 disponibles en el conjunto de datos original. Hemos eliminado algo menos del 4% de las observaciones originales. En el gráfico anterior se pueden observar como existen ciertas observaciones que “se separan” mucho del resto, estos son los outliers que hemos decidido eliminar.

#### 4.4 Discretización de variables

Otro procesamiento que se puede realizar para intentar mejorar la calidad de futuros clasificadores es la discretización de variables.

De realizar este proceso, sólo variables puramente numéricas deberían participar en él, por lo que, al igual que antes, sólo podríamos aplicarlo sobre las variables *BMI*, *MentHlth* y *PhysHlth*. Analicemos la viabilidad de este proceso.

*MentHlt* y *PhysHlth* no poseen demasiados valores únicos y más del 80% de las observaciones tienen un mismo valor (0), por lo que los intervalos estarían altamente descompensados y podríamos llegar a perder información. *BMI* podría ser una variable candidata a ser discretizada dada la mayor variabilidad de sus valores.

Para el caso de *BMI*, y tras haber consultado información adicional respecto de los valores normales que suele tomar esta variable, definiremos cinco intervalos: *Underweight* [0-18.5), *Normal* [18.5-25), *Overweight* [25, 30), *Obese* [30, 40) y *Severe Obese* [40, 100].

```
imputed_data_disc <- imputed_data_red_nout
intervalos_BMI <- c(0, 18.5, 25, 30, 40, 100)
labels_BMI <- c("Underweight", "Normal", "Overweight", "Obese", "SevereObese")
```

```
imputed_data_disc$BMI <- cut(imputed_data_disc$BMI,
                                breaks = intervalos_BMI, labels = labels_BMI)
```

Recordemos que variables como *Age*, *Income* o *Education* ya se encontraban discretizadas en el conjunto de datos original, aunque las etiquetas asignadas eran valores numéricos cuya interpretación estaba supeditadas a estándares americanos (como AGE5YR, INCOME2 o EDUCA), y no etiquetas textuales en sí mismas. Podríamos convertir dichos valores a etiquetas de texto con el siguiente código:

```
imputed_data_disc$Age <- cut(imputed_data_disc$Age,
                                breaks = c(0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50,
                                           , 55, 60, 65, 70, 75, 80, Inf),
                                labels = c("0-4", "5-9", "10-14", "15-19", "20-24",
                                           "25-29", "30-34", "35-39", "40-44",
                                           "45-49", "50-54", "55-59", "60-64",
                                           "65-69", "70-74", "75-79", "80+"))
imputed_data_disc$Education <- cut(imputed_data_disc$Education,
                                      breaks=c(0, 1, 2, 3, 4, 5, 6),
                                      labels=c("Never attended", "Elementary",
                                              "Some highschool",
                                              "Graduate highschool", "Some college"
                                              , "Graduate college"))
imputed_data_disc$Income <- cut(imputed_data_disc$Income,
                                   breaks=c(0, 1, 2, 3, 4, 5, 6, 7, 8),
                                   labels=c("<10.000$", "10.000-15.000$",
                                           "15.000-20.000$", "20.000-25.000$",
                                           "25.000-35.000$", "35.000-50.000$",
                                           "50.000-75.000$", ">75.000$"))
imputed_data_disc$GenHlth <- cut(imputed_data_disc$GenHlth,
                                     breaks=c(0, 1, 2, 3, 4, 5),
                                     labels=c("Excellent", "Very Good", "Good",
                                             "Fair", "Poor"))
```

Anteriormente no se había realizado este cambio dado que algoritmos como el de imputación sufren penalizaciones de rendimiento importantes al trabajar con datos de tipo texto en lugar de datos numéricos. Aunque parece una decisión menor y se podría pensar que este cambio solo ayuda a leer los datos al usuario, veremos posteriormente como, en función del algoritmo de clasificación, el uso de valores numéricos o etiquetas textuales puede proporcionar resultados muy distintos.

## 4.5 Normalización de variables

Otra de las técnicas de las que haremos uso consiste en la normalización de las variables numéricas dentro de unos límites más reducidos y trasladando sus valores. Haremos uso de la normalización de tipo *min\_max*, la cual establece entre límites de 0 y 1 todos los valores, siendo el menor de ellos el 0 y el mayor el 1.

En primer lugar, crearemos una función muy simple que implemente esta normalización *min\_max*.

```
normalize <- function(x) {
  (x -min(x)) / (max(x) - min(x))
}
```

A continuación se normalizan las variables puramente numéricas restantes, *MentHlth* y *PhysHlth*.

```
imputed_data_final <- imputed_data_disc
imputed_data_final$MentHlth <- normalize(imputed_data_final$MentHlth)
imputed_data_final$PhysHlth <- normalize(imputed_data_final$PhysHlth)
```

Con esta normalización damos por finalizado el pre-procesamiento de los datos. Pasemos a utilizar el conjunto de datos resultante para entrenar varios modelos de clasificación, examinar los resultados obtenidos y extraer conclusiones.

## 5 Clasificación con varios modelos

Tras haber pre-procesado los datos, emplearemos varios algoritmos de clasificación para generar diferentes modelos y comprobar la bondad de las predicciones realizadas por cada uno de ellos. Durante toda esta sección se hará uso del dataset pre-procesado anteriormente, en el que no se ha aplicado el cambio de etiqueta numérica a textual en las variables ya discretizadas inicialmente.

Antes de nada, deberemos realizar una partición de nuestros datos, de manera que dispongamos de un conjunto de entrenamiento y otro de validación. Para ello, haremos uso de la librería caret, la cual nos permite realizar validación cruzada mediante el controlador *trainControl*. Dedicaremos un 80% de los datos al entrenamiento de los diferentes modelos y el 20% restante a la validación. Se ha decidido repetir la validación cruzada 5 veces; en actividades anteriores se utilizaban 10 iteraciones, pero la gran cantidad de datos y el elevado tiempo de procesamiento me obliga a rebajar este parámetro.

```
set.seed(0)
imputed_data_final$Diabetes_binary <- factor(imputed_data_final$Diabetes_binary,
                                                labels = c("No", "Yes"))
trainIndex <- createDataPartition(imputed_data_final$Diabetes_binary,
                                    p = .8, list = FALSE)
training_set <- imputed_data_final[trainIndex, ]
validation_set <- imputed_data_final[-trainIndex, ]

trClassCtrl <- trainControl(classProbs = TRUE,
                             summaryFunction = twoClassSummary,
                             method = "cv", number = 5,
                             verboseIter=FALSE)
```

En particular, se usarán los siguientes algoritmos de clasificación y aprendizaje automático.

- **Árboles de decisión.** Se ha escogido este modelo ya que es uno de los modelos más sencillos y comúnmente utilizados a la hora de construir modelos de clasificación. Usaremos la versión proporcionada por la función *rpart*, del paquete homónimo, que hace uso de una variación del algoritmo CART (desarrollado en 1984). Podríamos optar por otros modelos de árboles de decisión, tales como C4.5 o C5.0. De hecho, inicialmente se optó por usar C5.0, ya que prometía mejores precisiones y menores tiempos de ejecución, pero parece existir algún tipo de error por el que la ejecución se demoraba infinitamente y colapsaba mi sistema.
- **Random Forest.** Se decide utilizar este modelo ya que su salida es el resultado de una “votación” entre multitud de árboles de decisión construidos de diferente forma. Puede resultar interesante compararlo con las decisiones tomadas por un único árbol, ya que se presupone que el modelo de *Random Forest* proporciona una mayor adaptabilidad a los cambios y por tanto una mayor precisión.
- **Naive Bayes.** Se ha escogido este modelo ya que se trata de uno de los más sencillos y a la vez más conocidos dentro del marco de los modelos probabilísticos. Puede proporcionarnos otro punto de vista, por lo que su utilización puede ser interesante. En el apartado correspondiente se justificará más en detalle su elección.
- **Redes neuronales artificiales.** Este modelo, de los más potentes y más utilizados en el marco de la inteligencia artificial, promete proporcionar buenos resultados. Gracias al uso de rejillas de parámetros se pueden utilizar diferentes combinaciones de hiperparámetros, facilitando la búsqueda del modelo que mejores resultados proporcione.

Nótese que podrían haberse utilizado otros modelos como *K-Nearest Neighbors (KNN)*. No se emplea este modelo ya que no esperamos grandes resultados del mismo. Este algoritmo se fundamenta en el cálculo de distancias, lo cual no tiene demasiado interés en un problema en el que la mayoría de variables son binarias.

### 5.1 Árboles de decisión.

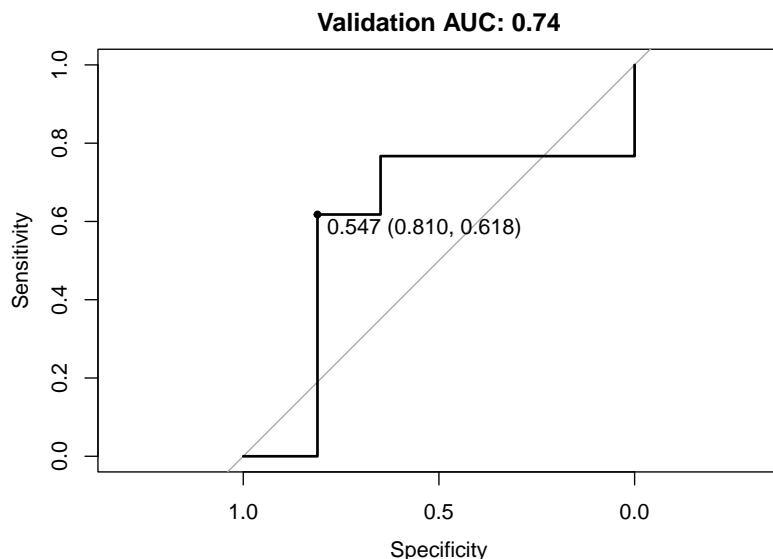
El primer modelo de clasificación del que haremos uso será el basado en árboles de decisión. Dado un conjunto de datos, el algoritmo divide repetidamente este conjunto en subconjuntos más pequeños

en función del valor de las diferentes variables (TRUE o FALSE para binarias, mayor o menor a un valor umbral para variables numéricas). Esto se repite hasta alcanzar un criterio de detención, como profundidad máxima o número de muestras mínimas por hoja. Construido el modelo, se pueden clasificar nuevas observaciones al seguir estas las ramas del árbol en función de las características oportunas.

```
# Crear modelo
modelo_rpart <- train(Diabetes_binary ~ .,
                        data = training_set,
                        method = "rpart",
                        metric = "ROC",
                        trControl = trClassCtrl)

# Generar predicciones
predictionValidationProb <- predict(modelo_rpart, validation_set, type = "prob")
predictionValidation <- predict(modelo_rpart, validation_set)

# Conseguir AUC en validación
auc_rpart <- roc(validation_set$Diabetes_binary,
                   predictionValidationProb[["Yes"]],
                   levels = unique(validation_set[["Diabetes_binary"]]))
roc_validation <- plot.roc(auc_rpart, ylim=c(0,1), type = "S" , print.thres = T,
                           main=paste('Validation AUC: ',
                                      round(auc_rpart$auc[[1]], 2)))
```



```
# Conseguir precisión y matriz de confusión en validación
acc_rpart <- confusionMatrix(predictionValidation,
                               validation_set$Diabetes_binary)
acc_rpart$overall["Accuracy"]

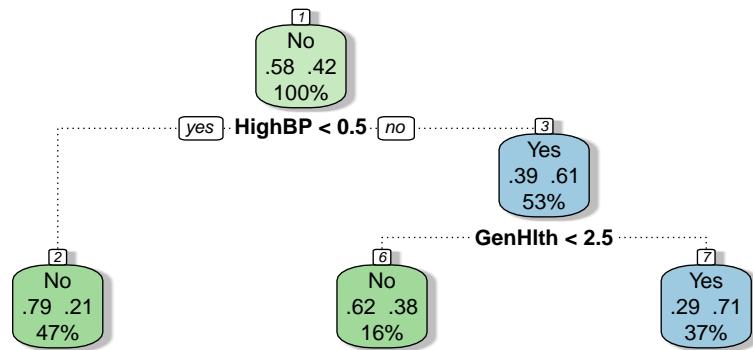
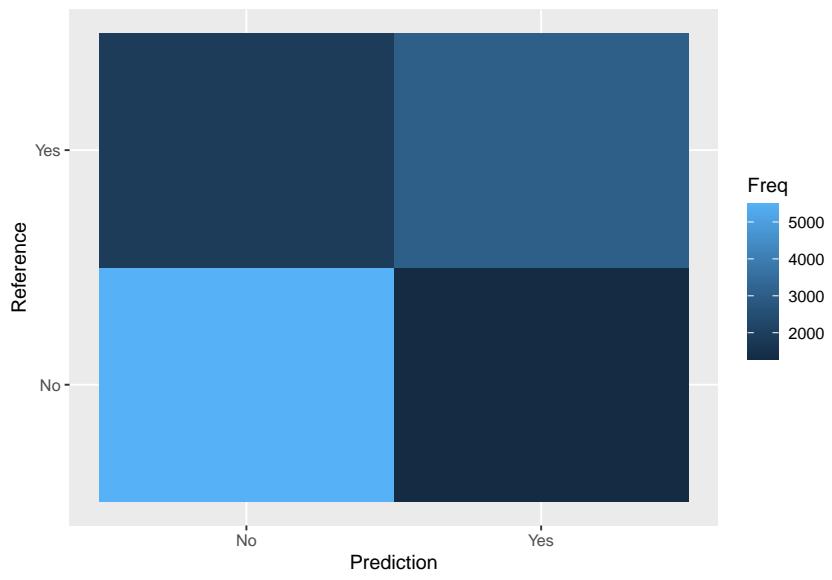
## Accuracy
## 0.7321979

matrix_table <- data.frame(acc_rpart$table)
ggplot(matrix_table, aes(x=Prediction, y=Reference, fill=Freq)) + geom_tile()

fancyRpartPlot(modelo_rpart$finalModel)
```

Este modelo, el más simple de cuantos probaremos, parece obtener unos resultados aceptables al proporcionar una precisión ligeramente superior al 73.2%. El valor del área bajo la curva es de 0.745 aproximadamente.

Con la función *fancyRpartPlot* se puede representar gráficamente parte del árbol. Gracias a esta



Rattle 2023-abr.-11 22:38:00 Usuario

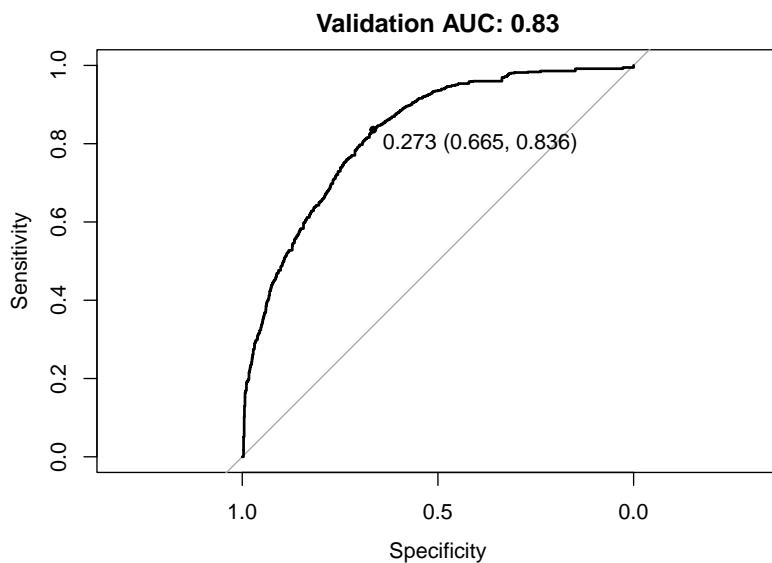
representación podemos observar que los factores, según el algoritmo, más determinantes para predecir la diabetes son la presencia de alta presión arterial y si el propio paciente considera que tiene buena salud. En otras ejecuciones se ha empleado si se ha controlado el colesterol y las dificultades para andar.

A continuación, implementaremos una variación del árbol de decisión anteriormente empleado, en la que se explora el árbol completo, con todas sus ramas. Este cambio podría proporcionar un mejor nivel en las predicciones realizadas.

```
# Rejilla de parámetros
rpartParametersGrid <- expand.grid(.cp = -1)
# Definir modelo
modelo_rpart_mejorado <- train(Diabetes_binary ~ .,
                                    data = training_set,
                                    method = "rpart",
                                    metric = "ROC",
                                    trControl = trClassCtrl,
                                    tuneGrid = rpartParametersGrid)

# Generar predicciones
predictionValidationProb <- predict(modelo_rpart_mejorado, validation_set,
                                       type = "prob")
predictionValidation <- predict(modelo_rpart_mejorado, validation_set)

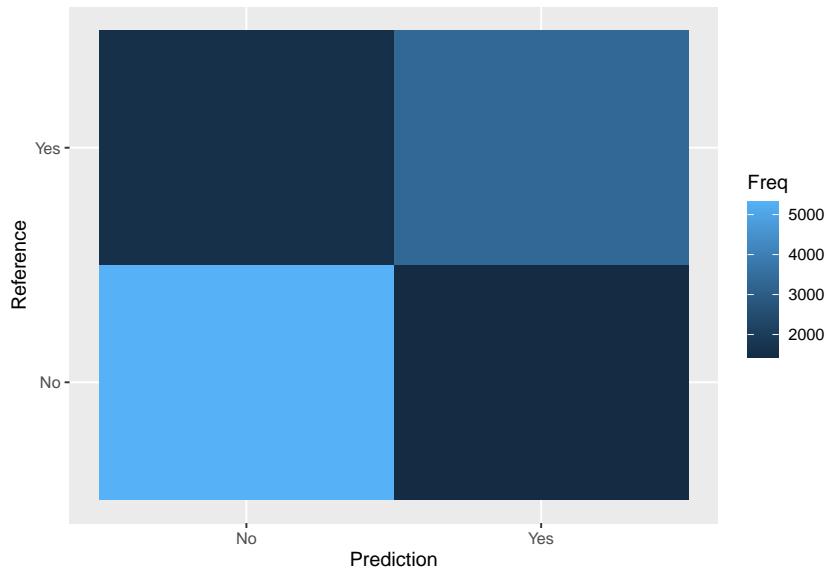
# Get AUC by ROC curve
auc_rpart_mejorado <- roc(validation_set$Diabetes_binary,
                            predictionValidationProb[["Yes"]],
                            levels = unique(validation_set[["Diabetes_binary"]]))
roc_validation <- plot.roc(auc_rpart_mejorado, ylim=c(0,1), type = "S" ,
                           print.thres = T, main=paste('Validation AUC:', round(auc_rpart_mejorado$auc[[1]], 2)))
```



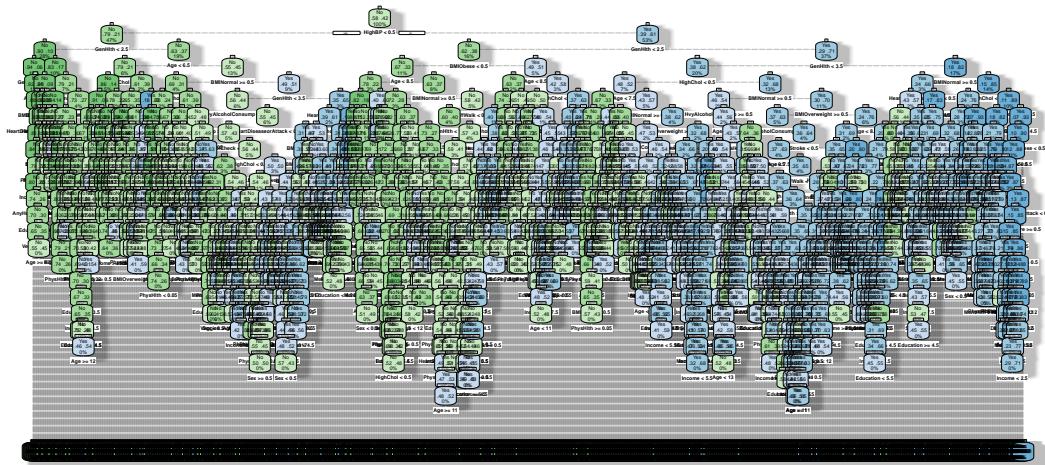
```
# Conseguir precisión y matriz de confusión en validación
acc_rpart_mejorado <- confusionMatrix(predictionValidation,
                                         validation_set$Diabetes_binary)
acc_rpart_mejorado$overall["Accuracy"]

## Accuracy
## 0.7423827

matrix_table <- data.frame(acc_rpart_mejorado$table)
ggplot(matrix_table, aes(x=Prediction, y=Reference, fill=Freq)) + geom_tile()
```



```
fancyRpartPlot(modelo_rpart_mejorado$finalModel)
```



Rattle 2023-abr.-11 22:38:07 Usuario

El hecho de explorar el árbol completo provoca una mejora en el nivel de acierto de las predicciones cercana al 1%, y una mejora cercana a 0.085 en el área bajo la curva. A título informativo se adjunta la representación gráfica del árbol completo; aunque no se pueda leer que variable determina el uso de una u otra rama, sirve para ilustrar la gran complejidad que puede adquirir el árbol generado.

## 5.2 Random Forest

El segundo modelo de clasificación que vamos a utilizar es el llamado *Random Forest*. Este algoritmo genera varios árboles de decisión construidos de diferente manera, de forma que sus ramas son distintas

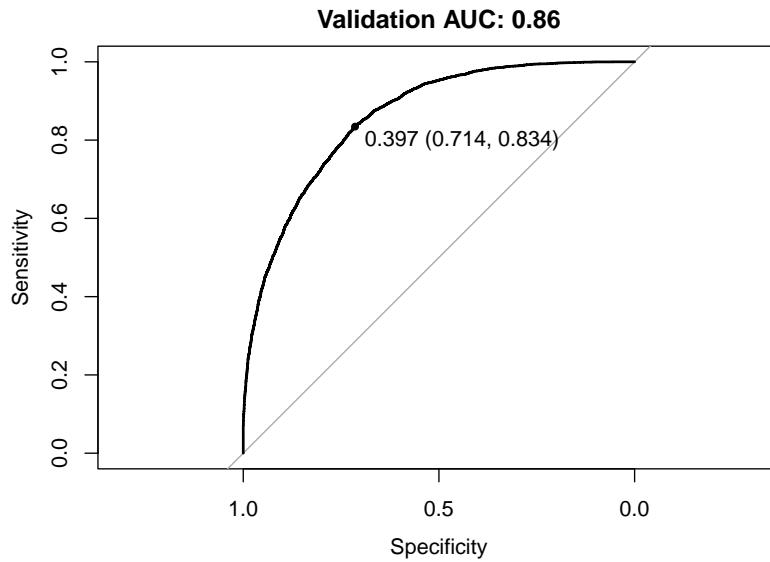
entre sí. Así pues, cuando se le proporciona una nueva observación al bosque, este devuelve el resultado de realizar una votación o promedio entre los resultados de cada uno de los árboles. De esta manera podemos tener en cuenta “diferentes puntos de vista”, en un intento de mejorar la precisión de las predicciones y reducir el sobre-ajuste al conjunto de entrenamiento.

A diferencia de las tareas entregadas anteriormente, se ha decidido utilizar la implementación disponible en la librería ranger, dado que esta promete ser más eficiente y proporcionar mejores resultados.

```
# Parámetros algoritmo
rfParametersGrid <- data.frame(
  mtry = sqrt(ncol(training_set)),
  splitrule = "gini",
  min.node.size = c(10, 20)
)
# Define modelo
modelo_rf <- train(Diabetes_binary ~ .,
                      data = training_set,
                      method = "ranger",
                      metric = "ROC",
                      trControl = trClassCtrl,
                      tuneGrid = rfParametersGrid)

# Generar predicciones
predictionValidationProb <- predict(modelo_rf, validation_set, type = "prob")
predictionValidation <- predict(modelo_rf, validation_set)

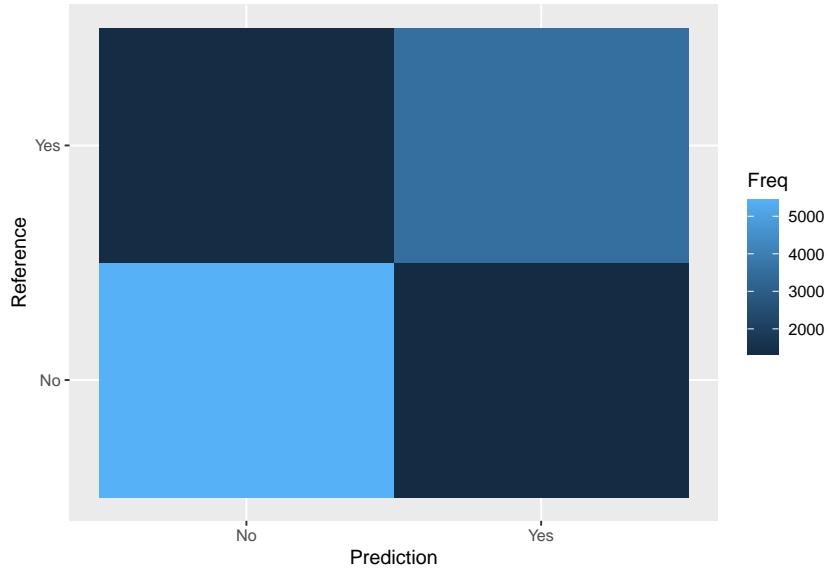
# Conseguir AUC en validación
auc_rf <- roc(validation_set$Diabetes_binary, predictionValidationProb[["Yes"]],
               levels = unique(validation_set[["Diabetes_binary"]]))
roc_validation <- plot.roc(auc_rf, ylim=c(0,1), type = "S" , print.thres = T,
                           main=paste('Validation AUC:',
                                      round(auc_rf$auc[[1]], 2)))
```



```
# Conseguir precisión y matriz de confusión en validación
acc_rf <- confusionMatrix(predictionValidation, validation_set$Diabetes_binary)
acc_rf$overall["Accuracy"]

## Accuracy
## 0.7692571
```

```
matrix_table <- data.frame(acc_rf$table)
ggplot(matrix_table, aes(x=Prediction, y=Reference, fill=Freq)) + geom_tile()
```



En esta ocasión, el modelo proporciona una precisión del 76.9%, mientras que el área bajo la curva es de 0.856. Haciendo uso de este modelo conseguimos una mejora en la precisión de las predicciones cercana al 3.7% y 2.7% respecto de los árboles anteriores, mientras que el área bajo la curva mejora en 0.11 y 0.026 aproximada y respectivamente.

### 5.3 Naive Bayes

El tercer modelo que vamos a utilizar es *Naive Bayes*, un modelo probabilístico que utiliza la probabilidad condicionada de cada atributo para calcular la probabilidad de que una observación pertenezca a una determinada clase.

Decidimos emplear este algoritmo dado que, bajo la suposición de que los atributos son independientes entre sí, suele funcionar bastante bien. Como vimos durante el análisis exploratorio, no existe demasiada correlación entre las distintas variables, salvo en un par de casos, por lo que podemos creer que este modelo puede proporcionar un buen resultado.

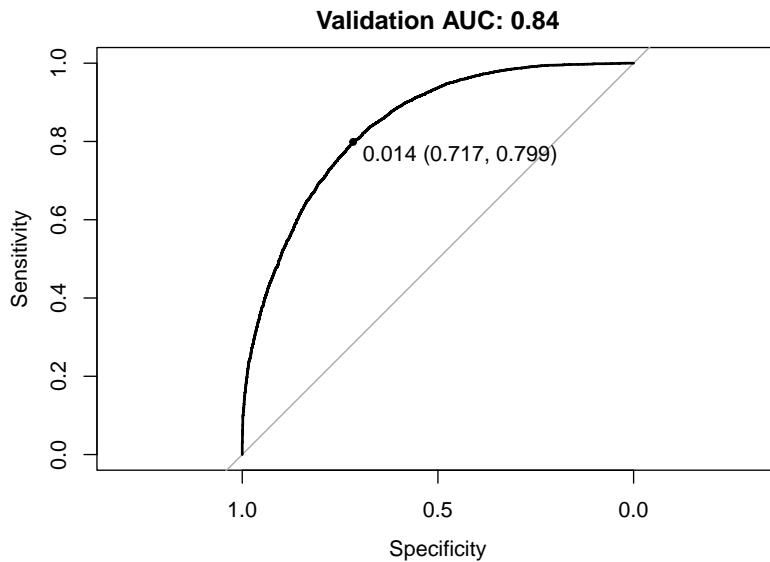
```
# Crear modelo
modelo_nb <- train(Diabetes_binary ~ ., data = training_set,
                      method = "naive_bayes", metric = "ROC",
                      trControl = trClassCtrl)

# Generar predicciones
predictionValidationProb <- predict(modelo_nb, validation_set, type = "prob")
predictionValidation <- predict(modelo_nb, validation_set)

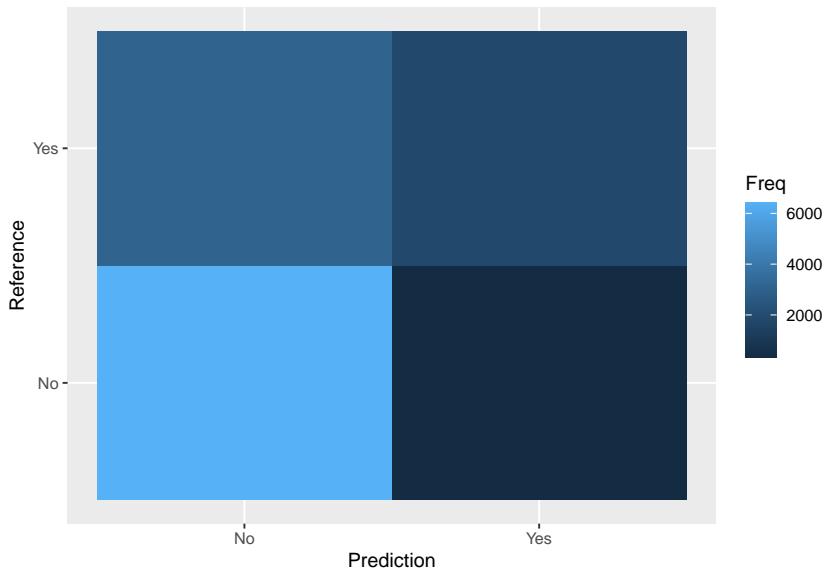
# Conseguir AUC en validación
auc_nb <- roc(validation_set$Diabetes_binary, predictionValidationProb[["Yes"]],
                levels = unique(validation_set[["Diabetes_binary"]]))
roc_validation <- plot.roc(auc_nb, ylim=c(0,1), type = "S" , print.thres = T,
                           main=paste('Validation AUC:',
                                      round(auc_nb$auc[[1]], 2)))

# Conseguir precisión y matriz de confusión en validación
acc_nb <- confusionMatrix(predictionValidation, validation_set$Diabetes_binary)
acc_nb$overall["Accuracy"]

## Accuracy
```



```
## 0.7075488
matrix_table <- data.frame(acc_nb$table)
ggplot(matrix_table, aes(x=Prediction, y=Reference, fill=Freq)) + geom_tile()
```



De usar la versión del conjunto de datos en que las variables discretizadas por defecto tenían valores numéricos (normalizados), la precisión es cercana al 70.75%. Sin embargo, de utilizar etiquetas de texto en su lugar, la precisión disminuye hasta un 60% aproximadamente. Por otra parte, el valor del área bajo la curva se mantiene prácticamente idéntico, oscilando en  $\pm 1\%$  de media aproximadamente.

## 5.4 Redes neuronales artificiales

Finalmente, haremos uso de redes neuronales artificiales. Este modelo, de forma muy resumida y simple, recibe en las neuronas de la capa de entrada una serie de datos, que procesará a través de diferentes capas ocultas (previamente definidas) de la red, ajustando una serie de pesos en el proceso. En la capa de salida se proporciona la predicción realizada para cada observación. El modelo generado estará conformado por los pesos que fueron calculados durante el proceso de aprendizaje automático descrito anteriormente.

```
# Definir modelo y evitar salida de las iteraciones
silent <- capture.output(modelo_rna <- train(Diabetes_binary ~ .,
```

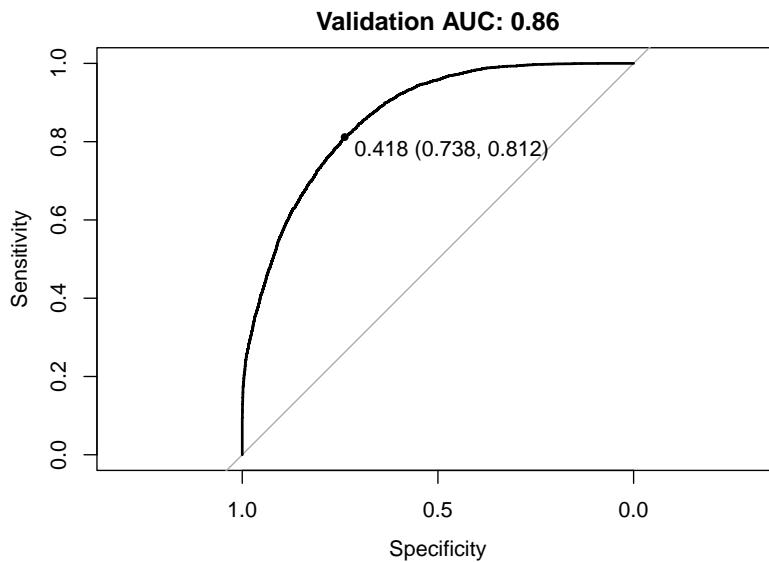
```

        data = training_set,
        method = "nnet",
        metric = "ROC",
        trControl = trClassCtrl))

# Generar predicciones
predictionValidationProb <- predict(modelo_rna, validation_set, type = "prob")
predictionValidation <- predict(modelo_rna, validation_set)

# Conseguir AUC en validación
auc_rna <- roc(validation_set$Diabetes_binary, predictionValidationProb[["Yes"]],
                 levels = unique(validation_set[["Diabetes_binary"]]))
roc_validation <- plot.roc(auc_rna, ylim=c(0,1), type = "S" , print.thres = T,
                           main=paste('Validation AUC:',
                                      round(auc_rna$auc[[1]], 2)))

```



```

# Conseguir precisión y matriz de confusión en validación
acc_rna <- confusionMatrix(predictionValidation, validation_set$Diabetes_binary)
acc_rna$overall["Accuracy"]

## Accuracy
## 0.7720229

matrix_table <- data.frame(acc_rna$table)
ggplot(matrix_table, aes(x=Prediction, y=Reference, fill=Freq)) + geom_tile()

```

En esta ocasión, la precisión mejora ligeramente, hasta alcanzar un 77.2% aproximadamente; el área bajo la curva es 0.861. Cabe mencionar que el tiempo de ejecución de este modelo ha sido considerablemente superior a los empleados anteriormente, más sencillos.

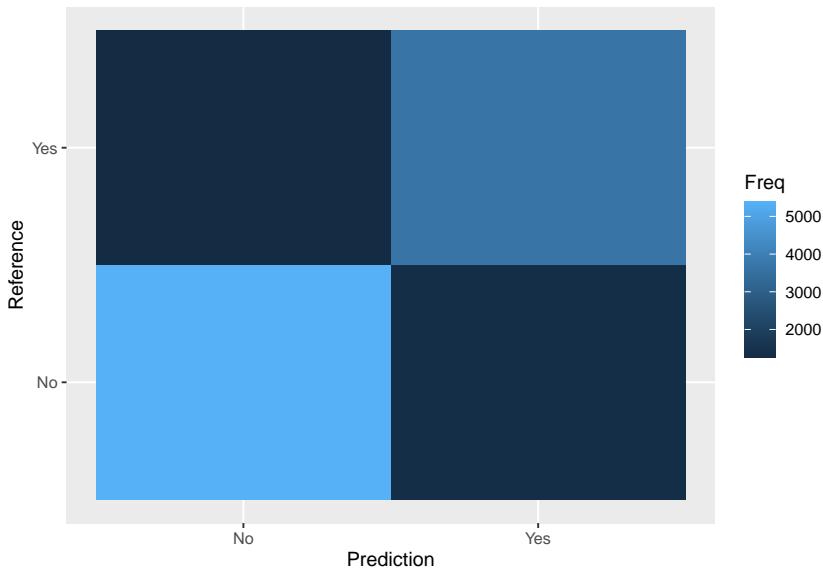
Posteriormente, hemos probado a hacer uso de una rejilla de parámetros, en un intento de conseguir una mejor combinación de hiper-parámetros que pudieran conseguir mejorar la precisión del modelo generado por la red neuronal.

```

# Definir rejilla de parámetros
paramGrid <- expand.grid(size = seq(from = 1, to = 8, by = 1),
                          decay = seq(from = 0.2, to = 0.5, by = 0.1))

# Entrenar modelo
silent <- capture.output(modelo_rna_mejorado <- train(Diabetes_binary ~ .,
                                                       data = training_set,
                                                       method = "nnet",
                                                       metric = "ROC",
                                                       trControl = trClassCtrl))

```



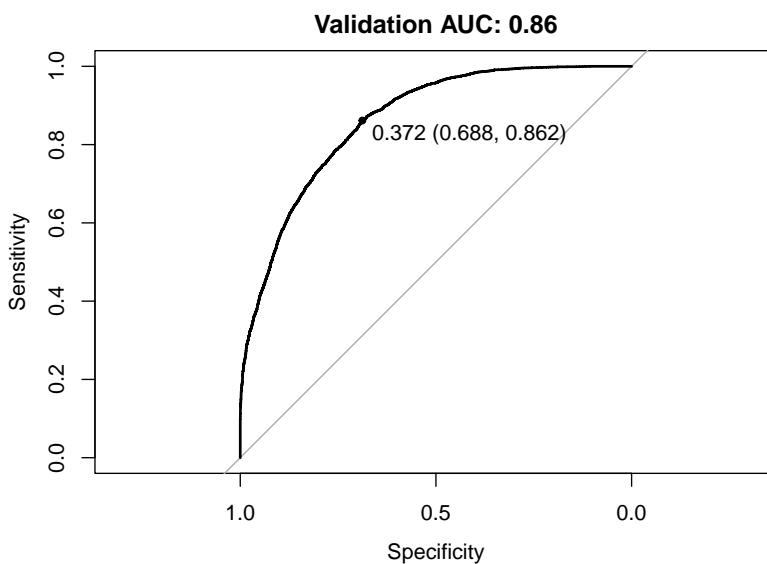
```

trControl = trClassCtrl,
tuneGrid = paramGrid))

# Generar predicciones
predictionValidationProb <- predict(modelo_rna_mejorado, validation_set,
                                      type = "prob")
predictionValidation <- predict(modelo_rna_mejorado, validation_set)

# Conseguir AUC en validación.
auc_rna_mejorado <- roc(validation_set$Diabetes_binary,
                           predictionValidationProb[["Yes"]],
                           levels = unique(validation_set[["Diabetes_binary"]]))
roc_validation <- plot.roc(auc_rna_mejorado, ylim=c(0,1), type = "S" ,
                           print.thres = T, main=paste('Validation AUC:',
                           round(auc_rna_mejorado$auc[[1]], 2)))

```



```

# Conseguir precisión y matriz de confusión en validación
acc_rna_mejorado <- confusionMatrix(predictionValidation,
                                       validation_set$Diabetes_binary)
acc_rna_mejorado$overall["Accuracy"]

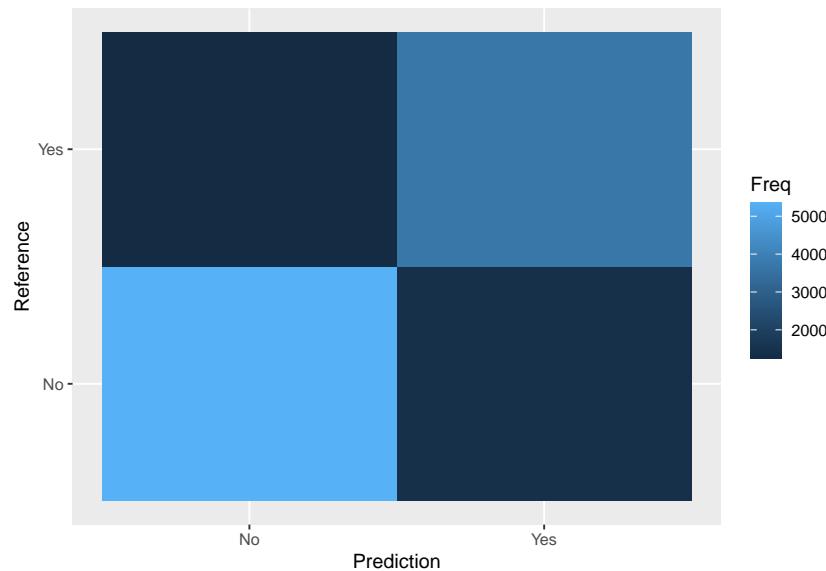
```

```

## Accuracy
## 0.7744779

matrix_table <- data.frame(acc_rna_mejorado$table)
ggplot(matrix_table, aes(x=Prediction, y=Reference, fill=Freq)) + geom_tile()

```



La combinación de parámetros que ha proporcionado el mejor resultado es `size=5` y `decay=0.4`; obteniéndose así una precisión del 77.45% y un valor del área bajo la curva de 0.8635, siendo estos los mejores resultados obtenidos a lo largo de la batería de experimentos. Cabe destacar que la búsqueda de la mejor combinación de parámetros ha tomado un tiempo bastante considerable, al ser el conjunto de datos utilizado bastante grande y querer probar bastantes combinaciones de parámetros.

## 6 Comparativa de los resultados

En este apartado, compararemos los resultados obtenidos por los diferentes modelos de clasificación para los datos resultados del procesamiento realizado.

Aunque no se ha dejado el código en el cuaderno, para evitar replicar código y facilitar la lectura, todos los modelos probados en el apartado anterior también han sido probados para el conjunto de datos original, al que solamente se le han eliminado las observaciones con valores nulos (por exigencia de algoritmos como los árboles de clasificación).

Para el conjunto de datos original, en el que sólo se han eliminado observaciones con algún valor de sus características nulo, se han obtenido los siguientes resultados:

Modelo	Precisión	AUC
Árbol de decisión	0.7168125	0.7291220
Árbol de decisión mejorado	0.7019129	0.7804903
Random Forest	0.7466116	0.8253129
Naive Bayes	0.6788426	0.8066738
Red neuronal por defecto	0.7456503	0.8255975
Red neuronal con rejilla de parámetros	0.7456503	0.8253691

Para conjunto de datos resultado de realizar el procesamiento descrito en secciones anteriores, se han obtenido los resultados que se muestran a continuación. Recordemos que para este conjunto de datos no se han cambiado los valores ya discretizados por etiquetas de texto.

Modelo	Precisión	AUC
Árbol de decisión	0.7321979	0.7449582
Árbol de decisión mejorado	0.7423827	0.8303629
Random Forest	0.7692571	0.8561485
Naive Bayes	0.7075488	0.8398569
Red neuronal por defecto	0.7720229	0.8637446
Red neuronal con rejilla de parámetros	0.7744779	0.8655603

Como se puede observar rápidamente, el procesamiento realizado permite obtener unos mejores resultados. En lo relativo a la precisión de las predicciones, la mejora oscila entre un 2 y 4% aproximadamente. En lo relativo al área bajo la curva, la mejora puede llegar a ser algo mayor, oscilando esta entre un 2 y 5% en función del algoritmo utilizado.

La mejora en las métricas expuestas puede parecer pequeña, pero teniendo en cuenta que el dataset de validación está compuesto por algo más de 12.000 muestras, el correcto diagnóstico unos 400 pacientes más puede resultar diferencial en la vida real.

También cabe analizar las matrices de confusión obtenidas con cada método. En esta memoria no se adjuntan los valores exactos, pero el número de verdaderos positivos y verdaderos negativos es aproximadamente proporcional a la precisión obtenida, salvo en el caso de Naive-Bayes, donde se tiende en mayor medida a identificar casos realmente positivos como negativos.

Desde el punto de vista de una persona no relacionada con el ámbito sanitario, como es nuestro caso, podemos pensar que es más conveniente diagnosticar gente realmente no diabética como diabética (falso positivo). De esta manera, podría aplicarse tratamiento preventivo o la realización de sucesivas para confirmar la enfermedad a posteriori. Consideramos que es bastante más grave dar por negativo a una persona que sí que es diabética.

En este sentido, los algoritmos que mejor se comportan son Random Forest y la red neuronal artificial (tanto con parámetros por defecto como con los óptimos, dado que los resultados obtenidos son prácticamente iguales), dado que son capaces de proporcionar una menor tasa de falsos negativos, pero sobre todo, de falsos positivos. Todo ello manteniendo las precisiones, más que aceptables, especificadas anteriormente.

A modo de curiosidad, se adjuntan los resultados de los modelos de clasificación si se reemplazan las etiquetas numéricas de los valores discretizados por etiquetas de texto.

Modelo	Precisión	AUC
Árbol de decisión	0.7329790	0.7612322
Árbol de decisión mejorado	0.7363061	0.8290408
Random Forest	0.7679733	0.8542094
Naive Bayes	0.6000514	0.8277808
Red neuronal por defecto	0.7727662	0.8627698
Red neuronal con rejilla de parámetros	0.7726806	0.8631080

Como se puede observar, no existen diferencias significativas respecto del dataset en el que no se realiza el cambio, salvo para el caso del algoritmo Naive Bayes, en el que la precisión puede llegar a disminuir algo más de un 11%, como se comentó anteriormente.

## 7 Conclusiones

Tras realizar un análisis exploratorio de los datos, un pre-procesamiento de los mismos y finalmente generar varios modelos de clasificación y evaluar su rendimiento, se han extraído una serie de conclusiones sobre el problema planteado, que se listan a continuación.

- El análisis de los datos de los que se dispone y su correcto procesamiento haciendo uso del conocimiento adquirido en dicho análisis resulta fundamental si se desean obtener los mejores modelos de clasificación posibles. Máxime en temas médicos como el abordado en esta práctica.

- Los modelos de árboles de decisión simples permiten obtener resultados bastante razonables, aunque inferiores a otras soluciones algo más complejas. Jugar con los parámetros de los mismos o el algoritmo concreto empleado para su construcción puede influir en la calidad de los resultados obtenidos, aunque para el conjunto de datos empleado en esta práctica la diferencia es prácticamente nula.
- El modelo que mejores resultados ha proporcionado en la práctica ha sido la red neuronal en la que se ha buscado iterativamente la mejor combinación de hiper-parámetros posible. Este modelo permite predecir correctamente cerca del 77.5% de las observaciones que recibe como entrada. Nótese que esta búsqueda de parámetros ha requerido un tiempo de ejecución considerable (cerca de 20 minutos en un equipo con procesador Intel Core i7 8700K)
- Los modelos de redes neuronales requieren de un tiempo de generación considerablemente mayor a los requeridos por otros algoritmos más simples. Conforme aumente el número de datos utilizados para el entrenamiento este tiempo puede aumentar en gran medida, pudiendo llegar a ser difícilmente asumible de no disponerse de grandes equipos. Para los datos de los que se disponía, la diferencia en la precisión de las predicciones con respecto del algoritmo Random Forest apenas es del 0.5%, por lo que podría estudiarse el uso de esta alternativa, cuyo tiempo de construcción del modelo es mucho menor, si se está dispuesto a asumir el sacrificio en precisión (poco probable en el caso de diagnóstico de enfermedades).
- Es importante definir el criterio por el que un algoritmo puede ser mejor que otro. Por lo general, utilizamos la precisión, pero podría ser interesante combinar esta métrica con la tasa de falsos positivos y falsos negativos, sobre todo en el ámbito de la salud.

## 8 Bibliografía

Durante la realización de esta práctica se ha consultado la siguiente bibliografía:

- [1] Diabetes Health Indicators Dataset
- [2] Documentación oficial de Caret.
- [3] AGE5YR: Fourteen-Level Age Category
- [4] EDUCA: Education Level
- [5] INCOME2: Income Level
- [6] Discretization functions for BMI
- [7] Outlier Treatment
- [8] OSLRR Package Documentation
- [9] RPART Package Documentation
- [10] C50 Package Documentation
- [11] RANGER Package Documentation
- [12] Diabetes Prediction | EDA + Preprocessing + Models
- [13] Diabetes detailed EDA with conclusion
- [14] Naive Bayes
- [15] Exploratory Data Analysis
- [16] Understand Random Forest Algorithms With Examples