UNIVERSIDAD
NACIONAL DEL OESTE

Explotación de Datos

Grupo 3

Actividad N° 4

Arboles de decisión

Docentes:

Dejean, Gustavo
Mendoza, Dante
Pérez, Silvia

Estudiantes:

Acosta, Rios Alejandro
Esperanza, Marcelo Fabián
Correa, Matias Daniel

Resumen

En el presente informe vamos a realizar un análisis de clasificación con varios métodos que nos permita determinar si dado un conjunto de variables se puede determinar si la persona tiene colesterol o no. Trataremos también de utilizar métodos que nos permitan mejorar la predicción de la variable *class*, es decir la variable que realiza la clasificación.

Índice

1 | Introducción

1.1 | Datos a utilizar

Utilizamos como dataset los datos contenidos en la *Encuesta Nacional de Factores de Riesgo* del año 2018, provisto por el INDEC (Instituto Nacional de Estadística y Censos).[1]

1.2 | Objetivo

Se desea corroborar si es posible predecir si una persona tiene colesterol o no en base a las demás variables del dataset, utilizando árboles de decisión y el algoritmo *Random Forest*.

2 | Desarrollo

2.1 | Preparación de los datos

Luego de haber importado los datos, eliminado los datos nulos y las respuestas *'NSNC'* (*'No Sabe No Contesta'*, pues al ser una encuesta, es posible responder con ese valor, el cual no nos aporta mucha información), procedemos a una limpieza de los datos. Renombramos las variables y algunos datos de tipo factor para mayor legibilidad, categorizamos la variable *IMC* para que se agrupe en intervalos, los cuales nos determinan si el peso de una persona es bajo, normal, con sobrepeso, etc. Nos basaremos en este cuadro para la categorización:

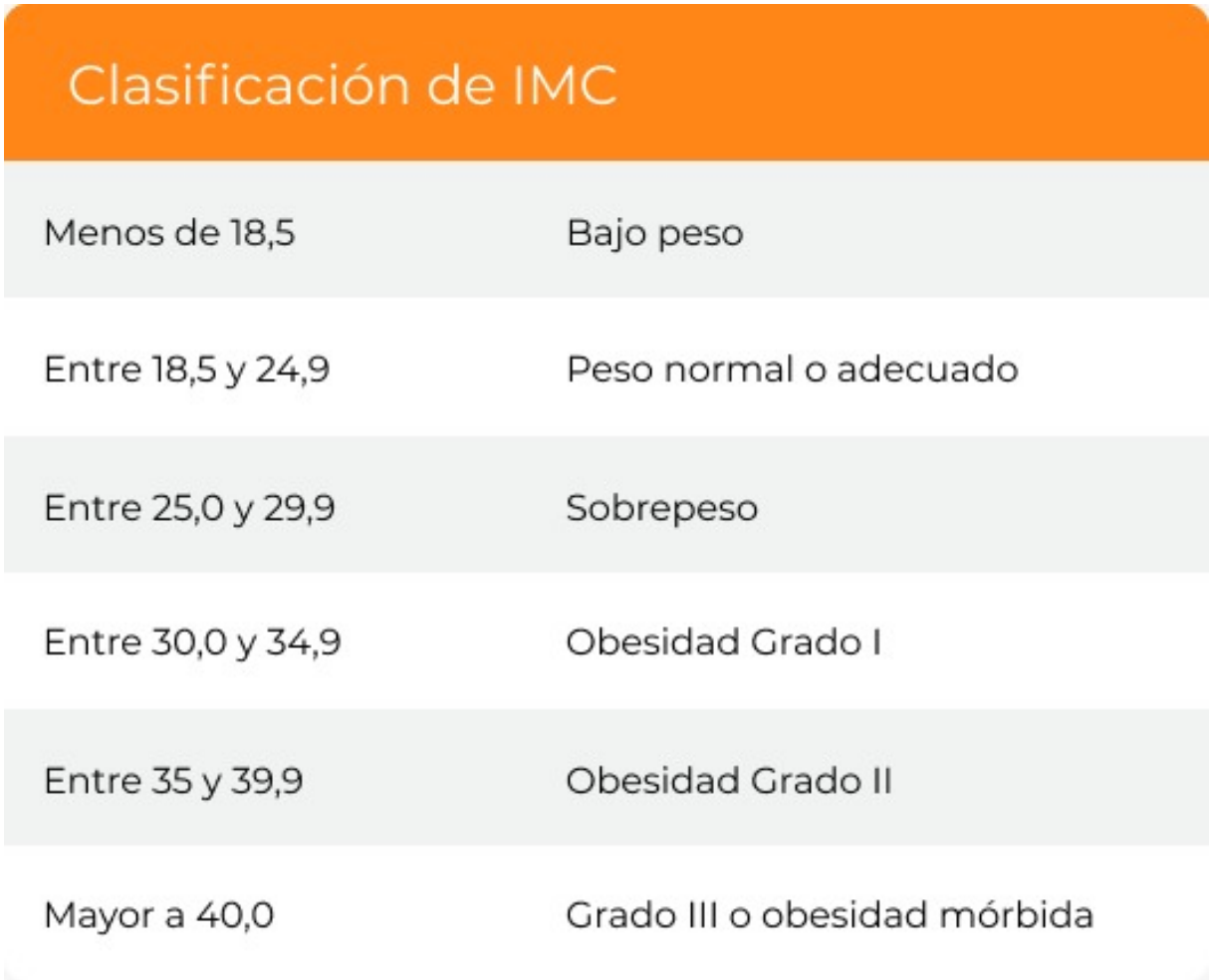

Figura 1: Tabla de clasificación del IMC[2]

3 | Análisis de Clasificación con Arboles de Decisión

3.1 | Entrenamiento del modelo

Primero seleccionamos una semilla inicial para que sea posible reproducir el modelo en diferentes equipos. Luego, separamos los datos de forma aleatoria: un 80 % será seleccionado para el entrenamiento del modelo y el 20 % restante para probar el modelo. Una vez realizado estos pasos aplicamos el rpart sobre los datos de entrenamiento.

```
set.seed(2022) # Número inicial a partir del cuál comenzará a generar una
secuencia aleatoria.
# Tomamos un 80% para el entrenamiento y un 20% para el test
split <- createDataPartition(encuesta.salud.clasificada$Colesterol, p=0.8,
list=FALSE) #List = Si el resultado devolverá una lista o una matriz.
train<- encuesta.salud.clasificada[split,]
test<-  encuesta.salud.clasificada[-split,]
```

Figura 2: Separación de los datos para entrenamiento y prueba

3.2 | Visualización del modelo

Una vez creado el modelo en base a los datos de entrenamiento, procedemos a visualizarlo. Realizar el gráfico del árbol de decisiones es importante para dar una idea de como se está llevando a cabo la clasificación y si realmente la clasificación tiene sentido o arroja buenos resultados.

En la figura 3 vemos que el total de personas que no tienen colesterol equivale a un 69 %, mientras que las personas con colesterol equivale a un 31 %. Si seguimos al siguiente nivel del árbol, vemos que este se divide en dos nodos. La variable divisora es la hipertensión, si vemos la rama en que se cumple la condición 'Hipertension = No' observamos que el 77 % de las personas no tienen colesterol, es decir, el no tener hipertensión hace que la mayoría de las personas no tengan colesterol. Luego, cuando la condición de 'Hipertensión = No' no se cumple, vemos que tenemos un 58 % de personas que no tienen colesterol contra un 42 % de personas que si lo tienen. Podemos notar que aunque las personas sean hipertensas sigue dando un porcentaje mayor de personas que no tienen colesterol aunque sean hipertensas.

Luego observamos que el árbol vuelve a dividir el nodo en dos por medio de la condición 'Prevalencia_Diabetes = No'. De los casos que cumplen con la condición, un 62 % no tiene colesterol y un 38 % tiene colesterol. De los casos que no cumplen con la condición 'Prevalencia_Diabetes = No', un 44 % no tiene colesterol y un 56 % si lo tiene.

También podemos ver en este gráfico cuales son las variables que el árbol consideró como importante a la hora de poder clasificar a las personas con colesterol. En este caso vemos que las variables más importantes son *Hipertension* y *Prevalencia_Diabetes*. A primera vista concluimos que la variable que mejor clasifica es la hipertensión.

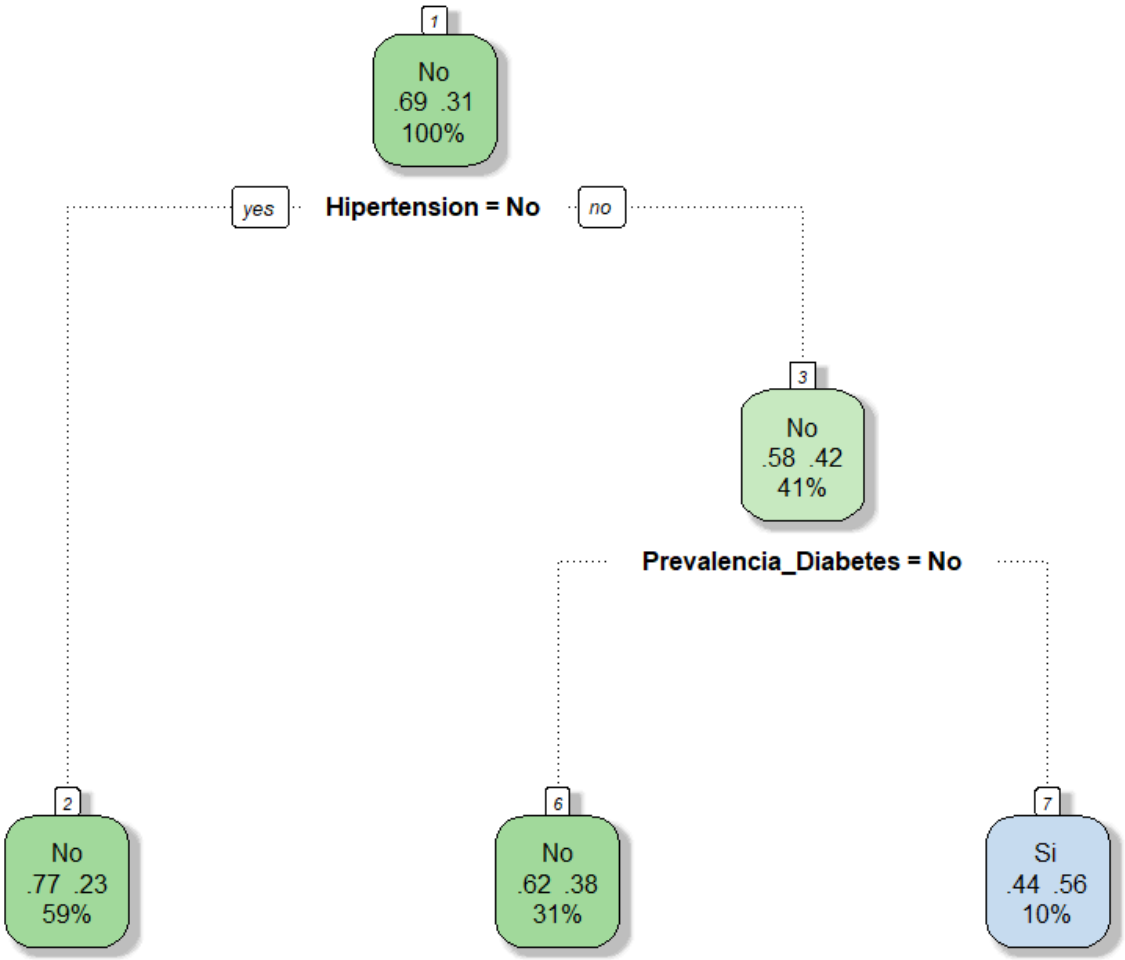

Figura 3: Árbol de decisión

3.3 | Matriz de confusión

Utilizando el modelo creado en base a los datos de entrenamiento, intentaremos predecir los datos de prueba. Luego, por medio de una matriz de confusión, podremos contrastar los casos correctamente clasificados contra los que no, así como la precisión, sensibilidad y especificidad.

```
Confusion Matrix and Statistics

          Reference
Prediction   No   Si
        No 2298  908
        Si  152  179

               Accuracy : 0.7003
                 95% CI : (0.6849, 0.7154)
    No Information Rate : 0.6927
    P-Value [Acc > NIR] : 0.1671

                  Kappa : 0.1273

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.16467
            Specificity : 0.93796
         Pos Pred Value : 0.54079
         Neg Pred Value : 0.71678
             Prevalence : 0.30732
         Detection Rate : 0.05061
   Detection Prevalence : 0.09358
      Balanced Accuracy : 0.55132
```

Figura 4: Matriz de confusión

De los 3206 casos que no tienen colesterol, 2298 fueron predichos correctamente y 908 casos incorrectamente (el modelo predijo que esos casos no tendrían colesterol pero resultaron si tenerlo). Por otro lado, de los 331 casos que si tienen colesterol, 179 fueron predichos correctamente y 152 incorrectamente (el modelo predijo que esos casos tendrían colesterol pero resultaron no tenerlo).

Vemos que la precisión fue del 70.03 %, la sensibilidad del 16.47 % y la especificidad del 93.8 %. La sensibilidad nos dice, en esencia, que tan bien el modelo predice los casos que tienen colesterol, que en este caso lo hace muy mal por el bajo valor de sensibilidad. En cambio, la especificidad nos dice que tan bien el modelo predice los casos que no tienen colesterol, que para este caso lo hace muy bien por el alto valor de especificidad.

3.4 | Tratamiento de clases desbalanceadas

La razón por la que nuestro modelo tenga una sensibilidad tan baja y una especificidad tan alta, se debe a que las clases a predecir no están balanceadas, como se puede ver en el siguiente gráfico:

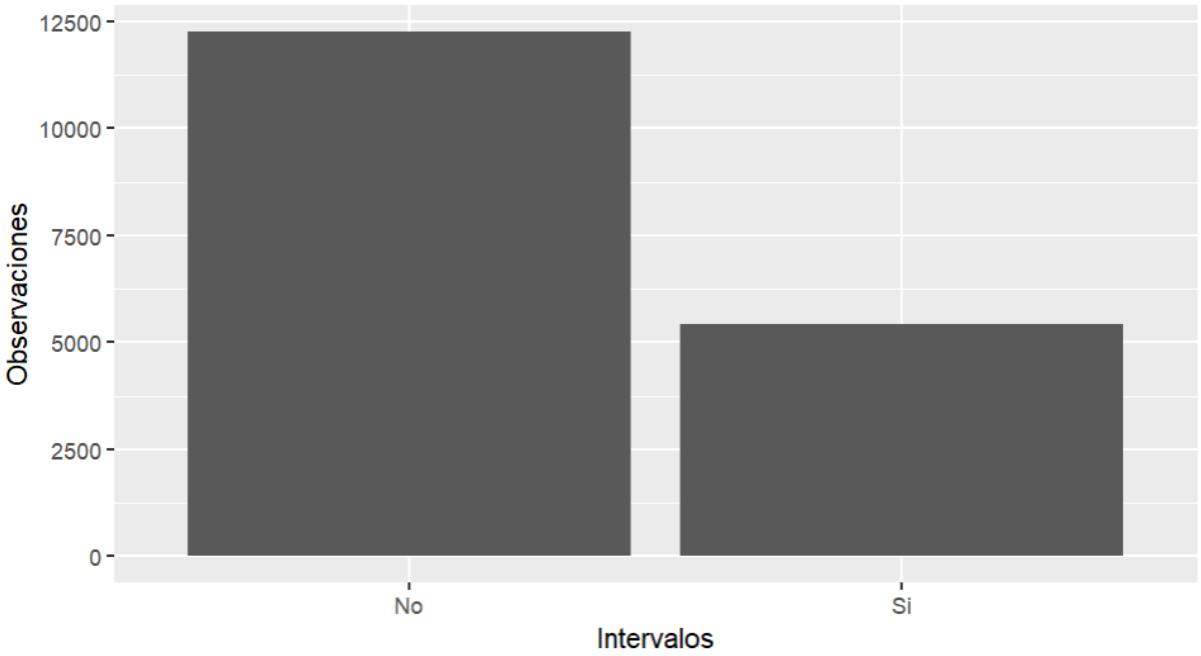

Figura 5: Cantidad de casos según clase

Del total de los datos, el 69.25 % de los casos no tienen colesterol, mientras que el 30.75 % si lo tienen. Esta diferencia tan grande entre los distintos casos hace que el modelo prediga mejor los casos que no tienen colesterol de los que si lo tienen, ya que los arboles de decisión, así como Random Forest, son muy sensibles al desbalanceo entre clases. Por ello, utilizaremos cuatro métodos distintos para el tratamiento de clases desbalanceadas y veremos como afecta a futuros modelos. Para todos ellos, utilizaremos la biblioteca *ROSE*.

3.4.1 | Primer método: Sobre-Muestreo

El método consiste en seleccionar de forma aleatoria casos de la clase con menor cantidad de casos y duplicarlos hasta igualar a la clase con más casos. La biblioteca lo hace por nosotros por medio de la función *ovun.sample*. Luego de haberla ejecutado, vemos que tenemos la misma cantidad de casos para cada clase (9801 casos para cada una). Posteriormente, procedemos a crear y visualizar el nuevo modelo a partir de los nuevos datos balanceados.

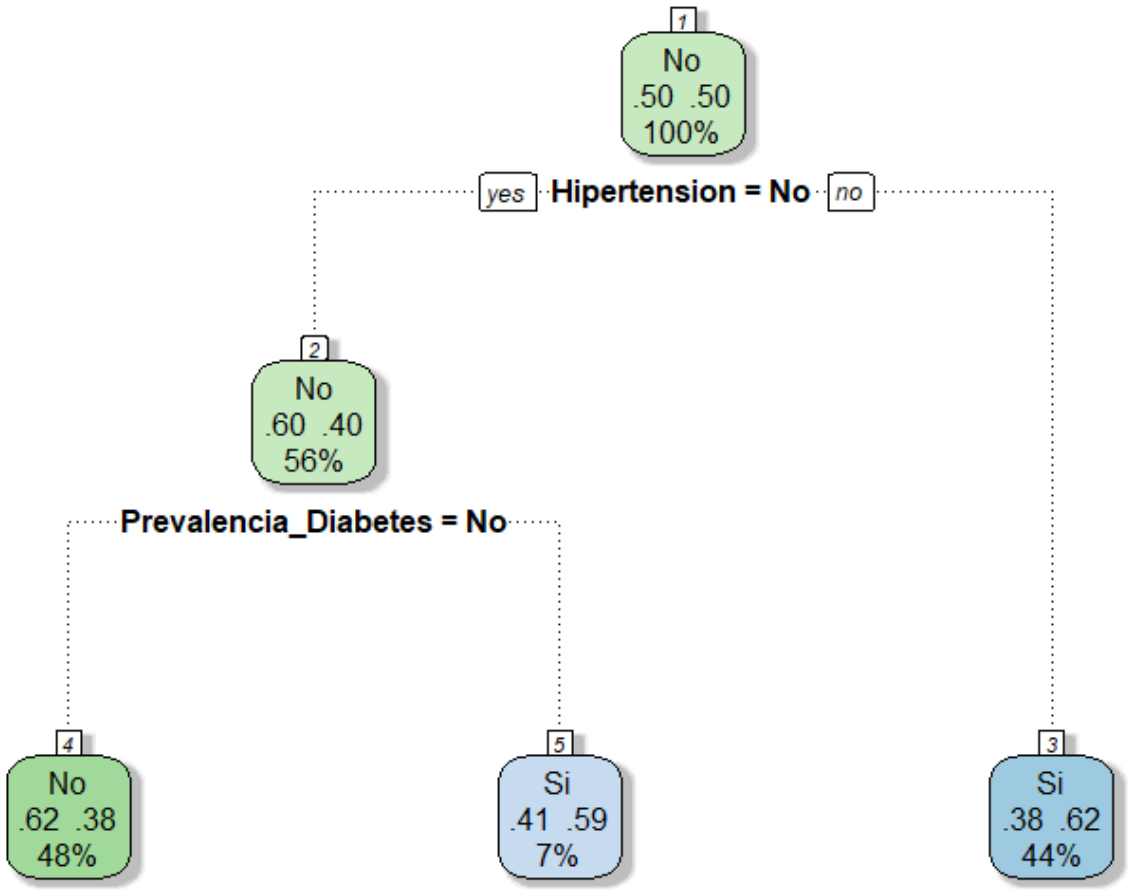

Figura 6: Árbol de decisión utilizando Sobre-Muestreo

Vemos que las variables no han cambiado, pero ahora aquellos que cumplen con la condición 'Hipertension = No' vuelven a ser divididos con la condición 'Prevalencia_Diabetes', además que los porcentajes de **Si** y **No** han cambiado en cada nodo.

Viendo la matriz de confusión en la figura 7, vemos que la precisión bajó (de 70.03 % a 61.07 %), así como la especificidad (de 93.8 % a 60.2 %), pero la sensibilidad subió considerablemente (de 16.47 % a 63.02 %). Este nuevo modelo predice mucho mejor los **Si**, a costa de predecir peor los **No** en comparación al primer modelo.

```
Confusion Matrix and Statistics

          Reference
Prediction   No   Si
        No 1475  402
        Si  975  685

               Accuracy : 0.6107
                 95% CI : (0.5944, 0.6268)
    No Information Rate : 0.6927
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2025

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6302
            Specificity : 0.6020
         Pos Pred Value : 0.4127
         Neg Pred Value : 0.7858
             Prevalence : 0.3073
         Detection Rate : 0.1937
   Detection Prevalence : 0.4693
      Balanced Accuracy : 0.6161
```

Figura 7: Matriz de confusión utilizando Sobre-Muestreo

3.4.2 | Segundo método: Sub-Muestreo

En este método, quitamos aleatoriamente casos de la clase que tenga más casos hasta igualar la cantidad de casos de la clase con menos casos.

Posteriormente a haber aplicado el método, tenemos la misma cantidad de casos para cada clase (4352 casos para cada una).

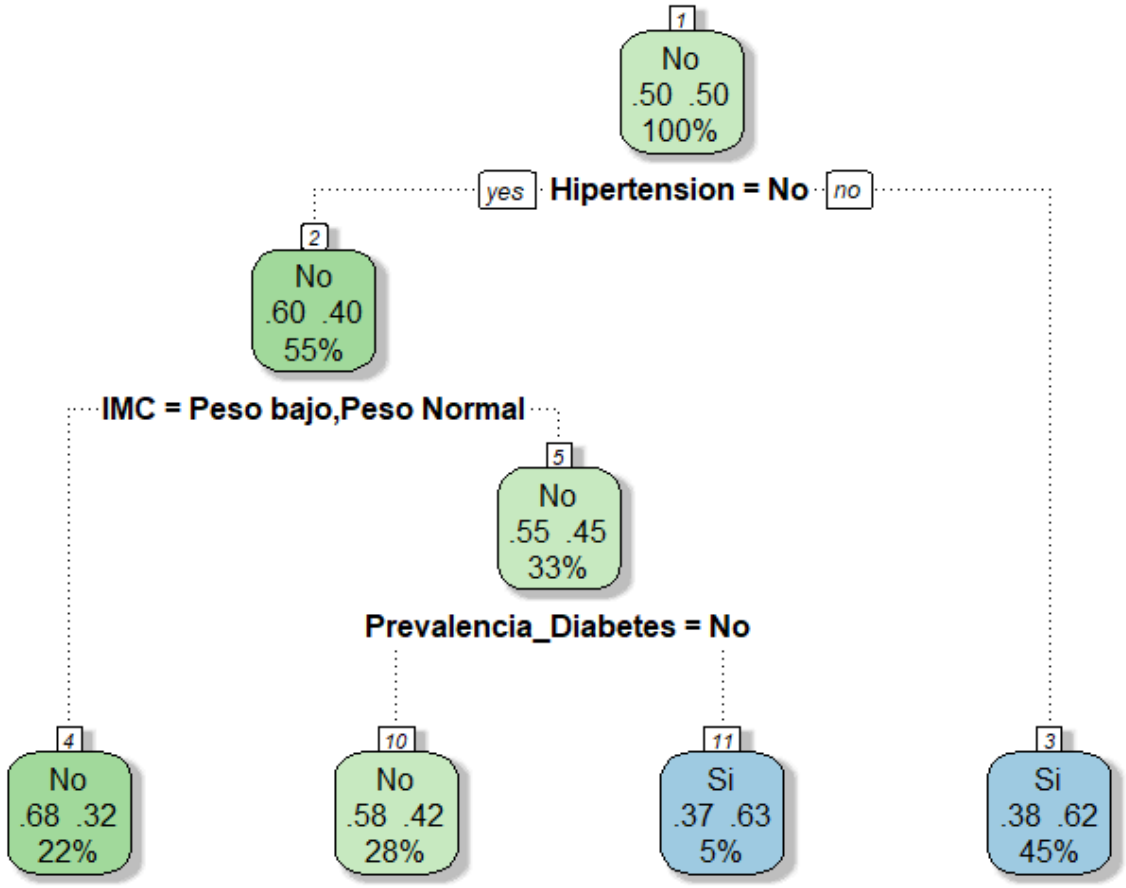

Figura 8: Árbol de decisión utilizando Sub-Muestreo

Vemos en la figura 8 que además de las dos variables ya mencionadas, este árbol añade una más: Aquellos casos que no tengan **Peso bajo** o **Peso Normal** son nuevamente divididos por la variable 'Prevalencia_Diabetes'. También vemos que los porcentajes para cada nodo son distintos.

```
Confusion Matrix and Statistics

          Reference
Prediction   No   Si
        No 1526  418
        Si  924  669

               Accuracy : 0.6206
                 95% CI : (0.6044, 0.6366)
    No Information Rate : 0.6927
    P-Value [Acc > NIR] : 1

                  Kappa : 0.211

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6155
            Specificity : 0.6229
         Pos Pred Value : 0.4200
         Neg Pred Value : 0.7850
             Prevalence : 0.3073
         Detection Rate : 0.1891
   Detection Prevalence : 0.4504
      Balanced Accuracy : 0.6192
```

Figura 9: Matriz de confusión utilizando Sub-Muestreo

Según la matriz de confusión en la figura 9, la precisión subió en comparación al modelo anterior (de 61.07 % a 62.06 %), así como la especificidad (de 60.2 % a 62.29 %), pero la sensibilidad bajó (de 63.02 % a 61.55 %).

3.4.3 | Tercer método: Ambos (Sobre y Sub-Muestreo)

Para este método, seleccionamos de manera aleatoria casos de ambas clases: los casos de la clase con más filas serán removidos y los de la clase con menos filas serán duplicados. Luego de haber utilizado el método (con la misma función que para los demás métodos), nos queda un total de 7069 casos que no tienen colesterol y 7084 casos que si tienen. A diferencia de los métodos anteriores, no tenemos la misma cantidad de casos para cada clase, pero aproximadamente es un 50 % para cada una de las mismas.

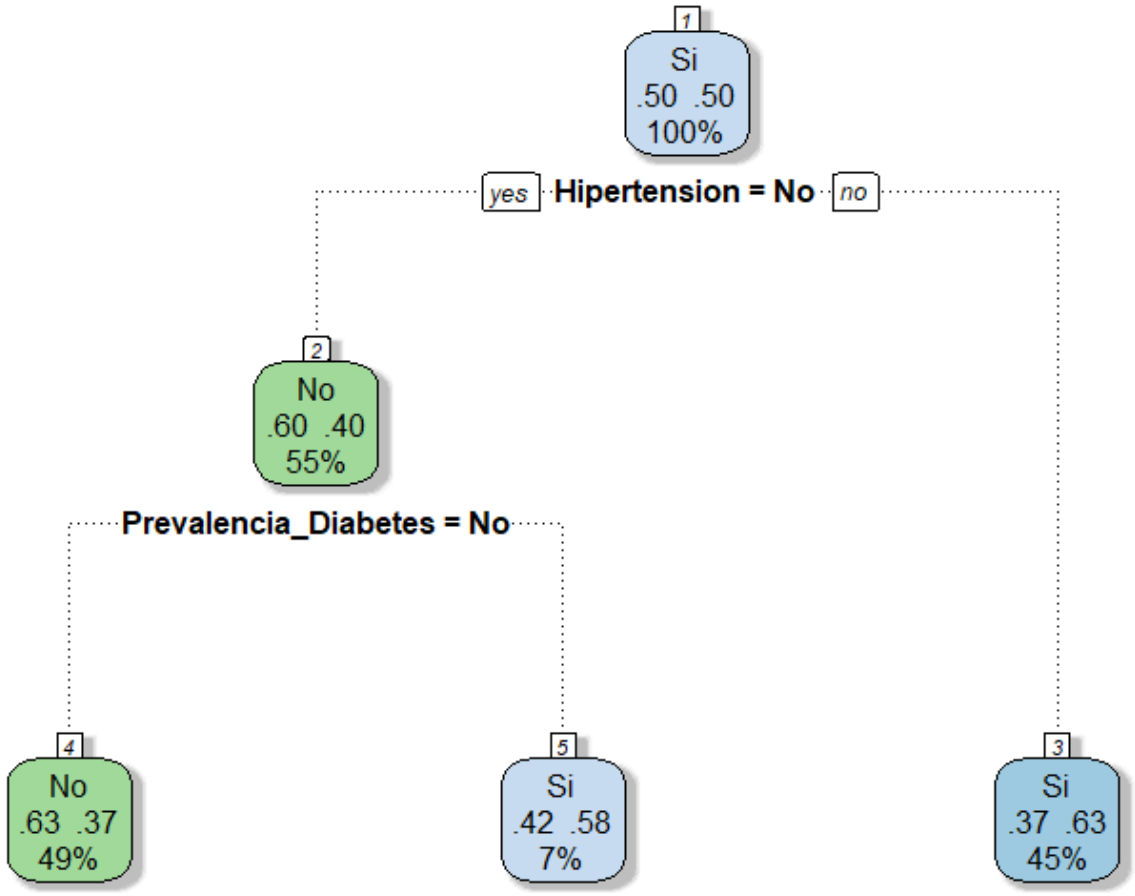

Figura 10: Árbol de decisión utilizando Sobre y Sub-Muestreo

Viendo la figura 10, nos damos cuenta que el árbol es similar al obtenido con el método de Sobre-Muestreo, aunque los porcentajes son distintos.

```
Confusion Matrix and Statistics

          Reference
Prediction   No   Si
        No 1475  402
        Si  975  685

               Accuracy : 0.6107
                 95% CI : (0.5944, 0.6268)
    No Information Rate : 0.6927
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2025

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6302
            Specificity : 0.6020
         Pos Pred Value : 0.4127
         Neg Pred Value : 0.7858
             Prevalence : 0.3073
         Detection Rate : 0.1937
   Detection Prevalence : 0.4693
      Balanced Accuracy : 0.6161
```

Figura 11: Matriz de confusión utilizando Sobre y Sub-Muestreo

Analizando la matriz de confusión en la figura 11, vemos que la precisión bajó (de 62.06 % a 61.07 %) así como la especificidad (de 62.29 % a 60.2 %), aunque la sensibilidad aumentó (de 61.55 % a 63.02 %). La precisión, especificidad y sensibilidad de este modelo son iguales al modelo obtenido utilizando el método de Sobre-Muestreo.

3.4.4 | Cuarto método: Muestreo Sintético

Para este último método utilizaremos la función *ROSE* (provista por la biblioteca del mismo nombre). Dicha función crea muestras sintéticas a partir de los datos originales (por medio de algoritmos). Ejecutamos la función pidiéndole 20000 muestras en total. Al final tenemos 10095 casos que no tienen colesterol y 9905 que si tienen colesterol, dándonos aproximadamente un 50 % de los casos para cada una de las clases.

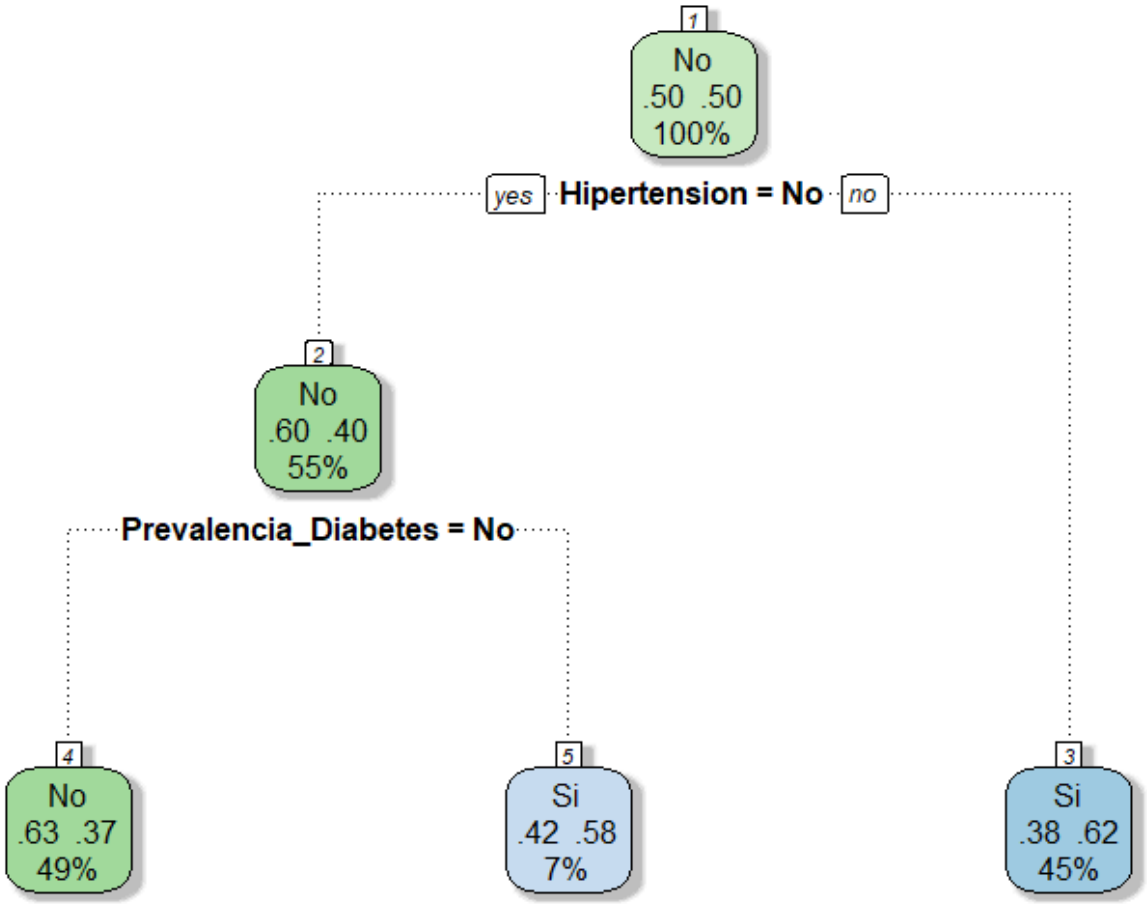

Figura 12: Árbol de decisión utilizando Muestreo Sintético

Dándole un simple vistazo al gráfico de la figura 12, vemos que es prácticamente similar al gráfico del modelo anterior, a excepción de un nodo que tiene porcentajes distintos.

```
Confusion Matrix and Statistics

          Reference
Prediction   No   Si
        No 1475  402
        Si  975  685

               Accuracy : 0.6107
                 95% CI : (0.5944, 0.6268)
    No Information Rate : 0.6927
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2025

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6302
            Specificity : 0.6020
         Pos Pred Value : 0.4127
         Neg Pred Value : 0.7858
             Prevalence : 0.3073
         Detection Rate : 0.1937
   Detection Prevalence : 0.4693
      Balanced Accuracy : 0.6161
```

Figura 13: Matriz de confusión utilizando Muestreo Sintético

Observando la matriz de confusión en la figura 13, notamos que la precisión, especificidad y sensibilidad son exactamente iguales que en el método anterior.

Para finalizar, vamos a mostrar una desventaja que presenta este método frente a los otros presentados. En la figura 14 vemos que el valor máximo para la variable *Actividad_Intensa* es de 2880 y para la variable *Actividad_Moderada* es 4200, y que el mínimo para ambas es de 0. Sin embargo, si observamos la figura 15 nos damos cuenta que los máximos y mínimos de dichas variables no son los mismos al utilizar el método de Muestreo Sintético. El máximo para la variable *Actividad_Intensa* es de 3978.11 y para *Actividad_Moderada* es de 4381.67. En cambio, los mínimos para ambas variables son valores negativos, y en nuestro dataset no tiene sentido que la actividad física sea un valor negativo.

```
Ansiedad_Depresion Actividad_Intensa Actividad_Moderada
Moderado: 2385     Min.   :   0.00   Min.   :   0.0
Mucho   :  461     1st Qu.:   0.00   1st Qu.:   0.0
No      :11307     Median :   0.00   Median :   0.0
                   Mean   :  55.91   Mean   : 117.4
                   3rd Qu.:   0.00   3rd Qu.: 120.0
                   Max.   :2880.00   Max.   :4200.0
```

Figura 14: Resumen de los datos de entrenamiento

```
> summary(rose)
 Ansiedad_Depresion Actividad_Intensa Actividad_Moderada
 Moderado: 3463     Min.   :-236.65   Min.   :-451.23
 Mucho   :  574     1st Qu.: -31.42   1st Qu.: -35.30
 No      :15963     Median :  16.59   Median :  51.37
                    Mean   :  56.58   Mean   : 116.78
                    3rd Qu.:  79.41   3rd Qu.: 168.89
                    Max.   :3978.11   Max.   :4381.67
```

Figura 15: Resumen de los datos de entrenamiento utilizando Muestreo Sintético

En resumen, no solo este método no presenta ventaja en cuanto al anterior (en términos de precisión, especificidad y sensibilidad), si no que no conviene utilizarlo por que crea muestras con valores que en nuestro caso no tienen sentido. Eso no quiere decir que este método no sea eficiente. En la realidad, se deben aplicar diferentes métodos y analizar uno por uno para seleccionar el que mejor resultados ofrezca, ya que no siempre el método que mejor resultados arroje será el mismo.

4 | Análisis de Clasificación con Random Forest

Los modelos Random Forest están formados por un conjunto de árboles de decisión individuales. La idea central es evaluar todos los arboles y obtener por mayoría de votos la variable que mejor clasifique a la observación.

A continuación se describen los siguientes pasos para el análisis de clasificación: en primer lugar se hace un análisis de la cantidad de variables a elegir, luego se entrena el modelo con random forest considerando la cantidad de variables óptima que obtuvimos en el paso anterior. El siguiente paso es analizar un árbol de desición del conjunto de arboles que genera el random forest, para tener una idea de la clasificación que se realiza y su calidad. Posteriormente, se hará un análisis de predicción. Por último se intentará mejorar el modelo.

4.1 | Entrenamiento del modelo

En esta sección aplicaremos la función *randomForest* con los datos de entrenamiento. Le pasaremos como parametro la cantidad de árboles igual a 700 y la cantidad de variables igual a 2. Vale decir que la cantidad de parámetros no se eligió considerando la raíz cuadrada del total de variables independientes del dataset (a pesar de ser una buena forma de determinar la cantidad de variables), ya que en nuestras pruebas, colocar una cantidad superior a 2 variables aumentaba considerablemente el error OOB. Sin embargo, existen métodos más complejos que permiten hacer una mejor elección.

En este caso, el OOB nos da un 30,03 %. Si analizamos la matriz de confusión, vemos que hay un 95 % de aciertos cuando se predice que no hay colesterol, pero existe un 12 % de aciertos en el caso de que se prediga que la persona tiene colesterol. Esto es aproximadamente el complemento del error de clase.

```
        OOB estimate of  error rate: 30.03%
Confusion matrix:
     No  Si class.error
No 9351 450  0.04591368
Si 3800 552  0.87316176
```

Figura 16: Primer modelo Random Forest

4.2 | Análisis de predicción

Para evaluar si realmente clasifica bien la variable respuesta, debemos realizar las predicciones sobre nuevos datos que se encuentran en el dataset test. Luego de calcular las predicciones debemos armar la matriz de confusión para analizar su resultado.

La figura 17 nos muestra la matriz mencionada anteriormente y nos indica que 2333 personas fueron predichas correctamente que no tenían colesterol y que también se predijo correctamente que 157 personas poseían colesterol. Esto significa que el 70.04 % de las predicciones fueron acertadas, es decir, se clasificó de manera correcta. Si sacamos el porcentaje de errores nos da un 30.06 %. Esto estaría, en concordancia con el error mostrado en el modelo hecho con los datos de entrenamiento.

```
          Reference
Prediction   No   Si
        No 2333  930
        Si  117  157

               Accuracy : 0.704
                 95% CI : (0.6886, 0.719)
    No Information Rate : 0.6927
    P-Value [Acc > NIR] : 0.07464

                  Kappa : 0.1221

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.14443
            Specificity : 0.95224
         Pos Pred Value : 0.57299
         Neg Pred Value : 0.71499
             Prevalence : 0.30732
         Detection Rate : 0.04439
   Detection Prevalence : 0.07747
      Balanced Accuracy : 0.54834
```

Figura 17: Mejor modelo.

4.3 | Mejora del Modelo

4.3.1 | Gráfico del error OOB en el modelo *rf1*

Para mejorar el modelo, lo que se intentará es ver a través de un gráfico la disminución de los errores en función de la cantidad de árboles. Como vemos en la figura 18, los errores empiezan a disminuir hasta llegar un punto en que se mantiene casi constante en sus valores. Esta linea constante comienza a notarse a partir de la cantidad de árboles igual 300.

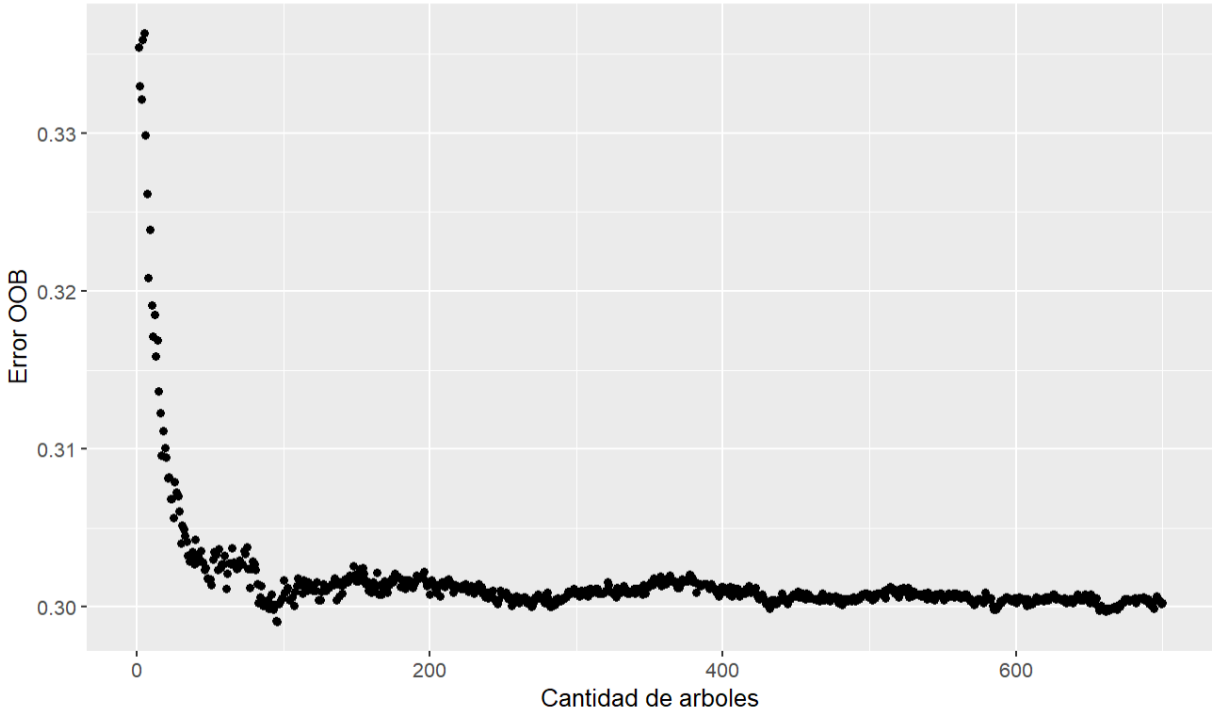

Figura 18: Gráfico OOB vs Cantidad de Árboles

Al realizar el modelo 2 de Random Forest (con cantidad de árboles igual a 300) junto con la predicción, vemos que los resultados no cambian mucho. Siguen dando valores parecidos de aciertos y errores. También, tanto para la especificidad como la sensibilidad.

```
        OOB estimate of  error rate: 30.01%
Confusion matrix:
     No  Si class.error
No 9360 441  0.04499541
Si 3807 545  0.87477022
```

Figura 19: Error OOB y de clases del segundo modelo

```
Confusion Matrix and Statistics

          Reference
Prediction   No   Si
        No 2351  956
        Si   99  131

               Accuracy : 0.7017
                 95% CI : (0.6863, 0.7168)
    No Information Rate : 0.6927
    P-Value [Acc > NIR] : 0.1253

                  Kappa : 0.1026

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.12052
            Specificity : 0.95959
         Pos Pred Value : 0.56957
         Neg Pred Value : 0.71092
             Prevalence : 0.30732
         Detection Rate : 0.03704
   Detection Prevalence : 0.06503
      Balanced Accuracy : 0.54005
```

Figura 20: Matriz de confusión del segundo modelo

4.3.2 | Analizando modelo 3

Para ver si nuestra predicción puede mejorar, decidimos, en este caso, realizar un Random Forest pero con el dataframe de train que resultó de haber aplicado la técnica de "tratamiento de clases desvalanceadas", en este caso con el método .ªmbos"que combina el Sub-Muestreo y el Sobre-Muestreo. También decidimos cambiar la cantidad de variables a 4 (en este caso se debería utilizar un método para la selección de la mejor variable, pero comprobamos que a medida que se aumenta la cantidad de variables por encima de 4, la reducción del error OOB se hace menos significativa).

```
Call:
 randomForest(formula = Colesterol ~ ., data = ambos, ntree = 300,      mtry = 4)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 4

        OOB estimate of  error rate: 13.43%
Confusion matrix:
     No   Si class.error
No 5979 1090   0.1541944
Si  811 6273   0.1144833
```

Figura 21: Media de los residuos estandarizados

Como se puede ver en la figura 21 el error disminuyó considerablemente de un 30 % a un 13.43 %. También vemos que a la hora de predecir que una persona no tiene colesterol el acierto es de un 85 %, mientras que para predecir que tienen colesterol se acierta un 89 % . Para estos últimos valores, observamos que la especificidad bajó, mientras que la sensitividad aumentó. Para nuestro caso en cuestión no es un buen indicio que haya bajado el porcentaje de aciertos de las personas que no tienen colesterol. Sí podemos destacamos que el error OOB disminuyó.

4.4 | Elección de las variables más importantes

Como vemos en la figura 22 las variables más importantes según el *Mean Decrese Accuracy* son la *Prevalencia_Diabetes*, *Hipertension*, el *IMC*, la *Ansiedad_Depresion* y así de forma descendente le siguen las demás variables en orden de importancia.

También podemos ver en el gráfico de *Mean Decrease Gini* que las variables que distribuyen de forma más proporcional los casos son el *Promedio_Frut_Ver_Diario*, luego la variable *Caminata*, *Sentado*, y así de forma descendente.

Con estos gráficos podemos destacar, por ejemplo, que si bien la variable *Prevalencia_Diabetes* es importante, ya que afecta a la precisión del modelo, también vemos que es una variable que no distribuye muy bien los casos.

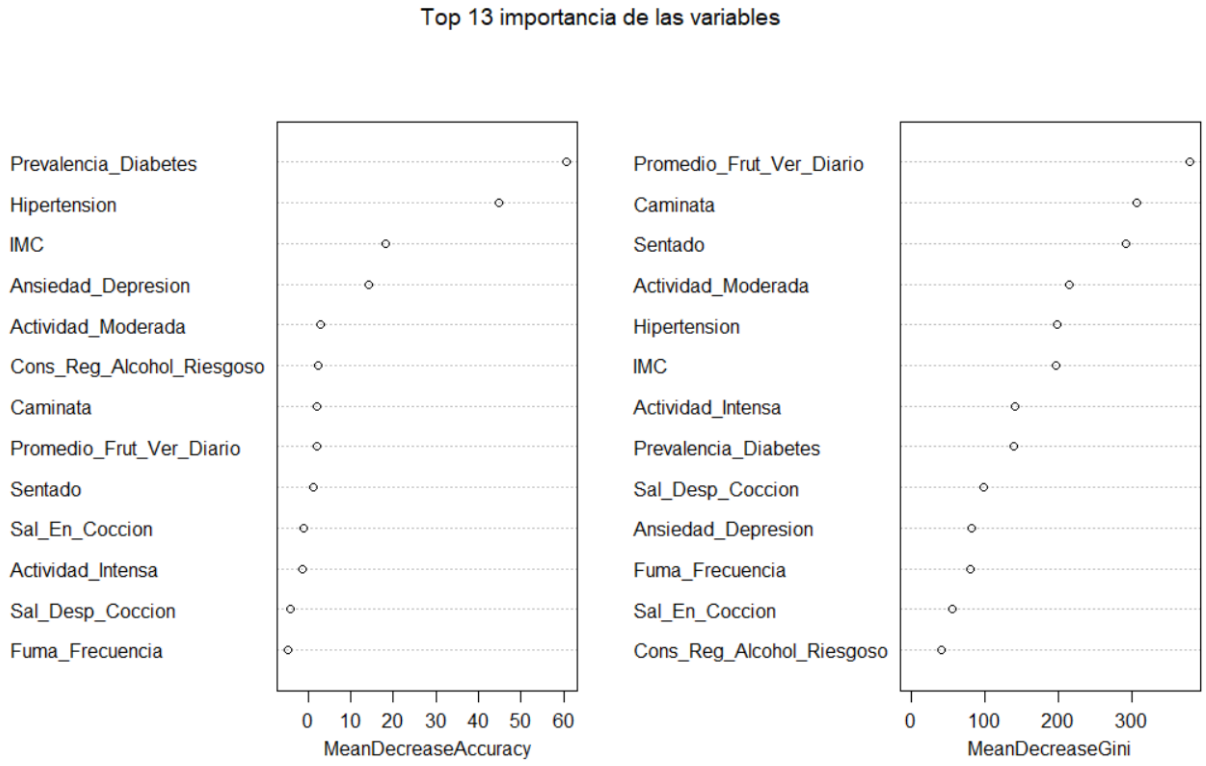

Figura 22: Variables según importancia

5 | Utilización de Ranger y Curva ROC

En esta ocasión, vamos a utilizar otro método para aplicar y evaluar un modelo de clasificación óptimo. Ranger es una implementación rápida de Bosques Aleatorios para R. Útil cuando la cantidad de variables es grande. Cuando se implemente este método, los parámetros por defectos son cantidad de arboles igual a 500 y variables igual a 2. En nuestro caso decidimos incluir las 4 variables más importantes. Al aplicar el algoritmo Ranger, vemos que el error de predicción es de un 29.76 % aproximadamente.

```
Type:                             Classification
Number of trees:                  500
Sample size:                      14153
Number of independent variables:  4
Mtry:                             2
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             29.76 %
```

Figura 23: Primer modelo con Ranger

Luego creamos un segundo modelo con Ranger. En este caso vamos a ajustar los hiperparametros para poder mejorar el modelo. El número de árboles que utilizamos es 1000, la cantidad de variables 2, la profundidad del árbol 20 y por último indicamos que se evalúe el modelo de clasificación según la probabilidad. Como se muestra en la figura 24 vemos que el porcentaje de error OOB es de un 19.8 %, lo cual indica que bajó un 10 % aproximadamente comparado al modelo de ranger anterior.

```
Call:
 ranger(formula = Colesterol ~ Prevalencia_Diabetes + Hipertension +      Ansiedad_Depresion +
MC, data = train, num.trees = 1000,      mtry = 2, max.depth = 20, seed = 1234, importance = "i
purity",      probability = TRUE)

Type:                             Probability estimation
Number of trees:                  1000
Sample size:                      14153
Number of independent variables:  4
Mtry:                             2
Target node size:                 10
Variable importance mode:         impurity
Splitrule:                        gini
OOB prediction error (Brier s.):  0.1981188
```

Figura 24: Modelo Ranger mejorado

Teniendo en cuenta este último modelo de Ranger mejorado, vamos a realizar el gráfico de curva ROC. Vemos en el gráfico que el área bajo la curva es de un 65 % casi. Es decir hay un 65 % de probabilidad de que se diagnostique de forma correcta que una persona tiene colesterol o no. Podemos decir que este modelo de predicción es regular. También podemos observar que el punto de corte para que se clasifique a una persona con colesterol o no es 0.3.

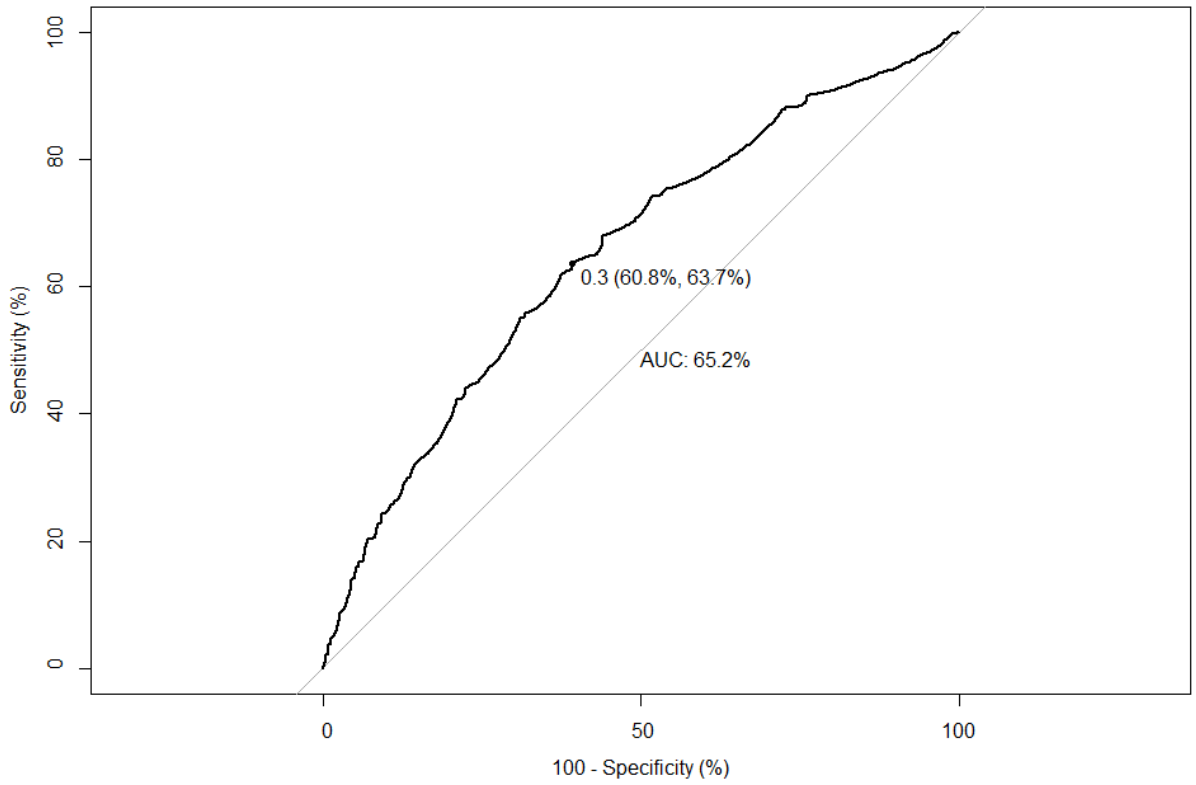

Figura 25: Gráfico de la curva ROC

6 | Visualización

Para finalizar, vamos a visualizar la variable dependiente en función de un par de variables independientes (las más importantes). La visualización se llevará a cabo sobre el set de datos original.

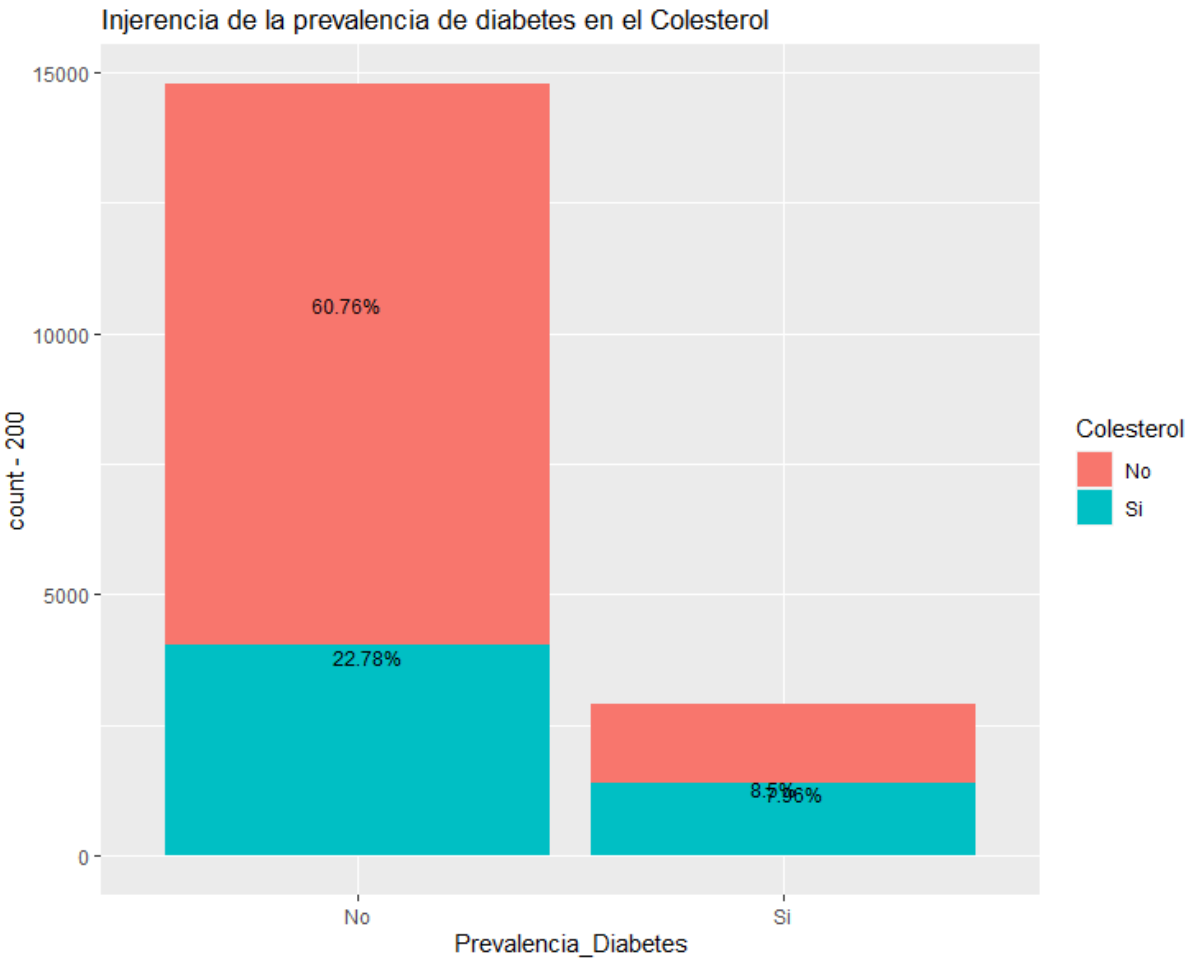

Figura 26: Existencia de colesterol en función de la prevalencia de diabetes

Como podemos ver en la figura 26, de los casos que no tienen diabetes, un 60.76 % no tiene colesterol y un 22.78 % si lo tienen, con lo que podemos decir que aquellos que no tengan diabetes probablemente tampoco tengan colesterol. No podemos decir lo mismo de los que si tienen diabetes, ya que para aquellos casos, el porcentaje de casos con o sin colesterol es aproximadamente igual.

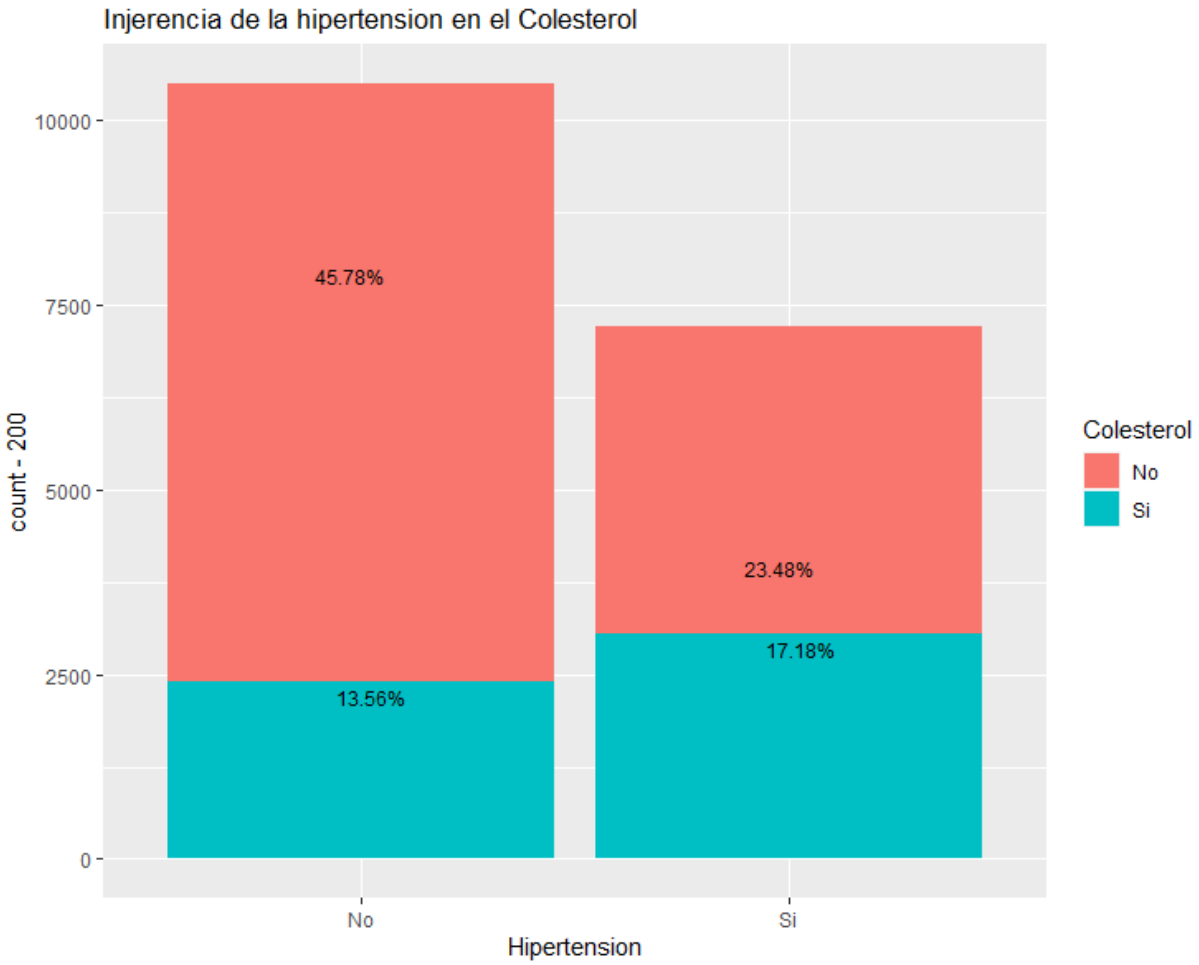

Figura 27: Existencia de colesterol en función de la existencia de hipertensión

Observando la figura 27, de los casos que no tienen hipertensión, un 45.78 % no tiene colesterol y un 13.56 % si lo tienen, con lo que podemos decir que aquellos que no tengan hipertensión probablemente tampoco tengan colesterol. No podemos decir lo mismo de los que si tienen hipertensión, ya que para aquellos casos, la diferencia entre el porcentaje de casos con o sin colesterol es baja.

Por medio de los gráficos, pudimos observar que el no tener diabetes o hipertensión hace que probablemente no se tenga colesterol, pero si algún caso tiene diabetes o hipertensión, no se puede afirmar con tanta seguridad que tengan o no colesterol. Esto se debe al desbalanceo de clases. Al no haber tantos casos con colesterol, hace que este par de variables elegidas (que son las más importantes según nuestro análisis previo) no predigan tan bien si un caso tiene o no colesterol.

7 | Conclusión

Entre los diferentes métodos de clasificación, hemos visto que los arboles de decisión son buenos a la hora de tener una visualización del modelo, aunque al ser aleatorios, con un mismo set de datos se pueden tener arboles muy distintos. Esto lo solucionamos utilizando Random Forest, que al crear muchos arboles, se reduce el efecto de la aleatoriedad en el modelo, aunque el costo computacional de implementarlo es alto. Por último, observamos que el algoritmo Ranger produce básicamente el mismo modelo que haría el algoritmo Random Forest, pero con un costo mucho menor.

También exploramos algunas técnicas para el tratamiento de clases desbalanceadas, demostrando que estos modelos de clasificación sufren mucho cuando tenemos clases desbalanceadas. A pesar de ello, no logramos obtener un modelo lo suficientemente robusto para predecir nuesta variable dependiente, como pudimos diagnosticar con la precisión, la especificidad, la sensibilidad y el área bajo la curva ROC.

En conclusión, ninguno de los modelos es lo suficientemente apto para poder predecir el colesterol, al menos con nuestro set de datos y con las técnicas de balanceo de clases. En un caso real, se tendrían que probar más técnicas o si es posible, buscar más datos de otras fuentes.

8 | Referencias

[1] Instituto Nacional de Estadística y Censos. *Dataset: Base de datos usuario ENFR 2018*. URL: https://www.indec.gob.ar/indec/web/Institucional-Indec-BasesDeDatos-2 (visitado 18-10-2022).

[2] *¿Qué es el IMC y cómo se relaciona con la obesidad?* URL: https://hablandodeobesidad.com/que-es-la-obesidad/que-es-imc-y-como-se-relaciona-con-la-obesidad/ (visitado 20-10-2022).

A | Anexo

A.1 | Código en R

```
#############################################################################
#
# Actividad 4: Arboles de decision.
# Fuente:
# https://www.indec.gob.ar/indec/web/Institucional-Indec-BasesDeDatos-2
# Alumnos: Correa, Matías; Esperanza, Marcelo Fabián; Ríos, Alejandro.
#
#############################################################################
#
# Utilizamos como dataset los datos contenidos en la Encuesta Nacional
# de Factores de Riesgo del año 2018, provisto por el INDEC
# (Instituto Nacional de Estadística y Censos).
#
#############################################################################

#### CONFIGURACION Y BIBLIOTECAS ####

rm(list = ls()) # Para limpiar la memoria
options(scipen = 6) # Para evitar notación científica

#### CARGA DE DATOS ####

##### Importamos las biliotecas #####

library(readr) #read_csv
library(randomForest) #turneRF()  randomForest()
library(caret) #createDataPartition()
library(ranger) #ranger()
library(rpart) #rpart()
library(rattle) #fancyRpartPlot() es otra opción
library(tidyr)  #expand_grid()
library(rpart.plot) #rpart.plot()
library(pROC) #plot.roc
library(ROCR) #roc
library(dplyr) #recode()

datos <- read_delim("Datos/encuesta.csv",
                    delim = ";", escape_double = FALSE, trim_ws = TRUE)

nrow(datos) # Cantidad de casos.
ncol(datos) # Cantidad de variables

# Numero de casos con null mediante iteracion con Sapply.
sapply(datos, function(x) sum(is.na(x)))

# Quitamos los casos con datos na
encuesta.salud <- na.omit(datos)

# Cantidad de casos luego de sacar los nulos.
nrow(encuesta.salud)

# Comprobamos el porcentaje de observaciones del nuevo dataframe
   respecto del original
```

```r
nrow(encuesta.salud) / nrow(datos)

# Convertimos en factor algunas variables
encuesta.salud <- encuesta.salud %>% mutate(hipertension = as.factor(
                                                    recode(
    hipertension,
    "SI" = "Si",
    "NO" = "No")),
                                            AnsiedadDepresion = as.
    factor(
                                                      recode
    (AnsiedadDepresion,
    "MUCHO" = "Mucho")),
                                            FumaFrecuencia = as.factor(
                                                      recode(
    FumaFrecuencia,
    "NingunDia" = "Ningun dia",
    "AlgunosDias" = "Algunos dias",
    "TodosLosDias" = "Todos los dias")),
                                            SalEnCoccion = as.factor(
    SalEnCoccion),
                                            Colesterol = as.factor(
    Colesterol),
                                            consumoRegAlcoholRiesgoso =
    as.factor(
      recode(consumoRegAlcoholRiesgoso,
             "SI" = "Si",
             "NO" = "No")),
                                            prevalencia_diabetes = as.
    factor(
    recode(prevalencia_diabetes,
             "SI" = "Si",
             "NO" = "No")),
                                            SalDespCoccion = as.factor(
                                                      recode(
    SalDespCoccion,
    "RaraVez" = "Rara vez")))
# Renombramos las variables para mayor legibilidad
names(encuesta.salud) <- c("Ansiedad_Depresion",
                           "Actividad_Intensa",
                           "Actividad_Moderada",
                           "Caminata",
                           "Sentado",
                           "Fuma_Frecuencia",
                           "Hipertension",
```

```
                              "IMC",
                              "Sal_En_Coccion",
                              "Sal_Desp_Coccion",
                              "Promedio_Frut_Ver_Diario",
                              "Colesterol",
                              "Cons_Reg_Alcohol_Riesgoso",
                              "Prevalencia_Diabetes")

# Categorizamos la variable IMC
imc.intervalos <- cut(encuesta.salud$IMC,
                      c(-Inf, 18.5, 25, 30, 35, 40, Inf), left=F)

encuesta.salud.clasificada <- mutate(encuesta.salud,
                                     IMC = imc.intervalos)

levels(encuesta.salud.clasificada$IMC) <- c("Peso bajo", "Peso Normal",
    "Sobrepeso",
                                            "Obesidad grado 1", "
    Obesidad grado 2",
                                            "Obesidad grado 3")

# Eliminamos los valores No sabe - No contesta
diabetes.sin.nsnc <- which(encuesta.salud.clasificada$Prevalencia_
    Diabetes
                           %in% "NSNC")

encuesta.salud.clasificada <- encuesta.salud.clasificada[-diabetes.sin.
    nsnc,]

# Comprobamos el porcentaje de observaciones del nuevo dataframe
    respecto del original
nrow(encuesta.salud.clasificada) / nrow(datos)

##### Definimos datos para entrenamiento y datos para test ######

set.seed(2022) # Número inicial a partir del cuál comenzará a generar
    una secuencia aleatoria.
# Tomamos un 80% para el entrenamiento y un 20% para el test
split <- createDataPartition(encuesta.salud.clasificada$Colesterol, p
    =0.8, list=FALSE) #List = Si el resultado devolverá una lista o una
    matriz.
train<- encuesta.salud.clasificada[split,]
test<-  encuesta.salud.clasificada[-split,]

############################################################
##### Modelado de árbol de desición aplicando RPART #####
############################################################

###### Genero un modelo 1 con rpart #####

tree <- rpart(Colesterol ~., data = train, method="class") # Indicamos
    que deseamos un arbol de clasificación
rpart.plot(tree) # Graficamos el árbol.
x11()
fancyRpartPlot(tree,type = 2) # Gráfico del árbol


###### Importancia de las variables ######
```

```
138
139 qplot(x = names(tree$variable.importance), y=tree$variable.importance,
140       xlab="Variable", ylab="Importancia", main="rpart - Importancia de
    las variables")
141
142 # Revisamos si hay desbalanceo de las clases
143 library(plotly)
144 qplot(encuesta.salud.clasificada$Colesterol, ylab="Observaciones", xlab=
    "Intervalos")
145
146 # El 69.25% de los casos no tienen colesterol
147 # El 30.75% tienen colesterol
148 prop.table(table(encuesta.salud.clasificada$Colesterol))
149
150
151 ###### Predicción del primer modelo ######
152
153 prediccion.rpart <- predict(tree, newdata=test, type="class")
154 # Precision: 70.03%
155 # Sensibilidad: 16.47%
156 # Especificidad: 93.8%
157 confusionMatrix(prediccion.rpart, test$Colesterol, positive = "Si")
158
159 ##### Tratamiento de clases desbalanceadas #####
160 library(ROSE)
161
162 # Cantidad de casos para cada clase
163 table(train$Colesterol)
164
165 ###### Sobre-muestreo ######
166 # Aumentamos la cantidad de "Si" creando copias de algunos casos
167 # seleccionados aleatoriamente
168 # N = cantidad de "No" * 2
169 over <- ovun.sample(Colesterol ~., data = train, method = "over", N =
    19602)$data
170
171 # Igual numero de clases
172 table(over$Colesterol)
173
174 # Comprobamos el nuevo modelo
175 tree.over <- rpart(Colesterol ~., data = over, method="class")
176
177 x11()
178 fancyRpartPlot(tree.over,type = 2)
179
180 prediccion.over <- predict(tree.over, newdata = test, type="class")
181 # Empeoramos la precision (de 70.03% a 61.07%), la especificidad empeoro
      (de 93.8% a 60.2%)
182 # pero mejoramos la sensibilidad (de 16.47% a 63.02%)
183 confusionMatrix(prediccion.over, test$Colesterol, positive = "Si")
184
185 ###### Sub-Muestreo ######
186 # Reducimos la cantidad de "No" de forma aleatoria, hasta igualar a los
    "Si"
187 # N = cantidad de "Si" * 2
188 under <- ovun.sample(Colesterol ~., data = train, method = "under", N =
    8704)$data
189
```

```r
# Igual numero de clases
table(under$Colesterol)

# Comprobamos el nuevo modelo
tree.under <- rpart(Colesterol ~., data = under, method="class")

x11()
fancyRpartPlot(tree.under,type = 2)

prediccion.under <- predict(tree.under, newdata = test, type="class")
# Mejoro la precision (de 61.07% a 62.06%), la especificidad mejoro (de
   60.2% a 62.29%)
# pero empeoro la sensibilidad (de 63.02% a 61.55%)
confusionMatrix(prediccion.under, test$Colesterol, positive = "Si")


##### Ambos (Sobre y Sub-Muestreo) #####
# Se reduce la cantidad de "No" y se aumenta la de "Si"
# N = cantidad de "Si" + cantidad de "No"
ambos <- ovun.sample(Colesterol ~., data = train, method = "both",
                     p = 0.5,
                     seed = 111,
                     N = 14153)$data

# Distinto numero de clases pero mucho mas balanceado
table(ambos$Colesterol)

# Porcentaje de cada clase
prop.table(table(ambos$Colesterol))

# Comprobamos el nuevo modelo
tree.ambos <- rpart(Colesterol ~., data = ambos, method="class")

x11()
fancyRpartPlot(tree.ambos,type = 2)

prediccion.ambos <- predict(tree.ambos, newdata = test, type="class")
# Empeoro la precision (de 62.06% a 61.07%), la especificidad empeoro (
   de 62.29% a 60.2%)
# pero mejoramos la sensibilidad (de 61.55% a 63.02%)
confusionMatrix(prediccion.ambos, test$Colesterol, positive = "Si")

##### Muestreo Sintetico con ROSE #####
# N puede ser cualquier numero, nosotros elegimos 20000
rose <- ROSE(Colesterol ~., data = train, seed = 222, N = 20000)$data

# Distinto numero de clases pero mucho mas balanceado
table(rose$Colesterol)

# Porcentaje de cada clase
prop.table(table(rose$Colesterol))

# Resumen
summary(ambos)
summary(rose)

# Comprobamos el nuevo modelo
tree.rose <- rpart(Colesterol ~., data = rose, method="class")
```

```
246
247 x11()
248 fancyRpartPlot(tree.rose,type = 2)
249
250 prediccion.rose <- predict(tree.rose, newdata = test, type="class")
251 # La precision, sensibilidad y especificidad se mantienen iguales
252 confusionMatrix(prediccion.rose, test$Colesterol, positive = "Si")
253
254 #### Resumen de los modelos ####
255
256 ##### Sin tratamiento de desbalanceo -----
257 # Precision: 70.03%
258 # Sensibilidad: 16.47%
259 # Especificidad: 93.8%
260
261 ##### Con Sobre-Muestreo
262 # Precision: 61.07%
263 # Sensibilidad: 63.02%
264 # Especificidad: 60.2%
265
266 ##### Con Sub-Muestreo
267 # Precision: 62.06%
268 # Sensibilidad: 61.55%
269 # Especificidad: 62.29%
270
271 ##### Ambos (Sobre y Sub-Muestreo)
272 # Precision: 61.07%
273 # Sensibilidad: 63.02%
274 # Especificidad: 60.2%
275
276 ##### Muestreo Sintetico con ROSE
277 # Precision: 61.07%
278 # Sensibilidad: 63.02%
279 # Especificidad: 60.2%
280
281 ###################################
282 # IMPLEMENTACION DE RANDOM FOREST #
283 ###################################
284
285 # ***** Aplicación de Random Forest *****
286
287 ##### MODELO 1 #####
288 # Genero el primer modelo seteando número de árboles igual a 700 y nú
    mero de variables igual a 2.
289 rf1 <- randomForest( Colesterol ~.
290                                 , data= train       # datos para
    entrenar
291                                 , ntree= 700    # cantidad de arboles
292                                 , mtry=2        # cantidad de
    variables
293                                 , replace = T            # muestras
    con reemplazo
294                                 , importance=T        # para poder
    mostrar la importancia de cada var
295                                 , class = NULL)
296
297
298 print(rf1)  #OOB estimate of  error rate: 29.98%
```

```
299
300 # Se estudia el error OOB (out-of-bag error): es el error promedio para
    cada observacion
301 # usando predicciones de los arboles que no contienen a dicha
    observacion.
302
303 ## Predicción del modeloRF1 ##
304 rf1.predict <- predict(rf1, newdata=test, type="class")
305
306 # Precision: 70.12%
307 # Sensibilidad: 12.51%
308 # Especificidad: 95.67%
309 confusionMatrix(rf1.predict, test$Colesterol, positive = "Si")
310
311
312 ## Gráfico del error OOB en modeloRF1 ##
313 qplot(y=rf1$err.rate[,1], main="randomForest, Error out-of-bag, 700
    arboles, 2 variables",
314       ylab="Error OOB", xlab="Cantidad de arboles")
315
316
317 ##### MODELO 2 #####
318 # Genero el primer modelo seteando número de árboles igual a 300 y nú
    mero de variables igual a 2.
319 rf2 <- randomForest(Colesterol ~ ., data=train, ntree=300, mtry = 2)
320 print(rf2) #         OOB estimate of  error rate: 30.01% aumenta
321
322 ## Predicción del modeloRF2 ##
323 rf2.predict <- predict(rf2, newdata=test, type="class")
324
325 # Precision: 70.17%
326 # Sensibilidad: 12.05%
327 # Especificidad: 95.96%
328 confusionMatrix(rf2.predict, test$Colesterol, positive = "Si")
329
330
331 ## Gráfico del error OOB en modeloRF2 ##
332 qplot(y=rf2$err.rate[,1], main="randomForest, Error out-of-bag, 200
    arboles, 2 variables",
333       ylab="Error OOB", xlab="Cantidad de arboles")
334
335
336 ##### MODELO 3 #####
337 # Creamos un modelo con el dataset con las clases balanceadas
338 rf3 <- randomForest(Colesterol ~ ., data=ambos, ntree=300, mtry = 4)
339
340 print(rf3) # OOB error: 13.63%
341
342 rf3.predict <- predict(rf3, newdata=test, type="class")
343
344 # Precision: 62.68%
345 # Sensibilidad: 46.64%
346 # Especificidad: 69.8%
347 confusionMatrix(rf3.predict, test$Colesterol, positive = "Si")
348
349 ## Gráfico del error OOB en modeloRF3 ##
350 qplot(y=rf3$err.rate[,1], main="randomForest, Error out-of-bag, 300
    arboles, 4 variables",
```

```
      ylab="Error OOB", xlab="Cantidad de arboles")


###### IMPORTANCIA DE LAS VARIABLES modeloRF1 ######

#La gráfica Mean Decrease Accuracy expresa cuánta
#precisión pierde el modelo al excluir cada variable
#Cuanto más sufre la precisión, más importante es la variable
#para la clasificación exitosa
#La disminución media del coeficiente de Gini es una medida de cómo cada variable contribuye
#a la homogeneidad de los nodos y hojas en el bosque aleatorio resultante.
#Cuanto mayor sea el valor de la precisión de disminución media o la puntuación de Gini
#de disminución media, mayor será la importa
x11()
varImpPlot(rf1, sort = T, n.var = 13 , main = 'Top 13 importancia de las variables')

### Seleccionamos las 4 variables más importantes ###

rf4 <- randomForest( Colesterol ~ Prevalencia_Diabetes
                                  + Hipertension + Ansiedad_Depresion
                                  + IMC
                                  , data= train       # datos para entrenar
                                  , ntree= 300    # cantidad de arboles
                                  , mtry=2        # cantidad de variables
                                  , replace = T            # muestras con reemplazo
                                  , importance=T        # para poder mostrar la importancia de cada var
                                  , class = NULL
)
#OOB estimate of  error rate: 29.58%

print(rf4)
prediccion.rf4 <- predict(rf4, newdata=test, type="class")

# Precision: 70.34%
# Sensibilidad: 14.81%
# Especificidad: 94.98%
confusionMatrix(prediccion.rf4, test$Colesterol, positive = "Si")

#######################################
# UTILIZACION DE RANGER Y CURVAS ROC #
#######################################

# Ranger es una implementacion rapida de bosques aleatorios para R,
# útil cuando la cantidad de variables es grande.
# Pruebo con implementacion ranger, parametros por defecto (500 arboles, 2 variables)
# Error OOB de aproximadamente  29.76 %
# Varía entre ejecución y ejecución.

ranger1 <- ranger(formula = Colesterol ~ Prevalencia_Diabetes
```

```
                    + Hipertension + Ansiedad_Depresion
                    + IMC,
                    data=train)
ranger1 # equivalente a : print(modeloRanger1)
prediccion.ranger1 <- predict(ranger1, data=test)

# Precision: 70.34%
# Sensibilidad: 14.81%
# Especificidad: 94.98%
confusionMatrix(prediccion.ranger1$predictions, test$Colesterol,
    positive = "Si")

# Se observan unos resultados practicamente similares a los anteriores,
#la ganancia se ve reflejada en los tiempos de ejecucion que son menores
    .

# Creamos un segundo modelo tomando mas arboles
ranger2 <- ranger(
  formula    = Colesterol ~ Prevalencia_Diabetes
               + Hipertension + Ansiedad_Depresion
               + IMC,
  data       = train,
  num.trees  = 1000,
  mtry       = 2,
  max.depth  = 20,
  seed       = 1234,
  importance= 'impurity',
  probability= TRUE
)

ranger2 # OOB error: 19.81%

roc_train <- roc(train$Colesterol, ranger2$predictions[,2], percent=
    TRUE,
                 auc= T, CI= T, plot= T )

plot.roc(roc_train, legacy.axes = T, print.thres = "best", print.auc = T
    )

print(roc_train)


##################
# Visualización #
##################

figura_1 <- ggplot(encuesta.salud.clasificada, aes(x = Prevalencia_
    Diabetes, fill = Colesterol)) +
  geom_bar() +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%'
    )),
            stat = 'count',
            position = position_dodge(.1),
            size = 3) +
  labs(title="Injerencia de la prevalencia de diabetes en el Colesterol"
    )

```

```
451 figura_1
452
453 figura_2 <- ggplot(encuesta.salud.clasificada, aes(x = Hipertension,
       fill = Colesterol)) +
454   geom_bar() +
455   geom_text(aes(y = ..count.. -200,
456                 label = paste0(round(prop.table(..count..),4) * 100, '%'
       )),
457             stat = 'count',
458             position = position_dodge(.1),
459             size = 3) +
460   labs(title="Injerencia de la hipertension en el Colesterol")
461
462 figura_2
```