MÁSTER UNIVERSITARIO EN ESTADÍSTICAS OFICIALES E INDICADORES SOCIALES Y ECONÓMICOS

UNIVERSIDAD
COMPLUTENSE
MADRID

TRABAJO FIN DE MÁSTER

CURSO 2018-2019

Uso de técnicas de aprendizaje automático para la depuración selectiva de variables categóricas

APELLIDOS Y NOMBRE: Santiago Iglesias, Enrique

DNI: 52000478P

TUTOR/A: Salgado Fernández, David / Rosa Pérez, Elena

Contenido

RESUMEN

En los últimos años ha habido una gran revolución en el mundo del análisis de datos, potenciado sobre todo por el avance del procesamiento computacional, el refinamiento de los algoritmos para optimizar su funcionamiento, nuevos métodos de recogida y almacenamiento de datos más económico. Todo esto ha conducido a la aparición de nuevas tecnologías con el fin de tratar estos datos y sacar el mayor rendimiento de ellos con infinidad de aplicaciones que van desde investigaciones en medicina, publicidad orientada a cada individuo, seguridad en nuestras cuentas del banco, procesamiento de imágenes, producción de energía. Estos son unos pocos ejemplos de la multitud de campos que usan las nuevas tecnologías y las técnicas de aprendizaje automático.

En el caso de este TFM se van a utilizar las técnicas de aprendizaje automático con el fin de mejorar la eficacia y calidad del proceso de depuración e imputación en la producción de estadísticas oficiales. Dentro de la depuración e imputación de los datos se centra en la depuración selectiva para variables cualitativas, ya que actualmente no existe una metodología para estas variables. Para ello se van a aplicar varias técnicas de aprendizaje automático conocidas (Random Forest, Neural Network y Support Vector Machine) a los datos de la Encuesta Nacional de Salud correspondiente al año 2011 sobre la variable objetivo CNO (Código Nacional de Ocupación) para tratar de obtener una ordenación de las unidades estadísticas con más probabilidad de error para su depuración e imputación.

ABSTRACT

In the last few years, the world of the data analysis has been a great revolution due to the progress of computational processing, the improvement of the algorithms, new methods of data collection and data storage. All of this has led to the emergence of new technologies in order to process these data and getting the best performance of them, using them in countless applications: medical research, marketing oriented to each individual, security in our bank accounts, processing of images, energy production. These are a few examples of the areas where you can use the new technologies and machine learning techniques.

In this TFM, we are going to use the machine learning techniques in order to improve the efficiency and quality of the Editing and Imputation process in the official statistical production. In the Editing and Imputation of the data, we are going to work with categorical variables because currently, there is not methodology to work with these variables. We are going to apply the machine learning techniques (Random Forest, Support Vector Machine and Neural Network) to the data of the National Health Survey corresponding to 2011. The analysis variable of this TFM is going to be the CNO (National Occupation Code) in order to get the data sorted according to the units with more probability of error.

INTRODUCCIÓN

Las estadísticas oficiales son el espejo de cualquier sociedad, con la información que se genera en su producción se puede saber la estructura de la sociedad, cómo se compone, cuánto de desarrollado está un país, el estado económico del mismo, etc. Es la fuente en la que se apoya las empresas, las administraciones públicas y el gobierno a la hora de tomar decisiones.

El INE tiene dos métodos de obtener la información, a través de los registros administrativos o a través de las encuestas o censos (seleccionando una muestra o del total de la población objetivo del estudio). En este trabajo se va a utilizar la información que proviene de las encuestas. Algunas de ellas se reciben tarde (fuera de plazo), o vienen con variables que se encuentran vacías o que contienen errores, en la mayoría de los casos debido a desconocimiento de la persona que lo rellena o errores de procesamiento. Debido a estos problemas, existe un proceso, dentro de la producción estadística, que se dedica a la depuración e imputación de estos valores: GSBPM-5.4 Edit & Impute[1], que es en el cual se centra este TFM.

La depuración e imputación (para abreviar se utilizará "E&I" del inglés Editing and Imputation) es un conjunto de actividades con el objetivo de detectar los errores y las variables vacías para posteriormente tratar estos casos y asignarles un dato de manera acorde. E&I es un importante subproceso de la producción estadística donde se consumen muchos recursos y se puede realizar en diferentes fases del proceso estadístico: recogida de los datos, en la grabación de los datos y en el procesamiento de los datos. Los principales objetivos de E&I son:

- Identificar errores en las fuentes de datos, para mejorar el proceso estadístico
- Proveer información sobre la calidad de los datos.
- Identificar y tratar los errores más significativos.
- Cuando sea necesario, proveer información, a nivel individual, completa y consistente.

[1] Estándar internacional desarrollado por Naciones Unidas que describe y define el conjunto de procesos necesarios para la producción de la estadística oficial.
https://statswiki.unece.org/display/GSBPM/GSBPM+v5.1

Existen diferentes tipos de depuración e imputación ([1] – EDIMBUS, 2007):

- Depuración Selectiva.
- Depuración Interactiva.
- Depuración Automática.
- Macro-depuración.

Existen diferentes estrategias a seguir para el proceso de E&I. En la ilustración 1 se puede ver una de las estrategias[2] que se puede seguir:

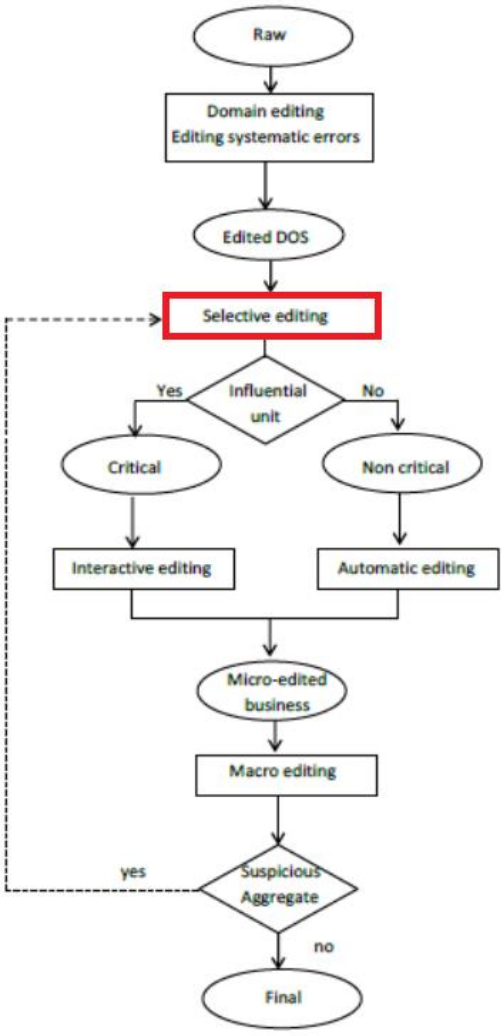

Ilustración 1: Fuente [1] EDIMBUS, 2007

[2] Estrategias incluidas en EDIMBUS https://ec.europa.eu/eurostat/documents/64157/4374310/30-Recommended+Practices-for-editing-and-imputation-in-cross-sectional-business-surveys-2008.pdf/6e51b229-8628-422d-8c4c-7ede411e107f

El trabajo se centra en la depuración selectiva ("Selective Editing" en la estrategia propuesta), donde las principales funciones de producción que se abordan en este punto del proceso son:

- Identificar errores influyentes, es decir, aquellos que tienen un gran impacto en el resultado final de los agregados.
- Optimizar los recursos disponibles (tanto humanos como económicos), para focalizar el trabajo a aquellas unidades identificadas en el punto anterior y que mejorarían la calidad de los datos.

Los trabajos sobre depuración selectiva, que se han realizado hasta el momento tanto a nivel nacional como internacional, sólo han tenido en cuenta variables cuantitativas, por lo que la principal idea de este análisis es investigar si la introducción de los mecanismos de "Machine Learning" en el proceso de depuración selectiva para variables categóricos es viable y ver si se obtienen unos resultados aceptables. De esta forma se garantiza que se puede hacer una ordenación de los errores en las unidades estadísticas, y poder tomar la decisión de qué unidades van a ser revisadas en la depuración interactiva ("Interactive Editing" en la estrategia propuesta) para aumentar la calidad y mejorar la eficiencia del proceso.

Datos utilizados

Para este fin, se va a trabajar con datos ya tratados y anonimizados, debido al tratamiento de datos personales y su confidencialidad, de la Encuesta Nacional de Salud correspondiente al año 2011, con una muestra de 18.486 individuos. El contenido del fichero es el siguiente:

CAMPO	DESCRIPCIÓN
IDENTHOGAR	Número de identificación del hogar: Sección + Vivienda + Hogar
EDADa	Adulto seleccionado: Edad
PROXY_0	¿El informante es la persona seleccionada?
A7_2a	Número de orden del adulto seleccionado
SEXOa	Adulto seleccionado: Sexo
F7_2	Adulto seleccionado: Actividad de la empresa en la que trabajó la persona que generó la pensión (código CNAE2009, 3 dígitos)
F16a_2	Adulto seleccionado: Actividad de la empresa donde trabaja actualmente (código CNAE2009, 3 dígitos)
F16m_2	Adulto seleccionado: Actividad de la empresa en la que trabajó (código CNAE2009, 3 dígitos)

F17a_2	Adulto seleccionado: Ocupación, profesión u oficio actual (código CNO2011, 3 dígitos)
F17m_2	Adulto seleccionado: Última ocupación, profesión u oficio (código CNO2011, 3 dígitos)
F8_2	Adulto seleccionado: Ocupación, profesión u oficio de la persona que generó la pensión (código CNO2011, 3 dígitos)
F18	Adulto seleccionado: Situación profesional de su último empleo
F9	Adulto seleccionado: Situación profesional en la ocupación que desempeñó la persona que generó la pensión
CCAA	Adulto seleccionado: Comunidad Autónoma de residencia
ESTRATO	Adulto seleccionado: Estrato al que pertenece el hogar (por tamaño de municipio)
FACTORADULTO	Adulto seleccionado: El peso de muestreo del individuo.
CNAE_AS	Adulto seleccionado: Actividad económica (grupo CNAE2009)
CNAE1_AS	Adulto seleccionado: Actividad económica (sección CNAE2009)
CNAE2_AS	Adulto seleccionado: Actividad económica (división CNAE2009)
CNO_AS_raw	Adulto seleccionado: Ocupación (Subgrupo CNO-11) - Información original del cuestionario
CNO_AS_ed	Adulto seleccionado: Ocupación (Subgrupo CNO-11) - Información después de ser editada en el proceso E&I
CNO1_AS_raw	Adulto seleccionado: Ocupación (Gran grupo CNO-11) - Información original del cuestionario
CNO1_AS_ed	Adulto seleccionado: Ocupación (Gran grupo CNO-11) - Información después de ser editada en el proceso E&I
CNO2_AS_ed	Adulto seleccionado: Ocupación (Subgrupo principal CNO-11) - Información original del cuestionario
CNO2_AS_raw	Adulto seleccionado: Ocupación (Subgrupo principal CNO-11) - Información después de ser editada en el proceso E&I
target1	Variable binaria que indica si la variable CNO1_AS es errónea y ha sido modificada
target2	Variable binaria que indica si la variable CNO2_AS es errónea y ha sido modificada
target3	Variable binaria que indica si la variable CNO_AS es errónea y ha sido modificada

El objeto de estudio es el Código según la clasificación Nacional de Ocupación (CNO2011[3]). El Código Nacional de Ocupación (CNO) tiene 3 dígitos en los datos que se utilizan. Se encuentra agrupado en 3 niveles de detalle: Gran Grupo (1 dígito "CNO1_AS"), Subgrupo principal (2 dígitos "CNO2_AS") o Subgrupo (3 dígitos "CNO_AS"). En cada uno de los niveles se tienen dos variables "_raw" correspondiente a la información que viene de origen y "_ed" el código asignado una vez pasado por el proceso de E&I. En la mayoría de los casos la información que viene de origen y la obtenida en el proceso de E&I es la misma, pero algunos códigos CNO han sido identificados como erróneos y se les ha asignado otro código CNO. Las 3 variables target (target1, target2 y target3) son variables dicotómicas que identifican los códigos CNO que han sido modificados en los 3 niveles de detalle. En la siguiente tabla se puede ver un ejemplo:

[3] Listado completo para el Código Nacional de Ocupación: https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736177033&menu=ultiDatos&idp=1254735976614

CNO Original	CNO Editado	target1	target2	target3
259	430	1	1	1
500	512	0	1	1
521	522	0	0	1

Las variables target se van a utilizar para la construcción de los modelos con el fin de asignar a cada individuo la probabilidad de que el código CNO incluido en el cuestionario no sea el correcto para los diferentes niveles de agregación. Una vez se tienen las probabilidades de error de cada individuo, se puede realizar una ordenación de los individuos con más probabilidad de error, para poder ser tratados en la fase de depuración interactiva. Se siguen dos criterios de ordenación:

- Probabilidad de Error: Se ordenan los individuos según la probabilidad de error encontrada en cada caso.

- Momento de Error: En este criterio de ordenación se tienen en cuenta los pesos de muestreo (campo “FACTORADULTO” de los datos de entrada), por lo que multiplica los pesos de muestro de cada individuo por su probabilidad de error para realizar la ordenación.

METODOLOGÍA

Depuración Selectiva

La depuración selectiva (Selective Editing), desarrollada hasta ahora para variables cuantitativas, se compone de 4 fases:

1- Construcción de los valores "Anticipados": $\hat{y}_k$

Los valores anticipados $\hat{y}_k$ son, en general, predicciones con no mucha calidad. Normalmente se toman como valores anticipados los valores del periodo anterior:

$\hat{y}_k = \hat{y}_k^{(*,t-1)}$ o si consideramos estacionalidad: $\hat{y}_k = \hat{y}_k^{(*,t-s)}$

2- Construcción de los valores "Local Score": s_k

Para obtener los Local Score, se utilizan las funciones Local Score, donde la estructura genérica de estas funciones es:

$s_k(y_k, \hat{y}_k) = F_k(y_k, \hat{y}_k) \times R_k(y_k, \hat{y}_k)$

Donde k es la unidad estadística, y es una variable cuantitativa, F_k es la componente que indica la influencia o contribución de la unidad k a la hora de estimar los agregados y R_k es el riesgo que indica la posibilidad de que en la unidad k se presente algún error.

Normalmente las funciones Local Score suelen construirse sobre la comparación: $s_k(y_k, \hat{y}_k) = w_k|y_k - \hat{y}_k|$ donde w_k denota el peso de muestreo de la unidad k.

3- Construcción de los valores "Global Score": S_k

Una vez que se tienen los valores Local Score de las variables de interés, entra en juego las funciones Global Score, que se utilizan para priorizar las observaciones. Se pueden utilizar bastantes funciones para obtener los valores Global Score, un ejemplo son las funciones de Minkowski (con pesos):

$$S_k^{(\alpha)} = \begin{cases} \left(\sum_{p=1}^{P} w_p \left[s_k^{(p)} \right]^{\alpha} \right)^{1/\alpha}, & para\ \alpha \neq \infty \\ max\left\{ w_1 s_k^{(1)}, \dots, w_P s_k^{(P)} \right\}, & para\ \alpha = \infty \end{cases}$$

4- Determinación del valor umbral: C_0

Una vez que se tienen los valores Global Score, es necesario determinar un umbral C_0 tal que, la unidad k es depurada si y solo si el valor Global Score ($S_k^{(\alpha)}$) es mayor que el umbral determinado C_0. Para determinar el umbral a elegir, se procede del siguiente modo:

- Se determina el Global Score ($S_k^{(\alpha)}$) para cada unidad k.
- Se elige un primer umbral $C_0^{(1)}$ con un valor alto, y se sustituye en todas las unidades k tales que $S_k^{(\alpha)} \geq C_0^{(1)}$, el valor que viene de origen por un valor depurado. De este paso, obtenemos un primer conjunto de unidades depuradas $E(C_0^{(1)})$.
- Se calcula los agregados usando el conjunto de unidades depuradas.
- Elegimos un nuevo valor umbral más bajo $C_0^{(2)}$, para obtener un nuevo conjunto de unidades $E(C_0^{(2)})$.
- Se repite este proceso n veces, para luego representar los resultados y elegir el valor C_0 más conveniente siguiendo la recomendación de un experto en la materia.

Como se ha comentado en la introducción, este proceso de depuración selectiva ha sido diseñado para variables cuantitativas, ya que la diferencia $w_k|y_k - \hat{y}_k|$ para la construcción de las Local Score no tiene sentido cuando se trabaja con variable categóricas. Actualmente no hay ninguna metodología para variables categóricas. Lo que se propone con este trabajo, es poder iniciar un camino metodológico para la optimización de los procesos en la producción estadística cuando se trabaja con variables categóricas.

La depuración selectiva para variables categóricas tendría las mismas fases:

1. Predicción de la probabilidad de error: $\mathcal{P}(y_k \neq y_k^0)$

En lugar de obtener un valor aproximado de la cifra real, como se realiza en el método para variables cuantitativas, se estima con qué probabilidad la variable de estudio es errónea $\mathcal{P}(y_k \neq y_k^0)$, donde:

- y_k : Valor de la variable de los datos de origen.
- y_k^0 : Valor real de la variable.

También se podría dar una predicción de la clase a la que correspondería el individuo, pero dependiendo de las categorías de la variable de estudio se tendría mucha más dificultad.

2. Construcción de los valores "Local Score": s_k

Por tanto, como se está trabajando con variables categóricas, no se puede calcular la diferencia $w_k|y_k - \hat{y}_k|$, en este caso, como Local Score se consideran las probabilidades de error que tiene la variable de estudio:

$$s_k(y_k, \hat{y}_k) = w_k \cdot \mathcal{P}(y_k \neq y_k^0)$$

Donde w_k denota los pesos de muestreo.

3. Construcción de los valores "Global Score": S_k

En esta fase no habría modificaciones, ya que disponemos de los valores numéricos de las Local Score. En este TFM sólo estamos tratando con una variable por lo que el cálculo de las Global Score no se va a aplicar, pero en el caso de que se desee mirar varias variables, el cálculo de los Global Score sería el mismo.

4. Determinación del valor umbral: C_0

Igual que en la fase anterior, no habría modificaciones. Para este trabajo, lo que se plantea es una ordenación de los individuos para su posterior depuración, por lo que el umbral dependerá de los recursos disponibles (económicos, tiempo, humanos, …) del departamento encargado de la depuración e imputación de los datos.

Lo que se desea es minimizar el error de no haber depurado una unidad errónea en la fase de E&I, en otras palabras, si se desea depurar un conjunto de datos y los recursos nos limitan el número de unidades que se puede revisar, entonces es necesario centrarse en las unidades más influyentes con más probabilidad de error.

Random Forest

Los **árboles de decisión** son una técnica de Machine Learning de tipo supervisado. La información de entrada y salida pueden ser variables tanto cuantitativas como categóricas. Si se desea construir un árbol de decisión, hay dos pasos:

1- Dividimos el espacio de las predicciones, que es el conjunto de los posibles valores que puede tomar la variable a estudiar, en J regiones distintas que tengan intersección nula: $R_1, R_2, R_3, \dots, R_J$ tal que $R_1 \cap R_2 \cap R_3 \cap \dots \cap R_J = \emptyset$

2- Para cada observación que caiga en la región R_j, la predicción que se realizaría sería la media de las observaciones de los datos de entrenamientos que están en dicha región.

La respuesta predicha por un árbol de clasificación, si se está trabajando con variables cualitativas, es dada por la proporción de las respuestas de los datos de entrenamiento. Entonces, para un árbol de clasificación, se puede decir que cada observación pertenece a la clase que más ocurrencia tenga en el conjunto de datos de entrenamiento de la región a la que pertenece. Normalmente el interés no sólo se centra en la predicción en sí, si no, en la proporción de categorías posibles que se encuentran en una región.

Para la construcción de las regiones se divide el espacio de las predicciones en rectángulos o cajas con el fin de minimizar la Suma de los Cuadrados de los Residuos (SCR) dada por:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Donde $\hat{y}_{R_j}$ es la media de las respuestas de los datos de entrenamiento que se encuentran en la región R_j. Computacionalmente es muy complicado, por lo que se utiliza una aproximación que se conoce como "recursive binary splitting", en la que al principio se empieza en una sola región en la que todas las observaciones se encuentran en dicha región, y a continuación se divide el espacio en dos regiones y así sucesivamente hasta obtener las regiones finales o terminales. La mejor partición se realiza en cada caso en particular, es decir, se mira la mejor partición para ese momento, sin tener en cuenta las futuras particiones para obtener una partición final óptima. Para llevar a cabo el "recursive binary splitting", primero se selecciona el predictor X_j y el punto de corte s tal que al dividir el espacio de las predicciones en las regiones $\{X|X_j < s\}$ y $\{X|X_j \geq s\}$ lleva a la mejor reducción de la SCR, entonces se consideran todos los predictores $X_1, X_2, \dots, X_p$ y todos los posibles valores del punto de corte s para cada uno de los predictores, y se elige el predictor y el punto de corte que minimice el SCR.

Para desarrollar un árbol de decisión con variables categóricas no es posible usar el SCR como criterio para hacer la división de las regiones. La alternativa que se utiliza es la "classification error rate", que asigna la observación a la región donde la clase se presenta con más frecuencia según los datos de entrenamiento, el **error de clasificación** se define como:

$$E = 1 - \max_k(\hat{p}_{mk})$$

Donde $\hat{p}_{mk}$ representa la proporción de observaciones de los datos de entrenamiento en la m-ésima región que pertenecen a la k-ésima clase, en definitiva, es el conjunto de observaciones de los datos de entrenamiento que en la región estudiada no pertenecen a la clase común. Este error de clasificación no es suficientemente bueno para desarrollar un árbol de decisión, por lo que en la práctica se utilizan otras dos medidas:

- **Índice de Gini**: Que se define como:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

  Es una medida del total de la varianza a lo largo de todas las K clases. Un valor pequeño en el índice indica que una región contiene observaciones predominantes de una sola clase.

- **Cross-Entropy**, que se define como:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \cdot \log \hat{p}_{mk}$$

  Si $0 \leq \hat{p}_{mk} \leq 1$, entonces $0 \leq -\hat{p}_{mk} \cdot \log \hat{p}_{mk}$. Como en el índice de Gini, tomará valores pequeños cuando la región m-ésima sólo contenga observaciones predominantes de una sola clase.

Cuando se construye un árbol de clasificación, el índice de Gini o el cross-entropy son los más usados para evaluar cada una de las divisiones, ya que estas dos aproximaciones son más precisas que el error de clasificación.

Tras la creación del árbol, las observaciones de entrenamiento quedarán clasificadas en las regiones finales o terminales. Si se dispone de una nueva observación del conjunto de datos a testear, irá recorriendo el árbol en función del valor de sus variables productoras hasta llegar a una de las regiones terminales.

Los árboles de decisión son métodos que tienen un gran inconveniente: la varianza que tienen es alta, por lo que para minimizar la varianza de los árboles de decisión se puede aplicar **bootstrap** o **bagging**. La manera de aplicar bagging es obtener diferentes muestras de un mismo

conjunto de datos de entrenamiento, entonces si se obtienen N diferentes muestras del conjunto de datos de entrenamiento y calculamos las predicciones $\widehat{f_1}(x), \widehat{f_2}(x), \dots, \widehat{f_N}(x)$,la media de dichas predicciones estará dada por:

$$\widehat{f_{md}}(x) = \frac{1}{N}\sum_{i=1}^{N}\widehat{f_i}(x)$$

La varianza de la media de las predicciones es mucho menor que la varianza de las predicciones por separado. Para los árboles de decisión, lo que se hace es la construcción de N árboles de decisión, como se está trabajando con variables cualitativas, se obtienen N predicciones de los N árboles construidos y se selecciona la categoría más predicha. Con bagging se gana mucho en la precisión de las predicciones, pero su inconveniente es que la interpretación del modelo es mucho más difícil.

El bagging tiene un problema: la correlación entre los árboles de decisión. Por ejemplo, si tenemos un predictor muy fuerte en el conjunto de datos de entrenamiento, entonces la mayoría de los árboles de decisión generados en el bagging usarán este predictor como separador principal de las categorías. Esto implica que la mayoría de los árboles serán muy parecidos, y tendrán una correlación muy alta, por lo que tener un solo árbol de decisión o tener N árboles es muy parecido. Aquí es donde entra en juego los Random Forest.

Los Random Forest consideran en cada separación del árbol sólo un subconjunto "*m*" aleatorio de predictores de los "*p*" totales predictores disponibles. Entonces de media $\frac{(p-m)}{p}$ de las separaciones de los árboles no considerarán el predictor fuerte y los otros predictores tendrán más oportunidad. La principal diferencia entre el Random Forest y el bagging es la elección del tamaño del subconjunto m de predictores, por lo que si m=p bagging y Random Forest serán iguales.

Support Vector Machine (SVM)

El método "Support Vector Machine" (SVM) fue desarrollado, dentro del campo de la ciencia computacional, en los noventa y ha crecido en popularidad desde entonces. Empezó siendo un clasificador binario, pero sus aplicaciones han mejorado y se puede aplicar a problemas de clasificación múltiple y regresión. El SVM es la generalización de un simple e intuitivo clasificador llamado "Maximal Margin Classifier".

- Maximal Margin Classifier

Esta técnica de clasificación está basada en el concepto de los hiperplanos. En un espacio *p*-dimensional, un hiperplano es un subespacio de dimensión *p*-1. Por ejemplo, en un espacio de dos dimensiones un hiperplano es un subespacio de una dimensión por lo que sería una recta, en un espacio de dimensión tres sería un plano, etc. Cuando pasamos de dimensión tres, es difícil de imaginar.

La definición matemática de un hiperplano en un espacio *p*-dimensional se define como:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + +\beta_p X_p = 0$$

Si un punto $X = (X_1, X_2, \dots, X_p)^T$ satisface la ecuación del hiperplano, entonces X cae en el hiperplano, pero si por el contrario X no cumple la ecuación, significa que X cae en un lado del hiperplano:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + +\beta_p X_p < 0$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + +\beta_p X_p > 0$$

Se puede ver al hiperplano como un divisor que divide el espacio *p*-dimensional en dos mitades (Ilustración 2):

¿Cómo se clasifica usando el hiperplano? Si se tiene una matriz X_{nxp} en la que tiene *n* observaciones de entrenamiento en un espacio *p*-dimensional:

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, \; x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

y la variable respuesta tiene dos clases $y_i \in \{-1,1\}$, donde -1 representa una clase y 1 la otra. Nuestro objetivo es construir un hiperplano que sirva de clasificador basándonos en los datos de entrenamiento.

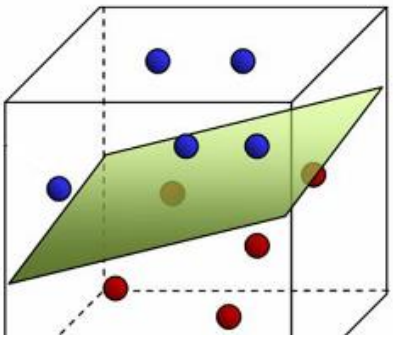

Ilustración 2: Imagen obtenida de https://33bits.wordpress.com/tag/spam/

Si el hiperplano separa las observaciones de los datos de entrenamiento según su clase de forma perfecta, entonces se cumple que:

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + +\beta_p X_p < 0$, si $y_i = -1$

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + +\beta_p X_p > 0$, si $y_i = 1$

En esta situación, el clasificador más sencillo es asignar cada observación a una clase dependiendo del lado del hiperplano en el que se encuentre. Es decir, las observaciones del test $x^* = (x_1^* \cdots x_p^*)^T$, se introducen en la ecuación del hiperplano, y dependiendo del signo resultante será de una clase u otra.

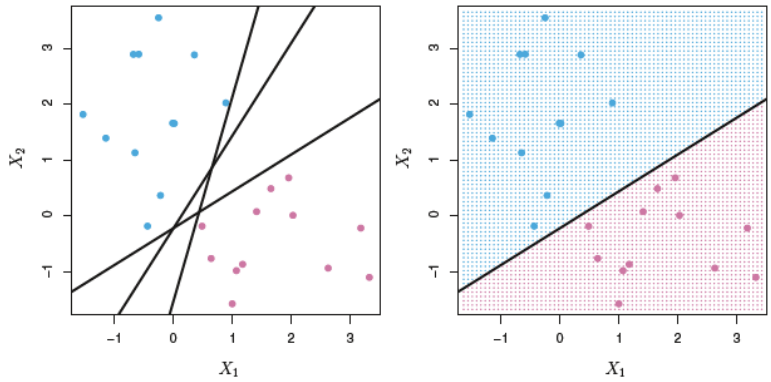

Ilustración 3. Fuente: [10] G. James, D. Witten, T. Hastie, R. Tibshirani (2013)

Cuando se tiene el caso de las clases perfectamente separadas se puede encontrar un número infinito de hiperplanos que separan ambas clases (Ilustración 3), por lo que es necesario aplicar un método para seleccionar cuál de los hiperplanos es el separador óptimo de ambas clases. Aquí es donde entra en juego el "Maximal Margin Classifier".

- Maximal Margin Classifier

El hiperplano óptimo de separación es el que se encuentra más alejado de todas las observaciones de los datos de entrenamiento (Ilustración 4). Para obtenerlo, es necesario calcular todas las distancias de cada observación a los hiperplanos de separación. La distancia más pequeña al hiperplano se conoce como "margen" y es la que determina la posición del plano con respecto al resto de las observaciones, por lo que el hiperplano óptimo es entonces el que obtiene un mayor margen. En la práctica, al haber infinitos planos, no es posible llevarla a cabo, por lo que es necesario recurrir a métodos de optimización.

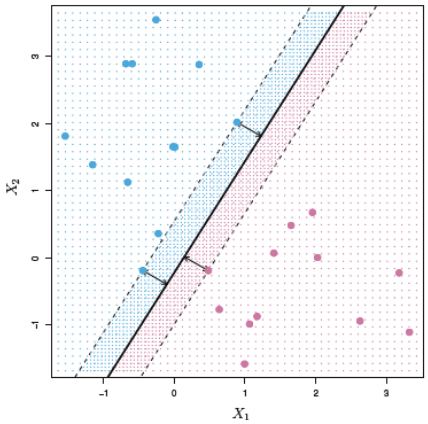

Ilustración 4. Fuente: [10] G. James, D. Witten, T. Hastie, R. Tibshirani (2013)

El Maximal Margin Classifier tiene varios inconvenientes:

- Difícil aplicación práctica: En la mayoría de los casos reales es difícil encontrar una separación de forma perfecta, por lo que no existe un hiperplano de separación para separar las dos clases.
- Poca Robustez: Es muy sensible a variaciones en los datos. Incluir una nueva observación puede producir cambios muy grandes en el hiperplano.

- Overfitting: Un ajuste perfecto del hiperplano sobre las observaciones puede llevar problemas de overfitting.

Para solucionar estos problemas se amplía el concepto de hiperplano de separación para desarrollar un hiperplano que casi separa las dos clases. La generalización del "Maximal Margin Classifier" para casos que no son separables se le conoce como "Support Vector Classifier".

- Support Vector Classifier

Los clasificadores de Support Vector Classifier (también conocidos como Soft Margin Classifiers) son más robustos y tienen una mayor capacidad predictiva al aplicarlo sobre nuevas observaciones. Lo que hace es que en lugar de buscar el mayor margen posible y que cada observación esté en un lugar del hiperplano, se permite que algunas observaciones estén dentro del margen o incluso en el lado incorrecto del hiperplano.

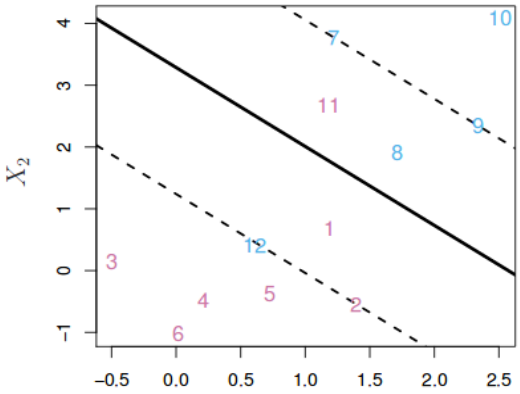

Ilustración 5. Fuente: [10] G. James, D. Witten, T. Hastie, R. Tibshirani (2013)

La ilustración 5 muestra un Support Vector Classifier ajustado para un conjunto de observaciones de entrenamiento. Se puede apreciar que las observaciones 1 y 8 se encuentran en el hiperplano correcto, pero dentro del margen, las observaciones 11 y 12 se encuentran en el lado incorrecto del plano, por lo que mal clasificadas.

Para seleccionar el hiperplano de un Support Vector Classifier es necesario resolver un problema de optimización convexa. No se va a entrar en detalle del problema, pero es importante mencionar que el proceso incluye un parámetro de tunning *C*. *C* es el encargado de controlar el número y la severidad de las observaciones que entran dentro del margen o en el hiperplano equivocado. A menor valor de *C* mayor son las restricciones y menos observaciones pueden entrar en el margen o caer en el otro hiperplano. Para *C*=0 estamos ante un Maximal Margin Classifier. El valor óptimo de este parámetro se identifica mediante cross-validation.

En el proceso de optimización del hiperplano, sólo las observaciones que se encuentran dentro del margen o que están en el lugar incorrecto del hiperplano son las que influyen a la hora de su construcción. A estas observaciones se las conoce como "Vectores Soporte" y son las que definen el hiperplano. Esto implica que el parámetro *C* controle el sesgo y la varianza, ya que es el que controla el número de observaciones que pueden entrar como "Vectores Soporte".

El Support Vector Classifier obtiene muy buenos resultados cuando el límite de separación entre las clases es aproximadamente lineal. Si no es el caso, sus resultados no son muy buenos, para solucionar este problema se puede utilizar los Support Vector Machine.

- Support Vector Machine

Para los grupos que no tienen una separación lineal entre clases, se puede expandir las dimensiones del espacio original, ya que si dos grupos no son separables en el espacio original no implica que en un espacio superior tampoco lo sean. En la ilustración 6 se puede ver un ejemplo.

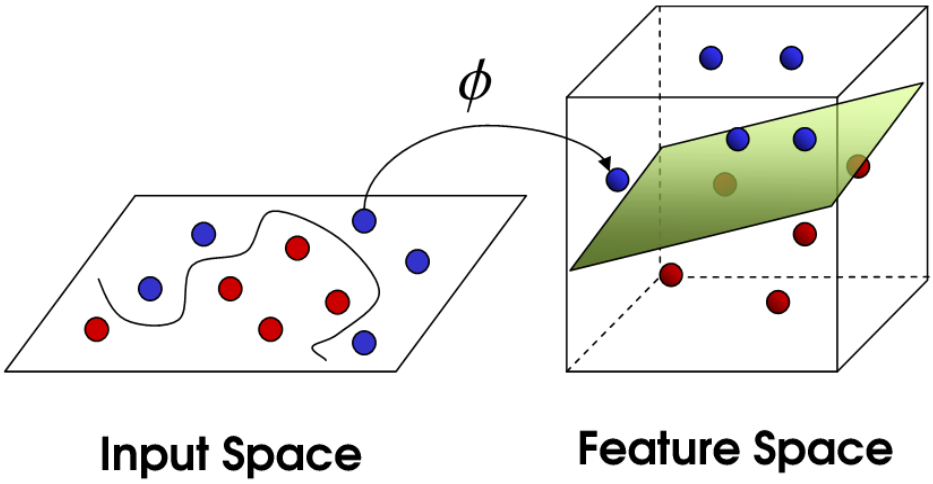

Ilustración 6: Imagen obtenida de: https://www.jeremyjordan.me/support-vector-machines/

Cuando se utilizan variables categóricas se aplica el One-Hot Encoding. Este procedimiento consiste en transformar los datos categóricos en datos numéricos mediante la creación de variables dicotómicas indicando en cada una de ellas si pertenece o no a la clase (1 si pertenece a la clase y 0 si no). Se crean tantas variables dicotómicas como clases tenga la variable categórica a la que se aplica el procedimiento.

Redes Neuronales

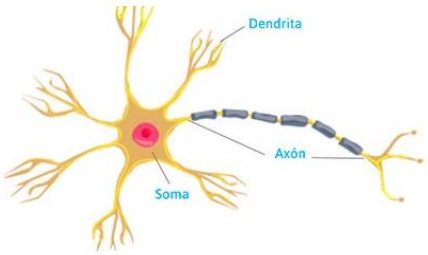

Ilustración 7: Imagen obtenida de la fundación Down21: https://www.down21.org

La neurona biológica (descubierta por Ramón y Cajal en 1888) consta de 3 partes tal como se puede observar en la ilustración 7:

- Soma: Cuerpo de la neurona, órgano de computo.
- Dendritas: Canal de entrada de información.
- Axón: Canal de salida de la información y envío de dicha información a otras neuronas.

La neurona es el elemento básico de un sistema neuronal, el cual, está formado por millones de neuronas organizadas en capas. Para construir un sistema artificial de neuronas, Rumelhart y McClelland (1986) describieron unos principios para construir el modelo estándar de neurona artificial. Los aspectos a tener en cuenta en la modelización con redes neuronales son:

- ELEMENTOS DE UNA RED NEURONAL

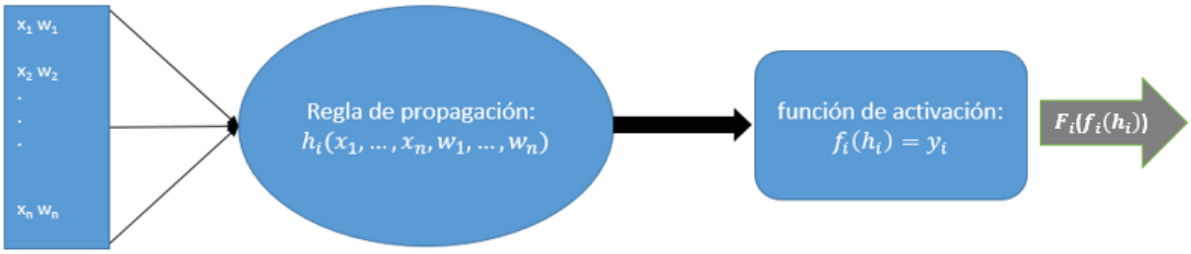

Ilustración 8: Elementos de una red neuronal

- Un conjunto de entradas x_j y unos **pesos sinápticos** w_j, con j = 1, …, n. Los pesos sinápticos representan la fuerza de una conexión sináptica entre dos neuronas. Si los pesos son próximos a cero se considera que no existe comunicación entre el par de neuronas.
- La **regla de propagación** $\boldsymbol{h_i}$ asociada a la neurona "i", determina el potencial post-sináptico resultante de la interacción de la neurona "i" con las neuronas precedentes. La regla usada más habitualmente es la combinación lineal de las entradas y los pesos $\sum_{i=1}^{n} x_j\, w_j$. Se puede añadir un parámetro adicional θ_i, denominado umbral, que se resta a la regla de propagación $\sum_{i=1}^{n} x_j\, w_j - \theta_i$.
- La **función de activación** $\boldsymbol{f_i}$, que representa la salida de la neurona y su estado de activación $y_i = f_i(h_i)$, donde y_i es su estado de activación. Un ejemplo de función de activación es:

$$y_i = \begin{cases} 1 \ \ si \sum_{i=1}^{n} x_j \, w_j \geq 0 \\ 0 \ \ si \sum_{i=1}^{n} x_j \, w_j < 0 \end{cases}$$

En la mayoría de los modelos se suele ignorar el estado de activación anterior de la neurona, pero podría incluirse $y_i(n) = f_i(y_i(n-1), h_i(n))$.

- La **función salida F_i**, proporciona el valor de salida de la neurona. En la práctica no se suele distinguir esta función y se utiliza la propia función de activación (F Identidad).

$$predicción_i(n) = F_i\left(y_j(n)\right) = y_j(n)$$

- <u>TOPOLOGÍA DE UNA RED NEURONAL</u>

La topología es la organización y disposición de las neuronas dentro de la red. Normalmente las neuronas se estructuran en capas. En función de su pertenencia a cada capa, existen varios tipos de neuronas:

- Entrada: Recibe datos o señales procedentes del entorno. A través de ellas se introducen las variables explicativas.
- Salida: Proporciona la respuesta de la red neuronal a los estímulos de la entrada. A través de ellas se introducen las variables target. Si la variable a explicar es de una variable categórica, se utilizan tantas neuronas de salida como clases de la variable -1.
- Ocultas: Es el procesamiento interno de la red, no reciben ni dan información al exterior. Propagan la información desde las neuronas de entrada a las de salida.
- Neurona sesgada: Se puede añadir para que actúe como termino constante.

Las conexiones entre neuronas pueden ser de dos tipos, si se produce entre neuronas de una misma capa son conexiones laterales o intra-capa, mientras que si la conexión se produce entre neuronas de distintas capas se denominan conexión inter-capa. Si se mira el sentido de las conexiones, si todas las conexiones entre neuronas siguen un mismo sentido, la red se denomina unidireccional o feedforward. Cuando las conexiones no siguen un solo sentido se denominan feedback o recurrente En la ilustración 9 se pueden ver un par de ejemplos.

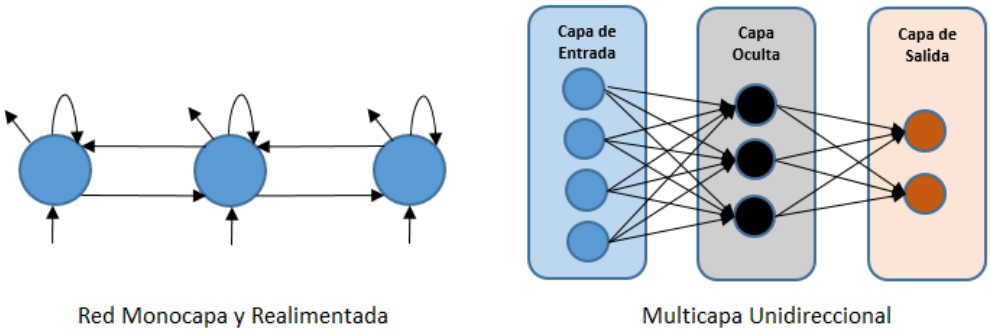

Ilustración 9: Diferentes tipos de redes

- APRENDIZAJE DE UNA RED NEURONAL

Una vez determinada la topología de la red es necesario entrenarla. El proceso de aprendizaje de una red neuronal es un proceso iterativo en el que, secuencialmente, se va analizando cada uno de los patrones de entrada a la red, para encontrar la relación de pesos (w_j) correcta para afinar las predicciones.

En el proceso de aprendizaje se establece una **función de error** $E(w)$ que mide el error generado por la fórmula definida para la relación de pesos sinápticos ajustados hasta el instante n. El objetivo del proceso de entrenamiento es encontrar la relación de pesos que minimizan dicha función de error. Para ello se pueden marcar etapas en el aprendizaje de la red neuronal:

1. **Inicialización de la red**: Se comienza en un instante inicial $n = 0$ partiendo de un conjunto de pesos sinápticos aleatorios (w_j) que generan un error inicial de predicción.
2. **Análisis de todo el conjunto de datos**: En cada instante de tiempo n, se analiza un nuevo conjunto de datos entrada $x(n)$ y se va realizando un ajuste de los pesos buscando mejorar el error actual del modelo.
3. **Asignación del error**: Cuando se ha leído el conjunto de datos entero, se ha completado un ciclo, y se vuelve a empezar de nuevo. El algoritmo se detiene cuando, al final de un ciclo el error se sitúa por debajo de una cota prestablecida, cuando de un ciclo a otro, el error no decrece sensiblemente o se ha alcanzado un número máximo de iteraciones prefijado.

El algoritmo más utilizado para el aprendizaje es el ***Backpropagation*** en el que se distinguen dos etapas:

- Cálculo de la salida que se obtiene por la red neuronal dado un conjunto de datos de entrada $x(n)$.

- Cálculo del error (entre la salida y el target) y propagación de dicho error hacia atrás desde la capa de salida, donde cada neurona precedente recibe un error proporcional a su contribución sobre el error total de la red.

El método está basado en el **método del gradiente descendiente** (Ilustración 10). El método parte de un conjunto de pesos iniciales $W(0)$, y calcula la dirección de máximo decrecimiento del error Dicha dirección viene dada por el vector gradiente $\nabla E(w)$. Los pesos se actualizan siguiendo el sentido contrario al gradiente, que indica la dirección de máximo decrecimiento:

$$W(n+1) = W(n) + \nabla W(n) = W(n) + \mu \cdot \nabla E(w(n))$$

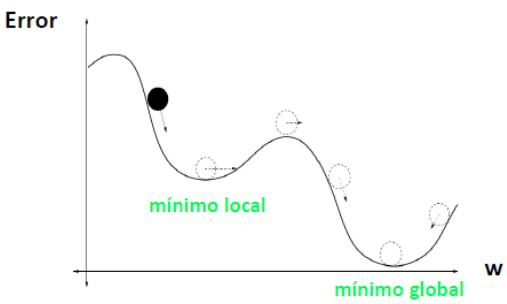

Ilustración 10. Fuente: [14] Vélez Serrano, Daniel (Curso 2017/2018)

Donde μ es la tasa de aprendizaje, e indica el paso que se da en cada iteración. Hay que tener cuidado, porque si μ es pequeño podría terminar convergiendo en mínimos locales, en cambio, si μ es grande podría quedar dando oscilaciones con respecto al mínimo y nunca convergería.

Podríamos decir, que el método del gradiente descendente consta de 3 pasos:

- Determinar una longitud para la tasa de aprendizaje μ.

- Buscar la dirección de máximo decrecimiento de la función error $\nabla E(w(n))$.

- Moverse en la dirección correspondiente y volver al paso anterior hasta que el vector gradiente se haga prácticamente 0.

IMPLEMENTACIÓN EN R Y ANÁLISIS DE RESULTADOS

Como paso inicial, y aplica a todos los métodos utilizados, se ha procedido a dividir el fichero inicial de datos (18.486 individuos que han contestado a la Encuesta Nacional de Salud del año 2011) en dos ficheros de datos:

- **Datos de Entrenamiento**: Corresponde al 80% de los datos totales. Son los datos que se utilizan para entrenar el modelo.
- **Datos de Test**: Son el 20% restante del fichero. Son los datos utilizados para testear cómo de bueno es el modelo, y obtener unas predicciones del mismo.

Para realizar la división del fichero, se ha realizado una división de los datos de manera proporcional al campo que se desea estudiar, en este caso la variable de estudio son las variables "Target" que son variables binarias que indican si el individuo ha contestado mal la variable CNO (1) y ha tenido que ser editada en el proceso de E&I o en caso contrario (0).

La proporción de unos y ceros en los datos de entrenamiento y los datos de test son iguales, pero la proporción de ceros es mucho mayor que la de unos. Este hecho es un problema para poder clasificar los individuos de la clase minoritaria, en este caso, son los que tienen mal asignado el CNO, ya que corresponden sólo a un 5.8% del total. Con esto se podría llegar fácilmente a una precisión (proporción de individuos clasificados correctamente) del 94.18%, con sólo indicar que todos los individuos pertenecen a la clase "0", obtendríamos una precisión muy alta para el modelo. Realmente esta precisión no es de mucho interés, ya que lo que interesa es saber cuántos individuos de la clase "1" están bien clasificados. Para esto, es necesario ver la matriz de confusión.

La **matriz de confusión** (Ilustración 11) es una tabla que contiene el número de individuos que:

		VALOR PREDICHO	
		Clase: 0	Clase: 1
VALOR REAL	Clase: 0	NV	PF
	Clase: 1	NF	PV

Ilustración 11: Matriz de confusión

- NV (Negativos Verdaderos): Número de individuos han sido clasificados como negativo, y su valor real es negativo.
- NF (Negativos Falsos): Número de individuos han sido clasificados como negativo, y su valor real es positivo.
- PF (Positivos Falsos): Número de individuos han sido clasificados como positivo, y su valor real es negativo.
- PV (Positivos Verdaderos): Número de individuos han sido clasificados como negativo, y su valor real es negativo.

De esta tabla, se puede obtener la precisión de positivos, que es la **precisión** de positivos que se han clasificado correctamente ($\frac{PV}{PV+PF}$), y la ***recall*** o tasa de positivos verdaderos, que son los individuos predichos correctamente ($\frac{PV}{PV+NF}$). Para saber cómo se comporta un modelo, es necesario ver cómo se comportan juntas. Para ver cómo se relacionan, tenemos las curvas ROC (Receiver Operating Characteristic).

Las curvas ROC (Ilustración 12) son curvas en las que se presenta la sensibilidad (tasa de positivos verdaderos $\frac{PV}{PV+NF}$) en función de la especificidad (tasa de negativos verdaderos $\frac{NV}{NV+PF}$ o 1 menos la fracción de falsos positivos $\frac{PF}{PF+NV}$). Cuanto más se acerque esta curva a la esquina superior izquierda, mejor será el rendimiento del modelo (es decir, se maximiza la tasa de positivos verdaderos a la vez que se minimiza la de falsos positivos).

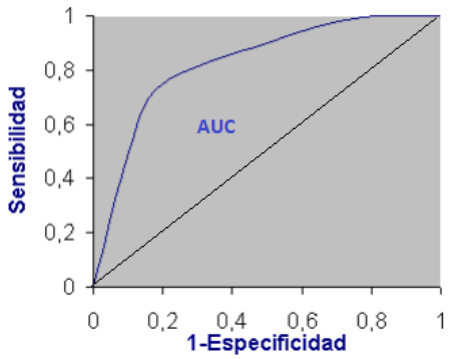

Ilustración 12: Curva ROC

La curva ROC nos permite visualizar el intercambio entre el beneficio (tasa de positivos verdaderos) y el coste que supone aumentar el beneficio (tasa de falsos positivos), cualquier modelo que incremente el número de positivos verdaderos, aumentará también los falsos positivos. Contra mayor sea el AUC (área debajo de la curva) mejor clasificará el modelo. El AUC se puede calcular de la siguiente manera:

$$AUC = \frac{1 + Tasa\ de\ positivos\ verdaderos - Tasa\ de\ falsos\ positivos}{2}$$

Para solucionar el problema de datos desbalanceados se recurre a realizar un muestreo con el fin de obtener unos datos de entrenamiento en que la proporción de ambas clases sean similares. Este problema se puede abordar con la función SMOTE[4] (Synthetic Minority Oversampling TEchnique) de R. Esta función aborda el problema a través de realizar un sub-muestreo sobre la clase dominante y un sobre-muestreo de la clase minoritaria. Para realizar el sobre-muestreo genera individuos ficticios de la clase minoritaria según los datos reales, es decir, no selecciona para la muestra un individuo de la clase minoritaria n veces, sino que genera un registro nuevo a través de la información de los datos reales teniendo en cuenta la cercanía de los individuos. Para el sub-muestreo si se realiza un sub-muestro de manera rigurosa.

[4] L. Torgo (2010): "Data Mining with R, learning with case studies" url = http://www.dcc.fc.up.pt/~ltorgo/DataMiningWithR

La función SMOTE tiene 3 parámetros de entrada:

- K: Cuando genera los individuos ficticios, se fija en los K-individuos más cercanos para generar los datos de la clase minoritaria.
- Sobre-muestreo: Indica para cada individuo de la clase minoritaria, cuantos individuos ficticios se van a generar para la clase minoritaria.
- Sub-muestreo: Indica cuantos casos de la clase mayoritaria son seleccionados en la muestra, para cada caso generado para la clase minoritaria.

Para ver que parámetros se deben de utilizar en la función SMOTE, se ha realizado una prueba tomando diferentes valores para los tres parámetros (Ilustración 13), con esta prueba se han generado 18 diferentes ficheros de entrenamiento. Para ver cuál de los ficheros generados es el que mejor resultado obtiene a la hora de entrenar los modelos, se va a proceder a ver los resultados de cada uno de ellos en la primera técnica a utilizar (Random Forest) y el que mejor ajuste el modelo se aplicará al resto de las técnicas propuestas.

```
# Hiperparámetros
hiperparametros_smote <- expand.grid(perc.over = c(50, 150, 300),
                                     k = c(3, 6),
                                     perc.under = c(25, 200, 400))
```

Ilustración 13: Parámetros utilizados en la función SMOTE

Una vez tratado los datos de entrada, se seleccionan las siguientes variables explicativas para entrenar los modelos. Tenemos las siguientes variables:

CAMPO	DESCRIPCIÓN
EDADa	Adulto seleccionado: Edad
RANGOEDAD	Adulto seleccionado: Agrupación de la variable edad (Menor de 30, 30-39, 40-49, 50-59 y Mayor 60)
PROXY_0	¿El informante es la persona seleccionada?
A7_2a	Número de orden del adulto seleccionado
SEXOa	Adulto seleccionado: Sexo
CNAE_AS	Adulto seleccionado: Actividad económica (división CNAE2009)
CNAE1_AS	Adulto seleccionado: Actividad económica (sección CNAE2009)
CNAE2_AS	Adulto seleccionado: Actividad económica (grupo CNAE2009)
CCAA	Adulto seleccionado: Comunidad Autónoma de residencia
CNO_AS_1	Adulto seleccionado: Ocupación (Gran grupo CNO-11)
CNO_AS_2	Adulto seleccionado: Ocupación (Subgrupo principal CNO-11)
CNO_AS_3	Adulto seleccionado: Ocupación (Subgrupo CNO-11)
F18	Adulto seleccionado: Situación profesional de su último empleo
FACTORADULTO	Factor de elevación. (Peso de muestreo)
ESTRATO	Estrato al que pertenece el hogar (por tamaño de municipio)

Todas las variables son cualitativas, excepto "EDADa" y "FACTORADULTO" que son cuantitativas. Para entrenar los modelos se han creado 4 grupos de variables explicativas para ver cuál de ellos entrena mejor los modelos. Los grupos son:

- GRUPO1: Eliminando la variable RANGOEDAD y FACTORADULTO.
- GRUPO2: Eliminando la variable EDADa y FACTORADULTO.
- GRUPO3: Eliminando la variable RANGOEDAD.
- GRUPO4: Eliminando la variable EDADa.

Igual que para los diferentes ficheros de train que hemos generado, se va a proceder a probar los 4 grupos de variables explicativas en la primera técnica utilizada (Random Forest) y viendo cuál de los 4 grupos es el que mejor se ajusta, procederemos con dicho grupo al resto de las técnicas.

En los siguientes apartados se entrenan las diferentes técnicas de Machine Learning con el objetivo de compararlas e identificar el que mejor resultado obtiene clasificando los individuos con el código CNO erróneo. Se va a utilizar la primera técnica propuesta (Random Forest) para probar las diferentes opciones de ficheros de entrada (18 distintos ficheros generados con la función SMOTE y el fichero original) y de variables explicativas (4 grupos variables mencionados anteriormente) para quedarnos con las opciones que mejor se ajustan y aplicarlas a las siguientes técnicas propuestas, SVM y Redes Neuronales.

Los modelos se ajustan, optimizan y se comparan utilizando la librería caret[5].

Random Forest

Para utilizar la técnica de Random Forest, se tienen en cuenta 3 parámetros:

- *mtry*: número de variables explicativas que se utilizan en cada una de las ramas del árbol.
- *Min.node.size*: Tamaño mínimo que tiene que tener un nodo para poder ser dividido.
- *Regla de división*: La regla que se utiliza para realizar la división.

Para este modelo se van a usar los siguientes valores para los parámetros:

```
# Hiperparámetros
hiperparametros <- expand.grid(mtry = c(2, 5, 10, 12),
                               min.node.size = c(2, 3, 4, 5, 10),
                               splitrule = "gini")
```

Ilustración 14: Parámetros usados para Random Forest

[5] Max Kuhn. Contributions from Jed Wing and Steve Weston and Andre Williams and Chris Keefer and Allan Engelhardt and Tony Cooper and Zachary Mayer and Brenton Kenkel and the R Core Team and Michael Ben esty and Reynald Lescarbeau and Andrew Ziem and Luca Scrucca and Yuan Tang and Can Candan and Tyler Hunt (2019): "Classification and Regression Training" .R package version 6.0-84. https://CRAN.R-project.org/package=caret

Se han tomado diferentes valores para el mtry y para el min.node.size (Ilustración 14), donde se verá la evolución de la precisión según los parámetros utilizados. La regla de división utilizada es el índice de Gini. Al utilizar el índice de Gini o Cross-Entropy, se obtienen resultados muy similares, por lo que se ha decidido utilizar el índice de Gini para evitar introducir funciones logarítmicas en el cálculo con el fin de hacerlo computacionalmente más rápido.

Para entrenar este modelo, se van a utilizar los 18 ficheros de entrada que han sido generados con la función SMOTE, y también se va a utilizar el fichero de entrenamiento original, con el fin de comparar los resultados entre los diferentes ficheros de entrenamiento y ver cuál es el que mejor resultado obtiene y aplicarlo a las siguientes técnicas propuestas.

- Resultados con el fichero Original:

Se ha utilizado el fichero de entrenamiento original para entrenar el modelo, en la ilustración 15 se puede observar cómo ha evolucionado la precisión del modelo según los diferentes valores de los parámetros de entrada.

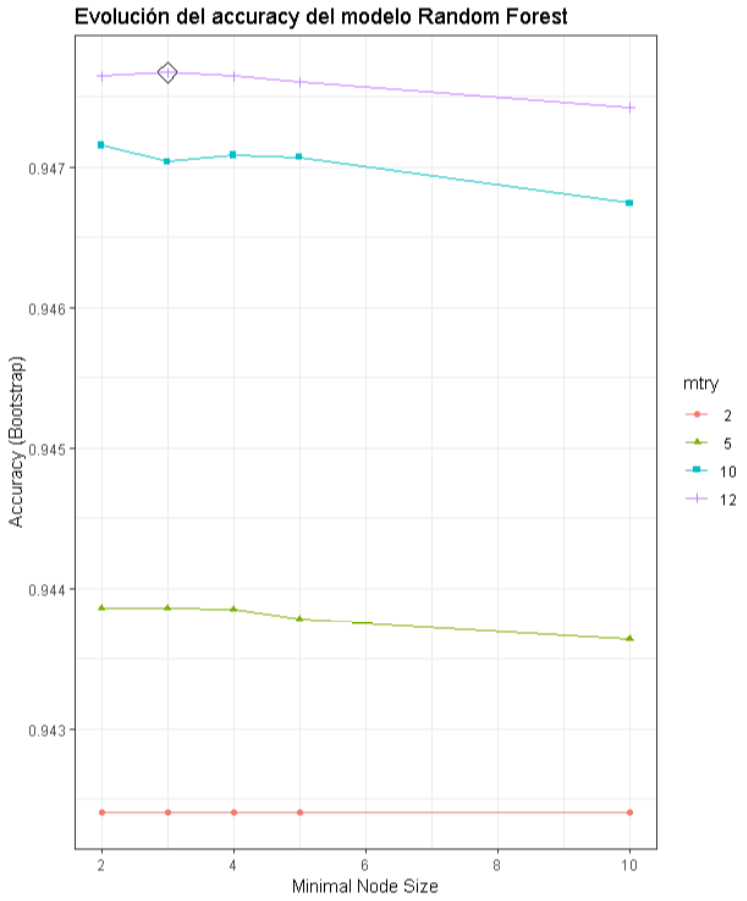

Ilustración 15: Evolución de la precisión del modelo según los diferentes valores de los parámetros de entrada.

La precisión máxima viene tomando el mayor número de variables explicativas en cada uno de los nodos (en este caso 12), y el tamaño mínimo que tiene que tener un nodo para ser dividido tiene que ser 3.

Tomando estos valores, se han entrenado 4 modelos diferentes de Random Forest, uno para cada grupo de variables explicativas de entrada. Para cada uno de estos 4 modelos se ha introducido el fichero test para ver las predicciones de cada uno de ellos. Las curvas ROC de los resultados de cada uno de los modelos las podemos ver en la ilustración 16:

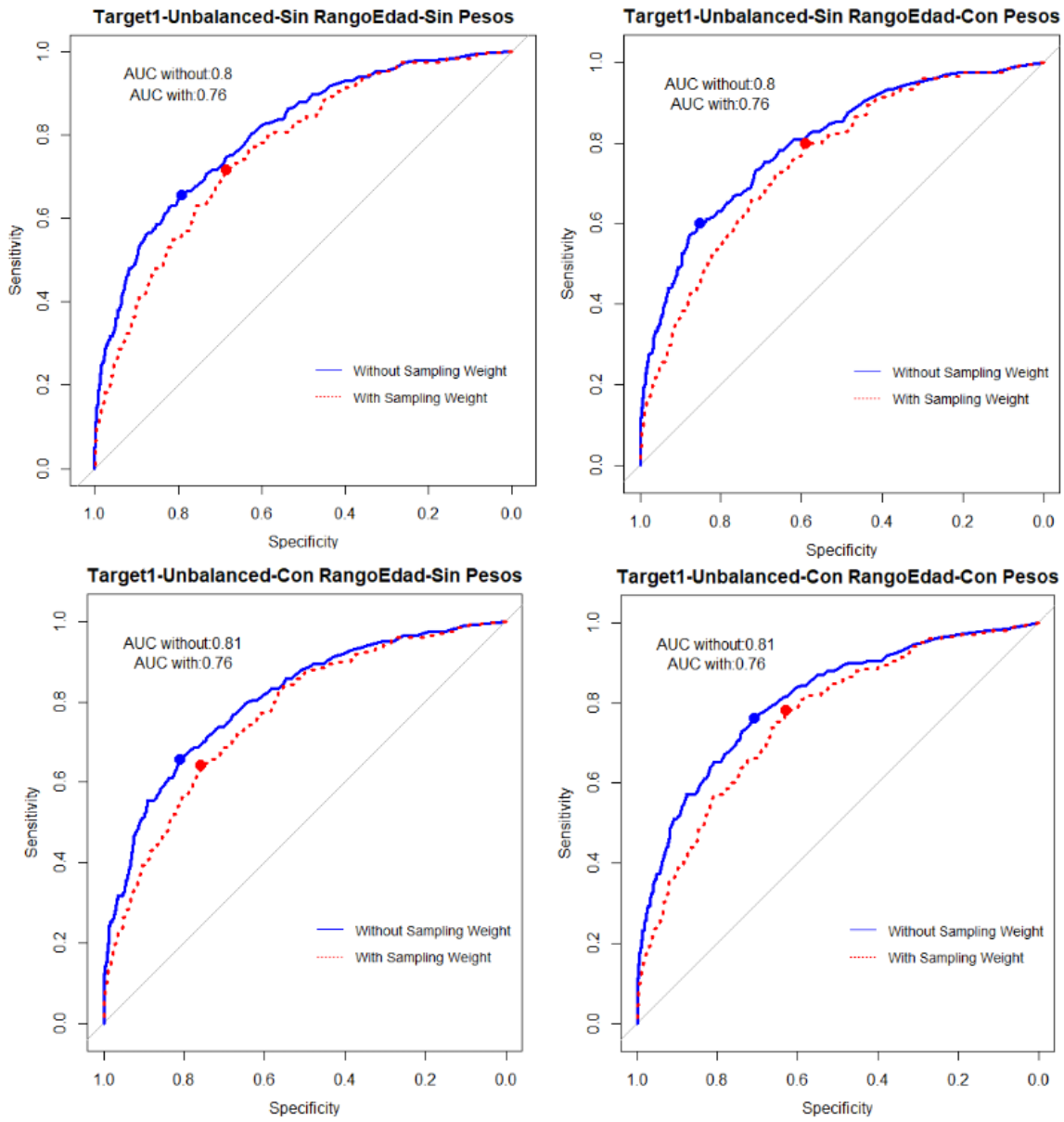

Ilustración 16: Curvas ROC para los 4 grupos de variables explicativas.

De izquierda a derecha y de arriba abajo son grupo 1, 3, 2 y 4)

Cómo se puede observar en la ilustración 16, se han dibujado dos curvas en cada uno de los modelos, la curva “azul” se corresponde a los Local Score obtenidos sin tener en cuenta los pesos de muestreo mientras que la curva “roja” refleja el resultado introduciendo los pesos de muestreo. Como primera conclusión se puede observar que al introducir los pesos de muestreo en los Local Score el AUC disminuye, con lo que la precisión del modelo disminuye. El otro resultado que se observa es relativo a la precisión en cada uno de los grupos. Por un lado, vemos que los grupos más precisos son el grupo 2 y 4 que utilizan la variable “RANGOEDAD” en lugar de utilizar la variable numérica “EDADa”, y por otro que introducir los pesos de muestreo como regresor no afecta mucho a la precisión del modelo.

- Resultados con los diferentes ficheros generados por la función SMOTE:

A continuación, se pasa a entrenar un Random Forest para cada fichero generado por la función SMOTE, por lo que se tienen un total de 18 diferentes modelos, cada uno de ellos entrenado con un fichero de entrenamiento diferente. Al igual que para el modelo del fichero original, cada fichero de entrenamiento generado por la función SMOTE. Se ha entrenado para cada grupo de variables explicativas (4 grupos de variables explicativas), por lo que hemos obtenido un total de 72 modelos diferentes.

En las siguientes Ilustraciones (Ilustración 17, 18, 19 y 20) se muestran las diferentes curvas ROC obtenidas para cada uno de los modelos entrenados por los ficheros generados por la función SMOTE para cada grupo de variables explicativas.

Realizando un análisis general de los resultados obtenidos:

- Se observa que los modelos con mejor AUC son los que toman los valores K=6, over=50 y under=400 en todos los grupos, ya que obtienen los AUC más altos.
- Como pasa también en los resultados de los modelos entrenados con el fichero original, la curva ROC que tiene en cuenta los pesos de muestreo (roja) es siempre inferior a la del modelo que no los tiene en cuenta.
- En cada uno de los 4 grupos el AUC es muy similar ∼0,8.

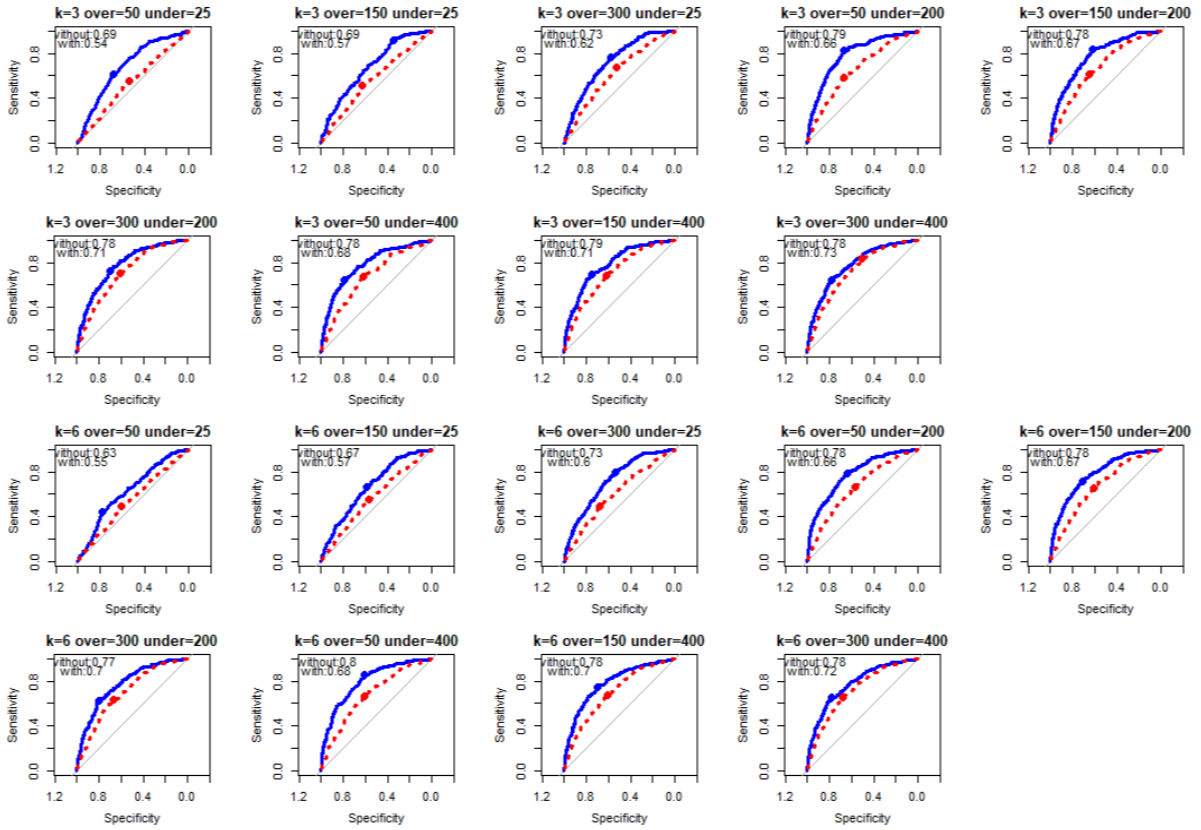

Ilustración 17: Curvas ROC para el grupo1 de variables explicativas

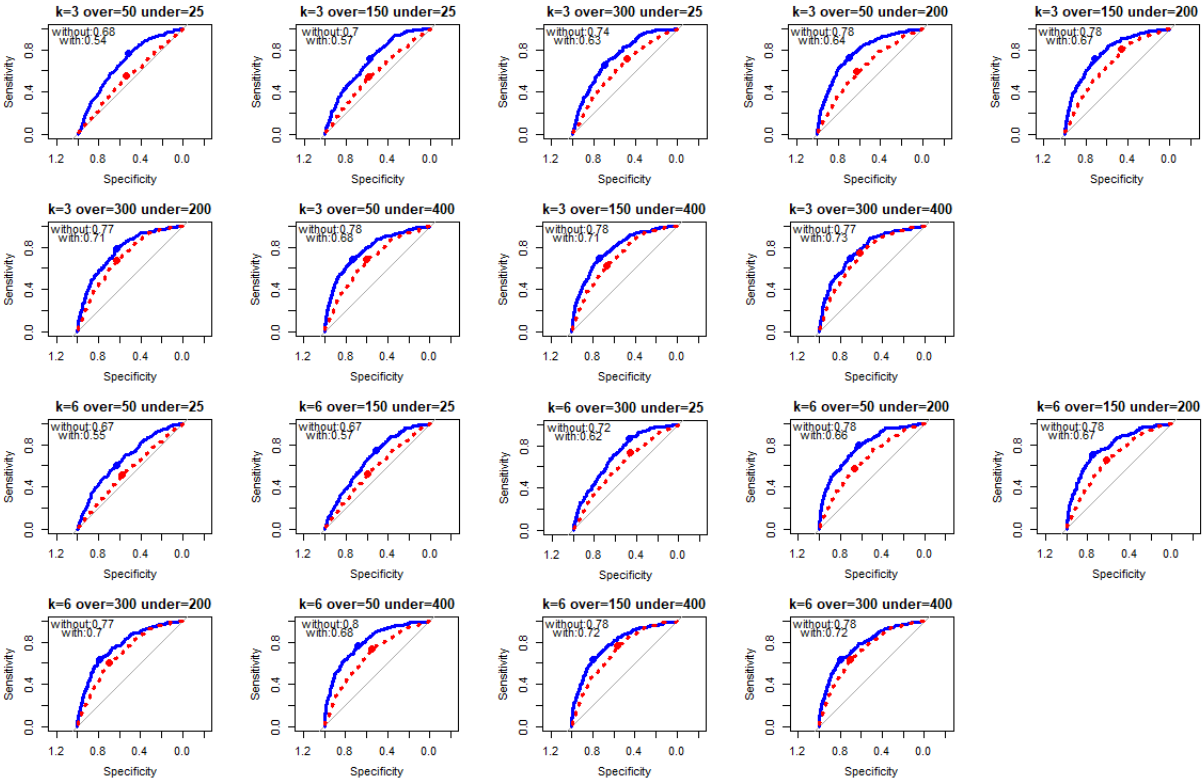

Ilustración 18: Curvas ROC para el grupo2 de variables explicativas

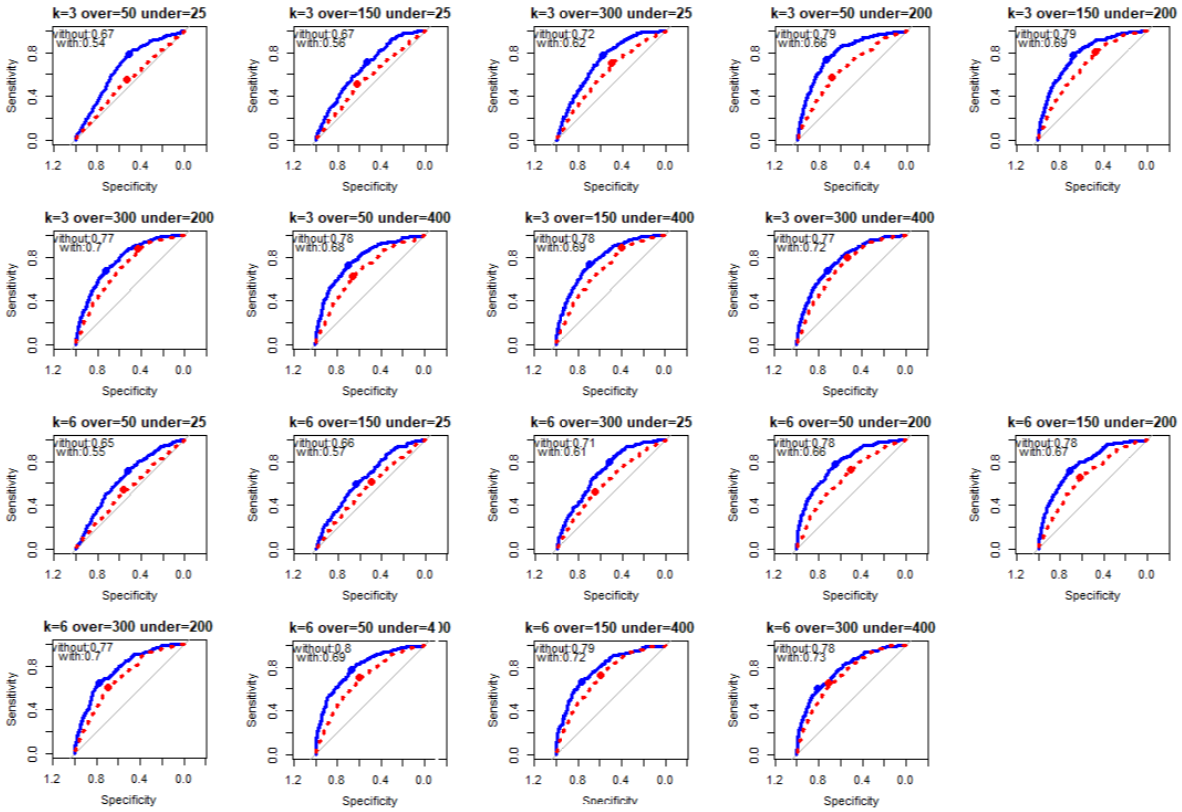

Ilustración 19: Curvas ROC para el grupo3 de variables explicativas

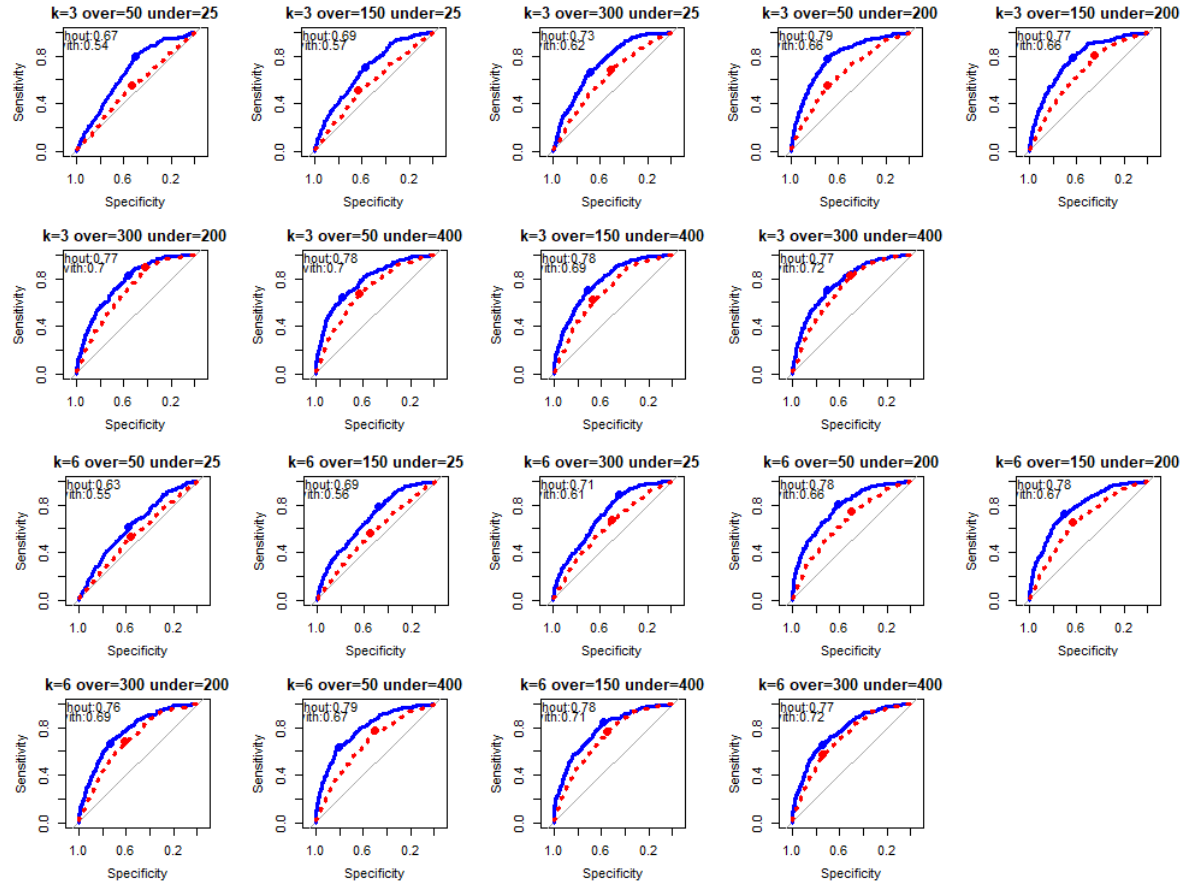

Ilustración 20: Curvas ROC para el grupo4 de variables explicativas

Para ver los resultados que se han obtenido de los mejores modelos (K=6, over=50 y under=400) de cada grupo se puede observar las matrices de confusión (Ilustración 21). (El resto de las matrices de confusión se pueden encontrar en la parte de anexos).

GRUPO 1						
				VALOR PREDICHO		SENSIBILIDAD
				Clase: 0	Clase: 1	0,609302326
16	K=6 over=50 under=400	VALOR REAL	Clase: 0	2783	699	ESPECIFICIDAD
			Clase: 1	84	131	0,799253303

GRUPO 2						
				VALOR PREDICHO		SENSIBILIDAD
				Clase: 0	Clase: 1	0,63255814
16	K=6 over=50 under=400	VALOR REAL	Clase: 0	2773	709	ESPECIFICIDAD
			Clase: 1	79	136	0,79638139

GRUPO 3						
				VALOR PREDICHO		SENSIBILIDAD
				Clase: 0	Clase: 1	0,623255814
16	K=6 over=50 under=400	VALOR REAL	Clase: 0	2761	721	ESPECIFICIDAD
			Clase: 1	81	134	0,792935095

GRUPO 4						
				VALOR PREDICHO		SENSIBILIDAD
				Clase: 0	Clase: 1	0,637209302
16	K=6 over=50 under=400	VALOR REAL	Clase: 0	2816	666	ESPECIFICIDAD
			Clase: 1	78	137	0,808730615

Ilustración 21: Matrices de confusión mejores modelos de cada grupo

Los grupos que mejor resultado obtienen en función de la sensibilidad y la especificidad son el grupo 2 y el grupo 4, igual que en el modelo entrenado con el fichero original.

Viendo los resultados obtenidos en los diferentes modelos de Random Forest se toman las siguientes decisiones para los modelos SVM y Neural Network:

- Se van a entrenar con el fichero original y con el fichero generado con la función SMOTE usando los parámetros K=6, over=50 y under=400, que es con el que se han obtenido los mejores resultados en las pruebas.
- Se ha visto que al utilizar intervalos de edad se obtienen mejores resultados que utilizando el campo numérico de edad y que incluir o no los pesos de muestreo en los regresores no altera los Local Score, por lo que se va a utilizar el grupo 2 como variables explicativas (utilizando intervalos de edad en lugar del campo numérico de edad, y excluyendo el peso de muestreo de los regresores).

Support Vector Machine (SVM)

Es común que las separaciones lineales no sean suficientes, ya que las iteraciones entre las variables no siguen una forma lineal. Como se ha visto en la metodología, para solucionar este problema, es necesario un aumento de la dimensión para encontrar una separación. Para aumentar la dimensión hay diferentes métodos, el más sencillo es añadir términos al cuadrado o al cubo en el polinomio:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_2^2 + \beta_5 x_2^3 + \cdots$$

Cuando se usan polinomios con un grado mayor para ajustar el modelo, al tener muchas variables explicativas, se suele obtener un sobre-ajuste y se suele perder precisión en las estimaciones. Por ello, otra manera de aumentar la dimensión es añadiendo otras funciones no lineales en el SVM, aquí entra en juego la función del Kernel Radial:

$$K(x, y) = e^{-\gamma \sum_{j=1}^{p} (x_{ij} - y_{ij})^2}$$

Donde γ es un parámetro que suaviza los límites para tomar decisiones y reduce la varianza del modelo. Si γ es muy grande se tiene problemas de sobre-ajuste y una alta varianza, mientras que si γ es pequeño los límites para tomar decisiones son más suaves y tiene menos varianza.

Dentro de la librería *caret*, tenemos el método *svmRadial* que emplea la función *ksvm()* del paquete *kernlab* de R. Este método tiene dos parámetros de entrada:

- γ: Parámetro para el Kernel radial.
- *C*: Encargado de controlar el número y la severidad de las observaciones que entran dentro del margen o en el hiperplano equivocado.

Para el modelo se prueban diferentes valores para ambos parámetros (Ilustración 22), para ver cuál es el que mejor se ajusta al modelo.

```
# Hiperparámetros
hiperparametros <- expand.grid(sigma = c(0.0001, 0.001, 0.01),
                               C = c(1, 10, 50, 100, 250, 500, 700, 1000))
```

Ilustración 22: Parámetros usados para SVM Kernel Radial

En los siguientes gráficos (Ilustración 23), se puede ver cómo afecta a la precisión los diferentes valores de los parámetros que se han asignado a la función del Kernel Radial. Para el modelo entrenado con el fichero original se tiene un valor del parámetro γ igual a 0.0001 y un valor de C de 1000 y para el modelo entrenado con el fichero generado con la función SMOTE se tiene un valor del parámetro γ igual a 0.01 y un valor de C de 50.

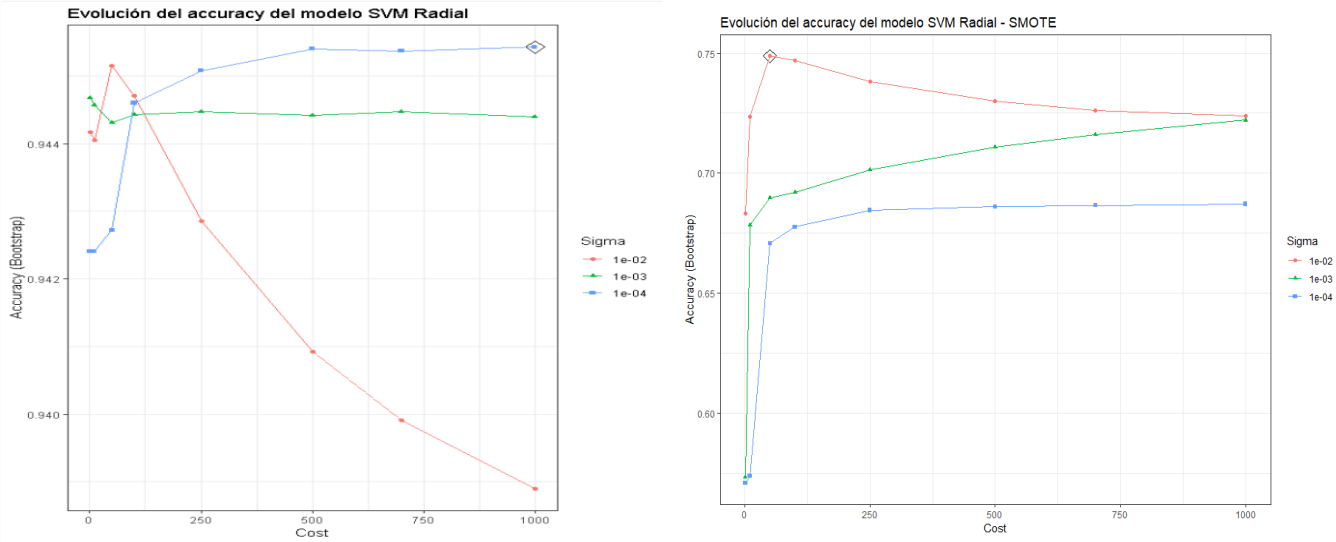

Ilustración 23: Evolución de la precisión según los diferentes parámetros de la función Kernel Radial.

La precisión obtenida en cada caso es bastante diferente, para los datos de entrenamiento originales se obtiene un 94.56% y para los datos generados por SMOTE se obtiene un 74.87%. Hay que tener en cuenta que lo que se está midiendo es la precisión total, es decir, los individuos clasificados correctamente entre el total de individuos.

Una vez se tienen los modelos entrenados, se introducen en el modelo los datos de test que se separaron al principio para poder ver lo bien o mal que predicen los modelos. Con los resultados de las predicciones se generan las matrices de confusión de cada modelo, y se puede observar la sensibilidad y especificidad de cada modelo:

SVM (KERNEL RADIAL)					
MODELO SMOTE			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,469767442
	VALOR REAL	Clase: 0	2942	540	ESPECIFICIDAD
		Clase: 1	114	101	0,844916715
MODELO NORMAL			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,069767442
	VALOR REAL	Clase: 0	3476	6	ESPECIFICIDAD
		Clase: 1	200	15	0,998276852

Ilustración 24: Matrices de confusión.

Como se observa en la (Ilustración 24), el modelo entrenado con los datos originales tiene una especificidad (Tasa de Negativos Verdaderos) de un 99.97%. Esto es debido a la alta proporción de la clase “0” que existe en los datos originales lo que lleva al modelo a tomar casi siempre la decisión de clasificar a los individuos en la clase 0. En cambio, la sensibilidad (Tasa de Positivos Verdaderos) es del 7.44%. Para el modelo entrenado con los datos obtenidos de la

función Smote se observa que la matriz de confusión está mucho más distribuida, es decir, el modelo toma mucho más a menudo la decisión de clasificar a los individuos en la clase 1, lo que conlleva a una peor especificidad 85.44% pero mejora considerablemente la sensibilidad, y crece hasta un 45.58%.

Las Curvas ROC de ambos modelos, el entrenado por el fichero original (Ilustración 25) y el entrenado por el fichero generado con la función SMOTE (Ilustración 26) son las siguientes:

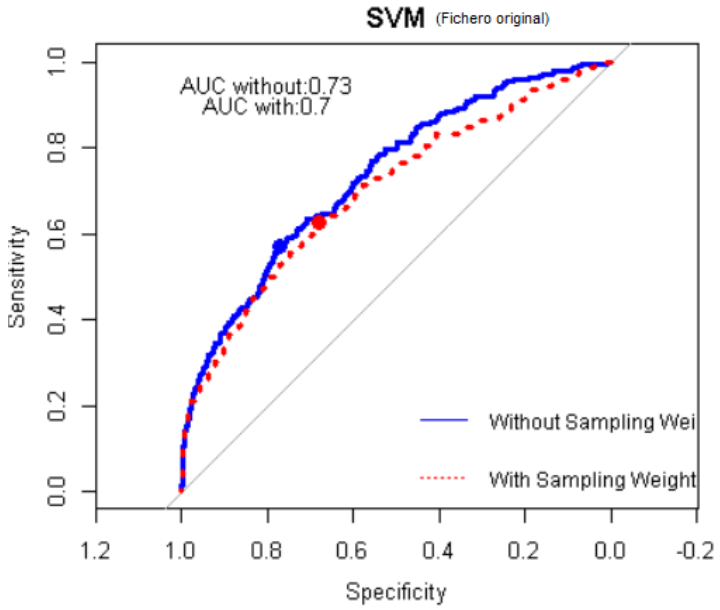

Ilustración 25: Curva ROC del SVM Radial para el fichero original

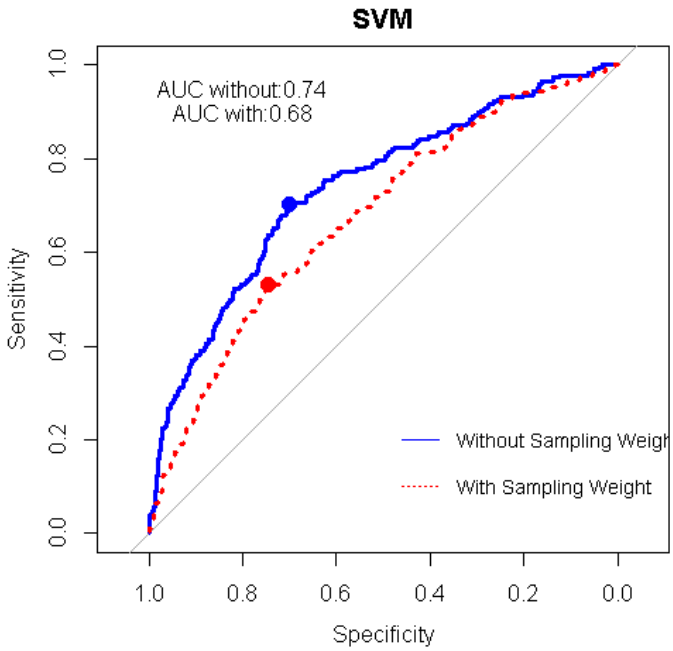

Ilustración 26: Curva ROC del SVM Radial para el fichero SMOTE

Ambas curvas son muy parecidas, tienen un valor del AUC cuando sólo tenemos en cuenta la probabilidad de error de 0.74 y 0.73 respectivamente. Para el AUC considerando los pesos de muestreo, se tienen uno valores de 0.7 y 0.68 respectivamente.

Redes Neuronales

Para utilizar la técnica de Neural Network, se tienen en cuenta 2 parámetros:

- *size*: indica el número de neuronas en la capa oculta.
- *decay*: controla la regularización durante el entrenamiento de la red para evitar el sobre-ajuste del modelo.

Para el modelo se prueban diferentes valores para ambos parámetros (Ilustración 27), para ver cuál es el que mejor se ajusta al modelo

```
# Hiperparámetros
hiperparametros <- expand.grid(size = c(5, 10, 15, 20, 40),
                               decay = c(0.01, 0.1))
```

Ilustración 27: Parámetros usados para NNET

En la ilustración 28 se puede ver cómo afecta a la precisión los diferentes valores de los parámetros utilizados para entrenar los modelos de redes neuronales. Para el modelo entrenado con el fichero original se tiene que el mejor modelo es el que utiliza 5 neuronas en la capa oculta, y el parámetro “Weight Decay” un valor de 0.1. Para el modelo entrenado con el fichero generado con la función SMOTE se tiene que el que mejor precisión presenta es el que utiliza 40 neuronas en la capa oculta y un valor del parámetro “Weight Decay” de 0.1.

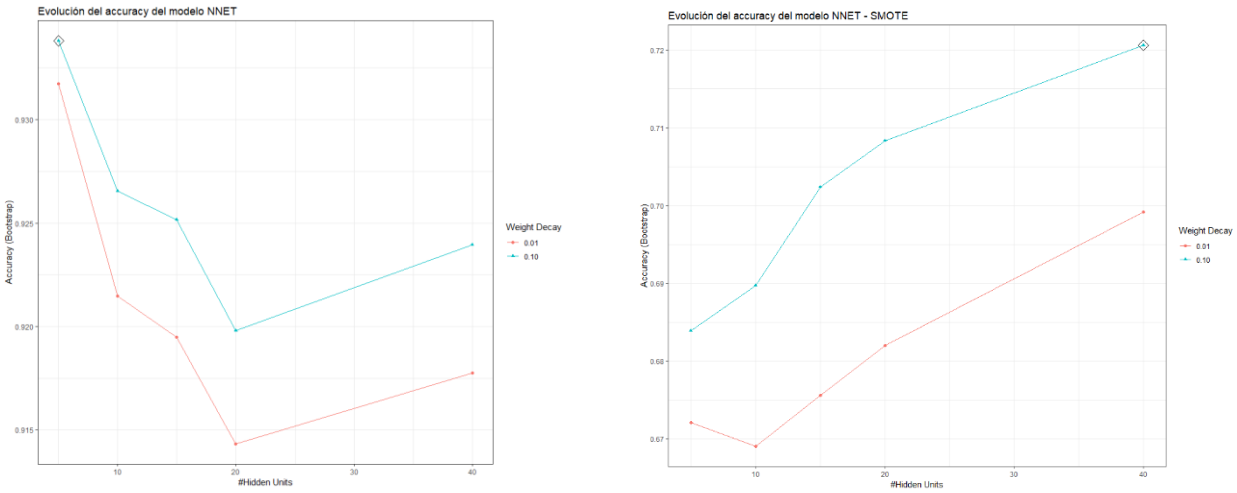

Ilustración 28: Evolución de la precisión según los diferentes parámetros del modelo Neural Network

La precisión obtenida en cada caso es bastante diferente, para los datos de entrenamiento originales se obtiene un 93.38% mientras que para los datos generados por SMOTE se obtiene un 72.06%. Hay que tener en cuenta que lo que se está midiendo es la precisión total, es decir, los individuos clasificados correctamente entre el total de individuos. Igual que se ha visto anteriormente, ésta es una medida poco fiable, ya que no sólo se desea obtener una precisión alta, sino lo que se busca es encontrar a las unidades con error, es decir, aumentar la tasa de positivos verdaderos.

Una vez se tienen los modelos entrenados, se introducen en el modelo los datos de test que se separaron al principio para poder ver lo bien o mal que predicen los modelos. Para poder observar no sólo la precisión total del modelo, sino la tasa de positivos verdaderos, tenemos las matrices de confusión:

NEURAL NETWORK					
MODELO SMOTE			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,553488372
	VALOR REAL	Clase: 0	2546	936	ESPECIFICIDAD
		Clase: 1	96	119	0,731188972
MODELO NORMAL			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,139534884
	VALOR REAL	Clase: 0	3445	37	ESPECIFICIDAD
		Clase: 1	185	30	0,989373923

Ilustración 29: Matrices de confusión

Como se observa en la (Ilustración 29), el modelo entrenado con los datos originales tiene una especificidad (Tasa de Negativos Verdaderos) de un 98.93%, esto es debido a la alta proporción de la clase “0” que existe en los datos originales lo que lleva al modelo a tomar casi siempre la decisión de clasificar a los individuos en la clase 0. En cambio, la sensibilidad (Tasa de Positivos Verdaderos) es del 7.44%. Para el modelo entrenado con los datos obtenidos de la función Smote, se observa que la matriz de confusión toma mucho más a menudo la decisión de clasificar a los individuos en la clase 1, lo que conlleva a una peor especificidad 73.1% pero una mejora de la sensibilidad, un 55.34%.

Las Curvas ROC de ambos modelos, el entrado por el fichero original (Ilustración 30) y el entrenado por el fichero generado con la función SMOTE (Ilustración 31) son las siguientes:

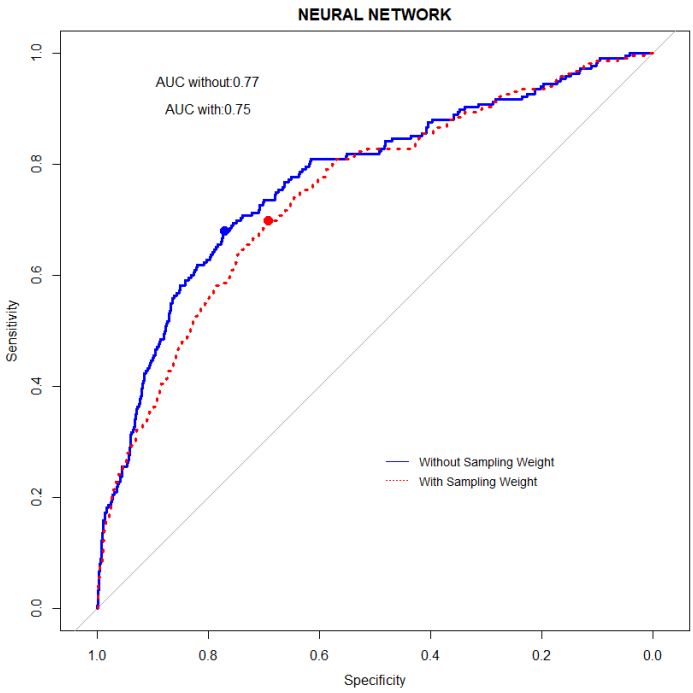

Ilustración 30: Curva ROC del modelo NNET para el fichero original

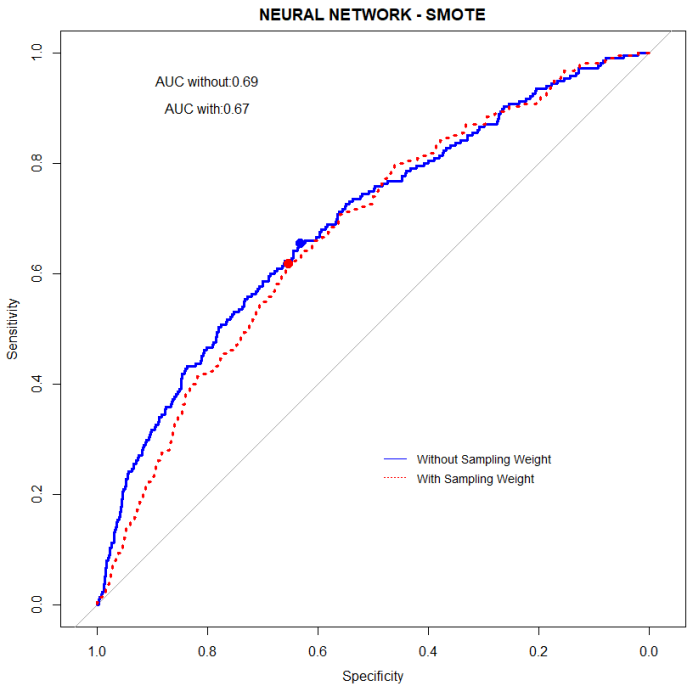

Ilustración 31: Curva ROC del modelo NNET para el fichero SMOTE

Se ve una diferencia entre la curva ROC del fichero original, donde tiene un AUC de 0.77 y 0.75 sin considerar los pesos de muestreo y teniéndolos en cuenta, respectivamente, con la curva ROC del fichero SMOTE, donde los valores del AUC son de 0.69 y 0.67, respectivamente.

Comparación de modelos

A lo largo de todo el TFM se han visto diferentes modelos con diferentes técnicas para predecir la probabilidad de error que tienen las unidades de una determinada encuesta. En este apartado, se van a ver los modelos vistos hasta el momento para poder compararlos entre sí. En total se van a comparar 6 modelos resultantes, 3 con datos originales para cada una de las técnicas (Random Forest, SVM y Neural Network) y otros 3 utilizando el fichero generado por la función SMOTE.

Fijando la mirada en la precisión esperada de los modelos, la ilustración 32 refleja la precisión de cada uno de ellos:

	modelo	Accuracy	Kappa
	<chr>	<dbl>	<dbl>
1	randfor_target1	0.948	0.186
2	svmrad_target1	0.945	0.155
3	nnet_target1	0.934	0.172
4	randfor_target1_Smote	0.769	0.517
5	svmrad_target1_Smote	0.745	0.475
6	nnet_target1_Smote	0.721	0.426

Ilustración 32: Precisión e índice Kappa de cada uno de los modelos

Los modelos que obtienen más precisión son los modelos entrenados con los datos originales, como se ha visto anteriormente. Esto se debe a la alta proporción de la clase "0" (94.18%) que existe en los datos originales es lo que lleva al modelo a tomar casi siempre la decisión de clasificar a los individuos en esa. Los modelos generados por SMOTE tienen una precisión más baja, pero en cambio tienen el índice Kappa mucho mejor.

El índice Kappa mide el grado de concordancia de las evaluaciones nominales u ordinales realizadas por múltiples evaluadores cuando se evalúan las mismas muestras. Esta medida es muy útil cuando se trabaja con datos desbalanceados, como es el caso, ya que esta medida tiene en cuenta las concordancias que ocurren por azar. El índice kappa oscila entre -1 y 1, donde la interpretación de los valores es:

- $-1 \leq \text{Kappa} < 0$, la concordancia es más débil que lo esperado en virtud de las probabilidades.
- Kappa = 0, la concordancia es la misma que se esperaría en virtud de las probabilidades
- $0 < \text{Kappa} < 1$, Según se acerca el índice a 1, la concordancia aumenta.
- Kappa = 1, existe concordancia perfecta.

Si se calcula las predicciones del conjunto de datos de test para cada uno de los modelos se obtienen las precisiones reales (Ilustración 33):

modelo	accuracy_validacion	accuracy_test
<chr>	<dbl>	<dbl>
randfor_target1	0.948	0.946
svmrad_target1	0.945	0.944
nnet_target1	0.934	0.940
randfor_target1_Smote	0.769	0.836
svmrad_target1_Smote	0.745	0.823
nnet_target1_Smote	0.721	0.721

Ilustración 33: Precisiones reales y esperadas de los modelos

Las precisiones reales de los modelos entrenados con los datos originales son muy parecidas a las precisiones esperadas. En cambio, los modelos entrenados con los ficheros generados con la función SMOTE, tienen una mejora considerable de la precisión real a la esperada.

La imagen de la ilustración 34 contiene las tablas de contingencia de los 6 modelos:

NEURAL NETWORK					
MODELO SMOTE			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,553488372
	VALOR REAL	Clase: 0	2546	936	ESPECIFICIDAD
		Clase: 1	96	119	0,731188972
MODELO NORMAL			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,139534884
	VALOR REAL	Clase: 0	3445	37	ESPECIFICIDAD
		Clase: 1	185	30	0,989373923

RANDOM FOREST					
MODELO SMOTE			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,511627907
	VALOR REAL	Clase: 0	2979	503	ESPECIFICIDAD
		Clase: 1	105	110	0,855542791
MODELO NORMAL			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,102325581
	VALOR REAL	Clase: 0	3477	5	ESPECIFICIDAD
		Clase: 1	193	22	0,998564044

SVM (KERNEL RADIAL)					
MODELO SMOTE			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,469767442
	VALOR REAL	Clase: 0	2942	540	ESPECIFICIDAD
		Clase: 1	114	101	0,844916715
MODELO NORMAL			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,069767442
	VALOR REAL	Clase: 0	3476	6	ESPECIFICIDAD
		Clase: 1	200	15	0,998276852

Ilustración 34: Tablas de contingencia de los modelos

Todos los modelos entrenados con los datos originales tienen una alta tasa de negativos verdaderos (98.94%, 99.86% y 99.83%), lo que implica una tasa de positivos verdaderos muy baja. Lo que se busca es conseguir encontrar el mayor número de positivos sin penalizar demasiado a la tasa de negativos verdaderos, por lo que los modelos que se han entrenado con los datos originales no son muy recomendables si se buscara sólo la predicción de si la unidad es correcta o no, ya que la mayoría de los positivos los pasaría por alto. Dentro de los modelos entrenados con el fichero

generado por la función SMOTE se observa que la tasa de negativos verdaderos disminuye considerablemente (73.12%, 85.55% y 84.49%), respectivamente. Con esto se consigue un aumento de la tasa de positivos verdaderos.

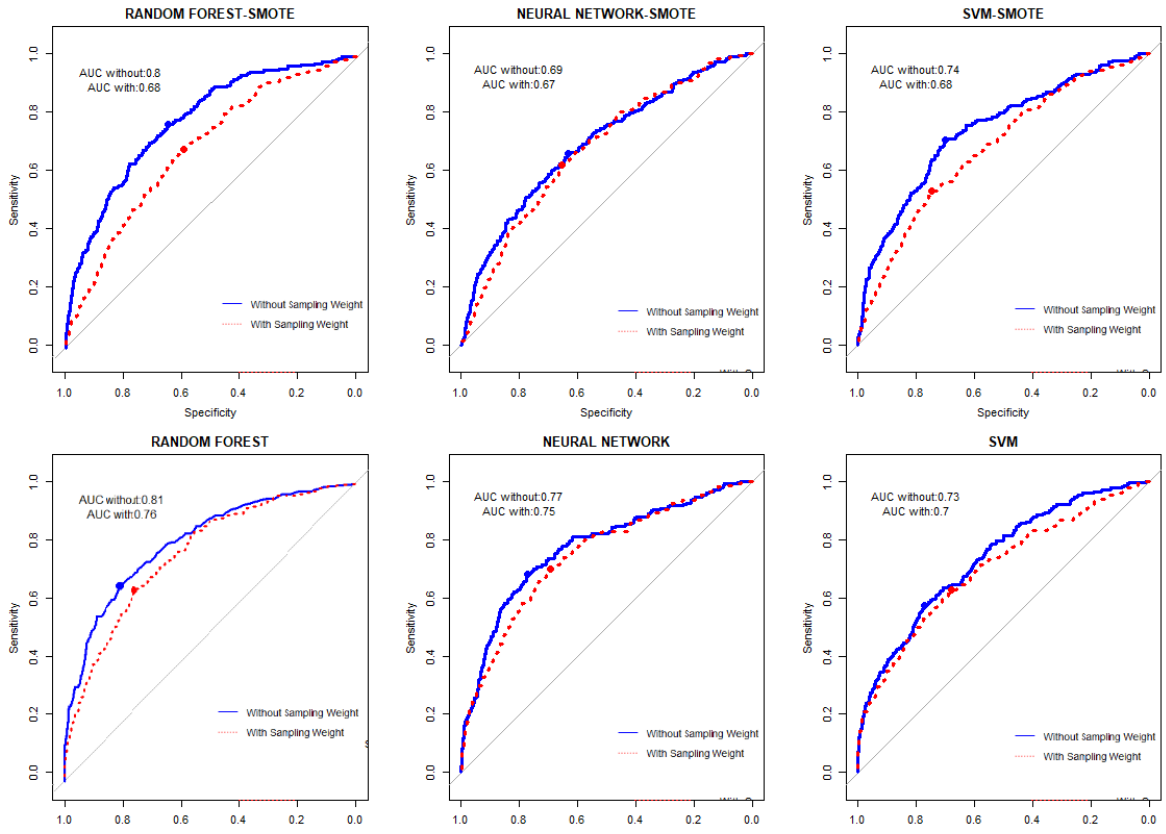

Ilustración 35: Curvas ROC de los 6 modelos

En la imagen anterior (Ilustración 35), se puede realizar una comparación de los 6 modelos. En general, el mejor AUC lo tiene la técnica del Random Forest, independientemente si estamos trabajando con los datos originales o con los datos balanceados. Si se considera la técnica Random Forest, dentro de ésta comparando los dos modelos obtenidos, el AUC es bastante similar, el AUC generado sin tener en cuenta los pesos de muestreo (azul) son 0.8 y 0.81, y en donde se obtiene mayor diferencia es en el AUC considerando los pesos de muestreo (rojo).

Ensamblaje de modelos

A partir de los modelos entrenados anteriormente se puede hacer un ensamblado de modelos. Existen diferentes métodos de ensamblaje de modelos, en este TFM vamos a centrarnos en tomar el promedio de las predicciones de los modelos.

Este método sólo consiste en tomar la moda de los modelos como predicción. Se van a realizar dos ensamblajes distintos, uno para los modelos entrenados con los datos originales, y otro ensamblaje para los modelos que han sido entrenados con el fichero generado con la función SMOTE.

Se construye una tabla (Ilustración 36) con las predicciones de todos los modelos, y como predicción del ensamblaje de modelos se toma la moda de los 3 modelos propuestos.

svmrad_target1_Smote	randfor_target1_Smote	nnet_target1_Smote	moda
X0	X0	X0	X0
X1	X1	X0	X1
X1	X1	X1	X1
X0	X0	X0	X0
X0	X1	X1	X1
X0	X0	X0	X0

Ilustración 36: Tabla para el ensamblaje de modelos

Después de realizar las predicciones que se obtienen en los ensamblajes de ambos modelos, las precisiones totales obtenidas de los modelos son 94.56% para el ensamblaje de los modelos entrenados con datos originales, y un 82.63% para el ensamblaje de los modelos entrenados con el fichero SMOTE. En ninguno de los dos casos se mejoran las precisiones que obtuvieron los modelos por separado, aunque esta métrica no es la que se busca.

La matriz de confusión de ambos ensamblajes de modelos es la siguiente ilustración:

NEURAL NETWORK

			VALOR PREDICHO		SENSIBILIDAD
MODELO SMOTE			Clase: 0	Clase: 1	0,553488372
	VALOR REAL	Clase: 0	2546	936	ESPECIFICIDAD
		Clase: 1	96	119	0,731188972
MODELO NORMAL			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,139534884
	VALOR REAL	Clase: 0	3445	37	ESPECIFICIDAD
		Clase: 1	185	30	0,989373923

SVM (KERNEL RADIAL)

			VALOR PREDICHO		SENSIBILIDAD
MODELO SMOTE			Clase: 0	Clase: 1	0,469767442
	VALOR REAL	Clase: 0	2942	540	ESPECIFICIDAD
		Clase: 1	114	101	0,844916715
MODELO NORMAL			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,069767442
	VALOR REAL	Clase: 0	3476	6	ESPECIFICIDAD
		Clase: 1	200	15	0,998276852

RANDOM FOREST

			VALOR PREDICHO		SENSIBILIDAD
MODELO SMOTE			Clase: 0	Clase: 1	0,511627907
	VALOR REAL	Clase: 0	2979	503	ESPECIFICIDAD
		Clase: 1	105	110	0,855542791
MODELO NORMAL			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,102325581
	VALOR REAL	Clase: 0	3477	5	ESPECIFICIDAD
		Clase: 1	193	22	0,998564044

ENSAMBLAJE DE MODELOS

			VALOR PREDICHO		SENSIBILIDAD
MODELO SMOTE			Clase: 0	Clase: 1	0,51627907
	VALOR REAL	Clase: 0	2944	538	ESPECIFICIDAD
		Clase: 1	104	111	0,845491097
MODELO NORMAL			VALOR PREDICHO		SENSIBILIDAD
			Clase: 0	Clase: 1	0,088372093
	VALOR REAL	Clase: 0	3477	5	ESPECIFICIDAD
		Clase: 1	196	19	0,998564044

Ilustración 37: Matrices de confusión de los modelos y los ensamblajes

Los resultados obtenidos en los ensamblajes (Ilustración 37), son muy parecidos a los obtenidos en los modelos, y en ningún caso mejoran los resultados que se han obtenido en el modelo del Random Forest.

Las curvas ROC de los modelos obtenidos con el ensamblaje se pueden ver en la siguiente ilustración:

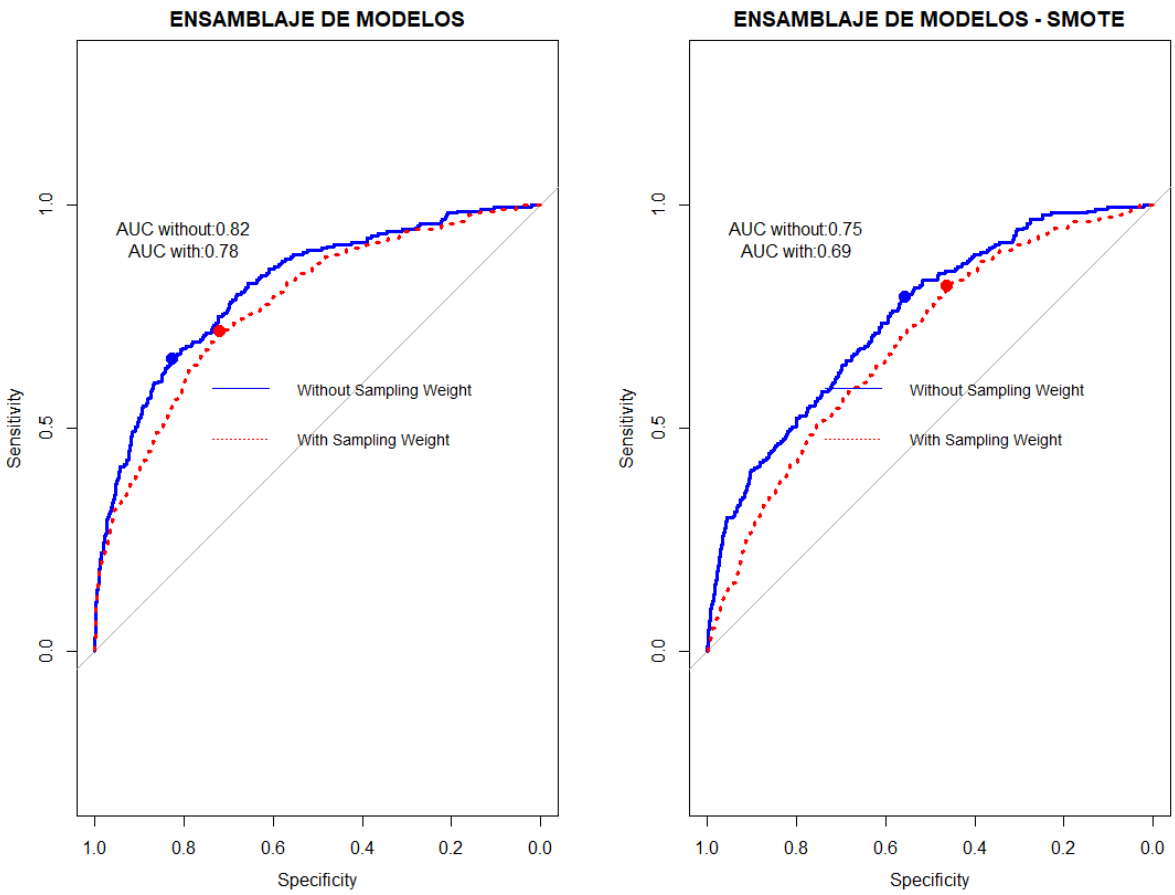

Ilustración 38: Curvas ROC del ensamblaje de los modelos

Las curvas ROC también son bastante similares a las de los modelos ya existentes (Ilustración 38), para el ensamblaje de los modelos entrenados con los datos originales se ve una ligera mejora en el AUC tanto en la curva teniendo en cuenta los pesos de muestreo, como en la que no los tiene en cuenta. Para las curvas ROC del ensamblaje de los modelos entrenados con el fichero generado por la función SMOTE no existe mejoría con respecto al modelo Random Forest.

CONCLUSIONES

El objetivo principal de este TFM es conseguir una ordenación de las unidades estadísticas para ser revisadas en la depuración interactiva siguiendo orden indicado. Teniendo en mente este objetivo, vamos a obtener el listado ordenado de las unidades del conjunto de datos test según su probabilidad de error y su momento de error para cada uno de los 6 modelos. Una vez se tiene el listado ordenado, vamos a imaginar que el Instituto Nacional de Estadística sólo tiene recursos para revisar 1000 unidades de ese listado, ¿cuál sería el resultado?

TOTAL UNIDADES	3697
UNIDADES CON ERROR	215
REVISIÓN	1000

PESO TOTAL	6.713.558,8
PESO UNID ERROR	411.772,4

Ilustración 39: Resumen de los datos de entrenamiento

(Ilustración 39) El conjunto de datos test tiene un total de **3697 unidades,** de las cuales hay **215 unidades** que tienen error. Las unidades han sido ordenadas según los Local Score obtenidos de los modelos (con y sin pesos de muestreo) para hacer una **revisión de 1000 unidades** según el orden indicado en cada uno de los modelos. El peso total de las unidades que contienen error (215 unidades) es de **411.772,4**.

En la ilustración 40 se pueden ver los resultados que se han obtenido por modelo. Dentro de cada modelo se ha considerado la ordenación de los Local Score con y sin pesos de muestreo, y se han obtenido los porcentajes de unidades corregidas de las 215 totales erróneas, y también se ha medido el porcentaje de pesos de muestreo de las unidades que han sido corregidas con respecto el total de peso de muestreo de las unidades erróneas (411.722,4).

SVM			
SIN PESOS	UNID CORREGIDAS	127	59,07%
	PESO CORREGIDAS	238.757,55	57,98%
CON PESOS	UNID CORREGIDAS	119	55,35%
	PESO CORREGIDAS	289.303,06	70,26%

RANDOM FOREST			
SIN PESOS	UNID CORREGIDAS	149	69,30%
	PESO CORREGIDAS	269739,372	65,51%
CON PESOS	UNID CORREGIDAS	135	62,79%
	PESO CORREGIDAS	303.032,95	73,59%

NEURAL NETWORK			
SIN PESOS	UNID CORREGIDAS	149	69,30%
	PESO CORREGIDAS	287.474,33	69,81%
CON PESOS	UNID CORREGIDAS	134	62,33%
	PESO CORREGIDAS	301.643,88	73,26%

SVM SMOTE			
SIN PESOS	UNID CORREGIDAS	132	61,40%
	PESO CORREGIDAS	250.875,92	60,93%
CON PESOS	UNID CORREGIDAS	114	53,02%
	PESO CORREGIDAS	290.195,06	70,47%

RANDOM FOREST SMOTE			
SIN PESOS	UNID CORREGIDAS	140	65,12%
	PESO CORREGIDAS	253.852,16	61,65%
CON PESOS	UNID CORREGIDAS	105	48,84%
	PESO CORREGIDAS	284.981,73	69,21%

NEURAL NETWORK SMOTE			
SIN PESOS	UNID CORREGIDAS	114	53,02%
	PESO CORREGIDAS	205.606,62	49,93%
CON PESOS	UNID CORREGIDAS	103	47,91%
	PESO CORREGIDAS	241.978,00	58,76%

Ilustración 40: Resultados de la revisión de las unidades por modelo y por datos de entrenamiento

Revisando los resultados de la prueba se puede ver que los modelos que han sido entrenados con los ficheros generados por SMOTE no han sido muy efectivos. El porcentaje de unidades que han sido revisadas y el total de pesos de muestreo corregidos son menores a los de los modelos entrenados con los datos originales, donde se observan mejores resultados. Dentro de estos, los mejores resultados se obtienen con las técnicas de Random Forest y Neural Network,

donde, si se tienen en cuenta los pesos de muestreo, se corrigen un 62.79% y 62.33%, respectivamente, de las unidades con error. Para la cantidad de peso de muestreo corregido se obtienen unos valores de 73.59 y 73.26, respectivamente. Si no se tiene en cuenta los pesos de muestreo a la hora de hacer las predicciones las unidades corregidas aumentan aproximadamente un 7%, pero en cambio se pierde peso de muestreo que ha sido corregido.

Si se quiere hacer el mismo análisis para el ensamblaje de los 3 modelos, los resultados que se han obtenido se pueden ver en la ilustración 41:

ENSAMBLAJE			
SIN PESOS	UNID CORREGIDAS	151	70,23%
	PESO CORREGIDAS	279707,858	67,93%
CON PESOS	UNID CORREGIDAS	144	66,98%
	PESO CORREGIDAS	321.763,34	78,14%

ENSAMBLAJE SMOTE			
SIN PESOS	UNID CORREGIDAS	122	56,74%
	PESO CORREGIDAS	223.165,08	54,20%
CON PESOS	UNID CORREGIDAS	109	50,70%
	PESO CORREGIDAS	280.713,90	68,17%

Ilustración 41: Resultados de la revisión de las unidades en el ensamblaje de los 3 modelos

Cabe destacar que se han obtenido los mejores resultados para el ensamblaje de los modelos entrenados con los datos origen, donde se ha conseguido corregir un 66.98% de las unidades con error, arreglando un 78.14% de los pesos de muestreo con error.

Las unidades más importantes y que no se deben dejar entrar con error, son las que tienen más peso sobre la población. Como primera conclusión, sí sería necesario introducir los pesos de muestreo a la hora de realizar la ordenación, ya que, aunque se estén corrigiendo menos unidades erróneas, las que se corrigen tiene un mayor peso.

Viendo el resultado anterior, se va a analizar la evolución de los métodos según la ordenación, teniendo en cuenta los pesos de muestreo para dicha ordenación. En el ejemplo anterior pusimos 1000 unidades a revisar, se va a ver la evolución de los diferentes métodos, según las unidades estadísticas que vayan a ser revisadas (100, 300, 500, 700 y 1000) viendo en cada caso cuantas unidades estadísticas erróneas se han encontrado y los pesos de error que han sido corregidos (Ilustración 42 y 43).

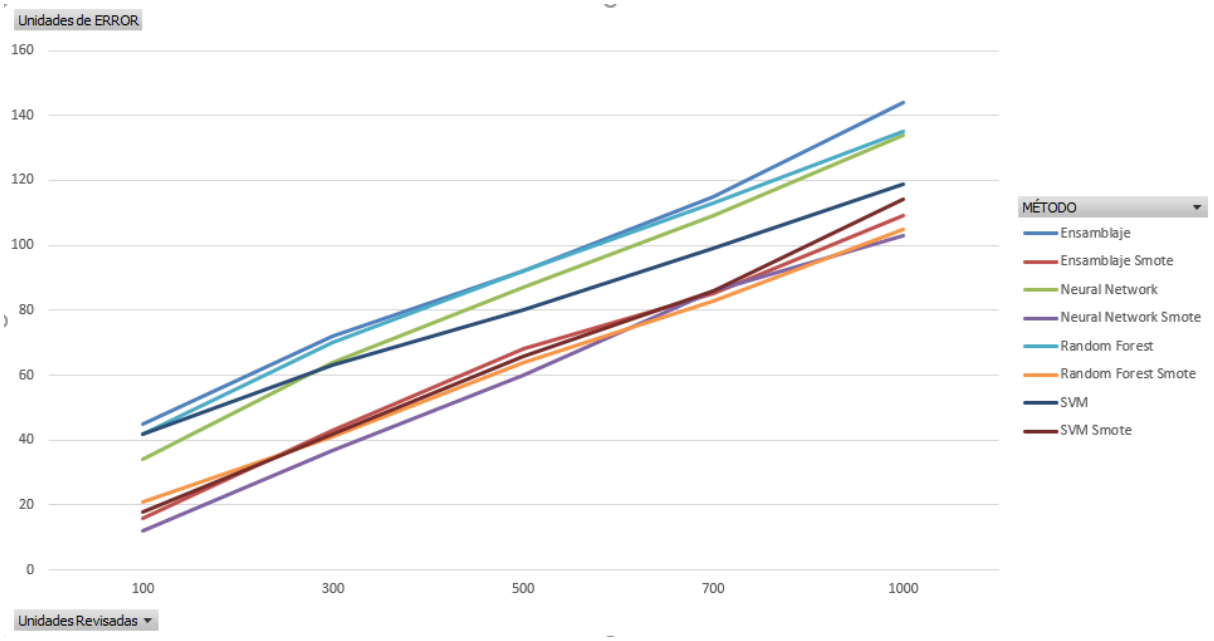

Ilustración 42: Evolución de las unidades de Error

En la Ilustración 42 se observa la progresión de las unidades de error corregidas según las unidades revisadas. Como primera observación, los métodos que han sido entrenados con el fichero generado con SMOTE corrigen muchas menos unidades erróneas que los métodos entrenados con los datos originales. Dentro de los métodos entrenados con el fichero origen, el que mejor resultado obtiene es el de ensamblaje de modelos en todos los casos.

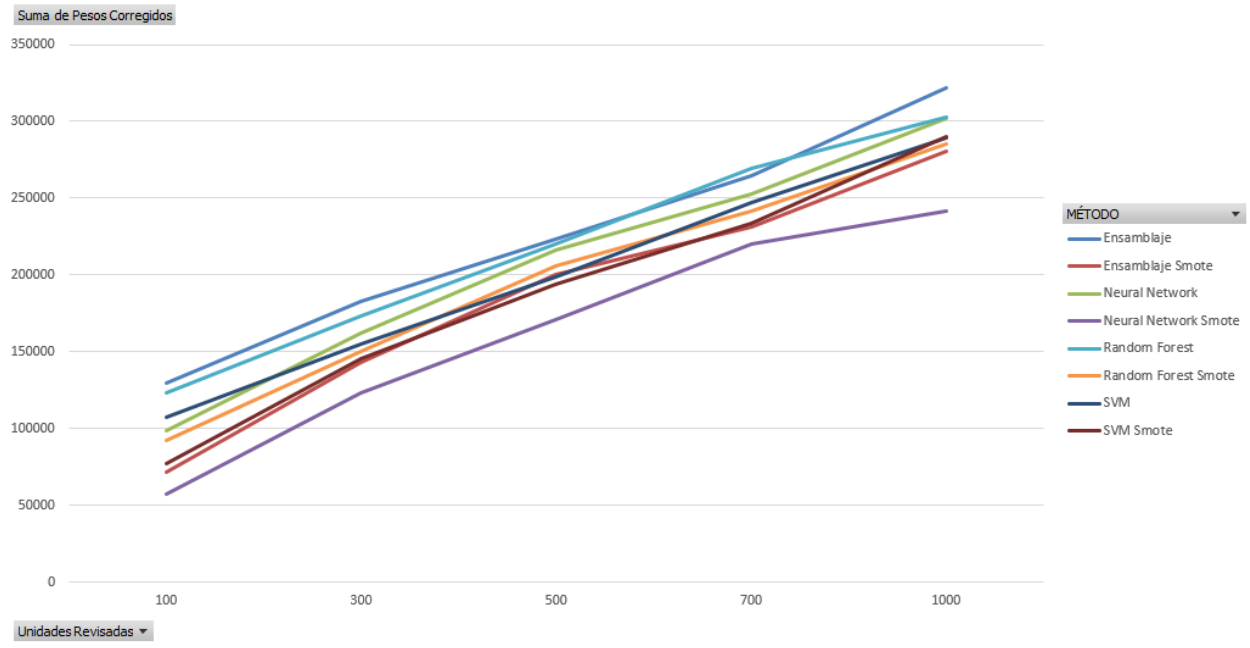

Ilustración 43: Evolución de la Suma de los pesos de las unidades corregidas

En la Ilustración 43, se observa la progresión de la suma de los pesos de las unidades de error corregidas según las unidades revisadas. Igual que para el caso anterior, los mejores resultados se obtienen con el ensamblaje de modelos.

Después de ver estos resultados, se podrían sacar las siguientes pautas a la hora de hacer una depuración selectiva de variables categóricas:

- No utilizar técnicas de muestreo para corregir los datos desbalanceados. La técnica utilizada (SMOTE) no ha conseguido mejorar los resultados ni corregir los datos desbalanceados. El principal problema de esta técnica es que se generan registros nuevos utilizando a unidades cercanas, por lo que al trabajar con unidades categóricas no es posible generar un registro nuevo haciendo. Por ejemplo, la media de los valores de los vecinos más próximos. Un poco más adelante se propone una línea futura de trabajo, la investigación de técnicas de muestreo para datos desbalanceados de variables categóricas.
- Utilizar un ensamblaje de modelos. Aunque cuando se ha analizado el ensamblaje de modelos, los resultados parecían que no iban a mejorar mucho con la situación inicial, se ha comprobado que se obtiene una mejora considerable en el objetivo marcado en este TFM.

Líneas de trabajo futuras:

- En este TFM sólo se ha visto introducir una variable de estudio “target1”, por lo que no existe la posibilidad de construir las Global Score de las unidades. Introducir 3 variables de estudio “target1”, “target2” y “target3” para la construcción de las Global Score, y analizar los resultados obtenidos.
- Como la función SMOTE no mejora los resultados de las predicciones, utilizar otros métodos de muestreo para intentar corregir los datos desbalanceados para variables categóricas.
- Utilización de otros métodos de ensamblaje para ver si se mejora la predicción de los modelos ya existentes.
- Utilización de otras técnicas de Machine Learning para incluirlas en el ensamblaje de los modelos, o para sustituir a alguno de los modelos ya planteados.
- En lugar de identificar las unidades erróneas con una variable dicotómica, intentar dar una predicción de la clase a la que correspondería el individuo.
- Introducir la variable “Coste” en el análisis, donde estudiar si se le asigna un coste fijo a revisar una unidad más del conjunto de unidades, con respecto al coste de que una unidad errónea no sea revisada, y optimizarlo.

BIBLIOGRAFÍA Y REFERENCIAS

1 Luzi, Orietta & Zio, Marco & Guarnera, Ugo & Manzari, Antonia & De Waal, Ton & Pannekoek, Jeroen & Hoogland, J & Templeman, C & Hulliger, Beat & Kilchman, D. (2007). Recommended practices for editing and imputation in cross-sectional business surveys.

2 Bercebal, J.M. & Maldonado, J.L. & Martínez-Vidal, M.A. & Salgado, David. (2015). Data collection and selec-tive data editing in a systematized and integrated way: an experience in progress at Statistics Spain.

3 Ton de Waal, Jeroen Pannekoek, Sander Scholtus (2011): "Handbook of Statistical Data Editing and Imputation"

4 Arbués, Ignacio & Revilla, Pedro & Salgado, David. (2013). An Optimization Approach to Selective Editing. Journal of official statistics. 29. 489. 10.2478/jos-2013-0037.

5 Depto. Metodología y Desarrollo de la Producción Estadística (INE): "Depuración Selectiva de la variable CNO en las Encuestas Nacional y Europea de Salud" (2018-10-17)

6 Depto. Metodología y Desarrollo de la Producción Estadística (INE): "Depuración Selectiva de la variable CNO en las Encuestas Nacional y Europea de Salud. Selección Manual de modelos" (2018-10-29)

7 "Memobust Handbook" on Methodology of Modern Business Statistics (2014-3-26).

8 Amat Rodrigo, Joaquín: "Clasificación de tumores con Machine Learning" (Mayo – 2018)
URL: https://rpubs.com/Joaquin_AR/387758

9 Efron, B.; Tibshirani, R. J. An Introduction to the Bootstrap; Monographs on Statistics and Applied Probability, 57; Chapman & Hall: Boca Raton, Fla., 1998.

10 Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2013): "An Introduction to Statistical Learning with Applications in R" (Springer)

11 Avinash Navlani: "Neural Network Models in R" (Fecha del artículo: 18/01/2019. Fecha de Acceso: 22/07/2019).
URL: https://www.datacamp.com/community/tutorials/neural-network-models-r

12 Pedro Larranaga, Iñaki Inza, Abdelmalik Moujahid Departamento de Ciencias de la Computación e Inteligencia Artificial - Universidad del País Vasco–Euskal Herriko Unibertsitatea: "Redes Neuronales". (Fecha de Acceso: 22/07/2019)
URL: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t14-neuronales.pdf

13 Vélez Serrano, Daniel: "Tratamiento Estadístico y Computacional de la Información: Redes neuronales" (Curso 2017/2018)

14 Documento de metodología para la Encuesta Nacional de Salud (INE)
URL: https://www.ine.es/metodologia/t15/t153041912.pdf

ANEXOS

Todos los documentos se encuentran en el siguiente repositorio:

https://github.com/Enrique-Santiago/Machine-Learning.git

Dentro del repositorio están disponibles los siguientes ficheros:

- **RandomForestSmote.R**: Código R que contiene toda la parte inicial del TFM, donde se entrena un modelo Random Forest para los 18 ficheros generados con la función SMOTE y los 4 grupos de variables explicativas distintos.
- **Bootstrap.R**: Código R que contiene la segunda parte del TFM en que se entrenan los modelos Neural network, Random Forest y SVM para la configuración elegida en la parte inicial.
- **Ensamblaje_Modelos.R**: Última parte del TFM donde se hace un promedio de los modelos entrenados anteriormente con el fin de mejorar la precisión de los resultados.
- **Random Forest**: Carpeta que incluye todos los modelos, guardados en formato "rds", de la parte inicial del TFM. También se incluye un Excel con todas las matrices de confusión generadas por los mismos. Dentro de esta carpeta se incluye la siguiente carpeta:
  - **Curvas ROC**: Todas las curvas ROC generadas en el análisis.
- **Modelos CARET**: Carpeta que incluye todos los modelos, guardados en formato "rds", de la segunda parte del TFM. También se incluye un Excel con todas las matrices de confusión generadas por los mismos. Dentro de esta carpeta se incluye la siguiente carpeta:
  - **Gráficos**: Incluye todos los gráficos generados en el proceso de análisis, así como las curvas ROC generadas.