CAMPUS
DE EXCELENCIA
INTERNACIONAL

Máster Universitario en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de

Ingenieros Informáticos

TRABAJO FIN DE MÁSTER

"Predicción de la evolución de pacientes tras daño cerebral causado por trauma."

Autor: Abel de Andrés Gómez

Director: Antonio LaTorre

MADRID, JULIO 2018

Agradecimientos

Desde estas líneas me gustaría expresar mi más sincero agradecimiento:

A mi tutor Antonio LaTorre, por ayudarme y guiarme durante todo el proyecto.

A mis padres, por haberme proporcionado la mejor educación y lecciones de vida.

En especial a mi padre, por haberme enseñado que, con esfuerzo, trabajo y constancia todo se consigue, y que en esta vida nadie regala nada.

En especial a mi madre, por hacerme ver cada día la vida de una forma diferente y confiar en mis decisiones. Sin olvidar su gran ayuda durante estos últimos años.

A todos mis familiares por haberme apoyado y animado.

A mis compañeros de clase, con los que he compartido buenos y malos momentos.

A mis amigos, por estar siempre a mi lado.

A todos aquellos que siguen estando cerca de mí y que le regalan a mi vida algo de ellos

Resumen

Las lesiones traumáticas cerebrales son un problema de salud mundial. Por tanto, se hace necesario el estudio de los predictivos considerados medicinalmente más importantes. El motivo es establecer estrategias, decisiones clínicas y protocolos para el tratamiento de estas lesiones de forma óptima y tratando siempre de obtener el mejor resultado en el paciente.

En los estudios que se han realizado se han utilizado datos de pacientes de distintos hospitales que han sufrido una lesión traumática cerebral. Con esta información se han construido varios modelos que permitan predecir el estado de los pacientes después de sufrir estas lesiones y poder construir así diagnósticos en futuros pacientes.

En este estudio se realiza una clasificación sobre la importancia de algunos predictores además se realizarán estudios estadísticos para obtener la relación que existen entre estos predictores para posteriormente utilizar distintos modelos con el objeto de conseguir la mayor predicción posible.

Para obtener el mejor modelo se utilizarán distintas métricas con el objetivo de poder compararlos entre sí en función de las necesidades que se requieran.

Abstract

Traumatic brain injuries are a worldwide health problem. Thus, it is completely necessary the study of the more relevant predictors. The aim of the study is to stablish the strategies, clinical decisions and protocols for the treatment of these injuries in an optimal way, always trying to achieve the best result for the patient.

In order to carry out the studies, data of patients who have suffered from traumatic brain injuries have been examined. Moreover, the data for the study have been selected from different hospitals. With the information obtained, different models have been created. These models allow to predict the state of the patients after suffering from these lesions. Additionally, the models will also allow to stablish correct diagnosis for future patients.

The current study carried out a classification about the importance of certain predictors. Furthermore, statistic studies have been conducted in order to know the existing relationship between there predictors. Once it has been stablished, different models will be used in order to attain the finest prediction possible.

In order to obtain the best model, different predictors will be used in order to compare them when different circumstances will require it.

Tabla de Contenidos

Listado de Figuras

Listado de Tablas

1. INTRODUCCIÓN Y OBJETIVOS

1.1 INTRODUCCIÓN

Las lesiones traumáticas cerebrales (**T**raumatic **B**rain **I**njury) conocidas también como lesiones cerebrales o de la cabeza ocurren cuando un golpe, impacto, sacudida u otras lesiones en la cabeza causan daño al cerebro. Son principalmente el resultado de accidentes vehiculares, caídas, actos de violencia y lesiones deportivas.

Se estima que aproximadamente 2 millones de personas sufren anualmente de lesiones cerebrales, el índice de incidencia estimado es de 100 cada 100000 personas. Cada año se producen alrededor de 500000 lesiones cerebrales lo suficientemente graves como para exigir la hospitalización, llegando a causar 52000 fallecimientos anuales.

Las lesiones más comunes se dan entre los varones de 15 y 24 años, pudiendo ocurrir con cualquier edad. Muchas de estas lesiones son benignas y los síntomas desaparecen con el tiempo si reciben la atención adecuada. En otros casos el daño es más grave y puede provocar una incapacidad permanente e incluso la muerte.

Las consecuencias negativas tras un TBI dependen de muchos factores entre los que se pueden destacar la rapidez del diagnóstico y el tratamiento adecuado, que puede contribuir a aliviar algunas consecuencias de las lesiones. Por lo general es complicado predecir las consecuencias de una TBI en las primeras horas e incluso en los primeros meses ya que las consecuencias pueden permanecer ocultas durante muchos meses después.

En este contexto podría ser muy útil realizar un estudio de predicción que nos permita saber las consecuencias de las lesiones en los próximos 6 meses justo después de haber tenido una lesión traumática cerebral.

1.2 OBJETIVOS

Con esta propuesta de Trabajo de Fin de Máster se persigue analizar una serie de datos provenientes de 10.008 pacientes de 239 hospitales situados en 49 países que han sufrido una lesión cerebral traumática (TBI).

El objeto de análisis de los datos es poder crear un modelo idóneo que nos permita predecir la evolución de los pacientes en los próximos seis meses, pudiendo también determinar su estado.

Esta predicción se realizará a partir del conjunto de datos dado, entre los que se encuentran variables tan importantes como la edad, los resultados del paciente habiendo evaluado su estado sobre la escala GSW (escala de Glasgow) y los escáneres realizados.

Para conseguir una buena predicción, se probarán varios modelos, teniendo en cuenta ciertas variables y descartando otras. Para la predicción se utilizará una serie de modelos y se tendrán en cuenta sus resultados con el objetivo de ver si el modelo es bueno para realizar predicciones y compararlos entre ellos.

La meta a la que se pretende llegar con este trabajo es, en definitiva, realizar un estudio completo de los datos, pasando por todas las etapas del análisis de datos y comparar varios tipos de modelos. Deberemos obtener el mejor modelo, teniendo en cuenta las características de los datos, que nos permita realizar predicciones tempranas a partir de la información de un paciente que haya sufrido un daño cerebral traumático.

La lista de tareas que se han definido para conseguir los objetivos establecidos es la siguiente:

- Estudio del estado del arte y familiarización con el conjunto de datos con el que se va a trabajar.
- Limpieza y preparación del conjunto de datos.
- Estudio y comparación de modelos de aprendizaje sobre el conjunto de datos.

- Validación final del modelo utilizado.
- Análisis y documentación de los resultados.

1.3 ESTRUCTURA

La estructura que va a definir el Trabajo Fin de Máster será la siguiente:

Capítulo 1. Introducción y Objetivos: En el primer capítulo se definen las necesidades que nos llevan al desarrollo de este trabajo. También se van a definir los objetivos que se persiguen con la realización de este. Por último, se presentará la estructura que tendrá el presente documento.

Capítulo 2. Estado del Arte: El objetivo que se pretende conseguir en este capítulo es introducir y situar al lector en el ámbito de Big Data, sus fases y las herramientas que se utilizan para lograr los objetivos deseados.

Capítulo 3. Desarrollo: En este capítulo se realiza una explicación detallada de todos los pasos que se van a seguir para conseguir obtener información de nuestros datos.

Capítulo 4. Resultado: En esta sección se valorarán los resultados que se han obtenido tras el uso de los distintos modelos analizados.

Capítulo 5. Líneas futuras: Este capítulo se centrará en las propuestas de líneas de trabajo futuro en este contexto.

2. ESTADO DEL ARTE

2.1 DATA SCIENCE

Cuando hablamos de la ciencia de datos ("Data Science") nos referimos a un campo de la estadística y de las ciencias de la computación que intenta descubrir patrones en grandes volúmenes de datos que nos puedan aportar información valiosa en la toma de futuras decisiones.

Cada día generamos una gran cantidad de información. Son muchos los motivos que nos llevan a generar este gran volumen de información, ya que la información nos puede ayudar a controlar, optimizar, administrar, examinar, investigar, predecir, negociar, tomar decisiones en cualquier ámbito en el que estemos.

En los últimos años, gracias al gran desarrollo tecnológico que hemos vivido, tanto a nivel de computo como a nivel de transmisión de datos y sin olvidarnos del abaratamiento de los sistemas de almacenamiento temporal como permanente, nos ha permitido llevar a cabo un gran abuso en el almacenamiento de información.

Centrándonos en el gran volumen de datos y de acuerdo al IDC (International Data Corporation), el volumen total de datos en 2.013 fue de 2.8ZB. Los seres humanos estamos creando y almacenando información constantemente y cada vez más en cantidades astronómicas y se espera que en 2.020 se alcancen los 40ZB, unos 5.247GB por persona.

Así pues, teniendo en cuenta este crecimiento exponencial de los datos, se hace necesario el uso de tecnologías que nos permitan explotar todo el potencial de este gran volumen de información.

Los principales objetivos que persigue la ciencia de datos son los siguientes:

- Descripción: El principal objeto de la ciencia de datos es el descubrimiento de reglas o patrones. Estas reglas mostraran nuevas relaciones entre las variables, ayudando a entender e interpretar los datos.
- Predicción: Las reglas y patrones descubiertos se utilizarán para estimar las variables de salidas.

2.2 ETAPAS EN EL PROCESAMIENTO DE DATOS

Las principales fases que se deben realizar a la hora de extraer información a partir de un conjunto de datos son las siguientes:

1. **Obtención de datos:** En esta primera fase lo que prima es la búsqueda y la obtención de los datos a partir de una fuente u origen. Los datos podrán ser estructurados, no estructurados, etc. Existen una serie de fuentes de donde se pueden sacar los datos, como, por ejemplo:
    a. GitHub
    b. Amazon
    c. Facebook
    d. Twitter
    e. Google (datos de mercado)
2. **Procesado de datos:** En esta fase lo que se persigue es la separación, agrupación y filtrado de datos, con el objetivo de producir que la información sea lo más significativa posible.
3. **Limpieza de datos:** Posteriormente, es necesario realizar una limpieza de los datos, puesto que muchas veces existen duplicaciones que son necesarias eliminar e incluso errores en los propios datos. Un ejemplo de estos errores podría ser los datos que se salen fuera de un intervalo cualitativo o cuantitativo determinado.
4. **Análisis exploratorio de datos:** Después de realizar la limpieza de datos estos se someterán a un tratamiento estadístico. Mediante este tratamiento estadístico se

buscarán tendencias, se obtendrán histogramas para detectar grupos y se visualizarán distintos gráficos (medias, modas, desviaciones, máximos y mínimos, correlación, normalidad, etc) con el fin de identificar el modelo teórico más adecuado para la representación de estos datos.

5. **Modelado:** Se utilizan los datos estadísticos obtenidos en la fase anterior con el objetivo de buscar el modelo que se adapte mejor a nuestros datos y que pueda proporcionarnos.
6. **Producto y visualización de datos**: Se utiliza aplicaciones como PowerBI, Pentaho, QlikView, PeriscopeData e incluso documentos Excel, con el objetivo de visualizar y obtener resultados dinámicos. Ademas, también podremos realizar informes por audiencias (comerciales, marketing, estrategia, dirección, técnicos, etc) con herramientas como STReport, JasperReport u otras herramientas de "*reporting*".

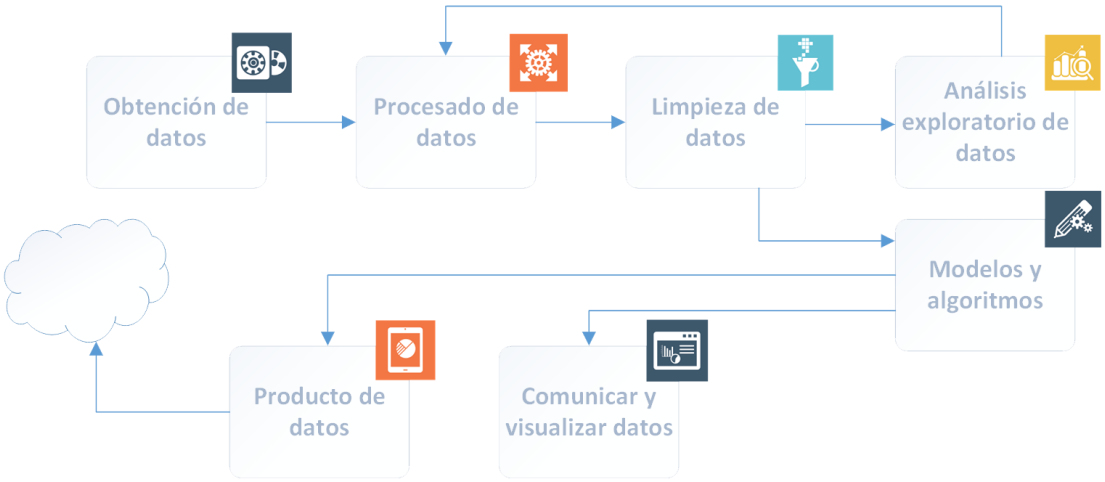


Ilustración 1. Fases de Big Data

Con nuestro proyecto alcanzaremos y trataremos los 5 primeros puntos anteriormente comentados.

2.3 HERRAMIENTAS PARA EL PROCESADO Y ANÁLISIS DE DATOS

A continuación, se van a describir algunas de las herramientas que se utilizan para realizar las fases del procesado y análisis de datos:

1. **Hadoop**

   Es una herramienta Big Data Open Source. Se considera el *"framework"* estándar para el almacenamiento de grandes volúmenes de datos. También se utiliza para el análisis y procesado de datos.

   Hadoop utiliza modelos de programación simple (aislando a los desarrolladores de las dificultades de la programación paralela) para el almacenamiento y el procesamiento distribuido. Hadoop distribuye por tanto el gran volumen de datos en nodos. Dispone, por consiguiente, de un sistema de archivos distribuido en cada nodo del cluster: HDFS (Hadoop Distributed File System) y se basa en el proceso de MapReduce.

2. **Apache Spark**

   Se trata de un motor de procesamiento de datos de código abierto. Realiza también una programación distribuida que consiste en distribuir el trabajo entre un conjunto de "clusters" que desde un punto de vista abstracto actúa como un solo ente que realiza el procesado. Se puede programar usando distintos lenguajes como Java, Scala, Python o R. Es bastante más rápido en memoria y en disco que Hadoop MapReduce.

3. **Apache Storm**

   Es un sistema de computación distribuida en tiempo real orientado a procesar flujos constantes de datos, por ejemplo, datos que provienen de Twitter, pudiendo realizar estudios sobre *"trending topics"* al momento.

4. **Lenguaje R.**

   Es un lenguaje y un entorno de software frecuentemente usado para el cálculo estadístico y la visualización de gráficos. Es utilizado para la minería de datos, la investigación bioinformática y las matemáticas financieras.

   R se asemeja más a un lenguaje matemático más que a un lenguaje de programación, por lo que puede ser un inconveniente para los programadores para realizar análisis de Big Data. Su punto fuerte es el gran número de librerías creadas por la comunidad entre otras herramientas.

5. **Python**

   Es un lenguaje avanzado cuya ventaja a otros lenguajes es su uso relativamente fácil para usuarios que no están familiarizados con la programación, pero que necesitan trabajar con análisis de datos.

   También dispone de una gran comunidad detrás de este lenguaje que proporcionan un gran número de librerías, haciendo de Python un lenguaje muy eficiente para realizar un procesado y análisis de datos.

2.4 CONCEPTOS BÁSICOS

2.4.1 MATRIZ DE CONFUSIÓN

Una matriz de confusión es una tabla que se usa a menudo para describir el rendimiento de un modelo de clasificación sobre en un conjunto de datos de prueba para los cuales se conocen los valores verdaderos.

Definamos ahora los términos más básicos para nuestro estudio:

- Verdaderos Positivos (TP): estos son casos en los que predijimos que sí (van a fallecer), y sí fallecen.

- Verdaderos Negativos (TN): Predijimos que no iban a fallecer, y los pacientes no fallecen.
- Falsos Positivos (FP): Predijimos que sí iban a fallecer, pero en realidad no han fallecido.
- Falsos Negativos (FN): Predijimos que no, pero en realidad tienen la enfermedad.

En nuestro estudio se ha tomado como positivo, el valor de "F", es decir, que el paciente fallezca.

Otra información a tener en cuenta en los gráficos:

- Reference V, Prediction V -> Verdaderos Negativos (TN).
- Reference F, Prediction V -> Falsos Negativos (FN)
- Reference V, Prediction F -> Falsos Positivos (FP)
- Reference F, Prediction F -> Verdaderos Positivos (TP)

Además, en la matriz de confusión nos encontraremos con los siguientes conceptos:

- Especificidad: representa la probabilidad de que un sujeto vivo tenga un resultado negativo en la prueba. Detecta los que verdaderamente están vivos.
- **Sensibilidad**: representa la probabilidad de que un sujeto fallecido tenga un resultado positivo en la prueba. Detecta los que verdaderamente han fallecido.
- **Prevalencia**: muestra con qué frecuencia ocurre realmente la condición positiva en nuestra muestra. Por ejemplo: si tenemos 100 pacientes y 85 viven, la prevalencia es del 15%. Teniendo en cuenta que la condición positiva es que fallezca.
- El **valor predictivo positivo** (VPP): muestra qué probabilidades hay de que una persona que da positivo (fallecido) haya realmente fallecido (verdadero positivo). En el caso en que las clases estén perfectamente equilibradas (lo que significa que la prevalencia es del 50%), el valor predictivo positivo (VPP) es equivalente a la precisión. Es la proporción de verdaderos positivos dentro del total de resultados positivos.

- El **valor predictivo negativo (VPN)**: muestra qué probabilidades hay de que una persona que obtiene un resultado negativo (vivo) realmente este vivo (verdadero negativo). Es la proporción de verdaderos negativos dentro del total de resultados negativos.

Teniendo los valores de sensibilidad y especificidad, deberemos saber que la aceptabilidad de una prueba diagnóstica dependerá de la condición clínica que más interese (en función de la enfermedad y de la población). Lo ideal sería obtener pruebas altamente sensibles y altamente específicas, pero es raro que estos casos se den.

Se preferirán las pruebas altamente sensibles en caso donde interese captar a toda la población enferma, aunque signifique un incremento en los falsos positivos.

- Un resultado falso positivo no tiene serios trastornos psicológicos o económicos en el paciente

Se preferirán una prueba altamente especifica en los casos en los que no se puede permitir una tasa alta de falsos positivos.

- Cuando un resultado falso positivo trae consecuencias psicológicas o económicas importantes al paciente.

2.4.2 MÉTRICAS DE EVALUACIÓN DE MODELOS

2.4.2.1 En modelos de regresión

RMSE y R^2 son las métricas predeterminadas que se usan para evaluar algoritmos de regresión.

- **R^2** (conocido como coeficiente de determinación): Proporciona una medida de bondad de las predicciones de las observaciones. Es un valor entre 0 y 1 para el estadístico T.

- Root mean Square error (**RMSE**): Es la desviación promedio de las predicciones de las observaciones. Es útil tener una idea general de lo óptimo (qué tan bien) que está funcionando un algoritmo.

2.4.2.2 En modelos de clasificación

Las métricas predeterminadas que se utilizan para evaluar los algoritmos de los conjuntos de datos de clasificación y multiclase son:

- **Overall Accuracy (precisión):** Es el porcentaje de instancias correctamente clasificadas de las instancias totales. Esta medida es más útil en una clasificación binaria que en un problema de clasificación multiclase puesto que es más complejo interpretar los resultados en este tipo de clases.
- **Kappa:** Es parecido a la precisión anteriormente comentada a excepción que está normalizada en base a una probabilidad aleatoria en el conjunto de datos. Es una medida útil para problemas que tienen un desequilibrio en las clases (por ejemplo, una división de 70% y 30% para las clases 0 y 1).

2.4.2.3 En modelos de clasificación binaria

Las mediciones de ROC son adecuadas para problemas de clasificación binarios (por ejemplo, dos clases). Para obtener la información de la curva ROC, debe cambiarse el *summaryFunction* en su *trainControl* para ser *twoClassSummary*. Esto calculará el área bajo la curva ROC (AUROC) también llamada área bajo curva (AUC), la sensibilidad y la especificidad.

El AUC representa una habilidad de los modelos para discriminar entre clases positivas y negativas. Un área de 1.0 representa un modelo que predice perfectamente. Un área de 0.5 representa un modelo tan bueno como aleatorio.

- **Sensibilidad**: nos indica la capacidad de nuestro estimador para dar como casos positivos los casos realmente positivos. Proporción de fallecidos correctamente identificados. Si predice que fallece, entonces que realmente fallezca.

- **Especificidad**: nos indica la capacidad de nuestro estimador para dar como casos negativos los casos realmente negativos (que los usuarios estén vivos); proporción de vivos correctamente identificados.

Recordemos que los ejes del gráfico de la curva ROC adoptan valores entre 0 y 1 (0% y 100%), delimitando un cuadrado de área = 1,00. Un test diagnóstico se considera no-discriminativo si su curva ROC coincide con la línea de no-discriminación, la cual posee un AUC = 0,50 (la línea de no-discriminación divide en dos mitades iguales el cuadrado de área = 1,00). A medida que el AUC de un test diagnóstico se acerca al valor 1,00 (test diagnóstico perfecto), mayor será su capacidad discriminativa.

2.4.2.4 Sobreajuste (overfitting)

El "*overfitting*", también llamado sobreajuste: es el efecto que se da al entrenar de más un algoritmo de aprendizaje, de este modo el algoritmo queda muy ajustado a características muy específicas y, por lo tanto, su respuesta a nuevos datos empeora.

Existen distintas técnicas para evitar el sobreajuste:

- **Cross-Validation:** esta técnica consiste en dividir los datos en varios conjuntos de datos y luego elegir uno de los conjuntos para medir la precisión de la predicción "test" y el resto para entrenar. Concretamente se divide la muestra en K sub-muestras, de forma que se utilizan K-1 para estimar el modelo y la restante como sub-muestra de evaluación, este proceso se repite K veces, de forma que cada sub-muestra es utilizada una vez para evaluar el modelo y K-1 veces para el ajuste.
- **Detención temprana:** proporciona información sobre cuántas iteraciones se pueden ejecutar antes de que el algoritmo de aprendizaje comience a sobrepasar el límite.
- **Poda:** Simplemente elimina los nodos que agregan poca capacidad de predicción para el problema en cuestión.

- **Regularización:** introduce un término de costo a la hora de obtener más variables con la función objetivo. Intenta reducir a cero los coeficientes de muchas variables consiguiendo así reducir el término de costo.

2.5 MOTIVACION EN EL USO DE R

El principal motivo para el uso de R ha sido que se trata de un lenguaje y posee un entorno bajo la licencia GNU GPL además de una gran cantidad de paquetes.

Podría haberse utilizado Python que también es un lenguaje bastante potente para realizar el análisis de datos, sin embargo, se ha intentado realizar un estudio simulando al previo realizado en el artículo original ("*Predictors of Outcome in Traumatic Brain Injury: New Insight Using Receiver Operating Curve Indices and Bayesian Network Analysis*"), en el que también se utiliza el lenguaje R para la obtención de evidencias.

Por otro lado, se ha decidido usar R debido a que la curva de aprendizaje es menor ya que se tenían conocimientos previos en el uso de esta herramienta.

El uso de paquetes dentro de R también se ha seleccionado dependiendo de las necesidades y de las funcionalidades y restricciones que nos aportaban.

Caret nos ha condicionado bastante en el uso de paquetes puesto que soporta ciertos modelos, que no son pocos.

2.5.1 CARET

El paquete caret *("classification and regression training")* es un framework que incluye un gran número de funciones que facilitan el uso de decenas de métodos complejos de clasificación y regresión. El uso de este paquete en lugar de usar las funciones originales presenta dos ventajas:

- Permite el uso de un código unificado para aplicar reglas de clasificación distintas, implementadas en distintos paquetes.
- Facilita el uso de algunos procedimientos usuales en problemas de clasificación.
    - Caret posee funciones específicas para dividir la muestra en datos de entrenamiento y datos de test o para ajustar parámetros mediante la validación cruzada.

En líneas generales, caret proporciona herramientas para:

- División de datos.
- Pre-procesado.
- Selección de características.
- Ajuste de modelo usando re-muestreo.
- Estimación de la importancia de variables.

Por otro lado, caret nos proporciona un modelo de rejillas (grids) de búsqueda. Estas rejillas se plantean cuando se necesita realizar una optimización de los parámetros de un modelo determinado.

Cuando un determinado algoritmo depende de, por ejemplo, 4 parámetros, se puede definir una rejilla.

Por su parte caret se encarga de ajustar el modelo bajo todas las combinaciones posibles de parámetros para ver cuál de estas combinaciones es la óptima.

En este documento se utilizará esta técnica de búsqueda en la mayoría de los modelos. Obviamente, esta técnica es bastante costosa ya que al fin y al cabo se obtiene el modelo optimo mediante “fuerza bruta”.

2.5.2 CARETENSEMBLE

CaretEnsemble es un paquete para hacer mezclas de modelos de caret. Posee 3 funciones principales: caretList, caretEnsemble y caretStack.

- **caretList** se usa para crear listas de modelos de caret con los mismos datos de entrenamiento y los mismos parámetros de muestreo. caretEnsemble y caretStack se utilizan para crear mezclas de modelos a partir de dichas listas de modelos de caret.
- **caretEnsemble** utiliza glm para crear una combinación lineal simple de modelos. Obviamente es similar al caretStack cuando este usa glm como modelo de orden superior.
- **caretStack** usa un modelo de intercalación para combinar las salidas de varios modelos de caret, es decir se utiliza un modelo de orden superior para combinarse con las predicciones de los sub-modelos.

3. DESARROLLO

Como ya se comentó en la sección 2.2. En este capítulo se desarrollarán dichas fases.

Las fases que se persiguen alcanzar con este trabajo fin de master, se van a dividir en tres secciones en este documento:

1. **Preparación de los datos:** Abarca las fases de obtención de datos, procesado de datos y limpieza de datos.
2. **Pre-procesamiento de los datos:** Abarca la fase del análisis exploratorio de datos
3. **Modelado de los datos:** Abarca la fase de modelados y algoritmos.

3.1 PREPARACIÓN DE LOS DATOS

En esta sección nos encargaremos de obtener los datos y realizar su preparación.

El propósito fundamental de la preparación de datos es manipular y transformar datos para que la información contenida en el conjunto de datos pueda ser expuesta o más fácilmente accesible con el objetivo de poder realizar tomar decisiones.

Una vez que se han obtenido los datos, se realizara una clasificación lógica de ellos, de esta forma podremos controlar y visualizar los datos anómalos que se encuentren en el conjunto.

3.1.1 OBTENCIÓN DE LOS DATOS

El nombre del repositorio de datos es el siguiente: **CRASH Corticosteroid Randomisation after Significant Head Injury.**

Estos datos provienen de la siguiente institución: **London School of Hygiene and Tropical Medicine.**

Por último, el artículo de investigación que nos ha servido de referencia para la realización de este trabajo de fin de master ha sido el siguiente: **“Predictors of Outcome in Traumatic**

Brain Injury: New Insight Using Receiver Operating Curve Indices and Bayesian Network Analysis" cuyos escritores son los siguientes: **Zsolt Zador, Matthew Sperrin y Andrew T. King.**

3.1.2 NOMENCLATURA DEL CONJUNTO DE DATOS

La pretensión de esta sección es que el lector entienda el significado de las variables que se van a utilizar en la preparación de los datos.

Las variables que se van a utilizar son las siguientes:

Nombre de la variable	**Significado**	**Valores**
SEX	Género del paciente	0 = Hombre 1 = Mujer
AGE	Edad del paciente	
GCS_EYE	Escala de Glasgow: Apertura de ojos	4 =Espontaneo 3 = Al sonido 2 = Al dolor 1 = Ninguno
GCS_MOTOR	Escala de Glasgow: Respuesta motriz	6 = Obedece ordenes 5 = Localiza el dolor 4 = Flexión normal 3 = Flexión anormal 2 = Extensión 1 = Ninguno
GCS_VERBAL	Escala de Glasgow: Respuesta verbal	5 = Orientado 4 = Desorientado y hablando 3 = Palabras 2 = Sonidos

		1 = Ninguno
PUPIL_REACT_LEFT	Reactividad de la pupila del ojo izquierdo.	1 = Si 2= No 3 = Incapaz de valuar
PUPIL_REACT_RIGHT	Reactividad de la pupila del ojo derecho.	1 = Si 2= No 3 = Incapaz de valuar
CAUSE	Causa del traumatismo	1 = Accidente de trafico 2 = Caída de más de 2 metros 3 = Otros
OUTCOME	Vivo o fallecido en las dos primeras semanas del traumatismo.	1 = Si 2 = No
SYMPTOMS	Condición del paciente en los primeros resultados	1 = Sin síntomas 2 = Síntomas menores 3 = Alguna restricción en el estilo de vida, pero independiente 4 = Dependiente, pero no requiere una constante atención. 5 = Completamente dependiente, requiere atención día y noche 6 = Fallecido

		9 = Se sabe que está vivo a los 6 meses pero no se sabe los síntomas en el día 14.
MAJOR_EC_INJURY	Lesión extra craneal mayor	1 = Si 2 = No
HEAD_CT_SCAN	Escáner CT realizado	1 = Si 2 = No
1_OR_MORE_PH	1 o más hemorragias petequias en la cabeza	1 = Si 2 = No
OBLITERATION_3RDVORBC	Obliteración del 3 ventrículo o deposito basal.	1 = Si 2 = No
SUBARACHNOID_BLEED	Hemorragia subaracnoidea	1 = Si 2 = No
MIDLINE_SHIFT>5MM	Cambio en la línea media del cerebro	1 = Si 2 = No
NON_EVAC_HAEM	Hematoma intracraneal evacuado	1 = Si 2 = No
EVAC_HAEM	Hematoma intracraneal	1 = Si 2 = No
GOS5	Evaluación general (Cuestionario 5 niveles)	GR = Buena recuperación MD = Discapacidad moderada SD = Discapacidad severa

		SD* = Discapacidad severa no relacionada con el traumatismo VS = Estado vegetativo D= Fallecido
GOS8	Evaluación general (Cuestionario 8 niveles)	GR- = Buena recuperación (baja) GR+ = Buena recuperación (alta) MD- = Discapacidad moderada (baja) MD+ = Discapacidad moderada (alta) SD- = Discapacidad severa (baja) SD+ = Discapacidad severa (alta) SD* = Discapacidad severa no relacionada con el traumatismo D= Fallecido

Tabla 1. Nomenclatura del conjunto de datos

Las **variables con prefijo "EO_"** son aquellas que pertenecen a las pruebas realizadas en las primeras dos semanas de ingreso en el hospital.

Las **variables con prefijo "TH_"** son aquellas que pertenecen a las pruebas realizadas en las primeras dos semanas si se ha transferido el paciente a otro hospital.

3.1.3 CLASIFICACIÓN ENTRE: ALIVE, DEATH, NO-DATA Y MD/GR

En esta fase, se ha realizado una clasificación de los datos dados según 4 resultados finales:

- Fallecidos o con discapacidades severas (SD-D).
- Con discapacidad moderada o buena recuperación (MR-GR).
- Vivos (pero sin resultados finales).
- Sin datos.

Al principio, se ha creado una columna extra llamada "outcome" vacía, que contiene un valor inicial de "NA". Por tanto, en un principio todos los datos iniciales pertenecen a la clasificación de "Sin datos". Posteriormente esta columna se modificará según los valores del resto de las columnas del conjunto de datos.

Para este procesado se han tenido en cuenta principalmente las siguientes variables:

- EO_Outcome.
- EO_Symptoms.
- TH_Outcome.
- TH_Symptoms.
- GOS5.
- GOS8.

3.1.3.1 Tratamiento cuando las columnas GOS5 y GOS8 contienen valores

El primer tratamiento que hemos realizado ha sido observar si las filas ya contenían datos en las columnas de GOS5 y GOS8. Si es así, directamente se han clasificado -según estas variables-. De lo contrario, se ha tenido que realizar un análisis del resto de variables.

```
head(datos.modelo[,c(17,18,27,28,29,30)])

##   EO_Outcome EO_Symptoms TH_Outcome TH_Symptoms GOS5 GOS8
## 1          4           1         NA          NA <NA> <NA>
## 2          4           3         NA          NA <NA>  MD+
## 3          4           2         NA          NA  SD* <NA>
## 4          4           2         NA          NA <NA>  GR+
## 5          4           1         NA          NA <NA> <NA>
## 6          4           2         NA          NA <NA>  SD-
```

3.1.3.2 Tratamiento cuando las columnas GOS5 y GOS8 se encuentran vacías

Una vez que ya hemos tratado las filas cuyas columnas de GOS5 y GOS8 contenían datos, ahora vamos a estudiar el caso contrario.

Para ello vamos a tener en cuenta los valores de otras columnas, que nos permitan obtener alguna información sobre el estado del paciente en los siguientes meses.

Si las columnas de *"outcome" (EO y TH)* contienen el valor de 1 (fallecimiento) o las columnas de *"Symptoms"* contenían el valor de 6, directamente esas filas del conjunto de se han clasificado como fallecidos, cambiando el valor de la columna "outcome" que hemos creado.

```
##        EO_Outcome  EO_Symptoms  TH_Outcome  TH_Symptoms  GOS5  GOS8  TH_Cause
## 22            1            6          NA           NA <NA> <NA>         NA
## 38            1            6          NA           NA <NA> <NA>         NA
## 50            1            6          NA           NA <NA> <NA>         NA
## 55            1            6          NA           NA <NA> <NA>         NA
## 61            1            6          NA           NA <NA> <NA>         NA
## 85            1            6          NA           NA <NA> <NA>         NA
```

Si las columnas de *"outcome" (EO y TH)* contenían el valor de 4 (alta) y las de "*Symptoms"* el valor de 1, entonces se han clasificado las respectivas filas como "Vivos (pero sin resultados finales)", cambiando el valor de la columna "outcome" que hemos creado.

```
##        EO_Outcome  EO_Symptoms  TH_Outcome  TH_Symptoms  GOS5  GOS8  TH_Cause
## 1             4            1          NA           NA <NA> <NA>         NA
## 5             4            1          NA           NA <NA> <NA>         NA
## 18            4            1          NA           NA <NA> <NA>         NA
## 20            4            1          NA           NA <NA> <NA>         NA
```

```
## 36            4             1           NA            NA <NA> <NA>         NA
## 51            4             1           NA            NA  <NA> <NA>         NA
```

Se clasificarán como "Sin datos" todas aquellas filas que no contengan valores en las columnas de "*Symptoms"*. En este tratamiento se tienen en cuenta los transferidos a otros hospitales.

```
##          EO_Outcome  EO_Symptoms  TH_Outcome  TH_Symptoms  GOS5  GOS8  TH_Cause
## 384            NA            NA           NA            NA <NA> <NA>         NA
## 417            NA            NA           NA            NA <NA> <NA>         NA
## 985            NA            NA           NA            NA <NA> <NA>         NA
## 997            NA            NA           NA            NA <NA> <NA>         NA
## 2270           NA            NA           NA            NA <NA> <NA>         NA
## 2292           NA            NA           NA            NA <NA> <NA>         NA
```

Si las columnas de "*Symptoms*" contienen valores de 4 o de 5 (Discapacidad Severa), entonces las correspondientes filas se clasificarán como "Fallecidos o con discapacidades severas", cambiando el valor de la columna "outcome" que hemos creado.

```
##     EO_Outcome EO_Symptoms TH_Outcome TH_Symptoms GOS5 GOS8 TH_Cause
## 160          5           5         NA           NA <NA> <NA>       NA
## 241          5           5         NA           NA <NA> <NA>       NA
## 317          5           5         NA           NA <NA> <NA>       NA
## 336          5           5         NA           NA <NA> <NA>       NA
## 357          5           5         NA           NA <NA> <NA>       NA
## 573          5           5         NA           NA <NA> <NA>       NA
```

Así mismo, si las columnas de *"outcome" (EO y TH)* contenían el valor de 4 y las de *"Symptoms"* el valor de 9, significa que el paciente ha sido dado de alta, pero no se tiene ningún dato sobre el estado final, por lo tanto, dichas filas se han incluido en la clasificación de "Vivos (pero sin resultados finales)". Cambiando el valor de la columna "outcome" que hemos creado.

```
##      EO_Outcome EO_Symptoms TH_Outcome TH_Symptoms GOS5 GOS8 TH_Cause
## 409          4           9         NA           NA <NA> <NA>       NA
## 1000         4           9         NA           NA <NA> <NA>       NA
## 4859         4           9         NA           NA <NA> <NA>       NA
```

Además, se han observado 3 elementos de la clasificación de "Sin datos", cuyos pacientes obtienen un valor en *"Symptoms"* de 4, por lo que se clasifica a un estado de fallecido. Estos datos son datos anómalos.

```
##      EO_Outcome EO_Symptoms TH_Outcome TH_Symptoms GOS5 GOS8 TH_Cause
## 52            2           4         NA           4 <NA> <NA>        3
## 699           2           4         NA           4 <NA> <NA>       NA
## 3025          2           4         NA           4 <NA> <NA>        1
```

3.1.3.3 Datos finales

A continuación, se muestra la clasificación realizada con el número de filas en cada clasificador.

- Fallecidos o con discapacidades severas*:* 3559
- Con discapacidad moderada o buena recuperación*:* 5997
- Vivos (pero sin resultados finales)*:* 127
- Sin datos*:* 86
- Con NA*:* 239

3.1.4 CLASIFICACIÓN ENTRE: ESCANEADOS Y NO ESCANEADOS

Esta clasificación se ha realizado debido a que se han encontrado una gran cantidad de datos anómalos. Debemos tener en cuenta que casi todas las columnas que vamos a tratar son resultados de escáneres. Por tanto, se ha utilizado esta clasificación para poder filtrar y sanar los datos.

En un futuro deberemos tener en cuenta si el escáner se ha hecho en el hospital inicial o al que se ha transferido posteriormente al paciente, por tanto, tener un control y una limpieza de los datos respecto a esta clasificación es fundamental.

En primer lugar, se han encontrado ciertos datos anómalos, en los que aparecen datos escaneados (con valor de 1) y no tienen los datos del escáner, entonces deberíamos ponerlos como no escaneado (cambiando su valor a 2).

```
##      EO_Head.CT.scan EO_1.or.more.PH EO_Subarachnoid.bleed
## 201                1              NA                    NA
## 314                1              NA                    NA
## 1277               1              NA                    NA
## 3234               1              NA                    NA
## 3687               1              NA                    NA
## 4256               1              NA                    NA
##      EO_Obliteration.3rdVorBC EO_Midline.shift..5mm EO_Non.evac.haem
## 201                        NA                    NA               NA
## 314                        NA                    NA               NA
## 1277                       NA                    NA               NA
## 3234                       NA                    NA               NA
## 3687                       NA                    NA               NA
## 4256                       NA                    NA               NA
##      EO_Evac.haem
## 201            NA
## 314            NA
## 1277           NA
## 3234           NA
## 3687           NA
## 4256           NA
```

A continuación, se van a clasificar los datos como:

- Escaneados: Serán aquellos datos que tengan valores en todas las variables de escáner, que son: "Head.CT.scan", "1.or.more.PH","Subarachnoid.bleed", "Obliteration.3rdVorBC"," Midline.shift..5mm"," Non.evac.haem" y " Evac.haem"
- No escaneados: Serán aquellos datos que no tengan valores en las variables anteriores o que directamente no se hayan hecho el escáner.
- En análisis: Serán aquellas filas que tengan datos anómalos y sean necesarios estudiarlos por separado.

Lo que vamos a realizar con esta clasificación es, a partir de los datos iniciales, separar dichos datos en 3 conjuntos de datos ("*dataset*") que correspondan con la clasificación establecida.

El primer tratamiento que realizaremos es clasificar las filas que contengan falta de información. Concretamente, si la columna de *"Outcome"* contiene un valor de 2 (el paciente se ha transferido a otro hospital), significa que el paciente se ha escaneado en otro hospital

("TH_SCAN") y no se posee ninguna información sobre los escáneres. Esos datos se clasificarán como "En análisis".

```
##     EO_Outcome TH_Head.CT.scan TH_1.or.more.PH TH_Subarachnoid.bleed
## 52           2            <NA>              NA                    NA
## 128          2            <NA>              NA                    NA
## 135          2            <NA>              NA                    NA
## 188          2            <NA>              NA                    NA
## 193          2            <NA>              NA                    NA
## 207          2            <NA>              NA                    NA
```

Sobre el dataset "**NO ESCANEADO"**: Si la columna de *"Outcome"* es 2 (el paciente se ha transferido a otro hospital) y no se ha realizado ningún escáner ("Head.CT.scan" es 1), pero si contiene datos en el escáner, entonces estas filas se clasificarán como "Escaneado".

```
##           EO_Outcome  TH_Head.CT.scan  TH_1.or.more.PH  TH_Subarachnoid.bleed
## 201            2                 1                2                      2
## 217            2                 1                2                      1
## 257            2                 1                1                      2
## 314            2                 1                1                      2
## 318            2                 1                2                      2
## 1184           2                 1                2                      2
```

En el dataset "**NO ESCANEADO"**, nos hemos dado cuenta que existen datos anómalos, que contienen varios escáneres, pero, sin embargo, no se indica como escaneado, son los registros: 2628,3276,3279,8469,8655, etc. (En total son 12). Estos datos se clasificarán como escaneados.

```
##      EO_Head.CT.scan EO_1.or.more.PH EO_Subarachnoid.bleed
## 2628               2               2                     2
## 3276               2               2                     2
## 3279               2               2                     2
## 3720               2               2                     2
## 7286               2               2                     2
## 8469               2               2                     2
```

En el dataset "**EN ANALISIS",** nos hemos dado cuenta que existen datos anómalos. Para las variables de los pacientes que se han transferido a otro hospital (TH), existen variables de escáner *("TH_Head.CT.scan")* que se encuentran vacías, junto con el resto de variables del escáner en particular. Por lo tanto, se ha asignado el valor de 2 a la variable de escáner

("TH_Head.CT.scan") y se han incluido en los escaneados, puesto que en todos ellos, en la variable *"EO_Head.CT.scan"* sí que existe un valor de 1 (escaneados) y no se han encontrado más anomalías en dichos datos.

```
##      EO_Head.CT.scan EO_1.or.more.PH EO_Outcome TH_Head.CT.scan
## 681                1               2          2            <NA>
## 1639               1               2          2            <NA>
## 5743               1               2          2            <NA>
## 8434               1               2          2            <NA>
## 8972               1               2          2            <NA>
##      TH_1.or.more.PH TH_Subarachnoid.bleed
## 681               NA                    NA
## 1639              NA                    NA
## 5743              NA                    NA
## 8434              NA                    NA
## 8972              NA                    NA
```

Sobre el dataset "**ESCANEADO"**: Se van a eliminar todas las filas que no tengan información en el *"TH_Major.EC.injury"* y en el *"EO_Major.EC.injury". Estas filas se añadirán al dataset de "**EN ANÁLISIS"**.*

```
##     EO_Outcome TH_Major.EC.injury
## 76           2                 NA
## 90           2                 NA
## 315          2                 NA
## 361          2                 NA
## 510          2                 NA
## 565          2                 NA
```

Sobre el dataset "**ESCANEADO"**: Comprobamos que las variables: *"EO_Cause"* y *"EO_Symptoms"*, no contengan valores nulos. Si contienen valores nulos, entonces se cambiarán al dataset **"EN ANALISIS".**

```
##     EO_Cause EO_Major.EC.injury
## 177        3                 2
## 211       NA                 1
## 242       NA                 2
## 255        2                 2
## 293       NA                 2
## 321       NA                 1
```

Sobre el dataset "**ESCANEADO"**: Comprobamos que la variable: *"EO_Outcome"* no se encuentre nula. (En total son 2 registros). Si se encuentra nula, se cambiará al dataset "**EN ANÁLISIS**".

```
##      EO_Cause EO_Outcome
## 9036        3         NA
## 9333        3         NA
```

Sobre el dataset "**ESCANEADO"**: Comprobamos que existe un valor anómalo (que se sale del rango de valores que puede tomar la columna) en un registro en la columna de *"EO_Major.EC.Injury"*. Este valor lo cambiaremos a positivo -> 1.

```
##      EO_Cause EO_Major.EC.injury
## 3862        2                -1
```

Datos finales:

- Vivos y escaneados: 4157
- Vivos y no escaneados: 1535
- Vivos en análisis: 305
- Fallecidos y escaneados: 2829
- Fallecidos y no escaneados: 439
- Fallecidos en análisis: 291

3.1.5 ELIMINACIÓN Y UNIFICACIÓN DE VARIABLES

En esta sección nos encargaremos de unir columnas que sean complementarias. En primer lugar, vamos a unir las variables de reacción de las pupilas, que en el conjunto de datos se dividen en dos columnas: reactividad de la pupila derecha y de la pupila izquierda y en el artículo inicial únicamente se utiliza una columna genérica que es reactividad de las pupilas (teniendo en cuenta ambas pupilas).

Por tanto, se van a unir las columnas de *"PUPIL_REACT_LEFT"* y *"PUPIL_REACT_RIGHT".*

Si las dos columnas (PUPIL_REACT_RIGHT y PUPIL_REACT_LEFT) poseen los siguientes valores: 1 y 3; entonces en la columna de unión ("*pupil*") asignara para dichas filas un valor de 1 ("*both reactive*").

Si las dos columnas (PUPIL_REACT_RIGHT y PUPIL_REACT_LEFT) poseen los siguientes valores: (1 y 2) o (3 y 2); entonces en la columna de unión ("*pupil*") asignara para dichas filas un valor de 2 ("*no response unilateral*").

Si las dos columnas (PUPIL_REACT_RIGHT y PUPIL_REACT_LEFT) poseen los siguientes valores: 2 y 2; entonces en la columna de unión ("*pupil*") asignara para dichas filas un valor de 3 ("*no response*").

Si las dos columnas (PUPIL_REACT_RIGHT y PUPIL_REACT_LEFT) poseen los siguientes valores: 3 y 3; entonces en la columna de unión ("*pupil*") asignara para dichas filas un valor de 4 ("*unable to asseses*").

```
##                   PUPIL_REACT_LEFT      PUPIL_REACT_RIGHT       ESTADOESCANER
##  1                              1                      1           SCANEADO
##  2                              1                      1           SCANEADO
##  3                              1                      1           SCANEADO
##  4                              1                      1           SCANEADO
##  5                              1                      1           SCANEADO
##  6                              1                      1           SCANEADO
```

La clasificación y el número de filas de la variable final de pupilas será la siguiente:

- Both reactive: 5662
- No response unilateral: 497
- No response: 634
- Unable to assess: 193

Ahora vamos a unir las variables de *"EO_Cause"* y *"TH_Cause"*. Para ello veremos en qué caso, ambas variables difieren:

```
##      EO_Cause TH_Cause
## 2307        1        3
## 2813        3        1
## 3285        2        3
## 4021        3        1
```

Como se puede observar, podríamos prescindir de la variable *"TH_Cause"*, puesto que recoge la misma información que *"EO_Cause"*.

A continuación, vamos a unir todas las variables del escáner. Si un paciente ha sido transferido a otro hospital y se han realizado los escáneres en dicho hospital, entonces, se mantendrán los últimos valores del escáner. En caso contrario, se usarán los resultados obtenidos en el primer escáner. De esta forma solamente nos quedaremos con los resultados más actuales de los escáneres.

Para tener un conjunto de datos similar al que se ha utilizado en el artículo de referencia, cambiaremos el nombre de las variables del conjunto de datos al nombre utilizado en dicho artículo.

Además, eliminaremos todas las variables que no se utilicen en el artículo de referencia.

A continuación, se muestra un ejemplo del formato de los datos finales:

```
##    sex  age  cause  ec  eye  motor  verbal  pupils  phm  sah  oblt  mdls  hmt  outcome
## 1    0   11      1   1    1      5       1       1    2    2     2     2    2     MDGR
## 2    0   14      1   2    1      2       1       1    1    2     2     2    1        D
## 3    0   14      1   2    2      5       1       1    2    2     2     2    1        D
## 4    0   14      1   2    2      5       2       1    2    2     2     2    2     MDGR
## 5    0   14      3   2    4      6       4       1    2    1     2     2    2     MDGR
## 6    0   15      1   2    1      5       1       1    2    2     2     2    2        D
```

3.2 PRE-PROCESADO DE LOS DATOS

Una vez preparado los datos, es necesario realizar un análisis de este, usando estadísticos descriptivos y visualizando los resultados.

Debemos destacar que un futuro modelo de aprendizaje que utilicemos, será tan bueno como los datos que se utilicen.

3.2.1 BÚSQUEDA DE OUTLIERS (DATOS ANÓMALOS)

En primer lugar, se ha realizado un diagrama para cada una de las variables del conjunto de datos.

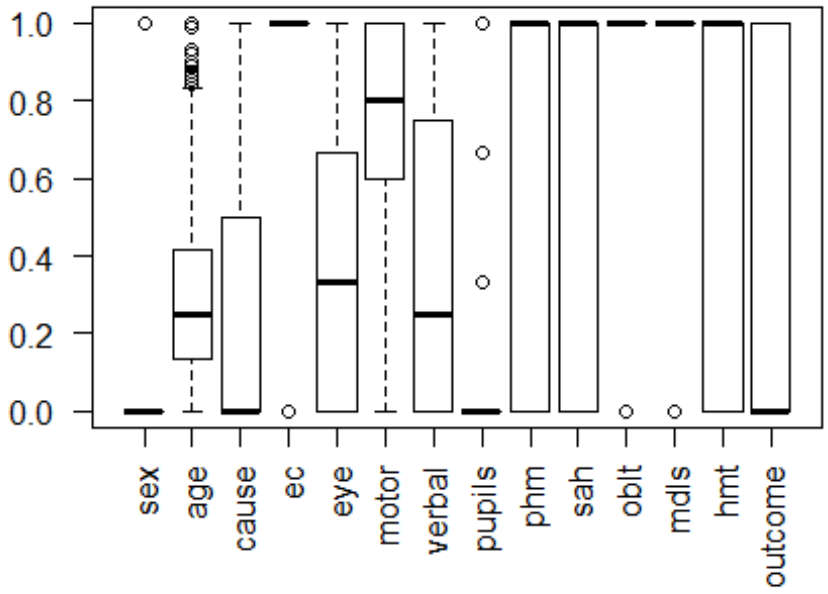


Ilustración 2. Diagrama de caja. Outliers

En este gráfico, nos hemos dado cuenta que la variable que contiene un gran numero datos anómalos es *"age"*.

A continuación, pasamos a estudiar a fondo los motivos que producen que esta variable tenga datos anómalos y comprobaremos si es necesario o no la eliminación de dichos datos.

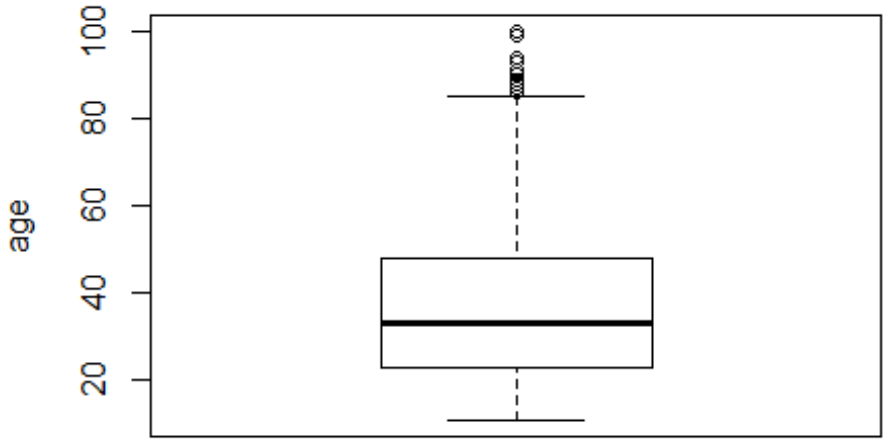


Ilustración 3. Diagrama de caja. Outliers. Edad

Estas son las edades anómalas:

```
##  [1]  86  86  86  86  86  86  87  87  87  87  88  88  88  88  89  89  90
## [18]  91  91  91  93  93  94  86  86  86  86  86  86  86  87  87  87  87
## [35]  87  87  87  88  88  88  88  89  89  89  89  89  89  89  90  90  91
## [52]  91  93  94  99 100
```

Antes de proceder a eliminar los datos anómalos, vamos a ver la correlación existente con la variable *"Outcome"*: 0.2598931

También vamos a observar como es la media y la mediana:

- Media: 37.02119
- Mediana: 33

A continuación, vamos a proceder con la eliminación relativa de los datos anómalos para ver cómo afecta al conjunto de los datos. Para ello hemos creado una función que nos elimine directamente los datos anómalos, es decir, todos aquellos datos que no se encuentren en el rango Q1-1.5·RIC o superiores a Q3+1.5·RIC se eliminaran. Siendo RIC el rango intercuartil (Q1-Q3)

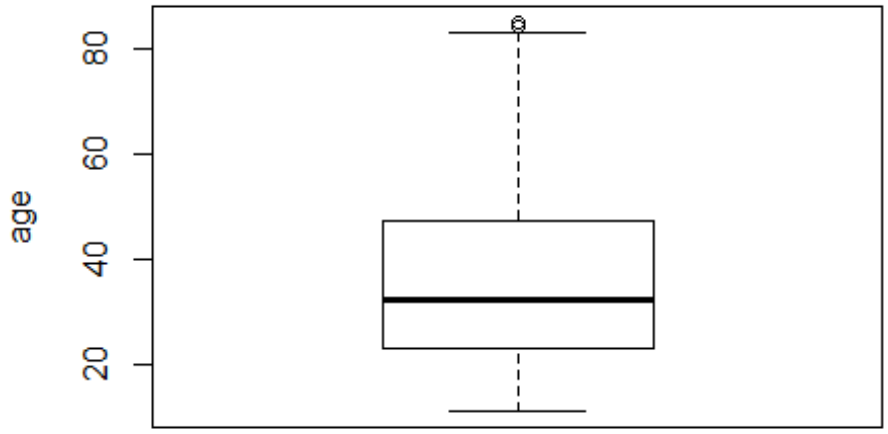


Ilustración 4. Diagrama de caja. Edad. Eliminando Outliers

La correlación obtenida posteriormente a la eliminación de los datos anómalos con la variable de *"Outcome"* es: 0.2452825. Vemos que la correlación ha empeorado un poco. De todas formas, la correlación entre la edad y el *"outcome"* es bastante débil.

También vamos a observar como es la media y la mediana:

- Media: 36.60303
- Mediana: 32

Como se puede comprobar, la eliminación de los "*outliers"* no ha afectado demasiado a las variables estadísticos por lo que no existe motivo para su eliminación.

Por otro lado, es necesario destacar que estos pacientes cuya edad es anómala (estadísticamente), en la naturaleza tampoco se consideran pacientes anómalos, ya que se encuentran en un rango de edades en las que sufrir un traumatismo craneocefálico es totalmente posible.

También vemos que existen "outliers" en la variable de pupils. La explicación que se dan a estos "outliers" es similar a la que se ha dado para la variable de "age". Como ya sabemos la variable de "pupils" contiene 4 tipos de valores: 0,1,2 y 3. La mayoría de las filas para esta

columna poseen un valor de 0 y esto hace que el resto de valores se salgan de RIC y se conviertan por tanto en valores anómalos (estadísticamente). Pese a que la gran mayoría de estos datos poseen valor de 0, existen muchos también que poseen valores distintos de 0 y no por eso se deben considerar anómalos. Esta misma problemática también la encontraremos en la variable “ec”, “oblt” y “mdls”.

3.2.2 ANÁLISIS DE NORMALIDAD

En primer lugar, visualizaremos la densidad de nuestras variables (individualmente), con el objetivo de observar a simple vista si cumplen o no con una distribución normal.

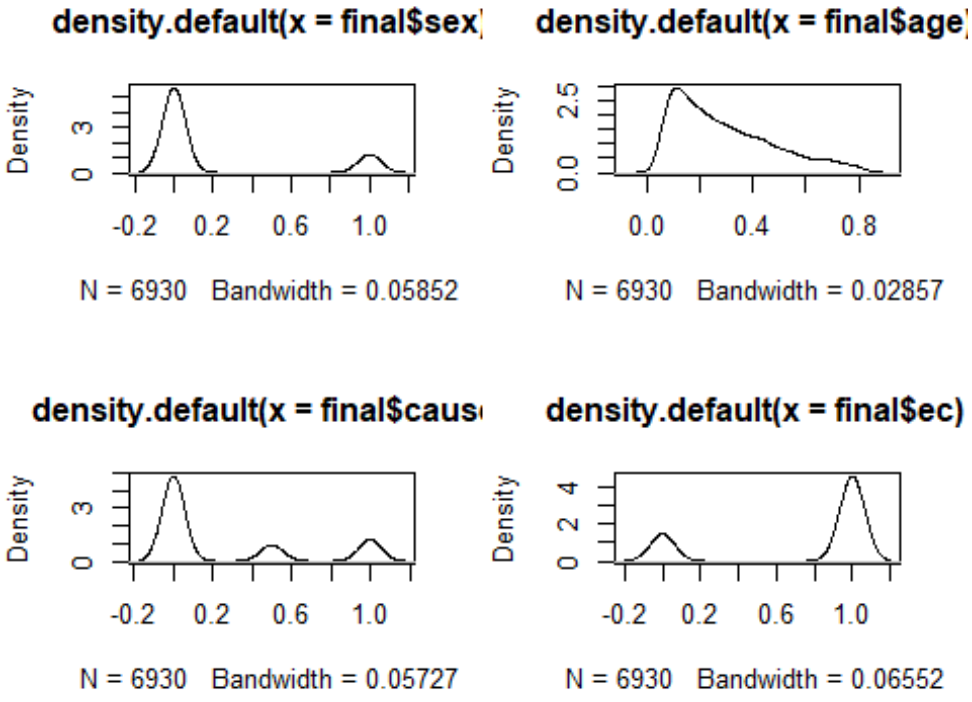


Ilustración 5. Gráfico Normalidad I

Como podemos comprobar, la tendencia a sufrir lesiones traumáticas es a aproximadamente entre los 15 y los 20 años. Después las lesiones decrecen hasta los 80 años. Se trata de una distribución normal asimétrica con un sesgo positivo hacia la derecha. Vemos también que

la tendencia a sufrir lesiones traumáticas es en varones. Y las causas más probables son los accidentes de tráfico.

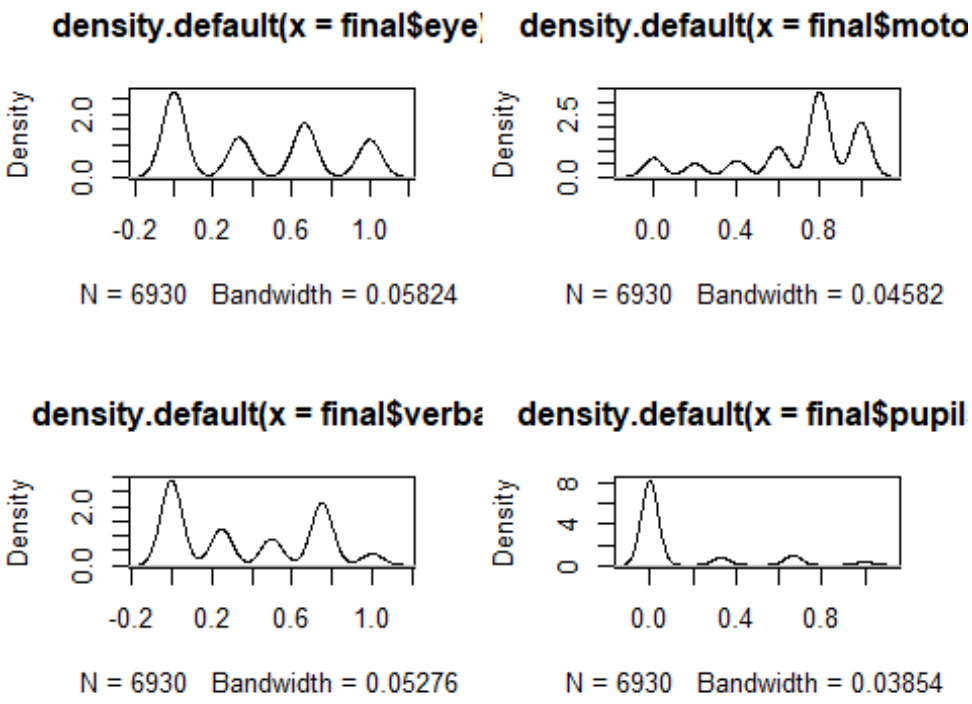


Ilustración 6. Gráfico Normalidad II

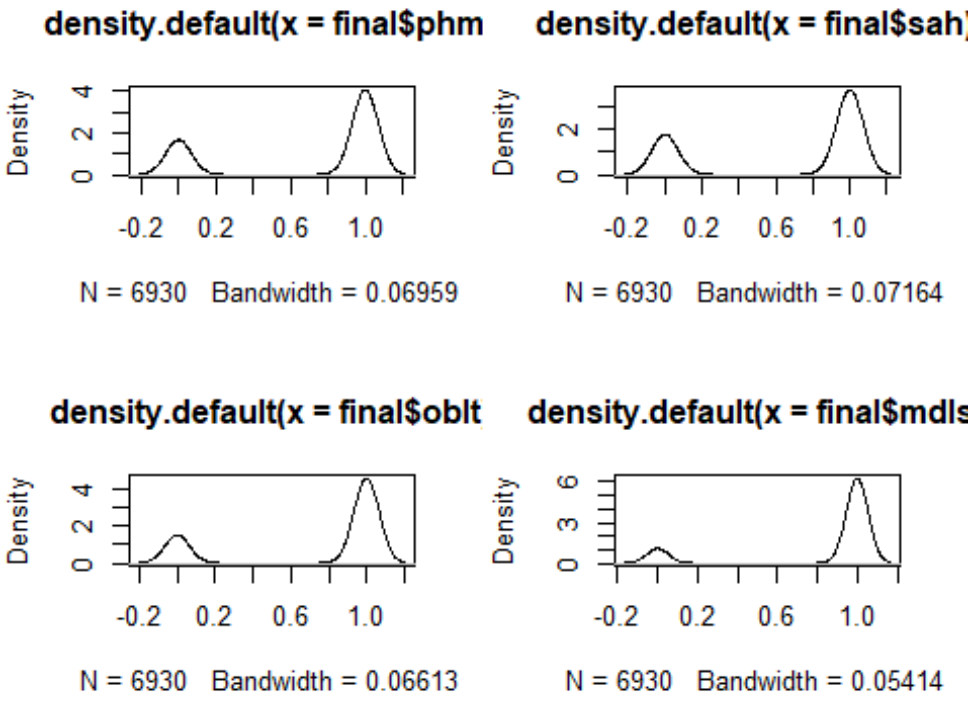


Ilustración 7. Gráfico Normalidad III

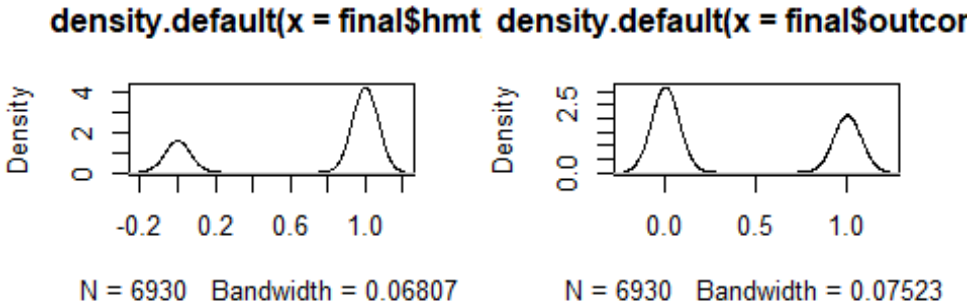


Ilustración 8. Gráfico Normalidad IV

Como podemos comprobar, al tratarse de variables discretas (excepto la variable de edad -*age*-, que es continua), no lograremos conseguir una distribución normal de forma individual.

Otro aspecto a tener en cuenta es que para que un conjunto de datos (teniendo en cuenta todas las variables) posea una distribución normal, es necesario que todas las variables verifiquen normalidad univariante, ya que es una condición necesaria (aunque no suficiente). Por lo tanto, rechazamos la hipótesis de normalidad del conjunto de datos.

Aun así, comprobaremos los resultados obtenidos mediante el Test de normalidad de Mardia:

```
## [1] 2
    ## Mardia's test for class 1
    ## mard1= 34629.84
    ## pvalue for m3= 0
    ## mard2= 74.78288
    ## p-value for m4= 0
    ## There is not statistical evidence for normality in class 1
    ## Mardia's test for class 2
    ## mard1= 6201.334
    ## pvalue for m3= 0
    ## mard2= -7.620724
    ## p-value for m4= 2.531308e-14
    ## There is not statistical evidence for normality in class 2
```

También vamos a utilizar el test de Henze-Zirkler:

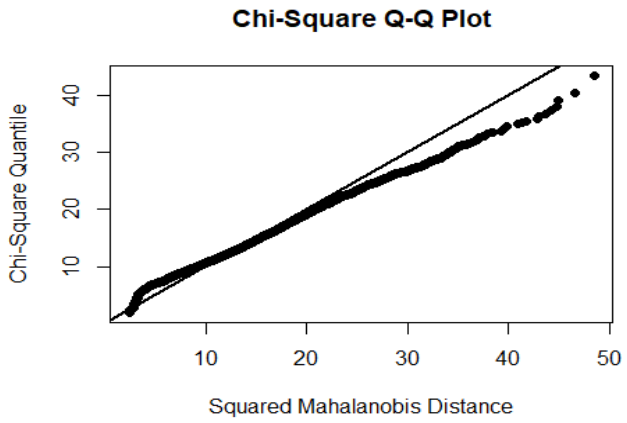


Ilustración 9. Chi-Square-Mahalanobis

Como se puede comprobar en la ilustración anterior, los datos no se ajustan a la línea recta teórica que mostraría que los datos si se ajustan a la normal.

```
##             Henze-Zirkler test for Multivariate Normality
##
##   data : final
##
##   HZ              : 15.38209
##   p-value         : 0
##
##   Result  : Data are not multivariate normal (sig.level = 0.05)
```

Como se puede comprobar, al ser el p-value menor de 0.05 en ambos test, los datos no se ajustan a una distribución normal.

Estos resultados no son positivos puesto que nuestro modelo no será tan eficiente como podría serlo si tuviéramos datos que sí que cumplieran con el supuesto de normalidad.

Tradicionalmente, la falta de normalidad afecta a las inferencias que se traten de hacer con el modelo, por ejemplo, los intervalos de confianza. La falta de normalidad no suele afectar a las predicciones puntuales, pero sí a los intervalos o "errores" en la predicción, llegando incluso a que los resultados puedan ser refutados.

Por otro lado, si quisiéramos utilizar la regresión como método de predicción, los datos requerirán el supuesto de normalidad. Por otro lado, existen modelos que son más sensibles a esta ausencia de normalidad, como por ejemplo el modelo de Naïves Bayes.

3.2.3 ESTUDIO DE CORRELACIÓN

Para el estudio de la correlación, utilizaremos el **coeficiente de correlación de Pearson (R)**. Mediante el siguiente gráfico, vamos a observar las relaciones que tienen los pares de variables entre sí.

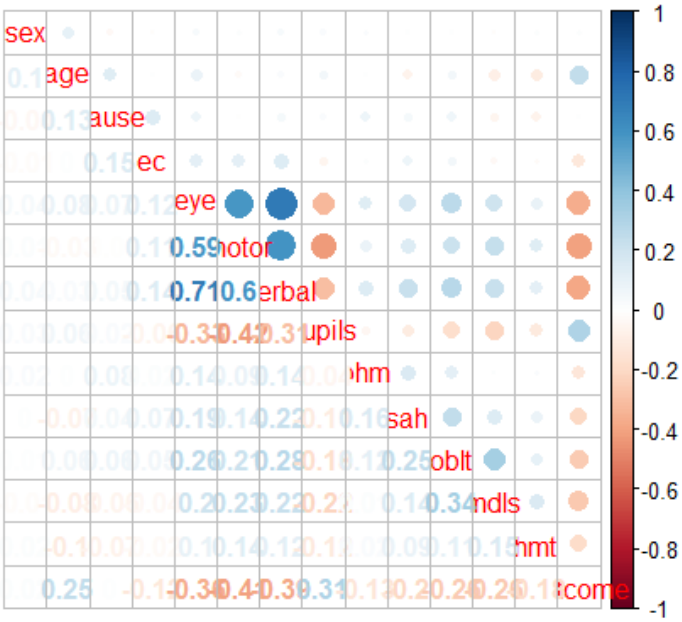


Ilustración 10. Matriz de correlación I

En este grafico podemos observar como por ejemplo las variables de *"motor"*, *"verbal"* y *"eye"* tienen bastante relación y dependencia entre sí. Sin embargo, hay algo que no nos cuadra y es que no existe una gran dependencia entre la variable "age" y la variable de "*outcome*", aspecto que podría ser más sustancial en la naturaleza.

Teniendo en cuenta los valores de la variable *"outcome"* (1 fallece y 0 vive), la correlación negativa de las variables del test de Glasgow ("*eye*", "*motor*", "*verbal*") tiene sentido, puesto que en general, cuanto mayor sea el valor de estas variables, mejor pronóstico de vida hay. La variable de "*pupils*" es, al contrario, cuanto mayores sean sus valores, más probable es el pronóstico de fallecimiento.

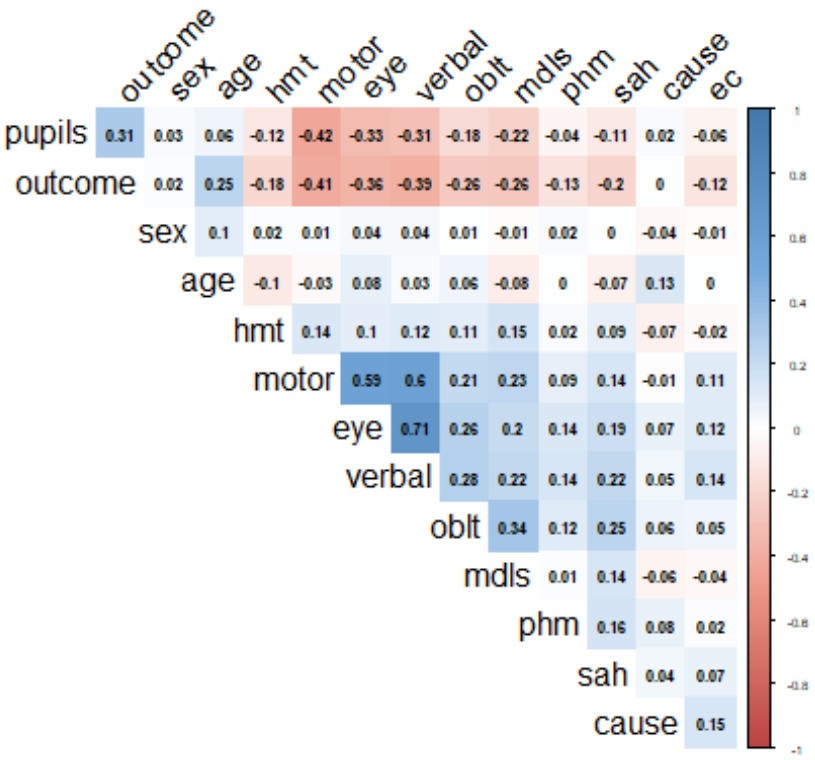


Ilustración 11. Matriz de correlación II

Como se ha podido apreciar en las dos graficas anteriores, existe una gran correlación entre las variables *"motor"*, *"eye"* y *"verbal"*.

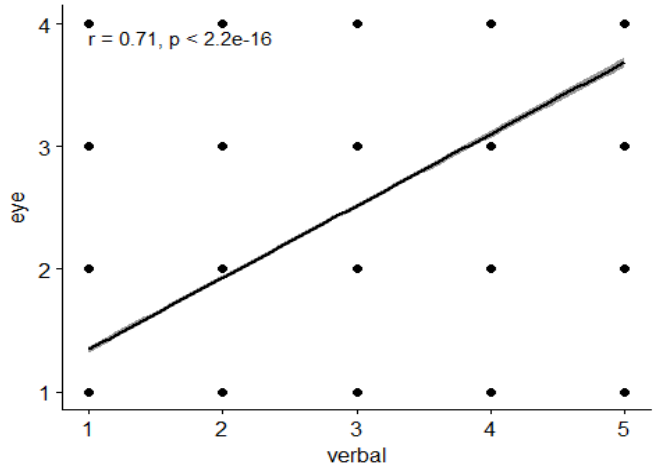


Ilustración 12. Relación lineal: verbal-eye

En el gráfico anterior podemos volver a comprobar que existe una gran relación lineal positiva entre las variables más correlacionadas que son "*verbal*" y "*eye*".

A continuación, se muestra la matriz de correlaciones parciales:

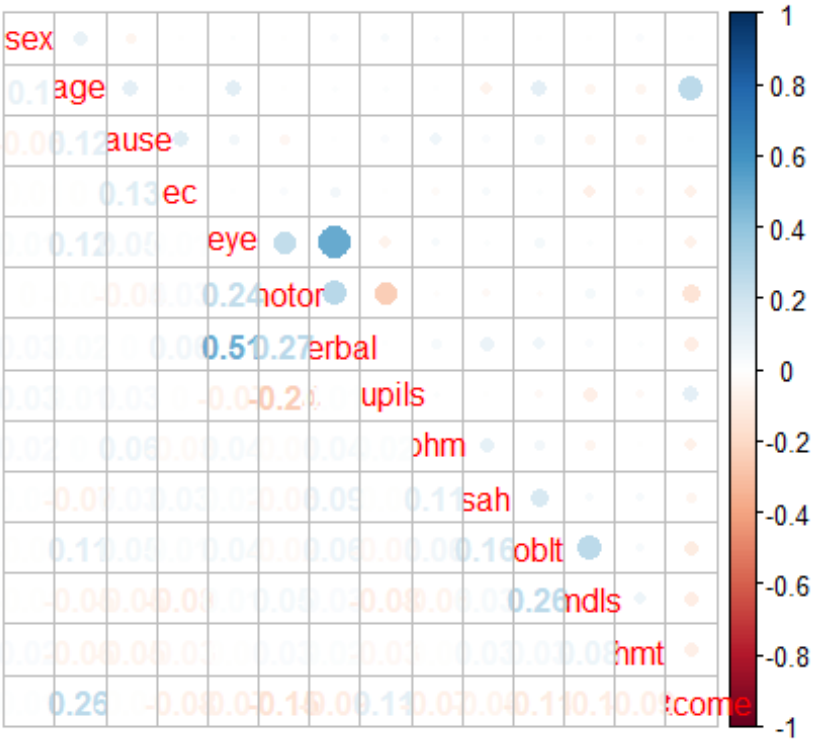


Ilustración 13. Matriz de correlación parcial

Con la matriz de correlaciones parciales, obtendremos las correlaciones parciales que existe entre los pares de variables eliminando el efecto de las restantes. Vemos que las correlaciones fuertes se encuentran entre los mismos pares de variables que en la matriz de correlación total.

A continuación, a modo de información, se muestra un listado en orden descendente con las mayores correlaciones existentes:

```
##     First.Variable Second.Variable Correlation
## 89             eye          verbal   0.7088056
## 90           motor          verbal   0.5989863
## 75             eye           motor   0.5863534
```

```
## 104          motor          pupils  -0.4224881
## 188          motor         outcome  -0.4098411
## 189         verbal         outcome  -0.3893489
## 187            eye         outcome  -0.3618281
## 165           oblt            mdls   0.3377311
## 103            eye          pupils  -0.3254853
## 190         pupils         outcome   0.3099949
## 105         verbal          pupils  -0.3073014
## 147         verbal            oblt   0.2765892
## 194           mdls         outcome  -0.2634093
## 145            eye            oblt   0.2609820
## 193           oblt         outcome  -0.2570853
```

Las correlaciones entre las variables y la clase ordenadas en orden descendente son las siguientes:

```
##     First.Variable Second.Variable Correlation
## 188          motor         outcome -0.40984112
## 189         verbal         outcome -0.38934889
## 187            eye         outcome -0.36182805
## 190         pupils         outcome  0.30999486
## 194           mdls         outcome -0.26340926
## 193           oblt         outcome -0.25708526
## 184            age         outcome  0.24528254
## 192            sah         outcome -0.20016386
## 195            hmt         outcome -0.18246423
## 191            phm         outcome -0.13254914
## 186             ec         outcome -0.12180868
## 183            sex         outcome  0.01988148
## 185          cause         outcome  0.00362152
## 196        outcome         outcome  0.00000000
```

Teniendo en cuenta los resultados anteriores, vemos que las variables de “cause” y “sex” no guardan apenas correlación con “outcome”, esto nos indica que estas variables pueden ser excluibles de nuestro análisis. En la naturaleza ocurre exactamente lo mismo, el sexo y las causas no son indicadores que vayan a afectar al estado del paciente en los próximos 6 meses después de sufrir un traumatismo. En estudios posteriores deberemos tener en cuenta estas variables y su posibilidad de exclusión.

Teniendo en cuenta todo lo anterior, consideramos que, aunque exista una correlación importante entre las variables “*eye*”, “*verbal*” y “*motor*”, no es lo suficientemente fuerte como para concluir que estas variables contienen la misma información y sea necesario la

eliminación de algunas de ellas. Por lo que no se procede a descartar ninguna de estas variables en estudios posteriores.

3.2.4 SELECCIÓN DE VARIABLES CON MAYOR IMPORTANCIA

En esta sección se utilizarán varios algoritmos para la selección de variables para la construcción de futuros modelos.

En muchas ocasiones se dispone de un gran conjunto de posibles variables explicativas, por lo que es necesario preguntarse si todas ellas deben utilizarse en los modelos a predecir, y en caso negativo, es necesario saber que variables deben utilizarse y cuáles no.

En general, existe el problema que, si se incluyen más variables en un modelo, el ajuste de los datos mejora, aumenta la cantidad de parámetros a estimar, pero disminuye la precisión individual (mayor varianza), y se produce un mayor sobreajuste. Por el contrario, si se menos variables de las necesarias en el modelo, las varianzas se reducen y se produciría, debido al sesgo, una peor descripción de los datos.

Por otro lado, algunas variables predictores podrían perjudicar la confiabilidad del modelo, concretamente si estas estuvieran correlacionadas con otras.

3.2.4.1 Uso de "Random Forest"

En este apartado, se buscará obtener un listado con las variables más importantes, usando el algoritmo de *"Random Forest"* para posteriormente tener en cuenta posibles descartes de variables en estudios posteriores.

Las variables más importantes utilizando el "mtry" son las siguientes:

```
##         X.IncMSE IncNodePurity
## sex     1.722602      43.18384
## cause   3.414805      67.70075
## phm    13.530135      49.85977
## hmt    18.625224      45.77766
## sah    23.916382      49.68159
## ec     27.739004      49.00917
## oblt   39.137262      50.54574
```

```
## mdls   39.767808      40.05219
## pupils 47.516489      76.13479
## eye    48.270418     101.46189
## verbal 66.382751     181.83672
## motor  70.147994     255.89314
## age    74.945217     426.70467
```

La variable **"IncNodePurity"** se la conoce también como la media de decrecimiento de de Gini. El índice de Gini es una "medida de desorden" en este caso *"IncNodePurity"* tiene el siguiente sentido, a mayor medida, mayor importancia en los modelos creados, puesto que valores próximos a 0 implican un mayor desorden. Por tanto, si computamos la media del "decrecimiento" del índice de Gini cuanto mayor sea esta medida, más variabilidad aporta a la variable dependiente.

Por otro lado, la variable **"IncMSE"** es la media de decrecimiento en la precisión, y es también un indicador sobre la importancia de las variables en el modelo.

El siguiente grafico representa la importancia de las variables según su media y los valores de *"Random Forest"* mostrados anteriormente:

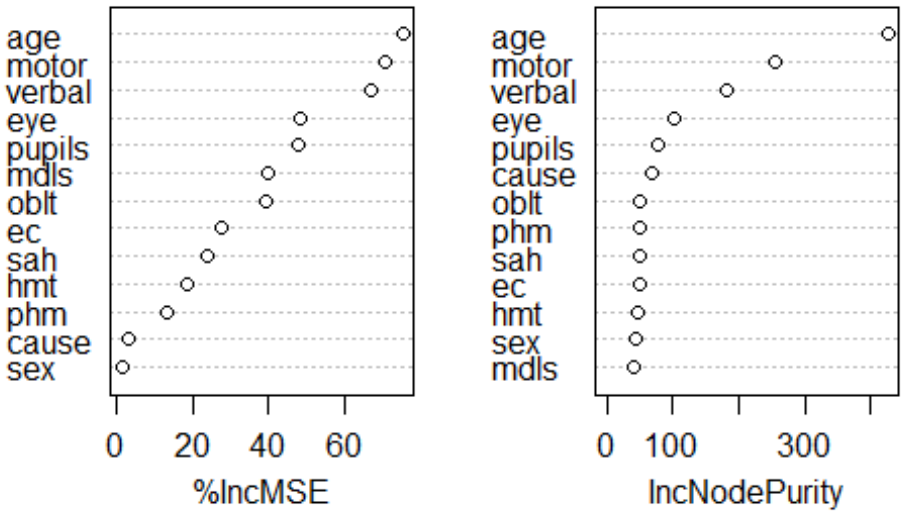


Ilustración 14. Random Forest. Importancia Variables

A continuación, se va a utilizar un árbol de clasificación, que nos mostrara la importancia de las variables según este algoritmo de clasificación.

Como conclusiones, utilizaremos las variables que se han considerado como más importantes en el algoritmo del árbol de clasificación y son las siguientes: *"motor"*, *"age"*, *"pupils"*, *"verbal"* y *"mdls"*.

3.2.4.2 Uso del método de clasificación paso a paso (Stepwise Fordward)

Este método es uno de los que se utilizan en la selección algorítmica del modelo. Se utiliza para identificar aquellas variables que se deberán integrar o no en los modelos a estudiar.

La lógica subyacente de este algoritmo consiste en conservar las variables independientes que contienen información relevante y a la vez prescindir de aquellas que resulten redundantes respecto de las que quedaron en el modelo.

Se ha realizado el estudio utilizando 2 funciones: la función de stepAIC, que compara los modelos utilizando el criterio de AIC y por otro lado utilizamos la función de greedy.Wilks, que utiliza el criterio lambda de Wilks.

Para utilizar la función de stepAIC, vamos primero a entrenar los datos mediante el modelo lineal generalizado.

```
##
## Call:
## glm(formula = outcome ~ ., family = binomial, data = final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7765  -0.7606  -0.3985   0.7762   2.6408
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.054157   0.310487  16.278  < 2e-16 ***
## sex          0.041738   0.080207   0.520    0.603
## age          0.041875   0.001976  21.187  < 2e-16 ***
## cause       -0.024701   0.040518  -0.610    0.542
```

```
## ec          -0.437802   0.069418  -6.307 2.85e-10 ***
## eye         -0.237643   0.038838  -6.119 9.43e-10 ***
## motor       -0.289910   0.025841 -11.219  < 2e-16 ***
## verbal      -0.237017   0.032541  -7.284 3.25e-13 ***
## pupils       0.378928   0.043887   8.634  < 2e-16 ***
## phm         -0.394038   0.065282  -6.036 1.58e-09 ***
## sah         -0.279699   0.065385  -4.278 1.89e-05 ***
## oblt        -0.638613   0.073955  -8.635  < 2e-16 ***
## mdls        -0.709317   0.091718  -7.734 1.04e-14 ***
## hmt         -0.488884   0.067500  -7.243 4.40e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9332.8  on 6929  degrees of freedom
## Residual deviance: 6751.1  on 6916  degrees of freedom
## AIC: 6779.1
##
## Number of Fisher Scoring iterations: 4
```

Como podemos comprobar a simple vista, todas las variables son estadísticamente significativos excepto *"age"* y *"cause"*, cuyo p-valor es mayor a 0.05.

A continuación, utilizamos el algoritmo de clasificación paso a paso con la función stepAIC():

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## outcome ~ sex + age + cause + ec + eye + motor + verbal + pupils +
##     phm + sah + oblt + mdls + hmt
##
## Final Model:
## outcome ~ age + ec + eye + motor + verbal + pupils + phm + sah +
##     oblt + mdls + hmt
##
##
##      Step Df  Deviance Resid. Df Resid. Dev      AIC
## 1                           6916   6751.097 6779.097
## 2   - sex  1 0.2704474      6917   6751.368 6777.368
## 3 - cause  1 0.4047151      6918   6751.772 6775.772
```

Una vez más podemos comprobar que las variables de *"cause"* y *"sex"* son las que se descartan usando este algoritmo.

Posteriormente, como ya se ha comentado, se va a utilizar la función greedy.wilks() del paquete klaR que proporciona un flujo de trabajo sencillo para realizar selecciones paso a paso. El modelo inicial se define comenzando por la variable que mejor clasifica a la clase. Después, el modelo se amplía incluyendo otras variables según el criterio lambda de Wilk.

```
Formula containing included variables:

outcome ~ motor + age + verbal + oblt + pupils + mdls + hmt +

    eye + ec + phm + sah

Values calculated in each step of the selection procedure:


     vars  Wilks.lambda F.overall p.value.overall F.diff  p.value.diff

1   motor    0.8334752  1395.3739   1.336586e-278 1395.37389 1.336586e-278

2     age    0.7718762  1031.8936   0.000000e+00  557.27242  0.000000e+00

3  verbal    0.7351229  838.5771    0.000000e+00  349.07310  0.000000e+00

4    oblt    0.7109362  709.6116    0.000000e+00  237.50022  0.000000e+00

5  pupils    0.6991578  600.6881    0.000000e+00  117.58909  0.000000e+00

6    mdls    0.6927267  515.9468    0.000000e+00   64.79132  8.881784e-16

7     hmt    0.6878549  452.3689    0.000000e+00   49.42258  2.259193e-12

8     eye    0.6831415  404.5139    0.000000e+00   48.13803  4.329426e-12

9      ec    0.6788188  366.7416    0.000000e+00   44.42257  2.848166e-11

10    phm    0.6745857  336.4680    0.000000e+00   43.76960  3.967648e-11

11    sah    0.6724682  308.7956    0.000000e+00   21.96026  2.836353e-06
```

Vemos que el mejor modelo contendría 11 variables y seria aquel que tendría en cuenta todos los predictores menos los predictores “sex” y “cause” como era de esperar y como nos había indicado también el criterio AIC.

3.2.5 ANÁLISIS PCA

En primer lugar, antes de proceder con el análisis de componentes principales, vamos a tener en cuenta la matriz de correlaciones, puesto que un PCA tiene sentido si existen altas correlaciones entre las variables, ya que como se ha comentado con anterioridad, esto es indicativo de que existe información redundante y, por tanto, pocos factores explicaran gran parte de la variabilidad total.

Como ya vimos con las matrices de correlaciones solo obtuvimos correlaciones medianamente fuertes entre las variables de *"motor"*, *"eye"* y *"verbal"*, pero la correlación no era significativa por lo que no se descartó ninguna variable.

Un problema en el análisis de datos multivariante es la reducción de la dimensionalidad: es decir, si se puede conseguir con precisión los valores de las variables (p) con un pequeño subconjunto de ellas (r<p), habremos conseguido reducir la dimensión a costa de una pequeña perdida de información.

El análisis de componentes principales tiene este objetivo. Dada n observaciones de p variables, se analiza si es posible representar adecuadamente esta información con un conjunto menor de variables (construidas como combinaciones lineales de las originales).

El primer paso en el análisis de componentes principales consiste en la obtención de los valores y vectores propios de la matriz de covarianzas muestral o de la matriz de coeficientes de correlación que se obtienen a partir de los datos.

Debemos saber que el análisis de componentes principales utiliza la versión normalizada de los predictores originales. Estas variables pueden encontrarse en distintas escalas (kilómetros, litros, euros, etc.) y, por lo tanto, las varianzas también tendrán varias escalas.

Realizar el PCA con variables no normalizadas dará lugar a que haya cargas bastante grandes para variables con una varianza alta y a su vez, esto llevará a la dependencia de una componente principal con la variable con la varianza más alta. Esto no es deseable. Por lo

que se llevara a cabo una normalización de las variables. Al normalizar las variables, la distribución de la variabilidad entre las componentes parece más racional.

Veamos qué ocurre si utilizamos la **matriz de covarianza**, sin haber normalizado las variables:

```
## Importance of components:
##                             PC1    PC2     PC3     PC4     PC5    PC6
## Standard deviation     16.5659 2.0860 0.96050 0.76858 0.68086 0.6514
## Proportion of Variance  0.9713 0.0154 0.00327 0.00209 0.00164 0.0015
## Cumulative Proportion   0.9713 0.9867 0.98999 0.99208 0.99373 0.9952
##                             PC7    PC8     PC9    PC10    PC11   PC12
## Standard deviation     0.51378 0.4441 0.42801 0.41667 0.40227 0.3766
## Proportion of Variance 0.00093 0.0007 0.00065 0.00061 0.00057 0.0005
## Cumulative Proportion  0.99616 0.9969 0.99751 0.99812 0.99870 0.9992
##                            PC13    PC14
## Standard deviation     0.37091 0.29861
## Proportion of Variance 0.00049 0.00032
## Cumulative Proportion  0.99968 1.00000
```

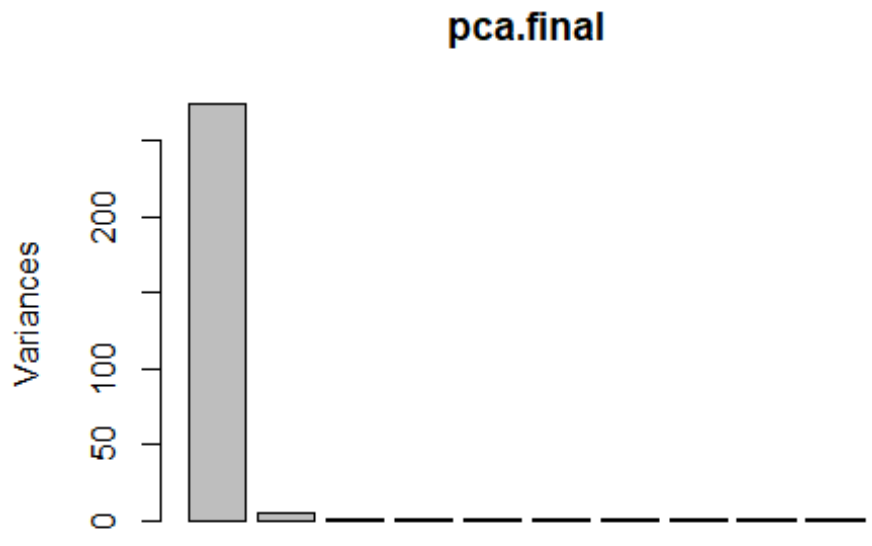


Ilustración 15. PCA. Varianzas sin normalizar

Como podemos comprobar en el grafico anterior, la componente PC1 posee toda la varianza.

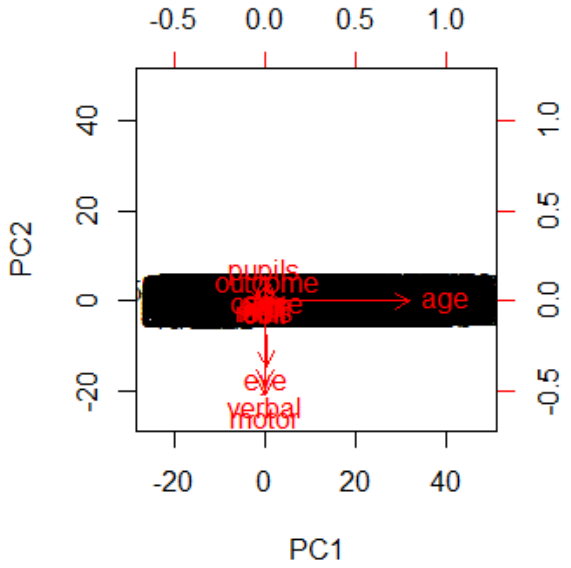


Ilustración 16. PCA. Gráfico sin normalizar

Como se puede comprobar en la gráfica anterior, al no haber escalado las variables, la primera componente principal (PC1) está dominada por la variable *"age"*, mientras que la segunda componente principal está dominada por las variables: *"eye"*, *"motor"* y *"verbal"*.

Ahora vamos a utilizar la **matriz de covarianza**, habiendo normalizado todas las variables.

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     0.7051 0.4706 0.45354 0.42754 0.41448 0.39605
## Proportion of Variance 0.2351 0.1047 0.09728 0.08644 0.08124 0.07418
## Cumulative Proportion  0.2351 0.3398 0.43709 0.52353 0.60477 0.67895
##                           PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.39038 0.37255 0.35681 0.29999 0.23364 0.20575
## Proportion of Variance 0.07207 0.06564 0.06021 0.04256 0.02582 0.02002
## Cumulative Proportion  0.75101 0.81665 0.87686 0.91941 0.94523 0.96525
##                           PC13    PC14
## Standard deviation     0.19667 0.18658
## Proportion of Variance 0.01829 0.01646
## Cumulative Proportion  0.98354 1.00000
```

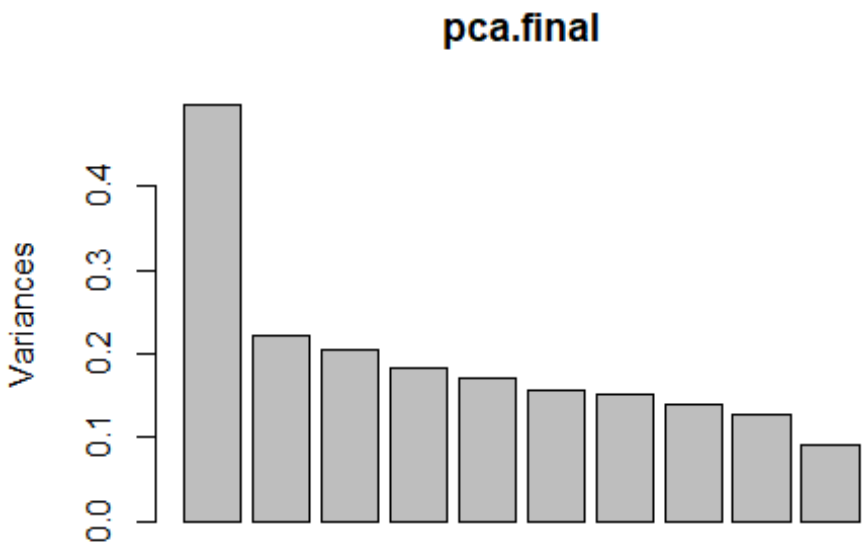


Ilustración 17. PCA. Varianzas normalizadas

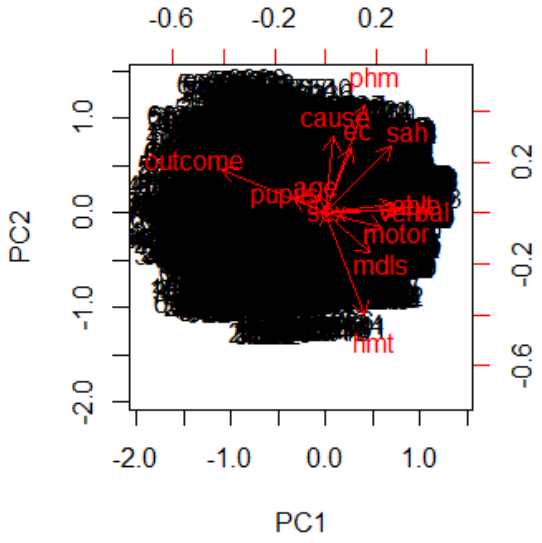


Ilustración 18. PCA. Gráfico normalizado

Como se puede comprobar en la gráfica anterior, al normalizar las variables, vemos que el peso de estas se distribuye de forma más uniforme entre las 2 componentes principales.

Para elegir nuestras componentes principales, podremos utilizar dos métodos:

- Por un lado, podemos utilizar el **criterio de Kaiser**, que consiste en conservar aquellos factores cuya desviación estándar al cuadrado asociada sea mayor que 1.

```
##  [1] 0.49713279 0.22143722 0.20570161 0.18278879 0.17178962 0.15685861
##  [7] 0.15239376 0.13879386 0.12731397 0.08999118 0.05458915 0.04233462
## [13] 0.03867736 0.03481073
```

    - Como se puede comprobar, utilizando este criterio, podríamos quedarnos con los componentes PC1, PC2, PC3, PC4 y PC5.

- Otra forma para saber cuántos componentes tener en cuenta es mantener el número de componentes necesarios para explicar al menos un porcentaje del total de la varianza. Por ejemplo, es importante **explicar al menos un 80%** de la varianza.

```
##              eigenvalue  variance.percent  cumulative.variance.percent
## Dim.1   0.49713279          23.509395                          23.50940
## Dim.2   0.22143722          10.471760                          33.98115
## Dim.3   0.20570161           9.727623                          43.70878
## Dim.4   0.18278879           8.644076                          52.35285
## Dim.5   0.17178962           8.123926                          60.47678
## Dim.6   0.15685861           7.417839                          67.89462
## Dim.7   0.15239376           7.206697                          75.10132
## Dim.8   0.13879386           6.563557                          81.66487
## Dim.9   0.12731397           6.020674                          87.68555
## Dim.10  0.08999118           4.255680                          91.94123
## Dim.11  0.05458915           2.581520                          94.52275
## Dim.12  0.04233462           2.002003                          96.52475
## Dim.13  0.03867736           1.829051                          98.35380
## Dim.14  0.03481073           1.646198                         100.00000
```

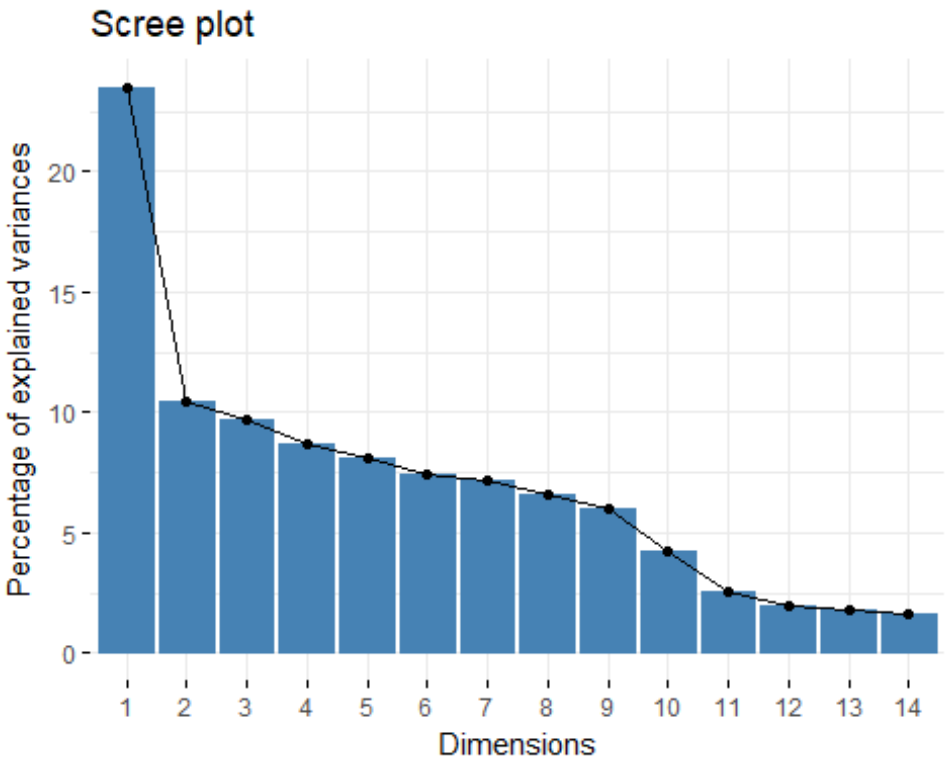


Ilustración 19. PCA. Porcentaje varianzas explicadas

Según este criterio, deberíamos quedarnos con los primeros componentes principales: PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8 y PC9.

A continuación, podremos ver la carga de cada variable respecto a las componentes principales.

```
##                PC1          PC2          PC3          PC4          PC5
##  sex          0.00754185   0.004825702 -0.05061100   0.09673923 -0.08169835
##  age         -0.03305719   0.089077063  0.02955650  -0.01094135 -0.09084363
##  cause        0.03634525   0.376804210  0.18316340  -0.09565336  0.12960828
##  ec           0.12719737   0.321982271  0.52579134  -0.25862893  0.53735713
##  eye          0.37584853   0.019196020  0.23805982   0.11908233 -0.21180882
##  motor        0.27860212  -0.072483884  0.17979746   0.09522108 -0.10943985
##  verbal       0.35002340   0.010691932  0.18829500   0.08182141 -0.15454678
##  pupils      -0.15858549   0.076136117 -0.06963948  -0.02843065  0.05073360
##  phm          0.19546358   0.532494470 -0.37457091   0.67612986  0.14592864
##  sah          0.32519856   0.328128704 -0.48960240  -0.55085227  0.09217214
##  oblt         0.34248291   0.040645898 -0.19093528  -0.28120171 -0.30335822
##  mdls         0.22183247  -0.197257325 -0.10251020  -0.10980630 -0.19521708
##  hmt          0.19633261  -0.505774550 -0.31239966   0.05260687  0.65098174
```

```
##  outcome   -0.51555059     0.217405899   -0.18078512   -0.17100730   -0.11188899
##                     PC6            PC7           PC8           PC9          PC10
##  sex          0.468631120  -0.545637298   0.56848224  -0.362100975   0.05449783
##  age          0.213465260  -0.004555270  -0.05438566   0.015539408  -0.06192319
##  cause        0.266773385   0.351493551  -0.30349198  -0.693069485   0.13967486
##  ec           0.075208510  -0.005328975   0.31834880   0.361266429   0.09920550
##  eye          0.216580398  -0.199155308  -0.32508940   0.111568280  -0.05697973
##  motor        0.059764349  -0.130958699  -0.16923921   0.085139555   0.03200627
##  verbal       0.158055809  -0.182516720  -0.24157401   0.110972727  -0.03748816
##  pupils       0.012761929   0.016462188   0.05708582  -0.041706107  -0.07740804
##  phm         -0.007730786   0.116800019   0.08439404   0.173756733   0.08844781
##  sah          -0.230815829  -0.386705035  -0.15835007  -0.053806287   0.02958798
##  oblt         0.305400757   0.515894927   0.33718561   0.134660703  -0.41366626
##  mdls         0.047519909   0.225944825   0.14468847   0.082413000   0.86550643
##  hmt          0.374508857   0.059210871  -0.18071620   0.002877392  -0.06071833
##  outcome     0.541525055  -0.066325752  -0.29153155    0.407445568    0.13989249
##                      PC11           PC12           PC13           PC14
##   sex            -0.046320481    0.051178890   -0.043546584    0.008349846
##   age             0.064650181   -0.794933844    0.526023857   -0.127687713
##   cause          -0.058199593    0.082271746   -0.018041653    0.006485046
##   ec              0.007524695   -0.018597326   -0.011587465    0.011292569
##   eye             0.283758366   -0.264686334   -0.509573504    0.352793604
##   motor          -0.444932122    0.256671073    0.528239380    0.513066463
##   verbal          0.222888103    0.350653837    0.224483179   -0.676707194
##   pupils          0.791456558    0.250663510    0.353092549    0.368086924
##   phm            -0.024586186   -0.001250305    0.003211589    0.006193667
##   sah            -0.020857062   -0.040982217    0.022648597    0.025366341
##   oblt           -0.045305471    0.083331468   -0.030150696    0.035015405
##   mdls            0.134062896   -0.046673682    0.029152475    0.008125267
##   hmt             0.024565131   -0.026644622    0.008717377    0.001351517
## outcome          -0.118307105    0.158441132   -0.069394421    0.018965211
```

Como conclusiones teniendo en cuenta el PCA y las matrices de correlaciones, no se puede descartar ninguna variable por los siguientes motivos:

- Las correlaciones entre las variables *"eye"*, *"motor"* y *"verbal"* no son lo suficientemente fuertes como para considerar que existe información redundante. El resto de pares de variables tienen una correlación poco significativa.
- Los criterios utilizados para elegir las componentes principales nos han indicado que se necesitan al menos 5 componentes principales usando el criterio de Kaiser y 9 utilizando el criterio del 80% de la proporción de la varianza. Teniendo en cuenta que poseemos 14 variables, la reducción no es significativa y se perdería interpretabilidad.

3.3 MODELADO DE DATOS

Una vez que hemos realizado el pre-procesado de datos, y hemos el respectivo análisis estadístico sobre nuestros datos para tener un mejor conocimiento de ellos, vamos a comenzar con la aplicación de modelos estadísticos con el objetivo de obtener el mejor modelo que nos permita una mejor predicción de los datos.

3.3.1 USO DE MODELOS

En esta sección realizaremos un estudio de un conjunto de modelos y los compararemos entre sí con el objetivo de ver cuál es el mejor. Para ello tendremos en cuenta la precisión de sus predicciones y los tiempos que tarda en realizar el entrenamiento y las predicciones.

Existe dos tipos de modelos predictivos: modelos de clasificación y de regresión. Los modelos de clasificación, que son los que vamos a utilizar en este documento, nos permiten predecir la pertenencia a una clase (vivo o fallecido). En nuestro caso clasificaremos entre los pacientes quienes son los más propensos a fallecer debido a los síntomas y los escáneres.

Las técnicas de modelado de análisis predictivo más usadas son las siguientes:

- Arboles de decisión
- Regresión lineal y logística
- Redes neuronales
- Análisis bayesiano
- Modelos combinados
- Gradient boosting
- Combinación de modelos
- Etc.

En este documento se realizará un estudio de modelos pertenecientes a cada una de las técnicas anteriores.

Para realizar el entrenamiento del conjunto de datos, vamos a utilizar el paquete de *Caret*, usando la validación cruzada con 10 iteraciones para todos los modelos que estudiemos. Para ajustar los hiperparámetros utilizaremos la rejilla de búsqueda que nos proporciona *Caret*.

Posteriormente también utilizaremos el paquete de *caretEnsemble* con el objetivo de conseguir un mejor resultado en la predicción mediante la combinación de varios modelos.

3.3.1.1 Datos de entrenamiento y test

En primer lugar, dividiremos los datos en un conjunto de entreno (train) y un conjunto de pruebas (test). Tenemos 6930 registros por lo que el 30% son datos de prueba (2096) y el 70% datos de entrenamiento del modelo (4890).

El subconjunto de datos de entrenamiento se utiliza para estimar los parámetros del modelo y el subconjunto de datos de test se emplea para comprobar el comportamiento del modelo estimado. Para dividir el conjunto de datos en ambos subconjuntos, se utiliza un procedimiento de muestreo. Lo ideal es entrenar el modelo con un conjunto de datos independiente de los datos con los que realizamos el test.

3.3.1.2 Regresión logística

Usando todos los predictores

En primer lugar, vamos a construir nuestro modelo de regresión logística con los datos de entrenamiento.

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6212  -0.7896   0.4030   0.7663   2.7535
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.025051   0.369242 -13.609  < 2e-16 ***
## sex         -0.088864   0.094387  -0.941 0.346456
## age         -0.042629   0.002311 -18.445  < 2e-16 ***
## cause       -0.011517   0.047834  -0.241 0.809739
## ec           0.437707   0.083247   5.258 1.46e-07 ***
## eye          0.273820   0.046107   5.939 2.87e-09 ***
## motor        0.261235   0.030660   8.520  < 2e-16 ***
## verbal       0.208675   0.038848   5.372 7.81e-08 ***
## pupils      -0.352803   0.051024  -6.914 4.70e-12 ***
```

```
## phm          0.460228   0.077954   5.904 3.55e-09 ***
## sah          0.261974   0.077846   3.365 0.000765 ***
## oblt         0.601150   0.088771   6.772 1.27e-11 ***
## mdls         0.819740   0.110993   7.386 1.52e-13 ***
## hmt          0.440056   0.080109   5.493 3.95e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6601.0  on 4889  degrees of freedom
## Residual deviance: 4774.3  on 4876  degrees of freedom
## AIC: 4802.3
##
## Number of Fisher Scoring iterations: 4
```

En el resultado anterior podemos ver el p-value de las variables, destacando que el p-value mayor a 0.05 viene dado por las variables de “sex” y “cause” una vez más. Esto indica que deberíamos probar el modelo sin estas variables. Se muestra también el AIC del modelo, que es un indicador comparativo entre modelos.

Ahora vamos a utilizar los datos de test para predecir el modelo que hemos construido y comprobaremos el ajuste de este. Una vez predicho el modelo, utilizaremos la matriz de confusión para visualizar los resultados de la predicción.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  546  191
##          V  303 1056
##
##                Accuracy : 0.7643
##                  95% CI : (0.7455, 0.7823)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5005
##  Mcnemar's Test P-Value : 5.91e-07
##
##             Sensitivity : 0.6431
##             Specificity : 0.8468
##          Pos Pred Value : 0.7408
##          Neg Pred Value : 0.7770
##              Prevalence : 0.4051
```

```
##          Detection Rate : 0.2605
##    Detection Prevalence : 0.3516
##       Balanced Accuracy : 0.7450
##
##        'Positive' Class : F
##
```

Como podemos comprobar, el modelo se ajusta bastante bien, con un 76%. Dicho de otra forma, el modelo es capaz de predecir correctamente un 76% de los datos de los pacientes. Eso sí, esta precisión nos viene marcada por el intervalo de confianza.

Además de mostrarse la sensibilidad y la especificidad, se muestran también el valor positivo predicho (PPV) y el valor negativo predicho (PPN).

Vemos en el gráfico de la matriz de confusión que 546 pacientes son verdaderos positivos, y 1056 son verdaderos negativos. Obviamente cuanto mayor sean los verdaderos positivos o negativos mejor, ya que seremos capaces de predecir con mayor precisión si un paciente vive o muere.

Eliminando los predictores "sex" y "cause"

Ahora vamos a realizar un estudio eliminando las variables de "sex" y "cause". Los parámetros utilizados en la rejilla de búsqueda de caret al entrenar los datos serán los mismos utilizados en el modelo de regresión logística para todas las variables. El resultado es el siguiente:

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6141  -0.7889   0.4004   0.7641   2.7605
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.034088   0.366752 -13.726  < 2e-16 ***
## age         -0.042908   0.002285 -18.776  < 2e-16 ***
## ec           0.435948   0.082564   5.280 1.29e-07 ***
## eye          0.272264   0.046034   5.914 3.33e-09 ***
## motor        0.261925   0.030627   8.552  < 2e-16 ***
```

```
## verbal       0.207757   0.038836   5.350 8.82e-08 ***
## pupils      -0.354631   0.050995  -6.954 3.55e-12 ***
## phm          0.457171   0.077761   5.879 4.12e-09 ***
## sah          0.261085   0.077763   3.357 0.000787 ***
## oblt         0.601978   0.088699   6.787 1.15e-11 ***
## mdls         0.820937   0.110781   7.410 1.26e-13 ***
## hmt          0.438623   0.079952   5.486 4.11e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6601.0  on 4889  degrees of freedom
## Residual deviance: 4775.3  on 4878  degrees of freedom
## AIC: 4799.3
##
## Number of Fisher Scoring iterations: 4
```

De los resultados anteriores podemos destacar que el AIC es menor al que obtuvimos con el modelo completo. Esto significa que ha mejorado.

A continuación, vamos a utilizar el modelo construido para realizar una predicción a partir de los datos de prueba. Una vez obtenida la predicción, mostraremos la matriz de confusión. A continuación, se muestra el resultado detallado.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  543  191
##          V  306 1056
##
##                Accuracy : 0.7629
##                  95% CI : (0.7441, 0.781)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4972
##  Mcnemar's Test P-Value : 3.161e-07
##
##             Sensitivity : 0.6396
##             Specificity : 0.8468
##          Pos Pred Value : 0.7398
##          Neg Pred Value : 0.7753
##              Prevalence : 0.4051
##          Detection Rate : 0.2591
##    Detection Prevalence : 0.3502
```

```
##       Balanced Accuracy : 0.7432
##
##        'Positive' Class : F
##
```

Teniendo en cuenta los resultados obtenidos eliminando las variables de “sex” y “cause”, se ha obtenido el mismo número de verdaderos negativos y, sin embargo, el número de verdaderos negativos ha decrecido en 3 unidades. Eso significa que la predicción es peor, sin embargo, a grandes rasgos quizá compense la eliminación de estas variables si quisiéramos reducir el tamaño del conjunto de datos.

Comparando el área parcial bajo la curva

A continuación, vamos a utilizar la curva de ROC con los valores de pAUC para la regresión logística y sus modelos:

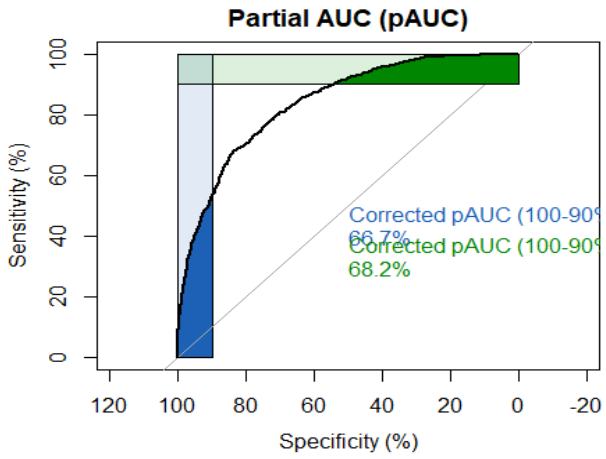


Ilustración 20. Regresión Logística. pAUC

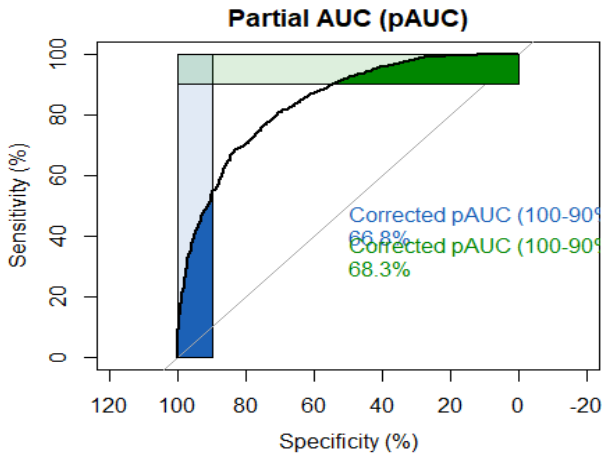


Ilustración 21. Regresión Logística. Sex-Cause. pAUC

Si tomamos un rango concreto de Especificidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 66,7% (cuando utilizamos todas las variables) y un 66,8% (cuando eliminamos el sexo y la causa) de casos negativos y que realmente son negativos, es decir seremos capaces de predecir que un paciente que vive, realmente va a vivir. Vemos que entre los dos modelos apenas existe una diferencia.

Si tomamos un rango concreto de sensibilidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 68,2% (cuando utilizamos todas las variables) y un 68,3% (cuando eliminamos el sexo y la causa) de casos positivos y que realmente son positivos, es decir seremos capaces de predecir que un paciente que fallece, realmente va a fallecer.

3.3.1.3 Naïves Bayes

Las redes bayesianas, junto con los árboles de decisión y las redes neuronales artificiales, han sido los tres métodos más usados en aprendizaje automático durante estos últimos años en tareas como la clasificación.

La idea de Naïve Bayes es sencilla y es usar las probabilidades condicionales de los valores de una variable para determinar a qué categoría pertenece, estas probabilidades se calculan con el teorema de Bayes.

Por ejemplo, si quisiéramos clasificar la valoración de un determinado servicio, podríamos hacerlo en dos categorías, "positiva" y "negativa"; de esta forma, tenemos que determinar que palabras son más probables de encontrar en cada una de estas categorías. Se puede intuir que es más probable que una valoración pertenezca a la categoría "positiva", si los valores son "bueno" o "excelente" y menos probable si contiene palabras como "malo" o "deficiente".

Por tanto, deberemos obtener la probabilidad de que una valoración pertenezca a la categoría "positiva", dado que la valoración contiene la palabra "excelente". Por tanto, deberíamos calcular p(positiva|excelente)

Debido a que este algoritmo calcula las probabilidades de cada valor por separado, como si los valores fueran independientes unos de otros, se conoce como "ingenuo". Lo que se hace, por lo tanto, es calcular la probabilidad condicional de cada valor, asumiendo de forma "ingenua", que en esta probabilidad no importa cuales valores de variables lo acompañan en cada registro.

En este caso, también vamos a utilizar el paquete de "caret" para realizar el entrenamiento del modelo. Utilizaremos también la validación cruzada con 10 iteraciones.

Los parámetros con los que hemos hecho la fuerza bruta ha sido los siguientes:

- useKernel: tipo de distribución. Si tiene valor de "verdadero", se utiliza una estimación de densidad de núcleo. En caso contrario se utiliza una densidad normal. Las posibles densidades son: Uniforme, Gaussiana, Triangular, etc.
- fL: la corrección de Laplace. Es un suavizado capaz de corregir los posibles problemas existentes cuando las probabilidades son nulas o muy bajas, por ausencia,

por ejemplo, de algunos valores de atributos en algunas categorías en el conjunto de entrenamiento. Este parámetro por tanto evita la influencia de probabilidades nulas.

- adjust: Ajuste de ancho de banda para la estimación del kernel. El ancho de banda controla el ancho de las funciones del kernel. Un ancho de banda grande da una estimación de densidad uniforme, un ancho de banda pequeño da estimaciones de densidad ruidosa/ondulante

Usando todos los predictores

En primer lugar, vamos a construir un modelo de Naïves Bayes utilizando todos los predictores. Para ello igual que en casos anteriores vamos a utilizar los datos de entrenamiento para construir el modelo.

En un principio tendremos que utilizar la rejilla de búsqueda de datos con parámetros que hayamos considerado dentro de un determinado rango.

El resultado obtenido después de utilizar la rejilla de búsqueda de caret es el siguiente:

```
## Naive Bayes
##
## 4890 samples
##   13 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results across tuning parameters:
##
##   usekernel  fL  adjust  Accuracy   Kappa
##   FALSE      0   0       0.7398773  0.4562820
##   FALSE      0   1       0.7398773  0.4562820
##   FALSE      0   2       0.7398773  0.4562820
##   FALSE      1   0       0.7398773  0.4562820
##   FALSE      1   1       0.7398773  0.4562820
##   FALSE      1   2       0.7398773  0.4562820
##   FALSE      2   0       0.7398773  0.4562820
##   FALSE      2   1       0.7398773  0.4562820
##   FALSE      2   2       0.7398773  0.4562820
##    TRUE      0   0             NaN        NaN
##    TRUE      0   1       0.7370143  0.4066122
```

```
##    TRUE      0   2       0.7372188  0.4098870
##    TRUE      1   0             NaN        NaN
##    TRUE      1   1       0.7370143  0.4066122
##    TRUE      1   2       0.7372188  0.4098870
##    TRUE      2   0             NaN        NaN
##    TRUE      2   1       0.7370143  0.4066122
##    TRUE      2   2       0.7372188  0.4098870
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = FALSE
##  and adjust = 0.
```

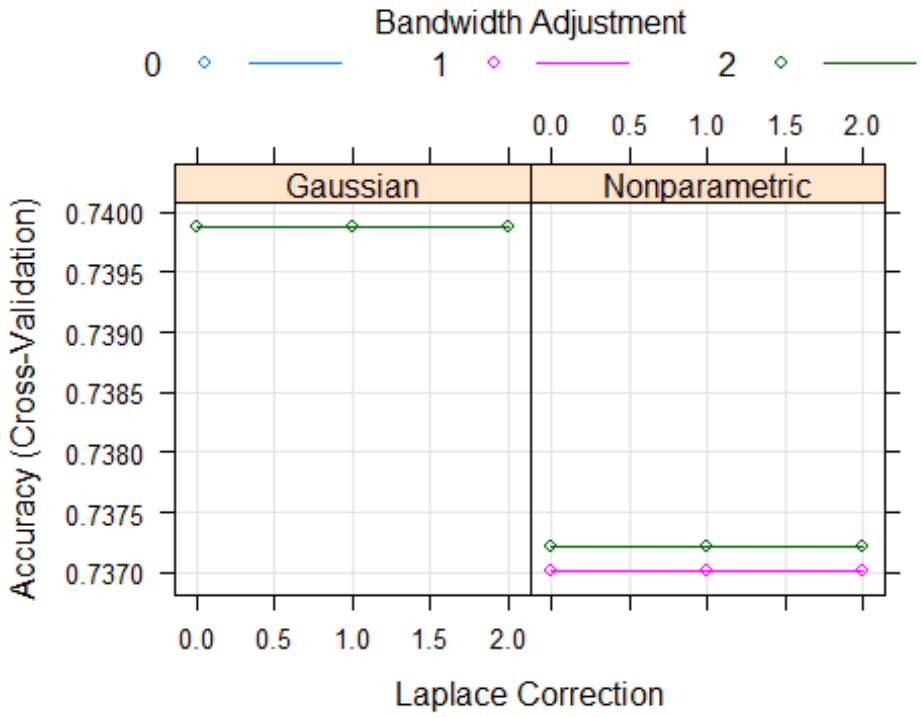


Ilustración 22. Naïves Bayes. Mejor ajuste.

Con estas iteraciones se ha conseguido que, para construir el mejor modelo, se necesitan los siguientes valores en los parámetros de configuración:

- fL: 0
- usekernel: FALSE

- adjust: 0

Con estos valores que hemos obtenido a través del paquete de caret, construimos el modelo de Naïves Bayes que se ha considerado como el que mejor precisión nos aporta.

```
Naive Bayes

4890 samples
  13 predictor
   2 classes: 'F', 'V'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
Resampling results:

  Accuracy   Kappa
  0.7398773  0.456282

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'usekernel' was held constant at a value of FALSE

Tuning parameter 'adjust' was held constant at a value of 0
```

Una vez que hemos construido el modelo, con los datos de test vamos a predecir el modelo construido. Los resultados que se van a obtener se van a mostrar a través de la siguiente matriz de confusión:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   F   V
##          F 551 249
##          V 298 998
##
##                Accuracy : 0.739
##                  95% CI : (0.7197, 0.7577)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.4535
##  Mcnemar's Test P-Value : 0.04014
##
##             Sensitivity : 0.6490
```

```
##             Specificity : 0.8003
##          Pos Pred Value : 0.6887
##          Neg Pred Value : 0.7701
##              Prevalence : 0.4051
##          Detection Rate : 0.2629
##    Detection Prevalence : 0.3817
##       Balanced Accuracy : 0.7247
##
##        'Positive' Class : F
##
```

Una vez más, con la matriz de confusión podemos obtener los resultados que nos aporten más interés.

Eliminando los predictores "sex" y "cause"

Ahora vamos a estudiar el modelo de Naïves Bayes eliminando las variables de "sex" y "cause".

Una vez más vamos a realizar la construcción del modelo con los parámetros utilizados en la versión con todos los predictores de Naïves Bayes (el estudiado con anterioridad).

```
## Naive Bayes
##
## 4890 samples
##   11 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7400818  0.4571251
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'usekernel' was held constant at a value of FALSE
##
## Tuning parameter 'adjust' was held constant at a value of 0
```

Una vez que hemos construido el modelo, vamos a utilizar los datos de prueba para obtener las predicciones. Utilizaremos una vez más la matriz de confusión para obtener los resultados de esta predicción.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   F   V
##          F 548 249
##          V 301 998
##
##                Accuracy : 0.7376
##                  95% CI : (0.7182, 0.7563)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.4502
##  Mcnemar's Test P-Value : 0.02966
##
##             Sensitivity : 0.6455
##             Specificity : 0.8003
##          Pos Pred Value : 0.6876
##          Neg Pred Value : 0.7683
##              Prevalence : 0.4051
##          Detection Rate : 0.2615
##    Detection Prevalence : 0.3802
##       Balanced Accuracy : 0.7229
##
##        'Positive' Class : F
##
```

Como se puede comprobar, con Bayes Naïves se ha conseguido unos resultados bastante negativos. Eso se debe a que como nuestros datos no poseen una distribución normal, Naïves Bayes obtiene un gran número de probabilidades nulas.

Comparando el área parcial bajo la curva

A continuación, vamos a utilizar la curva de ROC con los valores de pAUC para los modelos de Naïves Bayes.

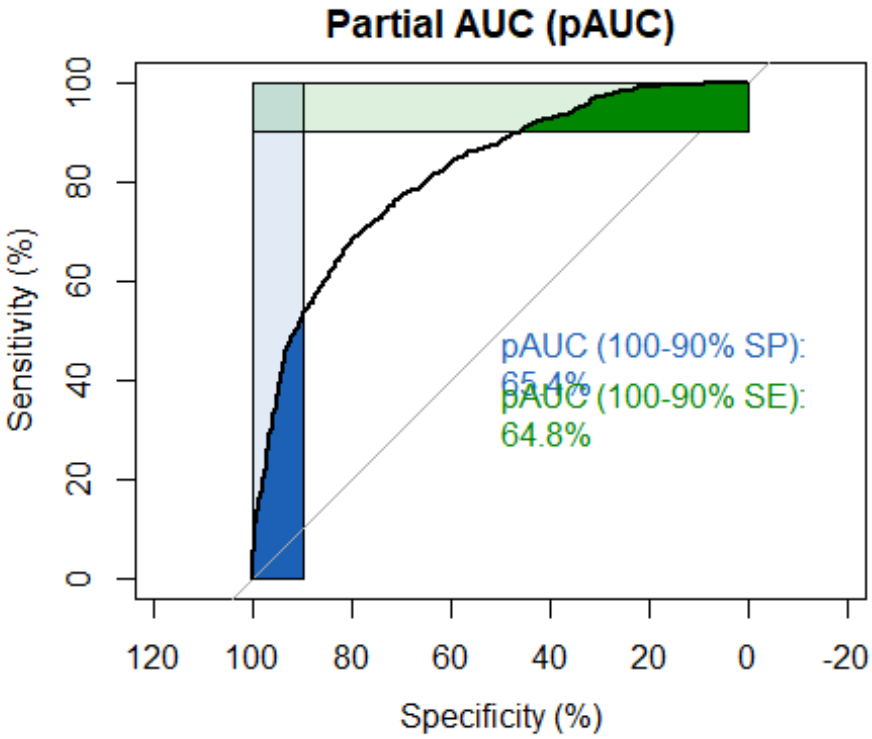


Ilustración 23. Naïve Bayes. pAUC

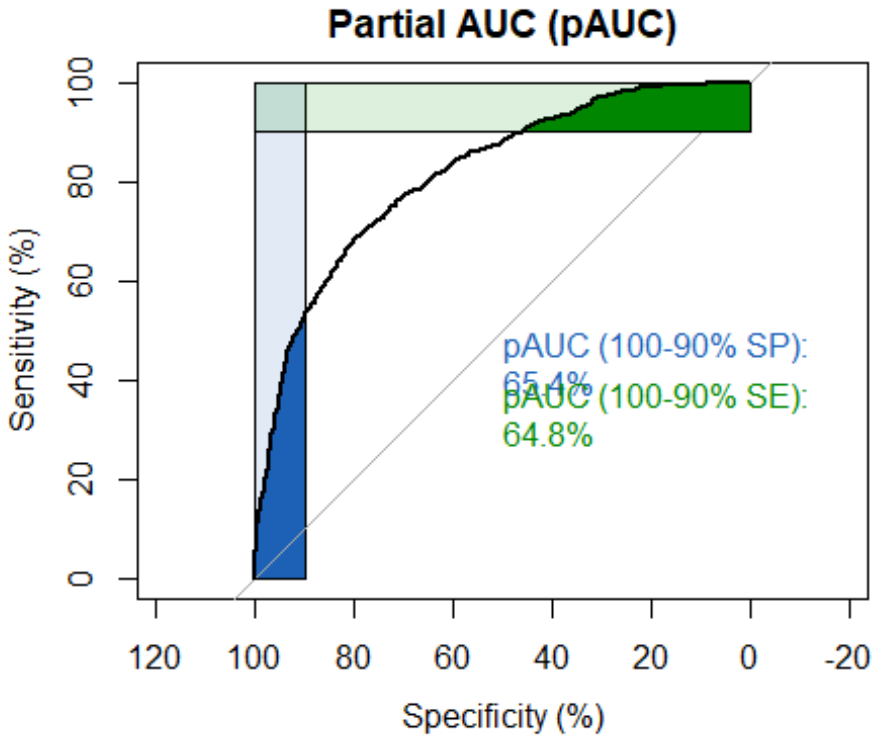


Ilustración 24. Naïves Bayes. pAUC II

Si tomamos un rango concreto de Especificidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 65,4% (cuando utilizamos todas las variables) y un 65,4% (cuando eliminamos el sexo y la causa) de casos negativos y que realmente son negativos, es decir seremos capaces de predecir que un paciente que vive, realmente va a vivir. Vemos que ambos modelos poseen los mismos valores para este rango.

Si tomamos un rango concreto de sensibilidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 64,8% (cuando utilizamos todas las variables) y un 64,8% (cuando eliminamos el sexo y la causa) de casos positivos y que realmente son positivos, es decir seremos capaces de predecir que un paciente que fallece, realmente va a fallecer. Vemos que ambos modelos poseen los mismos valores para este rango.

3.3.1.4 Random Forest

Random Forest es otra técnica de aprendizaje automático y nace como mejora sustancial de los árboles simples. Combina una cantidad grande de árboles de decisión independientes probados sobre conjuntos de datos aleatorios con igual distribución.

La fase de aprendizaje consiste en crear una gran cantidad de árboles de decisión independientes. Estos árboles se construyen a partir de los datos de entrada ligeramente modificados. Se modifica el conjunto inicial de partida, de la siguiente forma:

1. Se selecciona aleatoriamente con reemplazamiento un porcentaje de datos de la muestra total.
    a. Es habitual incluir un segundo nivel aleatoriedad, esta vez afectando los atributos:
2. En cada nodo, al seleccionar la partición óptima, tenemos en cuenta sólo una porción de los atributos, elegidos al azar en cada ocasión.
3. Una vez que se tienen muchos árboles -500 por ejemplo- la fase de clasificación se lleva a cabo de la siguiente manera:
4. Cada árbol se evalúa de forma independiente y la predicción del bosque será la media de los 500 árboles. La proporción de árboles que toman una misma respuesta se interpreta como la probabilidad de la misma.

El modelo de Random Forest también lo construiremos con los mismos datos de entrenamiento que utilizamos en la regresión logística.

Usando todos los predictores

En primer lugar, deberemos obtener el parámetro ajustable de “mtry”. “Mtry” es el número de variables aleatorias utilizadas en cada árbol. La reducción del “mtry” reduce tanto la correlación como la fuerza, aumentando ambas en caso contrario. Además, es el único parámetro ajustable al cual los bosques aleatorios son algo sensibles.

Para calcularlo tenemos dos opciones: utilizar Caret y la rejilla de búsqueda o utilizar la función de tuneRF que nos proporciona el paquete de Random Forest. En este documento se estudiarán ambas.

En primer lugar, si utilizamos Caret y su rejilla, debemos comprobar una serie de valores de "mtry". Nosotros hemos probado de forma secuencial desde 1 hasta 15 de uno en uno. Y se ha obtenido el siguiente resultado:

```
Random Forest
4890 samples
  13 predictor
   2 classes: 'F', 'V'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
Resampling results across tuning parameters:
  mtry  Accuracy   Kappa
   1    0.7470348  0.4466697
   2    0.7662577  0.4999483
   3    0.7660532  0.5032624
   4    0.7613497  0.4952490
   5    0.7548057  0.4831984
   6    0.7527607  0.4805898
   7    0.7449898  0.4648190
   8    0.7417178  0.4584319
   9    0.7398773  0.4550546
  10    0.7370143  0.4488703
  11    0.7339468  0.4427285
  12    0.7355828  0.4457404
  13    0.7300613  0.4354490
  14    0.7341513  0.4433639
  15    0.7335378  0.4419981
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

Ahora mostramos el grafico sobre el número de predictores usados y la mejora en la precisión del modelo.

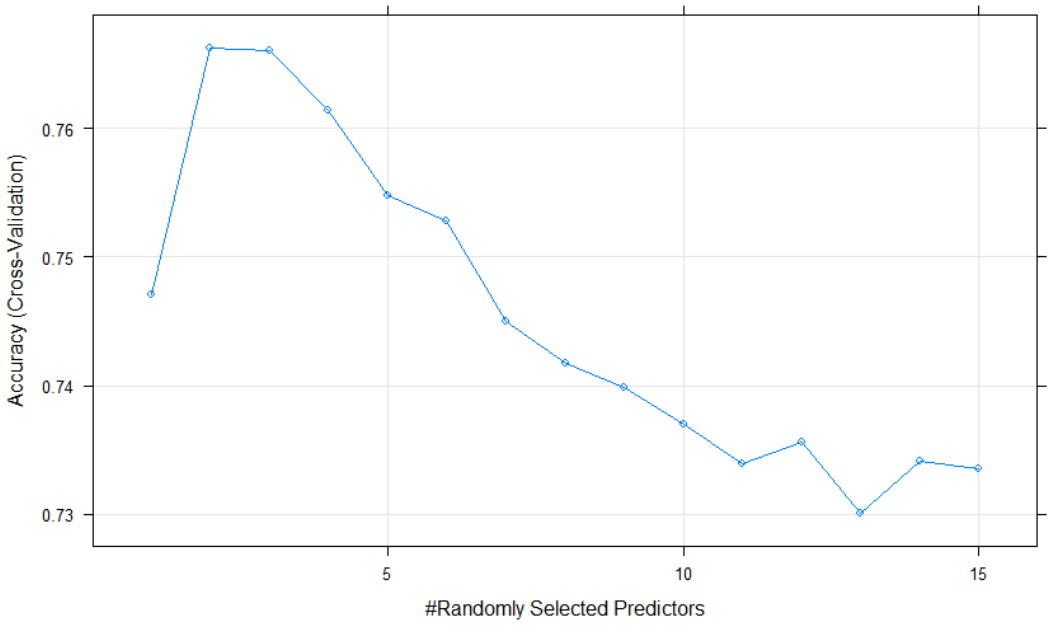


Ilustración 25.Random Forest. Accuracy-Predictors

Como podemos comprobar, el mejor valor de precisión se obtiene con un valor de "mtry" de 2.

En segundo lugar, se ha utilizado la función de tuneRF con los siguientes parámetros:

- mtryStart = 1.-> Cantidad de variables inicial.
- stepFactor = 2 -> Incremento de variables.
- ntreeTry = 50 -> Cantidad arboles a ejecutar en cada iteración.
- improve = .01 -> Mejora minina del OOB para seguir iteraciones.

Con estos parámetros vamos a obtener el siguiente resultado:

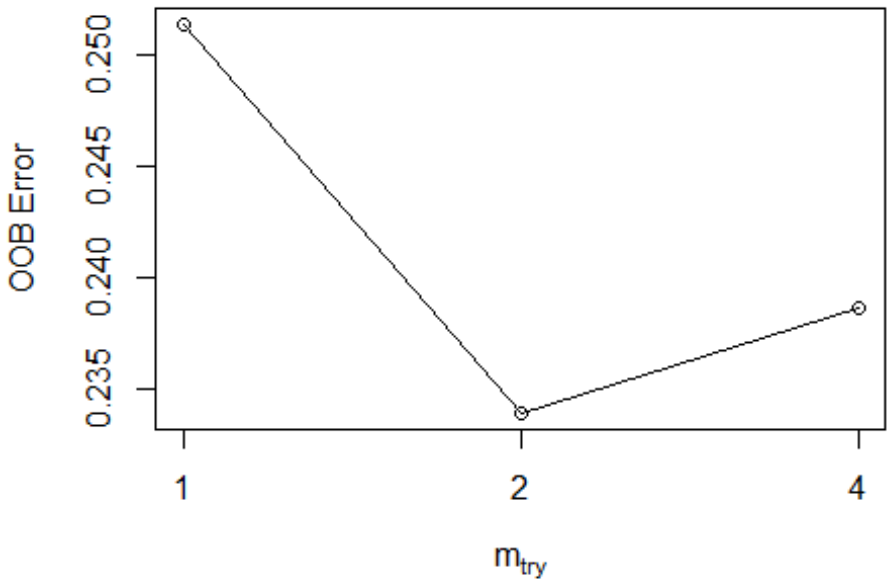


Ilustración 26. Random Forest. OOBError-MTRY

```
##       mtry  OOBError
## 1.OOB    1 0.2513292
## 2.OOB    2 0.2339468
## 4.OOB    4 0.2386503
```

En cada iteración del algoritmo de *"Random Forest"* se genera un error conocido como **OOB**, este error ira aumentando o disminuyendo en cada iteración y por cada variable que se incluya en el algoritmo. Utilizaremos el valor de 2 como "mtry", debido a que es el valor que menor error consigue. Como vemos es el mismo valor que se ha conseguido con caret.

Una vez obtenido el óptimo valor de "mtry", vamos a volver a construir el modelo con el "mtry" optimo.

```
## Random Forest
##
## 4890 samples
##   13 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
```

```
## Resampling results:
##
##   Accuracy   Kappa
##   0.7656442  0.4981153
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

Después de construir el modelo, vamos a utilizar los datos de test para predecirlo. Con las predicciones obtenidas, podremos obtener la matriz de confusión que se muestra en el siguiente resultado.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  520  147
##          V  329 1100
##
##                Accuracy : 0.7729
##                  95% CI : (0.7544, 0.7907)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5121
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6125
##             Specificity : 0.8821
##          Pos Pred Value : 0.7796
##          Neg Pred Value : 0.7698
##              Prevalence : 0.4051
##          Detection Rate : 0.2481
##    Detection Prevalence : 0.3182
##       Balanced Accuracy : 0.7473
##
##        'Positive' Class : F
##
```

Eliminando los predictores “sex” y “cause”

Como hemos hecho con los modelos estudiados hasta ahora, vamos a ver qué resultados obtenemos eliminando los predictores de “sex” y “cause”. Para ello vamos a utilizar los

mismos parámetros de la rejilla de búsqueda que se utilizaron para la versión con todos los predictores de Random Forest.

Una vez que hemos construido el modelo usando los datos de entrenamiento y los parámetros de la rejilla de búsqueda, obtenemos el siguiente resultado informativo donde se muestra el modelo utilizado, los predictores, el método de muestreo y el valor de configuración de la rejilla.

```
## Random Forest
##
## 4890 samples
##   11 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7672802  0.5022266
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

Una vez que hemos realizado la construcción del modelo, vamos a utilizar los datos de prueba para realizar la predicción. Con estas predicciones podremos realizar la matriz de confusión:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  519  148
##          V  330 1099
##
##                Accuracy : 0.7719
##                  95% CI : (0.7534, 0.7898)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5101
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6113
##             Specificity : 0.8813
```

```
##          Pos Pred Value : 0.7781
##          Neg Pred Value : 0.7691
##              Prevalence : 0.4051
##          Detection Rate : 0.2476
##    Detection Prevalence : 0.3182
##       Balanced Accuracy : 0.7463
##
##        'Positive' Class : F
##
```

Comparando el área parcial bajo la curva

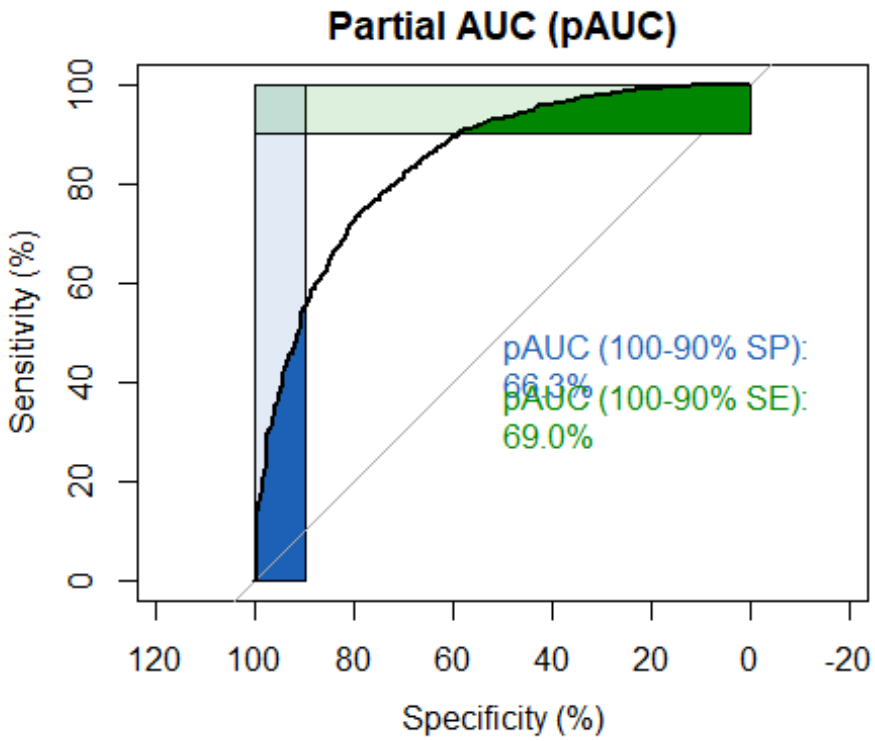


Ilustración 27. Random Forest. pAUC

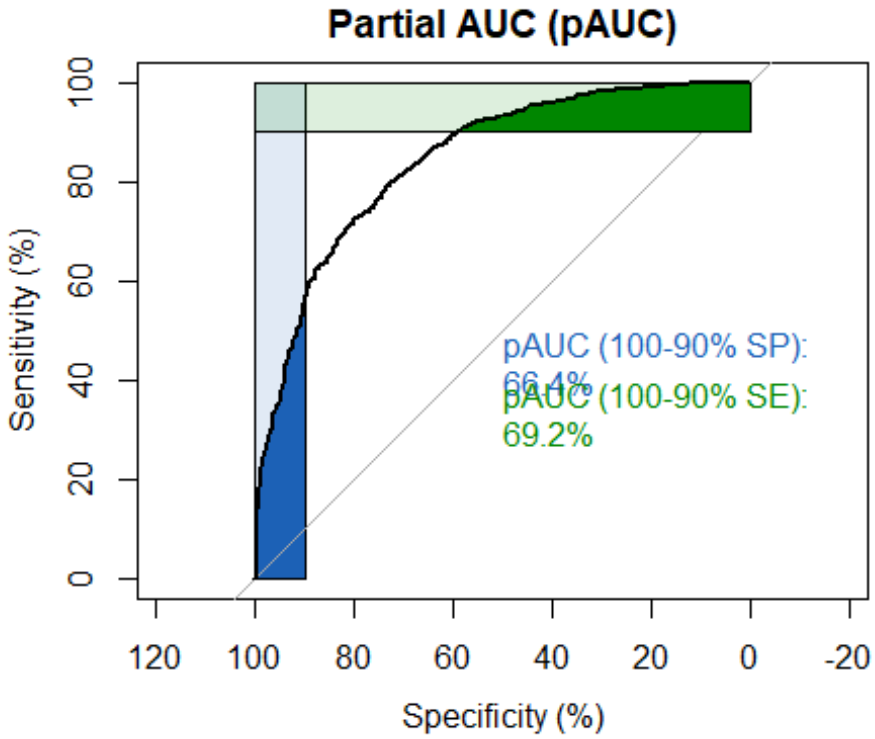


Ilustración 28. Random Forest. pAUC II

Si tomamos un rango concreto de Especificidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 66,3% (cuando utilizamos todas las variables) y un 66,4% (cuando eliminamos el sexo y la causa) de casos negativos y que realmente son negativos, es decir seremos capaces de predecir que un paciente que vive, realmente va a vivir. Vemos que el modelo que excluye las variables de sexo y causa obtiene mejores resultados.

Si tomamos un rango concreto de sensibilidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 69% (cuando utilizamos todas las variables) y un 69,2% (cuando eliminamos el sexo y la causa) de casos positivos y que realmente son positivos, es decir seremos capaces de predecir que un paciente que fallece, realmente va a fallecer. Vemos que el modelo que excluye las variables de sexo y causa obtiene mejores resultados.

3.3.1.5 Adaboost

El algoritmo AdaBoost propone entrenar iterativamente una serie de clasificadores base, de tal modo que cada nuevo clasificador preste mayor atención a los datos clasificados erróneamente por los clasificadores anteriores, y combinarlos de tal modo que se obtenga un clasificador con elevadas prestaciones.

Las características que lo convierten en un buen método son: su capacidad de evadir el **overfitting** y su menor porcentaje de error a cambio de tener un error mayor durante el entrenamiento.

Usando todos los predictores

En primer lugar, vamos a utilizar el paquete de "caret" con el objetivo de encontrar el modelo optimo usando adaboost. Aplicaremos la validación cruzada.

```
## Boosted Classification Trees
##
## 4890 samples
##   13 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results across tuning parameters:
##
##   maxdepth  iter  Accuracy   Kappa
##   1          50   0.7421268  0.4315823
##   1         100   0.7523517  0.4623808
##   1         150   0.7588957  0.4800449
##   2          50   0.7603272  0.4891654
##   2         100   0.7672802  0.5062127
##   2         150   0.7674847  0.5095219
##   3          50   0.7685072  0.5052091
##   3         100   0.7687117  0.5095332
##   3         150   0.7691207  0.5119779
##
## Tuning parameter 'nu' was held constant at a value of 0.1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were iter = 150, maxdepth = 3 and nu
##  = 0.1.
```

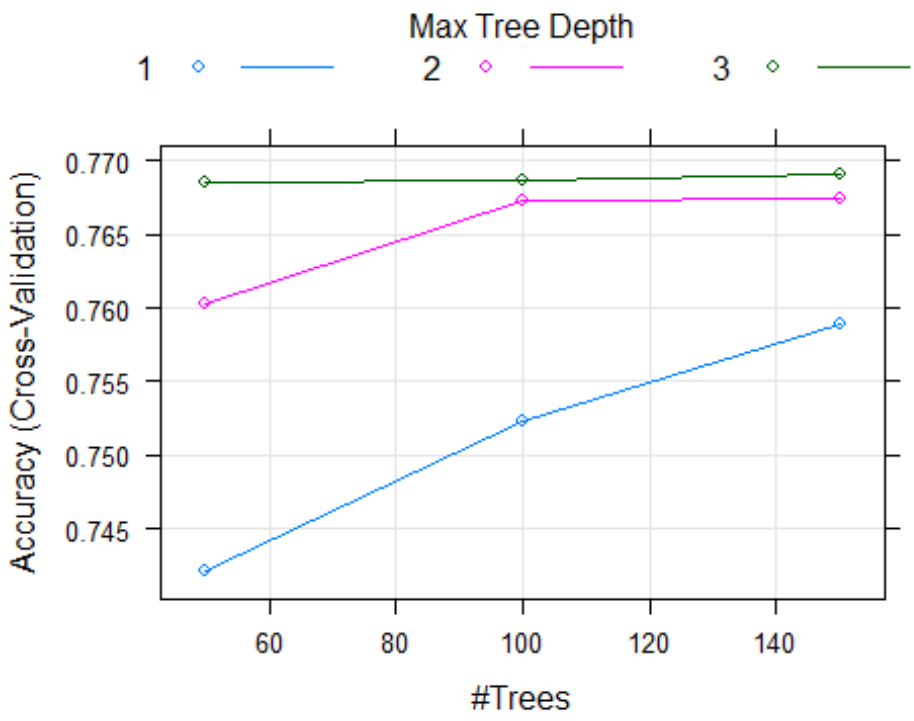


Ilustración 29. Adaboost. Mejor ajuste

Como se puede comprobar en el grafico anterior, con 3 iteraciones, conseguimos una mayor precisión (0.767).

También podemos comprobar como el paquete "caret" nos indica cuales son los mejores parámetros que debemos utilizar para obtener la mejor precisión.

Sin embargo, realizando pruebas más exhaustivas y utilizando la fuerza bruta con el paquete de "caret" y jugando con los valores de los argumentos necesarios en Adaboost, hemos obtenido mejores resultados.

Los parámetros con los que hemos utilizado la fuerza bruta ha sido con los siguientes:

- iter: número de iteraciones de refuerzo para realizar. Se ha iterado desde 1 hasta 200 de diez en diez.

- maxdepth: Profundidad (complejidad) del árbol. Se ha iterado desde 1 hasta 10 de uno en uno.
- nu: parámetro de contracción. Se ha iterado desde 0.1 a 0.4 de 0.1 en 0.1.

Con estas iteraciones se ha conseguido que, para construir el mejor modelo, se necesitan los siguientes parámetros:

- iter=150
- maxdepth=3
- nu=0.1

Con estos parámetros (número de iteraciones, máxima profundidad y nu -parámetro para truncar-) construimos nuestro modelo:

```
## Call:
## ada(outcome ~ ., data = train, iter = 150, nu = 0.1, control =
rpart.control(maxdepth = 3))
##
## Loss: exponential Method: discrete   Iteration: 150
##
## Final Confusion Matrix for Data:
##           Final Prediction
## True value    F    V
##          F 1318  662
##          V  440 2470
##
## Train Error: 0.225
##
## Out-Of-Bag Error:  0.224  iteration= 91
##
## Additional Estimates of number of iterations:
##
## train.err1 train.kap1
##        117        117
```

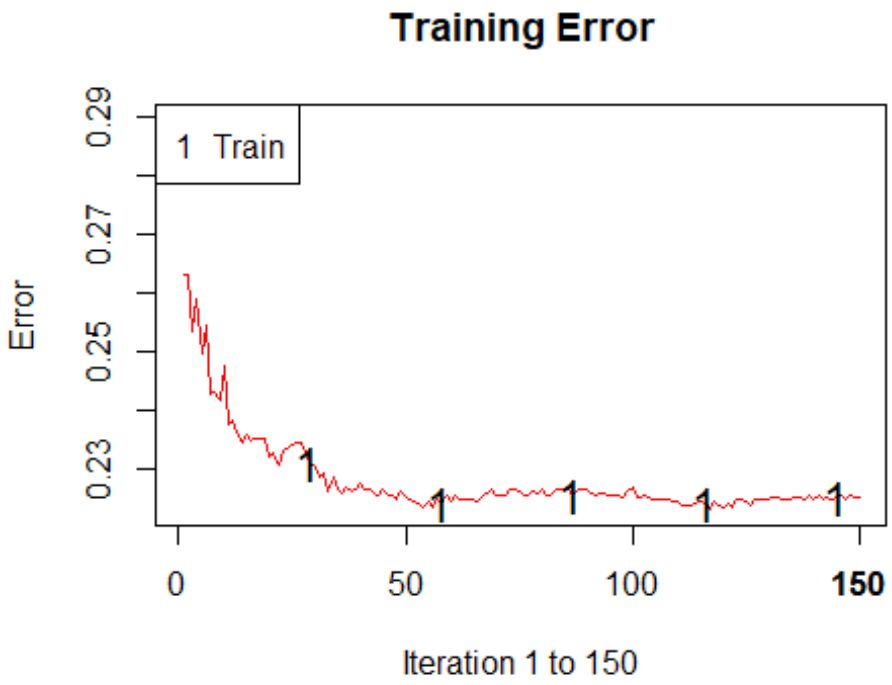


Ilustración 30. Adaboost. Error-Iteración

Vemos en la ilustración anterior que el menor error de OOB se obtiene alcanzando la iteración 150.

Después de construir el modelo con los datos de entrenamiento y habiendo predicho el modelo utilizando los datos de prueba, obtenemos la matriz de confusión:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  557  187
##          V  292 1060
##
##                Accuracy : 0.7715
##                  95% CI : (0.7529, 0.7893)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                    Kappa : 0.5163
##  Mcnemar's Test P-Value : 2.015e-06
##
##             Sensitivity : 0.6561
##             Specificity : 0.8500
##          Pos Pred Value : 0.7487
##          Neg Pred Value : 0.7840
##              Prevalence : 0.4051
##          Detection Rate : 0.2657
##    Detection Prevalence : 0.3550
##       Balanced Accuracy : 0.7531
##
##        'Positive' Class : F
##
```

Eliminando los predictores "sex" y "cause"

Como ya hemos realizado anteriormente, ahora vamos a construir un modelo obviando los predictores de "sex" y "cause". Para ello vamos a utilizar los parámetros de la rejilla de búsqueda que hemos utilizado para construir el modelo de adaboost con todos los predictores.

El resultado obtenido al construir el modelo con los datos de entrenamiento ha sido el siguiente:

```
## Call:
## ada(outcome ~ age + ec + eye + motor + verbal + pupils + phm +
##     sah + oblt + mdls + hmt, data = train, iter = 150, nu = 0.1,
##     control = rpart.control(maxdepth = 3))
##
## Loss: exponential Method: discrete   Iteration: 150
##
## Final Confusion Matrix for Data:
##           Final Prediction
## True value    F    V
##          F 1322  658
##          V  445 2465
##
## Train Error: 0.226
##
## Out-Of-Bag Error:  0.224  iteration= 91
##
```

```
## Additional Estimates of number of iterations:
##
## train.err1 train.kap1
##         56         129
```

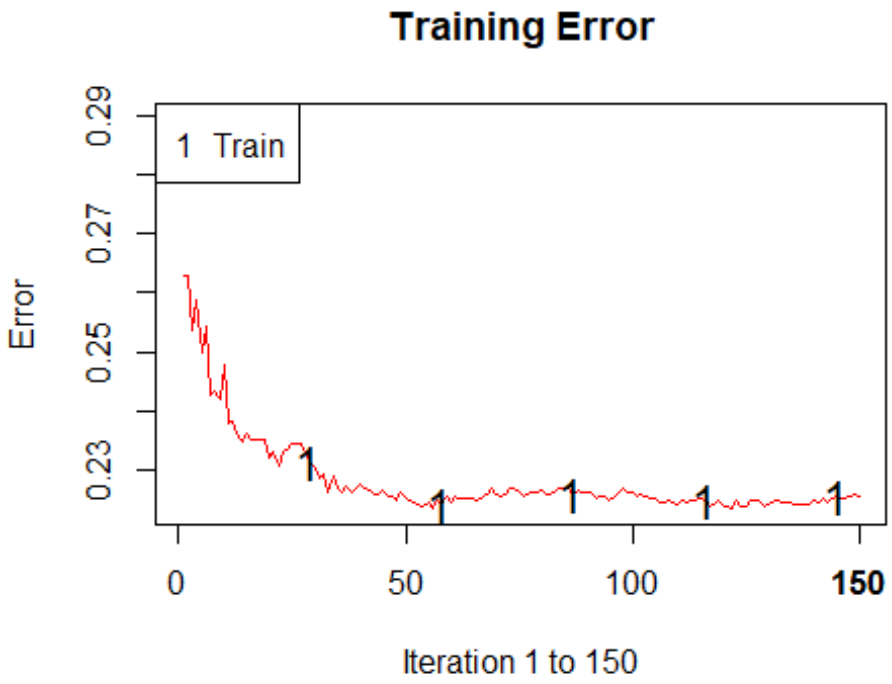


Ilustración 31. Adaboost. Error-Iteración II

Vemos una vez más como el error se reduce al converger con la iteración 150. Esto significa que será la iteración que caret considera que ayuda al modelo a obtener la mejor precisión.

Después de construir el modelo con los datos de entrenamiento y habiendo predicho el modelo con los datos de prueba, obtenemos la matriz de confusión:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  560  187
```

```
##          V  289 1060
##
##                Accuracy : 0.7729
##                  95% CI : (0.7544, 0.7907)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5196
##  Mcnemar's Test P-Value : 3.669e-06
##
##             Sensitivity : 0.6596
##             Specificity : 0.8500
##          Pos Pred Value : 0.7497
##          Neg Pred Value : 0.7858
##              Prevalence : 0.4051
##          Detection Rate : 0.2672
##    Detection Prevalence : 0.3564
##       Balanced Accuracy : 0.7548
##
##        'Positive' Class : F
##
```

Comparando el área parcial bajo la curva

A continuación, podemos observar la curva de ROC con los valores de pAUC para el modelo de Adaboost.

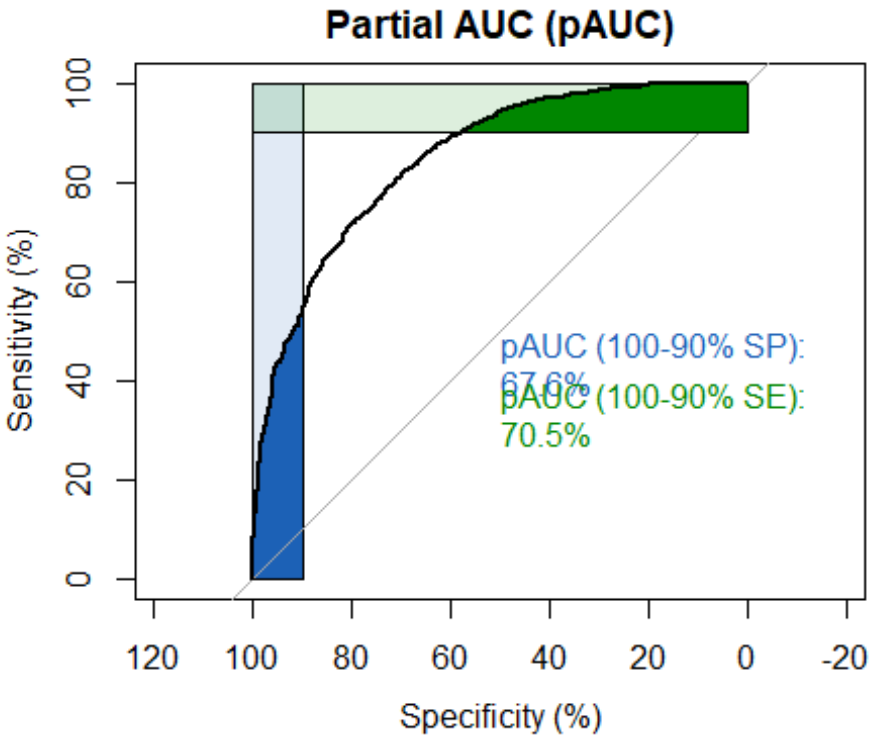


Ilustración 32. Adaboost. pAUC

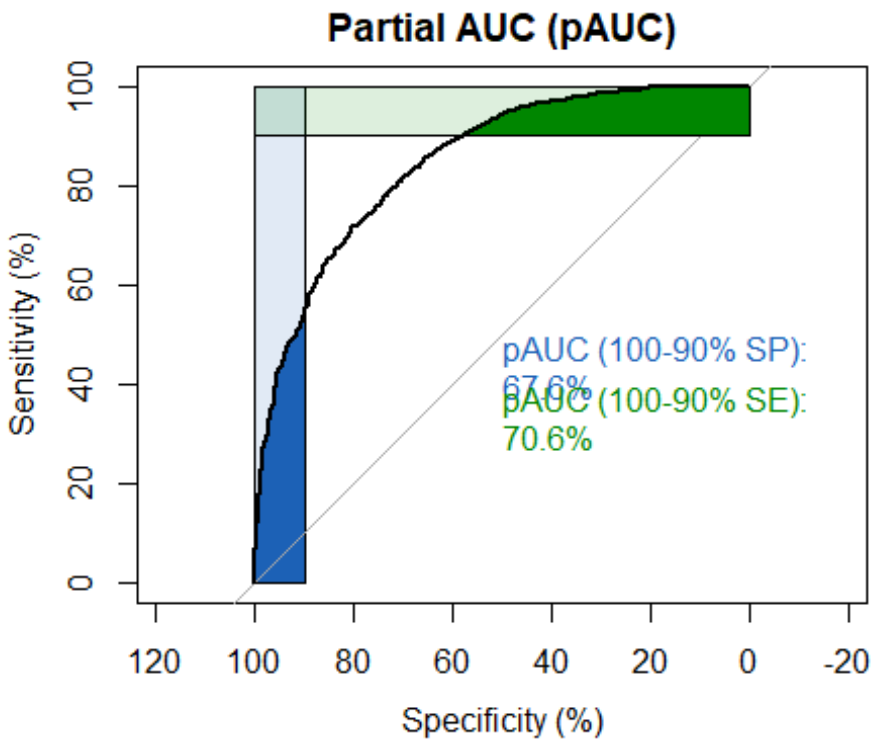


Ilustración 33.Adaboost. pAUC II

Si tomamos un rango concreto de Especificidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 67,6% (cuando utilizamos todas las variables) y un 67,6% (cuando eliminamos el sexo y la causa) de casos negativos y que realmente son negativos, es decir seremos capaces de predecir que un paciente que vive, realmente va a vivir. Vemos que ambos modelos obtienen el mismo resultado.

Si tomamos un rango concreto de sensibilidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 70,5% (cuando utilizamos todas las variables) y un 70,6% (cuando eliminamos el sexo y la causa) de casos positivos y que realmente son positivos, es decir seremos capaces de predecir que un paciente que fallece, realmente va a fallecer. Vemos que el modelo que excluye las variables de sexo y causa obtiene mejores resultados.

3.3.1.6 GBM

La idea general es obtener una secuencia de árboles (muy) simples, donde cada árbol sucesivo se construye con los residuos de predicción del árbol anterior.

Una vez generado los arboles uno a uno, se suman las predicciones de los árboles individuales:

$$D(x) = dtree1(x) + dtree2(x) + \ldots$$

El siguiente árbol de decisiones -dtree3- intenta reducir la diferencia entre la función objetivo f(x) y la predicción del conjunto actual al reconstruir el residuo (dtree1 +dtree2).

Además, el siguiente árbol -dtree3- en el conjunto debe complementarse bien con los árboles existentes y minimizar el error de entrenamiento del conjunto.

$$D(x) + dtree3(x) = f(x)$$

Para acercarnos a una predicción sin errores, entrenamos un árbol para reconstruir la diferencia entre la función objetivo y las predicciones actuales de un conjunto, esta diferencia se denomina residuo:

$$R(x)= f(x) - D(x)$$

Como podemos observar, si el árbol de decisión reconstruye completamente R(x), todo el conjunto daría predicciones sin errores, es decir, predicciones exactas. Esto en la práctica nunca sucede.

Uno de los principales problemas de todos los algoritmos de aprendizaje automático es “saber cuándo detenerse”, es decir, cómo evitar que el algoritmo de aprendizaje se ajuste tanto, que probablemente no mejore la validez predictiva del modelo. Este problema también se conoce como el problema del sobreajuste (overfitting).

Para establecer el límite de estos algoritmos, tenemos en cuenta la iteración (número de árboles) en la que se consigue un menor error.

El modelo de GBM también lo construiremos con los mismos datos de entrenamiento que utilizamos en los modelos anteriores.

Usando todos los predictores

A continuación, construimos el modelo automáticamente utilizando el paquete de "caret", igual que en ejemplos anteriores se ha utilizado la validación cruzada. Utilizamos la rejilla de búsqueda y obtenemos el siguiente resultado:

```
## Stochastic Gradient Boosting
##
## 4890 samples
##   13 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7591002  0.4831250
##   1                  100      0.7715746  0.5149340
##   1                  150      0.7707566  0.5148542
##   2                   50      0.7664622  0.5048869
##   2                  100      0.7707566  0.5156113
##   2                  150      0.7687117  0.5114616
##   3                   50      0.7693252  0.5119329
##   3                  100      0.7703476  0.5152530
##   3                  150      0.7695297  0.5131239
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 100,
##  interaction.depth = 1, shrinkage = 0.1 and n.minobsinnode = 10.
```

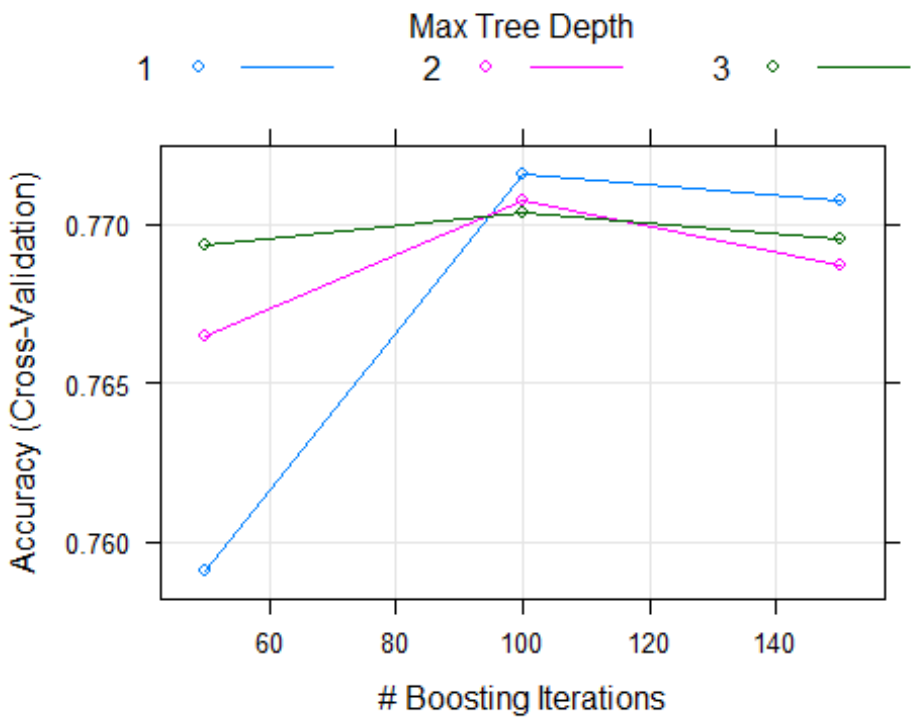


Ilustración 34. GBM. Mejor ajuste

Vemos que al construir al construir el modelo automáticamente, se utilizan los siguientes valores:

- n.trees = 100. Numero de iteraciones.
- shrinkage = 0.1. Es el ratio de aprendizaje. Simboliza la velocidad a la que se adapta el algoritmo.
- interaction.depth = 1. Este parámetro es la complejidad del árbol.
- n.minobsinnode = 10. Es el número mínimo de muestras del conjunto de entrenamiento para que un nodo comience a dividirse.

También podemos observar en el grafico es que la mejor precisión se consigue con una iteración de 100.

Vamos a construir otro modelo utilizando fuerza bruta. El objetivo es obtener una mejor precisión jugando con los parámetros ya vistos para este modelo anteriormente.

Para ello:

- Iteramos la variable n.trees de 1 a 200 aumentando 10.
- Iteramos la variable interaction.depth de 1 a 5 aumentando 1
- Iteramos la variable shrinkage de 0.1 a 0.4 aumentando 0.1.
- Iteramos la variable n.minobsinnode de 1 a 15 aumentando 1.

Utilizando esta configuración, se ha obtenido resultados similares en la precisión.

```
## Stochastic Gradient Boosting
##
## 4890 samples
##   13 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7713701  0.5140369
##
## Tuning parameter 'n.trees' was held constant at a value of 100
##  1
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
```

Después de construir el modelo, vamos a predecir utilizando los datos de test, con el objetivo de obtener la precisión de este.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  544  168
```

```
##          V  305 1079
##
##                Accuracy : 0.7743
##                  95% CI : (0.7558, 0.7921)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5194
##  Mcnemar's Test P-Value : 4.019e-10
##
##             Sensitivity : 0.6408
##             Specificity : 0.8653
##          Pos Pred Value : 0.7640
##          Neg Pred Value : 0.7796
##              Prevalence : 0.4051
##          Detection Rate : 0.2595
##    Detection Prevalence : 0.3397
##       Balanced Accuracy : 0.7530
##
##        'Positive' Class : F
##
```

Vemos que este modelo es bueno para predecir a los verdaderos negativos. Dicho de otra forma, es un modelo bastante bueno para predecir los pacientes que van a vivir.

Eliminando los predictores "sex" y "cause"

Una vez más, lo primero que realizaremos será el entrenamiento del modelo usando los datos de entrenamiento. Este entrenamiento se realizará con los parámetros obtenidos anteriormente usando la rejilla de búsqueda. El resultado obtenido es el siguiente:

```
## Stochastic Gradient Boosting
##
## 4890 samples
##   11 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7713701  0.5140369
##
## Tuning parameter 'n.trees' was held constant at a value of 100
```

```
##  1
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
```

Vemos que se muestra la precisión y el parámetro de Kappa para el modelo entrenado.

Una vez que se ha entrenado el modelo, vamos a predecir usando los datos de prueba. De esta forma como en los casos anteriores obtendremos la matriz de confusión.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  544  168
##          V  305 1079
##
##                Accuracy : 0.7743
##                  95% CI : (0.7558, 0.7921)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5194
##  Mcnemar's Test P-Value : 4.019e-10
##
##             Sensitivity : 0.6408
##             Specificity : 0.8653
##          Pos Pred Value : 0.7640
##          Neg Pred Value : 0.7796
##              Prevalence : 0.4051
##          Detection Rate : 0.2595
##    Detection Prevalence : 0.3397
##       Balanced Accuracy : 0.7530
##
##        'Positive' Class : F
##
```

Los resultados siguen siendo buenos aun a pesar de la exclusión de las variables que menos valor nos aporta en la construcción del modelo. Tenemos una buena precisión y una gran especificidad, que nos permite detectar verdaderos negativos.

Comparando el área parcial bajo la curva

A continuación, podemos observar la curva de ROC con los valores de pAUC para el modelo de GBM.

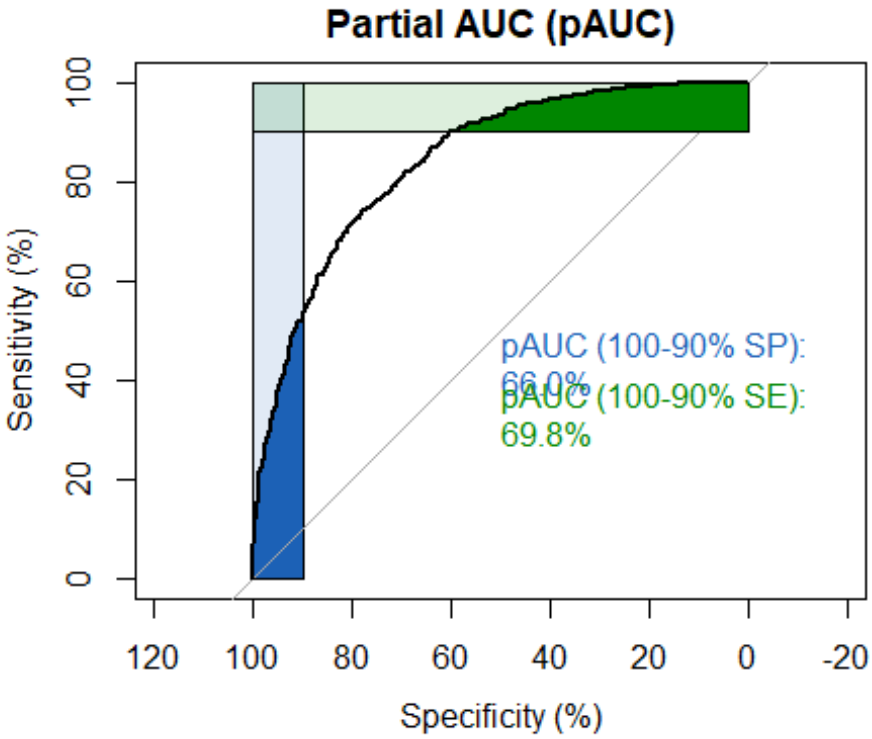


Ilustración 35. GBM. pAUC

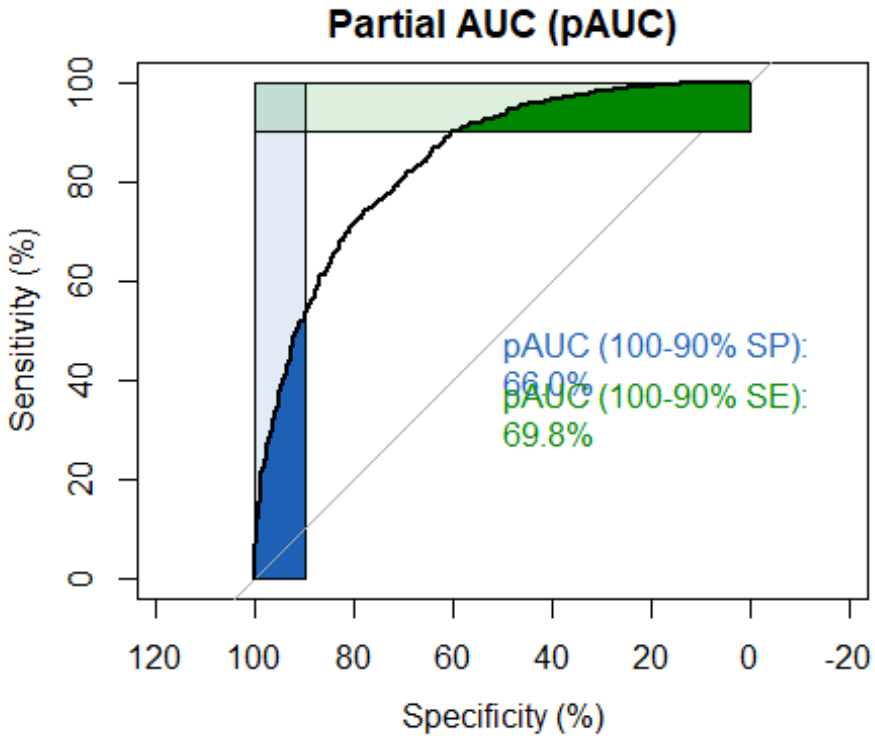


Ilustración 36.GBM. pAUC II

Si tomamos un rango concreto de Especificidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 66% (cuando utilizamos todas las variables) y un 66% (cuando eliminamos el sexo y la causa) de casos negativos y que realmente son negativos, es decir seremos capaces de predecir que un paciente que vive, realmente va a vivir. Vemos que ambos modelos han obtenido los mismos resultados.

Si tomamos un rango concreto de sensibilidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 69,8% (cuando utilizamos todas las variables) y un 69,8% (cuando eliminamos el sexo y la causa) de casos positivos y que realmente son positivos, es decir seremos capaces de predecir que un paciente que fallece, realmente va a fallecer. Vemos que ambos modelos han obtenido los mismos resultados.

3.3.1.7 XGBoost

XGBoost es la abreviatura de 'Extreme Gradient Boosting'. Está basado en el modelo original GBM (ya estudiado anteriormente).

A continuación, se muestran algunas de las ventajas de XGBOOST frente GBM:

- Regularización: La implementación de GBM no tiene regularización, lo que provoca que XGBOOST reduzca el sobreajuste.
- Procesamiento paralelo: XGBoost implementa el procesamiento paralelo y es sorprendentemente más rápido en comparación con GBM.
- Mayor flexibilidad: XGBoost permite a los usuarios definir objetivos de optimización personalizados y criterios de evaluación.
- Manejo de valores perdidos: XGBoost tiene una rutina incorporada para manejar los valores perdidos.
- Poda: El algoritmo GBM dejaría de dividir un nodo cuando encuentre una pérdida negativa en una división. Por lo tanto, XGBoost es más codicioso. XGBoost por otro lado hace divisiones hasta la "**max_depth"** especificada y luego comienza a podar el árbol hacia atrás y a eliminar divisiones de las cuales no hay ganancia positiva.
- Incorpora Cross-Validation: XGBoost permite al usuario ejecutar una validación cruzada en cada iteración del proceso de "boosting" y, por lo tanto, es fácil obtener el número óptimo de iteraciones de "boosting" en una sola ejecución. Con GBM tenemos que ejecutar una búsqueda en cuadrícula y solo se pueden probar valores limitados.

Igual que con los modelos construidos con anterioridad, vamos a volver a utilizar "caret" para la validación cruzada, además utilizaremos los siguientes parámetros de ajuste para XGBoost que nos ofrece el paquete "xgbtree":

- **nrounds:** Es el número de iteraciones que el modelo ejecuta antes de que se detenga. Con el valor más alto del modelo "nrounds" tomará más tiempo y viceversa.
- "**max_depth" (Profundidad máxima del árbol):** Un valor alto de "max_depth" creará árboles más profundos, es decir, creará un modelo más complejo. Un valor más alto de "max_depth" puede crear un ajuste excesivo y un valor inferior de "max_depth" puede crear un ajuste insuficiente. Todo depende de los datos disponibles.
- **eta (Shrinkage):** Con un valor alto, el modelo funcionará más rápido y viceversa. Con un "eta" más alto y un menor "round", el modelo tardará menos tiempo en ejecutarse. Con un "eta" menor y un modelo de "nround" más alto tomará más tiempo.
- **gamma (reducción de pérdida mínima):** Para crear una partición adicional en un nodo hoja del árbol Se requiere una reducción de pérdida mínima.
- **colsample_bytree (proporción de columnas de la submuestra):** Se escogen aleatoriamente columnas de entre todas las columnas o variables durante el proceso de construcción del árbol. Es similar al parámetro de "mtry" de los "Random Forest". Un valor alto puede crear un ajuste excesivo (overfitting) y un valor pequeño puede crear un ajuste insuficiente.
- **min_child_weight (suma mínima del peso de la instancia):** Si el paso de la partición del árbol da como resultado un nodo hoja con la suma del peso de la instancia menor que "min_child_weight", entonces el proceso de construcción dejará de particionar más.

Usando todos los predictores

Teniendo en cuenta los parámetros anteriores, vamos a construir el modelo de XGBoost utilizando la rejilla de búsqueda. El objetivo es obtener la mejor precisión jugando con los parámetros ya vistos para este modelo.

Para ello:

- Iteramos la variable Nrounds de 1 a 100 aumentando 10.
- Iteramos la variable eta de 0.1 a 0.3 aumentando 0.1
- Iteramos la variable Colsample_bytree de 0.4 a 0.8 aumentando 0.1
- Iteramos la variable Max_depth de 1 a 3 aumentando 1.
- Iteramos la variable Gamma de 1 a 2 aumentando 1.
- Iteramos la variable Min_child_weigh de 1 a 4 aumentando 1.
- Iteramos la variable Subsample de 1 a 2 aumentando 1.

Obtenemos el siguiente resultado:

```
eXtreme Gradient Boosting

4890 samples
  13 predictor
   2 classes: 'F', 'V'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
Resampling results across tuning parameters:

  eta  max_depth  colsample_bytree  subsample  nrounds  Accuracy   Kappa
  0.3  1          0.6               0.50        50      0.7683027  0.5094535
  0.3  1          0.6               0.50       100      0.7707566  0.5141968
  0.3  1          0.6               0.50       150      0.7709611  0.5145441
  0.3  1          0.6               0.75        50      0.7713701  0.5153801
  0.3  1          0.6               0.75       100      0.7715746  0.5162874
  0.3  1          0.6               0.75       150      0.7707566  0.5143055
  0.3  1          0.6               1.00        50      0.7719836  0.5158416
  0.3  1          0.6               1.00       100      0.7717791  0.5158738
  0.3  1          0.6               1.00       150      0.7721881  0.5166627
  0.3  1          0.8               0.50        50      0.7697342  0.5125936
  0.3  1          0.8               0.50       100      0.7707566  0.5143238
  0.3  1          0.8               0.50       150      0.7717791  0.5159266
  0.3  1          0.8               0.75        50      0.7691207  0.5108955
  0.3  1          0.8               0.75       100      0.7732106  0.5192796
  0.3  1          0.8               0.75       150      0.7736196  0.5199844
  0.3  1          0.8               1.00        50      0.7703476  0.5128821
  0.3  1          0.8               1.00       100      0.7728016  0.5179266
Tuning parameter 'gamma' was held constant at a value of 1
Tuning parameter 'subsample' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
```

```
The final values used for the model were nrounds = 71, max_depth = 1, eta = 0.2,
gamma = 1, colsample_bytree = 0.4, min_child_weight = 4
 and subsample = 1.
```

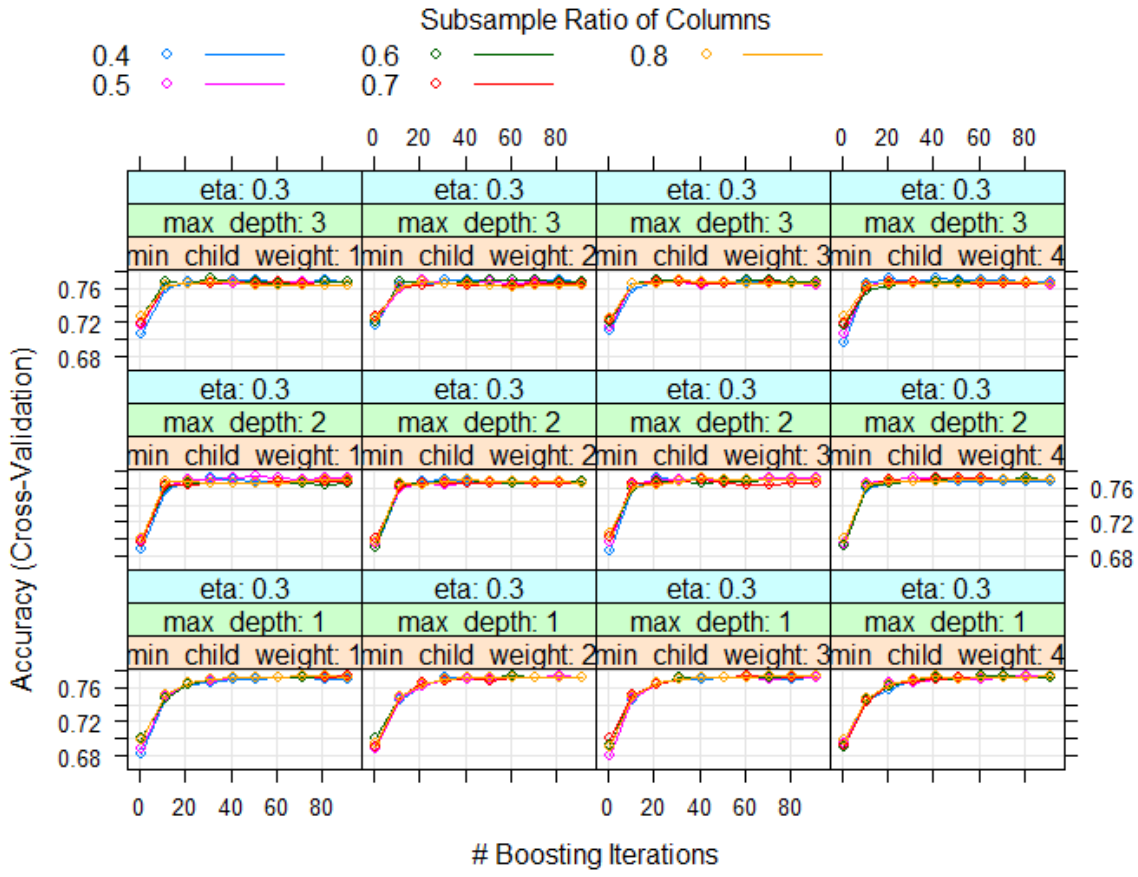


Ilustración 37.XGBoost. Mejor Ajuste

En la ilustración anterior podemos observar todas las pruebas que se han realizado ajustando los hiperparámetros mediante el uso de la rejilla de búsqueda de caret.

Como podemos comprobar, los valores para los parametros que generan el modelo optimo son los siguientes:

- Nrounds =71
- Max_depth = 1

- Eta = 0.2
- Gamma = 1
- Colsample_bytree = 0.4
- Min_child_weight = 4
- Subsample = 1

Una vez que tenemos los parámetros para construir el modelo con mejor precisión, construimos dicho modelo, a continuación, se puede observar los resultados.

```
## eXtreme Gradient Boosting
##
## 4890 samples
##   13 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7713701  0.5141202
##
## Tuning parameter 'nrounds' was held constant at a value of 71
##  0.4
## Tuning parameter 'min_child_weight' was held constant at a value of
##  4
## Tuning parameter 'subsample' was held constant at a value of 1
```

Una vez que tenemos el modelo construido, utilizaremos el conjunto de datos de prueba para predecir y obtener así la matriz de confusión que se muestra a continuación:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  546  171
##          V  303 1076
##
##                Accuracy : 0.7739
##                  95% CI : (0.7553, 0.7916)
##     No Information Rate : 0.5949
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5189
##  Mcnemar's Test P-Value : 1.776e-09
##
##             Sensitivity : 0.6431
##             Specificity : 0.8629
##          Pos Pred Value : 0.7615
##          Neg Pred Value : 0.7803
##              Prevalence : 0.4051
##          Detection Rate : 0.2605
##    Detection Prevalence : 0.3421
##       Balanced Accuracy : 0.7530
##
##        'Positive' Class : F
##
```

Eliminando los predictores "sex" y "cause"

En este caso y como se han hecho en modelos anteriores, eliminaremos las variables de "sex" y "cause". Usaremos los hiperparámetros obtenidos anteriormente para el modelo de XGBoost para construir este modelo. A continuación, se muestra el resultado obtenido:

```
## eXtreme Gradient Boosting
##
## 4890 samples
##   11 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7705521  0.5123433
##
## Tuning parameter 'nrounds' was held constant at a value of 71
##  0.4
## Tuning parameter 'min_child_weight' was held constant at a value of
##  4
## Tuning parameter 'subsample' was held constant at a value of 1
```

Una vez construido el modelo, vamos a pasar a predecirlo y a obtener la matriz de confusion para visualizer los resultados.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  541  171
##          V  308 1076
##
##                Accuracy : 0.7715
##                  95% CI : (0.7529, 0.7893)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5133
##  Mcnemar's Test P-Value : 5.165e-10
##
##             Sensitivity : 0.6372
##             Specificity : 0.8629
##          Pos Pred Value : 0.7598
##          Neg Pred Value : 0.7775
##              Prevalence : 0.4051
##          Detection Rate : 0.2581
##    Detection Prevalence : 0.3397
##       Balanced Accuracy : 0.7500
##
##        'Positive' Class : F
##
```

Volvemos que la precisión ha empeorado al eliminar los predictores. Sin embargo, la especificidad sigue igual. Es decir, el modelo sigue teniendo resultados buenos a la hora de predecir los verdaderos negativos.

Comparando el área parcial bajo la curva

A continuación, podemos observar la curva de ROC con los valores de pAUC para el modelo de XGBoost.

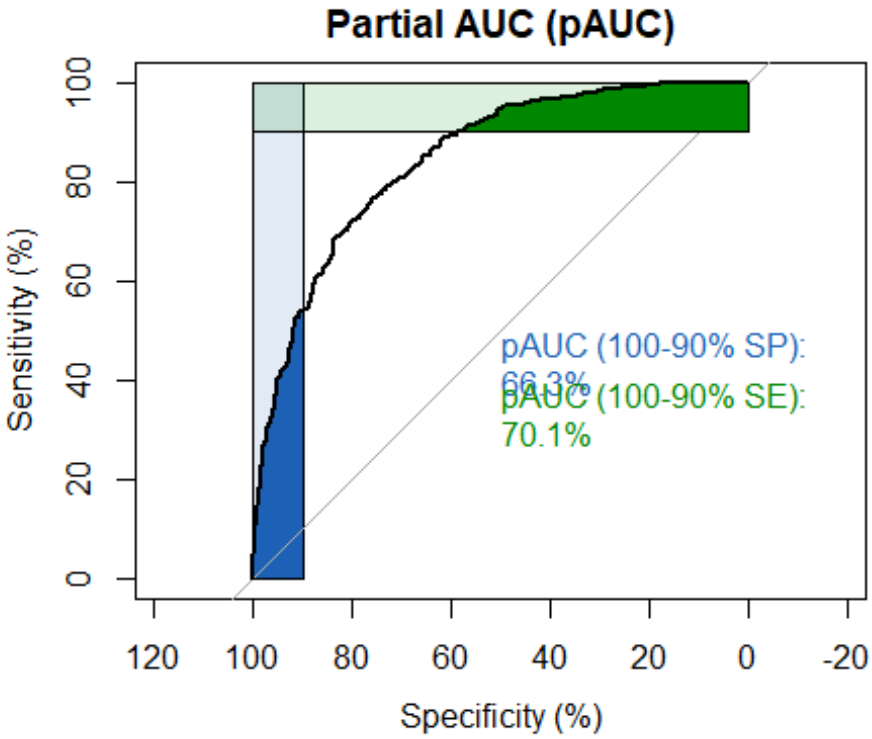


Ilustración 38. XGBoost. pAUC

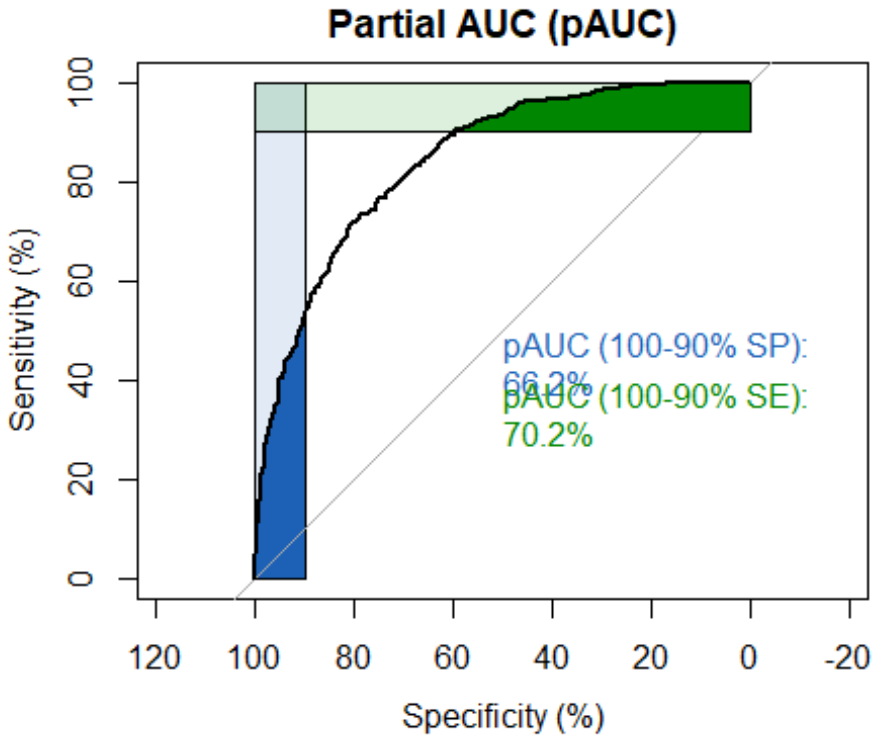


Ilustración 39.XGBoost. pAUC II

Si tomamos un rango concreto de Especificidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 66,3% (cuando utilizamos todas las variables) y un 66,2% (cuando eliminamos el sexo y la causa) de casos negativos y que realmente son negativos, es decir seremos capaces de predecir que un paciente que vive, realmente va a vivir. Vemos que el modelo completo ha obtenido un ligero mejor resultado.

Si tomamos un rango concreto de sensibilidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 70,1% (cuando utilizamos todas las variables) y un 70,2% (cuando eliminamos el sexo y la causa) de casos positivos y que realmente son positivos, es decir seremos capaces de predecir que un paciente que fallece, realmente va a fallecer. Vemos que el modelo que excluye las variables de sexo y causa ha obtenido ligeramente un mejor resultado.

3.3.1.8 Redes neuronales

Las redes neuronales son sistemas computacionales de aprendizaje basados en el funcionamiento de las redes neuronales biológicas presentes en el cerebro de los animales.

Estas redes están construidas a partir de una serie de nodos llamados "neuronas" que se organizan en forma de capas. Existe una capa de entrada, una capa de salida y al menos una capa intermedia o capa "oculta".

La capa de entrada debe tener tantas neuronas como variables de entrada tenga el sistema que se va a modelar. La capa de salida debe tener tantas neuronas como variables que estemos intentando predecir, en nuestro caso solo una.

En cuanto a las capas ocultas, sus neuronas realizan transformaciones no lineales sobre la información que los atraviesa mediante una función de activación. Las salidas de estas funciones de activación suelen variar entre 0 y 1. Estas neuronas se activan o no dependiendo de si la información que entra en estas neuronas Si la información que entra en esta neurona supera un valor umbral.

Las tareas de estas neuronas "ocultas" es realizar una suma ponderada de todas las entradas que tiene y aplicarle una función de activación para posteriormente pasar este valor a las neuronas de la siguiente capa, que a su vez repetirán el proceso.

En primer lugar, antes de comenzar con la construcción del modelo de redes neuronales, es necesario normalizar los datos.

Para poder obtener las mejores predicciones (precisión) con nuestro modelo de redes neuronales, deberemos tener en cuenta aspectos como:

- *Numero de capas ocultas*.
- *Numero de neuronas por capa oculta:* Si se usa una cantidad inadecuada de neuronas, la red no podrá modelar datos complejos y el ajuste resultante será deficiente. Si se utilizan demasiadas neuronas, el tiempo de entrenamiento puede ser excesivamente

largo y, lo que es peor, la red puede sobre ajustar los datos. No hay una regla práctica para elegir el número de neuronas, pero podemos tener en cuenta lo siguiente:

  - N es el número de neuronas ocultas.
  - N = 2/3 del tamaño de la capa de entrada, más el tamaño de la capa de salida.
  - N < dos veces el tamaño de la capa de entrada.

- *Función de activación de las capas ocultas:* Cambiar la función de activación puede ser un factor decisivo. Se pueden probar distintas funciones de activación: sigmoide, tanh y unidades lineales rectificadas.
- *Función de activación de las capas de salida:* Para una sola capa la elección de la función de activación para la capa de salida dependerá de la tarea que realizaremos con la red (es decir, categorización o regresión). Sin embargo, en redes multicapas, generalmente es deseable que las capas ocultas tengan funciones de activación no lineales (por ejemplo, sigmoide, logística o tanh).

Usando todas las variables

Para construir el modelo, una vez nos hemos ayudado con el paquete "Caret" y su validación cruzada con 10 iteraciones.

En la construcción del modelo, es necesario probar distintos parámetros, concretamente:

- actfun: es la función de activación.
- nhid: es el número de capas ocultas.

Estos valores, se han ido probando mediante "fuerza bruta" hasta obtener los mejores valores de precisión. El parámetro de "actfun" ira variando entre todas las funciones de activación. El parámetro de "nhid" se ha ido variando desde 0 hasta 50 en 1 unidad.

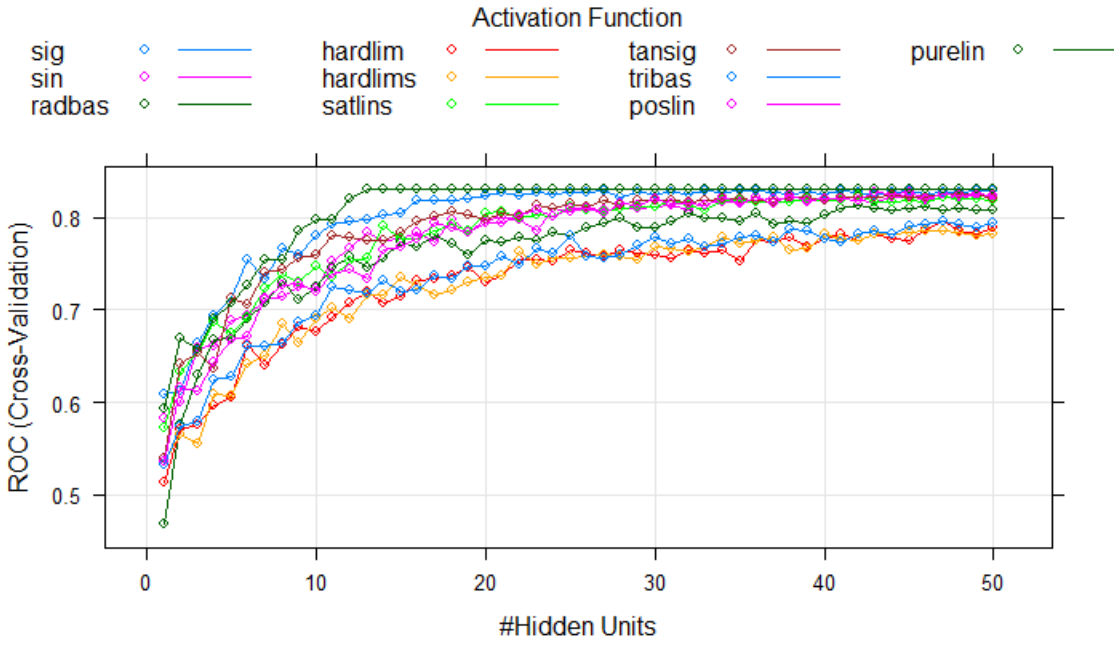


Ilustración 40. Redes Neuronales. Mejor ajuste

Como se puede observar en el grafico anterior, la función de activación que nos proporciona una mayor precisión es "purelin", que es una función de activación puramente lineal. Además, se usarán 14 capas ocultas (nhid=14).

```
## Extreme Learning Machine
##
## 4890 samples
##   13 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   ROC       Sens       Spec
##   0.832679  0.6388889  0.8549828
##
## Tuning parameter 'nhid' was held constant at a value of 14
##
## Tuning parameter 'actfun' was held constant at a value of purelin
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  558  200
##          V  291 1047
##
##                Accuracy : 0.7657
##                  95% CI : (0.747, 0.7837)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5055
##  Mcnemar's Test P-Value : 4.873e-05
##
##             Sensitivity : 0.6572
##             Specificity : 0.8396
##          Pos Pred Value : 0.7361
##          Neg Pred Value : 0.7825
##              Prevalence : 0.4051
##          Detection Rate : 0.2662
##    Detection Prevalence : 0.3616
##       Balanced Accuracy : 0.7484
##
##        'Positive' Class : F
##
```

Eliminamos las variables más correladas: Verbal

Ahora vamos a probar a eliminar las variables más correlacionadas (ya que podríamos considerar que contienen información redundante) puesto que quizá, mediante esta eliminación, podamos conseguir un mejor resultado en la precisión del modelo.

Como se puede comprobar en la Ilustración 11, las variables más correladas son “Verbal” y “Eye”. Teniendo en cuenta esto, vamos a eliminar la variable “Verbal”.

```
## Extreme Learning Machine
##
## 4890 samples
##   12 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
```

```
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8302102  0.6262626  0.8621993
##
## Tuning parameter 'nhid' was held constant at a value of 14
##
## Tuning parameter 'actfun' was held constant at a value of purelin
```

A continuación, se muestra la matriz de correlación una vez que hemos construido el modelo sin la variable "verbal".

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  546  196
##          V  303 1051
##
##                Accuracy : 0.7619
##                  95% CI : (0.7431, 0.78)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4959
##  Mcnemar's Test P-Value : 2.083e-06
##
##             Sensitivity : 0.6431
##             Specificity : 0.8428
##          Pos Pred Value : 0.7358
##          Neg Pred Value : 0.7762
##              Prevalence : 0.4051
##          Detection Rate : 0.2605
##    Detection Prevalence : 0.3540
##       Balanced Accuracy : 0.7430
##
##        'Positive' Class : F
##

##
```

Como podemos observar, con las redes neuronales también se han obtenido buenos resultados respecto a otros modelos. Este modelo presenta también una gran especificidad, indicándonos que es un modelo bastante bueno en la detección de verdaderos negativos.

Comparando el área parcial bajo la curva

A continuación, podemos observar la curva de ROC con los valores de pAUC para el modelo de Redes Neuronales.

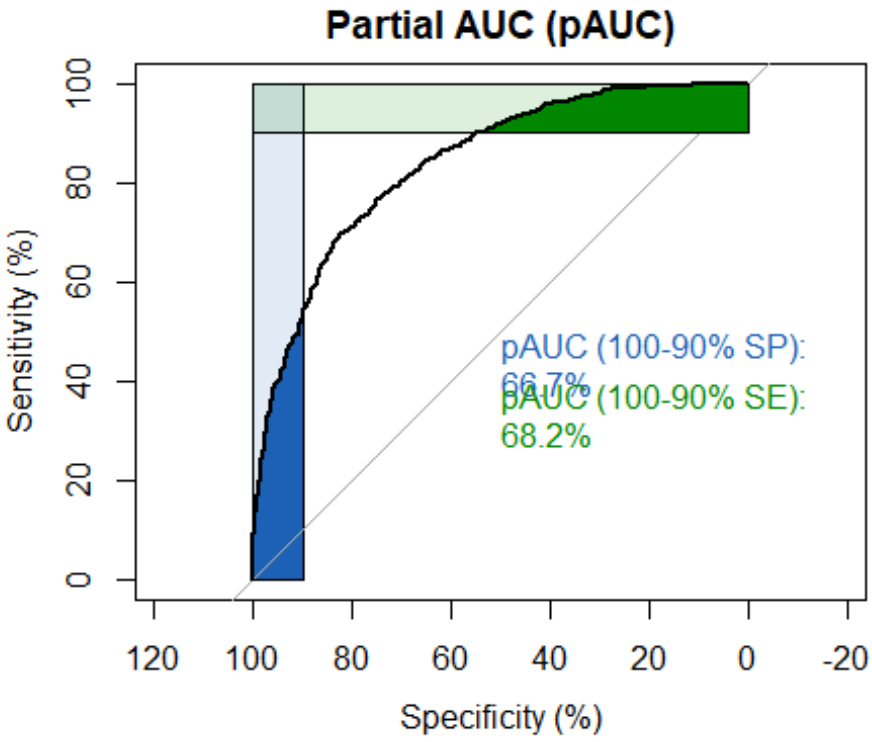


Ilustración 41. Redes Neuronales. pAUC

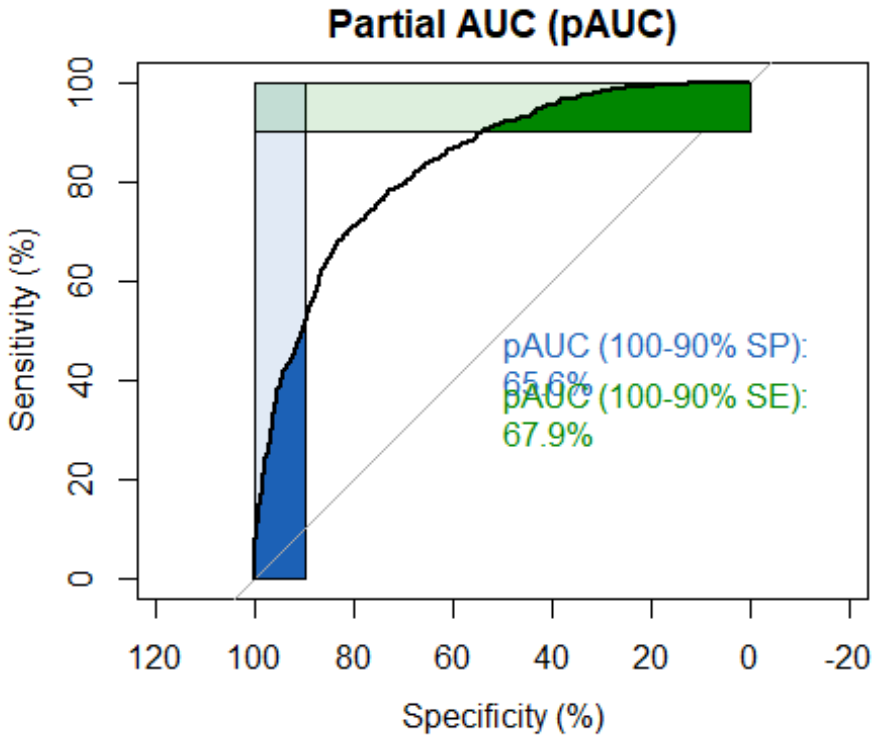


Ilustración 42. Redes Neuronales. Verbal. pAUC

Si tomamos un rango concreto de Especificidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 66,7% (cuando utilizamos todas las variables) y un 65,6% (cuando eliminamos el sexo y la causa) de casos negativos y que realmente son negativos, es decir seremos capaces de predecir que un paciente que vive, realmente va a vivir. Vemos que el modelo completo ha obtenido un mejor resultado. Podemos observar que, respecto a modelos anteriores, las redes neuronales son más sensibles a la eliminación de variables.

Si tomamos un rango concreto de sensibilidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 68,2% (cuando utilizamos todas las variables) y un 67,9% (cuando eliminamos el sexo y la causa) de casos positivos y que realmente son positivos, es decir seremos capaces de predecir que un paciente que fallece, realmente va a fallecer. Vemos que el modelo completo ha obtenido mejores resultados.

3.3.1.9 Combinación de modelos

La combinación de modelos es una técnica de agrupación de dos o más algoritmos similares o diferentes. Esta técnica se utiliza para generar un sistema más robusto que incorpore las predicciones de todos los modelos combinados. Una vez que tenemos un gran número de predicciones podremos tomar una decisión final. Esta decisión final será más robusta y precisa. Para ello utilizaremos caretEnsemble.

Utilizando todos los modelos

En primer lugar, se ha creado un listado de los modelos que se van a comparar y son los siguientes: glm (modelo lineal generalizado), nb (Naïves Bayes), rf (Random Forest), ada (AdaBoost), gbm (aumento de gradiente), xgboost (aumento de gradiente extremo), blackboost (aumento de gradiente con árboles de regresión), parRF (Random Forest Paralelo), knn (redes neuronales).

El siguiente resultado se obtiene usando caretList().

```
##
## Call:
## summary.resamples(object = results)
##
## Models: glm, nb, rf, ada, gbm, xgbTree, blackboost, parRF, knn, glmnet
## Number of resamples: 10
##
## Accuracy
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## glm        0.7464213 0.7592025 0.7689162 0.7685072 0.7755624 0.7893661
## nb         0.7075665 0.7321063 0.7361963 0.7400818 0.7397751 0.7893661
## rf         0.7505112 0.7530675 0.7627812 0.7635992 0.7663599 0.7934560
## ada        0.7484663 0.7586912 0.7678937 0.7701431 0.7745399 0.8036810
## gbm        0.7566462 0.7653374 0.7699387 0.7730061 0.7796524 0.7995910
## xgbTree    0.7566462 0.7689162 0.7740286 0.7748466 0.7827198 0.7955010
## blackboost 0.7505112 0.7617587 0.7689162 0.7717791 0.7822086 0.7934560
## parRF      0.7484663 0.7551125 0.7648262 0.7640082 0.7684049 0.7934560
## knn        0.7259714 0.7300613 0.7341513 0.7368098 0.7438650 0.7505112
## glmnet     0.7464213 0.7607362 0.7678937 0.7693252 0.7745399 0.7914110
##            NA's
## glm           0
## nb            0
## rf            0
## ada           0
```

```
## gbm           0
## xgbTree       0
## blackboost    0
## parRF         0
## knn           0
## glmnet        0
##
## Kappa
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## glm        0.4571628 0.4853150 0.5158085 0.5094980 0.5232337 0.5546566
## nb         0.3847433 0.4447245 0.4534797 0.4573486 0.4555877 0.5582888
## rf         0.4614343 0.4712777 0.4912302 0.4931436 0.5044885 0.5522587
## ada        0.4601922 0.4904086 0.5098643 0.5120830 0.5210424 0.5818279
## gbm        0.4921587 0.4999875 0.5125801 0.5190571 0.5338249 0.5724075
## xgbTree    0.4865243 0.5108158 0.5192573 0.5226248 0.5390859 0.5622281
## blackboost 0.4650274 0.4943380 0.5106504 0.5145010 0.5374375 0.5589520
## parRF      0.4547309 0.4767476 0.4989803 0.4939240 0.5042013 0.5537636
## knn        0.4064674 0.4185384 0.4265188 0.4319418 0.4458084 0.4623371
## glmnet     0.4571628 0.4941540 0.5076133 0.5105349 0.5216302 0.5586174
##            NA's
## glm           0
## nb            0
## rf            0
## ada           0
## gbm           0
## xgbTree       0
## blackboost    0
## parRF         0
## knn           0
## glmnet        0
```

En el resultado anterior se puede observar las medidas de precisión ("*Accuracy"*) y Kappa proporcionadas por cada modelo en la combinación de estos. Siendo el que mejor precisión media consigue el xgbTree.

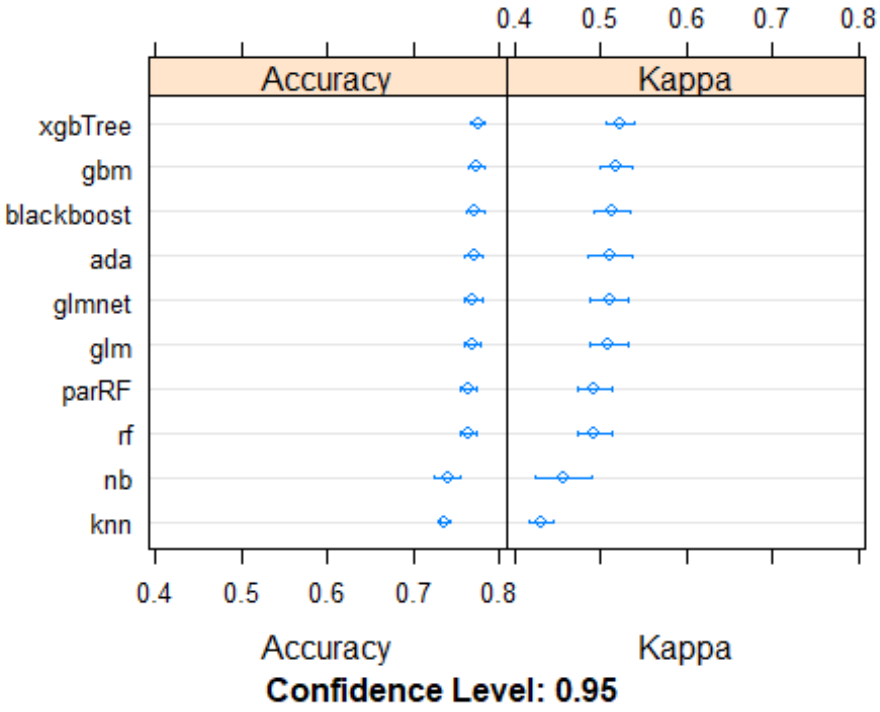


Ilustración 43. CaretEnsemble. Comparación de modelos

Como se puede comprobar en el grafico anterior, las mejores precisiones se obtienen con xgbTree (xgboost) y gbm, como ya se mostró anteriormente. Las peores se obtienen de nuevo con los modelos de Naïves Bayes y Knn (K vecinos más próximos).

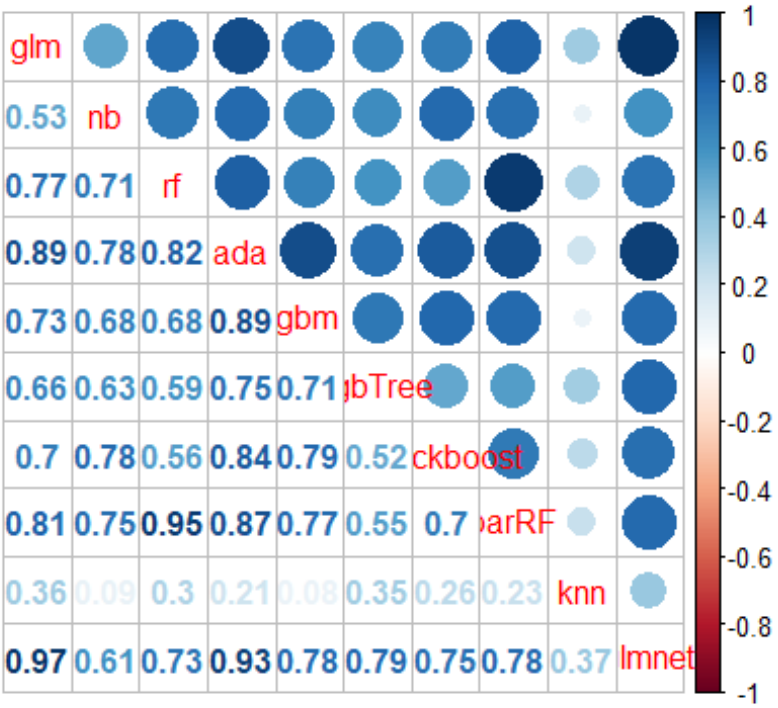


Ilustración 44.CaretEnsemble. Correlación Modelos

En la ilustración anterior podemos ver en los modelos que hemos seleccionado, existe un gran número de correlaciones bastante destacadas. Esto se debe a que existen algunos modelos que no son complementarios.

Concretamente, podemos destacar que se han utilizado dos funciones de Random Forest. RF y parRF, que poseen una alta correlación (0.95) y no aportan apenas ninguna característica nueva.

También podemos destacar que existe una gran correlación entre el modelo de glm y glmnet, siendo su correlación la más próxima a 1, con un valor de 0.97.

El modelo adaboost también tiene una gran correlación con otros modelos como glm, gbm, y glmnet.

En primer lugar, vamos a utilizar caretEnsemble, que como podemos recordar, realiza una combinación entre el modelo GLM y las predicciones de los sub-modelos.

```
## The following models were ensembled: glm, nb, rf, ada, gbm, xgbTree, blackboo
st, parRF, knn, glmnet
## They were weighted:
## 2.6283 6.3573 0.3319 0.4889 -0.2536 -1.242 -3.0928 1.2646 -1.2604 0.2029 -8.2
099
## The resulting Accuracy is: 0.7712
## The fit for each individual model on the Accuracy is:
##      method  Accuracy  AccuracySD
##         glm 0.7685072 0.014128694
##          nb 0.7400818 0.021349232
##          rf 0.7635992 0.013083673
##         ada 0.7701431 0.016337180
##         gbm 0.7730061 0.012383028
##     xgbTree 0.7748466 0.011785088
##  blackboost 0.7717791 0.014174663
##       parRF 0.7640082 0.012686997
##         knn 0.7368098 0.008890415
##      glmnet 0.7693252 0.014259639
```

Como podemos observar en el resultado anterior, el resultado en la precisión de la combinación de los modelos ha sido 0.7712. No es un resultado negativo, pero existen modelos como gbm y xgbTree que individualmente obtienen un mejor resultado.

Ahora vamos a realizar una predicción del modelo construido con los datos de test. El resultado es el siguiente:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  328 1089
##          V  521  158
##
##                Accuracy : 0.2319
##                  95% CI : (0.2139, 0.2505)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.44
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3863
```

```
##             Specificity : 0.1267
##          Pos Pred Value : 0.2315
##          Neg Pred Value : 0.2327
##              Prevalence : 0.4051
##          Detection Rate : 0.1565
##    Detection Prevalence : 0.6760
##       Balanced Accuracy : 0.2565
##
##        'Positive' Class : F
##
```

Como podemos observar en los resultados anteriores, la precisión no es buena, conseguiríamos obtener una precisión de 0.2319. Es decir, que de cada 100 pacientes podremos obtener una predicción de 23.

A continuación, se muestra la importancia de las variables en cada modelo utilizado:

```
##           overall        glm         nb        rf        ada       gbm
## cause   0.1536358  0.0000000  0.0000000  1.918434  0.0000000  0.000000
## sex     0.5243712  0.9050923  0.2084746  0.000000  0.2084746  0.000000
## sah     4.3276563  4.0358244  6.4223263  2.060381  6.4223263  1.746034
## ec      6.0939817  6.4804414  3.2005784  1.544308  3.2005784  1.821794
## hmt     6.7706915  6.7843414  5.5322575  1.847850  5.5322575  2.139092
## phm     6.7859202  7.3147435  4.1658753  1.172956  4.1658753  1.847201
## eye     8.5370367  7.3598631 14.8656127 12.924178 14.8656127 11.846577
## pupils  8.9750065  8.6200158  9.3304186  9.533392  9.3304186  9.718010
## verbal  9.3923284  6.6271770 15.5162247 13.548753 15.5162247 15.227608
## oblt    9.5647718  8.4359883  7.6156835  3.961918  7.6156835  5.474113
## mdls   12.0072974  9.2285446  6.7014626  5.060403  6.7014626  7.128575
## motor  12.2449674 10.6945162 16.8077350 19.205857 16.8077350 19.902390
## age    14.6223350 23.5134520  9.6333508 27.221571  9.6333508 23.148607
##           xgbTree blackboost     parRF        knn    glmnet
## cause   0.0000000  0.0000000  2.023410  0.0000000  0.000000
## sex     0.0258833  0.2084746  0.000000  0.2084746  0.687399
## sah     1.1159203  6.4223263  2.031542  6.4223263  6.117796
## ec      1.6351783  3.2005784  1.559661  3.2005784  9.811305
## hmt     3.0635779  5.5322575  1.918051  5.5322575 10.205130
## phm     2.5284030  4.1658753  1.315097  4.1658753 10.556740
## eye     7.9015995 14.8656127 12.858537 14.8656127  6.680039
## pupils  9.9053250  9.3304186  9.482576  9.3304186  8.586975
## verbal 16.8372971 15.5162247 14.606944 15.5162247  5.266673
## oblt    4.4536191  7.6156835  3.951069  7.6156835 14.665987
## mdls    6.4653262  6.7014626  4.909082  6.7014626 19.815022
## motor  20.9030350 16.8077350 18.164007 16.8077350  6.561309
## age    25.1648354  9.6333508 27.180024  9.6333508  1.045626
```

Vemos que la importancia de las variables, una vez más coincide con lo ya estudiado en la sección de pre-procesado de los datos. Siendo una vez más, las variables más importantes en la mayoría de los modelos son: "*age*", "*motor*" y "*mdls*" y las menos importantes son: "*cause*" y "*sex*".

Intercalando con los modelos de GLM y GBM

A continuación, utilizaremos un modelo de intercalación que se combinara con los resultados del resto de modelos. Para ello utilizaremos la función de caretStack().

En primer lugar, combinaremos las predicciones de los sub-modelos utilizando un modelo lineal simple (glm).

```
## A glm ensemble of 2 base models: glm, nb, rf, ada, gbm, xgbTree, blackboost,
parRF, knn, glmnet
##
## Ensemble results:
## Generalized Linear Model
##
## 4890 samples
##   10 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7711656  0.5164244
```

Una vez que hemos construido el modelo, vemos que se ha obtenido una buena precisión de entrenamiento.

Ahora vamos a entrenar el modelo combinado que hemos creado con el objetivo de ver que tal predice usando los datos de prueba. El resultado es el siguiente:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  328 1089
```

```
##          V  521  158
##
##                Accuracy : 0.2319
##                  95% CI : (0.2139, 0.2505)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.44
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3863
##             Specificity : 0.1267
##          Pos Pred Value : 0.2315
##          Neg Pred Value : 0.2327
##              Prevalence : 0.4051
##          Detection Rate : 0.1565
##    Detection Prevalence : 0.6760
##       Balanced Accuracy : 0.2565
##
##        'Positive' Class : F
##
```

Como podemos comprobar, los resultados son similares a los que obtuvimos con caretEnsemble(), esto es porque utilizamos glm como modelo principal en la combinación.

Si nos fijamos en los resultados, podemos ver que son bastante negativos. Es decir, la precisión a la hora de entrenar es muy baja.

Después, vamos a combinar las predicciones de los sub-modelos utilizando el modelo GBM.

```
## A gbm ensemble of 2 base models: glm, nb, rf, ada, gbm, xgbTree, blackboost,
parRF, knn, glmnet
##
## Ensemble results:
## Stochastic Gradient Boosting
##
## 4890 samples
##   10 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results across tuning parameters:
##
```

```
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7728016  0.5180401
##   1                  100      0.7728016  0.5174759
##   1                  150      0.7711656  0.5138098
##   2                   50      0.7709611  0.5126084
##   2                  100      0.7709611  0.5113523
##   2                  150      0.7689162  0.5070373
##   3                   50      0.7699387  0.5105254
##   3                  100      0.7654397  0.4998396
##   3                  150      0.7623722  0.4936029
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth
##  = 1, shrinkage = 0.1 and n.minobsinnode = 10.
```

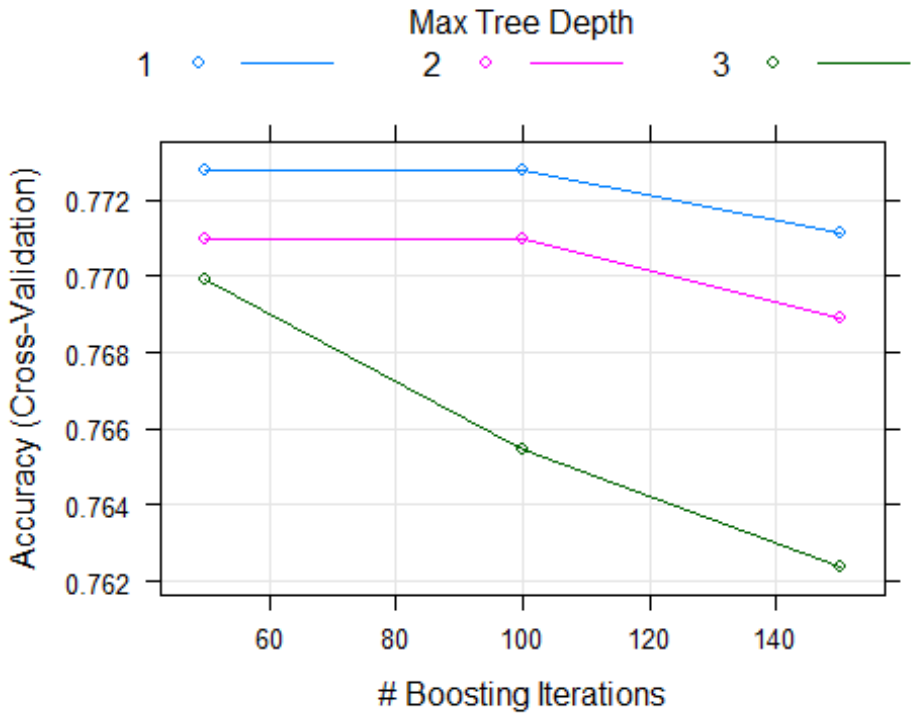


Ilustración 45.CaretEnsemble.GBM. Mejor Ajuste

La ilustración anterior nos muestra la variación de la precisión teniendo en cuenta el número de iteraciones y la profundidad del árbol.

Si comparamos los resultados obtenidos entre el modelo linear y el GBM con las predicciones combinadas, podemos ver que GBM obtiene mejores resultados (0.7728) mientras que el modelo linear obtiene un resultado de 0.7711.

Ahora vamos a usar caretStack() con el modelo que nos ha indicado la rejilla de búsqueda.

```
## A gbm ensemble of 2 base models: glm, nb, rf, ada, gbm, xgbTree, blackboost,
parRF, knn, glmnet
##
## Ensemble results:
## Stochastic Gradient Boosting
##
## 4890 samples
##   10 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7705521  0.5131897
##
## Tuning parameter 'n.trees' was held constant at a value of 50
##  1
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
```

Una vez que hemos construido el modelo, vamos a predecir dicho modelo para poder construir la matriz de confusión.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  296 1057
##          V  553  190
##
##                Accuracy : 0.2319
##                  95% CI : (0.2139, 0.2505)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.4558
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3486
##             Specificity : 0.1524
##          Pos Pred Value : 0.2188
##          Neg Pred Value : 0.2557
##              Prevalence : 0.4051
##          Detection Rate : 0.1412
##    Detection Prevalence : 0.6455
##       Balanced Accuracy : 0.2505
##
##        'Positive' Class : F
##
```

Como podemos comprobar, los resultados de la matriz de confusión vuelven a ser negativos. La precisión es muy baja, con este modelo apenas podremos conseguir predecir un 24% de los resultados del paciente. Ocurre lo mismo con la especificidad, su porcentaje es bajo para poder realizar diagnósticos.

Eliminando los modelos que son no son complementarios

Ahora vamos a realizar un estudio ignorando los modelos que no son complementarios. Si tenemos en cuenta la matriz de correlaciones entre los modelos de la Ilustración 44, vemos que los modelos ada, parRF y glmnet son los que tienen una mayor correlación con el resto de modelos, por lo tanto, hemos considerado su eliminación para los estudios próximos.

En primer lugar, usaremos la función de caretList para crear una lista con todos los modelos seleccionados.

```
##
## Call:
## summary.resamples(object = results)
##
## Models: glm, nb, rf, gbm, xgbTree, blackboost, knn
## Number of resamples: 10
##
## Accuracy
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## glm        0.7464213 0.7592025 0.7689162 0.7685072 0.7755624 0.7893661
## nb         0.7075665 0.7321063 0.7361963 0.7400818 0.7397751 0.7893661
## rf         0.7505112 0.7530675 0.7627812 0.7635992 0.7663599 0.7934560
## gbm        0.7525562 0.7689162 0.7719836 0.7728016 0.7750511 0.7975460
## xgbTree    0.7546012 0.7673824 0.7740286 0.7740286 0.7832311 0.7934560
```

```
## blackboost 0.7505112 0.7617587 0.7689162 0.7717791 0.7822086 0.7934560
## knn        0.7280164 0.7356851 0.7413088 0.7408998 0.7459100 0.7546012
##            NA's
## glm           0
## nb            0
## rf            0
## gbm           0
## xgbTree       0
## blackboost    0
## knn           0
##
## Kappa
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## glm        0.4571628 0.4853150 0.5158085 0.5094980 0.5232337 0.5546566
## nb         0.3847433 0.4447245 0.4534797 0.4573486 0.4555877 0.5582888
## rf         0.4614343 0.4712777 0.4912302 0.4931436 0.5044885 0.5522587
## gbm        0.4836235 0.5108158 0.5180438 0.5199856 0.5253661 0.5691170
## xgbTree    0.4737975 0.5023560 0.5259345 0.5218784 0.5407553 0.5596833
## blackboost 0.4650274 0.4943380 0.5106504 0.5145010 0.5374375 0.5589520
## knn        0.4104001 0.4239976 0.4408537 0.4405938 0.4568590 0.4693723
##            NA's
## glm           0
## nb            0
## rf            0
## gbm           0
## xgbTree       0
## blackboost    0
## knn           0
```

En el resultado anterior podemos observar todas las precisiones obtenidas, obtendremos sus precisiones mínimas, máximas y medias entre otras. Se utilizará como medida la precisión y la métrica de Kappa.

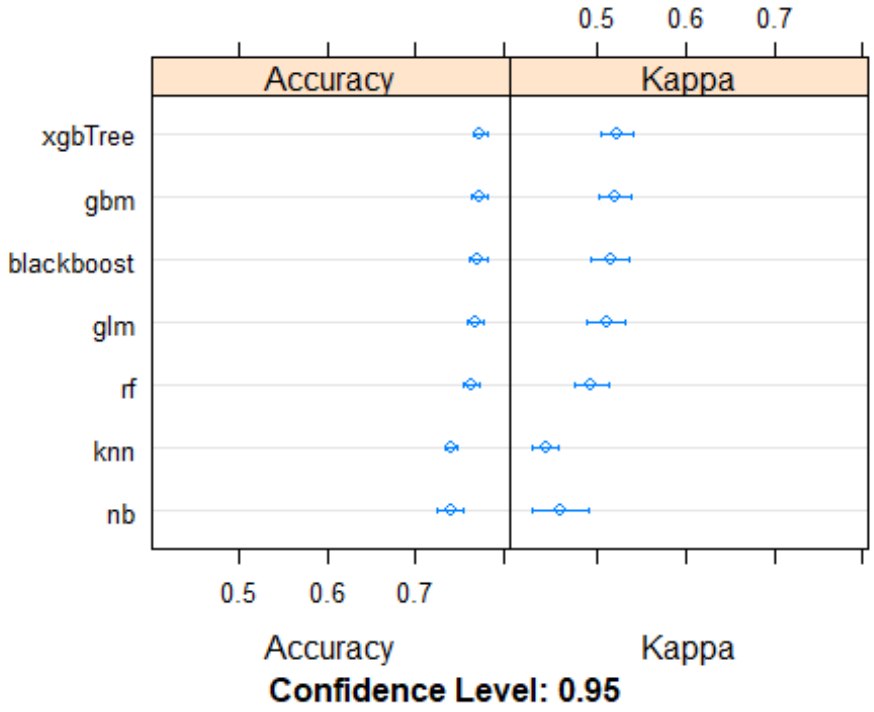


Ilustración 46.CaretEnsemble. Comparación de modelos II

En la ilustración anterior se puede observar las precisiones obtenidas en el entrenamiento de cada modelo.

A continuación, vamos a obtener la matriz de correlaciones entre los modelos seleccionados:

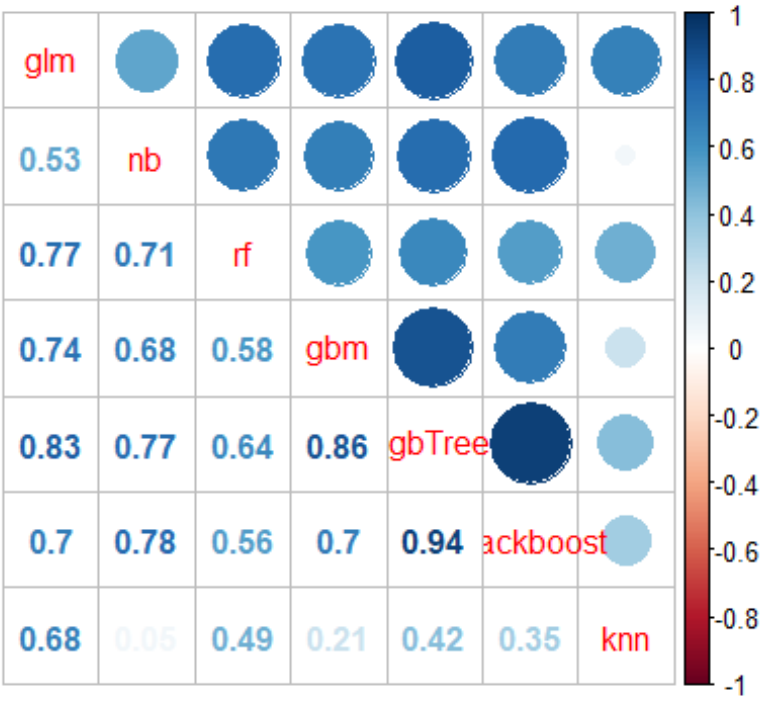


Ilustración 47.CaretEnsemble. Correlación Modelos II

Como se puede comprobar en la ilustración anterior, aun así, existe una gran correlación entre algunos modelos (blackboost y xgbtree), sin embargo, no se considerará la eliminación de ninguna de ellas, aunque podría considerarse este análisis en el futuro.

También podemos usar varImp para obtener las importancias de cada miembro del conjunto, así como también del modelo final.

```
##           overall        glm         nb        rf        gbm    xgbTree
## sex     0.2280114  0.9050923  0.2084746  0.000000  0.0000000  0.0000000
## cause   0.3161797  0.0000000  0.0000000  1.918434  0.2610329  0.1722038
## sah     2.9696561  4.0358244  6.4223263  2.060381  2.2263690  1.6511196
## ec      3.0699523  6.4804414  3.2005784  1.544308  1.8807722  2.2596388
## phm     3.2999419  7.3147435  4.1658753  1.172956  1.9098651  1.8986517
## hmt     3.4785187  6.7843414  5.5322575  1.847850  2.0028175  2.5898793
## oblt    5.8558889  8.4359883  7.6156835  3.961918  5.1368791  3.3528334
## mdls    7.0800560  9.2285446  6.7014626  5.060403  6.1189281  9.1234066
## pupils  9.3323992  8.6200158  9.3304186  9.533392  8.0972108 17.2643780
## verbal 11.3841230  6.6271770 15.5162247 13.548753 12.9467378  8.6832412
```

```
## eye    11.9564392  7.3598631 14.8656127 12.924178 14.1118303  8.1449136
## motor  18.4316891 10.6945162 16.8077350 19.205857 21.4261911 22.5898452
## age    22.5971445 23.5134520  9.6333508 27.221571 23.8813662 22.2698886
##        blackboost        knn
## sex     0.2084746  0.2084746
## cause   0.0000000  0.0000000
## sah     6.4223263  6.4223263
## ec      3.2005784  3.2005784
## phm     4.1658753  4.1658753
## hmt     5.5322575  5.5322575
## oblt    7.6156835  7.6156835
## mdls    6.7014626  6.7014626
## pupils  9.3304186  9.3304186
## verbal 15.5162247 15.5162247
## eye    14.8656127 14.8656127
## motor  16.8077350 16.8077350
## age     9.6333508  9.6333508
```

A continuación, vamos a usar caretEnsemble para construir una combinación de modelos usando como modelo principal el modelo glm, que se intercalara con las predicciones del resto.

```
## The following models were ensembled: glm, nb, rf, gbm, xgbTree, blackboost, k
nn
## They were weighted:
## 2.6132 -1.4376 0.3047 -0.5589 -3.067 -0.5927 -0.1288 0.1564
## The resulting Accuracy is: 0.7695
## The fit for each individual model on the Accuracy is:
##      method  Accuracy  AccuracySD
##         glm 0.7685072 0.014128694
##          nb 0.7400818 0.021349232
##          rf 0.7635992 0.013083673
##         gbm 0.7728016 0.012805476
##     xgbTree 0.7740286 0.011894975
##  blackboost 0.7717791 0.014174663
##         knn 0.7408998 0.008837995
```

Una vez construido la combinación de modelos, vamos a construir la matriz de confusión:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  313 1076
##          V  536  171
##
##                Accuracy : 0.2309
```

```
##                  95% CI : (0.213, 0.2496)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.4487
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3687
##             Specificity : 0.1371
##          Pos Pred Value : 0.2253
##          Neg Pred Value : 0.2419
##              Prevalence : 0.4051
##          Detection Rate : 0.1493
##    Detection Prevalence : 0.6627
##       Balanced Accuracy : 0.2529
##
##        'Positive' Class : F
##
```

Vemos que la probabilidad de sensibilidad y especificidad es bastante pequeña. Estos resultados no nos servirán en un diagnostico ya que no posee una potencia para detectar verdaderos positivos o negativos.

Con CaretEnsemble, podremos también observar la importancia de las variables en los distintos modelos.

```
##           overall        glm         nb        rf        gbm    xgbTree
## sex     0.2280114  0.9050923  0.2084746  0.000000  0.0000000  0.0000000
## cause   0.3161797  0.0000000  0.0000000  1.918434  0.2610329  0.1722038
## sah     2.9696561  4.0358244  6.4223263  2.060381  2.2263690  1.6511196
## ec      3.0699523  6.4804414  3.2005784  1.544308  1.8807722  2.2596388
## phm     3.2999419  7.3147435  4.1658753  1.172956  1.9098651  1.8986517
## hmt     3.4785187  6.7843414  5.5322575  1.847850  2.0028175  2.5898793
## oblt    5.8558889  8.4359883  7.6156835  3.961918  5.1368791  3.3528334
## mdls    7.0800560  9.2285446  6.7014626  5.060403  6.1189281  9.1234066
## pupils  9.3323992  8.6200158  9.3304186  9.533392  8.0972108 17.2643780
## verbal 11.3841230  6.6271770 15.5162247 13.548753 12.9467378  8.6832412
## eye    11.9564392  7.3598631 14.8656127 12.924178 14.1118303  8.1449136
## motor  18.4316891 10.6945162 16.8077350 19.205857 21.4261911 22.5898452
## age    22.5971445 23.5134520  9.6333508 27.221571 23.8813662 22.2698886
##         blackboost        knn
## sex     0.2084746  0.2084746
## cause   0.0000000  0.0000000
## sah     6.4223263  6.4223263
## ec      3.2005784  3.2005784
## phm     4.1658753  4.1658753
## hmt     5.5322575  5.5322575
```

```
## oblt    7.6156835  7.6156835
## mdls    6.7014626  6.7014626
## pupils  9.3304186  9.3304186
## verbal 15.5162247 15.5162247
## eye    14.8656127 14.8656127
## motor  16.8077350 16.8077350
## age     9.6333508  9.6333508
```

Como podemos comprobar, las variables de "sex" y "causa" siguen siendo las variables menos importantes para la construcción del modelo.

Intercalando con los modelos de GLM yGBM

Ahora vamos a intercalar los modelos usando caretStack. En primer lugar, utilizaremos como modelo principal el modelo GLM, que se intercalara con las predicciones del resto de modelos.

```
## A glm ensemble of 2 base models: glm, nb, rf, gbm, xgbTree, blackboost, knn
##
## Ensemble results:
## Generalized Linear Model
##
## 4890 samples
##    7 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7695297  0.5136152
```

Como podemos observar, se han obtenido un buen resultado en la precisión al construir el modelo. Aunque individualmente otros modelos han obtenido mejores resultados como GBM, XGB y Random Forest.

Nuevamente vamos a construir la matriz de confusión a partir de la predicción usando el modelo construido y los datos de prueba.

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

```
##           Reference
## Prediction    F    V
##          F  313 1076
##          V  536  171
##
##                Accuracy : 0.2309
##                  95% CI : (0.213, 0.2496)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.4487
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3687
##             Specificity : 0.1371
##          Pos Pred Value : 0.2253
##          Neg Pred Value : 0.2419
##              Prevalence : 0.4051
##          Detection Rate : 0.1493
##    Detection Prevalence : 0.6627
##       Balanced Accuracy : 0.2529
##
##        'Positive' Class : F
##
```

Como podemos observar, nuevamente se han obtenido resultados bastante negativos. La predicción es no es buena, ya que solo sólo se es capaz de predecir un 23% de los resultados. La sensibilidad y la especificidad son bastante negativas también.

Ahora vamos a utilizar el modelo de GBM como modelo principal en la combinación con las predicciones del resto de modelos.

```
## A gbm ensemble of 2 base models: glm, nb, rf, gbm, xgbTree, blackboost, knn
##
## Ensemble results:
## Stochastic Gradient Boosting
##
## 4890 samples
##    7 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
```

```
##   1                   50      0.7740286  0.5202522
##   1                  100      0.7705521  0.5123059
##   1                  150      0.7701431  0.5110697
##   2                   50      0.7713701  0.5139670
##   2                  100      0.7689162  0.5079795
##   2                  150      0.7654397  0.5007784
##   3                   50      0.7672802  0.5051463
##   3                  100      0.7652352  0.5003500
##   3                  150      0.7660532  0.5012908
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth
##  = 1, shrinkage = 0.1 and n.minobsinnode = 10.
```

Una vez que hemos construido el modelo usando la rejilla, obtenemos los parámetros para ajustar el modelo para conseguir la mayor precisión.

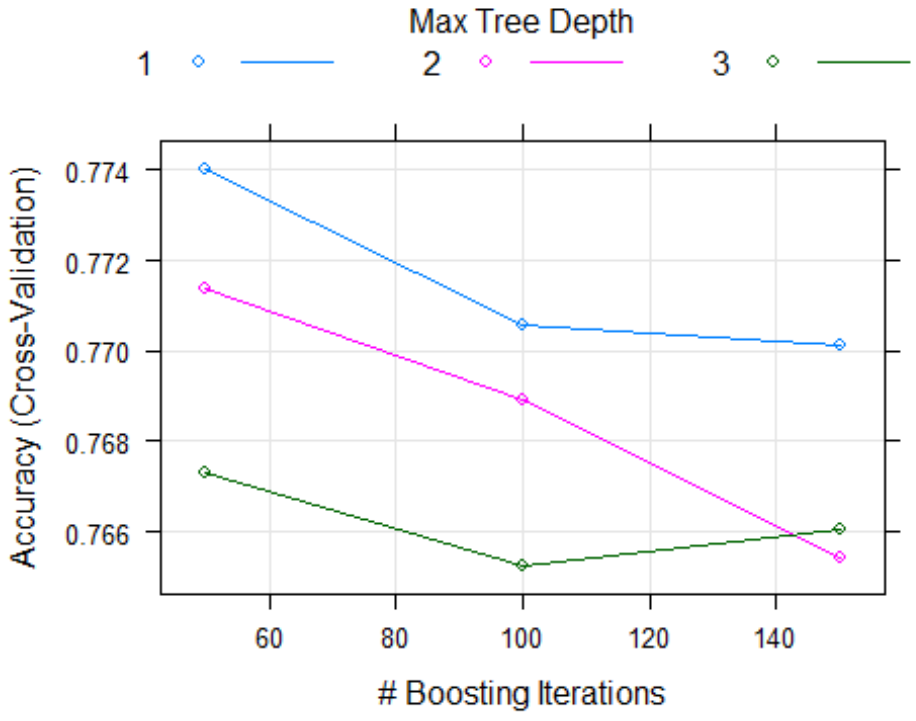


Ilustración 48.CaretEnsemble.GBM. Mejor Ajuste II

Teniendo en cuenta los parámetros obtenidos usando la rejilla de búsqueda, vamos a construir el modelo de GBM con estos parámetros.

```
## A gbm ensemble of 2 base models: glm, nb, rf, gbm, xgbTree, blackboost, knn
##
## Ensemble results:
## Stochastic Gradient Boosting
##
## 4890 samples
##    7 predictor
##    2 classes: 'F', 'V'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4401, 4401, 4401, 4401, 4401, 4401, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7699387  0.5116488
##
## Tuning parameter 'n.trees' was held constant at a value of 50
##  1
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
```

Una vez que se tiene el modelo construido, vamos a predecirlo usando los datos de prueba.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    F    V
##          F  302 1053
##          V  547  194
##
##                Accuracy : 0.2366
##                  95% CI : (0.2186, 0.2554)
##     No Information Rate : 0.5949
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.4463
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3557
##             Specificity : 0.1556
##          Pos Pred Value : 0.2229
##          Neg Pred Value : 0.2618
##              Prevalence : 0.4051
```

```
##          Detection Rate : 0.1441
##    Detection Prevalence : 0.6465
##       Balanced Accuracy : 0.2556
##
##        'Positive' Class : F
##
```

Tampoco se ha conseguido obtener buen resultado en las predicciones usando esta combinación de modelos. Ciertamente sí que se ha notado diferencias respecto al modelo que construimos de GBM para todos los modelos propuestos al principio, pero sin embargo la mejora no es suficiente como para utilizar este modelo en diagnósticos clínicos. El porcentaje de especificidad es muy bajo, igual que el de sensibilidad.

4. RESULTADOS

En este apartado nos centraremos en los resultados obtenidos al estudiar todos los modelos. Concretamente nos centraremos en las medidas usuales para la comparación entre modelos y que ya han sido explicadas en la sección 3.3.1. Además, se ha realizado un estudio sobre los tiempos de procesamiento de cada modelo con el objetivo de no solamente comparar los resultados de precisión sino el tiempo que necesita cada algoritmo para converger en el mejor resultado.

Para ello hemos tenido en cuenta los modelos siguientes:

MODELOS	**VARIABLES UTILIZADAS**
Redes Neuronales (NN)	TODAS
Redes Neuronales (NN2)	TODAS EXCEPTO “VERBAL”
XGBoost	TODAS
GBM	TODAS
Adaboost	TODAS
Random Forest	TODAS
N.Bayes	TODAS
RLog_Comp (Regresión Logística)	TODAS
RLog_1 (Regresión Logística)	Age, ec, verbal, hmt, motor, eye, oblt, mdls, sah, cause, pupils, sex
RLog_2 (Regresión Logística)	Age, ec, verbal, hmt, motor, eye, oblt, mdls, sah, cause, pupils
RLog_3 (Regresión Logística)	Age, ec, verbal, hmt, motor, eye, oblt, mdls, sah, cause
RLog_4 (Regresión Logística)	Age, ec, verbal, hmt, motor, eye, oblt, mdls, sah
RLog_5 (Regresión Logística)	Age, ec, verbal, hmt, motor, eye, oblt, mdls

RLog_6 (Regresión Logística)	Age, ec, verbal, hmt, motor, eye, oblt
RLog_7 (Regresión Logística)	Age, ec, verbal, hmt, motor, eye
RLog_8 (Regresión Logística)	Age, ec, verbal, hmt, motor
RLog_9 (Regresión Logística)	Age, ec, verbal, hmt
RLog_10 (Regresión Logística)	Age, ec, verbal
RLog_11 (Regresión Logística)	Age, ec
RLog_12 (Regresión Logística)	Age
RLog_13 (Regresión Logística)	TODAS EXCEPTO “SEX” y “CAUSE”

Tabla 2. Modelos utilizados

4.1.1 USANDO LA METRICA DE PRECISION Y KAPPA

4.1.1.1 Uso de Precisión

Teniendo en cuenta algunos de estos modelos, hemos obtenido la precisión y se ha graficado, obteniéndose el siguiente resultado:

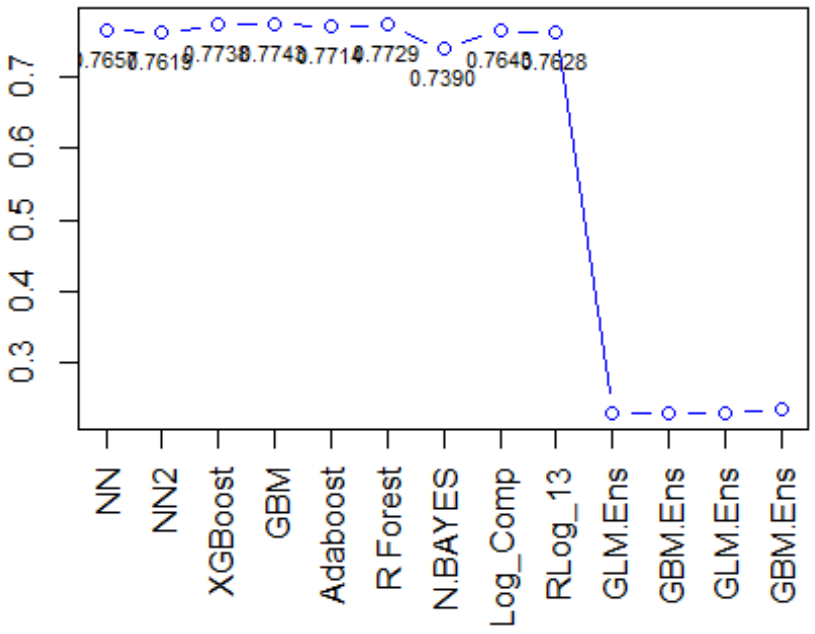


Ilustración 49. Comparativa de precisión

Como se puede observar en el grafico anterior, el algoritmo de GBM ha sido el que mejor resultado ha obtenido. Seguido del algoritmo de XGBoost.

Los peores resultados se han obtenido mediante la combinación de modelos, esta combinación de modelos no es apta para realizar predicciones. Por otro lado, el modelo individual con peores resultados ha sido el Naïves Bayes.

4.1.1.2 Uso de Kappa

De la misma forma que se ha obtenido la métrica de precisión, obtendremos la métrica e Kappa.

Utilizaremos algunos de los modelos utilizados en la Tabla 2.

Habiendo realizado el entrenamiento y posteriores pruebas con el modelo entrenado, se han obtenido los siguientes resultados.

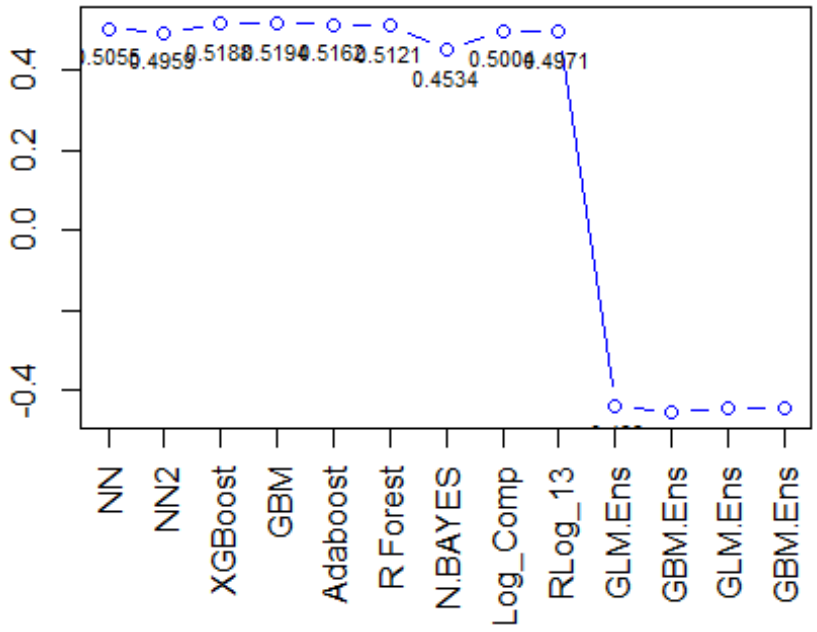


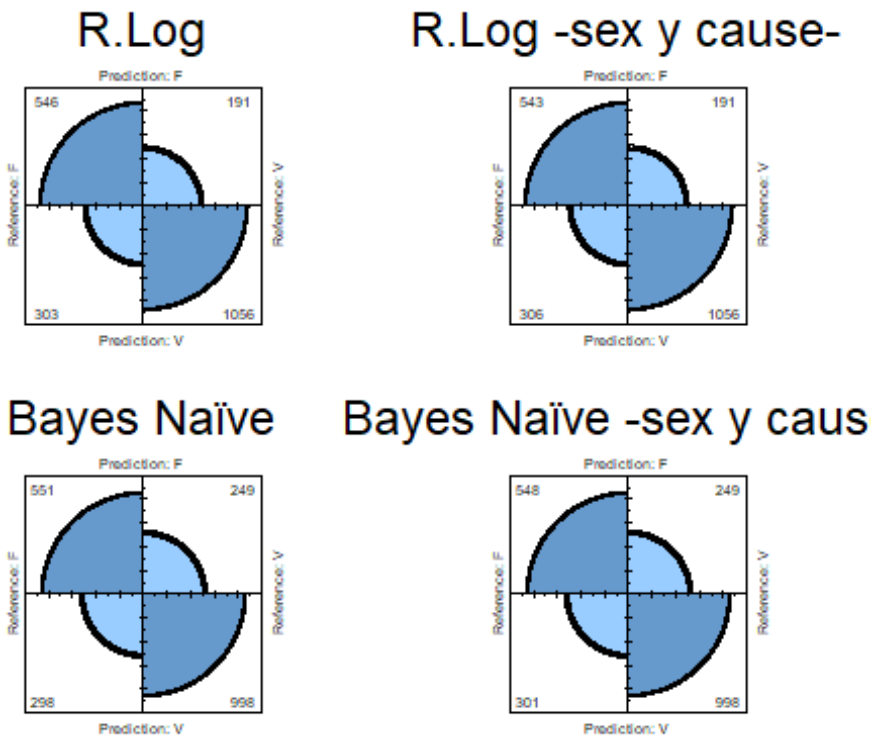
Ilustración 50. Comparativa de Kappa

Como se puede comprobar, aunque Kappa sea una medida distinta a la precisión, se ha obtenido un gráfico muy similar, que nos aporta la misma información que el anterior.

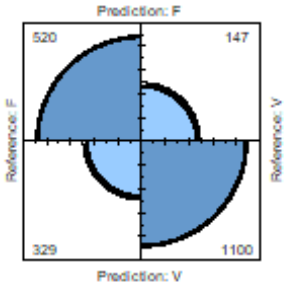
Destacando nuevamente como el modelo GBM el que obtiene las mejores predicciones para nuestro conjunto de datos y la combinación de modelos ha sido la peor.

4.1.2 COMPARATIVA DE MATRICES DE CONFUSIÓN

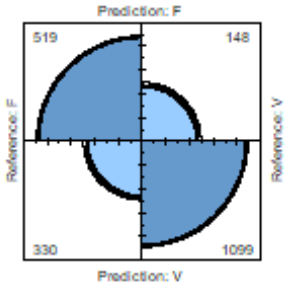
Una vez que hemos obtenido la predicción de todos los modelos, hemos agrupado todas las matrices de confusión para que el lector pueda de esta forma realizar una rápida comparación entre ellos.



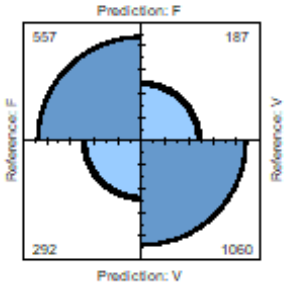
R.Forest



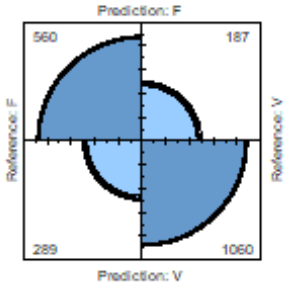
R.Forest -sex y cause-



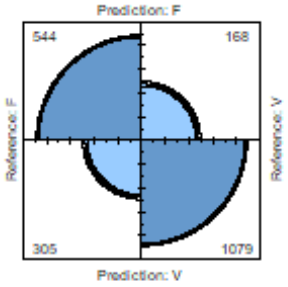
Adaboost



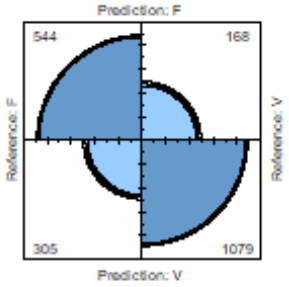
Adaboost -sex y cause-



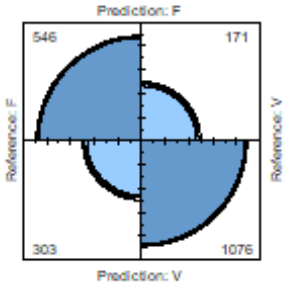
GBM



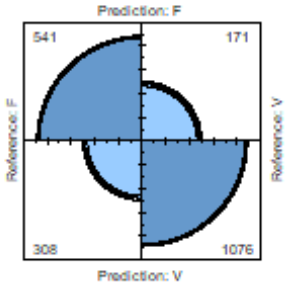
GBM -sex y cause-

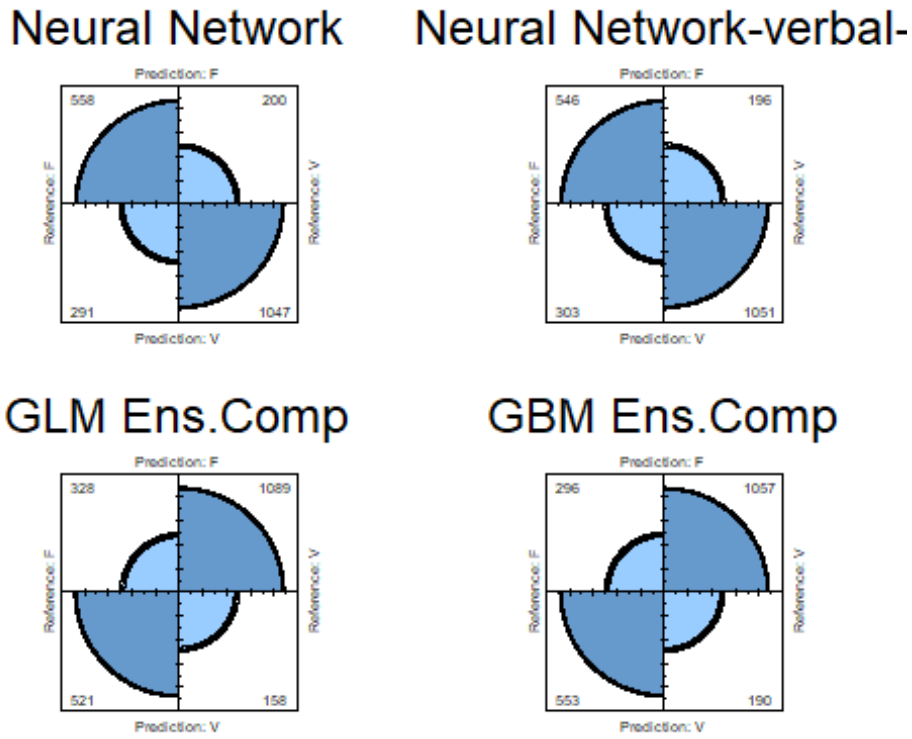


XGB



XGB -sex y cause-





Como se puede comprobar en las gráficas anteriores, existen un gran número de resultados existentes. Si descartamos los modelos con los que hemos tenido un peor resultado y utilizamos aquellos que mejores resultados han tenido (adaboost, gbm, xgb y redes neuronales), podemos ver que la tasa de verdaderos positivos y verdaderos negativos, que son las variables que se desea estudiar, es muy similar entre ellos.

Si queremos obtener la mayor tasa de verdaderos positivos, podemos utilizar el modelo de redes neuronales. En cambio, si queremos obtener el mayor número de verdaderos negativos, deberemos utilizar el modelo gbm o incluso xgboost.

Podemos observar también que los modelos combinados (Glm.ens.Comp y Gbm.ens.Comp), no obtienen buen resultado, no nos sirven apenas de ayuda ya que no son capaces de detectar los verdaderos positivos o verdaderos negativos que en este problema es lo que realmente nos interesa.

4.1.3 USANDO LA MÉTRICA DE LA CURVA DE ROC

Recordemos que el área bajo la curva ROC (AUC) estima la capacidad de distinguir o discriminar entre pacientes vivos y pacientes que fallecen.

Recientemente, se ha prestado más atención al AUC parcial (pAUC) que al AUC. Esto se debe a que pAUC permite analizar un rango en concreto. No solo eso, también permite poder diferenciar dos curvas con la misma área, puesto que estas curvas serán tendrán la misma área, pero podrán diferenciarse en rangos puntuales.

El AUC parcial se ha propuesto como una medida alternativa al AUC completo. Cuando se usa el AUC parcial, se consideran solo aquellas regiones del espacio ROC que corresponden a valores clínicamente relevantes de la sensibilidad o especificidad de la prueba.

El rango que se utiliza en la medicina es del 90% al 100% tanto para la sensibilidad como especificidad. Este rango es adecuado para obtener la tasa de falsos positivos de una determinada situación clínica.

Cuando se define un pAUC, este puede ser estandarizado (corregido). Esta corrección se controla mediante el argumento partial.auc.correct. Si partial.auc.correct = TRUE, se aplicará la corrección de McClish:

$$(1 + (auc - min) / (max - min)) / 2$$

donde auc es el pAUC no corregido calculado en la región definida por partial.auc, min es el valor del AUC no-discriminativo (con un AUC de 0,5 o 50 en la región) y max es el AUC máximo posible en la región. Con esta corrección, el AUC será de 0.5 si no es discriminante y 1.0 si es máximo, para cualquiera que sea la región definida.

Dicho de otra forma, con esta transformación conseguimos ver el área parcial en la misma escala que el área total. Esto nos facilita la interpretación de los verdaderos positivos

Hay que tener en cuenta que esta corrección no está definida para curvas debajo de la diagonal (auc <min).

A continuación, se muestran los resultados teniendo en cuenta el área bajo la curva ROC (AUC) en un rango de Especificidad y Sensibilidad del 90% al 100%.

MODELO	AUCsp	AUCse
NN	66.71931	68.22199
NN2	65.60533	67.88797
XGBoost	66.32966	70.12267
GBM	66.96680	69.84230
Adaboost	67.62618	70.52033
Random Forest	66.32509	68.96172
N.Bayes	65.43049	64.79555
RLog_13	66.80462	68.25485
RLog_Comp	66.71101	68.22468
RLog_1	67.25467	68.15001
RLog_2	67.33645	68.15831
RLog_3	67.11965	67.26436
RLog_4	67.12984	67.27197
RLog_5	66.71041	67.26292
RLog_6	66.55063	67.25164
RLog_7	65.61876	64.34792
RLog_8	66.29978	63.97593
RLog_9	65.08922	61.91413
RLog_10	64.23032	61.14322
RLog_11	52.89411	57.00550
RLog_12	51.97595	56.05466

Tabla 3. Comparativa AUCse y AUCsp

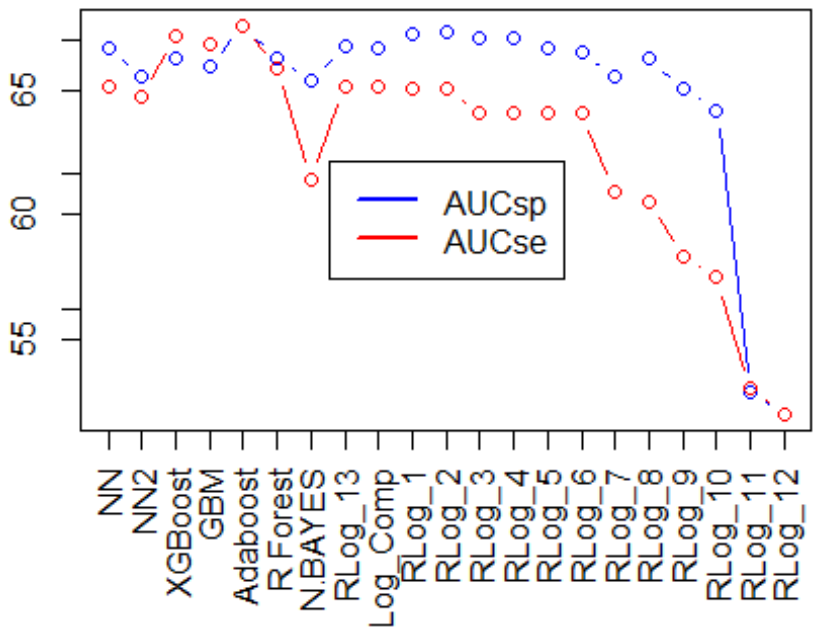


Ilustración 51. Comparativa AUCsp y AUCse

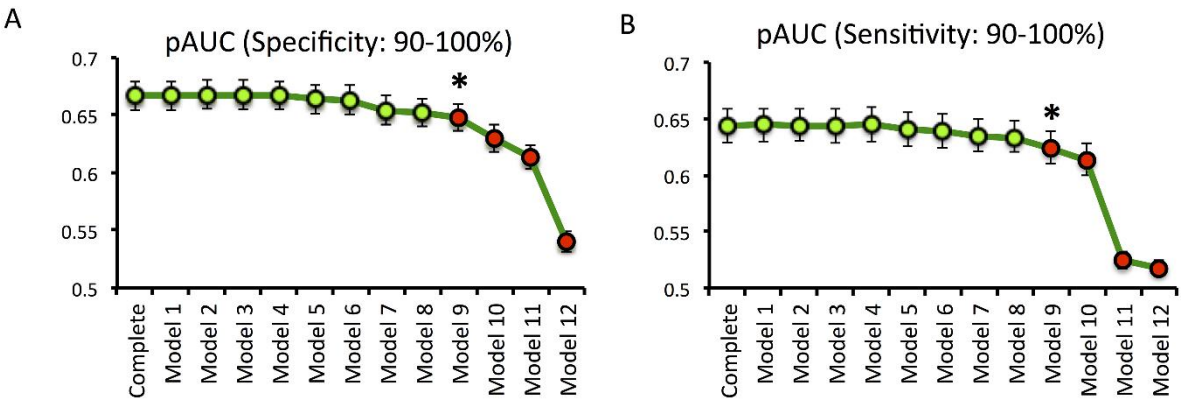
Como se puede observar en el grafico anterior, el máximo AUC de Especificidad se consigue con el modelo Adaboost y el menor valor se consigue con el mod_fit_12 (que corresponde a un modelo de regresión logística).

Si tenemos en cuenta el área AUC de Sensibilidad, el máximo resultado se obtiene también usando Adaboost y el menor se obtiene usando también el modelo de regresión logística, mod_fit_12.

Es relevante destacar que el mejor modelo tanto para Sensibilidad como para Especificidad es el Adaboost, sin embargo, si tenemos en cuenta y comparamos otros pares de modelos, quizá un modelo da mejores resultados para especificidad, y el otro obtenga mejores resultados en sensibilidad. Por ejemplo, podemos destacar los modelos mod_fit_7 y

mod_fit_8. El mod_fit_7 tiene mejor especificidad, sin embargo, el modelo mod_fit_8 tiene mayor sensibilidad, por lo tanto, a la hora de escoger uno de estos modelos, será necesario ver qué medida interesa más en el estudio.

A continuación, se muestran los resultados obtenidos en el artículo inicial:



Si observamos de forma detallada, vemos que los valores obtenidos en nuestro estudio se asimilan bastante a los obtenidos en el artículo de investigación. Vemos que el Modelo 2 obtiene mejores valores de AUCsp que el modelo 1 y el modelo completo. Eso mismo ocurre con nuestros resultados. Sin embargo, los mejores valores de AUCse para el articulo de investigación se obtienen con el modelo 4 mientras que en nuestro análisis lo hemos obtenido con el modelo completo. Esto se puede deber principalmente a la diferencia entre los tratamientos de datos realizados en el artículo de investigación original y aquellos que se han realizado en nuestro estudio.

En la siguiente ilustración, se muestra la curva ROC para todos los modelos utilizados. Se ha distribuido en dos gráficos para que sea más fácil de observar cada uno de los modelos y su respectiva curva.

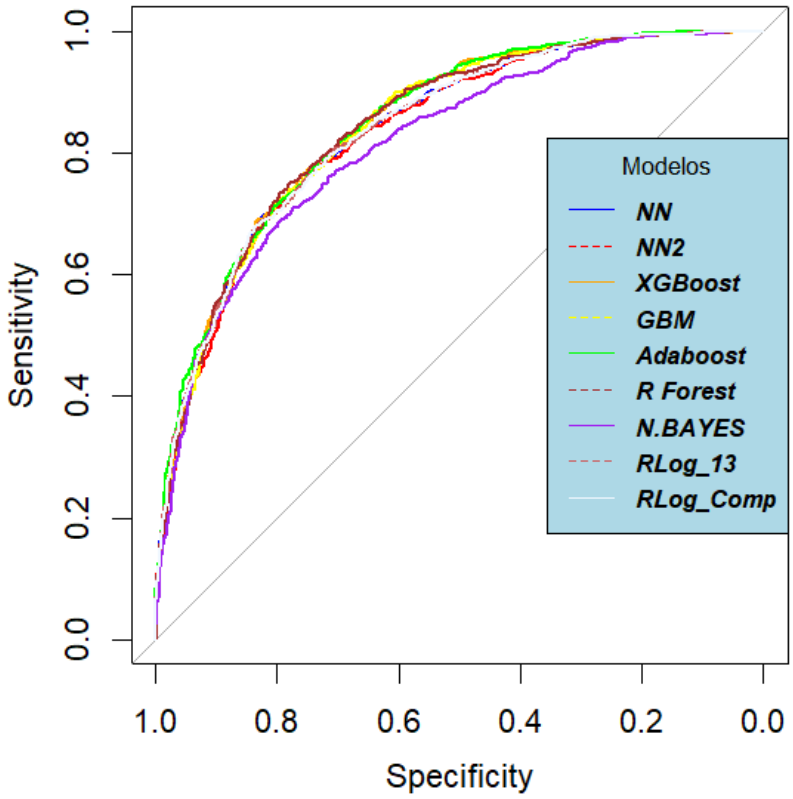


Ilustración 52. Comparativa curva ROC I

Vemos que entre los modelos con mejores resultados en áreas bajo la curva destacan nuevamente los modelos de Adaboost, GBM y XGBoost. Una vez más, el modelo de Naïves Bayes es el que peor área posee.

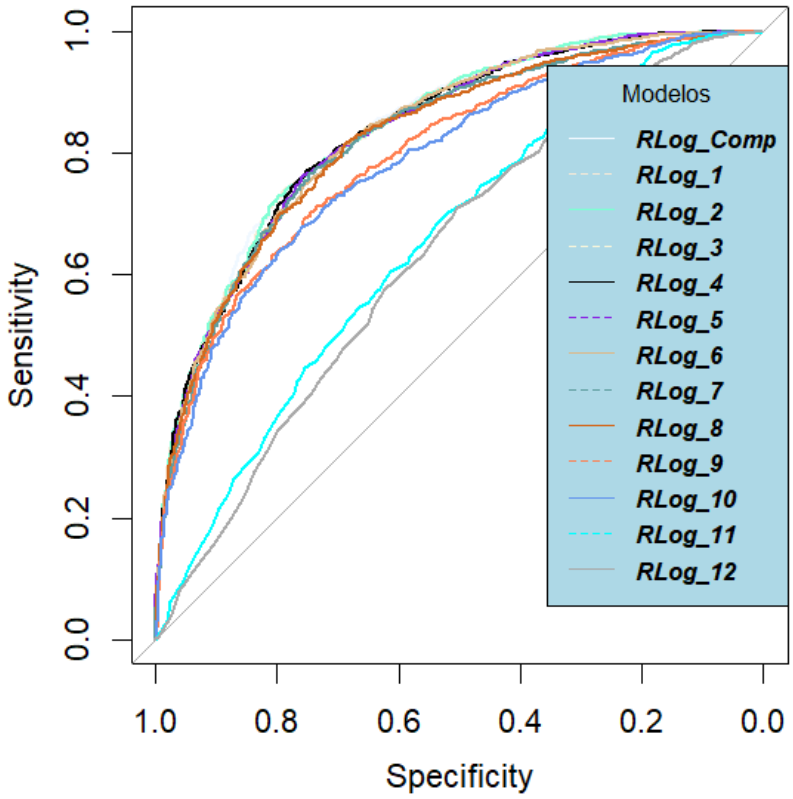


Ilustración 53. Comparativa curva ROC II

En la ilustración anterior, podemos observar únicamente los modelos construidos usando la Regresión Logística. Vemos que los mejores modelos son la mayoría los que se han construido utilizando el mayor número de predictores. Los peores modelos son: RLog_11 y RLog_12.

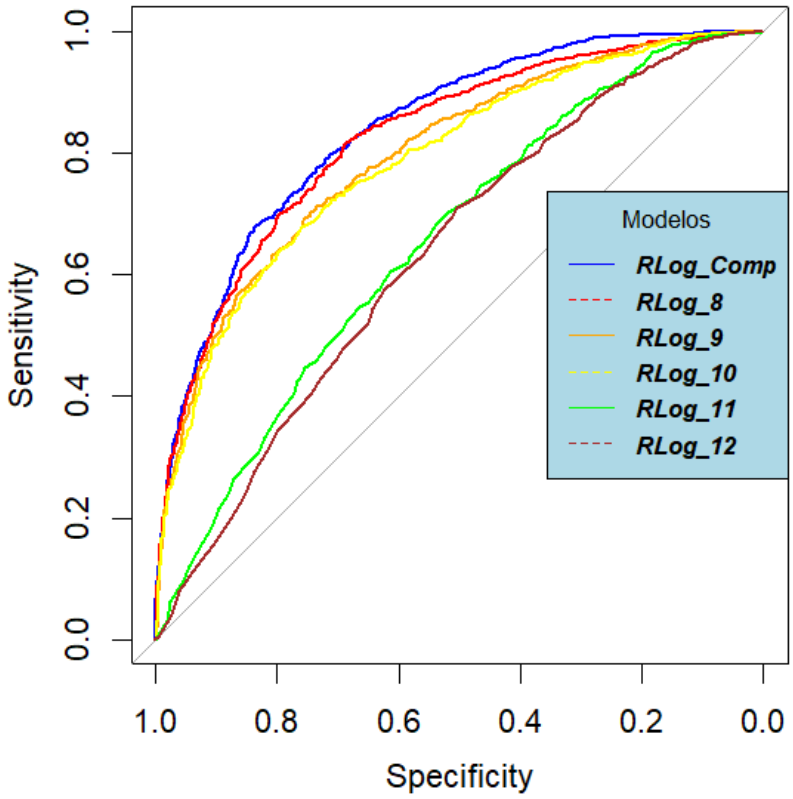


Ilustración 54. Comparativa curva ROC III

En la Ilustración 51 podemos observar la comparación entre el modelo que contempla todas los predictores y los modelos que contemplan hasta 5 predictores. Vemos que la diferencia entre el modelo completo y el modelo RLog_8 es bastante pequeña. Habiendo rangos en los que este último modelo obtiene mejores resultados que el modelo completo.

4.1.4 COMPARATIVA DE TIEMPOS

Además de tener en cuenta todas las métricas que indican la capacidad de predecir y en definitiva la bondad de los modelos, se ha tenido en cuenta su rendimiento, es decir cuánto tiempo se toma en obtener dichos resultados.

Por tanto, se han tomado medidas sobre el tiempo usado en cada algoritmo. Como podemos imaginar, la complejidad de los modelos implica un mayor costo en el tiempo de procesado; cuantas más variables debamos probar en un modelo, mayor va a ser el tiempo en obtener la mejor precisión. En este aspecto debemos tener en cuenta por tanto todas las variables de "tuning" de los modelos. Posiblemente, algunos algoritmos pueden tardar menos si se acota o se ajusta el límite de valores que puedan tomar cada variable de "tuning", si obviamos este aspecto y consideramos que el rango de valores es similar para todos los modelos, se ha obtenido el siguiente resultado:

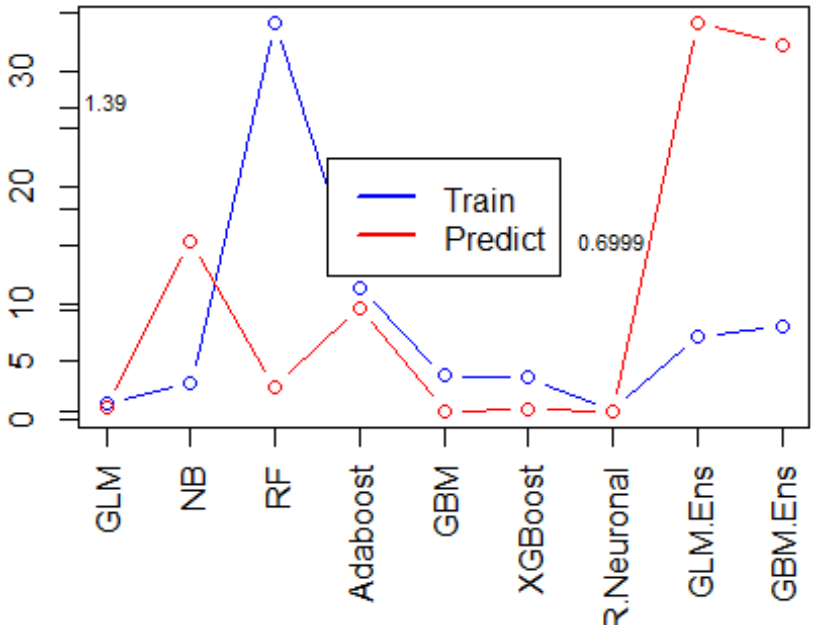


Ilustración 55. Comparativa de tiempos

Como se puede observar, el modelo de Random Forest es el que más tiempo ha utilizado en entrenarse. Seguido por la combinación de modelos.

Los modelos que destacaríamos de forma individual y que, además de obtener una precisión optima, no han consumido demasiado tiempo serian: adaboost, xgboost y gbm. Siendo este último modelo el que mejor precisión ha obtenido.

Teniendo estos resultados, fácilmente podríamos interpretar que modelos serian aconsejables y cuales no dependiendo no solo de la precisión sino también del tiempo. Concretamente podemos destacar 3 modelos que sobresalen del resto en cuanto a velocidad de cómputo y precisión y serían los ya comentados anteriormente: adaboost, xgboost y gbm.

5. CONCLUSIONES

Una vez que hemos realizado el estudio y se han obtenido los resultados, es hora de realizar una pequeña valoración en forma de conclusiones.

En primer lugar, la realización de este trabajo de fin de master ha tenido como objetivo realizar un estudio para la obtención de modelos que sean capaces de conseguir una gran predicción. En el artículo de investigación inicial sobre el que se ha trabajado en este trabajo, se ha considerado un único modelo, que ha sido la regresión logística. Sobre este modelo se han construido una serie de modelos mediante la eliminación de variables, y se ha utilizado indicadores como Nagelkerke R2 y la Curva de ROC, posteriormente se utilizarán redes Bayesianas como modelo de clasificación.

En nuestro caso, también hemos utilizado la curva de ROC y se han obtenido resultados muy similares. Obviamente nunca se llegarán a obtener los mismos resultados, sin embargo, sí que hemos llegado a conclusiones similares. Se han excluido las mismas variables en los modelos y se han obtenido resultados similares en el estudio de los modelos de regresión logística construidos basándose en esta exclusión de variables.

Sin embargo, nosotros hemos realizado pruebas y comparaciones con otros modelos que no se han tenido en cuenta en el artículo de investigación original.

En este documento se han recogido otros modelos distintos a los propuestos en el artículo y sus resultados y como conclusiones, hemos podido destacar que se han obtenido mejores resultados, concretamente el modelo GBM ha sido el que mejores precisiones ha obtenido junto con XGB, del que se puede destacar su gran eficiencia.

Sin embargo, el modelo que contempla los mejores resultados en especificidad y sensibilidad (para el rango establecido), ha sido el modelo de Adaboost.

6. LÍNEAS FUTURAS

Teniendo en cuenta los avances que se han realizado en este documento como continuación del artículo de investigación, podemos destacar que es evidente que en un futuro se puedan realizar un estudio más minucioso al trabajo realizado y se puedan realizar nuevos trabajos para obtener mejores resultados.

Concretamente, en un futuro, podríamos estudiar nuevos modelos o incluso volver a analizar los modelos utilizados teniendo en cuenta la construcción basándose en la eliminación o inclusión de variables. Esto nos ayudaría no solo a obtener mejores precisiones sino también tener presente qué variables son las más importantes y las que mejor ayudan a predecir los resultados finales. De esta forma se podrá aplicar clínicamente un tratamiento u otro para conseguir mejores resultados en la recuperación de los pacientes.

Relacionado con la idea de obtener mejores resultados finales, podríamos tener en cuenta no solo las variables que se han estudiado en el artículo de investigación y en este documento, sino que también podríamos incluir alguna variable más que no se haya tenido en cuenta. Concretamente en este documento, se ha utilizado la matriz de correlación con un número determinado de variables, pero estas variables no son todas las que se encuentran en el repositorio inicial, posiblemente teniendo en cuenta otras variables, podríamos haber obtenido mejores resultados.

Por otro lado, cabe destacar que se podría realizar nuevas investigaciones en la combinación de modelos, puesto que en este análisis no se han obtenido los resultados que podrían esperarse de una combinación de modelos. En este sentido podrían explotarse distintos modelos que no se han tenido en cuenta en este estudio.

7. BIBLIOGRAFÍA

[Manual abreviado de Análisis Estadístico Multivariante. Jesús Montanero Fernández] http://matematicas.unex.es Recuperado el 28 de marzo de 2018 de: https://ignsl.es/historia-del-big-data/

[Análisis Multivariente, usando R. José Carlos Vega Vilca] http://cicia.uprrp.edu Recuperado el 28 de marzo de 2018 de: http://cicia.uprrp.edu/publicaciones/Papers/ManualESTA5503.pdf

[Ambrosio Torres] https://www.r-bloggers.com. Recuperado el 2 de abril de 2018 de: https://www.r-bloggers.com/lang/spanish/940

[Selva Prabhakaran] http://r-statistics.co. Recuperado el 2 de abril de 2018 de: http://r-statistics.co/Outlier-Treatment-With-R.html

[Tema 3. Contraste de la normalidad multivariante. César A. Sanchez Sellero] http://eio.usc.es. Recuperado el 7 de abril de 2018 de: http://eio.usc.es/eipc1/base/BASEMASTER/FORMULARIOS-PHP/MATERIALESMASTER/Mat_142400_mmulti1011tema3.pdf

[Contrading. Victor A. Rico] http://www.cotradingclub.com. Recuperado el 7 de abril de 2018 de: http://www.cotradingclub.com/2017/05/25/prueba-de-normalidad-en-modelos-de-prediccion/

[Un análisis con R. Datos Multivariantes. Francesc Carmona] http://www.ub.edu. Recuperado el 9 de abril de 2018 de: http://www.ub.edu/stat/docencia/EADB/Ejemplo.pdf

[Javier Seoane, Carlos P. Carmona, Rocío Tarjuelo y Aimara Planillo] http://www.uam.es. Recuperado el 11 de abril de 2018 de: http://www.uam.es/personal_pdi/ciencias/jspinill/CFCUAM2014/RF_BRT-CFCUAM2014.html

[Manuel Sigüeñas Gonzales] https://rpubs.com Recuperado el 15 de abril de 2018 de: https://rpubs.com/MSiguenas/122473

[Analytics Vidhya Content Team] https://www.analyticsvidhya.com. Recuperado el 17 de abril de 2018 de: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/

[Grupo IGN] https://ignsl.es. Recuperado el 25 de abril de 2018 de: https://ignsl.es/historia-del-big-data/

[Instituto de Ingeniería del Conocimiento] http://www.iic.uam.es. Recuperado el 28 de abril de 2018 de: http://www.iic.uam.es/innovacion/herramientas-big-data-para-empresa/

Jason Brownlee. (2016). Machine Learning Mastery with R.

Michele Usuelli. (2014). R Machine Learning Essentials.