Agradecimientos

# Resumen

Las lesiones traumáticas cerebrales son un problema de salud mundial. Por tanto, se hace necesario el estudio de los predictores con el objetivo de establecer estrategias, decisiones clínicas y protocolos para el tratamiento de estas lesiones de forma más óptima y tratando siempre de obtener el mejor resultado en el paciente.

En este estudio se realiza una clasificación sobre la importancia de los predictores para posteriormente utilizar distintos modelos para conseguir la mayor predicción posible.

# Abstract

# Tabla de Contenidos

[Resumen 3](#_Toc517778900)

[Abstract 4](#_Toc517778901)

[Tabla de Contenidos 5](#_Toc517778902)

[Listado de Figuras 7](#_Toc517778903)

[Listado de Tablas 9](#_Toc517778904)

[1. INTRODUCCIÓN Y OBJETIVOS 11](#_Toc517778905)

[1.1 INTRODUCCIÓN 11](#_Toc517778906)

[1.2 OBJETIVOS 12](#_Toc517778907)

[1.3 ESTRUCTURA 13](#_Toc517778908)

[2. ESTADO DEL ARTE 15](#_Toc517778909)

[2.1 BIG DATA 15](#_Toc517778910)

[2.2 PRINCIPALES FASES DE BIG DATA 16](#_Toc517778911)

[2.3 HERRAMIENTAS UTILIZADAS EN BIG DATA 17](#_Toc517778912)

[3. EVALUACIÓN DE RIESGOS 19](#_Toc517778913)

[3.1 MOTIVACION EN EL USO DE R 19](#_Toc517778914)

[4. DESARROLLO 21](#_Toc517778915)

[4.1 PREPARACIÓN DE LOS DATOS 21](#_Toc517778916)

[4.1.1 Fuente de obtención de los datos 21](#_Toc517778917)

[4.1.2 Nomenclatura del conjunto de los datos 22](#_Toc517778918)

[4.1.3 Clasificación entre: ALIVE, DEATH, NO-DATA y MD/GR 25](#_Toc517778919)

[4.1.4 Clasificación entre: ESCANEADOS y NO ESCANEADOS 28](#_Toc517778920)

[4.1.5 Eliminación y centralización de variables 32](#_Toc517778921)

[4.2 PRE-PROCESADO DE LOS DATOS 33](#_Toc517778922)

[4.2.1 Búsqueda de outliers (datos anómalos) 33](#_Toc517778923)

[4.2.2 Análisis de normalidad 36](#_Toc517778924)

[4.2.3 Estudio de correlación 41](#_Toc517778925)

[4.2.4 Selección de variables con mayor importancia 46](#_Toc517778926)

[4.3 MODELADO DE DATOS 62](#_Toc517778927)

[4.3.1 Términos generales 62](#_Toc517778928)

[4.3.2 Caret 65](#_Toc517778929)

[4.3.3 Uso de modelos 66](#_Toc517778930)

[5. RESULTADOS 100](#_Toc517778931)

[5.1.1 USANDO LA METRICA DE PRECISION Y KAPPA 101](#_Toc517778932)

[5.1.2 USANDO LA METRICA DE LA CURVA DE ROC 103](#_Toc517778933)

[5.1.3 USANDO LA COMBINACION DE MODELOS 107](#_Toc517778934)

[5.1.4 COMPARATIVA DE TIEMPOS 113](#_Toc517778935)

[6. CONCLUSIONES 115](#_Toc517778936)

[7. LÍNEAS FUTURAS 116](#_Toc517778937)

[8. BIBLIOGRAFÍA 116](#_Toc517778938)

# Listado de Figuras

[Ilustración 1. Fases de Big Data 17](#_Toc517090394)

[Ilustración 2. Diagrama de caja. Outliers 34](#_Toc517090395)

[Ilustración 3. Diagrama de caja. Outliers. Edad 35](#_Toc517090396)

[Ilustración 4. Diagrama de caja. Edad. Eliminando Outliers 36](#_Toc517090397)

[Ilustración 5.Gráfico Normalidad I 37](#_Toc517090398)

[Ilustración 6.Gráfico Normalidad II 38](#_Toc517090399)

[Ilustración 7.Gráfico Normalidad III 39](#_Toc517090400)

[Ilustración 8.Gráfico Normalidad IV 39](#_Toc517090401)

[Ilustración 9.Chi-Square-Mahalanobis 41](#_Toc517090402)

[Ilustración 10.Matriz de correlación I 42](#_Toc517090403)

[Ilustración 11.Matriz de correlación II 43](#_Toc517090404)

[Ilustración 12.Relación lineal: verbal-eye 43](#_Toc517090405)

[Ilustración 13. Matriz de correlación parcial 44](#_Toc517090406)

[Ilustración 14.Random Forest. Gráfico OOBError-MTRY 47](#_Toc517090407)

[Ilustración 15.Random Forest. Importancia Variables 49](#_Toc517090408)

[Ilustración 16. Random Forest. Árbol Clasificación 50](#_Toc517090409)

[Ilustración 17. PCA. Varianzas sin normalizar **¡Error! Marcador no definido.**](#_Toc517090410)

[Ilustración 18.PCA. Gráfico sin normalizar 57](#_Toc517090411)

[Ilustración 19.PCA. Varianzas normalizadas 58](#_Toc517090412)

[Ilustración 20.PCA. Gráfico normalizado 58](#_Toc517090413)

[Ilustración 21.PCA. Porcentaje varianzas explicadas 60](#_Toc517090414)

[Ilustración 22. Regresión Logística. Matriz de confusión 68](#_Toc517090415)

[Ilustración 23. Regresión Logística. pAUC 68](#_Toc517090416)

[Ilustración 24. Regresión Logística. Sex-Cause. Matriz de confusión 70](#_Toc517090417)

[Ilustración 25.Regresión Logística. Sex-Cause. pAUC 70](#_Toc517090418)

[Ilustración 26.Naïves Bayes. Mejor ajuste. 73](#_Toc517090419)

[Ilustración 27. Naïve Bayes. Matriz de confusión 74](#_Toc517090420)

[Ilustración 28. Naïve Bayes. pAUC 75](#_Toc517090421)

[Ilustración 29. Random Forest. OOBError-MTRY 76](#_Toc517090422)

[Ilustración 30.Random Forest. Matriz de confusión 78](#_Toc517090423)

[Ilustración 31. Random Forest. pAUC 78](#_Toc517090424)

[Ilustración 32.Adaboost. Mejor ajuste 80](#_Toc517090425)

[Ilustración 33. Adaboost. Error-Iteración 82](#_Toc517090426)

[Ilustración 34. Adaboost. Matriz de confusión 83](#_Toc517090427)

[Ilustración 35. Adaboost. pAUC 83](#_Toc517090428)

[Ilustración 36. GBM. Mejor ajuste 86](#_Toc517090429)

[Ilustración 37. GBM. Matriz de confusión 88](#_Toc517090430)

[Ilustración 38. GBM. pAUC 88](#_Toc517090431)

# Listado de Tablas

[Tabla 1. Nomenclatura del conjunto de datos 25](#_Toc517766869)

[Tabla 2. Modelos utilizados 101](#_Toc517766870)

[Tabla 3. Comparativa AUCse y AUCsp 104](#_Toc517766871)

# INTRODUCCIÓN Y OBJETIVOS

## 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 otra lesionen la cabeza causa 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 se hace necesario realizar un estudio de predicción que nos permita saber *“a posteriori”* las consecuencias de las lesiones en los próximos 6 meses justo después de haber tenido una lesión traumática cerebral.

## OBJETIVOS

Con esta propuesta de Trabajo de Fin de Máster se persigue analizar una serie de datos provenientes de 10008 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, la forma en la que se produjo la lesión y los resultados del paciente habiendo evaluado su estado sobre la escala GSW (escala de Glasgow).

Para conseguir una buena predicción, se probarán varios modelos, teniendo en cuenta ciertas variables y descartando otras. Para la predicción se utilizará el análisis de regresión logística y se tendrá en cuenta la curva ROC, para detectar si el modelo utilizado se ajusta o no, es decir, si el modelo es bueno para realizar predicciones.

La meta a la que se pretende llegar con este trabajo es, en definitiva, obtener un patrón a partir de 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 objetivos que se han definido para este trabajo son los siguientes:

* 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.

## 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.

# ESTADO DEL ARTE

## BIG DATA

Cuando hablamos de Big Data nos referimos a conjuntos de datos o combinaciones de conjuntos de datos cuyo tamaño (volumen), complejidad (variabilidad) y velocidad de crecimiento (velocidad) dificultan su captura, gestión, procesamiento o análisis mediante tecnologías y herramientas convencionales, como por ejemplo bases de datos relacionales, estadísticas convencionales o paquetes de visualización que nos permitan observar estos volúmenes de datos de forma inmediata.

En estas últimas décadas, el termino de Big Data ha sido y es bastante revolucionario, puesto que, aunque antes ya se utilizaba este concepto (cuando en un artículo de la revista Harper’s Magazine se habla por primera vez de Big Data en 1989), no ha sido hasta comienzos del siglo XXI cuando se han empezado a plantear los problemas sobre el uso y almacenamiento de grandes volúmenes de datos. Ya en 2005 nace Hadoop, un entorno de trabajo de Big Data. Pero no es hasta 2016 cuando la palabra “Big Data” se convierte en la palabra de moda.

Centrándonos en el gran volumen de datos y de acuerdo al IDC (International Data Corporation), el volumen total de datos en 2013 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 2020 se alcancen los 40ZB, unos 5247GB por persona.

A su vez, se hace necesario el análisis de estas grandes cantidades de datos con el objetivo -entre otros- de facilitar a las organizaciones el aprovechamiento de estos datos para identificar nuevas oportunidades. Este aprovechamiento se consigue mediante la recopilación y la búsqueda de tendencia dentro de estos.

## PRINCIPALES FASES DE BIG DATA

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:
   1. GitHub
   2. Amazon
   3. Facebook
   4. Twitter
   5. 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 y algoritmos:** 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 de datos**: Se utiliza aplicaciones como PowerBI, Pentaho, QlikView, PeriscopeData e incluso documentos Excel, con el objetivo de visualizar y obtener resultados dinámicos.
7. **Comunicación / visualización de datos:** Se realizan informes por audiencia (comerciales, marketing, estrategia, dirección, técnicos, etc).

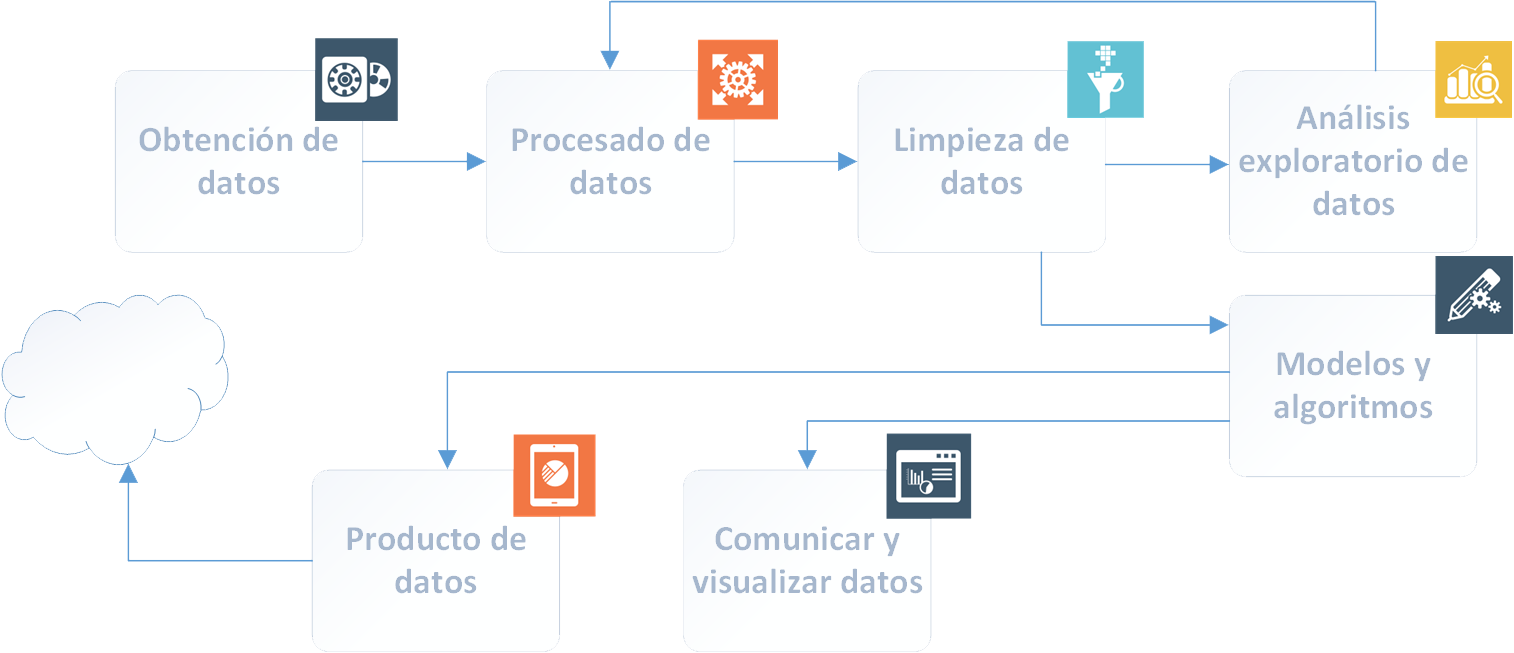


Ilustración 1. Fases de Big Data

Con nuestro proyecto buscaremos alcanzar y tratar los 5 primeros puntos anteriormente comentados.

## HERRAMIENTAS UTILIZADAS EN BIG DATA

A continuación, se van a describir algunas de las herramientas que se utilizan para realizar los procesos de Big Data:

1. **Hadoop**

Es una 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.

1. **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.

1. **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*.*

1. **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.

1. **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 Big Data.

# EVALUACIÓN DE RIESGOS

En esta sección, vamos a valorar las diferentes herramientas y tecnologías a utilizar en el desarrollo de nuestro estudio.

## 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 análisis simulando al previo realizado en el informe original, 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.

# DESARROLLO

Como ya se comentó en la sección: 2.2 Principales fases de Big Data. 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.

## 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 el contenido de información contenida en el conjunto de datos pueda ser expuesta o más fácilmente accesible.

Es de gran importancia realizar una preparación de los datos, para posteriormente realizar análisis sobre estos.

### Fuente de obtención de los datos

Los datos a utilizar en el estudio realizado con este trabajo, se han obtenido del siguiente repositorio de datos: <https://ctu-app.lshtm.ac.uk/freebird/index.php/available-trials/>

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**

### Nomenclatura del conjunto de los 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.

### 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.

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

* EO\_Outcome
* EO\_Symptoms
* TH\_Outcome
* TH\_Symptoms
* GOS5
* GOS8

#### Cuando las variables de GOS5 y GOS8 tienen datos

Si las filas ya contenían datos en las columnas de GOS5 y GOS8, directamente se han clasificado -según estas variables-. De lo contrario, se ha tenido que analizar las otras 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-

#### Cuando las variables de GOS5 y GOS8 no tienen datos

Si las variables de *“outcome”* contenían el valor de 1 (fallecimiento) o las variables de *“Symptoms”* contenían el valor de 6, directamente esas filas del conjunto de datos pasaban a clasificarse como fallecidos.

## 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 variables de *“outcome”* contenían el valor de 4 (alta) y las de “*Symptoms”* el valor de 1, entonces se han clasificado como “Vivos (pero sin resultados finales)”.

## 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 ni en las columnas de “*Symptoms”*. 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 los “*Symptoms*” son de 4 o de 5 (Discapacidad Severa), entonces se clasificarán como “Fallecidos o con discapacidades severas”.

## 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 variables de *“outcome”* 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, se han incluido en la clasificación de “Vivos (pero sin resultados finales)”.

## 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

Se han visto 3 elementos de “NODATA”, cuyos pacientes obtienen un estado de *“Symptoms”* de 4, por lo que se envía a estado de fallecido, 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

Datos Finales:

* 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

### Clasificación entre: ESCANEADOS y NO ESCANEADOS

En primer lugar, se han encontrado ciertos datos anómalos, en los que aparecen datos escaneados (1) y no tienen los datos del escáner, entonces deberíamos ponerlo como no escaneado (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
* No escaneados
* En análisis

Si el *“Outcome”* es 2 (el paciente se ha transferido a otro hospital), se ha escaneado en dicho hospital (“TH\_SCAN”) y no se tiene ninguna información en los escáneres, 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 el *“Outcome”* es 2 (el paciente se ha transferido a otro hospital) y no se ha realizado ningún escáner, pero si contiene datos en el escáner, entonces se clasificará 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

Sobre 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).

## 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

Sobre el dataset **EN ANALISIS**: Nos hemos dado cuenta de 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”.*

## 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.

## 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).

## 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) 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

### Eliminación y centralización de variables

Se va a centralizar las variables de *“PUPIL\_REACT\_LEFT”* y *“PUPIL\_REACT\_RIGHT”.*

## 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

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

Ahora vamos a ver si podemos prescindir o aunar 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 aunar 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 primeros resultados de escáner obtenidos en el primer. Además, eliminaremos todas las variables que no se utilicen.

## 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

## 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 y optimo como los datos que se utilicen.

### Búsqueda de outliers (datos anómalos)

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

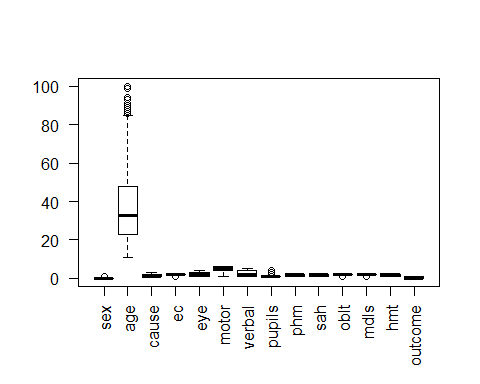


Ilustración 2. Diagrama de caja. Outliers

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.

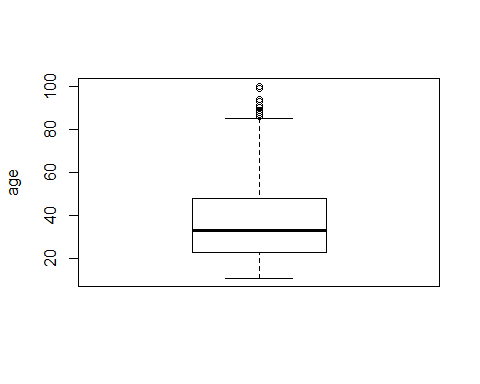


Ilustración 3. Diagrama de caja. Outliers. Edad

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)

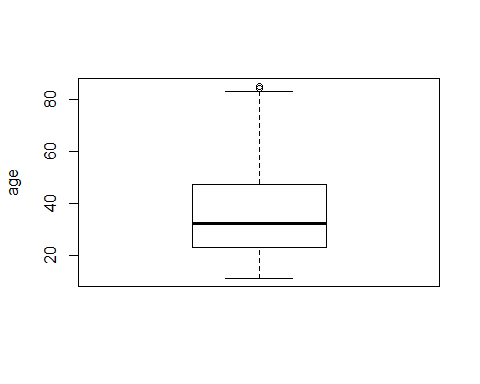


Ilustración 4. Diagrama de caja. Edad. Eliminando Outliers

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.

### 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.

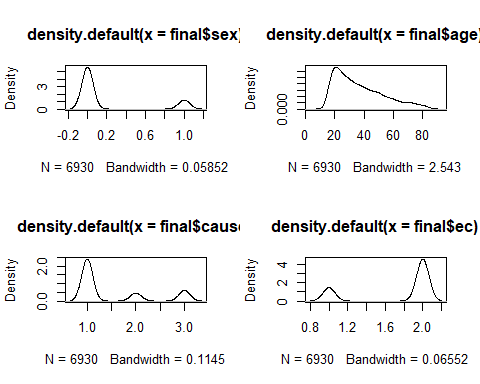


Ilustración 5. Gráfico Normalidad I

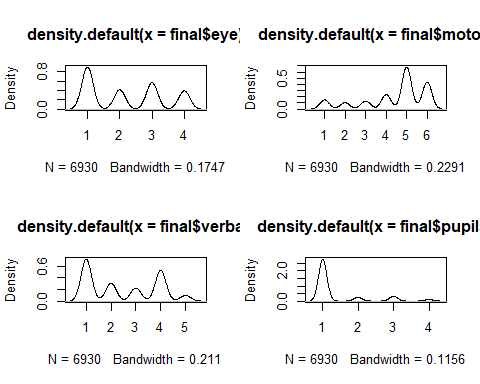


Ilustración 6. Gráfico Normalidad II

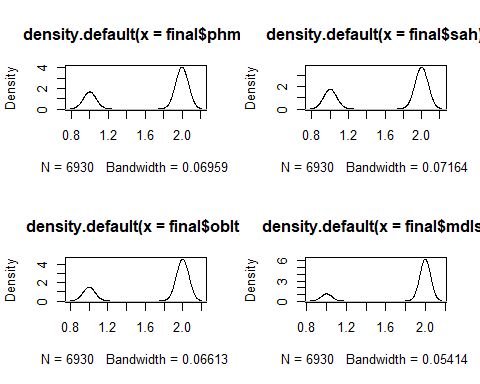


Ilustración 7. Gráfico Normalidad III

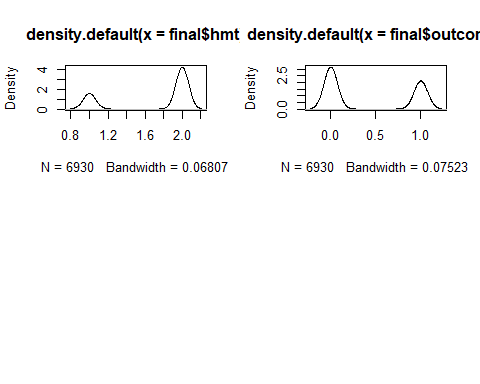


Ilustración 8. Gráfico Normalidad IV

Como podemos comprobar, al tratarse de variables discretas (excepto la variable de edad -*age*-), 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:

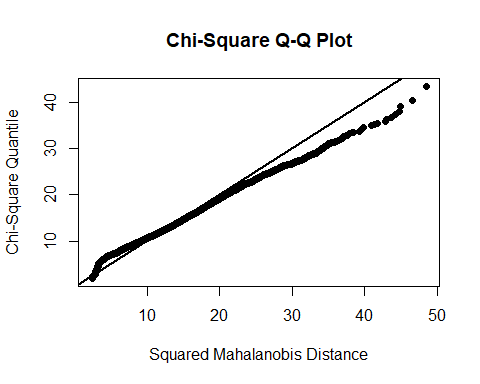


Ilustración 9. Chi-Square-Mahalanobis

## 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. Tampoco se pueden realizar otras labores para obtener un mejor rendimiento puesto que la mayoría de las variables son discretas.

### 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.

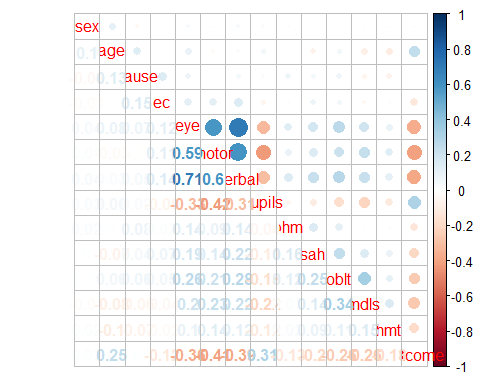


Ilustración 10. Matriz de correlación I

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.

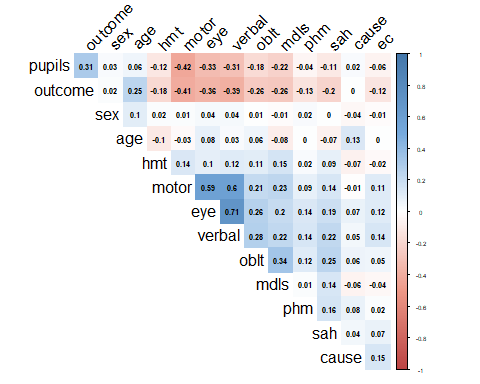


Ilustración 11. Matriz de correlación II

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

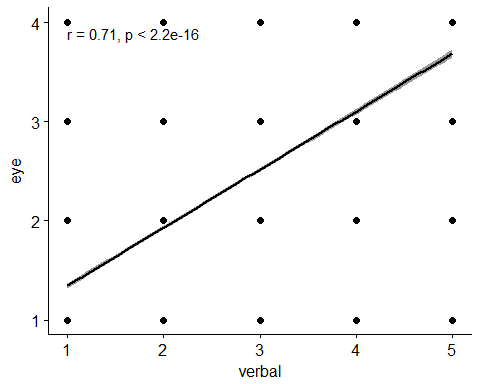


Ilustración 12. Relación lineal: verbal-eye

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:

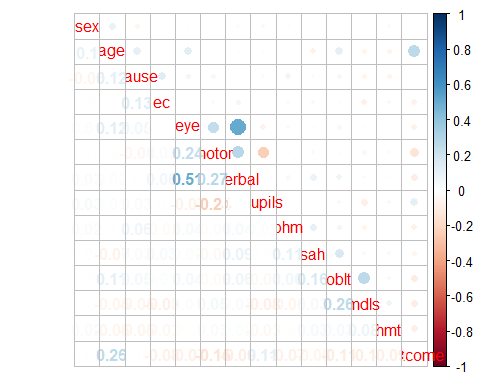


Ilustración 13. Matriz de correlación parcial

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

Por consiguiente, 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.

### 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.

#### 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.

La idea que existe detrás de los *“Random Forest”* es generar un número importante de árboles, entrenarlos y calcular el promedio de su salida.

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.

En cada paso (nodo) se recalcula el conjunto de **“m”** predictores permitidos. Lo más típico es elegir la raíz cuadrada del número total de variables. En nuestro caso, contamos con un total de 13 variables, por lo que se escogerían 4 variables (redondeando hacia arriba en caso de no ser un número entero) en el caso de **árboles de clasificación** y **m=p/3** en el caso de **árboles de regresión**. Siendo **“p”** el número de variables.

Aun así, es necesario calcular la variable **“mtry”**, puesto que es el único parámetro ajustable al cual los bosques aleatorios son algo sensibles. El “mtry” es el número de variables aleatorias utilizadas en cada árbol. La reduccion del “mtry” reduce tanto la correlación como la fuerza, aumentando ambas en caso contrario.

En algún punto intermedio hay un rango “óptimo” de “mtry”, generalmente bastante ancho. Usando la tasa de error de OOB, se puede encontrar rápidamente un valor de mínimo en el rango.

## mtry OOBError  
## 1 1 6.830054e-02  
## 2 2 9.155807e-03  
## 4 4 2.020574e-04  
## 8 8 8.633286e-07  
## 14 14 1.526225e-07

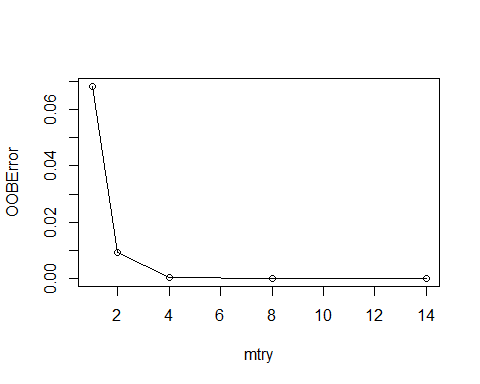


Ilustración 14. Random Forest. Gráfico OOBError-MTRY

Como se puede comprobar, el error OOB, se estabiliza, indicando cuantas particiones se deben realizar para obtener los mejores resultados En este caso, con **4 variables sería suficiente** (puesto que es el numero donde se estabiliza el error OOB).

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:

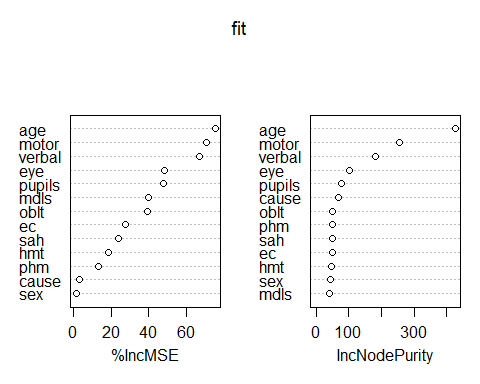


Ilustración 15. Random Forest. Importancia Variables

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

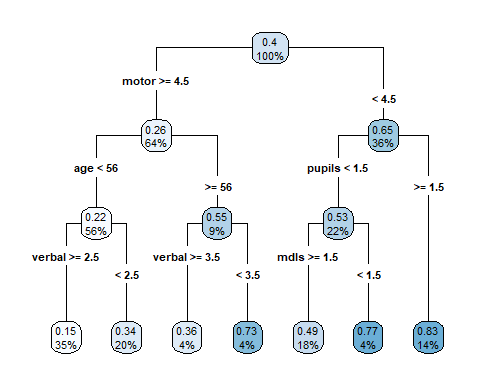


Ilustración 16. Random Forest. Árbol Clasificación

## n= 6930   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 6930 1664.39500 0.4008658   
## 2) motor>=4.5 4458 864.83090 0.2633468   
## 4) age< 56.5 3861 661.19090 0.2193732   
## 8) verbal>=2.5 2445 306.28790 0.1468303 \*  
## 9) verbal< 2.5 1416 319.81920 0.3446328 \*  
## 5) age>=56.5 597 147.88940 0.5477387   
## 10) verbal>=3.5 296 68.32095 0.3614865 \*  
## 11) verbal< 3.5 301 59.20266 0.7308970 \*  
## 3) motor< 4.5 2472 563.21680 0.6488673   
## 6) pupils< 1.5 1504 374.27060 0.5339096   
## 12) mdls>=1.5 1260 314.79680 0.4873016 \*  
## 13) mdls< 1.5 244 42.60246 0.7745902 \*  
## 7) pupils>=1.5 968 138.18900 0.8274793 \*

Si recordamos, un resultado en el *“outcome”* de 0 eran aquellos pacientes que a los 6 meses habían vivido mientras que un resultado de 1, significaba que los pacientes fallecían. Teniendo en cuenta este dato, podemos observar que los nodos del árbol son aquellas variables que el algoritmo considera más relevantes y que hacen que un paciente viva o fallezca.

La interpretación que se da al árbol es la siguiente: Cada nodo contiene el porcentaje de información que contiene además de la media de la variable *“outcome”* en cada partición. Por ejemplo, la media de *“outcome”* es de 0.4, que coincide con el 0.4 del nodo raíz. Sin embargo, cuando la variable *“motor”* es mayor de 4.5, entonces el número de datos se reduce al 64% y la media de *“outcome”* se vuelve a 0.26, significando para este caso que la mayoría de los pacientes viven, puesto que se aproxima a 0.

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”*.

#### Uso del método de regresión paso a paso (Stepwise Regression)

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.

##   
## 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 significantes excepto *“age”* y *“cause”*, cuyo p-valor es mayor a 0.05.

A continuación, utilizamos el algoritmo de regresión paso a paso:

## 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.

A continuación se va a utilizar la función train() del paquete *caret* que proporciona un flujo de trabajo sencillo para realizar selecciones paso a paso. Para esta prueba se ha utilizado la validación cruzada con 10 iteraciones para estimar el error promedio de predicción (RMSE) de cada uno de los modelos. La métrica estadística RMSE se usa para comparar los modelos y elegir automáticamente el mejor, donde el mejor se define como el modelo que minimiza el RMSE.

Se ha utilizado el parámetro “nvmax” que corresponde al número máximo de predictores que se incorporarán en el modelo. Nosotros hemos utilizado todos.

El parámetro “nvmax” busca el mejor modelo de 1 variable, el mejor modelo de 2 variables, de 3, etc.

Una vez que construido el modelo con la función train(), se muestra el número de variables que se utilizan para construir el mejor modelo.

## nvmax  
## 11 11

Por ultimo mostramos todos los modelos creados utilizando la función summary().

## Subset selection object  
## 13 Variables (and intercept)  
## Forced in Forced out  
## sex FALSE FALSE  
## age FALSE FALSE  
## cause FALSE FALSE  
## ec FALSE FALSE  
## eye FALSE FALSE  
## motor FALSE FALSE  
## verbal FALSE FALSE  
## pupils FALSE FALSE  
## phm FALSE FALSE  
## sah FALSE FALSE  
## oblt FALSE FALSE  
## mdls FALSE FALSE  
## hmt FALSE FALSE  
## 1 subsets of each size up to 11  
## Selection Algorithm: 'sequential replacement'  
## sex age cause ec eye motor verbal pupils phm sah oblt mdls hmt  
## 1 ( 1 ) " " " " " " " " " " "\*" " " " " " " " " " " " " " "  
## 2 ( 1 ) " " "\*" " " " " " " "\*" " " " " " " " " " " " " " "  
## 3 ( 1 ) " " "\*" " " " " " " "\*" "\*" " " " " " " " " " " " "  
## 4 ( 1 ) " " "\*" " " " " " " "\*" "\*" " " " " " " "\*" " " " "  
## 5 ( 1 ) " " "\*" " " " " " " "\*" "\*" "\*" " " " " "\*" " " " "  
## 6 ( 1 ) " " "\*" " " " " " " "\*" "\*" "\*" " " " " "\*" "\*" " "  
## 7 ( 1 ) " " "\*" " " " " " " "\*" "\*" "\*" " " " " "\*" "\*" "\*"  
## 8 ( 1 ) " " "\*" " " " " " " "\*" "\*" "\*" "\*" " " "\*" "\*" "\*"  
## 9 ( 1 ) " " "\*" " " "\*" " " "\*" "\*" "\*" "\*" " " "\*" "\*" "\*"  
## 10 ( 1 ) " " "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*" "\*"  
## 11 ( 1 ) " " "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

NOTA: Un asterisco especifica que la variable dada se incluye en el modelo correspondiente.

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.

#### Análisis de 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

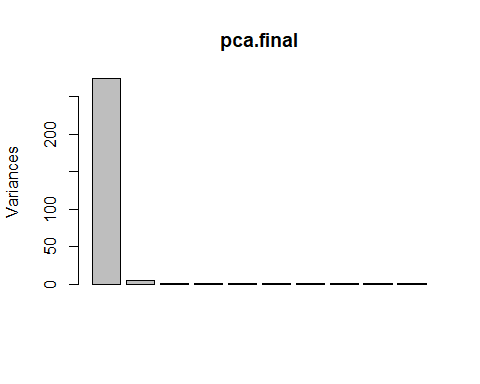


Ilustración 17. PCA. Varianzas sin normalizar

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

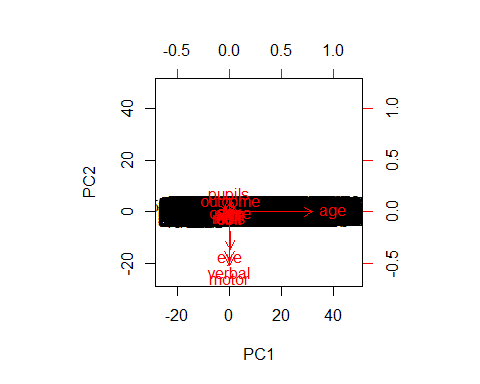


Ilustración 18. PCA. Gráfico sin normalizar

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

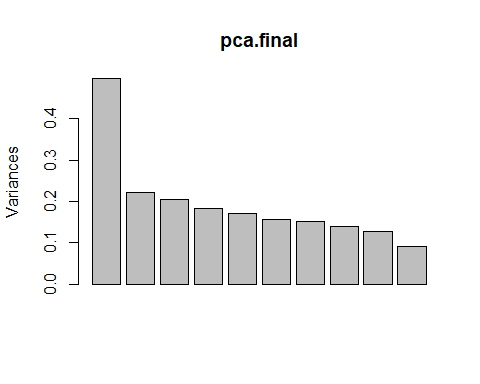


Ilustración 19. PCA. Varianzas normalizadas

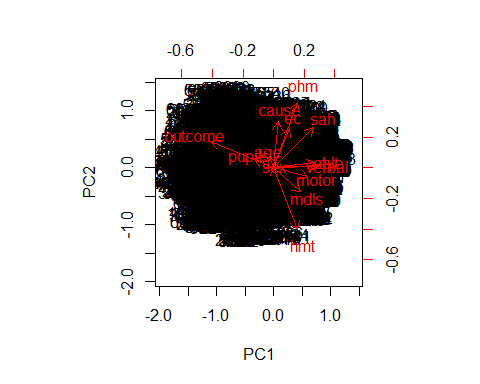


Ilustración 20. PCA. Gráfico normalizado

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

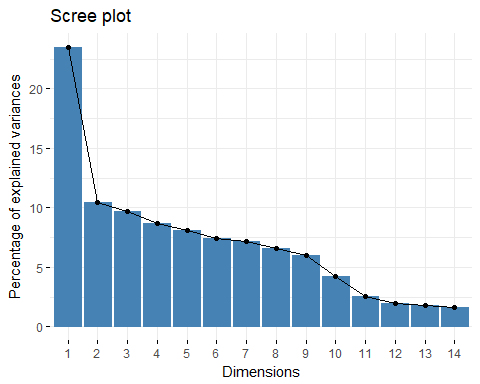
* + Como se puede comprobar, utilizando este criterio, podríamos quedarnos con los componentes PC1, PC2, PC3, PC4 y PC5.
* Otra forma para saber cuántos componentes tener en cuenta es mantener el número de componentes necesarios para explicar al menos un porcentaje del total de la varianza. Por ejemplo, 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 21. 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.

## MODELADO DE DATOS

Una vez que hemos realizado el pre-procesado de datos, y hemos el respectivo analisis 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.

### Términos generales

#### 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 siguientes 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)

#### Métricas de evaluación de modelos

##### En modelos de regresión

RMSE y R2 son las métricas predeterminadas que se usan para evaluar algoritmos de regresión.

* **R2** (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.

##### 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).

##### 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.

#### 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.

### 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.

### Uso de modelos

#### 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.

#### 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

Una vez hecho esto, vamos a utilizar los datos de test para predecir el modelo que hemos construido y comprobaremos el ajuste de este.

## 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   
##

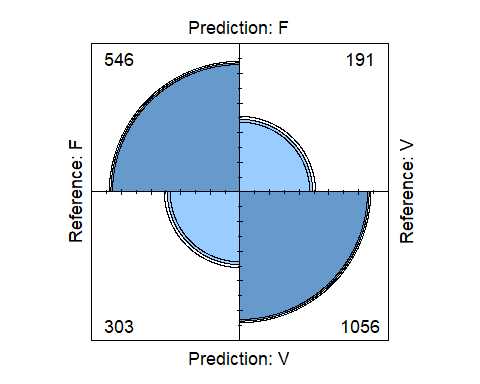


Ilustración 22. Regresión Logística. Matriz de confusión

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.

A continuación, vamos a utilizar la curva de ROC con los valores de pAUC:

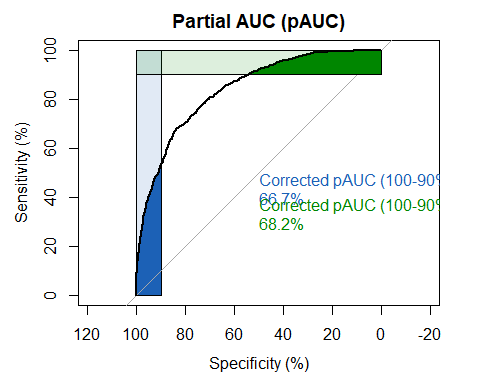


Ilustración 23. Regresión Logística. pAUC

Si nosotros tomamos un rango concreto de Especificidad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 66,7% de casos negativos y que realmente son negativos, es decir seremos capaces de predecir que un paciente que vive, realmente va a vivir.

Si nosotros tomamos un rango concreto de sensitividad, por ejemplo, del 90% a 100%, entonces seremos capaces de predecir un 68,2% de casos positivos y que realmente son positivos, es decir seremos capaces de predecir que un paciente que fallece, realmente va a fallecer.

##### Eliminando los predictores “sex” y “cause”

Al igual que hicimos en el modelo anterior, vamos a construir el modelo eliminando las variables de “sex” y “cause”.

Una vez hecho esto, vamos a utilizar los datos de test para predecir el modelo que hemos construido y comprobaremos el ajuste de este.

## 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   
##

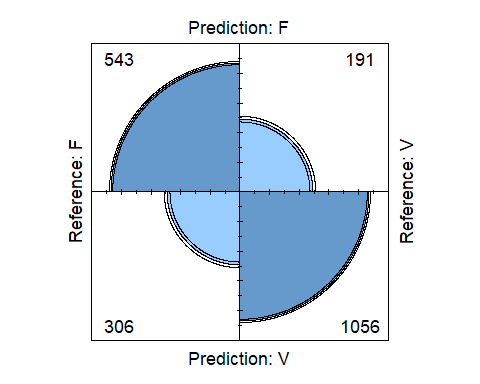


Ilustración 24. Regresión Logística. Sex-Cause. Matriz de confusión

Como podemos comprobar, aunque el AIC ha mejorado (reducido su valor) utilizando este modelo (sin las variables de sexo y causa) respecto al modelo con todas las variables, ha empeorado las predicciones.

A continuación, vamos a utilizar la curva de ROC con los valores de pAUC:

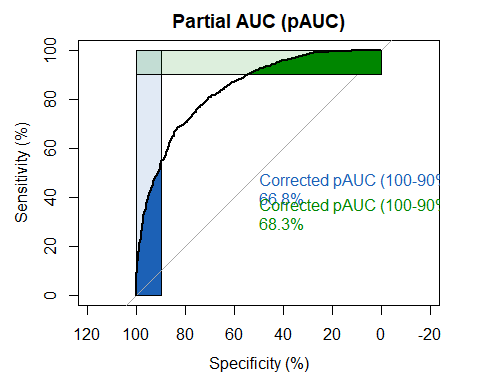


Ilustración 25. Regresión Logística. Sex-Cause. pAUC

#### 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
* fL: la corrección de Laplace
* adjust: Ajuste de ancho de banda

## 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.

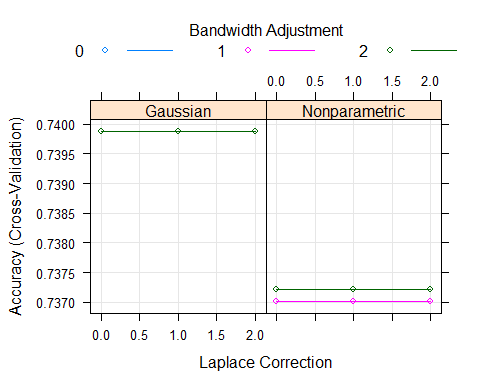


Ilustración 26. Naïves Bayes. Mejor ajuste.

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

* fL: 0
* usekernel: FALSE
* adjust: 0

Una vez hecho esto, pasamos a predecir usando el modelo construido y los datos de test. Mostramos la 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   
##

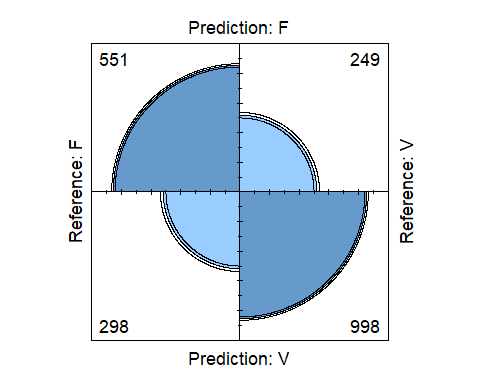


Ilustración 27. Naïve Bayes. Matriz de confusión

A continuación, vamos a utilizar la curva de ROC con los valores de pAUC:

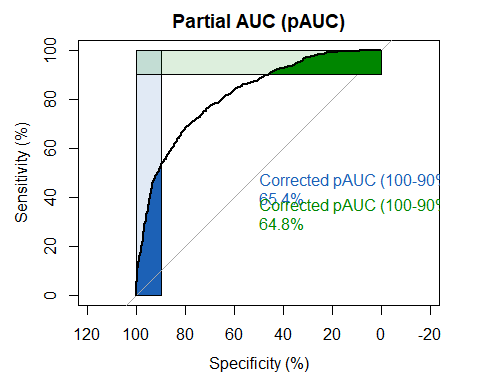


Ilustración 28. Naïve Bayes. pAUC

#### 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.
   1. 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.

##   
## Call:  
## randomForest(formula = outcome ~ ., data = train, ntree = 500)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 23.56%  
## Confusion matrix:  
## F V class.error  
## F 1265 715 0.3611111  
## V 437 2473 0.1501718

## mtry = 1 OOB error = 25.13%   
## Searching left ...  
## Searching right ...  
## mtry = 2 OOB error = 23.39%   
## 0.06916192 0.01   
## mtry = 4 OOB error = 23.87%   
## -0.0201049 0.01

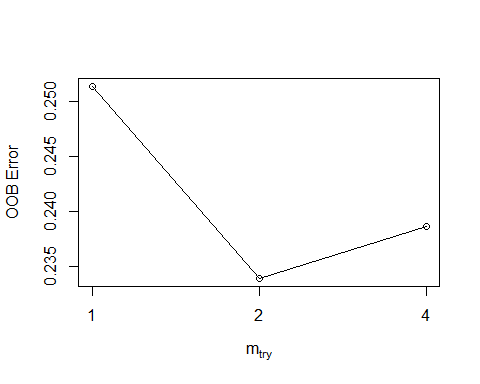


Ilustración 29. Random Forest. OOBError-MTRY

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

Utilizaremos el valor de 2 como “mtry”, debido a que es el que menos error consigue.

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

##   
## Call:  
## randomForest(formula = outcome ~ ., data = train, ntree = 500)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 23.56%  
## Confusion matrix:  
## F V class.error  
## F 1265 715 0.3611111  
## V 437 2473 0.1501718

Después de construir el modelo, vamos a utilizar los datos de test para predecirlo. Comprobaremos una vez más el ajuste de este.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction F V  
## F 525 153  
## V 324 1094  
##   
## Accuracy : 0.7724   
## 95% CI : (0.7539, 0.7902)  
## No Information Rate : 0.5949   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5121   
## Mcnemar's Test P-Value : 7.04e-15   
##   
## Sensitivity : 0.6184   
## Specificity : 0.8773   
## Pos Pred Value : 0.7743   
## Neg Pred Value : 0.7715   
## Prevalence : 0.4051   
## Detection Rate : 0.2505   
## Detection Prevalence : 0.3235   
## Balanced Accuracy : 0.7478   
##   
## 'Positive' Class : F   
##

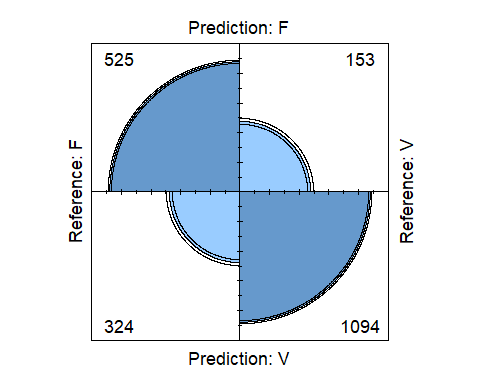


Ilustración 30. Random Forest. Matriz de confusión

A continuación, podemos observar la curva de ROC con los valores de pAUC:

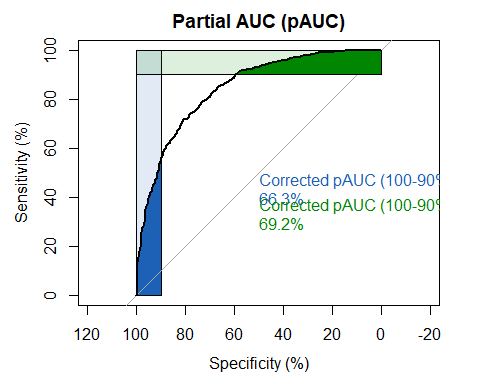


Ilustración 31. Random Forest. pAUC

#### 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.

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.

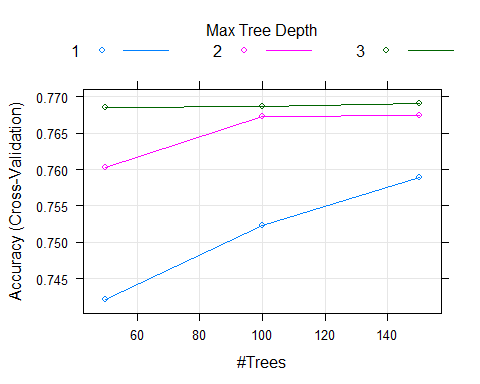


Ilustración 32. Adaboost. Mejor ajuste

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 = 141, nu = 0.3, control = rpart.control(maxdepth = 1))  
##   
## Loss: exponential Method: discrete Iteration: 141   
##   
## Final Confusion Matrix for Data:  
## Final Prediction  
## True value F V  
## F 1258 722  
## V 389 2521  
##   
## Train Error: 0.227   
##   
## Out-Of-Bag Error: 0.231 iteration= 132   
##   
## Additional Estimates of number of iterations:  
##   
## train.err1 train.kap1   
## 116 140

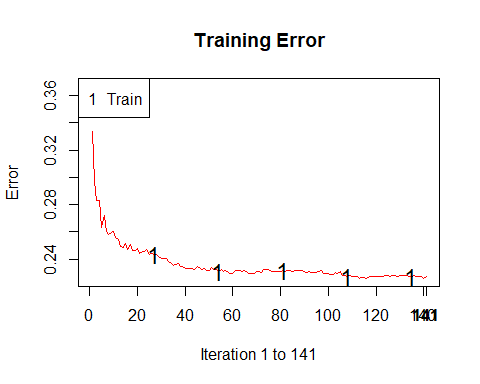


Ilustración 33. Adaboost. Error-Iteración

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction F V  
## F 533 164  
## V 316 1083  
##   
## Accuracy : 0.771   
## 95% CI : (0.7524, 0.7888)  
## No Information Rate : 0.5949   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5109   
## Mcnemar's Test P-Value : 5.495e-12   
##   
## Sensitivity : 0.6278   
## Specificity : 0.8685   
## Pos Pred Value : 0.7647   
## Neg Pred Value : 0.7741   
## Prevalence : 0.4051   
## Detection Rate : 0.2543   
## Detection Prevalence : 0.3325   
## Balanced Accuracy : 0.7481   
##   
## 'Positive' Class : F   
##

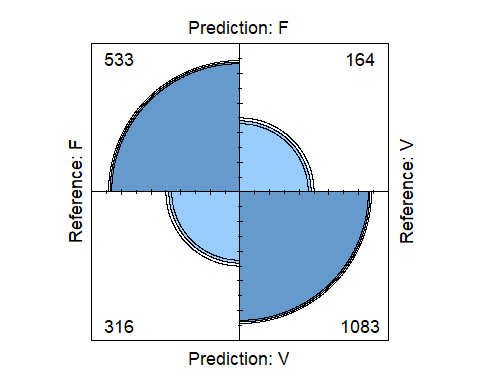


Ilustración 34. Adaboost. Matriz de confusión

A continuación, podemos observar la curva de ROC con los valores de pAUC:

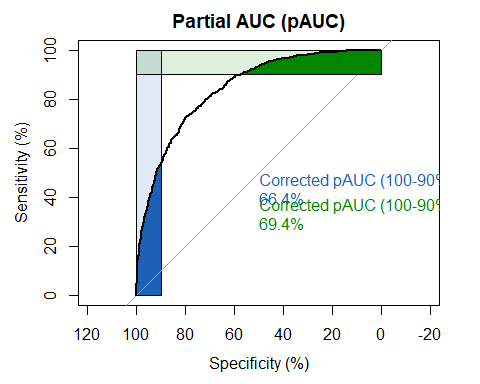


Ilustración 35. Adaboost. pAUC

#### 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) + …

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.

A continuación, construimos el modelo automáticamente utilizando el paquete de “caret”, igual que en ejemplos anteriores se ha utilizado la validación cruzada.

## 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.

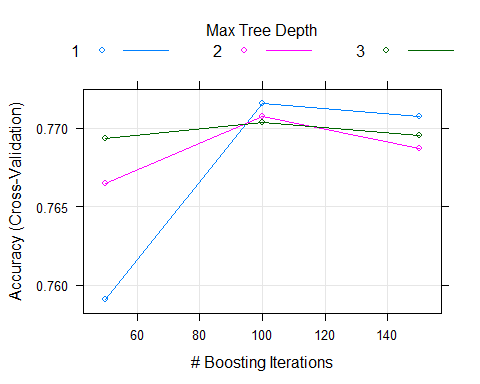


Ilustración 36. GBM. Mejor ajuste

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.

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 538 169  
## V 311 1078  
##   
## Accuracy : 0.771   
## 95% CI : (0.7524, 0.7888)  
## No Information Rate : 0.5949   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5118   
## Mcnemar's Test P-Value : 1.229e-10   
##   
## Sensitivity : 0.6337   
## Specificity : 0.8645   
## Pos Pred Value : 0.7610   
## Neg Pred Value : 0.7761   
## Prevalence : 0.4051   
## Detection Rate : 0.2567   
## Detection Prevalence : 0.3373   
## Balanced Accuracy : 0.7491   
##   
## 'Positive' Class : F   
##

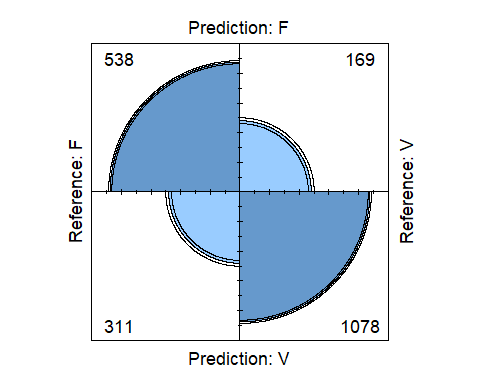


Ilustración 37. GBM. Matriz de confusión

A continuación, podemos observar la curva de ROC con los valores de pAUC:

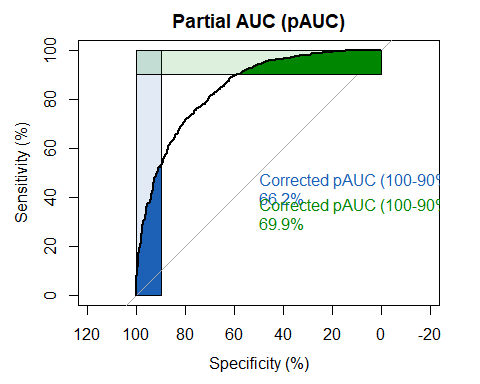


Ilustración 38. GBM. pAUC

#### 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.

Utilizando caret en para el entrenamiento obtenemos la siguiente información:

## 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:  
##   
## max\_depth colsample\_bytree Accuracy Kappa   
## 3 0.5 0.7691207 0.5103336  
## 3 0.7 0.7697342 0.5112361  
## 6 0.5 0.7666667 0.5055645  
## 6 0.7 0.7613497 0.4938099  
##   
## Tuning parameter 'nrounds' was held constant at a value of 100  
##   
## Tuning parameter 'min\_child\_weight' was held constant at a value of  
## 2  
## 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 = 100, max\_depth = 3,  
## eta = 0.1, gamma = 1, colsample\_bytree = 0.7, min\_child\_weight = 2  
## and subsample = 1.

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

* Nrounds =100
* Max\_depth = 3
* Eta = 0.1
* Gamma = 1
* Colsample\_bytree = 0.7
* Min\_child\_weight = 2
* Subsample = 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 551 177  
## V 298 1070  
##   
## Accuracy : 0.7734   
## 95% CI : (0.7548, 0.7911)  
## No Information Rate : 0.5949   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5189   
## Mcnemar's Test P-Value : 3.671e-08   
##   
## Sensitivity : 0.6490   
## Specificity : 0.8581   
## Pos Pred Value : 0.7569   
## Neg Pred Value : 0.7822   
## Prevalence : 0.4051   
## Detection Rate : 0.2629   
## Detection Prevalence : 0.3473   
## Balanced Accuracy : 0.7535   
##   
## 'Positive' Class : F   
##

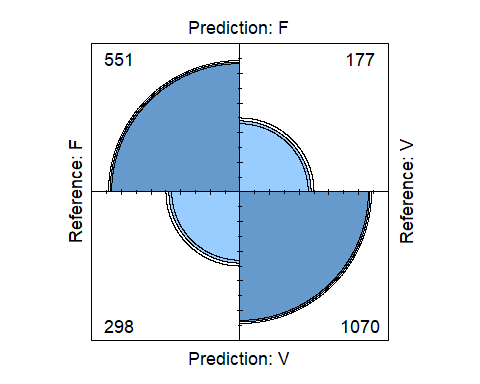


Ilustración 39. XGBoost. Matriz de confusión

A continuación, podemos observar la curva de ROC con los valores de pAUC:

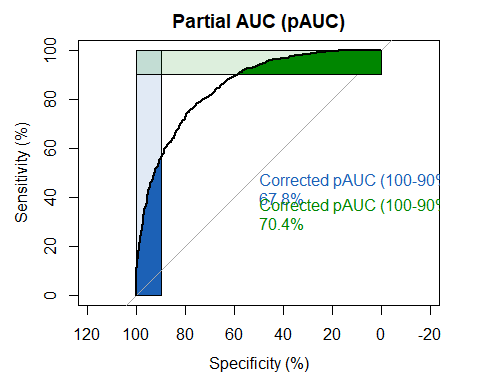


Ilustración 40. XGBoost. pAUC

#### 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.

##### Mejorar la precisión de nuestro modelo

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.

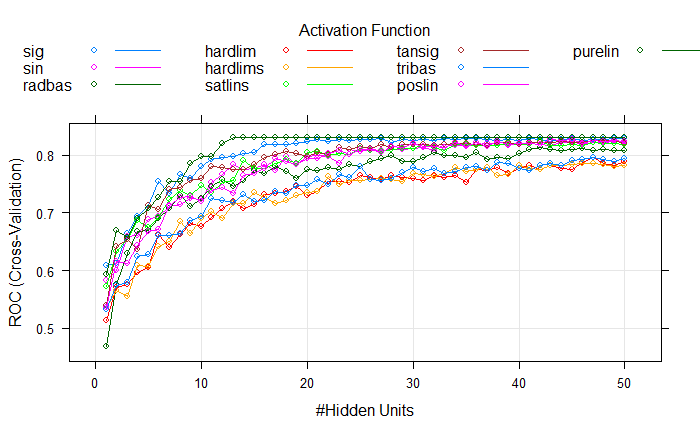


Ilustración 41. Redes Neuronales. Mejor ajuste

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.

## 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   
##

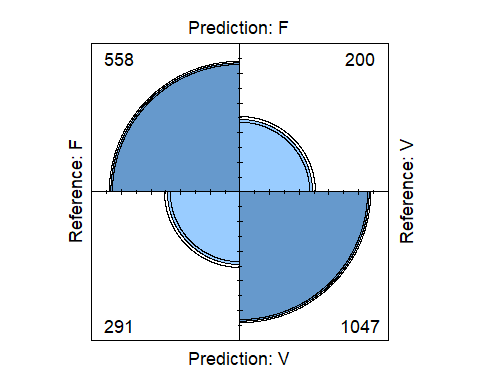


Ilustración 42. Redes Neuronales. Matriz de confusión

A continuación, podemos observar la curva de ROC con los valores de pAUC:

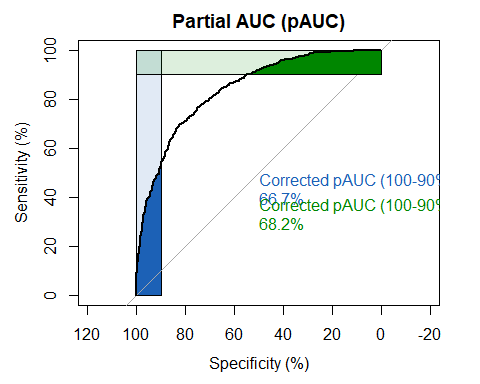


Ilustración 43. Redes Neuronales. pAUC

##### 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. Matriz de correlación II, las variables más correladas son “Verbal” y “Eye”. Teniendo en cuenta esto, vamos a eliminar la variable “Verbal”.

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   
##

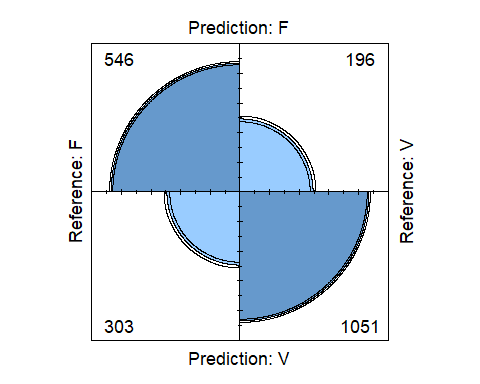


Ilustración 44. Redes Neuronales. Verbal. Matriz de confusión

A continuación, podemos observar la curva de ROC con los valores de pAUC:

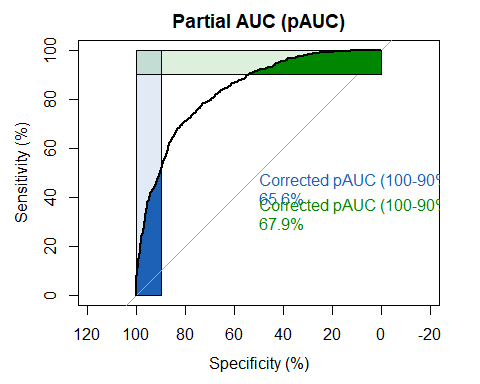


Ilustración 45. Redes Neuronales. Verbal. pAUC

# 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 Términos generales](#_Términos_generales). 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

### USANDO LA METRICA DE PRECISION Y KAPPA

#### Uso de Precisión

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

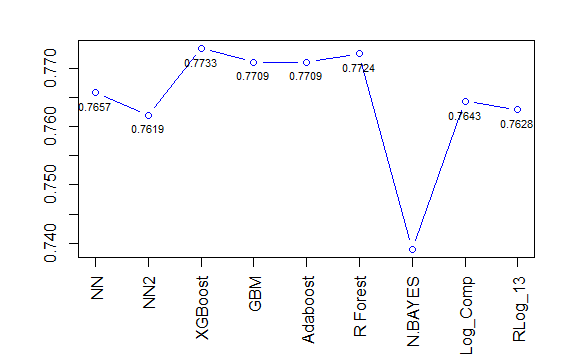


Ilustración 46. Comparativa de precisión

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

El peor resultado lo hemos obtenido con el algoritmo de Naïves Bayes.

#### 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. Modelos utilizados”

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

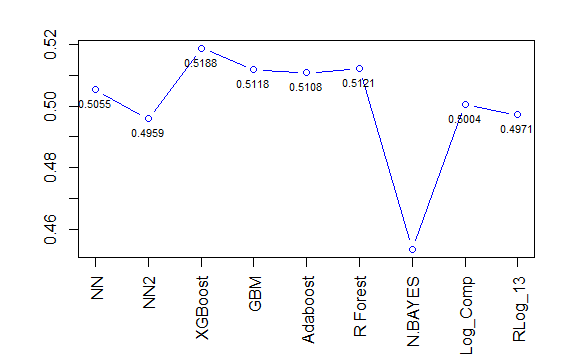


Ilustración 47. Comparativa de Kappa

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 XGBoost el que obtiene las mejores predicciones para nuestro conjunto de datos y el modelo de Naïves Bayes como el peor.

### USANDO LA METRICA 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.

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.

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).

|  |  |  |
| --- | --- | --- |
| **MODELO** | **AUCsp** | **AUCse** |
| NN | 66.71931 | 68.22199 |
| NN2 | 65.60533 | 67.88797 |
| XGBoost | 67.82727 | 70.38546 |
| GBM | 66.21591 | 69.93690 |
| Adaboost | 66.41074 | 69.43082 |
| Random Forest | 66.32583 | 69.19975 |
| 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

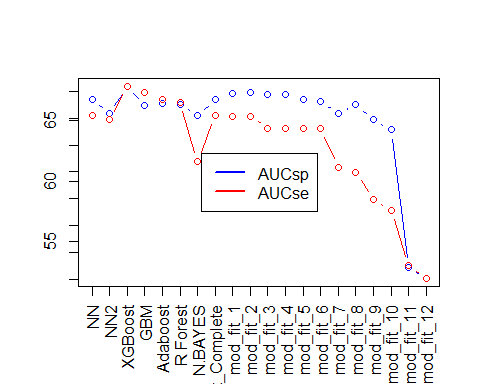


Ilustración 48. Comparativa AUCsp y AUCse

Como se puede observar en el grafico anterior, el máximo AUC de Especificidad se consigue con el modelo XGBoost 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 XGBoost 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 XGBoost, 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 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.

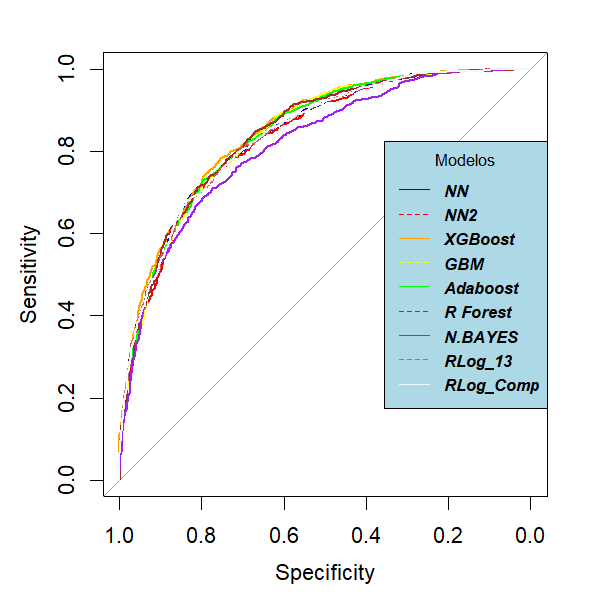


Ilustración 49. Comparativa curva ROC I

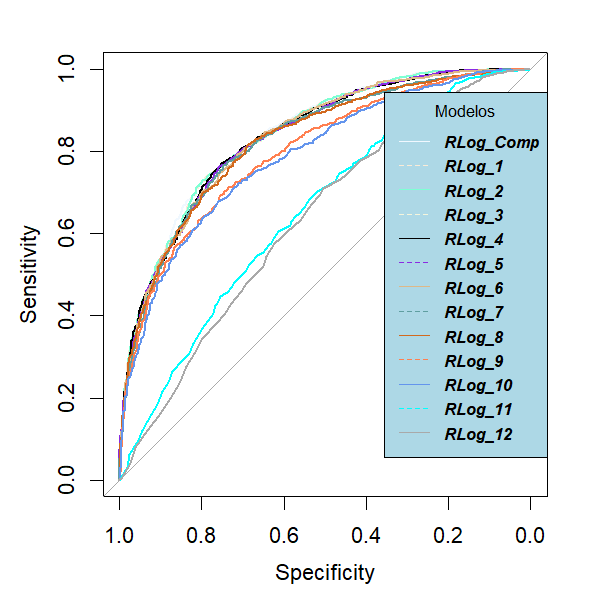


Ilustración 50. Comparativa curva ROC II

### USANDO LA COMBINACION 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.

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
* **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 submodelos.

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).

##   
## 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.

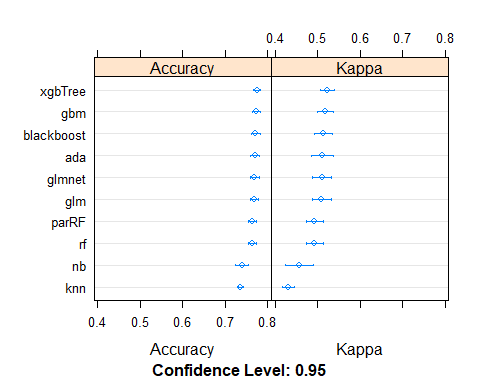


Ilustración 51. CaretEnsemble. Comparación de modelos

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).

## The following models were ensembled: glm, nb, rf, ada, gbm, xgbTree, blackboost, 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 -8.2099  
## 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

## glm nb rf ada gbm xgbTree  
## F vs. V 0.8387773 0.8163536 0.8440181 0.8485652 0.8439765 0.8438188  
## blackboost parRF knn glmnet ensemble  
## F vs. V 0.8440672 0.8439435 0.8202584 0.839054 0.8464272

Como podemos comprobar, en los resultados anteriores, que se han medido utilizando el AUC, vemos que la combinación (“*ensemble*”) de todos los modelos es el que mejor resultado AUC nos proporciona. No siempre es así, muchas veces, aunque se utilice una combinación, no se obtiene mejores resultados que utilizando modelos parciales.

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*”.

A continuación, utilizaremos un modelo de intercalación que se combinara con los resultados del resto de modelos.

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

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.

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. No obstante, también es importante comentar que GBM es un modelo más complejo y requiere de mayor tiempo de computación. Algo a tener en cuenta a la hora de seleccionar un modelo.

### 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:

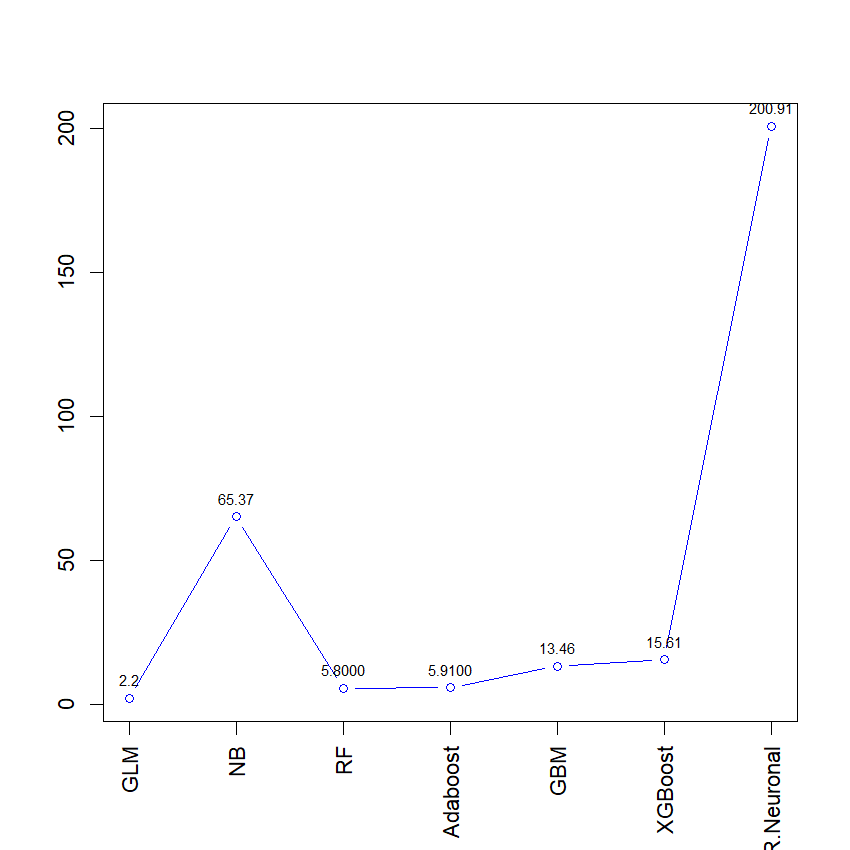


Ilustración 52. Comparativa de tiempos

Como se puede observar, el modelo de redes neuronales es el que más tiempo ha gastado y tampoco ha sido un modelo que ha tenido buenos resultados, por lo tanto, se trata de un modelo, que, para nuestro conjunto de datos, ha obtenido un rendimiento bastante negativo. Ocurre lo mismo con el modelo de Naïves Bayes, cuya precisión ha sido bastante insignificante comparada con otros modelos y que ha tomado bastante tiempo de procesamiento.

Los modelos que destacaríamos de forma individual y que, además de obtener una precisión optima, no han consumido demasiado tiempo serian: adaboost, gbm y xgboost. 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.

# CONCLUSIONES

Sección pendiente de desarrollo

# LÍNEAS FUTURAS

Sección pendiente de desarrollo

# 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/