Una herramienta clínica basada en el aprendizaje automático para diagnosticar la esclerosis múltiple mediante perfiles de expresión de microarrays de múltiples cohortes.

Amelia Martínez Sequera
Máster Bioinformática y Bioestadística
Área 4. Estadística y bioinformática.

Consultor/a
Romina Rebrij
Profesor/a responsable de la asignatura
Antoni Pérez Navarro

Barcelona, 8 de junio de 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Una herramienta clínica basada en el aprendizaje automático para diagnosticar la esclerosis múltiple mediante perfiles de expresión de microarrays de múltiples cohortes.*
Nombre del autor:	*Amelia Martínez Sequera*
Nombre del consultor/a:	*Romina Rebrij*
Nombre del PRA:	*Antoni Pérez Navarro*
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	*Máster Bioinformática y Bioestadística*
Área del Trabajo Final:	*Area 4*
Idioma del trabajo:	Español
Número de créditos:	15
Palabras clave	*Esclerosis múltiple, Aprendizaje automático,*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

La ***esclerosis múltiple*** **(EM)** es una enfermedad inflamatoria idiopática del sistema nervioso central y la segunda causa más común de discapacidad en adultos jóvenes. El diagnóstico y la clasificación de sus diferentes fases es un tema de gran interés en la medicina, ya que permite un resultado exitoso en los tratamientos médicos. Los avances recientes en inteligencia artificial aplicada a datos de resonancia magnética muestran resultados prometedores en la discriminación de las condiciones de EM de los controles sanos, pero tienen un poder limitado en la clasificación de etapas individuales de EM.

Los transcriptomas de células mononucleares de sangre periférica (PBMC) contienen información útil para construir clasificadores específicos para las diferentes formas de EM. A lo largo de este documento, se analiza la capacidad de distintos algoritmos de ***aprendizaje automático*** (AA) para encontrar patrones en los niveles que expresión genética que permitan clasificar las distintas etapas de EM de forma correcta. Además, se comparan los diferentes filtrados de genes. Los modelos se entrenan, optimizan y comparan en el conjunto de datos de entrenamiento antes de la validación final en el conjunto de prueba independiente. Los resultados demuestran un método de clasificación

eficiente y preciso basado en la expresión génica (datos de ***microarrays***) y algoritmos AA.

Abstract (in English, 250 words or less):

Multiple sclerosis (MS) is an idiopathic inflammatory disease of the central nervous system and the second most common cause of disability in young adults. The diagnosis and classification of its different phases is a topic of great interest in medicine, since it allows a successful result in medical treatments. Recent advances in artificial intelligence applied to MRI data show promising results in discriminating MS conditions from healthy controls, but have limited power in classifying individual MS stages.

Peripheral blood mononuclear cell (PBMC) transcriptomes contain useful information to construct specific classifiers for different forms of MS. Throughout this document, we analyze the ability of different **machine learning (ML)** algorithms to find patterns in the levels of gene expression that allow the different stages of MS to be classified correctly. In addition, the different gene filters are compared. Models are trained, optimized, and compared on the training data set before final validation on the independent test set. The results demonstrate an efficient and accurate classification method based on gene expression (microarray data) and AA algorithms.

Índice

4.7.1. División
4.7.2. Preprocesado:
Genes con varianza proxima a 0
Estandarización: Centrado y normalización

4.8. Selección de genes y reducción de la dimensionalidad
4.8.1. Anova p value
4.8.2. Signal to noise (S2N)
4.8.3. Comparación de filtrados
4.8.4. Reducción de dimensionalidad

4.9. Modelos
4.9.1. SVM: Máquinas de Vector Soporte (Support Vector Machines, SVMs)
4.9.2. Random Forest
4.9.3. Neural Network
4.9.4. Comparacion de modelos
4.9.5. Stacking (Ensemble)

5. Resultados
5.1. Comparación de modelos en función de hiperparámetros y filtrado.
5.2. Error de validación
5.3. Error de test
5.4. Mejor modelo
5.5. Model ensemble

6. Discusión
7. Conclusiones
8. Glosario de acrónimos
9. Bibliografia
10. Anexos

Lista de figuras

Lista de tablas

1 Resumen

La ***esclerosis múltiple*** **(EM)** es una enfermedad inflamatoria idiopática del sistema nervioso central y la segunda causa más común de discapacidad en adultos jóvenes. El diagnóstico y la clasificación de sus diferentes fases es un tema de gran interés en la medicina, ya que permite un resultado exitoso en los tratamientos médicos. Los avances recientes en inteligencia artificial aplicada a datos de resonancia magnética muestran resultados prometedores en la discriminación de las condiciones de EM de los controles sanos, pero tienen un poder limitado en la clasificación de etapas individuales de EM.

Los transcriptomas de células mononucleares de sangre periférica (PBMC) contienen información útil para construir clasificadores específicos para las diferentes formas de EM. A lo largo de este documento, se analiza la capacidad de distintos algoritmos de ***aprendizaje automático*** (AA) para encontrar patrones en los niveles que expresión genética que permitan clasificar las distintas etapas de EM de forma correcta. Además, se comparan los diferentes filtrados de genes. Los modelos se entrenan, optimizan y comparan en el conjunto de datos de entrenamiento antes de la validación final en el conjunto de prueba independiente. Los resultados demuestran un método de clasificación eficiente y preciso basado en la expresión génica (datos de ***microarrays***) y algoritmos AA.

2 Introducción

2.1. Contexto y justificación del trabajo

La esclerosis múltiple (EM) es una enfermedad crónica inflamatoria y desmielinizante del SNC, caracterizada por heterogeneidad clínica y biológica. Generalmente, la enfermedad comienza con un primer episodio clínico sugestivo de EM, clasificado como síndrome clínicamente aislado (CIS), que evoluciona a EM definida en casos de mayor actividad clínica o neurorradiológica[1].

Hasta el 85% de los pacientes con EM desarrollan el curso de EM recurrente-remitente (RR), que se caracteriza por un deterioro neurológico periódico

seguido de remisión parcial o completa. Algunos sujetos con EM-RR evolucionan eventualmente a secundaria progresiva (SP). Aproximadamente el 15% de los pacientes con EM desarrollan el curso primario progresivo (PP) desde el inicio de la enfermedad.

A pesar del continuo refinamiento de los criterios de diagnóstico, el grado de diagnóstico erróneo de EM sigue siendo bastante alto, especialmente para EM-PP, porque esta forma manifiesta hallazgos de RM que a menudo se superponen con EM-RR u otras enfermedades neurodegenerativas y vasculares [2][3]. Además el pronóstico de la enfermedad es poco predecible, la transición de EM RR a EM SP se puede definir retrospectivamente cuando se ha observado un período sostenido de deterioro neurológico que a menudo resulta en un período de 2 a 3 años de incertidumbre diagnóstica.[4]

Los avances recientes en inteligencia artificial aplicada a datos de resonancia magnética muestran resultados prometedores en la discriminación de las condiciones de EM de los controles sanos [5][6], pero tienen un poder limitado en la clasificación de etapas individuales de EM [7][8].

Por otro lado, las células mononucleares de sangre periférica (PBMC) presentan desregulaciones específicas en genes y vías en distintas etapas de la esclerosis múltiple (EM) que pueden ayudar a clasificar sujetos con EM y sin EM, especificar la etapa temprana de la enfermedad o discriminar entre cursos de EM [9][10][11][12]. Debido a que se han descrito perfiles transcripcionales aberrantes en sangre periférica de pacientes en distintos estadios de EM, se plantea la hipótesis de que los transcriptomas de células mononucleares de sangre periférica (PBMC) contienen información útil para construir clasificadores específicos para las diferentes formas de EM. La aplicación adecuada del aprendizaje automático a los perfiles transcripcionales de las células sanguíneas circulantes puede permitir la identificación del estado y estadio de la enfermedad en la EM.[13]

2.1. Objetivos del Trabajo

Objetivo general

Construir una herramienta clínica basada en el aprendizaje automático para predecir subtipos de esclerosis múltiple (EM) utilizando datos de expresión de microarrays de cohortes múltiples.

Objetivos específicos

- Analizar la capacidad de distintos algoritmos de aprendizaje automático para encontrar patrones en los niveles que expresión genética que permitan clasificar distintos tipos de estadíos de esclerosis múltiple (EM) de forma correcta.

- Determinar si existe algún método de aprendizaje automático capaz de predecir, con un porcentaje de acierto alto, el tipo de esclerosis múltiple en función sus niveles de expresión.

- Determinar, entre los varios miles de genes disponibles, si son todos importantes en vista a la clasificación o son solo unos pocos los que aportan información relevante.

Se trata, por lo tanto, de un **problema de clasificación multiclase supervisado.**

2.2. Enfoque y método a seguir

En este documento se analiza la capacidad de distintos algoritmos de ***aprendizaje automático*** (AA) para encontrar patrones en los niveles que expresión genética que permitan clasificar las distintas etapas de EM de forma correcta.

Es necesario identificar el subconjunto de genes que están realmente relacionados con la variable respuesta, es decir, genes que se expresan de forma diferente en una clase respecto al resto de clases. Al existir varios miles de posibles predictores, se recurre a métodos de filtrado y de reducción de dimensionalidad.

Los modelos se entrenan, optimizan y comparan en el conjunto de datos de entrenamiento antes de la validación final en el conjunto de prueba independiente. Los resultados demuestran un método de clasificación eficiente y preciso basado en la expresión génica (datos de ***microarrays***) y algoritmos AA.

Para la definición del clasificador óptimo, se tienen en cuenta varias cuestiones, incluida la selección del algoritmo de aprendizaje automático adecuado y las

características del conjunto de datos, como los números de características y muestras.

Las cohortes de EM recopiladas sufren desequilibrios entre las etapas de la EM, siendo la EM PP y la EM RR las formas más raras y frecuentes, respectivamente. Para superar estos problemas, se desarrolla un flujo de trabajo de aprendizaje automático, basado en el bootstraping, capaz de realizar una optimización imparcial, y una comparación de diferentes algoritmos en el conjunto de datos de entrenamiento antes de la validación final en el conjunto de prueba independiente.

2.3. Planificación del Trabajo

2.3.1. Recursos

Los códigos que se adjunta en los anexos se ejecutan en el programa R:

R version 4.0.4 (2021-02-15)

Platform: x86_64-w64-mingw32/x64 (64-bit)

Running under: Windows 10 x64 (build 19042)

Los modelos se entrenan, optimizan y comparan empleando las funcionalidades que ofrece el paquete ***caret.***

Debido al largo tiempo de ejecución de los códigos, se utiliza el paquete ***Parallel*** para paralelizar los loops.

```
library(parallel)
numCores <- detectCores()
numCores
```

```
## [1] 8
```

Se dispone de 8 núcleos lógicos. Se ocuparán la mitad (4 núcleos).

La anotación de genes, aunque no está dentro de los objetivos específicos de este trabajo, se realiza con el paquete ***illuminaHumanv3.***

2.3.2. Tareas

Tarea	Inicio	Fin	Dificultad
PEC0. PROPUESTA TFM	2021-02-17	2021-03-01	7
PEC1. PLAN DE TRABAJO	2021-03-02	2021-03-16	7
Estudio y selección de datos que contengan muestras de diferentes fases de EM	2021-03-17	2021-04-10	6
Selección de algoritmos clasificadores y paquetes en R	2021-03-17	2021-04-10	5
Reestructuración de los datos	2021-04-10	2021-04-15	6
Entreno y evaluación clasificadores	2021-04-15	2021-04-19	8
ENTREGA PEC2	2021-04-19	2021-04-19	NA
Selección y reestructuración de nuevos datos	2021-04-20	2021-05-02	9
División y preprocesado	2021-05-03	2021-05-06	6
Filtrado de genes	2021-05-07	2021-05-09	8
Aplicación de los algoritmos y comparación de modelos	2021-05-09	2021-05-17	8
ENTREGA PEC3	2021-05-17	2021-05-17	NA
Cambio de clases	2021-05-18	2021-05-21	5
Filtrado de genes, entreno y evaluación	2021-05-22	2021-06-05	6
Conclusiones	2021-06-05	2021-06-08	8
Redacción dela Memoria	2021-06-01	2021-06-08	7
ENTREGA PEC4	2021-06-08	2021-06-08	NA
PEC5a. Elaboración de la presentación	2021-06-9	2021-06-13	NA
PEC5b. Defensa pública	2021-06-16	2021-06-23	NA

Tabla 1. Lista de tareas realizadas en el TFM.

2.3.3. Planificación temporal

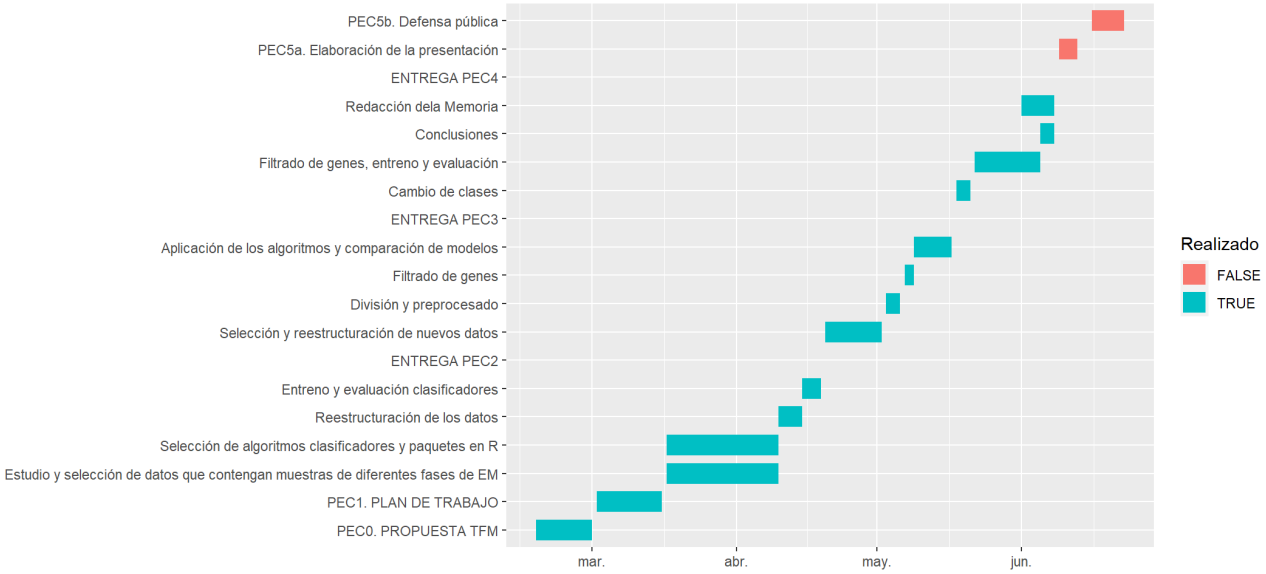

Fig 1. Diagrama de Gantt con la planificación temporal

2.4. Breve sumario de contribuciones y productos obtenidos

Se consigue construir un clasificador con una capacidad predictiva alta, tras el entrenamiento y validación de 3 algoritmos de aprendizaje supervisado: SVM, Random Forest y ANN. Inicialmente se emplean 6 clases, que incluyen controles sanos y muestras de individuos con otras enfermedades neurodegenerativas, además de las 4 clases de EM (síndrome aislado, PP, SP y RR). Se aplica la misma metodología utilizando sólo éstas últimas clases, prescindiendo de los EM negativos, y se obtiene una mejora en la predicción.

Los resultados también varían en función de la selección de genes diferencialmente expresados, siendo el filtrado por anova p value el que consigue mejores datos de *accuracy.* Se comparan resultados con un test de error y un test de validación y, finalmente, se realiza también un *Stacking* que no mejora los resultados obtenidos con los modelos.

2.5. Breve descripción de los otros capítulos de la memoria

Materiales y métodos empleados: Descripción del conjunto de datos utilizado, procedencia, procesado y reestructuración de los datos que se aplica para que sean útiles para generar el modelo de predicción.

Breve explicación de qué es un microarray, y de los algoritmos de aprendizaje supervisado más utilizados, así como los hiperparámetros que se modifican para conseguir el clasificador óptimo.

Descripción de los métodos de filtrado y reducción de dimensionalidad utilizados.

Conclusiones: análisis crítico del grado de consecución de los objetivos i del desarrollo del proyecto. Reflexión de las posibles línias de trabajo futuras.

Glosario de los acrónimos más relevantes utilizados en la memoria.

Bibliografia: lista numerada de las referencia bibliográficas.

Anexos: Scripts relevantes del proyecto, que incluyen el código en R utilizado y archivo HTML con los resultados. Carpeta con figuras obtenidas en R.

3. Estado del arte

Existen varios estudios recientes que aplican las técnicas de aprendizaje automático para el diagnóstico de la EM. Algunos han conseguido resultados prometedores utilizado imágenes de resonancia mágnética[5][6][8][22] solo o en combinación con datos clínicos[7], pero tienen un poder limitado en la clasificación de etapas individuales de la EM [7][8].

Otros estudios recientes han conseguido un buen poder predictivo a partir de biomarcadores en células mononucleares de sangre periférica (PBMC)[13][21], confirmando la solidez del procedimiento aplicado a los datos transcriptómicos, menos invasivos que las muestras de otros tejidos biológicos, como el líquido cefalorraquídeo[23]. Se han identificado genes con una desregulación significativa y constante en cada etapa de la EM a través del análisis de expresión diferencial, pero no hubo genes expresados diferencialmente entre

pacientes en los estadios PP-MS y SP-MS. Este resultado sugiere que a pesar de las claras diferencias en los síntomas clínicos de los pacientes con EM en las dos etapas, la mayoría de los genes tienen niveles de expresión relativamente similares[21].

En los últimos años, varios estudios de asociación de genoma completo (GWAS) han descubierto varias variantes de riesgo de EM y genes de susceptibilidad [24][25]. Aunque la EM afecta al SNC, existen evidencias de alteración de la inmunidad en la periferia en pacientes con EM [26]. Además, los fármacos terapéuticos más utilizados en la EM son agentes inmunosupresores o inmunomoduladores [26][27], lo que indica que dirigirse al sistema inmunitario periférico es beneficioso para los pacientes con esta enfermedad. Estas observaciones sustentan la justificación del empleo de células mononucleares de sangre periférica (PBMC) como una fuente informativa y de fácil acceso de material biológico en estudios de transcriptomas para la EM. Además, muchas investigaciones han demostrado recientemente la importancia de la transcriptómica sanguínea para descubrir cambios en la expresión génica y reguladores transcripcionales en la EM[27] [28] [29] [30] [31] [32] .

En el presente trabajo se pretende construir un clasificador de las diferentes formas de EM a partir de genes expresados diferencialmente de muestras de PBMC.

4. Metodología

4.1. Microarrays [14]

Desde mediados de los años noventa existe la técnica de los *microarrays* de ADN, que permite monitorizar simultáneamente el nivel de expresión de miles de genes en un conjunto de células, y se han consolidado como herramientas útiles en investigación genética con aplicaciones en medicina. Los datos que se generan con *microarrays*, aparte de tener un gran volumen, se caracterizan por ser altamente variables, por lo que serán básicos tanto el análisis estadístico como el diseño experimental.

El funcionamiento de los *microarrays* de expresión se basa en la capacidad de las moléculas complementarias de ADN de hibridar entre sí. Pequeñas cantidades de ADN, correspondientes a diversos genes cuya expresión se desea medir, son depositadas en una base de cristal. Para ello se utilizan robots de precisión que usan unas agujas especiales para obtener las moléculas de sus recipientes y depositarlas en las coordenadas adecuadas. A estas muestras de ADN depositadas en el *microarray* se las denomina dianas. De las células que queramos medir su expresión obtendremos una muestra de ARN que se convertirá en ADN complementario (ADNc) y se marcará con una molécula fluorescente. A esta muestra marcada la denominaremos sonda y se enfrentará a las dianas del *microarray*. Cada molécula de ADNc marcada de la sonda se moverá por difusión hacia la diana que contenga su molécula complementaria para hibridarse con ella y quedar fijada allí. Después de un tiempo para que la mayoría de las cadenas complementarias hibriden, el *microarray* se lava y se procede a hacer una medición relativa de la cantidad de ADN de la sonda que ha quedado fijada en cada diana.

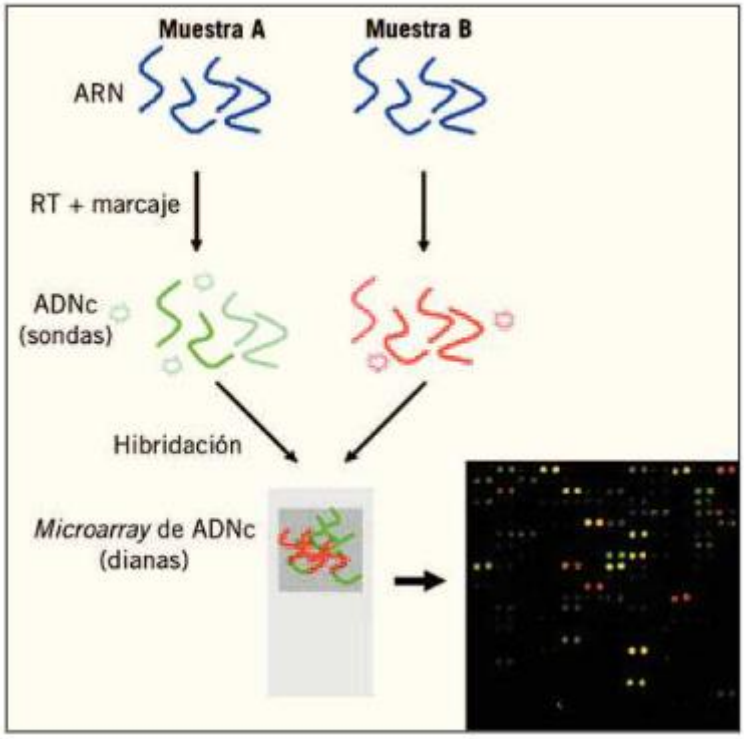

Fig 2. Esquema de un microarray.

Una vez se ha producido la hibridación, el *microarray* se lee con un escáner láser y se obtienen 2 imágenes, una para cada fluorocromo usado, con puntos de luz cuyas intensidades variarán según el nivel de hibridación que se haya producido en cada diana. Estas imágenes se procesan mediante un software que cuantifica la señal de cada punto (diana) para cada fluorocromo (muestra) y elabora una base de datos que será analizada con técnicas estadísticas. Con frecuencia se elabora una imagen superpuesta de las dos en forma de seudocolor. Una de las imágenes se colorea en rojo y la otra en verde. De esta manera, los puntos amarillos tienen señal alta en las 2 muestras en cantidad similar. Los puntos rojos o verdes indican que la expresión de ese gen predomina en una de las muestras.

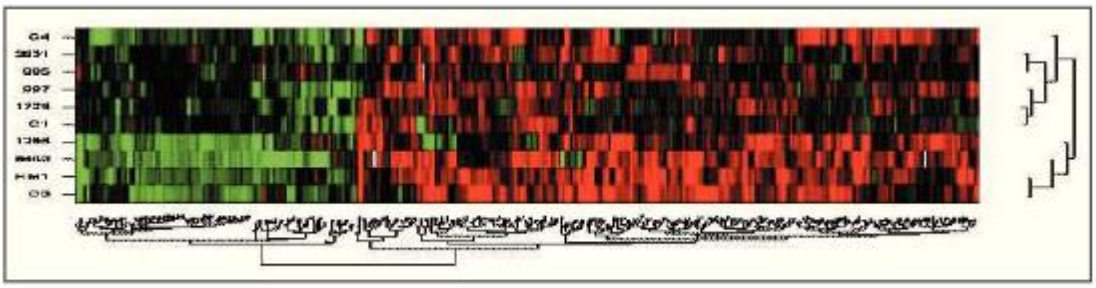

Fig 3. Imagen de un microarray.

4.1. Algoritmos de Aprendizaje Automático [15]

En este contexto, las técnicas de aprendizaje automático (AA) pueden aportar nuevas soluciones a este problema. El AA es una rama de la inteligencia artificial que crea sistemas con capacidad de identificar patrones complejos en grandes cantidades de datos de forma automática. A partir del análisis de datos, se generan modelos automáticamente, mediante algoritmos que aprenden de forma iterativa, identificando patrones i características dificilmente detectables mediante la observación y el análisis estadístico. La eleccción del algoritmo y los predictores que lo definen són claves para obtener predicciones fiables.

Los diferentes algoritmos de AA se pueden clasificar en aprendizaje supervisado y no supervisado. El aprendizaje supervisado consiste en modelar una serie de predictores (equivalentes a las variables independientes en regresión lineal) respecto a una variable de salida (equivalente a la variable dependiente). Durante el proceso de aprendizaje, se utilizan ejemplos con un valor de salida, numérico o categórico, conocido para generar un modelo que nos permita

predecir el valor de salida de nuevos ejemplos. Cuando finaliza la fase de aprendizaje, se evalúa el modelo comparando las predicciones efectuadas con los valores de salida conocidos. En el aprendizaje no supervisado no tenemos conocimiento a priori de la característica que se pretende analizar. Son algoritmos que permiten obtener descriptores de los datos, o detectar patrones, o relaciones entre los datos, o agruparlos según su grado de similitud, es decir, permiten generar clasificaciones (clustering).

Algunos de los algoritmos de AA supervisado más importantes se describen a continuación:

4.1.1. Artificial Neural Network

Una red neuronal artificial (ANN) modela la relación entre un conjunto de entradas eñales y una señal de salida utilizando un modelo derivado de nuestra comprensión de cómo un cerebro biológico responde a los estímulos de las entradas sensoriales. Así como un cerebro usa una red de células interconectadas, llamadas neuronas, para crear un procesador paralelo masivo, ANN utiliza una red de neuronas o nodos artificiales para resolver problemas de aprendizaje.

A pesar de su complejidad, se pueden aplicar fácilmente a problemas del mundo real. Son aprendices versátiles que se pueden aplicar a casi cualquier tarea de aprendizaje: clasificación, predicción numérica e incluso sin supervisión reconocimiento de patrones.

Características:

Las redes neuronales utilizan neuronas como bloques de construcción para construir modelos complejos de datos. Aunque existen numerosas variantes de redes neuronales, cada una puede definirse en términos de las siguientes características:

• Una **función de activación,** que transforma las señales de entrada combinadas de una neurona

en una sola señal de salida para ser transmitida más en la red

• Una **topología** (o arquitectura) de red, que describe el número de neuronas en el modelo, así como el número de capas y la forma en que están conectados (dirección en la que viaja la información)

• El **algoritmo de entrenamiento** que especifica cómo se establecen en orden los pesos de conexión para inhibir o excitar neuronas en proporción a la señal de entrada

Fortalezas	Debilidades
Puede adaptarse a problemas de predicción de clase o numérica	Computacionalmente, extremadamente intensivo y lento de entrenar, particularmente si la topología de la red es compleja
Capaz de modelar patrones más complejos que cualquier otro algoritmo	Propenso a sobreajustar el entrenamiento
Hace pocas suposiciones sobre las relaciones subyacentes de los datos	Da como resultado un modelo de caja negra compleja que es difícil, si no imposible, interpretar.

Tabla 2. Tabla de fortalezas y debilidades de ANN

4.1.2. Support Vector Machine.

El algoritmo SVM originariamente se desarrolló como un método de clasificación binaria, su aplicación se ha extendido a problemas de clasificación múltiple y regresión. Cuando se dispone de n observaciones, cada una con p predictores y cuya variable respuesta tiene dos niveles, se pueden emplear hiperplanos para construir un clasificador que permita predecir a que grupo pertenece una observación en función de sus predictores. Si la distribución de las observaciones es tal que se pueden separar linealmente de forma perfecta en las dos clases (+1 y −1), el clasificador más sencillo consiste en asignar cada observación a una clase dependiendo del lado del hiperplano en el que se encuentre.

La definición de hiperplano para casos perfectamente separables linealmente resulta en un número infinito de posibles hiperplanos, lo que hace necesario un método que permita seleccionar uno de ellos como clasificador óptimo.

La solución a este problema consiste en seleccionar como clasificador óptimo al que se conoce como maximal margin hyperplane o hiperplano óptimo de

separación, que se corresponde con el hiperplano que se encuentra más alejado de todas las observaciones de entrenamiento. Para obtenerlo, se tiene que calcular la distancia perpendicular de cada observación a un determinado hiperplano. La menor de estas distancias (conocida como margen) determina como de alejado está el hiperplano de las observaciones de entrenamiento. El maximal margin hyperplane se define como el hiperplano que consigue un mayor margen, es decir, que la distancia mínima entre el hiperplano y las observaciones es lo más grande posible. Aunque esta idea suena razonable, no es posible aplicarla, ya que habría infinitos hiperplanos contra los que medir las distancias. En su lugar, se recurre a métodos de optimización.

A las observaciones equidistantes respecto al maximal margin hyperplane se les conoce como vectores soporte, ya que son vectores en un espacio p-dimensional y soportan (definen) el maximal margin hyperplane. Cualquier modificación en estas observaciones (vectores soporte) conlleva cambios en el maximal margin hyperplane. Sin embargo, modificaciones en observaciones que no son vector soporte no tienen impacto alguno en el hiperplano.

CASOS CUASI-SEPARABLES LINEALMENTE

El maximal margin hyperplane descrito en el apartado anterior es una forma muy simple y natural de clasificación siempre y cuando exista un hiperplano de separación. En la gran mayoría de casos reales, los datos no se pueden separar linealmente de forma perfecta, por lo que no existe un hiperplano de separación y no puede obtenerse un maximal margin hyperplane.

Para solucionar estas situaciones, se puede extender el concepto de maximal margin hyperplane para obtener un hiperplano que casi separe las clases, pero permitiendo que cometa unos pocos errores. A este tipo de hiperplano se le conoce como Support Vector Classifier o Soft Margin.

El proceso incluye un hiperparámetro de tuning C. C controla el número y severidad de las violaciones del margen (y del hiperplano) que se toleran en el proceso de ajuste. Si $C=\infty$, no se permite ninguna violación del margen y por lo tanto, el resultado es equivalente al Maximal Margin Classifier (teniendo en cuenta que esta solución solo es posible si las clases son perfectamente separables). Cuando más se aproxima C a cero, menos se penalizan los errores y más observaciones pueden estar en el lado incorrecto del margen o incluso del hiperplano. C es a fin de cuentas el hiperparámetro encargado de controlar el balance entre bias y varianza del modelo. En la práctica, su valor óptimo se identifica mediante cross-validation.

Fortalezas	Debilidades
Puede adaptarse a problemas de predicción de clase o numérica	Encontrar el mejor modelo requiere probar varias combinaciones de kernels y parámetros
No se deja influenciar mucho por el ´ruido´ en los datos y no es muy propenso al sobreajuste	Puede ser lento en el entrenamiento, especialmente si los datos tienen un gran número de características o ejemplos
Puede ser más fácil de usar que una neural network	Da como resultado un modelo complejo de caja negra compleja que es difícil, si no imposible, interpretar.
Alta precisión	

Tabla 3. Tabla de fortalezas y debilidades de SVM

4.1.3. Arboles de Decisión.

Los árboles de decisión se construyen usando una heurística llamada partición recursiva. Este enfoque también se conoce comunmente como divide y vencerás porque divide los datos en subconjuntos, que luego se dividen repetidamente en subconjuntos aún más pequeños, y así sucesivamente.

El proceso se detiene cuando el algoritmo determina que los datos dentro de los subconjuntos son suficientemente homogéneos, o se ha cumplido otro criterio para finalizar la división.

Un árbol de decisión tiene un nodo raíz que crecer hasta convertirse en un árbol maduro. Al principio, el nodo raiz representa todo el conjunto de datos, ya que no se ha producido ninguna división. Luego, el algoritmo debe elegir una característica para dividir; idealmente, elige la característica más predictiva de la clase objetivo. Los ejemplos se dividen en grupos segun los distintos valores de esta característica y se forma el primer conjunto de ramas de árbol.

Sobre cada rama, el algoritmo continua dividiendo los datos y eligiendo la mejor característica candidata cada vez para crear otro nodo de decisión, hasta que se alcanza un criterio de detención.

Divide y vencerás podría detenerse en un nodo en un caso que:

. Todos (o casi todos) los ejemplos en el nodo tienen la misma clase.

. No hay más características.

. El árbol ha crecido a un límite de tamaño predefinido.

Usaremos el término de **Ganancia de Información** para seleccionar el atributo más útil, es decir, aquel que me permite separar mejor los datos respecto a la clasificación final.

Para poder aplicar el algoritmo hemos de comenzar sabiendo cómo se mide la ganancia de información, y para ello hay que introducir el concepto de Entropía de Shannon, que mide el grado de incertidumbre de una muestra:

* Muestra completamente homogénea, será aquella en la que todos sus elementos se clasifican igual, y por tanto tiene incertidumbre mínima, ya que no tenemos dudas de cuál es la clasificación de cualquiera de sus elementos. En este caso, fijaremos la incertidumbre (entropía) a 0.

* Muestra igualmente distribuida, será aquella que tiene el mismo número de ejemplos de cada posible clasificación, muestra por tanto una incertidumbre máxima, ya que es la peor situación para poder saber a priori cual sería la clasificación de unos de sus elementos si lo eligiésemos al azar. En este caso fijaremos la incertidumbre (entropía) a 1

La función matemática que mide el grado de incertidumbre es:

$$Entropy(S) = \sum_{i=1}^{C} -p_i log_2(p_i)$$

siendo S el conjunto de datos, C el número de clasificaciones que usaremos y p_i la proporción de datos que hay en la clasifición i en la muestra.

Asi lo que buscaremos será aquella característica que cree más ramas homogéneas y por tanto proporcione una ganancia de información mayor.

Proceso (obtenida en [@arbproc];)

1- Se calcula la entropía del total.

2- Se divide el conjunto de datos en función de los diferentes atributos.

3- Se calcula la entropía de cada rama y se suman proporcionalmente las ramas para calcular la entropía del total:

$$E(T, X) = \sum_{c \in X} p(c) E(S_c)$$

4- Se resta este resultado de la entropía original, obteniendo como resultado la Ganancia de Información (descenso de entropía) usando este atributo.

$$Gain(T, X) = E(T) - E(T, X)$$

5- El atributo con mayor Ganancia es selecciona como nodo de decisión.

Una rama con entropía 0 se convierte en una hoja (nodo-respuesta), ya que representa una muestra completamente homogénea, en la que todos los ejemplos tienen la misma clasificación. Si no es así, la rama debe seguir subdividiéndose con el fin de clasificar mejor sus nodos.

Tabla fortalezas y debilidades

Fortalezas	Debilidades
* Clasificador multiuso que funciona bien en la mayoría de los problemas	* Los modelos de árbol de decisión a menudo estan sesgados hacia divisiones en características que tienen una gran cantidad de niveles
* Proceso de aprendizaje altamente automático, pudiendo manejar variables numéricas o nominales, así como datos faltantes	* Es fácil sobreajustar o ajustar el modelo
* Excluye características sin importancia	* Puede tener problemas para modelar algunas relaciones debido a la dependencia de las divisiones paralelas al eje
* Se puede usar en conjuntos de datos pequeños y grandes	* Pequeños cambios en los datos de entrenamiento pueden generar grandes cambios en la lógica de decisión
* Resultados que pueden interpretarse sin una base matemática	* Los árboles grandes pueden ser difíciles de interpretar y las decisiones que toman pueden parecer contradictorias
* Más eficiente que otros modelos complejos	

Tabla 4. Tabla de fortalezas y debilidades de Árboles de decisión.

4.1.4. Random Forest

Un modelo Random Forest está formado por un conjunto (ensemble) de árboles de decisión individuales, cada uno entrenado con una muestra aleatoria extraída de los datos de entrenamiento originales mediante bootstrapping. Esto implica que cada árbol se entrena con unos datos ligeramente distintos. En cada árbol individual, las observaciones se van distribuyendo por bifurcaciones (nodos) generando la estructura del árbol hasta alcanzar un nodo terminal. La predicción de una nueva observación se obtiene agregando las predicciones de todos los árboles individuales que forman el modelo.

4.1.5. k-Nearest Neighbour.

k-NN no hace ningún aprendizaje, simplemente almacena textualmente los datos del trainin. Los ejemplos de prueba sin etiquetar se comparan con los registros más similares en el conjunto de entrenamiento usando una función de distancia, y el ejemplo sin etiqueta se asigna a la etiqueta de sus vecinos. A pesar de que k-NN es un algoritmo muy simple, es capaz de abordar tareas extremadamente complejas.

Fortalezas	Debilidades
Simple y efectivo	No produce un modelo, limitando la capacidad para comprender cómo las características están relacionados con la clase
No hace suposiciones sobre la distribución de datos subyacente	Requiere la selección de un k apropiado
Fase de entrenamiento rápida	Características nominales y datos faltantes requieren procesamiento adicional.
	Fase de clasificación lenta.

Tabla 5. Tabla de fortalezas y debilidades de k-NN

4.1.6. Naive Bayes.

El algoritmo de Naive Bayes se fundamenta en los conocidos como métodos bayesianos, un conjunto de principios matemáticos fundamentales desarrollados por Thomas Bayes cuyo fin es la descripción de eventos probabilísticos. Sobre esta base los clasificadores basados en los métodos bayesianos utilizan datos de entrenamiento para calcular una probabilidad observada de cada clase basada en los valores de las variables. Cuando el clasificador se utiliza posteriormente en datos no etiquetados, utiliza las probabilidades observadas para predecir la clase más probable de los nuevos datos dados para estas variables.

Típicamente, los clasificadores bayesianos se aplican mejor a los problemas en los que la información de numerosos atributos debe considerarse simultáneamente para estimar la probabilidad de un resultado. Mientras que muchos algoritmos ignoran aquellas características que tienen efectos débiles, los métodos bayesianos utilizan todas las pruebas disponibles para cambiar sutilmente las predicciones. Si un gran número de variables tienen efectos relativamente menores, en conjunto, su impacto combinado podría ser considerable.

Naive Bayes asume que todas las características del conjunto de datos son igualmente importantes e independientes. Estos supuestos rara vez son verdaderos en el mundo real.

El algoritmo de Naive Bayes presenta un problema importante que surge si un evento nunca ocurre para uno o más niveles de la clase, y es que, debido a que las probabilidades en los algoritmos de Naive Bayes se multiplican, un valor del cero por ciento causa que la probabilidad posterior de que se de x suceso sea cero, lo que da a este evento nulo la capacidad para anular y dominar efectivamente sobre todas las demás evidencias.

Una solución a este problema consiste en el uso de un elemento conocido como el Estimador de Laplace, que lleva el nombre del matemático francés Pierre-Simon Laplace. El estimador de Laplace lo que hace es añadir un pequeño número, una cifra residual, a cada uno de los recuentos realizados en la frecuencia, lo que asegura que cada característica tiene una probabilidad no nula de ocurrir con cada clase. Típicamente, el estimador de Laplace se fija en 1, lo que asegura que cada combinación de clases y características se encuentra en los datos al menos una vez.

El estimador de Laplace se puede ajustar a cualquier valor, y no necesariamente tiene que ser el mismo para cada una de las características. Se podría usar el estimador de Laplace para reflejar una presunta probabilidad a priori de cómo la característica se relaciona con la clase. En la práctica, dado un conjunto de datos de entrenamiento lo suficientemente grande, este paso es innecesario, y casi siempre se utiliza el valor de uno.

Puesto que el Naive Bayes utiliza tablas de frecuencia para el aprendizaje, cada característica debe ser categórica a fin de crear las combinaciones de valores de clase y característica que componen la matriz. Como los rasgos numéricos no tienen categorías de valores, este algoritmo no funciona directamente con los datos numéricos.

Una solución fácil y eficaz es la de discretizar las variables numéricas, lo que significa simplemente que los números se colocan en categorías conocidas como contenedores (bins). Hay que tener en consideración es que la discretización de una variable numérica siempre da lugar a una reducción de la información, ya que la granularidad original de la variable se reduce a un conjunto de categorías. Es importante lograr un equilibrio, un número demasiado reducido de bins puede dar lugar a que se oculten tendencias importantes, mientras que un número demasiado elevado de bins puede dar lugar a recuentos pequeños en la tabla de frecuencias de Naive Bayes.

Fortalezas	Debilidades
Simple, rápido y muy efectivo	Se basa en una suposición a menudo errónea de igualdad de importancia y variables independientes
Funciona bien con ruido y datos faltantes	No es ideal para conjuntos de datos con muchas variables numéricas
Requiere relativamente pocos ejemplos para entrenamiento, pero también funciona bien con una gran cantidad de ejemplos	Las probabilidades estimadas son menos fiables que las clases predichas
Fácil de obtener la probabilidad estimada de una predicción	

Tabla 6. Tabla de fortalezas y debilidades de Naive Bayes.

4.2. Estudio de la bibliografía relacionada: Selección de datos y algoritmos de clasificación.

Se lleva a cabo una revisión del estado actual de las investigaciones en el diagnóstico de esclerosis múltiple a partir de biomarcadores. Se descartan los estudios basados en imágenes y de muestras que no procedan de PBMC (microarrays).

De la bibliografia revisada, se determina:

- qué conjunto de datos podemos utilizar. Se opta por emplear muestras de otros estudios que ya están validadas y disponibles públicamente.

- qué clasificadores son los más empleados. Esta información también nos permite innovar en el análisis, utilizando alguna técnica nueva que no se haya utilizado.

Los datos de expresión empleados en este documento se han obtenido de la base de datos **Gene Expression Omnibus** (GEO), un repositorio público internacional que archiva y distribuye libremente la expresión génica de alto rendimiento y otros conjuntos de datos genómicos funcionales. Se trata de un estudio de Acquaviva et al. (2020) que pretende construir clasificadores óptimos basados en la expresión génica para la predicción del estado y estadio de la EM. Para ello, recopilaron perfiles transcriptómicos de todo el genoma de células mononucleares de sangre periférica (de ahora en adelante, PBMC) de 313 individuos (60 HC, 57 sujetos CIS, 108 sujetos RR MS, 26 sujetos SP MS, 35 PP MS sujetos y 27 sujetos OND) y generados con dos microarrays distintos (HumanRef-8 v.2 y HumanHT-12 v.4).

Para una comparación imparcial de distintos algoritmos de aprendizaje, desarrollaron un flujo de trabajo de validación cruzada anidado (NCV) seguido de la validación final en el conjunto de prueba independiente. Utilizando NCV comparan tres algoritmos basados en árboles de decisión: árboles funcionales (FT), Adaptive Boosting aplicado a FT (ADAboost-FT) y Random Forests (RF). El ajuste fino de los hiperparámetros específicos de cada algoritmo se realizó automáticamente en un ciclo interno de validación cruzada (innerCV) anidado dentro de un ciclo externo de validación cruzada (externalCV), que se utilizó para la estimación adecuada del modelo predictivo. [13]

Otros estudios relacionados:

Pinto et al. (2020) utilizan datos de la historia clínica de pacientes con EM que padecen cursos de RR y SP, donde los pacientes con PP fueron excluidos, ya que las manifestaciones clínicas de este curso son relativamente diferentes de las restantes en una fase inicial. Eligen tres conjuntos diferentes de pacientes, uno para la predicción del desarrollo de SP y dos para la gravedad de la enfermedad. Se entrenaron cuatro clasificadores: Máquina de vectores de soporte (SVM) con un kernel lineal, un k -NN, un Árbol de decisión y una Regresión lineal.[16]

Franks et al. (2019) utilizan clasificadores supervisados que incluyen red elástica multinomial (GLMnet), máquina de vectores de soporte (SVM) y bosque aleatorio (RF). La evaluación inicial de los clasificadores se realizó utilizando el rendimiento sobre la validación cruzada repetida.[17]

Shang et al.(2020) utilizan datos de microarrays procedentes de PBMC. Seleccionan los genes expresados diferencialmente y realizan un estudio de enriquecimiento. Este conjunto de datos sería una alternativa también válida para el presente trabajo. [21]

4.3. Instalación de librerías en R.

R es uno de los lenguajes de programación que domina dentro del ámbito de la estadística, data mining y machine learning. Al tratarse de un software libre, innumerables usuarios han podido implementar sus algoritmos, dando lugar a un número muy elevado de paquetes/librerías donde encontrar prácticamente todas las técnicas de machine learning existentes. Sin embargo, esto tiene un lado negativo, cada paquete tiene una sintaxis, estructura e implementación propia, lo que dificulta su aprendizaje. El paquete caret, desarrollado por Max Kuhn, es una interfaz que unifica bajo un único marco cientos de funciones de distintos paquetes, facilitando en gran medida todas las etapas de de preprocesado, entrenamiento, optimización y validación de modelos predictivos.[18]

4.4. Datos

En el campo del AA, se llama materiales al conjunto de datos que nos permitirá generar un modelo. [18] En esta sección se decribe el origen de los datos

utilizados en este trabajo, así como las modificaciones realizadas para obtener un conjunto de descriptores útiles para conseguir los objetivos marcados.

Los conjuntos de datos de microarrays sin procesar y procesados se obtuvieron de la base de datos GEO (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE136411).El conjunto de datos seleccionado se identifica con el número de acceso: GSE136411

4.4.1. Reestructuración de los datos.

Se realizan una serie de modificaciones para almacenar la información en un dataframe en el que cada fila representa una muestra y las columnas contienen la información relativa a ellas (información descriptiva sobre el tipo de EM y la expresión de los genes).

Se dispone de un total de 336 muestras (replicas técnicas y biológicas), y se parte de un total de 10160 genes.

Inicialmente se emplean 6 clases, que incluyen controles sanos y muestras de individuos con otras enfermedades neurodegenerativas, además de las 4 clases de EM (síndrome aislado, PP, SP y RR). Se aplica la misma metodología utilizando sólo éstas últimas clases, prescindiendo de los EM negativos, y se obtiene una mejora en la predicción.

4.4.2. Filtrado de calidad de datos.

La tecnología microarray empleada para cuantificar los niveles de expresión tiene unos límites de detección, fuera de los cuales, la lectura no es fiable. En este caso, si la señal es inferior a 10 unidades, se considera ruido y debe interpretarse como que no hay expresión de ese gen. En el extremo opuesto, 34 unidades es el valor máximo.

Prácticamente ninguno de los valores de este dataset se corresponden a lecturas por debajo del límite de detección (ruido), porque ya viene corregido.

Esta trasformación es un paso de limpieza, no un preprocesado de datos para mejorar el modelo, por lo tanto, sí puede hacerse sobre todas las observaciones antes de separarlas en conjunto de entrenamiento y test.

4.5. Exploración de los datos.

Cantidad y tipo de muestras.

El set de datos se dispone de 336 muestras, agrupadas en 3 grupos, que a su vez se subdividen en 6 tipos: controles sanos (HC), con EM (CIS,RR, PP, SP) y con otras enfermedades neurológicas (OND).

type <chr>	n <int>
CIS	60
HC	67
OND	27
PP	35
RR	121
SP	26

Tabla 7. Número de muestras de las diferentes clases

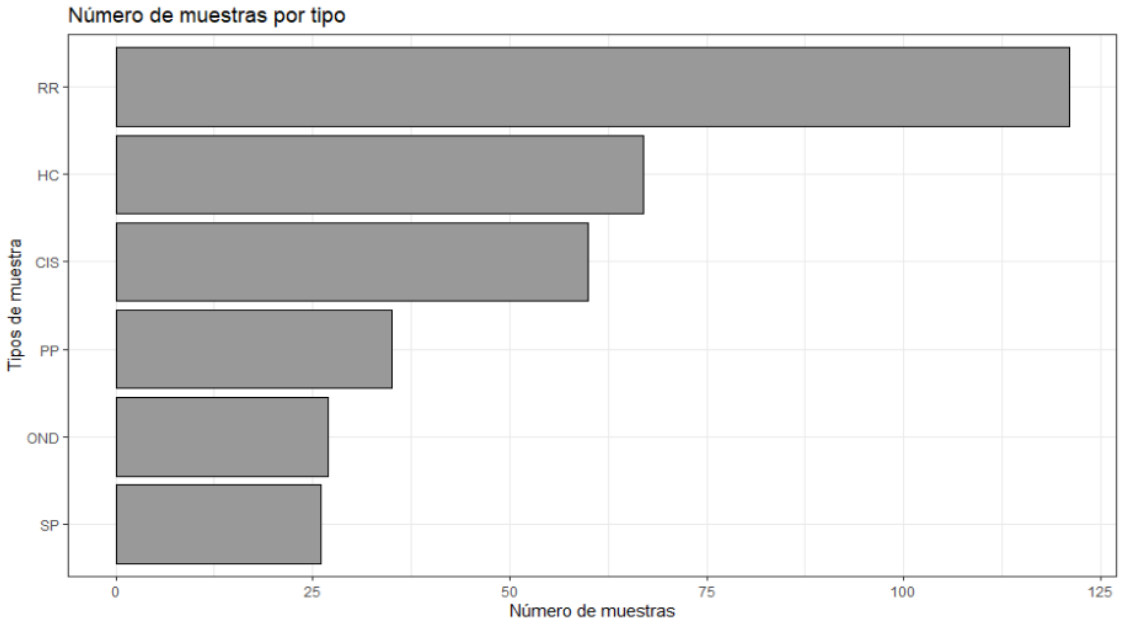

Figura 4. Número de muestras por tipo.

Cuando el número de observaciones por clase no es el mismo (clases desbalanceadas), los métodos estadísticos y de machine learning pueden verse afectados, por lo que es un aspecto a tener en cuenta en el análisis.

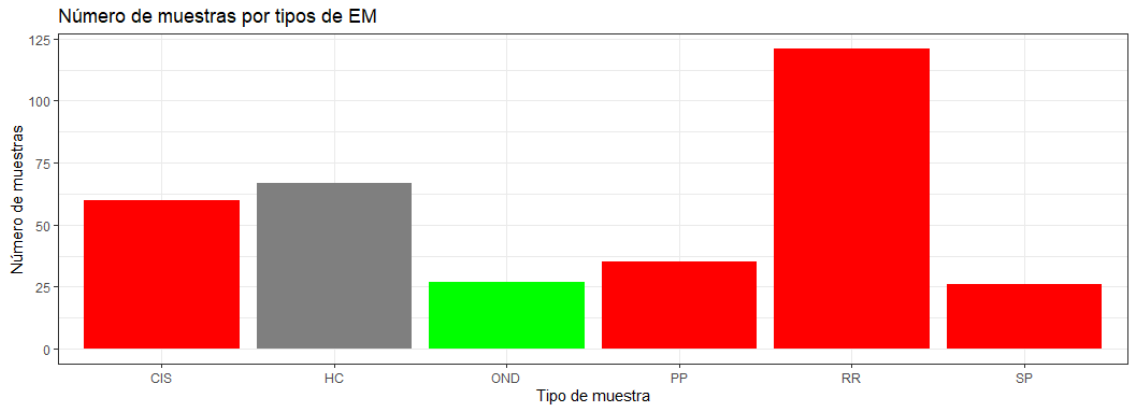

Figura 5. Histograma de número de muestras por tipo.

Valores ausentes.

Se comprueba que para todas las muestras se dispone del valor de expresión. El set de datos está completo, no hay valores ausentes.

4.6. División y preprocesado de los datos [18]

4.6.1. División de los datos.

Evaluar la capacidad predictiva de un modelo consiste en comprobar cómo de próximas son sus predicciones a los verdaderos valores de la variable respuesta. Para poder cuantificar de forma correcta este error, se necesita disponer de un conjunto de observaciones, de las que se conozca la variable respuesta, pero que el modelo no haya "visto", es decir, que no hayan participado en su ajuste. Con esta finalidad, se separan los datos disponibles en un conjunto de entrenamiento y un conjunto de test. El tamaño adecuado de las particiones depende en gran medida de la cantidad de datos disponibles y la seguridad que se necesite en la estimación del error, 80%-20% suele dar buenos resultados. El reparto debe hacerse de forma aleatoria o aleatoria-estratificada.

El objetivo de este estudio es poder predecir la fase de EM, por lo que se seleccionan los datos correspondientes a las 4 fases y se descartan las columnas que no son necesarias (controles sanos y otras enfermedades neurodegenerativas).

Es importante verificar que la distribución de la variable respuesta es similar en el conjunto de entrenamiento y en el de test. Por defecto, la función createDataPartition() garantiza una distribución aproximada (reparto estratificado).

Description: df[,4] [4 x 4]

train.Var1 <fctr>	train.Freq <dbl>	test.Var1 <fctr>	test.Freq <dbl>
CIS	0.265	CIS	0.273
PP	0.143	PP	0.091
RR	0.490	RR	0.545
SP	0.102	SP	0.091

Tabla 8. Tabla de frecuencias de test y training.

Este tipo de reparto estratificado asegura que el conjunto de entrenamiento y el de test sean similares en cuanto a la variable respuesta, sin embargo, no garantiza que ocurra lo mismo con los predictores. Es importante tenerlo en cuenta sobre todo cuando los predictores son cualitativos.

El tipo más frecuente es RR (49%), este es el porcentaje aproximado de aciertos que se espera si siempre se predice "RR". Por lo tanto, este es el porcentaje de aciertos (*accuracy*) que deben ser capaces de superar los modelos predictivos para considerarse mínimamente útiles.

4.6.2. Preprocesado.

El preprocesado de datos engloba aquellas transformaciones hechas sobre los datos con la finalidad de que puedan ser aceptados por el algoritmo de AA o que mejoren sus resultados. Todo preprocesado de datos debe aprenderse de las observaciones de entrenamiento y luego aplicarse al conjunto de entrenamiento y al de test. Esto es muy importante para no violar la condición de que ninguna información procedente de las observaciones de test puede participar o influir en el ajuste del modelo.

Genes con varianza próxima a cero.

En la mayoría de análisis discriminantes (diferenciación de grupos), el número de observaciones disponibles es mucho mayor que el número de variables, sin embargo, los estudios de expresión genética suelen caracterizarse por justo lo contrario. Por lo general, se dispone de un número bajo de muestras en comparación a los varios miles de genes disponibles como predictores. Esto dificulta en gran medida la creación de modelos predictivos por dos razones. En primer lugar, algunos algoritmos de machine learning, por ejemplo el análisis discriminante lineal (LDA), no puede aplicarse si el número de observaciones es inferior al número de predictores. En segundo lugar, aun cuando todos los genes pueden incorporarse en el modelo (SVM), muchos de ellos no aportan más que ruido al modelo, lo que disminuye su capacidad predictiva cuando se aplica a nuevas observaciones (overfitting). A este problema se le conoce como “alta dimensionalidad”.

Una forma de reducir este problema consiste en eliminar aquellos genes cuya expresión apenas varía en el conjunto de observaciones, y que, por lo tanto, no aportan información.

La función *nearZeroVar()* del paquete caret identifica como predictores potencialmente problemáticos aquellos que tienen un único valor (cero varianza) o que cumplen las dos siguientes condiciones:

- Ratio de frecuencias: ratio entre la frecuencia del valor más común y la frecuencia del segundo valor más común. Este ratio tiende a 1 si las frecuencias están equidistribuidas y a valores grandes cuando la frecuencia del valor mayoritario supera por mucho al resto (el denominador es un número decimal pequeño). Valor por defecto freqCut = 95/5.

- Porcentaje de valores únicos: número de valores únicos dividido entre el total de muestras (multiplicado por 100). Este porcentaje se aproxima a cero cuanto mayor es la variedad de valores. Valor por defecto uniqueCut = 10.

Entre los predictores incluidos en el modelo, no se detecta ninguno con varianza cero o próxima a cero.

Estos mismos genes identificados en el conjunto de entrenamiento, tendrían que ser excluidos también del conjunto de test.

Si bien la eliminación de predictores no informativos podría considerarse un paso propio del proceso de selección de predictores, dado que consiste en un filtrado por varianza, tiene que realizarse antes de estandarizar los datos, ya que después, todos los predictores tienen varianza 1.

Estandarización

Cuando los predictores son numéricos, la escala en la que se miden, así como la magnitud de su varianza, pueden influir en gran medida en el modelo. Muchos algoritmos de AA (SVM, redes neuronales, lasso…) son sensibles a esto, de forma que, si no se igualan de alguna forma los predictores, aquellos que se midan en una escala mayor o que tengan más varianza, dominarán el modelo aunque no sean los que más relación tienen con la variable respuesta.

Existen principalmente 2 estrategias para evitarlo:

Centrado: consiste en restarle a cada valor la media del predictor al que pertenece. Si los datos están almacenados en un dataframe, el centrado se consigue restándole a cada valor la media de la columna en la que se encuentra. Como resultado de esta transformación, todos los predictores pasan a tener una media de cero, es decir, los valores se centran en torno al origen.

Normalización: consiste en transformar los datos de forma que todos los predictores estén aproximadamente en la misma escala. Se procede a normalizar la expresión de cada gen (columna) para que tengan media 0 y varianza 1.

4.7. Selección de genes y reducción de dimensionalidad[18]

Es necesario aplicar una estrategia que permita identificar el subconjunto de genes que están realmente relacionados con la variable respuesta, es decir, genes que se expresan de forma diferente en una clase respecto al resto de clases. Incluir un exceso de variables suele conllevar una reducción de la capacidad predictiva del modelo cuando se expone a nuevos datos (overfitting).

Este problema a sido foco de investigación en el ámbito de la bioinformática durante años, lo que ha dado lugar a una amplia variedad de métodos conocidos como feature selection, cuyo objetivo es reducir el número de predictores para mejorar los modelos.

Métodos wrapper

Los métodos wrapper evalúan múltiples modelos, generados mediante la incorporación o eliminación de predictores, con la finalidad de identificar la combinación óptima que consigue maximizar la capacidad del modelo. Pueden entenderse como algoritmos de búsqueda que tratan a los predictores disponibles como valores de entrada y utilizan una métrica del modelo, por ejemplo, su error de predicción, como objetivo de la optimización.

Métodos de filtrado

Los métodos basados en filtrado evalúan la relevancia de los predictores fuera del modelo para, posteriormente, incluir únicamente aquellos que pasan un determinado criterio. Se trata por lo tanto de analizar la relación que tiene cada predictor con la variable respuesta. Por ejemplo, en problemas de clasificación con predictores continuos, se puede aplicar un ANOVA a cada predictor para identificar aquellos que varían dependiendo de la variable respuesta. Finalmente, se incorporan al modelo aquellos predictores con un p-value inferior a un determinado límite o los n mejores.

Ambas estrategias, wrapper y filtrado, tienen ventajas y desventajas. Los métodos de filtrado son computacionalmente más rápidos, por lo que suelen ser la opción factible cuando hay cientos o miles de predictores, sin embargo, el criterio de selección no está directamente relacionada con la efectividad del modelo. Además, en la mayoría de casos, los métodos de filtrado evalúan cada predictor de forma individual, por lo que no contemplan interacciones y pueden incorporar predictores redundantes (correlacionados). Los métodos wrapper, además de ser computacionalmente más costosos, para evitar overfitting, necesitan recurrir a validación cruzada o bootstrapping, por lo que requieren un número alto de observaciones. A pesar de ello, si se cumplen las condiciones, suelen conseguir una mejor selección.

En este caso, al existir varios miles de posibles predictores, se recurre a métodos de filtrado.

4.7.1. Anova p-value

El contraste de hipótesis ANOVA compara la media de una variable continua entre dos o más grupos. Para este estudio, la idea es que el ANOVA permita

identificar aquellos genes cuya expresión varía significativamente entre los distintos tipos de EM. Dos de las condiciones para que este test de hipótesis sea válido son: que la variable respuesta (nivel de expresión génica) se distribuya de forma normal y que tenga varianza constante en todos los grupos.

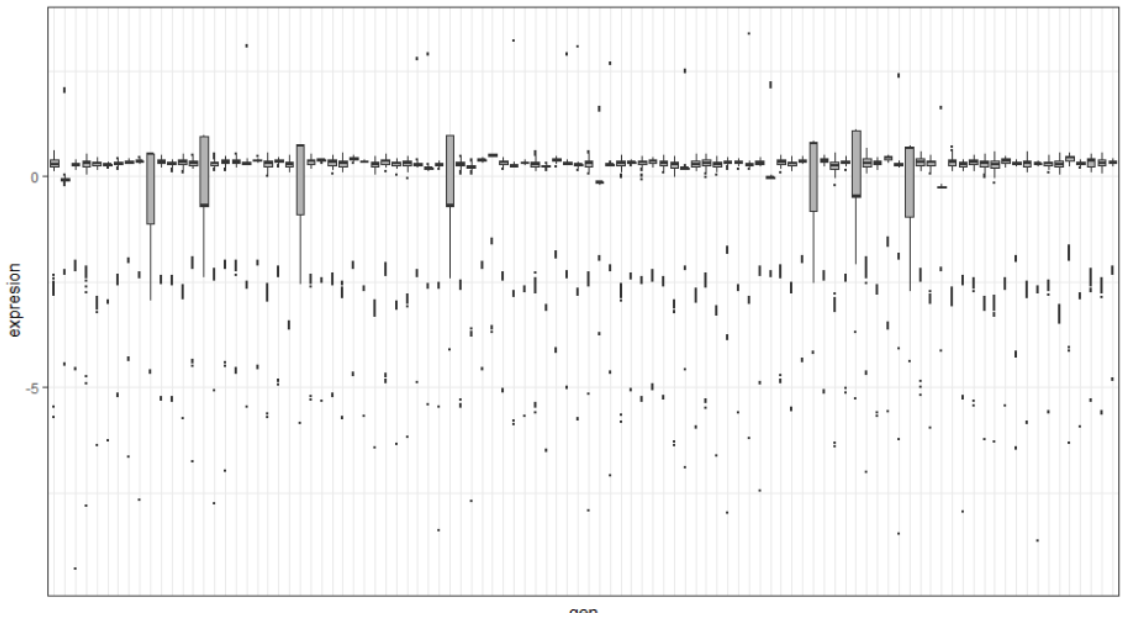

Figura 6. Representación de la expresión de 100 genes seleccionados de forma aleatoria.

Un rápido análisis gráfico muestra que la expresión de los genes no se distribuye de forma normal ni con varianza constante. Esto significa que los resultados del ANOVA no son precisos, por lo que no se deben emplear los p-value para determinar significancia estadística. Sin embargo, sí pueden resultar útiles como criterio para ordenar los genes de mayor a menor potencial importancia.

Se aplica un análisis ANOVA para cada uno de los genes, y para evitar que la selección de genes esté excesivamente influenciada por los datos de entrenamiento, y así minimizar el riesgo de overfitting, se implementa un proceso de bootstraping. El algoritmo seguido es el siguiente:

1. Para cada iteración de **bootstraping:**

1.1 Se genera una nueva pseudo-muestra por muestreo repetido con reposición, del mismo tamaño que la muestra original.

1.2 Se calcula el p-value asociado a cada gen mediante un ANOVA.

2. Se calcula el p-value promedio de cada gen.

3. Se seleccionan los top n genes con menor p-value promedio.

VERSIÓN NO PARALELIZADA

Se emplea un número de resampling bajo para que no tarde demasiado. Para valores más elevados se empleará la versión paralelizada que se describe más adelante. Se emplean semillas para que los muestreos sean reproducibles.

	gen <chr>	resample_1 <dbl>	resample_2 <dbl>	resample_3 <dbl>	pvalue_medio <dbl>
1	ILMN_1695902	9.015885e-09	5.078477e-08	4.350819e-08	3.443628e-08
2	ILMN_1753498	3.446316e-06	4.213725e-07	5.387510e-07	1.468813e-06
3	ILMN_1696708	1.219735e-71	1.652617e-05	8.826645e-12	5.508725e-06
4	ILMN_1755415	2.303400e-05	1.659638e-05	7.081651e-13	1.321013e-05
5	ILMN_1670256	6.515317e-05	6.414452e-07	5.130401e-05	3.903287e-05
6	ILMN_1796119	1.055412e-04	2.136416e-05	9.618019e-08	4.233383e-05

Tabla 9. Resultados ANOVA

Para agilizar el proceso, es recomendable paralelizar el loop externo.

VERSIÓN PARALELIZADA DE BOOTSTRAPPING PARA FILTRADO POR ANOVA

	gen <chr>	resample_1 <dbl>	resample_2 <dbl>	resample_3 <dbl>	resample_4 <dbl>	resample_5 <dbl>	resample_6 <dbl>
1	ILMN_1753498	3.446316e-06	4.213725e-07	5.387510e-07	3.429104e-05	5.726916e-06	4.391515e-05
2	ILMN_1718853	2.881590e-05	1.165828e-04	6.488268e-06	5.495113e-07	1.824826e-05	3.508611e-05
3	ILMN_1665655	3.165561e-02	8.314335e-04	2.588234e-04	2.250280e-02	9.153552e-04	7.741737e-06
4	ILMN_1809027	2.758335e-03	4.089726e-05	2.156115e-05	6.720942e-04	1.354974e-02	1.117978e-04
5	ILMN_1695902	9.015885e-09	5.078477e-08	4.350819e-08	6.756231e-08	8.507699e-11	1.346809e-02
6	ILMN_1724437	2.667046e-03	9.359005e-09	4.105013e-05	1.582740e-04	2.729669e-05	1.519411e-03

Tabla 10. Resultados ANOVA versión paralelizada.

	PROBEID <chr>	ENTREZID <chr>	SYMBOL <chr>	GENENAME <chr>
1	ILMN_1753498	80347	COASY	Coenzyme A synthase
2	ILMN_1718853	7385	UQCRC2	ubiquinol-cytochrome c reductase core protein 2
3	ILMN_1665655	51496	CTDSPL2	CTD small phosphatase like 2
4	ILMN_1809027	55101	DMAC2	distal membrane arm assembly complex 2
5	ILMN_1695902	162239	ZFP1	ZFP1 zinc finger protein
6	ILMN_1724437	23464	GCAT	glycine C-acetyltransferase
7	ILMN_1803742	829	CAPZA1	capping actin protein of muscle Z-line subunit alpha 1
8	ILMN_1712161	54880	BCOR	BCL6 corepressor
9	ILMN_1779252	10346	TRIM22	tripartite motif containing 22
10	ILMN_1815345	8669	EIF3J	eukaryotic translation initiation factor 3 subunit J

Tabla 11. Anotación de los 10 primeros genes obtenidos

Se filtran los 100, 50 y 25 genes identificados como más relevantes mediante anova.

4.7.2. Signal to noise (S2N)

Otra forma de identificar genes característicos de cada fase es ordenándolos acorde al valor de estadístico Signal-to-Noise (S2N). Este estadístico se calcula con la siguiente ecuación:

$$S2N = \frac{\mu_{grupoi} - \mu_{restodegrupos}}{\sigma_{grupoi} + \sigma_{restodegrupos}}$$

Cuanto mayor es la diferencia entre la expresión promedio de un gen en un grupo respecto a los demás, mayor es el valor absoluto de S2N. Puede considerarse que, para un determinado grupo, los genes con un valor alto de S2N son buenos representantes.

El algoritmo seguido para calcular los valores S2N es:

Para cada grupo i:

- Se separan los valores del grupo i y los del resto de grupos en dos dataframe distintos.

- Se calcula la media y la desviación típica de cada gen en el grupo i.

- Se calcula la media y la desviación típica de cada gen en resto de grupos (de forma conjunta).

- Empleando las medias y desviaciones típicas calculadas en los dos puntos anteriores, se calcula el estadístico Signal-to-Noise de cada gen para el grupo i.

Como resultado de este algoritmo, se ha obtenido el valor del estadístico Signal-to-Noise para cada uno de los genes, en cada una de las clases. Los resultados se han almacenado en una lista. A continuación, se seleccionan los 10 genes con mayor valor absoluto en cada uno de los grupos.

Si se cumple que el estadístico signal-to-noise es capaz de identificar en cada grupo genes cuya expresión es particularmente alta o baja en comparación al resto de grupos (genes representativos del estadío de la enfermedad), cabe esperar que, los genes con mayor valor absoluto signal-to-noise, sean distintos en cada clase. Si esto es cierto, la intersección de los top 10 genes de los 4 grupos, debería contener aproximadamente 40 genes, y así es.

	PROBEID <chr>	ENTREZID <chr>	SYMBOL <chr>	GENENAME <chr>
1	ILMN_1715169	3123	HLA-DRB1	major histocompatibility complex, class II, DR beta 1
2	ILMN_1778010	9235	IL32	interleukin 32
3	ILMN_1755990	92270	ATP6AP1L	ATPase H+ transporting accessory protein 1 like
4	ILMN_1769031	2475	MTOR	mechanistic target of rapamycin kinase
5	ILMN_1749892	54583	EGLN1	egl-9 family hypoxia inducible factor 1
6	ILMN_1775036	26984	SEC22A	SEC22 homolog A, vesicle trafficking protein
7	ILMN_1702534	51744	CD244	CD244 molecule
8	ILMN_1709936	90624	LYRM7	LYR motif containing 7
9	ILMN_1665838	29957	SLC25A24	solute carrier family 25 member 24
10	ILMN_1803686	100	ADA	adenosine deaminase

Tabla 12. Anotación de los 10 primeros genes obtenidos por filtrado S2N

Al igual, que en el filtrado por ANOVA, para evitar que la selección este excesivamente influenciada por la muestra de entrenamiento, es conveniente recurrir a un proceso de resampling y agregar los resultados. Esta vez, como método de agregación se emplea la media.

VERSIÓN PARALELIZADA DE BOOTSTRAPPING PARA FILTRADO POR SIGNAL TO NOISE

En cada elemento de la lista resultados_s2n se ha almacenado el resultado de una repetición bootstrapping, que a su vez, es otra lista con los valores S2N de cada gen en cada grupo. Para obtener un único listado final por clase, se tienen que agregar los valores obtenidos en las diferentes repeticiones.

Para cada clase se calcula el s2n medio de los genes y se devuelven los 10 genes con mayor S2N absoluto.

Se identifica la intersección entre los genes seleccionados para cada tipo (10x4=40), y se eliminan aquellos que son comunes para varios estadíos (aparecen más de dos veces): lo denominamos *S2N60*.

El mismo proceso se repite pero seleccionando únicamente los top 5 genes por grupo (5x4 = 20): lo denominamos *S2N30*.

4.7.3. Comparación de filtrados

Se estudia cuantos genes en común se han seleccionado con cada uno de los métodos. La selección de genes resultante con ambos métodos es muy distinta: se obtienen 16 en común entre el filtrado por anova pvalue100 y S2N de 40 genes, y 7 entre el filtrado anova pvalue50 y S2N de 20 genes.

4.7.4. Reducción de dimensionalidad (PCA)

Los métodos de reducción de dimensionalidad permiten simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conservan su información. Supóngase que existe una muestra con n individuos cada uno con p variables (X1, X2, …, Xp), es decir, el espacio muestral tiene p dimensiones. El objetivo de estos métodos es encontrar un número de factores subyacentes (z<p) que expliquen aproximadamente lo mismo que las p variables originales. Donde antes se necesitaban p valores para caracterizar a cada individuo, ahora bastan z valores. Dos de las técnicas más utilizadas son: PCA y t-SNE.

Se aplica un PCA a los niveles de expresión y se conservan las componentes principales hasta alcanzar un 95% de varianza explicada.

4.8. Modelos aplicados [18]

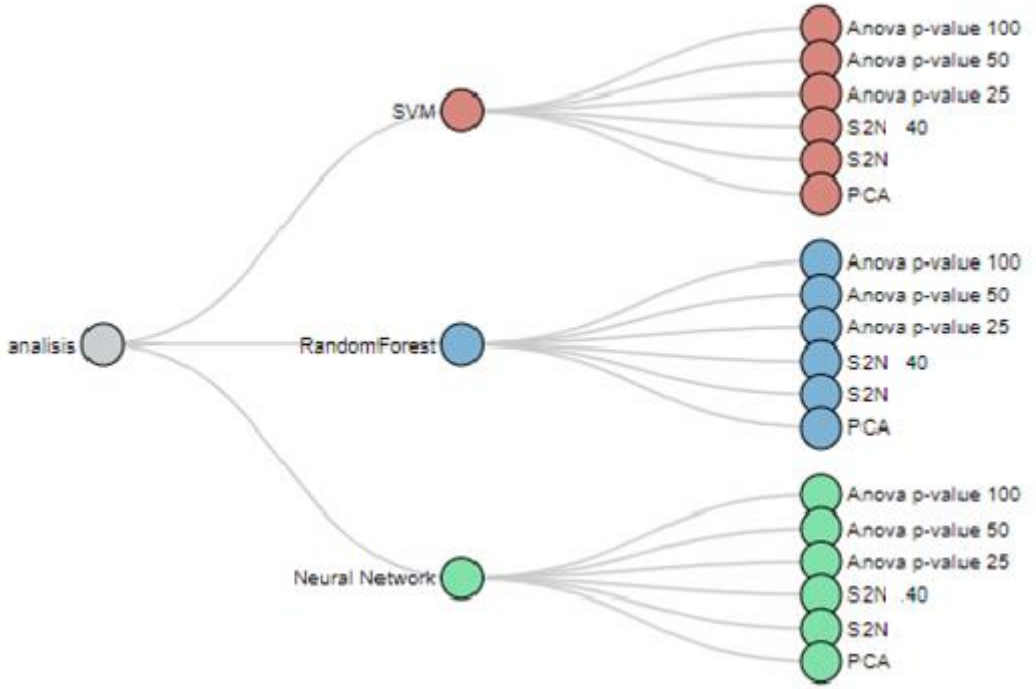

Figura 7. Diagrama de los modelos ajustados y los genes empleados

En los siguientes apartados se entrenan diferentes modelos de machine learning con el objetivo de compararlos e identificar el que mejor resultado obtiene clasificando los tumores. Además, se comparan los diferentes filtrados de genes. Los modelos se entrenan, optimizan y comparan empleando las funcionalidades que ofrece el ***paquete caret***.

Según la bibliografía consultada, estos tres modelos son los más utilizados en la construcción de clasificadores diagnósticos.

4.8.1. SVM: Máquinas de Vector Soporte (Support Vector Machines, SVMs)

El método svmRadial de caret emplea la función ksvm() del paquete kernlab. Este algoritmo tiene 2 hiperparámetros:

- sigma: coeficiente del kernel radial.

- C: penalización por violaciones del margen del hiperplano.

4.8.2. Random Forest

El método ranger de caret emplea la función ranger() del paquete ranger. Este algoritmo tiene 3 hiperparámetros:

- mtry: número predictores seleccionados aleatoriamente en cada árbol.

- min.node.size: tamaño mínimo que tiene que tener un nodo para poder ser dividido.

- splitrule: criterio de división.

4.8.3. Neural Network

El método nnet de caret emplea la función nnet() del paquete nnet para crear redes neuronales con una capa oculta. Este algoritmo tiene 2 hiperparámetros:

- size: número de neuronas en la capa oculta.

- decay: controla la regularización durante el entrenamiento de la red.

4.8.4. Comparación de modelos [18]

La finalidad última de un modelo es predecir la variable respuesta en observaciones futuras o en observaciones que el modelo no ha “visto” antes. El

error mostrado por defecto tras entrenar un modelo suele ser el error de entrenamiento, el error que comete el modelo al predecir las observaciones que ya ha “visto”. Si bien estos errores son útiles para entender cómo está aprendiendo el modelo (estudio de residuos), no es una estimación realista de cómo se comporta el modelo ante nuevas observaciones (el error de entrenamiento suele ser demasiado optimista). Para conseguir una estimación más certera, se tiene que recurrir a un conjunto de test o emplear estrategias de validación basadas en *resampling*.

Los métodos de validación, también conocidos como *resampling*, son estrategias que permiten estimar la capacidad predictiva de los modelos cuando se aplican a nuevas observaciones, haciendo uso únicamente de los datos de entrenamiento. La idea en la que se basan todos ellos es la siguiente: el modelo se ajusta empleando un subconjunto de observaciones del conjunto de entrenamiento y se evalúa (calcular una métrica que mida como de bueno es el modelo, por ejemplo, *accuracy*) con las observaciones restantes. Este proceso se repite múltiples veces y los resultados se agregan y promedian. Gracias a las repeticiones, se compensan las posibles desviaciones que puedan surgir por el reparto aleatorio de las observaciones. La diferencia entre métodos suele ser la forma en la que se generan los subconjuntos de entrenamiento/validación.

Diferencia entre el error de validación y error de test (*evaluation error* y *test error*): el primero es el error que comete el modelo al predecir observaciones del conjunto de validación y el segundo del conjunto de test. En ambos casos son observaciones que el modelo no ha “visto”.

Error de validación

Los errores de validación se obtienen por bootstrapping. Una muestra bootstrap es una muestra obtenida a partir de la muestra original por muestreo aleatorio con reposición, y del mismo tamaño que la muestra original. Muestreo aleatorio con reposición (*resampling with replacement*) significa que, después de que una observación sea extraída, se vuelve a poner a disposición para las siguientes extracciones. Como resultado de este tipo de muestreo, algunas observaciones aparecerán múltiples veces en la muestra bootstrap y otras ninguna. Las observaciones no seleccionadas reciben el nombre de out-of-bag (OOB). Por cada iteración de bootstrapping se genera una nueva muestra bootstrap, se ajusta el modelo con ella y se evalúa con las observaciones out-of-bag.

1. Obtener una nueva muestra del mismo tamaño que la muestra original mediante muestro aleatorio con reposición.
2. Ajustar el modelo empleando la nueva muestra generada en el paso 1.

3. Calcular el error del modelo empleando aquellas observaciones de la muestra original que no se han incluido en la nueva muestra. A este error se le conoce como error de validación.
4. Repetir el proceso n veces y calcular la media de los n errores de validación.
5. Finalmente, y tras las n repeticiones, se ajusta el modelo final empleando todas las observaciones de entrenamiento originales.

La naturaleza del proceso de bootstrapping genera cierto bias en las estimaciones que puede ser problemático cuando el conjunto de entrenamiento es pequeño.

4.8.5. Model ensemble (stacking) [18]

A modo general, el término model ensembling hace referencia a la combinación de las predicciones de dos o más modelos distintos, con el objetivo de mejorar las predicciones finales. Esta estrategia se basa en la asunción de que, distintos modelos entrenados independientemente, emplean distintos aspectos de los datos para realizar las predicciones, es decir, cada uno es capaz de identificar parte de la “verdad” pero no toda ella. Combinando la perspectiva de cada uno de ellos, se obtiene una descripción más detallada de la verdadera estructura subyacente en los datos.

La clave para que el ensembling consiga mejorar los resultados es la diversidad de los modelos. Si todos los modelos combinados son similares entre ellos, no podrán compensarse unos a otros. Por esta razón, se tiene que intentar combinar modelos que sean lo mejor posible a nivel individual y lo más diferentes entre ellos.

Las formas más simples de combinar las predicciones de varios modelos son: emplear la media para problemas de regresión y la moda para problemas de clasificación. También es posible ponderar estas agregaciones dando distinto peso a cada modelo, por ejemplo, en proporción al accuracy que han obtenido de forma individual.

5. Resultados

Se utilizan dos métodos de filtrado, por anova ***p value (100, 50 y 25),*** y por S2N (40 y 20 genes, pero denominados ***S2N60 y S2N30*** respectivamente), y una reducción de dimensionalidad por **PCA.**

5.1. Comparación de modelos en función de hiperparámetros y filtrado.

5.1.1. SVM Radial

Los mejores modelos se obtienen con los siguientes valores de sigma y C:

Filtrado	Sigma	C	Accuracy
p value 100	0.0001	100	**0.8950512**
p value 50	0.0001	500	0.8819139
p value 25	0.0001	100	0.8505650
S2N 40	0.001	10	0.7257777
S2N 20	0.0001	200	0.7100331
PCA	0.1	1	0.4841459

Tabla 13. Resultados de *accuracy* con SVM radial en función del filtrado y los hiperparámetros utilizados.

El mejor modelo se obtiene con el filtrado p value 100.

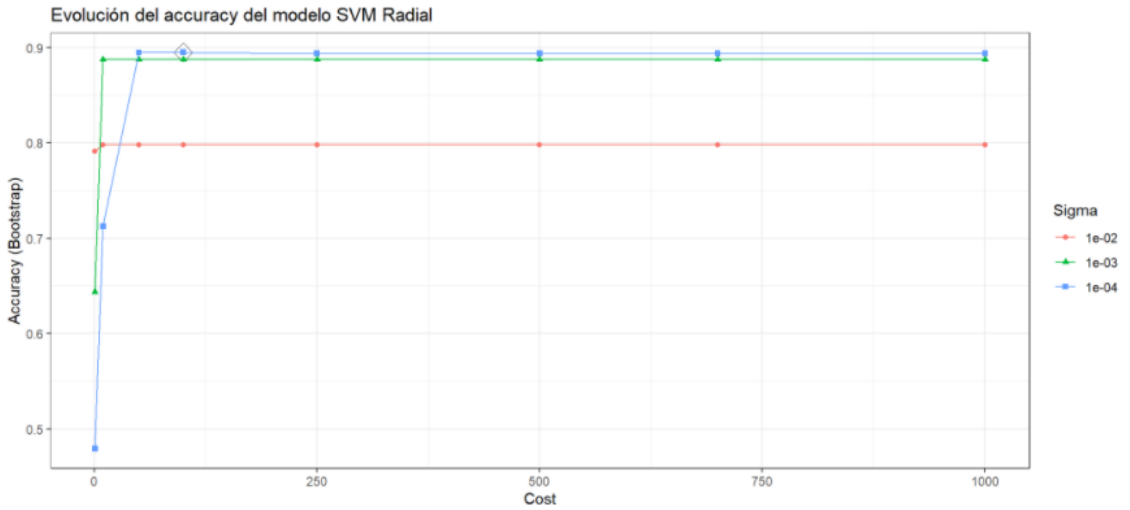

Figura 8. Evolución de accuracy del modelo SVM Radial filtrado por p value 100.

5.1.2. Random Forest

Los mejores modelos se obtienen con los siguientes valores de mtry y Min.node.size:

Filtrado	mtry	Min.node.size	Accuracy
p value 100	5	10	0.6975859
p value 50	5	2	0.7144791
p value 25	2	2	**0.7268851**
S2N 40	5	3	0.6788081
S2N 20	5	2	0.6570088
PCA	5	4	0.4483812

Tabla 14. Resultados de *accuracy* con Random Forest en función del filtrado y los hiperparámetros utilizados.

El mejor modelo se obtiene con el filtrado p value 25.

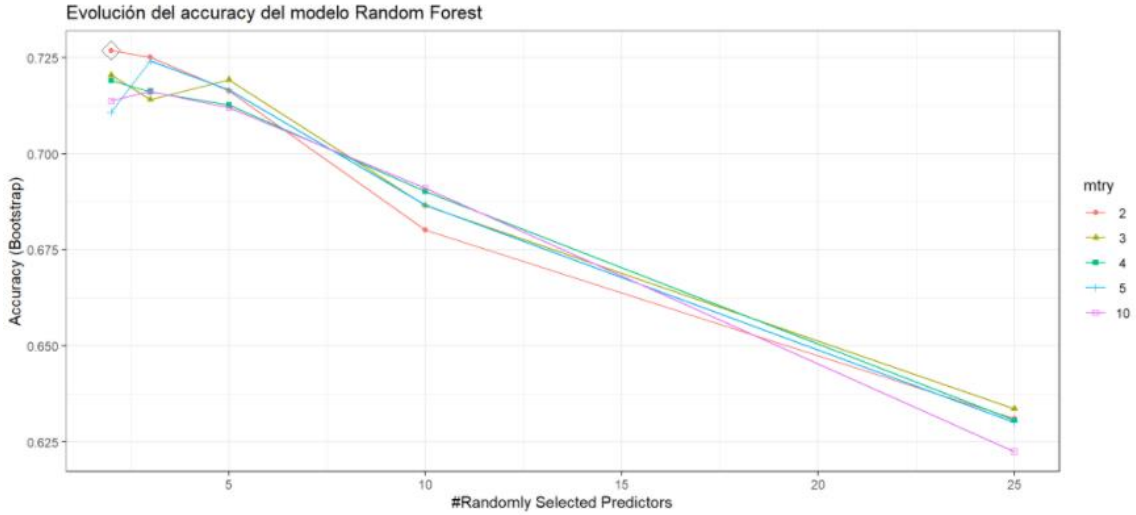

Figura 9. Evolución de accuracy del modelo Random Forest filtrado por p value 25.

5.1.3. ANN

Los mejores modelos ANN se obtienen con los siguientes size y decay:

Filtrado	size	decay	Accuracy
p value 100	10	0.01	**0.9554087**
p value 50	5	0.01	**0.9159172**
p value 25	10	0.1	0.8303917
S2N 40	15	0.1	0.6663865
S2N 20	30	0.1	0.6856681
PCA	5	4	0.4483812

Tabla 15. Resultados de *accuracy* con Neural Network en función del filtrado y los hiperparámetros utilizados.

El mejor modelo se obtiene con el filtrado p value 100, aunque con el filtrado p value 50 también se supera la accuracy obtenida en los modelos SVM y Random Forest.

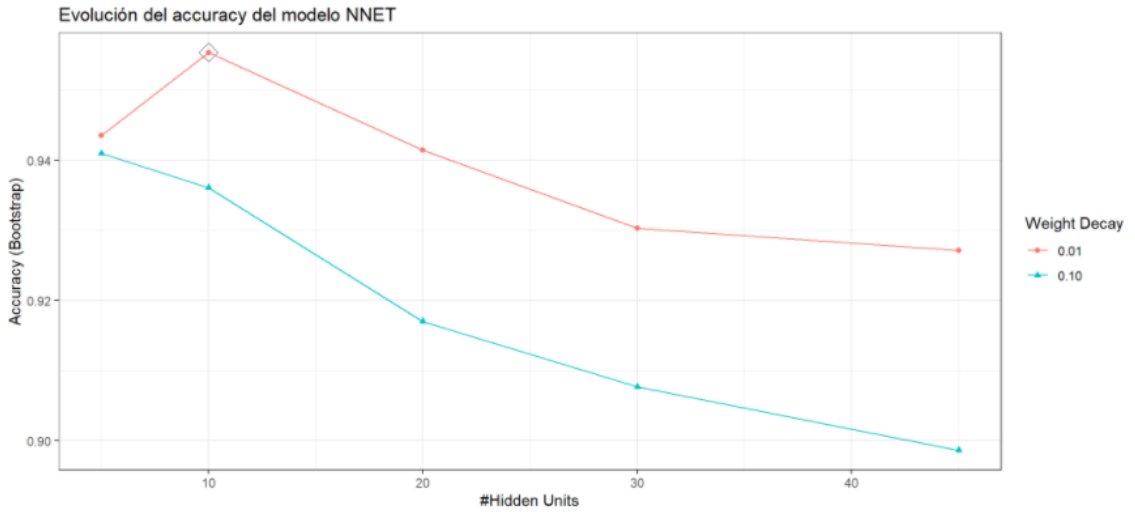

Figura 10. Evolución de accuracy del modelo ANN filtrado por p value 100.

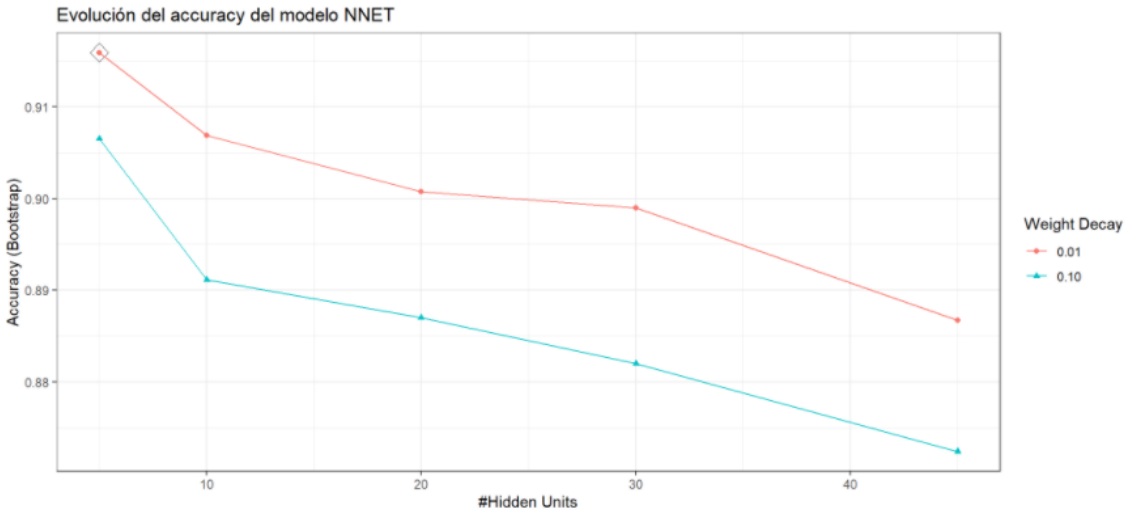

Figura 11. Evolución de accuracy del modelo ANN filtrado por p value 50.

5.2. Error de validación

modelo <chr>	Accuracy <dbl>	Kappa <dbl>
NNET_pvalue_100	0.9554087	0.9288216
NNET_pvalue_50	0.9159172	0.8692827
SVMrad_pvalue_100	0.8950512	0.8371880
SVMrad_pvalue_50	0.8819139	0.8078060
SVMrad_pvalue_25	0.8505650	0.7477939
NNET_pvalue_25	0.8303917	0.7199855
RF_pvalue_25	0.7268851	0.5186574
SVMrad_s2n_60	0.7257777	0.5316720
RF_pvalue_50	0.7144791	0.4988873
RF_s2n_30	0.7106013	0.5089746
SVMrad_s2n_30	0.7100331	0.4990588
RF_pvalue_100	0.6975859	0.4667225
NNET_s2n_60	0.6955528	0.5361675
NNET_s2n_30	0.6856681	0.5166723
RF_s2n_60	0.6788081	0.4618042
SVMrad_pca	0.4841459	0.0000000
RF_pca	0.4483812	-0.0155980

Tabla 16. Promedio métricas resamples.

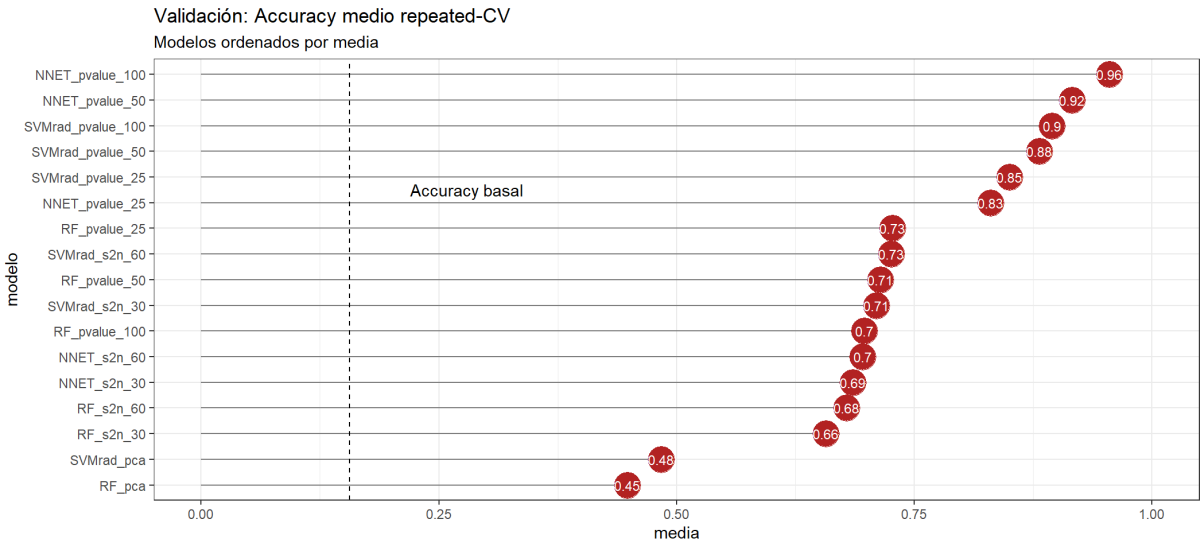

Figura 12. Validación: Accuracy medio repeated CV

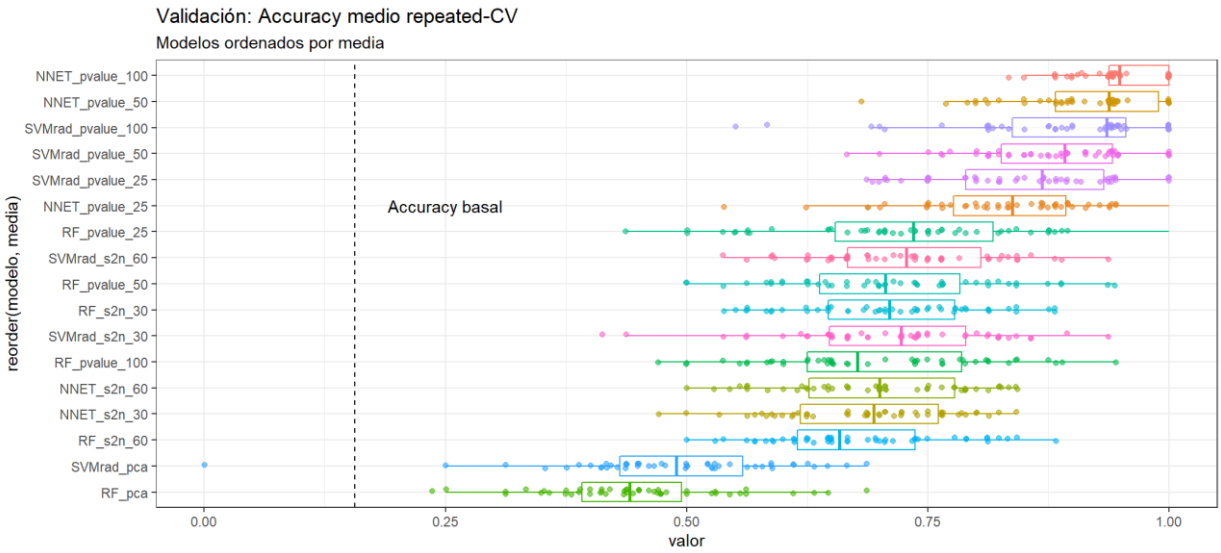

Figura 13. Validación: Accuracy medio repeated CV

Una de las ventajas de los métodos de validación es que, si se han empleado exactamente los mismos datos en todos los modelos (en este caso es así gracias a las semillas), se pueden aplicar test de hipótesis que permitan determinar si las diferencias observadas entre modelos son significativas o si solo son debidas a variaciones aleatorias.

Acorde a los errores de validación obtenidos por *bootstrapping*, los mejores modelos se consiguen empleando los genes seleccionados por el estadístico *p value100 y p value50,* y con los algoritmos *ANN y SVM*. Se comparan los 3 mejores algoritmos para determinar si hay evidencias de que uno de ellos sea superior a los demás.

Si se cumple la condición de normalidad, se pueden aplicar un t-test de datos pareados para comparar el *accuracy* medio de cada modelo.

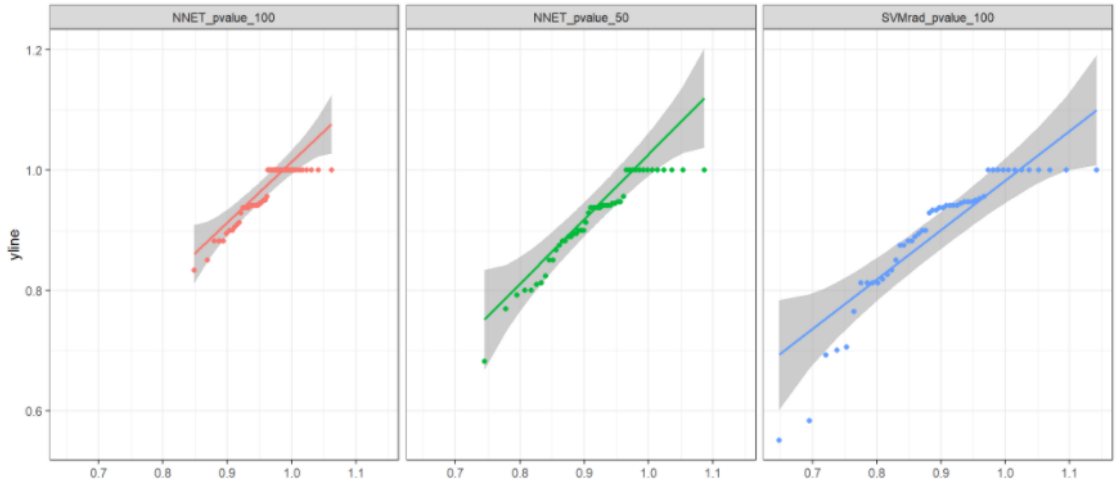

Figura 14. Análisis de normalidad de los modelos NNET_pvalue_100, NNET_pvalue_50 y SVMrad_pvalue_100.

El análisis gráfico no muestra grandes desviaciones de la normal, además, dado que se dispone de más de 30 valores por grupo, el t-test tiene cierta robustez. Se procede a comparar los modelos, y para un nivel de significancia α=0.05, se encuentran evidencias para aceptar que el accuracy promedio de los modelos es distinto. Por lo tanto**, según el proceso de validación por *bootstraping*, el mejor modelo es ANN_pvalue_100**

5.3. Error de test

	SVMrad_pvalue_100 <fct>	SVMrad_pvalue_50 <fct>	SVMrad_pvalue_25 <fct>	SVMrad_s2n_60 <fct>	SVMrad_s2n_30 <fct>	SVMrad_pca <fct>
1	CIS	CIS	CIS	RR	PP	RR
2	CIS	CIS	CIS	CIS	PP	RR
3	RR	RR	RR	CIS	RR	RR
4	RR	RR	RR	RR	RR	RR
5	RR	RR	RR	RR	RR	RR
6	CIS	RR	CIS	CIS	CIS	RR

RF_pvalue_100 <fct>	RF_pvalue_50 <fct>	RF_pvalue_25 <fct>	RF_s2n_60 <fct>	RF_s2n_30 <fct>	RF_p... <fct>	NNET_s2n_60 <fct>	NNET_s2n_30 <fct>
RR	RR	RR	RR	RR	RR	SP	PP
CIS	CIS	CIS	CIS	CIS	RR	CIS	CIS
RR	RR	RR	RR	CIS	RR	RR	CIS
RR	RR	RR	RR	RR	RR	RR	PP
RR	RR	RR	RR	RR	RR	RR	RR
CIS	CIS	CIS	RR	CIS	RR	CIS	PP

RF_pca <fct>	NNET_s2n_60 <fct>	NNET_s2n_30 <fct>	NNET_pvalue_100 <fct>	NNET_pvalue_50 <fct>	NNET_pvalue_25 <fct>	valor_real <chr>
RR	SP	PP	CIS	CIS	CIS	CIS
RR	CIS	CIS	CIS	CIS	CIS	CIS
RR	RR	CIS	RR	RR	RR	CIS
RR	RR	PP	RR	RR	RR	RR
RR	RR	RR	RR	RR	RR	RR
RR	CIS	PP	PP	PP	CIS	RR

Tabla 17. Predicciones de los diferentes modelos.

modelo <chr>	accuracy_validacion <dbl>	accuracy_test <dbl>
SVMrad_pvalue_25	0.8505650	0.9090909
NNET_pvalue_50	0.9159172	0.8181818
RF_pvalue_25	0.7268851	0.8181818
RF_pvalue_50	0.7144791	0.8181818
SVMrad_pvalue_50	0.8819139	0.8181818
NNET_s2n_60	0.6955528	0.7272727
RF_pvalue_100	0.6975859	0.7272727
RF_s2n_30	0.7106013	0.7272727
SVMrad_pvalue_100	0.8950512	0.7272727
NNET_pvalue_100	0.9554087	0.6363636

RF_pca	0.4483812	0.6363636
RF_s2n_60	0.6788081	0.6363636
SVMrad_pca	0.4841459	0.6363636
SVMrad_s2n_30	0.7100331	0.6363636
SVMrad_s2n_60	0.7257777	0.5454545
NNET_s2n_30	0.6856681	0.4545455
NNET_pvalue_25	0.8303917	*NA*

Tabla 18. Accuracy de validación y test de los diferentes modelos.

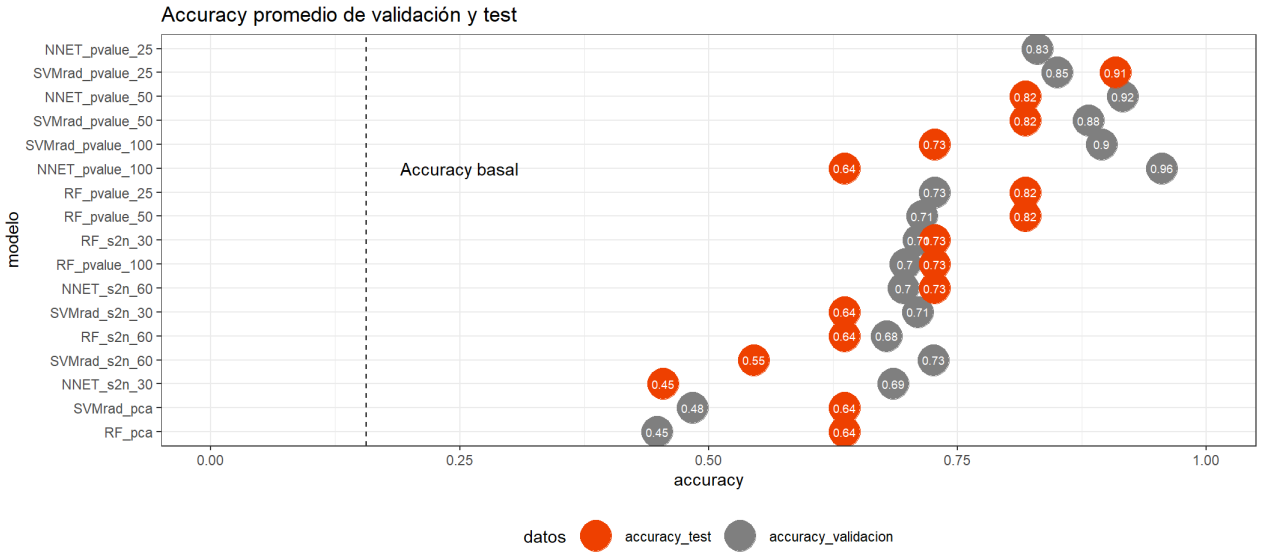

Figura 15. Accuracy promedio de validación y test.

Acorde al *accuracy* del test, los mejores modelos son SVMrad_pvalue_25, NNET_pvalue_50 y SVMrad_p value50 (con el mismo valor de *accuracy* que RF_pvalue_25 y 50).

La matriz de confusión muestra que el modelo clasifica peor unos tipos que otros.

Para SVMrad_pvalue_25:

Es muy específico para la clase CIS, y muy sensible para la clase RR. No es buen predictor para las clases SP y PP.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CIS PP RR SP
##        CIS   2  0  1  0
##        PP    0  0  0  0
##        RR    1  1  5  1
##        SP    0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.6364
##                  95% CI : (0.3079, 0.8907)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.3853
##
##                   Kappa : 0.3125
##
## Statistics by Class:
##
##                      Class: CIS Class: PP Class: RR Class: SP
## Sensitivity              0.6667   0.00000    0.8333   0.00000
## Specificity              0.8750   1.00000    0.4000   1.00000
## Pos Pred Value           0.6667       NaN    0.6250       NaN
## Neg Pred Value           0.8750   0.90909    0.6667   0.90909
## Prevalence               0.2727   0.09091    0.5455   0.09091
## Detection Rate           0.1818   0.00000    0.4545   0.00000
## Detection Prevalence     0.2727   0.00000    0.7273   0.00000
## Balanced Accuracy        0.7708   0.50000    0.6167   0.50000
```

Para NNET_pvalue_50:

Como en el caso anterior, no es buen predictor de las clases PP y SP.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CIS PP RR SP
##        CIS   2  0  0  0
##        PP    0  0  1  0
##        RR    1  1  5  1
##        SP    0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.6364
##                  95% CI : (0.3079, 0.8907)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.3853
##
##                   Kappa : 0.3333
##

##
## Statistics by Class:
##
##                      Class: CIS Class: PP Class: RR Class: SP
## Sensitivity              0.6667   0.00000    0.8333   0.00000
## Specificity              1.0000   0.90000    0.4000   1.00000
## Pos Pred Value           1.0000   0.00000    0.6250       NaN
## Neg Pred Value           0.8889   0.90000    0.6667   0.90909
## Prevalence               0.2727   0.09091    0.5455   0.09091
## Detection Rate           0.1818   0.00000    0.4545   0.00000
## Detection Prevalence     0.1818   0.09091    0.7273   0.00000
## Balanced Accuracy        0.8333   0.45000    0.6167   0.50000
```

Para RF_pvalue_50:
Ocurre lo mismo, las clases SP y PP son difíciles de predecir.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CIS PP RR SP
##        CIS   1  0  1  0
##        PP    0  0  0  0
##        RR    2  1  5  1
##        SP    0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5455
##                  95% CI : (0.2338, 0.8325)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.6214
##
##                   Kappa : 0.0984
##
## Statistics by Class:
##
##                      Class: CIS Class: PP Class: RR Class: SP
## Sensitivity             0.33333   0.00000    0.8333   0.00000
## Specificity             0.87500   1.00000    0.2000   1.00000
## Pos Pred Value          0.50000       NaN    0.5556       NaN
## Neg Pred Value          0.77778   0.90909    0.5000   0.90909
## Prevalence              0.27273   0.09091    0.5455   0.09091
## Detection Rate          0.09091   0.00000    0.4545   0.00000
## Detection Prevalence    0.18182   0.00000    0.8182   0.00000
## Balanced Accuracy       0.60417   0.50000    0.5167   0.50000
```

5.4. Mejor modelo

Para identificar cuál de todos es el mejor modelo, conviene tener en cuenta los dos tipos de error: el de validación, en este caso bootstrapping, y el de test. En vista de los resultados obtenidos, no puede afirmarse que, para este problema, exista un modelo que supere claramente a todos los demás, de ahí que, pequeñas diferencias, provoquen cambios en el orden de los modelos entre validación y de test. NNET_pvalue_50 parece un buen candidato, ya que obtiene valores muy altos en ambos test, seguido por SVMrad_pvalue_25.

En general, los filtrados por p value dan mejor resultado. En cambio la reducción por PCA no es conveniente.

5.5. Model ensemble (Stacking)

	SVMrad_pvalue_100 <fct>	NNET_pvalue_100 <fct>	NNET_pvalue_50 <fct>	RF_pvalue_50 <fct>	RF_pvalue_25 <fct>	mo... <chr>
1	CIS	CIS	CIS	RR	RR	CIS
2	CIS	CIS	CIS	CIS	CIS	CIS
3	RR	RR	RR	RR	RR	RR
4	RR	RR	RR	RR	RR	RR
5	RR	RR	RR	RR	RR	RR
6	CIS	PP	PP	CIS	CIS	CIS

Tabla 19. Predicciones ensemble

La media de aciertos del modelo es de 0.6363. En este caso, el *ensemble* de los modelos no consigue una mejora.

6. Discusión

En el presente trabajo se describe un flujo de trabajo de aprendizaje automático basado en la selección de genes diferencialmente expresados y técnicas de *bootstrapping*, que solventa la existencia de clases desbalanceadas y minimiza el riesgo de *overfitting*. Al existir varios miles de posibles predictores, se recurre a métodos de filtrado, que resultan efectivos pero computacionalmente costosos. Se comparan diferente algoritmos que demuestran un alto poder predictivo, en función del filtrado y de los hiperparámetros utilizados. Finalmente se determina el mejor modelo acorde a los errores de validación obtenidos por *bootstrapping.*

Resulta determinante la elección del método de filtrado, siendo en este caso mucho más efectivo el filtrado por anova p value, ya sea de 100, 50 o 25 genes. En cambio, reducir la dimensionalidad por PCA no es conveniente. Eso no significa que otros métodos, como el t-SNE, no puedan ser beneficiosos. Sería una línea a tener en cuenta.

Respecto a los algoritmos de clasificación, claramente ANN es un buen clasificador aunque, según el error de test, RF también es capaz de discriminar entre las diferentes fases de EM con bastante precisión. No puede afirmarse que, para este problema, exista un modelo que supere claramente a todos los demás, de ahí que, pequeñas diferencias, provoquen cambios en el orden de los modelos entre validación y de test. El modelo SVM, que según la bibliografía consultada es uno de los más utilizados en la construcción de clasificadores diagnósticos, pero en el presente estudio no da mejor resultados que los otros.

Hay que destacar que, aunque algunos modelos tengan una *accuracy* satisfactoria, no significa que sean buenos predictores para algunas de las clases, tal y como se ha podido comprobar en el error de test, donde los valores de sensibilidad y especificidad de estas clases no sonn buenos. Sería necesario aplicar los modelos a otros conjuntos de datos o modificar los algoritmos para averiguar si es un fenómeno debido al desequilibrio de clases.

Aunque una posible explicación a este suceso podría ser que entre las dos clases de EM progresiva no existen una expresión diferencial notable.

Los conjuntos de datos de expresión génica suponen un gran desafío para los algoritmos de clasificación porque la alta dimensionalidad que nos proporcionan los microarrays puede llevar a que los modelos se ajusten demasiado a los datos de entrenamiento y tengan un rendimiento deficiente en nuevas muestras. Además, los conjuntos de datos de EM están sujetos a un desequilibrio de clases, lo que agrega mayor complejidad al proceso de aprendizaje.

Aunque varios estudios han explorado los patrones de expresión génica de la sangre en la EM utilizando análisis estadísticos tradicionales,[9][10][11] solo un par de informes han intentado aplicar el aprendizaje automático a la transcriptómica sanguínea. Algunos se limitaron a la discriminación entre la forma RR MS y los controles[33], y sólo uno diferencia entre las 4 fases[13].

En conclusión, el presente estudio justifica la aplicación de métodos de aprendizaje automático a los transcriptomas sanguíneos y ofrece una línea

sólida para la generación de clasificadores que respalden el diagnóstico y pronóstico de la EM.

7. Conclusiones

El uso de las técnicas de aprendizaje automático empleadas en el análisis de datos del ámbito biomédico, va ligado al conocimiento de la programación (en este caso R) y al entendimiento de los algoritmos que se quieren utilizar. De esta manera, la elección del modelo a utilizar será más efectiva y se obtendrán mejores resultados predictivos.

Además, el origen y preprocesado de los datos influye en los resultados finales. La gran cantidad de predictores que proporcionan los microarrays hacen necesario un filtrado de los datos. La elección de esta metodología también es clave, independientemente del modelo de entrenamiento utilizado.

Los datos procedentes de microarrays pueden tener un desbalance de clases, por lo que también es necesario conocer y utilizar las técnicas de *resampling*. Para evitar que la selección de genes esté excesivamente influenciada por los datos de entrenamiento, y así minimizar el riesgo de *overfitting*, se implementa un proceso de *bootstraping*.

Los objetivos de este trabajo eran:

- *Construir un clasificador potente como herramienta clínica para el diagnóstico de las diferentes fases de la EM*

- *Analizar la capacidad de distintos algoritmos de aprendizaje automático para encontrar patrones en los niveles que expresión genética que permitan clasificar distintos tipos de estadíos de esclerosis múltiple (EM) de forma correcta.*

- *Determinar si existe algún método de aprendizaje automático capaz de predecir, con un porcentaje de acierto alto, el tipo de esclerosis múltiple en función sus niveles de expresión.*

- *Determinar, entre los varios miles de genes disponibles, si son todos importantes en vista a la clasificación o son solo unos pocos los que aportan información relevante.*

La aplicación de métodos de aprendizaje automático al conjunto de datos utilizado ofrece una línea sólida para la generación de clasificadores que respalden el diagnóstico y pronóstico de la EM, y ofrecen un porcentaje de acierto alto.

Con el filtrado se ha visto que los modelos consiguen buenos resultados sin necesidad de utilizar todos los genes del conjunto de datos inicial.

7.1. Líneas de futuro

La EM se considera la principal causa de discapacidad no traumática en todo el mundo en adultos jóvenes. La estimación más precisa de la Federación Internacional de EM hasta ahora muestra que hay al menos 1 de cada 3000 personas que viven con la enfermedad. A pesar de más de 40 años de investigación sobre la EM, su etiología sigue siendo desconocida, pero se conoce que tiene un componente genético significativo.

Debido a esto, y al avance contínuo de lás técnicas de aprendizaje automático, quedan muchos frentes abiertos y posibilidades de experimentación, que permitan realizar diagnósticos más precoces y adaptar los tratamientos farmacológicos inmunomoduladores a cada individuo. La identificación de los genes diferencialmente expresados y su valor como variable predictora es otra línea a seguir.

7.2. Seguimiento de la planificación

La planificación se ha ido redefiniendo a lo largo de este cuatrimestre. El principal motivo ha sido el definir las clases para que el porcentaje de aciertos fuera más elevado, y el largo tiempo de computación que implica ejecutar los códigos.

8. Glosario de acrónimos

AA. Aprendizaje automático
EM. Esclerosis múltiple.
RR. Remitente recurrente
PP. Primaria progresiva
SP. Secundaria progresiva
CIS. *Isolated Syndrome*
OND. *Other neurological diseases*
HC. *Healthy control*
PBMC. *Peripheral blood mononuclear cells*
S2N. *Signal to Noise*
ANN. Artificial Neural Network
SVM. Suport Vector Machines
RF. Random Forest.

9. Bibliografía

1. Miller DH, Chard DT, Ciccarelli O. Síndromes clínicamente aislados. Lancet Neurol. 2012; 11 : 157-169.

2. Solomon AJ, Naismith RT, Cross AH Diagnóstico erróneo de esclerosis múltiple: impacto de los criterios de McDonald de 2017 en la práctica clínica. Neurología. 2019; 92 : 26–33.

3. Rice CM, Cottrell D., Wilkins A., Scolding NJ Esclerosis múltiple progresiva primaria: avances y desafíos. J. Neurol. Neurourgo. Psiquiatría. 2013; 84 : 1100–1106.

4. Katz Sand I., Krieger S., Farrell C., Miller AE Incertidumbre diagnóstica durante la transición a la esclerosis múltiple secundaria progresiva. Mult. Scler. 2014; 20 : 1654-1657.

5. Marzullo A., Kocevar G., Stamile C., Durand-Dubief F., Terracina G., Calimeri F., Sappey-Marinier D. Clasificación de perfiles clínicos de esclerosis múltiple a través de redes neuronales convolucionales gráficas. Parte delantera. Neurosci. 2019; 13 : 594.

6. Wang SH, Tang C., Sun J., Yang J., Huang C., Phillips P., Zhang YD Identificación de esclerosis múltiple por red neuronal convolucional de 14 capas con normalización por lotes, abandono y agrupación estocástica. Parte delantera. Neurosci. 2018; 12 : 818.

7. Ion-Mărgineanu A., Kocevar G., Stamile C., Sima DM, Durand-Dubief F., Van Huffel S., Sappey-Marinier D. Enfoque de aprendizaje automático para clasificar los cursos de esclerosis múltiple mediante la combinación de datos clínicos con cargas de lesiones y características metabólicas de resonancia magnética. Parte delantera. Neurosci. 2017; 11 : 398.

8. Kocevar G., Stamile C., Hannoun S., Cotton F., Vukusic S., Durand-Dubief F., Sappey-Marinier D. Conectividad cerebral basada en la teoría de gráficos para la clasificación automática de los cursos clínicos de esclerosis múltiple. Parte delantera. Neurosci. 2016; 10 : 478.

9. Koch MW, Ilnytskyy Y., Golubov A., Metz LM, Yong VW, Kovalchuk O. Perfil de transcriptoma global de la esclerosis múltiple leve recurrente-remitente versus primaria progresiva. EUR. J. Neurol. 2018; 25 : 651–658.

10. Gandhi KS, McKay FC, Cox M., Riveros C., Armstrong N., Heard RN, Vucic S., Williams DW, Stankovich J., Brown M., ANZgene Multiple Sclerosis Genetics Consortium El transcriptoma de ARNm de sangre completa de esclerosis múltiple y las asociaciones genéticas indican una desregulación de las vías específicas de las células T en la patogénesis. Tararear. Mol. Gineta. 2010; 19 : 2134–2143.

11. Srinivasan S., Severa M., Rizzo F., Menon R., Brini E., Mechelli R., Martinelli V., Hertzog P., Salvetti M., Furlan R. Transcriptional disregulation of Interferome in experimental and human Multiple Esclerosis. Sci. Rep. 2017; 7 : 8981.

12. Srinivasan S., Di Dario M., Russo A., Menon R., Brini E., Romeo M., Sangalli F., Costa GD, Rodegher M., Radaelli M. Disregulación de genes y vías de riesgo de EM en distintos etapas de la enfermedad. Neurol. Neuroinmunol. Neuroinflamm. 2017; 4 : e337.

13. Acquaviva, M., Menon, R., Di Dario, M., Dalla Costa, G., Romeo, M., Sangalli, F., Colombo, B., Moiola, L., Martinelli, V., Comi, G., & Farina, C. (2020). Inferring Multiple Sclerosis Stages from the Blood Transcriptome via Machine Learning. Cell reports. Medicine, 1(4), 100053. https://doi.org/10.1016/j.xcrm.2020.100053

14. Moreno V, Solé X. Uso de chips de ADN (microarrays) en medicina: fundamentos técnicos y procedimientos básicos para el análisis estadístico de resultados [Use of DNA chips (microarrays) in medicine: technical foundations and basic procedures for statistical analysis of results]. Med Clin (Barc). 2004;122 Suppl 1:73-9. Spanish. doi: 10.1157/13057538. PMID: 14980164.

15. Brett Lantz. Machine Learning with R. *Second Edition (2015). Packt Publishing Ltd.(UK)*

16. Pinto, M.F., Oliveira, H., Batista, S. et al. Prediction of disease progression and outcomes in multiple sclerosis with machine learning. Sci Rep 10, 21038 (2020). https://doi.org/10.1038/s41598-020-78212-6

17. Franks, J. M., Martyanov, V., Cai, G., Wang, Y., Li, Z., Wood, T. A., & Whitfield, M. L. (2019). A Machine Learning Classifier for Assigning Individual Patients With Systemic

Sclerosis to Intrinsic Molecular Subsets. Arthritis & rheumatology (Hoboken, N.J.), 71(10), 1701–1710. https://doi.org/10.1002/art.40898

18. Ciencia de datos (Joaquín Amat Rodrigo). https://www.cienciadedatos.net/. Visitada en varias ocasiones durante mayo y junio de 2021.

19. Zhao Y, Healy BC, Rotstein D, Guttmann CRG, Bakshi R, Weiner HL, et al. (2017) Exploration of machine learning techniques in predicting multiple sclerosis disease course. PLoS ONE 12(4): e0174866.

20. Ulrich R, Kalkuhl A, Deschl U, Baumgärtner W. Machine learning approach identifies new pathways associated with demyelination in a viral model of multiple sclerosis. *J Cell Mol Med*. 2010;14(1-2):434-448. doi:10.1111/j.1582-4934.2008.00646.x

21. Shang Z, Sun W, Zhang M y col. Identificación de genes clave asociados con la esclerosis múltiple basada en datos de expresión génica de células mononucleares de sangre periférica. *PeerJ* . 2020; 8: e8357. Publicado el 3 de febrero de 2020 doi: 10.7717 / peerj.8357

22. Muthuraman M., Fleischer V., Kolber P., Luessi F., Zipp F., Groppa S. Las características de la red cerebral estructural pueden diferenciar el CIS de los primeros RRMS. *Parte delantera. Neurosci.* 2016; 10 : 14.

23. Barbour C., Kosa P., Komori M., Tanigawa M., Masvekar R., Wu T., Johnson K., Douvaras P., Fossati V., Herbst R. Diagnóstico molecular de la esclerosis múltiple y su etapa progresiva. *Ana. Neurol.* 2017; 82 : 795–812.

24. Beecham y col. (2013)Beecham AH, Patsopoulos NA, Xifara DK, Davis MF, Kemppinen A, Cotsapas C, Shah TS, Spencer C, Stand D, Goris A, Oturai A, Saarela J, Fontaine B, Hemmer B, Martin C, Zipp F, D ' Alfonso S, Martinelli-Boneschi F, Taylor B, Harbo HF, Kockum I, Hillert J, Olsson T, Ban M, Oksenberg JR, Hintzen R, Barcellos LF, Wellcome Trust Case Control Consortium 2 (WTCCC2); Consorcio Internacional de Genética de la EII (IIBDGC) Agliardi C, Alfredsson L, Alizadeh M, Anderson C, Andrews R, Søndergaard HB, Baker A, Band G, Baranzini SE, Barizzone N, Barrett J, Bellenguez C, Bergamaschi L, Bernardinelli L, Berthele A, Biberacher V, Binder TM, Blackburn H, Bomfim IL, Brambilla P, Broadley S, Brochet B, Brundin L, Buck D, Butzkueven H, Caillier SJ, Camu W, Carpentier W, Cavalla P, Celius EG, Coman I, Comi G, Corrado L, Cosemans L, Cournu-Rebeix I, Cree BA, Cusi D, Damotte V, Nicholas R, Nilsson P, Piehl F, Pirinen M, Price SE, Quach H, Reunanen M, Robberecht W, Robertson NP, Rodegher M, Rog D, Salvetti M, Schnetz-Boutaud NC, Sellebjerg F, Selter RC, Schaefer C, Shaunak S, Shen L, Shields S, Siffrin V, Slee M, Sorensen PS, Sorosina M, Sospedra M, Spurkland A, Strange A, Sundqvist E, Thijs V, Thorpe J, Ticca A, Tienari P, Van Duijn C, Visser EM, Vucic S, Westerlind H, Wiley JS, Wilkins A, Wilson JF, Winkelmann J, Zajicek J, Zindler E, Haines JL, Pericak-Vance MA, Ivinson AJ, Stewart G, Hafler D, Hauser SL, Compston A, McVean G, De Jager P, Sawcer SJ, McCauley JL. El análisis de loci relacionados con la inmunidad identifica 48 nuevas variantes de susceptibilidad a la esclerosis múltiple. Selter RC, Schaefer C, Shaunak S, Shen L, Shields S, Siffrin V, Slee M, Sorensen PS, Sorosina M, Sospedra M, Spurkland A, Strange A, Sundqvist E, Thijs V, Thorpe J, Ticca A, Tienari P , Van Duijn C, Visser EM, Vucic S, Westerlind H, Wiley JS, Wilkins A, Wilson JF, Winkelmann J, Zajicek J, Zindler E, Haines JL, Pericak-Vance MA, Ivinson AJ, Stewart G, Hafler D, Hauser SL, Compston A, McVean G, De Jager P, Sawcer SJ, McCauley JL. El análisis de loci relacionados con la inmunidad identifica 48 nuevas variantes de susceptibilidad a la esclerosis múltiple. Selter RC,

Schaefer C, Shaunak S, Shen L, Shields S, Siffrin V, Slee M, Sorensen PS, Sorosina M, Sospedra M, Spurkland A, Strange A, Sundqvist E, Thijs V, Thorpe J, Ticca A, Tienari P , Van Duijn C, Visser EM, Vucic S, Westerlind H, Wiley JS, Wilkins A, Wilson JF, Winkelmann J, Zajicek J, Zindler E, Haines JL, Pericak-Vance MA, Ivinson AJ, Stewart G, Hafler D, Hauser SL, Compston A, McVean G, De Jager P, Sawcer SJ, McCauley JL. El análisis de loci relacionados con la inmunidad identifica 48 nuevas variantes de susceptibilidad a la esclerosis múltiple. Ivinson AJ, Stewart G, Hafler D, Hauser SL, Compston A, McVean G, De Jager P, Sawcer SJ, McCauley JL. El análisis de loci relacionados con la inmunidad identifica 48 nuevas variantes de susceptibilidad a la esclerosis múltiple. Ivinson AJ, Stewart G, Hafler D, Hauser SL, Compston A, McVean G, De Jager P, Sawcer SJ, McCauley JL. El análisis de loci relacionados con la inmunidad identifica 48 nuevas variantes de susceptibilidad a la esclerosis múltiple.*Genética de la naturaleza.* 2013; 45 : 1353-1360. doi: 10.1038 / ng.2770.

25. Sawcer, Franklin y Ban (2014) Sawcer S, Franklin RJ, Ban M. Genética de la esclerosis múltiple. *Lancet Neurology.* 2014; 13 : 700–709. doi: 10.1016 / S1474-4422 (14) 70041-9.

26. Weiner (2009) Weiner HL. El desafío de la esclerosis múltiple: ¿cómo curamos una enfermedad crónica heterogénea? *Annals of Neurology.* 2009; 65 : 239–248. doi: 10.1002 / ana.21640.

27. Wiendl y Hohlfeld (2009) Wiendl H, Hohlfeld R. Terapéutica de la esclerosis múltiple: resultados inesperados que nublan los éxitos indiscutibles. *Neurología.* 2009; 72 : 1008–1015. doi: 10.1212 / 01.wnl.0000344417.42972.54.

28. Kemppinen y col. (2011) Kemppinen AK, Kaprio J, Palotie A, Saarela J. Revisión sistemática de estudios de expresión de todo el genoma en la esclerosis múltiple. *BMJ Open.* 2011; 1 : e000053.

29. Zhang y col. (2011) Zhang F, Shi Y, Wang L, Sriram S. Papel de HDAC3 en la expresión de p53 y apoptosis en células T de pacientes con esclerosis múltiple. *MÁS UNO.* 2011; 6 : e16795. doi: 10.1371 / journal.pone.0016795.

30. Gilli y col. (2010) Gilli F, Lindberg RL, Valentino P, Marnetto F, Malucchi S, Sala A, Capobianco M, Di Sapio A, Sperli F, Kappos L, Calogero RA, Bertolotto A. Aprender de la naturaleza: el embarazo cambia la expresión de la inflamación genes relacionados en pacientes con esclerosis múltiple. *MÁS UNO.* 2010; 5 : e8962. doi: 10.1371 / journal.pone.0008962.

31. Christophi y col. (2008) Christophi GP, Hudson CA, Gruber RC, Christophi CP, Mihai C, Mejico LJ, Jubelt B, Massa PT. Deficiencia de SHP-1 y aumento de la expresión de genes inflamatorios en PBMC de pacientes con esclerosis múltiple. *Investigación de laboratorio.* 2008; 88 : 243-255. doi: 10.1038 / labinvest.3700720.

32. Flores y col. (2003) Flores N, Duran C, Blasco MR, Puerta C, Dorado B, Garca-Merino A, Ballester S. NFkappaB y actividad de unión al ADN AP-1 en pacientes con esclerosis múltiple. *Revista de neuroinmunología.* 2003; 135 : 141-147. doi: 10.1016 / S0165-5728 (02) 00440-X.

33. Gurevich M., Miron G., Achiron A. Optimización del diagnóstico de esclerosis múltiple: expresión génica y asociación genómica. *Ana. Clin. Transl. Neurol.* 2015; 2 : 271-277.

10. Anexos

Codigo4clases.rmd. Código en R de los procesos explicados en este trabajo utilizando las 4 clases de EM (CIS,PP,RR,SP).

Codigo6clases.rmd. Código en R de los procesos explicados en este trabajo utilizando 6 clases: HC, OND, CIS,PP,RR,SP.

Codigo4clases.html. Código y resultados en HTML de los procesos explicados en este trabajo utilizando las 4 clases de EM (CIS,PP,RR,SP).

Codigo6clases.html. Código y resultados en HTML de los procesos explicados en este trabajo utilizando 6 clases: HC, OND, CIS,PP,RR,SP.

Figs2.Figuras obtenidas en R.

Todos los anexos se encuentran disponibles en https://github.com/gititub/TFM