Análisis de técnicas de clasificación de perfiles taxonómicos para predecir trastornos de la enfermedad inflamatoria intestinal.

Eva Castillo Rosa
Máster Bioinformática y Bioestadística
Área 1**.** Subárea 7: Desarrollo de herramientas de soporte a la ómica

Antonio Jesús Adsuar Gómez
Javier Luis Cánovas Izquierdo y Marc Maceira Duch

23/06/2020

B) GNU Free Documentation License (GNU FDL)

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Análisis de técnicas de clasificación de perfiles taxonómicos para predecir trastornos de la enfermedad inflamatoria intestinal.*
Nombre del autor:	*Eva Castillo Rosa*
Nombre del consultor/a:	*Antonio Jesús Adsuar Gómez*
Nombre del PRA:	*Javier Luis Cánovas Izquierdo y Marc Maceira Duch*
Fecha de entrega (mm/aaaa):	06/2020
Titulación:	*Máster Bioinformática y Bioestadística*
Área del Trabajo Final:	*Desarrollo de herramientas de soporte a la ómica*
Idioma del trabajo:	*Castellano*
Palabras clave	*microbiota, machine learning, shiny*

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

La enfermedad inflamatoria intestinal engloba una serie de desórdenes con síntomas muy similares. Por ello, saber qué bacterias están presentes en la microbiota de los pacientes con este tipo de trastornos es útil para su diagnóstico y tratamiento. Para poder estudiar la relación subyacente de los microorganismos que habitan en la microbiota con el fenotipo de la enfermedad se han utilizado técnicas de *machine learning*. Es crucial un buen estudio de los diferentes algoritmos de clasificación disponibles para poder aplicar aquellos óptimos al descubrimiento de marcadores específicos de la enfermedad o, en última instancia, al diagnóstico clínico.

En este trabajo se ha analizado la diversidad microbiana de muestras de biopsia de pacientes sanos, con enfermedad de Crohn o colitis ulcerosa usando el software QIIME 2^{TM}. A partir de los datos de abundancia relativa bacteriana se han aplicado diversas metodologías de aprendizaje supervisado para la clasificación de las muestras. Finalmente, se ha desarrollado una aplicación web interactiva para adaptar los modelos óptimos a los datos de entrada del usuario.

Aunque hay modelos lineales que muestran un rendimiento parecido a otros más complejos, el modelo que mayor capacidad predictiva tiene es random forest. Por otra parte, la elección de un buen método de reducción de la dimensionalidad es muy importante a la hora de aplicar *machine learning* sobre

datos de microbioma. Al igual que es crucial facilitar la disponibilidad de estos análisis a toda la comunidad científica, que permitan hacer estudios a mayor escala.

Abstract (in English, 250 words or less):

The inflammatory bowel disease comprises a wide range of disorders with similar symptoms. Therefore, studying the bacteria present in the microbiota of patients is key for the diagnosis and treatment of these diseases. A thorough study of different available classification algorithms is crucial to find the optimal ones and apply it to the discovery of biomarkers or, ultimately, clinical diagnosis.

In this study, the microbial diversity of biopsy samples from healthy, Crohn's disease or ulcerative colitis patients was analysed with QIIME 2™ software. Various supervised machine learning methods have been applied from bacterial relative abundance data to sample classification. Finally, an interactive web application has been developed in order to adapt the optimal models to the user's input data.

Although some linear models show similar performance to complex ones, the model with the highest performance is random forest. Besides, choosing a good dimensionality reduction method is important when applying machine learning on microbiome data. Just as crucial as making these analyses available to the entire scientific community, so that large-scale studies can be done.

Índice

1. Introducción..... 3
1.1 Contexto y justificación del Trabajo..... 3
1.2 Objetivos del Trabajo..... 4
1.3 Enfoque y método seguido..... 5
1.4 Planificación del Trabajo 5
1.5 Breve sumario de productos obtenidos 8
1.6 Breve descripción de los otros capítulos de la memoria..... 9
2. Análisis de la diversidad microbiana de muestras del 16S ARNr de pacientes con IBD. 10
2.1. Obtención de datos del repositorio europeo ENA. 10
2.2. Control de calidad y filtrado de lecturas..... 10
2.3. Obtención de una tabla de SVs. 13
2.4. Análisis de la α y β diversidad de las muestras. 13
2.5. Asignación de taxonomía a los SVs producidos. 15
2.6. Visualización de los resultados del análisis. 15
3. Estudio de técnicas de clasificación supervisada de los perfiles microbianos de pacientes sanos y con dos condiciones de IBD. 17
3.1. Estructuración de los datos para su uso en *machine learning*. 18
3.2. Creación de los conjuntos de entrenamiento y prueba. 18
3.3. Entrenamiento de los diferentes modelos. 18
3.4. Evaluación del rendimiento de los modelos con el conjunto de prueba. 20
3.5. Optimización de los modelos..... 21
3.6. Aplicación de ingeniería de características para reducir la dimensionalidad de los datos. 30
4. Desarrollo de una aplicación interactiva que engloba los modelos más relevantes...... 35
5. Conclusiones..... 39
6. Glosario..... 42
7. Bibliografía..... 43
8. Anexos 47
8.1. Anexo I. 47

Lista de figuras

Figura 1. Diagrama de Gantt. 7
Figura 2. Análisis de calidad de las lecturas forward de la muestra SRR6505433. 11
Figura 3. Boxplot del número de secuencias retenidas en cada paso del filtrado. 12
Figura 4. Gráfico de rarefacción. 14
Figura 5. PCoA basado en la distancia de Bray-Curtis. 14
Figura 6. Distribución taxonómica de las muestras agrupadas a nivel de Especie. 16
Figura 7. Gráfico de burbujas de los 30 géneros más abundantes en las muestras, agrupadas por clases. 17
Figura 8. Mejores valores de lambda tras cada iteración. 22
Figura 9. Error de clasificación para cada valor de k y de técnica de ponderación. 23
Figura 10. Tasa de error de clasificación para cada valor de *minsplit* y *cp*. 24
Figura 11. Tasa de error OOB total en función del número de árboles. 25
Figura 12. Tasa de error OOB total en función del valor de *mtry*. 25
Figura 13. Tasa de error en función del parámetro C. 26
Figura 14. Tasa de error en función del parámetro C y gamma. 27
Figura 15. SSE en función del número de capas y nodos de la red neuronal. 27
Figura 16. Tasa de error de clasificación en función de los valores de *eta* y *max_depth*. 28
Figura 17. Comparación de los valores de AUC y precisión de los distintos modelos predictivos. 29
Figura 18. Comparación de los valores de AUC antes y después del remuestreo. 30
Figura 19. Comparación de los valores de AUC y precisión de los distintos modelos predictivos tras aplicar HFE. 31
Figura 20. Comparación de los valores de AUC y precisión de los distintos modelos predictivos tras aplicar Boruta. 32
Figura 21. Comparación de los valores de AUC y precisión de los distintos modelos predictivos tras aplicar *forward selection*. 33
Figura 22. Boxplot comparando los valores de AUC con diferentes métodos de selección de variables. 34
Figura 23. Comparación del tiempo de ajuste de los modelos según el método de selección de variables. 35
Figura 24. Captura de pantalla de la apariencia de la aplicación tras introducir los datos de entrada. 36
Figura 25. Captura de pantalla de la apariencia de la aplicación en la división de datos con validación cruzada de Monte Carlo. 37
Figura 26. Captura de pantalla de la apariencia de la aplicación tras el tuneado y entrenamiento de un modelo de random forest. 37
Figura 27. Captura de pantalla de la apariencia de la aplicación tras el tuneado y entrenamiento de un modelo de SVM radial. 38

Lista de tablas

Tabla 1. Valores promedio de kappa, precisión y AUC para todos los modelos. 34

1. Introducción

1.1 Contexto y justificación del Trabajo

La enfermedad inflamatoria intestinal (IBD) engloba un grupo de desórdenes crónicos del tracto gastrointestinal. Las dos manifestaciones más comunes de la IBD son la enfermedad de Crohn (CD) y la colitis ulcerosa (UC). La enfermedad de Crohn afecta en mayor medida al íleon terminal y otras zonas del tubo digestivo; mientras que la colitis ulcerosa afecta exclusivamente al colon. A pesar de ello, los síntomas de estas dos enfermedades son, a menudo, muy similares y pueden llegar a confundirse.

Se ha relacionado el origen de la IBD con una predisposición genética, factores externos como la dieta o el estrés, la respuesta inmune del huésped o la microbiota intestinal. Aunque la etiología sigue siendo a día de hoy desconocida, el papel que juega la microbiota es ampliamente conocido y estudiado [1]. Muchos estudios se han centrado en buscar la relación de la respuesta inmunitaria y el microbioma de los pacientes durante el paso de la enfermedad de la inflamación intestinal fisiológica a patológica [2]. Dado el potencial de la modulación de la microbiota como tratamiento de esta enfermedad, es necesario conocer más en profundidad la comunidad microbiana característica de la enfermedad de Crohn y de la colitis ulcerosa.

El desarrollo de técnicas de secuenciación de segunda generación ha abierto la puerta al estudio profundo de las poblaciones microbianas en diferentes ambientes. Uno de los marcadores más utilizados es el gen 16S, que representa más del 80% del ARN bacteriano total. Su estructura, compuesta por regiones conservadas y variables, permite su amplificación y la posterior clasificación taxonómica. Estudiar la relación de los microorganismos entre si y con el ambiente que les rodea dentro del hospedador puede permitir descubrir el nexo con el fenotipo del individuo [3]. Por ello, conocer las bacterias presentes en la microbiota de un paciente puede ser una herramienta muy útil para descubrir biomarcadores de la enfermedad.

En los últimos años y, con el auge de técnicas de aprendizaje supervisado, se han publicado diversos estudios para predecir las características del hospedador basadas en patrones de microbioma [4]. Se han utilizado modelos de *machine learning* para el conocimiento en detalle de enfermedades como la cirrosis hepática, la obesidad o la diabetes tipo II. Este tipo de métodos aplicados a datos de metagenómica todavía presentan muchos problemas y obstáculos que resolver, por lo que su uso en diagnóstico clínico es aún un desafío [5].

Para tratar de solventar algunas de estas dificultades, en este trabajo se han analizado diferentes técnicas de clasificación supervisada sobre perfiles taxonómicos de pacientes con diferentes presentaciones de la enfermedad inflamatoria intestinal. Inicialmente se obtuvieron las lecturas del estudio PRJNA431126 disponible en el *European Nucleotide Archive* (ENA) [6]. En

dicho estudio se tomaron diferentes muestras (fecales y de mucosa) y se secuenció la región V4 del ARN ribosomal (ARNr) 16S [7]. Se ha procedido al análisis de la diversidad microbiana de estas muestras con el software QIIME 2™ [8]. Sobre la tabla de variantes de secuencia (SV) obtenida, se aplicaron diversos algoritmos de *machine learning* para predecir los diferentes trastornos de la enfermedad inflamatoria intestinal, basándose en las características del microbioma. Por último y, con el fin de hacer más accesibles estas técnicas a la comunidad científica, se ha desarrollado una aplicación web interactiva para la aplicación de los modelos con mejor rendimiento.

1.2 Objetivos del Trabajo

Los tres objetivos generales que se presentan en este trabajo son:

- Analizar la diversidad microbiana de las muestras del 16S ARNr.
- Estudiar diversas técnicas de clasificación supervisada de los perfiles microbianos en las dos condiciones de la enfermedad inflamatoria intestinal.
- Desarrollar una aplicación interactiva con Shiny.

Para la consecución de los objetivos generales planteados previamente, se exponen los siguientes objetivos específicos:

1. Analizar la diversidad microbiana de las muestras del 16S ARNr.

    - Obtener los datos del repositorio europeo ENA.
    - Realizar un control de calidad y filtrado de lecturas.
    - Obtener una tabla de SVs.
    - Proceder al análisis de la alfa y beta diversidad de las muestras.
    - Asignar taxonomía a los SVs producidos.
    - Visualizar los resultados del análisis.

2. Probar diferentes algoritmos de clasificación supervisada.

    - Explorar y estructurar los datos para su uso en *machine learning*.
    - Crear un conjunto de datos para entrenamiento y otro para prueba.
    - Entrenar los diferentes modelos con los datos de entrenamiento.
    - Evaluar el rendimiento de los modelos en el conjunto de datos de prueba.
    - Ajustar parámetros para mejorar los modelos y repetir los dos pasos anteriores.

3. Desarrollar una aplicación interactiva con Shiny.

    - Elegir el modelo o modelos de interés para la aplicación.
    - Integrar herramientas de visualización de datos.
    - Incorporar la personalización de parámetros para el desarrollo de modelos en la aplicación web.

1.3 Enfoque y método seguido

Se accedió al ENA para obtener las lecturas crudas procedentes de la secuenciación y así realizar el análisis taxonómico desde el inicio, controlando todo el proceso.

Aunque existen varios protocolos para el análisis de muestras microbianas procedentes de la amplificación del 16S ARNr, se optó por utilizar QIIME 2™. La mayoría de las herramientas bioinformáticas no difieren en la asignación taxonómica de las muestras en categorías taxonómicas superiores. Es a nivel de especie o Unidad Taxonómica Operativa (OTU) donde los resultados se diferencian, debido no solo al tipo de software utilizado sino a la base de datos elegida. Una de las ventajas de utilizar las herramientas englobadas en la plataforma QIIME 2™ es un aumento en la resolución, reflejada en la definición de SVs [9]. Otros métodos agrupan las lecturas de secuenciación por encima de un umbral de similitud, por ejemplo, el 97%. Mientras que la aplicación DADA2 [10] dentro de QIIME 2™ permite determinar SVs, discriminando las secuencias hasta en un solo nucleótido de diferencia.

Se consultó la bibliografía disponible y se seleccionaron los modelos más aptos para este tipo de datos, según su rendimiento, robustez en la generalización del modelo, complejidad y tiempo de ejecución. Tras este filtro inicial, se escogieron los algoritmos de interés dependiendo de los resultados obtenidos tras su aplicación e implementación. Para mejorar el rendimiento de los modelos se optimizaron hiperparámetros y se realizaron diferentes métodos de reducción de la dimensionalidad. Todos los modelos se desarrollaron en lenguaje de programación R y se utilizaron tanto paquetes específicos para cada solución como el paquete *caret* [11], que engloba herramientas para la implementación de modelos.

Después se seleccionaron varios modelos para su integración en una aplicación web. Dicha aplicación se creó con Shiny [12], un paquete de R que facilita la creación de aplicaciones interactivas con este mismo lenguaje. Esta herramienta se diseñó para una fácil visualización de los datos procedentes de una tabla de OTUs o SVs y para la aplicación de varios modelos, con la posibilidad de modificar parámetros para mejorar su rendimiento.

1.4 Planificación del Trabajo

En este apartado se concretan las tareas a realizar, el calendario definido para lograrlas, así como los hitos del proyecto. Finalmente, se evalúan los riesgos que pudiesen surgir.

A continuación, se exponen y explican las tareas para lograr los objetivos previamente definidos.

a) Análisis de la diversidad microbiana de las muestras del 16S ARNr.

- Búsqueda exhaustiva en ENA de posibles proyectos de secuenciación de amplicones de 16S, que tengan disponible toda la información referente al grupo al que pertenecen las muestras.
- Descarga de lecturas y control de calidad inicial de las mismas.
- Aprendizaje y familiarización del software de análisis QIIME 2™ y su funcionamiento.
- Procesado de lecturas dentro del software, con los siguientes pasos: filtrado y creación de tabla de SVs, generación de un árbol filogenético, análisis de diversidad (alfa y beta), asignación taxonómica de los SVs.
- Obtención de todas las figuras y tablas de resultados del análisis, para lo que es necesario la conversión entre formatos.

b) Redacción de la PEC2: Desarrollo del trabajo. Fase 1.

- Definición de los objetivos logrados y de los resultados obtenidos hasta el momento.

c) Estudio de diferentes algoritmos de clasificación supervisada.

- Búsqueda bibliográfica de los algoritmos de clasificación existentes y elección de aquellos que mejor se ajustan a los datos.
- Recolección y preparación de los datos en el formato necesario para el posterior análisis.
- Preparación del conjunto de entrenamiento y el conjunto de prueba, de forma equitativa y balanceada.
- Entrenamiento del modelo.
- Predicciones sobre el conjunto de prueba para evaluar el funcionamiento del modelo.
- Mejora del modelo, ajustando parámetros o utilizando paquetes diferentes. Repetición del paso anterior hasta conseguir el mejor modelo posible sin sobreajuste.
- Aplicación de ingeniería de características para reducir la dimensionalidad de los datos.

d) Redacción de la PEC3: Desarrollo del trabajo. Fase 2.

- Definición de los objetivos logrados, los resultados obtenidos y las posibles modificaciones sobre el plan inicial de tareas y objetivos.
- Descripción de los problemas encontrados durante el estudio de los diferentes algoritmos.

e) Desarrollo de una aplicación web con Shiny.

- Realización de un tutorial para conocer el funcionamiento del paquete.
- Diseño de una interfaz de usuario.
- Creación de la estructura y las características básicas de la aplicación.
- Establecimiento de los datos de entrada y los posibles datos de salida.
- Implementación de funciones y widgets dentro de la aplicación.
- Publicación de la aplicación para que esté disponible para toda la comunidad.

f) Redacción de la PEC4: Cierre de la memoria.

- Composición de los resultados obtenidos, agrupando la información de las PEC2 y 3.
- Redacción de conclusiones y glosario.
- Corrección general ortográfica y de erratas, tanto en el texto como en el formato de la bibliografía.
- Incorporación de anexos al escrito.

g) Realización de la PEC5a: Elaboración de la presentación.

- Elaboración de un esquema con los contenidos para la presentación.
- Desarrollo de diapositivas que incluyan gráficos, figuras y otras herramientas para la clara exposición de los resultados del trabajo.

h) Realización de la PEC5b: Defensa pública del TFM.

- Revisión de la memoria y la presentación.
- Preparación para las preguntas presentadas por el tribunal.

El calendario en el que se engloban las tareas, el tiempo dedicado a cada una y las entregas a realizar se ve reflejado en la Figura 1 con un diagrama de Gantt.

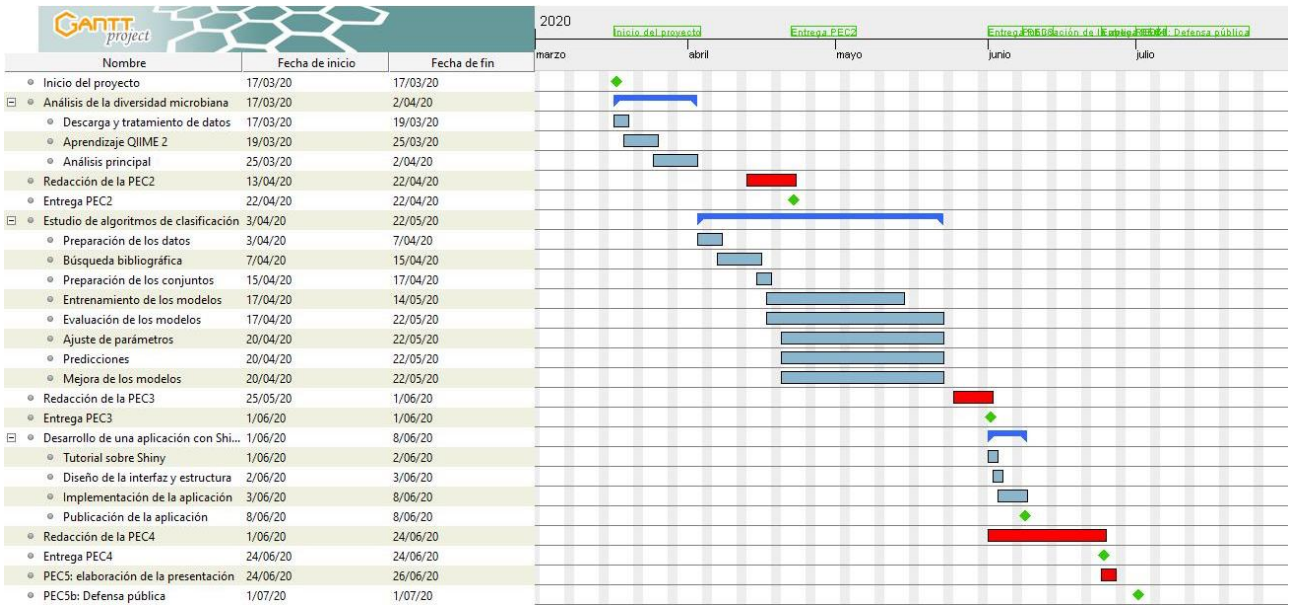

Figura 1. Diagrama de Gantt.

Las tareas principales se indican en azul oscuro, con tareas específicas englobadas en azul más claro. Los hitos se marcan en verde. Las redacciones y entregas de las diferentes PEC se señalan en rojo.

Los hitos del proyecto quedan establecidos por las fechas de entrega de las diferentes PEC y por los objetivos y tareas marcadas con anterioridad:

- Inicio del proyecto: 17 de marzo
- Análisis completo de la diversidad microbiana: 13 de abril

- Entrega PEC2: 22 de abril
- Producción de modelos de clasificación: 22 de mayo
- Entrega PEC3: 01 de junio
- Publicación de la aplicación con Shiny: 08 de junio
- Entrega PEC4: 24 de junio
- Entrega PEC5a: 28 de junio
- PEC5b, defensa pública: del 01 al 08 de julio

En este trabajo existe una mayor probabilidad de que factores de riesgo se concentren en el análisis de diferentes algoritmos de clasificación supervisada:

- Problemas con la dimensionalidad de los datos. Pero existen muchos algoritmos para superar esa barrera.
- Insuficiencia de poder computacional para el entrenamiento de modelos complejos. Se podría utilizar alguna plataforma de computación en la nube o acceder a algún nodo de computación como el del Consejo Superior de Investigaciones Científicas (CSIC).
- Falta de datos o sobreajuste de los modelos. Existe la posibilidad de acudir a técnicas de aumento de datos o de validación cruzada.
- Desequilibrio en las clases de las muestras. Se pueden aplicar técnicas que implican el remuestreo.
- Falta de tiempo para probar una gran cantidad de algoritmos. En tal caso, se priorizarán aquellos que han dado buen resultado según la bibliografía.
- Dificultad a la hora de entender algoritmos complejos e interpretar los resultados. Se puede acudir a libros de *machine learning* y a la ayuda del consultor del trabajo.
- Tiempo requerido para aprender a utilizar la herramienta de Shiny para la creación de una web interactiva. Hay muchos tutoriales y material de aprendizaje a disposición. En caso de no poder añadir muchas funciones y widget se diseñaría una interfaz más básica centrándose más en el contenido que en la apariencia de la web.

1.5 Breve sumario de productos obtenidos

Se presentan los productos obtenidos de la consecución de este trabajo:

- Plan de trabajo en el que se proponen los objetivos y la planificación concreta, teniendo en cuenta posibles problemas que puedan surgir.
- Memoria escrita final durante la PEC4, donde se engloban las redacciones entregadas en las PEC2 y 3.
- Productos obtenidos:

    1. Figuras y gráficos como resultado del análisis de la diversidad microbiana de las muestras.
    2. Un repositorio con el código R para implementar y entrenar los modelos analizados, así como figuras comparativas.
    3. Aplicación web para poder utilizar diferentes modelos de clasificación supervisada.

- Presentación visual de diapositivas en la que se plasmen con claridad los resultados y conclusiones derivados del trabajo.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria final del trabajo consta de los siguientes capítulos:

- Introducción: se centra en una revisión bibliográfica y del estado del arte tanto de la enfermedad inflamatoria intestinal y su relación con la disbiosis como de los diferentes algoritmos de aprendizaje supervisado.
- Resto de capítulos: se redactan tres capítulos acordes a los objetivos principales del trabajo, cada uno con subapartados.

    1. Análisis de la diversidad microbiana de muestras del 16S ARNr de pacientes con IBD. Se desglosan los pasos llevados a cabo para el análisis de la diversidad microbiana, de las lecturas crudas a la producción de figuras y resultados interpretables.
    2. Estudio de técnicas de clasificación supervisada de los perfiles microbianos de pacientes sanos y con dos condiciones de IBD. Se detallan los algoritmos elegidos y los pasos seguidos hasta la producción de modelos de clasificación óptimos. Todo el código utilizado para ello se ha almacenado en un repositorio.
    3. Desarrollo de una aplicación interactiva que engloba los modelos más relevantes. Se describe el desarrollo de una aplicación en Shiny y se muestra su apariencia y funcionamiento.

- Conclusiones: se plantean las conclusiones del trabajo con respecto a los objetivos propuestos. Se proponen futuras líneas de investigación.
- Glosario: se describen los términos y acrónimos utilizados a lo largo del trabajo.
- Bibliografía: se plasma la información de la bibliografía utilizada para el trabajo.
- Anexos: se incluye código utilizado durante la realización del trabajo.

2. Análisis de la diversidad microbiana de muestras del 16S ARNr de pacientes con IBD.

Seguidamente se exponen los subapartados para analizar la diversidad microbiana de las muestras del ARNr 16S.

2.1. Obtención de datos del repositorio europeo ENA.

Se accedió al estudio con identificador PRJNA431126 dentro del portal del ENA, con descripción: “Redes de interacción dieta-microbiota-metabolito revelan los actores claves en la enfermedad inflamatoria intestinal”. Se descargaron los 324 conjuntos de lecturas y el informe en el que se detalla el identificador y el alias de cada una de las muestras. De las 324 muestras, 113 pertenecen a muestras fecales y 211 a muestras de la mucosa intestinal tomadas a través de biopsia. Para intentar eliminar el sesgo añadido de la procedencia de las muestras, se decidió analizar únicamente un grupo de muestras. Se escogieron las muestras de biopsia al ser más abundantes y dado que, según apunta el estudio de Weng *et al*. [7], la α-diversidad o riqueza de especies dentro de la comunidad es mayor.

2.2. Control de calidad y filtrado de lecturas.

El paso principal para cualquier análisis de datos de secuenciación masiva es hacer un control de calidad de las lecturas. El programa más utilizado en el campo de la bioinformática es FastQC [13], una herramienta que permite la visualización de diferentes medidas de control de calidad en datos crudos de secuenciación masiva. En la Figura 2 se puede ver un ejemplo de los gráficos de calidad más representativos para una de las muestras.

En la parte superior izquierda de la Figura 2 se observa que la calidad por posición (codificada en escala Phred+*33*) mantiene un buen estándar en todos los ciclos con una calidad de base mediana superior a 28 (límite verde). En la parte superior derecha se muestra la distribución del contenido de bases. El patrón observado es muy variante sobre todo en las primeras 12-13 bases, que implica una falta de precisión al inicio de la secuenciación. Por último, la parte inferior de la Figura presenta un ligero desacuerdo de la distribución de G+C real con respecto a la teórica. Este tipo de comportamientos son esperables al tratarse de distribuciones no teóricas.

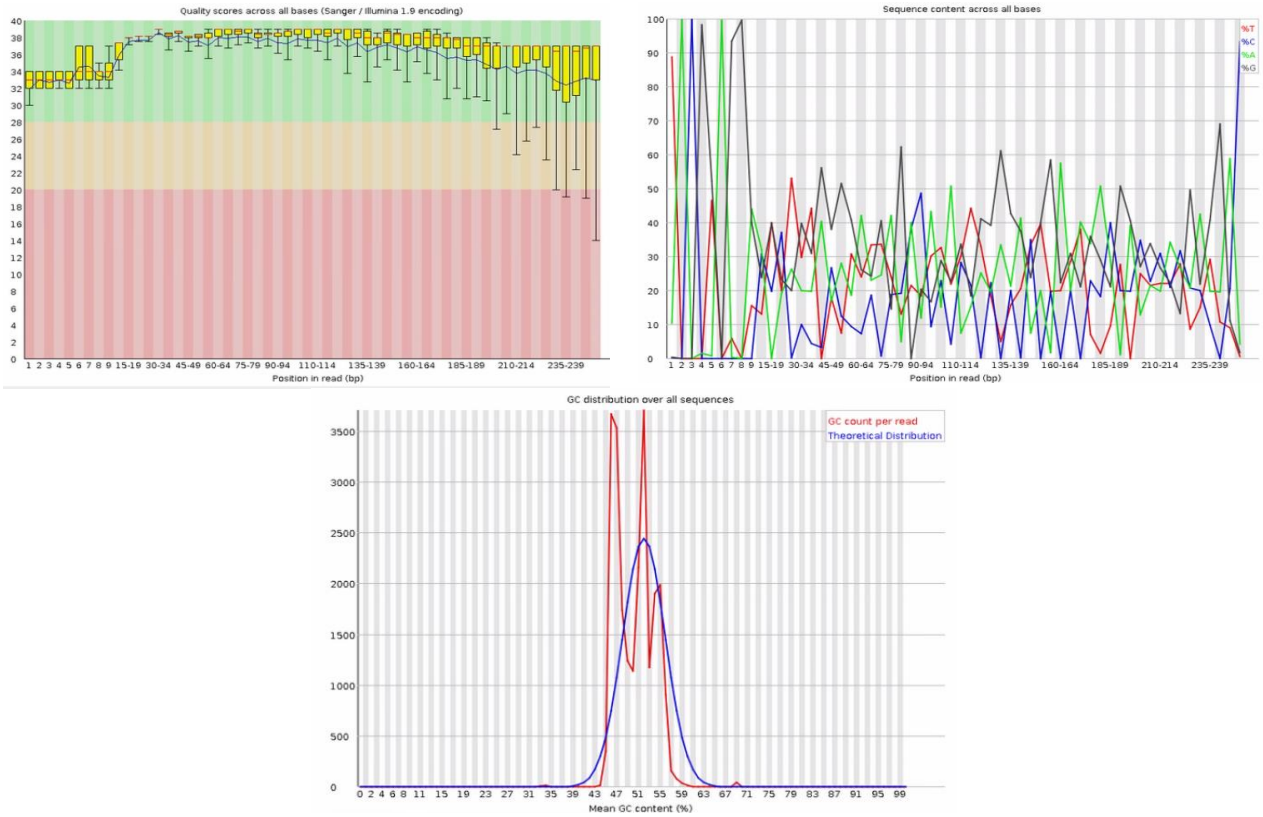

Figura 2. Análisis de calidad de las lecturas forward de la muestra SRR6505433.
En la parte superior izquierda se muestra la calidad por base y a la derecha el contenido de bases por posición. En la parte inferior se indica la distribución G+C.

Tras observar una buena calidad en las lecturas, el siguiente paso es la unión de las lecturas pareadas en un único archivo y la eliminación de los cebadores utilizados para la amplificación de la región del 16S ARNr. Existen muchos software capaces de unir las lecturas pareadas, que corrigen errores en la región solapante y eliminan las secuencias de mala calidad o que no han alineado correctamente entre sí. Para hacer todo el proceso más estandarizado y reducir al máximo los sesgos en el análisis, se van a eliminar los cebadores una vez se hayan importado las secuencias a la plataforma QIIME 2™, especializada en el análisis de datos microbianos de secuenciación masiva.

QIIME 2™ es un paquete bioinformático de código abierto para analizar microbiomas a partir de datos crudos de secuenciación masiva. QIIME 2™ incluye el filtrado de calidad de las lecturas, la selección de variantes de secuencia, la asignación taxonómica, la reconstrucción filogenética y el análisis de diversidad, generando como resultado gráficos y figuras de alta calidad.

El primer paso para un análisis lo más completo posible es el diseño de un archivo de metadatos. Este archivo separado por tabuladores va a contener toda la información importante sobre las muestras, como su procedencia, una descripción, un identificador, etc.

El siguiente paso consiste en importar las lecturas dentro de los archivos propios de la plataforma de QIIME 2™, que se denominan *artifact* (*.qza). Este tipo de ficheros contienen no solo los datos sino información acerca del tipo de datos y de su procedencia, para poder trazar el origen de todos los archivos

generados durante el análisis. Para ello hay que crear un archivo con el nombre de las muestras y la ruta específica donde están situadas (`manifest.csv`). Las líneas de comando utilizadas durante este y todos los pasos del análisis con QIIME 2™ se especifican en el Anexo I.

Tras este primer paso, todos los ficheros de lecturas quedan agrupados en el archivo `paired_end_reads.qza`. Posteriormente y, en un único paso, se eliminaron los cebadores, se filtró por calidad, se unieron las lecturas pareadas, se eliminó ruido y se filtraron las secuencias quimera. Dado que en el artículo de Weng *et al*. no se indican los cebadores utilizados, se eliminaron por longitud. En el informe obtenido con FastQC hay un apartado que permite ver si hay secuencias que están sobrerrepresentadas en las lecturas. Tanto en las lecturas *forward* como *reverse*, las secuencias sobrerrepresentadas, variables entre sí, tienen una longitud de 50 pb.
Para el proceso de filtrado se utiliza la aplicación DADA2 [14] de QIIME 2™, que permite la corrección de errores de amplicones secuenciados por Illumina y resuelve diferencias de tan solo un nucleótido entre secuencias (variantes de secuencia).

A partir de estos archivos ".qza" se generan otros con extensión ".qzv" que permiten visualizar los resultados con un simple comando. O bien subirlos a https://view.qiime2.org/, obteniendo de ambos modos una visualización interactiva.

En la Figura 3 se puede observar un boxplot de las secuencias retenidas tras cada paso del filtrado. Se puede ver como el número de secuencias mediano va disminuyendo con cada paso del filtrado. El porcentaje de secuencias que quedan para el estudio final son entre un 78 y un 93% del total de secuencias iniciales.

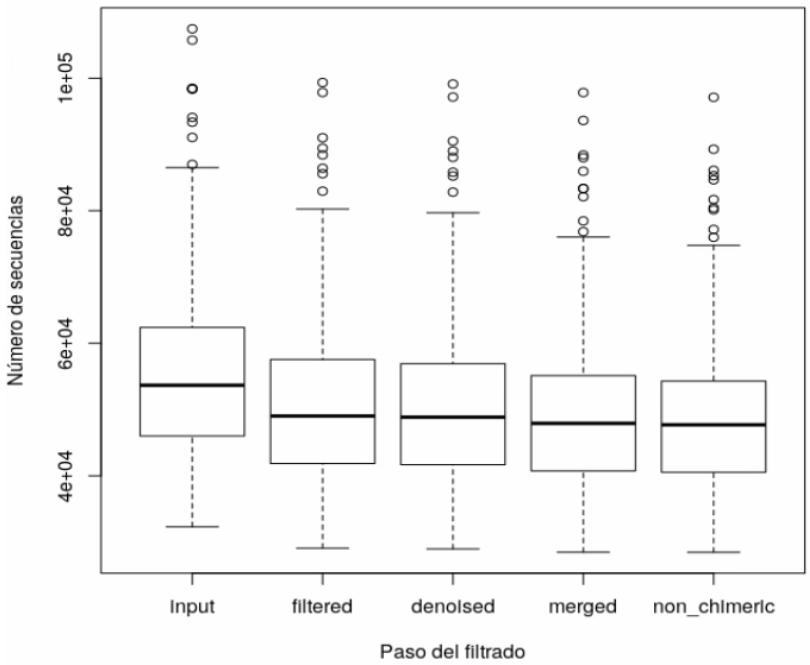

Figura 3. Boxplot del número de secuencias retenidas en cada paso del filtrado.

2.3. Obtención de una tabla de SVs.

Durante este último paso, gracias a la aplicación DADA2, se genera el archivo `feature_table.qza` que contiene las SVs y su frecuencia en las muestras. La modificación de QIIME 2™ sobre su versión anterior incluye la determinación de los SVs, con una resolución más alta que los OTUs. Se obtienen también las secuencias representativas de cada SV, en el archivo `rep-seqs.qza`. Se observaron un número total de 8493 SVs del análisis de 211 muestras.
La tabla producida contiene en número absoluto la frecuencia de los SVs en cada una de las muestras. Para trabajar posteriormente de forma más sencilla se generó una tabla con las frecuencias relativas para cada muestra.

2.4. Análisis de la α y β diversidad de las muestras.

QIIME 2™ es capaz de generar diferentes métricas de diversidad filogenética. Para ello necesita de la creación de un árbol filogenético con raíz en el que los SVs se relacionen entre sí. QIIME 2™ consta de una serie de parámetros condensados en una única línea de comandos. Primero se hace un alineamiento múltiple de las secuencias de los SVs, con el software MAFFT [15]. Este alineamiento se filtra para eliminar posiciones altamente variables que aportan ruido al árbol filogenético. Finalmente, utilizando el programa FastTree [16] se crea un árbol sin raíz al que posteriormente se le asigna una raíz en el punto medio de la distancia más larga de punta a punta del árbol.

Tras construir el árbol, se pueden calcular las medidas de α y β diversidad. Con la α-diversidad se pueden determinar cuántas y cómo de diferentes son las especies que se detectan en un ambiente. Mientras que la β-diversidad nos indica la diferencia entre comunidades microbianas de distintas muestras. Con un único comando, se producen diferentes medidas de la diversidad y gráficos de coordenadas principales (PCoA).

Dado que no se ha querido descartar ninguna muestra para los análisis de diversidad, se utiliza como límite el menor valor de profundidad de muestra. Los resultados del análisis quedan guardados, de forma automática, en una carpeta denominada `core-metrics-results`.

Con otro comando diferente, se calcula la curva de rarefacción. En este caso se toma como límite máximo el mayor valor de profundidad de muestra. La Figura 4 muestra el gráfico de rarefacción obtenido.

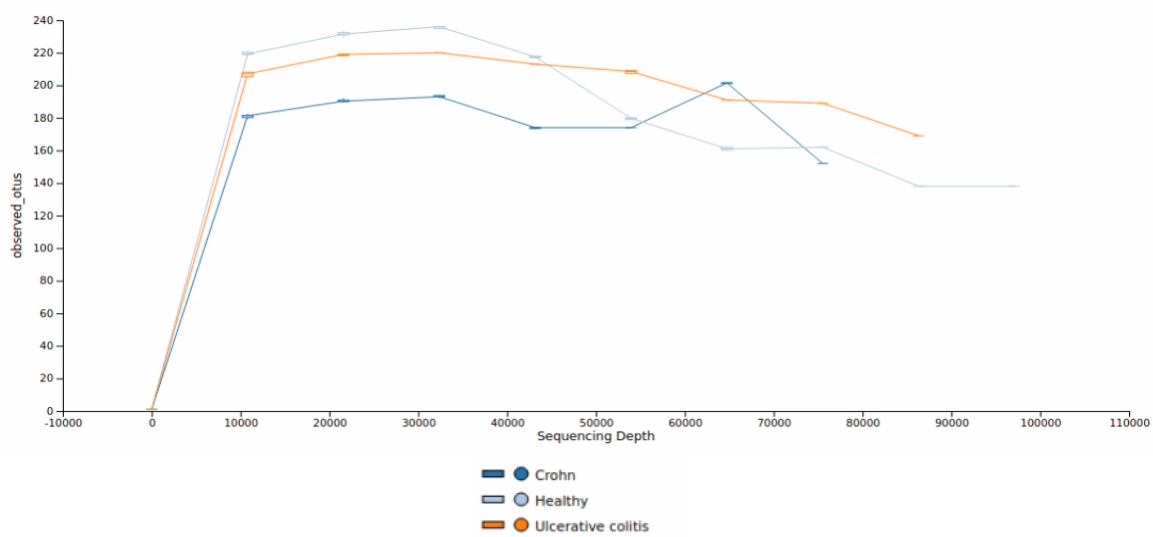

Figura 4. Gráfico de rarefacción.
SVs observados frente a la profundidad de secuenciación de las muestras, agrupadas por clase.

En la Figura 4 se puede ver que todos los grupos han llegado a saturación en cuanto a SVs observados, indicando que el muestreo ha sido el adecuado para estimar la riqueza total de las muestras. Aunque se puede observar cierta fluctuación una vez se ha llegado a saturación, provocada por una diferencia en los SVs observados entre muestras del mismo grupo.

En cuanto a las medidas de β-diversidad, se han utilizado diferentes métodos para generar gráficos interactivos de PCoA, incluyendo la distancia de Bray-Curtis, Jaccard o la distancia UniFrac. En la Figura 5 se puede ver un gráfico de PCoA con Emperor [17] utilizando la distancia de Bray-Curtis, una medida más sencilla que el resto ya que no tiene en cuenta las relaciones filogenéticas entre las muestras.

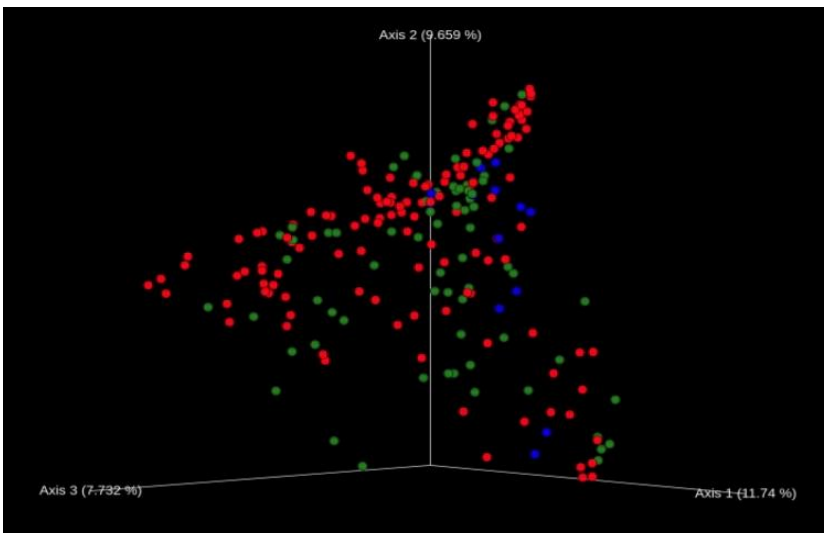

Figura 5. PCoA basado en la distancia de Bray-Curtis.
Las muestras rojas se corresponden con la clase Crohn, las azules con las sanas y las verdes con muestras de colitis ulcerosa.

Debido a la alta dimensionalidad y a la complejidad de los datos, no es evidente una asociación de las muestras según su grupo. Además, los tres ejes de coordenadas principales explican, en total, aproximadamente un 29% de la varianza presente en las muestras, por lo que las relaciones que pueda haber entre ellas no se observan a este nivel. Por ello es tan importante el uso del aprendizaje automático, que permite descubrir relaciones implícitas entre las muestras.

2.5. Asignación de taxonomía a los SVs producidos.

En este paso se explora la composición taxonómica de las muestras. Inicialmente, hay que disponer de una base de datos que albergue información de secuencias del 16S ARNr y su taxonomía para poder compararla. Se utilizó Greengenes [18], una base de datos del ARNr 16S que proporciona clasificación filogenética para las secuencias depositadas en GenBank [19], y cuyo uso es compatible con QIIME 2™.

Se importaron tanto las secuencias de referencia, agrupadas al 97% de identidad, como la taxonomía asociada en el formato requerido por QIIME 2™.

Los SVs detectados fueron asignados al grupo taxonómico al que pertenecen utilizando la herramienta de BLAST+ embebida en QIIME 2™. Se realiza un alineamiento local entre las secuencias representativas obtenidas y las secuencias de referencia de Greengenes y se asigna la taxonomía consenso entre las coincidencias principales. Se ha utilizado la herramienta de BLAST+ ya que presenta una alta precisión a nivel de especie comparado con otros métodos [20] utilizando poco tiempo de computación.

2.6. Visualización de los resultados del análisis.

El formato de visualización de los resultados en la plataforma de QIIME 2™ es un gráfico de barras donde se representan los porcentajes relativos de cada taxón detectado en las muestras.
Este gráfico de barras es interactivo y se puede seleccionar un nivel taxonómico para visualizar (de Dominio a Especie) y las características del archivo `map_file.txt` con respecto a las que ver los resultados. Por la magnitud de las muestras, no se puede incluir el gráfico de taxonomía completo. Se han agrupado las muestras por el grupo al que pertenecen, tomando la media de frecuencias por grupo. En la Figura 6 se exhiben los resultados obtenidos de la asignación taxonómica por grupo de muestras.

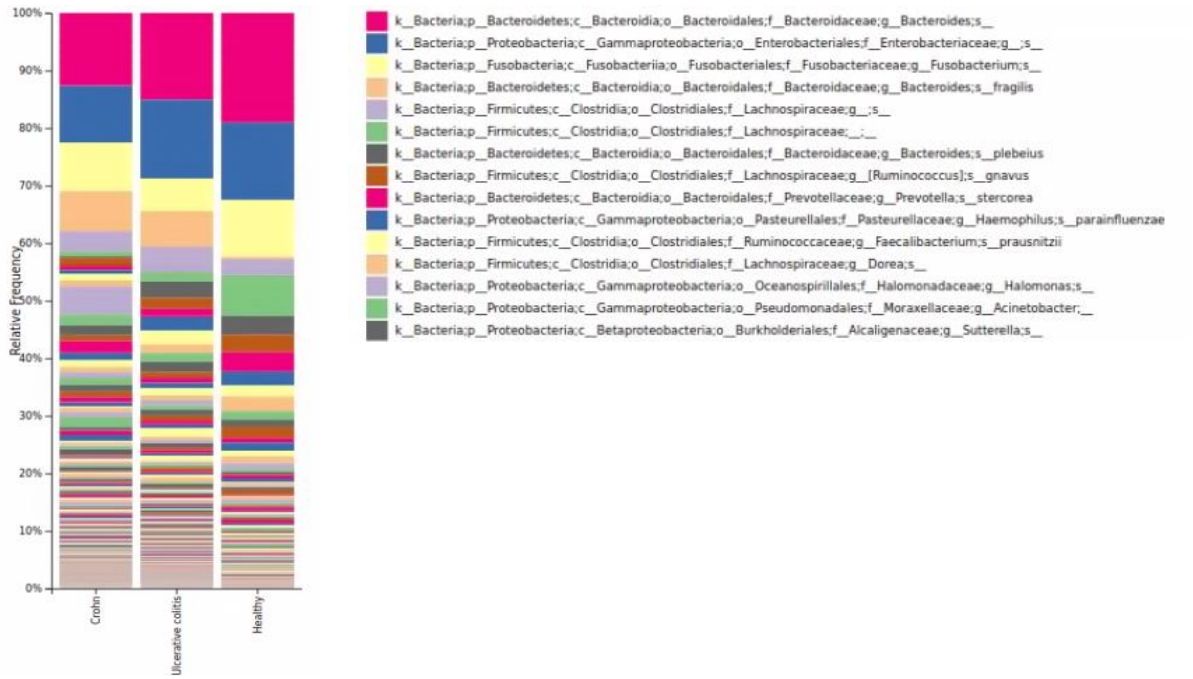

Figura 6. Distribución taxonómica de las muestras agrupadas a nivel de Especie.
Los colores de las barras indican las especies identificadas, dada la dimensión de la leyenda se ha incluido solo una parte, la correspondiente a las especies más abundantes.

Finalmente, es necesario extraer toda la información contenida en los archivos con formato .qza o .qzv para obtener archivos tabulares, imágenes y otros tipos de ficheros interpretables.

En la Figura 6 se puede ver que *Bacteroides fragilis* es más abundante en las muestras de pacientes con enfermedad de Crohn y colitis ulcerosa. Pero, debido a la alta dimensionalidad de los datos, observar la diferencia entre grupos es complejo.

Para una mejor visualización, en la Figura 7 puede verse un gráfico de burbujas, en el que se observan los 30 géneros más abundantes en cada grupo de muestras.

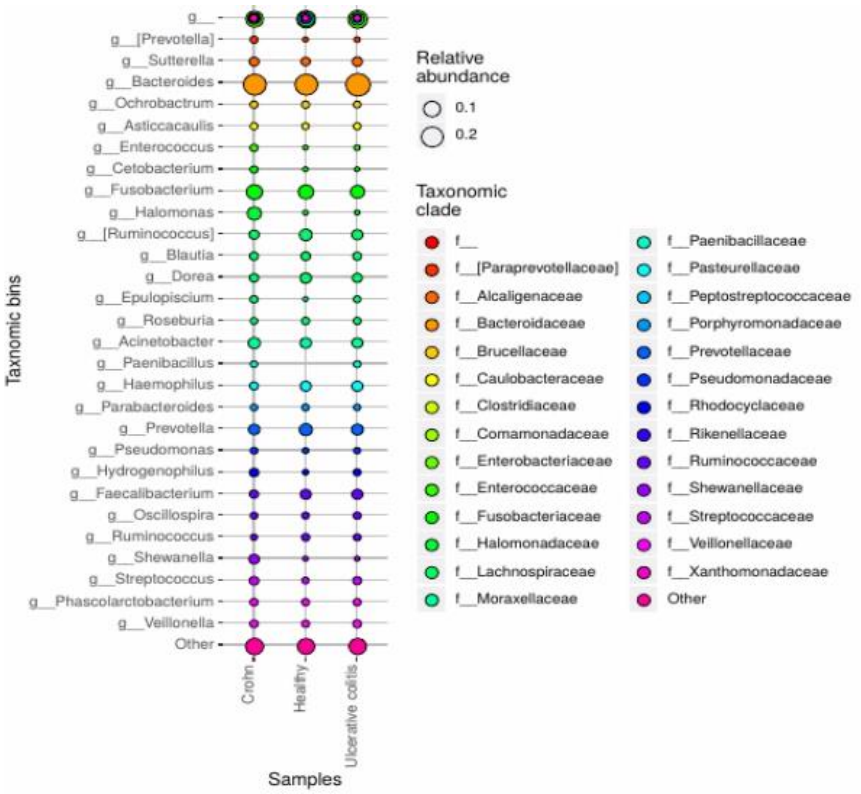

Figura 7. Gráfico de burbujas de los 30 géneros más abundantes en las muestras, agrupadas por clases.

El tamaño del punto indica la abundancia relativa media y el color la familia a la que pertenecen los géneros.

En este gráfico ya se pueden observar ciertas diferencias entre los grupos de muestras, como en las bacterias del género *Halomonas* y *Shewanella*, que están presentes en las muestras de pacientes de Crohn de forma más abundante que en el resto. O en el género *Paenibacillus*, cuya ausencia en las muestras de pacientes sanos podría dar a estas bacterias el papel de marcadores de la IBD.

Para conocer las especies diferencialmente abundantes en un grupo con respecto a los demás se requieren métodos estadísticos que permitan definir si la diferencia en los valores de abundancia relativa es mayor de lo que se esperaría por azar. Dado que la finalidad de este trabajo no es determinar aquellas bacterias diferenciales en las muestras, se va a pasar al estudio de técnicas de clasificación supervisada a partir de las tablas de asignación taxonómica obtenidas.

3. Estudio de técnicas de clasificación supervisada de los perfiles microbianos de pacientes sanos y con dos condiciones de IBD.

Se ha realizado un estudio en profundidad de diversas técnicas de clasificación supervisada con los datos de perfiles microbianos obtenidos para las muestras.

A continuación, se desglosan los subapartados que llevaron a la obtención de este segundo objetivo.

3.1. Estructuración de los datos para su uso en *machine learning*.

Para poder trabajar con la tabla de SVs, es necesario transponer los datos para que las muestras queden en filas y los identificadores de cada SV en columnas. Los datos utilizados ya están normalizados ya que se utiliza la tabla de frecuencias relativas, con valores entre 0 y 1. Se añadió una columna denominada "Class" que corresponde con el grupo al que pertenece cada muestra. Esta columna se transformó a un factor con tres niveles: "Healthy", "Crohn" y "Ulcerative". Se importaron los datos a RStudio, un entorno para trabajar con el lenguaje de programación R. De forma inicial se visualiza la estructura de datos para asegurarse de que se han cargado correctamente y se comprueba que no hay columnas vacías en los datos. La tabla de SVs original, así como todo el código relacionado con este trabajo se encuentra en Bitbucket (https://bitbucket.org/), un repositorio para desarrollar y compartir código en diferentes lenguajes de programación.

3.2. Creación de los conjuntos de entrenamiento y prueba.

De las 211 muestras iniciales 128 pertenecen al grupo con enfermedad de Crohn (Crohn), 14 al grupo de pacientes sanos (Healthy) y 69 al grupo con colitis ulcerosa (Ulcerative). El conjunto de datos está claramente desequilibrado, siendo la clase de pacientes sanos la menos abundante. Para tratar con un conjunto de datos no muy extenso y desequilibrado se generaron diferentes conjuntos de entrenamiento y prueba utilizando la validación cruzada de Monte Carlo [21]. Se selecciona de forma aleatoria y sin reemplazamiento una parte de los datos para ser designada como conjunto de entrenamiento, dejando el resto como conjunto de prueba. Este proceso se repite las veces deseadas produciendo nuevas particiones de entrenamiento y prueba cada vez. Tiene ventajas respecto a otros métodos de validación cruzada como *leave-one-out*, que suele llevar al sobreajuste de modelos, o como la validación cruzada de k iteraciones, más limitada ya que no permite que un mismo punto se repita en diferentes conjuntos de prueba.

Para ello, el consultor de este trabajo facilitó un código en R que requiere del conjunto de datos inicial, del número de iteraciones a realizar y del tamaño del conjunto de entrenamiento para cada iteración. Se utilizaron 10 iteraciones y un 67% de los datos como conjunto de entrenamiento, es decir, aproximadamente 2/3 del total. Este código genera los índices de las muestras correspondientes en cada repetición del proceso.

3.3. Entrenamiento de los diferentes modelos.

Siguiendo otros estudios en los que se aplica el aprendizaje supervisado a problemas de clasificación relacionados con datos de microbiota [4,5], se seleccionaron los siguientes modelos: lasso, k vecinos más cercanos (kNN), árboles de decisión, random forest, Máquinas de Soporte Vectorial (SVM) con

kernel lineal y radial, redes neuronales y XGBoost (eXtreme Gradient Boosting).

Lasso es un método de regresión que utiliza la selección de variables y la regularización para mejorar el rendimiento e interpretabilidad del modelo. A pesar de ser un método de regresión ha demostrado funcionar bastante bien con datos de microbioma [4], por su capacidad de seleccionar las variables importantes para el modelo. Se engloba dentro de otros métodos lineales que añaden una penalización a la complejidad del modelo para evitar el sobreajuste. Para aplicar este algoritmo se utilizó el paquete *glmnet* [22] que contiene herramientas para elegir el mejor valor de lambda con el que posteriormente entrenar el modelo.

kNN es un método de aprendizaje vago (*lazy learning*) en el que no se produce un modelo, sino que se trata de clasificar nuevas muestras basándose en el punto o puntos más cercanos. Estos puntos o vecinos se designan con la variable *k*, que permite determinar cuántos vecinos se deberían tener en cuenta para tomar la decisión. Aunque suele presentar un rendimiento bastante bajo, en este caso se optó por utilizar el método de kNN ponderado, obteniendo mejores resultados. Este tipo de algoritmo aumenta la influencia que tienen las muestras cercanas a la nueva observación frente a aquellas que están más alejadas [23]. Se utilizó el paquete *kknn* [24] de R, que utiliza validación cruzada para elegir el mejor valor de *k* y la mejor técnica de ponderación de las distancias entre muestras.

Los **árboles de decisión** permiten tomar decisiones complejas a partir de elecciones más simples. En el caso de los árboles de clasificación, utilizan una estructura de árbol para modelizar las relaciones entre las características de las muestras y la clase a la que estas pertenecen. Para el proceso de tuneado del modelo, es decir, encontrar los mejores parámetros, se utilizó una función del paquete *e1071* [25], que agrupa herramientas para optimizar modelos. Y para el entrenamiento posterior del modelo se utilizó el paquete *rpart* [26], que ajusta árboles de decisión.

Los bosques aleatorios o **random forest** utilizan conjuntos de árboles de decisión. Estos modelos realizan votaciones para combinar las predicciones de los diferentes árboles y obtener aquellas con mejor rendimiento. Permiten trabajar con grandes conjuntos de datos y son eficaces para casos como el que se aborda, en el que las variables predictivas superan en número a las muestras. Se han utilizado obteniendo los mejores resultados en problemas de clasificación de muestras de pacientes con diferentes enfermedades [5]. Para la elección de parámetros y el entrenamiento de este algoritmo se utilizó el paquete *randomForest* [27].

Las Máquinas de Soporte Vectorial o **SVM** son algoritmos de clasificación que maximizan la separación entre clases basándose en la frontera entre estas. Son modelos muy efectivos con muestras grandes y en casos de alta dimensionalidad como este. Permiten la clasificación de datos linealmente separables, así como aquellos que no se pueden separar de forma lineal. En este último caso, se aplica una transformación no lineal a un espacio de

dimensión mayor donde se pueda realizar la división linealmente. Se ha entrenado un SVM lineal y uno con función de núcleo radial. Para ello se utilizó la función *svm* del paquete *e1071* previamente citado.

Las **redes neuronales** utilizan redes de neuronas artificiales, o nodos, para encontrar las relaciones entre una serie de señales de entrada y la señal de salida. La variable respuesta es modelada por un conjunto de variables intermedias denominadas ocultas (*hidden*), que están organizadas en capas (*layers*). Para modificar el número de capas y nodos que contiene cada una y entrenar el modelo se utilizó el paquete *neuralnet* [28]. No son los modelos más populares ya que su interpretación es, a menudo, compleja.

El algoritmo **XGBoost** se engloba en los métodos multiclasificadores, que crean un modelo combinando un conjunto de métodos de aprendizaje. Se vale del método de *boosting,* en el que se utilizan algoritmos más débiles para generar otros más robustos. En la bibliografía se ha señalado a los algoritmos basados en *boosting* como los modelos que mejor clasifican comparados con otros clasificadores [29]. En este trabajo, como algoritmo de optimización se emplea el descenso de gradiente. Se ha utilizado el paquete *xgboost* [30] en la elección de los mejores parámetros y el entrenamiento del modelo.

Una vez se han definido los modelos y se ha familiarizado con el funcionamiento de los paquetes, funciones y parámetros que se pueden modificar, se va a evaluar el rendimiento de los modelos. Inicialmente, se hizo con los parámetros por defecto para establecer las bases de la evaluación que permitan, posteriormente, comparar los modelos de forma precisa.

3.4. Evaluación del rendimiento de los modelos con el conjunto de prueba.

Se evalúa la calidad de un modelo mediante su capacidad predictiva, calculada como la correlación entre los valores observados para la variable respuesta y los valores predichos por el modelo. En un modelo de clasificación las predicciones pueden ser obtenidas en dos formatos: como las clases a las que pertenecen las muestras o como la probabilidad de las muestras de pertenecer a una clase u otra. Para la evaluación de un modelo se deben utilizar datos diferente al conjunto empleado para entrenar el modelo. En caso contrario, se puede producir el sobreajuste (*overfitting*) del modelo. Para la evaluación de los modelos se utiliza *predict*, una función genérica integrada en R para hacer predicciones a partir de diferentes modelos.

Una vez realizadas las predicciones, se eligieron los estadísticos adecuados para evaluar la capacidad predictiva de los modelos entrenados. Al tener una variable predictora categórica, se genera una matriz de confusión, que contiene información de las observaciones correcta e incorrectamente clasificadas. Para ello se utiliza una función de *caret* [31], un paquete que incluye muchas funciones y facilita el uso de distintos métodos de clasificación y regresión. Aunque es un paquete muy completo, en este trabajo se ha optado por utilizar funciones específicas para cada modelo para poder modificar más hiperparámetros y aplicar la validación cruzada de Monte Carlo previamente.

De la matriz de confusión se obtuvieron los valores de precisión (*accuracy*), o la proporción de clasificaciones correctas, y la tasa de error (*error rate*), que es la proporción de clasificaciones incorrectas o la inversa de la precisión. También se obtiene el valor del coeficiente de la Kappa de Cohen [32], que mide la concordancia entre la precisión observada y la esperada. Por otra parte, y utilizando el paquete *pROC* [33], se calcula el área bajo la curva ROC (AUC), una medida de la capacidad predictiva global del modelo. El valor de AUC se utiliza para comparar la capacidad de predicción de diferentes modelos, usando las probabilidades predichas.

Se ha creado una función para que, a partir de un modelo entrenado y un conjunto de datos de prueba, se obtengan estos cuatro estadísticos, que evalúan el rendimiento del modelo.

3.5. Optimización de los modelos.

Este paso es el más importante para conseguir un modelo óptimo. El ajuste de los hiperparámetros se hizo utilizando las funciones específicas que ofrece cada paquete. Así, se automatizó la exploración de hiperparámetros de manera eficiente. Para cada modelo se han elegido los parámetros que se van a modificar en función de la influencia que tienen sobre la eficacia del modelo. Posteriormente, se prueban varios rangos de valores hasta llegar a obtener el menor error de clasificación posible, sin llegar a sobreajustar el modelo.

Idealmente, se contaría con un conjunto de validación para elegir el mejor modelo y configurar los hiperparámetros para obtener una estimación del error más generalizada. Debido a la cantidad de datos disponibles para este trabajo, este paso se ha sustituido por la validación cruzada de Monte Carlo sobre los datos de entrenamiento para optimizar cada uno de los modelos.

Una vez se han encontrado los parámetros óptimos, se entrenan los modelos con ellos y se evalúa el rendimiento para posteriormente hacer una comparación de todos ellos. A continuación, se detallan los parámetros que se han elegido para modificar en cada algoritmo y los resultados obtenidos.

Para optimizar el modelo de **lasso**, la variable que se puede controlar es el valor de lambda. Este valor determina la penalización para los coeficientes de las variables. Si lambda es lo suficientemente elevado, estima un valor de cero para ciertos coeficientes produciendo la exclusión de variables en el modelo. En la función implementada para este modelo únicamente se necesita el número de iteraciones que se quieran llevar a cabo para obtener el valor de lambda óptimo. Durante cada una de estas iteraciones, se acumula el error medio y la desviación estándar y se devuelve el valor de lambda que genera un error medio más bajo. En la Figura 8 se puede ver como los mejores valores de lambda tras cada una de las iteraciones de la validación cruzada varían de 0,06 a 0,11. Estos valores están dentro del rango de 0 a 1 en el que, si el valor de lambda fuese cercano a uno, nos estaría indicando el modelo nulo sin ninguna variable. Según el valor de lambda va aumentando, más variables son

añadidas al modelo. Tras todas las iteraciones, se elige el mejor valor de lambda para entrenar los modelos.

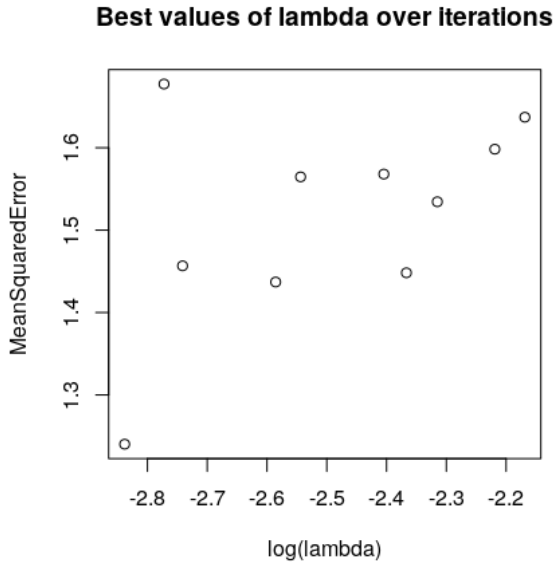

Figura 8. Mejores valores de lambda tras cada iteración.
Se representa el mejor valor de lambda (en escala logarítmica) junto al valor asociado del error medio cuadrado.

En el modelo de **kNN** se definió un valor máximo de *k* vecinos sobre el que iterar para obtener el óptimo. Normalmente se recomienda que el valor de *k* sea la raíz cuadrada de las observaciones del conjunto de entrenamiento. En este trabajo se ha utilizado un valor máximo de *k* de 11, utilizando un número impar para evitar el riesgo de empate a la hora de clasificar una nueva muestra. Adicionalmente, la función permite elegir entre 10 técnicas de ponderación de la distancia. En este caso se han utilizado 6, una de ellas no ponderada: rectangular (no ponderada), triangular, epanechnikov, gaussian, biweight y triweight. Se ha demostrado que utilizando técnicas de ponderación se corrige el error provocado por la elección una *k* muy alta y que son muy útiles con datos de alta dimensión, como pueden ser los microarrays [34]. En la Figura 9 se pueden ver los errores de clasificación obtenidos para cada valor de *k* con cada técnica de ponderación utilizada. Los valores más bajos del error de clasificación se obtienen con una *k* de 2 y con los métodos rectangular y gaussiano. El método rectangular no es ponderado, y asigna los mismos pesos a todos los vecinos de la observación a clasificar. Estos resultados parecen indicar que la elección del método de ponderación no es algo decisivo. El valor de vecinos óptimo es bajo, lo que implica que el ruido subyacente en los datos va a tener más peso en el resultado. En la Figura 9 se observa como con valores de *k* de 10 el error de clasificación es prácticamente similar que con una *k* de 2. Aumentar el valor de *k* puede aumentar el coste computacional, pero hacer un modelo más simple y suavizar los límites de decisión. Por ello, es importante prestar atención a los resultados y gráficos obtenidos durante la optimización de los parámetros y, en base a los datos, elegir los valores óptimos.

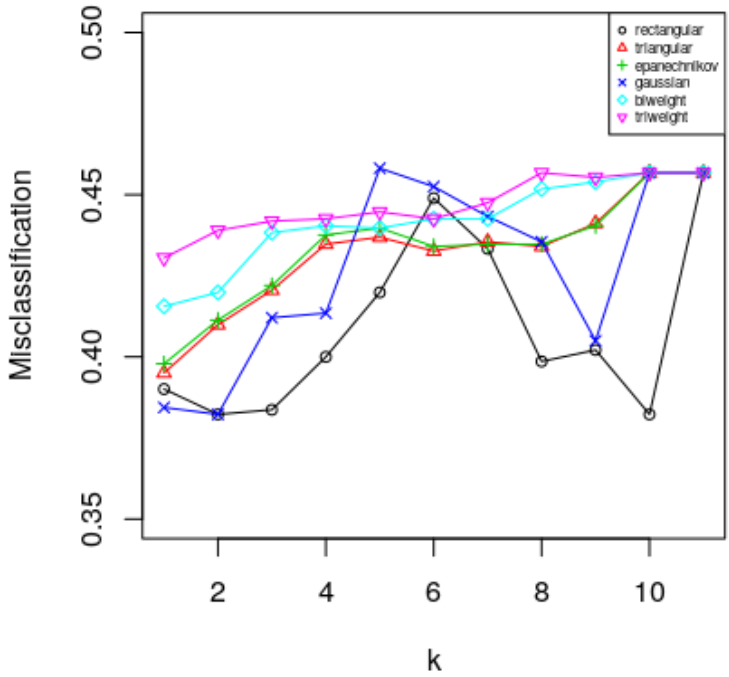

Figura 9. Error de clasificación para cada valor de k y de técnica de ponderación.
El error obtenido se corresponde con la media de los valores durante cada iteración.

Para tratar de mejorar los **árboles de decisión**, se modificaron los parámetros *minsplit* y *cp*, que son el mínimo número de observaciones en un nodo para intentar partirlo y el punto de corte del parámetro de complejidad, respectivamente. El parámetro *minsplit* es por defecto 20, pero se han probado valores por encima y por debajo de dicho número ya que una de las clases en el conjunto de datos tiene menos de 20 observaciones. El parámetro *cp* impone una penalización en la construcción del árbol, de forma que cuanto mayor sea su valor, los árboles generados van a tener menos nodos y van a ser más simples. Se han utilizado valores por encima y por debajo del valor por defecto (0,01) y validación cruzada para obtener el óptimo. El número mínimo de observaciones en un nodo antes de dividirlo con un error de clasificación menor es de 70, como indica la Figura 10. Este valor de *minsplit*, además de ayudar a evitar el sobreajuste, genera un árbol con menor profundidad y, por lo tanto, más fácil de entender. En este modelo, el valor de *cp*, cuyo óptimo es 0,05, parece menos crucial a la hora de obtener mejores resultados.

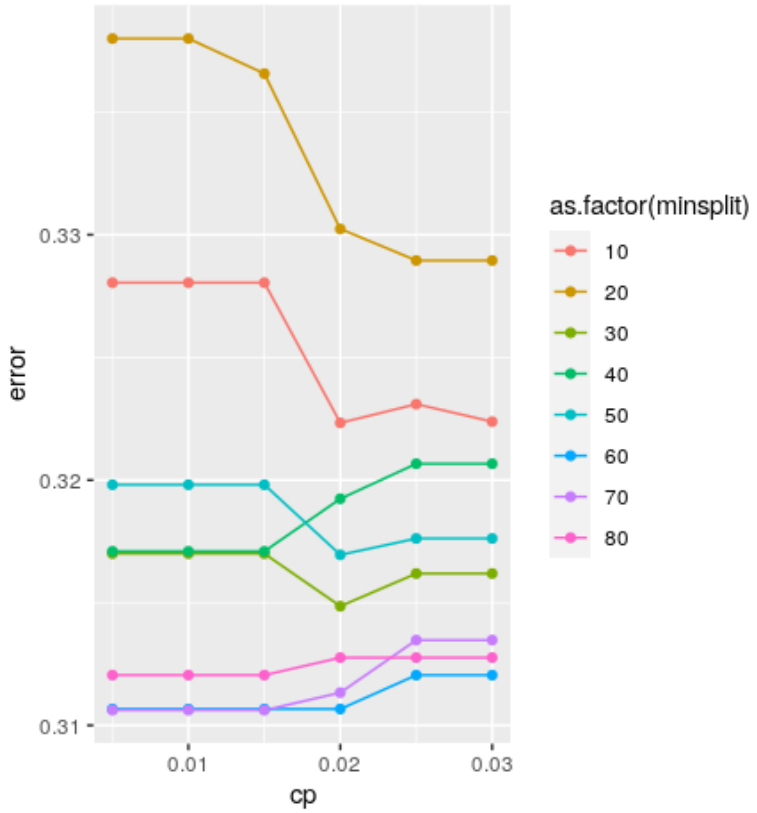

Figura 10. Tasa de error de clasificación para cada valor de *minsplit* y *cp*.
El error obtenido se corresponde con la media de los valores durante cada iteración. El color de las líneas indica el valor de *minsplit* utilizado.

En el modelo de **random forest**, dos de los parámetros que se pueden mejorar son el número de árboles (*ntree*) y el número de variables aleatorias que son seleccionadas en cada división (*mtry*). La elección del mejor valor del número de árboles es diferente a otros parámetros ya que, al seguir aumentando, también lo hace la precisión del modelo, hasta llegar a saturación. Por ello, hay que tratar de obtener un número de árboles elevado que no disminuya el rendimiento del modelo sin aumentar excesivamente el tiempo de computación. Normalmente, alrededor de 1000 árboles el error de clasificación se estabiliza. Como se puede observar en la Figura 11, en este caso a partir de 500 árboles el error *out-of-bag* (OOB) llega a saturación. Si se utiliza un menor número de árboles para construir el modelo, el coste computacional se reduce notablemente.

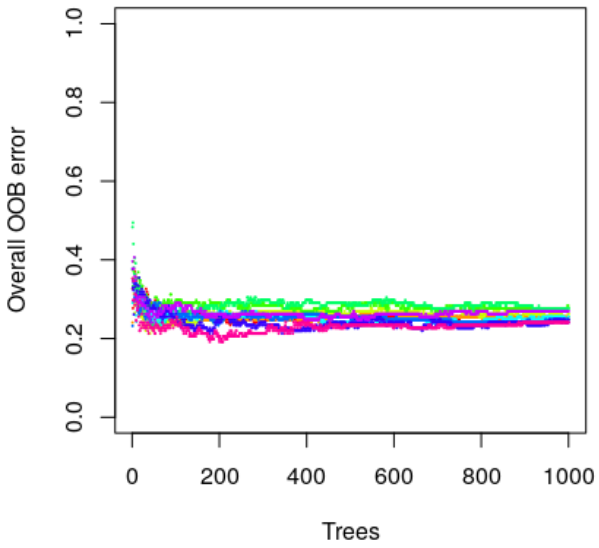

Figura 11. Tasa de error OOB total en función del número de árboles.

En cuanto al valor de mtry, el recomendado es la raíz cuadrada del número de variables en el conjunto de datos, por lo que va a ser modificada si se hace ingeniería de características y se reduce la dimensionalidad. Se eligieron valores alrededor de 92, la raíz cuadrada del número de variables predictivas. Valores más bajos de *mtry* producen árboles más diferentes y menos relacionados entre sí, dando mayor estabilidad a la hora de agregarlos [35]. Valores por debajo de 92 resultan en tasas de error más elevadas, mientras que valores ligeramente por encima llevan al óptimo. En la Figura 12 se puede ver que en este caso el mejor valor de *mtry* es 160.

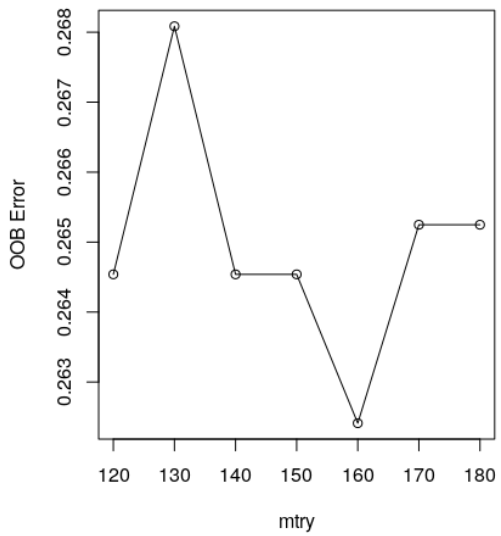

Figura 12. Tasa de error OOB total en función del valor de *mtry*.
El error obtenido se corresponde con la media de los valores durante cada iteración.

Se realizó un modelo de **SVM** lineal y uno ajustado con la función radial gaussiana de núcleo, ya que el SVM con núcleo polinomial no dio buenos resultados. Siguiendo lo recomendado por la bibliografía [36], se probaron secuencias del parámetro C (coste para malas clasificaciones) y de gamma (define la influencia de una sola observación de entrenamiento)

exponencialmente crecientes, para identificar los parámetros óptimos. Se debe encontrar el valor del parámetro C mayor que disminuya notablemente el error, pero sin llegar al sobreajuste del modelo. Valores más pequeños de C permiten un mayor número de errores y definen regiones de decisión más suaves, mientras que valores muy elevados permiten menos observaciones mal clasificadas hasta el punto de sobreajustar el modelo. Algo muy parecido pasa con el valor de gamma donde valores altos implican que el radio del área de influencia incluya solo al propio vector de soporte [37]. Para ajustar el SVM lineal el parámetro gamma no es necesario y solo se han probado diferentes valores del parámetro C. En la Figura 13 se observa que el valor más bajo del error se obtiene con un valor de C de 1000.

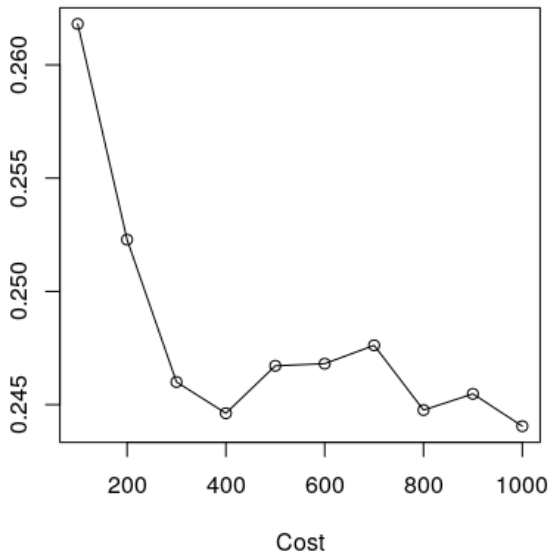

Figura 13. Tasa de error en función del parámetro C.
El error obtenido se corresponde con la media de los valores durante cada iteración

Mientras que en el SVM radial se modifican tanto los parámetros C como gamma. En este caso, como indica la Figura 14, el mejor valor de C es 200 y de gamma 0,7. Si seguimos aumentando el valor de gamma el error sigue bajando, aunque a un ritmo muy inferior hasta llegar a una meseta. Por ello se ha elegido un rango de valores con un máximo de 1 para obtener el menor valor de gamma que produzca un error de clasificación más pequeño.

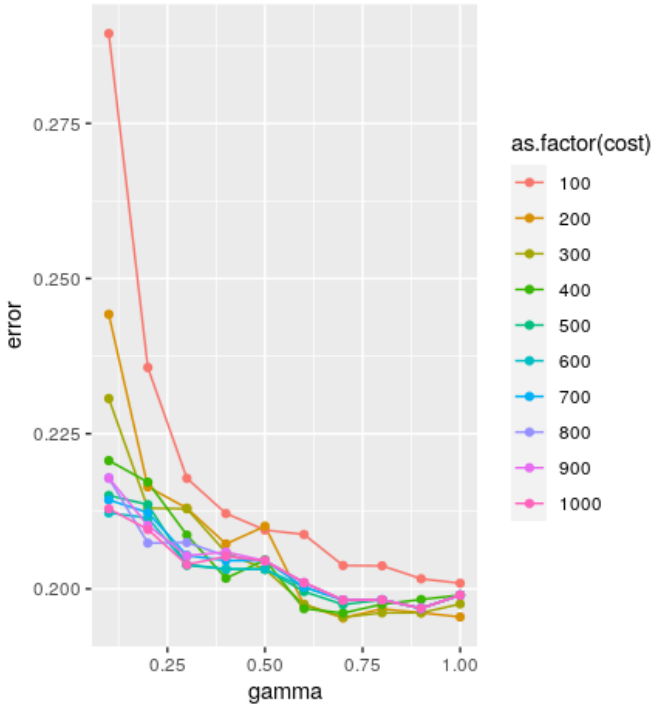

Figura 14. Tasa de error en función del parámetro C y gamma.
El error obtenido se corresponde con la media de los valores durante cada iteración. El color de las líneas indica el valor de C utilizado.

En el modelo de **redes neuronales** se modificaron el número de capas ocultas y de nodos contenidos en estas. Inicialmente se probaron hasta 5 capas ocultas, pero, en la mayoría de los casos, una única capa oculta con suficientes nodos es una función aproximadora universal [38]. En cuanto a qué número de nodos utilizar existen diversos criterios, pero lo más importante es tener en cuenta que un exceso de nodos puede llevar a que el modelo se ajuste a la perfección a los datos de entrenamiento y no a nuevas observaciones [39]. En nuestro caso, el menor valor de la suma de errores al cuadrado (SSE) se obtiene siempre con una única capa oculta. Y, en concreto, en la Figura 15 se puede visualizar que el error es menor utilizando una sola capa con 40 nodos.

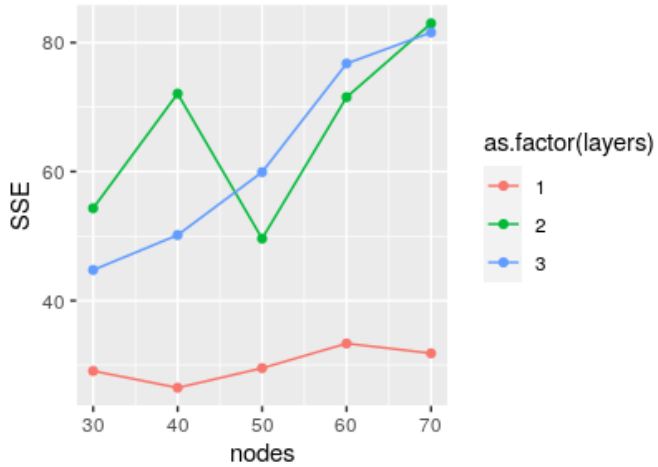

Figura 15. SSE en función del número de capas y nodos de la red neuronal.
El SSE obtenido se corresponde con la media de los valores durante cada iteración. El color de las líneas indica el número de capas.

Para el algoritmo de **XGBoost** existen muchos parámetros susceptibles de modificación. Tratando que la puesta a punto del modelo no fuese muy compleja y, tras un estudio de los parámetros con mayor influencia sobre el rendimiento del modelo, se han elegido dos hiperparámetros. El primero de ellos es *max_depth*, el número de nodos de bifurcación de los árboles de decisión utilizados en el entrenamiento. Es importante mantener la profundidad máxima en valores bajos que eviten la aparición de interacciones complejas debidas al ruido en los datos [40]. El segundo parámetro es *eta*, o la tasa de aprendizaje del modelo. Al igual en el caso anterior, un valor elevado de este parámetro puede producir mejores resultados a costa de sobreajustar el modelo. Lo ideal es encontrar valores lo más pequeños posibles que resulten en un óptimo del rendimiento. Las menores tasas de error de clasificación se consiguen a partir de un valor de 8 de *max_depth*. En la Figura 16 se superponen los resultados obtenidos con valores de *max_depth* de 8, 9 y 10. Además los mejores resultados se obtienen con un valor de *eta* de 0,5, aunque parece que esta variable no es tan crucial para el rendimiento.

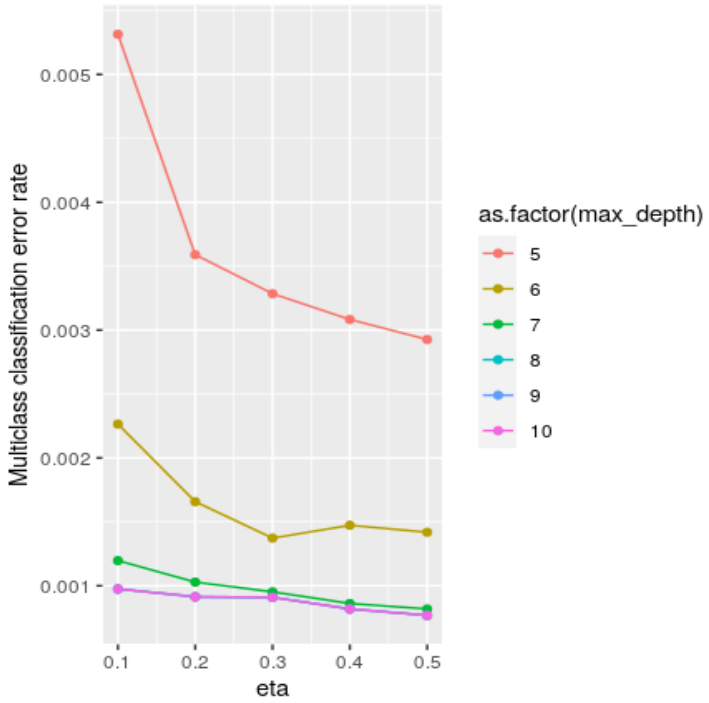

Figura 16. Tasa de error de clasificación en función de los valores de *eta* y *max_depth*. El error obtenido se corresponde con la media de los valores durante cada iteración. El color de las líneas indica los diferentes valores de *max_depth*.

Durante todo el proceso de ajuste de parámetros se ha utilizado el método de validación cruzada ya mencionado. Tras elegir los hiperparámetros que iban a ser modificados en cada modelo, se definieron funciones específicas de ajuste, entrenamiento y predicción de nuevos datos con cada algoritmo.

Después de ajustar los parámetros, se entrenaron los modelos con aquellos que dieron mejores resultados y se evaluaron sobre los datos de entrenamiento. En la parte izquierda de la Figura 17 se observan los valores de AUC obtenidos durante el proceso de validación cruzada para todos los

modelos entrenados. Mientras que en la parte derecha de la Figura se analizan conjuntamente los valores medios de AUC y de precisión de los modelos.

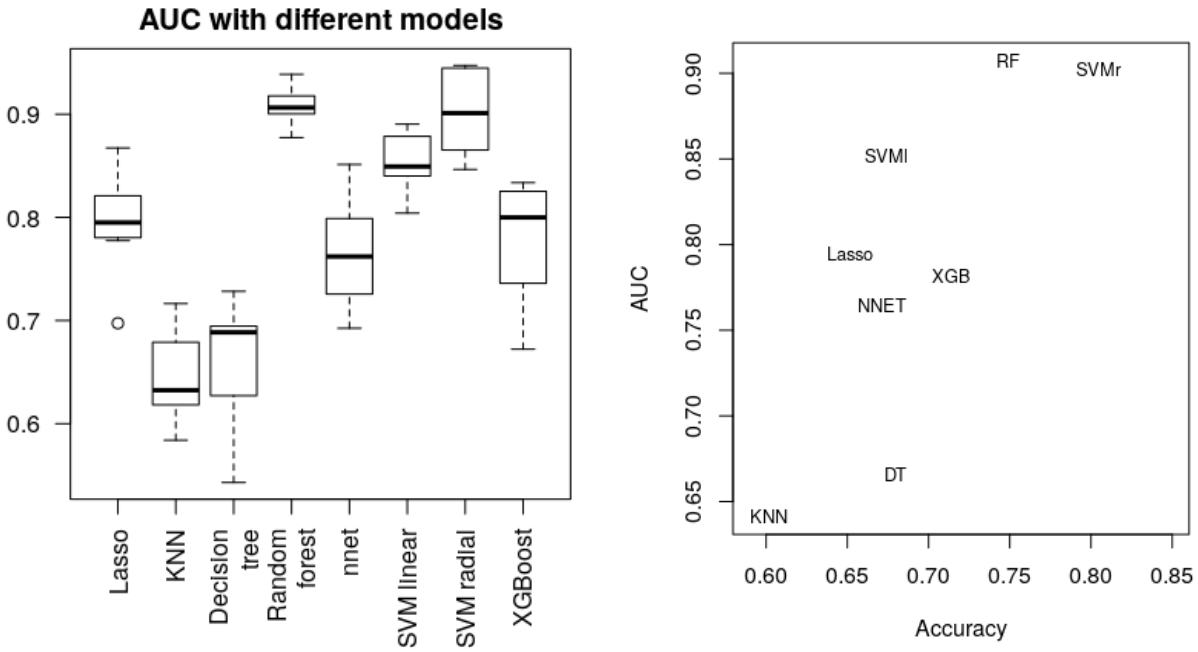

Figura 17. Comparación de los valores de AUC y precisión de los distintos modelos predictivos.
En la parte izquierda se comparan los valores de AUC obtenidos con validación cruzada y en la derecha se analizan conjuntamente los valores medios de AUC y precisión de todas las técnicas.

Los mejores resultados con el conjunto de datos original se obtienen con los modelos de random forest y SVM radial. La capacidad predictiva de estos dos modelos es muy buena. Mientras que el SVM radial tiene un rango de valores de AUC más ancho en diferentes iteraciones, random forest parece comportarse de forma más robusta. En cuanto a sus valores medios, SVM radial muestra mayor precisión frente a un mayor valor de AUC con random forest.

Se debe tener en cuenta también la posibilidad de que se hayan sobreajustado los modelos a los datos. Para hacer esta comprobación, las clases a las que pertenecen las observaciones se han muestreado de forma aleatoria y sin reemplazamiento. De esta forma, la mayoría de las muestras tendrán una clase diferente a la que tenían inicialmente. Se han vuelto a entrenar los modelos con estos datos y los valores de precisión y AUC obtenidos han sido mucho menores, disipando las sospechas de sobreajuste. En la Figura 18 se muestra la comparativa de los valores de AUC obtenidos en cada iteración de la validación cruzada en todos los modelos, antes y después de la reorganización de las clases.

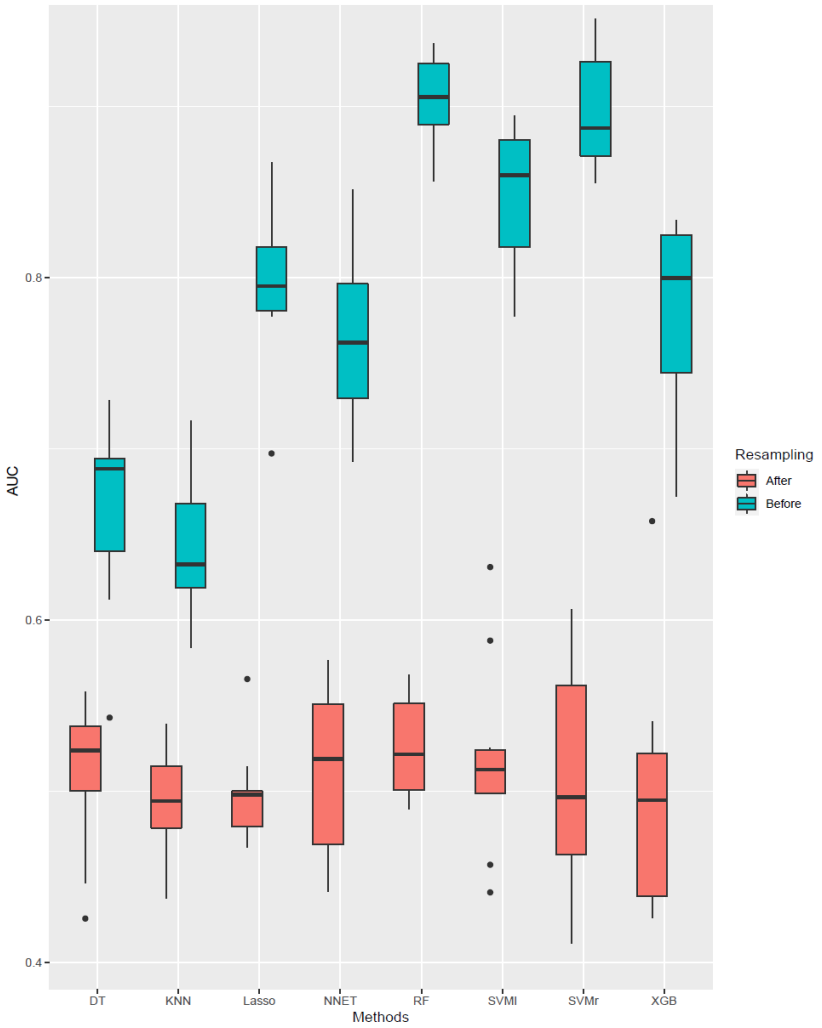

Figura 18. Comparación de los valores de AUC antes y después del remuestreo. En rojo se representan los valores obtenidos durante cada iteración tras reorganizar las clases de forma aleatoria.

En la Figura 18 se ve como, en todos los casos, los valores de AUC obtenidos oscilan en torno al 0,5. Esto implica que estos modelos no tienen una buena capacidad predictiva y que producen los mismos resultados que se obtendrían por azar. Mientras que la capacidad predictiva de los modelos con las clases correctamente designadas es considerablemente mayor. Cabe destacar que, al igual que se veía en la Figura 17, los modelos que peor se comportan son los árboles de decisión y kNN.

3.6. Aplicación de ingeniería de características para reducir la dimensionalidad de los datos.

La presencia de variables redundantes y que aporten ruido a los datos durante el entrenamiento de un modelo puede reducir drásticamente la capacidad predictiva y aumentar el coste computacional. La selección o ingeniería de características es el proceso que permite eliminar características poco o nada relevantes y es útil para reducir la dimensionalidad de los datos y mejorar la precisión de los modelos.

Por todo ello y, con el fin de aportar varias opciones para realizar este paso, en este trabajo se han abordado tres formas de reducir la dimensionalidad de los datos. Se han seleccionado tres métodos con enfoques diferentes. El primero de ellos es una herramienta desarrollada específicamente para datos derivados de análisis de la diversidad microbiana. Mientras que los otros dos están basados en la selección de variables de forma iterativa, añadiendo o eliminando características según su aporte al modelo.

El primer método es **HFE** (*hierarchical feature engineering*) [41], que utiliza la información taxonómica y su correlación para explorar la jerarquía subyacente de las variables, en este caso, los SVs. Produce como resultado un conjunto de variables informativas, algunas incluyendo los SVs originales y otras agrupándolos en diferentes niveles taxonómicos. Los resultados de AUC para los modelos y la relación entre los valores medios de AUC y precisión tras este paso de reducción de la dimensionalidad se pueden ver en la Figura 19.

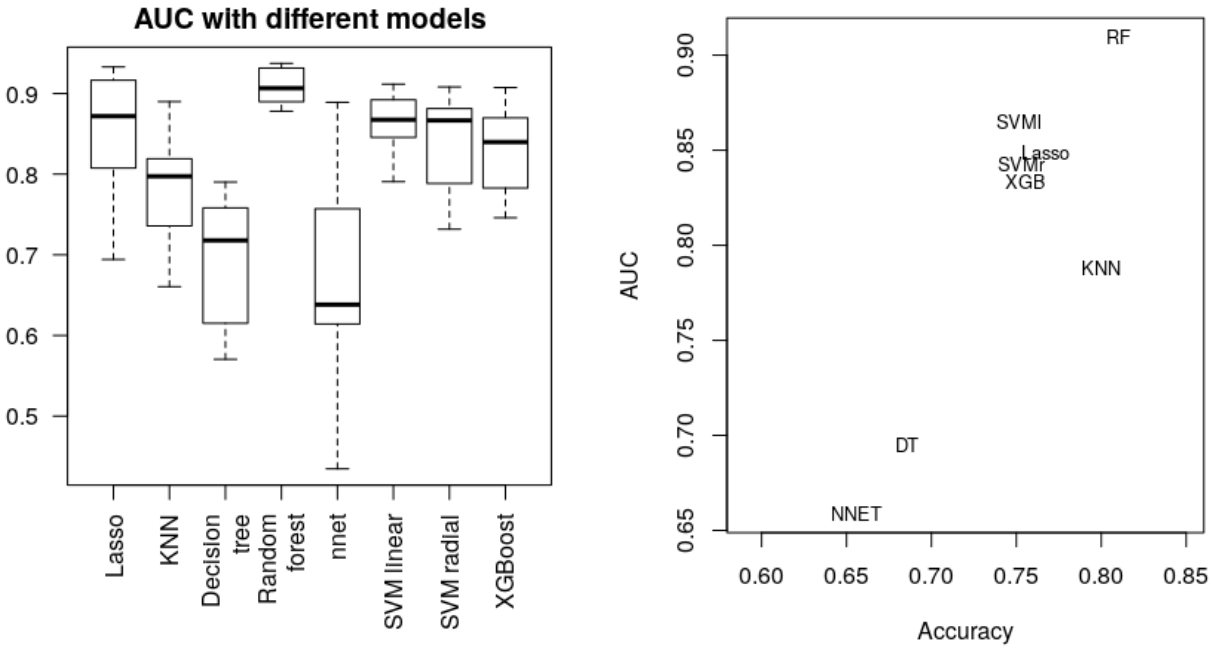

Figura 19. Comparación de los valores de AUC y precisión de los distintos modelos predictivos tras aplicar HFE.
En la parte izquierda se comparan los valores de AUC obtenidos con validación cruzada y en la derecha se analizan conjuntamente los valores medios de AUC y precisión de todas las técnicas.

La capacidad predictiva de alguno de los modelos aumenta al reducir la dimensionalidad de los datos con este método. Entre ellos, random forest supera al resto. Aunque también cabe destacar el buen rendimiento del modelo de SVM lineal, que presenta una precisión similar al SVM con kernel radial.

Para el segundo método de reducción de la dimensionalidad de los datos se utilizó el paquete ***Boruta***, que trata de encontrar todas las variables relevantes eliminando características de forma iterativa [42]. Este método utiliza la votación de muchos clasificadores débiles en los que se mide la importancia de las variables con la pérdida de precisión en la clasificación. Una vez se han

obtenido las características importantes del conjunto de datos, se entrenan los modelos y se evalúan, resultado que se puede observar en la Figura 20.

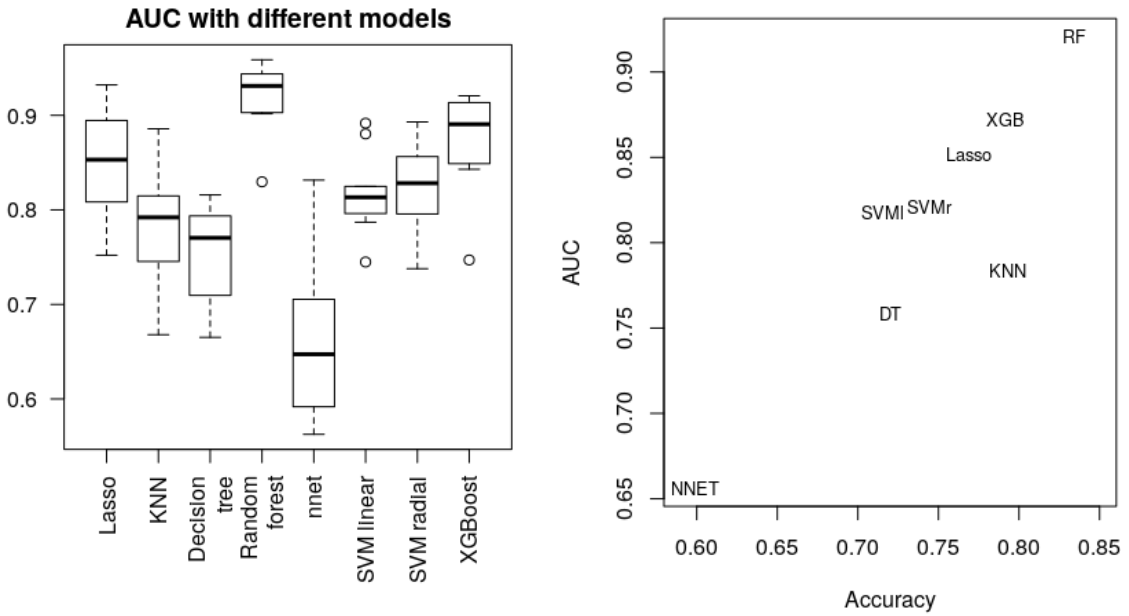

Figura 20. Comparación de los valores de AUC y precisión de los distintos modelos predictivos tras aplicar Boruta.
En la parte izquierda se comparan los valores de AUC obtenidos con validación cruzada y en la derecha se analizan conjuntamente los valores medios de AUC y precisión de todas las técnicas.

El método Boruta funciona especialmente bien con el modelo de random forest ya que originalmente se diseñó para integrarse en el entrenamiento de este tipo de algoritmos. También mejora notablemente el rendimiento del modelo de XGBoost, que está basado en árboles de decisión.

Se probó un último modelo de reducción de la dimensionalidad para tener un rango más amplio de comparaciones. Se utilizó el método de selección de variables por pasos, concretamente hacia adelante (***forward selection***). Este se basa en la selección secuencial de características añadiendo en cada paso la variable que más aporta al modelo, que inicialmente se crea sin ninguna variable. Para ello se utiliza la función *step* integrada en un paquete base de R y que utiliza como criterio el valor del AIC (criterio de información de Akaike), una versión de la suma de errores al cuadrado penalizada por el número de predictores. Los resultados de los modelos aplicando este método de selección de variables se pueden ver en la Figura 21.

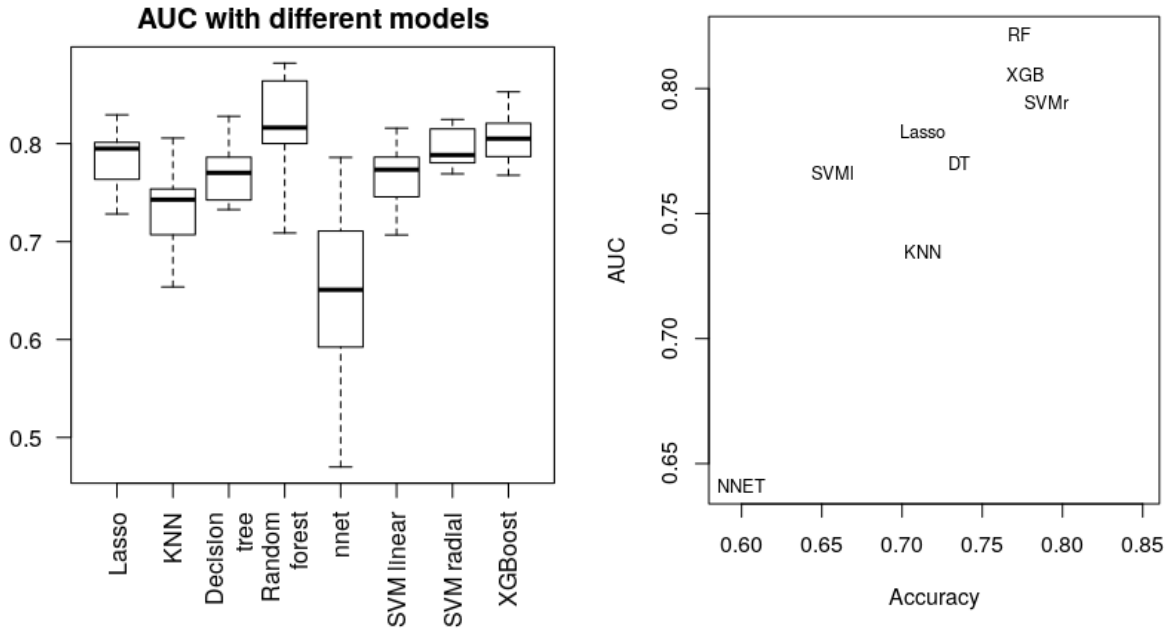

Figura 21. Comparación de los valores de AUC y precisión de los distintos modelos predictivos tras aplicar *forward selection*.
En la parte izquierda se comparan los valores de AUC obtenidos con validación cruzada y en la derecha se analizan conjuntamente los valores medios de AUC y precisión de todas las técnicas.

Con este método se obtienen peores resultados para todos los modelos, incluso comparado con el conjunto de datos iniciales sin reducción de dimensionalidad. Además, es el método de selección de variables que más tiempo computacional consume, por lo que no es el más adecuado para este tipo de datos.

Tras probar los diversos métodos de selección de variables, se han comparado los valores de kappa, precisión y AUC obtenidos con cada modelo, así como el tiempo empleado en el ajuste de hiperparámetros y en el entrenamiento y evaluación de los modelos. En la Tabla 1 se resumen los valores promedio de kappa, precisión y AUC para todos los modelos según el método de selección de variables utilizado. En esta Tabla están señalados los valores más altos y bajos de AUC en verde y en rojo respectivamente. Como se ha comentado previamente, con el método de forward selection se obtienen los valores de precisión y AUC más bajos para casi todos los modelos. Exceptuando el caso de los árboles de decisión que, tras aplicar esta reducción de variables se obtienen los mejores resultados. Cabe destacar que las redes neuronales sufren de una bajada en el rendimiento al aplicar cualquier tipo de reducción de la dimensionalidad. Puede que esto se deba a que las redes neuronales no funcionan especialmente bien con pocos datos o a que el paquete utilizado para entrenar el modelo no sea el ideal. Hay otros paquetes que ofrecen muchas más funciones y posibilidad de modificar diversas variables, pero se han probado en el desarrollo del trabajo y no se ha conseguido su correcto funcionamiento. En el modelo de SVM radial, los datos sin reducir generan mejores resultados que con el resto de los métodos.
Utilizando Boruta se obtienen los mejores resultados para 3 de los 8 modelos, siendo el método óptimo. Aunque dependiendo del modelo de interés, sería

interesante considerar HFE. Por ejemplo, si se quiere entrenar un SVM lineal, se ha visto que los resultados mejoran notablemente con este algoritmo de selección de variables.

Tabla 1. Valores promedio de kappa, precisión y AUC para todos los modelos.

	Sin reducir			HFE			Boruta			Forward selection		
	kappa	precisión	AUC	kappa	precisión	AUC	kappa	precisión	AUC	kappa	precisión	AUC
Lasso	0.210	0.651	0.795	0.526	0.767	0.849	0.528	0.769	0.852	0.397	0.713	0.783
kNN	0.203	0.601	0.641	0.594	0.800	0.788	0.576	0.793	0.784	0.468	0.713	0.735
Árboles de decisión	0.325	0.680	0.666	0.370	0.686	0.695	0.450	0.720	0.758	0.475	0.736	0.770
Random forest	0.436	0.747	0.904	0.616	0.810	0.909	0.664	0.834	0.921	0.548	0.773	0.822
Redes neuronales	0.301	0.611	0.756	0.198	0.656	0.659	0.017	0.599	0.656	0.000	0.600	0.641
SVM lineal	0.251	0.661	0.848	0.484	0.751	0.865	0.404	0.716	0.818	0.215	0.657	0.766
SVM radial	0.625	0.801	0.899	0.480	0.753	0.843	0.468	0.744	0.821	0.562	0.790	0.795
XGBoost	0.429	0.714	0.781	0.513	0.756	0.834	0.589	0.791	0.872	0.560	0.777	0.806

En resumen, destaca el rendimiento de random forest, seguido de SVM radial y XGBoost, dependiendo del método utilizado para seleccionar características. Concretamente, el modelo que mejor rendimiento tiene es random forest utilizando Boruta como método de reducción de la dimensionalidad, como se puede ver en la Figura 22.

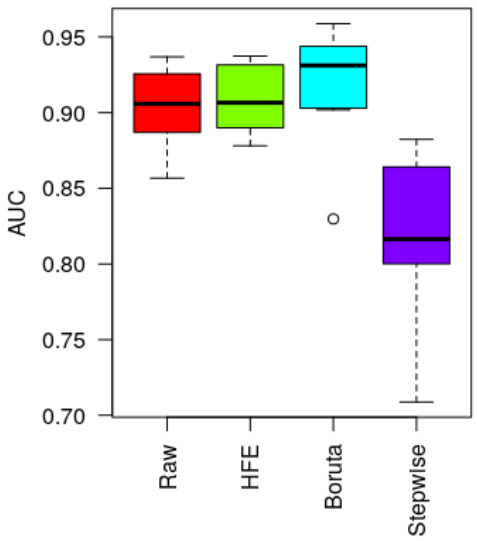

Figura 22. Boxplot comparando los valores de AUC con diferentes métodos de selección de variables.
En rojo si no se ha utilizado ningún método para reducir la dimensionalidad, verde con HFE, azul con Boruta y morado con *forward selection*.

El proceso de reducción de la dimensionalidad no solo mejora el resultado de algunos modelos si no que reduce drásticamente el tiempo necesario para el ajuste de los modelos en todos los casos, como se puede observar en la Figura 23.

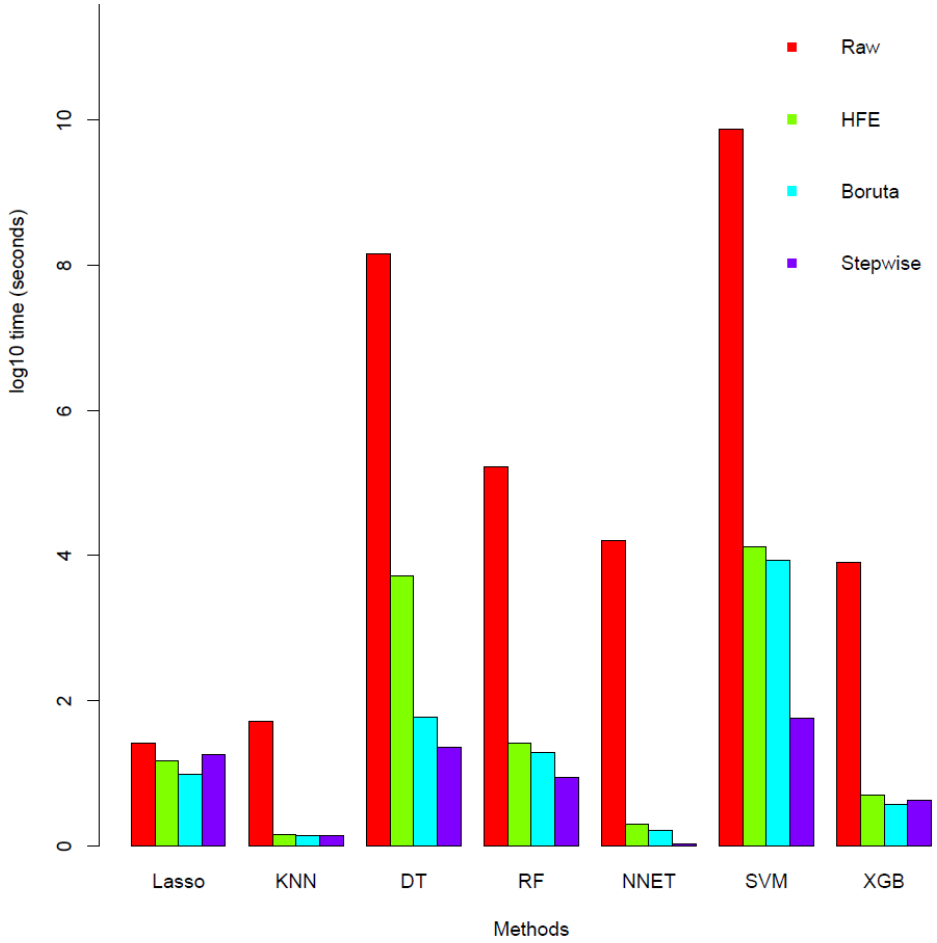

Figura 23. Comparación del tiempo de ajuste de los modelos según el método de selección de variables.
En rojo si no se ha utilizado ningún método para reducir la dimensionalidad, verde con HFE, azul con Boruta y morado con *forward selection*.

En vista de los resultados, para el desarrollo de la aplicación web con Shiny, se van a integrar los 3 modelos con los que se han obtenido los mejores resultados: random forest, SVM radial y XGBoost.

4. Desarrollo de una aplicación interactiva que engloba los modelos más relevantes.

Para hacer más accesible el estudio de técnicas de clasificación sobre datos de asignación taxonómica se ha desarrollado una sencilla aplicación en Shiny, un marco de trabajo que ofrece R enfocado a la creación de aplicaciones web. Previamente, se han desarrollado aplicaciones utilizando Shiny en el campo de la farmacometría [43] o la microbiología [44]. Este tipo de aplicaciones interactivas permiten al usuario introducir unos datos de entrada y elegir entre diferentes opciones para obtener la salida deseada.

Dado el tiempo disponible para el desarrollo de la aplicación, se han añadido opciones básicas y la posibilidad de entrenar tres modelos. La aplicación está alojada en la siguiente página web: https://evacastillo.shinyapps.io/host_train_pred/.

En la página inicial se invita introducir una tabla de OTUs o SVs en un formato concreto. El fichero debe ser tabular, con las muestras en filas y las características en columnas y las clases a las que pertenecen las muestras deben ser de tipo cadena de caracteres. Una vez se han cargado los datos de entrada se mostrará la estructura de los datos de las 10 primeras variables, ya que la dimensión de los datos dificulta su visualización completa. También se imprime un resumen de las variables, dependiendo este del tipo de datos que contenga cada característica. En la Figura 24 se muestra la apariencia de estos resultados con unos datos de ejemplo.

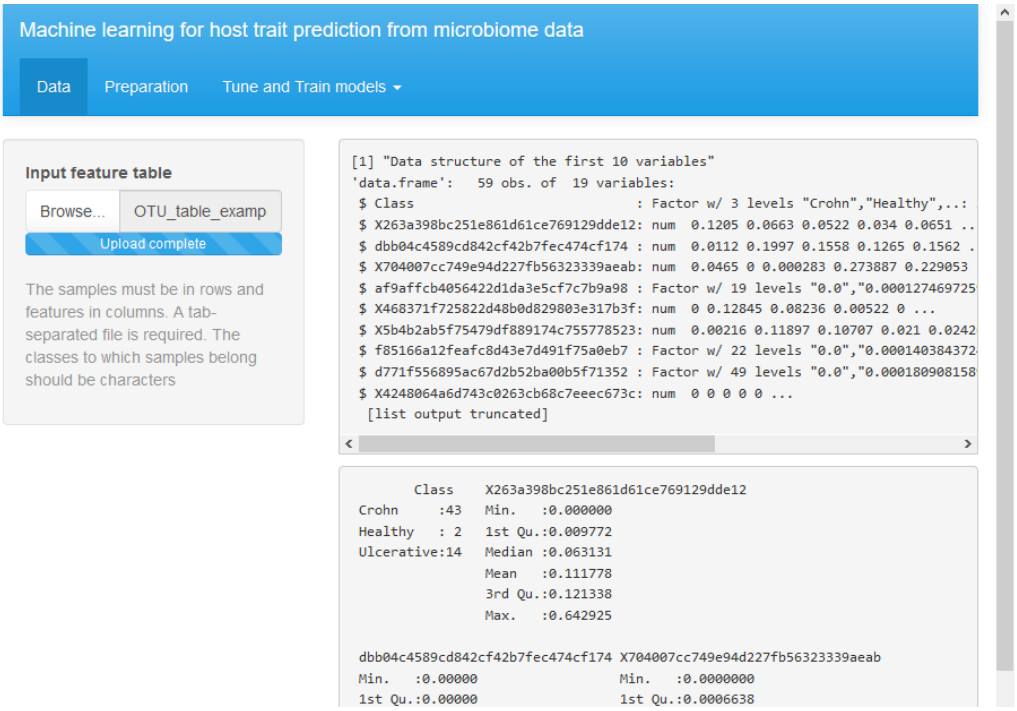

Figura 24. Captura de pantalla de la apariencia de la aplicación tras introducir los datos de entrada.

Se han creado otras dos pestañas que permiten la preparación de los datos y el entrenamiento y tuneado de los 3 modelos previamente indicados. En la preparación de los datos se incluye la opción de hacer validación cruzada de Monte Carlo eligiendo el número de iteraciones y el porcentaje de los datos que van a pertenecer al conjunto de entrenamiento. En la Figura 25 se puede ver la salida obtenida una vez se ha presionado el botón *Submit*, con un mensaje indicando que se ha completado el proceso y un gráfico de densidad de cada uno de los índices en el conjunto de validación cruzada.

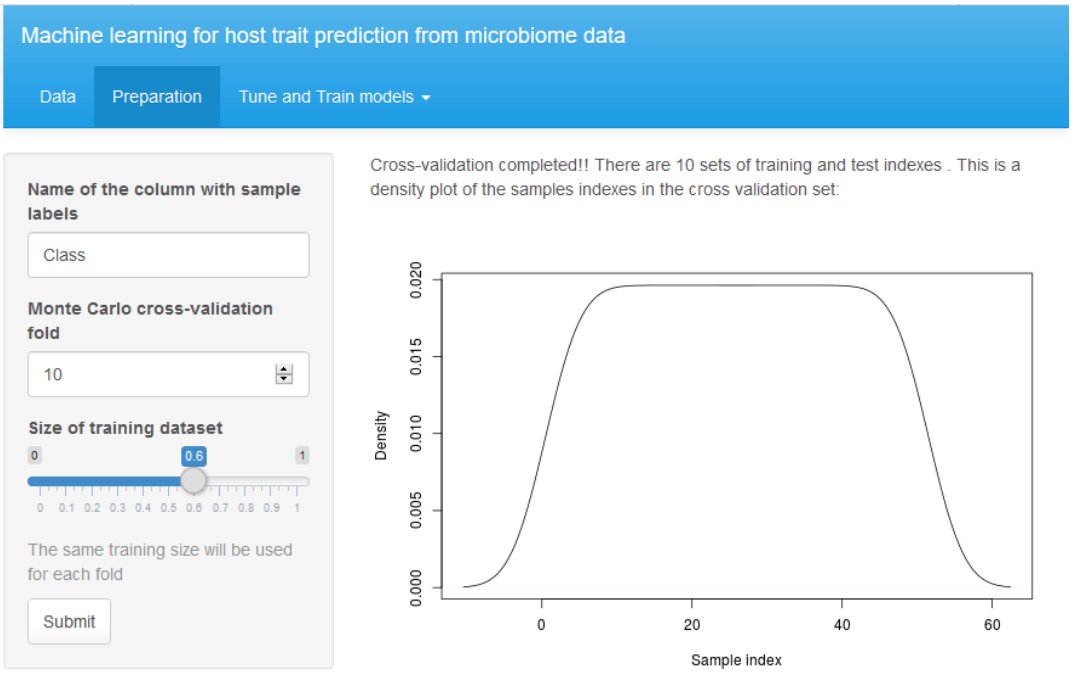

Figura 25. Captura de pantalla de la apariencia de la aplicación en la división de datos con validación cruzada de Monte Carlo.

Por último, para cada uno de los modelos incoporados en la aplicación, se pueden modificar los parámetros seleccionados en este trabajo. En el resultado de los modelos se imprimen una serie de características del modelo, como son el tiempo de tuneado y entrenamiento, el mejor o mejores valores de los hiperparámetros elegidos y una tabla resumen con las medidas de rendimiento para cada iteración. Un ejemplo de la visualización de todos estos resultados se puede observar en la Figura 26. Además, se incluyen gráficos del proceso de selección de los parámetros óptimos, como se indica en un ejemplo en la Figura 27.

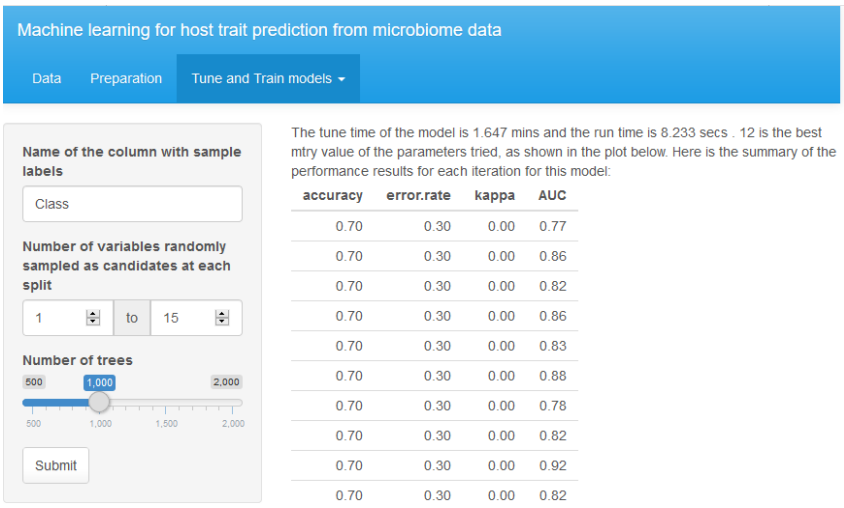

Figura 26. Captura de pantalla de la apariencia de la aplicación tras el tuneado y entrenamiento de un modelo de random forest.

La aplicación web está alojada en los servidores proporcionados de forma gratuita por shiny, por ello el tiempo de ejecución es mayor. Se han utilizado

datos de ejemplo con menos características ya que el tamaño del archivo de entrada es limitado.

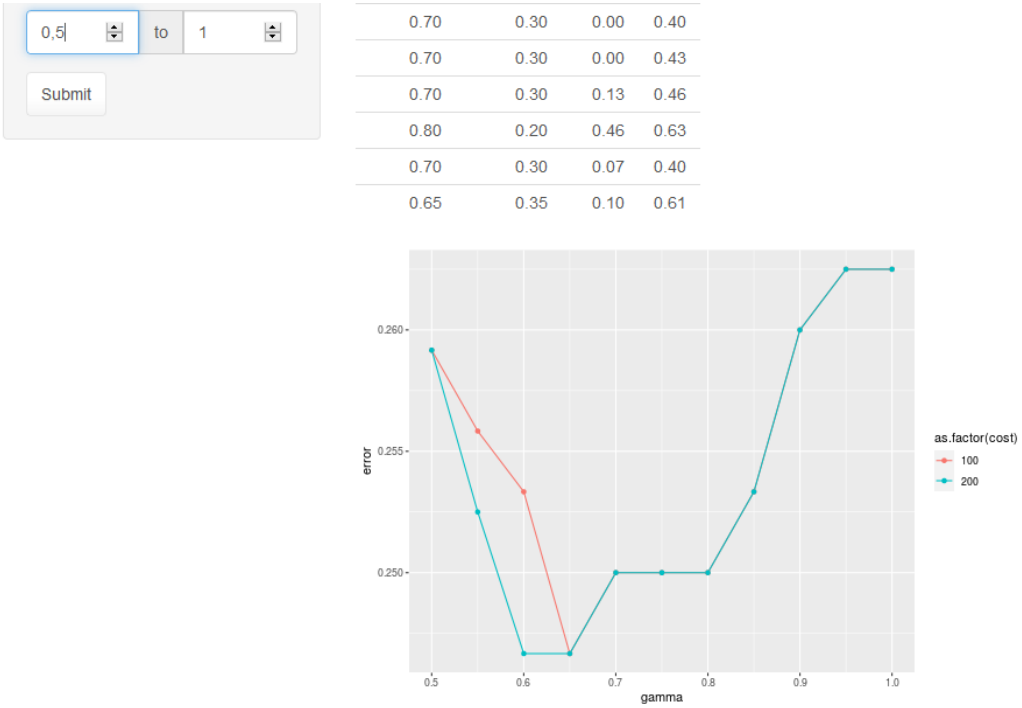

Figura 27. Captura de pantalla de la apariencia de la aplicación tras el tuneado y entrenamiento de un modelo de SVM radial.

5. Conclusiones

En este trabajo se ha presentado un flujo de trabajo para obtener una tabla de SVs a partir de lecturas de la región 16S ARNr de muestras de pacientes con distintas presentaciones de la enfermedad inflamatoria intestinal. Además, se han creado e implementado funciones para aplicar métodos de aprendizaje automático a datos de microbioma y se ha presentado una aplicación web para el uso de algunos de estos algoritmos.

En el análisis de la diversidad microbiana, se ha detallado paso a paso el flujo de trabajo para poder llevar a cabo cualquier otro estudio a partir de muestras del 16S ARNr utilizando diversas herramientas incluidas en QIIME2. A pesar de que la finalidad de este trabajo no es definir las bacterias diferencialmente presentes entre pacientes con IBD y sanos, se han encontrado en los gráficos y en las tablas de abundancia relativas géneros con poca o nula representación en algunos grupos. Este es el caso de *Bacteroides fragilis*, cuya abundancia es mucho menor en las muestras de pacientes sanos. Esta bacteria ha sido identificada en otros estudios en la mucosa de muestras luminales de pacientes con cáncer colorrectal, siendo definida como un miembro de la microbiota contribuyente a la carcinogénesis del colon [45]. También son menos abundantes en pacientes sanos bacterias del género *Paenibacillus*, que se han visto relacionadas con infecciones oportunistas, aunque no se ha podido establecer una relación clara entre infección y enfermedad [46]. Por otra parte, se han encontrado bacterias de los géneros *Halomonas* y *Shewanella* con una abundancia relativa mayor en pacientes con enfermedad de Crohn. Pero, por el momento, no hay estudios que relacionen a bacterias de estos géneros directamente con enfermedades relacionadas con el tracto gastrointestinal. De hecho, se ha descrito que el género *Shewanella* genera una respuesta reducida de neutrófilos produciendo un ambiente inflamatorio bajo en la cirrosis [47]. El efecto patogénico o protector de estas bacterias enriquecidas en mucosa inflamada está aún por descifrar [7].

Los algoritmos de clasificación con resultados más prometedores son random forest, SVM, XGBoost y lasso, dependiendo de si se aplica reducción de características y qué método es utilizado. Sin duda, destaca el potencial predictor de random forest donde, implementando los valores de dos parámetros sencillos, se obtienen muy buenos resultados en todos los casos. También cabe señalar la capacidad predictiva de modelos lineales como SVM lineal o lasso. Como ya se ha señalado en otros estudios [4], la complejidad de los modelos no va a la par de su rendimiento. Modelos más complejos de interpretar como XGBoost en algunos casos tienen una capacidad predictiva igual o incluso menor que algoritmos lineales. Aunque las técnicas de *machine learning* desarrolladas tienen una buena capacidad predictiva, sería importante interpretar los resultados combinándolos con conocimiento molecular y biológico de la enfermedad para entender el significado de las predicciones.

Se ha demostrado el potencial de reducir de la dimensionalidad de los datos. Tanto en el aumento de la precisión y el valor predictivo de los modelos como en la disminución del coste computacional. Es especialmente importante si

partimos de información taxonómica ya que hay muchas variables redundantes que no aportan información adicional al modelo. Aplicar este tipo de técnicas es tan crucial como elegir un método adecuado para realizarlas. En este caso Boruta es capaz de extraer la información importante a partir de datos de alta dimensionalidad, mitigando el ruido y la redundancia de la tabla de SVs. Y, para ello, utiliza algo tan sencillo como elegir las variantes más importantes tras entrenar un clasificador de random forest de forma iterativa, y descartar aquellas menos relevantes.
Adicionalmente, la selección de características basada en información taxonómica, con el método HFE, resulta en la mejora del rendimiento de ciertos modelos. Este método agrupa algunos de los SVs a niveles taxonómicos superiores para eliminar la redundancia en las características. Partir de datos de abundancia de taxones mayores como Especie, Género o Familia podría plantearse como línea de estudio futura.

La creación de funciones para el tuneado y entrenamiento de modelos y su disponibilidad en repositorios públicos de código es una herramienta muy útil para la comunidad científica. Pero dichas funciones solo se pueden aplicar si se tienen conocimientos de programación, en este caso en R, para poder entender los pasos seguidos y poder modificar lo necesario para llegar al resultado deseado. Esto genera una brecha para investigadores que no se han formado o están empezando en el campo de la bioinformática. Por ello, Shiny ofrece la oportunidad de desarrollar aplicaciones web interactivas que poner a disposición de toda la comunidad científica. La aplicación desarrollada en este trabajo es un mero ejemplo de lo que se podría lograr con más tiempo disponible. Esto es, un punto de partida para todo investigador que quiera conocer el campo del *machine learning* y los algoritmos de clasificación aplicados a datos de microbioma. Existen otro tipo de plataformas, como Galaxy [48], que permiten el desarrollo de aplicaciones web, aunque requieren de la creación de un usuario con privilegios de administrador. Mientras que Shiny permite la implementación de aplicaciones a cualquier tipo de usuario de R que tenga una cuenta en la página web de aplicaciones de Shiny. Como desventaja, dado que la aplicación funciona en los servidores gratuitos que ofrece Shiny, la velocidad de entrenamiento de los modelos es muy baja, llegando a veces a saturar el tiempo de espera de la página. Adicionalmente, no permite la introducción de datos de gran tamaño, lo que puede suponer un gran problema con el tipo de datos utilizados para este trabajo. Para mejorar el rendimiento de la aplicación sería necesario contratar una mejora de la cuenta.

Los factores de riesgo planteados al inicio no han impedido llegar a los objetivos y cumplir las actividades y tareas propuestas, aunque el desequilibrio de las muestras ha podido ser un factor limitante para el rendimiento de algunos modelos. Asimismo, el número de muestras sobre el que realizar el análisis puede ser pequeño para modelos como las redes neuronales, que requieren de grandes conjuntos de datos para un mejor funcionamiento. El tiempo disponible para desarrollar la aplicación ha llevado a la creación de una web más sencilla. Se ha generado una interfaz básica sobre la que poder añadir modelos y mejoras, sobre todo a la hora de visualizar los resultados. Y, por último, el poder computacional junto con el tiempo disponible no ha permitido aumentar el número de iteraciones en el proceso de validación

cruzada. En las Figuras 17-21 se puede observar que los intervalos de los valores de AUC obtenidos son amplios en algunos casos. Aumentando en 50 o 100 el número de repeticiones, los resultados serían más robustos. En una futura línea de trabajo se puede plantear variar tanto el número de repeticiones como el porcentaje de datos de entrenamiento elegidos en la división, como indican otros estudios [49].

Aunque se han alcanzado todos los objetivos planteados en este trabajo, hay margen de desarrollo en el análisis de algoritmos de clasificación y en la aplicación web. Los pasos de entrenamiento, evaluación y posterior optimización de los modelos requieren de un proceso de prueba y error en el que se han tenido que incluir o eliminar parámetros para conseguir los mejores rendimientos en cada caso. En esta dirección, sería interesante la creación de una única función para implementar los modelos más prometedores para datos de microbioma, en la que poder elegir los valores de los hiperparámetros a probar.

En vistas a líneas futuras de trabajo, es necesaria la estandarización de todos los procesos hasta llegar a la tabla de SVs u OTUs sobre la que aplicar técnicas de aprendizaje supervisado. Debido a los diferentes métodos de extracción de las muestras, secuenciación o tratamiento bioinformático de los datos, existe un problema de generalización de los modelos optimizados con unos datos concretos [50]. Es importante llegar a un consenso para obtener conjuntos de datos comparables y trasladar este trabajo a un estudio mayor utilizando datos de secuenciación almacenados en repositorios públicos. Estos nuevos datos permitirían la validación de los modelos planteados para que puedan ser ampliamente aceptados y posteriormente utilizados en la clínica.

6. Glosario

- **ARNr 16S**: componente de la subunidad menor (30S) de los ribosomas de procariotas. Los genes que codifican esta región son de gran utilidad para asignar y clasificar a diferentes niveles taxonómicos.
- **Unidad Taxonómica Operativa**: unidad de clasificación de grupos de individuos que están estrechamente relacionados.
- ***Phred***: medida de la calidad en la identificación de nucleótidos por secuenciación masiva.
- **Secuencias quimera**: secuencias que no son productos reales de la amplificación de un gen sino mezcla de otras secuencias.
- **PCoA**: método utilizado para obtener una representación geométrica de individuos basada en una matriz de similaridad.
- **Curva de rarefacción**: representan la relación entre el número de especies observadas y el número de individuos en las muestras. Si las curvas alcanzan una asíntota, el muestreo es acorde al número de especies identificadas.
- **Factor**: tipo de datos estadístico utilizado en R que almacena variables cualitativas o categóricas.
- **Hiperparámetro**: variables ajustables cuyo valor se elige para el entrenamiento de un modelo.
- **OOB (out-of-bag error)**: error medio en la predicción utilizando submuestras de entrenamiento para la predicción y el resto para construir el árbol.

7. Bibliografía

[1] Khan, I., Ullah, N., Zha, L., Bai, Y., Khan, A., Zhao, T., Che, T., & Zhang, C. (2019). Alteration of Gut Microbiota in Inflammatory Bowel Disease (IBD): Cause or Consequence? IBD Treatment Targeting the Gut Microbiome. *Pathogens (Basel, Switzerland)*, *8*(3), 126. https://doi.org/10.3390/pathogens8030126.

[2] Xavier, R. J., & Podolsky, D. K. (2007). Unravelling the pathogenesis of inflammatory bowel disease. *Nature*, *448*(7152), 427–434. https://doi.org/10.1038/nature06005.

[3] Qu, K., Guo, F., Liu, X., Lin, Y., & Zou, Q. (2019). Application of Machine Learning in Microbiology. *Frontiers in microbiology*, 10, 827. https://doi.org/10.3389/fmicb.2019.00827

[4] Zhou, Y.H., Gallins, P. (2019). A Review and Tutorial of Machine Learning Methods for Microbiome Host Trait Prediction. *Front Genet*. 10:579. https://doi.org/10.3389/fgene.2019.00579.

[5] Topçuoğlu, B.D., Lesniak, N.A., Ruffin, M., Wiens, J., Schloss, P.D. (2019). Effective application of machine learning to microbiome-based classification problems. *BioRxiv* 816090. https://doi.org/10.1101/816090.

[6] European Nucleotide Archive (2020). *European Nucleotide Archive Browser*. Recuperado de https://www.ebi.ac.uk/ena/browser/view/PRJNA431126.

[7] Weng, Y.J., Gan, H.Y., Li, X., Huang, Y., Li, Z.C., Deng, H.M., Chen, S.Z., Zhou, Y. *et al*. (2019). Correlation of diet, microbiota and metabolite networks in inflammatory bowel disease. *J Dig Dis*. Sep;20(9):447-459. https://doi.org/10.1111/1751-2980.12795.

[8] Bolyen, E., Rideout, J.R., Dillon, M.R. *et al.* (2019). Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. *Nat Biotechnol* 37**,** 852–857. https://doi.org/10.1038/s41587-019-0209-9.

[9] Prodan, A., Tremaroli, V., Brolin, H., Zwinderman, A. H., Nieuwdorp, M., & Levin, E. (2020). Comparing bioinformatic pipelines for microbial 16S rRNA amplicon sequencing. *PloS one*, *15*(1), e0227434. https://doi.org/10.1371/journal.pone.0227434.

[10] Callahan, B. J., McMurdie, P. J., Rosen, M. J., Han, A. W., Johnson, A. J., & Holmes, S. P. (2016). DADA2: High-resolution sample inference from Illumina amplicon data. *Nature methods*, *13*(7), 581–583. https://doi.org/10.1038/nmeth.3869.

[11] Kuhn, M. (2008). Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, 28(5), 1 - 26. http://dx.doi.org/10.18637/jss.v028.i05

[12] Chang, W., Cheng, J., Allaire, J.J., Xie, Y., McPherson, J. (2019). shiny: Web Application Framework for R. R package version 1.4.0. https://CRAN.R-project.org/package=shiny

[13] Andrews, S. (2010). FASTQC. A quality control tool for high throughput sequence data.

[14] Callahan, B. J., McMurdie, P. J., Rosen, M. J., Han, A. W., Johnson, A. J., & Holmes, S. P. (2016). DADA2: High-resolution sample inference from Illumina amplicon data. *Nature methods*, 13(7), 581-3.

[15] Katoh, K., Standley, D. M. (2013). MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molecular biology and evolution*, *30*(4), 772–780. https://doi.org/10.1093/molbev/mst010.

[16] Price, M. N., Dehal, P. S., & Arkin, A. P. (2010). FastTree 2--approximately maximum-likelihood trees for large alignments. *PloS one*, *5*(3), e9490. https://doi.org/10.1371/journal.pone.0009490.

[17] Vázquez-Baeza, Y., Gonzalez, A., Smarr, L., McDonald, D., Morton, J. T., Navas-Molina, J. A., & Knight, R. (2017). Bringing the Dynamic Microbiome to Life with Animations. *Cell host & microbe*, 21(1), 7–10. https://doi.org/10.1016/j.chom.2016.12.009.

[18] DeSantis, T. Z., Hugenholtz, P., Larsen, N., Rojas, M., Brodie, E. L., Keller, K., Huber, T., Dalevi, D., Hu, P., & Andersen, G. L. (2006). Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Applied and environmental microbiology*, 72(7), 5069–5072. https://doi.org/10.1128/AEM.03006-05.

[19] Santamaria, M., Fosso, B., Consiglio, A., De Caro, G., Grillo, G., Licciulli, F., Liuni, S., Marzano, M., Alonso-Alemany, D., Valiente, G., & Pesole, G. (2012). Reference databases for taxonomic assignment in metagenomics. *Briefings in bioinformatics*, *13*(6), 682–695. https://doi.org/10.1093/bib/bbs036.

[20] Bokulich, N.A., Kaehler, B.D., Rideout, J.R. *et al.* (2018). Optimizing taxonomic classification of marker-gene amplicon sequences with QIIME 2's q2-feature-classifier plugin. *Microbiome* 6**,** 90. https://doi.org/10.1186/s40168-018-0470-z.

[21] Xu, QS., Liang, YZ. (2001). Monte Carlo cross validation. *Chemometr. Intell. Lab. Syst*. 56 (1), 1–11. https://doi.org/10.1016/S0169-7439(00)00122-2.

[22] Friedman, J., Hastie, T., Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1-22.

[23] Lesmeister, C. (2015). *Mastering Machine Learning with R: master machine learning techniques with R to deliver complex and robust projects*. Birmingham: Packt Publishing.

[24] Schliep, K., Hechenbichler, K (2016). kknn: Weighted k-Neares Neighbors. R package version 1.3.1. https://CRAN.R-project.org/package=kknn.

[25] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F. (2019). Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-3. https://CRAN.R-project.org/package=e1071.

[26] Therneau, T., Atkinson, B. (2019). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-15. https://CRAN.R-project.org/package=rpart.

[27] Liaw, A., Wiener, A. (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

[28] Fritsch, S., Guenther, F., Wright, M.N. (2019). neuralnet: Training of Neuronal Networks. R package version 1.44.2. https://CRAN.R-project.org/package=neuralnet.

[29] Bang, S., Yoo, D., Kim, S. J., Jhang, S., Cho, S., & Kim, H. (2019). Establishment and evaluation of prediction model for multiple disease classification based on gut microbial data. *Scientific reports*, 9(1), 10189. https://doi.org/10.1038/s41598-019-46249-x.

[30] Chen, T., He, T., Benesty, M., Khotilovich V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., Li Y. (2020). xgboost: Extreme Gradient Boosting. R package version 1.0.0.2. https://CRAN.R-project.org/package=xgboost.

[31] Kuhn, M. (2020). caret: Classification and Regression Training. R package version 6.0-86. https://CRAN.R-project.org/package=caret.

[32] McHugh, M.L. (2012). Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3), 276–282.

[33] Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.C., Müller, M. (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves*. BMC Bioinformatics*, 12, p. 77. https://doi.org/10.1186/1471-2105-12-77.

[34] Schliep, K. P. (2004). Weighted k-nearest-neighbor techniques and ordinal classification. *Open Access LMU*. https://doi.org/10.5282/ubm/epub.1769.

[35] Probst, P., Wright, M., Boulesteix, A.L. (2018). Hyperparameters and Tuning Strategies for Random Forest. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 10.1002/widm.1301.

[36] Hsu, C.W., Chang, C.C. & Lin, C.J. (2003). A Practical Guide to Support Vector Classification (). Department of Computer Science, National Taiwan University.

[37] Ben-Hur, A., Weston, J. (2010). A user's guide to support vector machines. *Methods Mol Biol*. 2010;609: 223–239. 10.1007/978-1-60327-241-4_13.

[38] Lantz, B. (2015). *Machine Learning with R: Expert techniques for predictive modeling to solve all your data analysis problems*. Birmingham: Packt Publishing.

[39] Heaton, J. (2008). *Introduction to Neuronal Networks for Java*. Chesterfield, MO: Heaton Research.

[40] Peter, B., Andrew, B. (2017). *Practical Statistics For Data Scientists: 50 essential concepts*. USA: O'Reilly Media, Inc.

[41] Oudah, M., Henschel, A. (2018). Taxonomy-aware feature engineering for microbiome classification. *BMC Bioinformatics* 19, 227. https://doi.org/10.1186/s12859-018-2205-3.

[42] Kursa, M. B., Rudnicki, W. R. (2010). Feature Selection with the Boruta Package. *Journal of Statistical Software* 36(11), 1-13.

[43] Wojciechowski, J., Hopkins, A. M., & Upton, R. N. (2015). Interactive Pharmacometric Applications Using R and the Shiny Package. *CPT: pharmacometrics & systems pharmacology*, 4(3), e00021. https://doi.org/10.1002/psp4.21.

[44] McMurdie, P. J., & Holmes, S. (2015). Shiny-phyloseq: Web application for interactive microbiome analysis with provenance tracking. *Bioinformatics*, 31(2), 282–283. https://doi.org/10.1093/bioinformatics/btu616.

[45] Boleij, A., Hechenbleikner, E. M., Goodwin, A. C., Badani, R., Stein, E. M., Lazarev, M. G., Ellis, B., Carroll, K. C., Albesiano, E., Wick, E. C., Platz, E. A., Pardoll, D. M., & Sears, C. L. (2015). The Bacteroides fragilis toxin gene is prevalent in the colon mucosa of colorectal cancer patients. *Clinical infectious diseases*: an official publication of the Infectious Diseases Society of America, 60(2), 208–215. https://doi.org/10.1093/cid/ciu787.

[46] Grady, E. N., MacDonald, J., Liu, L., Richman, A., & Yuan, Z. C. (2016). Current knowledge and perspectives of Paenibacillus: a review. *Microbial cell factories*, *15*(1), 203. https://doi.org/10.1186/s12934-016-0603-7.

[47] Giannelli, V., Di Gregorio, V., Iebba, V., Giusto, M., Schippa, S., Merli, M., & Thalheimer, U. (2014). Microbiota and the gut-liver axis: bacterial translocation, inflammation and infection in cirrhosis. *World journal of gastroenterology*, *20*(45), 16795–16810. https://doi.org/10.3748/wjg.v20.i45.16795.

[48] Afgan, E., Baker, D., Batut, B., van den Beek, M., Bouvier, D., Cech, M., Chilton, J., Clements, D., Coraor, N., Grüning, B. A., Guerler, A., Hillman-Jackson, J., Hiltemann, S., Jalili, V., Rasche, H., Soranzo, N., Goecks, J., Taylor, J., Nekrutenko, A., & Blankenberg, D. (2018). The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic acids research*, 46(W1), W537–W544. https://doi.org/10.1093/nar/gky379.

[49] Beschorner, A., Voigt, M., Vogeler, K. (2014). Monte carlo cross validation for response surface benchmark. Proceedings of the 12th International Probabilistic Workshop, 2014. https://doi.org/10.1466/20141125.01.

[50] Sakowski, E., Uritskiy, G., Cooper, R., Gomes, M., McLaren, M. R., Meisel, J. S., Mickol, R. L., Mintz, C. D., Mongodin, E. F., Pop, M., Rahman, M. A., Sanchez, A., Timp, W., Vela, J. D., Wolz, C. M., Zackular, J. P., Chopyk, J., Commichaux, S., Davis, M., Dluzen, D., *et a*l. (2019). Current State of and Future Opportunities for Prediction in Microbiome Research: Report from the Mid-Atlantic Microbiome Meet-up in Baltimore on 9 January 2019. *mSystems*, 4(5), e00392-19. https://doi.org/10.1128/mSystems.00392-19.

8. Anexos

8.1. Anexo I.

Líneas de código utilizadas para el análisis de la microbiota dentro de la plataforma de QIIME 2™.

```
$ conda activate qiime2-2019.10

$ qiime tools import --type SampleData[PairedEndSequencesWithQuality] --input-path manifest.csv --output-path paired_end_reads.qza --input-format PairedEndFastqManifestPhred33

$ qiime dada2 denoise-paired --i-demultiplexed-seqs paired_end_reads.qza --p-trunc-len-f 0 --p-trunc-len-r 0 --p-trim-left-f 50 --p-trim-left-r 50 --o-table feature_table.qza --o-representative-sequences rep-seqs.qza --o-denoising-stats stats.qza –verbose

$ qiime metadata tabulate --m-input-file stats.qza --o-visualization stats.qzv

$ qiime feature-table summarize --i-table feature_table.qza --o-visualization feature_table.qzv --m-sample-metadata-file map_file.txt

$ qiime feature-table tabulate-seqs --i-data rep-seqs.qza --o-visualization rep-seqs.qzv

$ qiime tools view file.qzv

$ qiime feature-table relative-frequency --i-table feature_table.qza --o-relative-frequency-table feature_table_relative.qza

$ qiime phylogeny align-to-tree-mafft-fasttree --i-sequences rep-seqs.qza --o-alignment aligned-rep-seqs.qza --o-masked-alignment masked-aligned-rep-seqs.qza --o-tree unrooted-tree.qza --o-rooted-tree rooted-tree.qza

$ qiime diversity core-metrics-phylogenetic --i-phylogeny rooted-tree.qza --i-table feature_table.qza --p-sampling-depth 28456 --m-metadata-file map_file.txt --output-dir core-metrics-results

$ qiime diversity alpha-rarefaction --i-table feature_table.qza --i-phylogeny rooted-tree.qza --p-max-depth 97112 --m-metadata-file map_file.txt --o-visualization alpha-rarefaction.qzv

$ qiime tools import --type FeatureData[Sequence] --input-path 97_otus.fasta --output-path rep_gg97.qza

$ qiime tools import --type FeatureData[Taxonomy] --input-path 97_otu_taxonomy.txt --output-path taxonomy_gg97.qza --input-format HeaderlessTSVTaxonomyFormat

$ qiime feature-classifier classify-consensus-blast --i-query rep-seqs.qza --i-reference-taxonomy taxonomy_gg97.qza --i-reference-reads rep_gg97.qza --o-classification taxonomy.qza
```

```
$ qiime metadata tabulate --m-input-file taxonomy.qza --o-visualization taxonomy.qzv

$ qiime taxa barplot --i-table feature_table.qza --i-taxonomy taxonomy.qza --m-metadata-file map_file.txt --o-visualization taxa-bar-plots.qzv

$ qiime feature-table group --i-table feature_table.qza --p-axis sample --m-metadata-file map_file.txt --m-metadata-column State --p-mode mean-ceiling --o-grouped-table grouped_feature_table.qza

$ qiime tools export --input-path .qza --output-path output
```