Desarrollo de una GUI para el análisis y modelización de problemas de clasificación

Susana Hernández Ballesteros
Máster en Bioinformática y Bioestadística
Área 1

Nombre Consultor/a
Antonio Jesús Adsuar Gómez

Fecha Entrega 4 de junio de 2019

Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento 3.0 España de Creative Commons

B) GNU Free Documentation License (GNU FDL)

Copyright © AÑO TU-NOMBRE.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desarrollo de una GUI para el análisis y modelización de problemas de clasificación*
Nombre del autor:	*Susana Hernández Ballesteros*
Nombre del consultor/a:	*Antonio Jesús Adsuar Gómez*
Nombre del PRA:	*Antonio Jesús Adsuar Gómez*
Fecha de entrega (mm/aaaa):	06/2019
Titulación:	*Master en Bioinformática y Bioestadística*
Área del Trabajo Final:	*Area 1*
Idioma del trabajo:	*Español*
Palabras clave	*Clasificación, machine learning, shiny*

Resumen del Trabajo (máximo 250 palabras):

El problema de clasificación de machine learning en general es uno de los problemas mas comunes del mundo de la investigación y en concreto en el ámbito de la bioinformática. Ser capaces de determinar la clase de un conjunto de inputs biológicos dentro de un grupo de opciones desempeña un importante papel. Sin embargo, para realizar estos análisis se requiere que el usuario posea un cierto conocimiento de programación y conocimientos sobre los distintos modelos disponibles.

Este trabajo pretende crear una aplicación web que permita a usuarios que no dispongan de los conocimientos suficientes de programación, realizar un pipeline completo de análisis de datos con modelos de machine learning con objetivo de clasificación. Para ello, se desarrolla un pipeline en el lenguaje de programación R que marcará la ruta a seguir para el análisis y, posteriormente, se implementará este pipeline en una herramienta web creada mediante la herramienta Shiny.

Para comprobar la correcta funcionalidad tanto del pipeline como de la aplicación web, se descargan han utilizado dataset con target binario y multiclase que contengan los datos en uno o dos ficheros, en formato CSV con distintos separadores con variables tanto categóricas como numéricas que permitan realizar el análisis utilizando para ello distintos parámetros.

La herramienta permite cargar los datos, describirlos, separarlos y modelizarlos para determinar qué modelo de clasificación es el más adecuado para aplicar a cada conjunto de datos.

Abstract (in English, 250 words or less):

The problem of classification of machine learning in general is one of the most common problems in the world of research and specifically in the field of bioinformatics. Being able to determine the class of a set of biological inputs within a group of options plays an important role. However, in order to carry out these analyzes, the user must have some knowledge of programming and knowledge about the different models available.

This work aims to create a web application that allows users who do not have sufficient knowledge of programming, to perform a complete pipeline of data analysis with machine learning models for classification purposes. For this, a pipeline is developed in the R programming language that will mark the path to follow for the analysis and, later, this pipeline will be implemented in a web tool created by the Shiny tool.

To check the correct functionality of both the pipeline and the web application, datasets with binary and multiclass target containing the data in one or two files have been used, in CSV format with different separators with both categorical and numeric variables that allow performing the analysis using different parameters.

The tool allows to load the data, describe it, separate them and model them to determine which classification model is the most appropriate to apply to each data set.

AGRADECIMIENTOS

Me gustaría agradecer en primer lugar a mi tutor, *Antonio Jesús Adsuar Gómez*, la oportunidad que me ha brindado de realizar este trabajo. Gracias a su ayuda y consejos se ha conseguido que este Trabajo de Fin de Master llegue a buen puerto.

Así mismo, quería agradecer a todas las personas que se han implicado en la realización de este proyecto, aportando ideas, críticas y mejoras, para que el proyecto se completase de la mejor forma posible y fuese útil a todas aquellas personas a las que está orientado.

Quería acordarme de las personas que han influido en esta decisión de involucrarme en el mundo de la bioinformática, de los que siempre creen en mi cuando yo misma dudo de mis capacidades y de mis amigos que comprenden mi falta de tiempo y dedicación cuando me sobrepasan las horas de trabajo, gracias por seguir siempre a mi lado.

Y por último no me puedo olvidar de los grandes pilares que han soportado este nuevo reto a mi lado, mis padres, José María y Susana, y mi hermana María, quienes desde antes de yo saber que quería sumergirme en esta aventura me han conducido con sus recomendaciones a acercarme a mis sueños y una vez embarcada han luchado conmigo para superar las tormentas de cansancio, las tempestades de agobio y desde el primer día hasta el último me han dado los ánimos, el apoyo y el cariño necesario para poder superar cada uno de los obstáculos con los que me he encontrado. Sin ellos esto no habría sido posible. Gracias de todo corazón por todo, os quiero.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

En esta introducción se describirán los aspectos generales del proyecto, contexto, definición objetivos, tareas, planificación y que es lo que se puede esperar de este documento. Intenta reflejar estado actual del proyecto, el desarrollo de los objetivos especificados, exponer los resultados obtenidos en el TFM.

1.1. Contexto y justificación del Trabajo

El área escogida para el trabajo engloba el desarrollo de herramientas de soporte a la omica, es decir la parte mas de desarrollo informático para el apoyo al análisis de datos omicos al que se pueden enfrentar los profesionales en las áreas de la biología, medicina o bioinformática. Este proyecto pretende mitigar dos problemas, la necesidad de un profundo conocimiento de las herramientas y la falta de un entorno amigable y sencillo para su uso e integración, y los conocimientos de machine learning, procesamiento de datos y selección de variables necesarios para resolver problemas sobre datos omicos. Como resultado de estas observaciones, el objetivo principal del proyecto es desarrollar una pipeline integre los distintos pasos del análisis de datos con las técnicas más comúnmente utilizadas y que permita a un usuario sin conocimientos de informática avanzados el poder hacer uso de algunos de los algoritmos disponibles en el Machine Learning. Ademas, se contempla implementar una interfaz gráfica que abarcara el proceso de carga de datos, análisis descriptivo, preprocesamiento de datos, selección de variables y selección del algoritmo. La problemática que se pretende resolver es la unificación del proceso análisis de datos y de los distintos algoritmos de clasificación disponibles en el estado del arte bajo una herramienta intuitiva, sencilla y accesible a los usuarios de distintos perfiles científicos que trabajan con datos omicos. El análisis de datos omicos, o datos estructurados en general, sigue siendo un gran desafío, especialmente para los investigadores con menos experiencia en bioinformática, software y programación. Los flujos de trabajo de análisis de datos generalmente se componen de muchos procesos que involucran diferentes herramientas de software que demandan diferentes recursos y conocimientos. Por lo tanto, a menudo es difícil realizar un flujo de trabajo completo de manera automatizada, a pesar de que existen muchas herramientas y recursos computacionales disponibles. Con este proyecto se pretende integrar bajo una misma herramienta y lenguaje todos los pasos comunes al análisis de datos creando un pipeline de trabajo intuitivo y generalizado que permita a un usuario sin altos conocimientos de programación y de algoritmia llegar a tener un análisis de sus datos y resultados de predicciones de clasificación. Si bien es cierto que, cuando pensamos en bioinformática, nuestra intuición nos lleva siempre a pensar en datos genómicos, NGS(next generation sequencing) [1], HTS(high throughtput sequencing) [15] pero en este caso queremos centrarnos en conjuntos de datos de cualquier tipo, pudiendo ser datos de pacientes, información celular o cualquier otro conjunto que disponga de una variable objetivo de valor binario o múltiple y que se pretenda la clasificación de cada entrada de datos a una de estas clases. El motivo por el que se realiza este TFM es porque se quiere conseguir la unificación del proceso análisis de datos y de los distintos algoritmos de clasificación disponibles en el estado del arte bajo una herramienta intuitiva, sencilla y accesible a los usuarios de distintos perfiles científicos que trabajan con datos omicos. El análisis de datos omicos, o datos estructurados en general, sigue siendo un gran desafío, especialmente para los investigadores con menos experiencia en bioinformática, software y programación. Dada la gran variedad de métodos de análisis, algoritmos de predicción y flujos de análisis disponibles puede ser un poco abrumador para el usuario inexperto el decidir que metodología usar o en que lenguaje de programación. Con este proyecto se pretende in-

tegrar bajo una misma herramienta todos los pasos comunes al análisis de datos creando un pipeline de trabajo intuitivo y generalizado que permita a un usuario sin altos conocimientos de programación y de algoritmia llegar a tener un análisis de sus datos y resultados de predicciones de clasificación. Se ha elegido este tema debido a su alto contenido de análisis de datos, técnicas descriptivas de datos, transformaciones y predicción, programación y aplicación práctica para la vida diaria de los analistas de datos del campo de la bioinformática y casi cualquiera otra área de *data science*. Ademas, la posibilidad de desarrollar un software en R [17] para la interfaz de análisis permite reforzar los conocimientos adquiridos durante el máster de las librerías mas comunes y la generación de aplicaciones interactivas con Shiny [22],[8].

1.2. Objetivos del trabajo

En líneas generales del proyecto sentaría tres objetivos principales que se desarrollan a continuación:

- Definición de un pipeline de análisis de datos
- Desarrollo de las funciones de cada fase del análisis.
- Implementación e integración de la funcionalidad en una interfaz de usuario.

Estos tres objetivos principales se pueden detallar como se muestra en los siguientes objetivos específicos:

1. Selección de datasets: elegir dos conjuntos sobre los que se realizarán las pruebas.
    1.1. Selección de dataset con target binario
    1.2. Selección de dataset con target multiclase
2. Definición del pipeline de análisis de datos
    2.1. Selección del fichero de datos de entrada carga de datos y almacenamiento en estructura de datos.
    2.2. Selección de la variable objetivo
    2.3. Visualización descriptiva del conjunto de datos. Distribuciones de las variables, resumen estadístico, información de los metadatos [25].
    2.4. Análisis de tipo de variables y transformaciones.
    2.5. Selección manual de variables basada en conocimiento del usuario y análisis descriptivo.
    2.6. Preprocesamiento de los datos.
    2.7. Estandarización y normalización de los datos.
    2.8. Selección de algoritmos de clasificación y establecer las limitaciones de los mismos.
    2.9. Visualización de resultados.
3. Implementación de las funciones asociadas a cada una de las fases anteriores
    3.1. Estudio de las librerías disponibles para la implementación de la funcionalidad.
    3.2. Implementación de las funciones y métodos para cada una de las fases del análisis.
    3.3. Testing de las funciones.
    3.4. Desarrollar el flujo de llamadas a las funciones para creación del pipeline de análisis.
4. Implementación de la interfaz gráfica a través de Shiny dashboard.
    4.1. Diseño de las pantallas.

4.2. Implementación de la interacción entre pantallas.

4.3. Implementación de las acciones y activación de funciones en la interfaz.

4.4. Implementación de la interacción entre funciones y paso de valores.

4.5. Diseño e implementación de presentación de resultados y gráficas.

Definición

Para este proyecto se pretende realizar un análisis descriptivo de datos y un análisis predictivo de tipo clasificación. Como una primera aproximación se considera que los puntos de esta definición son:

- Selección de formato de los datos y establecer un par de conjuntos sobre los que se realizarán las pruebas. Se intentara realizar un proyecto que englobe tanto la clasificación binaria como múltiple por lo que se pretende escoger un conjunto de datos de cada tipo.
- Carga y lectura de datos. Extracción de variables y establecimiento de la variable objetivo. Además, se pretende tener una primera visualización y descripción de conjunto de datos .
- Preprocesamiento de los datos. Esta fase incluirá un análisis de los tipos de las variables y aplicación de filtros y transformaciones para que las variables cumplan los requisitos de los algoritmos de clasificación a utilizar. Con este proceso nos referimos a la eliminación o imputación de nulos, transformaciones de cadenas a valores numéricos o eliminación de variables no informativas o de valor para el análisis.
- Estandarización y normalización de los datos. Para un correcto funcionamiento e interpretación de los datos por parte de los algoritmos de machine learning muchas veces es necesario que los datos sean comparables entre ellos. Si bien algunos algoritmos como los árboles o random forest no necesitan de estas técnicas es recomendable añadirlas al pipeline de análisis para que el usuario disponga de la posibilidad de tener un grid de modelos de clasificación que utilicen el mismo input de datos.
- Algoritmos de clasificación y visualización de resultados. Se pretende incluir un conjunto de algoritmos de clasificación que permitan al usuario seleccionar el mas conveniente para su problema y comparar diferentes resultados.

Desarrollo

El desarrollo del pipeline definido en la sección anterior se quiere desarrollar en R basándonos en las librerías de tratamiento y análisis de datos como son *nnet[18], caret[13], class[19],forcats[29],pRoc [20], gmodels[11], neuralnets, ggplot2[28], rpart[26], kernlab, randomforest[7]* y otras librerías que implementan metodos y algoritmos específicos de machine learning. A partir de esta librería de funciones se podría realizar un análisis sin necesidad de desarrollo de código, simplemente a través de las llamadas a funciones y almacenamiento de resultados.

Integración

Para poder hacer uso de la librería y conjunto de funciones del pipeline se propone el desarrollo de una interfaz web basada en Shinydashboard a través de la cual el usuario pueda acceder a las distintas funcionalidades y visualización de los datos y resultados del análisis.

1.3. Enfoque y método seguido

Se plantean varias estrategias para llevar a cabo el trabajo, pero lo primero que se ha hecho es investigar cuales son los diferentes pasos que se deben seguir en un análisis de datos para la predicción de categorías definir un pipeline. En esta fase se puede incluir la investigación y selección de metodos y

algoritmos concretos a implementar en casa uno de los pasos. Es necesario limitar el *scope* dada la gran variedad de metodologías disponibles. A partir de aquí se pueden tomar varias alternativas:

- Estudio de los algoritmos seleccionados para cada fase como una primera parte teórica. Una vez entendida toda la teoría proceder a la implementación de todas las funciones asociadas a cada fase. Desarrollo de la interfaz gráfica y la interacción entre las distintas partes del pipeline. Realizar *testing* de las funciones y funcionalidad de la herramienta. Por último, escribir la documentación y documento de entrega.

- Realizar el estudio e investigación de los metodos de una fase, y desarrollar las funciones asociadas a la misma a la vez que realizar los test unitarios de dichas funciones y la documentación asociada. De forma paralela ir creando cada una de las pantallas asociadas a la fase del pipeline de la interfaz gráfica de usuario. Por último, hacer los test end to end y finalizar la memoria.

- Una tercera aproximación es una orientación incremental de la funcionalidad de la aplicación. Es decir realizar un pipeline completo y sencillo end to end, realizar las pruebas y la documentación del mismo.Con ello conseguiríamos tener un mínimo producto viable, a partir de ahora MVP, sobre el cual iterar y construir la herramienta e interfaz gráfica final.

La opción elegida para el desarrollo del TFM es la última ya que se crea de forma incremental y en todo momento el objetivo es tener un MVP lo que asegura entregables intermedios con funcionalidad completa, lo que podría definirse como cascada con retroalimentación[21]. Esta opción ayuda al desarrollador a ajustarse a los tiempo, proporcionando casi desde un primer momento la parte práctica y usable del producto centrándose la parte mas compleja y densa al principio del desarrollo del proyecto. Además, esta metodología incremental ayuda trabajar ajustando los tiempos y proporcionando un soporte a la parte practica, que va a ser la parte mas complicada del proyecto, puesto que se tiene que generar un paquete en R y la aplicación web. De esta forma, si por alguna razón fuese imposible analizar todos los posibles pipelines y algoritmos de clasificación encontrados, se podría decidir cuáles entran en el estudio, sin alargar la parte teórica del proyecto y dando el tiempo suficiente para realizar la parte práctica de desarrollo. También da la posibilidad de priorizar los pipelines, metodos y algoritmos de clasificación a analizar, investigando primero los mas utilizados, alguno más complejo, y dejando para los últimos aquellos que ya se sabe por el conocimiento adquirido durante el máster, que no van a ser los óptimos debido a las restricciones o inconvenientes que tienen. De esta forma, si por alguna razón la planificación temporal se ve afectada por algún problema o no fuese posible integrar todos los metodos seleccionados, la herramienta seguiría siendo funcional para algún tipo de análisis y clasificación de datos. También deja margen a realizar la parte mas compleja o mejoras en la totalidad de la funcionalidad al final del proyecto si la planificación temporal lo permite.

1.4. Planificación del Trabajo

Tareas

- Identificación del target del proyecto, pipeline a seguir para el análisis y los datasets a utilizar (Objetivo específico -OE- 1.1 y 1.2 con una duración de 8h -DT-)

- Para cada fase del pipeline se desarrollarán las siguientes subtareas (DT de entre 5 y 10 horas dependiendo de la fase y la complejidad de los métodos):

    - Estudiar los métodos disponibles y su cabida dentro del proyecto (OE 2.x)
    - Implementar las funciones para la utilización de los métodos adaptadas a los datos (OE 3.1, 3.2, 3.3)
    - Integrar las distintas fases del pipeline a medida que se van desarrollando métodos. (OE 3.4)

- Diseño y desarrollo de la interfaz gráfica de usuario (DT 20h-30h para cada pantalla).

    - Diseño del dashboard las interfaz web (OE 4.1).
    - Implementación de los elementos de interacción con las pantallas (OE 4.2)
    - Incluir y asociar las funciones y métodos a las pantallas y a los elementos de interacción (OE 4.3)
    - Implementar la interacción entre las fases del análisis (OE 4.4)
    - Implementación de gráficas y muestra de resultados (OE 4.5)
- Probar y validar el resultado obtenido tanto a nivel usuario como analítico (DT 10h)
- Crear un servicio para acceder a la aplicación en una máquina EC2 de AWS. (DT 20h)

Calendario

En el calendario de planificación por meses del Trabajo de Final de Máster se ha tenido en cuenta que la disponibilidad de la autora es principalmente fines de semana y festivos, y que entre semana la dedicación máxima es de unas 2 horas por día.

En marzo se finaliza con la selección de los datasets, el análisis de los posibles pipelines y selección de los métodos y modelos.

El mes de abril se dedica al desarrollo de las funciones, implementación de los flujos y análisis de datos. En paralelo se empezará a crear el diseño de la interfaz.

En el mes de mayo, se dedica casi exclusivamente a la creación de la aplicación, integración de la funcionalidad y análisis de los resultados Las tareas de junio son las propias de las entregas de las tres últimas pruebas de evaluación continua (PEC): Redacción de la memoria, elaboración de la presentación y defensa pública. En la figura 1.4.1 se presenta el Diagrama de Gantt en el que se puede ver detalladas las tareas, el tiempo de inicio y finalización.

Hitos

Se han analizado las tareas del proyecto y se han marcado los hitos, que son aquellos en los que si hay algún retraso en la tarea esto repercutiría negativamente en los plazos de otras tareas.

Hito	PEC	Fecha Crítica
Definición de proyecto	PEC-0	04-03-2019
Plan de trabajo	PEC-1	20-03-2019
Selección de datos	PEC-2	25-04-2019
Definición de Pipeline	PEC-2	25-04-2019
Implementación de funcionalidad	PEC-2 y 3	20-05-2019
Desarrollo de aplicación Shiny	PEC-2 y 3	20-05-2019
Entrega de memoria	PEC-4	04-06-2019
Elaboración de la presentación	PEC-5a	12-06-2019
Defensa pública	PEC-5b	25-06-2019

Cuadro 1.4.1: Hitos

Análisis de riesgos

Como en todos los proyectos hay factores que afectan negativamente en el planificación y en la ejecución del trabajo, lo que puede ocasionar que no se lleve a cabo el TFM. A continuación se enumeran algunos de los que se pueden encontrar en el estudio planteado:

1. El proyecto se realiza en un tiempo muy ajustado.

2. El TFM englobe muchos posibles pipelines, dificultando llegar al objetivo marcado.

3. Haber realizado mal el plan de trabajo; concretamente en la priorización y el tiempo que se debe dedicar a cada tarea.

4. Utilizar un conjunto de datos que no sean adecuados para testear los algoritmos de clasificación.

5. Falta de conocimiento en la creación de paquetes y desarrollo de interfaces web en R.

6. Depuración y corrección de errores del código en el desarrollo e interacción de los elementos de Shiny.

7. Diferentes versiones del lenguaje de programación y compatibilidades de las librerías utilizadas en el desarrollo del proyecto.

1.5. Productos obtenidos

Una vez se haya finalizado el proyecto se han obtenido los output que se especifican a continuación:

1.6. Plan de trabajo

Es el documento en el que se concreta el trabajo que se va a realizar, delimitando la labor e indicando como se va a proceder. También se especifican los objetivos del TFM, se planifica los hitos y el tiempo dedicado a cada tarea. Además de definir y analizar los posibles riesgos que conlleva el proyecto.

Producto

En este proyecto además de los resultados obtenidos que se presentarán en esta memoria, se ha desarrollado una aplicación web realizada con el programa R y con el paquete de documentos interactivos Shiny. Esta aplicación hace de interfaz gráfica para el análisis y clasificación de datos con target binario o multiclase.

Memoria

El objetivo de la memoria del Trabajo de Final de Máster es documentar el trabajo que se ha hecho. Indicando el contexto por el que se realiza y el motivo por el que es de interés para la sociedad. Exponiendo los objetivos que se pretenden alcanzar, detallando el método seguido para realizar el proyecto y, la parte más importante, exponiendo los resultados obtenidos en el TFM.

Presentación virtual

En esta entrega, se realizará una presentación que resumirá el trabajo y resultados obtenidos en el TFM. Donde se expondrán oral y visualmente la información más importante del proyecto (objetivos, planificación, desarrollo y resultados).

Autoevaluación del proyecto

En la autoevaluación del proyecto se medirá si se han alcanzado los objetivos iniciales. Identificando cuales han sido los motivos por los que se ha conseguido llegar al éxito o por el contrario, no se han superado. Además de realizar unas conclusiones, que tras la experiencia podrían sugerir mejoras en el planteamiento del proyecto para un mejor resultado final.

1.7. Estructuración del proyecto

El proyecto según el Plan Docente está estructurado en los siguientes elementos:

- Una memoria, que documenta el trabajo realizado. Concretamente la memoria constará de una introducción, descripción del problema una exposición de los resultados obtenidos, unas conclusiones, glosario, bibliografía y anexos. Este documento como máximo podrá ocupar 90 páginas y tendrá que estar redactado siguiendo las pautas de los textos académicos.
- Un producto, que en este caso es una aplicación accesible como servicio web y el código de la misma.
- Una presentación, en la que se sintetice los resultados obtenidos y la evolución del trabajo. La presentación será de unas 20 páginas o transparencias, en la que estará incluida la exposición oral (con una duración máxima de 20 minutos). En ella se recopilaran el trabajo realizado y los resultados adquiridos. Además de incluir los aspectos más importantes y una idea de la totalidad del TFM.
- Un informe de autoevaluación. Constará de una comparación de los resultados obtenidos respecto a los planificados, una evaluación de los indicadores de éxito/fracaso del trabajo, junto con explicaciones del no alcance de los objetivos y unas conclusiones del proyecto.
- Una defensa pública del TFM. En la defensa pública del Trabajo de Final del Máster, se defiende el trabajo ante el tribunal.

Además, durante la ejecución del proyecto se han realizado dos informes de seguimiento del proyecto correspondientes a las PEC 2 y 3. Ambos contienen una descripción del avance del trabajo, donde se indica el grado de cumplimiento de los objetivos y resultados según el plan del trabajo y si ha habido cambios una justificación. También contienen una relación de las actividades realizadas y, en caso de

ser necesario, una justificación de la actualización de la planificación temporal y acciones realizadas, junto con un resumen de los resultados obtenidos hasta ese momento.

1.8. Descripción del resto de capítulos

A continuación se resume el resto del contenido de este documento:

El problema de clasificación

En este capítulo se pretende dar una idea general, definición e ideas claves de qué es y en qué consiste la clasificación en este contexto del machine learning, que problemas puede resolver y cual es el *scope* dentro de este TFM.

Modelos de clasificación seleccionados

En este capítulo se hace una descripción de los modelos de machine learning de clasificación seleccionados, sus principales características, ventajas e inconvenientes.

Descripción de los datos

En este capítulo se hace una descripción de los requisitos de los datos para hacer uso en la aplicación y se describen 2 ejemplos de datasets seleccionados.

Pipeline

Este capítulo recoge cada una de las fases del análisis y clasificación de datos que se han seleccionado para este proyecto.

Desarrollo de la aplicación

Este capítulo se basa en la implementación de la aplicación. Recoge información sobre el lenguaje de programación, las librerías utilizadas para la implementación de las funciones, información sobre el paquete shiny [22] y shinydashboard [23]. Además, incluye una descripción detallada de los elementos del código de la interfaz, los elementos utilizados a nivel visual y las funciones de de activación para dotar de dinamismo a la interfaz. Por último recoge una descripción de cada una de las pantallas que se han desarrollado en la interfaz, como por ejemplo A.1.18.

Limitaciones y próximos pasos

Recoge los puntos débiles de la aplicación y todo aquello que podría desearse en este tipo de aplicación. Además, se exponen las mejoras o ampliaciones que se pueden desarrollar en futuros avances del proyecto.

Conclusión

Se resumen los trabajos desarrollados en el TFM y se imprimen las opiniones personales y aprendizajes adquiridos a lo largo del mismo.

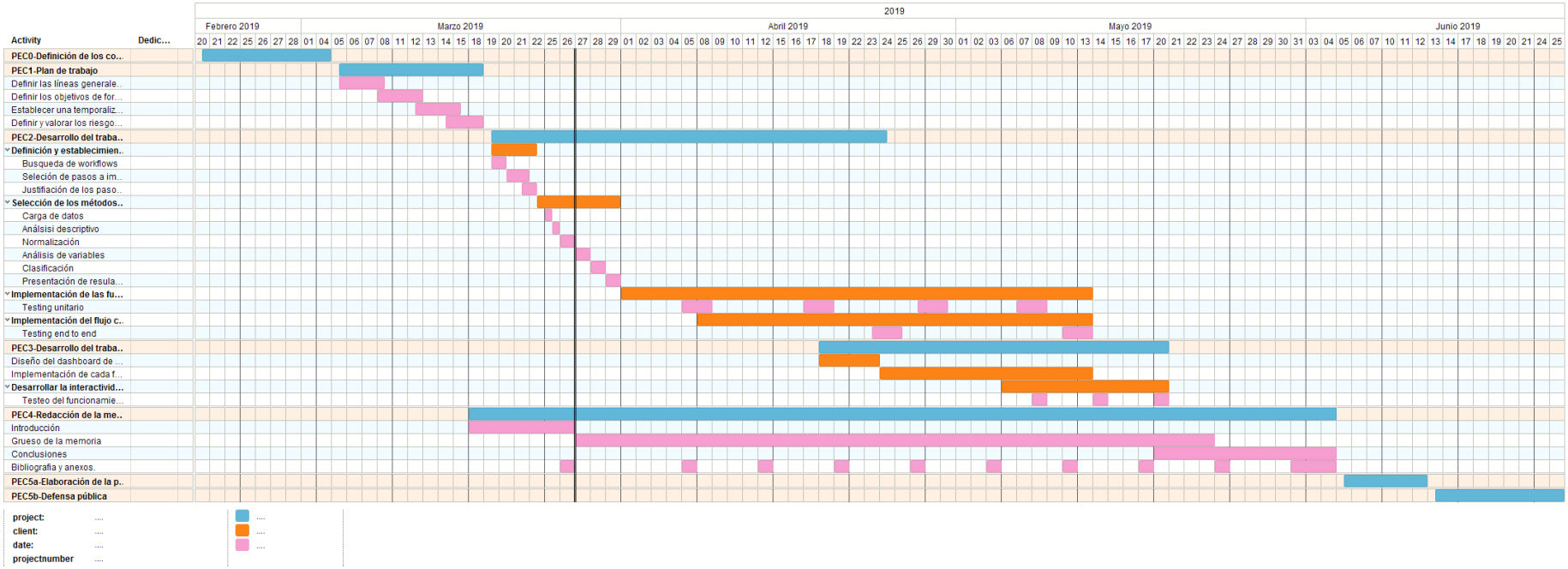


Figura 1.4.1: Diagrama de Gantt de tareas

2 El problema de clasificación

La clasificación [24] es un tema central en el aprendizaje automático que tiene que ver con la capacidad de aprendizaje de las máquinas a cómo agrupar los datos según criterios particulares. La clasificación es el proceso mediante el cual se intenta determinar a qué tipo pertenece cada uno de los registros de los datos del grupo basándose en características predeterminadas. Lo que se denomina aprendizaje supervisado consiste en tener ciertas muestras de las que se conoce el tipo para intentar asociar sus características al mismo. Existe una versión no supervisada de la clasificación [12] , denominada agrupación en clúster, donde las computadoras son características compartidas por las cuales se agrupan los datos cuando no se especifican las categorías. Este tipo de clasificación no se contempla en el alcance de este proyecto.

Un ejemplo común de clasificación muy utilizado es el de la detección de correos electrónicos no deseados o spam[27]. Para escribir un programa que filtre los correos electrónicos no deseados, un programador informático puede capacitarse en el algoritmo de aprendizaje automático con un conjunto de correos electrónicos similares a spam etiquetados como spam y los correos electrónicos regulares etiquetados como no spam. La idea es crear un algoritmo que pueda aprender las características de los correos electrónicos no deseados de este conjunto de capacitación para que pueda filtrar los correos no deseados cuando se encuentre con nuevos correos electrónicos. Este sería un caso de clasificación binaria. Por otra parte puede ser interesante la clasificación de diferentes muestras al tipo de célula al que pertenece, para este caso sería necesario disponer de diferentes muestras de distintos tipos de células para realizar la fase de aprendizaje y a continuación poder disponer de un modelo que nos permita determinar a qué tipo de célula de las que conocemos pertenece un nuevo registro.

La clasificación es una herramienta importante en el mundo de hoy, donde los datos grandes se utilizan para tomar todo tipo de decisiones en el gobierno, la economía, la medicina y o el tema que nos concierne, la bioinformática. Los investigadores tienen acceso a enormes cantidades de datos, y la clasificación es una herramienta que les ayuda a entender los datos y encontrar patrones.

La clasificación es algo que los humanos hacen naturalmente todos los días. La clasificación es simplemente agrupar las cosas de acuerdo con características y atributos similares. Mientras que la clasificación en aprendizaje automático requiere el uso de algoritmos complejos la mayoría de las veces. En bioinformática la clasificación se puede llevar a cabo sobre datos no estructurados, como pueden ser imágenes o estados de la célula de tipo grafo o bien sobre datos estructurados que son los tipos de datos que se explorarán en este proyecto. Podemos definir alguna terminología de la clasificación en términos de machine learning:

- Clasificador: Algoritmo que mapea un registro a una clase
- Modelo de clasificación: conjunto de reglas que intenta sacar algunas conclusiones sobre los valores de entrada en la fase de entrenamiento para predecir la clase o etiqueta de nuevos datos.
- Característica: Una característica es una propiedad individual medible de un fenómeno que se observa.
- Clasificación binaria: tarea de clasificación con dos posibles resultados
- Clasificación multiclase: tarea de clasificación con mas de dos clases. Cada registro se asocia a una única clase.

- Clasificación multilabel: Este tipo de clasificación tiene la peculiaridad de que cada registro se asocia a varias categorías a la vez.

En este caso nos vamos a centrar en la clasificación binaria y multiclase. Para construir un modelo de clasificación se suelen tener que llevar a cabo los siguientes pasos:

- Inicialización del modelo: se definen los parámetros del modelo,
- Entrenamiento: En este punto se le provee al modelo de un conjunto de datos que están clasificados para que busque las relaciones entre las características y la target.
- Predicción: Se le proporcionan al modelo los registros de los datos sin clasificar para que sea el propio modelo el que defina la target de cada uno.
- Evaluación: Si se conoce la clase real de los datos del conjunto de predicción se puede comparar el resultado del modelo con la realidad y determinar la calidad del modelo.
- Presentación de resultados

3 Modelos de clasificación seleccionados

Los siguientes modelos [14] son los elegidos, estudiados e implementados en este TFM

- **Modelo de los K-vecinos** Este algoritmo usa información sobre los k vecinos más cercanos a un ejemplo para clasificar los datos no etiquetados. La letra k es una variable que implica que se podría usar cualquier número de vecinos más cercanos. Después de elegir k, el algoritmo requiere un conjunto de datos de entrenamiento compuesto de ejemplos que se han clasificado en varias categorías, como lo indica una variable categórica. Luego, para cada registro no etiquetado en el conjunto de datos de test, *k-NN* identifica k registros en los datos de entrenamiento que son los más cercanos en similitud. A la instancia de test sin etiqueta se le asigna la clase de la mayoría de los k vecinos más cercanos. El algoritmo *k-NN* trata las características como coordenadas en un espacio de características multidimensional. La determinación de vecinos más cercanos requiere una función de distancia o una fórmula que mida la similitud entre dos instancias. La decisión de cuántos vecinos usar para *k-NN* determina como de bien se predirán y clasificaran los datos de test o futuros. El equilibrio entre el ajuste excesivo y el ajuste insuficiente de los datos de entrenamiento es un problema conocido como compensación de sesgo-varianza[9]. Por ello se suelen probar diferentes valores de k para al final elegir el que mejor resultados en el conjunto de test nos de. Como fortalezas de este modelo podemos destacar:

  - Simple y efectivo
  - No hace suposiciones sobre la distribuciónn de datos subyacente
  - Fácil y rápido de entrenar

  También tiene desventajas a la hora de utilizarlo:

  - No produce un modelo, lo que limita la capacidad de comprender cómo se relacionan las características con la clase Requiere selección de un k apropiado
  - Es lento en la fase de clasificación.
  - Las características categóricas y los datos nulos requieren un procesamiento adicional

- **Modelo de Naive-Bayes** Los modelos bayesianos asignan la clase mas probable a posteriori. Se basa en el supuesto de independencia condicional de las variables predictivas dada el target, es decir considera que cada una de estas características contribuye de manera independiente a la probabilidad de que un registro pertenezca a una clase u otra , independientemente de la presencia o ausencia de las otras características.

$$\text{classify}(f_1, \ldots, f_n) = \underset{c}{\text{argmax}}\ p(C = c) \prod_{i=1}^{n} p(F_i = f_i | C = c).$$

- **Redes Neuronales Artificiales**

  Una red neuronal queda definida por:

  - Una función de activación, que transforma las variables de entrada combinadas de una neurona en una única senal de salida para ser transmitida en la red, como pueden ser la sigmoidal, lineal, tangente hiperbólica, gausiana, etc. . .

- Una topología de red (o arquitectura), que describe la cantidad de neuronas en el modelo, así como la cantidad de capas y la forma en que estan conectadas. Esta parte define también si la informaciónn viaja hacia delante o también hacia atrás (RNN) y cuantos nodos tiene cada capa oculta, si hay varias capas ocultas se habla de deep neural networks (DNN). Esto define la complejidad y capacidad de aprendizaje de la red neuronal.
- El algoritmo de entrenamiento que especifica cómo se configuran los pesos de conexión para inhibir o excitar las neuronas en proporción al signo de entrada.

El entrenamiento se realiza con el backpropagation que permite aprender y reajustar los pesos a partir de la salida de la red neuronal. Consiste en:

- Una fase hacia adelante en la que las neuronas se activan en secuencia desde la capa de entrada a la capa de salida, aplicando los pesos y la función de activación de cada neurona. Al llegar a la capa final, se produce una senal de salida.
- Una fase hacia atrás en la que la senal de salida de la red resultante de la primera fase se compara con el verdadero valor objetivo en los datos de entrenamiento. La diferencia entre la senal de salida de la red y el valor verdadero da como resultado un error que se propaga hacia atrás en la red para modificar los pesos de conexión entre las neuronas y reducir los errores futuros.

Las fortalezas de este modelo se pueden resumir en los siguientes puntos:

- Simple y efectivo
- Capaz de modelar patrones más complejos que casi cualquier algoritmo
- Hace algunas suposiciones sobre las relaciones subyacentes de los datos

También tiene algunas desventajas que hay que considerar a la hora de decidir utilizarlas:

- Extremadamente intensivo en computación y lento en el entrenamiento, particularmente si la topología de la red es compleja
- Muy propensos a sobre ajustar los datos de entrenamiento.
- Resulta en un modelo complejo de caja negra que es difícil, si no imposible, de interpretar

][Estos modelos utilizan nodos organizados en capas conectadas de forma consecutiva de forma que cada unidad recibe la información de otras unidades de la capa anterior en lo que se denomina proceso Forward, y el ajuste de los parámetros de la red se realiza en el proceso de Backward.

Una red neuronal queda definida por:

- Una función de activación, que transforma las variables de entrada combinadas de una neurona en una única senal de salida para ser transmitida en la red, como pueden ser la sigmoidal, lineal, tangente hiperbólica, gausiana, etc. . .
- Una topología de red (o arquitectura), que describe la cantidad de neuronas en el modelo, así como la cantidad de capas y la forma en que estan conectadas. Esta parte define también si la informaciónn viaja hacia delante o también hacia atrás (RNN) y cuantos nodos tiene cada capa oculta, si hay varias capas ocultas se habla de deep neural networks (DNN). Esto define la complejidad y capacidad de aprendizaje de la red neuronal.
- El algoritmo de entrenamiento que especifica cómo se configuran los pesos de conexión para inhibir o excitar las neuronas en proporción al signo de entrada.

El entrenamiento se realiza con el backpropagation que permite aprender y reajustar los pesos a partir de la salida de la red neuronal. Consiste en:

- Una fase hacia adelante en la que las neuronas se activan en secuencia desde la capa de entrada a la capa de salida, aplicando los pesos y la función de activación de cada neurona. Al llegar a la capa final, se produce una senal de salida.

- Una fase hacia atrás en la que la senal de salida de la red resultante de la primera fase se compara con el verdadero valor objetivo en los datos de entrenamiento. La diferencia entre la senal de salida de la red y el valor verdadero da como resultado un error que se propaga hacia atrás en la red para modificar los pesos de conexión entre las neuronas y reducir los errores futuros.

Las fortalezas de este modelo se pueden resumir en los siguientes puntos:

- Simple y efectivo
- Capaz de modelar patrones más complejos que casi cualquier algoritmo
- Hace algunas suposiciones sobre las relaciones subyacentes de los datos

También tiene algunas desventajas que hay que considerar a la hora de decidir utilizarlas:

- Extremadamente intensivo en computación y lento en el entrenamiento, particularmente si la topología de la red es compleja
- Muy propensos a sobre ajustar los datos de entrenamiento.
- Resulta en un modelo complejo de caja negra que es difícil, si no imposible, de interpretar

]Estos modelos utilizan nodos organizados en capas conectadas de forma consecutiva de forma que cada unidad recibe la información de otras unidades de la capa anterior en lo que se denomina proceso Forward, y el ajuste de los parámetros de la red se realiza en el proceso de Backward.

Una red neuronal queda definida por:

- Una función de activación, que transforma las variables de entrada combinadas de una neurona en una única senal de salida para ser transmitida en la red, como pueden ser la sigmoidal, lineal, tangente hiperbólica, gausiana, etc. . .
- Una topología de red (o arquitectura), que describe la cantidad de neuronas en el modelo, así como la cantidad de capas y la forma en que estan conectadas. Esta parte define también si la informaciónn viaja hacia delante o también hacia atrás (RNN) y cuantos nodos tiene cada capa oculta, si hay varias capas ocultas se habla de deep neural networks (DNN). Esto define la complejidad y capacidad de aprendizaje de la red neuronal.
- El algoritmo de entrenamiento que especifica cómo se configuran los pesos de conexión para inhibir o excitar las neuronas en proporción al signo de entrada.

El entrenamiento se realiza con el backpropagation que permite aprender y reajustar los pesos a partir de la salida de la red neuronal. Consiste en:

- Una fase hacia adelante en la que las neuronas se activan en secuencia desde la capa de entrada a la capa de salida, aplicando los pesos y la función de activación de cada neurona. Al llegar a la capa final, se produce una senal de salida.
- Una fase hacia atrás en la que la senal de salida de la red resultante de la primera fase se compara con el verdadero valor objetivo en los datos de entrenamiento. La diferencia entre la senal de salida de la red y el valor verdadero da como resultado un error que se propaga hacia atrás en la red para modificar los pesos de conexión entre las neuronas y reducir los errores futuros.

Las fortalezas de este modelo se pueden resumir en los siguientes puntos:

- Simple y efectivo
- Capaz de modelar patrones más complejos que casi cualquier algoritmo
- Hace algunas suposiciones sobre las relaciones subyacentes de los datos

También tiene algunas desventajas que hay que considerar a la hora de decidir utilizarlas:

- Extremadamente intensivo en computación y lento en el entrenamiento, particularmente si la topología de la red es compleja
- Muy propensos a sobre ajustar los datos de entrenamiento.
- Resulta en un modelo complejo de caja negra que es difícil, si no imposible, de interpretar

- **Support Vector Machine** Este metodo de clasificación se basa en el preprocesamiento de los datos para representar patrones en dimensiones altas [2]. Encontramos la situación de datos linealmente separables y no linealmente separables. Una característica clave de los SVM es su capacidad para mapear el problema en un espacio de dimensión superior usando una función kernel. Al hacerlo, una relación no lineal puede parecer repentinamente bastante lineal. Existen kernels lineales, polinomiales, sigmoidales o gausianos. Gracias a los mapeos no lineales dos categorías se pueden separar con un hiperplano. También se pueden usar polinomios gausianos o funciones lineales, dependiendo eso del tipo de datos que tengamos en nuestro conjunto. La clasificación con hiperplanos se mide con el MMH para separar lo mejor posible las clases. Los vectores de soporte son los puntos de cada clase que estan más cerca del MMH; cada clase debe tener al menos un vector de soporte, pero es posible tener más de uno. Usando solo los vectores de soporte, es posible definir el MMH. Esta es una característica clave de SVMs. Como ventajas de este método de clasificación podemos destacar:

  - Puede ser utilizado para problemas de clasificación o predicción numérica
  - No esta demasiado influido por datos ruidosos y no es muy propenso a sobrealimentar
  - Puede ser más fácil de usar que las redes neuronales, particularmente debido a la existencia de varios algoritmos SVM bien soportados

  Como desventajas a tener en cuenta:

  - Encontrar el mejor modelo requiere probar varias combinaciones de núcleos y parametros de modelo.
  - Puede ser lento para entrenar, especialmente si el conjunto de datos de entrada tiene una gran cantidad de funciones o ejemplos
  - Resulta en un modelo complejo de caja negra que es difícil, si no imposible, de interpretar.

- **Decission Tree** Estos modelos son bastante intuitivos. Se basan en la consecución de preguntas y división y subdivisión de los datos de acorde a la respuesta a dicha pregunta. Además, las preguntas consecutivas dependen de la respuesta a la anterior. A través de las ramas del árbol se conecta la raíz con las hojas del mismo, las cuales determinan la clasificación de cada una de las muestras. Puede que en cada hoja se concentre una mezcla de clases, pero esto se acepta como la imperfección de la hoja. Al clasificar una muestra, la probabilidad de ser de una clase u otra la dará el conjunto de muestras de cada categoría que definan dicha hoja.

  Las ventajas de este tipo de modelos son:

  - Un clasificador de uso múltiple que funciona bien en la mayoría de los problemas.
  - Proceso de aprendizaje altamente automático, que puede manejar características numéricas o nominales, así como datos faltantes.
  - Excluye características sin importancia.
  - Se puede utilizar en conjuntos de datos pequeños y grandes.
  - Resultados en un modelo que se puede interpretar sin un fondo matemático (para árboles relativamente pequeños).
  - Más eficiente que otros modelos complejos.

  Las desventajas también se han de mencionar y son las siguientes:

- Los modelos de árbol de decisión a menudo están sesgados hacia divisiones en entidades que tienen un gran número de niveles.
- Es fácil adaptar o adaptar el modelo.
- Puede tener problemas para modelar algunas relaciones debido a la dependencia de divisiones de eje-paralelo.
- Pequeños cambios en los datos de entrenamiento pueden resultar en grandes cambios en la lógica de decisión.
- Los árboles grandes pueden ser difíciles de interpretar y las decisiones que toman pueden parecer contradictorias.

El primer desafío al que se enfrentará un árbol de decisión es identificar con qué característica dividirse. El grado en que un subconjunto de ejemplos contiene solo una clase única se conoce como pureza, y cualquier subconjunto compuesto de una sola clase se llama puro. Hay varias medidas de pureza que se pueden usar para identificar el mejor candidato para la división del árbol de decisión. El árbol de decisión espera encontrar divisiones que reduzcan la entropía, lo que en última instancia aumenta la homogeneidad dentro de los grupos [3]. Normalmente, la entropía se mide en bits. Si solo hay dos clases posibles, los valores de entropía pueden variar de 0 a 1. Para n clases, la entropía varía de 0 a $\log_2(n)$. En cada caso, el valor mínimo indica que la muestra es completamente homogénea, mientras que el valor máximo indica que los datos son lo más diversos posible, y ningún grupo tiene incluso una pequeña pluralidad.

$$Entropia(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

Para utilizar la entropía para determinar la característica óptima a partir de la cual dividirse, el algoritmo calcula el cambio en la homogeneidad que resultaría de una división en cada característica posible, que es una medida conocida como ganancia de información. La ganancia de información para una característica se calcula como la diferencia entre la entropía en el segmento antes de la división (S1) y las particiones resultantes de la división (S2):

$$InfoGain(F) = Entropia(S_1) - Entropia(S_2)$$

- **Random forest** Es la combinación de múltiples árboles de decisión cuya selección de variables discriminatorias y de corte en la clasificación es aleatoria. El conjunto de resultados de cada árbol es lo que determina como se clasifica una muestra. Este tipo de modelos combinan versatilidad y potencia en un solo enfoque de aprendizaje automático. Como el conjunto utiliza solo una pequeña porción aleatoria del conjunto completo de características, los random forest pueden manejar conjuntos de datos extremadamente grandes, donde la llamada *maldición de la dimensionalidad* puede hacer que otros modelos fallen. Al mismo tiempo, sus tasas de error para la mayoría de las tareas de aprendizaje están a la par con casi cualquier otro método y son mejores a la hora de no caer en el overfiting.

  Otras ventajas de estos modelos son:

  - Un modelo de uso múltiple que funciona bien en la mayoría de los problemas
  - Puede manejar datos faltantes o ruidosos, así como características categóricas o continuas
  - Selecciona solo las características más importantes.
  - Se puede usar en datos con una gran cantidad de funciones o ejemplos

  Como desventajas podemos destacar:

  - A diferencia de un árbol de decisión, el modelo no es fácil de interpretar.
  - Puede requerir esfuerzo para ajustar el modelo a los datos.

4 Descripción de los datos

La idea principal con la que se ha creado esta herramienta es que cualquier conjunto de datos con las características que se describirán a continuación pueda ser utilizado para hacer predicción de dos o más categorías.

Los datos deben cumplir los siguientes requerimientos:

- Tamaño: los ficheros no pueden superar los 100Mb.
- Deben contener datos estructurados
- Las columnas pueden tener nombre o no, en caso de no disponer del mismo, se debe conocer en que posición está la variable objetivo ya que R le otorgará nombres según el orden.
- Los datos pueden estar en 1 o 2 ficheros. En un único fichero se debe identificar la variable objetivo en la aplicación. En el caso de dos ficheros, el primero deberá contener las variables predictoras y el segundo contener únicamente una columna correspondiente a la variable objetivo, ordenadas en el mismo orden que los registros del fichero de características.
- El formato de los ficheros debe ser CSV separado por *";","tab","|."* o *" "*.
- Podrán contener datos sin informar, pero se ha de indicar el valor que toman estos datos en caso de ser distinto de NULL.
- La variable objetivo ha de ser categórica, es decir, debe interpretarse como la clase a la que pertenece cada registro. Podrá ser binaria o múltiple.

4.1. Dataset Binario

Cervical cancer (Risk Factors) Data Set

El conjunto de datos fue recolectado en el Hospital Universitario de Caracas en Caracas, Venezuela. El conjunto de datos incluye información demográfica, hábitos y registros médicos históricos de 858 pacientes. Varios pacientes decidieron no responder algunas de las preguntas debido a problemas de privacidad (valores perdidos).

Las variables que contiene son las siguientes:

Kelwin Fernandes, Jaime S. Cardoso, and Jessica Fernandes. 'Transfer Learning with Partial Observability Applied to Cervical Cancer Screening.' Iberian Conference on Pattern Recognition and Image Analysis. Springer International Publishing, 2017.

4.2. Dataset Multiclase

Este conjunto de datos tiene target multiclase y está dividido en dos ficheros. En uno se encuentran los atributos o valores de los genes para cada registro, y en el segundo fichero se encuentra el tipo de tumor al que pertenece la muestra.

Data Set Characteristics:	Multivariate
Number of Instances:	858
Area:	Life
Attribute Characteristics:	Integer, Real
Number of Attributes:	36
Date Donated:	42797
Associated Tasks:	Classification
Missing Values?:	Yes
Number of Web Hits:	83824

Cuadro 4.1.1: Datos de clasificación binaria

Tipo	**Nombre**
(int)	Age
(int)	Number of sexual partners
(int)	First sexual intercourse (age)
(int)	Num of pregnancies
(factor)	Smokes
(factor)	Smokes (years)
(factor)	Smokes (packs/year)
(factor)	Hormonal Contraceptives
(int)	Hormonal Contraceptives (years)
(factor)	IUD
(int)	IUD (years)
(factor)	STDs
(int)	STDs (number)
(factor)	STDs:condylomatosis
(factor)	STDs:cervical condylomatosis
(factor)	STDs:vaginal condylomatosis
(factor)	STDs:vulvo-perineal condylomatosis
(factor)	STDs:syphilis
(factor)	STDs:pelvic inflammatory disease
(factor)	STDs:genital herpes
(factor)	STDs:molluscum contagiosum
(factor)	STDs:AIDS
(factor)	STDs:HIV
(factor)	STDs:Hepatitis B
(factor)	STDs:HPV
(int)	STDs: Number of diagnosis
(int)	STDs: Time since first diagnosis
(int)	STDs: Time since last diagnosis
(factor)	Dx:Cancer
(factor)	Dx:CIN
(factor)	Dx:HPV
(factor)	Dx
(factor)	Hinselmann: target variable
(factor)	Schiller: target variable
(factor)	Cytology: target variable
(factor)	Biopsy: target variable

Cuadro 4.1.2: Nombres y tipos de las variables del dataset

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Real
Associated Tasks:	Classification, Clustering
Number of Instances:	801
Area:	Life
Date Donated:	42530
Number of Web Hits:	37358
Number of Attributes:	20531
Missing Values?:	N/A

Cuadro 4.2.1: Datos de clasificación múltiple

Las muestras (instancias) se almacenan por filas. Las variables (atributos) de cada muestra son los niveles de expresión del gen RNA-Seq medidos por la plataforma illumina HiSeq. Se le da un nombre ficticio (gene_XX) a cada atributo. Es una extracción aleatoria de expresiones genéticas de pacientes con diferentes tipos de tumores: BRCA, KIRC, COAD, LUAD y PRAD.

5 Pipeline

Para llevar a cabo un análisis de datos y resolver problemas de clasificación normalmente se llevan a cabo los siguientes pasos definidos para el proyecto.

- Carga y definición de los datos de entrada:
  - Se da la opción de definir si los datos vienen en un único fichero, o en dos diferenciados. Esta dualidad se ha decidido incluir ya que en el estudio preliminar de posibles datasets se ha observado que en muchos casos se proporciona un fichero para las features, y otro para las labels.
  - Especificación de si el conjunto es binario o multiclase.
  - Selección de los ficheros de datos.
  - Especificación es de lectura de los datos, separador, cabecera elementos nulos.
  - Para el caso de un único fichero de datos se proporciona un selector de la variable objetivo. En el caso de dos ficheros de datos se asume que el segundo fichero contiene el conjunto de datos de objetivo.
  - Muestra preliminar del conjunto de datos seleccionado por el usuario.
- Inferencia y descripción de datos. Para tener una idea general de los datos disponibles se realiza el análisis descriptivo de cada una de las features proporcionadas en el conjunto de datos. En esta fase del análisis se separan las features de los datos en varios tipos, a saber, numéricas y categóricas. Para cada uno de estos grupos se presenta una visualización de la distribución para las variables numéricas, por grupos para las variables categóricas (conteo de cada clase), considerando en este caso la misma visualización para las variables lógicas. Además, se añade una comparativa de la distribución de los datos numéricos para cada una de los target.
- Selección del tamaño del conjunto de train y test. De esta forma se puede determinar que porcentaje de datos se quiere dedicar al entrenamiento y evaluación de predicción de los modelos. Se muestra el conjunto de datos seleccionado para cada grupo.
- Preprocesamiento del conjunto de datos en función del tipo de modelo de clasificación. Como sabemos, los diferentes modelos de machine learning disponibles en las librerías de R requieren un conjunto de datos de entrada que cumplan ciertas características como tipo de dato, gestión de nulos o normalización y estandarización de los datos para obtener resultados de calidad y fiables.
  - Para el conjunto de variables numéricas se permite la estandarización, la normalización y la imputación de los datos.
  - Para el conjunto de datos de variables categóricas se permite la factorización y aplicación del OneHotEncoding [10].
  - Para las variables lógicas se hace una transformación a 0 y 1, en el caso binario o 0 a N en el caso de multiclase.
- Modelado de los datos, que incluye la fase de entrenamiento y test. Se han implementado los algoritmos de clasificación binaria y multiclase. Dependiendo del conjunto de datos seleccionado por el usuario, sea binario o no, se le dan al usuario un conjunto de modelos y selección de hiperparámetros. Estos datos servirán para realizar los entrenamientos de los modelos con dichas

características y mostrar los resultados sobre el conjunto de test que en este caso se han decidido las tablas de matrices de confusión, las distintas métricas asociadas a la performance del modelo y la curva ROC.

6 Desarrollo de la aplicación

El desarrollo de una aplicación web puede ser llevado a cabo a través de una gran cantidad de lenguajes de programación como podían ser JavaScript, Python, ASP, Ruby, PHP,[8] ... Sin embargo, el lenguaje escogido para desarrollar esta aplicación es R y su paquete para el desarrollo de aplicaciones web Shiny, debido a que el pipeline diseñado para el análisis de datos ha sido desarrollado en R. El propósito de esta aplicación es facilitar el trabajo de análisis de datos y predicción con modelos de clasificación a cualquier persona que no posea un amplio conocimiento de programación el análisis de datos, por tanto, es conveniente que la interfaz sea simple e intuitiva y el dashboard de Shiny permite cumplir este requerimiento.

6.1. R y las librerías utilizadas

El Integrated Development Environment (IDE) utilizado para la programación de la GUI ha sido RStudio [**?**] ya que nos permite la ejecución *in situ* del código así como la representación de gráficas y otros resultados. Desde ella se han podido integrar las diversas librerías (paquetes de R) necesarias para la implementación de la interfaz y las herramientas del análisis y clasificación. R es un lenguaje de programación dirigido, principalmente, al análisis estadístico. Sin embargo, es ampliamente usado en minería de datos, investigación biomédica y bioinformática. Se distribuye mediante licencia GNU GPL y esta disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux. R fue desarrollado por Robert Gentleman y Ross Ihaka en el Departamento de Estadística de la Universidad de Auckland [30]. La interfaz se ha diseñado utilizando shinyFiles [16], shinydashboard [23] y Shiny [22].

6.2. El machine learning con R

Muchos de los algoritmos necesarios para el aprendizaje automático con R no se incluyen como parte de la instalación básica. En cambio, los algoritmos necesarios para el aprendizaje automático están disponibles a través de una gran comunidad de expertos que han compartido su trabajo libremente. Estos deben instalarse en la parte superior de la base R manualmente. Gracias al estado de R como software libre de código abierto, no hay cargo adicional por esta funcionalidad. Para la programación de los modelos se ha usado como referencia el libro de texto *Machine Learning with R* [14].

6.3. Shiny y ShinyDashboard

Shiny es un paquete *open source* de R, que permite desarrollar aplicaciones web usando código de R41. Shiny posee una gran variedad de widgets para crear de forma rápida interfaces. Sin embargo, Shiny es muy extensible y es fácil integrar contenido web utilizando HTML, CSS, JavaScript y jQuery [6]. La estructura de un programa desarrollado con Shiny es sencilla, el programa se divide en dos partes, que pueden localizarse en dos archivos distintos (app.R y ui.R) o en un mismo archivo (app.R). La parte o archivo 'ui' o interfaz se centra en crear la interfaz gráfica, es decir, el resultado visual de la aplicación, mientras que el archivo o parte 'app' o controlador se centra en realizar las operaciones y procesamiento de los datos. El dashboard de Shiny permite desarrollar aplicaciones con una potente estética visual y mayor funcionalidad48.

6.4. Desarrollo de la aplicación

El código de la aplicación se incluye en el anexo.

En el desarrollo de la aplicación se han utilizado distintos elementos de Shiny UI (cada pantalla contiene un código) y Shiny SERVER (incluido en app.R) así como funciones de R que permitan llevar a cabo la funcionalidad, en la figura 6.4.1 podemos ver como interaccionan ambos módulos [5].

UI Dentro de la parte visual de la aplicación se utilizan los elementos de layout como son el menú lateral, cabecera y cuerpo de la aplicación.

- El menú lateral servirá para pasar de una fase del pipeline a otra, ya que cada una de estas fases está asociada a una pantalla de la aplicación.
- El cuerpo de la aplicación se llevarán a cabo todas las interacciones con el usuario, datos y resultados.

Para crear la interfaz gráfica se han utilizado varios tipos de elementos:

- Input de datos: Para la interacción con el usuario se utilizan elementos como selectores, check box, radio buttons, entradas de texto, selectores de ficheros, sliders y botones.
  - *radioButtons*: elemento utilizado para la selección única de opciones.
  - *sliderInput*: widget que permite establecer un valor numérico entre un rango de valores determinado en intervalos.
  - *textInput*: campo de texto en el cual el usuario puede introducir datos mediante entrada de teclado.
  - *selectInput*: ventana desplegable con valores predeterminados mediante el cual el usuario puede seleccionar uno de los valores.
  - *checkboxGroupInput*: elemento de selección múltiple.
  - *Shinydirbutton*: botón específico del paquete shinyFiles que permite la selección de directorios o carpetas, así como la creación de nuevos directorios.
  - *fileInput*: elemento de selección de archivos, mediante el cual se cargan archivos de forma temporal en la aplicación para su uso. Permite determinar los tipos y tamaños de dichos ficheros.
- Output de datos: Para mostrar resultados se utilizan salidas de texto textOutput, tablas dataTableOutput y gráficos plotOutput. Estos elementos en muchos casos se combinan dentro de uiOutputs, ya que han de generarse dentro de una misma caja o pestaña asociada a un tipo de modelo.
  - *uiOutput*: función encargada de actualizar valores dinámicos de la interfaz a partir de valores de entrada. Este elemento se ha utilizado en las pantallas que dependan del número variable features y modelos. La creación de tabBoxes va asociada a las variables y a los modelos.
  - *dataTableOutput*: elemento de generación de tablas de manera dinámica. Se ha utilizado para mostrar los datos asociados al análisis descriptivo de los datos y los resultados de las métricas de los modelos que se han generado mediante las funciones.
  - *plotOutput*: esta función de Shiny nos permite actualizar y mostrar elementos gráficos.
  - *textOutput*: función encargada de mostrar valores de tipo texto.
- Elementos de agrupación:
  - Los *boxes* se han utilizado para agrupar elementos.
  - Los *tabBox* se han utilizado para agrupar los resultados de los modelos o tipos de variables.

- Los inputs son los parámetros que utilizará la aplicación para determinar el path del pipeline a seguir.
- Además, se han incluido algunos elementos de control, por ejemplo, durante el proceso de generación de modelos de machine learning se proporciona un elemento que indica el avance *progress bar* y también se han incluido cuadros de control cuando no se introducen datos correctos para informar al usuario.

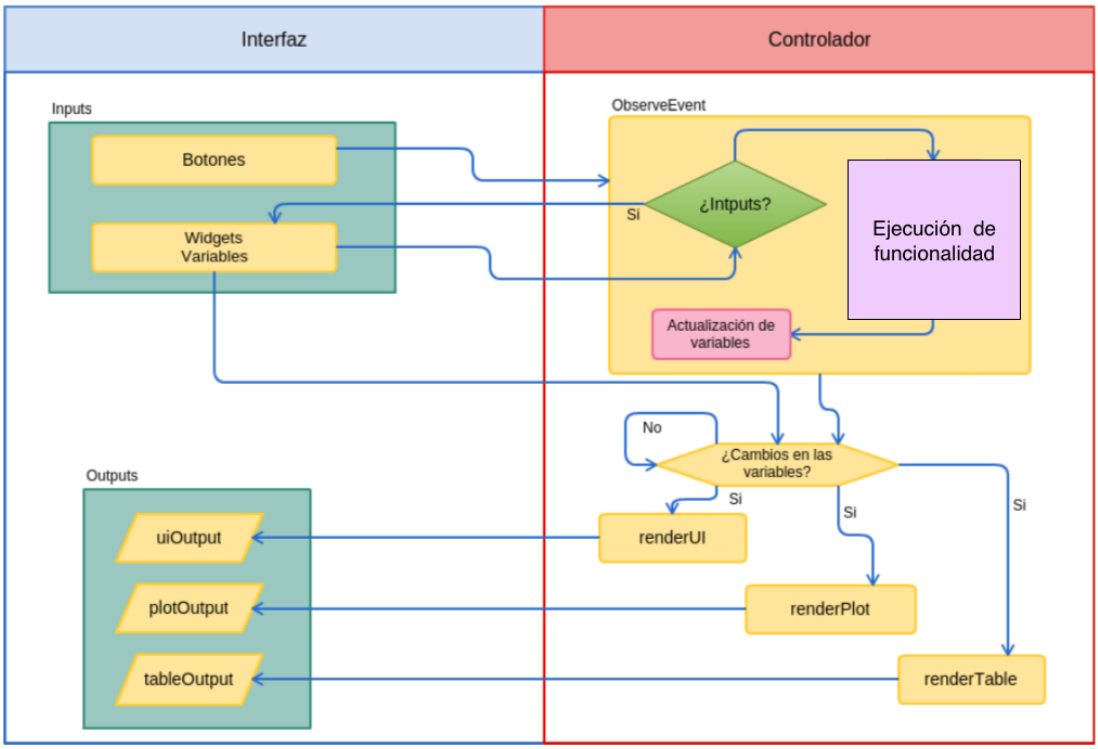


Figura 6.4.1: Funcionalidad UI y SERVER

SERVER La parte del servidor es la mas compleja y extensa de la aplicación. En ella se incluyen todos los elementos reactivos, actualizaciones de valores de la interfaz gráfica, procesamiento de datos y generación de resultados.

- Definición y actualización de variables. Para ello se definen estructuras de valores reactivos que permitan hacer uso de dichas variables a lo largo de las distintas fases del pipeline. Si aún no se ha llegado a una fase del pipeline o se quiere volver a realizar una ejecución las variables se reinician al valor FALSE.
- Ejecución de la funcionalidad. Estas acciones se llevan a cabo principalmente a través de las funciones de observ y observEvent. En la mayoría de los casos se activan a través de los botones disponibles en la aplicación. La invocación de las funciones se controla mediante la interacción con botones de la interfaz gráfica. Para ello se ha utilizado la función observEvent. Esta es una función que se activa mediante el clic en dichos botones. Una vez se ha presionado el botón asociado a un evento se procede a hacer una recogida de los datos de entrada, utilización de estos datos para seguir el pipeline y procesamiento de los resultados obtenidos. Los eventos que se han implementado con esta funcionalidad son la carga de datos y la ejecución de los modelos.
    - La carga de datos desencadena el proceso de lectura de datos, separación de variables por tipo, procesado del target y descripción de los datos.

- La ejecución de modelos procede a hacer el preprocesamiento de los datos definido por el usuario y ejecutar los modelos con los hiperparámetros elegidos. Cuando finaliza la ejecución se muestran los gráficos y resultados. Una vez generados, se recogen los resultados y se utilizan para la representación gráfica.

- Generación de gráficas, datos y tablas de resultados. Estos procesos se encargan de generar los outputs visuales a partir de los datos generados durante el pipeline. Estos resultados visuales son incrustados en la interfaz gráfica y actualizados cada vez que los datos cambian de forma automática.

  - *renderUI*: actualización o creación de elementos como widgets, box o tabBox.
  - *renderPlot*: actualización o creación de gráficas con los resultados del análisis.
  - *renderDataTale*: actualización o creación de tablas y los datos de estas.

Las funciones son reactivas a uno o varios valores de entrada gracias lo que se conoce como la estructura Modelo Vista Controlador (MVC)[5].

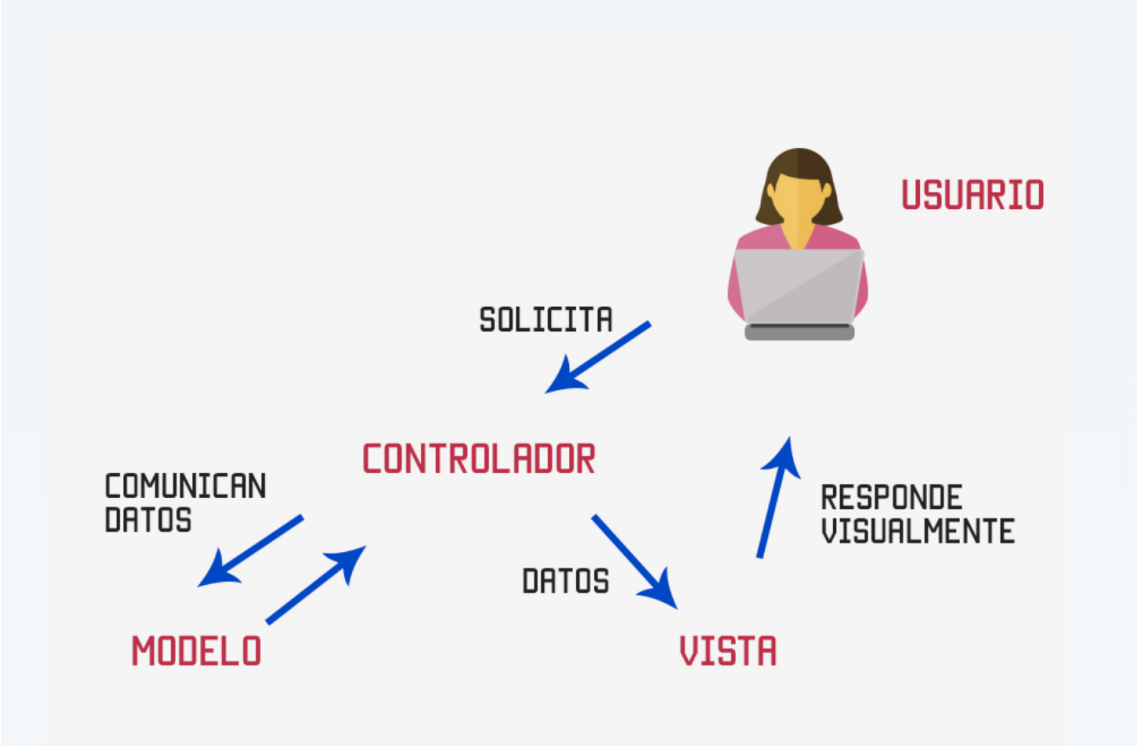


Figura 6.4.2: Modelo-Vista-Controlador

A partir de estos valores se generan los elementos necesarios para generar los elementos gráficos. De esta forma se ha conseguido generar un número variable de elementos con identificadores que permitirán mostrar unos resultados adaptados a lo que haya solicitado el usuario. Es importante mencionar que, si se modifican los valores de entrada, las funciones se desencadenarán automáticamente y los elementos dinámicos asociados a estas funciones se perderán. Se generarán los nuevos elementos y los datos asociados a los elementos antiguos habrán de ser introducidos de nuevo o cambiarlos por los nuevos valores. En la mayoría de los casos, las funciones *renderUI* son reactivas a las variables asociadas a los widgets. Por el contrario, las funciones *renderDataTable* y *renderPlot* tiene como valores de entrada las variables generadas en las funciones observeEvent, ya que se generarán los datos para las tablas y gráficas con los resultados generados tras la ejecución de los modelos.

Diseño de la interfaz La interfaz consiste en varias pantallas conectadas entre ellas y que siguen el orden lógico del análisis de datos.

1. Pantalla de inicio: En esta pantalla se explican las distintas pantallas de las que se compone la aplicación. Para cada una de las fases del pipeline se explican los pasos que hay que llevar a cabo en cada pantalla y se pueden ver las imágenes A.1.18, A.1.19 y A.1.20. Para esta pantalla se utilizan cajas desplegables.

2. Carga de datos: Se corresponde con la funcionalidad del pipeline 2.1.1.2 y se muestran en las imágenes A.1.1 ,A.1.2 y A.1.21 En el se incluyen los elementos descritos en este punto, así como parametros de lectura de los ficheros, comprobaciones de los tipos de ficheros A.1.3 y selección de target en caso de ser necesario A.1.4.

3. Descripción de los datos: Se corresponde con la funcionalidad de 2.3.. Para ello se muestra la inferencia del tipo de datos y sus valoresA.1.5, y por cada tipo de variable se muestra su distribución en histograma o tabla por valores [4] y por último una comparativa de las variables para cada valor de la variable objetivo A.1.6.

4. Selección de los conjuntos de entrenamiento y test para los modelos A.1.7. Esta pantalla permite al usuario definir el porcentaje de datos que se utilizará como conjunto de entrenamiento..

5. Modelado y procesamiento de datos. Esta pantalla incluye todos los elementos referentes al modelado de datos 1.2,1.2 y presentación de los resultados obtenidos para cada uno de ellos. Consiste en las siguientes partes.

   - Selección de los modelos que se querrán utilizar A.1.8.
   - Para cada uno de los modelos seleccionados se podrán definir los parametros y el preprocesamiento recomendable que se le ha de hacer por lo general a un conjunto de datos. Puesto que el usuario puede haber llevado a cabo este proceso por su cuenta puede decidir no seleccionar ningún preprocesamiento. Además, podrá decidir si quiere una salida probabilística o de clase. Ejemplos de ello los podemos ver en A.1.9, A.1.13, A.1.11, A.1.12, A.1.10..
   - Por último, se muestran los resultados y visualizaciones obtenidas para cada uno de los modelos seleccionados. En las siguientes imágenes podemos ver los ejemplosA.1.17, A.1.15, A.1.16, A.1.14.

6.5. Ejecución de la aplicación y batería de pruebas

La forma de probar la aplicación se ha realizado en varias fases. Por una parte se han implementado las distintas funciones y se han probado en un entorno controlado, en el que se llamaba a la función con los inputs esperados y se comprobaba que el output era el deseado. Una segunda fase de pruebas es la creación interactiva de los elementos de la interfaz a medida que se ha ido desarrollando. Por último, sobre la aplicación ya implementada se han hecho pruebas sobre el funcionaminto del pipeline.

En las siguientes imágenes vemos un ejemplo de selección de los modelos de clasificación múltiple A.1.22 y de los resultados A.1.23

Todas estas prueba han tenido las siguientes consecuencias:

- Se ha refactorizado parte de la implementación, mejorando su estructura interna, claridad y mantenibilidad, fruto de las inspecciones del código.
- Se han corregido errores de ejecución de las herramientas y errores leves gracias a las pruebas de caja negra (entrada y salida).
- Se han podido detectar y corregir errores de la interfaz gráfica, resultado de las pruebas realizadas sobre la aplicación. Esto ha llevado a mejorar el control de errores y mensajes al usuario.

Prueba	**Binario**	**Multiclase**
Carag de datos a partir de un fichero	x	x
Carga de datos de dos ficheros	x	x
Selección de la variable target	x	x
Visualización de la descripción de datos binarios	x	x
Visualización de la descripción de datos binarios	x	x
Visualización de la descripción de datos por cada categoría en boxplot	x	x
Visualización de histogramas en pestañas	x	x
Selección del porcentaje de datos para entrenamiento y test	x	x
Visualización de los datos de entrenamiento y test	x	x
Selector de modelos	x	x
Selector de preprocesamiento para cada modelo	x	x
Selección de los hiperparámetros de los modelos	x	x
Ejecución de los modelos con todos los parámetros	x	x
Intentos de ejecución con selecciones erróneas	x	x
Visualización de los resultados de los modelos	x	x
Generación interactiva de pestañas para diferentes números de elementos	x	x
Control de errores	x	x

Cuadro 6.5.1: Batería de pruebas de la GUI

- Se han realizado modificaciones sobre la interfaz con el fin de mejorar aquellos aspectos que presentaban la mayor complejidad.

Se han encontrado fallos en la aplicación cuando se ha desplegado en un entorno distinto al de desarrollo.

7 Limitaciones y próximos pasos

A continuación se exponen algunas de las limitaciones que se han detectado, en algunos casos de deben a la falta de expertise en el desarrollo de aplicaciones, otras al tiempo disponible y otras a la tecnología

1. Capacidad de carga y entrada de ficheros
   a) Formato: solo admite csv.
   b) Tamaño: el tamaño está limitado a 100Mb
   c) Organización de las estructuras
2. Visualización de los datos. El look and feel y la variedad de gráficas.
3. Incluir la opción de cross validation.
4. Preprocesamiento y tratamiento de datos. Una parte muy importante en el análisis y creación de modelos es el tratamiento de datos, limpieza y selección de variables. En este proyecto no se ha incluido toda esta parte por las limitaciones de tiempo.
5. Modelos. Existen muchos modelos de clasificación que no se han incluido en este proyecto, además, de los modelos elegidos se han elegido hiperparámetros muy básicos.
6. Presentación de resultados. Se han elegido resultados muy trasversales a este tipo de problemas y visualizaciones.
7. Descarga de resultados.
8. Mejora del look de la aplicación. Se ha intentado generar un diseño lo mas sencillo posible y organizado en pantallas de acuerdo al pipeline.

Todos los temas anteriores son posibles ampliaciones de la funcionalidad y requisitos de la aplicación web y como primera aproximación a próximos pasos de este proyecto podrían integrarse en la misma.

Desde el punto de vista de la tecnología se ha encontrado la limitación debido a las diferentes versiones de R instaladas en la máquina de desarrollo R versión 3.5.0 (2018-04-23) en un *MACOS*, y la versión de la máquina virtual R 3.4 se han encontrado muchos problemas a la hora de instalar las librerías requeridas. Por ejemplo, no se ha podido utilizar la librería *C5.0* por una incompatibilidad en una de las dependencias y se ha tenido que cambiar a *rpart* [26] para la implementación de los árboles de decisión. Por otra parte, unos problemas de shiny server que no se han encontrado solución por ahora como son RPLOT.pdf (not found) y la función *train* de SVM.

8 Conclusiones

El presente Trabajo de Fin de Máster ha permitido el desarrollo de una aplicacion web destinada a integrar un workflow para el analisis de análisis de datos y clasificación permitiendo un uso sencillo e intuitivo de los métodos recogidos en el mismo. Ha supuesto aprender las bases del machine learing relacionada con la clasificación, en particular lo referente a la clasificación supervisada de datos estructurados, conocer los algoritmos mas utilizados e investigar las soluciones como integrar estos en una interfaz gráfica. Este trabajo se ha realizado con el fin de poder crear una interfaz que utilice un pipeline unificado y que a la vez simplifique el uso de estos modelos y lectura de los resultados al usuario no experto.

El proyecto ha seguido un ciclo de vida en cascada con retroalimentación en el que se han seguido las fases de análisis de requisitos, diseño, implementación, pruebas y validación hasta llegar a un producto que satisface la idea inicial del proyecto, pero que a su vez ha ido evolucionando a medida que se ha desarrollado. Actualmente se encuentra en fase mantenimiento y ampliación de funcionalidad. El proyecto se ha diseñado siguiendo un patrón arquitectura MVC (modelo-vista-controlador) y se ha desarrollado en R. Ademas, he aprendido a utilizar la herramienta 'Shiny' del software estadístico R, la cual permite crear herramientas web utilizando código de R. Una vez completado tanto el proceso de desarrollo del pipeline para el análisis de datos como la aplicación que permitirá llevar a cabo estos análisis de forma sencilla para aquellas personas que no tengan un amplio conocimiento de programación, se puede concluir que los objetivos propuestos en el plan de trabajo han sido alcanzados.

En cuanto a la planificación del trabajo, considero que ha sido llevada a cabo de forma adecuada, por lo que no se han excedido los plazos para cada una de las tareas propuestas. Sin embargo, a medida que se iba desarrollando tanto el pipeline como la aplicación, surgían nuevas dudas y errores en el código que han supuesto esfuerzos y dedicación temporal no planificada. Además se han incluido algunas tareas como la búsqueda de una forma de exponer la aplicación al público.

Como conclusión y aprendizaje de este proyecto destaco la dificultad de integrar en un entorno interactivo todo el proceso de análisis, la gran cantidad de variables a tener en cuenta y creación de una aplicación amigable intuitiva y sencilla. Estos problemas no los encuentras en una ejecución secuencial de código de programación. Shiny me ha parecido interesante para hacer desarrollos sencillos pero altamente complejo y poco intuitivo a la hora de realizar una aplicación como esta.

9 Glosario

- Bioinformática: Informática aplicada a la biología.
- Ómica: estudio de la totalidad o del conjunto de algo en el campo de la bioinf
- Clasificacion: predice una categoría.: predice una categoría
- Machine Learning
- Pipeline: conjunto de pasos o procesos concatenados que son necesarios para concluir un trabajo de, en este caso, análisis de datos y clasificación.
- Algoritmo: conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas.
- Modelo: algoritmo capaz de generalizar comportamientos e inferencias para un conjunto amplio (potencialmente infinito) de datos.
- MVP: Minimal viable product. Producto con suficientes características para satisfacer a los requisitos iniciales, y proporcionar retroalimentación para el desarrollo futuro.
- KNN: K nearest neighbourg
- Naive Bayes
- ANN: Artificial Neural Netework
- SVM: support vector machine
- MMH: maximal margin hyperplane
- Random Forest
- Decision Tree
- shiny
- shinydashboard
- UI: conjunto de código de Shiny que implementa la parte visual y estática de la app.
- Server: conjunto de código de Shiny que implementa la funcionalidad y parte visuald dinámica de la aplicación.
- MVC: modelo vista controlador

Bibliografía

[1] Introduction to ngs.learn how the technology works and what it can do for you.

[2] Support vector machines: A guide for beginners.

[3] Support vector machines: A guide for beginners.

[4] Visualización de datos en r.

[5] Modelo, vista y controlador, 2019. Último acceso 27 de Abril.

[6] Beeley, C. *Web Application Development with R Using Shiny*, first edition ed. Packt open source, 2013.

[7] Breiman, L., and Cutler, A. Breiman and cutler's random forests for classification and regression.

[8] Carmona, F. Taller 2: shiny: aplicaciones web interactivas con r. vi genaeio i 2015: Vi jornadas de ensenanza y aprendizaje de la estadistica y la investigacion operativa, 208, 21, 2016. Último acceso 15 de Mayo.

[9] González, N. V. C. Técnicas de m achine learning para el post-proceso de la predicción de la irradiancia. *UG* (2015), 6,7.

[10] Graves, E. E. Fast onehot encoding for data.frame.

[11] Gregory R. Warnes, Ben Bolker, T. L., and C, R. Various r programming tools for model fitting.

[12] Jones, M. T. Unsupervised learning for data classification.

[13] Kuhn, M. Classification and regression training.

[14] Lantz, B. *Machine Learning with R. Discover how to build machine learning algorithms, prepare data, and dig deep into data prediction techniques with R*, seccond edition ed. Packt open source, 2015.

[15] M Churko, J., Mantalas, G., P Snyder, M., and Wu, J. Overview of high throughput sequencing technologies to elucidate molecular pathways in cardiovascular diseases. *Circulation research 112* (06 2013), 1613–23.

[16] Pedersen, T. L. shinyfiles.

[17] project, R. The r project for statistical computing, 2019. Último acceso 15 de Mayo.

[18] Ripley, B. Feed-forward neural networks and multinomial log-linear models.

[19] Ripley, B. Functions for classification.

[20] Robin, X. Display and analyze roc curves.

[21] Rodríguez, J. H. . S. Cascada con retroalimentación, 2012. Último acceso 2 de Abril.

[22] RStudio. Shiny, 2019. Último acceso 15 de Mayo.

[23] RStudio. Shiny dashboard, 2019. Último acceso 18 de Mayo.

[24] Science, T. D. Supervised machine learning: Classification.

[25] Science, T. D. Classification on large datasets, 2019. Último acceso 12 Mayo.

[26] Therneau, T. rpart.

[27] W.A, A., and S.M, E. Machine learning methods for spam e-mail classification. *International Journal of Computer Science Information Technology 3* (02 2011).

[28] Wickham, H. Create elegant data visualisations using the grammar of graphics.

[29] Wickham, H. Tools for working with categorical variables (factors).

[30] Wikipedia. R, 2019. Último acceso 2 de Junio.

A Imágenes

A.1.

A continuación se muestra el estado de la interfaz a través de las imágenes:

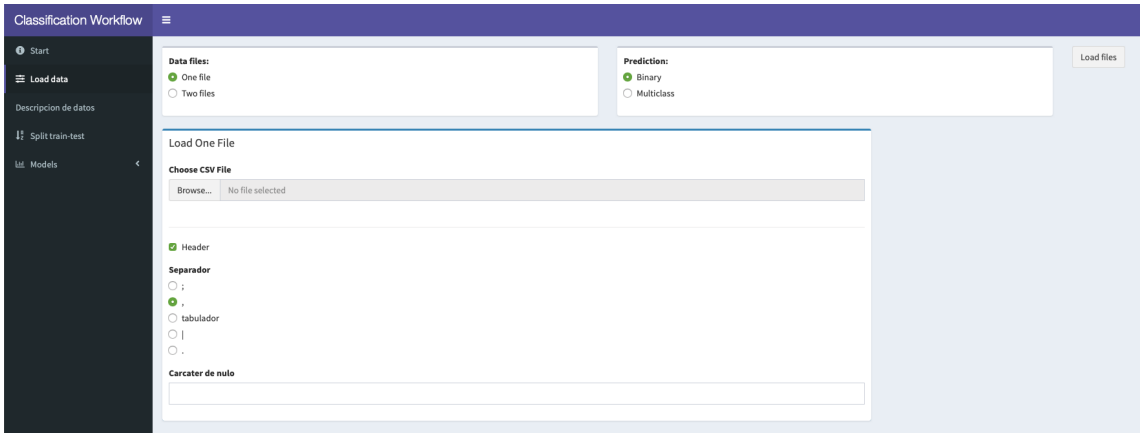


Figura A.1.1: Carga de datos con un fichero

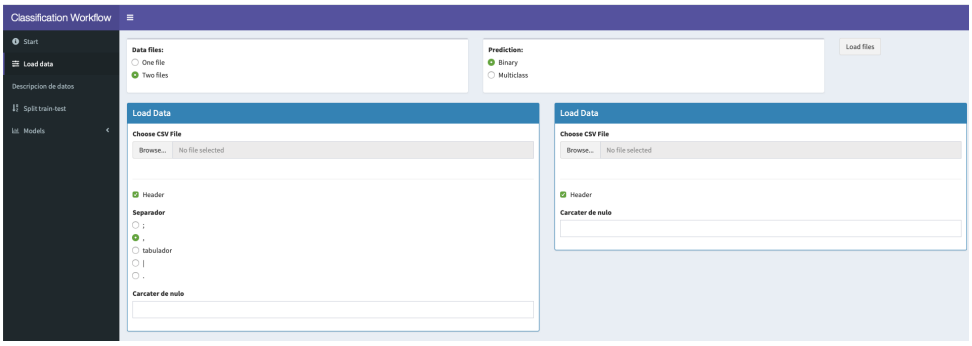


Figura A.1.2: Carga de datos con dos ficheros

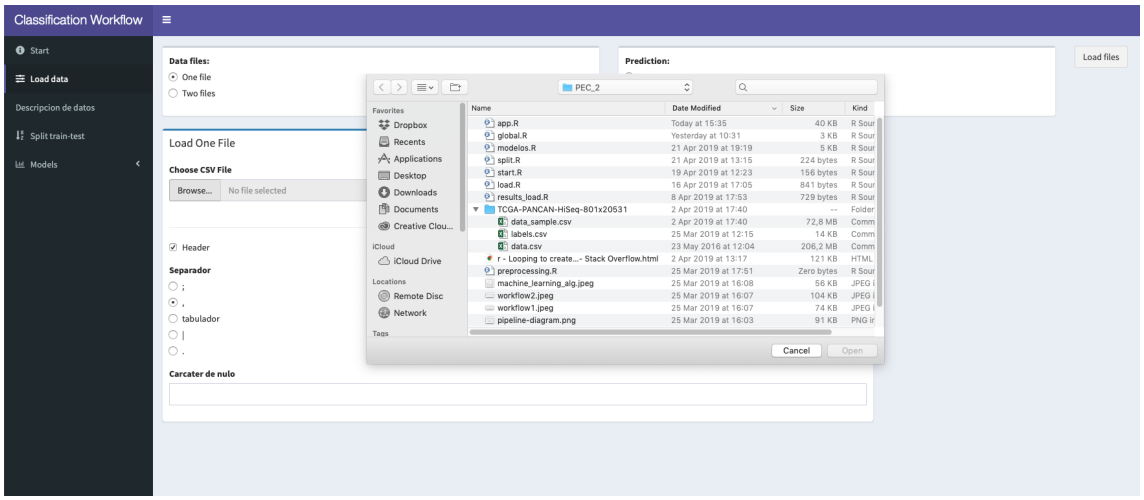


Figura A.1.3: Selección de fichero

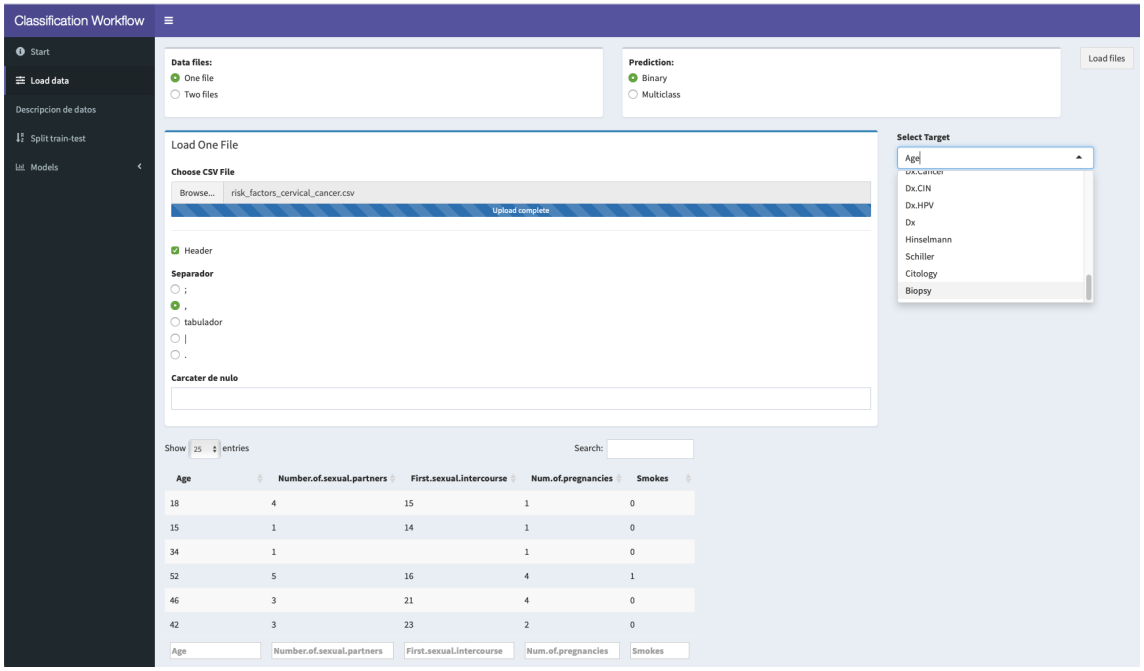


Figura A.1.4: Selección de la Target

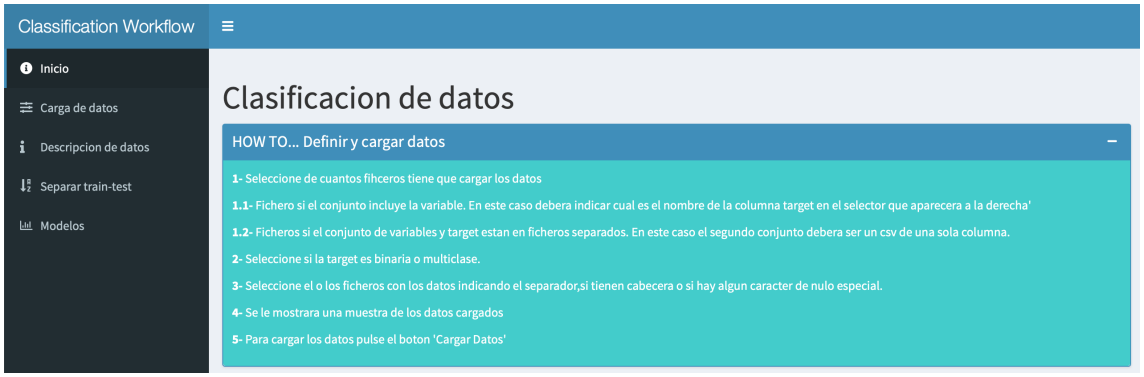


Figura A.1.18: Pagina de inicio. Explicacion-1

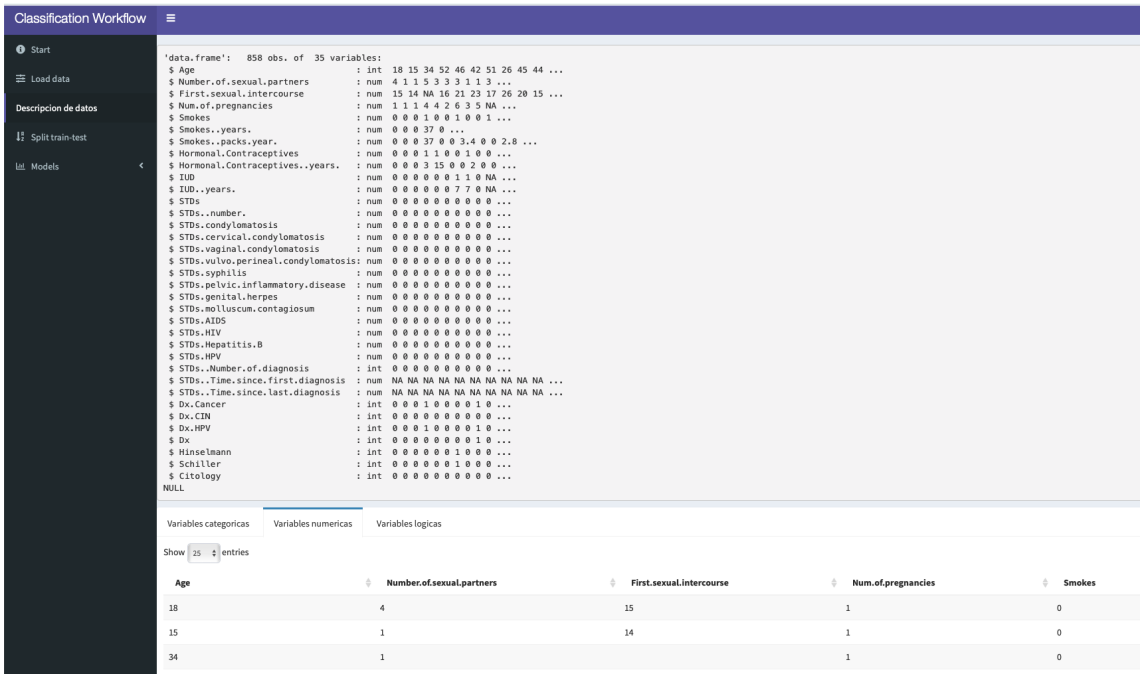


Figura A.1.5: Descripción del conjunto de datos

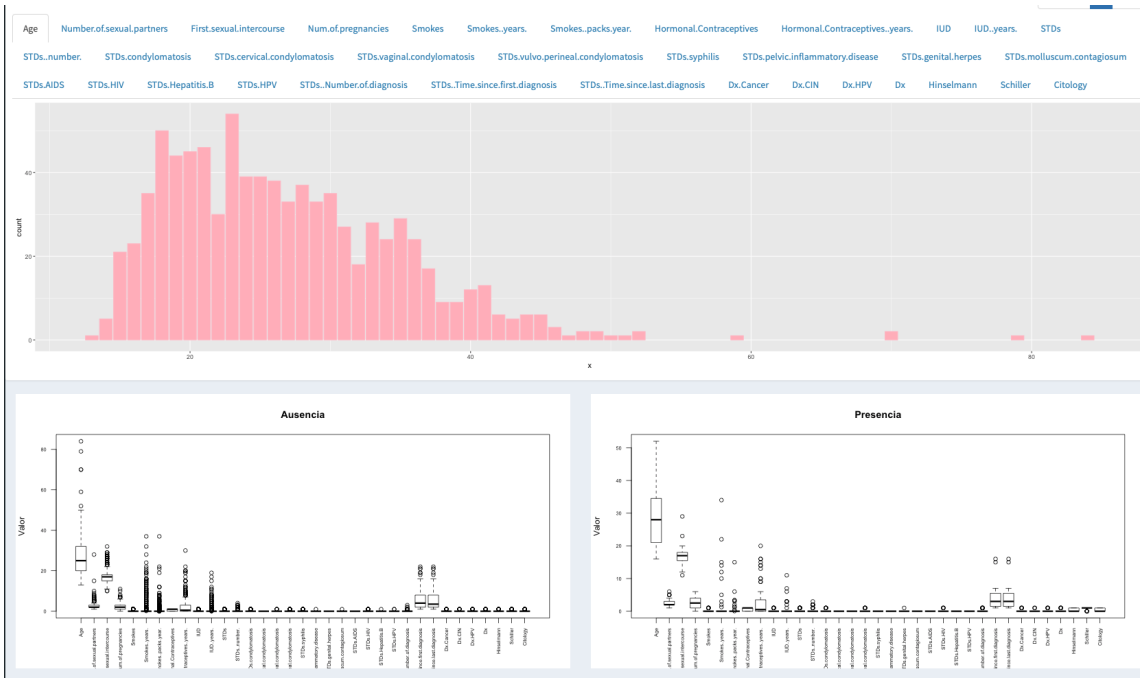


Figura A.1.6: Visualización de los datos

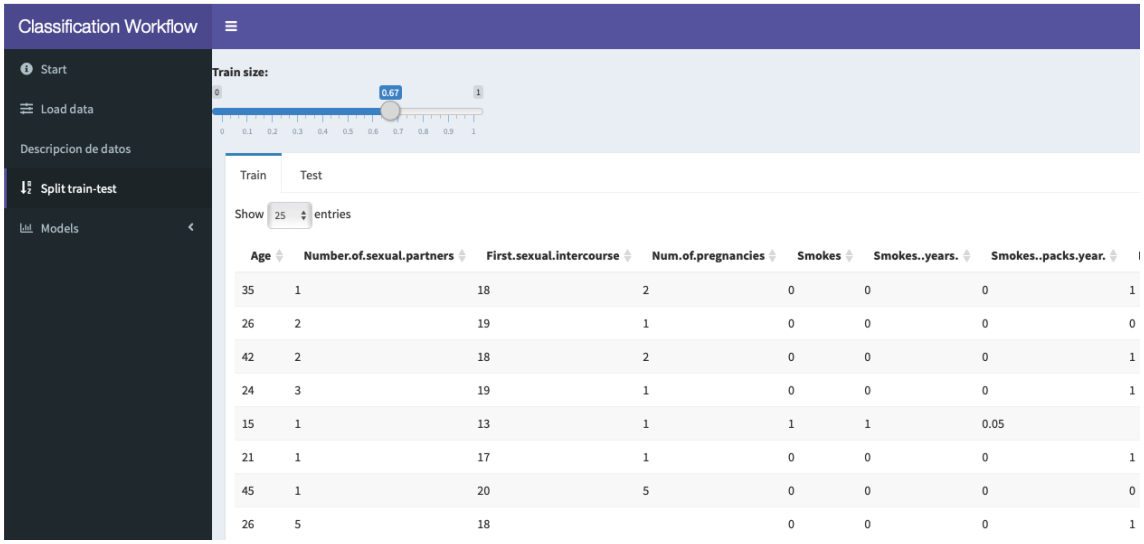


Figura A.1.7: Selección conjunto de entrenamiento

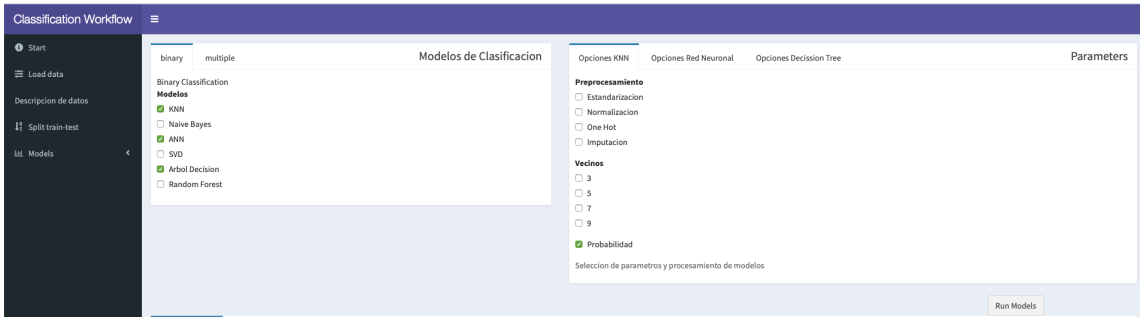


Figura A.1.8: Selección de modelos

Figura A.1.20: Pagina de inicio. Explicacion-3

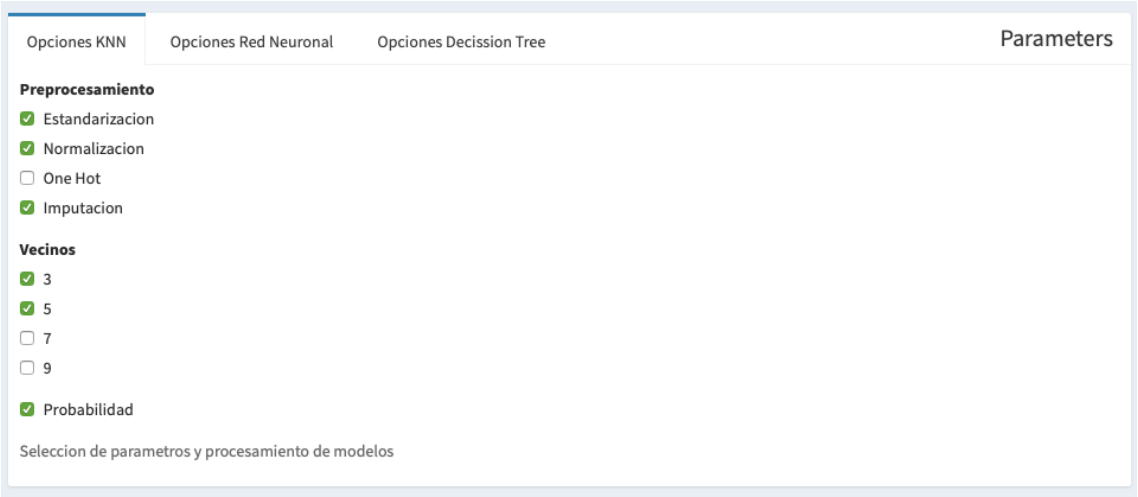


Figura A.1.9: Parametros K-NN

Figura A.1.10: Parámetros ANN

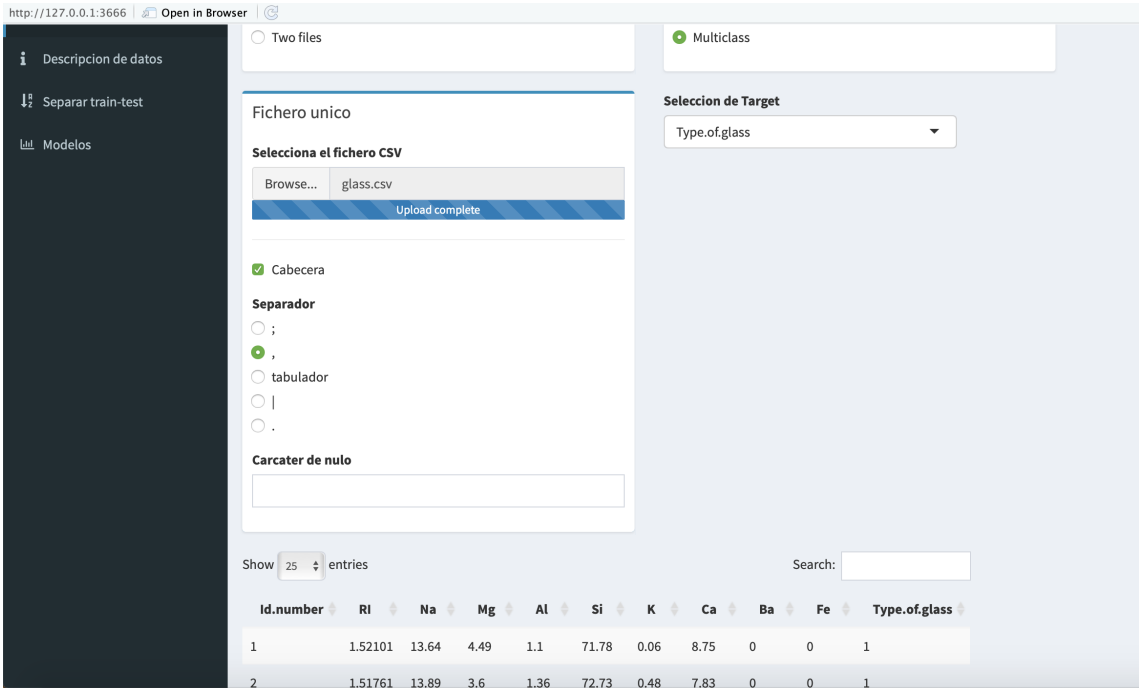


Figura A.1.21: Carga de datos con target multiple

Figura A.1.11: Parámetros Random Forest

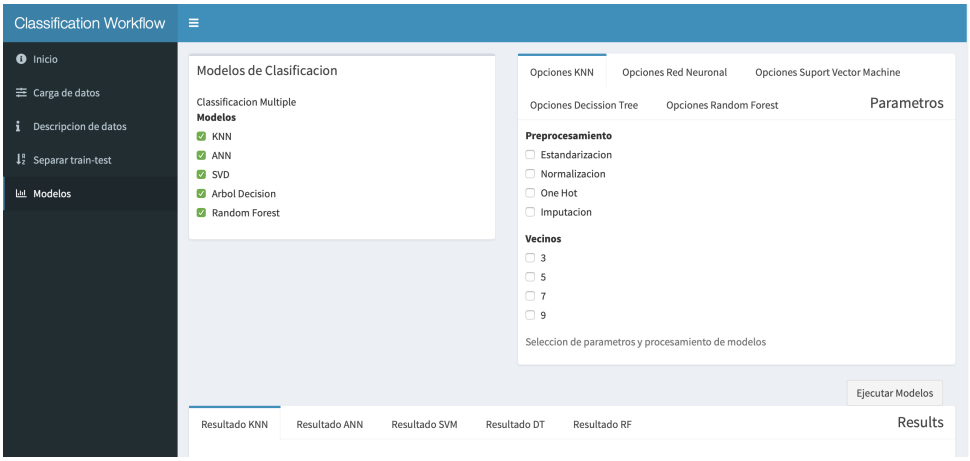


Figura A.1.22: Modelos para clasificacion múltiple

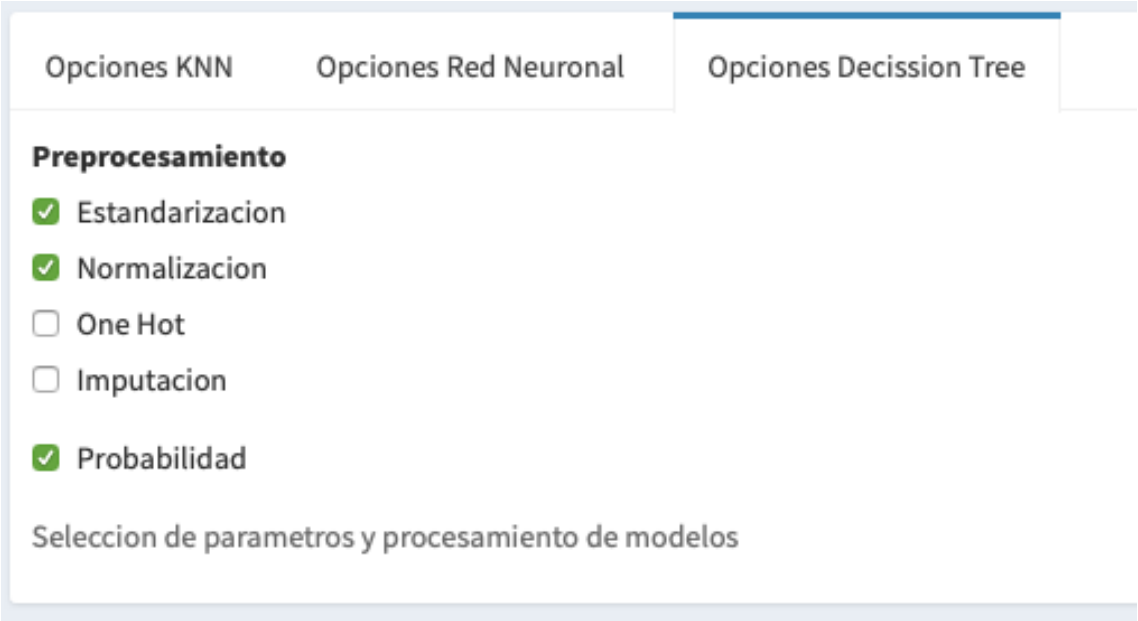


Figura A.1.12: Parámetros Árbol de Decisión

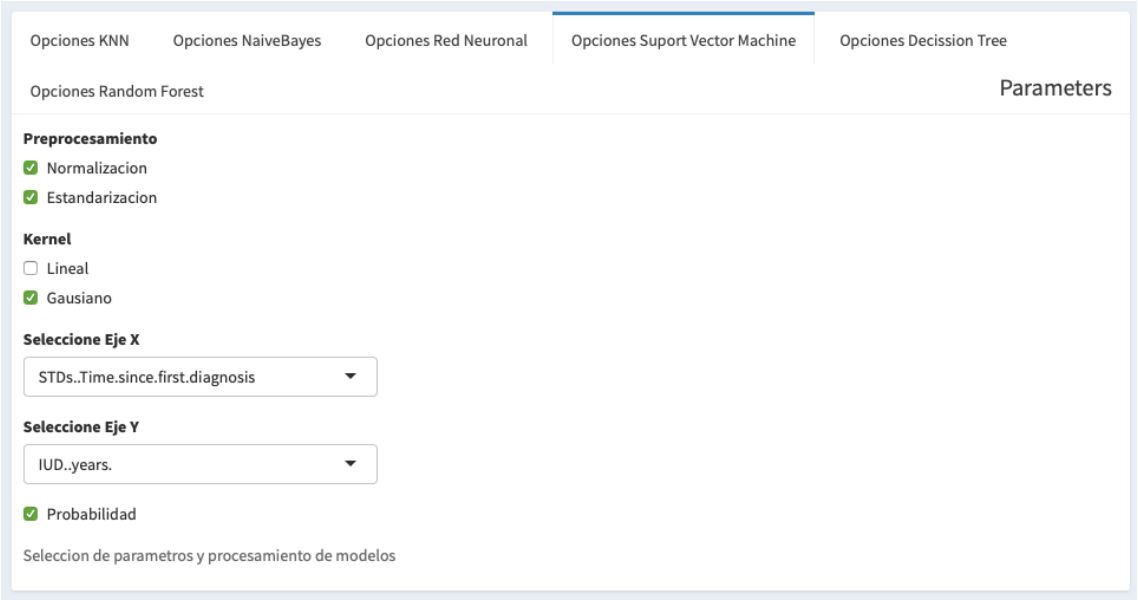
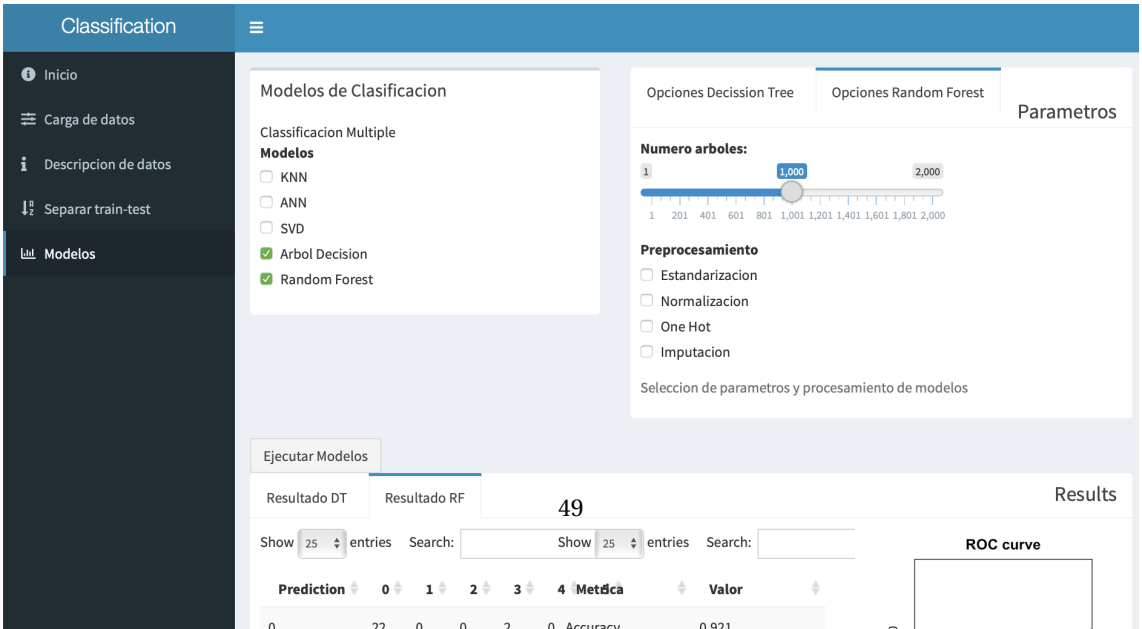


Figura A.1.13: Parámetros Support vector machine



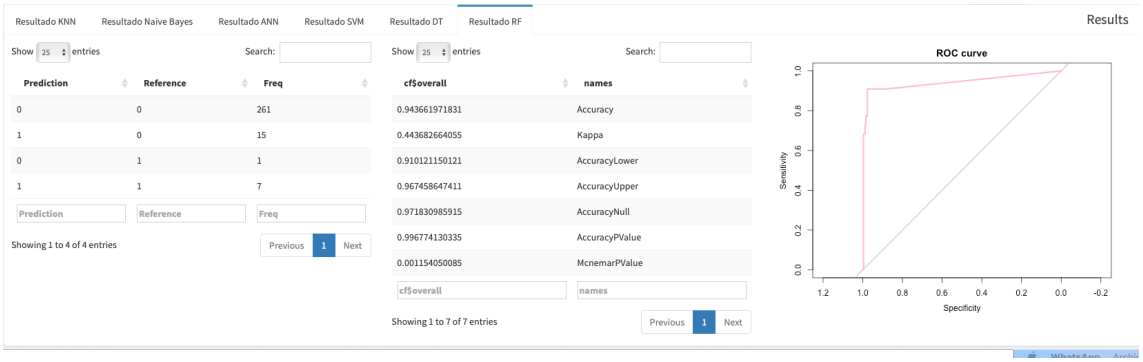


Figura A.1.14: Resultados random forest

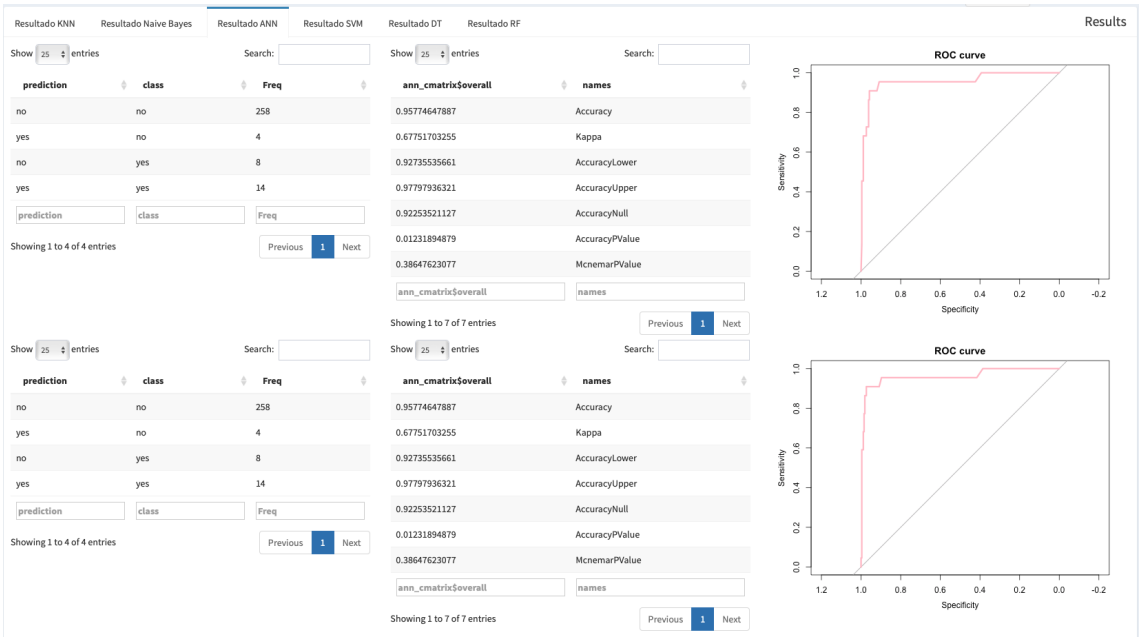


Figura A.1.15: Resultado red neuronal

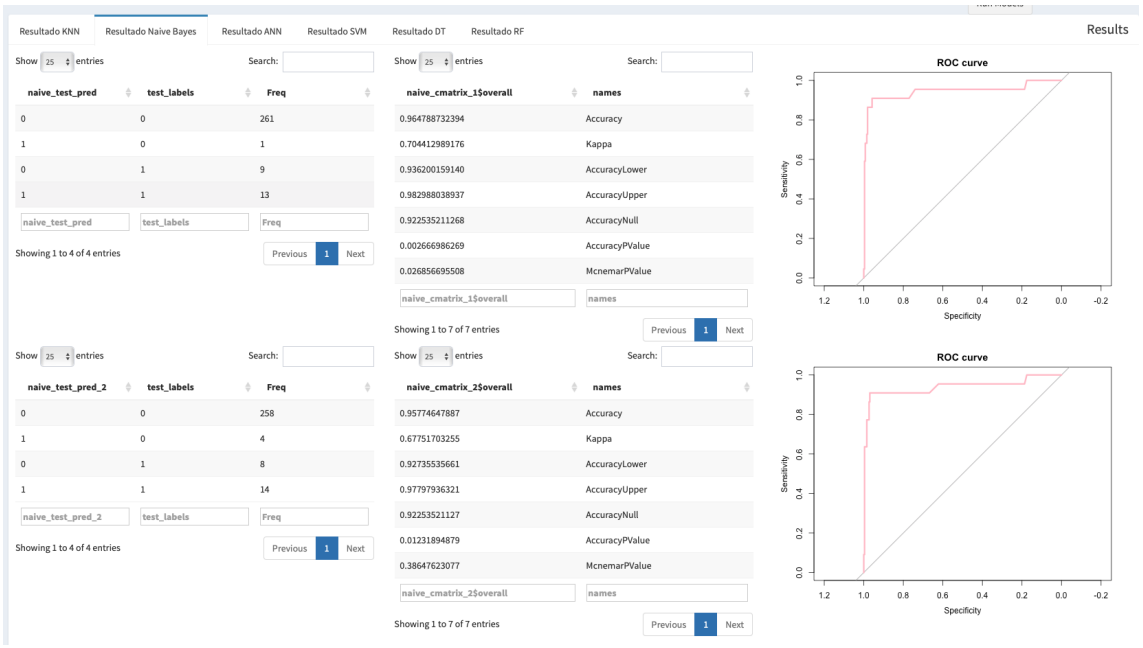


Figura A.1.16: Resultado Naive Bayes

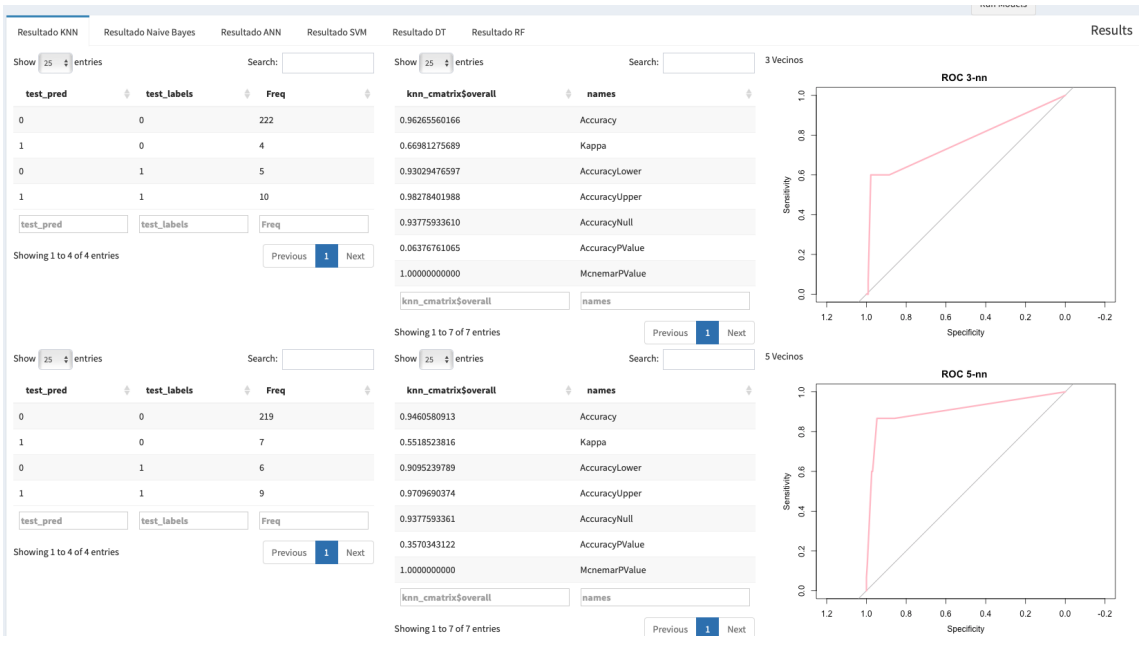


Figura A.1.17: Resultado Vecinos cercanos

B Código de la aplicación

B.1. APP Code

```
## app.R ##
source("global.R")
source("load.R")
source('split.R')
source('start.R')
source('export.R')
source('preprocessing.R')
source('modelos.R')
source("results_load.R")
#Dashboard header carrying the title of the dashboard

################################################################################################
#####                                        UI                                            #####
################################################################################################
#    * Titulo
#    * Men?? laterarl
#    * Cuerpo
################################################################################################

header <- dashboardHeader(title = "Classification Workflow")
#Sidebar content of the dashboard
sidebar <- dashboardSidebar(
  sidebarMenu(
    # Setting id makes input$tabs give the tabName of currently-selected tab

    id = "tabs",
    menuItem("Inicio", tabName = "inicio", icon = icon("info-circle")),
    customMenuItem(
      "Carga de datos",
      tabName = "load",
      icon = icon("sliders")
    ),
    menuItem(
      "Descripcion de datos",
      tabName = "describe",
      icon = icon("fas fa-info")
    ),
    menuItem(
      "Separar train-test",
      tabName = "split",
      icon = icon(" glyphicon glyphicon-sort-by-alphabet")
    ),
    customMenuItem(
      "Modelos",
      tabName = "models",
      icon = icon("bar-chart-o")
    )
    #menuItem("Exportacion", tabName = "export", icon = icon("download"))
  )
)

body <- dashboardBody(tabItems(start_data, load_data, results_load, split_data, models))
#completing the ui part with dashboardPage
ui <-
  dashboardPage(title = 'This is my Page title', header, sidebar, body, skin =
                  'blue')


################################################################################################
#####                                    SERVER                                            #####
################################################################################################
#    * Funciones de carga de datos
#    * Funciones de descripcion y analisis exploratorio de datos
#    * Funciones de separacion en train y test
#    * Funciones de generacion  de interfaz grafica dinamicos
#    * Funciones de selecci??n de modelos y par??metros
#    * Funciones de modelizacion
#    * Funciones de generaci??n de graficos
################################################################################################

```

```
71
72 # create the server functions for the dashboard
73 server <- function(input, output, session) {
74   ################################################
75   #  Inicializaci??n de variables reactivas     #
76   ################################################
77   loaded <- FALSE
78   final_eje_models <- reactiveVal(FALSE)
79   options(shiny.maxRequestSize = 100 * 1024 ^ 2)
80   hist_out <- reactiveVal(NULL)
81   df_num_norm <- reactiveVal(NULL)
82   df_cat_encoded <- reactiveVal(NULL)
83   df_cat <- reactiveVal(NULL)
84   df_num <- reactiveVal(NULL)
85   df_log <- reactiveVal(NULL)
86   df_features <- reactiveVal(NULL)
87   df_label <- reactiveVal(NULL)
88   train_indices  <- reactiveVal(NULL)
89   inFile1 <- reactive({
90     input$data_file
91   })
92   inFile2 <- reactive({
93     input$feature_file
94   })
95   inFile3 <- reactive({
96     input$label_file
97   })
98   box_a <- reactiveVal(NULL)
99   box_p <- reactiveVal(NULL)
100  boxes <- reactiveValues(
101    box_1 = FALSE,
102    box_2 = FALSE,
103    box_3 = FALSE,
104    box_4 = FALSE,
105    box_5 = FALSE,
106    box_6 = FALSE,
107    box_7 = FALSE
108  )
109
110
111  ################################################
112  #  Funciones asociadas a la lectura de datos #
113  ################################################
114
115  data <- reactive({
116    if (is.null(inFile1()))
117      return(NULL)
118    read.csv(
119      inFile1()$datapath,
120      header = input$header,
121      sep = input$sep,
122      stringsAsFactors = FALSE,
123      dec = ".",
124      na.strings = c("?", "", input$nan_string)
125    )
126  })
127  feature <- reactive({
128    if (is.null(inFile2()))
129      return(NULL)
130    df <-
131      read.csv(
132        inFile2()$datapath,
133        header = input$header,
134        sep = input$sep,
135        stringsAsFactors = FALSE,
136        na.strings = c("?", "", input$nan_string),
137        dec = "."
138      )
139    return(df)
140  })
141
142  label <- reactive({
143    if (is.null(inFile3()))
144      return(NULL)
145    df <-
146      read.csv(
147        inFile3()$datapath,
148        header = input$header,
149        quote = "\"",
150        sep = input$sep,
151        stringsAsFactors = FALSE,
152        na.strings = input$nan_string
153      )
154    a <- ncol(df)
155    if (a != 1) {
156      shinyalert("Oops!",
157                 "El dataset de label debe tener solo una columna.",
158                 type = "error")
159      showNotification("This is a notification.",
```

```
                      type = "error",
                      duration = 5)
    }
    return(df)
  })

  ########################################################
  #   GUI reactiva para la pantalla de carga de datos    #
  ########################################################

  output$distPlot <- renderUI({
    n_files <- switch(input$num_files,
                      one = "one",
                      two = "two",
                      one)

    if (n_files == 'one') {
      fluidRow(column(
        5,
        box(
          title = "Fichero unico",
          status = "primary",
          width = NULL,
          fileInput(
            "data_file",
            "Selecciona el fichero CSV",
            accept = c(
              "text/csv",
              "text/comma-separated-values,text/plain",
              ".csv"
            )
          ),
          tags$hr(),
          checkboxInput("header", "Cabecera", TRUE),
          radioButtons(
            "sep",
            "Separador",
            choices =  c(
              ";" = ";",
              "," = ",",
              "tabulador" = "\t",
              "|" = "|",
              "." = "."
            ),
            selected = ','
          ),
          textInput("nan_string", "Carcater de nulo")
        )
      ), uiOutput("radio"))

    } else{
      fluidRow(column(
        5,
        box(
          title = "Fichero de caracteristicas",
          width = NULL,
          status = "primary",
          solidHeader = TRUE,
          fileInput(
            "feature_file",
            "Selecciona el fichero CSV",
            accept = c(
              "text/csv",
              "text/comma-separated-values,text/plain",
              ".csv"
            )
          ),
          tags$hr(),
          checkboxInput("header", "Cabecera", TRUE),
          radioButtons(
            "sep",
            "Separador",
            choices =  c(
              ";" = ";",
              "," = ",",
              "tabulador" = "\t",
              "|" = "|",
              "." = "."
            ),
            selected = ','
          ),
          textInput("nan_string_f", "Carcater de nulo")
        )
      ),
      column(
        5,
        box(
          title = "Fichero de etiquetas",
          width = NULL,
```

```
249           status = "primary",
250           solidHeader = TRUE,
251           fileInput(
252             "label_file",
253             "Selecciona el fichero CSV",
254             accept = c(
255               "text/csv",
256               "text/comma-separated-values,text/plain",
257               ".csv"
258             )
259           ),
260           tags$hr(),
261           checkboxInput("header", "Cabecera", TRUE),
262           textInput("nan_string_l", "Carcater de nulo")
263         )
264       ))
265     }
266   })
267
268   ########################################################
269   #  Visualizacion de los datos cargados por el usuario  #
270   ########################################################
271   observeEvent(input$num_files, {
272     fluidRow(uiOutput("tables"))
273   })
274   output$tables <- renderUI({
275     n_files <- switch(input$num_files,
276                       one = "one",
277                       two = "two",
278                       one)
279
280     if (n_files == 'one') {
281       fluidRow(column(9, dataTableOutput("table1")))
282
283     } else{
284       fluidRow(column(9, dataTableOutput("table2")),
285                column(1),
286                column(2, dataTableOutput("table3")))
287     }
288
289   })
290
291   output$table1 <- renderDataTable(as.data.frame(data()),
292                                    options = list(autoWidth = TRUE, scrollX = T))
293
294
295   output$table2 <- renderDataTable({
296     df <- as.data.frame(feature())
297     if (ncol(df) > 5) {
298       sample <- head(df[, 1:17])
299     } else{
300       sample <- head(df)
301     }
302     df
303   }, options = list(autoWidth = TRUE, scrollX = T))
304
305   output$table3 <- renderDataTable({
306     df <- as.data.frame(label())
307     df
308   }, options = list(autoWidth = TRUE, scrollX = T))
309
310
311   ########################################################
312   # GUI reactiva para la pantalla de selecci??n de target #
313   ########################################################
314
315   output$radio <- renderUI({
316     if (is.null(inFile1()))
317       return(NULL)
318     vchoices <- names(data())
319     column(5,
320            selectInput(
321              inputId =  "label_col",
322              "Seleccion de Target",
323              choices = vchoices
324            ))
325   })
326
327
328   observe({
329     vchoices <- names(data())
330     updateRadioButtons(
331       session = session,
332       inputId =  "column1",
333       choices = vchoices,
334       selected = vchoices[1]
335     )
336   })
337
```

```
338
339   ########################################################
340   #   Ejecucion de la carga de datos. Paso al describe   #
341   ########################################################
342   observeEvent(input$go_describe, {
343     if (is.null(data()) & is.null(feature())) {
344       return(NULL)
345     }
346     n_files <- input$num_files
347
348     if (n_files == 'one') {
349       df <- as.data.frame(data())
350       df[is.na(df)] <- 0
351       selected <- input$label_col
352       df_features <- df[,!(names(df) %in% selected)]
353       df_label <- df[, selected, drop = FALSE]
354       names(df_label) <- c('Class')
355       df_label$Class <- as.factor(df_label$Class)
356       l1 <- unique(df_label[[1]])
357       max_v <- 1
358       if (class(l1) == "numeric") {
359         l1 <- l1 - min(l1)
360         max_v <- max(l1)
361       }
362       l2 <- seq(max_v + 1) - 1
363       if (!(identical(l1, l2))) {
364         ranks <- rank(-table(df_label$Class), ties.method = "first")
365         df_label$Class <- ranks[df_label$Class] - 1
366       } else{
367         df_label$Class <- df_label$Class - min(l1)
368       }
369       df_label[is.na(df_label)] <- 0
370     }
371     else{
372       df_features <- as.data.frame(feature())
373       df_features[is.na(df_features)] <- 0
374       df_label <- as.data.frame(label())
375       names(df_label) <- c('Class')
376       df_label$Class <- as.factor(df_label$Class)
377       l1 <- unique(df_label[[1]])
378       max_v <- 1
379       if (class(l1) == "numeric") {
380         l1 <- l1 - min(l1)
381         max_v <- max(l1)
382       }
383       l2 <- seq(max_v + 1) - 1
384       if (!(identical(l1, l2))) {
385         ranks <- rank(-table(df_label$Class), ties.method = "first")
386         df_label$Class <- ranks[df_label$Class] - 1
387       } else{
388         df_label$Class <- df_label$Class - min(l1)
389       }
390       df_label[is.na(df_label)] <- 0
391
392     }
393
394     ####################################
395     # Separaci??n de columnas por tipo #
396     ####################################
397
398     df_features(df_features)
399     df_label(df_label)
400     cls <- sapply(df_features(), class)
401     new_char <-
402       df_features() %>% dplyr::select(which(cls == "factor" |
403                                             cls == "character"))
404     new_num <-
405       df_features() %>% dplyr::select(which(cls == "numeric" |
406                                             cls == 'integer'))
407     new_log <-
408       df_features() %>% dplyr::select(which(cls == "logical"))
409     df_num(new_num)
410     df_cat(new_char)
411     df_log(new_log)
412     values_label <- as.list(unique(df_label[1]))[[1]]
413     a <- c()
414     if (length(names(df_cat())) > 0) {
415       df_cat_sample = df_cat()
416       for (i in 1:length(names(df_cat_sample))) {
417         if (nrow(unique(df_cat_sample[i])) > min(52, (nrow(df_cat_sample) / 2))) {
418           a <- append(a, i)
419         } else{
420           ranks <-
421             rank(-table(df_cat_sample[[paste0(names(df_cat_sample)[i])]]), ties.method =
422                    "first")
423           df_cat_sample[[paste0(names(df_cat_sample)[i])]] <-
424             ranks[df_cat_sample[[paste0(names(df_cat_sample)[i])]]]
425         }
426       }
```

```
427
428       df_cat_use <- df_cat_sample[-a]
429       cat_data <- as.data.frame(lapply(df_cat_use, factor))
430       df_cat(cat_data)
431     }
432
433     numeric_data_scaled <- as.data.frame(scale(new_num))
434     numeric_data_norm <-
435       as.data.frame(lapply(numeric_data_scaled, normalize))
436     df_num_norm(numeric_data_norm)
437     if (input$num_files == "one") {
438       hist_out = lapply(df_features(), function(x) {
439         ggplot(data.frame(x), aes(x)) + geom_histogram(
440           binwidth = 1,
441           color = "pink",
442           fill = "lightpink",
443           stat = "count"
444         )
445       })
446       hist_out(hist_out)
447     } else{
448       hist_out = lapply(df_features(), function(x) {
449         ggplot(data.frame(x), aes(x)) + geom_histogram(binwidth = 1,
450                                                        color = "pink",
451                                                        fill = "lightpink")
452       })
453       hist_out(hist_out)
454     }
455
456
457     if (input$type_label == "bin") {
458       ausencia <- df_num()[df_label()$Class == 0, ]
459       presencia <- df_num()[df_label()$Class == 1, ]
460     } else{
461       for (i in 1:length(values_label)) {
462         df_class <-
463           as.data.frame(df_features()[df_label()$Class == values_label[i], ])
464         boxes[[paste0("box_", i)]] <- df_class
465       }
466     }
467
468   })
469   ##########################################################
470   # GUI reactiva para la visualizacion de datos por tipo #
471   ##########################################################
472
473   output$tables_by_type <- renderUI({
474     fluidRow(dataTableOutput("table_cat"))
475     fluidRow(dataTableOutput("table_num"))
476     fluidRow(dataTableOutput("table_log"))
477   })
478
479
480   #show features by type
481   output$table_cat <- renderDataTable({
482     if (is.null(df_cat())) {
483       return(NULL)
484     }
485     as.data.frame(df_cat())
486   }, options = list(autoWidth = TRUE, scrollX = T))
487
488   output$table_num <- renderDataTable({
489     if (is.null(df_num())) {
490       return(NULL)
491     }
492     if (ncol(df_num()) > 5) {
493       sample <- head(df_num()[, 1:5])
494     }
495     else{
496       sample <- head(df_num())
497     }
498     as.data.frame(df_num())
499   }, options = list(autoWidth = TRUE, scrollX = T))
500
501   output$table_log <- renderDataTable({
502     if (is.null(df_log())) {
503       return(NULL)
504     }
505     if (ncol(df_log()) > 5) {
506       sample <- head(df_log()[, 1:5])
507     }
508     else{
509       sample <- head(df_log())
510     }
511     as.data.frame(df_log())
512   }, options = list(autoWidth = TRUE, scrollX = T))
513
514   ##########################################################
515   # Graficas para los diagarmamas descriptivos de los datos   #
```

```
516   ############################################################
517   for (i in 1:7) {
518     local({
519       my_i <- i
520       plotname <- paste0("plot_box_", my_i)
521       output[[plotname]] <- renderPlot({
522         boxplot(
523           boxes[[paste0("box_", my_i)]],
524           col = 'brown',
525           main = paste0("Clase ", my_i),
526           cex.axis = 0.4
527         )
528         abline(h = 0.5, lwd = 2)
529       })
530     })
531   }
532
533   for (i in 1:100) {
534     local({
535       my_i <- i
536       plotname <- paste0("plot_hist_", my_i)
537       output[[plotname]] <- renderPlot({
538         if (my_i > ncol(df_num()))
539           return(NULL)
540         hist_out()[[my_i]]
541       })
542     })
543   }
544
545   output$str_data <- renderPrint({
546     if (is.null(df_features()))
547       return("Aun no ha cargado los datos")
548     (str(df_features()))
549   })
550
551   output$box_aus <- renderPlot({
552     if (is.null(df_label()))
553       return(NULL)
554     c <- unique(df_label()$Class)[1]
555     ausencia <- df_num()[df_label()$Class == c, ]
556     boxplot(
557       ausencia,
558       cex.axis = 0.6,
559       ylab = 'Valor',
560       main = paste0("Clase ", c),
561       las = 2
562     )
563     abline(h = 500, col = 'red')
564
565   })
566
567   output$box_pre <- renderPlot({
568     if (is.null(df_label()))
569       return(NULL)
570     c <- unique(df_label()$Class)[2]
571     presencia <- df_num()[df_label()$Class == c, ]
572     boxplot(
573       presencia,
574       cex.axis = 0.6,
575       ylab = 'Valor',
576       main = paste0("Clase ", c),
577       las = 2
578     )
579     abline(h = 500, col = 'red')
580
581   })
582
583   output$plot_box_class <- renderUI({
584     if (is.null(df_features())) {
585       return()
586     }
587     vchoices <- as.list(unique(df_label()[1]))[[1]]
588     do.call(tabsetPanel, lapply(1:length(vchoices), function(i) {
589       tabPanel(paste0(vchoices[i]), plotOutput(paste0("plot_box_", i)))
590     }))
591   })
592
593   output$plot_hist_num <- renderUI({
594     if (is.null(df_num())) {
595       return()
596     }
597     vchoices <- names(df_num())
598     do.call(tabsetPanel, lapply(1:min(100, length(vchoices)), function(i) {
599       tabPanel(paste0(vchoices[i]), plotOutput(paste0("plot_hist_", i)))
600     }))
601   })
602
603   ############################################################
604   # Divisi??n de datos en train y test visualizaci??n de tablas #
```

```
605   ############################################################
606
607   output$tables_split <- renderUI({
608     #67%
609     if (is.null(df_features()) | is.null(df_label())) {
610       return()
611     }
612     if (input$percent_train == 0.67) {
613       perc <- 0.67
614     } else{
615       perc <- input$percent_train
616     }
617     smp_size <- floor(perc * nrow(df_features()))
618
619     #fijamos la semilla para que los resultados sean reproducibles
620     set.seed(1234)
621     train_indices <-
622       sample(seq_len(nrow(df_features())), size = smp_size)
623     train_indices(train_indices)
624     fluidRow(column(width = 12, tabBox(
625       width = NULL,
626       tabPanel("Entrenamiento", dataTableOutput("table_train")),
627       tabPanel("Validacion", dataTableOutput("table_test"))
628     )))
629   })
630
631   output$table_train <- renderDataTable({
632     train <- df_features()[train_indices(),]
633   }, options = list(autoWidth = TRUE, scrollX = T))
634
635   output$table_test <- renderDataTable({
636     test <- df_features()[-train_indices(),]
637     test
638   }, options = list(autoWidth = TRUE, scrollX = T))
639
640   output$vals <- renderTable({
641     col <- input$vl
642     table(df_cat()$col)
643   })
644   output$cat_dist <- renderUI({
645     vchoices <- names(df_cat())
646     shiny:::buildTabset(id = "vl",
647                         lapply(1:length(vchoices), function(i) {
648                           tabPanel(paste0(vchoices[i]), renderTable({
649                             col <- vchoices[i]
650                             print(col)
651                             tableOutput(table(df_cat()$col))
652                           }))
653                         }),
654                         paste0("nav nav-", "tabs")) %>% (function(x) {
655                           tags$div(class = "tabbable", x[[1]], x[[2]])
656                         })
657   })
658
659
660   ############################################################
661   # GUI interactiva de modelos de clasficiacion disponibles   #
662   ############################################################
663
664   observeEvent(input$alg_models_m, {
665     all_modelos = c(1, 2, 3, 4, 5, 6)
666     name_modelos = c("knn", "nb", "ann", "svm", "dt", "rf")
667     modelos <- input$alg_models_m
668     for (m in modelos) {
669       showTab(inputId = "tabset1", target = name_modelos[as.numeric(m)])
670       showTab(inputId = "model_results", target = name_modelos[as.numeric(m)])
671     }
672   })
673
674   observeEvent(input$alg_models_m, {
675     all_modelos = c(1, 2, 3, 4, 5, 6)
676     name_modelos = c("knn", "nb", "ann", "svm", "dt", "rf")
677     modelos <- input$alg_models_m
678     for (m in all_modelos) {
679       if (!is.element(m, modelos)) {
680         hideTab(inputId = "tabset1", target = name_modelos[as.numeric(m)])
681         hideTab(inputId = "model_results", target = name_modelos[as.numeric(m)])
682       }
683     }
684   })
685   observeEvent(input$alg_models_b, {
686     all_modelos = c(1, 2, 3, 4, 5, 6)
687     name_modelos = c("knn", "nb", "ann", "svm", "dt", "rf")
688     modelos <- input$alg_models_b
689     for (m in modelos) {
690       showTab(inputId = "tabset1", target = name_modelos[as.numeric(m)])
691       showTab(inputId = "model_results", target = name_modelos[as.numeric(m)])
692     }
693   })
```

```
694
695   observeEvent(input$alg_models_b, {
696     all_modelos = c(1, 2, 3, 4, 5, 6)
697     name_modelos = c("knn", "nb", "ann", "svm", "dt", "rf")
698     modelos <- input$alg_models_b
699     for (m in all_modelos) {
700       if (!is.element(m, modelos)) {
701         hideTab(inputId = "tabset1", target = name_modelos[as.numeric(m)])
702         hideTab(inputId = "model_results", target = name_modelos[as.numeric(m)])
703       }
704     }
705   })
706
707   ##########################################
708   #  Ejecucion de los modelos seleccionados #
709   ##########################################
710
711   observeEvent(input$model_eje, {
712     reset_models("1")
713     if (input$type_label == 'bin') {
714       modelos <- input$alg_models_b
715     } else{
716       modelos <- input$alg_models_m
717     }
718     print(modelos)
719     if (is.null(train_indices())) {
720       perc <- 0.67
721       smp_size <- floor(perc * nrow(df_features()))
722
723       #fijamos la semilla para que los resultados sean reproducibles
724       set.seed(1234)
725       if (nrow(df_features()) > 0) {
726         train_indices <-
727           sample(seq_len(nrow(df_features())), size = smp_size)
728       } else{
729         train_indices <- FALSE
730       }
731     } else{
732       train_indices <- train_indices()
733     }
734     if (!identical(train_indices, FALSE)) {
735       train_ind <- train_indices
736
737       data_num <- df_num()
738       data_cat <- df_cat()
739       y <- df_label()$Class
740       data_cat[is.na(data_cat)] <- 0
741       data_num[is.na(data_num)] <- 0
742
743       for (m in modelos) {
744         print("Entro en el loop de modelos")
745         if (m == 1) {
746           #knn
747           choices <- input$knn_opt
748           for (c in choices) {
749             if (c == 1) {
750               data_num = as.data.frame(scale(data_num))
751               print(head(data_num))
752             } else if (c == 2) {
753               data_num = as.data.frame(lapply(data_num, normalize))
754               print(head(data_num))
755             } else if (c == 4) {
756               data_num[is.na(data_num)] <- 0
757               data_cat = data_cat %>% mutate_if(is.factor,
758                                                 fct_explicit_na,
759                                                 na_level = 0)
760             } else{
761
762             }
763
764           }
765
766           knn_data <- data_num
767           knn_data[, colnames(data_cat)] <- data_cat
768           knn_data[, colnames(df_log())] <- df_log()
769           knn_data[is.na(knn_data)] <- 0
770           train <- knn_data[train_ind,]
771           test <- knn_data[-train_ind,]
772           train_labels <- y[train_ind]
773           test_labels <- y[-train_ind]
774           set.seed(1234)
775           ks <- input$k_values
776           withProgress(message = 'Generando modelos KNN',
777                        detail = "part 0",
778                        value = 0,
779                        {
780                          if (length(ks) == 0) {
781                            showNotification("Seleccione un numero de vecinos")
782                          } else{
```

```
                        resum <- data.frame(ks,
                                            FN = NA,
                                            FP = NA,
                                            mal_clas = NA)
                        #iteracion sobre las k
                        j <- 0
                        if (input$type_label == 'mult') {
                          prob <- FALSE
                        } else{
                          prob <-  input$prob_knn
                        }

                        for (i in ks) {
                          j <- j + 1
                          #Algoritmo KNN
                          test_pred <-
                            knn(
                              train = train,
                              test = test,
                              cl = train_labels,
                              k = as.numeric(i),
                              prob = prob
                            )
                          #matriz de confusi?n
                          mat <-
                            CrossTable(x = test_labels,
                                       y = test_pred,
                                       prop.chisq = FALSE)

                          #almacenamiento de los falsos negativos, positivos
                          #y calculo de porcentaje de elementos mal clasificados

                          res <- table(test_pred, test_labels)
                          (knn_cmatrix <- caret::confusionMatrix(res))
                          if (input$type_label == 'bin') {
                            (knn_cmatrix <- caret::confusionMatrix(res))
                            b <- as.data.frame(knn_cmatrix$table)
                            models_knn[[paste0("knn_", i)]] <-
                              spread(b, test_labels, Freq)
                            info <- as.data.frame(round(knn_cmatrix$overall, 3))
                            info$names <- rownames(info)
                            models_knn[[paste0("knn_", i, "_vals")]] <- info
                            if (prob) {
                              models_knn[[paste0("knn_", i, "_prob")]] <- attr(test_pred, "prob")
                            }

                            models_knn[[paste0("knn_", i, "_lab")]] <- test_labels
                          } else{
                            (knn_cmatrix <- caret::confusionMatrix(res))
                            b <- as.data.frame(mat$t)

                            models_knn[[paste0("knn_", i)]] <- spread(b, y, Freq)
                            info <- as.data.frame(round(knn_cmatrix$overall, 3))
                            info$names <- rownames(info)
                            models_knn[[paste0("knn_", i, "_vals")]] <- info
                            models_knn[[paste0("knn_", i, "_lab")]] <- test_labels
                          }
                        }
                      }
                    })


      } else if (m == 2)
      {
        #nb
        choices <- input$nb_opt
        data_num[is.na(data_num)] <- 0
        naive_data <- data_num
        withProgress(message = 'Generando Modelos Naive Bayes',
                     detail = "part 0",
                     value = 0,
                     {
                       val_cual = c("_bajo", "_medio", "_alto", "_muy_alto")
                       for (name in c(1:ncol(naive_data))) {
                         min = min(naive_data[, name])
                         max = max(naive_data[, name])
                         if ((max - min) >= 4) {
                           diff = (max - min) / 4
                           breaks =  c(min - 0.001,
                                       min + diff,
                                       min + (2 * diff),
                                       min + (3 * diff),
                                       min + (4 * diff))
                           labels = paste0(rep(names(naive_data)[[name]], 4), val_cual)
                           naive_data[, name] <- cut(naive_data[, name],
                                                     breaks = breaks,
                                                     labels = labels)
                         } else if ((max - min) < 4) {
                           naive_data[, name] <- as.factor(naive_data[, name])
```

```
                        }
                      }
                      cat_data <- as.data.frame(lapply(data_cat, factor))
                      naive_data[, colnames(cat_data)] <- cat_data
                      naive_data$class <- as.factor(y)

                      if (length(choices) == 0) {
                        showNotification("Seleccione un el modelo lineal o laplace")
                      } else{
                        train <- naive_data[train_ind,]
                        test <- naive_data[-train_ind, 1:ncol(naive_data) - 1]
                        train_labels <- naive_data[train_ind, ncol(naive_data)]
                        test_labels <- naive_data[-train_ind, ncol(naive_data)]
                        set.seed(1234)
                        naive_model <- naiveBayes(class ~ ., data = train)
                        if (input$prob_nb) {
                          naive_test_pred <- predict(naive_model, test, type = "raw")
                          models_nb$nb_1_prob <- naive_test_pred[, 1]
                          models_nb$nb_1_lab <- test_labels

                        }


                        naive_test_pred <- predict(naive_model, test)
                        res <- table(naive_test_pred, test_labels)
                        (naive_cmatrix_1 <- caret::confusionMatrix(res))
                        accuracy <- naive_cmatrix_1$overall["Accuracy"]
                        models_nb$nb_1 <- as.data.frame(naive_cmatrix_1$table)
                        info <- as.data.frame(round(naive_cmatrix_1$overall, 3))
                        info$names <- rownames(info)
                        models_nb$nb_1_vals <- as.data.frame(info)
                        if (is.element(2, choices)) {
                          naive_model_2 <- naiveBayes(class ~ ., data = train, laplace = 1)
                          if (input$prob_nb) {
                            naive_test_pred_2 <- predict(naive_model_2, test, "raw")
                            models_nb$nb_2_prob <- naive_test_pred_2[, 1]
                            models_nb$nb_2_lab <- test_labels
                          }
                          naive_test_pred_2 <- predict(naive_model_2, test)
                          res <- table(naive_test_pred_2, test_labels)
                          (naive_cmatrix_2 <- caret::confusionMatrix(res))
                          accuracy_laplace <- naive_cmatrix_2$overall["Accuracy"]
                          models_nb$nb_2 <- as.data.frame(naive_cmatrix_2$table)
                          info <- as.data.frame(round(naive_cmatrix_2$overall, 3))
                          info$names <- rownames(info)
                          models_nb$nb_2_vals <- as.data.frame(info)
                        }
                      }
                    })


      } else if (m == 3)
      {
        #ann
        ann_data <- data
        choices <- input$knn_opt
        for (c in choices) {
          if (c == 1) {
            data_num = as.data.frame(scale(data_num))
          } else if (c == 2) {
            data_num = as.data.frame(lapply(data_num, normalize))
          } else if (c == 3) {
            if (length(ncol(data_cat)) > 1) {
              cat_data <- as.data.frame(lapply(data_cat, factor))
              dmy <- dummyVars(" ~ .", data = cat_data)
              data_cat <-
                data.frame(predict(dmy, newdata = cat_data))
            }
          } else if (c == 4) {
            data_cat[is.na(data_cat)] <- 0
            data_num[is.na(data_num)] <- 0
          } else{

          }
        }
        withProgress(message = 'Generando Redes Neuronales',
                     detail = "part 0",
                     value = 0,
                     {
                       data_num[is.na(data_num)] <- 0
                       ann_data <- data_num
                       ann_data[, colnames(data_cat)] <- data_cat
                       ann_data[, colnames(df_log())] <- df_log()
                       ann_data[is.na(ann_data)] <- 0
                       if (length(input$k_layers) == 0) {
                         showNotification("Seleccione un numero de capas para la RN")
                       } else{
                         if (input$type_label == 'bin') {
                           ann_data$Y <- y
```

```
                        ann_data$Y <- ann_data$Y == 1
                        ann_data$N <- !y
                        xnam <- names(ann_data[1:(ncol(ann_data) - 2)])
                        (fmla <-
                            as.formula(paste(
                              "Y+N ~ ", paste(xnam, collapse = "+")
                            )))
                        train <- ann_data[train_ind,]
                        test <- ann_data[-train_ind, ]


                        set.seed(1234)
                        k_layers <- input$k_layers
                        for (k in k_layers) {
                          train_prob <- train[, 1:(ncol(ann_data) - 2)]
                          train_prob$Y <- as.factor(y[train_ind])
                          train_prob[is.na(train_prob)] <- 0
                          nnetGrid <-  expand.grid(size = k,
                                                   decay = seq(
                                                     from = 0.1,
                                                     to = 0.5,
                                                     by = 0.2
                                                   ))

                          model <- train(
                            Y ~ .,
                            data = train_prob,
                            method = "nnet",
                            trControl = trainControl(method = 'cv'),
                            tuneGrid = nnetGrid,
                            verbose = FALSE
                          )

                          test_prob <- test[, 1:(ncol(ann_data) - 2)]
                          test_prob$Y <- as.factor(y[-train_ind])
                          test_prob[is.na(test_prob)] <- 0
                          if (input$prob_ann) {
                            probs <- predict(model, test_prob, type = 'prob')
                            models_ann[[paste0("ann_", k, "_prob")]] <- probs[, 1]
                            models_ann[[paste0("ann_", k, "_lab")]] <- test$Y
                          }
                          pred <- predict(model, test_prob, type = 'raw')
                          res <- table(pred, test_prob$Y)

                          # Results
                          (ann_cmatrix <- caret::confusionMatrix(res))
                          b <- as.data.frame(ann_cmatrix$table)
                          models_ann[[paste0("ann_", k)]] <-
                            spread(b, names(b)[2], Freq)
                          info <- as.data.frame(round(ann_cmatrix$overall, 3))
                          info$names <- rownames(info)
                          models_ann[[paste0("ann_", k, "_vals")]] <- info
                          models_ann[[paste0("ann_", k, "_plot")]] <- model
                        }
                      } else{
                        ann_data$Y <- y
                        values <- unique(ann_data$Y)

                        for (v in values) {
                          ann_data[[paste0(v)]] <- ann_data$Y == v
                        }
                        train <- ann_data[train_ind,]
                        train$Y <- NULL
                        test <- ann_data[-train_ind, ]
                        xnam <- names(ann_data[1:(ncol(train) - length(values))])
                        (fmla <-
                            as.formula(paste(
                              paste(values, collapse = "+"),
                              " ~ ",
                              paste(xnam, collapse = "+")
                            )))

                        set.seed(1234)
                        k_layers <- input$k_layers
                        for (k in k_layers) {
                          data_model_1 <- neuralnet(
                            fmla,
                            data = train,
                            hidden = c(as.numeric(k) + 3, as.numeric(k) + 3),
                            rep = 2,
                            err.fct = "ce",
                            linear.output = F,
                            lifesign = "minimal",
                            stepmax = 10000,
                            threshold = 0.001
                          )
                          model_results_1 <-
                            neuralnet::compute(data_model_1, test[, 1:(ncol(test) - length(values) -
                                                                         1)])  #$net.result
```

```
                            idx <- apply(model_results_1$net.result, 1, which.max)
                            predicted <- values[idx]
                            res <- table(predicted, test$Y)
                            # Results
                            (ann_cmatrix <- caret::confusionMatrix(res))
                            b <- as.data.frame(ann_cmatrix$table)
                            models_ann[[paste0("ann_", k)]] <- spread(b, Var2, Freq)
                            info <- as.data.frame(round(ann_cmatrix$overall, 3))
                            info$names <- rownames(info)
                            models_ann[[paste0("ann_", k, "_vals")]] <- info
                            models_ann[[paste0("ann_", k, "_plot")]] <- data_model_1
                            NeuralNetTools::plotnet(models_ann[[paste0("ann_", k, "_plot")]])
                          }
                        }

                      }
                    })
      } else if (m == 4) {
        choices <- input$svm_opt
        for (c in choices) {
          if (c == 1) {
            data_num = as.data.frame(scale(data_num))
          } else if (c == 2) {
            data_num = as.data.frame(lapply(data_num, normalize))
          } else if (c == 3) {
            if (length(ncol(data_cat)) > 1) {
              cat_data <- as.data.frame(lapply(data_cat, factor))
              dmy <- dummyVars(" ~ .", data = cat_data)
              data_cat <-
                data.frame(predict(dmy, newdata = cat_data))
            }
          } else if (c == 4) {
            data_cat[is.na(data_cat)] <- 0
            data_num[is.na(data_num)] <- 0
          } else{

          }
        }
        #SVM
        withProgress(message = 'Generando Maquinas de Soporte Vectorial',
                     detail = "part 0",
                     value = 0,
                     {
                       if (input$type_label == 'bin') {
                         data_num[is.na(data_num)] <- 0
                         svm_data <- data_num
                         svm_data$Y <- y
                         svm_data$y <- c('no', 'yes')[svm_data$Y + 1]
                         svm_data$y <- as.factor(svm_data$y)
                         svm_data <- subset(svm_data, select = -c(Y))
                         train <- svm_data[train_ind,]
                         test <- svm_data[-train_ind, ]
                         kernel <- input$kernel
                         for (k in kernel) {
                           if (k == 1) {
                             model_linearK <- svm(y ~ ., data = train, kernel = "linear")
                             linear_pred <- predict(model_linearK, test)
                             table(linear_pred, test$y)
                             matches_linear <- linear_pred == test$y
                             #Modelo lineal
                             (
                               svm_cmatrix_lineal <-
                                 caret::confusionMatrix(table(linear_pred, test$y),
                                                        positive = 'yes')
                             )
                             b <- as.data.frame(svm_cmatrix_lineal$table)

                             models_svm$svm_1 <- spread(b, names(b)[2], Freq)
                             info <-
                               as.data.frame(round(svm_cmatrix_lineal$overall, 3))
                             info$names <- rownames(info)
                             models_svm$svm_1_vals <- info
                             models_svm$svm_1_plot <- train

                           } else if (k == 2) {

                           } else if (k == 3) {

                           } else{
                             model_rbfK <- svm(y ~ ., data = train, kernel = "sigmoid")
                             rbf_pred <- predict(model_rbfK, test)
                             matches_rbf <- rbf_pred == test$y
                             prop.table(table(matches_rbf))
                             (
                               svm_cmatrix_gauss <- caret::confusionMatrix(table(rbf_pred, test$y),
                                                                           positive = 'yes')
                             )
                             print(round(svm_cmatrix_gauss$overall["Accuracy"], 3))
                             b <- as.data.frame(svm_cmatrix_gauss$table)
```

```
                            models_svm$svm_2 <- spread(b, names(b)[2], Freq)
                            info <-
                              as.data.frame(round(svm_cmatrix_gauss$overall, 3))
                            info$names <- rownames(info)
                            models_svm$svm_2_vals <- info
                            models_svm$svm_2_plot <- train
                          }
                        }
                      } else{
                        data_num[is.na(data_num)] <- 0
                        svm_data <- data_num
                        svm_data[is.na(svm_data)] <- 0
                        svm_data$y <- as.factor(y)
                        train <- svm_data[train_ind,]
                        test <- svm_data[-train_ind, ]
                        train <-
                          train[, apply(train, 2, function(col) {
                            length(unique(col)) > 1
                          })]
                        kernel <- input$kernel_mult
                        for (k in kernel) {
                          if (as.numeric(k) == 1) {
                            svm_model <-
                              e1071::svm(
                                y ~ .,
                                type = "C-classification",
                                data = train,
                                kernel = 'linear',
                                gamma = 0.1
                              )
                          } else if (as.numeric(k) == 2) {
                            svm_model <-
                              e1071::svm(
                                y ~ .,
                                type = "C-classification",
                                data = train,
                                kernel = 'polynomial',
                                degree = 3,
                                coef0 = 0.1
                              )
                          } else if (as.numeric(k) == 3) {
                            svm_model <-
                              e1071::svm(
                                y ~ .,
                                type = "C-classification",
                                data = train,
                                kernel = 'radial'
                              )
                          } else if (as.numeric(k) == 4) {
                            svm_model <-
                              e1071::svm(
                                y ~ .,
                                type = "C-classification",
                                data = train,
                                kernel = 'sigmoid',
                                coef0 = 0.1
                              )
                          } else{

                          }
                          x <- subset(test, select = -y)
                          pred <- predict(svm_model, x)
                          svm_cmatrix <-
                            caret::confusionMatrix(table(pred, test$y))
                          b <- as.data.frame(svm_cmatrix$table)
                          models_svm[[paste0("svm_", k)]] <- spread(b, Var2, Freq)
                          info <- as.data.frame(round(svm_cmatrix$overall, 3))
                          info$names <- rownames(info)
                          models_svm[[paste0("svm_", k, "_vals")]] <- info
                          models_svm[[paste0("svm_", k, "_plot")]] <- train
                        }

                      }
                    })

    } else if (m == 5) {
      choices <- input$dt_opt
      for (c in choices) {
        if (c == 1) {
          data_num = as.data.frame(scale(data_num))
        } else if (c == 2) {
          data_num = as.data.frame(lapply(data_num, normalize))
        } else if (c == 3) {
          if (length(ncol(data_cat)) > 1) {
            cat_data <- as.data.frame(lapply(data_cat, factor))
            dmy <- dummyVars(" ~ .", data = cat_data)
            data_cat <-
              data.frame(predict(dmy, newdata = cat_data))
```

```
            }
          } else if (c == 4) {
            data_cat[is.na(data_cat)] <- 0
            data_num[is.na(data_num)] <- 0
          } else{

          }
        }
        #dt
        dt_data <- data_num
        dt_data[, colnames(data_cat)] <- data_cat
        xnam <- names(dt_data)
        dt_data$Class <- y
        dt_data$Class - as.factor(dt_data$Class)
        train <- dt_data[train_ind,]
        test <- dt_data[-train_ind, ]
        set.seed(1234)
        fmla <-
          as.formula(paste("Class ~ ", paste(xnam, collapse = "+")))
        data.model <- rpart(fmla, data = train, method = "class")
        if (input$prob_dt) {
          probs <- predict(data.model, test, type = 'prob')
          models_dt$dt_1_prob <- as.data.frame(probs)[, 1]
          models_dt$dt_1_lab <- test$Class
        }
        resumen <- summary(data.model)
        predict <- predict(data.model, test, type = "class")
        cf <- caret::confusionMatrix(predict, as.factor(test$Class))
        b <- as.data.frame(cf$table)
        models_dt$dt_1 <- spread(b, Reference, Freq)
        info <- as.data.frame(round(cf$overall, 3))
        info$names <- rownames(info)
        models_dt$dt_1_vals <- info
        models_dt$dt_1_plot <- data.model
      } else if (m == 6)
      {
        #rf
        choices <- input$rf_opt
        data_cat[is.na(data_cat)] <- 0
        data_num[is.na(data_num)] <- 0
        for (c in choices) {
          if (c == 1) {
            data_num = as.data.frame(scale(data_num))
          } else if (c == 2) {
            data_num = as.data.frame(lapply(data_num, normalize))
          } else if (c == 3) {
            if (length(ncol(data_cat)) > 1) {
              cat_data <- as.data.frame(lapply(data_cat, factor))
              dmy <- dummyVars(" ~ .", data = cat_data)
              data_cat <-
                data.frame(predict(dmy, newdata = cat_data))
            }
          } else if (c == 4) {
            data_cat[is.na(data_cat)] <- 0
            data_num[is.na(data_num)] <- 0
          } else{

          }
        }
        rf_trees <- input$rf_trees
        rf_data <- data_num
        rf_data[, colnames(data_cat)] <- data_cat
        rf_data[, colnames(df_log())] <- df_log()
        rf_data$Class <- as.factor(y)
        rf_data[is.na(rf_data)] <- 0
        train <- rf_data[train_ind,]
        test <- rf_data[-train_ind, ]
        set.seed(1234)
        rf <-
          randomForest(Class ~ .,
                       data  = train,
                       ntree = input$rf_trees)
        importance <- importance(rf)
        varImportance <- varImpPlot(rf)
        predict.rf <- predict(rf, test)
        if (input$prob_rf) {
          probs <- predict(rf, test, type = 'prob')
          models_rf$rf_1_prob <- probs[, 1]
          models_rf$rf_1_lab <- test$Class
        }
        cf <- caret::confusionMatrix(test$Class, predict.rf)
        accuracy <- round(cf$overall["Accuracy"], 3)
        b <- as.data.frame(cf$table)
        models_rf$rf_1 <- spread(b, Reference, Freq)
        info <- as.data.frame(round(cf$overall, 3))
        info$names <- rownames(info)
        models_rf$rf_1_vals <- info
        models_rf$rf_1_plot <- rf
      }
```

```
1317        }
1318      }
1319    })
1320
1321    ######################################################
1322    # GUI interactiva de los resultados de los modelos  #
1323    ######################################################
1324    output$result_models_bin <- renderUI({
1325      tabBox(
1326        title = "Results",
1327        width = NULL,
1328        # The id lets us use input$tabset1 on the server to find the current tab
1329        id = "model_results",
1330        tabPanel(
1331          value = 'knn',
1332          title = "Resultado KNN",
1333          fluidRow(
1334            column(4, dataTableOutput("res_knn_3")),
1335            column(4, dataTableOutput("res_knn_3_vals")),
1336            column(4, plotOutput("res_knn_3_plot"))
1337          ),
1338          fluidRow(
1339            column(4, dataTableOutput("res_knn_5")),
1340            column(4, dataTableOutput("res_knn_5_vals")),
1341            column(4, plotOutput("res_knn_5_plot"))
1342          ),
1343          fluidRow(
1344            column(4, dataTableOutput("res_knn_7")),
1345            column(4, dataTableOutput("res_knn_7_vals")),
1346            column(4, plotOutput("res_knn_7_plot"))
1347          ),
1348          fluidRow(
1349            column(4, dataTableOutput("res_knn_9")),
1350            column(4, dataTableOutput("res_knn_9_vals")),
1351            column(4, plotOutput("res_knn_9_plot"))
1352          )
1353        ),
1354
1355        tabPanel(
1356          value = 'nb',
1357          title = "Resultado Naive Bayes",
1358          fluidRow(
1359            column(4, dataTableOutput("res_nb_1")),
1360            column(4, dataTableOutput("res_nb_1_vals")),
1361            column(4, plotOutput("res_nb_1_plot"))
1362          ),
1363          fluidRow(
1364            column(4, dataTableOutput("res_nb_2")),
1365            column(4, dataTableOutput("res_nb_2_vals")),
1366            column(4, plotOutput("res_nb_2_plot"))
1367          )
1368        ),
1369        tabPanel(
1370          value = 'ann',
1371          title = "Resultado ANN",
1372          fluidRow(
1373            column(4, dataTableOutput("res_ann_1")),
1374            column(4, dataTableOutput("res_ann_1_vals")),
1375            column(4, plotOutput("res_ann_1_plot"))
1376          ),
1377          fluidRow(
1378            column(4, dataTableOutput("res_ann_3")),
1379            column(4, dataTableOutput("res_ann_3_vals")),
1380            column(4, plotOutput("res_ann_3_plot"))
1381          ),
1382          fluidRow(
1383            column(4, dataTableOutput("res_ann_5")),
1384            column(4, dataTableOutput("res_ann_5_vals")),
1385            column(4, plotOutput("res_ann_5_plot"))
1386          )
1387        ),
1388        tabPanel(
1389          value = 'svm',
1390          title = "Resultado SVM",
1391          fluidRow(column(6, dataTableOutput("res_svm_1")), column(
1392            6, dataTableOutput("res_svm_1_vals")
1393          )),
1394          fluidRow(column(6, dataTableOutput("res_svm_2")), column(
1395            6, dataTableOutput("res_svm_2_vals")
1396          )),
1397          fluidRow(column(6, dataTableOutput("res_svm_3")), column(
1398            6, dataTableOutput("res_svm_3_vals")
1399          )),
1400          fluidRow(column(6, dataTableOutput("res_svm_4")),
1401                   column(
1402                     6, dataTableOutput("res_svm_4_vals")
1403                   ))
1404        ),
1405        tabPanel(
```

```
        value = 'dt',
        title = "Resultado DT",
        fluidRow(
          column(4, dataTableOutput("res_dt_1")),
          column(4, dataTableOutput("res_dt_1_vals")),
          column(4, plotOutput("res_dt_1_plot"))
        )
      ),

      tabPanel(
        value = 'rf',
        title = "Resultado RF",
        fluidRow(
          column(4, dataTableOutput("res_rf_1")),
          column(4, dataTableOutput("res_rf_1_vals")),
          column(4, plotOutput("res_rf_1_plot"))
        )
      )
    )
  })

  ########################################
  # GUI plot de resultados del modelo KNN #
  ########################################
  #tables matriz de confusion
  for (i in seq(3, 10, 2)) {
    local({
      my_i <- i
      tablename <- paste0("res_knn_", my_i)
      resname <- paste0("res_knn_", my_i, "_vals")
      plotname <- paste0("res_knn_", my_i, "_plot")

      # Matriz de confusion
      output[[tablename]] <- renderDataTable({
        models_knn[[paste0("knn_", my_i)]]

      })

      # Datos de las metricas
      output[[resname]] <- renderDataTable({
        if (models_knn[[paste0("knn_", my_i, "_vals")]] == FALSE) {
          return(NULL)
        }
        names(models_knn[[paste0("knn_", my_i, "_vals")]]) <-
          c("Valor", "Metrica")
        df <- models_knn[[paste0("knn_", my_i, "_vals")]][, c(2, 1)]
        df


      })

      # Plot especifica del modelo
      output[[plotname]] <- renderPlot({
        prob_input <- input$prob_knn
        prob <- models_knn[[paste0("knn_", my_i, "_prob")]]
        test_labels <- models_knn[[paste0("knn_", my_i, "_lab")]]
        if (!prob_input | !prob) {
          return(NULL)
        }
        plot(
          pROC::roc(test_labels, prob, direction = ">"),
          col = "pink",
          lwd = 3,
          main = paste0("ROC ", my_i, "knn ")
        )
      })

    })
  }

  ########################################
  # GUI plot de resultados del modelo NB  #
  ########################################
  for (i in seq(1, 2, 1)) {
    local({
      my_i <- i
      tablename <- paste0("res_nb_", my_i)
      resname <- paste0("res_nb_", my_i, "_vals")
      plotname <- paste0("res_nb_", my_i, "_plot")

      # Matriz de confusion
      output[[tablename]] <- renderDataTable({
        print(tablename)
        models_nb[[paste0("nb_", my_i)]]

      })
      # Datos de las metricas
      output[[resname]] <- renderDataTable({
        if (models_nb[[paste0("nb_", my_i, "_vals")]] == FALSE) {
```

```
1495          return(NULL)
1496        }
1497        names(models_nb[[paste0("nb_", my_i, "_vals")]]) <-
1498          c("Valor", "Metrica")
1499        df <- models_nb[[paste0("nb_", my_i, "_vals")]][, c(2, 1)]
1500        df
1501
1502      })
1503
1504      # Plot especifica del modelo
1505      output[[plotname]] <- renderPlot({
1506        prob_input <- input$prob_nb
1507        prob <- models_nb[[paste0("nb_", my_i, "_prob")]]
1508        test_labels <- models_nb[[paste0("nb_", my_i, "_lab")]]
1509
1510        if (!prob_input | !prob) {
1511          return(NULL)
1512        }
1513        plot(
1514          pROC::roc(test_labels, prob, direction = ">"),
1515          col = "pink",
1516          lwd = 3,
1517          main = "ROC curve"
1518        )
1519      })
1520
1521    })
1522  }
1523
1524  ########################################
1525  # GUI plot de resultados del modelo ANN #
1526  ########################################
1527  #tables matriz de confusion
1528  for (i in seq(1, 10, 2)) {
1529    local({
1530      my_i <- i
1531      tablename <- paste0("res_ann_", my_i)
1532      resname <- paste0("res_ann_", my_i, "_vals")
1533      plotname <- paste0("res_ann_", my_i, "_plot")
1534
1535      # Matriz de confusion
1536      output[[tablename]] <- renderDataTable({
1537        models_ann[[paste0("ann_", my_i)]]
1538
1539      })
1540
1541      # Datos de las metricas
1542      output[[resname]] <- renderDataTable({
1543        if (models_ann[[paste0("ann_", my_i, "_vals")]] == FALSE) {
1544          return(NULL)
1545        }
1546        names(models_ann[[paste0("ann_", my_i, "_vals")]]) <-
1547          c("Valor", "Metrica")
1548        df <- models_ann[[paste0("ann_", my_i, "_vals")]][, c(2, 1)]
1549        df
1550
1551      })
1552
1553      # Plot especifica del modelo
1554      output[[plotname]] <- renderPlot({
1555        prob <- models_ann[[paste0("ann_", my_i, "_prob")]]
1556        test_labels <- models_ann[[paste0("ann_", my_i, "_lab")]]
1557        if (input$prob_ann &
1558            !(identical(models_ann[[paste0("ann_", my_i, "_prob")]], FALSE))) {
1559          plot(
1560            pROC::roc(test_labels, prob, direction = ">"),
1561            col = "pink",
1562            lwd = 3,
1563            main = "ROC curve"
1564          )
1565        }
1566        else if (!(identical(models_ann[[paste0("ann_", my_i, "_plot")]], FALSE))) {
1567          NeuralNetTools::plotnet(models_ann[[paste0("ann_", my_i, "_plot")]])
1568        }
1569        else{
1570          return(NULL)
1571        }
1572
1573      })
1574
1575    })
1576  }
1577
1578  ########################################
1579  # GUI plot de resultados del modelo SVM #
1580  ########################################
1581  for (i in seq(1, 4)) {
1582    local({
1583      my_i <- i
```

```
      tablename <- paste0("res_svm_", my_i)
      resname <- paste0("res_svm_", my_i, "_vals")
      plotname <- paste0("res_svm_", my_i, "_plot")

      # Matriz de confusion
      output[[tablename]] <- renderDataTable({
        print(tablename)
        models_svm[[paste0("svm_", my_i)]]

      })

      # Datos de las metricas
      output[[resname]] <- renderDataTable({
        if (models_svm[[paste0("svm_", my_i, "_vals")]] == FALSE) {
          return(NULL)
        }
        names(models_svm[[paste0("svm_", my_i, "_vals")]]) <-
          c("Valor", "Metrica")
        df <- models_svm[[paste0("svm_", my_i, "_vals")]][, c(2, 1)]
        df

      })

      # Plot especifica del modelo
      output[[plotname]] <- renderPlot({
        model_linear <-
          svm(y ~ ., data = models_svm[[paste0("svm_", my_i, "_plot")]], kernel = "linear")
        formula_plot <-
          as.formula(paste0(input$eje_x_svm, "~", input$eje_y_svm))
        plot(model_linear, models_svm[[paste0("svm_", my_i, "_plot")]], formula_plot)

      })

    })
  }


  ########################################
  # GUI plot de resultados del modelo DT  #
  ########################################
  for (i in seq(1, 1, 1)) {
    local({
      my_i <- i
      tablename <- paste0("res_dt_", my_i)
      resname <- paste0("res_dt_", my_i, "_vals")
      plotname <- paste0("res_dt_", my_i, "_plot")

      # Matriz de confusion
      output[[tablename]] <- renderDataTable({
        print(tablename)
        models_dt[[paste0("dt_", my_i)]]

      })

      # Datos de las metricas
      output[[resname]] <- renderDataTable({
        if (models_dt[[paste0("dt_", my_i, "_vals")]] == FALSE) {
          return(NULL)
        }
        names(models_dt[[paste0("dt_", my_i, "_vals")]]) <-
          c("Valor", "Metrica")
        df <- models_dt[[paste0("dt_", my_i, "_vals")]][, c(2, 1)]
        as.data.frame(df)

      })

      # Plot especifica del modelo
      output[[plotname]] <- renderPlot({
        prob_input <- input$prob_dt
        prob <- models_dt[[paste0("dt_", my_i, "_prob")]]
        pl <- models_dt[[paste0("dt_", my_i, "_plot")]]
        if (identical(prob, FALSE) & identical(pl, FALSE)) {
          return(NULL)
        }
        test_labels <- models_dt[[paste0("dt_", my_i, "_lab")]]
        if (input$prob_dt & !(identical(prob, FALSE))) {
          plot(
            pROC::roc(test_labels, prob, direction = ">"),
            col = "pink",
            lwd = 3,
            main = "ROC curve"
          )
        } else if (!identical(pl, FALSE)  & !input$prob_dt) {
          plot(models_dt[[paste0("dt_", my_i, "_plot")]])
          text(models_dt[[paste0("dt_", my_i, "_plot")]])
        } else{
          return()
        }
      })
```

```
1673
1674      })
1675    }
1676
1677    ########################################
1678    #  GUI plot de resultados del modelo RF #
1679    ########################################
1680    for (i in seq(1, 1, 1)) {
1681      local({
1682        my_i <- i
1683        tablename <- paste0("res_rf_", my_i)
1684        resname <- paste0("res_rf_", my_i, "_vals")
1685        plotname <- paste0("res_rf_", my_i, "_plot")
1686
1687        # Matriz de confusion
1688        output[[tablename]] <- renderDataTable({
1689          print(tablename)
1690          models_rf[[paste0("rf_", my_i)]]
1691
1692        })
1693
1694        # Datos de las metricas
1695        output[[resname]] <- renderDataTable({
1696          if (models_rf[[paste0("rf_", my_i, "_vals")]] == FALSE) {
1697            return(NULL)
1698          }
1699          names(models_rf[[paste0("rf_", my_i, "_vals")]]) <-
1700            c("Valor", "Metrica")
1701          df <- models_rf[[paste0("rf_", my_i, "_vals")]][, c(2, 1)]
1702          df
1703
1704
1705        })
1706
1707        # Plot especifica del modelo
1708        output[[plotname]] <- renderPlot({
1709          please_work <- models_rf[[paste0("rf_", my_i, "_prob")]]
1710          pl <- prob <- models_rf[[paste0("rf_", my_i, "_plot")]]
1711          test_labels <- models_rf[[paste0("rf_", my_i, "_lab")]]
1712          if (identical(prob, FALSE)  & identical(pl, FALSE)) {
1713            return(NULL)
1714          }
1715          if (input$prob_rf & !(identical(please_work, FALSE))) {
1716            plot(
1717              pROC::roc(test_labels, please_work, direction = ">"),
1718              col = "pink",
1719              lwd = 3,
1720              main = "ROC curve"
1721            )
1722          } else if (!identical(pl, FALSE)  & !input$prob_rf) {
1723            varImpPlot(pl, type = 2)
1724          } else{
1725            return()
1726          }
1727
1728        })
1729
1730      })
1731    }
1732
1733    output$downloadData <- downloadHandler(
1734      file = function() {
1735        paste("results_", input$dataset, ".csv", sep = ",")
1736      },
1737      content = function(file) {
1738        write.csv(datasetInput(), file, row.names = FALSE)
1739      }
1740    )
1741
1742  }
1743
1744
1745
1746  shinyApp(ui, server)
```

B.2. UI estática

Inicio

```
inicio <- fluidRow(actionButton(inputId = "start", "START"))
fin <- fluidRow(textOutput("start_text"))


#Paginas de explicacion
explanation <-
  tabItem(
    tabName = "inicio",
    tags$h1("Clasificacion de datos"),
    fluidRow(column(
      12,
      box(
        title = "HOW TO... Definir y cargar datos",
        width = NULL,
        solidHeader = TRUE,
        collapsed = TRUE,
        status = "primary",
        background = "teal",
        collapsible = TRUE,

        p(
          strong("1-"),
          "Seleccione de cuantos fihceros tiene que cargar los datos "
        ),
        p(
          strong("\t1.1-"),
          "Fichero si el conjunto incluye la variable. En este caso debera indicar cual es el nombre de la columna
               target en el selector que aparecera a la derecha'"
        ),
        p(
          strong("\t1.2-"),
          "Ficheros si el conjunto de variables y target estan en ficheros separados.
          En este caso el segundo conjunto debera ser un csv de una sola columna."
        ),
        p(strong("2-"), "Seleccione si la target es binaria o multiclase. "),
        p(
          strong("3-"),
          "Seleccione el o los ficheros con los datos indicando el separador,si tienen cabecera o si hay algun
               caracter de nulo especial. "
        ),
        p(strong("4-"), "Se le mostrara una muestra de los datos cargados "),
        p(strong("5-"), "Para cargar los datos pulse el boton 'Cargar Datos' ")
        )
    )),
    fluidRow(column(
      12,
      box(
        title = "HOW TO... Visualizacion de la descripcion de los datos",
        width = NULL,
        solidHeader = TRUE,
        collapsed = TRUE,
        status = "primary",
        background = "teal",
        collapsible = TRUE,
        p(
          strong("1-"),
          "En primer lugar se mostrara la descripcion de cada columna del conjunto de datos "
        ),
        p(
          strong("2-"),
          "Para cada tipo de datos, numericos, categoricos o logicos se muestra un sample"
        ),
        p(
          strong("3-"),
          "Para cada variable del tipo, se muestra o bien un histograma para las numericas o la distribucion por
               clases en el resto."
        ),
        p(
          strong("4-"),
          "Por ultmo se muestra un boxplot de las variables para cada conjunto de datos asociado a cada target de
               calsificacion."
        )
      )
    )),
    fluidRow(column(
      12,
      box(
        title = "HOW TO... Separar en conjunto de entrenamiento y validacion",
        width = NULL,
        solidHeader = TRUE,
        collapsed = TRUE,
        status = "primary",
        background = "teal",
```

```
80          collapsible = TRUE,
81
82          p(
83            strong("1-"),
84            "Se propociona un slider para indicar el porcentaje de los datos que se asignaran al conjunto de
                  entrenamiento."
85          )
86        )
87      )),
88      fluidRow(column(
89        12,
90        box(
91          title = "HOW TO... Modelos  de clasificacion",
92          width = NULL,
93          solidHeader = TRUE,
94          collapsed = TRUE,
95          status = "primary",
96          background = "teal",
97          collapsible = TRUE,
98
99          p(
100           strong("1-"),
101           "Se selecciona si se quieren hacer modelos de clasifiacion binaria  o multiclase."
102         ),
103         p(
104           strong("2-"),
105           "De los modelos disponibles se indican los que se quieren probar."
106         ),
107         p(
108           strong("3-"),
109           "Para cada uno de los modelos elegidos se mostraran pesta??as asociadas a los parametros implementados."
110         ),
111         p(
112           strong("4-"),
113           "Para cada modelo se deben elegir los parametros y prepocesados de datos necesarios"
114         ),
115         p(
116           strong("5-"),
117           "Una vez completados estos datos, se ha de pulsar el boton 'Ejecutar modelos'"
118         ),
119         p(
120           strong("6-"),
121           "En la parte inferior, para cada modelo elegido se mostraran las pesta??as con, la matriz de confusion, los
                  scores del modelo y en algunos casos una visualizacion de los resultados o el modelo."
122         )
123       )
124     ))
125     )
126
127 start_data <- explanation
```

Carga y lectura de datos

```
# Definicion y carga de datos
number_files_input <-
  column(5, box(
    radioButtons(
      "num_files",
      "Data files:",
      choices =  c("One file" = "one",
                   "Two files" = "two"),
      selected = 'one'
    ),
    width = NULL
  ))

type_pred <- column(5, box(
  radioButtons(
    "type_label",
    "Prediction:",
    choices =  c("Binary" = "bin",
                 "Multiclass" = "mult"),
    selected = 'bin'
  ),
  width = NULL
))

load_button <- column(2, actionButton("go_describe", "Load files"))
definition_data <-
  fluidRow(number_files_input, type_pred, load_button)



files_input <- uiOutput('distPlot')
data_table <- fluidRow(column(12, uiOutput("tables")))


load_data <-
  tabItem(tabName = "load", definition_data, files_input, data_table)
```

Descripción y EDA

```
#Descripcion de los datos
types_features <- fluidRow(
  tabBox(
    title = "Variables por tipo",
    width = NULL,
    id = "type_feature",
    selected = "num",
    tabPanel(
      value = "cat",
      "Variables categoricas",
      width = NULL,
      dataTableOutput("table_cat")
    ),
    tabPanel(
      value = "num",
      "Variables numericas",
      width = NULL,
      dataTableOutput("table_num"),
      uiOutput("plot_hist_num")
    ),
    tabPanel(
      value = "log",
      "Variables logicas",
      width = NULL,
      dataTableOutput("table_log")
    )

  )
)
str_features <- fluidRow(verbatimTextOutput("str_data"))


box_features <-
  fluidRow(
    conditionalPanel(condition = "input.type_label=='bin'",
                     column(6, plotOutput("box_aus")),
                     column(6, plotOutput("box_pre"))),
    conditionalPanel(condition = "input.type_label=='mult'",
                     uiOutput("plot_box_class"))
  )


results_load <-
  tabItem(tabName = "describe", str_features,  types_features, box_features)
```

Selección de modelos, parámetros y resultados

```
1
2
3  models <- fluidRow(column(
4    width = 5,
5    box(
6      title = "Modelos de Clasificacion",
7      width = NULL,
8      id = "models_sel",
9      height = NULL,
10     conditionalPanel(
11       condition = "input.type_label == 'bin'",
12       "Classification Binaria",
13       checkboxGroupInput(
14         inputId = "alg_models_b",
15         label = "Modelos",
16         choices = list(
17           "KNN" = 1,
18           "Naive Bayes" = 2,
19           "ANN" = 3,
20           "SVM" = 4,
21           "Arbol Decision" =
22             5,
23           "Random Forest" = 6
24         )
25       )
26     ),
27     conditionalPanel(
28       condition = "input.type_label == 'mult'",
29       "Classificacion Multiple",
30       checkboxGroupInput(
31         inputId = "alg_models_m",
32         label = "Modelos",
33         choices = list(
34           "KNN" = 1,
35           "ANN" = 3,
36           "SVD" = 4,
37           "Arbol Decision" =
38             5,
39           "Random Forest" = 6
40         )
41       )
42     )
43   )
44 ),
45 column(
46   7,
47   tabBox(
48     title = "Parametros",
49     width = NULL,
50     id = "tabset1",
51     tabPanel(
52       value = "knn",
53       title = "Opciones KNN",
54       checkboxGroupInput(
55         label = "Preprocesamiento",
56         inputId = "knn_opt",
57         choices = list(
58           "Estandarizacion" = 1,
59           "Normalizacion" = 2,
60           "One Hot" = 3,
61           "Imputacion" = 4
62         )
63       ),
64       checkboxGroupInput(
65         label = "Vecinos",
66         inputId = "k_values",
67         choices = list(
68           "3" = 3,
69           "5" = 5,
70           "7" = 7,
71           "9" = 9
72         )
73       ),
74       conditionalPanel(condition = "input.type_label == 'bin'",
75                        checkboxInput("prob_knn", "Probabilidad", TRUE))
76     ),
77     tabPanel(
78       value = "nb",
79       title = "Opciones NaiveBayes",
80       checkboxGroupInput(
81         "Preprocesamiento",
82         inputId = "nb_opt",
83         choices = list("Estratificar" =
84                          1, "Laplace" = 2)
85       ),
86       conditionalPanel(condition = "input.type_label == 'bin'",
```

```
87                      checkboxInput("prob_nb", "Probabilidad", TRUE))
88    ),
89    tabPanel(
90      value = "ann",
91      title = "Opciones Red Neuronal",
92      checkboxGroupInput(
93        "Preprocesamiento",
94        inputId = "ann_opt",
95        choices = list("Normalizacion" =
96                       1, "Estandarizacion" = 2)
97      ),
98      checkboxGroupInput(
99        label = "Hidden Layers",
100       inputId = "k_layers",
101       choices = list("1" =
102                      1, "3" = 3, "5" = 5)
103     ),
104     conditionalPanel(condition = "input.type_label == 'bin'",
105                      checkboxInput("prob_ann", "Probabilidad", TRUE))
106   ),
107   tabPanel(
108     value = "svm",
109     title = "Opciones Suport Vector Machine",
110     checkboxGroupInput(
111       "Preprocesamiento",
112       inputId = "svm_opt",
113       choices = list("Normalizacion" =
114                      1, "Estandarizacion" = 2)
115     ),
116     conditionalPanel(
117       condition = "input.type_label == 'bin'",
118       checkboxGroupInput(
119         label = "Kernel",
120         inputId = "kernel",
121         choices = list("Lineal" =
122                        1, "Gausiano" = 4)
123       )
124     ),
125     conditionalPanel(
126       condition = "input.type_label == 'mult'",
127       checkboxGroupInput(
128         label = "Kernel",
129         inputId = "kernel_mult",
130         choices = list(
131           "Lineal" = 1,
132           "Polinomial" = 2,
133           "Radial" = 3,
134           "Gausiano" = 4
135         )
136       )
137     ),
138     conditionalPanel(condition = "input.type_label == 'bin'",
139                      checkboxInput("prob_svm", "Probabilidad", TRUE))
140   ),
141   tabPanel(
142     value = "dt",
143     title = "Opciones Decission Tree",
144     checkboxGroupInput(
145       label = "Preprocesamiento",
146       inputId = "dt_opt",
147       choices = list(
148         "Estandarizacion" = 1,
149         "Normalizacion" = 2,
150         "One Hot" =
151           3,
152         "Imputacion" = 4
153       )
154     ),
155     conditionalPanel(condition = "input.type_label == 'bin'", checkboxInput("prob_dt", "Probabilidad", TRUE))
156   ),
157   tabPanel(
158     value = "rf",
159     title = "Opciones Random Forest",
160     sliderInput(
161       "rf_trees",
162       "Numero arboles:",
163       min = 1,
164       max = 2000,
165       value = 1000
166     ),
167     checkboxGroupInput(
168       label = "Preprocesamiento",
169       inputId = "rf_opt",
170       choices = list(
171         "Estandarizacion" = 1,
172         "Normalizacion" = 2,
173         "One Hot" =
174           3,
175         "Imputacion" = 4
```

```
        )
      ),
      conditionalPanel(condition = "input.type_label == 'bin'",
                       checkboxInput("prob_rf", "Probabilidad", TRUE))
    ),
    helpText("Seleccion de parametros y procesamiento de modelos")
  )
))
exec_button <- fluidRow(column(10), column(width = 2,
                                           actionButton("model_eje", "Ejecutar Modelos")))
viz_models <- uiOutput("result_models_bin")

models <-
  tabItem(tabName = "models", models, exec_button, viz_models)
```

B.3. Variables globales

```
library(shiny)
library(shinyFiles)
library(reshape)
library(shinydashboard)
library(shinyjs)
library(shinyalert)
library(ggplot2)
library(forcats)
library(tidyr)
library(caret)
library(nnet)
library(psych)
library(rpart)
library(pROC)
library(kernlab)
library(class)
library(randomForest)
library(neuralnet)
library(NeuralNetTools)
library(e1071)
library(gmodels)
library(dplyr)


volumes <-
  c(
    'Home' = "~",
    'Desktop' = "~/Desktop",
    'Dropbox' = "~/Dropbox",
    "Mis Documentos" = "~/Documents"
  )
customMenuItem <-
  function (text,
            ...,
            icon = NULL,
            badgeLabel = NULL,
            badgeColor = "green",
            tabName = NULL,
            href = NULL,
            newtab = TRUE,
            selected = NULL,
            hasTab = F)
  {
    subItems <- list(...)
    if (!is.null(icon))
      shinydashboard:::tagAssert(icon, type = "i")
    if (!is.null(href) + (!is.null(tabName) + (length(subItems) >
                                                 0) != 1)) {
      stop("Must have either href, tabName, or sub-items (contained in ...).")
    }
    if (!is.null(badgeLabel) && length(subItems) != 0) {
      stop("Can't have both badge and subItems")
    }
    shinydashboard:::validateColor(badgeColor)
    isTabItem <- FALSE
    target <- NULL
    if (!is.null(tabName)) {
      shinydashboard:::validateTabName(tabName)
      isTabItem <- TRUE
      href <- paste0("#shiny-tab-", tabName)
    }
    else if (is.null(href)) {
      href <- "#"
    }
    else {
      if (newtab)
        target <- "_blank"
    }
    if (!is.null(badgeLabel)) {
      badgeTag <- tags$small(class = paste0("badge pull-right bg-",
                                            badgeColor), badgeLabel)
    }
    else {
      badgeTag <- NULL
    }
    if (length(subItems) == 0) {
      return(tags$li(
        a(
          href = href,
          `data-toggle` = if (isTabItem)
            "tab",
          `data-value` = if (!is.null(tabName))
            tabName,
          `data-start-selected` = if (isTRUE(selected))
            1
          else
```

```
            NULL,
          target = target,
          icon,
          span(text),
          badgeTag
        )
      ))
    }
    else if (!hasTab)
      return(tags$li(
        class = "treeview",
        a(
          href = href,
          icon,
          span(text),
          shiny::icon("angle-left", class = "pull-right")
        ),
        tags$ul(class = "treeview-menu",
                subItems)
      ))
    else
      return(tags$li(
        class = "treeview",
        a(
          href = href,
          icon,
          span(text),
          'data-toggle' = if (isTabItem)
            "tab",
          'data-value' = if (!is.null(tabName))
            tabName,
          `data-start-selected` = if (isTRUE(selected))
            1
          else
            NULL,
          shiny::icon("angle-left", class = "pull-right")
        ),
        tags$ul(class = "treeview-menu",
                subItems)
      ))
  }

read_file <- function(input, header, separ) {
  read.csv(input, header = header, sep = separ)
}

normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

simple_roc <- function(labels, scores) {
  labels <- labels[order(scores, decreasing = TRUE)]
  data.frame(
    TPR = cumsum(labels) / sum(labels),
    FPR = cumsum(!labels) / sum(!labels),
    labels
  )
}

models_knn <- reactiveValues(
  knn_3 = FALSE,
  knn_3_vals = FALSE,
  knn_3_prob = FALSE,
  knn_3_lab = FALSE,
  knn_5 = FALSE,
  knn_5_vals = FALSE,
  knn_5_prob = FALSE,
  knn_5_lab = FALSE,
  knn_7 = FALSE,
  knn_7_vals = FALSE,
  knn_7_prob = FALSE,
  knn_7_lab = FALSE,
  knn_9 = FALSE,
  knn_9_vals = FALSE,
  knn_9_prob = FALSE,
  knn_9_lab = FALSE
)
models_nb <- reactiveValues(
  nb_1 = FALSE,
  nb_1_vals = FALSE,
  nb_1_prob = FALSE,
  nb_1_lab = FALSE,
  nb_2 = FALSE,
  nb_2_vals = FALSE,
  nb_2_prob = FALSE,
  nb_2_lab = FALSE

)
models_ann <- reactiveValues(
```

```
  ann_1 = FALSE,
  ann_1_vals = FALSE,
  ann_1_plot = FALSE,
  ann_1_prob = FALSE,
  ann_1_lab = FALSE,
  ann_3 = FALSE,
  ann_3_vals = FALSE,
  ann_3_plot = FALSE,
  ann_3_prob = FALSE,
  ann_3_lab = FALSE,
  ann_5 = FALSE,
  ann_5_vals = FALSE,
  ann_5_plot = FALSE,
  ann_5_prob = FALSE,
  ann_5_lab = FALSE
)
models_svm <- reactiveValues(
  svm_1 = FALSE,
  svm_1_vals = FALSE,
  svm_1_plot = FALSE,
  svm_1_prob = FALSE,
  svm_1_lab = FALSE,
  svm_2 = FALSE,
  svm_2_vals = FALSE,
  svm_2_plot = FALSE,
  svm_2_prob = FALSE,
  svm_2_lab = FALSE,
  svm_3 = FALSE,
  svm_3_vals = FALSE,
  svm_3_plot = FALSE,
  svm_3_prob = FALSE,
  svm_3_lab = FALSE,
  svm_4 = FALSE,
  svm_4_vals = FALSE,
  svm_4_plot = FALSE,
  svm_4_prob = FALSE,
  svm_4_lab = FALSE
)
models_dt <- reactiveValues(
  dt_1 = FALSE,
  dt_1_vals = FALSE,
  dt_1_plot = FALSE,
  dt_1_prob = FALSE,
  dt_1_lab = FALSE
)
models_rf <- reactiveValues(
  rf_1 = FALSE,
  rf_1_vals = FALSE,
  rf_1_plot = FALSE,
  rf_1_prob = FALSE,
  rf_1_lab = FALSE
)
reset_models <- function(x) {
  models_knn$knn_3 = FALSE
  models_knn$knn_3_vals = FALSE
  models_knn$knn_3_prob = FALSE
  models_knn$knn_3_lab = FALSE
  models_knn$knn_5 = FALSE
  models_knn$knn_5_vals = FALSE
  models_knn$knn_5_prob = FALSE
  models_knn$knn_5_lab = FALSE
  models_knn$knn_7 = FALSE
  models_knn$knn_7_vals = FALSE
  models_knn$knn_7_prob = FALSE
  models_knn$knn_7_lab = FALSE
  models_knn$knn_9 = FALSE
  models_knn$knn_9_vals = FALSE
  models_knn$knn_9_prob = FALSE
  models_knn$knn_9_lab = FALSE
  models_nb$nb_1 = FALSE
  models_nb$nb_1_vals = FALSE
  models_nb$nb_1_prob = FALSE
  models_nb$nb_1_lab = FALSE
  models_nb$nb_2 = FALSE
  models_nb$nb_2_vals = FALSE
  models_nb$nb_2_prob = FALSE
  models_nb$nb_2_lab = FALSE
  models_ann$ann_1 = FALSE
  models_ann$ann_1_vals = FALSE
  models_ann$ann_1_plot = FALSE
  models_ann$ann_1_prob = FALSE
  models_ann$ann_1_lab = FALSE
  models_ann$ann_3 = FALSE
  models_ann$ann_3_vals = FALSE
  models_ann$ann_3_plot = FALSE
  models_ann$ann_3_prob = FALSE
  models_ann$ann_3_lab = FALSE
  models_ann$ann_5 = FALSE
  models_ann$ann_5_vals = FALSE
```

```
  models_ann$ann_5_plot = FALSE
  models_ann$ann_5_prob = FALSE
  models_ann$ann_5_lab = FALSE
  models_svm$svm_1 = FALSE
  models_svm$svm_1_vals = FALSE
  models_svm$svm_1_plot = FALSE
  models_svm$svm_1_prob = FALSE
  models_svm$svm_1_lab = FALSE
  models_svm$svm_2 = FALSE
  models_svm$svm_2_vals = FALSE
  models_svm$svm_2_plot = FALSE
  models_svm$svm_2_prob = FALSE
  models_svm$svm_2_lab = FALSE
  models_svm$svm_3 = FALSE
  models_svm$svm_3_vals = FALSE
  models_svm$svm_3_plot = FALSE
  models_svm$svm_3_prob = FALSE
  models_svm$svm_3_lab = FALSE
  models_svm$svm_4 = FALSE
  models_svm$svm_4_vals = FALSE
  models_svm$svm_4_plot = FALSE
  models_svm$svm_4_prob = FALSE
  models_svm$svm_4_lab = FALSE
  models_dt$dt_1 = FALSE
  models_dt$dt_1_vals = FALSE
  models_dt$dt_1_plot = FALSE
  models_dt$dt_1_prob = FALSE
  models_dt$dt_1_lab = FALSE
  models_rf$rf_1 = FALSE
  models_rf$rf_1_vals = FALSE
  models_rf$rf_1_plot = FALSE
  models_rf$rf_1_prob = FALSE
  models_rf$rf_1_lab = FALSE
}
```

C Manual de usuario

C.1. Instalación en local

Para el uso de la aplicación en local es necesario disponer de:

- Los ficheros de código adjuntos en el apénice B.
- Tener insinstalado RStudio.
- Disponer de la version *R 3.5.0 (2018-04-23) – "Joy in Playing"*
- Descargas los datos deseados para analizar.
- Instalar los paquetes que aparecen en global.R
- Disponer de un navegador web.

Una vez que se hayan cumplido los pasos anteriores, dsde el fichero app.R se podrá pulsar *Run App* y se abriraá el navegador con la aplicación.

C.2. Uso web

Se ha creado durante un servicio web en AWS para que se pueda acceder a la aplicación durante los días de evaluación de la defensa. En caso de querer acceder a la url del servicio, por favor comuníquemelo para levantar de forma temporal la instancia y proporcionarle la ip y puerto de acceso al servicio.

En el documento de DEFENSA se indicará la url definitiva.

C.3. La aplicación

Para hacer uso de la aplicación hay que ir siguiendo el flujo del pipeline definido en esta memoria. Es decir, no se podrán realizar acciones sobre los datos sin haberlos cargado antes. O intentar ejecutar modelos sin haber hecho una selección de los modelos que se quieren probar. En el documento de defensa se incluirá una demo que servirá de quía para el uso de la app.