# Desarrollo de una aplicación web para la clasificación de tumores renales, utilizando técnicas de aprendizaje automático

**Hugo Suárez González**

Bioinformática Estadística y
Aprendizaje Automático

Máster en Bioinformática y Bioestadística

Nombre del tutor/a de TF:
**Romina Astrid Rebrij**

Nombre del/de la PRA:
Carles Ventura Royo

14 de enero de 2023

Universitat Oberta
de Catalunya

# Ficha Del Trabajo Final

| | |
|---|---|
| **Título del trabajo:** | Desarrollo de una aplicación web para la clasificación de tumores renales, utilizando técnicas de aprendizaje automático |
| **Nombre del autor/a:** | Hugo Suárez González |
| **Nombre del tutor/a de TF:** | Romina Astrid Rebrij |
| **Nombre del/de la PRA:** | Carles Ventura Royo |
| **Fecha de entrega:** | 14 de enero de 2023 |
| **Titulación o programa:** | Máster en Bioinformática y Bioestadística |
| **Área del trabajo final:** | Bioinformática Estadística y Aprendizaje Automático |
| **Idioma del trabajo:** | Castellano |
| **Palabras clave:** | Aprendizaje automático, Riñón, Carcinoma renal de células claras, Carcinoma papilar de células renales, Carcinoma renal de células cromófobas |

## Resumen del trabajo

El aprendizaje automático se ha establecido como un gran aliado en áreas como la biomedicina, la medicina y la oncología. Debido a la enorme cantidad de datos que estas disciplinas médicas producen, la inteligencia artificial, con su gran capacidad de detectar patrones, es una herramienta clave para obtener diagnósticos más fiables y personalizados. El carcinoma de células renales es el cáncer de riñón más frecuente. En la mayoría de casos se detecta fortuitamente. Cuando se encuentra un tumor, es fundamental saber en qué etapa se encuentra y qué tipo de cáncer es para un mejor pronóstico y tratamiento del mismo. En este trabajo se han entrenado y evaluado los algoritmos de aprendizaje automático ANN, DT, KNN, NB RF y SVM utilizando datos de expresión recopilados por The Cancer Genome Atlas, consistentes en 60660 genes obtenidos de 981 muestras de pacientes diagnosticados con RCC para hacer una selección del mejor modelo en función de su rendimiento medio en la predicción del subtipo del tumor, es decir, si es de células claras, papilar o cromófobo, y también si está en etapa temprana o tardía. El algoritmo escogido fue SVM, cuyo rendimiento medio fue de 94,82 %, mostrando una exactitud media al clasificar las muestras de 93,55 %. Posteriormente, este modelo fue implementado en una aplicación web en Shiny.

**Abstract**

Machine learning has established itself as a great ally in areas such as biomedicine, medicine and oncology. Due to the enormous amount of data that these medical disciplines produce, artificial intelligence, with its great ability to detect patterns, is a key tool to obtain more reliable and personalized diagnoses. Renal cell carcinoma is the most common kidney cancer. In most cases it is detected by chance. When a tumor is found, it is essential to know what stage it is in and what type of cancer it is for a better prognosis and treatment of the tumor. In this work, ANN, DT, KNN, NB RF and SVM machine learning algorithms have been trained and evaluated using expression data collected by The Cancer Genome Atlas of 60660 genes obtained from 981 RCC patient samples to make a selection of the best model based on their average performance in predicting the subtype of the tumor, i.e. whether it is clear cell, papillary or chromophobe, and also whether it is early or late stage. The algorithm chosen was SVM, whose average performance was 94,82 %, showing an average accuracy in classifying samples of 93,55 %. Subsequently, this model was implemented in a web application in Shiny.

# Índice general

# Índice de figuras

# Índice de cuadros

# Capítulo 1

# Introducción

## 1.1. Contexto y justificación del trabajo

El cáncer de riñón se encuentra entre los 10 tumores más comunes en los Estados Unidos, y su incidencia en Europa y Norteamérica ha aumentado a más del doble en los últimos 50 años [1]. En 2012 fue el séptimo cáncer más común globalmente, siendo el 3.3 % de los nuevos casos de cáncer diagnosticados. La proporción más elevada de cáncer de riñón se produce en Norteamérica, Europa, Australia y Nueva Zelanda, y el 52 % de las muertes globales se produjeron en estas zonas [2]. Según la American Cancer Society, en 2021 hubo 79000 nuevos casos y 13920 muertes, siendo en ambas el 64 % hombres, aproximadamente. Sin embargo, a pesar de que la incidencia es más alta en países con más recursos, el acceso a tratamientos y la detección temprana ha provocado en estos países una tendencia estabilizadora, reduciendo el tamaño de los tumores a la hora del diagnóstico [3]. Por otro lado, en países de Europa del este ha habido un aumento en las tasas de mortalidad, y se prevé que en 2030, Brasil y Ecuador podrían tener el mayor aumento en la incidencia de este cáncer [4].

En su artículo *Cancer Statistics* [5], se menciona que en general en diferentes tipos de cáncer, además de diferencias entre sexos, resaltan las diferencias étnicas en la supervivencia según el tipo de tumor, siendo normalmente los pacientes afroamericanos los que peor pronóstico muestran. Sin embargo, en el caso particular del cáncer de riñón, se ve una menor diferencia en supervivencia general basándose en la etnia del paciente, pero esto se debe a que el carcinoma renal de células claras (ccRCC, por sus siglas en inglés), el subtipo más frecuente de neoplasia renal, es más común entre la población blanca, y éste es el que peor pronóstico suele tener [6].

El carcinoma de células renales (RCC, por sus siglas en inglés) es la afección más común entre los cánceres de riñón, comprendiendo entre el 90 % y el 95 % de los casos. Los tres tipos más comunes de RCC son el carcinoma renal de células claras, siendo aproximadamente el 65-70 % de los casos, el carcinoma papilar de células renales, 15-20 % y el carcinoma de células renales cromófobo, 6-11 % [7]. El ccRCC es, además del subtipo más común, el que cuenta con más muertes entre todos los cánceres de riñón.

A pesar de haber aumentado su incidencia y ser un cáncer tan mortífero, el RCC sigue siendo difícil de detectar y muchas veces es asintomático hasta que llega a las etapas tardías, en las que su pronóstico se ve gravemente empeorado y la probabilidad de supervivencia se vuelve muy baja, lo que hace que un diagnóstico rápido y eficaz sea muy necesario. La mayoría de los tumores se detectan, de hecho, investigando otras afecciones abdominales [8]. El 20 % de los tumores menores de 4 centímetros hallados en riñones de forma incidental son benignos, pero actualmente hay pocos métodos para diferenciar entre tumores benignos y malignos [9].

En los últimos años, ha habido muchos avances en la utilización del aprendizaje automático para fines biomédicos gracias a su gran versatilidad, ya que se pueden emplear múltiples tipos de datos diferentes. Por ello, la inteligencia artificial se ha utilizado ampliamente para diagnóstico o clasificación de enfermedades basándose en imágenes, dada su gran capacidad de hallar patrones [10]. En el caso de los RCC, hay varias disciplinas involucradas en su estudio, como puede ser la urología, patología, oncología o radiología, y cada una de esas disciplinas genera una gran cantidad de datos médicos [11]. Por ello, la inteligencia artificial se presenta como una herramienta muy útil en el estudio del cáncer de riñón, y podemos encontrar múltiples proyectos en los que se ha utilizado para diagnóstico y clasificación de los RCC mediante datos de imagen, utilizando imágenes obtenidas de histopatologías, resonancias magnéticas o tomografías computarizadas [9].

Utilizando datos de expresión génica, a pesar de que ya ha habido esfuerzos en utilizar aprendizaje automático para la clasificación del tipo de tumor y la detección de si es tumor temprano o tardío, todavía se podría mejorar el rendimiento de los algoritmos de clasificación propuestos. En los últimos años el rendimiento de los modelos ha aumentado notablemente, pero no existe ningún modelo que clasifique subtipo de tumor e indique su estadío. El rendimiento máximo obtenido al diferenciar el estadío del tumor ha sido de 81.15 %, con un área bajo la curva (AUC) de 0.86 para carcinoma renal de células claras, el subtipo más común [12]. En 2018, se alcanzó un rendimiento de 88 %, con un AUC de 0.804 en carcinoma papilar [13] y éste se aumentó al 94,2 % en 2020 [14]. Se ha hecho un mejor trabajo al clasificar el tipo de tumor utilizando datos de miRNA, con un rendimiento medio de 95 % [15], pero poder determinar el estadío del tumor aparte del tipo es muy relevante, ya que de él depende gravemente el pronóstico de la enfermedad. En los últimos años se han estudiado nuevos biomarcadores y diferencias genéticas entre hombres y mujeres que podrían ayudar a un mejor rendimiento de los clasificadores [16].

El Atlas del Genoma del Cáncer (The Cancer Genome Atlas, o TCGA) es un programa supervisado por el National Cancer Institute (NCI) y el National Human Genome Research Institute, que desde el 2006 recoge datos del genoma, transcriptoma, epigenoma y proteoma humano con el objetivo de obtener una mejor comprensión del cáncer [17]. A lo largo de los años este programa ha generado una cantidad masiva de datos que se han utilizado para mejorar el diagnóstico, prevención y tratamiento del cáncer. Ya que los datos son de acceso público, muchos estudios de aprendizaje automático los utilizan, como los mencionados anteriormente [12,13,15]. Es por ese motivo que los datos utilizados en este trabajo son descargados de TCGA, de los proyectos TCGA-KIRC, TCGA-KIRP y TCGA-KICH. Estos son tres proyectos dedicados a

la investigación del RCC. El primero estudia el ccRCC, el segundo estudia el carcinoma renal papilar o pRCC, y el tercero el carcinoma renal cromófobo. Estos proyectos intentan conectar fenotipos con genotipos para estos tipos de cáncer, y recopilan datos de imágenes clínicas como histopatologías, datos transcriptómicos y datos epigenómicos entre otros. Estos datos se recogieron en diferentes partes del mundo, lo que la hace una muestra bastante heterogénea. Se obtuvieron normalmente no para un estudio específico, sino simplemente para la rutina de cuidado de cada paciente y más tarde se recopilaron en las base de datos [18].

Como ya se ha mencionado, hay muchos estudios que utilizan datos obtenidos en TCGA para entrenar modelos de aprendizaje automático. Sin embargo, utilizando datos de expresión génica, encontramos relativamente pocos estudios basados en RCC, y éstos no han tenido un rendimiento tan elevado como otros estudios realizados con datos de expresión de muestras de tumores localizados en diferentes órganos [19], y podemos encontrar algoritmos con un rendimiento mayor al 90 % en artículos que, como este proyecto, utilizan datos obtenidos de TCGA, basados en leucemia [20], cáncer de mama [21] o cáncer de pulmón [22]. Es por eso que este artículo se centró en utilizar datos de TCGA de los tres proyectos, KIRC, KIRP y KICH, para desarrollar un modelo que pudiera superar el rendimiento medio de los ya publicados. Teniendo en cuenta que ningún algoritmo publicado puede clasificar entre los tres subtipos más comunes de RCC y además indicar el estado del tumor, este proyecto, al tener como objetivo combinar la clasificación de subtipo y estado, no se puede basar en el rendimiento de uno sólo de los trabajos mencionados, ya que sintetiza los objetivos de los tres [12, 14, 15]. Por lo tanto, alcanzar un rendimiento superior al modelo de predicción de estadíos de ccRCC sería ya un avance sobre lo publicado. Se calculó la media del rendimiento de los tres artículos en los que nos hemos basado, y se intentará superar ese rendimiento.

## 1.2. Objetivos del trabajo

- **Objetivo General**
  Desarrollar una aplicación web que permita predecir el tipo de tumor renal, y su estado (temprano/tardío) a partir de datos de expresión génica.

- **Objetivos específicos**

  1. Identificar genes relevantes mediante un análisis de expresión diferencial
  2. Evaluar los modelos en base a cuatro métricas: Exactitud, Sensibilidad, Especificidad y el valor de Kappa.
  3. Optimizar el modelo seleccionado para lograr un rendimiento mínimo del 90,13 % (usando la métrica Exactitud) en diferenciar el tipo de tumor y diferenciar entre estadío temprano o tardío.
  4. Desarrollar una aplicación web interactiva en base al modelo seleccionado.

## 1.3. Impacto en sostenibilidad, ético-social y de diversidad

Este trabajo se encuadra dentro del Objetivo de Desarrollo Sostenible (ODS) 3 - Salud y Bienestar [23], en concreto se relaciona con la meta 3.4, *Para 2030, reducir en un tercio la mortalidad prematura por enfermedades no transmisibles mediante la prevención y el tratamiento y promover la salud mental y el bienestar*, ya que la detección temprana del RCC es un factor fundamental para una mayor supervivencia. También se encuadra dentro del Objetivo 5 - Igualdad de género, y el Objetivo 10 - Reducción de las desigualdades, ya que en el trabajo se consideran ambos sexos y distintas etnias. Existe la necesidad de conseguir más muestras de los grupos minoritarios para que se encuentren mejor representados en los modelos y que no exista una discriminación por infrarrepresentación para el diagnóstico, tratamiento y/o seguimiento de enfermedades.

- **Dimensión sostenibilidad**
  Dado que este trabajo se basará en un análisis bioinformático de datos obtenidos de un repositorio público, sin utilizar ninguna materia prima ni consumir ningún recurso más allá del que pueda generar un ordenador personal, se considera que no tendrá ningún impacto ni positivo ni negativo en aspectos de sostenibilidad medioambiental y/o huella ecológica.

- **Dimensión comportamiento ético y de responsabilidad social (RS)**
  Los datos utilizados en este trabajo son de acceso público, la privacidad de las personas participantes en el estudio que recogió estos datos ha sido cuidada y mantenida en el momento de crear la base de datos. Por otro lado, el producto utilizado no reportará ningún beneficio económico. Por estos motivos, se ha considerado que no tendrá impactos positivos ni negativos en aspectos ético-sociales.

- **Dimensión diversidad, género y derechos humanos.**
  En la realización de este trabajo, se ha tratado de tener una visión amplia sobre las diversidades de género, etnia, etc., empleando un lenguaje no discriminatorio y tratando, en la medida de lo posible, de resaltar los sesgos encontrados en los datos e información empleada al realizar el trabajo, teniendo en cuenta que históricamente ha existido una tendencia a sobrerrepresentar a hombres caucásicos de clase media/alta, y por lo tanto, desfavorecer a personas de otros géneros, etnias y/o grupos sociales.

## 1.4. Enfoque y método seguido

Desde el desarrollo de las técnicas de secuenciación masiva, el aprendizaje automático es ampliamente utilizado en el campo de la salud, ya que ayuda a trabajar con la gran cantidad de datos que estas técnicas están generando. En este caso, se utilizaron datos del repositorio del Atlas del Genoma del Cáncer (TCGA) [17], de los proyectos TCGA-KIRC, TCGA-KIRP y TCGA-KICH, que recogen un gran número de datos genómicos, transcriptómicos, proteómicos y epigenómicos, con la finalidad de ayudar en mejorar el conocimiento, diagnóstico y tratamiento del cáncer de células renales.

Teniendo en cuenta la estructura de los datos, se ha decidido emplear el método de clasificación mediante aprendizaje supervisado, ya que se disponía de etiquetas de diagnóstico asociadas a los pacientes en el conjunto de datos. El enfoque de aprendizaje supervisado se centra en la optimización de un modelo, con el objetivo de determinar la combinación óptima de los valores de las características presentes en los datos. Esta combinación conduce a la predicción precisa de una etiqueta específica, en este caso, el diagnóstico de los pacientes en la base de datos. El trabajo se desarrolló en el lenguaje de programación R, utilizando el entorno RStudio. De los algoritmos de aprendizaje supervisado que existen, se usaron, por tanto, los dedicados a clasificación, como son: Nearest Neighbor, Naive Bayes, Decision Trees, Random Forests, Neural Networks y Support Vector Machines.

El método a seguir fue el siguiente:

1. Recopilación de los datos
2. Exploración y preprocesamiento de los datos
3. Análisis de expresión de los genes
4. Construcción y entrenamiento de los modelos de clasificación
5. Evaluación de los modelos y elección del mejor modelo
6. Mejora del modelo escogido
7. Evaluación del modelo mejorado

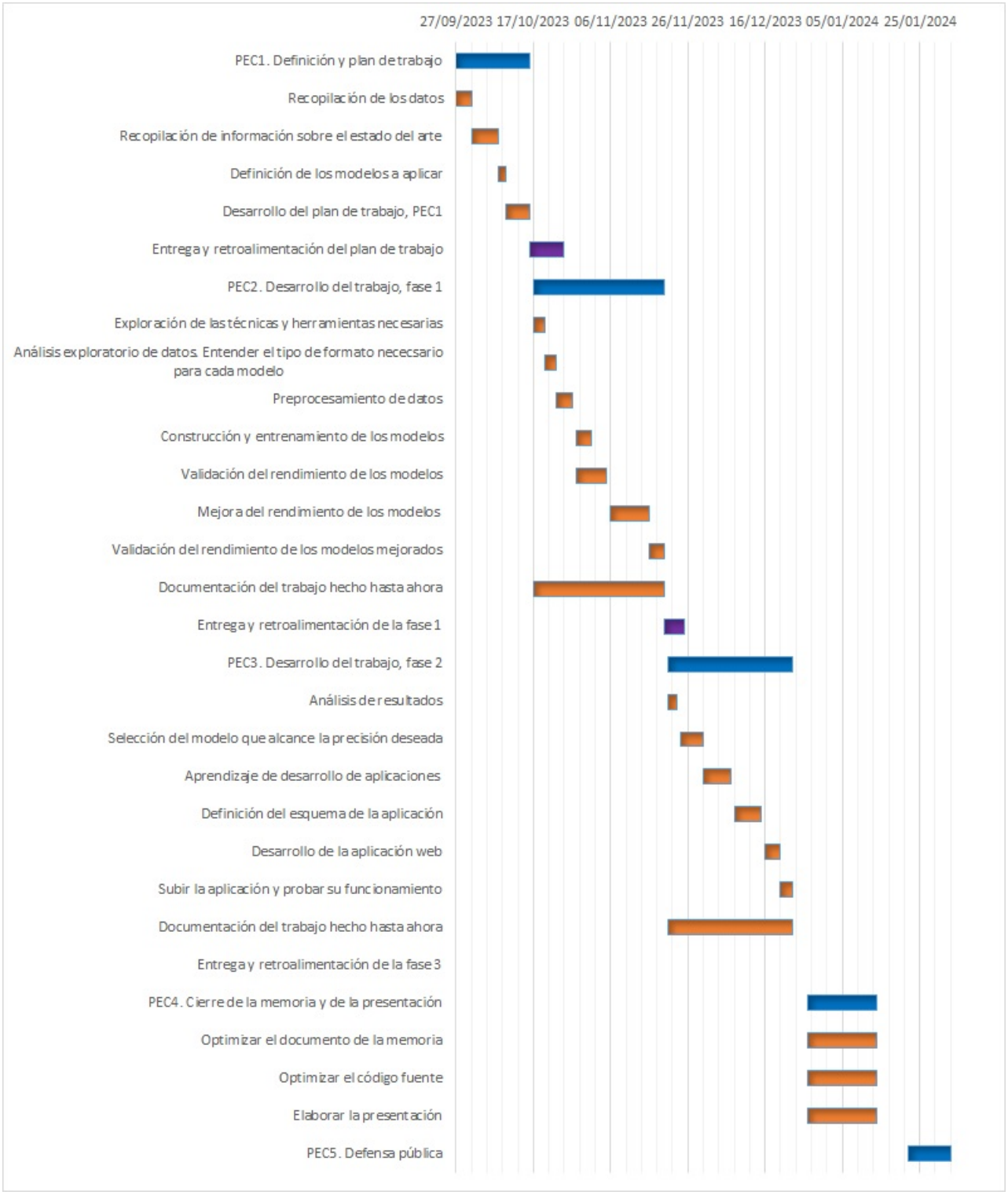
Posteriormente, una vez alcanzado el rendimiento requerido, se elaboró una aplicación web utilizando el paquete Shiny de R en la cual se puedan introducir datos nuevos de pacientes con RCC y la aplicación nos indicará el tipo de tumor y su estadío.

## 1.5. Planificación del trabajo

1. **Tareas**

| Descripción | Fecha inicio | Fecha fin |
|---|---|---|
| **PEC1. Definición y plan de trabajo** | **27/09/2023** | **16/10/2023** |
| Recopilación de los datos | 27-sep | 01-oct |
| Recopilación de información sobre el estado del arte | 01-oct | 08-oct |
| Definición de los modelos a aplicar | 08-oct | 10-oct |
| Desarrollo del plan de trabajo, PEC1 | 10-oct | 16-oct |
| Entrega y retroalimentación del plan de trabajo | 16-oct | 25-oct |
| **PEC2. Desarrollo del trabajo, fase 1** | **17-oct** | **20-nov** |
| Exploración de las técnicas y herramientas necesarias | 17-oct | 20-oct |
| Análisis exploratorio de datos. Entender el tipo de formato nececsario para cada modelo | 20-oct | 23-oct |
| Preprocesamiento de datos | 23-oct | 27-oct |
| Construcción y entrenamiento de los modelos | 28-oct | 01-nov |
| Validación del rendimiento de los modelos | 28-oct | 05-nov |
| Mejora del rendimiento de los modelos | 06-nov | 16-nov |
| Validación del rendimiento de los modelos mejorados | 16-nov | 20-nov |
| Documentación del trabajo hecho hasta ahora | 17-oct | 20-nov |
| Entrega y retroalimentación de la fase 1 | 20-nov | 25-nov |
| **PEC3. Desarrollo del trabajo, fase 2** | **21-nov** | **23-dic** |
| Análisis de resultados | 21-nov | 23-nov |
| Selección del modelo que alcance la precisión deseada | 24-nov | 30-nov |
| Aprendizaje de desarrollo de aplicaciones | 30-nov | 07-dic |
| Definición del esquema de la aplicación | 08-dic | 15-dic |
| Desarrollo de la aplicación web | 16-dic | 20-dic |
| Subir la aplicación y probar su funcionamiento | 20-dic | 23-dic |
| Documentación del trabajo hecho hasta ahora | 21-nov | 23-dic |
| Entrega y retroalimentación de la fase 3 | 23-dic | 27-dic |
| **PEC4. Cierre de la memoria y de la presentación** | **27-dic** | **14-ene** |
| Optimizar el documento de la memoria | 27-dic | 14-ene |
| Optimizar el código fuente | 27-dic | 14-ene |
| Elaborar la presentación | 27-dic | 14-ene |
| **PEC5. Defensa pública** | **22-ene** | **02-feb** |

## 2. Calendario



3. **Hitos**

| Descripción | Fecha |
|---|---|
| Entrega del plan de trabajo | 16/10/2023 |
| Entrega del desarrollo de trabajo, fase 1 | 20/11/2023 |
| Entrega del desarrollo de trabajo, fase 2 | 23/12/2023 |
| Entrega de la memoria y presentación | 14/01/2024 |
| Defensa pública | 02/02/2024 |

4. **Análisis de Riesgos**

| Descripción del riesgo | Severidad | Probabilidad | Mitigación |
|---|---|---|---|
| Problemas con el funcionamiento de las librerías requeridas | Alta | Baja | Investigar sobre librerías alternativas |
| No poder aplicar los modelos seleccionados a los datos | Alta | Baja | Investigar sobre el correcto preprocesamiento de los datos |
| No alcanzar el rendimiento mínimo objetivo | Alta | Moderada | Investigar sobre todos los factores para mejorar el rendimiento |
| No poder desarrollar la aplicación web en Shiny | Alta | Baja | Investigar sobre Shiny, considerar programas alternativos |

## 1.6. Breve sumario de productos obtenidos

1. **Plan de Trabajo**
   Documento en forma de PDF que contenga pautas y tiempos estimados para todas las tareas a realizar para el correcto desarrollo del proyecto.

2. **Memoria**
   Documento en forma de PDF en el que se recopile toda la investigación, desarrollo, resultados, conclusiones y código utilizado a lo largo del trabajo

3. **Producto**
   Enlace de acceso a la aplicación web desarrollada en Shiny para la clasificación de carcinomas renales. Acceso a un repositorio público que permita acceder al código utilizado.

4. **Presentación virtual**
   Presentación en formato PPT para defender el trabajo realizado

## 1.7. Breve descripción de los otros capítulos de la memoria

En esta sección mostramos una breve explicación de los contenidos de cada capítulo y su relación con el proyecto global.

1. **Estado del arte**: En esta sección se ofrece un análisis bibliográfico sobre el estado actual de la investigación de algoritmos de aprendizaje automático en el diagnóstico o clasificación del RCC. También explica la relevancia de este proyecto en contexto con los ya publicados.

2. **Materiales y métodos**: En esta sección se describen en detalle los materiales y métodos empleados para lograr los objetivos del proyecto, que incluyen la creación de un modelo de aprendizaje automático para clasificación de subtipos y estadío de RCC y su implementación en una aplicación web. Muestra una explicación concisa del tratamiento que sufrieron los datos, los principales algoritmos utilizados, las métricas de evaluación y el desarrollo de la aplicación web.

3. **Resultados**: En esta sección se muestran los resultados más significativos obtenidos a lo largo del proyecto.

4. **Discusión**: En esta sección se justifica la elección del modelo de clasificación utilizado, profundizando en los resultados obtenidos.

5. **Conclusiones y trabajos futuros**: En esta sección se presentan las conclusiones derivadas del trabajo realizado, así como las posibles direcciones para investigaciones futuras. También se realiza una evaluación sobre el cumplimiento de los objetivos planificados inicialmente.

6. **Glosario**: Este apartado contiene una recopilación alfabética de las definiciones de los términos técnicos empleados a lo largo del proyecto.

7. **Bibliografía**: En esta sección se incluye una lista numerada de todas las fuentes de información referenciadas en la memoria, ordenadas según su aparición en el texto.

8. **Anexos**: Esta sección muestra el código utilizado para la obtención de resultados a lo largo del proyecto, junto con sus explicaciones correspondientes.

# Capítulo 2

# Estado del arte

Como ya se ha mencionado anteriormente, actualmente hay muchos esfuerzos puestos en desarrollar modelos de aprendizaje automático con fines biomédicos, como por ejemplo el diagnóstico de enfermedades o la evaluación del efecto de un tratamiento. Para el caso de los carcinomas renales, hay múltiples investigaciones utilizando diversos tipos de datos de diagnóstico. Muchas de estas investigaciones utilizan los datos obtenidos mediante el proyecto TCGA, que reúne tanto datos genómicos, transcriptómicos y epigenómicos como imágenes obtenidas mediante distintas técnicas para diagnóstico. Este proyecto se ha fundamentado principalmente en los cuatro trabajos citados a continuación.

En 2017, Sherry Bhalla et. al. [24] desarrollaron un modelo que clasifica muestras de carcinoma renal de células claras según su estadío, utilizando datos de RNA-seq obtenidos en la base de datos de TCGA bajo el nombre de proyecto TCGA-KIRC. Su modelo clasifica las muestras en dos estadíos, temprano y tardío, con un rendimiento del 72.64 %. Más tarde, en 2020, Fangjun Li et. al. [12] mejoraron este modelo utilizando los mismos datos gracias a una mejor selección de las variables, aumentando el rendimiento hasta un 81.15 %.

En 2018, Noor Pratap Singh et. al. [13] publican un modelo que clasifica tumores en estadío temprano y tardío para muestras de carcinoma renal papilar con un rendimiento del 80 %, también utilizando datos de expresión génica obtenidos en TCGA, en el proyecto TCGA-KIRP. Posteriormente, en 2020, Sugi Lee et al elaboraron un modelo de aprendizaje profundo dedicado a clasificar datos de pRCC en estadío temprano o tardío, con un rendimiento del 94,2 % [14].

En 2018, Ali Muhamed Ali et. al [15] desarrollan un modelo para clasificar los tumores en cinco subtipos de RCC utilizando datos de miRNA y obteniendo un rendimiento del 95 %. Los subtipos son: carcinoma renal de células claras o ccRCC, carcinoma renal papilar o pRCC, carcinoma renal cromófobo o chRCC, tumor de Wilms y tumor rabdoide.

Por otro lado, dado que las técnicas más comúnmente usadas de diagnóstico por imagen están normalmente digitalizadas, hay muchos proyectos que emplean imágenes obtenidas de estas técnicas para utilizarlas entrenando algoritmos de aprendizaje automático para la detección, clasificación o pronóstico de tumores renales. Hay múltiples artículos en los que desarrollan un algoritmo utilizando imágenes obtenidas mediante resonancia magnética (MRI por sus siglas

en inglés) o tomografía computarizada (CT por sus siglas en inglés) para diferenciar entre tumores renales benignos y malignos [25–28] o para clasificar subtipos de tumor [29–31]. En 2021, Stefan Schulz et. al. [11] desarrollaron un modelo de aprendizaje multimodal para la predicción del pronóstico de ccRCC utilizando imágenes obtenidas mediante histopatologías, MRI y CT, y también datos genéticos obtenidos por secuenciación del genoma completo, obteniendo un rendimiento en la clasificación del 83.43 %.

En este trabajo se utilizan datos transcriptómicos obtenidos en los proyectos TCGA-KIRC, TCGA-KIRP y TCGA-KICH para desarrollar un algoritmo que clasifica los datos en tres tipos de tumor ccRCC, pRCC o chRCC y nos indica el estadío de dicho tumor. El proyecto tiene como objetivo desarrollar un modelo de predicción mejorado en comparación con los estudios anteriores, y su distinción radica en dos cosas: La primera, es su capacidad para indicar el subtipo de tumor y su estadío, evitando así tener que usar herramientas diferentes para diagnosticar el subtipo y posteriormente la etapa. La segunda es su capacidad para ofrecer de manera fácil y práctica resultados a través de una aplicación web. Esta aplicación podrá estar al alcance de cualquier persona, lo que la convierte en una herramienta poderosa para respaldar a los profesionales de la salud en la toma de decisiones sobre el diagnóstico de pacientes con cáncer de riñón.

# Capítulo 3

# Materiales y métodos

## 3.1. Software

Para el desarrollo de todo el trabajo se ha utilizado el lenguaje de programación R en RStudio, versión 2023.09.1+494. Las librerías utilizadas para la creación de los modelos son ROCR versión 1.0-11 para el KNN, e1071 versión 1.7-14 para Naive Bayes, neuralnet versión 1.44.2 para la ANN, kernlab versión 0.9-32 para el SVM, C50 versión 0.1.8 para el árbol de decisión y randomForest versión 4.7-1.1 para el Random Forest. Se utilizó ggplot versión 3.4.4 para los gráficos; data.table versión 1.14.10, reshape versión 0.8.9, tidyr versión 1.3.0, tibble versión 3.2.1 y dplyr versión 1.1.4 para el manejo de los datos. Se utilizó DESeq2 versión 1.34.0 para el análisis de expresión y scutr versión 0.2.0 para el balanceo de datos mediante SMOTE. Para el Análisis de Componentes Principales se usó prcomp de R. Para el desarrollo de la aplicación, se usaron las librerías shiny versión 1.8.0, shinycssloaders versión 1.0.0 y shinyjs versión 2.1.0.

## 3.2. Hardware

El proyecto se realizó en un ordenador personal Asus R510V, con CPU Intel Core i5-6300HQ, memoria RAM 8GBx2, disco duro SDD de 1TB.

## 3.3. Origen de los datos

Los datos están disponibles en la página web de TCGA [17], accediendo al portal de Genomic Data Commons (GDC), seleccionando los proyectos TCGA-KIRC, TCGA-KIRP y TCGA-KICH. La estrategia experimental seleccionada fue RNA-seq, y el tipo de datos escogido fue Gene Expression Quantification.

## 3.4. Métodos

### 3.4.1. Obtención de los datos

Con los datos de los proyectos TCGA-KIRC, TCGA-KIRP y TCGA-KICH seleccionados, se descargan los siguientes archivos: un archivo TSV con datos de expresión génica por cada muestra, que se encuentra dentro de una carpeta, y tanto el archivo como la carpeta tienen nombres específicos y diferentes para cada muestra; una hoja con datos clínicos, una tercera hoja que nos indica el nombre de la carpeta que contiene el archivo de RNAseq, el nombre del archivo de RNAseq, y el identificador del paciente para que podamos obtener los datos recorriendo cada carpeta y unir los datos de RNAseq con los datos clínicos; y por último otra que nos proporciona datos sobre cada muestra, entre otras cosas la procedencia de la muestra y si pertenece a tumor o es una muestra control.

El primer paso a aplicar en R fue crear una función que defina una ruta para cada uno de los archivos de RNAseq, leyendo el nombre de la carpeta y el nombre del archivo de cada muestra para pegarlos en la ruta e ir obteniéndolos. De cada archivo, escogemos las columnas que nos indican el identificador del gen, y los datos de expresión. Son dos columnas, de las cuales haremos la traspuesta transformándolas en filas de nuestro dataframe. A esas filas se les añadirá posteriormente el tipo de tumor al que corresponden, su estadío, y si son muestra control. Como el objetivo del trabajo es clasificar el tipo de tumor en tres tipos (células claras, papilar o cromófobo) y en su estado (temprano o tardío), excluímos del dataframe de datos clínicos las muestras que no tengan definido el estado o el tipo de tumor.

Para indicar el estadío del tumor, tenemos en los datos clínicos tumores en 4 estados: Stage I, Stage II, Stage III y Stage IV, procedemos a nombrar como estado temprano los estados I y II, y como tardío los III y IV. Por lo tanto, nuestras muestras se organizan de la siguiente forma:

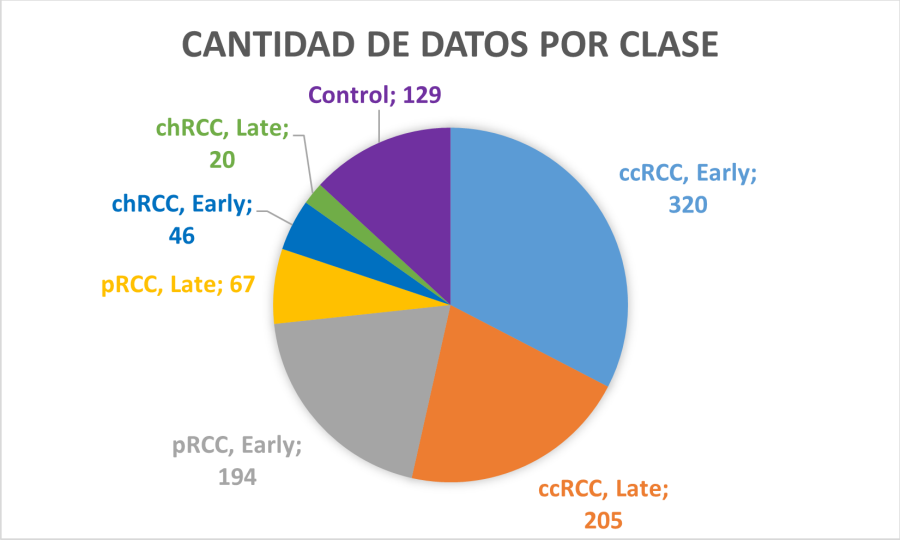



Figura 1: Número de datos por cada clase en Diagnóstico

En la Figura 2 podemos ver que, a pesar de que los proyectos TCGA-KIRC, TCGA-KIRP y TCGA-KICH recopilaron datos en distintas partes del mundo e intentan tener una muestra variada, sigue existiendo un sesgo de género y de etnia, ya que la mayor parte de los datos recogidos proviene de hombres blancos. Es muy importante que se obtengan más muestras de los grupos minoritarios para que se encuentren mejor representados en los modelos, ya que si un modelo se entrena con datos de mayoritariamente hombres caucásicos, éste tendrá buen rendimiento para este grupo poblacional, y sin embargo no funcionará bien para personas que pertenezcan a otro grupo.

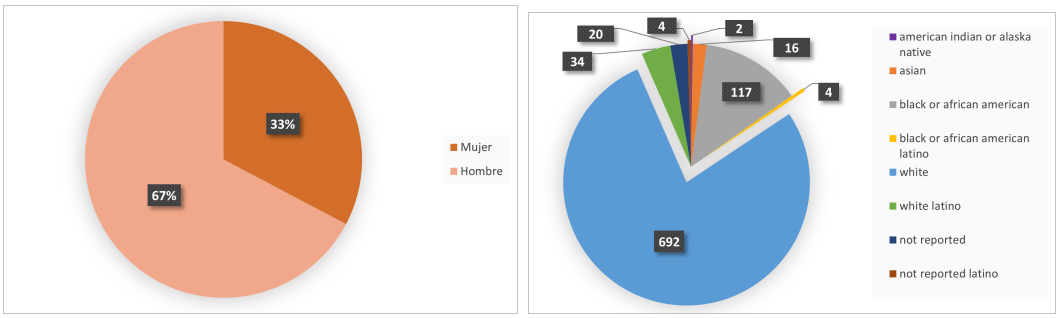



Figura 2: Porcentaje de datos pertenecientes a cada sexo o etnia
(a) Muestras por sexo. (b) Muestras por etnia

### 3.4.2. Análisis exploratorio

Los diagramas de cajas del análisis exploratorio se realizaron con ggplot2 en R. Para mostrar cómo se distribuyen los datos según la clase, que será nuestra variable respuesta, se reorganizan los datos en un nuevo dataframe, que contiene una columna con los nombres de los genes, otra columna con los datos de expresión de cada gen, y otra columna con la variable respuesta *Diagnosis*, que es el tipo de tumor y el estadío. De esta forma tenemos los datos de expresión de todos los genes para cada clase. Se comprobaron datos faltantes en varias fases del proyecto; entre ellas al unir los datos de expresión con los datos de tipo de tejido, al unir estos datos con los datos clínicos, tras normalizar los datos, y en el código de la aplicación.

### 3.4.3. Análisis de expresión y filtrado de genes

Para reducir el número de variables, se empleó un análisis de expresión diferencial con DESeq2, escogiendo como fórmula el estadío del tumor y el tipo de tumor. DESeq2 utiliza la fórmula proporcionada para comparar el número de counts de cada gen en distintas muestras. Hay dos conceptos clave utilizando DESeq2: el Log2FoldChange o LFC y el valor de p o p-value. DESeq2 establece la hipótesis nula de que no hay expresión diferencial entre los grupos a comparar. En ese caso, el LFC sería 0, ya que es la medida que nos muestra si los genes están expresados diferencialmente o no. Un valor positivo de LFC nos indicaría que los genes están sobreexpresados en las muestras tardías, ya que las estamos comparando con las tempranas. El valor de p nos muestra el valor de falsos positivos. Para reducir el número de falsos positivos, DESeq2 utiliza la corrección de Benjamin-Hochberg o FDR y nos proporciona valores de p ajustados, o

p-adj. Se filtraron los genes escogiendo los genes expresados diferencialmente entre los estados temprano y tardío de las muestras, usado un valor de $p_{adj} < 0{,}05$.

### 3.4.4. Preprocesamiento

Tras filtrar los genes con el análisis de expresión, los datos se balancearon con scutr en R, utilizando la función *oversample_smote()*. Synthetic Minority Over-sampling TEchnique, o SMOTE, es una técnica que se utiliza para modelos de clasificación cuando una o varias de las clases a clasificar están sobre o infrarrepresentadas. El hecho de que los datos no estén balanceados puede afectar al rendimiento de los modelos, por lo tanto se emplean técnicas que o bien reducen el número de muestras de la clase mayoritaria, o generan artificialmente nuevas muestras para las clases infrarrepresentadas. Para no perder datos, SMOTE genera nuevas muestras para las clases minoritarias basándose en los vecinos cercanos, calculando la diferencia entre la muestra y su vecino cercano y generando datos nuevos multiplicando esta diferencia por un número entre 0 y 1. La función *oversample_smote()* permite escoger el número de vecinos más cercanos y el número de muestras finales que tendrá la clase, sumando las originales con las generadas. Como la clase que más datos tenía, ccRCC Early, tenía 320 muestras, se procedió a generar las muestras necesarias para acabar teniendo 320 muestras de cada clase.

Los datos se normalizaron en dos pasos; utilizando la función $\log_2(x+1)$, donde x son los datos de expresión por gen y por muestra, a los que se le suma un 1 para no tener datos con expresión 0, y utilizando la normalización Min-Max, que convierte nuestros datos en valores entre 0 y 1. Los datos se separaron en un porcentaje 80 % y 20 % para entrenamiento y para prueba, respectivamente. Se obtuvieron los máximos y mínimos de los datos de entrenamiento para utilizarlos en la función Min-Max, cuya fórmula es la siguiente:

$$x_{norm} = \frac{x - min}{max - min} \tag{3.1}$$

Donde $x$ es el valor y $x_{norm}$ es el nuevo valor normalizado. Con esta función y los máximos y mínimos obtenidos anteriormente, se normalizan todos los datos, y también se utilizará para normalizar los datos futuros.

### 3.4.5. Algoritmos utilizados

Mostramos una breve explicación de los algoritmos que se han utilizado en el proyecto:

- **k-Nearest Neighbours:** El algoritmo k-NN (k Nearest Neighbours, k-vecinos más cercanos) es un algoritmo de machine learning que se encarga de identificar a qué clase pertenecen nuevos datos desconocidos a partir de los "vecinos más próximos", es decir, dado un set de datos ya clasificado, tomando en cuenta la clase de los k vecinos más cercanos a los datos por clasificar, escogerá como resultado la clase mayoritaria entre ellos [32]. Este algoritmo funciona de la siguiente manera: Primero, se define un valor de k, que representa el número de vecinos más cercanos que se considerarán al clasificar una muestra de datos. Luego, para clasificar una muestra, el algoritmo busca los k puntos de datos más cercanos y proporciona como resultado la clase de la muestra en función de

la mayoría de las clases de sus vecinos más cercanos. El valor de k se define antes de la aplicación del algoritmo, y es un parámetro que puede ser ajustado. Si se selecciona un valor pequeño para k, el algoritmo es más sensible al ruido y puede producir una clasificación más inestable, mientras que un valor grande para k puede hacer que el algoritmo pierda precisión en la clasificación. Una de las principales ventajas del algoritmo k-NN es su simplicidad y facilidad de implementación. Además, es un algoritmo no paramétrico, lo que significa que no hace suposiciones sobre la distribución de los datos subyacentes. Esto lo hace adecuado para conjuntos de datos no lineales. Sin embargo, el algoritmo también tiene debilidades, como su sensibilidad a los valores atípicos, la necesidad de una selección cuidadosa del valor de k para una precisión óptima, y la disminución del rendimiento con grandes conjuntos de datos o una gran cantidad de atributos o dimensiones.

- **Naive Bayes:** El algoritmo Naive Bayes es una técnica de aprendizaje automático que estima la probabilidad de diferentes resultados considerando la información proporcionada por las variables. Se basa en el teorema de Bayes, el cual calcula la probabilidad de un evento dadas las condiciones relacionadas con él. Este algoritmo asume que todas las variables son independientes entre sí y que tienen igual importancia, lo que le otorga el adjetivo "Naive"(ingenuo en inglés). Al construir un modelo, crea una tabla de probabilidades que incluye las probabilidades de cada clase y las probabilidades condicionales de las clases con respecto a cada variable. Utilizando estas probabilidades, el modelo puede predecir la probabilidad de que un nuevo conjunto de datos pertenezca a cada clase, seleccionando finalmente la clase con la probabilidad más alta [32]. El modelo Naive Bayes aplica el estimador de Laplace para evitar que la tabla de probabilidades contenga valores de cero. Esto se debe a que las probabilidades se calculan mediante multiplicaciones sucesivas, y la presencia de un cero podría distorsionar significativamente el resultado final sin tener una justificación válida. Entonces, el estimador de Laplace suma un pequeño número, normalmente 1, a cada instancia en la tabla de probabilidades para evitar este inconveniente.

- **Artificial Neural Network:** Las redes neuronales artificiales, o ANN por sus siglas en inglés, son un algoritmo de aprendizaje automático que imita cómo funcionan las neuronas cerebrales, en tanto que modelan la relación entre los datos de entrada y los datos de salida utilizando una red de neuronas artificiales o nodos que funcionan como procesadores en paralelo para resolver un problema de clasificación, regresión o reconocimiento de patrones. Este modelo se compone de una función de activación, que transforma las señales de entrada en una señal de salida; la arquitectura de la red, que consiste en el número de neuronas y capas del modelo y cómo están conectadas; y el entrenamiento, mediante el cual el algoritmo va modificando sus respuestas, devolviéndolas a la red neuronal y de este modo volviéndose cada vez más precisa.

- **Support Vector Machine:** El algoritmo Support Vector Machine (SVM) es un algoritmo de machine learning que se basa en crear particiones en el espacio mediante una división en una dimensión superior llamada hiperplano, que separará de manera óptima las muestras de diferentes clases. Este algoritmo se usa principalmente en problemas de clasificación y regresión, y es capaz de modelar relaciones entre datos altamente complejas [32]. Para separar de manera óptima las muestras, el algoritmo busca un hiperplano

que maximice la distancia entre las muestras más cercanas de cada clase, es decir, los vectores de soporte. A esta máxima distancia entre el hiperplano y los vectores de soporte se le llama Maximum Margin Hyperplane (MMH), y al encontrarlo se pretende que el algoritmo se ajuste lo mejor posible a datos futuros. Los vectores de soporte son puntos críticos en la construcción del hiperplano, ya que tienen mayor influencia en la construcción del modelo. Cuando los datos no son linearmente separables, se utiliza el truco del kernel, que consiste en crear una dimensión nueva a través de la cual las muestras de diferentes clases se separarían. Hay distintos tipos de kernels, y en este trabajo nos centraremos en el kernel lineal y el kernel RBF.

- **Decision Trees:** Los árboles de decisión, o Decision Trees, son algoritmos de aprendizaje automático usados para clasificación y regresión, que dividen los datos en subconjuntos basándose en reglas, del mismo modo que un árbol se divide en ramas. Este modelo comienza con el nodo raíz, que contiene todos los datos, divide los datos en nodos secundarios o nodos hijos basándose en la característica que origine la mejor división. Esto se repite recursivamente, dividiendo los datos en subconjuntos más pequeños, hasta alcanzar un criterio de parada. La elección de los puntos de mejor división se basa en la variable más predictiva. Los subconjuntos finales son llamados nodos hoja. Una desventaja de los árboles de decisión es que, cuando el árbol se vuelve muy grande, es decir, cuando se ha subdividido muchas veces, los datos que contienen los nodos hoja podrían ser no homogéneos, y esto llevaría al *overfitting*.

- **Random Forest:** El algoritmo Random Forest construye árboles de decisión a partir de subconjuntos aleatorios de los datos, entrenando cada árbol con cada subconjunto de datos, y combina los resultados de cada árbol para obtener como resultado final el resultado más frecuente.

### 3.4.6. Entrenamiento y evaluación de los modelos

Tras la normalización, se ejecutaron los modelos de aprendizaje automático, cada uno de ellos en un loop *for* que fue cambiando parámetros para después quedarnos con el parámetro que devolviera la mejor evaluación. Se obtuvieron cuatro métricas de evaluación para todos los modelos, que se explican más adelante.

- El algoritmo **KNN** se empleó con el paquete de R *ROCR* versión 1.0-11, y con valores de $k = 1, 3, 5, 7, 9, 11$.

- El algoritmo **ANN** se empleó con el paquete de R *neuralnet* versión 1.44.2 , con una capa oculta con un nodo, y con 2 capas ocultas, con 10 y 5 nodos respectivamente.

- El algoritmo **SVM** se empleó con el paquete de R *kernlab* versión 0.9-32, con kernel lineal y con kernel rbf.

- El algoritmo **Decision tree** se empleó con el paquete de R *c50* versión 0.1.8,con boost 1 y 10. El parámetro boost nos permite utilizar en combinación más de un árbol de decisiones para mejorar el rendimiento del modelo.

- El algoritmo **RF** se empleó con el paquete de R *randomForest* versión 4.7-1.1,con 100, 200 y 500 árboles.

Además de realizar los modelos con sus respectivos paquetes, se utilizó también el paquete *caret* versión 6.0-94, que permite generar modelos a partir de la misma función, train() y también permite ejecutar los modelos con validación cruzada de una forma sencilla, y paralelizar el procesamiento utilizando más de un núcleo del ordenador.

Las métricas de evaluación fueron las siguientes:

- **Exactitud:** Proporción que representa el número de muestras clasificadas de forma correcta

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

  Siendo:

  - TP: Verdaderos positivos. Número de muestras positivas clasificadas correctamente como positivas, es decir, número de muestras que se clasifican como la clase de interés correctamente
  - TN: Verdaderos negativos. Número de muestras negativas clasificadas correctamente como negativas, es decir, número de muestras que se clasifican como que no son la clase de interés, de forma acertada
  - FP: Falsos positivos. Número de muestras negativas clasificadas incorrectamente como positivas
  - FN: Falsos negativos. Número de muestras positivas clasificadas incorrectamente como negativas

- *Valor Kappa:* EL valor de kappa es una corrección de la exactitud, en la que se tiene en cuenta si el predictor ha acertado por azar. En la fórmula, $Pr(a)$ es la proporción de aciertos que se han obtenido y $Pr(e)$ es la concordancia entre el clasificador y los valores verdaderos bajo la asunción de que fueron clasificados al azar.

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \tag{3.3}$$

- *Especifidad:* La especificidad, o el ratio de verdaderos negativos, es la proporción de muestras negativas que fueron clasificadas correctamente

$$specificity = \frac{TN}{TN + FP} \tag{3.4}$$

- *Sensibilidad:* La sensibilidad, o el ratio de verdaderos positivos, es la proporción de muestras positivas que fueron clasificadas correctamente

$$sensitivity = \frac{TP}{TP + FN} \tag{3.5}$$

Estas fueron las métricas utilizadas al escoger el mejor modelo. Sin embargo, la matriz de confusión ofrece más medidas que fueron mostradas en la optimización del modelo y se explican a continuación:

- *PPV:* El *Positive Predictive Value* o valor predictivo positivo, es la proporción de verdaderos positivos frente a las muestras que fueron clasificadas como positivas.
- *NPV:* El *Negative Predictive Value* o valor predictivo negativo, es la proporción de verdaderos negativos frente a las muestras que fueron clasificadas como negativas.
- *Prevalencia:* Proporción de positivos, clasificados correcta o incorrectamente, que hay en nuestra muestra.

$$Prevalencia = \frac{TP + FN}{TP + TN + FP + FN} \tag{3.6}$$

- *Ratio de detección (DR):* O Detection Rate, proporción de verdaderos positivos con respecto al total

$$DR = \frac{TP}{TP + TN + FP + FN} \tag{3.7}$$

- *Exactitud Balanceada:* O Balanced Accuracy, es la media aritmétrica de sensibilidad y especificidad.

$$DR = \frac{sensibilidad + especificidad}{2} \tag{3.8}$$

A pesar de que la matriz de confusión ofrece estas medidas, algunas de ellas mostraron valores nulos al ejecutar los modelos, por lo tanto al escoger el mejor modelo nos quedamos con Exactitud, Valor de kappa, Sensibilidad y Especificidad, y además de observar todas por individual, realizamos una media de las cuatro para mostrar el porcentaje de rendimiento del modelo. Sin embargo, para el modelo final se muestran todas las métricas de evaluación.

### 3.4.7. Mejora de los modelos

Después del análisis de expresión diferencial y de obtener las métricas de evaluación de los modelos con estos datos, se empleó un algoritmo Random Forest para obtener la importancia de las variables restantes y así mejorar los modelos. Este algoritmo nos proporciona un valor llamado *Mean Decrease Accuracy*, que nos indica cómo afecta cada variable al rendimiento del modelo. Cuanto más positivo el valor, indicaría mayor importancia de la variable en el modelo. Se filtraron las variables escogiendo las que mostraban una *Mean Decrease Accuracy* mayor a 0.

También, como otro método de reducción de dimensionalidad, se realizó un Análisis de Componentes Principales (PCA) con prcomp en R, y tras calcular la varianza acumulada, se escogieron las componentes principales que explicaban el 95 % de la varianza.

### 3.4.8. Implementación del modelo en una Aplicación web

Para poder utilizar el modelo en la aplicación web, hubo que crear un script de R con una función que obtiene las predicciones del modelo y contiene los pasos previos para poder utilizarlo. En la aplicación se llama a la función *modelo_SVM*, que tiene cuatro variables: *data* y *data_labels*, que son dos dataframes que contienen, el primero, los datos de expresión con un gen por columna, y el segundo, las etiquetas identificadoras de cada muestra, una etiqueta por fila que tenga el dataframe *data*; *modelo*, el modelo escogido, previamente entrenado; y *genes_modelo*, una lista de las variables escogidas finalmente para el modelo tras el filtrado de datos. La función recibe el modelo y la lista de los genes escogidos, procede a filtrar los datos proporcionados en la aplicación (data) y escoge solo las columnas escogidas tras el filtrado. Se normalizan los datos con las funciones que se utilizaron para normalizar los datos de entrenamiento, y se obtienen las predicciones. La función devuelve un dataframe en forma de tabla con dos columnas, la primera es la etiqueta de la muestra y la segunda es el tipo de tumor que el modelo predijo. Por lo tanto, la función funciona de la siguiente manera:

1. Filtrado de las variables
2. Normalización de los datos utilizando logaritmo en base 2, normalización de los datos usando el máximo y el mínimo de los datos de entrenamiento.
3. Comprobación de valores faltantes.
4. Ejecución del modelo
5. Obtención de las predicciones

La aplicación web se desarrolló utilizando la librería Shiny, versión 1.8.0, en R. La estructura a seguir para desarrollar una aplicación web en Shiny es la siguiente:

- **Interfaz de usuario (UI):** Elemento donde se guarda la apariencia de la aplicación. En ella configuras el título, los elementos que muestra la app, como por ejemplo botones para adjuntar archivos, pestañas donde mostrar distintos elementos, las imágenes que se muestran o las funciones que tendrá la aplicación.
- **Servidor (server):** Contiene el código que hace funcionar la aplicación. En nuestro caso, el server es un elemento que recibe los datos que el usuario adjunta, y llama a la función que generamos en nuestro paquete de R, que realizará las predicciones, devolviendo la tabla con las etiquetas y las predicciones. En el servidor también especificamos la ruta de las imágenes a mostrar en la aplicación, y la ruta de los archivos de ejemplo que se podrán descargar.
- La **función** que ejecuta la aplicación, y que tiene como argumentos la UI y el server.

### 3.4.9. Productos generados

La aplicación web es accesible desde el siguiente enlace
`https://hugosuarez.shinyapps.io/RenalCellPredictor/`,
sin embargo puede no estar en funcionamiento debido a problemas con el límite de memoria del paquete gratuíto de Shiniapps.io. En el siguiente enlace hay un vídeo explicativo de su funcionamiento
`https://drive.google.com/file/d/17dL-MIllm4zvu2ezUUuETA20LmxZ51v_/view?usp=sharing`

El código usado para el proyecto se encuentra en el siguiente enlace
`https://github.com/hugosuarezglez/SVM_RCC/blob/main/TFM_comoAnexo.R`

Y, en este repositorio en general se encuentran el código de la app, el código de el Script con la función usada en la app, y un archivo README que los identifica
`https://github.com/hugosuarezglez/SVM_RCC`

# Capítulo 4

# Resultados

## 4.1. Análisis exploratorio

Tras eliminar los datos con etiquetas ambiguas y comprobar que no haya datos faltantes, nos quedamos con un total de 981 muestras, y 60660 genes. Los datos de expresión génica suelen estar muy dispersos, ya que su rango puede ser muy amplio, por lo tanto se realizó un boxplot con los datos de expresión de todos los genes divididos por su diagnóstico, de esta forma se ve cómo se distribuyen los datos para cada clase:

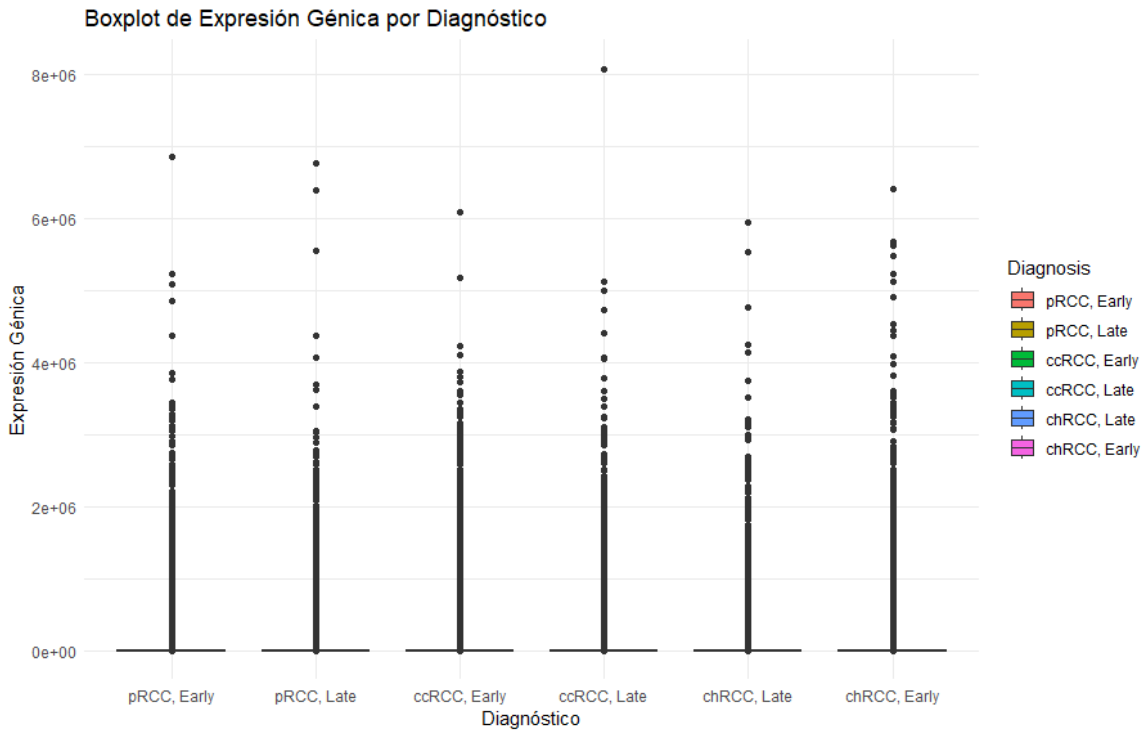



Figura 3: Diagrama de cajas que muestra los datos de expresión de todos los genes según el diagnóstico de las muestras

Como muestra la Figura 3, los datos tienen un rango demasiado amplio, y por lo tanto sería

más efectivo realizar una transformación logarítmica para disminuírlo. Realizamos la transformación utilizando el logaritmo en base 2, y sumando un 1 a cada dato ya que esta transformación no soporta datos que tengan el valor 0. Hacemos un diagrama de cajas nuevamente para ver los datos tras la transformación.

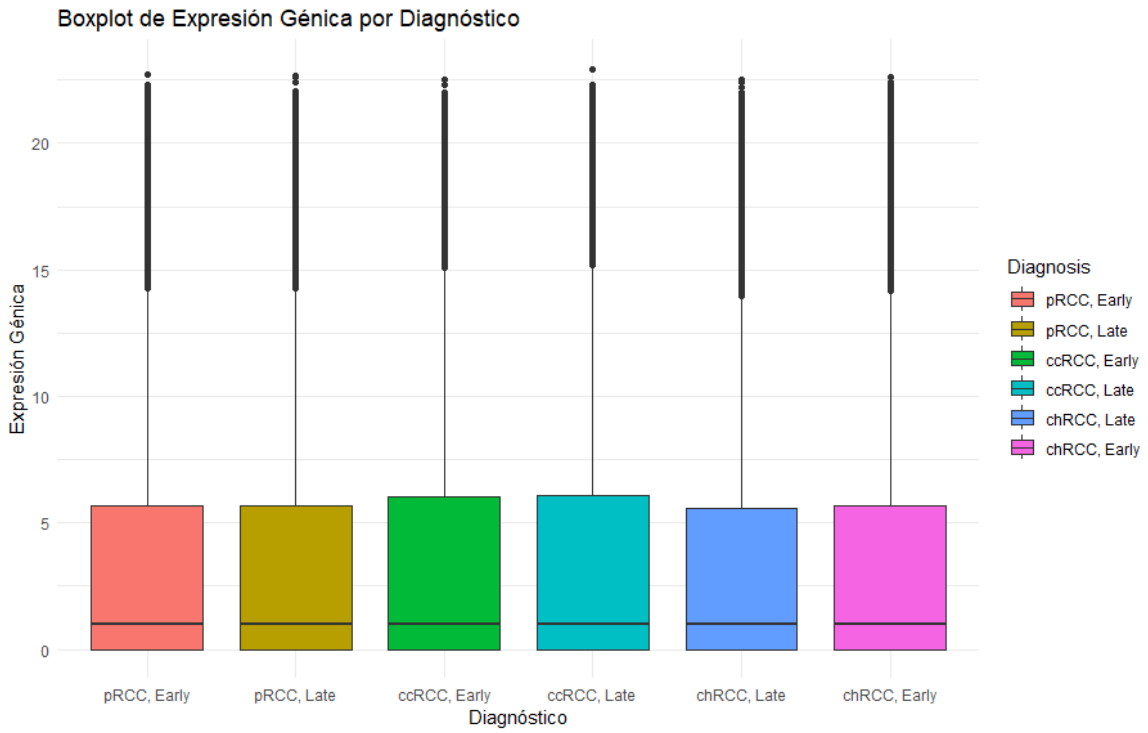



Figura 4: Diagrama de cajas que muestra los datos de expresión de todos los genes según el diagnóstico de las muestras, tras la transformación logarítmica de los datos.

Ahora nuestros datos se muestran más legibles, y en el diagrama de cajas podemos ver que hay valores atípicos, pero en datos de expresión es normal que haya genes con expresiones muy elevadas. Por lo tanto, los datos fueron normalizados, pero después del análisis de expresión diferencial, ya que DESeq2 utiliza datos crudos.

Comprobamos si los datos están balanceados,

En la Tabla 1 podemos ver claramente que los datos no están balanceados. El tipo de tumor ccRCC muestra más datos que los otros dos tipos, y en general hay también más cantidad de datos de estadío temprano que tardío. Este es un problema que se debe solucionar, ya que probablemente afectará al rendimiento de los modelos. Por ello, se procedió a balancear los datos utilizando la técnica SMOTE. Tras esto, obtuvimos muestras generadas artificialmente que harán que tengamos 320 muestras en total para cada clase. Sin embargo, este paso se realizó tras el análisis de expresión, ya que a DESeq2 no le afecta el desbalance de las muestras, y es preferible realizar el análisis de expresión con nuestros datos originales.

| Clase | Número de muestras |
|---|---|
| ccRCC, Early | 320 |
| ccRCC, Late | 205 |
| pRCC, Early | 194 |
| pRCC, Early | 67 |
| chRCC, Early | 46 |
| chRCC, Early | 20 |
| Control | 129 |

Tabla 1: Número de muestras de cada clase

## 4.2. Análisis de expresión y filtrado de genes

Tras el análisis de expresión con DESeq2 y posterior filtrado de los resultados, se obtuvo una lista de genes diferencialmente expresados entre las muestras etiquetadas como tempranas y las muestras etiquetadas como tardías, lo que redujo nuestra lista de 60660 genes a 15497 genes.

Tras esto, ya se procedió a balancear los datos. Se probó el rendimiento de los modelos con estos datos. Para mejorar los modelos, se procedió a filtrar los genes más profundamente, elaborando un análisis de importancia de las variables con randomForest, lo cual nos dio un total de 15233 genes.

## 4.3. Modelos

A continuación se muestra una tabla con el rendimiento de los modelos, calculado con las métricas de evaluación Exactitud, Valor de kappa, Especificidad y Sensibilidad. Se mostrarán las métricas obtenidas con cada paquete correspondiente, explicado en el apartado métodos. Los algoritmos utilizados fueron KNN, ANN, SVM, DT, RF y NB, cada uno realizado más de una vez, variando los parámetros indicados en la tabla, para observar si al cambiar los parámetros había mejoras.

Como podemos ver en la Tabla 2, la exactitud de los modelos ha estado por encima de 0.5 para todos los modelos excepto las ANN y Naive Bayes, mostrando un rango desde 0.34 en el modelo ANN con 8 capas ocultas hasta 0.82 en SVM con kernel lineal. El estadístico kappa es ligeramente más bajo para todos los modelos, siendo el modelo con mayor valor de kappa el SVM con kernel lineal. La sensibilidad muestra un rango parecido a la Exactitud, entre 0.34 para la ANN y 0.82 para SVM lineal. La especificidad, sin embargo, es elevada; muestra valores por encima de 0.89 para todos los modelos, siendo el SVM el modelo con el valor máximo, de 0.97. La media hecha con todas las métricas de evaluación muestra un rango entre 45 % y 85 %, aproximadamente. El modelo que mejor rendimiento ha obtenido es el SVM con kernel lineal, con un 85 % de rendimiento de media entre las 4 métricas de evaluación y un 82.14 % en la métrica Exactitud.

| Modelo | Parámetro | Exactitud | Valor de kappa | Sensibilidad | Especificidad | Media (%) |
|---|---|---|---|---|---|---|
| KNN | k=1 | 0,7767857 | 0,7393212 | 0,7724919 | 0,9627977 | 81,2849125 |
| | k=3 | 0,6785714 | 0,6243843 | 0,6735504 | 0,946333 | 73,0709775 |
| | k=5 | 0,6584821 | 0,6006898 | 0,6539639 | 0,9428781 | 71,4003475 |
| | k=7 | 0,6160714 | 0,5511284 | 0,6111836 | 0,9358243 | 67,8551925 |
| | k=11 | 0,5892857 | 0,5199683 | 0,5856971 | 0,9313755 | 65,658165 |
| ANN | 8 capas | 0,3392857 | 0,2308522 | 0,3351592 | 0,8903796 | 44,8919175 |
| | 10, 5 capas | 0,3570982 | 0,25149975 | 0,3506183 | 0,90217985 | 46,5349025 |
| NB | Laplace = 0 | 0,3549107 | 0,2521473 | 0,3460774 | 0,8939801 | 46,1778875 |
| | Laplace = 0,1 | 0,3549107 | 0,2521473 | 0,3460774 | 0,8939801 | 46,1778875 |
| SVM | kernel lineal | 0,8214286 | 0,7915649 | 0,8179783 | 0,9703003 | 85,0318025 |
| | kernel rbf | 0,7165179 | 0,6690226 | 0,7136409 | 0,9527478 | 76,29823 |
| DT | boost=1 | 0,6116071 | 0,5462367 | 0,6063824 | 0,9354271 | 67,4913325 |
| | boost=10 | 0,765625 | 0,7261853 | 0,7661226 | 0,9609681 | 80,472525 |
| RF | ntrees=100 | 0,7924107 | 0,7575236 | 0,7929415 | 0,9654618 | 82,70844 |
| | ntrees=200 | 0,7879464 | 0,7521936 | 0,7877155 | 0,9646631 | 82,312965 |
| | ntrees=500 | 0,7834821 | 0,7470238 | 0,7834821 | 0,9639413 | 81,9482325 |

Tabla 2: Rendimiento de los modelos ejecutados con los datos obtenidos tras el análisis de expresión

Para comprobar si había mejoras, se realizaron los mismos modelos utilizando el paquete *caret*, pero el rendimiento resultó ser ligeramente más bajo para todos los modelos, por lo tanto se utilizarán las otras librerías. En la siguiente tabla vemos un ejemplo de la diferencia entre los rendimientos:

| Modelo | Librería | Parámetro | Exactitud | Valor de kappa | Sensibilidad | Especificidad | Media (%) |
|---|---|---|---|---|---|---|---|
| KNN | caret | k = 5 | 0,652 | 0,593 | 0,652 | 0,942 | 65,179 |
| KNN | ROCR | k = 5 | 0,658 | 0,601 | 0,654 | 0,943 | 71,400 |
| SVM | caret | Lineal | 0,799 | 0,765 | 0,800 | 0,967 | 79,911 |
| SVM | kernlab | Lineal | 0,821 | 0,792 | 0,818 | 0,970 | 85,032 |
| SVM | caret | Radial | 0,725 | 0,679 | 0,727 | 0,954 | 72,545 |
| SVM | kernlab | Radial | 0,717 | 0,669 | 0,714 | 0,953 | 76,298 |
| DT | caret | | 0,304 | 0,183 | 0,300 | 0,883 | 30,357 |
| DT | c50 | | 0,766 | 0,726 | 0,766 | 0,961 | 80,473 |

Tabla 3: Comparación del rendimiento con distintas librerías

## 4.4. Optimización de los modelos

Para ver si se podía mejorar el rendimiento de los modelos, se hizo una selección de las variables más importantes, midiendo la importancia con el algoritmo Random Forest. En la siguiente figura podemos ver una pequeña muestra de los 20 genes que mostraron la mayor importancia.

Aunque en la figura se muestren los 20 genes más importantes, la importancia de todas

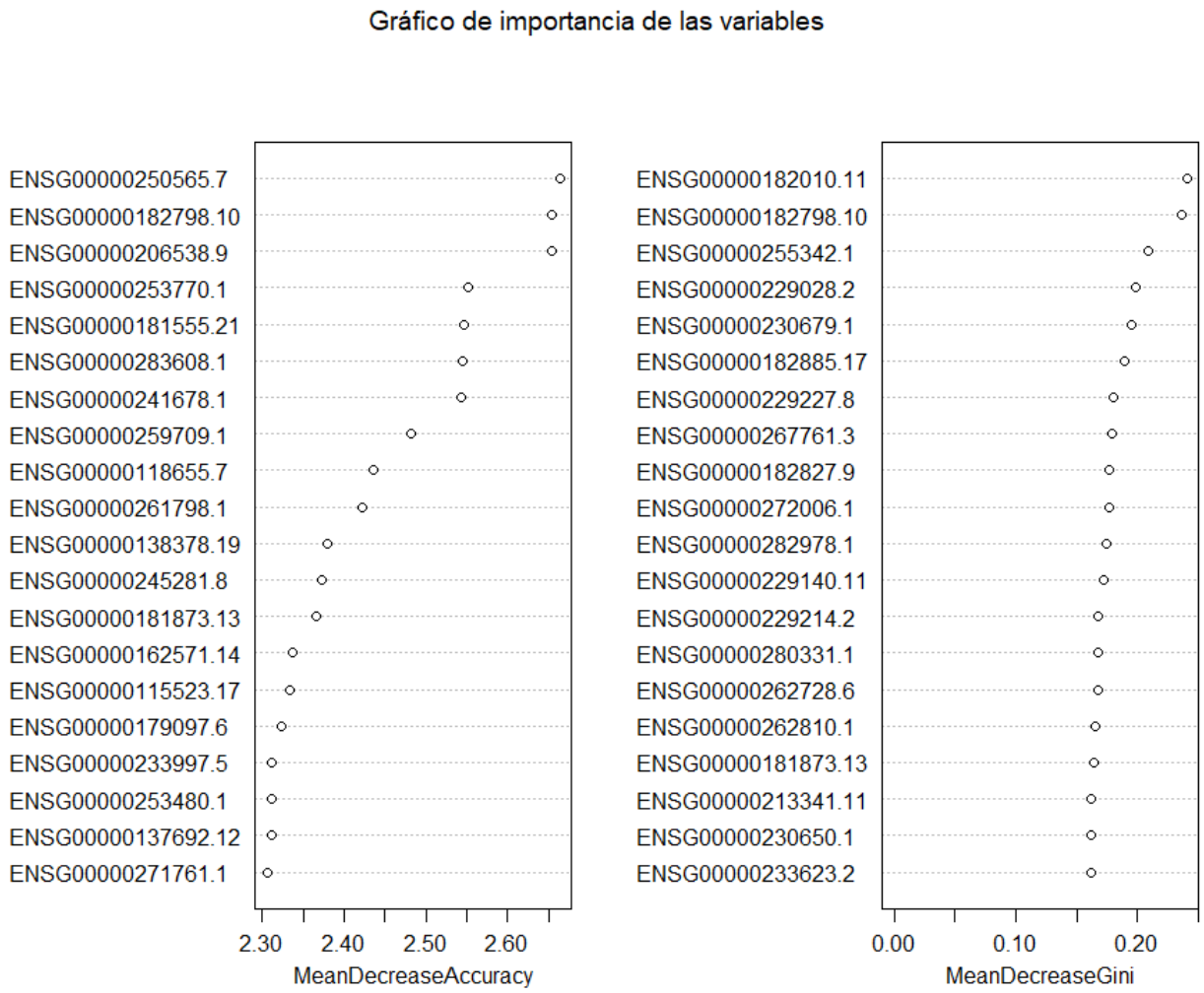



Figura 5: Top 20 genes más importantes en el modelo

nuestras variables tenía un rango que incluía números negativos, por lo tanto se procedió a eliminar todas las variables que tenían una importancia menor o igual a cero, y que por lo tanto influían o negativamente o de forma nula al modelo. Así, nos quedamos sólo con las variables con un efecto positivo, que resultaron ser 15233 genes.

A continuación se muestra otra tabla con el rendimiento de los modelos, utilizando los genes seleccionados tras el análisis de importancia En esta tabla, a pesar de haber variado parámetros dentro de cada modelo, se muestra ya el modelo con el parámetro que mejor rendimiento obtuvo.

Como podemos ver en la Tabla 4, el análisis de importancia ha mejorado ligeramente el rendimiento del modelo KNN, pero ha empeorado el rendimiento de los modelos SVM, DT, RF y NB. Como el rendimiento más alto sigue siendo el obtenido con SVM en el paso anterior, fue este el modelo escogido.

| Modelo | Parámetro | Exactitud | Valor de kappa | Sensibilidad | Especificidad | Media (%) |
|---|---|---|---|---|---|---|
| KNN | k=1 | 0,7857143 | 0,7496813 | 0,785568 | 0,9643536 | 82,13293 |
| SVM | lineal | 0,796875 | 0,7626953 | 0,7978934 | 0,9661665 | 79,6875 |
| DT | boost=10 | 0,7611607 | 0,721012 | 0,7618734 | 0,9602419 | 76,11607 |
| RF | n = 500 | 0,7901786 | 0,7548564 | 0,7900458 | 0,9650683 | 82,5037275 |
| NB | Laplace=0 | 0,3549107 | 0,2521473 | 0,3460774 | 0,8939801 | 35,49107 |

Tabla 4: Rendimiento de los modelos tras el filtrado de los datos según la importancia de las variables

Posteriormente, se hizo un Análisis de Componentes Principales, PC, para reducir la dimensionalidad de los datos, y tras calcular la varianza acumulada, se escogieron las componentes que explicaran el 95 % de la varianza, que fueron 212. En la siguiente Figura podemos ver un gráfico con las primeras 10 PC y el porcentaje de varianza explicado por cada una de ellas.

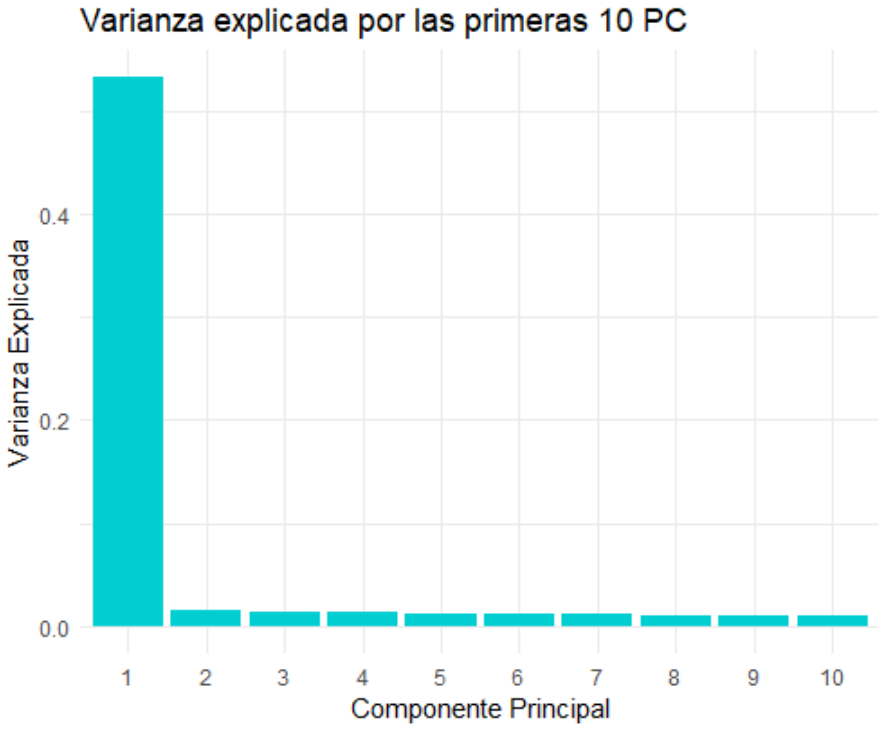



Figura 6: Top 20 genes más importantes en el modelo

Se transformaron los datos de prueba y de entrenamiento según el modelo generado por el PCA, y se escogieron las primeras 212 PC, y estos fueron los resultados.

Como se puede comprobar, la reducción de la dimensionalidad ha empeorado el rendimiento del modelo. Tampoco el uso de validación cruzada y diferentes parámetros para el modelo mejoró el rendimiento. No ha habido ningún modelo mejor que el actual, SVM con kernel lineal

| Modelo | Exactitud | Valor de kappa | Sensibilidad | Especificidad | Media (%) |
|---|---|---|---|---|---|
| SVM, PCA | 0,7723214 | 0,7340364 | 0,7729550 | 0,9620992 | 77,23214 |

Tabla 5: Rendimiento del modelo SVM tras hacer una reducción de la dimensionalidad con PCA

y $C = 0,01$, que presenta un 85 % de rendimiento de media, y un 82,14 % de Exactitud. Para simplificar, a este modelo se le llama $SVM_1$.

A continuación se muestra una tabla con todas las medidas de evaluación por clase, disponibles en la información que proporciona la función *confusionMatrix* de *caret*.

Como última opción, y ya que podemos ver que las clases que peor rendimiento muestran son las de ccRCC, que eran las mayoritarias, se probó a sobremuestrear los datos otra vez, utilizando SMOTE, pero en lugar de no sobremuestrear la clase mayoritaria, se utilizó un número máximo de muestras de 400, lo que significa que la clase *ccRCC, Early* tendrá 80 muestras generadas artificialmente en vez de ninguna. Esto podría aumentar el rendimiento general del modelo, pero también podría generar ruido, y significa un riesgo para el modelo porque puede provocar *overfitting*, es decir, que el modelo esté entrenado de forma muy específica para nuestros datos y no funcione bien ante datos nuevos. Para simplificar, a este modelo se le llama $SVM_2$.

Efectivamente, tras el sobremuestreo de los datos, ha aumentado el rendimiento del modelo $SVM_2$, mostrando una media de 90,36 % entre las cuatro medidas de evaluación, y una Exactitud de 90,36 %, superando así el rendimiento objetivo de este trabajo. En la Tabla 6 podemos ver las métricas de evaluación para dos modelos SVM, uno con kernel lineal y otro con kernel radial, donde se puede ver que el rendimiento ha aumentado en más de un 5 % para el kernel lineal y en aproximadamente un 9 % para el kernel radial. En la Tabla 8 observamos las medidas de evaluación por clase, que nos son proporcionadas por la matriz de confusión. Al comparar las medidas con el modelo antes de hacer el sobremuestreo, se ve una mejora en todas ellas. La clase chRCC Late muestra un rendimiento perfecto, clasificando de forma correcta todas sus muestras. Ésta era la clase minoritaria y para la que se generaron más muestras aleatorias; en este último sobremuestreo se generaron 380 de 400, ya que la clase sólo tenía 20 muestras. Las siguientes clases minoritarias son chRCC Early y pRCC Late, las cuales tienen unas medidas muy altas de todas las métricas. La clase minoritaria, ccRCC Early, es la que peores valores muestra. Se concluye que el sobremuestreo afecta positivamente al rendimiento.

| Modelo | Exactitud | Valor de kappa | Sensibilidad | Especificidad | Media (%) |
|---|---|---|---|---|---|
| SVM, lineal | 0,9035714 | 0,8873109 | 0,8985325 | 0,9840689 | 91,838 |
| SVM, radial | 0,8535714 | 0,8290267 | 0,8503876 | 0,9758208 | 87,720 |

Tabla 8: Rendimiento del modelo $SVM_2$ tras hacer un sobremuestreo mayor de los datos

| | ccRCC, Early | ccRCC, Late | chRCC, Early | chRCC, Late C | Control | pRCC, Early | pRCC, Late | Media (%) |
|---|---|---|---|---|---|---|---|---|
| Sensibilidad | 0,50794 | 0,67188 | 1 | 1 | 0,9516 | 0,7794 | 1 | 84,4402857 |
| Especificidad | 0,93247 | 0,96615 | 0,9949 | 0,9973 | 0,9767 | 0,9763 | 0,9743 | 97,4017143 |
| PPV | 0,55172 | 0,76786 | 0,9649 | 0,9872 | 0,8676 | 0,8548 | 0,8551 | 83,5597143 |
| NPV | 0,92051 | 0,94643 | 1 | 1 | 0,9921 | 0,9611 | 1 | 97,4305714 |
| Prevalencia | 0,14062 | 0,14286 | 0,1228 | 0,1719 | 0,1384 | 0,1518 | 0,1317 | 14,2868571 |
| DR | 0,07143 | 0,09598 | 0,1228 | 0,1719 | 0,1317 | 0,1183 | 0,1317 | 12,0544286 |
| Exactitud Balanceada | 0,7202 | 0,81901 | 0,9975 | 0,9987 | 0,9641 | 0,8779 | 0,9871 | 90,9215714 |

Tabla 6: Métricas de evaluación por clase del modelo $SVM_1$

| | ccRCC, Early | ccRCC, Late | chRCC, Early | chRCC, Late C | Control | pRCC, Early | pRCC, Late | Media (%) |
|---|---|---|---|---|---|---|---|---|
| Sensibilidad | 0,72727 | 0,8272 | 1 | 1 | 0,9286 | 0,8214 | 0,9853 | 89,8538571 |
| Especificidad | 0,95547 | 0,9854 | 0,9915 | 1 | 0,9874 | 0,9748 | 0,9939 | 98,4067143 |
| PPV | 0,68571 | 0,9054 | 0,956 | 1 | 0,9286 | 0,8519 | 0,9571 | 89,7815714 |
| NPV | 0,96327 | 0,9712 | 1 | 1 | 0,9874 | 0,9687 | 0,998 | 98,4081429 |
| Prevalencia | 0,11786 | 0,1446 | 0,1554 | 0,1607 | 0,15 | 0,15 | 0,1214 | 14,2851429 |
| DR | 0,08571 | 0,1196 | 0,1554 | 0,1607 | 0,1393 | 0,1232 | 0,1196 | 12,9072857 |
| Exactitud Balanceada | 0,84137 | 0,9063 | 0,9958 | 1 | 0,958 | 0,8981 | 0,9896 | 94,131 |

Tabla 7: Métricas de evaluación por clase del modelo $SVM_2$ tras hacer un sobremuestreo de los datos

Hacer una comparación entre la Tabla 7 y la Tabla 8 ayuda a tomar la decisión de qué modelo escoger para la aplicación web. En ambas tablas vemos valores bajos de Prevalencia y Ratio de detección, lo cual es razonable porque son la proporción de positivos y la proporción de muestras clasificadas como positivas, respectivamente. Cuanta menor sea la diferencia entre esas dos medidas, más predictivo es nuestro modelo, pues los casos positivos están siendo clasificados como positivos. Vemos una mejora general en todas las métricas de evaluación para el segundo modelo.

Desgraciadamente, en el último día antes de acabar el plazo de entrega, llegué a la conclusión de que había balanceado el conjunto entero de datos y no el conjunto de entrenamiento, y eso era lo que había elevado tanto el rendimiento de los modelos al generar muestras artificialmente. Sin tiempo para corregir mi error, el modelo escogido fue el que contiene menos datos generados, ya que al menos tenemos medidas fiables para la clase mayoritaria, ya que no se infiltraron datos de entrenamiento de esa clase en el conjunto de prueba. Este trabajo ha sido un reto desde el comienzo; se probaron múltiples formas de elevar el rendimiento y no dieron resultado, hubo que aprender a utilizar estos recursos desde cero y lidiar con problemas de falta de poder de computación ya que el conjunto de datos era muy grande. En la primera etapa ni siquiera se podían ejecutar los modelos ya que el ordenador no tenía suficiente memoria RAM. Ese fue el motivo de reducir la dimensionalidad de los datos haciendo el análisis de expresión de los genes. Tras él, se probaron otros métodos de reducción de dimensionalidad, y no sólo no aumentaban el rendimiento de los modelos, sino que lo disminuían. La falta de tiempo imposibilitó rehacer el proyecto con los datos bien balanceados, por lo tanto se escogió el modelo $SVM_1$.

Para poder ver unas métricas más realistas de este modelo, se procedió a retirar del conjunto de evaluación todas las entradas que fueran generadas artificialmente, lo cual se podía saber ya que el nombre de fila de las originales comienza con *TCGA-*, mientras que las generadas tienen como nombre de fila un número. De esta forma, nos quedamos con las siguientes muestras:

| Clase | Número de muestras |
|---|---|
| ccRCC, Early | 63 |
| ccRCC, Late | 42 |
| pRCC, Early | 38 |
| pRCC, Early | 10 |
| chRCC, Early | 6 |
| chRCC, Early | 4 |
| Control | 23 |

Tabla 9: Número de muestras de cada clase en el conjunto de evaluación

Claramente, la falta de equilibrio en el número de muestras por clase en la Tabla 9 no es ideal. Sin embargo, debido a restricciones de tiempo, no fue posible realizar nuevamente todo el trabajo, por lo que se optó por mantener esta medida y obtener las predicciones con este conjunto de evaluación. Al utilizar estos datos, que excluyen la inclusión de datos generados

artificialmente, se procedió a realizar nuevas predicciones con el modelo $SVM_1$, dando como resultado las métricas de evaluación que se presentan en la Tabla **??**.

Podemos ver que los resultados de rendimiento son bastante altos, con una Exactitud del 93,55 %, una Sensibilidad del 96,53 % y una Especificidad del 98,8 % de media entre las clases, y una media de todas las métricas de evaluación exceptuando DT y Prevalencia del 93 % para ccRCC Early, 96 % para ccRCC Late, 100 % y 99,73 % para chRCC Early y Late, 97 % para las muestras control y 96,4 y 96,8, % para pRCC Early y Late. Como ya se ha mencionado, lo correcto sería tener un conjunto de datos de evaluación mayor, con el mismo número de datos para cada clase, pero como se disponía de un tiempo limitado para obtener estos últimos resultados, estas serán las métricas de evaluación que se usarán para el modelo. En la Figura 7 podemos ver las medidas de evaluación de nuestro modelo final, el $SVM_1$ con kernel lineal y valor de $c = 0{,}01$.

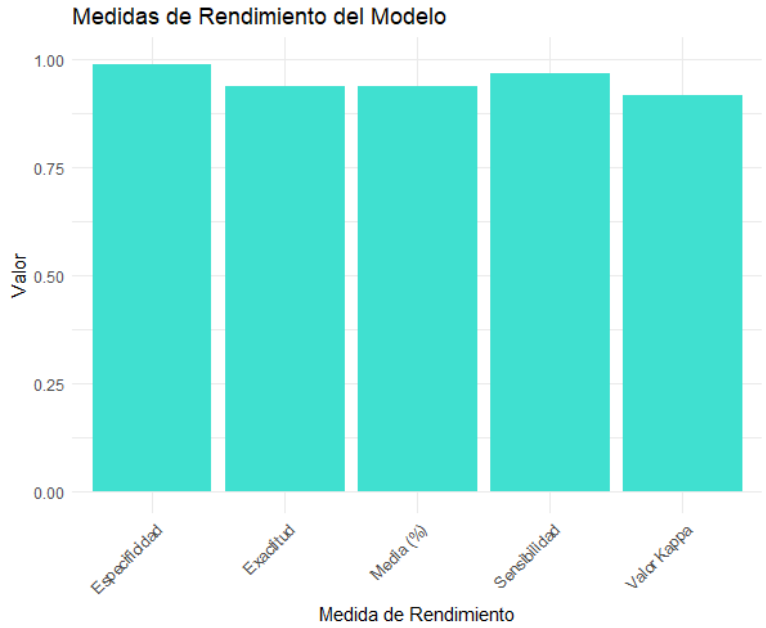



Figura 7: Diagrama de barras que muestra las medidas de evaluación del rendimiento del modelo

## 4.5. Aplicación web

Se implementó el modelo a una aplicación web creada con Shiny en R. En las siguientes imágenes podemos ver un ejemplo de su funcionamiento. La aplicación está disponible a través del enlace
`https://hugosuarez.shinyapps.io/RenalCellPredictor/`
y se puede ver un vídeo mostrando su funcionamiento en
`https://drive.google.com/file/d/17dL-MIllm4zvu2ezUUuETA20LmxZ51v_/view?usp=sharing`.

La Figura 8 muestra la pantalla principal de la aplicación. Podemos ver que tiene un panel a la izquierda en el que se introducen los datos, en formato CSV, por un lado los datos de

| | Sensitivity | Specificity | Pos Pred Value | Neg Pred Value | Precision | Prevalence | Detection Rate | Balanced Accuracy | Media (sin DT y Prevalencia) |
|---|---|---|---|---|---|---|---|---|---|
| ccRCC, Early | 0,9047619 | 0,9593496 | 0,9193548 | 0,9516129 | 0,9193548 | 0,3387097 | 0,3064516 | 0,9320557 | 0,9310816 |
| ccRCC, Late | 0,9047619 | 0,9930556 | 0,9743590 | 0,9727891 | 0,9743590 | 0,2258065 | 0,2043011 | 0,9489087 | 0,9613722 |
| chRCC, Early | 1,0000000 | 1,0000000 | 1,0000000 | 1,0000000 | 1,0000000 | 0,0322581 | 0,0322581 | 1,0000000 | 1,0000000 |
| chRCC, Late | 1,0000000 | 0,9945055 | 0,8000000 | 1,0000000 | 0,8000000 | 0,0215054 | 0,0215054 | 0,9972527 | 0,9319597 |
| Control | 1,0000000 | 0,9877301 | 0,9200000 | 1,0000000 | 0,9200000 | 0,1236559 | 0,1236559 | 0,9938650 | 0,9702658 |
| pRCC, Early | 0,9473684 | 0,9864865 | 0,9473684 | 0,9864865 | 0,9473684 | 0,2043011 | 0,1935484 | 0,9669275 | 0,9636676 |
| pRCC, Late | 1,0000000 | 0,9943182 | 0,9090909 | 1,0000000 | 0,9090909 | 0,0537634 | 0,0537634 | 0,9971591 | 0,9682765 |
| Media | 0,9652703 | 0,9879208 | 0,9243104 | 0,9872698 | 0,9243104 | 0,1428571 | 0,1336406 | 0,9765955 | 0,9609462 |

| Modelo | Exactitud | Valor de kappa | Sensibilidad | Especificidad | Media (%) |
|---|---|---|---|---|---|
| SVM, lineal | 0,9354839 | 0,9169241 | 0,9652703 | 0,9879208 | 94,82166 |

Tabla 10: Rendimiento del modelo $SVM_1$ utilizando los datos de evaluación sin datos generados artificialmente

expresión y por otro lado las etiquetas. Al pulsar el botón Ejecutar predicción, saldrán los resultados en la pestaña Resultado.

En la Figura 9 podemos ver una muestra de los resultados, tras subir un CSV creado a partir de 10 muestras del conjunto de evaluación. En la Figura 10 se muestra la pestaña How to Use, en la cual hay dos imágenes ejemplo de cómo deberían ser los datos, tanto de expresión como de etiquetas, y la opción de descargar un CSV de 10 muestras.

Por último, en la Figura 11 vemos la pestaña Rendimiento, en la cual se muestra un gráfico con las medidas del rendimiento del modelo que se han usado a lo largo del trabajo: Exactitud, Valor de Kappa, Sensibilidad y Especificidad, y la media de las cuatro.

Se ha conseguido implementar el modelo en la aplicación web y que funcione de manera correcta, recibiendo dos archivos CSV y realizando predicciones a partir de los datos de estos archivos. Sin embargo, al publicar la aplicación en Shinyapps.io nos encontramos con un problema con el límite de memoria establecido para el paquete gratis de la web, por lo tanto la aplicación no podrá ejecutar las predicciones. Para un ejemplo de cómo las obtiene, se puede consultar el vídeo indicado anteriormente.

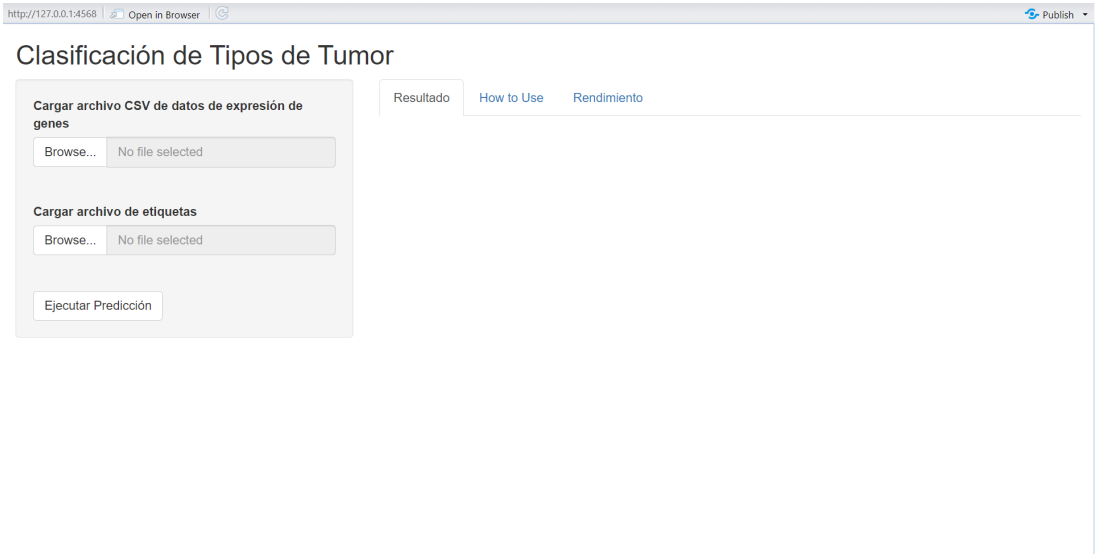



Figura 8: Pantalla principal de la aplicación. Muestra el panel lateral y la pestaña Rendimiento vacía

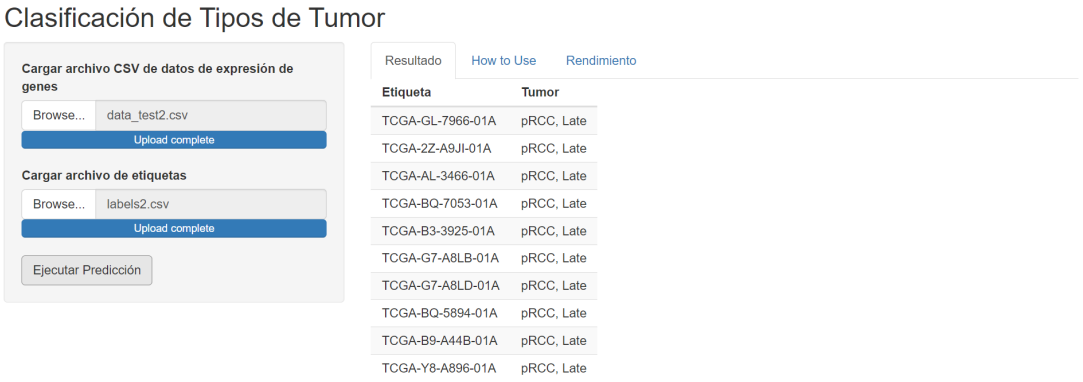



Figura 9: Pantalla principal de la aplicación tras haber ejecutado el modelo para obtener predicciones

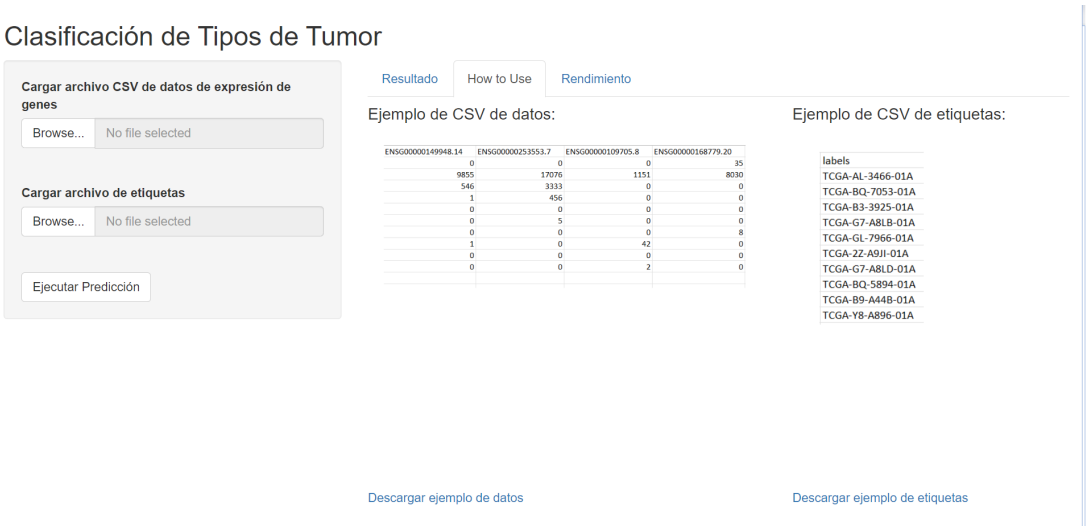



Figura 10: Pestaña How to Use, que muestra un ejemplo de cómo deben de ser los datos a introducir

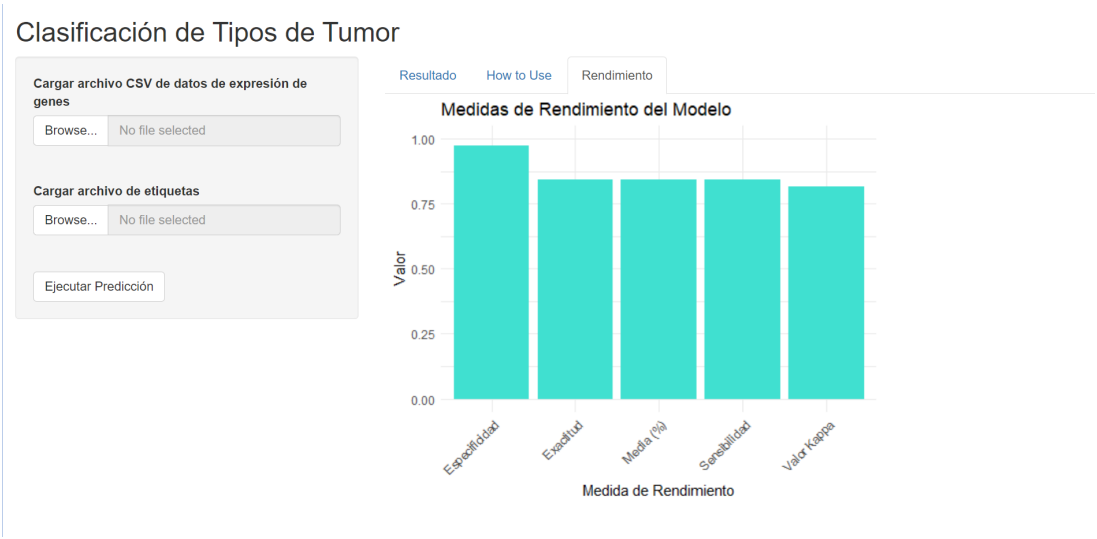



Figura 11: Pestaña Rendimiento, que contiene una gráfica con las métricas de evaluación del modelo

# Capítulo 5

# Discusión

En la Tabla 2 podemos ver que tanto los modelos ANN como los NB no produjeron buenos resultados, ya que muestran los cuatro una exactitud y una sensibilidad menores al 36 %, lo cual indica que el número de muestras clasificadas de forma correcta fue muy bajo. En la mayor parte de los modelos podemos ver valores de Especificidad altos, lo cual significa que las muestras negativas fueron en gran parte clasificadas correctamente por los modelos, y que la tasa de falsos positivos es baja en general; las muestras clasificadas como positivas serán, generalmente, positivas. Sin embargo, una Sensibilidad baja indica una baja proporción de muestras positivas clasificadas. Eso indica que, las muestras clasificadas como positivas fueron positivas en gran parte, pero pocas fueron detectadas.

El resto de modelos muestran un rendimiento mejor, mostrando todos una Exactitud por encima del 80 %. Los dos mejores modelos fueron el SVM con kernel lineal y el Random Forest con 100 árboles. En general, los modelos generados con la librería *caret* muestran rendimientos más bajos, a pesar de haber utilizado validación cruzada $k = 5$. El modelo que mejor rendimiento obtuvo en este caso también fue el SVM con kernel lineal.

Los datos más elevados de rendimiento se obtuvieron tras el análisis de expresión.En la Tabla 7 podemos ver valores de más métricas de evaluación por clase para el modelo escogido. La diferencia entre la Prevalencia y el Ratio de Detección nos da el porcentaje de falsos negativos, es decir, muestras positivas clasificadas como negativas. Vemos que la media de FN para todas las clases es de un 2,23 %, y depende mucho de la clase. Ninguna de las muestras de chRCC ni las muestras tardías de pRCC mostraron falsos negativos, lo cual indica que fueron clasificadas correctamente en todos los casos. Sin embargo, se observa que la clase ccRCC Early, que es la clase mayoritaria, obtiene las peores métricas de evaluación. Por ese motivo, se concluyó que ésto se debía a que, al ser la clase mayoritaria, en el proceso de balanceo de datos no se generaron datos artificiales sobre esta clase, por lo tanto se procedió a hacer un sobremuestreo de todas las clases, incluída la mayoritaria, lo cual elevó el rendimiento de modelo por encima del 90 % tanto en Exactitud como en la media de las cuatro métricas de evaluación. Las podemos ver en la Tabla 8. También vemos que el número de FN aumentó muy ligeramente para la clase pRCC Late, y se mantuvo en 0 % para ambos estados de chRCC. Para chRCC Late, la clase minoritaria, vemos que las métricas son perfectas, obteniendo un 100 % en todo, lo cual concuerda con el razonamiento de que las clases más sobremuestreadas han obtenido mejores

resultados.

Este resultado llevó a la conclusión de que se gestionaron los datos de entrenamiento y evaluación de forma incorrecta, balanceando los datos antes de crear la partición entrenamiento-evaluación, y utilizando datos generados artificialmente en el conjunto de evaluación, lo que provoca una infiltración de datos de entrenamiento en este conjunto. Este error se intentó mitigar excluyendo del conjunto de evaluación los datos generados artificialmente. La decisión más adecuada hubiera sido rehacer el trabajo utilizando unos datos obtenidos de forma correcta, pero debido a una falta de tiempo, hubo que continuar de esta forma. Las medidas de evaluación tras realizar este cambio resultaron tener valores bastante elevados, pero esto no convierte nuestro modelo en un modelo fiable, y debería volver a entrenarse y evaluarse con unos datos bien gestionados.

# Capítulo 6

# Conclusiones

En este proyecto se ha abordado la tarea de clasificar muestras de pacientes con tumores renales a partir de datos de expresión génica, explorando diversas técnicas de aprendizaje automático. A pesar de los retos encontrados, se presenta el modelo Support Vector Machine (SVM) con kernel lineal como el más destacado, alcanzando una exactitud del 93,55 % y un rendimiento medio del 94,82 %. El resultado cumple con la meta inicial del superar 90,13 % de exactitud, pero no supera el rendimiento para el modelo existente sobre la clasificación de estadío de pRCC ni el rendimiento del modelo que clasifica subtipos de RCC.

Este trabajo muestra la viabilidad de clasificar el carcinoma de células renales (RCC) y su diagnóstico mediante modelos de clasificación generados a partir de datos de expresión génica obtenidos en una base de datos pública. Se ha logrado desarrollar un modelo de aprendizaje automático para clasificar pacientes con tumores renales y el estado del tumor, ahorrando un paso al proporcionar ambos datos en una misma clasificación. La implementación del modelo SVM en una aplicación web amplía su accesibilidad a profesionales de la salud y al público en general, proporcionando una herramienta valiosa para el diagnóstico de enfermedades renales.

En relación con los objetivos planteados, se lograron satisfactoriamente tanto los principales como los secundarios. La evaluación crítica del logro de objetivos revela tanto éxitos como desafíos. Aunque se han alcanzado los objetivos principales, como la identificación de genes relevantes y el desarrollo de la aplicación web, el fracaso en el intento de reducción de dimensionalidad destaca la dificultad en la tarea. Por otro lado, se considera que los resultados podrían estar sesgados, debido a una mala gestión de los datos de entrenamiento y validación. La adaptación de metas y la introducción de cambios, como el análisis de expresión diferencial y el uso de SMOTE para abordar desbalances en las clases, resaltan la flexibilidad y adaptabilidad necesarias para abordar obstáculos inesperados, y a pesar de haber cometido errores al ejecutar estas técnicas, se ha intentado en todo momento mantenerse en un razonamiento coherente y mitigar en la medida de lo posible las consecuencias de los errores cometidos.

El seguimiento de la planificación y metodología revela desviaciones muy significativas. Factores como la magnitud del conjunto de datos y la necesidad de aprender nuevas técnicas, como el análisis de expresión diferencial, la reducción de la dimensionalidad mediante el análisis de importancia o el análisis de componentes principales, impactaron gravemente el cumplimiento

de la planificación. La introducción de cambios, como el análisis de expresión y el sobremuestreo de los datos para evitar desbalance de clases, y sobre todo el nuevo sobreajuste de los datos de evaluación, fue esencial para garantizar el éxito del trabajo. Estas modificaciones ilustran la importancia de la flexibilidad y la adaptación en la investigación. Este trabajo ha proporcionado lecciones valiosas en términos de gestión de datos transcriptómicos, ajuste de modelos y adaptabilidad metodológica. La importancia de la exploración continua y la capacidad de respuesta a desafíos inesperados son aspectos cruciales que se han destacado durante el desarrollo del proyecto.

### 6.0.1. Líneas de futuro

**Re-entrenamiento y evaluación del modelo**

Realizar de nuevo el entrenamiento y evaluación del modelo, utilzando datos que sean balanceados después de la partición entrenamiento y evaluación, para obtener unos valores más realistas del rendimiento.

**Selección de variables más completa** Realizar una selección de variables más exhaustiva, incorporando la significancia biológica de los genes y la información disponible en la literatura científica. Esta aproximación permitirá identificar genes más directamente relacionados con el cáncer de células renales, contribuyendo a una mejor comprensión de la base molecular de la enfermedad.

**Entrenamiento con datos adicionales** Ampliar el entrenamiento de los modelos utilizando conjuntos de datos adicionales provenientes de diversas fuentes. Esto permitirá tener un conjunto de datos más grande, verificar y validar el rendimiento del modelo en diferentes contextos, lo que aumentaría la robustez del modelo. Existen diferentes programas de los que podríamos obtener datos de expresión génica de carcinoma de células renales, para complementar nuestro set de datos.

**Aprendizaje Multimodal** Explorar el uso de algoritmos de aprendizaje multimodal, integrando información de imágenes citopatológicas o datos de miRNA. Esta estrategia podría enriquecer el modelo al incorporar múltiples modalidades de datos, brindando una visión más completa y precisa de las características moleculares o histológicas asociadas al cáncer de células renales.

**Inclusión de variables demográficas** Considerar la inclusión de variables demográficas, como el sexo o la etnia, en el modelo. Dado que existen diferencias genéticas entre distintas etnias y sexos, incorporar estas variables podría mejorar la capacidad del modelo para capturar la variabilidad genética y sus implicaciones en la predicción del cáncer renal.

**Mejora del diseño de la aplicación** Refinar el diseño de la aplicación web, incluyendo elementos visuales explicativos, como gráficos que muestren el porcentaje de muestras clasificadas para cada clase y el rendimiento específico del modelo para cada categoría. Esta mejora no solo

hará la aplicación más informativa, sino que también proporcionará una comprensión detallada del rendimiento del modelo en distintos contextos clínicos.

**Validación externa del modelo** Realizar una validación externa del modelo utilizando conjuntos de datos independientes recopilados en diferentes entornos clínicos. Esta validación externa proporcionará una evaluación más sólida del rendimiento del modelo en diferentes poblaciones y contextos, aumentando la confianza en su aplicabilidad clínica.

**Utilizar datos más inclusivos**

Ejecutar el modelo con datos que incluyan a más proporción de los grupos que fueron infrarrepresentados en este modelo, ya que, a pesar de que la base de datos recogió muestras en distintas partes del mundo e intentó tener una población variada, sigue habiendo un sesgo de género y de etnia en los datos que provoca que haya una desigualdad de representación en los modelos.

### 6.0.2. Seguimiento de la planificación

Seguir la planificación a lo largo del trabajo resultó ser complicado. Aparecieron problemas inesperados, como no tener poder de computación suficiente en el ordenador para manejar un conjunto de datos tan grande, a los que hubo que buscar soluciones, lo que provocó que los objetivos esperados para cada entrega se retrasaran. La metodología prevista no fue suficiente; no se previó el filtrado de genes con el análisis de expresión, el balanceo de datos con SMOTE, ni el análisis de importancia o el análisis de componentes principales para la reducción de la dimensionalidad. Todas estas téncicas fueron añadidas posteriormente para garantizar el éxito del trabajo, ya que aseguraron que se alcanzase el rendimiento requerido.

# Capítulo 7

# Glosario

**ANN** Siglas para Artificial Neural Network, red neuronal artificial en inglés. Es un algoritmo de aprendizaje automático supervisado inspirado en las redes neuronales cerebrales.

**Aprendizaje automático** sistema que toma un gran conjunto de datos y lo usa para detectar patrones, aprendiendo o mejorando su funcionamiento de forma autónoma.

**Carcinoma** Cáncer que comienza en el tejido que cubre el interior o exterior de un órgano

**ccRCC** siglas para clear cell Renal cell carcinoma: Carcinoma renal de células claras

**chRCC** siglas para chromophobe type Renal cell carcinoma: Carcinoma renal cromófobo

**DT** Siglas para Decision Tree, árbol de decisiones en inglés. Algoritmo de aprendizaje automático supervisado inspirado en la estructura de un árbol

**Especificidad** : La especificidad, o el ratio de verdaderos negativos, es la proporción de muestras negativas que fueron clasificadas correctamente

**FN** Falsos negativos. Número de muestras positivas clasificadas incorrectamente como negativas

**FP** Falsos positivos. Número de muestras negativas clasificadas incorrectamente como positivas

**KNN** siglas para k Nearest Neighbours, o k vecinos más cercanos. Algoritmo de aprendizaje automático que clasifica datos nuevos a partir de la clase de los vecinos más cercanos, es decir, las muestras más parecidas.

**LFC** Log2 Fold Change: Medida usada en el análisis de expresión que nos muestra si un gen está sobre o infraexpresado entre dos condiciones diferentes.

**Matriz de confusión** Matriz que se utiliza en aprendizaje automático para evaluar el desempeño de un algoritmo. Indica el número de muestras clasificadas correcta e incorrectamente como positivas o negativas.

**NB** Siglas para Naive Bayes, algoritmo de aprendizaje automático que obtiene los resultados calculando las probabilidades de cada clase.

**pRCC** Siglas para Papillary Renal cell carcinoma. Carcinoma papilar de células renales.

**p valor ajustado** valor de p tras la corrección de Benjamin-Hochberg o FDR utilizada para reducir el número de falsos positivos

**p valor** El valor de p o p valor se utiliza en contraste de hipótesis e indica la probabilidad de haber obtenido los resultados de forma arbitraria.

**RCC** Siglas para Renal cell carcinoma, carcinoma de células renales.

**RF** Algoritmo de aprendizaje automático formado por árboles de decisiones, que obtiene resultados escogiendo la clase mayoritaria obtenida por ellos.

**Sensibilidad** La sensibilidad, o el ratio de verdaderos positivos, es la proporción de muestras positivas que fueron clasificadas correctamente

**Server** Servidor. Parte de una aplicación de Shiny que contiene las funciones de la aplicación.

**SMOTE** Siglas para Synthetic Minority Over-sampling Technique. Transformación de datos que se utiliza cuando no existe el mismo número de muestras en todas las clases, y que genera artificialmente muestras para las clases minoritarias.

**SVM** Siglas para Support Vector Machine, algoritmo de aprendizaje automático que se basa en crear particiones en el espacio mediante una división en una dimensión superior llamada hiperplano, que separa de manera óptima las muestras de diferentes clases.

**TCGA** Siglas para The Cancer Genome Atlas. proyecto que recoge datos del genoma, transcriptoma, epigenoma y proteoma humano con el objetivo de obtener una mejor comprensión del cáncer.

**TN** Verdaderos negativos. Número de muestras negativas clasificadas correctamente como negativas, es decir, n´umero de muestras que se clasifican como que no son la clase de interés, de forma acertada.

**TP** Verdaderos positivos. Número de muestras positivas clasificadas correctamente como positivas.

**UI** Siglas para User Interface: Inferfaz de Usuario. Parte de la aplicación de Shiny que contiene la apariencia de la aplicación.

# Capítulo 8

# Bibliografía

[1] Sandeep Anand Padala, Adam Barsouk, Krishna Chaitanya Thandra, Kalyan Saginala, Azeem Mohammed, Anusha Vakiti, Prashanth Rawla, and Alexander Barsouk. Epidemiology of renal cell carcinoma. *World journal of oncology*, 11(3):79, 2020.

[2] Jacques Ferlay, Isabelle Soerjomataram, Rajesh Dikshit, Sultan Eser, Colin Mathers, Marise Rebelo, Donald Maxwell Parkin, David Forman, and Freddie Bray. Cancer incidence and mortality worldwide: sources, methods and major patterns in globocan 2012. *International journal of cancer*, 136(5):E359–E386, 2015.

[3] Mike M Nguyen, Inderbir S Gill, and Lars M Ellison. The evolving presentation of renal carcinoma in the united states: trends from the surveillance, epidemiology, and end results program. *The Journal of urology*, 176(6):2397–2400, 2006.

[4] Martin CS Wong, William B Goggins, Benjamin HK Yip, Franklin DH Fung, Colette Leung, Yuan Fang, Samuel YS Wong, and CF Ng. Incidence and mortality of kidney cancer: temporal patterns and global trends in 39 countries. *Scientific reports*, 7(1):15698, 2017.

[5] Rebecca L Siegel, Kimberly D Miller, Hannah E Fuchs, Ahmedin Jemal, et al. Cancer statistics, 2021. *Ca Cancer J Clin*, 71(1):7–33, 2021.

[6] Bhavani Krishnan, Tracy L Rose, Jordan Kardos, Matthew I Milowsky, and William Y Kim. Intrinsic genomic differences between african american and white patients with clear cell renal cell carcinoma. *JAMA oncology*, 2(5):664–667, 2016.

[7] F Bazan and M Busto. Radiología del carcinoma renal. *Radiología*, 56(1):61–75, 2014.

[8] Mary Jayson and Holt Sanders. Increased incidence of serendipitously discovered renal cell carcinoma. *Urology*, 51(2):203–205, 1998.

[9] Zijie Wang, Xiaofei Zhang, Xinning Wang, Jianfei Li, Yuhao Zhang, Tianwei Zhang, Shang Xu, Wei Jiao, and Haitao Niu. Deep learning techniques for imaging diagnosis of renal cell carcinoma: current and emerging trends. *Frontiers in Oncology*, 13:1152622, 2023.

[10] Ahmed Hosny, Chintan Parmar, John Quackenbush, Lawrence H Schwartz, and Hugo JWL Aerts. Artificial intelligence in radiology. *Nature Reviews Cancer*, 18(8):500–510, 2018.

[11] Stefan Schulz, Ann-Christin Woerl, Florian Jungmann, Christina Glasner, Philipp Stenzel, Stephanie Strobl, Aurélie Fernandez, Daniel-Christoph Wagner, Axel Haferkamp, Peter Mildenberger, et al. Multimodal deep learning for prognosis prediction in renal cancer. *Frontiers in oncology*, 11:788740, 2021.

[12] Fangjun Li, Mu Yang, Yunhe Li, Mingqiang Zhang, Wenjuan Wang, Dongfeng Yuan, and Dongqi Tang. An improved clear cell renal cell carcinoma stage prediction model based on gene sets. *BMC bioinformatics*, 21(1):1–15, 2020.

[13] Noor Pratap Singh, Raju S Bapi, and PK Vinod. Machine learning models to predict the progression from early to late stages of papillary renal cell carcinoma. *Computers in biology and medicine*, 100:92–99, 2018.

[14] Sugi Lee, Jaeeun Jung, Ilkyu Park, Kunhyang Park, and Dae-Soo Kim. A deep learning and similarity-based hierarchical clustering approach for pathological stage prediction of papillary renal cell carcinoma. *Computational and structural biotechnology journal*, 18:2639–2646, 2020.

[15] Ali Muhamed Ali, Hanqi Zhuang, Ali Ibrahim, Oneeb Rehman, Michelle Huang, and Andrew Wu. A machine learning approach for the classification of kidney cancer subtypes using mirna genome data. *Applied Sciences*, 8(12):2422, 2018.

[16] Maximilian Pallauf, Yasser Ged, and Nirmish Singla. Molecular differences in renal cell carcinoma between males and females. *World journal of urology*, 41(7):1727–1739, 2023.

[17] The cancer genome atlas. `https://www.cancer.gov/tcga`.

[18] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, et al. The cancer imaging archive (tcia): maintaining and operating a public information repository. *Journal of digital imaging*, 26:1045–1057, 2013.

[19] Mahmood Khalsan, Lee R Machado, Eman Salih Al-Shamery, Suraj Ajit, Karen Anthony, Mu Mu, and Michael Opoku Agyeman. A survey of machine learning approaches applied to gene expression analysis for cancer prediction. *IEEE Access*, 10:27522–27534, 2022.

[20] Sara Tarek, Reda Abd Elwahab, and Mahmoud Shoman. Gene expression based cancer classification. *Egyptian Informatics Journal*, 18(3):151–159, 2017.

[21] Padideh Danaee, Reza Ghaeini, and David A Hendrix. A deep learning approach for cancer detection and relevant gene identification. In *Pacific symposium on biocomputing 2017*, pages 219–229. World Scientific, 2017.

[22] Joe W Chen and Joseph Dhahbi. Lung adenocarcinoma and lung squamous cell carcinoma cancer classification, biomarker identification, and gene expression analysis using overlapping feature selection methods. *Scientific reports*, 11(1):13323, 2021.

[23] Naciones Unidas. Informe de los objetivos de desarrollo sostenible, 2023.

[24] Sherry Bhalla, Kumardeep Chaudhary, Ritesh Kumar, Manika Sehgal, Harpreet Kaur, Suresh Sharma, and Gajendra P.S. Raghava. Gene expression-based biomarkers for discriminating early and late stage of clear cell renal cancer. *Scientific Reports*, 7(March):1–13, 2017.

[25] Leilei Zhou, Zuoheng Zhang, Yu-Chen Chen, Zhen-Yu Zhao, Xin-Dao Yin, and Hong-Bing Jiang. A deep learning-based radiomics model for differentiating benign and malignant renal tumors. *Translational oncology*, 12(2):292–300, 2019.

[26] Takashi Tanaka, Yong Huang, Yohei Marukawa, Yuka Tsuboi, Yoshihisa Masaoka, Katsuhide Kojima, Toshihiro Iguchi, Takao Hiraki, Hideo Gobara, Hiroyuki Yanai, et al. Differentiation of small renal masses on multiphase contrast-enhanced ct by deep learning. *American journal of roentgenology*, 214(3):605–612, 2020.

[27] Fatemeh Zabihollahy, Nicola Schieda, Satheesh Krishna, and Eranga Ukwatta. Automated classification of solid renal masses on contrast-enhanced computed tomography images using convolutional neural network with decision fusion. *European Radiology*, 30:5183–5190, 2020.

[28] Ianto Lin Xi, Yijun Zhao, Robin Wang, Marcello Chang, Subhanik Purkayastha, Ken Chang, Raymond Y Huang, Alvin C Silva, Martin Vallieres, Peiman Habibollahi, et al. Deep learning to distinguish benign from malignant renal lesions based on routine mr imaging. *Clinical Cancer Research*, 26(8):1944–1952, 2020.

[29] Seokmin Han, Sung Il Hwang, and Hak Jong Lee. The classification of renal cancer in 3-phase ct images using a deep learning method. *Journal of digital imaging*, 32:638–643, 2019.

[30] Yao Zheng, Shuai Wang, Yan Chen, and Hui-qian Du. Deep learning with a convolutional neural network model to differentiate renal parenchymal tumors: a preliminary study. *Abdominal Radiology*, 46:3260–3268, 2021.

[31] Teng Zuo, Yanhua Zheng, Lingfeng He, Tao Chen, Bin Zheng, Song Zheng, Jinghang You, Xiaoyan Li, Rong Liu, Junjie Bai, et al. Automated classification of papillary renal cell carcinoma and chromophobe renal cell carcinoma based on a small computed tomography imaging dataset using deep learning. *Frontiers in Oncology*, 11:746750, 2021.

[32] Brett Lantz. *Machine learning with R: expert techniques for predictive modeling*. Packt publishing ltd, 2019.