Universidad Nacional Autónoma de México

facultad de ciencias

Métodos estadísticos de aprendizaje supervisado para la detección de cobertura del suelo a través de Imágenes Satelitales

T E S I S

que para obtener el título de:

Actuario

presenta:

Héctor Miguel Olivares García

Director de tesis:

Dr. Gonzalo Pérez de la Cruz

Ciudad de México 2023

Quien con monstruos lucha cuide de no convertirse a su vez en monstruo.
Cuando miras largo tiempo a un abismo, el abismo también mira dentro de ti.

-Friedrich Nietzsche

Sobre todo, no te mientas a ti mismo.
El hombre que se miente a sí mismo y escucha su propia mentira llega a un punto en el que no puede distinguir la verdad dentro de él, ni a su alrededor, y por lo tanto pierde todo respeto por sí mismo y por los demás.
Y al no tener respeto, deja de amar.

-Fiodor Dostoievski

Índice general

1 Introducción **1**
- 1.1 Motivación . . . 1
- 1.2 Objetivos . . . 2
  - 1.2.1 Objetivo general . . . 2
  - 1.2.2 Objetivos específicos . . . 2
  - 1.2.3 Región de aplicación . . . 3

2 Estructuración de datos de imágenes satelitales **5**
- 2.1 Sensor remoto . . . 7
- 2.2 Preprocesamiento de las imágenes satelitales . . . 9
  - 2.2.1 Corrección radiométrica . . . 10
  - 2.2.2 Reproyección . . . 11
  - 2.2.3 Índices . . . 11
- 2.3 Obtención de la variable sobre tipo de cobertura de suelo . . . 16

3 Metodología y algoritmos de clasificación **18**
- 3.1 Clasificación supervisada . . . 18
- 3.2 Estimación del poder predictivo . . . 18
  - 3.2.1 Repeated Hold-Out . . . 19
  - 3.2.2 K-Cross Validation . . . 20
- 3.3 Métodos para construir reglas de clasificación . . . 22
  - 3.3.1 Regresión Binaria . . . 22
  - 3.3.2 Máquinas de soporte vectorial (SVM) . . . 27
  - 3.3.3 K vecinos más cercanos (KNN) . . . 33
  - 3.3.4 Árboles de Decisión y Bosques Aleatorios (Random Forest, RF) . . . 34

4 Aplicación **40**
- 4.1 Construcción del conjunto de datos . . . 40
  - 4.1.1 Obtención de los datos . . . 40
  - 4.1.2 Transformación de la imagen satelital a un conjunto de datos . . . 42
- 4.2 Análisis descriptivo . . . 45
- 4.3 Aplicación de modelos de clasificación con datos de junio de 1984 . . . 46
  - 4.3.1 Construcción de las reglas de clasificación y resultados de la estimación del poder predictivo . . . 47
- 4.4 Aplicación y resultados del mejor modelo con datos de diciembre de 1984 . . . 55

5 Conclusiones **57**

Bibliografía **59**

Anexo A: Código de los ejemplos del capítulo 3 **62**

Anexo B: Funciones **65**

Anexo C: Código de la aplicación del capítulo 4 **80**

Resumen

La clasificación de la cobertura del suelo facilita la implementación de políticas públicas y el monitoreo ambiental, lo que a su vez ayuda en la toma de decisiones informadas.

La segmentación, al usar métodos de clasificación supervisada, así como aprovechar las imágenes satelitales, puede agilizar la tarea de generar información, ya que en muchas ocasiones la clasificación de la cobertura del suelo proporciona información no oportuna con métodos tradicionales, debido a que implica un gran costo económico y un largo proceso de estudio.

El objetivo general del presente trabajo es la aplicación de métodos de clasificación supervisada para segmentar el tipo de suelo en dos categorías: agricultura y no agricultura, considerando información de la Ciudad de Harare. La aplicación parte de datos no estructurados, de imágenes satelitales, así como su adecuación para el uso de métodos de clasificación supervisada.

Los datos utilizados están disponibles en Kamusoko (2019), y una aplicación de esto está en Kamusoko (2013). Los datos corresponden a dos temporadas diferentes del mismo año, que refieren a los meses de julio y diciembre de 1984. Con la información del mes de julio se entrenaron y se compararon los modelos. Posteriormente se aplicó la regla de clasificación que tuvo mejor rendimiento predictivo a los datos del mes de diciembre para simular una situación real y analizar el desempeño de la regla entrenada.

Se consideraron métodos de clasificación supervisada: KNN, regresión binaria, SVM y bosques aleatorios. Posteriormente, se realizó una comparación de los métodos a través de la estimación del poder predictivo. Dado que algunos modelos pueden ser complejos o tardados en su aplicación, pues incluyeron varios hiperparámetros, se implementaron técnicas de programación en paralelo para distribuir la carga de trabajo del CPU. Además, se empleó el método de remuestreo Repeated Hold Out para el cálculo del poder predictivo y K Cross Validation para definir los hiperparámetros óptimos con K=10.

Destacando los bosques aleatorios como el modelo ganador, pues obtuvo una exactitud del 87% y una precisión del 86% al identificar al grupo de agricultura a pesar de contar con un desbalanceo entre grupos: un 31% de observaciones para agricultura y un 69% para no agricultura. Además, se observó una enorme diferencia en los tiempos de ejecución entre el SVM y los bosques aleatorios.

A pesar del buen desempeño de los bosques aleatorios, este cambió totalmente al aplicar la regla de clasificación en una estación del año diferente al los datos de entrenamiento, resaltando la importancia del monitoreo y la extracción de información cuando existen cambios bruscos en la cobertura del suelo.

CAPÍTULO 1

Introducción

1.1 Motivación

Una de las motivaciones para la realización del presente estudio se encuentra asociada con el Indicador SDG 15.3.1 de la "Agenda 2030 para el Desarrollo Sostenible". El propósito del Objetivo de Desarrollo Sostenible (ODS) 15 es proteger, restaurar y promover el uso sostenible de los ecosistemas terrestres. La meta 15.3.1 se concentra en combatir la desertificación y la restauración tanto de la tierra como del suelo degradado. Esta problemática se ha convertido en un compromiso internacional que establece la necesidad de evaluar el progreso del indicador 15.3.1 que mide la proporción de tierra degradada sobre la superficie total (Trends Earth, 2023, Sección 2.1).

Otra motivación se asocia con los compromisos internacionales celebrados por México, particularmente el Acuerdo de París de 2015, cuyo objetivo principal es establecer un marco global para abordar temas relacionados con el cambio climático mediante la reducción de las emisiones de gases de efecto invernadero y la adaptación a sus efectos. Por tal motivo, es importante monitorear y conocer las condiciones de la cobertura del suelo con el propósito de:

- Proporcionar información esencial sobre cómo están cambiando los ecosistemas y cómo estos se ven afectados por el cambio climático.
- Detectar y analizar los cambios en la vegetación, la deforestación, la expansión urbana y otros factores clave que ayuden a comprender mejor las tendencias climáticas y a evaluar la eficacia de las políticas de mitigación y adaptación.

Sin embargo, desde la década de los noventa, algunos países dejaron de actualizar los mapas de cobertura del suelo debido a su complejidad y alto costo. Esto se debe a que involucra personal altamente especializado, tanto para realizar determinaciones en campo y laboratorio, así como para utilizar herramientas de teledetección. Actualmente, es importante revisar métodos modernos que exploren fuentes de datos iterativos: tal es el caso de las imágenes de satélite que, pueden servir para obtener indicadores sobre la cobertura de suelo.

En México se han llevado a cabo dos esfuerzos para monitorear el cambio de suelo y aplicar políticas públicas que contribuyan al cuidado de los diferentes tipos de suelo. Uno de ellos realizado por la Comisión Nacional Forestal (CONAFOR) que ha implementado el sistema Satelital de Monitoreo Forestal (SAMOF), con las siguientes especificaciones de acuerdo con (CONAFOR, 2020,p.4):

> "En el 2011, considerando nuevos avances tecnológicos, la CONAFOR, en colaboración con otras dependencias federales, comenzó el desarrollo del software de Monitoreo de Datos de Actividad de México (MAD-Mex, V2.0) para la obtención de mapas de cobertura del suelo de forma automatizada; sin embargo, la experiencia en la utilización de este software evidenció que la diversidad ecológica y fisiográfica del país limitan, de manera importante, la posibilidad de realizar el monitoreo de la cubierta forestal a través de procesos totalmente automatizados".

En otras palabras, la complejidad y variabilidad del entorno natural mexicano requerían enfoques más flexibles y adaptativos en lugar de depender completamente de procesos automáticos. La CONAFOR

(2020, p.5) informa que:

> "...los mapas elaborados tienen una precisión o exactitud temática mayor al 80% a nivel de las 34 clases de coberturas de vegetación y otras coberturas del suelo, y de 90% a nivel de las clases para reporte adoptadas por las directrices del Panel Intergubernamental de Expertos sobre el Cambio Climático (IPCC por sus siglas en inglés)."

Por otro lado, está el sistema MAD-Mex que tiene el objetivo de producir información anual sobre cobertura de suelo a nivel nacional, a partir del procesamiento automático de imágenes satelitales, realizado por la Comisión Nacional para el Conocimiento y Uso de la Biodiversidad (CONABIO), quien "provee el mantenimiento técnico del sistema de procesamiento automatizado MAD-Mex con la implementación de nuevas tecnologías y algoritmos para procesar de manera eficiente los insumos de información que reciba" (CONABIO, 2020, p.5).

En ambos proyectos se emplearon métodos de segmentación: procesos mediante los cuales se divide una imagen en regiones o segmentos que comparten características específicas, uno de ellos es conocido como Segmentación de Berkeley. Adicionalmente, también se aplicaron métodos comunes de preprocesamiento que son empleados en este trabajo, los cuales se centran en corregir la reflectancia en cada píxel. De igual modo, se generaron variables relacionadas con los índices espectrales e índices de textura.

1.2 Objetivos

En este trabajo se muestra el uso de algunos métodos estadísticos clásicos usados para la clasificación de nuevas observaciones mediante una aplicación correspondiente a la detección de cobertura de suelo a partir de información satelital. La aplicación se restringe a un conjunto de imágenes satelitales disponibles de manera gratuita.

Dicha aplicación se enfoca en describir lo correspondiente a una parte del procesamiento de las imágenes y en la aplicación de los métodos de aprendizaje supervisado: regresión binaria, máquinas de soporte vectorial, KNN y bosques aleatorios, para clasificar las categorías de suelo: agricultura y no agricultura.

Una vez obtenidos los resultados y seleccionando el método con mejor estimación predictiva, se interpretarán los resultados y se comentará sobre los requerimientos computacionales, así como los posibles métodos alternativos que se podrían usar para este tipo de aplicaciones. Para el procesamiento de los datos y la ejecución de los algoritmos se utilizará el lenguaje de programación R y Rstudio como IDE.

1.2.1 Objetivo general

Identificar dos tipos de suelo, agricultura y no agricultura, en la ciudad de Harare, Zimbabue, donde se aplicarán modelos de clasificación supervisada.

1.2.2 Objetivos específicos

Los objetivos específicos son los siguientes:

- Describir el proceso de obtención y aplicación del preprocesamiento de la información no estructurada del satélite Landsat 5.
- Calcular y comparar el poder predictivo de los métodos de clasificación siguientes con información perteneciente al 22 de junio de 1984: regresión binaria (regresión logística, regresión probit, regresión lasso), KNN, máquinas de soporte vectorial y bosques aleatorios.

Adicionalmente, para los datos correspondientes a diciembre de 1984, se empleará el algoritmo que demostró mejor desempeño predictivo en los datos de junio. Esto con el objetivo de:

- Explorar la posibilidad de usar el algoritmo en todas las estaciones meteorológicas o determinar si sería necesario ajustar reglas de clasificación específicas para cada estación.

1.2.3 Región de aplicación

El diseño de un sistema de clasificación para imágenes satelitales se realiza considerando las necesidades del usuario, el área que abarca un píxel en la imagen, los algoritmos disponibles y las limitaciones de tiempo y recursos. Las muestras de entrenamiento deben incluir etiquetas sobre el tipo de suelo que son esenciales para el trabajo.

Las etiquetas representan el tipo de suelo en una zona geográfica específica y su creación implica la asignación manual de etiquetas a cada píxel en la imagen, ya sea a partir de la interpretación visual de imágenes satelitales o mediante trabajo de campo. Generalmente, este tipo de proyectos son realizados por especialistas en teledetección, ciencias de la tierra o personas con conocimiento a priori de la zona geográfica.

Por otro lado, es importante tener en cuenta diversas consideraciones adicionales como la necesidad de volver a generar etiquetas para adaptarse a los cambios estacionales, la dimensión del píxel que puede influir en la cantidad de etiquetas en una imagen. Por lo tanto, para el diseño y enfoque usado en la construcción de las etiquetas deben tenerse en cuenta estos factores, así como la relación entre la fecha de captura de la imagen y la fecha de creación de la etiqueta.

La intención inicial de este trabajo era aplicar métodos de clasificación para la cobertura del suelo en México. Sin embargo, la falta de una base de datos sólida y gratuita sobre la composición del suelo mexicano nos llevó a optar por otras alternativas. En consecuencia, se buscó información gratuita y etiquetada. De tal manera que el presente trabajo se limitará a la información de Harare, Zimbabue, asegurando la disponibilidad de imágenes y etiquetas del suelo.

Los conjuntos de datos que se usarán se recopilaron originalmente para un proyecto piloto de cobertura del suelo, como detalla Kamusoko et al.(2013, p.325), donde se menciona que el etiquetado se realizó: *"...based on the Forestry Commission (Zimbabwe) and the Surveyor-General national cover classes classification schemes as well as the author's a priori knowledge of the study area".*

Cabe mencionar que Harare experimenta variaciones ambientales notables durante todo el año, pues en octubre presenta un clima cálido, de noviembre a abril la zona es húmeda, mientras que de mayo a agosto es seca. La información disponible corresponde al año 1984, sugiriendo que las coordenadas geográficas y los patrones climáticos podrían haber experimentado cambios desde entonces.

Por último, cuando se trabaja con proyectos de imágenes satelitales puede ser necesario utilizar más de una imagen para cubrir una zona específica. La imagen proporcionada en Kamusoko (2019) representa una reducción de dimensiones con respecto al tamaño original del satélite Landsat 5 del cual proviene. Por lo general, las imágenes captadas por este satélite cubren alrededor de 34,225 km^2, mientras que la Ciudad de Harare abarca aproximadamente 942 km^2, así que se puede suponer que una sola imagen puede abarcar esta zona, ver Figura 1.1. Esto ahorra una parte del preprocesamiento que es la unión de diferentes imagénes satelitales para cubrir un área de estudio.

Figura 1.1: Ciudad de Harare, Zimbabue. La provincia metropolitana se extiende aproximadamente entre los 17°40' y los 18°00' sur, y entre los 30°55' y los 31°15' este, abarcando un área de unos 942 km2.

CAPÍTULO 2

Estructuración de datos de imágenes satelitales

Una imagen satelital es una representación gráfica de la información capturada por un sensor en órbita alrededor de la Tierra. Esta representación no es meramente una imagen satelital, sino una estructura de datos compleja que alberga gran cantidad de información almacenada en distintos canales o bandas.

El píxel es la unidad básica de una imagen satelital. Cada píxel en almacena información geográfica que indica su ubicación específica en la Tierra, así como información espectral o espectro electromagnético. Esta última es una medida de la intensidad de la radiación electromagnética reflejada o emitida por el objeto representado en ese píxel.

Es importante subrayar que una imagen satelital está compuesta por múltiples bandas o canales, como se puede ver en la Figura 2.1. Cada banda captura una porción específica del espectro electromagnético. Al analizar estas bandas en conjunto se obtiene una visión detallada y completa de la zona de interés.

Desde la perspectiva del manejo y análisis de datos, las propiedades de los datos de la imagen satelital dependen de la resolución radiométrica, ésta se expresa en términos de números digitales que representan la intensidad de brillo del objeto en el píxel, generando un estructura parecida a un dataframe o conjunto de datos. Por lo tanto, la resolución radiométrica es un factor importante en la interpretación de las imágenes satelitales.

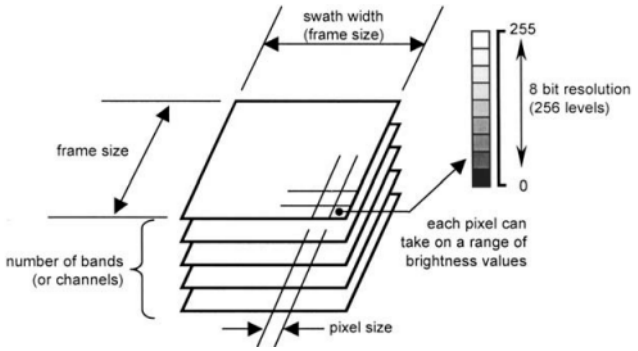


Figura 2.1: Características técnicas de la composición de las imágenes Satelitales. La combinación de bandas espectrales corresponden en conjunto a una imagen satelital. [1]

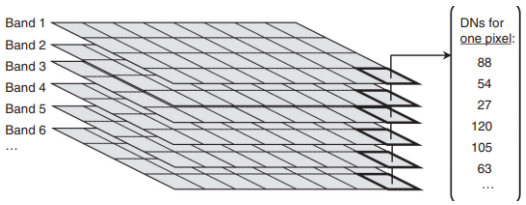
El análisis de imágenes satélitales se realiza, generalmente, utilizando estructuras de datos denominadas RASTER, donde se almacena información geográfica y además una estructura de dataframe, donde los canales o bandas representan las columnas, mientras que las filas corresponden a cada píxel, tal como se muestra en la Figura 2.2.

En este contexto, se identifican a los píxeles como (x, y), donde x y y representan las coordenadas geográficas de la zona de estudio representada por el píxel. Para cada banda y píxel se asigna un valor de intensidad de brillo, $f_j(x, y),\ j = 1, 2, ..., B$ donde B es el número de bandas. Esto significa que para cada par de coordenadas (x, y), se tiene varios valores correspondientes a la intensidad de brillo en las diferentes bandas.

[1] Imagen obtenida de *Remote Sensing Digital Image Analysis*(p.3), Richards J. A & Jia X. Berlin/Heidelberg, Germany: springer.

$$\begin{bmatrix} f_1(x_1, y_1) & f_2(x_1, y_1) & \dots & f_B(x_1, y_1) \\ f_1(x_2, y_2) & f_2(x_2, y_2) & \dots & f_B(x_2, y_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(x_R, y_R) & f_2(x_R, y_R) & \dots & f_B(x_R, y_R) \end{bmatrix}$$

(A) Raster



(B) Imagen satelital

Figura 2.2: En A) se presenta la matriz de las bandas en conjunto, donde B representa el total de las bandas espectrales y R es el total de píxeles que contiene la imagen satelital, cada fila se refiere a un píxel en específico representado por las coordenadas (x_j, y_j). En B) la representación de una imagen satelital, el píxel que resalta representa una fila en A) y los valores numéricos hacen referencia al nivel de intensidad de brillo de cada banda. [2]

Es importante destacar que la calidad y cantidad de los píxeles en una imagen satelital dependen en gran medida del sensor remoto. Cada píxel de la imagen satelital se ve afectado en términos de intensidad de brillo, así como en aspectos de tamaño, número de bandas, entre otros. Estos factores pueden agruparse en cuatro tipos de resoluciones para el análisis de las imágenes:

- Resolución espacial: Es la capacidad que posee el sensor para distinguir los detalles espaciales en la imagen, como la nitidez y el nivel de detalle de los objetos presentes en la superficie terrestre. Se mide en metros por píxel. En la Figura 2.3 presentan diferentes tipos de resolución espacial, por ejemplo, en a) se nota una alta resolución espacial que permite identificar claramente objetos pequeños aunque ocasiona que el volumen de datos sea inmenso, mientras que en f) se nota una baja resolución muestra objetos más grandes ocasionando una falta de claridad en sí mismos.
- Resolución espectral: Se refiere a la capacidad del sensor para captar la energía reflejada o emitida en diferentes bandas del espectro electromagnético. Los sensores multiespectrales pueden registrar información en varias bandas. La resolución espectral permite identificar diferentes tipos de superficies terrestres y estudiar sus propiedades físicas y químicas.
- Resolución radiométrica: Representa la sensibilidad del sensor para detectar variaciones en la intensidad de la energía reflejada o emitida en la imagen. Una alta resolución radiométrica permite distinguir pequeñas diferencias en los niveles de brillo, lo que es importante para la detección de sutiles cambios en la superficie o la detección de contaminantes, así como diferencias en la vegetación.
- Resolución temporal: Es la referencia a la frecuencia con la que el sensor vuelve a pasar sobre una misma área para tomar imágenes. Una alta resolución temporal implica que el sensor puede captar cambios en la superficie con mayor frecuencia, lo que es útil para monitorear fenómenos dinámicos como cambios en el uso del suelo, crecimiento de cultivos o desastres naturales.

Además, cuando se analizan áreas extensas con una baja resolución espacial, es probable que diferentes tipos de resolución espectral coexistan en un mismo píxel, ocasionando que éste tenga características de diferentes tipos de suelo. Generalmente, cuando existen estos problemas de baja resolución espacial, se recurren a las firmas espectrales que yacen en cada banda espectral.

Una firma espectral se refiere a un patrón distintivo en el espectro electromagnético que es único para diversos tipos de objetos o materiales presentes en la Tierra, por ejemplo, el agua, la vegetación y el suelo desnudo exhibirán firmas espectrales diferentes entre sí. Estas características son valiosas para identificar dichos objetos en imágenes satelitales.

La obtención de firmas espectrales se realiza a partir de la información capturada en diversas bandas espectrales dentro del sensor remoto. Cada banda contribuye a conformar el patrón espectral característico de un objeto o material específico. En consecuencia, un píxel contiene datos provenientes

[2]Imagen obtenida de *Remote sensing and image interpretation* (p.34). Lillesand, T., Kiefer, R. W., & Chipman, J. John Wiley & Sons.

[3]Imagen obtenida de *Remote sensing and image interpretation* (p.74). Lillesand, T., Kiefer, R. W., & Chipman, J. John Wiley & Sons.

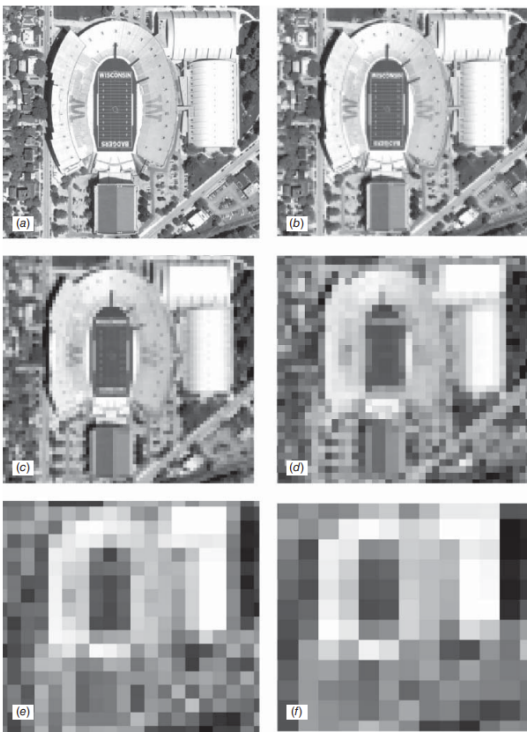


Figura 2.3: Ejemplo de diferentes tipos de resolución espacial sobre la misma área. La resolución corresponde a metros por píxel: La figura (a) 1m, (b) 2.5m, (c) 5m, (d) 10m, (e) 20m y (f) 30m. Se puede observar la disminución de la claridad de los objetos conforme la resolución va aumentando, así como la reducción del número de píxeles.[3]

de múltiples bandas espectrales que al combinarlas generan las firmas espectrales correspondientes. Este proceso permite asignar a cada píxel un valor numérico que representa su firma espectral única, es decir, se puede clasificar.

En la Tabla 2.1 se puede apreciar cómo la información contenida en estas bandas ha sido útil en diferentes contextos, pues al combinarlas proporcionan información de la firma espectral. Es importante destacar que las bandas 1, 2, 3 y 4 contribuyen a la identificación del tipo de suelo, ofreciendo una perspectiva sobre las posibles bandas más relevantes para este estudio.

Banda	Descripción	Aplicaciones
Banda 1	Azul	Mapeo de aguas costeras. Diferencias de suelo y vegetación.
Banda 2	Verde	Mapeo de vegetación y calidad de agua.
Banda 3	Roja	Absorción de la clorofila. Diferenciación de especies vegetales; áreas urbanas y uso del suelo.
Banda 4	Infrarrojo cercano 1	Identificación de áreas de incendios y húmedas. Agricultura y vegetación.
Banda 5	Infrarrojo cercano 2	Diferenciación entre nubes y nieve.
Banda 6	Infrarrojo térmico	Mapeo de stress térmico en plantas. Propiedades termales del suelo.
Banda 7	Infrarrojo medio	Identificación de minerales. Mapeo hidrotermal.

Tabla 2.1: Descripción de las Bandas más comunes y sus posibles usos.

2.1 Sensor remoto

Es fundamental entender que la calidad del análisis de una imagen satelital depende en gran medida del tipo de información que sea obtenida y esta información, a su vez, depende de cómo fue adquirida.

En este sentido, el sensor remoto juega un papel importante para determinar la calidad de la imagen satelital.

Hoy en día existen varios sensores remotos que ayudan a obtener información de las imágenes satelitales. Además, existen sitios web que proporcionan exclusivamente información satelital (esto no incluye etiquetas de suelo), mientras que otros sitios se dedican a vender únicamente imágenes satelitales, por dar un ejemplo el sitio Rapid Eye ofrece imágenes de alta calidad. Sin embargo, puede variar el costo dependiendo del área y calidad disponible.

Afortunadamente, también existen sitios web de acceso gratuito o público donde se pueden obtener las imágenes satelitales que se utilizan para su estudio. Por ejemplo, los datos obtenidos por Kamusoko (2019, p.13) corresponden al satélite Landsat 5 provenientes del Servicio Geológico de los Estados Unidos (USGS).

El satélite Landsat fue el primer sensor remoto que envió Estados Unidos con el objetivo de monitorear los recursos terrestres, su nombre proviene de Land (tierra) y Sat (satélite) (INEGI,2022).

Las características distintivas del satélite Landsat 5 que destacan son las siguientes: la imagen satélital está compuesta por ocho bandas espectrales (resolución espectral), cubre un área de 185 km por 185 km que equivale a 34,225 km^2 en la superficie de la Tierra, la resolución espacial y espectral se encuentra detallada en la Tabla 2.2. En cuanto a la resolución radiométrica, se cifra en 8 bits(2^8), permitiendo la representación de 256 niveles de brillo o tonos grisáceos. En términos temporales, la resolución se sitúa en un intervalo de 16 días.

Con el paso del tiempo, la familia de satélites Landsat ha experimentado un proceso evolutivo. Aunque se han desarrollado varias versiones, algunas de ellas tienen mayor uso, tal es el caso de los modelos Landsat 5 y Landsat 8.

Banda	Nombre	Resolución espacial (m)	Resolución Espectral(μm)
Banda 1	Azul	30	0.45-0.52
Banda 2	Verde	30	0.52-0.60
Banda 3	Roja	30	0.63-0.69
Banda 4	Infrarrojo cercano 1	30	0.76-0.90
Banda 5	Infrarrojo cercano 2	30	1.55-1.75
Banda 6	Infrarrojo térmico	120	10.40-12.50
Banda 7	Infrarrojo medio	30	2.08-2.35

Tabla 2.2: Resolución espacial y espectral de las bandas de las imágenes Landsat 5 (INEGI, 2022).

De acuerdo con Chuvieco (1995, p.106) la captura de los datos que utiliza la familia Landsat se le conoce como barrido whiskbroom o radiómetro barredor. Dispone de un espejo móvil que oscila perpendicularmente a la dirección de la trayectoria y que permite explorar una franja de terreno en ambos lados de esta. Cada movimiento del espejo envía información de una franja distinta al conjunto de sensores. Un ejemplo del proceso se puede ver en la Figura 2.4.

El resultado que genera la obtención de esta información se ve reflejado en una representación visual que se asemeja a una malla cuadrangular y puede ser descargada de diferentes sitios web. La resolución de la malla puede variar, tal como se ilustra en la Figura 2.3. Sin embargo, es importante señalar que estos datos están comprimidos en diferentes tipos de archivos. Entre ellos, se incluyen archivos en formato txt, además de varias representaciones visuales de la zona de interés. Estas representaciones corresponden a distintas bandas o canales que componen la imagen satelital y visualmente se presentan en diferentes tonos grisáceos que simulan la intensidad de brillo.

El archivo en formato txt con mayor peso de almacenamiento, conocido como metadato, es necesario en el procesamiento de imágenes, ya que tiene la función de recopilar y organizar toda la información contenida en los archivos comprimidos, preparándola para un análisis posterior.

[4]Imagen tomada de *Fundamentos de la teledetección espacial*(p. 106) por Emilio Chuvieco.

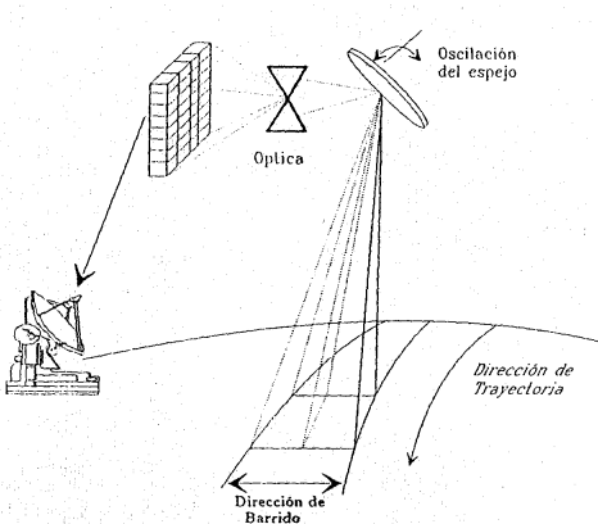


Figura 2.4: Proceso de captura de datos de la familia Landsat mediante radiómetro barredor[4]

De esta manera, la función del satélite es obtener energía necesaria para su funcionamiento a través de la absorción de las ondas reflejadas por cada objeto en la Tierra, para ello la contribución del sol es fundamental en este proceso. No obstante, hay otros factores que influyen en esta captación de datos, generando que los objetos en la imagen no sean claramente distinguibles, algunos de estos factores son la captación de nubes, presencia de neblina, entre otros. Por esta razón, es esencial llevar a cabo un preprocesamiento a la información de las imágenes satelitales.

2.2 Preprocesamiento de las imágenes satelitales

Al obtener el conjunto de datos con el que se va trabajar es importante adecuarlo para los fines del trabajo, en este caso el preprocesamiento consiste en mejorar la calidad de cualquier imagen satelital y prepararla adecuadamente para su análisis en las diversas aplicaciones que puedan surgir.

La obtención de información satelital está sujeta al sensor remoto y a la calidad de captura en la zona. Como se mencionó anteriormente, factores como las nubes o la neblina pueden interferir en la captura de las imágenes. Incluso, es posible que existan errores de captura, lo que puede causar problemas al vincular la información con sus coordenadas geográficas. Por tal motivo, se abordará el tema de la corrección radiométrica, el cual es un proceso esencial para ajustar las imágenes donde se eliminan o minimizan los errores y las distorsiones que puedan insertarse durante la captura de las imágenes.

Después de la corrección radiométrica, se aborda lo correspondiente a la reproyección, debido a que es un paso determinante en el preprocesamiento, ya que permite transformar las imágenes satelitales a un sistema de coordenadas común.

Posteriormente, se describe la creación de los índices, por ejemplo, los índices espectrales que permiten cuantificar ciertas propiedades de la imagen como la reflectancia de la superficie o la presencia de agua. Ambas características pueden ser útiles para la segmentación de suelos y respondería al objetivo general de este estudio: la identificación del tipo de suelo de "agricultura".

Por último, se discutirán los índices de textura que permiten cuantificar la variabilidad de una imagen, permitiendo observar el comportamiento que tienen los píxeles respecto al tipo de suelo.

Es importante mencionar que, en el procesamiento de información satelital, a menudo se requiere unir y recortar imágenes satelitales para analizar una ciudad completa, ya que una sola imagen puede no ser suficiente. Sin embargo, basándonos en la información disponible y a falta de datos adicionales, se procederá bajo el supuesto de que la ciudad de Harare puede ser representada adecuadamente mediante una única imagen satelital debido a su tamaño. Aunque la unión y el recorte de imágenes

para el estudio de una zona específica son parte del preprocesamiento de datos, en este caso no se realizó dicho proceso.

2.2.1 Corrección radiométrica

La corrección radiométrica emerge como un procedimiento fundamental destinado a calibrar los valores de los píxeles en una imagen. Tiene el propósito de eliminar las variaciones originadas por factores atmosféricos, condiciones lumínicas y características del sensor de captura. En muchas ocasiones los valores de los píxeles se encuentran expresados en números enteros, de ahí que se implemente una transformación que convierte el valor de cada píxel a valor flotante o real, a través del uso de dos parámetros:

- offset: empleado para subsanar errores constantes en las mediciones.
- gain: factor de escala que se aplica a cada píxel de la imagen para ajustar su contraste.

Ambos conceptos están preestablecidos en los satélites y se mantienen como constantes. Es esencial recalcar que estos valores pueden experimentar variaciones según el tipo de satélite considerado.

La ecuación para convertir los números digitales del satélite a radiancias en el satélite es:

$$Lsat = \frac{(ND - offset)}{Gain} \tag{2.1}$$

Donde Lsat es la radiancia espectral en cada Banda y ND es el número digital en el píxel.

Así las radiancias en el satélite son convertirdas a reflectancias de superficie corrigiendo tanto los efectos solares como atmosféricos. De acuerdo con Chavez (1996), la reflectancia espectral se define de la siguiente manera:

$$REF = \frac{PI * (Lsat - Lhaze)}{TAUv * (Eo * Cos(TZ) * TAUz + Bdown} \tag{2.2}$$

Donde

- REF = Reflectancia espectral de la superficie.
- Lhaze = Radiancia espectral atmosférica en la dirección del punto de entrada del sensor.
- TAUv = Transmitancia atmosférica a lo largo del trayecto entre la superficie del suelo y el sensor.
- Eo = Irradiancia espectral solar sobre una superficie perpendicular a los rayos del sol fuera de la atmósfera.
- TZ = Ángulo de incidencia del flujo solar directo sobre la superficie terrestre.
- Twuz = Transmitancia atmosférica a lo largo de la trayectoria desde el sol hasta la superficie del suelo.
- Edown = Irradiancia espectral descendente en la superficie.
- Lsat = Radiancia espectral del satélite para las bandas espectrales.

Los procedimientos de corrección radiométrica que se describirán a continuación se basan en la ecuación (2.2). Aunque los valores de los parámetros pueden variar dependiendo del modelo utilizado. Estos métodos no se discutirán a detalle en este trabajo, por lo que se sugiere que para tener mayor información se consulte a Chavez (1996, p.1027).

Generalmente el uso de estos modelos se hace de manera individual y la ayuda de expertos es crucial seleccionar algún modelo en específico. Para este trabajo se utilizó el modelo COSTZ aplicado en

Kamusoko (2019); sin embargo, se considera necesario describir brevemente los modelos APREF, SDOS con el fin de proporcionar un panorama completo de las opciones disponibles.

En programas como R o Python existen librerías donde la aplicación de los modelos APREF, SDOS y COSTZ se realizan mediante funciones que indican el método que se desean aplicar. Por ejemplo, la librería *RStoolbox* proporciona una amplia gama de funciones y herramientas para realizar tareas como el preprocesamiento, la extracción de información, la manipulación y el análisis de datos de imágenes remotas (Leutner et al., 2023)

2.2.1.1 Reflectancia Aparente (APREF)

Este método consiste en ajustar el brillo de la imagen en función de la proporción de luz que la superficie "devuelve" al ojo del observador, en comparación con la luz que "recibe" de la fuente de iluminación. Es importante mencionar que este método no corrige la dispersión ni la absorción atmosférica, sino que se enfoca en la corrección de la iluminación (Chavez, 2019).

2.2.1.2 Simple dark object subtraction (SDOS)

Este modelo asume que los objetos oscuros presentes en la imagen tienen un valor de reflectancia cero, quiere decir que no reflejan luz. Por lo tanto, cualquier valor cercano a cero se considera dispersión y este método se encarga de eliminar los píxeles más oscuros de la imagen (Chavez, 2019).

2.2.1.3 Cosine of the Solar Zenith Angle (COSTZ)

El modelo de corrección radiométrica COST es ampliamente utilizado para mejorar la calidad de las imágenes satelitales; permite eliminar las diferencias de luminosidad en las imágenes causadas por factores como la distancia al sol, la presencia de nubes o la reflexión de la luz en la atmósfera (Chavez, 2019).

2.2.2 Reproyección

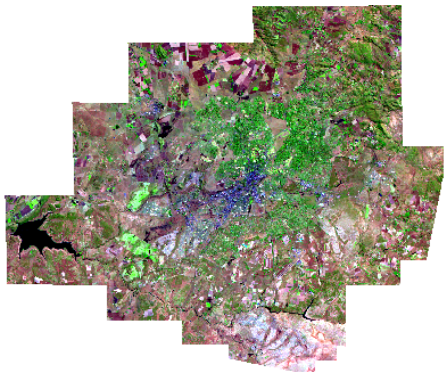
La reproyección es un proceso que transforma una imagen satélital de un sistema de coordenadas espaciales a otro distinto, lo que implica cambiar la proyección.

Durante este proceso se realiza un análisis en las coordenadas geográficas y el Sistema de Referencia de Coordenadas Geográficas (SRC) para garantizar su correcto registro, de esta manera se procede a enfocar la zona de estudio, por lo que es necesario hacer uso de archivos shapefile (*.shape*), que almacenan información geográfica y ubicación geométrica en forma de puntos (representan una ubicación específica para un par de coordenadas), líneas (comúnmente se utilizan para el análisis de ríos o carreteras) y polígonos (se utilizan en la delimitación de áreas territoriales).

El método de reproyección permite focalizar el análisis en la ciudad de estudio, mientras que a nivel de costo computacional se reduce la cantidad de observaciones, o píxeles útiles, debido a que no pertenecen al área territorial de interés.

2.2.3 Índices

Es determinante subrayar que si la meta primordial no consistiera en mejorar la resolución de la imagen satelital, sino en sólo trabajar con los datos derivados de la corrección radiométrica, se podrían calcular índices de textura e índices espectrales directamente. No obstante, considerando que el propósito del preprocesamiento es realzar la calidad de la imagen satelital, existe la posibilidad de generar variables adicionales tras haber rectificado la naturaleza intrínseca de la información, a través



(A) Aplicación del archivo shapefile de polígono

(B) Delimitación territorial de la Ciudad de Harare.

Figura 2.5: En A) se muestra la aplicación del archivo shapefile de polígono y en B) se presenta la delimitación territorial de la zona de la ciudad de Harare. El archivo shape trata de acoplarse lo mejor posible a la delimitación roja.

Indice Espectral	Bandas	Definición
NDVI	Roja(banda 3) y NIR(banda 4)	$\frac{-Roja}{NIR+Roja}$
SAVI	Roja(banda 3) y NIR(banda 4)	$\frac{(NIR-Roja)(1+L)}{NIR+Roja+L}$
MSAVI	Roja(banda 3) y NIR(banda 4)	$NIR + 0.5 - (0.5\frac{\sqrt{(2NIR+1)^2-8(NIR-(2Roja))}}{2})$

Tabla 2.3: Índices espectrales: los índices espectrales como NDVI, SAVI y MSAVI se encuentran entre -1 y 1 indicando el nivel de vegetación donde a -1 indica poca vegetación y a 1 indica lo contrario.

de la corrección radiométrica y la reproyección. Este proceso puede facilitar la construcción de algoritmos para la identificación de coberturas de suelo, permitiendo una clasificación más precisa de los datos.

2.2.3.1 Índices espectrales

Los índices espectrales proporcionan información valiosa sobre la salud de la vegetación, la cantidad de agua presente en un área, entre otros aspectos del terreno. Estos índices se calculan mediante diferentes combinaciones de las bandas espectrales disponibles en la imagen satelital. Algunos ejemplos comunes de índices son: el NDVI, que ayuda a diferenciar y determinar el estado de la vegetación; el SAVI que evita la influencia del suelo en la vegetación; y el MSAVI, útil para áreas con escasa vegetación. Para mayor información consultar (Lillesand et al., 2015, p. 520-522).

Para realizar cálculos precisos entre bandas es necesario que estén correctamente alineadas en el espacio. En otras palabras, cada píxel en una banda debe corresponder al mismo punto geográfico de las demás bandas.

La Tabla 2.3 la ecuación para construir los índices espectrales donde intervienen las bandas 3 y 4. El cálculo de estos índices se realiza por píxel para determinar las características de la vegetación. Aunque existen otros índices espectrales que pueden requerir de la intervención otras bandas, en este trabajo se usarán únicamente en los índices que se describen en la Tabla 2.3 .

La librería *RStoolbox*(Leutner et al., 2023) permite simplificar el cálculo de los índices espectrales descritos en la Tabla 2.3 entre otros. Sólo se debe de especificar las bandas y el índice espectral que se desea calcular.

2.2.3.2 Índices de textura

La textura en el contexto de las imágenes se refiere a la distribución de las tonalidades de intensidad de brillo en los píxeles vecinos, esta característica permite identificar regiones en la imagen satelital. Para este trabajo, se adoptó la idea de Kamusoko (2019, p.68) con patrones donde se utiliza la Matriz de Co-ocurrencia de Niveles de Gris (Grey Level Co-occurrence Matrix, GLCM) para algunas bandas de la imagen satelital.

La GLCM describe la frecuencia con la que una cierta intensidad de brillo aparece junto a sus píxeles vecinos dentro de una ventana determinada. La ventana móvil define la cantidad de píxeles a usar, ver Figura 2.6. Presutti (2004, p.2) señala lo siguiente:

> "Respecto del tamaño de la ventana, ésta debe ser cuadrada y con número impar de píxeles. El resultado del cálculo de la textura es un único número que representa la ventana completa, el cual es colocado en el lugar del píxel central. Luego, la ventana se mueve un píxel y el cálculo se repite calculando una nueva matriz de co-ocurrencia para esta nueva ventana y resultando un nuevo valor para el píxel central de esta nueva posición de la ventana."

Cuando la ventana no considera un número impar de píxeles, se asigna el valor de textura a uno de los píxeles centrales.

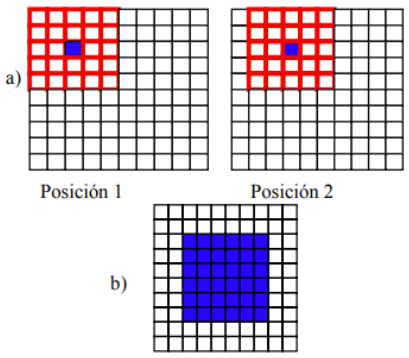


*Figura 2.6: En **a)** se muestra una imagen de dimensión 10x10, mientras que su ventana es de dimensión 5x5. El píxel central recibe el resultado de los cálculos (varianza, media, entropia, etc.). En **b)** se muestra el resultado una vez que la ventana móvil ha pasado por todos estos píxeles que se toman como referencia, mientras que las columnas y filas en el borde al no asignar ningún valor debido a la construcción de la GLCM reciben los valores de sus vecinos más cercanos.*[5]

Un vez definida la ventana, se analizan los L niveles de grises que conviven en dicha ventana generando la matriz co-ocurrencia de $L \times L$:

$$M_{L\times L} = \begin{bmatrix} p(0,0) & \cdots & p(0,L-1) \\ \vdots & \ldots & \vdots \\ p(L-1,0) & \cdots & p(L-1,L-1) \end{bmatrix} \tag{2.3}$$

$p(i,j)$ se define como el número de veces que una pareja de pixeles con intensidad i, j, ocurren en una ventana móvil dado una dirección especifica d , donde $1 \leq i,j \leq L$. La dirección define la manera de comparar un píxel con otro y puede ser: horizontal, vertical, etcétera, ver Figura 2.7. De acuerdo a Presutti (2004, p.4), el cálculo de la matriz de co ocurrencia "se logra si cada par de pixeles se cuentan dos veces: una vez a la derecha y otra vez a la izquierda (se intercambian los pixeles de referencia y vecino en el segundo calculo)"

[5]Imagen obtenida de *La matriz de co-ocurrencia en la clasificación multiespectral* por Presutti Miriam.
[6]Textural Features for Image Classification (Haralick, Shanmugam, & Dinstein, 1973, p.612)

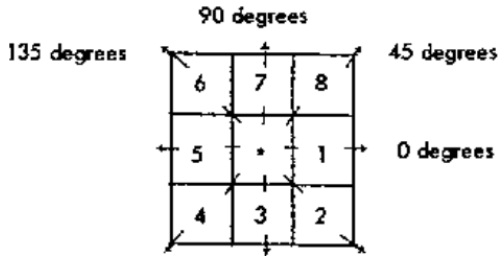
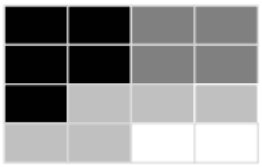


*Figura 2.7: Los grados señalan la dirección en que se calcularán los valores de textura. Las celdas 1 y 5 están en dirección horizontal ($0°$) con respecto a la celda central *; las celdas 4 y 8 en diagonal a $45°$; 7 y 3 en vertical ($90°$) y 6 y 2 en diagonal a $135°$.* [6]



$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix}$$

Figura 2.8: Ejemplo de representación de una ventana móvil de dimensión 4x4 con 4 niveles de intensidad de brillo.

Por ejemplo: en la Figura 2.8 se presenta una matriz ventana de $4x4$, donde los píxeles sólo presentan cuatro intensidades de brillo: 0,1,2,3. En este caso la matriz de co-ocurrencia es de $4x4$. Si se define la dirección horizontal con $d = 1$, la matriz de co-ocurrencia se presenta en la matriz (2.4):

$$M = \begin{bmatrix} p(0,0) & p(0,1) & p(0,2) & p(0,3) \\ p(1,0) & p(1,1) & p(1,2) & p(1,3) \\ p(2,0) & p(2,1) & p(2,2) & p(2,3) \\ p(3,0) & p(3,1) & p(3,2) & p(3,3) \end{bmatrix} = \begin{bmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \tag{2.4}$$

Por ejemplo, en la matriz M, $p(0,1)$=2+0, ya que hay 2 píxeles que tienen intensidad 0 y que a su derecha ($0°$), a una distancia de 1, hay un píxel con intensidad 1. Nótese que no hay píxeles que tengan intensidad 0 y a la izquierda ($0°$) haya un píxel con intensidad 1 a una distancia de 1. Ver Figuras 2.9 A) y B) en donde se ponen en amarillo los píxeles con intensidad 0, y de los cuales se encierran en un rectángulo verde los que tienen, Figura 2.9 A), a su derecha o, Figura 2.9 B), a su izquierda a un píxel con intensidad 1.

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix}$$

(A)

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix}$$

(B)

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix}$$

(C)

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix}$$

(D)

Figura 2.9: Píxeles que tienen una intensidad 0 (A, B) o 1 (C, D). De estos, en rectángulo verde los que a su derecha (A, C) o izquierda (A, B) tienen un píxel con intensidad 1 a una distancia 1.

Otra instancia es $p(1,1) = 2 + 2 = 4$, ver Figuras 2.9 C) y D), en donde la representación de cuadros

amarillos corresponden a todos los píxeles de intensidad 1, los cuadros que se encuentran de color verde son aquellos que tienen una distancia 1 y una intensidad 1.

Para el cálculo de probabilidades se realiza la división de cada elemento de la matriz M entre la suma total de todos los elementos de la matriz, que en este caso es 24, lo que da como resultado:

$$P = \begin{bmatrix} 0.166 & 0.083 & 0.041 & 0 \\ 0.083 & 0.166 & 0 & 0 \\ 0.041 & 0 & 0.25 & 0.041 \\ 0 & 0 & 0.041 & 0.083 \end{bmatrix}. \tag{2.5}$$

Finalmente, con P se calculan las medidas de de textura que son definidas a continuación:

- Media: El valor de un píxel es ponderado por la frecuencia de su co-ocurrencia en combinación con un determinado valor del píxel vecino. Dado que la matriz $M_{L\times L}$ fue construida de acuerdo con la definición dada en (**??**), la media se puede calcular de la siguiente manera:

$$\mu = \sum_{j=0}^{L-1}\sum_{i=0}^{L-1} iP_{i,j} = \sum_{j=0}^{L-1}\sum_{i=0}^{L-1} jP_{i,j} \tag{2.6}$$

- Varianza: Esta medida aumenta cuando la intensidad de brillo de un píxel difiere significativamente de la media de las intensidades de brillo en la ventana. Una varianza alta indica una textura más rugosa o variada. Dado que la matriz es simétrica, la varianza se puede calcular como:

$$\sigma^2 = \sum_{j=0}^{L-1}\sum_{i=0}^{L-1} P_{i,j}(i-\mu_i)^2 = \sum_{j=0}^{L-1}\sum_{i=0}^{L-1} P_{i,j}(j-\mu_j)^2 \tag{2.7}$$

- Homogeneidad: Esta medida indica la similitud de la imagen digital. Una homogeneidad alta sugiere que la imagen tiene una textura uniforme. Se calcula como:

$$\sum_{j=0}^{L-1}\sum_{i=0}^{L-1} \frac{P_{i,j}}{1+(i-j)^2} \tag{2.8}$$

- Entropía: Esta medida cuantifica la aleatoriedad de la distribución de intensidad. Una entropía alta resalta cuando la imagen no tiene una textura uniforme, indicando una mayor complejidad o desorden en la textura de la imagen. Se calcula como:

$$\sum_{j=0}^{L-1}\sum_{i=0}^{L-1} -P_{i,j}ln(P_{i,j}) \tag{2.9}$$

Siguiendo con el ejemplo, el cálculo de la homogeneidad queda de la siguiente manera:

$$\begin{aligned} &\frac{0.166}{1+(0-0)^2} + \frac{0.083}{1+(0-1)^2} + \frac{0.041}{1+(0-2)^2} + \frac{0}{1+(0-3)^2} \\ &+ \frac{0.083}{1+(1-0)^2} + \frac{0.166}{1+(1-1)^2} + \frac{0}{1+(1-2)^2} + \frac{0}{1+(1-3)^2} \\ &+ \frac{0.041}{1+(2-0)^2} + \frac{0}{1+(2-1)^2} + \frac{0.25}{1+(2-2)^2} + \frac{0.042}{1+(2-3)^2} \\ &+ \frac{0}{1+(3-0)^2} + \frac{0}{1+(3-1)^2} + \frac{0.041}{1+(3-2)^2} + \frac{0.083}{1+(3-3)^2} = 0.807 \end{aligned}$$

Para calcular la media como:

$$\mu = (0*0.166)+(1*0.083)+(2*0.041)+(3*0)+(0*0.083)+(1*0.166)+(2*0)+(3*0)+(0*0.041)+$$

$$(1*0)+(2*0.25)+(3*0.041)+(0*0)+(1*0)+(2*0.041)+(3*0.083) = (0*0.166)+(1*0.083)+(2*0.041)+$$

$$(3*0)+(0*0.083)+(1*0.166)+(2*0)+(3*0)+(0*0.041)+(1*0)+(2*0.25)+$$

$$(3*0.041)+(0*0)+(1*0)+(2*0.041)+(3*0.083) = 1.287$$

De modo que se asigna el valor de homogeneidad y media a uno de los píxeles centrales; por ejemplo, al píxel con coordenadas (2,2) en la Figura 2.8, se le asigna el valor 0.807 en homogeneidad y 1.287 en la media, debido a su pertenencia a los píxeles centrales. Sin embargo, se recomienda que el número de píxeles para definir la ventana sea impar.

Cabe señalar que en el lenguaje de programación R existe la función *glcm* de la librería *glcm* (Zvoleff, 2022) que permite calcular valores anteriores; el argumento que viene por default es la dirección a 45°.

Para más detalles sobre la matriz de co-ocurrencia, la métricas a calcular y sus aplicaciones, ver Haralick et al,. (1973) y Presutti, (2004).

2.3 Obtención de la variable sobre tipo de cobertura de suelo

La segmentación de cobertura es un proceso complejo, ya que se basa en características o patrones que definen grupos de pertenencia.

El primer paso en este proceso es la generación de diversas variables a partir de las bandas espectrales contenidas en archivos comprimidos, como la corrección radiométrica y creación de índices. Este paso puede ser realizado mediante imágenes satelitales disponibles en sitios web como USGS, Earth Explorer o LANDViewer.

Sin embargo, la auténtica complejidad radica en la creación u obtención de la variable objetivo **Y** que representa el tipo de suelo en una zona geográfica específica y es esencial para aplicar modelos de clasificación supervisada. La variable **Y** actúa como guía para definir reglas que permitan predecir la clase de las nuevas observaciones.

La creación de la variable objetivo **Y** es un proceso que implica la interpretación visual de imágenes satelitales y la asignación manual de etiquetas de cobertura del suelo a cada ubicación o píxel de la imagen satelital. Este trabajo es realizado por especialistas en teledetección, ciencias de la tierra o personas con conocimiento a priori de la zona en cuestión. Estos especialistas poseen conocimientos específicos sobre las características de la superficie terrestre, las clases de cobertura del suelo y la interpretación de imágenes satelitales. La creación de esta variable implica costos tanto económicos como computacionales, además de un tiempo prolongado de evaluación. En este trabajo, la variable **Y** debe definirse como: "Agr" o "NoAgr".

Cada píxel en las imágenes satelitales representa una observación en los datos y abarca un área específica. En imágenes gratuitas generalmente un píxel está representado en metros cuadrados. Dado que una imagen contiene millones de píxeles, un número considerable de ellos deben estar definidos de antemano para entrenar un modelo presentando un enorme desafío.

Existen diversas formas de definir esta variable, cada una con sus propios desafíos y costos. Por ejemplo, en el caso de México la definición se puede basar en las etiquetas proporcionadas por las series del INEGI, pero éstas tienen una escala de 1:250 m, lo que puede llevar a errores significativos de etiquetado debido a su escala y la resolución espacial de la imagen satelital. Otra opción es contar con expertos en imágenes satelitales y cobertura de suelo, pero la subjetividad de cada experto en la determinación de la clase puede introducir errores.

Un enfoque alternativo implica la generación de modelos de muestreo que permitan seleccionar zonas representativas de las áreas de interés. Esto limitará el trabajo de etiquetado. A pesar de ello, es necesario contar con personas especializadas para crear las etiquetas del tipo de suelo en las áreas seleccionadas. Sin embargo, este enfoque también enfrenta la subjetividad de los expertos, aunado a la selección y a la forma de analizar cuando el tipo de suelo difiere en el mismo píxel.

También se pueden generar modelos de muestreo y enviar a las zonas seleccionadas brigadas especializadas en el estudio de campo para determinar el tipo de suelo. Las brigadas deben cubrir la resolución espacial que abarca un píxel, por ejemplo, si la imagen proviene del Landsat, la brigada debe estudiar un área de 30 metros cuadrados lo que representa solamente un píxel. Además, se debe considerar el tiempo de captura para que la imagen pueda coincidir con la etiqueta. Hacer esto para varios píxeles que posteriormente sean utilizados en el entrenamiento de los modelos disminuye el error de etiquetado, pero aumenta el costo económico y el tiempo de etiquetado.

Además, se debe considerar la existencia de imágenes satelitales de la zona y su tiempo de captura, así como factores naturales y estacionales que afecten el tipo de suelo. Este proceso debe repetirse cada vez que el suelo cambie, pues, los modelos de clasificación deben ser reentrenados. La resolución de la imagen es un factor crucial, ya que determina el tamaño del píxel y el área cubierta. Estas consideraciones, entre otras, hacen que la tarea de crear las etiquetas del tipo de suelo sea demasiada difícil y costosa en general.

Las etiquetas que se usan en este trabajo, de acuerdo con Kamusoko (2013) tienen la siguiente característica *"..based on the Forestry Commission (Zimbabwe) and the Surveyor-General national cover classes classification schemes as well as the author's a priori knowledge of the study area."*. Esto resulta en una muestra de datos de alrededor de 7 mil observaciones o píxeles, a pesar de que la imagen original contenía solo un total de 4 millones de píxeles.

CAPÍTULO 3

Metodología y algoritmos de clasificación

3.1 Clasificación supervisada

La clasificación se refiere al proceso de agrupar observaciones o elementos en categorías que suelen compartir atributos similares. Las categorías no siempre están predefinidas, cuando se establecen previamente se utilizan métodos supervisados con el objetivo de entrenar un modelo que pueda asignar automáticamente nuevas observaciones a estas categorías preestablecidas. Por otro lado, en la clasificación no supervisada, donde no hay información previa sobre las categorías de los elementos, el objetivo es agrupar a los elementos en conjuntos (clusters o conglomerados) basándose en medidas de disimilaridad. Este trabajo está enfocado en métodos de clasificación supervisada.

En la clasificación supervisada, se crea una regla que permite identificar el grupo al que pertenecen nuevas observaciones. Esta regla se ajusta utilizando las n observaciones disponibles en el conjunto de datos, es decir, $(y_1, \mathbf{x}_1), \ldots, (y_n, \mathbf{x}_n)$, donde y_i representa la etiqueta o grupo al que pertenece la i-ésima observación, y $\mathbf{x}_i$ son las características asociadas a dicha observación, con $i = 1, 2, ..., n$. Una vez creada la regla de clasificación, surge el interés de predecir los valores $y_{h_1}, ..., y_{h_m}$ de nuevas observaciones asociadas a $\mathbf{x}_1, ..., \mathbf{x}_m$, respectivamente. Esto se logra contrastando los valores reales de $y_{h_1}, ..., y_{h_m}$ con los valores predichos.

Sin embargo, aplicar la regla de clasificación a nuevas observaciones y comparar los resultados reales con los predichos podría implicar grandes pérdidas o costos de cualquier índole, por lo que esta estrategia es poco común.

A continuación, se describe una alternativa para aplicar la regla de clasificación con mayor base y fiabilidad.

3.2 Estimación del poder predictivo

La estimación del poder predictivo parte de la idea de que, una vez que se ha ajustado una regla de clasificación, es importante analizar si servirá para clasificar nuevas observaciones. Para esto y cuando n es pequeña se emula el proceso con el que se ajusto la regla de clasificación dividiendo las n observaciones originales en dos subconjuntos: el conjunto de entrenamiento (train), con el que se ajusta la regla de clasificación, y el conjunto de prueba (test), que representa las nuevas observaciones.

Dado que el conjunto de prueba trata de emular las nuevas observaciones no sólo se puede evaluar la regla obtenida con n observaciones usando las n_{test} observaciones. Se debe ajustar la regla con $n_{train} < n$ observaciones para que la regla ajustada con n_{train} observaciones se parezca a la ajustada con n,se debe de repetir por completo todo el proceso de ajuste, pero sólo considerando las n_{train} observaciones. Esto asegura las n_{test} observaciones son "nuevas" al momento de evaluar la regla calculada con n_{train} observaciones. Por lo general el conjunto de entrenamiento es de 80% o 90%.

No obstante, el uso del 10% o 20% de las observaciones para el conjunto de prueba podría no captar todas las características que puedan surgir en nuevas observaciones en la práctica. Para mejorar la

estimación del poder predictivo se recurre a métodos de remuestreo que permiten tener diferentes conjuntos y emular las posibles nuevas observaciones, es decir, este proceso se repite B veces.

La medición del poder predictivo se basa en métricas como la exactitud (Accuracy), la precisión (Recall), especificidad (Especificity), todas calculadas aplicando la regla ajustada con el conjunto de train a todas las observaciones del conjunto test.

El cálculo de estas métricas se hace a través de la matriz de confusión, la cual proporciona una visión de cómo un modelo está clasificando nuevas observaciones. Cuando la variable **Y** solamente tiene dos categorías, una puede considerar como positiva y otra como negativa. La matriz organiza las predicciones del modelo en cuatro categorías distintas, ver Tabla 3.1.

		Categoría predicha	
		Positiva	Negativa
Categoría observada	Positiva	Verdadero Positivo (VP)	Falso Negativo (FN)
	Negativa	Falso Positivo (FP)	Verdadero Negativo (VN)

Tabla 3.1: Matriz de confusión para el caso de clasificación binaria

Donde cada celda de la Tabla 3.1 corresponde a:

- Verdadero Positivo (VP): Número de observaciones positivas que fueron clasificadas correctamente.
- Falso Positivo (FP): Número de observaciones negativas que el modelo predijo incorrectamente, es decir, eran negativas y fueron clasificadas como positivas.
- Falso Negativo (FN): Número de observaciones positivas que el modelo predijo incorrectamente, es decir, eran positivas y fueron clasificadas como negativas.
- Verdadero Negativo (VN): Número de observaciones negativas que fueron clasificadas correctamente.

Algunos ejemplos de métricas para medir el desempeño de la clasificación son las siguientes:

1. Exactitud (Accuracy): Es el porcentaje de clasificación correcta global.
   - $Exactitud = \dfrac{VP + VN}{VP + FP + FN + VN}$
2. Sensibilidad (Recall): Es el porcentaje que se logra clasificar de forma correcta del grupo positivo.
   - $Sensibilidad = \dfrac{VP}{VP + FN}$
3. Especificidad (Especificity): Es el porcentaje que se logra clasificar de forma correcta del grupo negativo.
   - $Especificidad = \dfrac{VN}{VN + FP}$

Estas métricas se calculan para cada partición y promedian los resultados de las B repeticiones, lo que permite evaluar el desempeño de la regla de clasificación.

A continuación se describen los métodos de remuestreo más comunes.

3.2.1 Repeated Hold-Out

El método Hold-Out que involucra la división del conjunto de datos en dos grupos: entrenamiento (*train*) y prueba (*test*), ver Figura 3.1. Para obtener una estimación más robusta de poder predictivo, la división se repite B veces y se calcula el promedio de las métricas de desempeño obtenidas en cada una de las iteraciones. Al repetir el proceso varias veces con diferentes particiones aleatorias de los

datos, se obtiene una estimación del rendimiento que es menos sensible a la elección específica de los conjuntos de entrenamiento y prueba, reduciendo la variabilidad de la estimación del poder predictivo.

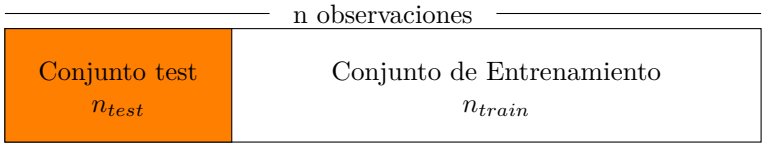


Figura 3.1: Ejemplo de la división en conjunto de entrenamiento y test, ambos subconjuntos pertenecen a nuestros datos reales o históricos ($n = n_{train} + n_{test}$).

3.2.2 K-Cross Validation

El método de validación cruzada parte de la idea del método Hold-out (dividir nuestros datos en dos subconjuntos), pero directamente se particiona el conjunto de datos en K subconjuntos de igual tamaño llamado pliegues o folds. De esta manera se realizan K rondas de entrenamiento y prueba, en cada ronda un pliegue se utiliza como conjunto de prueba y los K-1 pliegues restantes son utilizados como conjuntos de entrenamiento, garantizando que todos los datos van a ser considerados una vez en el conjunto test. Al final, se promedian los resultados obtenidos en las K iteraciones para obtener una medida general del rendimiento del modelo, ver Figura 3.2.

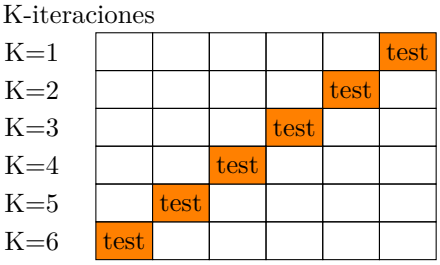


Figura 3.2: Ejemplo del método K Cross Validation, con K=6, donde el conjunto de datos se divide en conjunto de entrenamiento (cuadros blancos) y en conjunto de test (cuadros naranjas) en cada una de K rondas.

De la misma manera que en Hold-Out, este método puede ser aplicado varias veces, lo que se conoce como Repeated K-Cross Validation.

Calibración tuneo de hiperparámetros

Los métodos modernos de clasificación supervisada requieren la definición de hiperparámetros para ajustar las reglas de clasificación. El objetivo del proceso conocido como calibración o ajuste de hiperparámetros es identificar la combinación óptima de valores de hiperparámetros que minimice una métrica de error de predicción. Por lo general, el ajuste se realiza en el subconjunto de entrenamiento.

Por ejemplo, si un algoritmo de clasificación utiliza dos hiperparámetros y se decide probar tres valores diferentes para cada uno, se crea una malla de 2x3, es decir, seis combinaciones posibles de hiperparámetros. Para cada combinación, se entrena el modelo y se calcula el error en el conjunto de validación. La combinación con el menor error se selecciona como la mejor para el modelo. Este proceso se repite varias veces para garantizar la robustez de los resultados y elegir así el conjunto de hiperparámetros.

Una vez definidos los mejores hiperparámetros, se ajusta el modelo utilizando el subconjunto de entrenamiento y luego se evalúa su rendimiento con el conjunto de prueba, como se muestra en la Figura 3.3. Este proceso se repite B veces y, al final, se calcula la media de las estimaciones del poder predictivo.

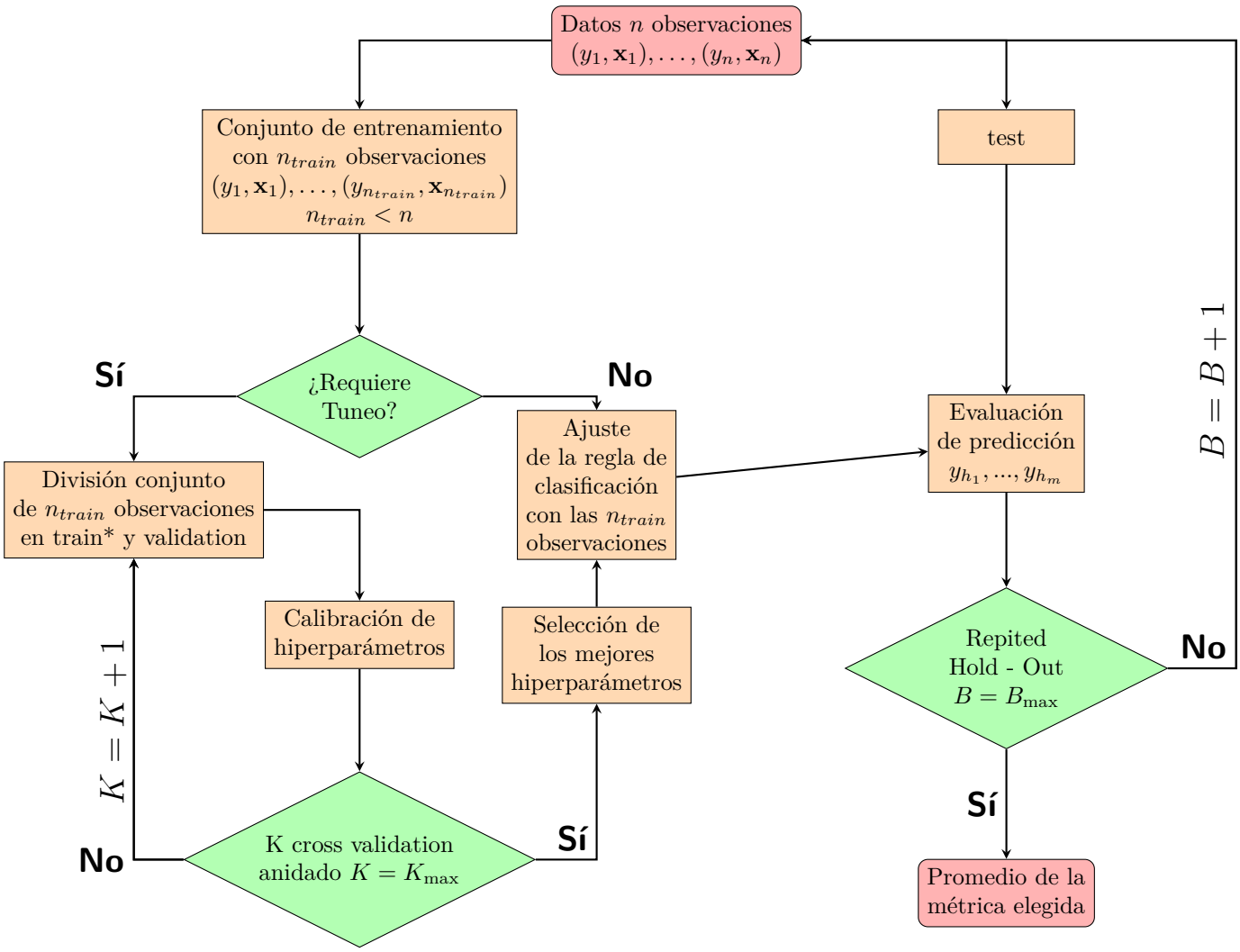


Figura 3.3: Proceso de estimación del poder predictivo en este trabajo, aplicado a dos casos distintos: uno en el cual existe una malla de valores a tunear y otro en el que no requiere tuneo. En color rojo representa la entrada y salida de información que necesita para el cálculo del poder predictivo.

En la Figura 3.3 se observa el flujo de la estimación del poder predictivo realizado en este trabajo:

1. El conjunto de datos de n observaciones se divide en dos subconjuntos: train y test, con n_{train} y n_{test} observaciones respectivamente. Se utiliza el método Repeated Hold-out para calcular el poder predictivo, repitiendo el proceso 100 veces. Además, se aplica K-Fold Cross Validation con K=10 para ajustar los hiperparámetros.
2. Si el método de clasificación requiere ajuste de hiperparámetros, se procede de la siguiente manera:
   (a) El subconjunto train se divide nuevamente en dos partes: train* y validation.
   (b) Se prueban todas las combinaciones posibles de los hiperparámetros dentro de la malla definida utilizando el conjunto de train*.
   (c) Para cada combinación, se evalúa el rendimiento con el conjunto de validation y se estima el error.
   (d) Este proceso se repite 10 veces, ya que K=10.
   (e) Finalmente, se selecciona la combinación de hiperparámetros que ofrece el mejor rendimiento.
3. Si no se requiere ajuste de hiperparámetros, se entrena el modelo utilizando las n_{train}

observaciones. Si se ha realizado el ajuste de hiperparámetros, el entrenamiento se realiza con la mejor combinación de hiperparámetros seleccionados a partir de K-Fold Cross Validation.

4. El modelo entrenado se evalúa utilizando el conjunto de test para evaluar su capacidad predictiva.
5. El proceso desde el punto 1 hasta el punto 4 se repite 100 veces, variando las particiones de los subconjuntos train y test.
6. El resultado final se toma como una estimación "real" de la capacidad de predicción del modelo o regla aplicada. Este resultado se utiliza para compararse con otras reglas de clasificación

Algunas consideraciones que se deben de tomar en cuenta son las siguientes: si se pretenden usar varios modelos de predicción y se quieren comparar, estos deben de ser tratados con el mismo proceso de remuestreo que se aplica en todos. Este proceso resulta costoso computacionalmente para algunos modelos donde se debe tunear, dado el tamaño de muestra, la cantidad de hiperparámetros y los valores que se asignen en las mallas.

3.3 Métodos para construir reglas de clasificación

3.3.1 Regresión Binaria

Cuando la variable de interés presenta solo dos opciones (0 o 1, sí o no, positivo o negativo, etc.) se busca predecir su valor usando un conjunto de características $x_1 \dots x_p$, llamadas covariables o variables independientes. De esta manera, en la regresión binaria la predicción se lleva a cabo mediante modelos probabilísticos que buscan modelar:

$$\mu = \mathrm{E}(y; x_1, \dots, x_p) = \mathrm{P}(y = 1; x_1, \dots, x_p), \text{ donde } y \sim Bernoulli(\mu). \tag{3.1}$$

En estos métodos la relación entre la variable y y $x_1, x_2, ..., x_p$ se modela usando un componente lineal dado por

$$\eta_i = \eta(\boldsymbol{\beta}, \mathbf{x}_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}, \tag{3.2}$$

donde η_i hace referencia a la combinación lineal de los parámetros del modelo que incluye como constantes a los valores de las covariables de la observación i-ésima.

Dado que se busca modelar una probabilidad, los valores deben estar entre cero y uno, sin embargo, el rango de variación del componente lineal no necesariamente está dentro de estos valores, incluso podría ser negativo. De esta manera se busca una función $g(.)$, conocida como como función liga, monótona y diferenciable tal que:

$$g(\mu_i) = \eta_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}, \tag{3.3}$$

$$\text{con } g^{-1}(\eta_i) \in (0, 1).$$

Suponiendo $y_i \sim Bernoulli(\mu_i)$ e independencia, la verosimilitud de la muestra de tamaño n es:

$$L(\mu_i) = \prod_{i=1}^{n} \mu_i^{y_i} (1 - \mu_i)^{1 - y_i}, \tag{3.4}$$

Función liga	$g(\mu_i) = \eta_i$	$g(\eta_i)^{-1} = \mu_i$
logit	$ln(\frac{\mu_i}{1-\mu_i})$	$\frac{exp(\eta_i)}{1+exp(\eta_i)}$
probit	$\phi^{-1}(\mu_i)$	$\phi(\eta_i)$
Log-log	$-ln(-ln(\mu_i))$	$exp(-exp(\eta_i))$
C-Log-log	$-ln(-ln(1-\mu_i))$	$1-exp(-exp(\eta_i))$

Tabla 3.2: Algunos ejemplos de funciones liga para modelos de regresión binaria son:

y la log-verosimilitud es

$$ln(L(\mu_i)) = \sum_{i=1}^{n}[y_i ln(\mu_i) + (1-y_i)ln(1-\mu_i)]. \tag{3.5}$$

En Tabla 3.2 se muestran algunos ejemplo de las funciones ligas más comunes para el caso de $\mathbf{Y}$ binaria.

Para la estimación de las $\beta' s$ se considera la función liga, dereiva e iguala a cero.

La regresión logística es uno de los métodos de clasificación más conocidos en el aprendizaje automatizado y se obtiene aplicando la liga logit en (3.3) de donde (3.5) se convierte en

$$ln(L(\mu_i)) = \sum_{i=1}^{n}\left\{y_i\Big[\frac{exp(\eta_i)}{1+exp(\eta_i)}\Big] + (1-y_i)ln\Big[1-\frac{exp(\eta_i)}{1+exp(\eta_i)}\Big]\right\} \tag{3.6}$$

$$= \sum_{i=1}^{n}\left\{y_i\Big[\eta_i - ln(exp(\eta_i)+1)\Big] + (1-y_i)\Big[-[ln(exp(\eta_i)+1)]\Big]\right\} \tag{3.7}$$

$$= \sum_{i=1}^{n}\Big\{y_i\eta_i - ln(exp(\eta_i)+1)\Big\}. \tag{3.8}$$

En este caso, se busca maximizar la ecuación (3.8) para encontrar los valores estimados, $\hat{\beta}' s$, de los $\beta' s$. Para maximizar o resolver el sistema obtenido fijando las derivadas con respecto a β como cero, se aplican métodos numéricos como el de Newton Raphson.

Además, se pueden utilizar métodos que ayuden a seleccionar variables, esperando que mejore la predicción de nuevas observaciones. Algunos métodos para selección de variables son los métodos por pasos (stepwise) y los métodos de regularización (lasso). Estos permiten identificar las variables más relevantes para el modelo de acuerdo con un criterio.

Una vez estimado un modelo, para predecir nuevas observaciones se ingresan los valores de las variables independientes en el modelo ajustado, y este proporciona una predicción de la probabilidad para cada uno de los dos grupos. Por ejemplo, en la regresión logística:

$$\hat{P}(y=1) = \frac{1}{1+exp^{-(\hat{\beta}_0+\hat{\beta}_1 x_1+\hat{\beta}_2 x_2+\ldots+\hat{\beta}_n x_p)}}, \tag{3.9}$$

donde:

$\hat{P}(y=1)$ es la probabilidad estimada de pertenecer al grupo 1,

$\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p$ son los coeficientes estimados del modelo,

$x_1, x_2, \ldots, x_p$ son los valores de las variables independientes en una nueva observación,

$\hat{P}(y=1)$ es un valor entre 0 y 1, de esta manera para que la observación pueda ser clasificada se introduce el concepto del punto de corte (C): es un umbral para asignar observaciones a una de las dos categorías, es decir, si $\hat{P}(y=1) > C$ se asigna al grupo 1 o si $\hat{P}(y=0) \leq C$ se asigna al grupo 2. La selección del punto de corte C es de vital importancia, ya que impacta en el desempeño de la regla de clasificación. Por lo general el punto de corte es 0.5, sin embargo, puede variar dependiendo de los resultados que se busquen y este valor puede ser modificado manualmente o también bajo otro criterio calculdao de los datos de entrenamiento, por ejemplo, maximizando el criterio de Youden (1950) definido a como:

$$Cyouden = Sensitividad + Especificidad - 1,$$

de esta manera se busca el punto de corte que optimice tanto la capacidad del modelo para detectar los verdaderos positivos como para evitar los falsos positivos.

Para temas relacionados sobre la regresión binaria y uso de la regresión lineal generalizada, ver (Agresti, A. Capítulo 4-5, 2015)

Las librerías o paquetes utilizadas para implementar modelos de regresión binaria en este trabajo son las siguientes:

glmnet: es un paquete en R que se utiliza para ajustar modelos lineales geralizados con penalización lasso y ridge. Aunque sólo se utilizará lasso (Friedman, et al., 2023).

pROC: es una herramienta diseñada para trabajar con curvas ROC (Receiver Operating Characteristic) y análisis de rendimiento de pruebas diagnósticas. Se utilizará para encontrar el punto de corte (Robin et al., 2023).

En entornos de programación como R, pROC, facilita a la selección del punto de corte óptimo para modelos de regresión binaria.

Cabe mencionar que la funcion *glm* se utiliza para ajustar modelos lineales generalizados y la función *step* se utiliza para la selección de variables con el método stepwise, donde las variables se agregan o eliminan del modelo en función de criterios predefinidos, como AIC o BIC. Ambas funciones son parte de la librería *"base"* de R. A continuación se detalla lo relacionado a la selección de variables.

Métodos por pasos (stepwise)

Estos métodos comparan un criterio de optimización como el AIC o BIC, al agregar (forward) o quitar (backward) una variable en el modelo de regresión en cada paso, también se puede combinar ambos métodos (both). Estos algoritmos trabajan con modelos anidados, es decir, están contenidos considerando un paso subsecuente.

Los métodos por pasos facilitan encontrar un modelo parsimonioso que a la vez proporcione un buen ajuste al modelo. El procedimiento para el método forward generalmente comienza con un modelo sin variables predictoras, mientras que el backward incluye todas las variables disponibles. A partir de este punto, se realizan iteraciones en las que se agrega o elimina una variable hasta llegar al modelo con mejor ajuste según un criterio establecido como el AIC o BIC. Una interacción se produce cuando el impacto de una variable predictora en la variable objetivo cambia según los valores de otra variable predictora.

Para ejemplificar el procedimiento, usaremos la base Diabetes in Pima Indian Women incluida en la librería *MASS*(Venables, 2002). Es un conjunto de datos sobre mujeres indígenas de la tribu Pima ubicada en el suroeste de los Estados Unidos, la cual tiene una de las tasas más altas de diabetes tipo 2 en el mundo, la base esta compuesta por 7 variables descritas a continuación:

- npreg: Número de veces que la mujer ha estado embarazada.
- glu: Concentración de glucosa en plasma sanguíneo a 2 horas en una prueba de tolerancia a la glucosa.

- bp: Presión sanguínea diastólica (mm Hg).
- skin: Grosor del pliegue cutáneo del tríceps (mm).
- bmi: Índice de masa corporal, calculado como el peso.
- ped: Función que proporciona una medida de la agregación genética de diabetes en familiares y parientes.
- age: Edad de la persona.
- type: Variable de interés que indica si la persona tiene diabetes (1) o no (0).

Si la selección de variables por pasos se realiza con dirección forward, comenzando sin covariables y usando el criterio AIC, entonces algunos pasos del algoritmo se presentan en la Figura 3.4.

```
Start:  AIC=422.3
type ~ 1

        Df Deviance    AIC
+ glu    1   325.99 329.99
+ bmi    1   386.84 390.84
+ age    1   394.44 398.44
+ skin   1   395.98 399.98
+ ped    1   399.79 403.79
+ npreg  1   401.55 405.55
+ bp     1   410.44 414.44
<none>       420.30 422.30
```

(A) Modelo sin variables predictoras

```
Step:  AIC=329.99
type ~ glu

        Df Deviance    AIC
+ npreg  1   311.02 317.02
+ bmi    1   312.49 318.49
+ age    1   314.87 320.87
+ skin   1   315.16 321.16
+ ped    1   317.52 323.52
+ bp     1   323.88 329.88
<none>       325.99 329.99
```

(B) Modelo con una variable predictora

```
Step:  AIC=302.54
type ~ glu + npreg + bmi

       Df Deviance    AIC
+ ped   1   287.44 297.44
<none>     294.54 302.54
+ age   1   293.35 303.35
+ skin  1   293.93 303.93
+ bp    1   294.28 304.28
```

(C) Modelo con dos variables predictoras

```
Step:  AIC=297.44
type ~ glu + npreg + bmi + ped

       Df Deviance    AIC
<none>     287.44 297.44
+ age   1   286.73 298.73
+ skin  1   286.96 298.96
+ bp    1   287.23 299.23
```

(D) Modelo sin agregar más variables

Figura 3.4: Algunos pasos del método forward aplicado al conjunto de datos Diabetes in Pima Indian Women.

En la Figura 3.4 A) se presenta el modelo inicial sin covariables predictoras, con un valor de AIC de 422.3. Al incorporar la covariable `glu` al modelo, como se muestra en la Figura 3.4 B), el AIC se reduce a 329.99, lo que indica un mejor ajuste.

Siguiendo el mismo proceso de selección de covariables, al añadir `npreg` y después `bmi` al modelo, el AIC disminuye a 302.54, como se observa en la Figura 3.4 C). Esto sugiere continuar con el proceso de selección mejorar el modelo.

La Figura 3.4 D) muestra el modelo final con un AIC de 297.44 con las covariables`glu, npreg, bmi`, agregar otra covariable no mejoraría el valor del AIC.

Finalmente, la regla de clasificación se ajusta con las covariables del modelo final, cuyos valores $\hat{\beta}$'s son aplicados en (3.9) para predecir nuevas observaciones.

Método lasso aplicado a la regresión logística binaria

En la anterior subsección, se discutió sobre una combinación de un modelo de clasificación y el uso del método stepwise. Ahora, se abordará el método de regularización lasso. En este enfoque, se agrega un término de regularización a la función de verosimilitud del modelo, que penaliza los coeficientes de las variables predictoras y fomenta la selección de un subconjunto relevante de variables:

$$\underset{\beta_0,\beta}{max}[\frac{1}{n}\sum_{i=1}^{n}[y_i\eta_i - ln(exp(\eta_i)+1) - \lambda\sum_{j=1}^{p}|\beta_j|]. \tag{3.10}$$

El hiperparamétro λ controla la magnitud de la penalización. Esta función al optimizarse para algunos valores de λ grandes lleva a que algunos valores de los coeficientes β sean 0, lo que implica la eliminación de la participación de ciertas variables explicativas en el modelo. Este hiperparámetro debe ser tuneado.

Los resultados al maximizar (3.10), representados por los coeficientes $\hat{\beta}$, varían dependiendo del valor de λ. En este contexto, los coeficientes estimados se denominarán como $\hat{\beta}_\lambda$, indicando su dependencia al hiperparámetro λ. De manera que para un valor de λ fijo, la $\hat{P}(y=1)$ se estima con los $\hat{\beta}_\lambda$'s calculados. Por ejemplo, con la regresión logística:

$$\hat{P}(y=1) = \frac{1}{1+\exp^{-(\hat{\beta}0_\lambda+\hat{\beta}1_\lambda x_1+\hat{\beta}2_\lambda x_2+...+\hat{\beta}p_\lambda x_p)}}. \tag{3.11}$$

Al aplicar la regularización lasso en el conjunto de datos *Diabetes in Pima Indian Women*, el valor que minimiza el error de clasificación está dado por $\lambda = 0.06492767$. Generalmente se utiliza la validación cruzada, para evaluar el rendimiento del modelo con diferentes valores de lambda. Los datos se dividen en K grupos, y el modelo se entrena en k - 1 grupos y se prueba en el grupo restante. De esta manera se utiliza métricas de evaluación para medir el rendimiento del modelo con cada valor de lambda.

Para esto se usó la función *cv.glmnet* del paquete *glmnet* (Friedman et al., 2023) indicando 200 λ's, ver la Figura 3.5. Por otro lado, en la parte superior del gráfico se visualizan la cantidad de variables a considerar. Es decir, fueron eliminadas dos covariables de la regla de clasificación.

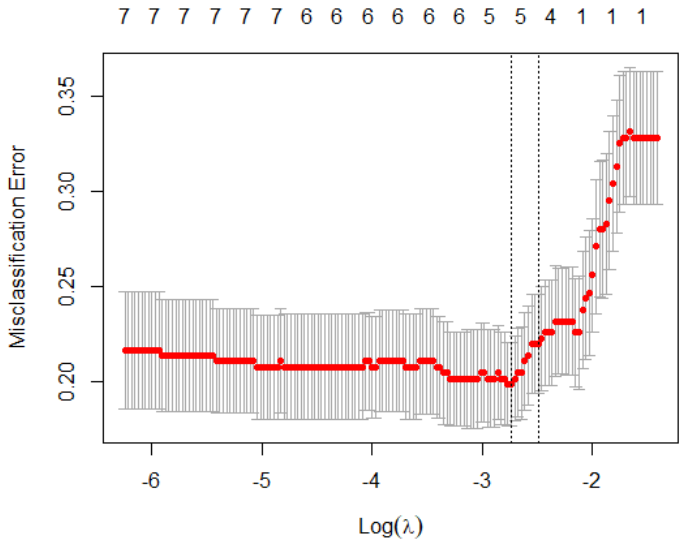


Figura 3.5: Ejemplo del tuneo de λ con cv.glmnet y los datos Diabetes in Pima Indian Women.

Cabe señalar que dentro de R, la función *"cv.glmnet()"* posee 2 parámetros a considerar: gamma = 0 indica que la implementación es lasso, de lo contrario se emplea la regresión Ridge y el paramétro

relax=False indica que se aplica sólo la regresión lasso, en caso contrario se usaría otra función liga conocida como ElasticNet.

Para mayor información sobre los métodos de regularización lasso, rigde y ElasticNet, ver (Hastie et al., 2009, sección 4.4.4) o (Hastie et al., 2023, Capitulo 3)

3.3.2 Máquinas de soporte vectorial (SVM)

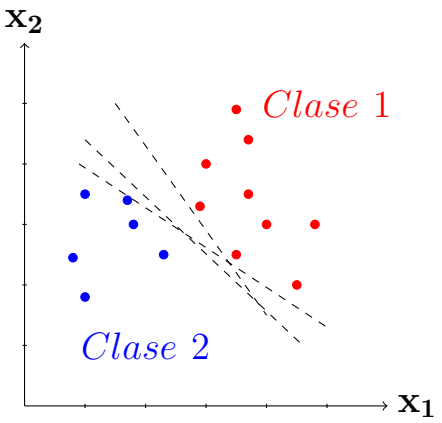


Figura 3.6: SVM con separabilidad lineal, donde multiples hiperparametros pueden separar las dos clases.

En el contexto de las máquinas de soporte vectorial (SVM), el objetivo es construir una regla que permita separar C clases mediante hiperplanos. Para simplificar, será considerado el caso de clasificación binaria.

Un hiperplano puede ser entendido como un subespacio "plano" de dimensión $p-1$ que se encuentra dentro de un espacio p dimensional. Para describir este "plano", se consideran todos los puntos que satisfacen la ecuación siguiente:

$$\forall x \in X; w_1x_1 + w_2x_2 + \cdots + w_px_p + b = 0 \tag{3.12}$$

$$o\ bien\ \mathbf{w}^t\mathbf{x} + \mathbf{b} = 0. \tag{3.13}$$

Donde $\mathbf{x}$ son las variables predictoras, $\mathbf{w}$ es un vector perpendicular al plano y $\mathbf{b}$ es un parámetro que influye en la posición del hiperplano.

3.3.2.1 Caso donde existe separabilidad lineal

En algunos casos, puede ocurrir que varios hiperplanos sean candidatos válidos para separar los dos grupos, como se ilustra en la Figura 3.6. El propósito del método es encontrar aquel hiperplano que maximice la mínima distancia entre el hiperplano y cada uno de los puntos asociadas a las n observaciones, lo que se conoce como "margen". Este enfoque permite obtener un umbral amplio para clasificar de manera más precisa a nuevas observaciones.

En esencia, SVM busca crear un margen que permita mantener la pureza de las clases separadas por el hiperplano, es decir, se pretende que la distancia entre los puntos más cercanos de cada clase y el hiperplano sea la mayor posible. Este margen es crucial, ya que proporciona mayor confianza al clasificar nuevas observaciones, permitiendo una mejor generalización del modelo. En consecuencia, SVM busca una solución óptima que no solo separa los datos, sino que también maximiza la separación y el espacio disponible entre las clases para lograr una clasificación más precisa y eficaz. El problema se traduce en definir $\mathbf{w}$ y b. Para esto se realiza la siguiente restricción en el conjunto de entrenamiento, en donde por facilidad y sin pérdida de generalidad se define como $y = 1$ para una clase y $y = -1$ para otra clase:

$$\mathbf{w}^t\mathbf{x}_i + b \geq 1 \; para \; y_i = 1 \tag{3.14}$$

$$\mathbf{w}^t\mathbf{x}_i + b \leq -1 \; para \; y_i = -1. \tag{3.15}$$

Las ecuaciones (3.14) y (3.15) se pueden representar en una sola ecuación como:

$$y_i(\mathbf{w}^t\mathbf{x}_i + b) \geq 1 \text{ ó} \tag{3.16}$$

$$y_i(\mathbf{w}^t\mathbf{x}_i + b) - 1 \geq 0. \tag{3.17}$$

En especial, si se cumple

$$y_i(\mathbf{w}^t\mathbf{x}_i + b) - 1 = 0, \tag{3.18}$$

los vectores $\mathbf{x}_i$ son vectores soporte, pues son los que se encuentran en el margen, tal que $y_i(\mathbf{w}^t\mathbf{x}_i + b) = 1$, ver Figura 3.7.

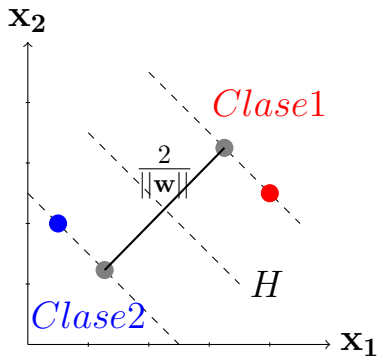


*Figura 3.7: Distancia euclidiana entre dos vectores soporte y el hiperplano H, donde H cumple $\mathbf{w}^t\mathbf{x} + \mathbf{b} = 0$. La distancia está representada por la línea perpendicular al hiperplano H. Los puntos grises hacen referencia al punto azul y rojo una vez proyectados con el vector **w**.*

De manera geométrica se puede considerar a $\mathbf{w}$ como un vector perpendicular al hiperplaplano H. Por lo tanto, cualquier punto en el espacio puede ser proyectado con el vector $\mathbf{w}$, es decir, $\mathbf{x} \cdot \frac{\mathbf{w}}{||\mathbf{w}||}$.

Para dos puntos $\mathbf{x}, \mathbf{x}'$ su distancia es:

$$d(\mathbf{x}, \mathbf{x}') = (\mathbf{x}' - \mathbf{x})\frac{\mathbf{w}}{||\mathbf{w}||}. \tag{3.19}$$

Dado la ecuación (3.18), si $\mathbf{x}$ es de la clase 1 y $\mathbf{x}'$ es de la clase 2 se tiene $\mathbf{x}\mathbf{w} = 1 + b$ *y* $\mathbf{x}'\mathbf{w} = 1 - b$, de donde:

$$d(\mathbf{x}, \mathbf{x}') = \frac{2}{||\mathbf{w}||}. \tag{3.20}$$

Es decir, la distancia entre los vectores soporte se encuentra en términos de $\mathbf{w}$, y no de los vectores $(\mathbf{x}, \mathbf{x}')$.

Por lo tanto, el problema se puede plantear como encontrar $\mathbf{w}$ y b tal que se maximice a:

$$\frac{2}{||\mathbf{w}||}. \tag{3.21}$$

Lo que es equivalente a minimizar

$$||\mathbf{w}||^2 \text{ o bien } \frac{1}{2}||\mathbf{w}||^2, \tag{3.22}$$

con las siguientes restricciones:

$$y_i(\mathbf{w}^t\mathbf{x}_i + b) - 1 \geq 0, i = 1, \ldots, n$$

Por lo tanto, el objetivo de un SVM con separabilidad lineal yace en resolver:

$$\underset{\mathbf{w},b}{min}\frac{1}{2}||\mathbf{w}||^2 = \underset{\mathbf{w},b}{min}\frac{1}{2}(\mathbf{w}^t \cdot \mathbf{w}) \tag{3.23}$$

con la restricción $y_i(\mathbf{w}^t\mathbf{x}_i + b) - 1 \geq 0, i = 1, \ldots, n$.

Para resolver este problema de n observaciones, se introduce el Lagrangiano.

$$\underset{\mathbf{w},b}{min}\ \underset{\lambda}{max}\ L(\mathbf{w}, b, \lambda) = \underset{\mathbf{w},b}{min}\ \underset{\lambda}{max}\ [\frac{1}{2}(\mathbf{w}^t \cdot \mathbf{w}) - \sum_{i=1}^{n}\lambda_i(y_i(\mathbf{w}^t\mathbf{x}_i + b) - 1)]. \tag{3.24}$$

Entonces, en lugar de tener un problema con restricciones se convierte en un problema sin restricciones. El costo en este tipo de problema es agregar los hiperparámetros λ, conocidos como multiplicadores de Lagrange.

Del problema (3.24) se busca encontrar los valores de $\mathbf{w}$ y b que minimicen la función. Además, dado que en la ecuación (3.24) el primer término es cuadrático, entonces si se encuentra su mínimo local se estaría encontrando su mínimo global. Dado que el valor b es un término que también puede variar, la solución para el problema (3.24) satisface:

$$\frac{\partial L(\mathbf{w}, b, \lambda)}{\partial \mathbf{w}} = 0 \tag{3.25}$$

$$\frac{\partial L(\mathbf{w}, b, \lambda)}{\partial b} = 0. \tag{3.26}$$

De (3.25) se tiene:

$$\frac{\partial L(\mathbf{w}, b, \lambda)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\lambda_i y_i x_i = 0 \tag{3.27}$$

$$\implies \mathbf{w} = \sum_{i=1}^{n}\lambda_i y_i \mathbf{x}_i. \tag{3.28}$$

De (3.26)

$$\frac{\partial L(\mathbf{w}, b, \lambda)}{\partial b} = -\sum_{i=1}^{n}\lambda_i y_i = 0 \tag{3.29}$$

$$\implies \sum_{i=1}^{n}\lambda_i y_i = 0. \tag{3.30}$$

Sustituyendo (3.28) en (3.24):

$$L(\mathbf{w}, b, \lambda) = \frac{1}{2}(\sum_{i=1}^{n}\lambda_i y_i \mathbf{x}_i \cdot \sum_{j=1}^{n}\lambda_j y_j \mathbf{x}_j) - \sum_{i=1}^{n}\lambda_i(y_i(\sum_{j=1}^{n}\lambda_j y_j \mathbf{x}_j \cdot \mathbf{x}_i + b) - 1) \tag{3.31}$$

$$= \frac{1}{2}(\sum_{i=1}^{p}\sum_{j=1}^{n}\lambda_j y_j \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{i=1}^{n}\lambda_i(y_i(\sum_{j=1}^{n}\lambda_j y_j \mathbf{x}_j \cdot \mathbf{x}_i) - b\sum_{i=1}^{n}\lambda_i y_i + \sum_{i=1}^{n}\lambda_i \tag{3.32}$$

$$= \frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_j y_j \lambda_i y_i x_i \cdot \mathbf{x}_j) - \sum_{i=1}^{n}\lambda_i(y_i(\sum_{j=1}^{n}\lambda_j y_j \mathbf{x}_j \cdot \mathbf{x}_i) - b\sum_{i=1}^{n}\lambda_i y_i + \sum_{i=1}^{n}\lambda_i \tag{3.33}$$

$$= \frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_j y_j \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_j y_j \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n}\lambda_i \cdots \text{ por (3.30)} \tag{3.34}$$

$$\implies L(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_j y_j \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x}_j). \tag{3.35}$$

De donde el problema (3.24) se reduce a

$$\max_{\lambda} L(\lambda) = \max_{\lambda} [\sum_{i=1}^{n} \lambda_i - \frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_j y_j \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x}_j)], \tag{3.36}$$

$$\text{con } \lambda_1, \ldots, \lambda_n \geq 0 \text{ y } \sum_{i=1}^{n} \lambda_i y_i = 0.$$

De manera que con los vectores de soporte se puede calcular el hiperplano óptimo que separa nuestras dos clases. Más aún, esta solución se produce a través del producto punto.

3.3.2.2 Caso donde no hay separabilidad total

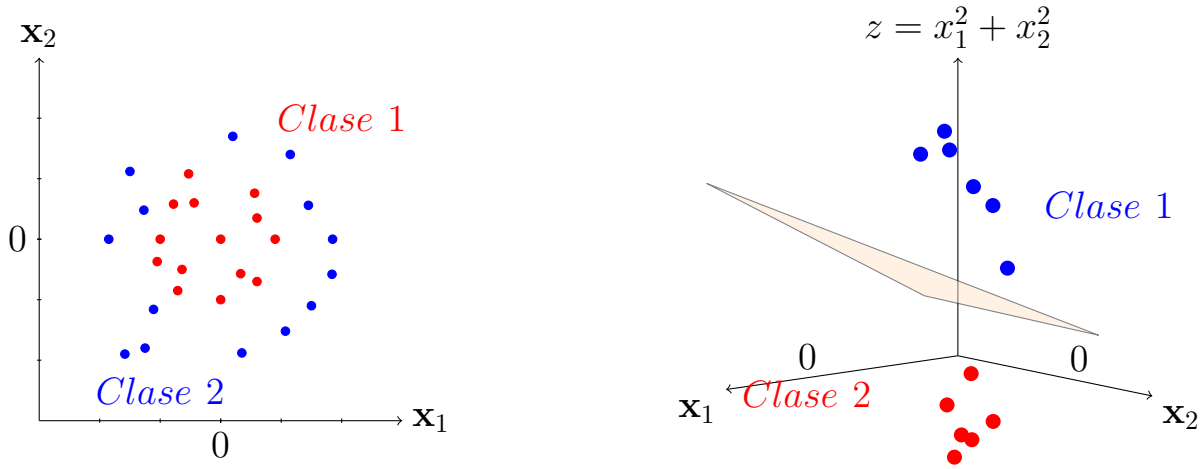
En casos donde los datos no pueden ser separados linealmente en el espacio original, el SVM lineal puede no funcionar.

Por ello, se introducen dos conceptos: Kernel y la función de costo. El Kernel permite transformar los datos a un espacio de mayor dimensión donde la separación puede ser posible.

Por otro lado, la función de costo en SVM, controla el equilibrio entre maximizar el margen y minimizar el error de clasificación.

Por lo general, ambos conceptos son utilizados de manera conjunta en el tuneo para mejorar la regla de clasificación.

Uso de Kernels



(A) Ejemplo de SVM sin completa separabilidad lineal con $\mathbf{x}_1$ y $\mathbf{x}_2$

(B) Ejemplo de SVM con separabilidad lineal con uso de una variable adicional: es decir $\phi(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_1^2 + \mathbf{x}_2^2)$.

Figura 3.8: Ejemplo de donde no hay completa separabilidad lineal si sólo se usan $\mathbf{x}_1$ y $\mathbf{x}_2 \in \mathcal{R}^2$, pero donde si se considera una variable adicional z se puede conseguir la separabilidad lineal.

En casos donde no hay separabilidad lineal, se podría encontrar una función $\phi()$ en una mayor dimensión, tal que el problema sea modelado en otra dimensión para tener una mejor segmentación de clases. Ver Figura 3.8 donde incluso podría lograr separabilidad total.

Partiendo de la ecuación (3.36) se puede observar que el producto interno de los vectores soporte ayuda en encontrar el hiperplano óptimo para segmentar las dos clases, pero si no hay separabilidad total se podría usar en lugar de $\mathbf{x}_i \cdot \mathbf{x}_j$ a $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$, donde con $\phi()$ si es posible resolver el problema. Para el cálculo de $\phi()$ para todos los puntos, se define el concepto de Kernel:

Definición 1 (Kernel) *Sea $K(\mathbf{x}, \mathbf{x}')$ una función definida en $\mathbb{R}^p \times \mathbb{R}^p$. Decimos que K es un kernel si existe un mapeo ϕ tal que*

$$K(\mathbf{x}, \mathbf{x}') = (\phi(\mathbf{x}) \cdot \phi(\mathbf{x}')) \ \forall x, x' \in \mathbb{R}^p. \tag{I}$$

Dada esta definición, si tenemos un conjunto de observaciones con características donde no existe separabilidad lineal, el problema de optimización estaría dado por:

$$\max_{\lambda}[\sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_j y_j \lambda_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)] \tag{3.37}$$

ó

$$\max_{\lambda}[\sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_j y_j \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}_j)]. \tag{3.38}$$

La función Kernel es de gran utilidad, ya que permite trabajar con datos que no son linealmente separables. Una de las ventajas más importantes del kernel es que no requiere conocer explícitamente la transformación $\phi()$, lo que lo hace computacionalmente eficiente.

Kernels comúnmente utilizados:

- **Kernel Polinomial**:
$$K(\mathbf{x}, \mathbf{x}') = ((\mathbf{x} \cdot \mathbf{x}') + c)^d$$
donde d y c son parámetros que pueden ajustarse.
- **Kernel Radial Gaussiano**:
$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\gamma^2}\right)$$
donde γ es un parámetro ajustable.

Para ilustrar cómo funciona el kernel, consideremos un problema en $\mathbb{R}^2$ que puede resolverse en $\mathbb{R}^3$. Supongamos que tenemos dos vectores $\mathbf{x}$ y $\mathbf{x}'$, donde:

$$\mathbf{x} = [x_1, x_2],$$
$$\mathbf{x}' = [x_1', x_2'].$$

Podemos usar el Kernel Polinomial con $d = 2$ y $c = 0$, de donde:

$$K(\mathbf{x}, \mathbf{x}') = ((\mathbf{x} \cdot \mathbf{x}'))^2 = ((x_1 x_1' + x_2 x_2'))^2,$$
$$= (x_1 x_1')^2 + 2 x_1 x_1' x_2 x_2' + (x_2 x_2')^2.$$

Por otro lado, supongamos que de antemano tenemos la transformación $\phi()$:

$$\phi(\mathbf{x}) = [x_1^2, 2 x_1 x_2, x_2^2],$$
$$\phi(\mathbf{x}') = [x_1'^2, 2 x_1' x_2', x_2'^2].$$

En este caso, el producto punto de estos dos vectores transformados sería:

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{x}') = x_1^2 x_1'^2 + 2x_1 x_2 x_1' x_2' + x_2^2 x_2'^2 = K(\mathbf{x}, \mathbf{x}').$$

El kernel nos permite calcular el producto escalar de dos vectores en un espacio de mayor dimensión sin tener que calcular explícitamente las transformaciones ϕ. Esta característica es lo que hace que el uso del kernel sea computacionalmente eficiente y nos permita generar diferentes soluciones mediante una malla de valores en el tuneo.

Inclusión del parámetro de costo en SVM

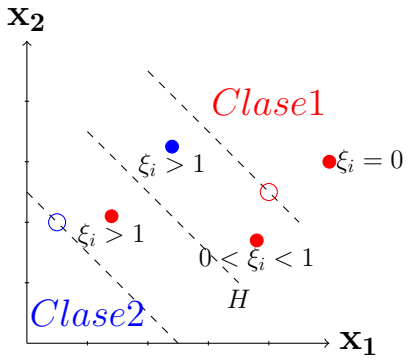


Figura 3.9: Caso donde se relajan las restricciones y se permiten posibles errores (observaciones que caen dentro de los margenes que marcan los vectores soporte) al definir el hiperparámetro. Los vectores de soporte cumplen con la condición $y_i(\mathbf{w}^t\mathbf{x}_i + b) = 1$ y se encuentran en los márgenes asociados al hiperplano H. Por otro lado ξ es el error asociado a cada observación que cae dentro del margen de los vectores de soporte, y se define como $\xi_i = \max(0, 1 - y_i(\mathbf{w}^t\mathbf{x}i + b))$

En SVM el término "costo" se refiere al costo asociado con la clasificación errónea de las observaciones de entrenamiento. Como se mencionó, el objetivo de SVM es encontrar el hiperplano que mejor separa las clases en el espacio de características. Sin embargo, es posible que no exista un hiperplano que pueda separar perfectamente todas las observaciones.

La función de costo se introduce para controlar cómo se penaliza la clasificación incorrecta durante el proceso de entrenamiento del modelo SVM. Un valor más alto de costo indica una penalización más severa por los errores de clasificación, lo que lleva a un modelo con una separación más estricta entre las clases. Por otro lado, un valor más bajo de costo permite una mayor flexibilidad al permitir que algunas observaciones queden mal clasificadas. Al problema de optimización en (3.23) se añade la función costo, dada por :

$$\underset{\mathbf{w},b,\xi}{min} \frac{1}{2}||\mathbf{w}||^2 + cost \sum_{i=1}^{n} \xi_i \tag{3.39}$$

$$\text{sujeto a } y_i(\mathbf{w}^t\mathbf{x}_i + b) \geq 1 - \xi_i, \text{ para } i = 1, 2, ..., n, \tag{3.40}$$

$$\xi_i \geq 0, \text{ para } i = 1, 2, ..., n, \tag{3.41}$$

Donde $\xi_i = \max(0, 1 - y_i(\mathbf{w}^t\mathbf{x}i + b))$ y $Cost > 0$ es un hiperparámetro de penalización por cada observación que caiga dentro del margen de los vectores soporte. A partir de (3.39), se concluye que, además de minimizar $||\mathbf{w}||^2$, también se minimiza $\sum_{i=1}^{p} \xi_i$, que representa la medida de violación en las restricciones $y_i(\mathbf{w}^t\mathbf{x}_i + b) \geq 1$ para $i = 1, 2, ..., n$. El parámetro *cost* determina el peso entre estos dos términos y su ajuste, de esta manera *cost* puede ser tuneado en lenguajes de programación.

Dado que el problema en (3.39) es genérico se puede usar en conjunto con cualquier tipo de kernel.

Para mayor profundidad sobre SVM, así como de los kernels, se puede consultar Deng et al., (2012).

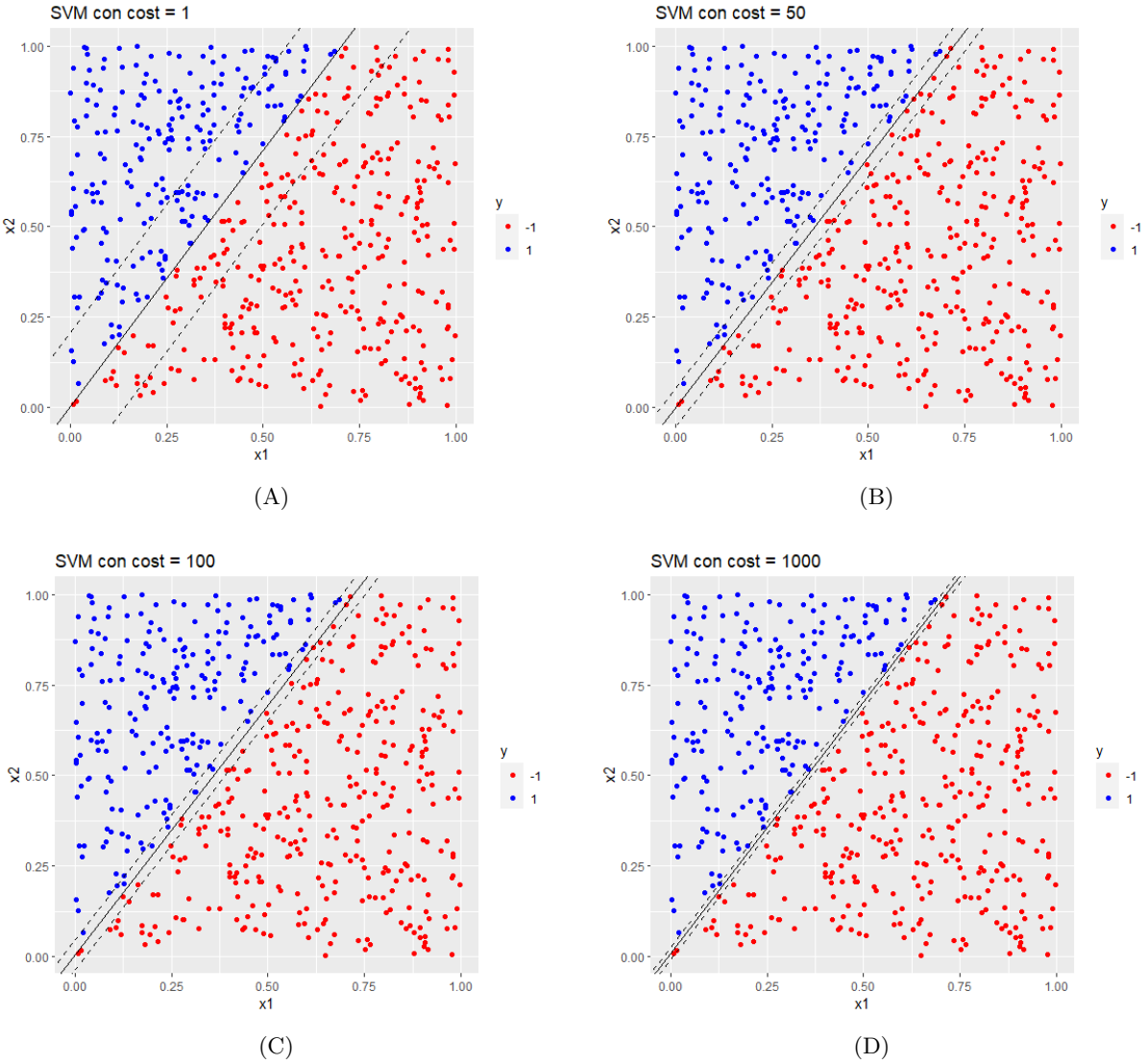


Figura 3.10: Resultado de aplicar SVM con kernel lineal y modificando el hiperparámetro $Cost$, en datos simulados.

La librería utilizada en este trabajo para aplicar el método de SVM es: *e1071* (Meyer, 2023).

En lenguajes de programación como R o Python, existen varios tipos de kernels preprogramados. Dependiendo del kernel que se elija serán los valores de los hiperparámetros a tunear. Más aún, se considera que el kernel es un hiperparámetro que puede ser tuneado.

Con el propósito de visualizar el efecto del parámetro *cost* al ser tuneado, se generó una muestra de 700 observaciones simuladas, con dos covariables numéricas generadas aleatoriamente, mientras que la creación de la variable Y corresponde a dos grupos: 1 y -1. Los datos garantizan separabilidad lineal, los resultados se muestran en la Figura 3.10. Se puede observar cómo el margen que divide a dos clases (-1,1) se ve afectado a medida que se modifica el hiperparámetro *cost*. En 3.10 D) se observa la mejor opción de las 4 para separar de manera lineal las clases 1 y -1.

3.3.3 K vecinos más cercanos (KNN)

El algoritmo de K-vecinos más cercanos (KNN) clasifica una nueva observación basándose en las disimilaridades o distancias entre ésta y todas y cada una de las n observaciones. La distancia o disimilaridad se calcula con la información de las variables $x_1, \ldots, x_p$.

Para una nueva observación con todas las disimilaridades calculadas con respecto a las n observaciones existentes, se seleccionan las K disimilaridades más pequeñas y la nueva observación se asigna al grupo

al que pertenezca la mayoría las K observaciones asociadas. A estas K observaciones se le llaman vecinos más cercanos, es decir, este método se basa en identificar los K vecinos más cercanos de cada nueva observación.

Una de las ventajas que tiene KNN es que la implementación del algoritmo no requiere que las observaciones sean linealmente separables ni que sigan una distribución específica. Es un método no paramétrico. La desventaja es que requiere mucho poder computacional, pues para cada nueva observación se calculan n disimilaridades o distancias, de donde si n es grande este punto es importante. Además, dado que depende de una disimilaridad o distancia, se sugiere que $x_1, \ldots, x_p$ estén en una escala comparable.

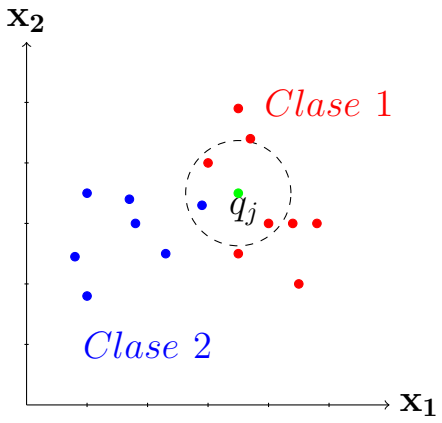


Figura 3.11: KNN con $K = 3$ en un espacio bidimensional usando la distancia euclidiana. Los puntos rojos pertenecen a la clase 1, mientras que los puntos azules son de la clase 2, el punto verde q_j corresponde a una nueva observación.

En la Figura 3.11 se muestra un ejemplo de clasificación de dos grupos con datos de dos variables explicativas $\mathbf{x}_1$ *y* $\mathbf{x}_2$. Asumiendo $K = 3$, para la nueva observación q_j se tienen dos vecinos de la clase 1 y uno de la clase 2. La regla más común es que se le asigne la clase 1, pues es la que tiene más observaciones cercanas a q_j.

Es importante prestar atención al valor de K. Si se elige una K muy pequeña la decisión podría estar sujeta a ruido, mientras que una K muy grande podría incluir puntos que no son relevantes para la clasificación, reduciendo la precisión del modelo. Por lo tanto, K es un hiperparámetro que se debe tunear según el problema y los datos disponibles.

Además, la función que mide las diferencias juega un papel importante en el proceso de clasificación. La medida más comúnmente utilizada para esto es la distancia Euclidiana:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{r=1}^{p} (x_{r,j} - x_{r,i})^2},$$

donde $\mathbf{x}_i$ y $\mathbf{x}_j$ son dos elementos a comparar, p es el número variables y $x_{r,j}$ y $x_{r,i}$ son los valores de la variable $r - ésima$. Además de la distancia euclidiana, existen otras medidas de disimilaridad que se podrían usar.

Para más información de este método se puede consultar Hastie et al. (2021, sección 3.5). La librería utilizada en la aplicación de este método fue **class** (Ripley, 2023).

3.3.4 Árboles de Decisión y Bosques Aleatorios (Random Forest, RF)

La estructura de los árboles de decisión se puede representar como un esquema con un nodo raíz y nodos hijos, como se muestra en la Figura 3.12, donde a partir del nodo raíz, se definen condiciones con base en los valores de las variables $x_1, \ldots, x_n$, de manera que el espacio generado por $x_1, \ldots, x_p$ se particionan para crear un hijo derecho o un hijo izquierdo. Estas condiciones resultan en la formación

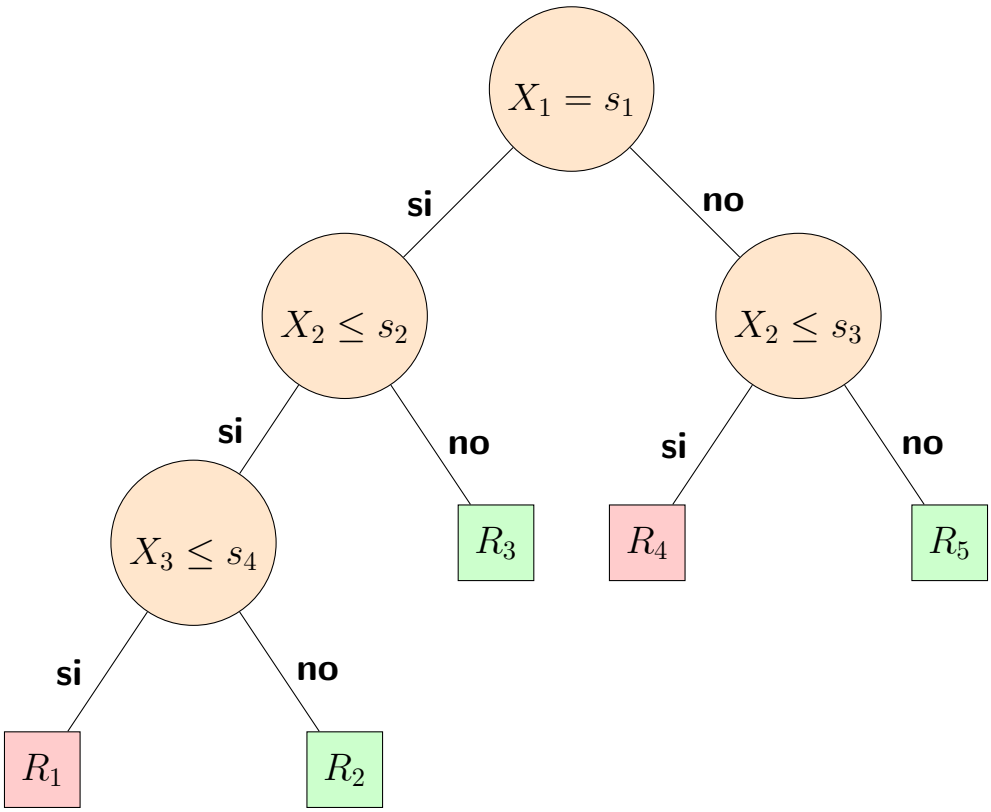


Figura 3.12: Ejemplo de árbol de decisión para clasificación binaria, donde las regiones R_j con mayor proporción del grupo 1 están representados en color rojo y con mayor proporción del grupo 2 en color verde. El árbol resultante se divide en cinco regiones terminales, considerando una variable categórica (X_1) y dos variables continuas (X_2, X_3).

de regiones terminales $R_1, R_2, ..., R_j$, que representan subdivisiones del espacio de características. Las aristas corresponden a la decisión que se toma, sobre cada condición de arriba hacia abajo.

La manera en que se definen las condiciones para la partición binara depende el tipo de variables $x_1, \ldots, x_p$ que existen en el conjunto de datos:

- Variable x_j es continua. Dado que la variable es continua, generalmente se encontrará en un rango especifico, digamos [a,b], de esta manera la partición binaria podría ser de la forma $x_j < c$ o $x_j \leq c$, donde $a < c < b$. Para facilitar este proceso, se eligen algunos valores específicos de c basados en la muestra, como percentiles o los valores observados.

- Variable x_j es categórica con L niveles. Si hay al menos dos niveles presentes, se considera cualquier partición en dos conjuntos no vacíos que involucra a todas las categorías.

- Variable x_j es categórica ordinal: Podemos tratarla como continua asignando valores ordenados de 1 a L.

El problema del árbol consiste en la definición de las regiones $R_1, R_2, \ldots, R_j$ con base en las $x_1, x_2, \ldots, x_p$ covariables. La definición del árbol es iterativa. Consideremos un nodo no particionado en el árbol en la $i - ésima$ iteración; para lograr una partición efectiva, entre todas las posibles divisiones, debemos elegir la que mejor contribuya al aumento de la homogeneidad de sus dos hijos. Esto se logra definiendo una medida de impureza.

Se considera que una división 100% pura cuando, después de aplicarla, todas las observaciones en los nodos hijos pertenecen a la misma clase. En otras palabras, la proporción de elementos que son de la clase c en la región o nodo R_j es del 100%.

En problemas de clasificación, donde la variable objetivo $\mathbf{Y}$ es categórica con C niveles, se emplean comúnmente dos índices para medir la impureza:

$$\text{Gini impurity:} \quad Imp(R_j) = 1 - \sum_{c=1}^{C} p_{cR_j}^2, \tag{3.42}$$

$$\text{Entropy impurity:} \quad Imp(R_j) = -\sum_{c=1}^{C} p_{cR_j} \log_2(p_{cR_j}), \tag{3.43}$$

donde:

- R_j es un nodo en el árbol de decisión.
- p_{cR_j} es la proporción de observaciones de la clase c en el nodo R_j.
- C es el número total de clases.

Supongamos, que mediante una regla o condición sobre la variable x_j se parte la región o nodo R_j en dos, R_{j1} y R_{j2}. Sea $n_{R_{j1}}$ el número de elementos de la región R_j que caen en R_{j1} y $n_{R_{j2}}$ que caen en R_{j2}.

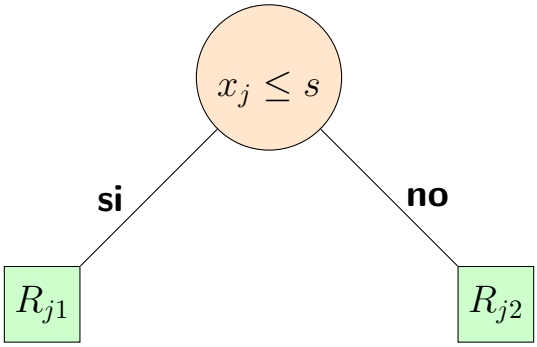


Figura 3.13: Partición de un nodo, correspondiente a la condición de la covariable x_j con característica s

Con esta información se establece una medida que evalué la ganancia en la partición para el nodo R_j:

$$\triangle(Imp) = Imp(R_j) - \frac{n_{R_{j1}}}{n_{R_j}} Imp(R_{j1}) - \frac{n_{R_{j2}}}{n_{R_j}} Imp(R_{j2}). \tag{3.44}$$

Así, el criterio de la mejor partición se define con la máxima ganancia que se obtiene al particionar un nodo.

Por ejemplo, supongamos que se cuenta con 50 observaciones y dos covariables: x_1 y x_2. Después de la división con la covariable x_1 se tiene 30 observaciones en el nodo derecho R_{j1} y 20 observaciones en el nodo izquierdo R_{j2}.

Ahora, para R_{j1} hay 5 observaciones de la clase 1 y 25 en la clase 2, de manera similar supongamos que hay 7 elementos de la clase 1 y 13 de clase 2 en R_{j2}. Calculando la impureza Gini en (3.42), se tiene:

$$Imp(R_{j1}) = 1 - [(\frac{5}{30})^2 + (\frac{25}{30})^2] = 0.455,$$
$$Imp(R_{j2}) = 1 - [(\frac{7}{20})^2 + (\frac{13}{20})^2] = 0.277.$$

Ahora, supongamos que antes de la partición, el nodo R_j tenía una impureza de Gini de 0.5.

Usando (3.44) se tiene:

$$\triangle(Imp_1) = 0.5 - \frac{20}{50} * 0.455 - \frac{30}{50} * 0.277 = 0.1518.$$

Por otro lado, si usando la covariable x_2 se tiene una división en donde hay 10 elementos en el nodo derecho y 40 elementos en el nodo izquierdo, cuya impureza fue de 0.45 y 0.5, respectivamente, se tiene:

$$\triangle(Imp_2) = 0.5 - \frac{10}{50} * 0.45 - \frac{40}{50} * 0.5 = 0.01.$$

Entonces, en este caso se elige la partición con la covariable x_1, pues la impureza se reduce 15% respecto al nodo R_j, mientras que con la partición con x_2 la impureza se reduce 1% respecto al nodo R_j.

La creación de un árbol conlleva un costo computación grande, pues depende del número de covariables, así como de los nodos que están disponibles en cada iteración $x_1, .., x_p$; comparando cada partición posible.

Para usar el árbol con una nueva observación, se van evaluando las reglas de decisión y se llega hasta un nodo final, por ejemplo R_j. En este nodo se tienen las probabilidades de pertenecer a cada clase y se asigna a la de mayor probabilidad.

En determinadas situaciones, existe un volumen considerablemente grande de variables y observaciones. En este contexto, a medida que aumenta la cantidad de información disponible, el árbol de decisión tiende a expandirse, lo que podría desencadenar un fenómeno de sobre ajuste.

Para detener este proceso de crecimiento desmesurado, son empleados diversos elementos auxiliares, conocidos como criterios de paro. Uno de estos, consiste en establecer un número mínimo de observaciones requerido para cada región, por ejemplo, si se considera un mínimo de observaciones de 5, significa que si $R_j < 5$, R_j ya no se seguirá particionando. Asimismo, se puede definir la profundidad máxima de los árboles para el modelo.

Junto con la posibilidad de ajustar el crecimiento del árbol, también existe la opción de reducir su tamaño a través del proceso de "poda". Este consiste en eliminar todas aquellas aristas o ramas que determinen beneficios pequeños respecto a la impureza.

Los árboles de decisión se presentan como estructuras jerárquicas que se asemejan a flujos de decisiones organizados en forma de árbol. Esta configuración propicia su comprensión y visualización valiosa en términos de explicación. Las reglas y trayectorias de decisión son explícitas y pueden interpretarse sin la necesidad de un conocimiento profundo.

No obstante, los árboles de decisión son susceptibles a ligeras variaciones en los datos de entrenamiento. Es decir, existe una propensión al sobreajuste, lo que implica la posibilidad de que el modelo adquiera detalles específicos del conjunto de entrenamiento que no sean generalizables a nuevas observaciones.

Una modificación o extensión de los árboles de decisión incluye el concepto de ensamblado de árboles, conocido como "bagging" (Hastie, 2013, pp. 316-319) en particular "Random Forests" o bosques aleatorios (Hastie, 2013, pp. 319-321).

La premisa que subyace en los Bosques Aleatorios consiste en la creación de un modelo compuesto por árboles de diferentes conformaciones. Estos árboles se caracterizan por poseer distintas estructuras de división y decisiones, y también se han entrenado con conjuntos de datos diversos. Todo esto se lleva a cabo con el propósito de minimizar el error general y aumentar la robustez del modelo final.

En términos sencillos los bosques aleatorios consisten en:

- Creación de árboles de decisión: Se seleccionan muestras aleatorias con reemplazo de tamaño n del conjunto de datos y se crean árboles de decisión para cada muestra seleccionada, donde se define el número de variables que se consideran en cada división al definir los árboles. En R este hiperparámetro es conocido como mtry.
- Número de árboles. Otro de los parámetros más comunes a tunear es la cantidad de árboles que hay en el bosque, en R se conoce como ntree y generalmente es de 500 o 1000 árboles.

- Predicción de cada árbol. Cada árbol de decisión realiza una predicción independiente.
- Votación: Una vez que todos los árboles en el bosque han hecho sus predicciones, estas predicciones se combinan para producir la predicción final del bosque. Este proceso de combinación se realiza a través de un proceso de votación. Cada árbol "vota" por una clase, y la clase con la mayoría de los votos correspondiente a la predicción final del bosque.

Para mayor información sobre la construcción de los árboles, así como el proceso de podado y criterios de paro, consultar (Alpaydin, 2014, seccion 9.2), (Hastie et al., 2021, sección 8.2.2) y Pérez de la Cruz (2023).

Para la construcción de los bosques aleatorios fue considerada la librería **ranger**, que es una herramienta eficiente y rápida para ajustar modelos de bosques aleatorios. La librería ranger está diseñada para manejar grandes conjuntos de datos y aprovechar cálculos en paralelo para acelerar el proceso de ajuste del modelo (Marvin, 2023).

Siguiendo con el mismo ejemplo de la base Diabetes in Pima Indian Women incluida en la librería *MASS*(Venables, 2002), supongamos que fue generado un bosque aleatorio con 3 árboles, donde el nivel de profundidad es 3, ver Figura 3.14.

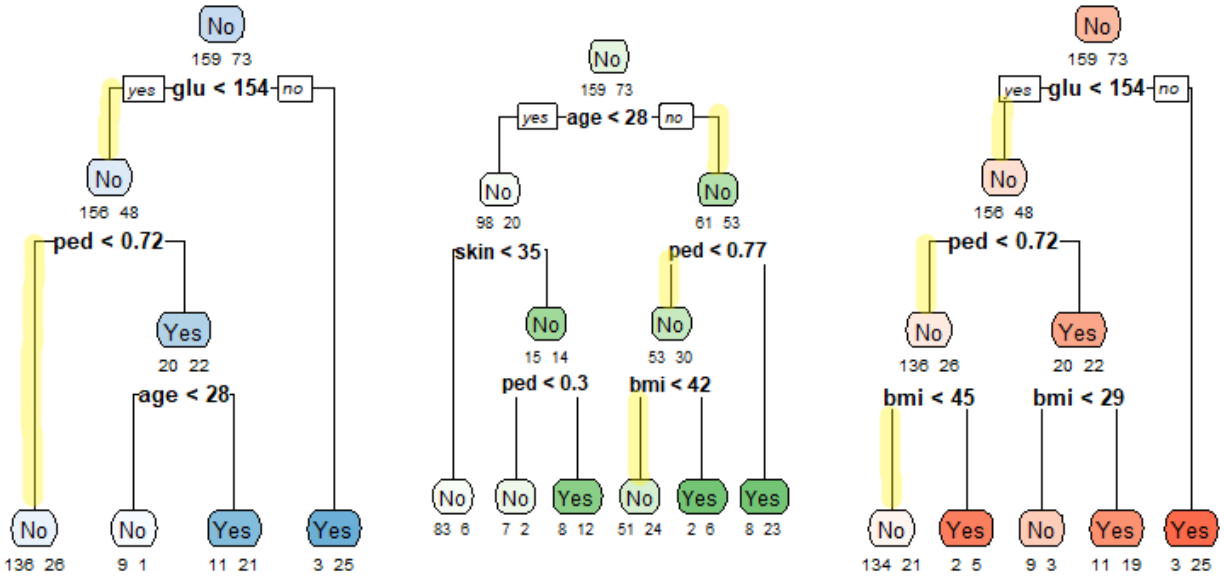


Figura 3.14: Ejemplo de un bosque aleatorio con 3 árboles, donde diferente número de variables y tipo de variables intervienen. En amarillo las trayectorias que se deben seguir con valores: glu = 140, age = 28, bmi = 32, skin=32, ped = 0.5 y npreg = 5.

La operación de clasificación para una nueva observación usando un bosque aleatorio puede ilustrarse de la siguiente manera. Supongamos que se presenta una nueva observación con las siguientes características: glu = 140, age = 28, bmi = 32, skin=32, ped = 0.5 y npreg = 5.

Para el primer árbol, la clasificación se realiza considerando que el nivel de glu es menor a 154 y ped es menor a 0.72. Indicando que la nueva observación no tiene diabetes (grupo 'No'), ver Figura 3.14 (árbol izquierdo de color azul).

En el segundo árbol, la clasificación de la nueva observación se basa en las variables age, skin, ped y bmi, clasificándose como 'No', dado que la nueva observación tiene 28 años, su ped es menor a 0.77 y bmi es menor a 42, ver Figura 3.14 (árbol central de color verde).

Finalmente, en el tercer árbol cuenta con un nivel de glu menor a 154, ped menor a 0.72 y un bmi menor a 45, se clasifica como 'No', ver Figura 3.14 (árbol derecho de color naranja).

Generalmente, el criterio para determinar el grupo al que pertenece la nueva observación se calcula por votación. Dado que dos de los tres árboles clasificaron como no diabética, se concluye que la nueva observación pertenece a la clase 'No'. La votación ayuda a compensar las posibles debilidades individuales de los árboles de decisión y proporciona una clasificación más robusta y precisa.

Este ejemplo pretende ilustrar el funcionamiento de un bosque aleatorio, donde el objetivo es que los árboles sean diferentes entre sí con el fin de reducir la varianza

CAPÍTULO 4

Aplicación

4.1 Construcción del conjunto de datos

4.1.1 Obtención de los datos

Al momento de obtener información satelital sobre una zona específica, generalmente se descarga un archivo comprimido que incluye diversos formatos con datos relacionados con la región en cuestión, entre ellos destacan los archivos en formato tiff, empleados para almacenar datos de las bandas espectrales; archivos de texto o formato txt que almacenan detalles técnicos para la interpretación de la imagen satelital y archivos conocidos como metadatos que poseen información como resolución espacial, coordenadas geográficas del área de estudio, valores radiométricos, etc.

Cabe mencionar que el conjunto de datos fue recopilado para un proyecto piloto centrado en la clasificación de zonas edificadas y no edificadas, de manera que las etiquetas de suelo no son homogéneas, es decir, se presenta un mayor volumen de datos en áreas de asentamiento y agricultura, ver Kamusoko (2019) y Kamusoko (2013).

Los datos de Kamusoko (2019) contienen dos archivos comprimidos: uno corresponde al mes de junio y otro al mes diciembre de 1984. Estos dos momentos representan condiciones climáticas diferentes: seca y húmeda, las cuales se pueden observar en la igura 4.1 A) y 4.1 B), respectivamente.

Cada archivo comprimido contenía la información satelital para la obtención y creación de las covariables y, al mismo tiempo, una muestra de píxeles, etiquetada sobre la cobertura del suelo. En este sentido, para la generación o creación de las etiquetas, Kamusoko (2013) detalla *"...based on the Forestry Commission (Zimbabwe) and the Surveyor-General national cover classes classification schemes as well as the author's a priori knowledge of the study area."*.

En la Tabla 4.1 se muestra un desglose de las categorías del tipo de suelo para la creación u obtención del tipo de cobertura, dicha Tabla fue obtenida de Kamusoko (2019).

En este trabajo, se buscó diferenciar entre dos tipos de suelo: agricultura (etiquetado como "Agr") y no agricultura (etiquetado como "NoAgr"). Se decidió fijar como objetivo la identificación de agricultura dado que los índices espectrales están vinculados a los índices de vegetación, y esto podría ayudar a distinguir este tipo de suelo.

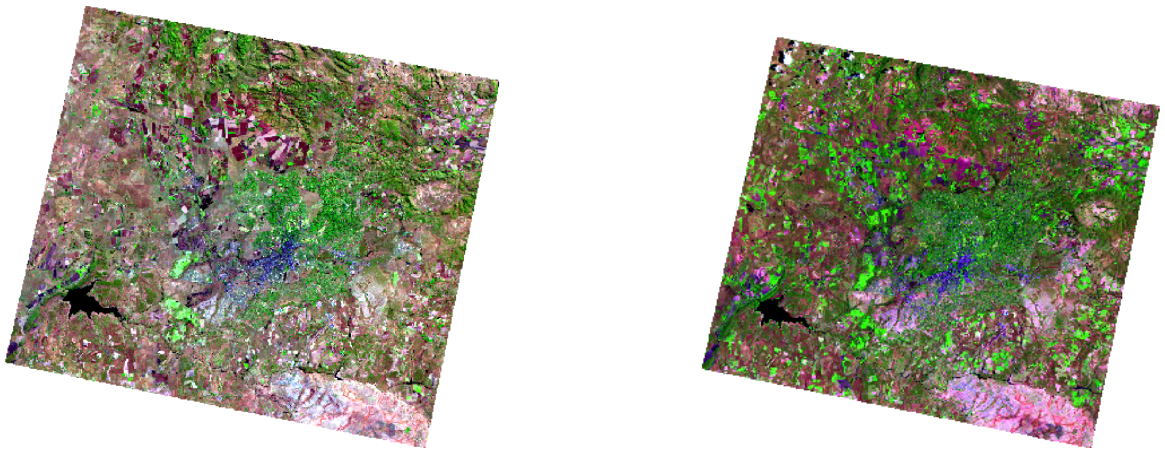
En la Tabla 4.1 se muestra la proporción de datos etiquetados con los que se cuenta, al fijar como objetivo la identificación de agricultura, se generó un desequilibrio en las clases, pues el 31% de las observaciones (2,453) pertenecen al grupo "Agr" sólo con la categoría de agricultura, mientras que grupo "NoAgr" contiene el 69% (5,279) restante de los datos.

Es importante considerar este desbalanceo al construir la regla de clasificación, ya que puede influir en su desempeño.

Además, se decidió entrenar los modelos utilizando los datos del mes de junio de 1984, y sólo usar la información de diciembre para evaluar el rendimiento del modelo que tuvo mejor desempeño predictivo con los datos del mes de junio. Esta elección se realizó con la intención de resaltar la importancia de disponer de datos y etiquetas de suelo en diferentes condiciones estacionales.

Cobertura de suelo	Descripción	Siglas en inglés	Proporción
Asentamiento urbano	Sitios de construcción, edificios, casas, transporte, industrias, etc.	Set	18%
Área verde	Bosques, vegetación ribereña, arbustos y matorrales, cubierta de hierba, áreas abiertas de hierba, campos de golf y parques	GS	9%
Agricultura	Tierra cultivada o tierra en preparación para cultivos, agricultura en barbecho tierra, y tierra bajo riego	Agr	31%
Suelo desnudo	Áreas desnudas expuestas, áreas de transición y sitios de excavación.	Br	41%
Agua	Ríos y embalses	Wt	1%

Tabla 4.1: Descripción y distribución de las etiquetas de suelo del archivo de estudio. La proporción se calcula sobre un total de 7,732 datos.



(A) Tomada el 22 de junio de 1984

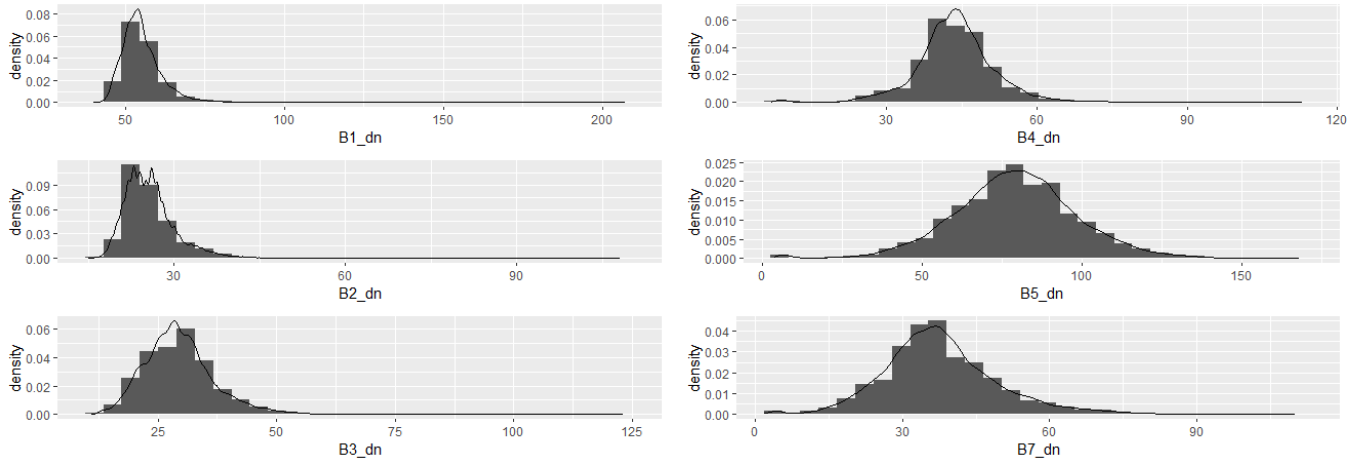
(B) Tomada el 13 de diciembre de 1984

Figura 4.1: Imagen satelital de la ciudad de Harare tomadas en dos tiempos diferentes (combinación de bandas representadas por color rojo, verde y azul)

La Figura 4.2 presenta la distribución de cada banda espectral. Se puede observar que los valores se encuentran en un rango de 0 a 256, con una distribución asimétrica en las primeras tres bandas espectrales y una distribución más simétrica en las bandas 4, 5 y 7. Estos valores son la base del preprocesamiento de la información satelital. Para el preprocesamiento de imágenes satelitales se hizo uso de las siguientes librerías:

- *raster*: es una paquetería que ofrece una amplia gama de funcionalidades para manipular, analizar y visualizar datos,en aplicaciones de análisis espacial y geoespacial (Bivand et al., 2023).
- *Rstoolbox*: es una paquetería con herramientas para importar datos, preprocesamiento, análisis de datos y visualización gráfica (Leutner et al., 2024).
- *sp*: es una que proporciona clases y métodos para representar y analizar objetos geográficos en R, lo que es fundamental para el análisis geoespacial y la cartografía (Pebesma et al., 2023).
- *sf*: es una paquetería que trabaja con datos espaciales y geoespaciales. A diferencia de la paquetería "sp", que se basa en clases y estructuras de datos propias, "sf" es una paquetería más moderna y eficiente en términos de rendimiento para el manejo de datos espaciales (Pebesma et al., 2023).
- *glcm*: es una paquetería para calcular índices de textura usando las matrices de co-ocurrencia de niveles de gris (GLCM). El tamaño de la ventana y el desplazamiento son determinados por el usuario (Zvoleff, 2022).

- *rgdal*: es una herramienta que realiza operaciones de lectura y escritura en diversos formatos de datos geográficos y espaciales; se basa en la biblioteca GDAL (Geospatial Data Abstraction Library), que es una biblioteca de código abierto ampliamente utilizada para trabajar con datos geoespaciales en diferentes formatos y proyecciones cartográficas (Hijmans et al., 2023).



(A) Distribución de los ND de las bandas 1, 2 y 3

(B) Distribución de los ND de las bandas 3, 4 y 7

Figura 4.2: Distribución de los números digitales (ND) en las bandas espectrales de la zona de estudio. Bi_dn *se refiere a la i-ésima banda espectral, con* $i = 1, 2, 3, 4, 5, 7$.

4.1.2 Transformación de la imagen satelital a un conjunto de datos

Este proceso, es importante comprender la naturaleza de la información que se tiene, por ejemplo, su origen y procedencia. Retomando los conceptos del Capítulo 2, la única banda que posee una resolución espacial distinta es la banda 6, cuya resolución es de 120x120, debido a que no está sujeta a las mismas condiciones de resolución de otras bandas, la cual es de 30x30, y siguiendo la idea de Kamusoko (2019) se opta por eliminarla del análisis y el preprocesamiento, pues al comparar directamente, se estaría mezclando información de diferentes escalas, lo que podría distorsionar los resultados. De esta manera, se cuenta con seis covariables, distinguidas en este trabajo por `B1_s`, `B2_s`, `B3_s`, `B4_s`, `B5_s`, `B7_s`.

A pesar de que es viable realizar un análisis empleando los datos de las imágenes satelitales sin una corrección radiométrica, los resultados de tales análisis podrían verse influenciados por variaciones en la iluminación y otras distorsiones radiométricas presentes en los datos no corregidos.

Se implementa la corrección radiométrica que corrige problemas como el ruido visual causado por nubes, neblina, entre otros, como se describe en la sección 2.2.1. Los valores de $offset$ y $gain$ contenidos en el metadato se presentan en la Tabla 4.2.

Banda	Offset	Gain
Banda 1	-2.191	0.671
Banda 2	-4.162	1.322
Banda 3	-2.214	1.044
Banda 4	-2.386	0.876
Banda 5	-0.490	0.120
Banda 6	1.182	0.055
Banda 7	-0.216	0.066

Tabla 4.2: Medidas de offset y gain procedente de los datos del satelite Landsat 5

En virtud de la existencia de varios métodos para llevar a cabo la corrección radiométrica y de acuerdo

a lo descrito en la sección 2.2. Se opto por usar el método COSTZ para la corrección radiométrica como en Kamusoko et al., (2019). esto se realizó usando la función *radCor* del paquete *RStoolbox* (Leutner et al., 2024).

Es clave señalar que cada método se debe aplicar de manera uniforme en todas las bandas, pues usar bandas con distintos métodos de corrección radiométrica podría generar incoherencias en nuestros resultados y contaminar la información obtenida. Además, los resultados derivados de esta corrección serán empleados en la construcción de variables subsiguientes.

El conjunto de datos resultante se presenta como un archivo Raster, en el cual se identifican los píxeles mediante sus coordenadas (x, y). Aquí, x y y corresponden a las coordenadas geográficas que representan la ubicación en la zona de estudio asociada al píxel. Para cada banda y píxel (x, y) se asigna un valor de intensidad de brillo, $f_j(x, y),\ j = 1, 2, \ldots, B$ donde B es el número de bandas. De esta manera la estructura de los datos tiene 6 columnas y R renglones:

$$\begin{bmatrix} f_1(x_1, y_1) & \ldots & f_5(x_1, y_1) & f_7(x_1, y_1) \\ f_1(x_2, y_2) & \ldots & f_5(x_2, y_2) & f_7(x_2, y_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(x_R, y_R) & \ldots & f_5(x_R, y_R) & f_7(x_R, y_R) \end{bmatrix}. \tag{4.1}$$

Es decir cada columna corresponde a las bandas Bi_dn, $i = 1, 2, 3, 4, 5, 7$.

La aplicación de la reproyección es un requisito para vincular las etiquetas de la cobertura del suelo con las variables predictivas. En su estado original cada píxel se asocia con una coordenada geográfica, estableciendo así la base para asignar una etiqueta a dicho píxel.

Durante el proceso de reproyección se identificó un error en la ubicación de la imagen. Para corregir este aspecto se empleó la función *CRS* del paquete *sp* (Pebesma et al., 2024). Esta función se aplicó al metadato inicial, confirmando que la imagen se encontraba en el hemisferio norte. Por lo tanto, se llevó a cabo la reproyección, ya que los píxeles no fueron georreferenciados correctamente en el momento de la captura. La función *projectRaster* del paquete *raster* (Hijmans et al., 2023) permitió ajustar la ubicación de la imagen satelital al sur de África.

La creación de los índices espectrales fue realizada a través de la librería *raster* (Hijmans et al., 2023), haciendo uso de la función *spectralIndices*. A través de este proceso, fueron calculados sólo tres índices espectrales (NDVI, SAVI y MSAVI) cuyos detalles se pueden encontrar en la Tabla 2.3.

Se usará óla notación para $g_k(x_i, y_j)$ para los índices tanto espectrales como de textura. De manera que NDVI, SAVI y MSAVI son identificados como: $g_1(x_i, y_j), g_2(x_i, y_j), g_3(x_i, y_j)$ con $i.j = 1, 2, .., R$ respectivamente. Es decir, el conjunto de datos se presenta de la siguiente manera es:

$$\begin{bmatrix} f_1(x_1, y_1) & f_2(x_1, y_1) & \ldots & f_7(x_1, y_1) & g_1(x_1, y_1) & g_2(x_1, y_1) & g_3(x_1, y_1) \\ f_1(x_2, y_2) & f_2(x_2, y_2) & \ldots & f_7(x_2, y_2) & g_1(x_2, y_2) & g_2(x_2, y_2) & g_3(x_2, y_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f_1(x_R, y_R) & f_2(x_R, y_R) & \ldots & f_7(x_R, y_R) & g_1(x_R, y_R) & g_2(x_R, y_R) & g_3(x_R, y_R) \end{bmatrix}. \tag{4.2}$$

En cuanto a los índices de textura, se usó la función *glcm* de la librería *glcm*(Zvoleff, 2022). Que requiere como entrada el conjunto de datos de tipo raster y las bandas pertinentes para el cálculo, así como la dirección (por defecto dirección de 45°), además del rango o ventana y los índices de textura deseados.

Sólo se calculan los índices de las bandas 4, 5 y 7 usando una ventana de tamaño 3x3 y los índices seleccionados fueron: media, varianza, homogeneidad y entropía. Se optó por calcular los índices en dos direcciones opuestas a 0° (izquierda y derecha), como se describe en la sección 2.2

En este proceso se generaron cuatro covariables por cada banda elegida. En total, se crearon 12 variables correspondientes a los índices de textura, denotadas como: $g_4(x_i, y_i), g_5(x_i, y_i), \ldots, g_{15}(x_i, y_i)$ con $i = 1, 2, \ldots, R$, respectivamente.

En consecuencia, las reglas de clasificación se construirán con la siguiente información:

$$\begin{bmatrix} f_1(x_1,y_1) & f_2(x_1,y_1) & \ldots & f_6(x_1,y_1) & g_1(x_1,y_1) & g_2(x_1,y_1) & \ldots & g_{15}(x_1,y_1) \\ f_1(x_2,y_2) & f_2(x_2,y_2) & \ldots & f_6(x_2,y_2) & g_1(x_2,y_2) & g_2(x_2,y_2) & \ldots & g_{15}(x_2,y_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ f_1(x_R,y_R) & f_2(x_R,y_R) & \ldots & f_6(x_R,y_R) & g_1(x_R,y_R) & g_2(x_R,y_R) & \ldots & g_{15}(x_R,y_R) \end{bmatrix}, \tag{4.3}$$

donde cada columna está asociada a: `B1_s, B2_s, B3_s, B4_s, B5_s,B7_s, NDVI, SAVI, MSAVI, glcm_entropy_4, glcm_homogenity_4, glcm_mean_4, glcm_variance_4, glcm_entropy_5, glcm_homogenity_5, glcm_mean_5, glcm_variance_5, glcm_entropy_7, glcm_homogenity_7, glcm_mean_7, glcm_variance_7`. Una columna adicional es sobre la etiqueta del tipo de suelo, es decir, la variable objetivo $\mathbf{Y}$.

El número de renflones corresponde a R= 7,769 píxeles: 2,453 son de la clase "Agr" y 5,279 de "NoAgr". En Figura 4.3 se representa la ubicación geográfica de un píxel en la imagen satelital como un punto, de tal manera que un punto geográfico representa a todo el píxel. Esta representación puede llevar a errores, ya que en un mismo píxel pueden llegar a convivir más de un tipo de suelo.

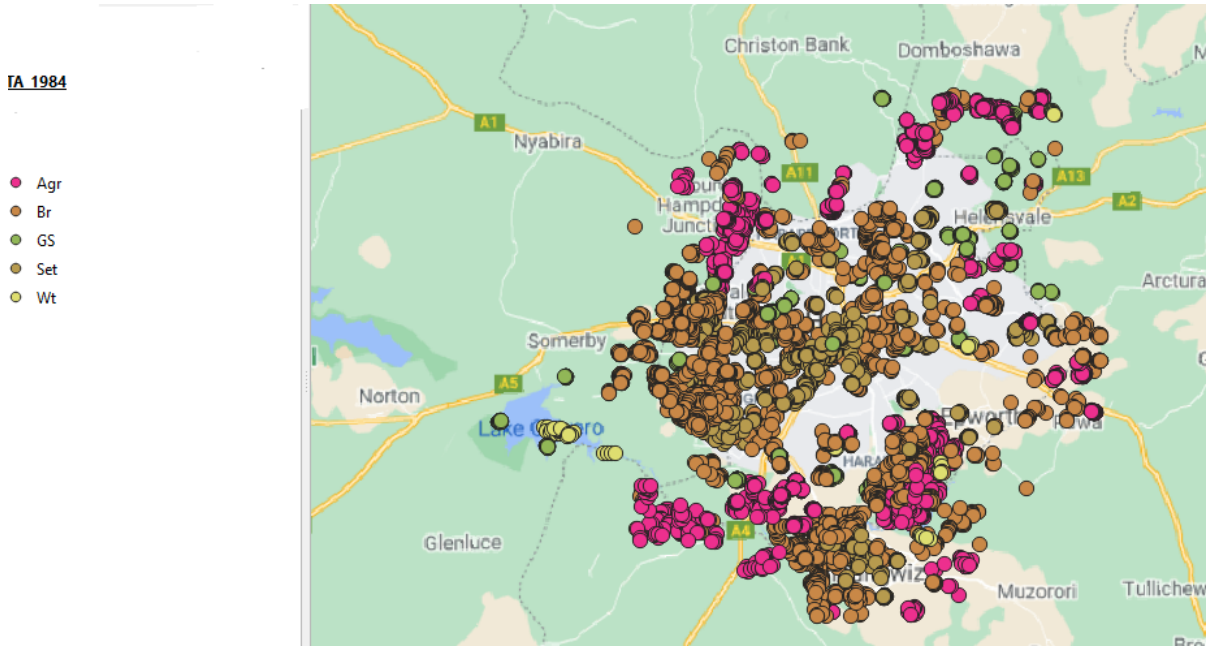


Figura 4.3: Etiquetas de cobertura del suelo y su distribución geográfica.[1]

En la Figura 4.3 se puede observar la dispersión de las etiquetas de suelo sobre la Ciudad de Harare con una concentración en el centro del suelo desnudo (Set) y (Br), mientras que agricultura se concentra en las orillas de la ciudad.

[1]Imagen creada en QGIS: Descarga el software en https://www.qgis.org/es/site/

4.2 Análisis descriptivo

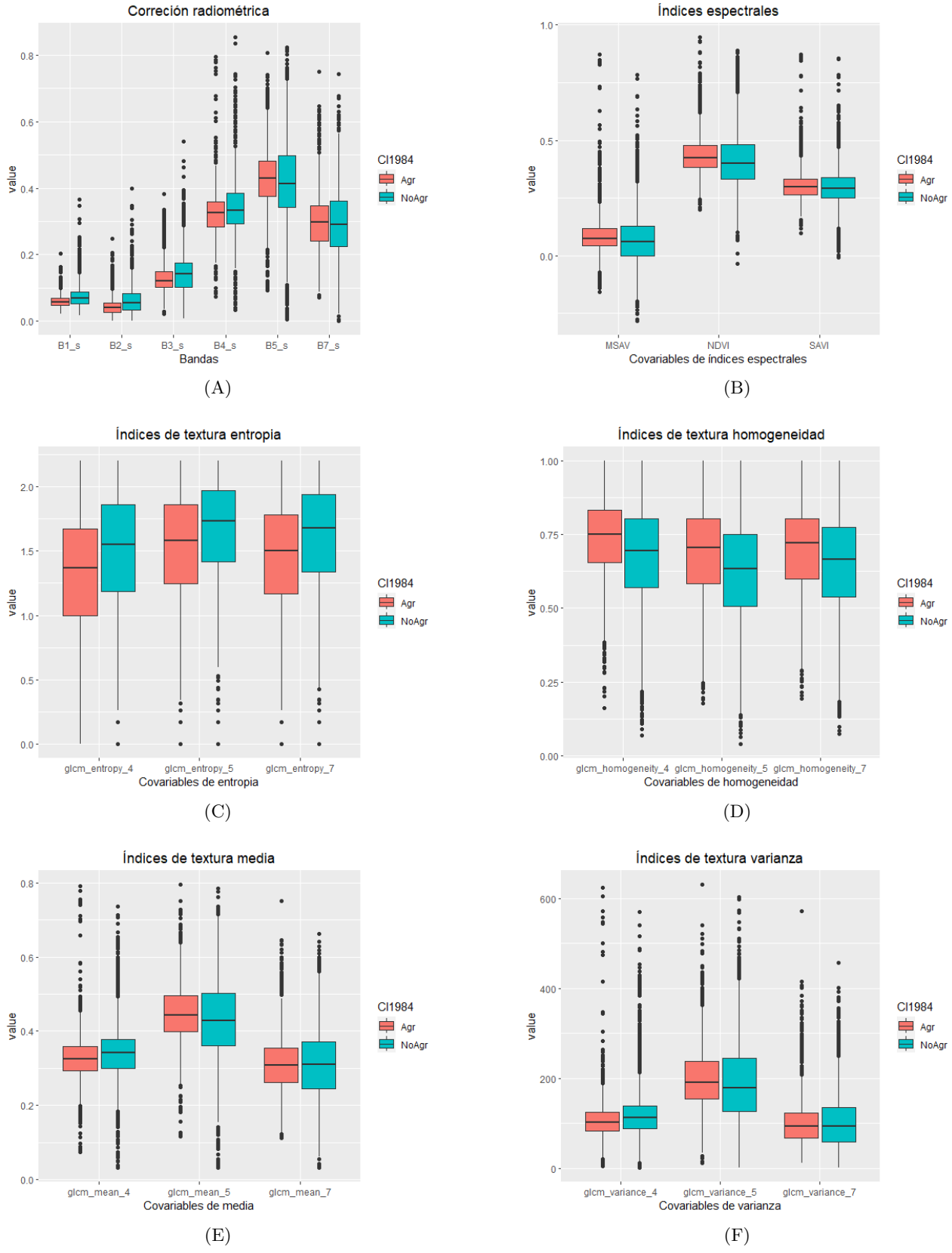


Figura 4.4: Boxplot por tipo de cobertura: A) correción radiométrica de las bandas,B) índices espectrales, así como índices de textura considerando: C) entropía,D) homogeneidad,E) media y F) varianza. Los tipos de suelo son definidos como: "Agr" y "NoAgr" con un total de 2,453 y 5,279 observaciones, que representan 31% y 69% respectivamente.

La Figura 4.4, presenta la distribución de las covariables distinguidas por grupo "Agr" (caja roja) y "NoAgr" (caja azul). Notando en la Figura 4.4 A) la distribución de las bandas, en todas las bandas se notan datos varios atípicos. Además, las bandas 4 y 7 las medias de sus valores son similares entre grupos indicando que posiblemente no contribuyan en la distinción entre clases. También en la Figura 4.4 B) se observa que la covariable SAVI tiene una distribución similar en ambos grupos.

En cuanto a los índices de textura, 2 de las 12 variables generadas hay comportamientos similares en su distribución: mean (4.4 E)) y variance (4.4 F)) de la banda 4 y 7. El resto de los índices si presentan diferencias.

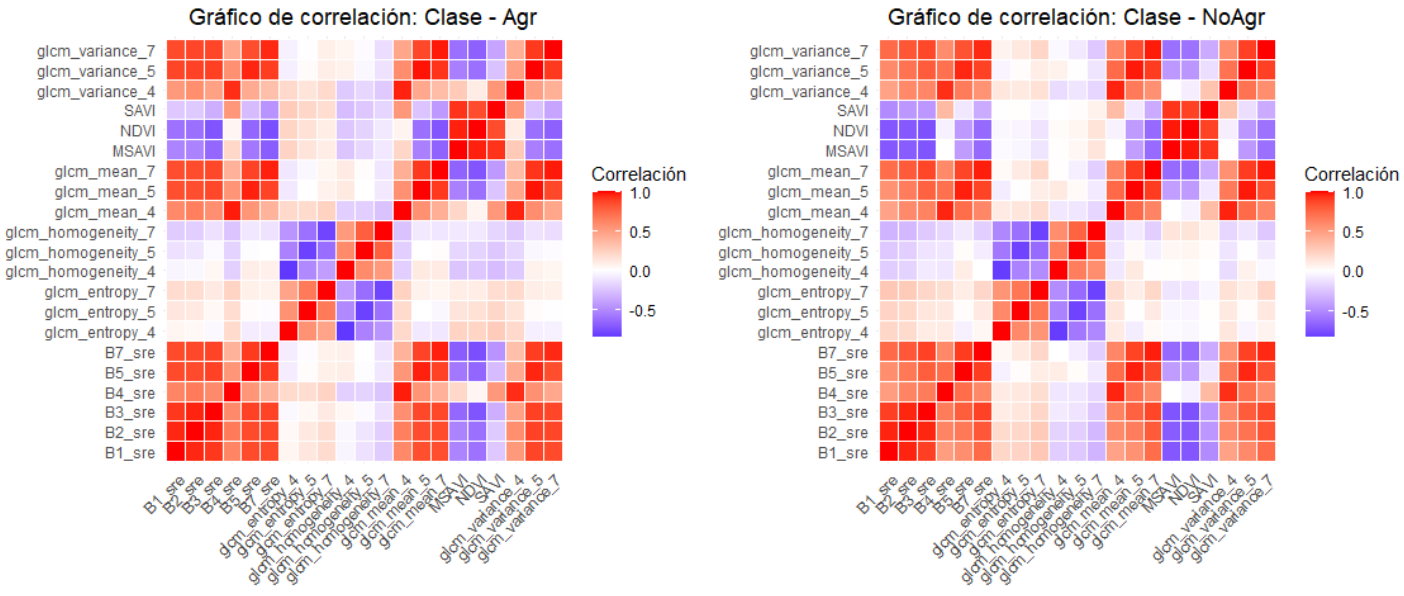


Figura 4.5: Gráficos de correlación entre las 21 covariables para el grupo "Agr" y el grupo "NoAgr".

La Figura 4.5 muestra las correlaciones entre las covariables de los grupos 'Agr' y 'NoAgr'. Las covariables `B1_sre, B2_sre, B3_sre` presentan correlaciones distintas con las 18 covariables restantes en los dos grupos. Esta diferencia también es notoria en las covariables relacionadas con entropía, homogeneidad, índices espectrales, `B1_sre` y varianza. Identificar estas diferencias de correlación entre variables puede proporcionar información sobre qué variables distinguen mejor los grupos cuando interactúan

4.3 Aplicación de modelos de clasificación con datos de junio de 1984

En esta seccion se presentan los resultados de la aplicación de modelos de clasificación supervisada descritos en la sección 3.3: regresión binaria, K-vecinos más cercanos (KNN), máquinas de soporte vectorial (SVM) y bosques aleatorios.

Primero se ajustó una regla para cada modelo utilizando las 21 variables obtenidas en el procesamiento y el conjunto completo de observaciones. Para los algunos modelos que requieren la definición de hiperparámetros se definió una malla.

La selección de los valores dentro de la malla se realizó considerando los recursos computacionales y valores recomendados por las librerías usadas para cada método de clasificación.

Tal como se abordó en el capítulo 3, para cada regla de clasificación se estimó su poder predictivo con el método Repeated hold out y se considera la sensitividad, especificidad y exactitud como métricas a comparar entre modelos.

La división en el conjunto de entrenamiento (train) y de prueba(test) se realizó usando muestreo estratificado, es decir, la proporción de 31% en "Agr" y 69% en "NoAgr" se mantuvo en ambos subconjuntos.

Las reglas de clasificación se ajustaron con dos enfoques: con datos desbalanceados y datos balanceados. Es decir, cuando se trabajó con datos desbalanceados el conjunto de entrenamiento considera: 1,926 observaciones en “Agr” y 4,289 en “NoAgr”. Por otro lado, cuando se trabaja con datos balanceados se aplicó un submuetreo en el conjunto de entrenamiento (train), con el fin de tener un balance entre los grupos de interés, es decir, contar 1,926 observaciones en "Agr" y 1,926 en "NoAgr". Para obtener mayor información sobre técnicas para datos desbalanceados, se puede consultar (Tarawneh et al., 2022).

4.3.1 Construcción de las reglas de clasificación y resultados de la estimación del poder predictivo

A continuación se describen algunas consideraciones y resultados al construir las reglas usando las n = 7,732 observacines.

Regresión binaria

Los modelos de regresión binaria probit y logit comparten la misma estructura, con la única diferencia en la función liga utilizada. Esto significa que el procedimiento para los modelos de regresión logit y probit será similar.

Se desarrollaron dos versiones para cada modelo:

1. Una versión simple que utiliza sólo los efectos principales de todas las variables en el conjunto de datos (sin interacciones).
2. Una versión que incorpora interacciones de segundo orden y selección de variables con un método por pasos.

Para la selección de variables se empleó el método por pasos (stepwise) con dirección ’forward’ debido al tiempo de ejecución del algoritmo. Este proceso comienza con un modelo que no incluye ninguna covariable predictora. Posteriormente, se agregan covariables una a una al modelo basándose en el criterio BIC.

También se exploró la regresión lasso con las dos versiones, pero sólo con la loga logit usando una matriz diseño de 21 columnas para el modelo sin interacciones y usando una matriz de 231 columnas para el modelo con interacciones. Además, se usó el método de validación cruzada con 10 pliegues para calibrar el valor óptimo de λ con base en la tasa de clasificación errónea global. En total, se generaron seis reglas de clasificación

En la Tabla 4.3 se presentan las cinco variables más relevantes identificadas mediante la aplicación de regresión binaria sin interacciones. Destacando la importancia de la corrección radiométrica de las seis bandas espectrales, así como de los índices de textura, en la predicción de los modelos.

La segunda versión de estos modelos considera interacción entre variables y selección de variables.

En la Tabla 4.4 se muestran los valores de las cinco covariables más importantes para cada modelo de regresión binaria. Se resaltan las interacciones entre covariables, como: `B7:NDVI`, `glcm_mean_7:NDVI`, `B3_s:SAVI`, `glcm_mean_4:MSAVI`, `B4_s:MSAVI`. La Figura 4.5 revela diferencias en las correlaciones de las covariables dentro de cada grupo.

Ademas, cada una de las 6 reglas se uso con punto de corte 0.5 o bien definiendo un umbral usando las funciones *coords* y *roc* de la librería *pROC* (Robin, 2011), de acuerdo con el criterio de Youden. Como resultado, se generaron 12 versiones de modelos de regresión binaria: seis modelos con y sin

Variable	Regresión lasso	Regresión logit	Regresión probit
glcm_mean_4	✓	✓	×
glcm_mean_5	✓	✓	×
glcm_mean_7	×	✓	×
B1_sre	✓	✓	✓
B2_sre	✓	×	✓
B3_sre	×	✓	✓
B4_sre	×	×	✓
B5_sre	×	×	✓
B7_sre	✓	×	×

Tabla 4.3: Las 5 covariables más importantes en cada modelos representadas por ✓ si fueron consideradas en los modelos de regresión logit y probit no se aplicaron interacciones y ni selección de variables. En la regresión lasso no se aplicaron interacciones de covariables.

Variable	Regresión lasso	Regresión logit	Regresión probit
B3_sre:SAVI	✓	×	×
glcm_mean_7:NDVI	✓	×	×
B1_sre:glcm_homogeneity_7	✓	×	×
B1_sre:B2_sre	✓	×	×
B7_sre:NDVI	✓	×	×
B1_sre	×	✓	✓
glcm_mean_4:MSAVI	×	✓	×
MSAVI:B4_sre	×	✓	×
glcm_mean_4	×	✓	✓
glcm_mean_5	×	✓	×
B5_sre	×	×	✓
glcm_variance_7	×	×	✓
MSAVI	×	×	✓

Tabla 4.4: Las 5 covariables más importantes en cada modelo representadas por ✓ si fueron consideradas en los modelos de regresión logit y probit aplicando selección de variables e interacciones.

interacciones, y en algunos casos con selección de variables, utilizando un punto de corte de 0.5; así como seis modelos adicionales en los que se ajustó el punto de corte.

Máquinas de soporte vectorial con kernel radial

El método SVM tiene varios hiperparámetros y depende de analizar todas los datos para encontrar los vectores de soporte, por lo que tiene mayor costo computacional. Dependiendo de la cantidad de valores en la malla de hiperparámetros, el tuneo exige un mayor tiempo.

Además que el método SVM se basa en distancias, es importante que las covariables estén en la misma escala. La estandarización ayuda a garantizar que todas las variables tengan un impacto comparable en el modelo, previniendo así que una variable con valores más grandes o pequeños influya de manera desproporcionada en el resultado.

En este trabajo, se optó por usar el kernel radial y se calibraron dos hipérparametros: costo (cost) con una malla de 4 valores y gamma con una malla de 5 valores, además se estandarizaron todas las covariables. Cabe mencionar que durante el proceso de tuneo se podría considerar la inclusión de los diferentes tipos de kernel disponibles (radial, lineal, polinomial, entre otros); sin embargo, debido al costo computacional que implica no se realizó el tuneo del kernel. En la Figura 4.6 se puede observar que a medida que va en aumento el parámetro gamma disminuye el error. Además, el hiperparámetro cost resulta ser mejor con un valor aproximado a 100.

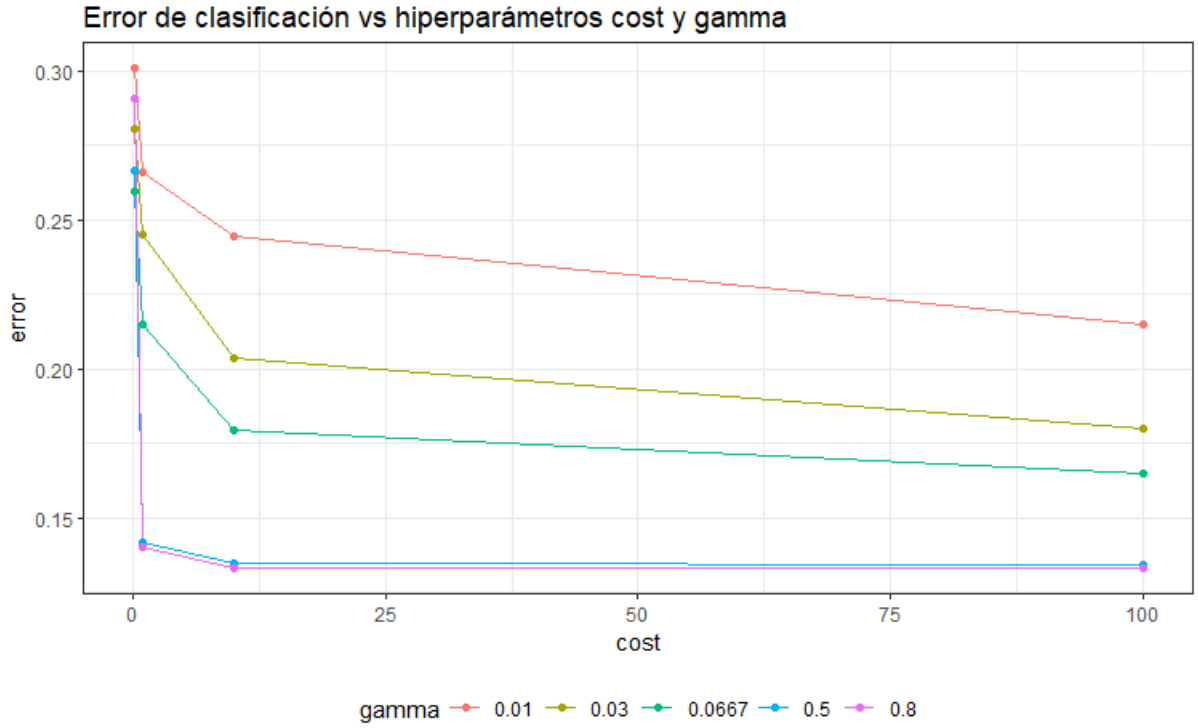


Figura 4.6: Errores estimados usando el método SVM con el kernerl radiasl para cada valor en las mallas de los hiperparámetros del costo y gamma, considerando K-Cros validation con K=10.

Bosques aleatorios

En el contexto de los bosques aleatorios, el parámetro mtry representa el número de variables predictoras consideradas en cada división de un árbol de decisión durante la construcción del bosque. Por defecto, se utiliza la raíz cuadrada del número total de covariables, donde p denota la cantidad de covariables en el conjunto de datos. En este caso, se utilizan 7 variable. En este trabajo se tuneo usando la malla para $mtry = \Big\{3, 4, 5, 6, 7\Big\}$.

Adicionalmente, se evaluaron diferentes aspectos de la estructura de cada árbol en el bosque, incluyendo diferentes valores de profundidad, estableciendo en la malla de tuneo a $nodesize = \Big\{2, 10, 15\Big\}$. Además, el número de árboles se calibró con una malla de valores entre 100 y 500. Para medir la impureza se estableció el criterio Gini. En la Figura 4.7 se destaca la importancia de las variables consideradas. Se puede observar que las variables derivadas de los índices de textura vuelven a ser importantes, además de las

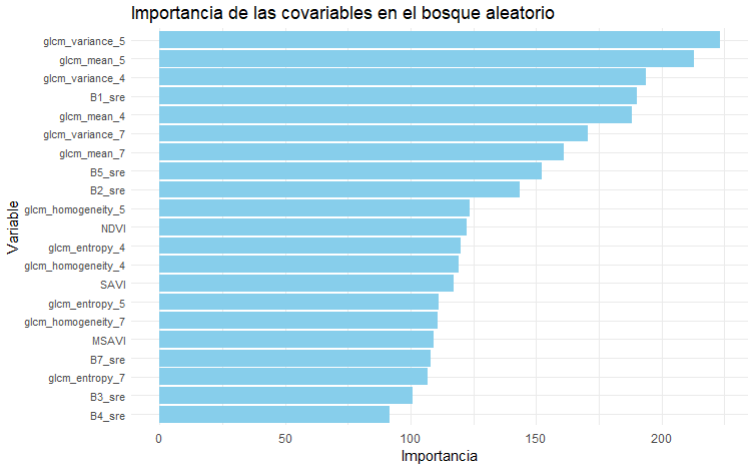


Figura 4.7: Importancia de las covariables, ordenadas de mayor a menor al usar bosques aleatorios.

K vecinos más cercanos (KNN)

Para este método se consideró una malla de 2 a 25 vecinos. La Figura 4.8 muestra el error obtenido mediante el método—. También se puede observar que a partir del vecino 3 el error de predicción aumenta. Cabe mencionar que antes de aplicar esta regla se estandarizan las variables.

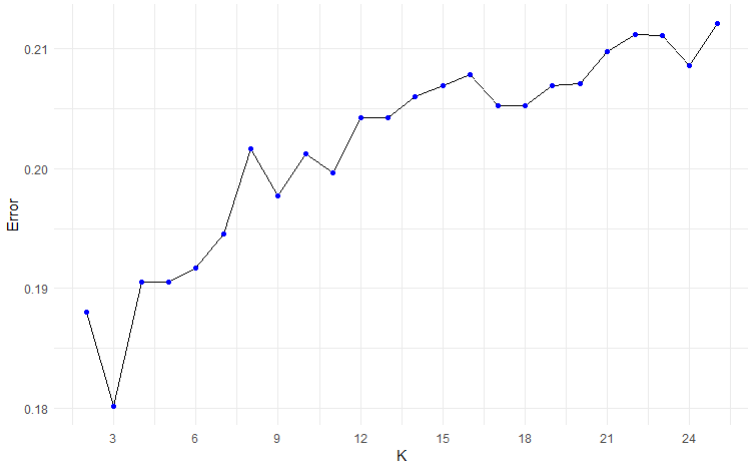


Figura 4.8: Error de clasificación que consideró de 2 a 25 vecinos

Resultados de la estimación del poder predictivo para datos de junio de 1984

Considerando todas las reglas descritas en el apartado anterior, se procedió a evaluar su capacidad predictiva.

Para estimar el poder predictivo se aplicó el método *Repeated Hold-Out* con 100 repeticiones. Además, como ya se describió antes se usó *K-Cross Validation* para encontrar los hiperparámetros óptimos (con K = 10). El 80% de los datos se utilizó como conjunto de entrenamiento y el restante se reservó como conjunto de prueba.

Para optimizar este proceso se implementaron funciones de programación en paralelo. Esto resulta en una reducción del tiempo de procesamiento y permite probar diversas configuraciones de manera simultánea.

Para la aplicación de programación en paralelo se utilizaron las siguientes librerías:

- *parallelly*: es una librería de R que proporciona funciones que mejoran el paquete parallel. Por ejemplo, la función *availableCores* devuelve el número de núcleos de CPU (Bengtsson, 2024)
- *purrr*: es una librería que mejora el kit de herramientas de programación funcional de R (Wickham et al., 2023).
- *furrr*: es una librería de R que combina la familia de funciones de mapeo de *purrr* con las capacidades de procesamiento paralelo (Vaughan et al., 2022).

En la Tabla 4.5 se muestran los métodos de clasificación supervisada que fueron considerados en este trabajo, junto con la malla o capa de hiperparámetros que fue considerada en el tuneo. En la Tabla 4.5 también se muestran los valores obtenidos en la creación de la regla de clasificación cuando se usaron las n observaciones, así como los valores obteneidos en la estimación del poder predictivo, representados con el primer, segundo y tercer cuantil.

Tabla 4.5: Resultado de tuneo e hiperparámetros con clases desbalanceadas

Tuneo de hiperparámetros			
Modelo	Capa de hiperparámetros	Valores de la Regla de clasificación con n observaciones	Regla de clasificación con n_{train}
R. logit simple	No aplica	pt = 0.5	Q1: pt = 0.5, Q2: pt = 0.5, Q3: pt = 0.5
R. probit simple	No aplica	pt = 0.5	Q1: pt = 0.5, Q2: pt = 0.5, Q3: pt = 0.5
R. lasso simple	$\lambda_1, \lambda_2, ..., \lambda_{300}$	pt = 0.5, $\lambda = 0.00095$	Q1: pt = 0.5, $\lambda = 0.00025$, Q2: pt = 0.5, $\lambda = 0.00045$, Q3: pt = 0.5, $\lambda = 0.00103$
R. logit con interacciones y selección de variables	No aplica	pt = 0.5	Q1: pt = 0.5, Q2: pt = 0.5, Q3: pt = 0.5
R. probit con interacciones y selección de variables	No aplica	pt = 0.5	Q1: pt=0.5, Q2: pt = 0.5, Q3: pt = 0.5
R. lasso con interacciones	$\lambda_1, \lambda_2, ..., \lambda_{300}$	pt = 0.5, $\lambda = 0.00048$	Q1: pt = 0.5, $\lambda = 0.00019$, Q2: pt = 0.5, $\lambda = 0.00036$, Q3: pt = 0.5, $\lambda = 0.00480$
R. logit simple	$pt \in (0, 1)$	pt = 0.67	Q1: pt = 0.68, Q2: pt = 0.67, Q3: pt = 0.69
R. probit simple	$pt \in (0, 1)$	pt = 0.67	Q1: pt = 0.67, Q2: pt = 0.65, Q3: pt = 0.68

R. lasso simple	$\lambda_1, \lambda_2, ..., \lambda_{300}$	pt = 0.67, $\lambda = 0.00022$	Q1: pt = 0.67, $\lambda = 0.00013$, Q2: pt = 0.67, $\lambda = 0.00022$, Q3: pt = 0.68, $\lambda = 0.00035$
R. logit con interacciones y selección de variables	$pt \in (0, 1)$	pt = 0.67	Q1: pt = 0.67, Q2: pt = 0.66, Q3: pt= 0.69
R. probit con interacciones y selección de variables	$pt \in (0, 1)$	pt=0.68	Q1: pt = 0.68, Q2: pt = 0.66, Q3: pt = 0.69
R. lasso con interacciones y selección de variables	$\lambda_1, \lambda_2, ..., \lambda_{300}$	pt = 0.73, $\lambda = 0.00048$	Q1: pt = 0.63, $\lambda = 0.00009$, Q2: pt = 0.65, $\lambda = 0.00021$, Q3:pt=0.66, $\lambda = 0.00034$
SVM	cost=$[10^{(-1:2)}]$ $\gamma = \Big\{0.01, 0.03, 0.0667,\ 0.5,\ 0.8\Big\}$	cost = 100, $\gamma = 0.5$	Q1: cost = 10, $\gamma = 0.5$, Q2: cost = 10, $\gamma = 0.65$, Q3: cost = 10, $\gamma = 0.8$
Bosques aleatorios	$ntree = \Big\{100, 500\Big\}$, $nodesize = \Big\{2, 10, 15\Big\}$, $mtry = \Big\{3, 4, 5, 6, 7\Big\}$	ntree = 500, nodesize = 10, mtry = 4	Q1: ntree = 100, nodesize = 2, mtry =3 , Q2: ntree = 100, nodesize = 10, mtry =4 , Q3: ntree = 500, nodesize = 15, mtry = 5
KNN	$K = \Big\{1, 2, \cdots, 25\Big\}$	K=3	Q1: K=3, Q2: K=3, Q3: K=4

Los resultados del poder predictivo para grupos desbalanceados se presentan en la Tabla 4.6. La columna 'Desempeño aparente' muestra las métricas calculadas utilizando únicamente el conjunto de entrenamiento, mientras que la columna 'Desempeño real' muestra las métricas calculadas con el conjunto de prueba.

Se observa que, con datos desbalanceados y el modelo de bosques aleatorios, el grupo de interés ('Agr') obtuvo una precisión del 82% similar a la del conjunto con una muestra mayor de entrenamiento. Esto sugiere que los resultados para nuevas observaciones podrían ser similares.

En términos generales, el método de SVM con kernel radial tuvo un buen desempeño en su exactitud (85.9%), pero fue menos eficaz en la predicción del grupo 'Agr' (72%) en comparación con otros modelos de regresión binaria con ajustes de hiperparámetros. Los modelos de regresión binaria con el punto de corte tradicional presentaron el peor desempeño.

La estimación del poder predictivo resalta estas diferencias en las predicciones. Como se puede ver los modelos más complejos o modernos no necesariamente superarán a los métodos más simples o tradicionales. A pesar de ello, el mejor modelo que resultó es bosques aleatorios por tener la mejor

precisión y exactitud.

Modelo	tuneo	Desempeño aparente (datos train)			Desempeño real (datos test)		
		Grupo Agr	Grupo NoAgr	Global	Grupo Agr	Grupo NoAgr	Global
Bosques Aleatorios	si	96.9	99.8	98.9	86.0	87.5	87.1
SVM	si	99.7	99.9	99.8	72.0	92.3	85.9
KNN	si	82.5	92.3	89.2	70.4	86.3	81.3
R. lasso interacciones	si	76.7	73.9	74.8	75.2	72.9	73.6
R. logit interacciones y selección de variables	si	77.2	67.9	70.9	76.4	67.5	70.3
R. probit interacciones y selección de variables	si	52.6	86.9	70.7	52.0	86.4	70.1
R. probit simple	si	76.5	69.1	71.5	75.7	68.8	71.0
R. logit simple	si	77.5	68.4	71.3	76.7	68.2	70.9
R. lasso simple	si	78.0	67.7	71.0	77.2	67.5	70.6
R. lasso con interacciones	no	50.3	89.5	77.0	48.6	88.9	76.1
R. logit interacciones y selección de variables	no	43.5	88.6	74.3	43.3	88.5	74.2
R. probit interacciones y selección de variables	no	42.2	89.1	74.2	42.0	89.0	74.1
R. probit simple	no	42.2	89.1	74.2	41.9	88.9	74.0
R. logit simple	no	43.3	88.7	74.3	43.0	88.6	74.1
R. lasso simple	no	42.6	89.0	74.3	42.3	88.9	74.1

Tabla 4.6: Poder predictivo de las reglas de clasificación para datos desbalanceados

Modelo	tuneo	Desempeño aparente (datos train)			Desempeño real (datos test)		
		Grupo NoAgr	Grupo Agr	Global	Grupo NoAgr	Grupo Agr	Global
Bosques Aleatorios	si	99.8	99.9	99.9	84.5	81.7	82.6
SVM	si	99.9	99.8	99.9	81.9	81.0	81.3
KNN	si	89.7	84.7	81.6	81.6	70.7	74.1
R. lasso interacciones	si	76.3	69.2	72.5	78.0	71.0	73.2
R. logit interacciones y selección de variables	si	77.8	68.7	72.9	75.9	67.7	70.3
R. probit interacciones y selección de variables	si	73.3	70.9	71.1	78.9	65.4	69.7
R. probit simple	si	76.3	69.6	72.9	75.7	68.8	71.0
R. logit simple	si	77.5	69.6	73.6	62.2	61.9	63.6
R. lasso simple	si	71.0	63.7	70.0	72.2	68.5	73.6
R. lasso interacciones	no	78.2	68.2	73.5	70.7	69.6	71.2
R. logit interacciones y selección de variables	no	77.4	67.6	72.5	76.7	66.9	70.0
R. probit interacciones y selección de variables	no	77.1	67.5	72.3	76.6	66.9	70.0
R. probit simple	no	77.1	66.6	71.6	77.7	66.8	70.0
R. logit simple	no	77.9	67.6	72.8	77.6	67.3	70.5
R. lasso simple	no	78.0	67.2	72.6	77.6	66.9	70.3

Tabla 4.7: Poder predictivo de las reglas de clasificación cuando se balancean los grupos en el conjunto de entrenamiento (train)

Por otro lado, la Tabla 4.7 presenta los resultados del poder predictivo cuando se equilibró el conjunto de entrenamiento, es decir, cuando se ajustó la proporción entre los grupos 'Agr' y 'NoAgr'. Aunque la precisión en los bosques aleatorios disminuyó a 84%, este método continúa siendo el mejor.

El cambio más notable se observa en los modelos SVM y KNN. Con los datos equilibrados, la precisión para el grupo 'Agr' aumentó, acercándose a la exactitud de SVM. A pesar de que SVM mantiene un buen desempeño en términos de exactitud, el modelo KNN ofrece resultados similares en 81% de precisión con un ajuste de hiperparámetros más sencillo.

En general, al entrenar un modelo con datos equilibrados, se logró un mejor desempeño predictivo en el grupo 'Agr', a pesar de la disminución en la precisión de los bosques aleatorios. Esta disminución podría deberse a que el modelo fue entrenado con un número menor de observaciones para mantener el equilibrio entre los grupos

En la estimación del poder predictivo, es común comparar el tiempo de ejecución de cada regla de clasificación. En la Tabla 4.8 se muestran los tiempos de ejecución, en minutos, que tomó cada regla para evaluar su poder predictivo utilizando datos desbalanceados. Los métodos más simples fueron los

más rápidos en ejecutarse, mientras que los métodos más complejos o aquellos con un mayor número de hiperparámetros tuvieron tiempos de ejecución más prolongados.

Entre los métodos evaluados, los bosques aleatorios ocuparon el segundo lugar en mayor tiempo de ejecución, en contraste con los modelos de regresión binaria, que fueron los más rápidos y también obtuvieron las mejores precisiones en el grupo de interés. SVM fue el método que presentó el tiempo de ejecución más prolongado, superando a los bosques aleatorios en aproximadamente dos días y medio.

Con base en el tiempo de ejecución y las métricas observadas, se concluye que el mejor modelo de clasificación fue el de bosques aleatorios.

Modelo	Tiempo (minutos)
Regresión probit sin interacciones ni selec.var	0.29
Regresión logit simple sin interacciones ni selec.var	0.31
Regresión lasso simple sin interacciones	5.9
Regresión logit con interacciones y sel.var	8.13
Regresión probit con interacciones y sel.var	11.37
KNN	17.97
Regresión lasso con interacciones	48.69
Bosques aleatorios	296.05
SVM	3734.832

Tabla 4.8: Comparación del tiempo de ejecución aplicando programación en paralelo 8 núcleos. Dado que el punto de corte no implica gran costo computacional, sólo se calculó s'olo se calcul'o para la versión con tuneo.

4.4 Aplicación y resultados del mejor modelo con datos de diciembre de 1984

Como se mencionó anteriormente, durante la recopilación de información se contaba con dos carpetas comprimidas que correspondían a diferentes períodos temporales. En la Figura 4.9 se muestra la imagen satelital correspondiente a diciembre de 1984.

Para esta fecha se aplicaron las mismas técnicas de preprocesamiento en los datos, así como el modelo que tuvo el mejor desempeño predictivo, ver Tabla 4.6. Esto se planteó debido a que el objetivo general de este tipo de proyectos es monitorear y dar seguimiento constante al comportamiento que tiene la cobertura del suelo con el paso de tiempo y así poder clasificar la cobertura de suelo para la generación de mapas.

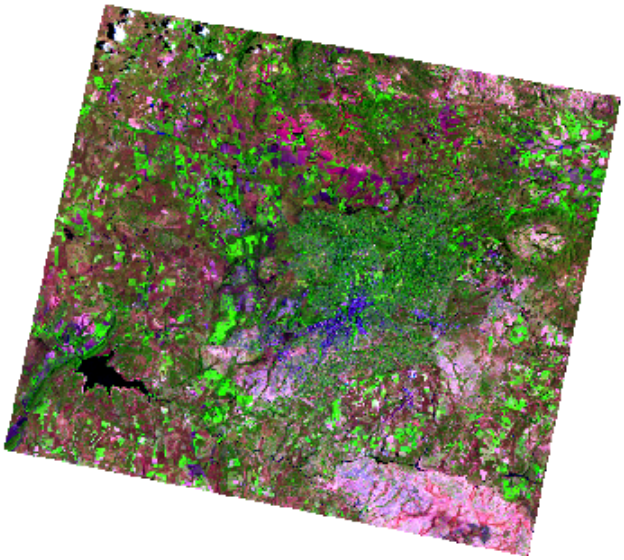


Figura 4.9: Imagen satelital de la Ciudad de Harare capturada el 13 de diciembre de 1984

Se puede observar en la Figura 4.1 que los datos de entrenamiento y las características extraídas de las imágenes satelitales presentaron variaciones, pues la cobertura del suelo parece estar húmeda. Estas variaciones son resultado de los cambios en la reflectancia de la superficie terrestre, ocasionados por el cambio en la vegetación, la humedad del suelo y otros factores ambientales.

Los resultados que se exponen en la Tabla 4.9 se observa que el modelo de bosques aleatorios no logró identificar las características por grupo de la información sobre el grupo de interés agricultura. Estos resultados son de esperarse, pues la nueva información es totalmente diferente con la que fue entrenado el modelo.

Con este resultado se podría decir que, la precisión de los modelos de clasificación supervisada, empleados para la segmentación de la cobertura del suelo mediante imágenes satelitales es susceptible a la variabilidad estacional. Ya que los cambios estacionales alteraron las condiciones de la cobertura del suelo y su representación en las imágenes satelitales, lo que sugiere que es necesario el reentrenamiento de los modelos en lapsos más cortos o cada cambio de estación.

Por lo tanto, es esencial considerar estos factores al entrenar y aplicar modelos de clasificación supervisada.

Modelo	Agr	NoAgr	Global
Bosques aleatorios	0.21	0.90	0.68

Tabla 4.9: Resultados obtenidos al predecir usando el modelo de bosques aleatorios entenado en junio

CAPÍTULO 5

Conclusiones

El principal desafío que enfrenté en la realización de este trabajo fue la limitación de datos nacionales de etiquetado sobre la cobertura del suelo que había disponibles, pues, a pesar de existir recursos disponibles de fuentes alternas en el INEGI u otros organismos, la fiabilidad y manejo de la información fue un aspecto que impidió implementarlos. Esta limitación llevó a la búsqueda de alternativas de datos que incluyan el etiquetado de la cobertura del suelo. A pesar de trabajar con datos de otro país y trabajar con datos desbalanceados, se logró entrenar una regla de clasificación con tasas:

En éste trabajo se proporcionó una guía introductoria para estructurar información a partir de imágenes satelitales con énfasis en consideraciones clave, como el uso de la reproyección y la información que comúnmente es utilizada. En particular, se describe parte del proceso de preparar los datos de una imagen para crear un conjunto de datos que pudiera ser usado en el análisis, aspecto fundamental en la actualidad.

Se observó que índices de textura fueron importantes en la clasificación, de manera que en futuros trabajos se podrían analizar otros que no fueron utilizados en este trabajo, por ejemplo: los índices de textura relacionados con la correlacion, disimilaridad, segundo momento y contraste, todos implementados en la librería *glcm*, así como explorar con diferentes tamaños de ventanas y diferentes direcciones para el cálculo de estas medidas. Además, también se puede explorar el uso de otro tipo de variables relacionadas al análisis espacial, por ejemplo *Sharpening highlights*, *lowpass gaussian*, etc. Para mayor información ver en Gonzalez et al. (2018, Capitulo 3). El aumento de covariables podría generar un aumento en el costo computacional, para resolver este problema se podría hacer uso en el procesamiento de componentes principales (PCA) y así reducir la dimensión del conjunto de datos.

Se recomienda que en futuros trabajos se exploren las redes neuronales convolucionales (CNN). Las redes neuronales convolucionales están estrechamente relacionadas con los índices de textura, ya que este preprocesamiento se basa en la aplicación de convoluciones. Básicamente una convolución consiste en aplicar un conjunto de filtros a través de la imagen de entrada. Cada filtro detecta características específicas en la imagen, como bordes, texturas, entre otros. Durante la convolución, el filtro se desliza sobre la imagen aplicando diferentes operaciones, como fue en el caso del cálculo de los índices de textura, para mayor detalle ver Gonzalez et al. (2018, Capítulo 12).

A pesar de la existencia de modelos de clasificación más avanzados como XGBoost redes neuronales o redes neuronales convolucionales, se optó por modelos clásicos debido a su menor complejidad en términos de hiperparámetros. Las redes neuronales convolucionales no fueron consideradas en este trabajo, debido al costo computacional que representa al momento de aplicar la estimación del poder predictivo, ya que cuenta con varios hiperparámetros a considerar. Además, la cantidad de observaciones etiquetadas se consideró pequeña para aplicar este tipo de información. Sin embargo, en Rai et al., (2020) se puede ver una aplicación de redes neuronales convoluionales y análisis de componentes principales (PCA).

Bosques aleatorios presentó el mejor desempeño, ya que en términos de costo computacional y de estimación del poder predictivo fue el más óptimo con una precisión del 86% y una exactitud del 87%. En contraste, las máquinas de soporte vectorial cuyos resultados de predicción tambien fueron buenos, aunque en términos del tiempo computacional resultó ser el más alto, aproximadamente dos días y medio en el tiempo de ejecución durante la estimación del poder predictivo. El objetivo general de

identificar el tipo de suelo con datos correspondientes a las mismas condiciones del tipo de suelo se logró.

Sin embargo, cuando la regla se uso para predecir en otra estación del año, se observó que el desempeño de redujo a 21% en identificar el grupo "Agr" y 68% de exactitud en la predicción de la regla aplicada. En otras palabras, la regla sólo puede aplicarse a datos donde las condiciones climáticas o estacionales sean similares a los datos utilizados para el entrenamiento. Una opción es obtener datos históricos por estaciones y reentrenar los modelos. También se podría aplicar metodologías relacionadas a series temporales.

Como sugerencia para futuras investigaciones, se propone explorar el uso de imágenes satelitales de mayor calidad y la colaboración con expertos en teledetección. Además, se podría profundizar en la creación de variables específicas para objetivos concretos como la identificación de zonas deforestadas. La aplicación de este enfoque en el contexto mexicano sería especialmente valiosa y podría tener un impacto significativo en la gestión de recursos naturales y la conservación del medio ambiente. Además se sugiere el uso un hardware robusto o máquinas virtuales para el proceso de estimación del poder predictivo y la exploración de modelos más avanzados o modernos, en este tipo de modelos más avanzados o modernos.

Por último, se recomienda aprovechar datos abiertos y enfoques interdisciplinarios para abordar desafíos relacionados con la clasificación de la cobertura del suelo, el tratamiento de datos espaciales y los aspectos que se deben considerar antes de aplicar proyectos relacionados.

Bibliografía

Agresti, A. (2015). *Foundations of Linear and Generalized Linear Models*. Florida: John Willey & Sons.

Bengtsson, H. (2023). Parallelly: Enhancing the 'parallel' Package (R package version 1.36.0). Recuperado de https://CRAN.R-project.org/package=parallelly

Bivand, R., Keitt, T., & Rowlingson, B. (2023). rgdal: Bindings for the 'Geospatial' Data Abstraction Library (R package version 1.5-27) [Software]. Disponible en https://CRAN.R-project.org/package=rgdal

Bivand, R., Pebesma, E., & Gomez-Rubio, V. (2013). *Applied spatial data analysis with R* (2da ed.). Springer.

Chavez, J. P. (1996). Image-Based Atmospheric Corrections—Revisited and Improved. *Photogrammetric Engineering and Remote Sensing*(62), 1025-1036.

Chuvieco Salinero, E. (1996). *Fundamentos de teledetección espacial* (3ra ed.). Madrid: Rialp.

CONABIO. (2020). Programa de información MAD-Mex (Sistema de Monitoreo de Datos de Actividad del Programa REDD+ México, CONABIO-CONAFOR). Recuperado el 12 de diciembre del 2021 de Comisión Nacional para el Conocimiento y Uso de la Biodiversidad: https://monitoreo.conabio.gob.mx/descargas/documentos/MPEG-INEGI-CONABIO-Mad-Mex.pdf

CONAFOR. (2020). *Guía de uso de los productos del Sistema Satelital de Monitoreo Forestal*. Recuperado el 12 de diciembre del 2021 de Comisión Nacional Forestal: https://www.gob.mx/cms/uploads/attachment/file/587193/Gui_a__de__uso__de_los__productos_del_sistema_SAMOF_v1.pdf

Deng, N., Tian, Y., & Zhang, C. (2013). *Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions*. United Satates of America: CRS Press

Friedman, J., Hastie, T., & Tibshirani, R. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2da ed.). New York: Springer. doi:https://doi.org/10.1007/978-0-387-84858-7

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of statistical software*, 33(1), 1-22.

Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4ta ed). Pearson.

Haralick, R., Shanmugam, K., & Dinstein, I. (1973). Textural Features for Image Classification. IEEE *Transactions on Systems, Man and Cybernetics*, 3(6), 610-621.

Hijmans, R. (2023). raster: Geographic Data Analysis and Modeling (R package Version 3.6-26) [Software]. Disponible en https://CRAN.R-project.org/package=raster

INEGI. (2018). *Aspectos técnicos de las imágenes Landsat*. Recuperado el 11 de abril de 2023, de https://en.www.inegi.org.mx/contenidos/temas/imagenes/imgLandsat/doc/Aspec

tos_tecnicos_Landsat.pdf

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2023). *An Introduction to Statistical Learning with Appliations in R.* (2da ed.). New York: Springer. doi: 10.1007/978-1-4614-7138-7

Kamusoko, C. (2019). *Remote Sensing Image Classification in R.* Japón: Springer. doi:10.1007/978-981-13-8012-9_5

Kamusoko, C., Gamba, J., & Murakami, H. (2013). Monitoring Urban Spatial Growth in Harare Metropolitan Province, Zimbabwe. *Advances in Remote Sensing*, 2(4), 322-331. doi:10.4236/ars.2013.24035

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

Leutner, B., Horning, N., Schwalb-Willmann, J., Hijmans, R. J., & Mueller, K. (2024). RStoolbox: Remote Sensing Data Analysis Toolbox (R package Version 0.4.0) [Software]. Disponible en https://github.com/bleutner/RStoolbox

Lillesand, T., Kiefer, R. W., & Chipman, J. (2015). *Remote sensing and image interpretation.* John Wiley & Sons.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2023). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien (R package version 1.7-13) [Software]. Disponible en https://CRAN.R-project.org/package=e1071

Naciones Unidas. (2018). *Marco de indicadores mundiales para los Objetivos de Desarrollo Sostenible y metas de la Agenda 2030 para el Desarrollo Sostenible.* Recuperado 11 de diciembre del 2023, de Naciones Unidas: https://agenda2030.mx/docs/doctos/Global_Indicator_Framework_after_2021_refinement_Spa.pdf

Pebesma, E., & Bivand, R. (2005). Classes and methods for spatial data in R. *R News*, 5(2), 9-13.

Pebesma, E., & Bivand, R. (2023). *Spatial Data Science: With Applications in R.* Chapman and Hall/CRC. doi: https://doi.org/10.1201/9780429459016

Pebesma, E. (2018). Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 10(1), 439-446. doi: https://doi.org/10.32614/RJ-2018-009

Pérez de la Cruz, G. (2023). Notas de clase. Seminario Estadística. Semestre 2023-2. Facultad de Ciencias, UNAM.

Presutti, M. (2004). La matriz de co-ocurrencia en la clasificación multiespectral: Tutorial para la enseñanza de medidas texturales en cursos de grado universitario. *4ª Jornada de Educação em Sensoriamento Remoto no Âmbito do Mercosul*, (págs. 1-9). São Leopoldo, Brasil. Obtenido de http://www3.inpe.br/unidades/cep/atividadescep/jornada/programa/t-9_trab_27.pdf

Rai, A. K., Mandal, N., Singh, A., & Singh, K. K. (2020). Landsat 8 OLI Satellite Image Classification using Convolutional Neural Network. Procedia Computer Science, 167, 987-993.https://doi.org/10.1016/j.procs.2020.03.398

Richards, J. A., & Jian, X. (2006). *Remote sensing digital image analysis: an introduction* (Vol. 4). Berlin: Springer.

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

Tarawneh, A. S., Hassanat, A. B. A., Tarawneh, A. S., & Almuhaimeed, A. (2022). Stop Oversampling for Class Imbalance Learning: A Review. *IEEE Access*, 10, 47643-47660. https://doi.org/10.1109/access.2022.3169512

Trends, E. (06 de febrero de 2023). *Trends.Earth - User Guide.* Recuperado el 11 de diciembre del 2023, de Trends.Earth: https://docs.trends.earth/es/1.0.10/background/understanding_indicators15.html

Vaughan, D., & Dancho, M. (2022). furrr: Apply Mapping Functions in Parallel using Futures (R package version 0.3.1) [Software]. Disponible en https://CRAN.R-project.org/package=furrr

Venables, W. N., & Ripley, B. D. (2002). MASS: *Modern Applied Statistics with S* (4ta ed) [Software]. New York: Springer.

Wright, M. N., & Ziegler, A. (2017). Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, 77(1), 1-17. [Software]. https://doi.org/10.18637/jss.v077.i01

Wickham, H., & Henry, L. (2023). purrr: Functional Programming Tools (R package version 1.0.2) [Software]. Disponible en https://CRAN.R-project.org/package=purrr

Youden, W. J. (1950). Index for rating diagnostic tests. *American Cancer Society*, 3(1), 32-35

Zvoleff, A. (2020). glcm: Calculate Textures from Grey-Level Co-Occurrence Matrices (GLCMs) (R package version 1.6.5) [Software]. Disponible en https://CRAN.R-project.org/package=glcm

Héctor Miguel Olivares García Dr. Gonzalo Pérez de la Cruz

Anexo A: Código de los ejemplos del capítulo 3

```r
# Librerías
library(dplyr)
library(tidyverse)
library(MASS)
library(glmnet)
library(e1071)
library(rpart.plot)
library(rpart)
```

```r
# semilla
set.seed(12323)
# Cargar conjunto de Datos
# Diabetes in Pima Indian Women

data("Pima.te")

# Ejemplo seleccion de variables
# Logística sin interacciones
AjusteNulo <- glm(formula = type ~ 1,
                  family =  binomial(link = "logit"),
                  data = Pima.te)

### Ajuste saturado.
AjusteSaturado <- glm(formula = type ~.,
                      family =  binomial(link = "logit"),
                      data = Pima.te)

# Selección de variables por pasos

probit_forward <- step(object = AjusteNulo,
                       scope = list(lower = AjusteNulo,
                                    upper = AjusteSaturado),
                       trace = TRUE,
                       direction = "forward" )
```

```r
# Ejemplo LASSO
XVar <- model.matrix(type ~., data = Pima.te)[,-1]
YRes <- Pima.te$type

lasso.tun <- cv.glmnet(x = XVar,
                       y = YRes,
                       nfolds = 10,
                       type.measure = "class",
```

```
                        gamma = 0,
                        relax = FALSE,
                        family = "binomial",
                        nlambda = 300)

plot(lasso.tun)
```

```
# Ejemplo SVM
# Visualizar paramétro cost

# Establecer conjunto de valores
n <- 700

# Genrar un dataframe con dos covariables con distribución uniforme entre 0 y 1
# La variable y se distingue por dos grupos: 1 y -1
df <- data.frame(x1 = runif(n),
                 x2 = runif(n))
df$y <- factor(ifelse(df$x2 - 1.4*df$x1 < 0, -1, 1),
    levels = c(-1, 1))

# Creacion de funcion para iterar
# hiperparámetro cost con kernel lineal

create_svm_plot <- function(trainset, cost_value) {
  svm_model <- svm(y ~ .,
                   data = trainset,
                   kernel = "linear",
                   gamma = 1,
                   cost = cost_value,
                   scale = FALSE)

  # calculo de margenes
  # w1*x1 +....wp*xp
  w <- t(svm_model$coefs) %*% svm_model$SV

  #pendiente
  slope_1 <- -w[1] / w[2]
  # calculo de b
  intercept_1 <- svm_model$rho / w[2]

  scatter_plot <- ggplot(data = trainset, aes(x = x1, y = x2, color = y)) +
    geom_point() +
    scale_color_manual(values = c("red", "blue"))

  #
  plot_decision <- scatter_plot + geom_abline(slope = slope_1, intercept = intercept_1)

  #
  plot_margins <- plot_decision +
    geom_abline(slope = slope_1, intercept = intercept_1 - 1 / w[2], linetype = "dashed") +
    geom_abline(slope = slope_1, intercept = intercept_1 + 1 / w[2], linetype = "dashed")

  #
```

```
  plot_with_title <- plot_margins + labs(title = paste("SVM con cost =", cost_value))

  #
  return(plot_with_title)
}

#Division de tamaño de muestra
tamano_m <- floor(0.7 * nrow(df))
train <- sample(seq_len(nrow(df)), size = tamano_m)

trainset <- df[train, ]
testset <- df[-train, ]

# Tuneo de Costo
cost_values <- c(1,40,45,50,100, 1000)

# Iteraccion de costos
for (cost_value in cost_values) {
  svm_plot <- create_svm_plot(trainset, cost_value)
  print(svm_plot)
}

# Tuneo
svm_tuned <- tune(svm, y ~ ., data = trainset, kernel = "linear",
                  ranges = list(cost = cost_range), scale = FALSE)

# Detalles del tuneo
print(svm_tuned)
```

```
# Particion de conjutno de datos
indice_entrenamiento <- sample(1:nrow(Pima.te), 0.7 * nrow(Pima.te))
conjunto_entrenamiento <- Pima.te[indice_entrenamiento, ]
conjunto_prueba <- Pima.te[-indice_entrenamiento, ]

# Entrenar tres árboles de decisión con diferentes covariables
arbol1 <- rpart(type ~ . , data = conjunto_entrenamiento, method = "class", control = rpart.co
arbol2 <- rpart(type ~  skin+ ped +age +bmi, data = conjunto_entrenamiento, method = "class",
arbol3 <- rpart(type ~ glu+ped+bmi+npreg, data = conjunto_entrenamiento, method = "class", con

# Bosque con tres árboles
par(mfrow = c(1, 3))  # Configurar el diseño de la ventana gráfica

# Árbol 1
rpart.plot(arbol1, extra = 1, under = TRUE, box.palette = "Blues", tweak = 1.2)

# Árbol 2
rpart.plot(arbol2, extra = 1, under = TRUE, box.palette = "Greens", tweak = 1.2)

# Árbol 3
rpart.plot(arbol3, extra = 1, under = TRUE, box.palette = "Reds", tweak = 1.2)
```

Anexo B: Funciones

```r
set.seed(1234)
# Función de de macheo y recorte para delimitar la zona de estudio
clip<-function(raster,shape) {
  a1_crop<-crop(raster,shape)
  step1<-rasterize(shape,a1_crop)
  a1_crop*step1}

# Funciones de estimacion predictiva y en paralelo

# Repeated Hould out oara datos originales
Repeat_HO <- function(df,Y,B,p1=0.8,balanceo=0){
  #i=1
  lista_Datos=list()
  for(i in 1:B){
    indice= caret::createDataPartition(df[,Y], p = p1, list = FALSE)
    Train <- df[indice, ]
    Test <- df[-indice, ]
    lista_Datos[[i]]=list(Train,Test)
    names(lista_Datos[[i]]) <- c("Train", "Test")
  }
  ListaD=lista_Datos %>%
    transpose()
  return(ListaD)
}

# Repeated Hould out para balabncear datos de entrenamiento
Repeat_HOBalanceo <- function(df,Y,B,p1=0.8){
  #i=1
  lista_Datos=list()
  for(i in 1:B){
    indice= caret::createDataPartition(df[,Y], p = 0.8, list = FALSE)
    Train1 <-  downSample(df[indice, ], df[indice, ]$Cl1984, list = T)
    Train <- as.data.frame( Train1$x)
    Test <- df[-indice, ]
    lista_Datos[[i]]=list(Train,Test)
    names(lista_Datos[[i]]) <- c("Train", "Test")
  }
  ListaD=lista_Datos %>%
    transpose()

  return(ListaD)
}
```

```r
ErroresClasificacion <- function(MC.Train, MC.Test){

  # metrica aparente

  ## Clase Agr / sensibilidad / recall
  Y1Train <- MC.Train[1,1]/sum(MC.Train[1,])

  ## Clase NoAgr  especificidad / specificity
  Y2Train <- MC.Train[2,2]/sum(MC.Train[2,])

  ## Tasa Clas correcta global/accuracy
  GlobalTrain <- sum(diag(MC.Train))/sum(MC.Train)

  # Errores de predicción

  ## Clase Agr / sensibilidad / recall
  Y1Test <- MC.Test[1,1]/sum(MC.Test[1,])

  ## Clase NoAgr  especificidad / specificity
  Y2Test <- MC.Test[2,2]/sum(MC.Test[2,])

  ## Tasa Clas correcta global/accuracy
  GlobalTest <- sum(diag(MC.Test))/sum(MC.Test)

  Errores <- data.frame(Y1Train, Y2Train, GlobalTrain, Y1Test, Y2Test, GlobalTest)

  return(Errores)

}



Result_Par <- function(Metodo, workers){

  opciones_furrr <- furrr_options(seed = TRUE)

  plan(strategy = multisession,
       workers = 5)

  Aux1 <- datos_analisis %>%
    future_pmap(.f = get(Metodo),
                .progress = TRUE,
                .options = opciones_furrr) %>%
    transpose()



  Aux1.1 <-  Aux1 %>%
    pmap(.f = ~ErroresClasificacion(.x,.y)) %>%
    plyr::ldply(data.frame) %>%
    mutate_if(is.numeric, ~.*100) %>%
    mutate_if(is.numeric, round, 3) %>%
```

```
    tibble::add_column(Metodo = Metodo,
                .before = "Y1Train")

  Aux2 <- Aux1.1 %>%
    reframe(Y1Train = mean(Y1Train),
              Y2Train = mean(Y2Train),
              GlobalTrain = mean(GlobalTrain),
              Y1Test = mean(Y1Test),
              Y2Test = mean(Y2Test),
              GlobalTest = mean(GlobalTest)) %>%
    mutate_if(is.numeric, round, 3) %>%
    tibble::add_column(Metodo = Metodo,
                .before = "Y1Train")

  return(list(Individual = Aux1.1,
              Global = Aux2,
              hiper=Aux1$hipe))

}
```

```
# Funciones de modelos de clasificación

Logit <- function(Train, Test) {

  logit <- glm(formula = Cl1984 ~.,
               family = binomial(link = "logit"),
               data = datos_analisis[["Train"]][[1]])


  # Punto de corte optimo
  pr<-pROC::coords(pROC::roc(logit$y, logit$fitted.values, quiet=T), 'best',
                   best.method='youden')$threshold

  hip=list(pto=pr,coe= as.list(coef(logit)))

  PredTrain <- (predict(object = logit,
                        newdata = datos_analisis[["Train"]][[1]],
                        type = "response") > pr) %>%
    ifelse(levels(logit$data[,1])[2],
           levels(logit$data[,1])[1])

  PredTest <- (predict(object = logit,
                       newdata = as.data.frame( datos_analisis[["Test"]][[1]] ),
                       type = "response") > pr) %>%
    ifelse(levels(logit$data[,1])[2],
           levels(logit$data[,1])[1])

  MC.Train <- table(Observados = datos_analisis[["Train"]][[1]]$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))
```

```
}

Probit <- function(Train, Test) {

  probitt <- glm(formula = Cl1984 ~.,
                 family = binomial(link = "probit"),
                 data = Train)


  # Punto de corte optimo
  pr<-pROC::coords(pROC::roc(probitt$y, probitt$fitted.values, quiet=T), 'best',
                   best.method='youden')$threshold

  hip=list(pto=pr,coe= as.list(coef(probitt)))
  PredTrain <- (predict(object = probitt,
                        newdata = Train,
                        type = "response") > pr) %>%
    ifelse(levels(probitt$data[,1])[2],
           levels(probitt$data[,1])[1])

  PredTest <- (predict(object = probitt,
                       newdata = Test,
                       type = "response") > pr) %>%
    ifelse(levels(probitt$data[,1])[2],
           levels(probitt$data[,1])[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

Logit_Lasso <- function(Train, Test) {

  XTrain <- model.matrix(Cl1984 ~., data = Train)[,-1]
  YTrain <- Train$Cl1984

  XTest <- model.matrix(Cl1984 ~., data = Test)[,-1]

  set.seed(1234)
  lasso.tun <- cv.glmnet(x = XTrain,
                         y = YTrain,
                         nfolds = 10,
                         type.measure = "class",
                         gamma = 0,
                         relax = FALSE,
                         family = "binomial",
                         nlambda = 300)
```

```r
  pred_prob <- predict(lasso.tun, newx = XTrain, s = "lambda.min", type = "response")

  roc_curve <- roc(YTrain, pred_prob)
  pr <- coords(roc_curve, "best", best.method = "youden")$threshold
  lambdaop <- lasso.tun$lambda.min

  hip=list(pto=pr,coe= lambdaop)


  PredTrain <- (predict(object = lasso.tun,
                        newx = XTrain,
                        type = "response",
                        s = "lambda.min") > pr) %>%
    ifelse(levels(YTrain)[2],
           levels(YTrain)[1])

  PredTest <- (predict(object = lasso.tun,
                       newx = XTest,
                       type = "response",
                       s = "lambda.min") > pr) %>%
    ifelse(levels(YTrain)[2],
           levels(YTrain)[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

Logit2_Step <- function(Train, Test) {

  ### Ajuste nulo.
  AjusteNulo <- glm(formula = Cl1984 ~ 1,
                    family =  binomial(link = "logit"),
                    data = Train)

  ### Ajuste saturado.
  AjusteSaturado <- glm(formula = Cl1984 ~.^2,
                        family =  binomial(link = "logit"),
                        data = Train)

  logit_forward <- step(object = AjusteNulo,
                        scope = list(lower = AjusteNulo,
                                     upper = AjusteSaturado),
                        trace = FALSE,
                        direction = "forward",
                        k = log(dim(Train)[1]))

  pr<-pROC::coords(pROC::roc(logit_forward$y, logit_forward$fitted.values, quiet=T), 'best',
                   best.method='youden')$threshold
```

```r
  #pr=0.5
  hip=list(pto=pr,coe= as.list(coef(logit_forward)))

  PredTrain <- (predict(object = logit_forward,
                        newdata = Train,
                        type = "response") > pr) %>%
    ifelse(levels(logit_forward$data[,1])[2],
           levels(logit_forward$data[,1])[1])

  PredTest <- (predict(object = logit_forward,
                       newdata = Test,
                       type = "response") > pr) %>%
    ifelse(levels(logit_forward$data[,1])[2],
           levels(logit_forward$data[,1])[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

Probit2_Step <- function(Train, Test) {

  ### Ajuste nulo.
  AjusteNulo <- glm(formula = Cl1984 ~ 1,
                    family =  binomial(link = "probit"),
                    data = Train)

  ### Ajuste saturado.
  AjusteSaturado <- glm(formula = Cl1984 ~.^2,
                        family =  binomial(link = "probit"),
                        data = Train)

  probit_forward <- step(object = AjusteNulo,
                         scope = list(lower = AjusteNulo,
                                      upper = AjusteSaturado),
                         trace = FALSE,
                         direction = "forward",
                         k = log(dim(Train)[1]))

  pr<-pROC::coords(pROC::roc(probit_forward$y, probit_forward$fitted.values, quiet=T), 'best',
                   best.method='youden')$threshold

  hip=list(pto=pr,coe= as.list(coef(probit_forward)))

  PredTrain <- (predict(object = probit_forward,
                        newdata = Train,
                        type = "response") > pr) %>%
    ifelse(levels(probit_forward$data[,1])[2],
           levels(probit_forward$data[,1])[1])
```

```
  PredTest <- (predict(object = probit_forward,
                       newdata = Test,
                       type = "response") > pr) %>%
    ifelse(levels(probit_forward$data[,1])[2],
           levels(probit_forward$data[,1])[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

Logit2_Lasso <- function(Train, Test) {

  XTrain <- model.matrix(Cl1984 ~.^2, data = Train)[,-1]
  YTrain <- Train$Cl1984

  XTest <- model.matrix(Cl1984 ~.^2, data = Test)[,-1]

  set.seed(1234)
  lasso.tun <- cv.glmnet(x = XTrain,
                         y = YTrain,
                         nfolds = 10,
                         type.measure = "class",
                         gamma = 0,
                         relax = FALSE,
                         family = "binomial",
                         nlambda = 300)

  pred_prob <- predict(lasso.tun, newx = XTrain, s = "lambda.min", type = "response")

  roc_curve <- roc(YTrain, pred_prob)
  pr <- coords(roc_curve, "best", best.method = "youden")$threshold
  lambdaop <- lasso.tun$lambda.min

  hip=list(pto=pr,coe= lambdaop)


  PredTrain <- (predict(object = lasso.tun,
                        newx = XTrain,
                        type = "response",
                        s = "lambda.min") > pr) %>%
    ifelse(levels(YTrain)[2],
           levels(YTrain)[1])

  PredTest <- (predict(object = lasso.tun,
                       newx = XTest,
                       type = "response",
                       s = "lambda.min") > pr) %>%
```

```
    ifelse(levels(YTrain)[2],
           levels(YTrain)[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

#punto de corte 0.5

LogitN <- function(Train, Test) {

  logit <- glm(formula = Cl1984 ~.,
               family = binomial(link = "logit"),
               data = Train)

  pr=0.5
  hip=list(pto=0.5,coe= as.list(coef(logit)))

  PredTrain <- (predict(object = logit,
                        newdata = Train,
                        type = "response") > pr) %>%
    ifelse(levels(logit$data[,1])[2],
           levels(logit$data[,1])[1])

  PredTest <- (predict(object = logit,
                       newdata = Test,
                       type = "response") > pr) %>%
    ifelse(levels(logit$data[,1])[2],
           levels(logit$data[,1])[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

ProbitN <- function(Train, Test) {

  probitt <- glm(formula = Cl1984 ~.,
                 family = binomial(link = "probit"),
                 data = Train)


  pr=0.5
  hip=list(pto=0.5,coe= as.list(coef(probitt)))
```

```
  PredTrain <- (predict(object = probitt,
                        newdata = Train,
                        type = "response") > pr) %>%
    ifelse(levels(probitt$data[,1])[2],
           levels(probitt$data[,1])[1])

  PredTest <- (predict(object = probitt,
                       newdata = Test,
                       type = "response") > pr) %>%
    ifelse(levels(probitt$data[,1])[2],
           levels(probitt$data[,1])[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

Logit_LassoN <- function(Train, Test) {

  XTrain <- model.matrix(Cl1984 ~., data = Train)[,-1]
  YTrain <- Train$Cl1984

  XTest <- model.matrix(Cl1984 ~., data = Test)[,-1]

  set.seed(1234)
  lasso.tun <- cv.glmnet(x = XTrain,
                         y = YTrain,
                         nfolds = 10,
                         type.measure = "class",
                         gamma = 0,
                         relax = FALSE,
                         family = "binomial",
                         nlambda = 300)

  pred_prob <- predict(lasso.tun, newx = XTrain, s = "lambda.min", type = "response")


  lambdaop <- lasso.tun$lambda.min
  pr=0.5
  hip=list(pto=0.5,coe= lambdaop)


  PredTrain <- (predict(object = lasso.tun,
                        newx = XTrain,
                        type = "response",
                        s = "lambda.min") > pr) %>%
    ifelse(levels(YTrain)[2],
           levels(YTrain)[1])
```

```
  PredTest <- (predict(object = lasso.tun,
                       newx = XTest,
                       type = "response",
                       s = "lambda.min") > pr) %>%
    ifelse(levels(YTrain)[2],
           levels(YTrain)[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

Logit2_StepN <- function(Train, Test) {

  ### Ajuste nulo.
  AjusteNulo <- glm(formula = Cl1984 ~ 1,
                    family =  binomial(link = "logit"),
                    data = Train)

  ### Ajuste saturado.
  AjusteSaturado <- glm(formula = Cl1984 ~.^2,
                        family =  binomial(link = "logit"),
                        data = Train)

  logit_forward <- step(object = AjusteNulo,
                        scope = list(lower = AjusteNulo,
                                     upper = AjusteSaturado),
                        trace = FALSE,
                        direction = "forward",
                        k = log(dim(Train)[1]))


  pr=0.5
  hip=list(pto=0.5,coe= as.list(coef(logit_forward)))

  PredTrain <- (predict(object = logit_forward,
                        newdata = Train,
                        type = "response") > pr) %>%
    ifelse(levels(logit_forward$data[,1])[2],
           levels(logit_forward$data[,1])[1])

  PredTest <- (predict(object = logit_forward,
                       newdata = Test,
                       type = "response") > pr) %>%
    ifelse(levels(logit_forward$data[,1])[2],
           levels(logit_forward$data[,1])[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)
```

```
  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

Probit2_StepN <- function(Train, Test) {

  ### Ajuste nulo.
  AjusteNulo <- glm(formula = Cl1984 ~ 1,
                    family =  binomial(link = "probit"),
                    data = Train)

  ### Ajuste saturado.
  AjusteSaturado <- glm(formula = Cl1984 ~.^2,
                        family =  binomial(link = "probit"),
                        data = Train)

  probit_forward <- step(object = AjusteNulo,
                         scope = list(lower = AjusteNulo,
                                      upper = AjusteSaturado),
                         trace = FALSE,
                         direction = "forward",
                         k = log(dim(Train)[1]))

  pr=0.5
  hip=list(pto=0.5,coe= as.list(coef(probit_forward)))

  PredTrain <- (predict(object = probit_forward,
                        newdata = Train,
                        type = "response") > pr) %>%
    ifelse(levels(probit_forward$data[,1])[2],
           levels(probit_forward$data[,1])[1])

  PredTest <- (predict(object = probit_forward,
                       newdata = Test,
                       type = "response") > pr) %>%
    ifelse(levels(probit_forward$data[,1])[2],
           levels(probit_forward$data[,1])[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

Logit2_LassoN <- function(Train, Test) {

  XTrain <- model.matrix(Cl1984 ~.^2, data = Train)[,-1]
```

```
  YTrain <- Train$Cl1984

  XTest <- model.matrix(Cl1984 ~.^2, data = Test)[,-1]

  set.seed(1234)
  lasso.tun <- cv.glmnet(x = XTrain,
                         y = YTrain,
                         nfolds = 10,
                         type.measure = "class",
                         gamma = 0,
                         relax = FALSE,
                         family = "binomial",
                         nlambda = 300)

  pred_prob <- predict(lasso.tun, newx = XTrain, s = "lambda.min", type = "response")

  pr=0.5
  lambdaop <- lasso.tun$lambda.min

  hip=list(pto=0.5,coe= lambdaop)


  PredTrain <- (predict(object = lasso.tun,
                        newx = XTrain,
                        type = "response",
                        s = "lambda.min") > pr) %>%
    ifelse(levels(YTrain)[2],
           levels(YTrain)[1])

  PredTest <- (predict(object = lasso.tun,
                       newx = XTest,
                       type = "response",
                       s = "lambda.min") > pr) %>%
    ifelse(levels(YTrain)[2],
           levels(YTrain)[1])

  MC.Train <- table(Observados = Train$Cl1984, Predic=PredTrain)
  MC.Test <- table( Observados = Test$Cl1984, Predic=PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}



Knn <- function(Train, Test) {
  set.seed(1234)
  TrainPre <- preProcess(Train, method = c("center", "scale"))
  TestPre <- preProcess(Test, method = c("center", "scale"))

  # Transformar los datos con los parámetros de estandarización
```

```
  datos_train <- predict(TrainPre, Train)
  datos_test <- predict(TestPre, Test)


  tuneknn=tune.knn(x=datos_train[,-1],
                   y= datos_train$Cl1984,
                   k=2:25,tunecontrol=tune.control(sampling = "cross"),cross=10)

  hiper= list(kbest=tuneknn$best.parameters[1])
  Maux1=knn(train=datos_train[,-1], test= datos_train[,-1],
            cl=datos_train$Cl1984, k = tuneknn$best.parameters[1])


  Maux2=knn(train=datos_train[,-1], test= datos_test[,-1],
            cl=datos_train$Cl1984, k = tuneknn$best.parameters[1])

  MC.Train <- table( Observados = Train$Cl1984, Predic=Maux1)
  MC.Test <- table(Observados = Test$Cl1984, Predic= Maux2)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hiper1=hiper))

}

RandomForest <- function(Train, Test) {

  malla_hyper <- expand.grid(
    mtry       = seq(3,7,1),
    node_size  = c(2,10,15),
    num.trees  = c(100,500))

  malla_hyper$OOBerr <- NA
  set.seed(123)
  folds <- cut(seq(1, nrow(Train)), breaks = 10, labels = FALSE)

  for(i in 1:nrow(malla_hyper)) {
    errors <- NULL

    # Perform k-fold cross validation
    for(j in 1:10) {
      # Segregate training and validation sets
      TrainCV <- Train[folds != j, ]
      ValidCV <- Train[folds == j, ]

      rf <- ranger(
        formula         = Cl1984 ~.,
        data            = TrainCV,
        num.trees       = malla_hyper$num.trees[i],
        mtry            = malla_hyper$mtry[i],
        min.node.size   = malla_hyper$node_size[i],
        importance = 'impurity')
```

```
      PredCV <- predict(object = rf, data = ValidCV)
      errors[j] <- mean(PredCV$predictions != ValidCV$Cl1984)
    }

    malla_hyper$OOBerr[i] <- mean(errors)
  }

  position <- which.min(malla_hyper$OOBerr)

  hip= list(num.trees = malla_hyper$num.trees[position],
            min.node.size = malla_hyper$node_size[position],
            mtry = malla_hyper$mtry[position],
            importance = 'impurity' )

  RF.Tune <- ranger(Cl1984 ~.,
                    data = Train,
                    num.trees = malla_hyper$num.trees[position],
                    min.node.size = malla_hyper$node_size[position],
                    mtry = malla_hyper$mtry[position],
                    importance = 'impurity',
                    probability = FALSE)

  PredTrain <- predict(object = RF.Tune,
                       data = Train)

  PredTest <- predict(object = RF.Tune,
                      data = Test)



  MC.Train <- table( Observados=Train$Cl1984, Predic=PredTrain$predictions)
  MC.Test <- table( Observados=Test$Cl1984 , Predic=PredTest$predictions)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}

SVM <- function(Train, Test) {

  set.seed(1234)
  TrainPre <- preProcess(Train, method = c("center", "scale"))
  TestPre <- preProcess(Test, method = c("center", "scale"))

  # Transformar los datos con los parámetros de estandarización
  datos_train <- predict(TrainPre, Train)
  datos_test <- predict(TestPre, Test)

  cad_svmtun <- e1071::tune("svm", Cl1984 ~ ., data=datos_train,kernel="radial",
                            ranges=list(cost=10^(-1:2),
                                        gamma=c(.01,.03,0.0667,.5,0.8)), cross = 10)
```

```
  hip= list(cost = cad_svmtun$best.parameters[[1]],
            gamma = cad_svmtun$best.parameters[[2]])

  svm <- e1071::svm(formula = Cl1984 ~.,kernel="radial",
                    cost=cad_svmtun$best.parameters[[1]],
                    gamma=cad_svmtun$best.parameters[[2]],
                    data=datos_train )

  PredTrain <- predict(object = svm, newdata = datos_train, type = "response")

  PredTest <- predict(object = svm, newdata = datos_test, type = "response")

  MC.Train <- table( Observados = datos_train$Cl1984, Predic=PredTrain)
  MC.Test <- table(Observados = datos_test$Cl1984, Predic= PredTest)

  return(list(MC.Train = MC.Train,
              MC.Test = MC.Test,
              hipe=hip))

}
```

Anexo C: Código de la aplicación del capítulo 4

```
rm(list = ls(all.names = TRUE))
gc()

# Librerías usadas

library(sp)
library(rgdal)
library(raster)
library(RStoolbox)
library(ggplot2)
library(gridExtra)
library(dplyr)
library(RColorBrewer)
library(glcm)
library(reshape)
library(grid)
library(rasterVis)
library(corrplot)
library(doParallel)
library(NeuralNetTools)
library(parallelly)
library(glmnet)
library(caret)
library(pROC)
library(rsample)
library(purrr)
library(furrr)
library(glmnet)
library(MASS)
library(ranger)
library(e1071)
library(ggradar)
library(beepr)

set.seed(1234)

#Lectura de datos desde el metado
metaDato<- RStoolbox::readMeta("\\LT51700721984174XXX08_MTL.txt")

# Combinar la informacion de las imagenes que se descargan en el documento
MT_6_84 <- stackMeta(metaDato)
```

```r
# # checa lo que trae el objeto
MT_6_84

# Verificando posición geográfica (CRS)
crs(MT_6_84)
#UTM zone 36N

## Se remueve la banda 6 debido a la resolución que tiene
TM5_6_84 <- dropLayer(MT_6_84, c(6))

# Imagen Satelital RGB (combinaciondes bandas)
plotRGB(TM5_6_84, r=5, g=4, b=3, stretch="lin")

## Distribución de los pixeles
B1_dn <- ggplot(TM5_6_84, aes(B1_dn)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B2_dn <- ggplot(TM5_6_84, aes(B2_dn)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B3_dn <- ggplot(TM5_6_84, aes(B3_dn)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B4_dn <- ggplot(TM5_6_84, aes(B4_dn)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B5_dn <- ggplot(TM5_6_84, aes(B5_dn)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B7_dn <- ggplot(TM5_6_84, aes(B7_dn)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

# Unir visualizaciones
grid.arrange(B1_dn, B2_dn, B3_dn, ncol = 1)
grid.arrange(B4_dn, B5_dn, B7_dn, ncol = 1)

# VAlores de GAIN y Offset
# Ranciancia=Gain*ND+Ooffset
TM5_radParameters <- metaDato$CALRAD
TM5_radParameters

## Corrección Radiometrica

# APREF

apref_6_84 <- radCor(TM5_6_84, metaData = metaDato, method = "apref")
```

```r
# Imagen Satelital aplicando correción APREF
plotRGB(apref_6_84, r=5, g=4, b=3, stretch="lin")

# Plot the histogram and density plots of the top-of-the-atmosphere reflectance/ apparent refl
B1_apref <- ggplot(apref_6_84, aes(B1_tre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B2_apref <- ggplot(apref_6_84, aes(B2_tre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B3_apref <- ggplot(apref_6_84, aes(B3_tre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B4_apref <- ggplot(apref_6_84, aes(B4_tre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

# Histogramas en algunas bandas aplicadas.
grid.arrange(B1_apref, B2_apref, B3_apref, B4_apref, ncol = 1)

#SDOS

# Estamación Haze
hazeDN <- estimateHaze(TM5_6_84, hazeBands = 1:4, darkProp = 0.01, plot = TRUE)

# Aplicación SDOS.
sdos_6_84 <- radCor(TM5_6_84, metaData = metaDato, method = "sdos",
                        hazeValues = hazeDN, hazeBands = 1:4)

# Imagen Satelital aplicando correción SDOS
plotRGB(sdos_6_84, r=5, g=4, b=3, stretch="lin")


B1_sdos <- ggplot(sdos_6_84, aes(B1_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B2_sdos <- ggplot(sdos_6_84, aes(B2_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B3_sdos <- ggplot(sdos_6_84, aes(B3_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B4_sdos <- ggplot(sdos_6_84, aes(B4_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

grid.arrange(B1_sdos, B2_sdos, B3_sdos, B4_sdos, ncol = 1)
```

```r
# Aplicación DOS
dos_6_84 <- radCor(TM5_6_84, metaData = metaDato, method = "dos")

# Imagen Satelital aplicando correción DOS
plotRGB(dos_6_84, r=5, g=4, b=3, stretch="lin")

B1_dos <- ggplot(dos_6_84, aes(B1_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B2_dos <- ggplot(dos_6_84, aes(B2_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B3_dos <- ggplot(dos_6_84, aes(B3_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B4_dos <- ggplot(dos_6_84, aes(B4_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

grid.arrange(B1_dos, B2_dos, B3_dos, B4_dos, ncol = 1)

# COSTZ

# Estamacion Haze
hazeDN <- estimateHaze(TM5_6_84, hazeBands = 1:4, darkProp = 0.01, plot = TRUE)

# Aplicación CostZ
costz_6_84 <- radCor(TM5_6_84, metaData = metaDato, method = "costz",
                         hazeValues = hazeDN, hazeBands = 1:4)

costz_6_84

nombres <-  names(costz_6_84)

plotRGB(costz_6_84, r=5, g=4, b=3, stretch="lin")

# Histogramas de la radiancia
B1_costz <- ggplot(costz_6_84, aes(B1_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B2_costz <- ggplot(costz_6_84, aes(B2_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

B3_costz <- ggplot(costz_6_84, aes(B3_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()
```

```
B4_costz <- ggplot(costz_6_84, aes(B4_sre)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density()

grid.arrange(B1_costz, B2_costz, B3_costz, B4_costz, ncol = 1)

# Comparacion de bandas
grid.arrange(B1_apref, B1_sdos, B1_costz)
grid.arrange(B2_apref, B2_sdos, B2_costz)
grid.arrange(B3_apref, B3_sdos, B3_costz)
grid.arrange(B4_apref, B4_sdos, B4_costz)

# REPROYECCION

# Georeferenciar la zona al hemisferio sur
reproj<-"+proj=utm+zone=36+south+datum=WGS84+units=m+no_defs+ellps=WGS84+towgs84=0,0,0"

# Georeferenciar la zona de las bandas corregidas al hemisferio sur
# Ver la resolucion del pixel en res, aplicadas en la reproyeccion
costz_6_84_proy <- projectRaster(costz_6_84, crs=reproj, res=30)

#Verificar georeferencia
t1 <- crs(costz_6_84_proy) # UTM zone 36S
t1

# Lectura de archivo shapefile, forma poligonal
# para delimitar zona
clip_boundary <- readOGR(setwd(), "Hre_Boundary")


# Recorte de la imagen satelital de acuerdo a la delimitación geografica
# Aplicar la función clip hecha en el apendice de funciones
costz_6_84R = clip(costz_6_84_proy, clip_boundary)

# Imagen Satelital RGB con la que se va a trabajar,
# una vez aplicada la corrección radiométrica
plotRGB(costz_6_84R, r=5, g=4, b=3, stretch="lin")

# Guardar información de la correción radiometrica
raster::writeRaster(costz_6_84R, "\\CostzPr_6_84.img",
                    datatype='FLT4S', overwrite = TRUE,
                    format = "GTiff")

####         Tratamiento de los indices espectrales

#Lista de archivos que contienen imagenes
ListaImg<-list.files(setwd(),pattern="img$",
                     full.names=TRUE)

# extracción de información
costz_6_84R <- stack(ListaImg)
```

```
#Checar Atributos
summary(costz_6_84R)

#Cálculo de indices espectrales
# Dependiendo de los indices que se quieran
# calcular se colocan las bandas o layer denomindas
IE_6_84 <- spectralIndices(costz_6_84R,
                           red = "Layer_3",
                           nir = "Layer_4",
                           indices = c("NDVI", "SAVI", "MSAVI"))

names(IE_6_84)
nombres=c(nombres,names(IE_6_84))

# Guardar información espectral
writeRaster(IE_6_84$NDVI,"\NDVI_22_06_84.img", datatype='FLT4S', overwrite = TRUE)
writeRaster(IE_6_84$SAVI,"\\SAVI_22_06_84.img", datatype='FLT4S', overwrite = TRUE)
writeRaster(IE_6_84$MSAVI,"\\MSAVI_22_06_84.img", datatype='FLT4S', overwrite = TRUE)

# INDICES de TEXTURA

# GLCM para la BANDA 4 con ventana de 3x3
# Se seleccionan las bandas que participan,
# los estadísticos deseados
# y la direccion
# por default esta la direccion a 45 grados
# si se requiere y teoricamente se debe de calcular la
# direccion y su opuesto, por lo que
# se selecciona list(c(1,0), c(-1,0))
glcm_6_84_4 <- glcm(raster(costz_6_84R, layer=4),
                    window = c(3, 3),
                    statistics = c("mean", "variance", "homogeneity", "entropy"),
                    shift=list(c(1,0), c(-1,0)))

paste0(names(glcm_6_84_4),"_4")
nombres=c(nombres, paste0(names(glcm_6_84_4),"_4") )

writeRaster(glcm_6_84_4$glcm_mean,"\\mean_22_6_84b4.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_6_84_4$glcm_variance,"\\variance_22_6_84b4.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_6_84_4$glcm_homogeneity,"\\homogeneity_22_6_84b4.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_6_84_4$glcm_entropy,"\\entropy_22_6_84b4.img",
            datatype='FLT4S', overwrite = TRUE)

# GLCM para la BANDA 5 con ventana de 3x3
glcm_6_84_5 <- glcm(raster(costz_6_84R, layer=5),
                    window = c(3, 3),
                    statistics = c("mean", "variance","homogeneity", "entropy"),
                  shift=list(c(1,0), c(-1,0)))

nombres=c(nombres, paste0(names(glcm_6_84_5),"_5") )
```

```
writeRaster(glcm_6_84_5$glcm_mean,"\\mean_22_6_84b5.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_6_84_5$glcm_variance,"\\variance_22_6_84b5.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_6_84_5$glcm_homogeneity,"\\homogeneity_22_6_84b5.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_6_84_5$glcm_entropy,"\\entropy_22_6_84b5.img",
            datatype='FLT4S', overwrite = TRUE)


# GLCM para la BANDA 7, pero dado que la banda 6
# originalmente no fue considerada,
# la banda 7 se considera como banda 6
# con ventana de 3x3
glcm_6_84_6 <- glcm(raster(costz_6_84R, layer=6),
                    window = c(3, 3),
                    statistics = c("mean", "variance", "homogeneity", "entropy"),
                    shift=list(c(1,0), c(-1,0)))

nombres=c(nombres, paste0(names(glcm_6_84_6),"_6") )
saveRDS(nombres, file = "\\nombres.rds")

writeRaster(glcm_6_84_6$glcm_mean,"\\mean_22_6_84b6.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_6_84_6$glcm_variance,"\\variance_22_6_84b6.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_6_84_6$glcm_homogeneity,"\\homogeneity_22_6_84b6.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_6_84_6$glcm_entropy,"\\entropy_22_6_84b6.img",
            datatype='FLT4S', overwrite = TRUE)

# UNIFICACI?N del DATASET

# Funcion para leer las variabels explicativas tratadas
Lista_VX <- list.files(file.path("\\Input_Tesis"),pattern="img$",
                       full.names=TRUE)
rasVars<- stack(Lista_VX)

#asignar nombres de las covariables covariables
x <- sort( readRDS(file = "\\nombres.rds") )
x2=c("B1_sre","B2_sre", "B3_sre", "B4_sre", "B5_sre","B7_sre",
     "glcm_entropy_4", "glcm_entropy_5", "glcm_entropy_7","glcm_homogeneity_4",
     "glcm_homogeneity_5", "glcm_homogeneity_7", "glcm_mean_4",
     "glcm_mean_5","glcm_mean_7","MSAVI", "NDVI", "SAVI",
     "glcm_variance_4", "glcm_variance_5", "glcm_variance_7")

names(rasVars) <- x2

# Extracción de etiquetas del tipo de suelo
# en formato shape
ta_data <- readOGR(file.path("\\conjunto"), "TA_1984")
```

```r
# Ta_data es una data frame de puntos espaciales
# Se extrae los valores de reflectancia por pixel de landSat5 y
# por pixel se etiquetan los puntos de entrenamiento
ta<-as.data.frame(raster::extract(rasVars, ta_data))

# Union de las variables explicativas y la variable Y
ta_data@data=data.frame(ta_data@data,
                        ta[match(rownames(ta_data@data),
                                 rownames(ta)),])

#Transformar la variable Y a formato factor
ta_data@data$Cl1984 <- factor(ta_data@data$Cl1984)

TM5_6_84_F <- plotRGB(rasVars, r=5, g=4, b=3, stretch="lin")

## Agrupar las etiquetas a un problema de clasificación binaria

ta_data@data$Cl1984 <-ifelse(ta_data@data$Cl1984=="Agr", "Agr", "NoAgr")

# transformar la variable Y como una variable categorica
ta_data@data$Cl1984<-as.factor(ta_data@data$Cl1984)
ta_data@data <- na.omit(ta_data@data)
summary(ta_data@data$Cl1984)

### Análisis descriptivo

# Primero Creamo un boxplot y se va a ranguear o dividir por pasos de preprocesameinto

datos_analisis <- tidyr::gather(ta_data@data, variable, value, -Cl1984)

# Boxplot
ggplot( subset(datos_analisis,
               variable %in% c("B1_sre","B2_sre", "B3_sre","B4_sre","B5_sre","B7_sre")),
        aes(x = variable, y = value, fill = Cl1984 )) +
  geom_boxplot() +
  scale_x_discrete(labels = abbreviate, name = "Bandas") +
  labs(title = "Correción radiométrica") +
  theme(plot.title = element_text(hjust = 0.5))

# Indices Espectrales

ggplot(subset(datos_analisis, variable %in% c("MSAVI", "NDVI", "SAVI") ), ç
       aes(x = variable, y = value, fill = Cl1984 )) +
  geom_boxplot() +
  scale_x_discrete(labels = abbreviate,
                   name = "Covariables de índices espectrales")+
  labs(title = "Índices espectrales") +
  theme(plot.title = element_text(hjust = 0.5))

# Indices de TEXTURA
ggplot( subset(datos_analisis,
               variable %in% c("glcm_entropy_4","glcm_entropy_5","glcm_entropy_7")),
        aes(x = variable, y = value, fill = Cl1984 )) +
```

```
  geom_boxplot() +
  scale_x_discrete(name = "Covariables de entropia")+
  labs(title = "Índices de textura entropia") +
  theme(plot.title = element_text(hjust = 0.5))

ggplot(subset(datos_analisis,
              variable %in% c("glcm_homogeneity_4","glcm_homogeneity_5", "glcm_homogeneity_7")
       aes(x = variable, y = value, fill = Cl1984 )) +
  geom_boxplot() +
  scale_x_discrete(name = "Covariables de homogeneidad")+
  labs(title = "Índices de textura homogeneidad") +
  theme(plot.title = element_text(hjust = 0.5))

ggplot( subset(datos_analisis,
               variable %in% c("glcm_mean_4","glcm_mean_5","glcm_mean_7")),
       aes(x = variable, y = value, fill = Cl1984 )) +
  geom_boxplot() +
  scale_x_discrete( name = "Covariables de media")+
  labs(title = "Índices de textura media") +
  theme(plot.title = element_text(hjust = 0.5))

ggplot( subset(datos_analisis,
               variable %in% c("glcm_variance_4","glcm_variance_5","glcm_variance_7")),
       aes(x = variable, y = value, fill = Cl1984 )) +
  geom_boxplot() +
  scale_x_discrete(name = "Covariables de varianza")+
  labs(title = "Índices de textura varianza") +
  theme(plot.title = element_text(hjust = 0.5))

# Correlaciones

# Calcular la matriz de correlación para cada grupo
cor_matrix_group1 <- cor(ta_data@data[ta_data@data$Cl1984 == "Agr", 2:22])
cor_matrix_group2 <- cor(ta_data@data[ta_data@data$Cl1984 == "NoAgr", 2:22])


# Convertir matrices a marcos de datos para ggplot2
cor_df_group1 <- as.data.frame(as.table(cor_matrix_group1))
cor_df_group2 <- as.data.frame(as.table(cor_matrix_group2))

par(mfrow=c(1,2))
# Crear gráficos de calor con ggplot2
ggplot(cor_df_group1, aes(Var1, Var2, fill = Freq)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                       name = "Correlación", midpoint = 0) +
  theme_minimal() +
  labs(title = "Gráfico de correlación: Clase - Agr") +
  theme(axis.title.x=element_blank(), axis.title.y=element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))

ggplot(cor_df_group2, aes(Var1, Var2, fill = Freq)) +
```

```
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                       name = "Correlación", midpoint = 0) +
  theme_minimal() +
  labs(title = "Gráfico de correlación: Clase - NoAgr") +
  theme(axis.title.x=element_blank(), axis.title.y=element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))

#  Construcción de las Reglas de clasificación

# Regresion Binaria

# Regresión logit simple
RL <- glm(formula = Cl1984 ~.,
          family = binomial(link = "logit"),
          data = ta_data@data)

# Regresión probit simple
RP <-  glm(formula = Cl1984 ~.,
           family = binomial(link = "probit"),
           data = ta_data@data)

# LASSO SIMPLE y seleccion de variables
XTrain <- model.matrix(Cl1984 ~., data = ta_data@data)[,-1]
YTrain <- ta_data@data$Cl1984

lassoS <- cv.glmnet(x = XTrain,
                        y = YTrain,
                        nfolds = 10,
                        type.measure = "class",
                        gamma = 0,
                        relax = FALSE,
                        family = "binomial",
                        nlambda = 300)

# Variables importantes


# Obtener los coeficientes de la regresión logística
coeficientes_RL <- coef(RL)
coeficientes_RL=as.data.frame(coeficientes_RL)
coeficientes_RL$index <- row.names(coeficientes_RL)
names(coeficientes_RL)[names(coeficientes_RL) == "index"] <- "Variable"
coeficientes_RL=coeficientes_RL[-1, ]
indices_ordenados1 <- order(abs(coeficientes_RL$coeficientes_RL), decreasing = TRUE)
coeficientes_RL <- coeficientes_RL[indices_ordenados1, ]

#Top 5 de variables importantes
TopRL= head(coeficientes_RL, 5) %>%
  mutate(Modelo="RL") %>%
  dplyr::select(Modelo,Variable,"valor"=coeficientes_RL)
```

```r
# Obtener los coeficientes de la regresión logística
coeficientes_RP <- coef(RP)
coeficientes_RP=as.data.frame(coeficientes_RP)
coeficientes_RP$index <- row.names(coeficientes_RP)
names(coeficientes_RP)[names(coeficientes_RP) == "index"] <- "Variable"
coeficientes_RP=coeficientes_RP[-1, ]
indices_ordenados2 <- order(abs(coeficientes_RP$coeficientes_RP), decreasing = TRUE)
coeficientesP <- coeficientes_RP[indices_ordenados2, ]

TopRP= head(coeficientes_RP, 5) %>%
  mutate(Modelo="RP") %>%
  dplyr::select(Modelo,Variable,"valor"=coeficientes_RP)


# Obtener los coeficientes de LASSO
coeficientes_Lasso <- coef(lassoS)
coeficientes_LassoToM <- as.matrix(coeficientes_Lasso)
coeficientes_Lasso2=as.data.frame(coeficientes_LassoToM)
coeficientes_Lasso2$index <- row.names(coeficientes_Lasso2)
names(coeficientes_Lasso2)[names(coeficientes_Lasso2) == "index"] <- "Variable"
coeficientes_Lasso2=coeficientes_Lasso2[-1, ]
indices_ordenados <- order(abs(coeficientes_Lasso2$s1), decreasing = TRUE)
coeficientes_Lasso2 <- coeficientes_Lasso2[indices_ordenados, ]

TopRLasso= head(coeficientes_Lasso2, 5) %>%
  mutate(Modelo="RLasso") %>%
  dplyr::select(Modelo,Variable,"valor"=s1)


TopSimple <- rbind(TopRLasso, TopRL, TopRP) %>%
  pivot_wider( names_from = Variable, values_from = valor) %>%
  mutate_all(~ replace(.,is.na(.),0))

limite_min <- min(apply(TopSimple[, 2:ncol(TopSimple)], 2, min))
limite_max <- max(apply(TopSimple[, 2:ncol(TopSimple)], 2, max))

lcols <- c("#EEA236", "#5CB85C", "#46B8DA")
ggradar::ggradar(TopSimple,
                 grid.min =limite_min,
                 grid.max = limite_max,
                 values.radar = c(as.character(round(limite_min,0)),
                                  "0",
                                  as.character(round(limite_max,0)) ),
                 legend.title = "Modelo",
                 legend.position = "bottom",
                 group.colours = lcols,
                 legend.text.size = 10,
                 fill.alpha = 0.3)


# Regresion Binaria con Interacciones y seleccion de variables
```

```r
# Regresion logit

AjusteNulo <- glm(formula = Cl1984 ~ 1,
                  family =  binomial(link = "logit"),
                  data = ta_data@data)

### Ajuste saturado.
AjusteSaturado <- glm(formula = Cl1984 ~.^2,
                      family =  binomial(link = "logit"),
                      data = ta_data@data)

RL2 <- step(object = AjusteNulo,
            scope = list(lower = AjusteNulo,
                         upper = AjusteSaturado),
            trace = FALSE,
            direction = "forward",
            k = log(dim(ta_data@data)[1]))

# Regresion probit

AjusteNulo <- glm(formula = Cl1984 ~ 1,
                  family =  binomial(link = "probit"),
                  data = ta_data@data)

### Ajuste saturado
AjusteSaturado <- glm(formula = Cl1984 ~.^2,
                      family =  binomial(link = "probit"),
                      data = ta_data@data)

RP2 <- step(object = AjusteNulo,
            scope = list(lower = AjusteNulo,
                         upper = AjusteSaturado),
            trace = FALSE,
            direction = "forward",
            k = log(dim(ta_data@data)[1]))

# LASSO con interacciones y seleccion de variables.

XTrain <- model.matrix(Cl1984 ~.^2, data = ta_data@data)[,-1]
YTrain <- ta_data@data$Cl1984

lasso2 <- cv.glmnet(x = XTrain,
                    y = YTrain,
                    nfolds = 10,
                    type.measure = "class",
                    gamma = 0,
                    relax = FALSE,
                    family = "binomial",
                    nlambda = 300)
```

```r
# Obtener los coeficientes de la regresión logística
coeficientes_RL <- coef(RL2)
coeficientes_RL=as.data.frame(coeficientes_RL)
coeficientes_RL$index <- row.names(coeficientes_RL)
names(coeficientes_RL)[names(coeficientes_RL) == "index"] <- "Variable"
coeficientes_RL=coeficientes_RL[-1, ]
indices_ordenados1 <- order(abs(coeficientes_RL$coeficientes_RL), decreasing = TRUE)
coeficientes_RL <- coeficientes_RL[indices_ordenados1, ]

#Top 5 de variables importantes
TopRL= head(coeficientes_RL, 5) %>%
  mutate(Modelo="RL") %>%
  dplyr::select(Modelo,Variable,"valor"=coeficientes_RL)


# Obtener los coeficientes de la regresión logística
coeficientes_RP <- coef(RP2)
coeficientes_RP=as.data.frame(coeficientes_RP)
coeficientes_RP$index <- row.names(coeficientes_RP)
names(coeficientes_RP)[names(coeficientes_RP) == "index"] <- "Variable"
coeficientes_RP=coeficientes_RP[-1, ]
indices_ordenados2 <- order(abs(coeficientes_RP$coeficientes_RP), decreasing = TRUE)
coeficientesP <- coeficientes_RP[indices_ordenados2, ]

TopRP= head(coeficientes_RP, 5) %>%
  mutate(Modelo="RP") %>%
  dplyr::select(Modelo,Variable,"valor"=coeficientes_RP)


# Obtener los coeficientes de LASSO
coeficientes_Lasso <- coef(lasso2)
coeficientes_LassoToM <- as.matrix(coeficientes_Lasso)
coeficientes_Lasso2=as.data.frame(coeficientes_LassoToM)
coeficientes_Lasso2$index <- row.names(coeficientes_Lasso2)
names(coeficientes_Lasso2)[names(coeficientes_Lasso2) == "index"] <- "Variable"
coeficientes_Lasso2=coeficientes_Lasso2[-1, ]
indices_ordenados <- order(abs(coeficientes_Lasso2$s1), decreasing = TRUE)
coeficientes_Lasso2 <- coeficientes_Lasso2[indices_ordenados, ]

TopRLasso= head(coeficientes_Lasso2, 5) %>%
  mutate(Modelo="RLasso") %>%
  dplyr::select(Modelo,Variable,"valor"=s1)


TopSimple <- rbind(TopRLasso, TopRL, TopRP) #%>%
  pivot_wider( names_from = Variable, values_from = valor) %>%
  mutate_all(~ replace(.,is.na(.),0))

limite_min <- min(apply(TopSimple[, 2:ncol(TopSimple)], 2, min))
limite_max <- max(apply(TopSimple[, 2:ncol(TopSimple)], 2, max))

lcols <- c("#EEA236", "#5CB85C", "#46B8DA")
ggradar::ggradar(TopSimple,
```

```
                grid.min =-18.009250620,
                grid.max = 94.257316474,
                values.radar = c(as.character(round(limite_min,0)),
                                 "0",
                                 as.character(round(limite_max,0)) ),
                legend.title = "Modelo",
                legend.position = "bottom",
                group.colours = lcols,
                legend.text.size = 10,
                axis.label.size = 2.5,
                grid.label.size = 4,
                fill.alpha = 0.3)

# KNN

# Estandarizar los datos
preproc <- preProcess(ta_data@data, method = c("center", "scale"))

# Transformar los datos con los parámetros de estandarización
datos_trans <- predict(preproc, ta_data@data)

#tuneo con k-CV k=10
tuneknn=tune.knn(x=datos_trans[,-1],
                 y= datos_trans$Cl1984,
                 k=2:25,
                 tunecontrol=tune.control(sampling = "cross"),
                 cross=10)

# regla de clasificación
KnnFit=knn(train=datos_trans[,-1],
           test= datos_trans[,-1],
           cl=datos_trans$Cl1984,
           k = tuneknn$best.parameters[1])

ggplot(data = tuneknn$performances, aes(x = k, y = error)) +
  geom_line() +
  geom_point(color = "blue") +
  labs( x = "K", y = "Error") +
  theme_minimal()+
  scale_x_continuous(breaks = seq(0,26, by = 3))

# Bosques Aleatorios

# Tuneo
malla_hyper <- expand.grid(
  mtry       = seq(3,7,1),
  node_size  = c(2,10,15),
  num.trees  = c(100,500))

malla_hyper$error <- NA

# Aplicar k-CV
folds <- cut(seq(1, nrow(ta_data@data)), breaks = 10, labels = FALSE)
```

```r
# Aplicar tuneo
for(i in 1:nrow(malla_hyper)) {
  errors <- NULL

  # K cross validation
  for(j in 1:10) {
    # Segregate training and validation sets
    TrainCV <- ta_data@data[folds != j, ]
    ValidCV <- ta_data@data[folds == j, ]

    rf <- ranger(
      formula         = Cl1984 ~.,
      data            = TrainCV,
      num.trees       = malla_hyper$num.trees[i],
      mtry            = malla_hyper$mtry[i],
      min.node.size   = malla_hyper$node_size[i],
      importance = 'impurity')

    PredCV <- predict(object = rf, data = ValidCV)
    errors[j] <- mean(PredCV$predictions != ValidCV$Cl1984)
  }

  malla_hyper$OOBerr[i] <- mean(errors)
}

position <- which.min(malla_hyper$OOBerr)

BosquesRegla <- ranger(Cl1984 ~.,
                  data = ta_data@data,
                  num.trees = malla_hyper$num.trees[position],
                  min.node.size = malla_hyper$node_size[position],
                  mtry = malla_hyper$mtry[position],
                  importance = 'impurity',
                  probability = FALSE)


saveRDS(BosquesRegla, file ="\\BosquesA.rds")

# importancia de las variables
importancia_variables <- importance(BosquesRegla)

# Convertir objeto a dataframe
importancia_df <- data.frame(variable = names(importancia_variables),
                             importance = importancia_variables)

# Crea el gráfico de barras con ggplot2
ggplot(importancia_df,
       aes(x = reorder(variable, importance), y = importance)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +
  labs(title = "Importancia de las covariables en el bosque aleatorio",
       x = "Variable", y = "Importancia") +
```

```
  theme_minimal() +
  theme(axis.text.y = element_text(size = 8))

# SVM
# estandarizar datos

cad_svmtun <- e1071::tune("svm", Cl1984 ~ .,
                          data=datos_trans, kernel="radial",
                          ranges=list(cost=10^(-1:2),
                                      gamma=c(.01,.03,0.0667,.5,0.8)),
                          cross = 10)

svm <- e1071::svm(formula = Cl1984 ~.,kernel="radial",
                  cost=cad_svmtun$best.parameters[[1]],
                  gamma=cad_svmtun$best.parameters[[2]],
                  data=datos_trans )

ggplot(data = cad_svmtun$performances,
       aes(x = cost, y = error, color = as.factor(gamma)))+
  geom_line() +
  geom_point() +
  labs(title = "Error de clasificación vs hiperparámetros cost y gamma",
       color = "gamma") +
  theme_bw() +
  theme(legend.position = "bottom")

# Aplicacion poder predictivo

# Preparar datos de entradaç
# para aplicar Repeated Hold Out
datos_analisis=Repeat_HO(df=ta_data@data, Y="Cl1984", B=100,p1=0.80)

# Regresion binaria

start_time <- proc.time()
logitEP=Result_Par(Metodo = "Logit", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
probitEP=Result_Par(Metodo = "Probit", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
LassoEp=Result_Par(Metodo = "Logit_Lasso", workers = 8)
t=proc.time() - start_time
beep(2)

# Binario con interacciones

start_time <- proc.time()
logit2EP=Result_Par(Metodo = "Logit2_Step", workers = 8)
```

```r
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
probit2EP=Result_Par(Metodo = "Probit2_Step", workers = 8)
t=proc.time() - start_time
beep(2)

# Lasso

start_time <- proc.time()
lasso2EP=Result_Par(Metodo = "Logit2_LassoN", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
RF=Result_Par(Metodo = "RandomForest", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
Knnfit=Result_Par(Metodo = "Knn", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
SvM=Result_Par(Metodo = "SVM", workers = 8)
t=proc.time() - start_time
beep(2)
```

```r
# Aplicación del podeer predictivo con datos balanceados
datos_analisis=Repeat_HOBalanceo(df=ta_data@data, Y="Cl1984", B=100,p1=0.8)

# PAradatos Balanceados
start_time <- proc.time()
logitEP=Result_Par(Metodo = "LogitN", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
probitEP=Result_Par(Metodo = "ProbitN", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
LassoEp=Result_Par(Metodo = "Logit_LassoN", workers = 8)
t=proc.time() - start_time
beep(2)

# Binario con interacciones

start_time <- proc.time()
logit2EP=Result_Par(Metodo = "Logit2_StepN", workers = 8)
```

```
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
probit2EP=Result_Par(Metodo = "Probit2_StepN", workers = 8)
t=proc.time() - start_time
beep(2)


# Lasso

start_time <- proc.time()
lasso2EP=Result_Par(Metodo = "Logit2_LassoN", workers = 8)
t=proc.time() - start_time
beep(2)

# Regresion binaria punto de corte Youden

start_time <- proc.time()
logitEP=Result_Par(Metodo = "Logit", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
probitEP=Result_Par(Metodo = "Probit", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
LassoEp=Result_Par(Metodo = "Logit_Lasso", workers = 8)
t=proc.time() - start_time
beep(2)

# Binario con interacciones

start_time <- proc.time()
logit2EP=Result_Par(Metodo = "Logit2_Step", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
probit2EP=Result_Par(Metodo = "Probit2_Step", workers = 8)
t=proc.time() - start_time
beep(2)

# Lasso

start_time <- proc.time()
lasso2EP=Result_Par(Metodo = "Logit2_Lasso", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
```

```
RF=Result_Par(Metodo = "RandomForest", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
Knnfit=Result_Par(Metodo = "Knn", workers = 8)
t=proc.time() - start_time
beep(2)

start_time <- proc.time()
SvM=Result_Par(Metodo = "SVM", workers = 8)
t=proc.time() - start_time
beep(2)
```

```
# Aplicación de la mejor regla en datos
# de diciembre

metaData<- readMeta("\\LT51700721984366XXX02_MTL.txt")

prueba <- stackMeta(metaData)

print( prueba)
TM5_31_12_84 <- dropLayer(prueba, c(6))
#VEr imagen de diciembre
plotRGB(TM5_31_12_84, r=5, g=4, b=3, stretch="lin")

hazeDN <- estimateHaze(TM5_31_12_84,
                       hazeBands = 1:4,
                       darkProp = 0.01,
                       plot = FALSE)

costz_31_12_84 <- radCor(TM5_31_12_84,
                         metaData = metaData,
                         method = "costz",
                         hazeValues = hazeDN,
                         hazeBands = 1:4)

reproj<-"+proj=utm+zone=36+south+datum=WGS84+units=m+no_defs+ellps=WGS84+towgs84=0,0,0"

costz_31_12_84pr <- projectRaster(costz_31_12_84, crs=reproj, res=30)

t1 <- crs(costz_31_12_84pr)

#verificar zona
t1

clip_boundary <- readOGR(setwd(), "Hre_Boundary")

costz31_12_84 = clip(costz_31_12_84pr, clip_boundary)

plotRGB(costz31_12_84, r=5, g=4, b=3, stretch="lin")

writeRaster(costz31_12_84,"\\CostzPr_31_12_8.img",
```

```
            datatype='FLT4S', overwrite = TRUE)

plotRGB(costz31_12_84, r=5, g=4, b=3, stretch="lin")

VI31_12_84 <- spectralIndices(costz31_12_84,
                              red = "layer.3",
                              nir = "layer.4",
                              indices = c("NDVI", "SAVI", "MSAVI"))


writeRaster(VI31_12_84$NDVI,"\\NDVI_31_12_8.img", datatype='FLT4S', overwrite = TRUE)
writeRaster(VI31_12_84$SAVI,"\\SAVI_31_12_8.img", datatype='FLT4S', overwrite = TRUE)
writeRaster(VI31_12_84$MSAVI,"\\MSAVI_31_12_8.img", datatype='FLT4S', overwrite = TRUE)


rlist<-list.files(setwd(),pattern="img$",
                  full.names=TRUE)

costz31_12_84 <- stack(rlist)


glcm_31_12_84.4 <- glcm(raster(costz31_12_84, layer=4),
                        window = c(3, 3),
                        statistics = c("mean", "variance", "homogeneity", "entropy"),
                        shift=list(c(1,0), c(-1,0)))

writeRaster(glcm_31_12_84.4$glcm_mean,"\\mean_31_12_8b4.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_31_12_84.4$glcm_variance,"\\variance_31_12_8b4.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_31_12_84.4$glcm_homogeneity,"\\homogeneity_31_12_8b4.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_31_12_84.4$glcm_entropy,"\\entropy_31_12_8b4.img",
            datatype='FLT4S', overwrite = TRUE)


glcm_31_12_84.5 <- glcm(raster(costz31_12_84, layer=5),
                        window = c(3, 3),
                        statistics = c("mean", "variance", "homogeneity", "entropy"),
                        shift=list(c(1,0), c(-1,0)))

writeRaster(glcm_31_12_84.5$glcm_mean,"\\mean_31_12_8b5.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_31_12_84.5$glcm_variance,"\\variance_31_12_8b5.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_31_12_84.5$glcm_homogeneity,"\\homogeneity_31_12_8b5.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_31_12_84.5$glcm_entropy,"\\entropy_31_12_8b5.img",
            datatype='FLT4S', overwrite = TRUE)


glcm_31_12_84.6 <- glcm(raster(costz31_12_84, layer=6),
                        window = c(3, 3),
```

```
                        statistics = c("mean", "variance","homogeneity", "entropy"),
                        shift=list(c(1,0), c(-1,0)))

writeRaster(glcm_31_12_84.6$glcm_mean,"\\mean_31_12_8b6.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_31_12_84.6$glcm_variance,"\\variance_31_12_8b6.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_31_12_84.6$glcm_homogeneity,"\\homogeneity_31_12_8b6.img",
            datatype='FLT4S', overwrite = TRUE)
writeRaster(glcm_31_12_84.6$glcm_entropy,"\\entropy_31_12_8b6.img",
            datatype='FLT4S', overwrite = TRUE)

#

rasList<-list.files(setwd(),pattern="img$",
                    full.names=TRUE)
rasVars <- stack(rasList)

# Establecer nombres
x <- sort( readRDS(file = "\\nombres.rds") )

x2=c("B1_sre","B2_sre", "B3_sre", "B4_sre", "B5_sre","B7_sre",
     "glcm_entropy_4", "glcm_entropy_5", "glcm_entropy_7","glcm_homogeneity_4",
     "glcm_homogeneity_5", "glcm_homogeneity_7", "glcm_mean_4",
     "glcm_mean_5","glcm_mean_7","MSAVI", "NDVI", "SAVI",
     "glcm_variance_4", "glcm_variance_5", "glcm_variance_7")

names(rasVars) <- x2

TM5_31_12_84 <- plot(rasVars)

TM5_31_12_84 <- plotRGB(rasVars, r=5, g=4, b=3, stretch="lin")

ta_data <- readOGR(setwd("\\Imagenes31_12_84"), "TA_1984")

ta<-as.data.frame(raster::extract(rasVars, ta_data))

ta_data@data=data.frame(ta_data@data,
                        ta[match(rownames(ta_data@data),
                                 rownames(ta)),])


ta_data@data$Cl1984 <-ifelse(ta_data@data$Cl1984=="Agr", "Agr", "NoAgr")
# transformar la variable Y como una variable categorica
ta_data@data$Cl1984<-as.factor(ta_data@data$Cl1984)
ta_data@data <- na.omit(ta_data@data)

# Leer regla de clasificación que tuvo mejor desempeño
# Leer la regla
RF=readRDS("\\BosquesA.rds")

BApredict=predict(RF, data=ta_data@data)
```

```
taux= table( Observaciones=ta_data@data$Cl1984,
             Predic=BApredict$predictions)
## Clase Agr
Y1Test <- taux[1,1]/sum(taux[1,])

## Clase NoAgr
Y2Test <- taux[2,2]/sum(taux[2,])

## excatitud del modelo
GlobalTest <- sum(diag(taux))/sum(taux)
```