



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

EL USO DE MODELOS GRÁFICOS
PROBABILÍSTICOS EN PREDICCIÓN. UNA
APLICACIÓN CON DATOS SOBRE LA
SECUENCIACIÓN GENÓMICA PARA
DETECTOR CÁNCER DE CEREBRO

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

ACTUARIO

P R E S E N T A :

LEONARDO DANIEL DE LA CRUZ CUAXILOA

TUTOR

DR. GONZALO PÉREZ DE LA CRUZ

CIUDAD UNIVERSITARIA, CDMX, 2025



AGRADECIMIENTOS

Investigación realizada gracias al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM IA101224. Agradezco a la DGAPA-UNAM la beca recibida.

RESUMEN

Los modelos gráficos probabilísticos han demostrado tener un potencial uso en problemas de clasificación supervisada, especialmente en contextos de alta dimensionalidad y tamaños de muestra limitados. En este trabajo, se explora el uso de modelos gráficos gaussianos no dirigidos (UGGM) para la clasificación de datos de expresión genética, utilizando el algoritmo *Graphical Lasso* para estimar la estructura de la gráfica y la matriz de concentración; así como para optimizar el poder predictivo del modelo.

Para evaluar el desempeño de la regla basada en UGGM, se realizó un ejercicio comparativo con otros métodos de clasificación supervisada usando tanto datos simulados como los de expresión genética, incluyendo: la regresión logística multinomial, el análisis de discriminante lineal y cuadrático, las máquinas de soporte vectorial y los bosques aleatorios.

En particular, se analizaron las expresiones genéticas de las principales enzimas de la vía de las kinureninas, una ruta metabólica clave en la degradación del triptófano, cuya actividad ha sido asociada con el desarrollo y progresión del cáncer de cerebro. Se utilizaron datos de expresión genética de estas enzimas obtenidos de los proyectos TCGA, GTEx y UCSC Xena, con el objetivo de encontrar reglas para clasificar entre individuos sanos, pacientes con gliomas de bajo grado y pacientes con gliomas altamente agresivos.

Esto es relevante, ya que actualmente existen diversos métodos de diagnóstico para detectar el cáncer de cerebro, sin embargo, algunos de estos métodos identifican la enfermedad en etapas avanzadas o requieren procedimientos invasivos que pueden implicar riesgos adicionales para un paciente. Por ello, es importante desarrollar métodos de detección que cumplan con las siguientes características:

- Oportunos: Capaces de detectar el cáncer en etapas iniciales, lo que aumentaría la efectividad de los tratamientos posteriores.
- No invasivos: Que no requieran procedimientos quirúrgicos invasivos, evitando así riesgos adicionales para el paciente.

Dado lo anterior, es interesante analizar el potencial uso de la vía de las kinureninas en la definición de reglas de diagnóstico.

Los resultados muestran que el modelo basado en UGGM supera en ciertas métricas a otros modelos, obteniendo el mejor rendimiento en la clasificación de individuos sanos (90.8 %) y gliomas altamente agresivos (82.6 %). Sin embargo, se identifican áreas de mejora a partir de lo realizado en este trabajo, como la calibración del hiperparámetro λ del método *Graphical Lasso* que podría realizarse para cada clase de forma independiente y la exploración de criterios alternativos para la selección de la estructura del modelo.

Se concluye que la integración de los modelos gráficos probabilísticos en problemas de clasificación es una alternativa viable y con potencial. Además, los hallazgos sugieren que la expresión de las enzimas de la vía de las kinureninas podría utilizarse como un potencial biomarcador

para la predicción temprana del cáncer de cerebro, lo que abre nuevas oportunidades para estudios clínicos y el desarrollo de herramientas diagnósticas menos invasivas.

ÍNDICE GENERAL

1. Introducción	6
1.1. Motivación	6
1.2. Objetivos	8
1.2.1. Generales	8
1.2.2. Específicos	8
1.3. Estructura de la tesis	9
2. Metodología	10
2.1. Modelos gráficos probabilísticos	11
2.1.1. Definiciones	11
2.1.2. Distribución normal o gaussiana multivariada	12
2.1.3. Modelos gráficos gaussianos no dirigidos	13
2.1.4. Datos simulados	14
2.1.5. Problemas de estimación en los modelos gráficos gaussianos	18
2.2. Aprendizaje supervisado con el objetivo de predicción	23
2.2.1. Métricas de desempeño	27
2.2.2. Repeated u-v Fold Cross Validation	29
2.3. Algoritmos de clasificación	30
2.3.1. Análisis de discriminante y el uso de los modelos gráficos gaussianos no dirigidos	31
2.3.2. Otros métodos	39
2.4. Comparación empírica de los modelos usando datos simulados	48
3. Caso práctico	50
3.1. Datos	50
3.1.1. La vía de las kinureninas	50
3.1.2. Relación con el cáncer de cerebro	51
3.1.3. La vía de las kinureninas desde un enfoque de clasificación supervisada	51
3.2. Análisis exploratorio	53
3.2.1. Pruebas no paramétricas	53
3.2.2. Diagramas de dispersión, correlaciones de Pearson y correlaciones parciales de Pearson	55
3.2.3. Componentes principales	58
3.3. Estimación del poder predictivo	61
3.4. Análisis de los resultados	69
4. Conclusiones	72
4.1. Modelos gráficos probabilísticos	72
4.2. Vía de las kinureninas	73
Bibliografía	75
A. Anexo 1	79
B. Anexo 2	81

I

INTRODUCCIÓN

1.1 MOTIVACIÓN

El cáncer de cerebro es uno de los tipos de cáncer más mortales a nivel mundial [Bray et al., 2024]. Esta elevada mortalidad se debe a la dificultad para tratarlo y a su naturaleza altamente invasiva. A pesar de los avances en tratamientos como la cirugía y la quimioterapia, la tasa de supervivencia sigue siendo baja para muchos pacientes, especialmente en casos de tumores agresivos como el glioblastoma, con una supervivencia media de 14 meses [Mohammed et al., 2022].

Actualmente, no existe un tratamiento curativo para varios tipos de cáncer de cerebro, y los tratamientos disponibles suelen enfocarse más en prolongar la vida y aliviar los síntomas que en eliminar el tumor [Batash et al., 2017]. Ante esta situación, los esfuerzos en investigación buscan identificar posibles puntos de intervención terapéutica que permitan desarrollar tratamientos específicos más efectivos [Touat et al., 2017]. Asimismo, la creación de métodos de diagnóstico temprano más precisos es de gran importancia, ya que detectar la enfermedad en sus etapas iniciales podría mejorar significativamente la efectividad de las intervenciones [Gao and Jiang, 2013].

Un método de diagnóstico efectivo para detectar el cáncer de cerebro debería ser capaz de distinguir con precisión entre personas enfermas y no enfermas, permitiendo así intervenciones tempranas y específicas. Sin embargo, la mayoría de los métodos de diagnóstico actuales presentan diversas limitaciones: algunos dependen de tecnologías de imagen que solo detectan tumores cuando estos han alcanzado un tamaño considerable o han invadido otras áreas, reduciendo la efectividad de los tratamientos. Otros métodos, como las biopsias, requieren procedimientos quirúrgicos invasivos, que pueden implicar riesgos adicionales para los pacientes. Por esta razón, la investigación hacia el desarrollo de métodos menos invasivos que permitan identificar etapas tempranas del cáncer de cerebro sin necesidad de intervenciones quirúrgicas es de gran importancia.

Una alternativa para encontrar posibles variables o biomarcadores que tengan un potencial uso para el diagnóstico es el análisis de la información de las expresiones genéticas. Una de las bases más utilizadas, y la que se considera en este trabajo, está disponible bajo el nombre *A combined cohort of TCGA, TARGET and GTEx samples*. Esta base de datos es el resultado de un preprocesamiento que permite la integración de datos de los proyectos TCGA, TARGET y GTEx, como se describe en Vivian et al. (2017); y está disponible en el proyecto UCSC Xena [Goldman et al., 2020], que es una plataforma desarrollada por la Universidad de California, Santa Cruz, que integra y facilita el acceso a grandes conjuntos de

datos genómicos, particularmente aquellos relacionados con la investigación del cáncer. Esta plataforma permite a los investigadores analizar de manera conjunta datos de secuenciación genómica provenientes de diversos proyectos coordinados por las agencias de salud de Estados Unidos, como TCGA (*The Cancer Genome Atlas*) [Tomczak et al., 2015], GTEx (*Genotype-Tissue Expression*) [Carithers and Moore, 2015] y otros estudios internacionales, ofreciendo herramientas para la visualización y análisis de datos.

En este trabajo, se utilizaron datos obtenidos de tejido cerebral de 972 personas, 283 sanas y 689 con algún cáncer de cerebro, para identificar si las variables asociadas con las expresiones genéticas de las enzimas de la vía de las kinureninas (11 variables cuantitativas) podrían servir para construir herramientas para un diagnóstico efectivo del cáncer de cerebro. Se da un mayor énfasis al estudio de estas expresiones de la vía de las kinureninas, ya que se ha demostrado que algunas de estas enzimas están asociadas a la progresión del cáncer de cerebro [Cervenka et al., 2017; Pérez de la Cruz et al., 2023], o la manera en que el sistema inmunológico responde a la enfermedad [Vázquez Cervantes et al., 2022], e incluso que podrían estar relacionadas con la producción de ciertos compuestos que favorecen la inflamación y la proliferación de células tumorales [Summers et al., 2024].

Aunque los resultados mostrados en este trabajo pueden tener limitaciones, como basarse en información de tejido cerebral y no de sangre o suero, este enfoque representa un primer paso hacia la identificación de nuevos biomarcadores que ayuden a predecir la presencia de tumores cerebrales. Además, se espera que los hallazgos de este trabajo permitan definir futuras investigaciones y protocolos clínicos en los cuales estas variables puedan ser medidas de forma menos invasiva, aumentando su viabilidad para el diagnóstico temprano.

En cuanto a la metodología estadística que se usa en este trabajo, dado que el objetivo es analizar si las variables pueden servir para definir herramientas de diagnóstico, la hipótesis principal de este trabajo es que si la información de esas variables es relevante para la predicción del estatus o tipo de cáncer, entonces al usar reglas de clasificación basadas sólo en la información de esas variables, se deberían de observar buenas tasas de clasificación. A partir de lo anterior, en este trabajo se considera la construcción de reglas de clasificación obtenidas con métodos de clasificación supervisada como regresión logística multinomial Agresti (2015, sección 6), que es un método muy sencillo y simple de explorar; así como métodos que requieren mayor poder computacional como la máquina de soporte vectorial James et al. (2021, sección 9), Hastie et al. (2009, sección 12) y los bosques aleatorios James et al. (2021, sección 8) y Hastie et al. (2009, sección 15).

Además, dado que las variables asociadas a las expresiones genéticas son cuantitativas, una opción para construir las reglas de clasificación es usar el análisis de discriminante lineal y cuadráticos James et al. (2021, sección 4.4), Hastie et al. (2009, sección 4) y Murphy (2022, sección 9). En estos métodos lo importante es estimar la matriz de covarianzas en cada población, lo que para casos con poca muestra podría ser un inconveniente. Es por esta razón que en este trabajo se opta por explorar los modelos gráficos probabilísticos [Lauritzen, 1996], en particular, los modelos gráficos gaussianos que sirven para modelar de forma multivariada la distribución de variables cuantitativas. En estos modelos las relaciones de independencia marginal y condicional entre variables se representan por medio de una gráfica, la cual puede ser dirigida [redes bayesianas; Scutari and Denis, 2021] o no dirigida Lauritzen (1996, sección 3.2.1).

En este trabajo nos centramos en los modelos con gráficas no dirigidas. Estos modelos son una opción cuando el tamaño de muestra no es tan grande, ya que si las gráficas tienen pocas aristas, entonces algunas de las entradas de la matriz de concentración (inversa de la matriz de covarianzas) con cero, lo que implica que el número de parámetros a estimar es menor y se requiere menos muestra. Uno de los retos en estos modelos es la definición de la estructura de la gráfica que se usará para una aplicación en particular; para esto existen varios métodos, algunos que se basan en métodos por pasos y criterios AIC o BIC, así como algunos que se basan en pruebas de hipótesis simultáneas, ver por ejemplo, [Højsgaard et al. \(2012, sección 4.4\)](#) y [Drton and Maathuis \(2017, sección 3\)](#). En particular, se puede usar el método *Graphical Lasso* [[Friedman et al., 2008](#)] que representa el problema de la búsqueda de la gráfica como un problema de optimización convexa con penalización L_1 , en donde se debe de calibrar un parámetro $\lambda > 0$, que tiene la característica de que entre mayor sea su valor menor el número de aristas tiene la gráfica. La calibración del parámetro λ se puede realizar usando criterios como AIC o BIC, o como se propone en [Chen \(2022\)](#) con base en el poder predictivo de reglas de clasificación basadas en modelos gráficos. Este último es el enfoque que se considera en este trabajo.

1.2 OBJETIVOS

1.2.1 Generales

El objetivo de este trabajo es mostrar y comparar los resultados obtenidos al usar los modelos gráficos probabilísticos dentro de un problema de clasificación supervisada; usando tanto datos simulados como datos sobre cáncer de cerebro.

Los resultados corresponden a las Tasas de Clasificación Correctas Globales y por grupo obtenidas usando métodos de remuestreo. La comparación se realiza con los resultados correspondientes a las reglas obtenidas con otros modelos de clasificación supervisada: regresión logística multinomial, análisis de discriminante lineal y cuadrática, máquina de soporte vectorial y los bosques aleatorios.

1.2.2 Específicos

1. Implementar y evaluar una regla de clasificación basada en el uso de los modelos gráficos gaussianos. La regla consiste es suponer que las distribuciones de un conjunto de variables continuas dentro de cada grupo o población de interés se pueden modelar con un modelo gráfico gaussiano. Con esas distribuciones se considera la regla de clasificación obtenida mediante el logaritmo del cociente de las distribuciones, como se hace en análisis de discriminante cuadrático. La estimación de la regla se realiza estimando las distribuciones (estructura de la gráfica y parámetros) en cada grupo usando el método *Graphical Lasso*, y su evaluación se realiza a partir de las tasas de clasificación obtenidas usando métodos de remuestreo.
2. Comparar las tasas de clasificación de la regla anterior con las tasas de otros métodos de clasificación supervisada en escenarios con tamaños de muestra limitados, usando

datos simulados, así como con datos sobre cáncer de cerebro. Los métodos de clasificación supervisada considerados en la comparación son: regresión logística multinomial, análisis de discriminante lineal y cuadrático, máquina de soporte vectorial y los bosques aleatorios.

3. Analizar el uso de las expresiones genéticas de las enzimas de la vía de las kinureninas para la construcción de reglas predictivas sobre el estatus de cáncer de cerebro: sano, glioma de bajo grado y glioblastoma. Para esto, las reglas de clasificación, en particular la que se basa en los modelos gráficos gaussianos, se construyen usando sólo la información de 11 enzimas de la vía de las kinureninas, es decir, con 11 variables predictoras cuantitativas; y el objetivo es analizar si es posible obtener buenas tasas de clasificación con algunos de los métodos; lo que sugeriría que la información de la vía puede servir para el diseño de herramientas de diagnóstico de cáncer de cerebro.

1.3 ESTRUCTURA DE LA TESIS

La estructura del resto de la tesis es la siguiente. En el capítulo 2 se describe la teoría básica de los modelos gráficos probabilísticos, así como la forma como se define una regla de clasificación a partir de estos; también se mencionan los conceptos fundamentales del aprendizaje supervisado con el objetivo de predicción, así como una breve descripción de los otros modelos de clasificación supervisada que se consideran en la comparación. Se concluye el capítulo con una comparación del desempeño de los métodos de clasificación usando datos simulados. En el capítulo 3, se introduce la base de datos sobre cáncer de cerebro que se utiliza para ajustar los modelos que se basan en las expresiones genéticas de las enzimas de las kinureninas; también se realiza un análisis exploratorio y se presentan las tasas de clasificación obtenidas por los diferentes modelos. Finalmente, en el capítulo 4 se discuten los resultados y se mencionan algunas áreas de mejora y posible trabajo futuro.

2

METODOLOGÍA

En este capítulo se describen las bases teóricas y las metodologías necesarias para la implementación de los diferentes modelos de clasificación utilizados en el capítulo 3. El contenido se organiza en tres secciones principales: Modelos Gráficos Probabilísticos, Aprendizaje Supervisado con el objetivo de predicción y Algoritmos de Clasificación. Cada sección aborda aspectos básicos y fundamentales para comprender el marco teórico y la metodología de este trabajo. A continuación, se detallan los objetivos específicos de cada sección:

- **Modelos Gráficos Probabilísticos:** Introducir los conceptos básicos de los modelos gráficos probabilísticos, destacando su aplicación en la representación y análisis de relaciones entre variables cuantitativas con base principal en [Højsgaard et al. \(2012, secciones 1 y 4\)](#) y [Drton et al. \(2019, sección 9\)](#). En esta sección se explora también el algoritmo *Graphical Lasso* presentado en [Hastie et al. \(2015, Sección 9\)](#), el cual será de importancia cuando se retome la discusión de los Modelos Gráficos Probabilísticos para construir una regla de clasificación.
- **Aprendizaje Supervisado:** Presentar los conceptos fundamentales del aprendizaje supervisado con el objetivo de predicción. Esto incluye una descripción de los enfoques metodológicos para la calibración de hiperparámetros y la estimación del poder predictivo de las reglas de clasificación usando métodos de remuestreo. La base principal de esta sección es [James et al. \(2021, secciones 4 y 5\)](#), [Hastie et al. \(2009, secciones 2 y 7\)](#) y [Murphy \(2022, secciones 1, 2 y 4\)](#).
- **Algoritmos de Clasificación:** Describir los algoritmos de clasificación utilizados en este trabajo, con énfasis en su implementación práctica. Se presenta el modelo basado en los modelos gráficos gaussianos y el análisis de discriminante cuadrático, el cual se denota como UGGM-QDA y que usa el algoritmo *Graphical Lasso*.

Además, se incluye una sección con un ejercicio utilizando datos simulados para comparar el rendimiento del modelo UGGM-QDA con el modelo clásico de análisis de discriminante cuadrático (QDA), así como con los otros modelos descritos en esta sección, evaluando su poder predictivo.

2.1 MODELOS GRÁFICOS PROBABILÍSTICOS

2.1.1 Definiciones

Los modelos gráficos probabilísticos son modelos multivariados que permiten describir una distribución de probabilidad que involucra múltiples variables mediante la estructura de una gráfica. En estos modelos, cada variable se representa como un nodo, y las conexiones entre nodos indican relaciones de dependencia probabilística entre las variables.

Una **gráfica** se puede definir como un conjunto G que consta de un conjunto de vértices (*nodos*) $V = \{v_1, \dots, v_p\}$ y un conjunto de aristas (*enlaces*) $E \subseteq \{\{v_i, v_j\} : v_i, v_j \in V, v_i \neq v_j\}$:

$$G = \{V, E\}. \quad (1)$$

Una **gráfica no dirigida** $G = \{V, E\}$ es aquella donde E es un conjunto no ordenado, es decir:

$$\forall v_l, v_m \in V, \{v_l, v_m\} = \{v_m, v_l\}. \quad (2)$$

Un **modelo gráfico probabilístico no dirigido** es un modelo probabilístico basado en la estructura de una gráfica no dirigida. En estos modelos cada vértice v_i es asociado con una variable aleatoria x_{v_i} .

Sea $G = \{V, E\}$ un modelo gráfico probabilístico no dirigido, con $x_V = (x_{v_1}, \dots, x_{v_p})$ un vector aleatorio asociado al conjunto de vértices $V = \{v_1, \dots, v_p\}$, diremos que G cumple con la propiedad de Markov por pares, si la función de densidad o probabilidad conjunta asociada a x_V satisface que:

$$\{v_l, v_m\} \notin E \iff x_{v_l} \perp\!\!\!\perp x_{v_m} | x_{V \setminus \{v_l, v_m\}}, \quad (3)$$

Ver [Lauritzen \(1996, sección 3.2.1\)](#).

En otras palabras, cuando no exista una arista entre dos vértices en una gráfica, las variables aleatorias asociadas a dichos vértices serán condicionalmente independientes, dado el resto de las variables. A un modelo de este tipo lo llamaremos **modelo gráfico de Markov**.

En la Figura 1 se presentan dos ejemplos de modelos gráficos probabilísticos. En estos ejemplos, las gráficas están compuestas por los siguientes elementos:

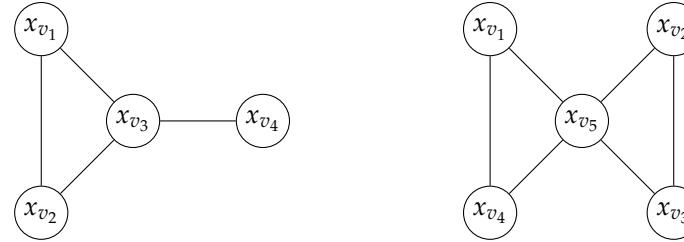
(a) $G_1 = \{V_1, E_1\}$, donde:

- $V_1 = \{v_1, \dots, v_4\}$.
- $E_1 = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_3, v_4\}\}$.

(b) $G_2 = \{V_2, E_2\}$, donde:

- $V_2 = \{v_1, \dots, v_5\}$.

- $E_2 = \{\{v_1, v_4\}, \{v_1, v_5\}, \{v_2, v_3\}, \{v_2, v_5\}, \{v_3, v_5\}, \{v_4, v_5\}\}$.



(a) $G_1 = \{V_1, E_1\}$ y $x_{V_1} = (x_{v_1}, \dots, x_{v_4})$ (b) $G_2 = \{V_2, E_2\}$ y $x_{V_2} = (x_{v_1}, \dots, x_{v_5})$

Figura 1: Ejemplos de modelos gráficos probabilísticos no dirigidos.

Notar que, para que los ejemplos de la Figura 1 se consideren modelos gráficos de Markov, deben cumplir con las siguientes condiciones:

- (a) • $x_{v_1} \perp\!\!\!\perp x_{v_4} | x_{V \setminus \{v_1, v_4\}}$
 • $x_{v_2} \perp\!\!\!\perp x_{v_4} | x_{V \setminus \{v_2, v_4\}}$
- (b) • $x_{v_1} \perp\!\!\!\perp x_{v_2} | x_{V \setminus \{v_1, v_2\}}$
 • $x_{v_1} \perp\!\!\!\perp x_{v_3} | x_{V \setminus \{v_1, v_3\}}$
 • $x_{v_2} \perp\!\!\!\perp x_{v_4} | x_{V \setminus \{v_2, v_4\}}$
 • $x_{v_3} \perp\!\!\!\perp x_{v_4} | x_{V \setminus \{v_3, v_4\}}$

Este trabajo se centra en los modelos gráficos gaussianos, un caso particular de los modelos gráficos de Markov. En estos modelos, se asume que las variables son cuantitativas y siguen una distribución normal o gaussiana multivariada.

2.1.2 Distribución normal o gaussiana multivariada

Sea $\mathbf{x} = (x_1, \dots, x_p) \in \mathbb{R}^p$ un vector aleatorio de dimensión p que sigue una distribución normal multivariada, es decir $\mathbf{x} \sim N(\boldsymbol{\mu}, \Sigma)$, entonces la función de densidad está dada por:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}, \quad (4)$$

donde:

- $\boldsymbol{\mu} = (\mu_1, \dots, \mu_p) \in \mathbb{R}^p$ es un vector de medias de dimensión p que representa las medias de cada variable.
- $\Sigma \in \mathbb{R}^{p \times p}$ es la matriz de covarianza de dimensión $p \times p$, que captura la varianza de cada variable en la diagonal y las covarianzas entre pares de variables en las posiciones fuera de la diagonal.
- $|\Sigma|$ es el determinante de la matriz de covarianza.

- Σ^{-1} es la inversa de la matriz de covarianza, también llamada matriz de precisión o concentración.

Sea $\{\mathbf{x}_1 = (x_{11}, \dots, x_{1p}), \dots, \mathbf{x}_n = (x_{n1}, \dots, x_{np})\}$ una muestra aleatoria de tamaño n de $\mathbf{x} = (x_1, \dots, x_p) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Los parámetros $\boldsymbol{\mu}$ y $\boldsymbol{\Sigma}$ pueden ser estimados a partir de la muestra utilizando los estimadores de máxima verosimilitud (MLE, *Maximum Likelihood Estimators*).

Los estimadores MLE, de acuerdo con [Mardia et al. \(1979\)](#), son los siguientes:

$$\hat{\boldsymbol{\mu}}_{\text{MLE}} = (\hat{\mu}_1, \dots, \hat{\mu}_p), \quad (5)$$

donde $\hat{\mu}_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$ con $j \in \{1, \dots, p\}$ y

$$\hat{\boldsymbol{\Sigma}}_{\text{MLE}} = S, \quad (6)$$

donde S es la matriz de covarianza muestral y se define como:

$$S = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{\text{MLE}}) (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{\text{MLE}})^T.$$

2.1.3 Modelos gráficos gaussianos no dirigidos

Sea $G = \{V, E\}$ un modelo gráfico probabilístico no dirigido, donde $\mathbf{x}_V = (x_{v_1}, \dots, x_{v_p})$ es el vector aleatorio asociado al conjunto de vértices $V = \{v_1, \dots, v_p\}$. Diremos que G es un modelo gráfico gaussiano no dirigido si se cumple que $\mathbf{x}_V \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Este tipo de modelo gráfico se denominará como UGGM (*Undirected Gaussian Graphical Model*).

De esta forma:

$$p(\mathbf{x}_V) = \frac{1}{(2\pi)^{\frac{p}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2} (\mathbf{x}_V - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_V - \boldsymbol{\mu}) \right\}. \quad (7)$$

De acuerdo con [Hastie et al. \(2015\)](#), la función de densidad descrita en la expresión (7) puede parametrizarse como un miembro de la familia exponencial multivariada de la siguiente forma:

$$p(\mathbf{x}_V) = \exp \left\{ \sum_{s=1}^p \gamma_s x_{v_s} - \frac{1}{2} \sum_{s=1}^p \sum_{t=1}^p \theta_{st} x_{v_s} x_{v_t} + \frac{1}{2} \ln \left(\left| \frac{\boldsymbol{\Theta}}{2\pi} \right| \right) \right\}, \quad (8)$$

donde

$$\boldsymbol{\mu} = -\boldsymbol{\Theta}^{-1} \boldsymbol{\gamma}, \quad \text{y} \quad \boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \theta_{11} & \cdots & \theta_{1p} \\ \vdots & \ddots & \vdots \\ \theta_{p1} & \cdots & \theta_{pp} \end{pmatrix}; \quad (9)$$

Ver [Wainwright and Jordan \(2008, ejemplo 3.3\)](#).

Aquí, $\gamma \in \mathbb{R}^p$ y $\Theta \in \mathbb{R}^{p \times p}$ son los parámetros canónicos, donde Θ es una matriz simétrica que se conoce como **matriz de concentración** y es fundamental para describir las relaciones de independencia condicional entre las variables aleatorias en un modelo UGGM, como se establece en el Resultado 1.

Resultado 1 (Equivalencias UGGM). *Sea $l \neq m$. En un modelo UGGM, las siguientes condiciones son equivalentes:*

1. $\theta_{lm} = 0$
2. $x_{v_l} \perp\!\!\!\perp x_{v_m} | \mathbf{x}_{V \setminus \{v_l, v_m\}}$
3. $\{v_l, v_m\} \notin E$

Una demostración de este resultado puede ser consultada en [Lauritzen \(1996, sección 5.1.3\)](#).

En otras palabras, en los modelos gráficos gaussianos, las independencias condicionales, representadas por la ausencia de aristas en la gráfica, se reflejan como ceros en la matriz de concentración.

De esta forma, si los modelos de la Figura 1 son UGGM, entonces las matrices de concentración asociadas tendrán la siguiente forma:

$$\Theta_{V_1} = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} & 0 \\ \theta_{21} & \theta_{22} & \theta_{23} & 0 \\ \theta_{31} & \theta_{32} & \theta_{33} & \theta_{44} \\ 0 & 0 & \theta_{43} & \theta_{44} \end{pmatrix} \quad \text{y} \quad \Theta_{V_2} = \begin{pmatrix} \theta_{11} & 0 & 0 & \theta_{14} & \theta_{15} \\ 0 & \theta_{22} & \theta_{23} & 0 & \theta_{25} \\ 0 & \theta_{32} & \theta_{33} & 0 & \theta_{35} \\ \theta_{41} & 0 & 0 & \theta_{44} & \theta_{45} \\ \theta_{51} & \theta_{52} & \theta_{53} & \theta_{54} & \theta_{55} \end{pmatrix}. \quad (10)$$

2.1.4 Datos simulados

Para ilustrar los conceptos discutidos anteriormente, se generaron datos simulados basados en un modelo UGGM. Estos datos permitirán ejemplificar algunos aspectos del modelo, como la estructura de dependencia entre variables, la estimación de la gráfica asociada, y la estimación de los parámetros asociados al modelo.

Posteriormente, se utilizará el mismo procedimiento para simular diferentes conjuntos de datos que servirán para ejemplificar el uso de un modelo UGGM con el objetivo de predicción.

Para simular los datos, se utilizó una estructura para la matriz de concentración basada en una gráfica aleatoria. Esta gráfica fue generada utilizando la función `sample_pa()` del paquete `igraph` [[Csardi and Nepusz, 2006](#)], que implementa un modelo de adjunción preferencial (*preferential attachment*).

De acuerdo con la documentación de la función `sample_pa()`, el proceso de construcción de la gráfica comienza con un solo vértice y sin aristas en el primer paso. Luego, en cada paso siguiente, se añade un nuevo vértice que inicia algunas aristas hacia vértices ya existentes. La

probabilidad de que un vértice anterior sea seleccionado para conectar con el vértice nuevo está dada por:

$$p(v_i) \sim k_{v_i}^\alpha + a, \quad (11)$$

donde k_{v_i} es el grado de entrada (*indegree*) del vértice v_i en el paso actual, es decir, el número de aristas conectadas a v_i que no fueron iniciadas por el propio v_i . Los parámetros α y a son parámetros dados por los argumentos `power` y `zero.appeal` en la función `sample_pa()`.

Para los datos simulados de esta sección, los parámetros de la función `sample_pa()` se definieron como $\alpha = 1.75$ y $a = 1$, utilizando un total de 10 vértices. La estructura de la gráfica puede ser visualizada en la Figura 2, que en este caso resultó en un árbol, aunque el algoritmo podría dar cualquier otro tipo de gráfica.

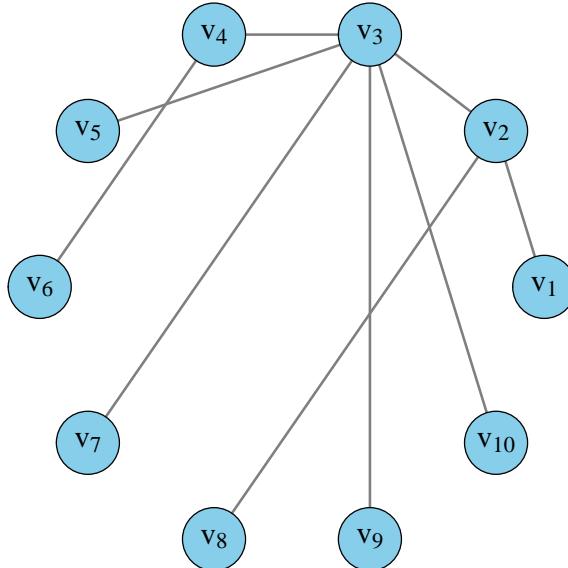


Figura 2: Estructura de la gráfica asociada generada mediante la función `sample_pa()` con los parámetros `power = 1.75` y `zero.appeal = 1`. La gráfica sigue un modelo de adjunción preferencial que captura relaciones basadas en la conectividad acumulativa de los vértices.

Una vez determinada la estructura de la gráfica asociada, el procedimiento para especificar la matriz de covarianza se describe en el Algoritmo 1, ver Perez-de-la Cruz and Eslava-Gomez (2016, Anexo A.2).

input :

- Gráfica no dirigida: $G = \{V, E\}$
- Constante $R > 1$
- Intervalo D

output:

- Matriz de covarianza: Σ

1 Definición de la matriz A con entradas a_{ij} :

$$a_{ij} \leftarrow \begin{cases} u_{ij}, & \text{si } \{v_i, v_j\} \in E \\ 0, & \text{si } \{v_i, v_j\} \notin E \end{cases}, \text{ donde } u_{ij} \text{ son valores de una distribución U}(D).$$

2 Matriz A diagonal dominante:

$$a_{ii} \leftarrow R \sum_{j \neq i} |a_{ij}|, \text{ donde } R > 1.$$

3 Definición de la matriz Σ con entradas σ_{ij} :

$$\sigma_{ij} \leftarrow \frac{a^{ij}}{\sqrt{a^{ii}a^{jj}}}, \text{ donde } a^{ij} \text{ es la entrada } ij \text{ de la matriz } A^{-1}.$$

Algoritmo 1: Generación de una matriz de covarianza Σ a partir de una gráfica no dirigida $G = \{V, E\}$. El procedimiento define una matriz A basada en las aristas de G , donde los valores en las entradas se obtienen de una distribución uniforme $U(D)$. La matriz A se transforma en diagonal dominante mediante el parámetro $R > 1$, y finalmente se define Σ utilizando la normalización basada en A^{-1} .

Dada la gráfica $G = \{V, E\}$ asociada a la Figura 2, se utilizó el Algoritmo 1 con los parámetros $R = 1.01$ y $D = [-1, -0.5] \cup [0.5, 1]$ para obtener la matriz de covarianza descrita en la expresión (12).

$$\Sigma = \begin{pmatrix} 1.00 & -0.94 & -0.88 & 0.77 & -0.81 & 0.72 & -0.78 & -0.88 & 0.81 & 0.80 \\ -0.94 & 1.00 & 0.93 & -0.82 & 0.86 & -0.77 & 0.83 & 0.93 & -0.87 & -0.85 \\ -0.88 & 0.93 & 1.00 & -0.88 & 0.92 & -0.83 & 0.89 & 0.87 & -0.93 & -0.91 \\ 0.77 & -0.82 & -0.88 & 1.00 & -0.81 & 0.93 & -0.79 & -0.77 & 0.82 & 0.81 \\ -0.81 & 0.86 & 0.92 & -0.81 & 1.00 & -0.76 & 0.82 & 0.80 & -0.86 & -0.84 \\ 0.72 & -0.77 & -0.83 & 0.93 & -0.76 & 1.00 & -0.74 & -0.72 & 0.77 & 0.76 \\ -0.78 & 0.83 & 0.89 & -0.79 & 0.82 & -0.74 & 1.00 & 0.77 & -0.83 & -0.82 \\ -0.88 & 0.93 & 0.87 & -0.77 & 0.80 & -0.72 & 0.77 & 1.00 & -0.81 & -0.79 \\ 0.81 & -0.87 & -0.93 & 0.82 & -0.86 & 0.77 & -0.83 & -0.81 & 1.00 & 0.85 \\ 0.80 & -0.85 & -0.91 & 0.81 & -0.84 & 0.76 & -0.82 & -0.79 & 0.85 & 1.00 \end{pmatrix}. \quad (12)$$

De esta forma, la matriz de concentración asociada $\Theta = \Sigma^{-1}$ es descrita en la expresión (13).

$$\Theta = \begin{pmatrix} 8.67 & 8.16 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 8.16 & 21.67 & -7.03 & 0.00 & 0.00 & 0.00 & 0.00 & -6.93 & 0.00 & 0.00 \\ 0.00 & -7.03 & 32.06 & 4.04 & -6.14 & 0.00 & -4.33 & 0.00 & 6.77 & 5.60 \\ 0.00 & 0.00 & 4.04 & 11.51 & 0.00 & -7.42 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & -6.14 & 0.00 & 6.66 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & -7.42 & 0.00 & 7.94 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & -4.33 & 0.00 & 0.00 & 0.00 & 4.86 & 0.00 & 0.00 & 0.00 \\ 0.00 & -6.93 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 7.45 & 0.00 & 0.00 \\ 0.00 & 0.00 & 6.77 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 7.29 & 0.00 \\ 0.00 & 0.00 & 5.60 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 6.12 \end{pmatrix}. \quad (13)$$

Con la matriz de covarianza generada, se simula una muestra de 1000 observaciones a partir de una distribución normal multivariada, utilizando como parámetros la matriz Σ , definida en la expresión (12), y un vector de medias centrado en cero. Para ello, se emplea la función `mvnrm()` del paquete MASS [Ripley and Venables, 2023].

Una vez simulados los datos, se calcularon las correlaciones parciales entre cada par de variables, cuyos resultados se presentan en la Figura 3. El objetivo principal de este análisis es examinar cómo las correlaciones parciales cercanas a cero están asociadas a la ausencia de aristas en la gráfica representada en la Figura 2, así como a las entradas con valor cero en la matriz de concentración, descrita en la expresión (13).

Matriz de Correlaciones Parciales de Pearson

Datos Simulados

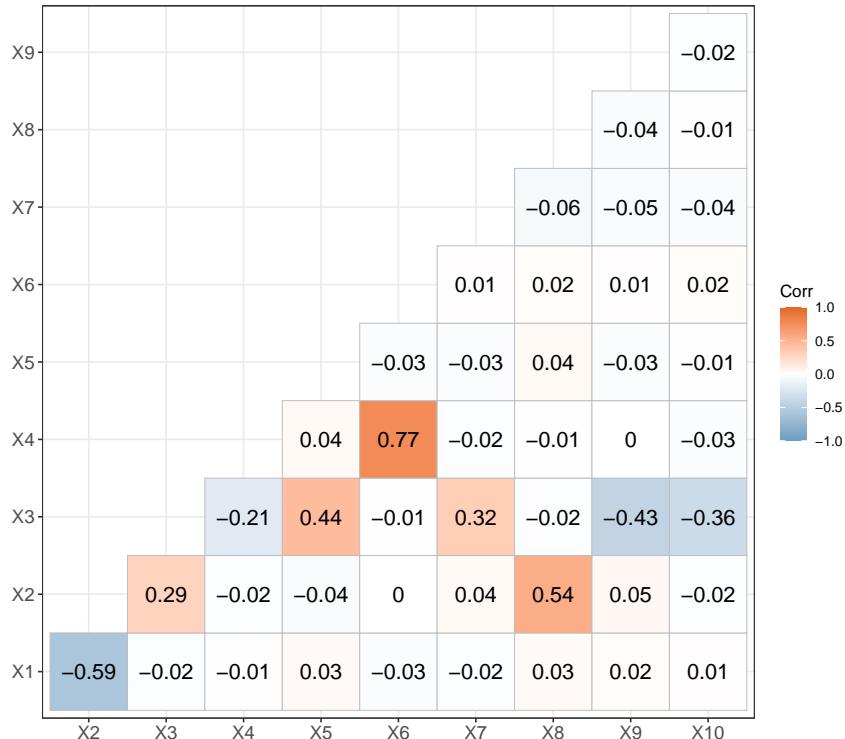


Figura 3: Correlaciones parciales entre variables de los datos simulados. Los valores cercanos a cero se asocian a la ausencia de aristas de la Figura 2 y a las entradas con valor cero de la expresión (13).

2.1.5 Problemas de estimación en los modelos gráficos gaussianos

Al trabajar con un modelo UGGM, existen dos problemas fundamentales en el proceso de estimación:

- Estimación de los parámetros: Estimación de μ y de la matriz de concentración Θ cuando se asume que la estructura de la gráfica G es conocida.
- Estimación de la gráfica: Estimación de la estructura o selección del modelo, donde el objetivo es identificar o inferir la estructura de la gráfica de dependencias entre las variables.

Estimación de los parámetros

El objetivo es la estimación de μ y de la matriz de concentración Θ cuando se asume que la estructura de la gráfica G es conocida.

Sea $\{x_1, \dots, x_n\}$ una muestra aleatoria de tamaño n asociada a un modelo UGGM, con $G = \{V, E\}$ la gráfica asociada al modelo. Se puede demostrar que la función de log verosimilitud ℓ_{x_V} asociada es:

$$\ell_{x_V}(\Theta) = \frac{1}{n} \sum_{i=1}^n \ln [p(x_i)] \quad (14)$$

$$= \ln(|\Theta|) - \text{tr}(S\Theta). \quad (15)$$

De esta forma la matriz de concentración Θ puede ser estimada a partir de la muestra utilizando el estimador MLE, como se plantea a continuación:

$$\hat{\Theta}_{\text{MLE}} = \arg \max_{\Theta \in \Theta_G} \{\ell_{x_V}(\Theta)\}, \quad (16)$$

donde

$$\Theta_G = \{\Theta | \theta_{lm} = 0, \forall \{v_l, v_m\} \notin E\} \quad \text{y} \quad \ell_{x_V}(\Theta) = \ln(|\Theta|) - \text{tr}(S\Theta), \quad (17)$$

con

$$\ln(|\Theta|) = \begin{cases} \sum_{j=1}^p \ln(\lambda_j(\Theta)), & \text{si } \Theta \text{ es definida positiva} \\ -\infty, & \text{e.o.c} \end{cases}. \quad (18)$$

Aquí, $\lambda_j(\Theta)$ es el j -ésimo eigenvalor de Θ .

Estimación de la gráfica

El objetivo es la estimación de la estructura o selección del modelo, donde el objetivo es identificar o inferir la estructura de la gráfica de dependencias entre las variables.

Existen diversos métodos para determinar la gráfica asociada a un modelo UGGM. Un método popular para la selección de modelos es utilizar un enfoque paso a paso (*stepwise*). Este puede comenzar con una gráfica vacía (o completa) y ejecutar una búsqueda hacia adelante (*forward*) o hacia atrás (*backward*), añadiendo o eliminando aristas si esto mejora algún criterio. Alternativamente, se puede buscar la arista que minimice un criterio específico y añadirla o eliminarla, aunque este método es considerablemente más lento. Dos funciones objetivo populares para este proceso son el Criterio de Información de Akaike (AIC) y el Criterio de Información Bayesiana (BIC), los cuales penalizan la verosimilitud en función de la complejidad del modelo, es decir:

$$-2\ell_{x_v}(\Theta) + \lambda|E|, \quad (19)$$

donde ℓ_{x_v} es la función de log verosimilitud, λ es un hiperparámetro que penaliza la complejidad del modelo y $|E|$ denota el número de vértices. En este caso el criterio AIC se define con $\lambda = 2$, mientras que el criterio BIC se define con $\lambda = \ln(n)$.

El problema principal de los métodos por pasos es que resultan poco prácticos para problemas de gran escala, ya que solo se puede cubrir una pequeña parte del espacio de búsqueda relevante.

Una alternativa es el uso del método *Graphical Lasso* [Friedman et al., 2008], el cual penaliza la función de verosimilitud, como se describe a continuación:

$$\ell_{x_v}(\Theta) - \lambda \sum_{l \neq m} |\theta_{lm}|. \quad (20)$$

Aquí, $\lambda \geq 0$ se considera un hiperparámetro que penaliza la complejidad del modelo y $\sum_{l \neq m} |\theta_{lm}|$ es la suma de los valores absolutos de los elementos fuera de la diagonal de la matriz de concentración. Valores muy grandes de λ hacen que varias entradas de la matriz de concentración Θ sean cero, de manera que variando λ se exploran varios modelos.

Una de las principales ventajas del método *Graphical Lasso* para la selección de modelos gráficos gaussianos es su aplicación en problemas de alta dimensión. Para un análisis más detallado de las ventajas del algoritmo *Graphical Lasso* en contextos de alta dimensión, se puede consultar [Ravikumar et al., 2008].

De esta forma, dado un valor de $\lambda \geq 0$, [Hastie et al. \(2015\)](#) plantea el siguiente problema de optimización:

$$\widehat{\Theta}_{\text{glasso}} = \arg \max_{\Theta} \left\{ \ell_{x_v}(\Theta) - \lambda \sum_{l \neq m} |\theta_{lm}| \right\}. \quad (21)$$

El problema de optimización descrito en la expresión (21) es un problema de optimización convexo, lo que permite el uso de métodos clásicos de punto interior para encontrar la solución. Sin embargo, estos métodos resultan ineficientes en problemas de alta dimensión.

Una versión más eficiente para problemas de alta dimensión se discute en [Witten et al. \(2011\)](#), donde la optimización se lleva a cabo mediante una descomposición bloque a bloque. En este enfoque, las columnas (o filas, debido a la simetría de Θ) se actualizan iterativamente resolviendo un problema de regresión *Lasso* en cada paso. Este método, conocido como *glasso* está implementado en el paquete *glasso* [[Friedman et al., 2021](#)] y se describe de manera general en el Algoritmo 2.

De esta forma, el método *Graphical Lasso* (*glasso*) cumple dos funciones simultáneamente:

- Estimación de la matriz de concentración Θ , la cual representa las relaciones condicionales entre variables.
- Aprendizaje estructural, es decir, la selección de la gráfica que representa las conexiones entre las variables.

Al igual que en un método *Lasso* estándar de regresión, la penalización en *glasso* fuerza a ciertos elementos de Θ a ser exactamente cero, eliminando las aristas correspondientes de la gráfica. De esta forma, el método *glasso* no solo estima los parámetros del modelo, sino que también determina qué relaciones son estructuralmente importantes.

Es importante notar que el método *glasso* (Algoritmo 2) resuelve el problema de optimización planteado en la expresión (21), utilizando un valor fijo del hiperparámetro λ . Dependiendo del objetivo del problema, se pueden emplear distintas metodologías para la selección de λ . A continuación se mencionan algunas de estas metodologías:

- **Validación Cruzada:** Si el objetivo del problema es predictivo, se puede definir un conjunto de valores candidatos para λ y seleccionar aquel que minimice una métrica de desempeño mediante un esquema de validación cruzada. Dado que el propósito de este trabajo es explorar el uso de modelos gráficos gaussianos en un contexto predictivo, se implementó el Algoritmo 6 para el tuneo de λ utilizando validación cruzada.
- **Criterios de información AIC o BIC:** Si el objetivo es estructural y se busca obtener una gráfica interpretable, se pueden utilizar criterios de información como AIC o BIC para la selección de λ . En [Yuan and Lin \(2007\)](#), se analiza el uso de estos criterios en modelos gráficos gaussianos.
- **Selección de Estabilidad (*Stability Selection*):** Este enfoque genera múltiples subconjuntos de datos (por ejemplo, usando *bootstrap*), ajusta el algoritmo *glasso* con distintos valores de λ y mide qué tan consistentemente parecen las conexiones en la gráfica resultante. Este método es preferible cuando se quiere aprender una estructura robusta en datos de alta dimensión y fue propuesto en [Meinshausen and Bühlmann \(2009\)](#).

Para obtener más información sobre la penalización *Lasso*, se recomienda consultar [Bühlmann and van de Geer \(2011\)](#). Asimismo, para un análisis más detallado sobre modelos gráficos en alta dimensionalidad, se sugiere revisar [Liang and Jia \(2023\)](#).

input :

- Matriz de covarianza muestral: S
- Hiperparámetro: $\lambda \geq 0$

output:

- Matriz de concentración estimada: $\widehat{\Theta}$

1 *Iniciar:*

$$W \leftarrow S + \lambda I, \quad \text{con } I \text{ la matriz identidad}$$

2 **repeat**

3 **for** $j \leftarrow 1$ **to** p **do**

4 *Particionar la matriz W:*

- W_{11} : Todos los elementos excepto la j -ésima fila y columna.
- w_{12} : La j -ésima columna.

5 *Resolver el problema, usando algún algoritmo de descenso de coordenadas cíclicas:*

$$W_{11}\beta - s_{12} + \lambda \operatorname{sign}(\beta) = 0$$

6 *Actualizar:*

$$w_{12} \leftarrow W_{11}\widehat{\beta}$$

7 **end**

8 *Verificar criterio de convergencia.*

9 **until** *Convergencia;*

10 **for** $j \leftarrow 1$ **to** p **do**

11 *Actualizar:*

$$\widehat{\theta}_{12} \leftarrow -\widehat{\beta} \cdot \widehat{\theta}_{22}, \quad \text{con } \frac{1}{\widehat{\theta}_{22}} = w_{22} - w_{12}^T \widehat{\beta}.$$

12 **end**

13 **if** se cumple criterio de convergencia **then**

14 | Devolver $\widehat{\Theta}$

15 **end**

16 **else**

17 | devolver la mejor solución obtenida hasta el momento.

18 **end**

Algoritmo 2: Algoritmo glasso, utilizado para estimar la matriz de concentración.

El algoritmo se basa en la partición de la matriz de covarianza en bloques y la optimización penalizada mediante descenso de coordenadas cíclicas.

Para ejemplificar el uso del algoritmo `glasso`, se calculó S a partir de los datos simulados. Posteriormente, se utilizó la función `glasso()` con distintos valores de λ para evaluar cómo cambia la estructura de la gráfica asociada a la matriz de concentración. La comparación se realizó con la estructura teórica asociada a la expresión (13) y visualizada en la Figura 4, con el objetivo de determinar si la estructura obtenida a partir de los datos simulados es similar a la esperada.

Modelo Gráfico basado en Correlaciones Parciales

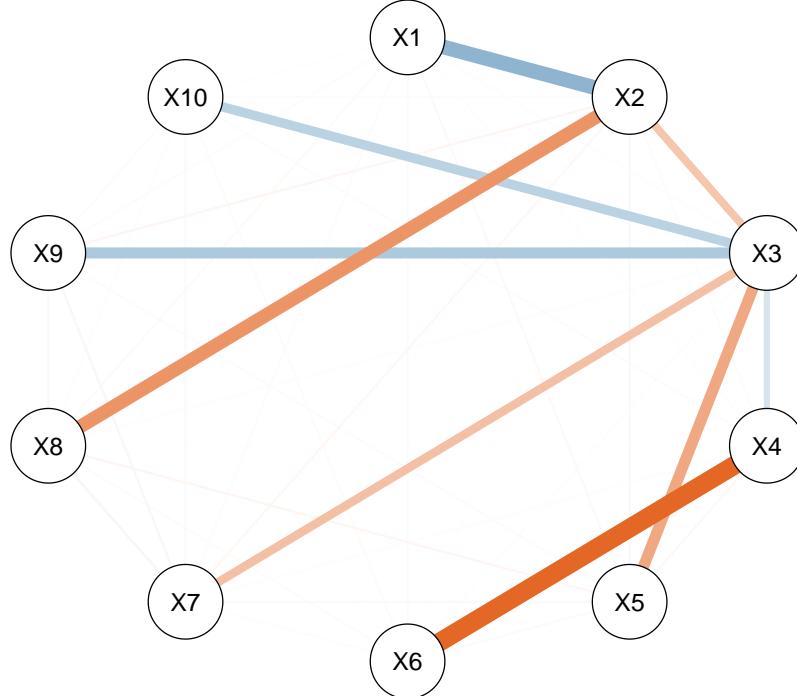


Figura 4: Representación del modelo gráfico asociado a la Figura 2, construido a partir de la estructura teórica descrita en la expresión (13). Las aristas rojas indican correlaciones parciales positivas, mientras que las azules representan correlaciones parciales negativas.

En la Figura 5 se muestra la estimación de la gráfica representada a través de las correlaciones parciales, las cuales están acotadas en el intervalo $[-1, 1]$. Para la estimación, se utilizó una malla de 6 valores de λ , donde $\lambda \in \{0.82, 0.86, \dots, 0.94\}$. Al comparar con la Figura 4, se observa que la gráfica estimada que mejor recupera la estructura teórica es aquella que utiliza el valor $\lambda = 0.88$ aunque falta la arista entre x_3 y x_7 . Notar que a mayor valor de λ se obtiene una gráfica con menos aristas, es decir, λ es un hiperparámetro que sirve para calibrar la complejidad del modelo.

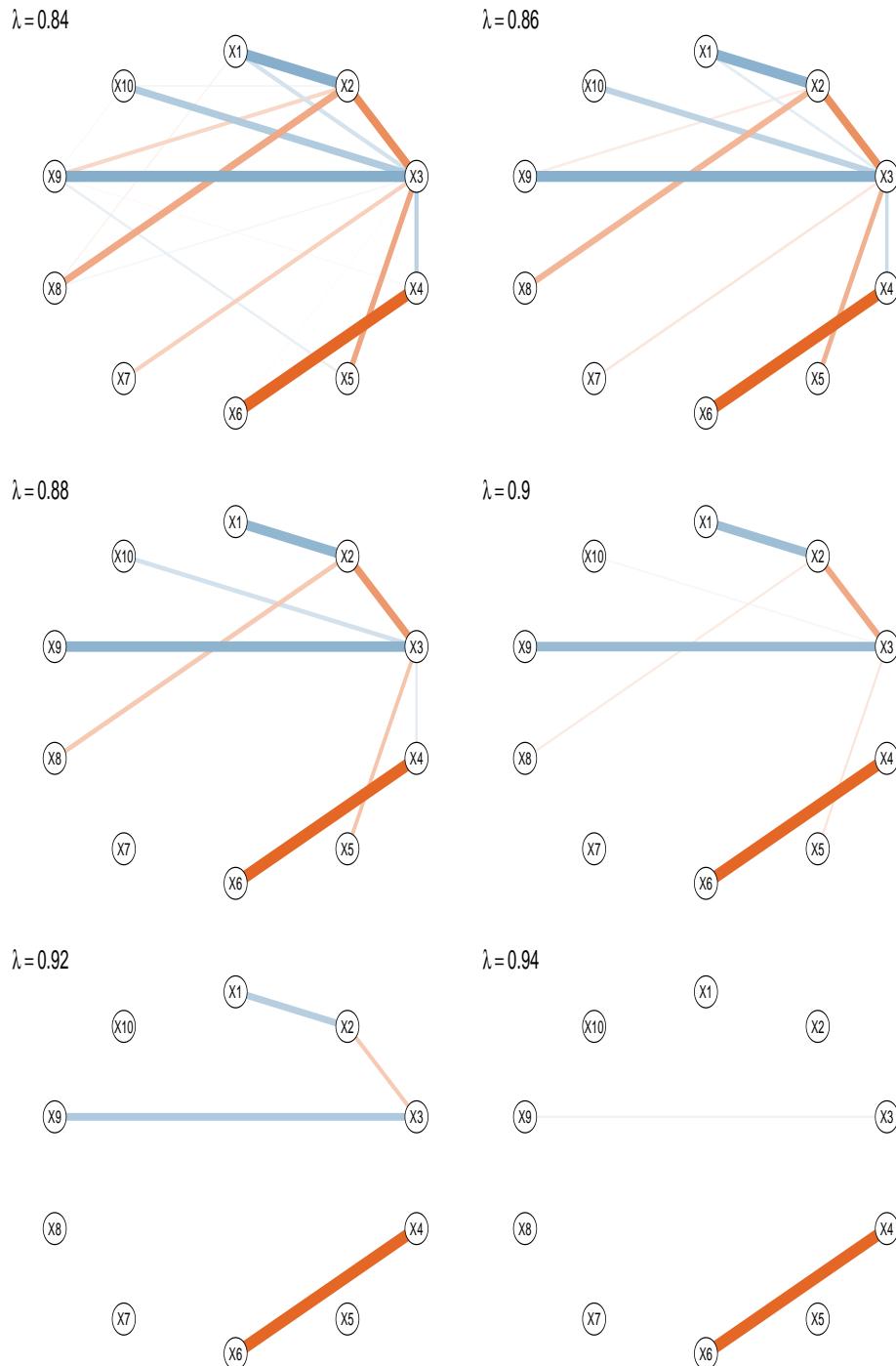


Figura 5: Correlaciones parciales obtenidas para distintos valores del hiperparámetro λ en el algoritmo glasso. Las aristas rojas indican correlaciones parciales positivas, mientras que las azules representan correlaciones parciales negativas. Esta Figura ilustra cómo varía la estructura de la gráfica de dependencias entre variables para diferentes valores de lambda λ .

2.2 APRENDIZAJE SUPERVISADO CON EL OBJETIVO DE PREDICCIÓN

Consideremos un conjunto de datos que consta de n observaciones, representadas a través de p variables distintas:

$$\mathcal{X} = \left\{ \mathbf{x}_i \in \mathbb{R}^p | i = 1, \dots, n \right\}. \quad (22)$$

Se denomina al conjunto \mathcal{X} como el conjunto de datos de entrada, y a sus elementos \mathbf{x}_i se les llama observaciones. Es común el uso de una representación matricial de \mathcal{X} , donde los datos se expresan como valores x_{ij} . Aquí, $i \in \{1, \dots, n\}$ se refiere a la i -ésima observación, mientras que $j \in \{1, \dots, p\}$ indica la j -ésima variable de cada observación. Es decir:

$$\mathcal{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}. \quad (23)$$

Además, en **aprendizaje supervisado** se supone que cada elemento $\mathbf{x}_i \in \mathcal{X}$ está etiquetado con un único valor $y_i \in \mathcal{Y}$. Se denomina al conjunto \mathcal{Y} como el conjunto de etiquetas, y en general, se pueden distinguir dos casos:

- *Regresión*: Cuando el conjunto \mathcal{Y} representa una variable cuantitativa.
- *Clasificación*: Cuando el conjunto \mathcal{Y} toma una cantidad finita de valores $\{C_1, \dots, C_k\}$ con $k \in \mathbb{N}$. Los elementos C_1, \dots, C_k son conocidos como *clases* o *etiquetas*.

Al conjunto conformado por la relación entre los datos de entrada \mathcal{X} y las etiquetas \mathcal{Y} se le denomina como conjunto de datos etiquetados y se denota como \mathcal{D} :

$$\mathcal{D} = \left\{ (y_i, \mathbf{x}_i) \in \mathbb{R}^{p+1} | i = 1, \dots, n \right\}. \quad (24)$$

También es posible utilizar una representación matricial del conjunto \mathcal{D} :

$$\mathcal{D} = [(y_1, \mathbf{x}_1), \dots, (y_n, \mathbf{x}_n)]^T = \begin{pmatrix} y_1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ y_n & x_{n1} & \cdots & x_{np} \end{pmatrix}. \quad (25)$$

El aprendizaje supervisado con el objetivo de predicción es un enfoque dentro del campo del aprendizaje automático (*machine learning*), donde se **entrena** un modelo utilizando un conjunto de datos etiquetados \mathcal{D} con el propósito de realizar **predicciones** sobre y para **nuevas observaciones**. En el contexto del aprendizaje supervisado, se considera que una observación es nueva cuando no se dispone de información previa acerca de la etiqueta asociada a dicha observación. El conjunto de nuevas observaciones se denota como \mathcal{X}^* , y sus elementos como \mathbf{x}^* .

Un algoritmo de aprendizaje supervisado \mathcal{A}_Λ se define como una función que mapea un conjunto de datos etiquetados \mathcal{D} a una función \hat{f} .

$$\begin{aligned} \mathcal{A}_\Lambda : \mathcal{D} &\longrightarrow f \\ \mathcal{D} &\longmapsto \hat{f}. \end{aligned} \quad (26)$$

A su vez, la función \hat{f} mapea nuevas observaciones x^* a etiquetas dentro del conjunto \mathcal{Y} . Estas etiquetas se denominan **predicciones** y se denotan como \hat{y} .

$$\begin{aligned}\hat{f} : \mathcal{X}^* &\longrightarrow \mathcal{Y} \\ x^* &\longmapsto \hat{y}.\end{aligned}\tag{27}$$

La función \hat{f} recibe el nombre de modelo ajustado, y es el resultado del algoritmo \mathcal{A}_Λ . Es importante mencionar que en varios casos un algoritmo de aprendizaje \mathcal{A}_Λ puede depender de ciertas configuraciones que son calibradas (*tuneadas*) a través de un proceso conocido como **calibración de hiperparámetros** o **tuneo**. Estas configuraciones son conocidas como hiperparámetros y se denotan como $\Lambda = \{\Lambda_1, \dots, \Lambda_J\}$. Es decir, el ajuste se realiza asumiendo valores de los hiperparámetros pero se debe realizar también el proceso de *tuneo*.

Dentro del proceso de ajuste y tuneo se introduce el concepto de función de pérdida, denotada por \mathcal{L} . Esta función se utiliza para estimar los parámetros (ajuste) y calibrar (*tunear*) los hiperparámetros del modelo durante las etapas de ajuste y tuneo, respectivamente.

$$\begin{aligned}\mathcal{L} : \mathcal{Y} \times \hat{\mathcal{Y}} &\longrightarrow \mathbb{R} \\ (y, \hat{y}) &\longmapsto r.\end{aligned}\tag{28}$$

Es importante mencionar que la función de pérdida puede diferir entre las etapas de ajuste y tuneo. Además, el proceso de ajuste y tuneo abarca todas las fases y aspectos relacionados con el algoritmo \mathcal{A}_Λ , incluyendo la estimación de los parámetros del modelo y, en caso de ser necesario, la calibración (*tuneo*) de hiperparámetros. Cabe destacar que el tuneo no siempre se lleva a cabo, ya que su aplicación depende de la naturaleza del problema o del enfoque adoptado.

Después de completar el proceso de ajuste y tuneo, es necesario someter el algoritmo \mathcal{A}_Λ a una evaluación que permita visualizar su capacidad de generalización y desempeño predictivo para nuevas observaciones. En otras palabras, se busca evaluar la capacidad del modelo para hacer predicciones correctas (poder predictivo).

Dentro del proceso de evaluación, se pueden emplear diversas métricas para evaluar el rendimiento del modelo ajustado. Una métrica se define como una función que proporciona una medida cuantitativa de aspectos específicos, tales como la exactitud, la sensibilidad y la precisión, entre otros. Esta métrica podría ser usada como la función de pérdida para el *tuneo*.

Para llevar a cabo el proceso de ajuste, *tuneo* (si es requerido) y evaluación, es común realizar una división (*split*) de los datos etiquetados \mathcal{D} , dependiendo de si se considera tunear o no los hiperparámetros. La forma más sencilla, aunque no la única, es:

- Con tuneo: $\mathcal{D}_{\text{Train}^*} = \mathcal{D}_{\text{Train}} \cup \mathcal{D}_{\text{Val}}$ y $\mathcal{D}_{\text{Test}}$ (ver Figura 6).
- Sin tuneo: $\mathcal{D}_{\text{Train}}$ y $\mathcal{D}_{\text{Test}}$ (ver Figura 7).

Este enfoque, aunque sencillo y directo, es fundamental para entender las bases del aprendizaje supervisado. No obstante, más adelante se abordará el método de validación cruzada, que

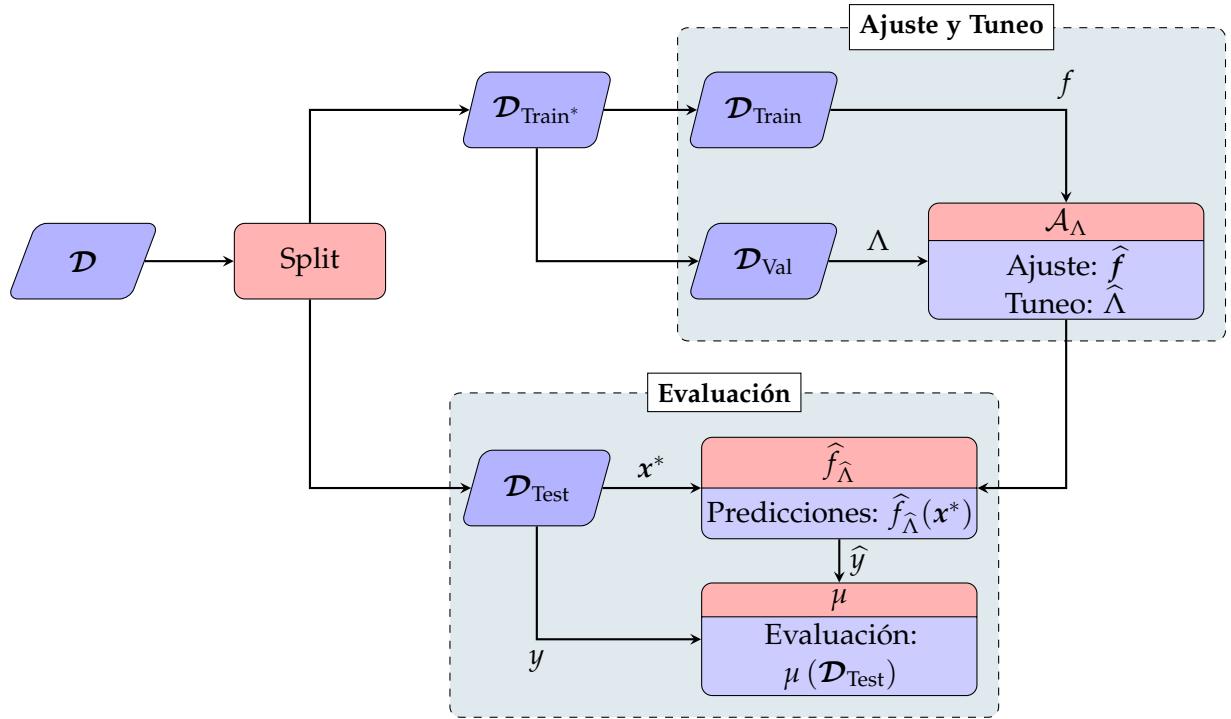


Figura 6: Esquema general de la división del conjunto \mathcal{D} y de los procesos de Ajuste, Tuneo y Evaluación.

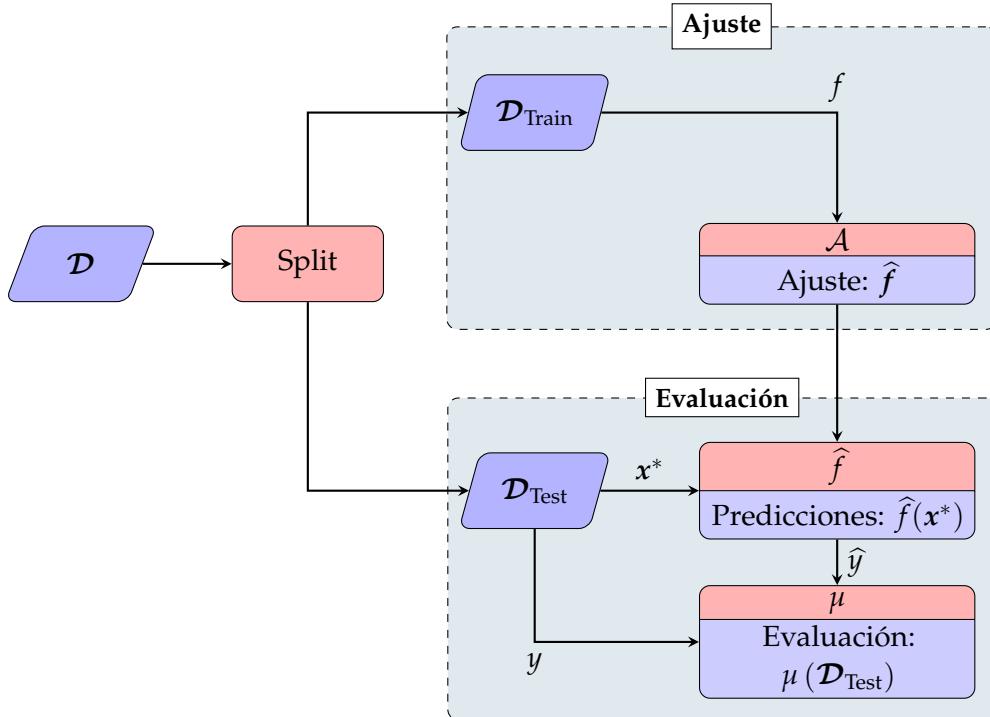


Figura 7: Esquema general de la división del conjunto \mathcal{D} y de los procesos de Ajuste y Evaluación.

permiten realizar los procesos de tuneo y evaluación de manera más robusta, al aprovechar de manera más eficiente la totalidad de los datos disponibles.

Por ejemplo, en la Figura 6, una vez realizada la división, cada conjunto se utilizará para la estimación de parámetros, tuneo de hiperparámetros y para evaluar el rendimiento del modelo ajustado, respectivamente. Es decir:

- $\mathcal{D}_{\text{Train}}$: Se busca obtener el modelo f_{Λ}^* que minimice la función de pérdida \mathcal{L} asumiendo un conjunto de hiperparámetros Λ fijos. En la sección 2.3 se discutirán diversos algoritmos que permitan estimar el modelo \hat{f}_{Λ} tal que:

$$\hat{f}_{\Lambda} \approx f_{\Lambda}^* = \arg \min_{f_{\Lambda}} \mathcal{L}(y, f_{\Lambda}(\mathbf{x})), \quad (\mathbf{x}, y) \in \mathcal{D}_{\text{Train}}. \quad (29)$$

- \mathcal{D}_{Val} : Se busca obtener la configuración de hiperparámetros Λ que minimice alguna métrica de desempeño, μ , dado el modelo ajustado con los datos de entrenamiento $\mathcal{D}_{\text{Train}}$, es decir:

$$\hat{\Lambda} \approx \Lambda^* = \arg \min_{\{\lambda \in \Lambda_i\}_{i=1,\dots,l}} \mu_{\hat{f}_{\Lambda}}(\mathcal{D}_{\text{Val}}). \quad (30)$$

Una vez elegido $\hat{\Lambda}$, se vuelve a resolver (29) con $\hat{\Lambda}$ y todo el conjunto $\mathcal{D}_{\text{Train}}^*$.

Para mayor detalle sobre el proceso de calibración de hiperparámetros, en general, se puede consultar [Bartz et al. \(2023\)](#).

- $\mathcal{D}_{\text{Test}}$: Una vez completados los procesos de ajuste y *tuneo*, se procede a evaluar el modelo ajustado $\hat{f}_{\hat{\Lambda}}$ utilizando alguna métrica, la cual se denota como μ . La descripción de las métricas empleadas para evaluar los modelos ajustados en el capítulo 3 serán revisadas en la siguiente sección.

2.2.1 Métricas de desempeño

En la sección 2.2 se mencionó la necesidad de someter un algoritmo de aprendizaje \mathcal{A}_{Λ} a una evaluación en términos de su poder predictivo. Esto implica evaluar el modelo ajustado $\hat{f}_{\hat{\Lambda}}$, resultado del algoritmo \mathcal{A}_{Λ} , usando nuevas observaciones. En particular, el conjunto $\mathcal{D}_{\text{Test}}$ es utilizado para simular o emular nuevas observaciones, aprovechando los siguientes aspectos:

- El conjunto $\mathcal{D}_{\text{Test}}$ no es utilizado en el proceso de aprendizaje para definir el modelo ajustado $\hat{f}_{\hat{\Lambda}}$.
- Cada observación del conjunto $\mathcal{D}_{\text{Test}}$ tiene asociada una etiqueta y .

Con base en esto último, es común utilizar la matriz de confusión para la definición de algunas métricas de desempeño. La matriz de confusión para k clases, denotada por $\mathcal{M}_{\text{Test}}$, se define de la siguiente manera:

$$\mathcal{M}_{\text{Test}} = \begin{pmatrix} C_{1\hat{1}} & C_{1\hat{2}} & \cdots & C_{1\hat{k}} \\ C_{2\hat{1}} & C_{2\hat{2}} & \cdots & C_{2\hat{k}} \\ \vdots & \vdots & \ddots & \vdots \\ C_{k\hat{1}} & C_{k\hat{2}} & \cdots & C_{k\hat{k}} \end{pmatrix}, \quad (31)$$

donde

$$C_{ij} = \sum_{(x,y) \in \mathcal{D}_{\text{Test}}} I(y = C_i, \hat{f}(x) = C_j), \quad (32)$$

con $I(\cdot)$ la función indicadora.

La diagonal de una matriz de confusión es de gran importancia, ya que representa las predicciones correctas realizadas por un modelo de clasificación. Cada elemento de la diagonal corresponde a la cantidad de casos en los que el modelo ha predicho correctamente la clase de una observación específica. Es decir, si consideramos una matriz de confusión para un problema de clasificación con k clases, el elemento C_{ii} en la diagonal indica el número de observaciones que pertenecen a la clase i y que han sido clasificadas correctamente como clase i por el modelo. Una diagonal con valores altos y elementos fuera de la diagonal bajos sugiere un modelo que está funcionando bien, ya que está realizando la mayoría de las predicciones correctamente. Por lo tanto, el análisis de la diagonal de la matriz de confusión es crucial para evaluar la efectividad de un modelo de clasificación.

Con base en los valores que componen la matriz de confusión $\mathcal{M}_{\text{Test}}$, es posible definir diversas métricas para medir el desempeño del modelo ajustado \hat{f}_{Λ} . Las métricas que serán utilizadas en el caso práctico del capítulo 3 son las siguientes:

- Tasa de Clasificación Correcta de la clase C_l (TCCC $_l$): Evalúa el rendimiento del modelo para clasificar correctamente la clase C_l , brindando una medida más detallada del desempeño del modelo para dicha clase en particular. En este caso μ se define como:

$$\mu = \frac{C_{l\hat{l}}}{\sum_{i=1}^k C_{i\hat{i}}}, \quad l \in \{1, \dots, k\}. \quad (33)$$

- Tasa de Clasificación Correcta Global (TCCG, Exactitud o *Accuracy*): Evalúa el rendimiento del modelo para clasificar correctamente de manera general, indicando la proporción de predicciones correctas. Aquí μ se define como:

$$\mu = \frac{\sum_{i=1}^k C_{i\hat{i}}}{\sum_{i=1}^k \sum_{j=1}^k C_{ij}}. \quad (34)$$

Es importante mencionar que existen otras métricas, además de las descritas en las expresiones (33) y (34), que pueden ser utilizadas para cuantificar el desempeño de un modelo ajustado. Para obtener más detalles sobre este tipo de métricas alternativas, se puede consultar [Grandini et al. \(2020\)](#).

2.2.2 Repeated $u\text{-}v$ Fold Cross Validation

Una vez especificada la métrica que se utilizará para evaluar el modelo ajustado \hat{f}_Λ es posible definir el esquema con el cual se estimará el poder predictivo, denotado por τ , del algoritmo \mathcal{A}_Λ .

Una técnica comúnmente utilizada, es la que se ilustra en las Figuras 6 y 7, en la cual se realiza una única división del conjunto de datos \mathcal{D} . Sin embargo, este enfoque puede resultar inadecuado para estimar con precisión el poder predictivo del algoritmo \mathcal{A}_Λ . Esto debido a que una sola proporción de datos reservada para la evaluación podría no capturar completamente las características relevantes de nuevas observaciones. Dado lo anterior, es común emplear técnicas como la validación cruzada (*cross validation*), la cual permite una estimación más robusta del poder predictivo al considerar múltiples divisiones del conjunto \mathcal{D} .

La técnica de validación cruzada consiste en formar una partición de v subconjuntos (*folds*) del conjunto \mathcal{D} para luego entrenar y evaluar el modelo v veces, utilizando diferentes combinaciones de la partición como datos de entrenamiento y evaluación como se describe en el Algoritmo 3, donde, en su caso, el tuneo se realiza dentro del paso 5 considerando que $\mathcal{D}_{\text{Train}}$ se reemplaza por $\mathcal{D}_{\text{Train}}^*$ como en la Figura 6. Cabe mencionar que el *tuneo* también podría usar validación cruzada.

```

input :
    ■ Datos:  $\mathcal{D}$ 
    ■ Algoritmo:  $\mathcal{A}$ 
    ■ Métrica:  $\mu$ 

output:
    ■ Estimación del poder predictivo:  $\hat{\tau}$ 

```

1 *Partición del conjunto \mathcal{D} en v subconjuntos:*

$$\mathcal{D} = \bigcup_{i=1}^v \mathcal{D}^{[i]}, \quad \mathcal{D}^{[l]} \cap \mathcal{D}^{[m]} = \emptyset \forall l \neq m.$$

```

2 for  $i \leftarrow 1$  to  $v$  do
3    $\mathcal{D}_{\text{Test}} \leftarrow \mathcal{D}^{[i]}$ ;
4    $\mathcal{D}_{\text{Train}} \leftarrow \bigcup_{j \neq i} \mathcal{D}^{[j]}$ ;
5    $\hat{f}_i \leftarrow \mathcal{A}(\mathcal{D}_{\text{Train}})$ ;
6    $\mu_i \leftarrow \mu_{\hat{f}_i}(\mathcal{D}_{\text{Test}})$ ;
7 end
8 Estimación del poder predictivo  $\hat{\tau}$ :

```

$$\hat{\tau} = \frac{1}{v} \sum_{i=1}^v \mu_i.$$

Algoritmo 3: v Fold Cross Validation

Para dar una mayor robustez a la estimación del poder predictivo, se suele repetir u veces el proceso descrito en el algoritmo 3. Este procedimiento recibe el nombre de *Repeated u - v Fold Cross Validation*, y es descrito en el algoritmo 4, en este caso el tuneo se realiza dentro del paso 6.

```

input :
    ■ Datos:  $\mathcal{D}$ 
    ■ Algoritmo:  $\mathcal{A}$ 
    ■ Métrica:  $\mu$ 

output:
    ■ Estimación del poder predictivo:  $\hat{\tau}$ 

1 for  $i \leftarrow 1$  to  $u$  do
2   | Partición del conjunto  $\mathcal{D}$  en  $v$  subconjuntos:
3     |  $\mathcal{D} = \bigcup_{j=1}^v \mathcal{D}^{[i,j]}, \quad \mathcal{D}^{[i,l]} \cap \mathcal{D}^{[i,m]} = \emptyset \forall l \neq m.$ 
4     | for  $j \leftarrow 1$  to  $v$  do
5       |  $\mathcal{D}_{\text{Test}} \leftarrow \mathcal{D}^{[i,j]};$ 
6       |  $\mathcal{D}_{\text{Train}} \leftarrow \bigcup_{k \neq j} \mathcal{D}^{[i,k]};$ 
7       |  $\hat{f}_{ij} \leftarrow \mathcal{A}(\mathcal{D}_{\text{Train}});$ 
8       |  $\mu_{ij} \leftarrow \mu_{\hat{f}_{ij}}(\mathcal{D}_{\text{Test}});$ 
9     | end
10   | end
11   | Estimación del poder predictivo  $\hat{\tau}$ :
12     | 
$$\hat{\tau} = \frac{1}{u} \sum_{i=1}^u \frac{1}{v} \sum_{j=1}^v \mu_{ij}.$$


```

Algoritmo 4: Repeated u - v Fold Cross Validation

En este trabajo se usará - para la evaluación y en los casos en donde se realice el tuneo -.

2.3 ALGORITMOS DE CLASIFICACIÓN

En esta sección se presentan los fundamentos teóricos de los modelos que serán implementados en el capítulo 3. Se proporciona una descripción general de cada modelo, resaltando sus principios teóricos esenciales y los hiperparámetros más relevantes empleados en su implementación. Además se mencionan los paquetes en R que serán utilizados para su implementación.

Los modelos considerados son:

- Análisis de discriminante lineal.

- Análisis de discriminante cuadrático.
- El uso de los modelos gráficos gaussianos en clasificación.
- Regresión logística.
- Regresión logística con penalización *ElasticNet*.
- Máquina de soporte vectorial.
- Bosques aleatorios.

2.3.1 Análisis de discriminante y el uso de los modelos gráficos gaussianos no dirigidos

En los modelos de análisis de discriminante se supone un supuesto distribucional sobre las observaciones x_i y sobre las etiquetas y_1, \dots, y_n . En general el objetivo es estimar la siguiente probabilidad:

$$P(y = C_j | x), \quad j \in \{1, \dots, k\}. \quad (35)$$

En lo que sigue, se denota a la función de densidad o probabilidad como $p(\cdot)$ y la función de densidad condicional como $p(\cdot | \cdot)$. De acuerdo con los axiomas básicos de probabilidad, el teorema de Bayes y el de probabilidad total, las siguientes igualdades son equivalentes:

$$\begin{aligned} P(y = C_j | x) &= \frac{p(y = C_j, x)}{p(x)} \\ &= \frac{p(x|y = C_j)p(y = C_j)}{p(x)} \\ &= \frac{p(x|y = C_j)p(y = C_j)}{\sum_{l=1}^k p(x|y = C_l)p(y = C_l)}. \end{aligned} \quad (36)$$

Las funciones de probabilidad descritas en las expresiones (35) y (36) a menudo reciben los siguientes nombres:

- $p(y = C_j)$: Probabilidad *a priori*.
- $P(y = C_j | x)$: Probabilidad *a posteriori*.

De esta manera, el objetivo será estimar la probabilidad descrita en la expresión (35) a través de la estimación de las probabilidades descritas en la expresión (36). A este proceso se le conoce como *generative learning*.

Por lo tanto, el modelo ajustado \hat{f} será la etiqueta C_j para la cual la probabilidad *a posteriori* sea máxima. Es decir:

$$\hat{f}(x) = \arg \max_{C_j} \left\{ \hat{P}(y = C_j | x) \right\}. \quad (37)$$

Es importante notar que:

$$\arg \max_{C_j} \{P(y = C_j | \mathbf{x})\} = \arg \max_{C_j} \{\ln [P(y = C_j | \mathbf{x})]\}. \quad (38)$$

Además, el denominador dentro de la expresión (36) será siempre el mismo, sin importar el valor C_j para el cual se estime la probabilidad *a posteriori*. Por lo tanto:

$$\arg \max_{C_j} \{P(y = C_j | \mathbf{x})\} = \arg \max_{C_j} \{\ln [p(y = C_j)] + \ln [p(\mathbf{x}|y = C_j)]\}. \quad (39)$$

Sea

$$\delta_{C_j}(\mathbf{x}) = \ln [p(y = C_j)] + \ln [p(\mathbf{x}|y = C_j)]. \quad (40)$$

A la función δ_{C_j} descrita en la expresión (40) se le conoce como función discriminante y los algoritmos descritos en esta sección buscarán estimar un valor de dicha función. Por lo tanto, el modelo ajustado descrito en la expresión (37) es equivalente a:

$$\hat{f}(\mathbf{x}) = \arg \max_{C_j} \{\hat{\delta}_{C_j}(\mathbf{x})\}. \quad (41)$$

Análisis de discriminante lineal

Para este primer modelo, basado en la función de discriminante, se asume el siguiente supuesto distribucional:

$$\mathbf{x} | \{y = C_j\} \sim N(\boldsymbol{\mu}_{C_j}, \boldsymbol{\Sigma}), \quad j \in \{1, \dots, k\}. \quad (42)$$

Donde $\boldsymbol{\mu}_{C_j} \in \mathbb{R}^p$ es conocido como el vector de medias de la clase C_j , y $\boldsymbol{\Sigma}$ es una matriz de dimensión $p \times p$ llamada matriz de covarianza. Es importante notar que $\boldsymbol{\Sigma}$ no depende del valor que tome la variable y ; es decir, se asume el mismo valor de $\boldsymbol{\Sigma}$ para cualquier valor C_j , con $j \in \{1, \dots, k\}$. De esta forma:

$$p(\mathbf{x}|y = C_j) = \frac{1}{(2\pi)^{\frac{p}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_{C_j})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{C_j}) \right\}. \quad (43)$$

Sustituyendo (43) en la función discriminante descrita en la expresión (40) se obtiene:

$$\delta_{C_j}(\mathbf{x}) = \ln [p(y = C_j)] - \frac{p}{2} \ln(2\pi) - \frac{1}{2} \ln |\boldsymbol{\Sigma}| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_{C_j})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{C_j}). \quad (44)$$

Notemos que $-\frac{p}{2} \ln(2\pi)$ es una constante con respecto a C_j , por lo tanto es posible prescindir de ella. De esta forma, la función discriminante puede quedar expresada de la siguiente manera:

$$\delta_{C_j}(\mathbf{x}) = \ln [p(y = C_j)] - \frac{1}{2} \ln |\Sigma| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_{C_j})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{C_j}). \quad (45)$$

Es posible analizar la región de decisión entre dos clases donde las funciones discriminantes son exactamente iguales. Sea $l \neq m$, entonces $\delta_{C_l}(x_i) = \delta_{C_m}(x_i)$ ocurre si y solo si:

$$\ln \left[\frac{p(y = C_l)}{p(y = C_m)} \right] - \frac{1}{2} (\boldsymbol{\mu}_{C_l} - \boldsymbol{\mu}_{C_m})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_{C_l} - \boldsymbol{\mu}_{C_m}) + \mathbf{x}^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_{C_l} - \boldsymbol{\mu}_{C_m}) = 0. \quad (46)$$

Cuando se utilizan los estimadores de máxima verosimilitud (MLE), siguiendo la regla *plug-in*, para estimar el valor de la función discriminante descrita en la expresión (44), el método se denomina LDA (*linear discriminant analysis*). En este caso, los estimadores MLE son:

$$\hat{p}(y = C_j) = \frac{n_{C_j}}{n}, \quad n_{C_j} = \sum_{i=1}^n I(y_i = C_j), \quad (47)$$

$$\hat{\boldsymbol{\mu}}_{C_j} = \frac{1}{n_{C_j}} \sum_{\{i:y_i=C_j\}} \mathbf{x}_i. \quad (48)$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{j=1}^k \sum_{\{i:y_i=C_j\}} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{C_j}) (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{C_j})^T. \quad (49)$$

Cabe mencionar que LDA también se puede obtener sin un supuesto distribucional, planteando el problema como uno de optimización en donde se busca la proyección que separa las medias de los grupos.

Análisis de Discriminante Cuadrático

A diferencia del modelo anterior, para este modelo se asume el siguiente supuesto distribucional:

$$\mathbf{x} | \{y = C_j\} \sim N(\boldsymbol{\mu}_{C_j}, \boldsymbol{\Sigma}_{C_j}), \quad j \in \{1, \dots, k\}. \quad (50)$$

Es importante notar que en este caso $\boldsymbol{\Sigma}_{C_j}$ depende del valor que tome la variable y . De esta forma:

$$p(\mathbf{x}|y = C_j) = \frac{1}{(2\pi)^{\frac{p}{2}} |\boldsymbol{\Sigma}_{C_j}|^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_{C_j})^T \boldsymbol{\Sigma}_{C_j}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{C_j}) \right\}. \quad (51)$$

Similar al proceso descrito en la sección anterior, la función de discriminante asociada a este modelo puede quedar expresada de la siguiente manera:

$$\delta_{C_j}(\mathbf{x}) = \ln [p(y = C_j)] - \frac{1}{2} \ln (\boldsymbol{\Sigma}_{C_j}) - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_{C_j})^T \boldsymbol{\Sigma}_{C_j}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{C_j}). \quad (52)$$

Sea $l \neq m$, entonces $\delta_{C_l}(x) = \delta_{C_m}(x)$ ocurre si y solo si:

$$\ln \left[\frac{p(y = C_l)}{p(y = C_m)} \right] + \frac{1}{2} \ln \frac{|\Sigma_{C_m}|}{|\Sigma_{C_l}|} + \frac{1}{2} (x - \mu_{C_m})^T \Sigma_{C_m}^{-1} (x - \mu_{C_m}) - \frac{1}{2} (x - \mu_{C_l})^T \Sigma_{C_l}^{-1} (x - \mu_{C_l}) = 0. \quad (53)$$

Es importante notar que, en este caso la función discriminante contiene términos de la forma $x^T A x$, lo que introduce términos no lineales en x . Esto permite capturar relaciones más complejas entre las variables que pueden ser diferentes entre las clases, a diferencia de función discriminante en LDA, donde la frontera de decisión es siempre lineal.

En este caso, al utilizar los estimadores MLE, el método se denomina QDA (*quadratic discriminant analysis*). Estos estimadores son los mismos que en LDA para la probabilidad a priori y el vector de medias de la clase j , con excepción de la matriz de covarianza, que en este caso se determina de la siguiente forma:

$$\hat{\Sigma}_{C_j} = \frac{1}{n_{C_j}} \sum_{\{i:y_i=C_j\}} (x_i - \hat{\mu}_{C_j}) (x_i - \hat{\mu}_{C_j})^T. \quad (54)$$

Para la implementación de los modelos LDA y QDA se utilizará el paquete MASS [Venables and Ripley, 2002].

Modelo UGGM-QDA

Como alternativa a los estimadores MLE, usados para definir las expresiones en (52), en este trabajo se hace el supuesto donde $x | \{y = C_j\}$ sigue un modelo gráfico gaussiano y se usa el algoritmo *glasso* para la estimación de la estructura y parámetros, los que, siguiendo la regla *plug-in*, se sustituyen en la función discriminante de acuerdo con la expresión (52), para obtener un modelo de clasificación que denominaremos UGGM-QDA. Este modelo se describe en el Algoritmo 5 y se ha propuesto en [Chen, 2022].

Es decir, *glasso* se utiliza para estimar la estructura de dependencia entre variables, representada por la matriz de concentración, donde algunas entradas son cero, lo que implica una estructura de independencia condicional en un modelo gráfico gaussiano. Esta estimación introduce una restricción estructural en la matriz de covarianza que no está presente en el método QDA.

De esta forma, el modelo resultante asume un modelo gráfico implícito. Luego, la clasificación sigue un enfoque similar al método QDA, ya que cada clase tiene su propia matriz de covarianza, obtenida mediante *glasso* en lugar de una estimación sin restricciones.

Ahora bien, para determinar el valor de λ a utilizar en el Algoritmo 5, se siguió la metodología descrita en el Algoritmo 6, donde se ajusta λ a través de un proceso de validación cruzada utilizando una malla de valores de λ y un conjunto de entrenamiento $\mathcal{D}_{\text{Train}}^*$.

En el Algoritmo 6 el ajuste del hiperparámetro λ se realiza mediante un proceso de validación cruzada para optimizar el rendimiento del modelo. Este procedimiento implica dividir el conjunto de datos en varios subconjuntos (*folds*) y evaluar el modelo en diferentes configuraciones de λ . Para cada valor de λ , el modelo se entrena en un subconjunto de los datos y se valida

en otro, calculando una métrica de rendimiento, por ejemplo, utilizando *Accuracy* descrito en la expresión (34), en cada iteración. Este proceso se repite para todos los subconjuntos, y la métrica de rendimiento promedio se calcula para cada valor de λ .

Finalmente, se selecciona el valor de λ que maximiza la métrica de rendimiento promedio, lo que garantiza que el modelo esté optimizado y se espera que generalice bien a datos no vistos.

```

input :
    ■ Datos:  $\mathcal{D}$ 
    ■ Lambda:  $\lambda$ 
    ■ Nueva observación:  $x^*$ 

output:
    ■ Clase predicha:  $\hat{y}$ 

1 for  $i \leftarrow 1$  to  $k$  do
2
3   for  $j \leftarrow 1$  to  $p$  do
4
5   end
6
7    $n_{C_i} \leftarrow \sum_{l=1}^n I(y_l = C_i)$ 
8    $\hat{\pi}_{C_i} \leftarrow \frac{n_{C_i}}{n}$ 
9
10   $\hat{\mu}_{ij} \leftarrow \frac{1}{n_{C_i}} \sum_{\{l:y_l=C_i\}} x_{lj}$ 
11
12   $\hat{\mu}_{C_i} \leftarrow (\hat{\mu}_{i1}, \dots, \hat{\mu}_{ip})$ 
13   $\hat{\Sigma}_{C_i} \leftarrow \frac{1}{n_{C_i}} \sum_{\{l:y_l=C_i\}} (x_l - \hat{\mu}_{C_i})(x_l - \hat{\mu}_{C_i})^T$ 
14   $\hat{\Theta}_{C_i} \leftarrow \text{glasso}(\hat{\Sigma}_{C_i}, \lambda)$ 
15   $\hat{\delta}_{C_i}(x^*) \leftarrow \ln(\hat{\pi}_{C_i}) - \frac{1}{2} \ln(|\hat{\Theta}_{C_i}|) - \frac{1}{2} (x^* - \hat{\mu}_{C_i})^T \hat{\Theta}_{C_i} (x^* - \hat{\mu}_{C_i})$ 
16
17 end
18
19   $\hat{y} \leftarrow \arg \max_{C_i \in \{C_1, \dots, C_k\}} \{\hat{\delta}_{C_i}(x^*)\}.$ 

```

Algoritmo 5: Regla de clasificación dado un valor de λ del modelo UGGM-QDA.

El modelo estima los parámetros de la distribución condicional para cada clase, ajustando la matriz de precisión mediante el método *glasso*, y asigna la nueva observación a la clase que maximiza la función discriminante.

input :

- Datos de entrenamiento: $\mathcal{D}_{\text{Train}^*}$
- Malla de valores λ : $\{\lambda_1, \dots, \lambda_q\}$
- Número de subconjuntos (*folds*): v

output:

- Valor de λ óptimo: $\hat{\lambda}$

1 Partición del conjunto \mathcal{D} en v subconjuntos:

$$\mathcal{D}_{\text{Train}^*} = \bigcup_{i=1}^v \mathcal{D}^{[i]}, \quad \mathcal{D}^{[l]} \cap \mathcal{D}^{[m]} = \emptyset \forall l \neq m.$$

2 **for** $\lambda_j \in \{\lambda_1, \dots, \lambda_q\}$ **do**

3 **for** $i \leftarrow 1$ **to** v **do**

4 $\mathcal{D}_{\text{Val}} \leftarrow \mathcal{D}^{[i]}$;

5 $\mathcal{D}_{\text{Train}} \leftarrow \bigcup_{j \neq i} \mathcal{D}^{[j]}$;

6 $\hat{f}_{\lambda_j}^{[i]} \leftarrow \text{UGGM-QDA}(\mathcal{D}_{\text{Train}}, \lambda_j, \mathcal{D}_{\text{Val}})$;

7 $\mu_{\lambda_j}^{[i]} \leftarrow \mu_{\hat{f}_{\lambda_j}^{[i]}}(\mathcal{D}_{\text{Val}})$;

8 **end**

9 Estimación del poder predictivo para λ_j :

$$\hat{\tau}_{\lambda_j} = \frac{1}{v} \sum_{i=1}^v \mu_{\lambda_j}^{[i]}.$$

10 **end**

11 Selección del valor λ_j óptimo:

$$\hat{\lambda} \leftarrow \arg \max_{\lambda_j \in \{\lambda_1, \dots, \lambda_q\}} \{\hat{\tau}_{\lambda_j}\}$$

Algoritmo 6: Tuneo del hiperparámetro λ mediante un proceso de validación cruzada con v subconjuntos (*folds*).

Es decir, el Algoritmo 6 se usa en el Algoritmo 5 considerando a $\mathcal{D} = \mathcal{D}_{\text{Train}}$ para el tuneo, una vez elegido $\hat{\lambda}$ se vuelve a ejecutar el Algoritmo 5 con $\mathcal{D} = \mathcal{D}_{\text{Train}^*}$. Este último es el método que será evaluado.

Ejemplo con datos simulados

Para evaluar el desempeño del modelo UGGM-QDA en comparación con el modelo QDA en términos de poder predictivo, se llevaron a cabo simulaciones considerando distintos tamaños de muestra y número de variables. En total, se generaron 16 conjuntos de datos, cada uno compuesto por tres clases (Clase 1, Clase 2 y Clase 3). La metodología de simulación siguió

el Algoritmo 1 descrito en la sección 2.1.4, y las estructuras de las gráficas utilizadas pueden consultarse en el Anexo A.

Cada simulación consideró tamaños de muestra de 50, 100, 500 y 1000 observaciones por clase, combinados con distintos números de variables: 20, 25, 30 y 35. Esto resultó en un total de 16 configuraciones distintas. Para cada uno de estos conjuntos, se generó un conjunto de evaluación $\mathcal{D}_{\text{Test}}$, compuesto por 10000 observaciones por clase, asegurando así una evaluación robusta del desempeño predictivo.

En la Figura 8 se visualiza el proceso de *tuneo* para cada uno de los escenarios simulados. Este proceso se llevó a cabo siguiendo el Algoritmo 6, donde se fijó el número de subconjuntos (*folds*) en 10 y se exploró una malla de valores de λ en el intervalo $[0, 0.1]$ con incrementos de 0.001, evaluando un total de 101 valores. Además, se calculó la variabilidad de la Tasa de Clasificación Correcta Global (TCCG) obtenida en cada iteración de la validación cruzada, lo que permitió determinar los valores óptimos de λ en cada escenario.

A partir de los visualizado en la Figura 8 se observa que los valores óptimos de λ tienden a ser mayores en los casos donde el tamaño de muestra es reducido y el número de variables es elevado (por ejemplo, el cuadrante $n = 50, p = 35$), lo que indica una mayor necesidad de penalización en estos escenarios. Esto tiene sentido, ya que al contar con menos observaciones y más parámetros, el modelo podría ajustarse demasiado a las particularidades del conjunto de entrenamiento, reduciendo su capacidad para hacer predicciones precisas en nuevas observaciones.

Además, se nota que las desviaciones estándar del poder predictivo son mayores en los casos donde el tamaño de muestra es bajo (cuadrantes de la primera fila $n = 50$), lo que sugiere una mayor variabilidad en el rendimiento del modelo. Esto también es esperable, dado que un modelo construido con pocas observaciones y muchas variables tiende a ser menos estable, lo que implica un desempeño más sensible a la variabilidad en los datos de entrenamiento.

Por otra parte, en la Figura 9 se presenta la comparación del poder predictivo, con base en la Tasa de Clasificación Correcta Global (TCCG), entre los modelos UGGM-QDA, QDA y las predicciones teóricas. Estas últimas se calcularon a partir de la estructura de la matriz de covarianza correspondiente a cada clase en la simulación. En particular, se consideraron distribuciones normales multivariadas con medias centradas en cero y matrices de covarianza generadas mediante el Algoritmo 1. Luego, para obtener las predicciones teóricas, se aplicó la regla de decisión del modelo QDA, basada en la función discriminante definida en la expresión (52), reemplazando directamente las matrices de covarianza de las que se simulan los datos. De este modo, la regla de decisión se construyó utilizando los parámetros μ y Σ empleados en la generación de los datos, proporcionando una referencia teórica para evaluar el desempeño de los modelos entrenados.

En el Cuadro 1, se resumen los métodos utilizados para estimar la matriz de concentración $\Theta = \Sigma^{-1}$ en cada modelo.

A partir de los visualizado en la Figura 9, se pueden destacar los siguientes puntos:

- Para todas las combinaciones de tamaño de muestra y número de variables, el modelo UGGM-QDA obtuvo un mejor desempeño en comparación con el modelo QDA.

- Conforme aumenta el tamaño de muestra, el poder predictivo de ambos modelos tiende a acercarse a su valor teórico.
- El modelo UGGM-QDA mostró una mayor precisión respecto al modelo QDA especialmente en los escenarios donde el número de variables es grande y el tamaño de muestra por clase es bajo.

Modelo	Método de estimación de $\Theta = \Sigma^{-1}$
QDA	Máxima Verosimilitud: $\hat{\Theta} = S^{-1}$.
UGGM-QDA	Algoritmo <i>glasso</i> : $\hat{\Theta} = \hat{\Theta}_{\text{glasso}}$.
Teórico	Se usa Θ directamente.

Cuadro 1: Métodos de estimación de la matriz de concentración Θ en cada modelo.

Tuneo del hiperparámetro λ

Modelo UGGM-QDA

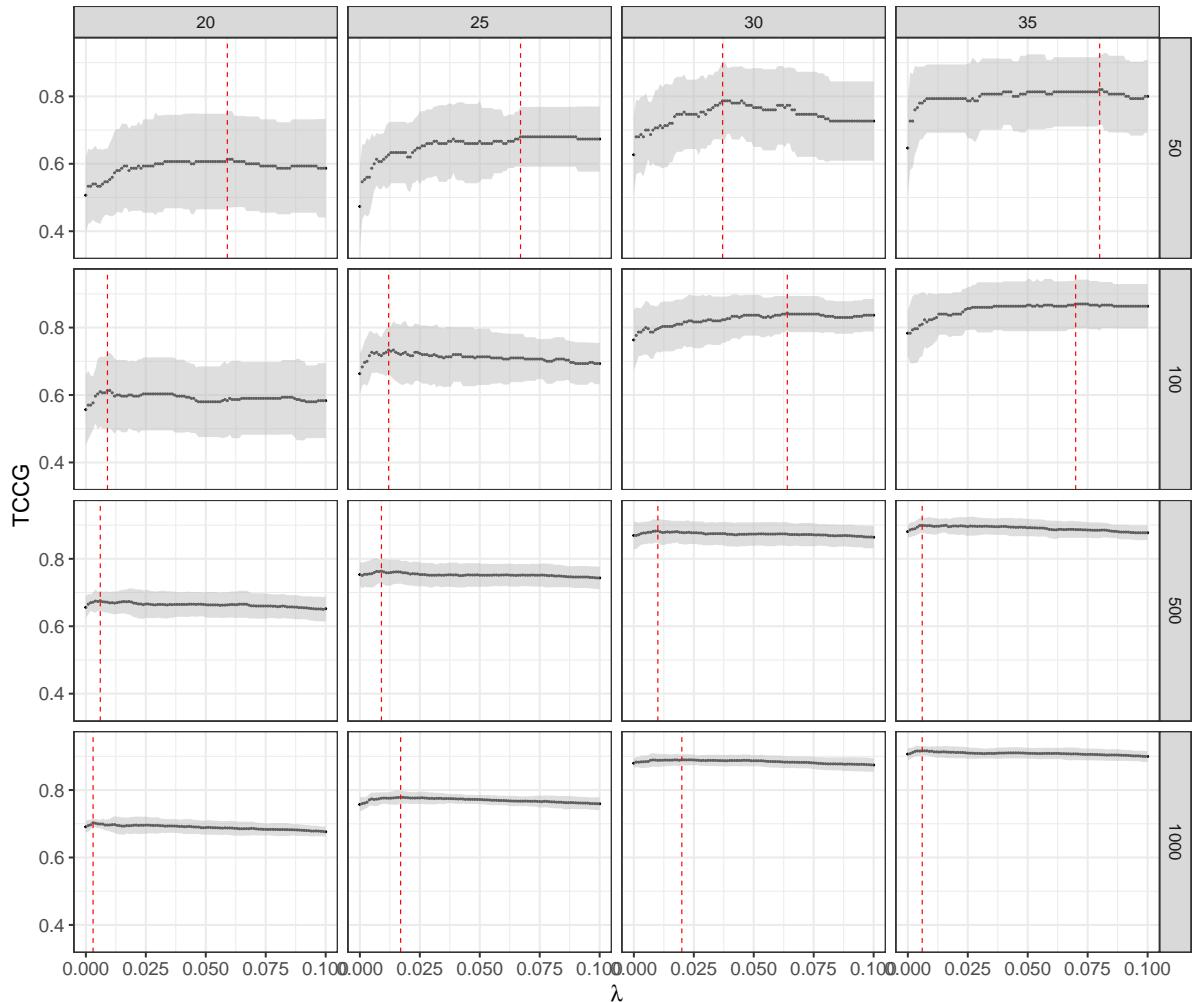


Figura 8: Tuneo del hiperparámetro λ mediante el Algoritmo 6: Cada panel representa una combinación del número de variables p (vertical) y el tamaño de muestra por clase n (horizontal), la línea roja punteada representa el valor de λ para el cual se alcanza el valor más alto de la Tasa de Clasificación Correcta Global (TCCG) y el área sombreada la desviación estándar de cada iteración de los 10 subconjunto (*folds*) usados en el proceso de validación cruzada.

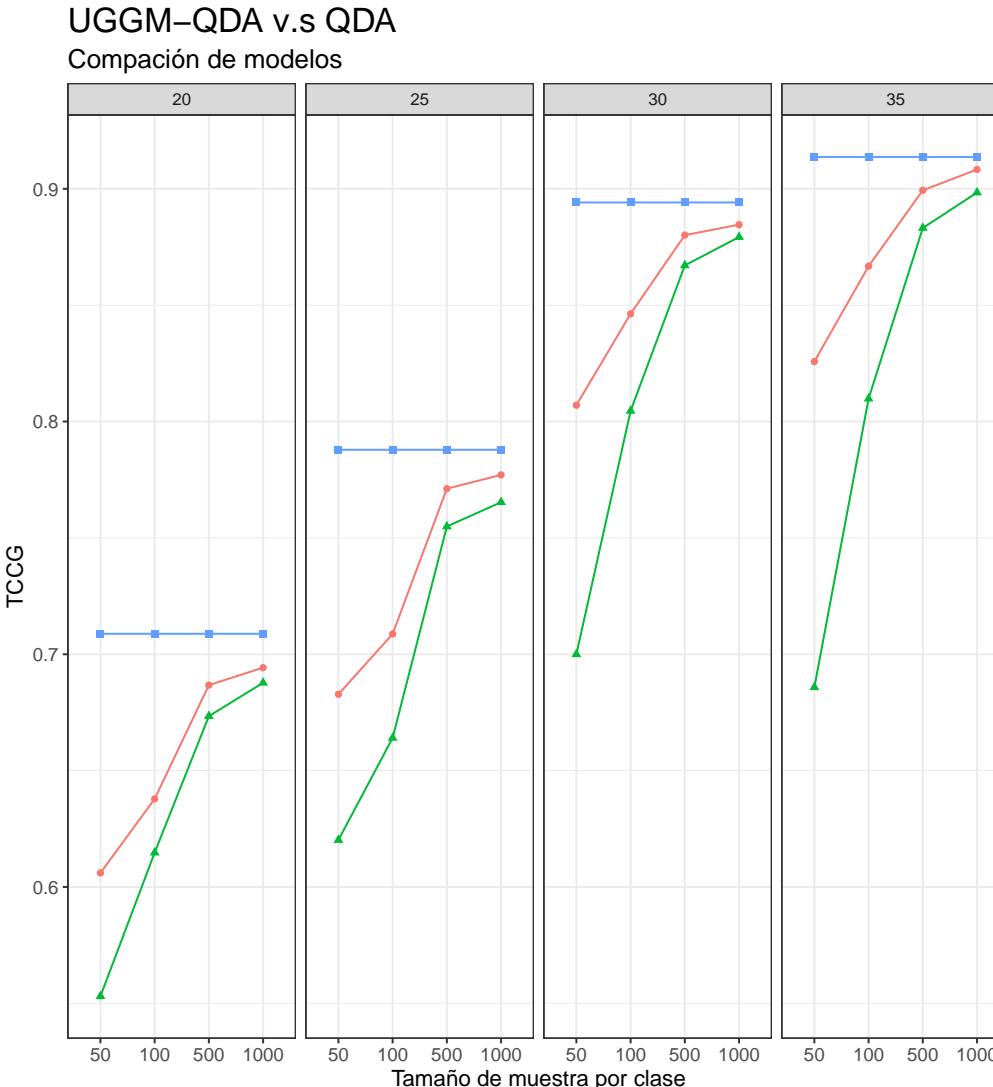


Figura 9: Comparación del poder predictivo entre los modelos UGGM-QDA, QDA y las predicciones teóricas. Cada panel muestra el cambio de la Tasa de Clasificación Correcta Global (TCCG) en función del tamaño de muestra (eje x) y el número de variables (columnas). Se comparan los modelos QDA, UGGM-QDA y las predicciones teóricas, permitiendo analizar su desempeño en distintos escenarios y evaluar cómo la penalización implementada en el algoritmo *glasso* influye en la precisión del modelo.

2.3.2 Otros métodos

Regresión logística

La regresión logística forma parte de un conjunto de modelos llamados *modelos lineales generalizados*. En un modelo GLM (*Generalized Linear Model*) se asume que las etiquetas y_1, \dots, y_n son variables aleatorias independientes de la familia exponencial cuya función de densidad o de probabilidad, en su forma natural, es de la siguiente forma:

$$f(y_i; \theta_i, v_i / \varphi) = \exp \left\{ \frac{y_i \theta_i - \kappa(\theta_i)}{\varphi/v_i} + a(y_i; v_i / \varphi) \right\}, \quad i \in \{1, \dots, n\}, \quad (55)$$

donde

- $\kappa(\cdot)$ es una función doblemente diferenciable respecto a θ_i .
- $\theta_i = \theta(x_i) \in \mathbb{R}$ es un parámetro de localización llamado parámetro canónico.
- $v > 0$ es una constante positiva llamada peso.
- $\varphi > 0$ es un parámetro de dispersión o escala.
- $a(\cdot; \cdot)$ es una función de normalización que asegura que la función descrita en la expresión (55) corresponde a una función de densidad o de probabilidad.

De acuerdo con ([Agresti, 2015](#), sección 4), se puede verificar que:

$$\mathbb{E}_{\theta_i}[y_i] = \mathbb{E}[y_i; x_i] = \frac{\partial}{\partial \theta_i} \kappa(\theta_i). \quad (56)$$

En un modelo GLM generalmente se supone que los datos de entrada \mathcal{X} incluyen una variable $x_{i0} = 1$ denominada intercepto, es decir, $x_i = (x_{i0}, x_{i1}, \dots, x_{ip})$. La representación matricial de estos datos se denomina como matriz diseño y se denota como \mathcal{X}_0 . Es decir:

$$\mathcal{X}_0 = (x_1, \dots, x_n)^T = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}. \quad (57)$$

Además, se asume la existencia de un conjunto de parámetros $\beta = (\beta_0, \beta_1, \dots, \beta_p) \in \mathbb{R}^{p+1}$ y una función $g(\cdot)$ monótona y diferenciable, de tal forma que la expresión (56) se modela de manera lineal respecto a los parámetros β de la siguiente manera:

$$\begin{aligned} g(\mathbb{E}[y_i; x_i]) &= \sum_{j=0}^p \beta_j x_{ij} \\ \implies \mathbb{E}[y_i; x_i] &= g^{-1} \left(\sum_{j=0}^p \beta_j x_{ij} \right). \end{aligned} \quad (58)$$

En general, dentro de los modelos GLM se busca obtener un modelo ajustado minimizando el valor negativo de la función de log-verosimilitud $\ell_y(\beta)$ en términos de los parámetros $\beta = (\beta_0, \beta_1, \dots, \beta_p)$. Es decir:

$$\hat{\beta} = \arg \min_{\beta} \{-\ell_y(\beta)\}. \quad (59)$$

De esta forma, el modelo ajustado \hat{f} depende de:

$$\widehat{E}[y_i; \mathbf{x}_i] = g^{-1} \left(\sum_{j=0}^p \widehat{\beta}_j x_{ij} \right). \quad (60)$$

En particular, para un problema de clasificación binaria ($y_i \in \{C_1 = 1, C_2 = 0\}$) el modelo *GLM* utilizado en este trabajo es la *regresión logística*. Es decir:

- y_1, \dots, y_n siguen una distribución Bernoulli(π_i), con $\pi_i = P(y_i = C_1)$.
- $g(z) = \ln(\frac{z}{1-z})$.

En este caso, $f(y_i; \pi_i) = \pi_i^{I(y_i=C_1)} (1 - \pi_i)^{1-I(y_i=C_1)}$. Aplicando la función $g(z)$ a π_i se obtiene el parámetro canónico:

$$\theta_i = g(\pi) = \ln \left(\frac{\pi_i}{1 - \pi_i} \right). \quad (61)$$

De esta forma:

$$\pi_i = \frac{e^{\theta_i}}{1 + e^{\theta_i}}. \quad (62)$$

Por lo tanto,

$$\begin{aligned} f(y_i; \theta_i) &= \left(\frac{e^{\theta_i}}{1 + e^{\theta_i}} \right)^{I(y_i=C_1)} \left(\frac{1}{1 + e^{\theta_i}} \right)^{1-I(y_i=C_1)} \\ &= e^{I(y_i=C_1)\theta_i} (1 + e^{\theta_i})^{-1} \\ &= \exp \left\{ I(y_i = C_1) \theta_i - \ln(1 + e^{\theta_i}) \right\}. \end{aligned} \quad (63)$$

Concluyendo así, que la distribución Bernoulli pertenece a la familia exponencial, es decir, su función de probabilidad es de la forma descrita en la expresión (55), con $\kappa(\theta_i) = -\ln(1 + e^{\theta_i})$, $v = 1$ y $\varphi = 1$

Por otra parte, en este modelo se asume que:

$$\pi_i = P(y_i = C_1; \mathbf{x}_i) = E[y_i; \mathbf{x}_i] = \frac{\exp \left\{ \sum_{j=0}^p \beta_j x_{ij} \right\}}{1 + \exp \left\{ \sum_{j=0}^p \beta_j x_{ij} \right\}}. \quad (64)$$

De manera equivalente:

$$\ln \left(\frac{\pi_i}{1 - \pi_i} \right) = \ln \left[\frac{P(y_i = C_1; \mathbf{x}_i)}{P(y_i = C_2; \mathbf{x}_i)} \right] = \sum_{j=0}^p \beta_j x_{ij}. \quad (65)$$

Por lo tanto la probabilidad estimada es la siguiente:

$$\hat{\pi}_i = \widehat{P}(y_i = C_1; \mathbf{x}_i) = \frac{\exp\left\{\sum_{j=0}^p \hat{\beta}_j x_{ij}\right\}}{1 + \exp\left\{\sum_{j=0}^p \hat{\beta}_j x_{ij}\right\}}. \quad (66)$$

La estimación de los parámetros se obtiene siguiendo la expresión (59), en este caso:

$$\begin{aligned} \ell_y(\boldsymbol{\beta}) &= \ln \left(\prod_{i=1}^n \pi_i^{I(y_i=C_1)} (1 - \pi_i)^{1-I(y_i=C_1)} \right) \\ &= \sum_{i=1}^n I(y_i = C_1) \ln(\pi_i) + I(y_i = C_2) \ln(1 - \pi_i). \end{aligned} \quad (67)$$

Por lo tanto, se busca:

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left\{ - \sum_{i=1}^n I(y_i = C_1) \ln(\pi_i) + I(y_i = C_2) \ln(1 - \pi_i) \right\}. \quad (68)$$

Se pueden emplear métodos iterativos estándar para resolver el problema planteado en la expresión (68), por ejemplo el método de Newton-Raphson. Para una revisión detallada de estos métodos, se recomienda consultar ([Agresti, 2015](#), sección 5.4).

Es importante notar que el modelo *logístico* permite estimar las probabilidades de los eventos $\{y_i = C_1\}$ y $\{y_i = C_2\}$. Por lo tanto la regla ajustada es:

$$\hat{f}(\mathbf{x}_i) = \begin{cases} C_1, & \text{si } \widehat{P}(y_i = C_1; \mathbf{x}_i) \geq \alpha \\ C_2, & \text{si } \widehat{P}(y_i = C_1; \mathbf{x}_i) < \alpha \end{cases}. \quad (69)$$

Comúnmente $\alpha = \frac{1}{2}$, donde α es un punto de corte (*cut off*). Sin embargo, dependiendo del objetivo del modelo, α puede tomar cualquier valor entre 0 y 1. Por lo tanto α puede ser considerado un hiperparámetro que puede ser calibrado (*tuneado*).

Una de las extensiones del modelo *logístico* para casos donde el conjunto de etiquetas $\{C_1, \dots, C_k\}$ tiene más de dos elementos ($k > 2$) es la *regresión logística multinomial*. Un modelo *MLR* (*multinomial logistic regression*) se puede expresar de la siguiente manera:

$$\ln \left[\frac{P(y_i = C_j; \mathbf{x}_i)}{P(y_i = C_k; \mathbf{x}_i)} \right] = \sum_{l=0}^p \beta_{lj} x_{il}, \quad j \in \{1, \dots, k-1\}. \quad (70)$$

De manera equivalente:

$$P(y_i = C_j; \mathbf{x}_i) = \frac{\exp\left\{\sum_{l=0}^p \beta_{lj} x_{il}\right\}}{\sum_{m=1}^k \exp\left\{\sum_{l=0}^p \beta_{lm} x_{il}\right\}}, \quad j \in \{1, \dots, k\}, \quad (71)$$

donde $\beta_{0k}, \beta_{1k}, \dots, \beta_{pk}$ son cero.

De esta forma, las probabilidades estimadas bajo el modelo corresponden a:

$$\hat{P}(y_i = C_j; \mathbf{x}_i) = \frac{\exp\left\{\sum_{l=0}^p \hat{\beta}_{lj} x_{il}\right\}}{\sum_{m=1}^k \exp\left\{\sum_{l=0}^p \hat{\beta}_{lm} x_{il}\right\}}, \quad j \in \{1, \dots, k\}. \quad (72)$$

La estimación de los parámetros se obtiene resolviendo el siguiente problema de minimización:

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left\{ - \sum_{i=1}^n \sum_{j=1}^k I(y_i = C_j) \ln [\hat{P}(y_i = C_j; \mathbf{x}_i)] \right\}, \quad (73)$$

con $P(y_i = C_j; \mathbf{x}_i)$ como en la expresión (71). Para mayor detalle sobre el planteamiento y los métodos de estimación de los parámetros del modelo MLR, se puede consultar [McCullagh and Nelder \(1989, sección 5\)](#).

Por lo tanto, la regla de máxima probabilidad es:

$$\hat{f}(\mathbf{x}_i) = \arg \max_{C_j} \{\hat{P}(y_i = C_j)\}. \quad (74)$$

Para la implementación del modelo *MLR* se utilizará el paquete `glmnet` [[Friedman et al., 2010](#)].

Métodos de regularización

Los métodos de regularización son técnicas utilizadas en la implementación de modelos GLM con el objetivo de reducir el número de parámetros que influyen en el modelo ajustado. Estas técnicas son especialmente útiles cuando se trabaja con un gran número de variables o en presencia de alta correlación entre ellas.

Dos de los métodos de regularización más conocidos son la penalización L₁ (*Lasso*) [[Tibshirani, 1996](#)] y la penalización L₂ (*Ridge*) [[Hoerl and Kennard, 2006](#)].

- *Lasso*: Agrega una penalización L₁ a la función de log-verosimilitud, lo que favorece la selección de variables al forzar algunos coeficientes a ser exactamente cero:

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg \min_{\boldsymbol{\beta}} \left\{ -\frac{1}{n} \ell_y(\boldsymbol{\beta}) + \lambda \sum_{j=1}^p |\beta_j| \right\}, \quad \text{con } \lambda \geq 0. \quad (75)$$

- *Ridge*: Agrega una penalización L₂ a la función de log-verosimilitud, lo que permite controlar la magnitud de los coeficientes sin forzarlos a ser exactamente cero:

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \arg \min_{\boldsymbol{\beta}} \left\{ -\frac{1}{n} \ell_y(\boldsymbol{\beta}) + \lambda \sum_{j=1}^p \beta_j^2 \right\}, \quad \text{con } \lambda \geq 0. \quad (76)$$

De acuerdo con [Tay et al. \(2023\)](#), la penalización L₁ y L₂ son introducidas en un modelo GLM penalizando la función de log-verosimilitud, donde el problema de optimización es:

$$\hat{\boldsymbol{\beta}}_{\text{ElasticNet}} = \arg \min_{\boldsymbol{\beta}} \left\{ -\frac{1}{n} \ell_y(\boldsymbol{\beta}) + \lambda P_\alpha(\boldsymbol{\beta}) \right\}, \quad (77)$$

donde $\lambda \geq 0$, $\alpha \in [0, 1]$ y

$$P_\alpha(\boldsymbol{\beta}) = \sum_{j=1}^p \left[\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j| \right]. \quad (78)$$

La función descrita en la expresión (78) es conocida como la penalización *ElasticNet* [[Zou and Hastie, 2005](#)] y es una combinación entre la penalización *lasso* ($\alpha = 1$) y la penalización *ridge* ($\alpha = 0$).

Es importante mencionar que el modelo MLR no pertenece a los modelos GLM, sin embargo, de acuerdo con [Friedman et al. \(2010\)](#), la penalización *ElasticNet* puede ser utilizada dentro del modelo MLR de la siguiente manera:

$$\hat{\boldsymbol{\beta}}_{\text{ElasticNet}} = \arg \min_{\boldsymbol{\beta}} \left\{ -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^k [I(y_i = C_j) \ln [P(y_i = C_j; \boldsymbol{x}_i)]] + \lambda \sum_{l=1}^{k-1} P_\alpha(\beta_l) \right\}. \quad (79)$$

Es importante notar que la última suma indica que la regularización se aplica únicamente a los coeficientes de las categorías del *output* que poseen parámetros, es decir, a todas excepto la categoría de referencia. Esto es necesario porque en un modelo multinomial hay $k - 1$ conjuntos de coeficientes β_l que se deben penalizar, ver expresión (70).

Para la implementación del modelo MLR con penalización *ElasticNet* se utilizará el paquete `glmnet` [[Friedman et al., 2010](#)]. En este caso, el hiperparámetro α se evaluará dentro de una malla de valores, mientras que el valor óptimo del hiperparámetro λ se determinará mediante un esquema de validación cruzada con 3 subconjuntos (*folds*). Es decir, para cada valor de α , se ajustará un modelo y se seleccionará el valor de λ que minimice la Tasa de Clasificación Correcta Global (TCCG) en la validación cruzada. Finalmente, se elegirá la combinación de α y λ que genere el menor valor de la TCCG entre todas las configuraciones evaluadas.

Máquina de soporte vectorial

Un modelo SVM (*support vector machine*) es un algoritmo de aprendizaje supervisado utilizado para problemas de clasificación. Su objetivo principal es encontrar un hiperplano que divida los datos en clases de la mejor manera posible, maximizando el margen entre los puntos de datos más cercanos de cada clase, llamados vectores de soporte. Este enfoque hace que el modelo SVM sea robusto y eficaz en problemas tanto lineales como no lineales, gracias al uso de funciones *kernel* que permiten mapear los datos a espacios de mayor dimensión, donde pueden ser separados de manera lineal, es decir es un método no lineal cuando se usan *Kernels*.

Para la implementación del modelo, se utilizará el paquete e1071 [Meyer et al., 2022]. Dependiendo del *kernel* seleccionado, se pueden ajustar distintos hiperparámetros que influyen en el desempeño del modelo. En el Cuadro 2, se presentan los hiperparámetros principales del paquete e1071 y su descripción.

Para mayor detalle sobre este algoritmo de clasificación se puede consultar Cristianini and Shawe-Taylor (2000), James et al. (2021, sección 9) o Hastie et al. (2009, sección 12).

Hiperparámetro	Argumento en e1071	Descripción
<i>kernel</i>	<i>kernel</i>	Define la transformación utilizada para mapear los datos a un espacio de mayor dimensión. Opciones: <code>linear</code> , <code>polynomial</code> , <code>radial</code> y <code>sigmoid</code> .
Parámetro de regularización.	<i>cost</i>	Controla el equilibrio entre maximizar el margen y minimizar los errores de clasificación. Valores altos intentan clasificar correctamente todas las observaciones, mientras que valores bajos permiten cierta tolerancia a errores para evitar sobreajuste.
Parámetro de escala del <i>kernel radial</i> .	<i>gamma</i>	Controla el alcance de la influencia de una muestra en la clasificación con <i>kernel radial</i> . Valores altos generan modelos más complejos, mientras que valores bajos suavizan la frontera de decisión.
Grado del <i>kernel polynomial</i> .	<i>degree</i>	Especifica el grado del polinomio utilizado cuando se selecciona un <i>kernel polynomial</i> .

Cuadro 2: Hiperparámetros del modelo SVM en el paquete e1071 y su descripción.

En este trabajo se explorará el *kernel radial* moviendo los hiperparámetros *cost* y *gamma*.

Bosques aleatorios

Un modelo RF (*random forest*) es un algoritmo de aprendizaje supervisado basado en la combinación de múltiples árboles de decisión. Cada árbol es entrenado sobre diferentes subconjuntos del conjunto de datos original (muestras *bootstrap* a partir de muestreo con reemplazo) y del conjunto de variables de entrada (*inputs*) (muestreo aleatorio de *inputs* en cada división del árbol), lo que genera un modelo robusto. Al final, las predicciones individuales de los árboles se combinan (por voto mayoritario para clasificación) para generar la predicción final.

Para la implementación del modelo, se utilizará el paquete `ranger` [Wright and Ziegler, 2023]. Dependiendo de la configuración del modelo, se pueden ajustar distintos hiperparámetros que influyen en su desempeño. En el Cuadro 3, se presenta una tabla con los hiperparámetros principales del paquete `ranger` y su descripción.

Para mayor detalle sobre este algoritmo de clasificación, se puede consultar James et al. (2021, sección sección 8) o Hastie et al. (2009, sección 15).

En este trabajo se calibraran los hiperparámetros `num.trees`, `mtry` y `min.node.size`.

Hiperparámetro	Argumento en <code>ranger</code>	Descripción
Número de árboles.	<code>num.trees</code>	Controla la cantidad de árboles en el bosque. Un número mayor mejora la estabilidad pero aumenta el tiempo de cómputo.
Número de variables por división.	<code>mtry</code>	Define cuántas variables se consideran en cada división del árbol. Un valor bajo introduce más variabilidad, mientras que un valor alto puede llevar a sobreajuste.
Mínimo de observaciones en un nodo.	<code>min.node.size</code>	Especifica el número mínimo de observaciones necesarias para dividir un nodo. Valores altos generan árboles más pequeños y menos complejos.

Cuadro 3: Hiperparámetros del modelo RF en el paquete `ranger` y su descripción.

A continuación se presenta el Cuadro 4, a manera de resumen, describiendo las ventajas y desventajas de cada modelo discutido en las secciones 2.3.1 y 2.3.2.

Modelo	Descripción	Ventajas	Desventajas	Hiperparámetros
UGGM-QDA	Extensión de QDA utilizando <i>glasso</i> para la estimación de la matriz de concentración.	Manejo eficiente de alta dimensionalidad mediante penalización. Modela asociaciones cuadráticas.	Sensible a la selección del hiperparámetro λ .	λ , de acuerdo con lo descrito en la expresión (20).
QDA	Clasificación cuadrática basada en estimaciones de la matriz de concentración.	Modela asociaciones cuadráticas.	Requiere grandes cantidades de datos para una buena estimación de la matriz de concentración.	No requiere ajuste
RF	Ensamble de árboles de decisión con selección aleatoria de variables en cada partición.	Robustez ante datos con relaciones complejas.	Alta complejidad computacional.	Número de árboles, número de variables por división y mínimo de observaciones en un nodo.
SVM	Encuentra un hiperplano óptimo en un espacio transformado.	Manejo efectivo de alta dimensionalidad y relaciones complejas.	Requiere ajuste de hiperparámetros.	<i>kernel</i> , parámetro de regularización, parámetro de escala del <i>kernel</i> y grado del <i>kernel</i> polinomial.
LDA	Clasificación lineal basada en la estimación de una sola matriz de concentración.	Simplicidad, interpretabilidad y eficiencia computacional.	Sólo considera relaciones lineales.	No requiere ajuste
MLR	Extiende la regresión logística para múltiples clases.	Interpretabilidad y robustez.	Puede tener problemas en alta dimensionalidad sin penalización.	Penalización <i>lasso</i> λ , de acuerdo con lo descrito en la expresión (75)

Cuadro 4: Comparación de modelos de clasificación utilizados, incluyendo ventajas, desventajas e hiperparámetros.

2.4 COMPARACIÓN EMPÍRICA DE LOS MODELOS USANDO DATOS SIMULADOS

Para concluir este capítulo, se complementa el ejercicio de comparación con datos simulados presentado en la Figura 10, incluyendo las tasas de clasificación correcta global de los métodos: análisis de discriminante lineal (LDA), regresión logística multinomial (MLR), árboles aleatorios (RF) y máquinas de soporte vectorial (SVM). Las especificaciones sobre el uso de los métodos se pueden encontrar en el Cuadro 5. Este ejercicio de comparación es muy sencillo, ya que los métodos RF y SVM sólo se usaron con los valores de los hiperparámetros por defecto, es decir, no se incluyó un proceso de calibración de hiperparámetros. Una comparación más exhaustiva se llevará a cabo en el siguiente capítulo con los datos sobre cáncer de cerebro.

Modelo	Descripción	Hiperparámetros	Malla de valores
UGGM-QDA	Análisis de Discriminante Cuadrático con penalización $glasso$.	$glasso: \lambda$	$\lambda \in [0, 0.1]$ con incrementos de 0.001
QDA	Análisis de Discriminante Cuadrático.	-	-
SVM	Máquina de Soporte Vectorial.	$kernel: \text{radial}$ $cost: 1$ $gamma: \frac{1}{n}$	-
RF	Bosques Aleatorios.	$num.trees: 500$ $mtry: \lfloor \sqrt{p} \rfloor$ $node.size: 1$	-
MLR	Regresión Logística Multinomial con efectos principales.	$cut off: 0.5$	-
LDA	Análisis de Discriminante Lineal.	-	-

Cuadro 5: Descripción de los modelos de clasificación utilizados en el ejercicio con datos simulados.

En la Figura 10 se comparan los modelos a partir de la Tasa de Clasificación Correcta Global (TCCG) utilizando los datos simulados. Es importante señalar que los datos fueron simulados específicamente para permitir una comparación directa entre QDA y UGGM-QDA, lo que podría no ser completamente justo para otros modelos como RF, SVM, LDA y MLR. Por lo tanto, los resultados deben interpretarse únicamente como un ejercicio comparativo y no como una evaluación general de la efectividad de estos modelos en otros contextos.

A partir de los resultados mostrados en la Figura 10, es posible hacer varias observaciones relevantes. En primer lugar, se confirma que los modelos LDA y MLR tuvieron un bajo desempeño, lo cual era esperado debido a la estructura de los datos simulados. Como se mencionó previamente, ambos modelos modelan la relación entre las variables de forma lineal, lo que no resulta adecuado para los datos analizados. Además, dado que LDA y MLR son algebraicamente equivalentes en sus expresiones, aunque no en sus estimaciones, sus resultados son similares, lo que refuerza la idea de que estos modelos no son óptimos para este tipo de datos. Para conocer la relación entre los modelos LDA y MLR, se puede consultar Murphy (2022, sección 9.2.3).

Por otro lado, los modelos RF y Support Vector Machine (SVM) mostraron un desempeño aceptable, a pesar de haber sido implementados con los hiperparámetros por defecto. Esto

sugiere que, aunque estos modelos pueden adaptarse a problema con estructuras de datos más complejas, su rendimiento podría mejorar con un proceso de *tuneo* de hiperparámetros, lo que a su vez implicaría un mayor costo computacional para el usuario.

Finalmente, los modelos QDA y UGGM-QDA superaron a los demás métodos al aprovechar la estructura de los datos definida desde la simulación. En particular, UGGM-QDA obtuvo el mejor desempeño en escenarios de alta dimensionalidad y tamaños de muestra reducidos, lo que valida la ventaja de utilizar el algoritmo *glasso* para la estimación de la matriz de concentración. Este resultado refuerza la efectividad de UGGM-QDA en situaciones donde la estimación de la matriz de concentración se vuelve inestable, destacando su utilidad frente a modelos más tradicionales como QDA.

Como ya se mencionó, este ejercicio de comparación fue muy simple y está limitado ya que para algunos métodos no se consideró un proceso de calibración de hiperparámetros; también dado que sólo se consideró una partición de datos para entrenar y de datos para estimar el poder predictivo y, en general, en los experimentos de simulación esto se debe repetir varias veces. Aun así, el ejercicio sirvió para observar que los modelos gráficos probabilísticos pueden ser útiles para definir reglas de clasificación cuando el tamaño de la muestra es pequeña y las diferencias entre las poblaciones se centran en las relaciones entre las variables.

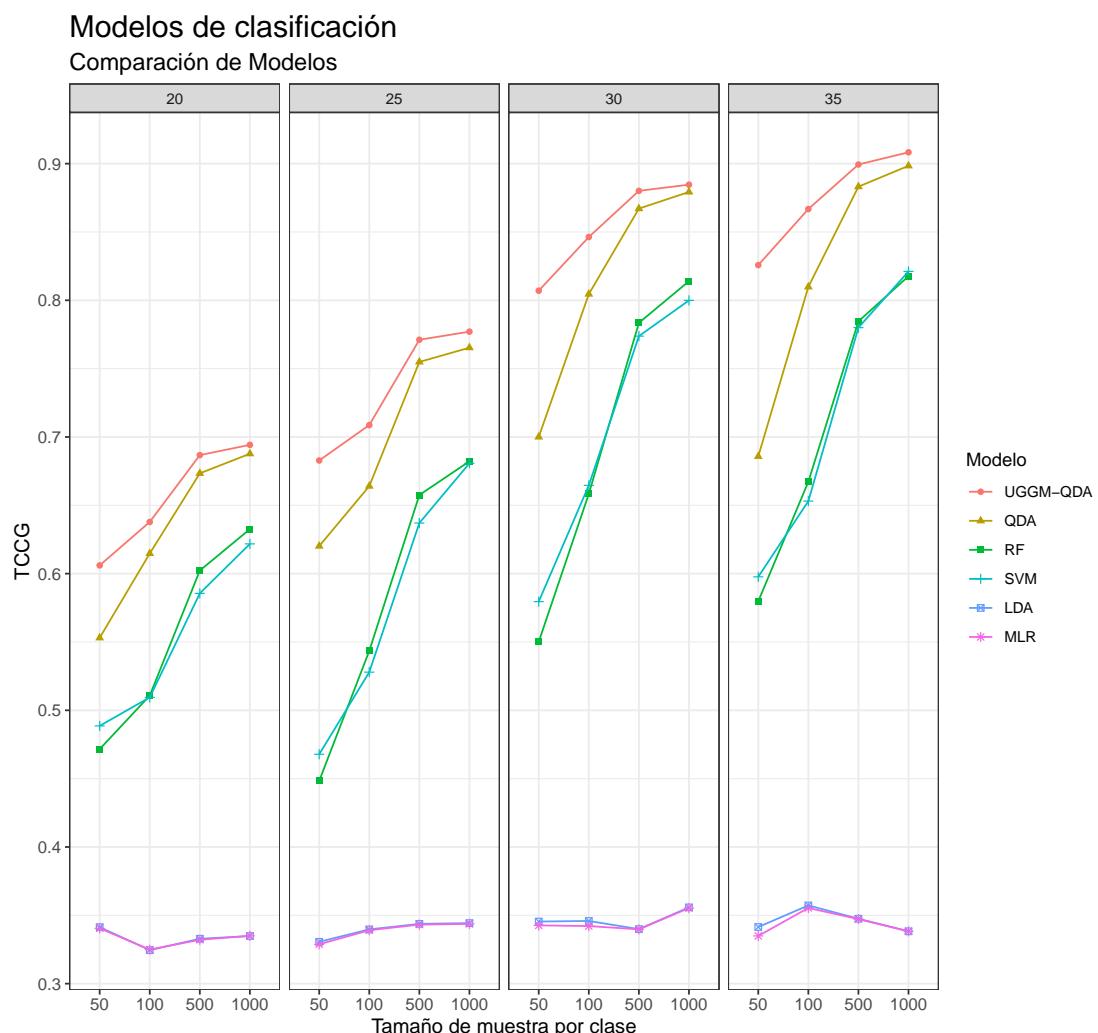


Figura 10: Comparación del poder predictivo, utilizando la Tasa de Clasificación Correcta Global (TCCG), entre los modelos descritos en el Cuadro 5.

3

CASO PRÁCTICO

En este capítulo se analiza y evalúa el poder predictivo de los distintos modelos de clasificación, explorados en el capítulo 2, aplicados a los datos de expresión genética de las enzimas de la vía de las kinureninas en distintos tipos de gliomas y tejido sano. El contenido se organiza en cuatro secciones principales: Datos, Análisis exploratorio, Estimación del poder predictivo y Análisis de los resultados. A continuación, se detallan los objetivos específicos de cada sección:

- **Datos:** Describir la relevancia de la vía de las kinureninas en el cáncer de cerebro y presentar las fuentes de datos utilizadas para el análisis, incluyendo su estructura y preprocesamiento.
- **Análisis exploratorio:** Identificar patrones y relaciones en los datos mediante visualización de datos, correlaciones parciales y análisis de componentes principales. Evaluar diferencias significativas entre grupos mediante pruebas no paramétricas.
- **Estimación del poder predictivo:** Implementar y comparar los distintos modelos de clasificación, descritos en el capítulo 2, para evaluar su capacidad de distinguir entre los grupos de estudio. Evaluar el desempeño de los modelos y validar su estabilidad.
- **Análisis de los resultados:** Interpretar los resultados obtenidos en la clasificación, identificando los modelos con mejor desempeño en cada grupo y evaluar la robustez y eficiencia computacional de los modelos implementados.

3.1 DATOS

3.1.1 *La vía de las kinureninas*

El triptófano es un aminoácido esencial que obtenemos a través de la alimentación y cumple funciones clave en nuestro organismo. Es conocido por ser el precursor de la serotonina, un neurotransmisor que influye en nuestro estado de ánimo y en diversos procesos conductuales, así como de la melatonina, una hormona que regula el sueño. La mayor parte del triptófano se procesa a través de una serie de reacciones químicas que conforman la llamada vía de las kinureninas (VK). Esta vía, se lleva acabo en el hígado, los riñones y el sistema nervioso central. Aunque el proceso es similar en diferentes órganos, en el hígado y el cerebro hay algunas diferencias en cómo comienza esta vía.

La VK está controlada por dos enzimas principales: la indoleamina 2,3-dioxigenasa (IDO) y la triptófano 2,3-dioxigenasa (TDO), las cuales inician la degradación del triptófano en el cerebro y el hígado, respectivamente. Estudios recientes han demostrado que esta vía está estrechamente relacionada con procesos inflamatorios y enfermedades como el cáncer cerebral [Cervantes et al., 2017].

3.1.2 Relación con el cáncer de cerebro

Se ha demostrado que la VK está estrechamente relacionada con el desarrollo y la progresión del cáncer cerebral. En particular, el glioblastoma, uno de los tumores más agresivos del cerebro, presenta una mayor actividad de esta vía. Enzimas como la kinureninasa (KYN) han sido encontradas en niveles elevados en estos tumores, lo que sugiere que su actividad podría estar favoreciendo el crecimiento y la supervivencia de las células cancerígenas [Pérez de la Cruz et al., 2023].

Además, se ha observado que el aumento en la actividad de IDO y KYN en los tumores cerebrales puede afectar la manera en que el sistema inmunológico responde a la enfermedad. En lugar de atacar el tumor, el cuerpo parece generar un ambiente que permite su desarrollo sin muchas barreras [Vázquez Cervantes et al., 2022]. Otros estudios han sugerido que la VK no solo influye en la inmunidad, sino que también podría estar relacionada con la producción de ciertos compuestos que favorecen la inflamación y la proliferación de células tumorales [Summers et al., 2024].

Investigaciones recientes también han explorado cómo los metabolitos de la VK pueden influir en la progresión del cáncer cerebral. Se ha identificado que el ácido quinolínico, un subproducto de esta vía, puede estar involucrado en procesos que afectan la estabilidad del ADN en las células tumorales, lo que podría contribuir a la agresividad del glioblastoma [Cervenka et al., 2017].

En conclusión, la VK, a través de sus enzimas clave como IDO, TDO y KYN, desempeña un papel importante en el crecimiento y la progresión del cáncer cerebral. Su estudio es fundamental para entender mejor cómo se desarrollan estos tumores y cómo se podrían diseñar nuevas estrategias terapéuticas para combatirlos.

3.1.3 La vía de las kinureninas desde un enfoque de clasificación supervisada

El análisis de clasificación presentado en este trabajo se fundamenta en investigaciones previas sobre la expresión de las enzimas de la vía de las kinureninas (VK) en diferentes tipos de gliomas y su relación con características tumorales clave [Pérez de la Cruz et al., 2023]. Para ello, en este trabajo se emplean datos de expresión genética de las principales enzimas involucradas en esta vía, obtenidos de muestras de cáncer cerebral y tejido sano. La descripción detallada de estas enzimas se presenta en el Cuadro 6.

Los datos utilizados en este análisis corresponden a valores de expresión genética normalizados (*RSEM norm count*) de muestras provenientes de los proyectos TCGA, TARGET y GTEx, accesibles a través de la plataforma Xena [Goldman et al., 2020]. Estas bases de datos incluyen

información de pacientes con distintos tipos de cáncer y tejidos sanos, recopilada mayormente en instituciones de Estados Unidos.

El objetivo es evaluar la capacidad predictiva de distintos modelos de clasificación para distinguir entre tres grupos de muestras, utilizando como variables cuantitativas los niveles de expresión de las enzimas descritas en el Cuadro 6. La definición y características de estos grupos se presentan en el Cuadro 7.

Datos y Preprocesamiento

De un total de 19,131 muestras disponibles, se seleccionaron aquellas correspondientes a gliomas de bajo grado (TCGA LGG, $n = 523$), glioblastoma multiforme (TCGA GM, $n = 166$) y muestras de tejido cerebral sano (GTEx brain cortex, $n = 105$; GTEx brain anterior cingulate cortex (Ba24), $n = 83$; y GTEx brain frontal cortex (Ba9), $n = 95$).

Las enzimas incluidas en este análisis se detallan en el Cuadro 6. Cada variable representa la expresión genética de una enzima en la VK, las cuales han sido estudiadas por su implicación en la progresión tumoral y la inmunosupresión en gliomas. En particular, se destaca el papel de KYNU, IDO1 y KMO en la modulación de la respuesta inmune y la resistencia terapéutica [Vázquez Cervantes et al., 2022].

Nombre	Abreviación	Variable
Tryptophan dioxygenase	TDO	x_1
Indoleamine dioxygenase 1	IDO1	x_2
Indoleamine dioxygenase 2	IDO2	x_3
Arylformamidase	AFMID	x_4
Glutamic-oxaloacetic transaminase	GOT2	x_5
Aminoadipate aminotransferase	AADAT	x_6
Kynureninase	KYNU	x_7
Kynurenone monooxygenase	KMO	x_8
Quinolinic acid phosphoribosyl transferase	QPRT	x_9
3-HANA dioxygenase	HAAO	x_{10}
Aminocarboxymuconate semialdehyde decarboxylase	ACMSD	x_{11}

Cuadro 6: Principales enzimas asociadas en el proceso de la vía de las kinureninas.

Definición de Grupos

El problema de clasificación se definió con base en tres grupos, detallados en el Cuadro 7. Estos grupos representan distintos estados, lo que permite evaluar la capacidad de las enzimas de la VK para clasificar entre tejido sano y distintos tipos de gliomas.

Utilizando el valor $\log_2(\text{norm count} + 1)$ de las principales expresiones genéticas de las enzimas involucradas en la VK, se exploraron diferentes modelos de clasificación para comparar su poder predictivo para clasificar entre 3 diferentes clases dado un conjunto de 11 variables, como se describe en los Cuadros 6 y 7.

Clase	Grupo	Abreviación	Tamaño de muestra
1	GTEX Brain	GTEX Brain	283
2	TCGA Brain Lower Grade Glioma	TCGA LGG	523
3	TCGA Glioblastoma Multiforme	TCGA GM	166

Cuadro 7: Clases asociadas a los grupos y su tamaño de muestra

3.2 ANÁLISIS EXPLORATORIO

Para comprender mejor la estructura y relaciones en los datos de expresión genética de las enzimas de la VK, se realizó un análisis exploratorio en diferentes etapas.

Inicialmente, para evaluar diferencias estadísticamente significativas entre grupos, se aplicó la prueba de Kruskal-Wallis, seguida de pruebas Dunn para analizar diferencias entre pares de grupos.

Posteriormente, se generó una matriz de diagramas de dispersión junto con las correlaciones de Pearson y la distribución de cada variable, diferenciando entre grupos. Esto permitió visualizar la dispersión de las variables y detectar posibles patrones o relaciones directas entre las expresiones genéticas. Además, se calcularon las correlaciones parciales de Pearson, con el fin de evaluar la relación entre pares de variables mientras se controla el efecto de las demás.

Finalmente, se realizó un análisis de componentes principales (PCA) para reducir la dimensionalidad de los datos y explorar la variabilidad explicada por combinaciones lineales de las enzimas involucradas en la VK.

3.2.1 Pruebas no paramétricas

Se realizó una comparativa de las variables del Cuadro 6 entre los 3 grupos del Cuadro 7. En la Figura 11 se ilustran los resultados de esta comparación mediante gráficas de caja (*boxplot*), las cuales muestran la distribución de los valores de las expresiones genéticas para cada variable de interés en los distintos grupos.

Para determinar si existían diferencias significativas entre los grupos, se aplicó la prueba no paramétrica Kruskal-Wallis (K-W), cuyo resultado reveló diferencias significativas en la mayoría de las variables analizadas ($p\text{-value} < 0.05$), a excepción de la variable IDO2, donde no se encontró evidencia suficiente para concluir que los grupos difieren significativamente ($p\text{-value} \geq 0.05$).

Posteriormente, con el objetivo de identificar qué grupos específicos se observaban dichas diferencias, se realizaron pruebas simultáneas Dunn con corrección Bonferroni para comparar los grupos de forma múltiple. Este análisis permitió identificar si existía una diferencia significativa entre cada par de grupos ($p\text{-value} < 0.05$), proporcionando una visión más detallada sobre las distribuciones de las variables entre los grupos. Se puede observar que entre las variables que resultaron significativas con K-W también se observa una diferencia entre los 3 grupos, excepto en AFMID (TCGA LGG v.s TCGA GM) y QPRT (GTEX Brain v.s TCGA LGG).

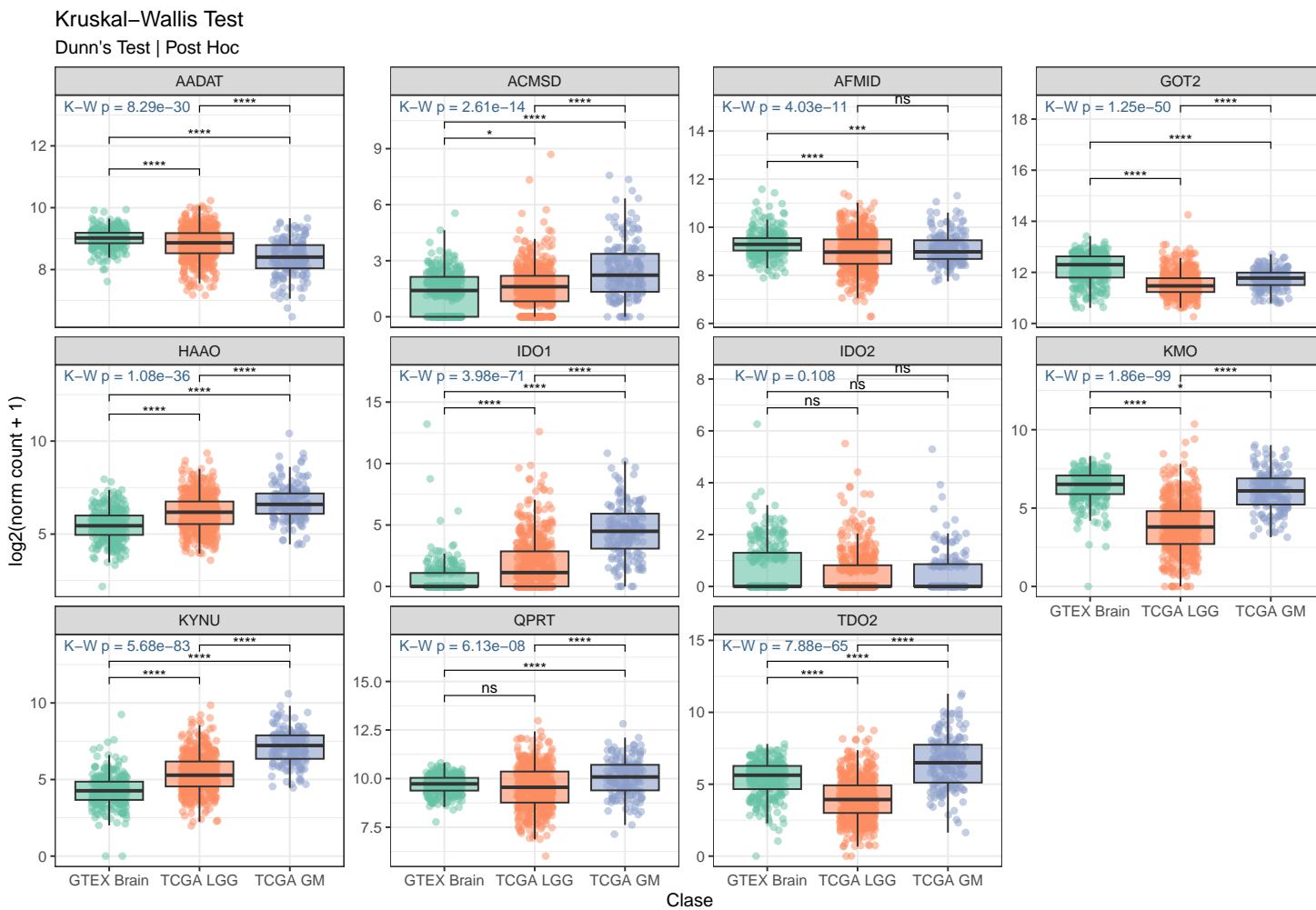


Figura 11: Análisis comparativo de la expresión genética entre grupos mediante las pruebas Kruskal-Wallis y Dunn. Cada panel representa la expresión de una enzima de la vía de las kinureninas en los tres grupos de estudio (GTEx Brain, TCGA LGG y TCGA GM). Los valores en azul corresponden a la prueba de Kruskal-Wallis, utilizada para evaluar diferencias globales entre los grupos. Las comparaciones entre pares se realizaron mediante la prueba de Dunn, con los siguientes niveles de significancia: ns ($p \geq 0.05$) no significativo, * ($p < 0.05$), ** ($p < 0.01$), *** ($p < 0.001$) y **** ($p < 0.0001$).

3.2.2 Diagramas de dispersión, correlaciones de Pearson y correlaciones parciales de Pearson

La Figura 12 muestra la relación entre los niveles de expresión de las principales enzimas, distinguiendo entre cada grupo GTEx Brain, TCGA LGG y TCGA GM. Destacando los siguientes puntos:

- Se identificaron relaciones lineales positivas entre algunas variables, destacándose las asociaciones más fuertes entre KYNU-IDO1 (0.67), KMO-TDO2 (0.58), HAAO-KYNU (0.58) y KMO-GOT2 (0.49). Por otro lado, se observó una relación negativa entre KYNU y AADAT (-0.46). En particular, KYNU mostró una mayor asociación con varias de las demás variables, lo que sugiere su relevancia dentro del conjunto de datos analizado.
- Aunque hay superposición entre clases, en ciertas combinaciones se aprecia una mayor separación entre los distintos grupos: KYNU-KMO y KYNU-QPRT.
- Enzimas como IDO1, KYNU y KMO muestran una tendencia a estar más expresadas en los grupos TCGA LGG y TCGA GM.

Con el objetivo de identificar posibles ceros en la matriz de concentración, se calcularon las correlaciones parciales de Pearson. En las Figuras 13, 14 y 15 se presentan dichas correlaciones para los grupos GTEx Brain, TCGA LGG y TCGA GM, respectivamente. A partir de los resultados observados, se pueden destacar los siguientes puntos:

- Grupo GTEx Brain:
 - GOT2 presenta las correlaciones parciales positivas más altas con TDO2 (0.52) y KMO (0.46). Además, presenta la correlación parcial negativa más fuerte con HAAO (-0.36).
 - En general, las enzimas TDO2, IDO1 y KYNU presentan correlaciones parciales en valores absolutos inferiores a 0.25.
- Grupo TCGA LGG:
 - IDO1 presenta una alta correlación con TDO2 (0.38) y KYNU (0.36).
 - KYNU y HAAO presentan la correlación parcial más alta (0.49).
 - GOT2 muestra una correlación positiva con KMO (0.34), a diferencia del grupo sano, donde la relación era más débil.
- Grupo TCGA GM:
 - KYNU presenta las correlaciones parciales positivas más altas con KMO (0.51) y TDO2 (0.47). Además, presenta la correlación parcial negativa más fuerte con AADAT (-0.29).
 - Comparado con el grupo TCGA LGG, las correlaciones en TCGA GM tienden a ser más altas.

Diagramas de dispersión

Principales enzimas involucradas en la vía de las Kinureninas

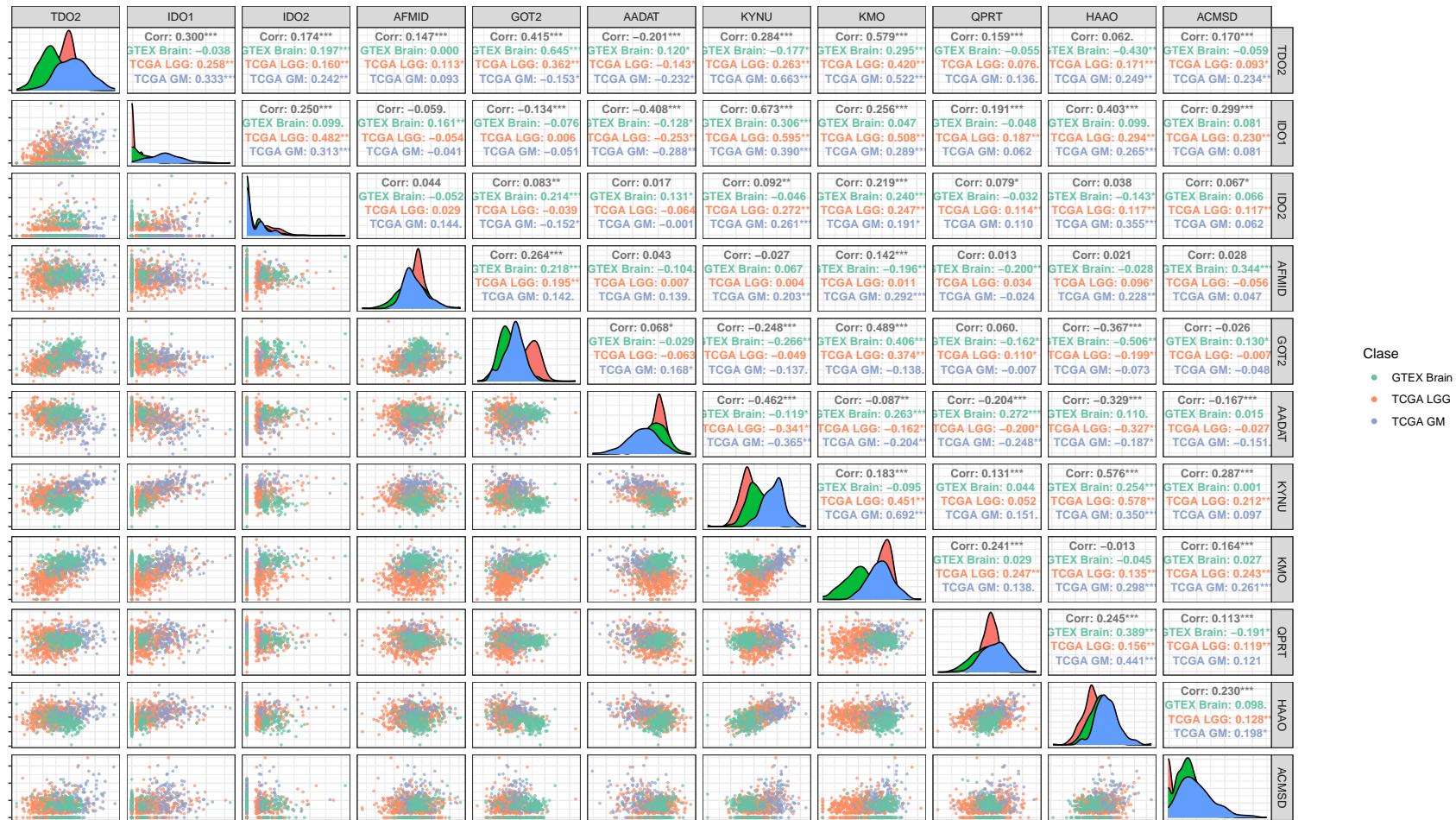


Figura 12: Matriz de diagramas de dispersión de las principales enzimas de la VK. Los colores representan diferentes grupos: GTEX Brain (verde), TCGA LGG (naranja) y TCGA GM (azul).

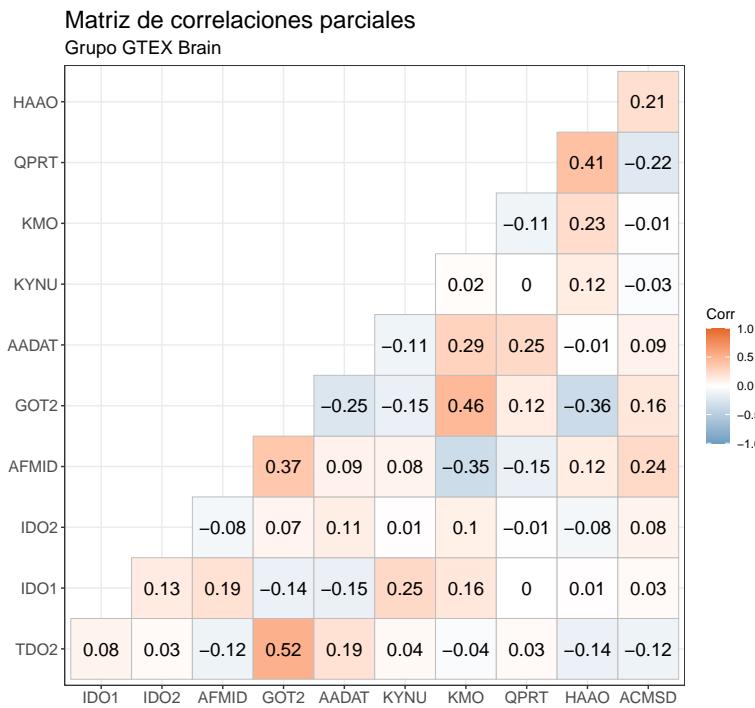


Figura 13: Matriz de correlaciones parciales entre las enzimas de la vía de las kinureninas en el grupo GTEx Brain. Los valores indican la relación entre pares de enzimas controlando por el efecto del resto de las variables.

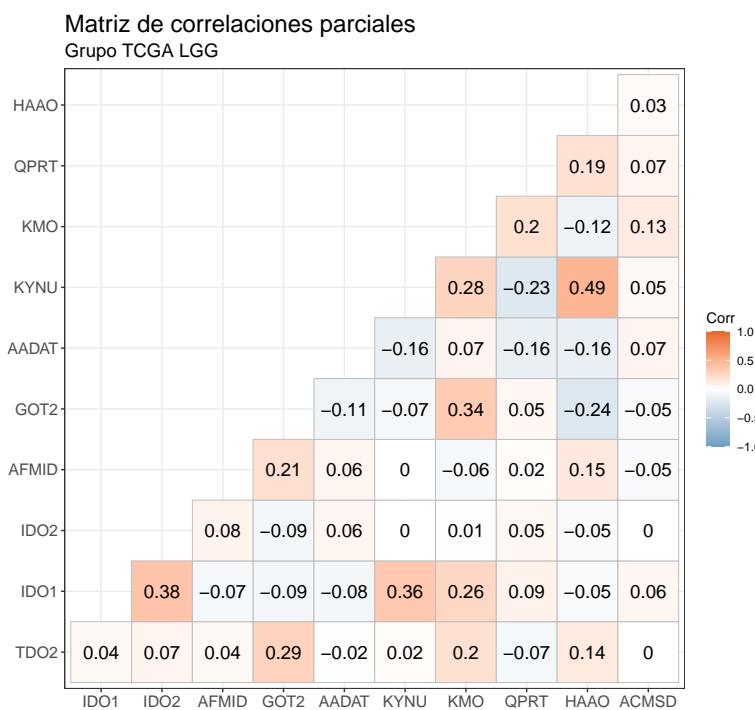


Figura 14: Matriz de correlaciones parciales entre las enzimas de la vía de las kinureninas en el grupo TCGA LGG. Los valores indican la relación entre pares de enzimas controlando por el efecto del resto de las variables.

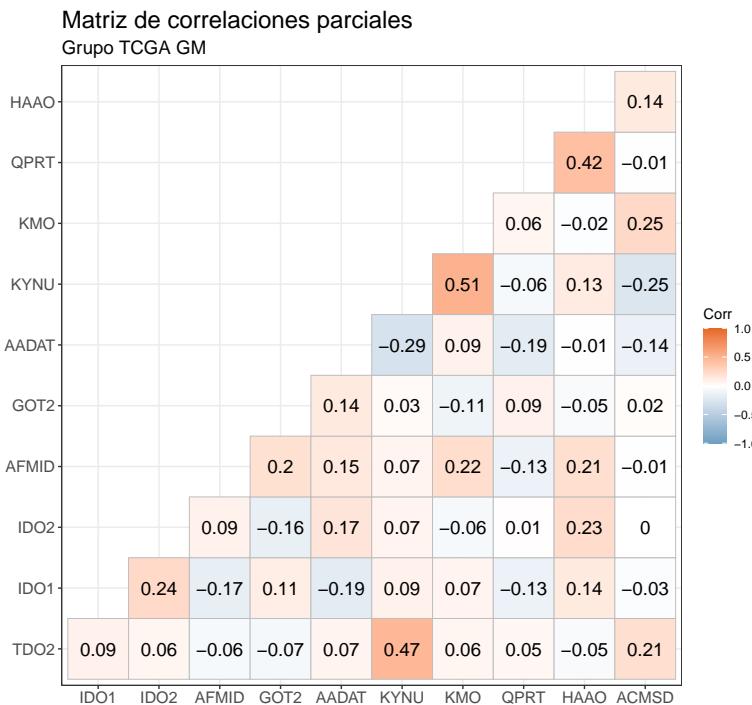


Figura 15: Matriz de correlaciones parciales entre las enzimas de la vía de las kinureninas en el grupo TCGA GM. Los valores indican la relación entre pares de enzimas controlando por el efecto del resto de las variables.

3.2.3 Componentes principales

Para analizar los datos reduciendo la dimensionalidad, se utilizó el Análisis de Componentes Principales (PCA). El análisis se realizó considerando las variables en su escala original, es decir, centrándolas pero sin escalarlas, dado que todas las variables comparten la misma unidad de medida y varianzas comparables [Jolliffe and Cadima, 2016]. Este método permite transformar las variables originales en combinaciones lineales llamadas componentes principales, las cuales capturan la mayor variabilidad posible en los datos.

En las Figuras 16, 17 y 18 se presentan las proyecciones de los datos en distintos pares de componentes principales, a partir de ellas se puede comentar lo siguiente:

- PC₁ v.s PC₂: Representa la distribución de las muestras en los dos primeros componentes principales (PC₁ vs PC₂), donde PC₁ explica el 43.1 % de la varianza y PC₂ el 22 %. Se observa una separación importante entre los 3 grupos. Mayores valores en el PC₁ y en el PC₂ se relacionan con el grupo TCGA LGG, mientras que valores negativos en el PC₁ al grupo TCGA GM y valores negativos de PC₂ al grupo GTEX Brain, es decir a tejido sano.
- PC₁ v.s PC₃: Muestra la relación entre PC₁ y PC₃, donde PC₃ explica el 7.61 % de la varianza. En esta proyección, la separación entre grupos es menos evidente, con mayor superposición entre los grupos en comparación con PC₁ vs PC₂.
- PC₂ v.s PC₃: Se observa que la diferenciación es menos marcada, de esta forma la mayor variabilidad está explicada por PC₁ y PC₂.

Para complementar el análisis de los componentes principales se presenta el Cuadro 8 con las correlaciones de Pearson entre las variables originales y los primeros tres componentes principales, lo que permite interpretar qué características principales son capturadas por cada componente.

- El PC₁ presenta altas correlaciones negativas con las enzimas IDO₁ (-0.87), KYNU (-0.74), TDO₂ (-0.64) y KMO (-0.60), lo que indica que este componente representa una combinación de estas enzimas en sentido opuesto.
- El PC₂ se encuentra altamente correlacionado en sentido negativo con KMO (-0.69), GOT₂ (-0.67) y TDO₂ (-0.58), mientras que muestra correlaciones positivas con IDO₁ (0.40), HAAO (0.41) y KYNU (0.38).
- El PC₃ muestra correlaciones moderadas en distintos sentidos. Destaca su relación positiva con TDO₂ (0.46), HAAO (0.24) y KYNU (0.24), y negativa con KMO (-0.34) e IDO₂ (-0.21).

	Componentes principales		
	PC ₁	PC ₂	PC ₃
TDO ₂	-0.64	-0.58	0.46
IDO ₁	-0.87	0.40	-0.16
IDO ₂	-0.29	-0.07	-0.21
AFMID	-0.04	-0.22	0.08
GOT ₂	-0.09	-0.67	-0.09
AADAT	0.43	-0.20	-0.15
KYNU	-0.74	0.38	0.24
KMO	-0.60	-0.69	-0.34
QPRT	-0.29	-0.08	-0.11
HAAO	-0.43	0.41	0.24
ACMSD	-0.39	0.09	0.15

Cuadro 8: Correlaciones de Pearson entre variables y los primeros 3 componentes principales.

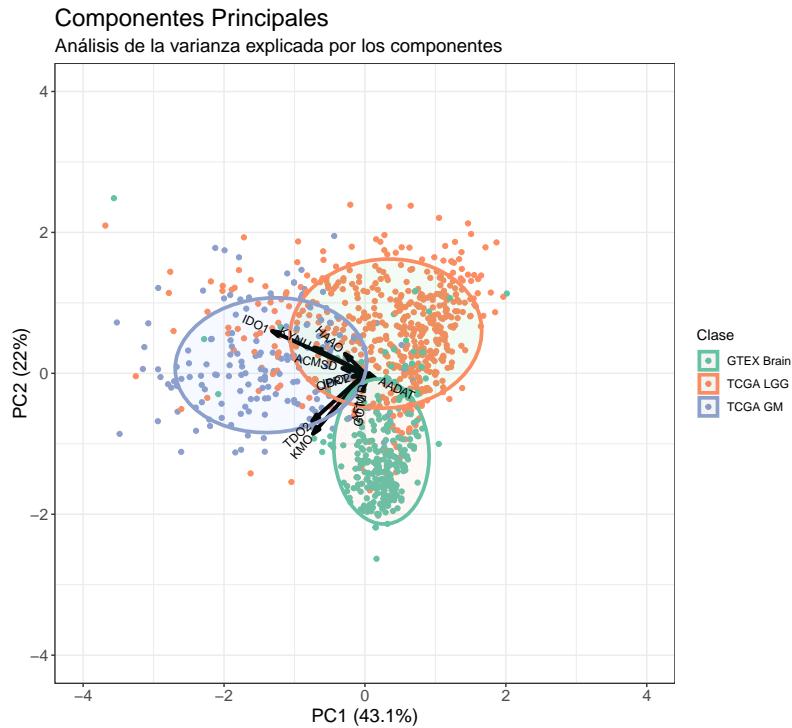


Figura 16: Distribución de los datos en función de los dos primeros componentes principales. PC1 explica el 43.1 % de la variabilidad total y PC2 el 22 %.

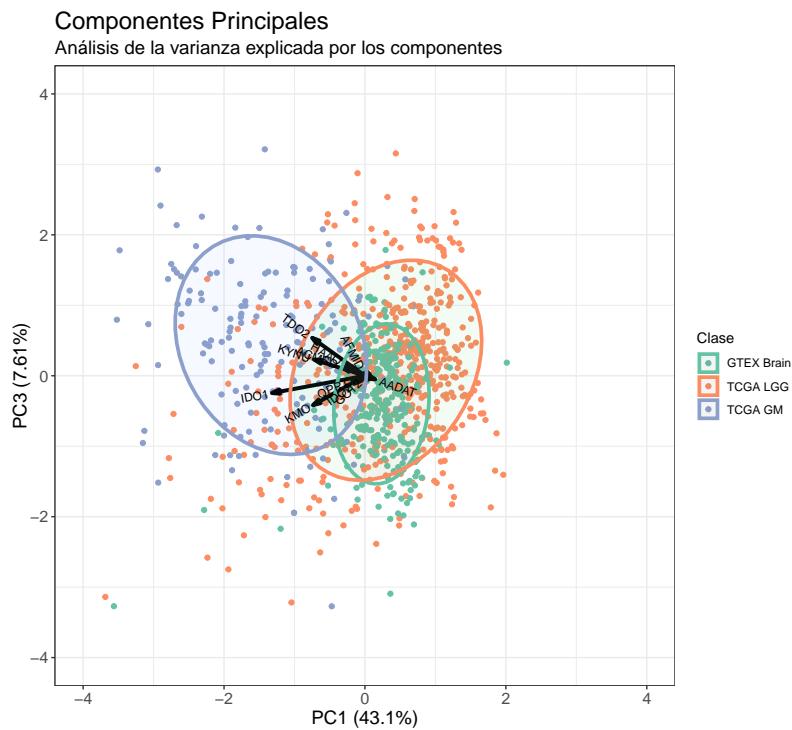


Figura 17: Distribución de los datos en función del primer y tercer componente principal. PC1 explica el 43.1 % de la variabilidad total y PC3 el 7.61 %.

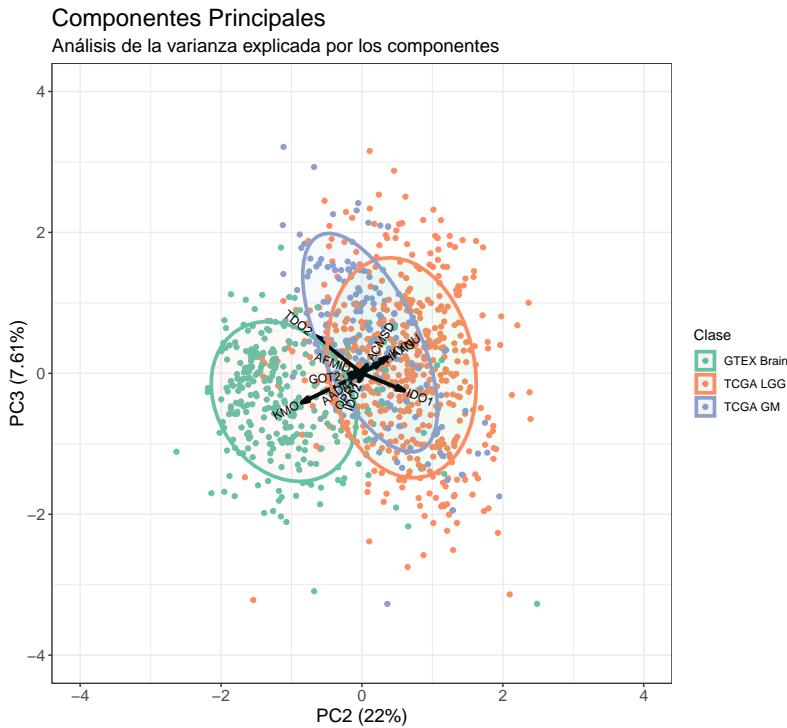


Figura 18: Distribución de los datos en función del segundo y tercero componente principal. PC2 explica el 22 % de la variabilidad total y PC3 el 7.61 %.

3.3 ESTIMACIÓN DEL PODER PREDICTIVO

En esta sección se presenta el análisis de la estimación del poder predictivo de los distintos modelos de clasificación aplicados a los datos previamente descritos. La descripción detallada de los modelos utilizados se muestra en el Cuadro 10. En los casos en los que se realizó la calibración de hiperparámetros, la métrica empleada fue la tasa de clasificación correcta general, definida en la expresión (34).

Los modelos con el sufijo 1 no incluyen calibración de hiperparámetros, mientras que los modelos con sufijo 2 incorporan un proceso de calibración de al menos un hiperparámetro. La única excepción es LDA 2, cuyo único cambio es la inclusión de la penalización con *glasso* fijo, conforme lo especifica el paquete NetDA [Chen, 2022]. En los casos donde se realiza *tuneo* de hiperparámetros esta se hace mediante un esquema de validación o una evaluación exhaustiva de cada combinación de hiperparámetros, es decir, probando cada combinación de las mallas de cada hiperparámetro.

Para calcular el poder predictivo de los modelos, se empleó un esquema de Repeated u - v Fold Cross Validation (Algoritmo 4), con $u = 100$ y $v = 5$, lo que significa que se realizaron 100 repeticiones de una validación cruzada de 5 pliegues, dando un total de 500 procesos diferentes. En los casos en donde se realiza calibración de hiperparámetros, esto se realiza en cada uno de los 500 procesos.

Los resultados obtenidos se presentan en los Cuadros 9 y 11. El primero muestra las métricas de evaluación a partir del conjunto de entrenamiento ($\mathcal{D}_{\text{Train}}$), mientras que el segundo refleja las mismas métricas calculadas a partir del conjunto de evaluación ($\mathcal{D}_{\text{Test}}$). La diferencia entre

estos valores permite visualizar el error de generalización (*generalization error*), de acuerdo con [Hastie et al. \(2009\)](#), sección 2.9).

En general, los valores de $\mu(\mathcal{D}_{\text{Train}})$ son superiores a los de $\mu(\mathcal{D}_{\text{Test}})$, lo que indica que los modelos tienden a ajustarse mejor a los datos de entrenamiento y pueden experimentar una reducción en su desempeño al evaluar nuevas observaciones. Este efecto es especialmente notable en modelos complejos como Random Forest (RF) y Support Vector Machine (SVM), que alcanzan valores cercanos al 100 en el entrenamiento, pero sufren una disminución notable en el conjunto de evaluación.

Para evaluar la estabilidad del desempeño, el Cuadro 12 presenta las desviaciones estándar de cada métrica en el conjunto de evaluación. Modelos con menor desviación estándar, como LDA 2 (0.24), muestran un desempeño más consistente, mientras que modelos con mayor variabilidad, como SVM 2 (0.54), reflejan una mayor inestabilidad.

Finalmente, las Figuras 19, 20, 21 y 22 muestran gráficas de caja (*boxplot*) que ilustran la distribución de los valores obtenidos en cada paso de la validación cruzada para las tasas de clasificación calculadas con el conjunto de evaluación ($\mathcal{D}_{\text{Test}}$), proporcionando una representación visual del poder predictivo de cada modelo en función de las métricas evaluadas.

	$\mu(\mathcal{D}_{\text{Train}})$			
	TCCG	TCCC ₁	TCCC ₂	TCC C ₃
Multinomial Logistic Regression (MLR 1)	86.1	89.2	90.3	67.3
Multinomial Logistic Regression (MLR 2)	88.0	91.0	91.5	71.8
Linear Discriminant Analysis (LDA 1)	84.6	88.6	86.4	71.9
Linear Discriminant Analysis (LDA 2)	83.3	82.4	91.5	58.8
Quadratic Discriminant Analysis (QDA 1)	87.8	91.6	88.6	78.9
Quadratic Discriminant Analysis (QDA 2)	81.7	89.8	74.5	90.4
Support Vector Machine (SVM 1)	91.4	93.7	94.7	76.6
Support Vector Machine (SVM 2)	90.6	94.0	93.3	76.3
Random Forest (RF 1)	100	100	100	100
Random Forest (RF 2)	97.6	98.5	97.0	98.2

Cuadro 9: Comparación de Modelos en términos de 4 métricas calculadas con el conjunto $\mathcal{D}_{\text{Train}}$. En verde se muestra el valor más alto y en rojo el más bajo. TCCG: Tasa de Clasificación Correcta Global y TCCC_{*j*}: Tasa de Clasificación Correcta para la Clase 1 (GTEX Brain), Clase 2 (TCGA LGG) y Clase 3 (TCGA GM).

Modelo	Descripción	Hiperparámetros	Malla de valores	Esquema de <i>tuneo</i>
MLR 1	Regresión Logística Multinomial con efectos principales.	<i>cut off</i> : α	$\alpha = 0.5$	-
MLR 2	Regresión Logística Multinomial con efectos principales e interacciones de segundo orden con penalización <i>ElasticNet</i> .	<i>ridge</i> : α <i>lasso</i> : λ	$\alpha \in \{0, 0.5, 1\}$. 100 valores por default de la función <code>cv.glmnet()</code> del paquete <code>glmnet</code> para λ .	Esquema de validación cruzada con 3 subconjuntos (<i>folds</i>), mediante de la función <code>cv.glmnet()</code> .
LDA 1	Análisis de Discriminante Lineal.	-	-	-
LDA 2	Análisis de Discriminante Lineal en su versión de UGGM.	<i>glasso</i> : λ	$\lambda = 0.01$.	-
QDA 1	Análisis de Discriminante Cuadrático.	-	-	-
QDA 2	Modelo UGGM-QDA.	<i>glasso</i> : λ	$\lambda \in [0, 1]$ con incrementos de 0.01.	Esquema de validación cruzada con 5 subconjuntos (<i>folds</i>), mediante la implementación del Algoritmo 6.
SVM 1	Máquina de Soporte Vectorial.	<i>kernel</i> : radial <i>cost</i> : 1 <i>gamma</i> : $\frac{1}{n}$	-	-
SVM 2	Máquina de Soporte Vectorial.	<i>kernel</i> : radial <i>cost</i> : c <i>gamma</i> : γ	$c \in \{1, 11, 21, \dots, 91\}$ $\gamma \in \{0.1, 0.2, \dots, 1\}$	Esquema de validación cruzada con 5 subconjuntos (<i>folds</i>), mediante la función <code>svm.tune()</code> .
RF 1	Bosques Aleatorios.	<i>num.trees</i> : 500 <i>mtry</i> : $\lfloor \sqrt{11} \rfloor$ <i>node.size</i> : 1	-	-
RF 2	Bosques Aleatorios.	<i>num.trees</i> : nt <i>mtry</i> : m <i>node.size</i> : ns	$nt \in \{50, 100, 500\}$ $m \in \{1, 2, \dots, 11\}$ $ns \in \{1, 10, 15\}$	Evaluación exhaustiva de cada combinación de hiperparámetros.

Cuadro 10: Descripción de los modelos de clasificación utilizados. Los modelos con el sufijo 1 no incluyen *tuneo* de hiperparámetros, mientras que los modelos con sufijo 2 si incluyen *tuneo* de hiperparámetros, excepto LDA 2, cuyo único cambio es la inclusión de la penalización con *glasso* fijo, como se define en el paquete `NetDA`.

	$\mu (\mathcal{D}_{\text{Test}})$			
	TCC G	TCC C ₁	TCC C ₂	TCC C ₃
Multinomial Logistic Regression (MLR 1)	85.0	88.4	89.3	65.5
Multinomial Logistic Regression (MLR 2)	85.5	89.0	89.4	67.1
Linear Discriminant Analysis (LDA 1)	83.7	87.8	85.6	70.7
Linear Discriminant Analysis (LDA 2)	82.6	81.8	91.1	57.2
Quadratic Discriminant Analysis (QDA 1)	85.5	89.8	87.4	72.0
Quadratic Discriminant Analysis (QDA 2)	80.3	88.6	73.4	87.8
Support Vector Machine (SVM 1)	86.3	89.1	91.0	66.7
Support Vector Machine (SVM 2)	86.3	90.6	89.9	67.3
Random Forest (RF 1)	85.2	87.0	90.2	66.5
Random Forest (RF 2)	85.0	90.8	84.3	77.7

Cuadro 11: Comparación de Modelos en términos de 4 métricas calculadas con el conjunto $\mathcal{D}_{\text{Test}}$. En verde se muestra el valor más alto y en rojo el más bajo. TCCG: Tasa de Clasificación Correcta Global y TCCC_j: Tasa de Clasificación Correcta para la Clase 1 (GTEX Brain), Clase 2 (TCGA LGG) y Clase 3 (TCGA GM).

	Desviaciones Estándar			
	TCC G	TCC C ₁	TCC C ₂	TCC C ₃
Multinomial Logistic Regression (MLR 1)	0.35	0.62	0.51	1.16
Multinomial Logistic Regression (MLR 2)	0.49	0.76	0.68	1.48
Linear Discriminant Analysis (LDA 1)	0.32	0.52	0.45	0.93
Linear Discriminant Analysis (LDA 2)	0.24	0.51	0.26	0.99
Quadratic Discriminant Analysis (QDA 1)	0.42	0.47	0.49	1.55
Quadratic Discriminant Analysis (QDA 2)	0.41	0.59	0.63	0.94
Support Vector Machine (SVM 1)	0.43	0.47	0.49	1.53
Support Vector Machine (SVM 2)	0.54	0.82	0.67	1.63
Random Forest (RF 1)	0.39	0.54	0.54	1.31
Random Forest (RF 2)	0.46	0.73	0.51	1.55

Cuadro 12: Desviaciones estándar del proceso con el conjunto de prueba. En verde se muestra el valor más alto y en rojo el más bajo. TCCG: Tasa de Clasificación Correcta Global y TCCC_j: Tasa de Clasificación Correcta para la Clase 1 (GTEX Brain), Clase 2 (TCGA LGG) y Clase 3 (TCGA GM).

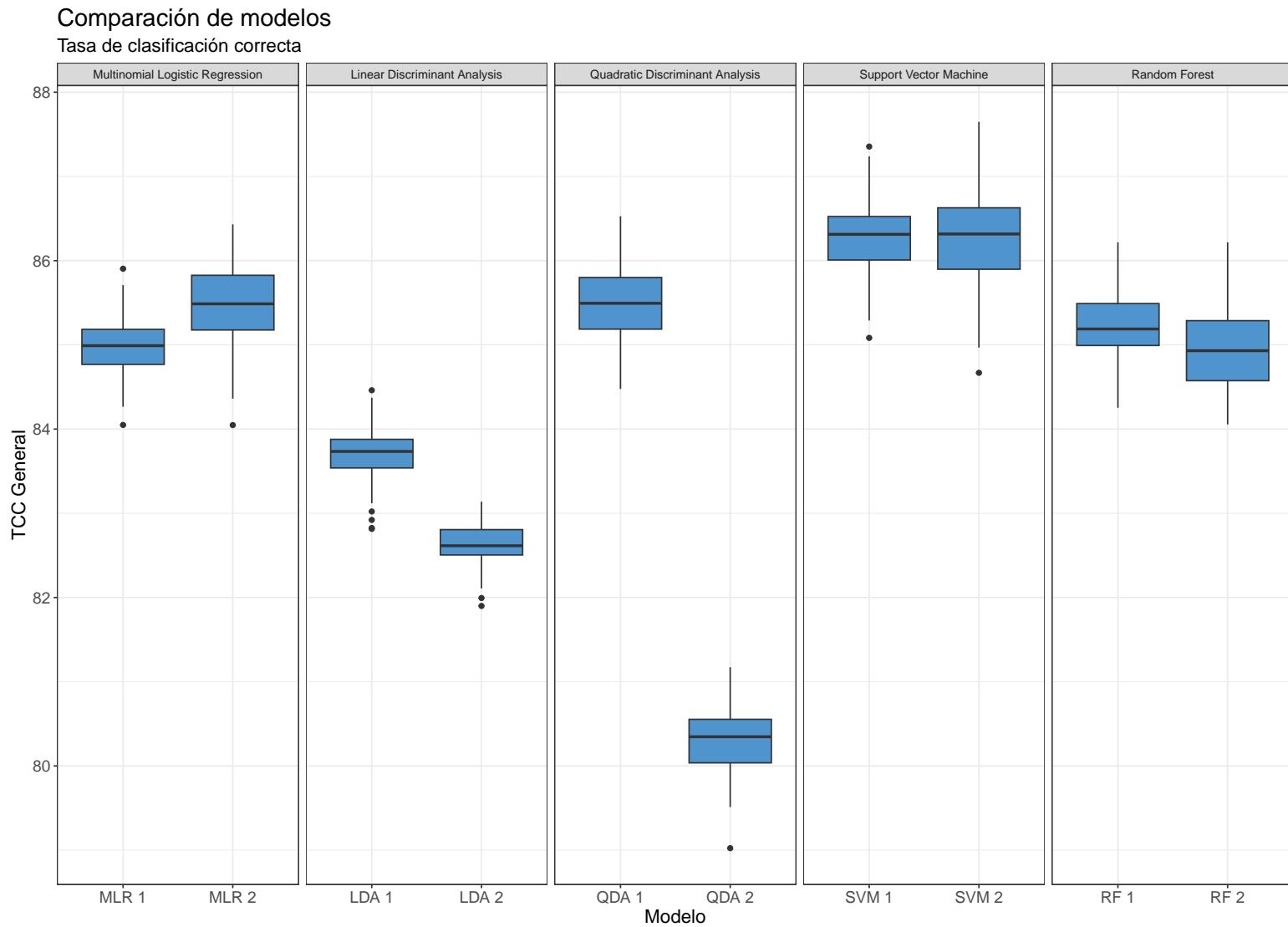


Figura 19: Tasa de Clasificación Correcta Global en el conjunto $\mathcal{D}_{\text{Test}}$.

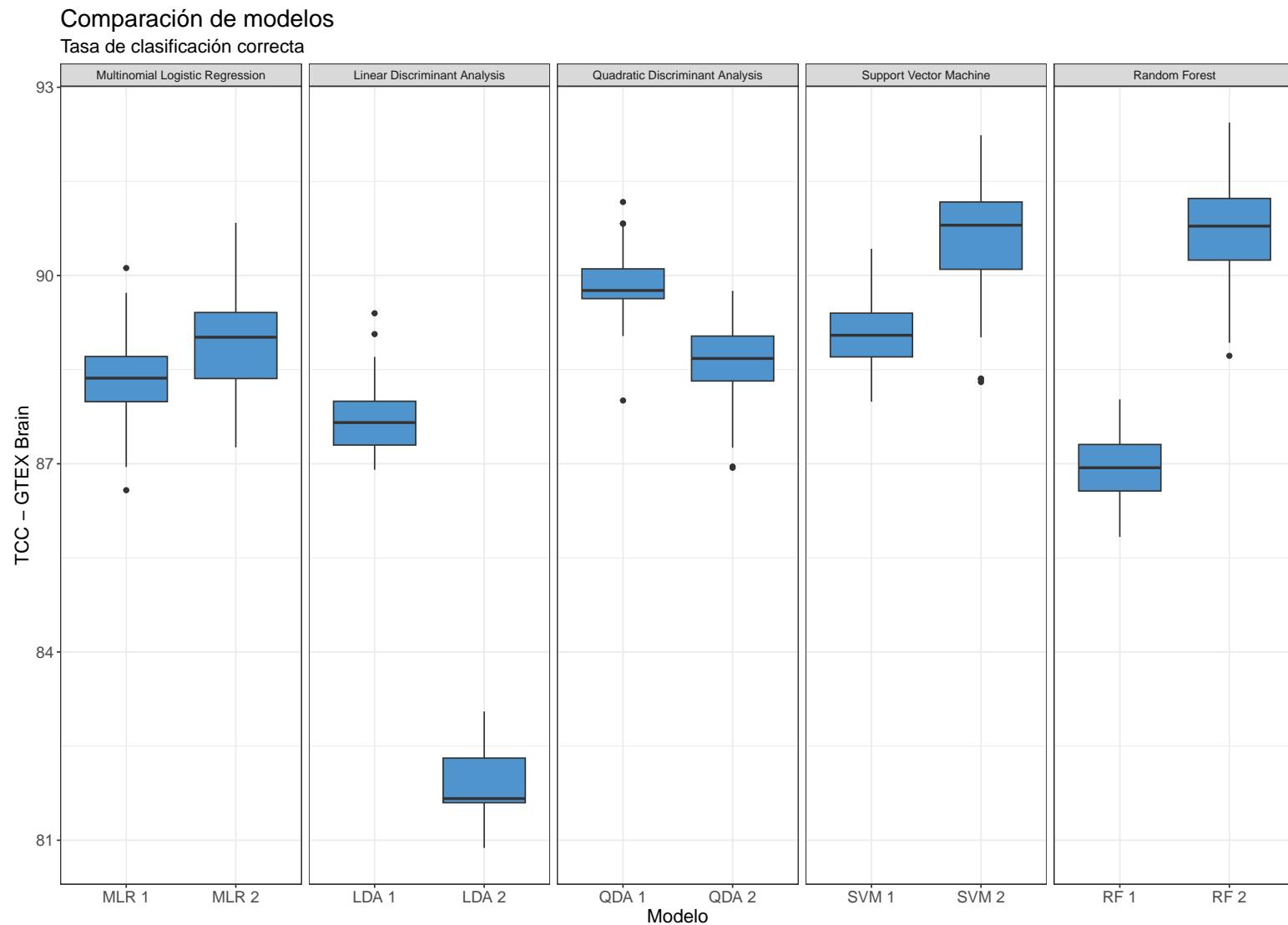


Figura 20: Tasa de Clasificación Correcta para la clase GTEX Brain en el conjunto $\mathcal{D}_{\text{Test}}$.

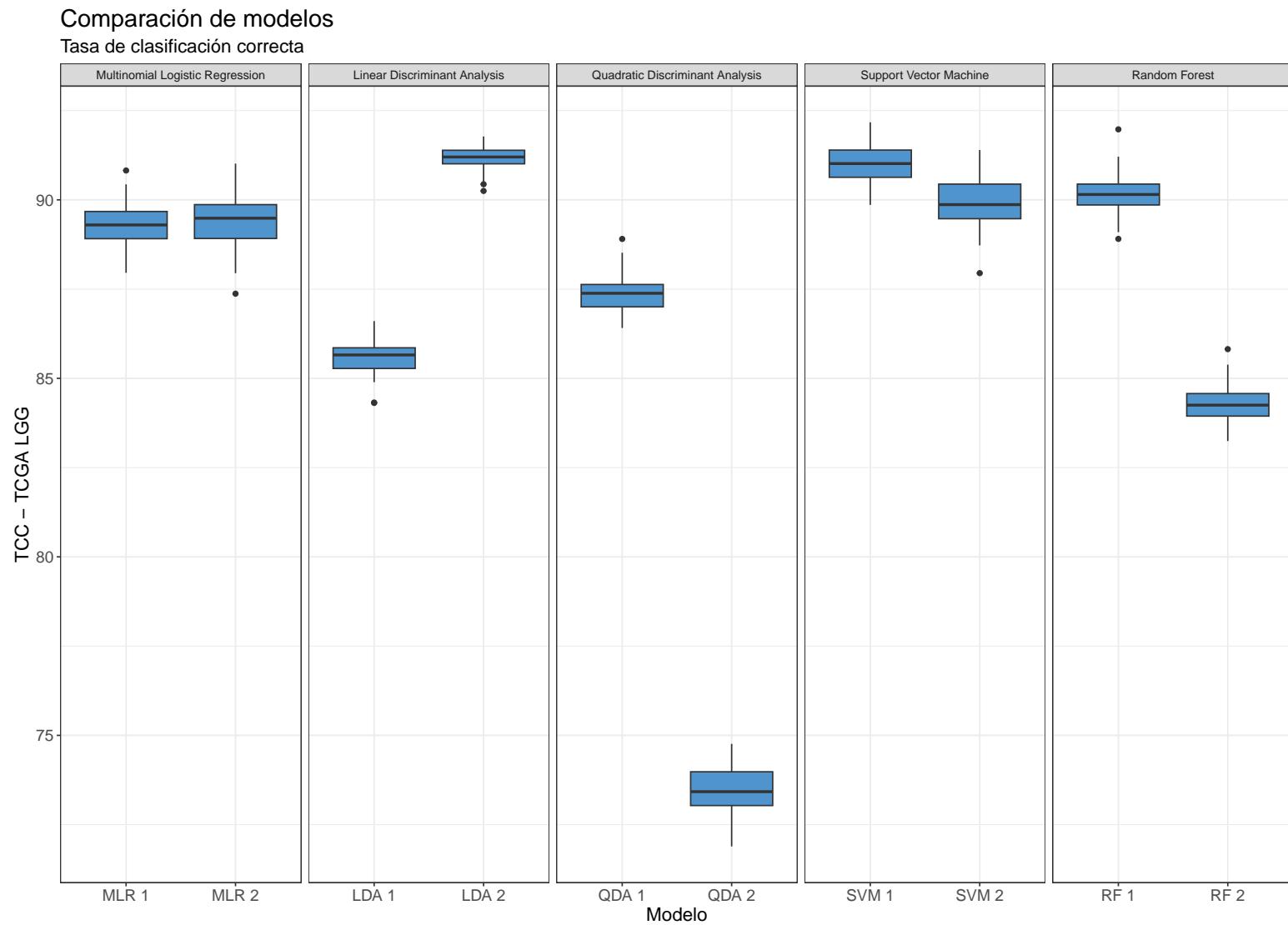


Figura 21: Tasa de Clasificación Correcta para la clase TCGA Lower Grade Glioma en el conjunto $\mathcal{D}_{\text{Test}}$.

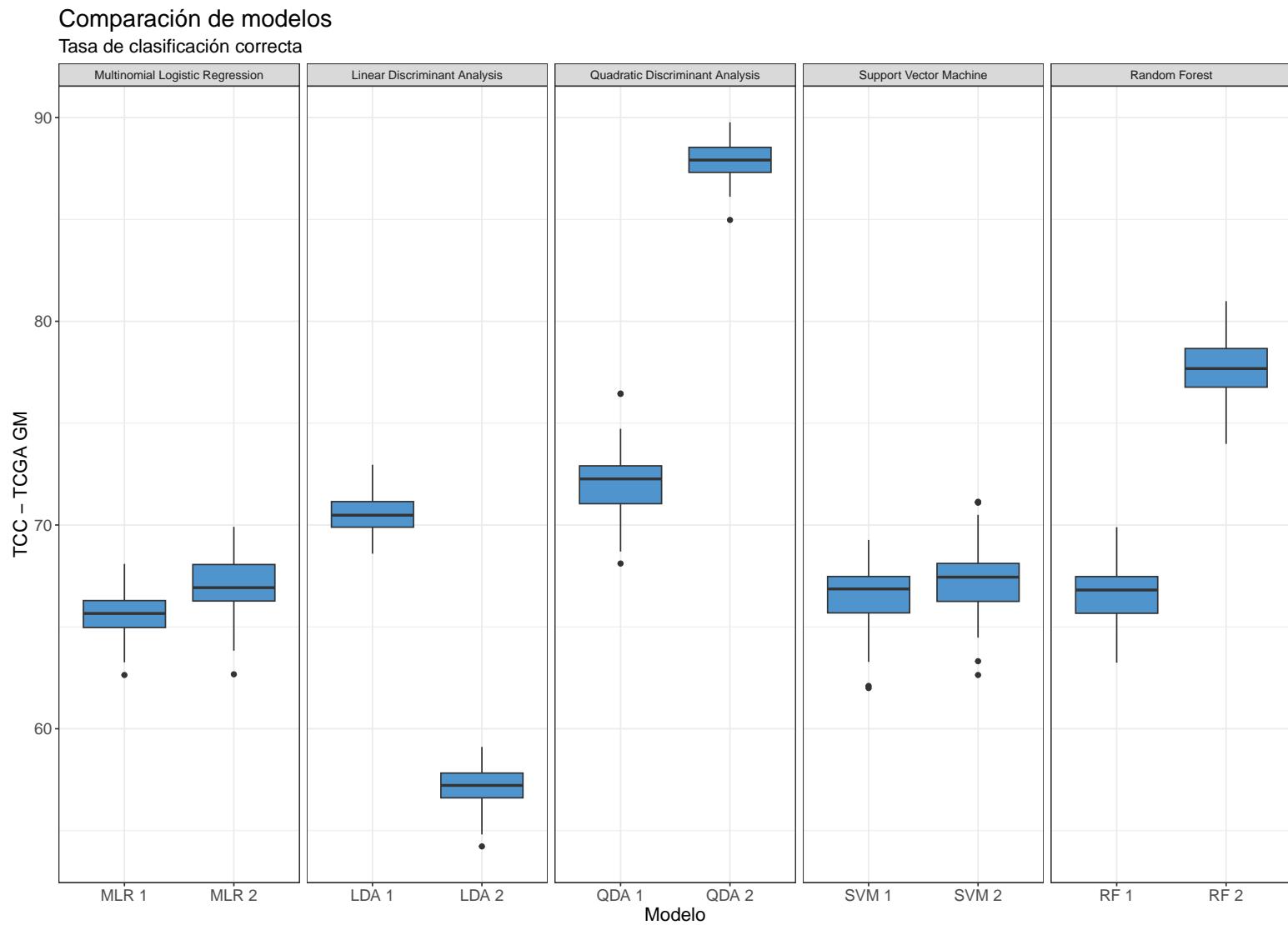


Figura 22: Tasa de Clasificación Correcta para la clase TCGA Glioblastoma Multiforme en el conjunto $\mathcal{D}_{\text{Test}}$.

3.4 ANÁLISIS DE LOS RESULTADOS

Esta sección está dedicada al análisis de los resultados obtenidos en la sección 3.3. Según el Cuadro 11, los modelos SVM 1 y SVM 2 lograron el mejor desempeño global, con una tasa de clasificación correcta global del 86.3 %, seguidos por los modelos MLR 2 (85.5 %) y QDA 1 (85.5 %).

Sin embargo, los modelos RF 2 y SVM 2 destacaron en la clasificación del grupo de personas sanas con 90.8 % y 90.6 %, respectivamente. Además, el modelo QDA 2 (UGGM-QDA) obtuvo el mejor resultado en la clasificación de personas con glioblastoma multiforme (87.8 %), lo que sugiere que la penalización a través del algoritmo *glasso* mejora el poder predictivo en escenarios específicos, particularmente cuando la estructura de correlación entre las variables es relevante.

A partir de los resultados de la sección 3.3, se pueden destacar los siguientes puntos:

- SVM: Ambos modelos SVM muestran el mejor desempeño general, destacándose en el grupo de personas con gliomas de bajo grado (91 % para SVM1 y 89.9 % para SVM 2). Sin embargo, en el grupo de personas con glioblastoma multiforme (con 66.7 % y 67.3 % respectivamente), presentan una caída considerable en su rendimiento, lo que sugiere que, para este subconjunto de datos, las fronteras no lineales del modelo SVM no capturan completamente las características del grupo. Notar que no hay mayores diferencias entre SVM 1 y SVM 2, es decir, no parece que el *tunear* fue relevante.
- QDA y LDA: Los modelos de análisis discriminante cuadrático y lineal (QDA y LDA) muestran una variabilidad en su desempeño. QDA 2 (UGGM-QDA), sobresale en el grupos de personas sanas y el grupo de personas con glioblastoma multiforme. Sin embargo, este mismo modelo presenta el desempeño más bajo en la Tasa de Clasificación Correcta Global (80.3 %) y el grupo de personas con gliomas de bajo grado (73.4 %), lo que indica que su ventaja podría ser más específica a ciertos contextos o subgrupos.
- El modelo LDA 2 muestra una tendencia similar, con un desempeño destacado en el grupo de personas con gliomas de bajo grado (91.1 %), pero con el peor en el grupo de personas con glioblastoma multiforme (57.2 %). Esto podría reflejar que la estructura de correlación utilizada en el modelo tiene efectos positivos en algunos subgrupos, pero puede no ser adecuada para otros.
- Bosques Aleatorios (RF): Los modelos de bosques aleatorios muestran un desempeño consistente, aunque no sobresaliente. RF 2, el cual considera un proceso de *tunear* en sus hiperparámetros, mejora notablemente en el grupo de personas con glioblastoma multiforme (77.7 %) comparado con RF 1 (66.5 %).
- Regresión Logística Multinomial (MLR): Los modelos MLR 1 y MLR 2 también muestran un buen desempeño, especialmente en el grupo de personas sanas y el grupo de personas con gliomas de bajo grado. MLR 2, que incluye interacciones de segundo orden con penalización *ElasticNet*, tiene un ligero mejor rendimiento en general.

En cuanto a los tiempos de ejecución, el Cuadro 13 presenta los valores obtenidos para la estimación del poder predictivo de cada modelo. Como era de esperarse, los modelos que incluyeron un proceso de *tunear* registraron tiempos de ejecución más prolongados. En

particular, el modelo LDA 1 fue el más eficiente, con un tiempo de 6.61 segundos, mientras que MLR 2 tuvo el tiempo de ejecución más alto, alcanzando 5520.24 segundos.

Finalmente, la Figura 23, que muestra las matrices de confusión promediadas obtenidas en el proceso de evaluación, evidencia que los modelos tienden a clasificar erróneamente observaciones del grupo TCGA GM en el grupo TCGA LGG y viceversa. Esto sugiere una confusión recurrente entre estos dos grupos. Por lo tanto, agrupar los grupos TCGA LGG y TCGA GM podría mejorar el rendimiento de los modelos y mejorar las tasas de clasificación.

Modelo	Tiempo de Ejecución (s)
Multinomial Logistic Regression (MLR 1)	30.25
Multinomial Logistic Regression (MLR 2)	5520.24
Linear Discriminant Analysis (LDA 1)	6.61
Linear Discriminant Analysis (LDA 2)	7.46
Quadratic Discriminant Analysis (QDA 1)	15.30
Quadratic Discriminant Analysis (QDA 2)	820.34
Support Vector Machine (SVM 1)	11.36
Support Vector Machine (SVM 2)	1173.68
Random Forest (RF 1)	57.46
Random Forest (RF 2)	2783.61

Cuadro 13: Tiempos de ejecución de los modelos en segundos.



Figura 23: Matrices de confusión promediadas obtenidas durante el proceso de evaluación de los modelos. Se observa una mayor confusión entre los grupos TCGA LGG y TCGA GM, lo que sugiere la posibilidad de mejorar la clasificación combinando estos grupos.

4

CONCLUSIONES

4.1 MODELOS GRÁFICOS PROBABILÍSTICOS

En este trabajo se exploró el uso de modelos gráficos probabilísticos en problemas de clasificación supervisada, con énfasis en la implementación del modelo UGGM-QDA. Este modelo se basa en el análisis de discriminante cuadrático, incorporando una penalización en la estimación de la matriz de concentración mediante el algoritmo *Graphical Lasso*. Esta estrategia permitió mejorar la precisión del modelo en escenarios donde el tamaño de muestra era limitado y la dimensionalidad era alta.

Los resultados mostraron que UGGM-QDA ofrece ventajas en el manejo de estructuras de correlación entre variables, especialmente en el ejercicio con datos simulados, donde la estructura de la matriz de concentración era conocida. No obstante, su desempeño en los datos de cáncer de cerebro fue variable dependiendo del grupo de clasificación, evidenciando dificultades en la diferenciación entre individuos con gliomas de bajo grado y aquellos con gliomas altamente agresivos.

Es importante mencionar que una implementación del modelo UGGM-QDA se encuentra en el paquete NetDA [Chen, 2022]. Sin embargo, en esa versión, el hiperparámetro λ se establece de manera fija en $\lambda = 0.01$, lo que representa una limitación en problemas de clasificación supervisada. La calibración de λ es un proceso clave para optimizar la precisión del modelo, ya que regula el nivel de penalización (ceros) en la estimación de la matriz de concentración. Para abordar esta limitación, en este trabajo implementó el Algoritmo 6, el cual permite evaluar una malla de valores para λ definida por el usuario, utilizando un esquema de validación cruzada para seleccionar el valor óptimo.

A partir de los resultados obtenidos tanto en los datos simulados como en el caso práctico, se concluye que el modelo UGGM-QDA puede ser útil en problemas de clasificación. Sin embargo, existen aspectos que podrían mejorar significativamente su desempeño. Algunos de estos aspectos quedaron fuera del alcance de este trabajo y podrían representar futuros trabajos:

- Calibración del Algoritmo 6:
 - Uso de valores λ diferenciados por clase: Actualmente, la implementación utiliza un único valor de λ para todo el problema de clasificación. Sin embargo, permitir la calibración de λ de manera independiente para cada clase podría mejorar la estimación de la gráfica asociada a cada grupo. Si bien esta modificación incrementaría

significativamente el espacio de búsqueda de hiperparámetros, su implementación podría capturar mejor las diferencias estructurales entre los grupos y optimizar la clasificación en escenarios con relaciones de dependencia heterogéneas.

- Evaluación de diferentes métricas para la calibración de λ : En la implementación actual, la selección de λ se basa exclusivamente en la Tasa de Clasificación Correcta Global. Sin embargo, sería beneficioso permitir la optimización basada en otras métricas, como la Tasa de Clasificación Correcta mínima entre las clases (para equilibrar el desempeño del modelo en todos los grupos) o la calibración dirigida a maximizar la clasificación en una clase específica, lo que podría ser útil en problemas con clases desbalanceadas o de especial interés.
- Incorporación de la calibración de las probabilidades a priori: Tanto en el ejercicio con datos simulados como en el caso práctico, las probabilidades a priori fueron definidas como la frecuencia relativa de cada grupo. No obstante, permitir la calibración de estas probabilidades podría mejorar la precisión del modelo, especialmente en escenarios donde los grupos estén desbalanceados.
- Definición de un modelo UGGM-LDA: Extender el enfoque de UGGM-QDA al análisis de discriminante lineal (UGGM-LDA), incorporando una penalización en la estimación de la matriz de concentración mediante el algoritmo *Graphical Lasso*.
- Desarrollo de un paquete en R: Integrar todas las funcionalidades mencionadas en un paquete dentro de R, facilitando su implementación y replicación. Este paquete incluiría la calibración avanzada de λ , la posibilidad de utilizar valores diferenciados por clase, la evaluación flexible de métricas de desempeño y la calibración de probabilidades a priori.
- Exploración de algoritmos alternativos a *Graphical Lasso*: Investigar otros enfoques para la estimación de la gráfica asociada en modelos gráficos probabilísticos e incorporarlos en un enfoque de clasificación supervisada, por ejemplo, métodos basados en criterios de información como AIC y BIC, como en [Højsgaard et al. \(2012, sección 4.4\)](#) o [Drton and Maathuis \(2017, sección 3\)](#).

4.2 VÍA DE LAS KINURENINAS

En este trabajo, se evaluó el potencial de las expresiones genéticas de las principales enzimas de la vía de las kinureninas como biomarcadores para la clasificación de distintos grupos de individuos. Los resultados obtenidos mostraron que los modelos de clasificación lograron Tasas de Clasificación Correcta Global superiores al 82 %, lo que sugiere que estas variables tienen el potencial de ser utilizadas en el diagnóstico de las siguientes condiciones:

- Personas sanas.
- Personas con gliomas de bajo grado.
- Personas con glioblastoma multiforme, un tipo de tumor cerebral altamente agresivo.

Estos hallazgos abren la posibilidad de desarrollar estudios adicionales y protocolos clínicos que profundicen en el papel de la vía de las kinureninas en el diagnóstico y pronóstico del

cáncer cerebral. En particular, sería relevante analizar si el poder predictivo de estas variables se mantiene cuando se miden en muestras de sangre en lugar de tejido cerebral, lo que permitiría el desarrollo de métodos de diagnóstico menos invasivos.

A pesar de los resultados obtenidos, existen diversos aspectos que podrían optimizar el desempeño de los modelos de clasificación utilizados. Algunas limitaciones identificadas en este trabajo, que representan oportunidades de mejora en futuras investigaciones, incluyen:

- Desbalance en el tamaño de muestra por grupo: Para ciertos modelos, como la regresión logística y los modelos de análisis de discriminante, la diferencia en la cantidad de observaciones por grupo puede sesgar las predicciones hacia la clase mayoritaria. Para abordar este problema, se podrían explorar técnicas de generación de datos sintéticos, como el algoritmo SMOTE [Chawla et al., 2002], que permitirían mejorar la representación de los grupos con menor cantidad de datos y mejorar el rendimiento de los modelos.
- Calibración de hiperparámetros: El desempeño de los modelos de clasificación supervisada depende en gran medida de la configuración de sus hiperparámetros. En este trabajo, la calibración de hiperparámetros se realizó dentro de un conjunto predefinido de valores; sin embargo, el espacio de búsqueda podría ampliarse o refinarse con estrategias de calibración más eficientes.
- Implementación de modelos avanzados: Técnicas más sofisticadas, como XGBoost o redes neuronales profundas, podrían mejorar significativamente la capacidad predictiva del modelo. No obstante, su implementación implica mayores requerimientos computacionales para el usuario y la necesidad de ajustar múltiples hiperparámetros, lo que representa un desafío en términos de costo computacional.
- Implementación de un modelo en producción: Desarrollar un modelo productivo entrenado con la totalidad de los datos disponibles permitiría realizar un análisis exhaustivo de las variables con mayor potencial predictivo. Además, esta implementación facilitaría la interpretación del modelo y su posible integración en entornos clínicos.
- Incorporación de nuevas variables: Ampliar el conjunto de variables utilizadas en los modelos de clasificación podría mejorar la capacidad predictiva. Se recomienda explorar otras expresiones genéticas reportadas en la Tabla 1 de Vázquez Cervantes et al. (2022) y la Figura 6 de Pérez de la Cruz et al. (2023) para evaluar su impacto en la clasificación de los distintos grupos.
- Extensión del modelo a otros tipos de cáncer: La metodología propuesta podría adaptarse para la clasificación de otros tipos de cáncer más allá del cáncer cerebral. En la Figura 2 de Pérez de la Cruz et al. (2023) se muestra una lista de distintos tipos de cáncer con datos disponibles tanto de tejido sano como de tejido tumoral. Un análisis comparativo permitiría evaluar si las mismas variables utilizadas en este trabajo tienen poder predictivo en otros tipos de cáncer y, en caso afirmativo, determinar si su efecto es consistente en sentido y magnitud.

En general, los resultados obtenidos en este trabajo establecen un punto de partida para futuras investigaciones sobre el uso de la vía de las kinureninas en el diagnóstico del cáncer cerebral. Sin embargo, aún quedan diversas áreas de mejora que podrían optimizar el desempeño de los modelos y ampliar su aplicabilidad en un contexto clínico.

BIBLIOGRAFÍA

- Agresti, A. (2015). *Foundations of Linear and Generalized Linear Models*. John Wiley & Sons, Hoboken, NJ.
- Bartz, E., Bartz-Beielstein, T., Zaeffferer, M., and Mersmann, O. (2023). *Hyperparameter Tuning for Machine and Deep Learning with R: A Practical Guide*. Springer Nature.
- Batash, R., Asna, N., Schaffer, P., Francis, N., and Schaffer, M. (2017). Glioblastoma multiforme, diagnosis and treatment; recent literature review. *Curr Med Chem*, 24(27):3002–3009.
- Bray, F., Laversanne, M., Sung, H., Ferlay, J., Siegel, R. L., Soerjomataram, I., and Jemal, A. (2024). Global cancer statistics 2022: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: A Cancer Journal for Clinicians*, 74(3):229–263.
- Bühlmann, P. and van de Geer, S. (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Series in Statistics. Springer.
- Carithers, L. J. and Moore, H. M. (2015). The genotype-tissue expression (gtex) project. *Biopreserv Biobank*, 13(5):307–308.
- Cervantes, G. I. V., Arellano, N. K. O., Ortega, D. R., Ramiro, A. S., Esquivel, D. F. G., Ríos, C., Olvera, B. P., and de la Cruz, V. P. (2017). Role of kynurenine pathway in glioblastoma. In Abreu, G. E. A., editor, *Mechanisms of Neuroinflammation*, chapter 10. IntechOpen, Rijeka.
- Cervenka, I., Agudelo, L. Z., and Ruas, J. L. (2017). Kynurenines: Tryptophan's metabolites in exercise, inflammation, and mental health. *Science*, 357(6349).
- Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.
- Chen, L.-P. (2022). Netda: An r package for network-based discriminant analysis subject to multilabel classes. *Journal of Probability and Statistics*, 2022(1):1041752.
- Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, UK.
- Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*:1695.
- Drton, M. and Maathuis, M. H. (2017). Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4:365–393.
- Drton, M., Maathuis, M. H., Lauritzen, S. L., and Wainwright, M. J., editors (2019). *Handbook of Graphical Models*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. Chapman and Hall/CRC.
- Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

- Friedman, J., Hastie, T., and Tibshirani, R. (2021). *glasso: Graphical Lasso - Estimation of Gaussian Graphical Models*. R package version 1.11.
- Friedman, J., Tibshirani, R., and Hastie, T. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.
- Gao, H. and Jiang, X. (2013). Progress on the diagnosis and evaluation of brain tumors. *Cancer Imaging*, 13(4):466–481.
- Goldman, M. J., Craft, B., Hastie, M., Repečka, K., McDade, F., Kamath, A., Banerjee, A., Luo, Y., Rogers, D., Brooks, A. N., Zhu, J., and Haussler, D. (2020). Visualizing and interpreting cancer genomics data via the xena platform. *Nature Biotechnology*, 38(6):675–678.
- Grandini, M., Bagli, E., and Visani, G. (2020). Metrics for multi-class classification: an overview. *ArXiv*, abs/2008.05756.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2 edition.
- Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC.
- Hoerl, A. and Kennard, R. (2006). Ridge regression. *Encyclopedia of Statistical Sciences*, 8:129–136.
- Højsgaard, S., Edwards, D., and Lauritzen, S. (2012). *Graphical Models with R*. Use R! Springer New York.
- James, G., Witten, D., Hastie, T., and Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R*. Springer, New York, NY, 2nd edition edition.
- Jolliffe, I. T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philos Trans A Math Phys Eng Sci*, 374(2065):20150202.
- Lauritzen, S. L. (1996). *Graphical Models*. Oxford University Press, Oxford, UK.
- Liang, F. and Jia, B. (2023). *Sparse Graphical Modeling for High-Dimensional Data: A Paradigm of Conditional Independence Tests*. Chapman and Hall/CRC.
- Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis*. Academic Press, London, UK.
- McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*. Chapman & Hall/CRC, 2nd edition.
- Meinshausen, N. and Buehlmann, P. (2009). Stability selection.
- Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.-C., and Lin, C.-C. (2022). *e1071: Misc functions of the Department of Statistics, Probability Theory Group (Formerly: E1071)*, TU Wien. R package version 1.7-14.
- Mohammed, S., Dinesan, M., and Ajayakumar, T. (2022). Survival and quality of life analysis in glioblastoma multiforme with adjuvant chemoradiotherapy: a retrospective study. *Rep Pract Oncol Radiother*, 27(6):1026–1036.
- Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press.

- Perez-de-la Cruz, G. and Eslava-Gomez, G. (2016). Discriminant analysis with gaussian graphical tree models. *ASTA Advances in Statistical Analysis*, 100(2):161–187.
- Pérez de la Cruz, G., Pérez de la Cruz, V., Navarro Cossio, J., Vázquez Cervantes, G. I., Salazar, A., Orozco Morales, M., and Pineda, B. (2023). Kynureninase promotes immunosuppression and predicts survival in glioma patients: In silico data analyses of the chinese glioma genome atlas (cgga) and of the cancer genome atlas (tcga). *Pharmaceuticals (Basel)*, 16(3).
- Ravikumar, P., Wainwright, M. J., Raskutti, G., and Yu, B. (2008). High-dimensional covariance estimation by minimizing ℓ_1 -penalized log-determinant divergence.
- Ripley, B. D. and Venables, W. N. (2023). MASS: Support Functions and Datasets for Venables and Ripley's MASS. R package version 7.3-60.
- Scutari, M. and Denis, J.-B. (2021). Bayesian Networks: With Examples in R. Chapman and Hall/CRC, 2nd edition.
- Summers, B. S., Thomas Broome, S., Pang, T. W. R., Mundell, H. D., Koh Belic, N., Tom, N. C., Ng, M. L., Yap, M., Sen, M. K., Sedaghat, S., Weible, M. W., Castorina, A., Lim, C. K., Lovelace, M. D., and Brew, B. J. (2024). A review of the evidence for tryptophan and the kynurenine pathway as a regulator of stem cell niches in health and disease. *Int J Tryptophan Res*, 17:11786469241248287.
- Tay, J. K., Narasimhan, B., and Hastie, T. (2023). Elastic net regularization paths for all generalized linear models. *Journal of Statistical Software*, 106(1):1–31.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.
- Tomczak, K., Czerwińska, P., and Wiznerowicz, M. (2015). The cancer genome atlas (tcga): an immeasurable source of knowledge. *Contemp Oncol (Pozn)*, 19(1A):A68–77.
- Touat, M., Idbaih, A., Sanson, M., and Ligon, K. L. (2017). Glioblastoma targeted therapy: updated approaches from recent biological insights. *Ann Oncol*, 28(7):1457–1472.
- Vázquez Cervantes, G. I., Navarro Cossio, J. Á., Pérez de la Cruz, G., Salazar, A., Pérez de la Cruz, V., and Pineda, B. (2022). Bioinformatic analysis of kynurenine pathway enzymes and their relationship with glioma hallmarks. *Metabolites*, 12(11).
- Venables, W. N. and Ripley, B. D. (2002). Modern Applied Statistics with S. Springer, New York, fourth edition. ISBN 0-387-95457-0.
- Vivian, J., Rao, A. A., Nothaft, F. A., Ketchum, C., Armstrong, J., Novak, A., Pfeil, J., Narkizian, J., Deran, A. D., Musselman-Brown, A., Schmidt, H., Amstutz, P., Craft, B., Goldman, M., Rosenbloom, K., Cline, M., O'Connor, B., Hanna, M., Birger, C., Kent, W. J., Patterson, D. A., Joseph, A. D., Zhu, J., Zaranek, S., Getz, G., Haussler, D., and Paten, B. (2017). Toil enables reproducible, open source, big biomedical data analyses. *Nature Biotechnology*, 35(4):314–316.
- Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305.
- Witten, D. M., Friedman, J. H., and Simon, N. (2011). New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4):892–900.

- Wright, M. N. and Ziegler, A. (2023). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17.
- Yuan, M. and Lin, Y. (2007). Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35.
- Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320.

A

ANEXO 1

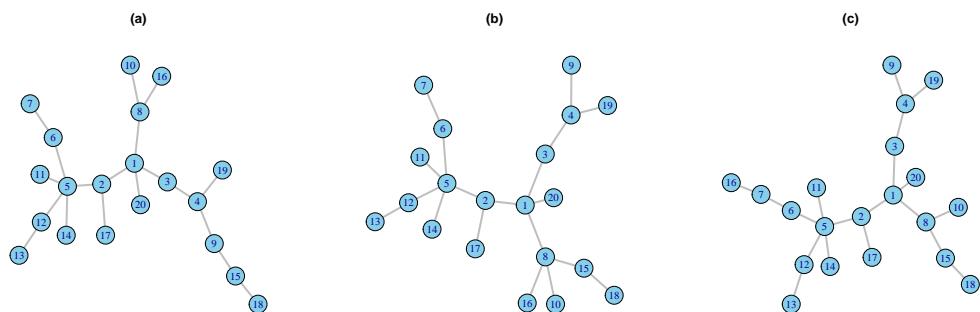


Figura 24: Estructura de la gráfica para la simulación de datos con 20 variables: (a): Clase 1, (b): Clase 2 y (c): Clase 3.

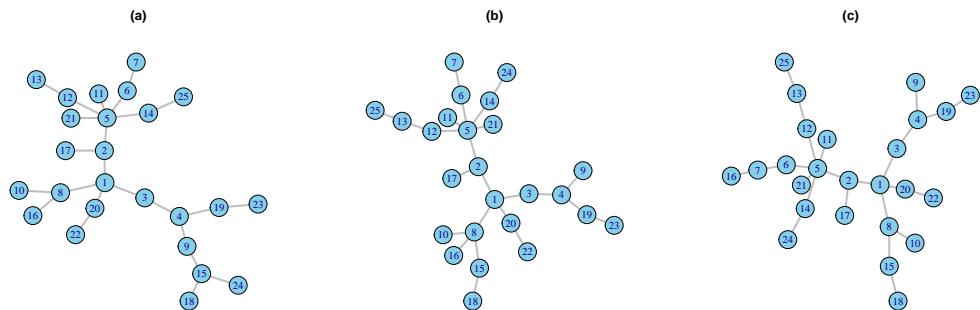


Figura 25: Estructura de la gráfica para la simulación de datos con 25 variables: (a): Clase 1, (b): Clase 2 y (c): Clase 3.

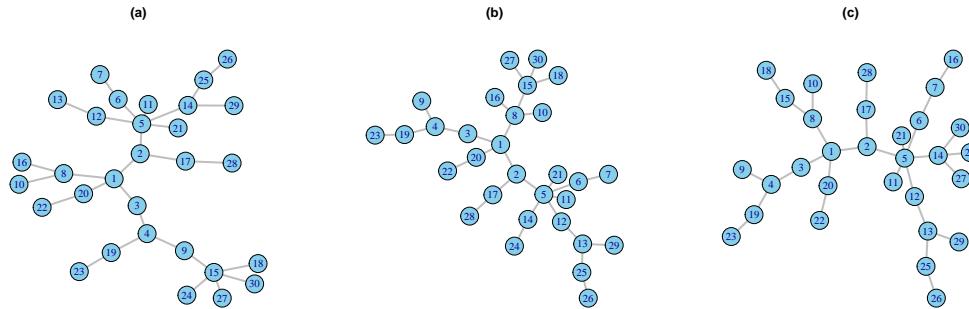


Figura 26: Estructura de la gráfica para la simulación de datos con 30 variables: (a): Clase 1, (b): Clase 2 y (c): Clase 3.

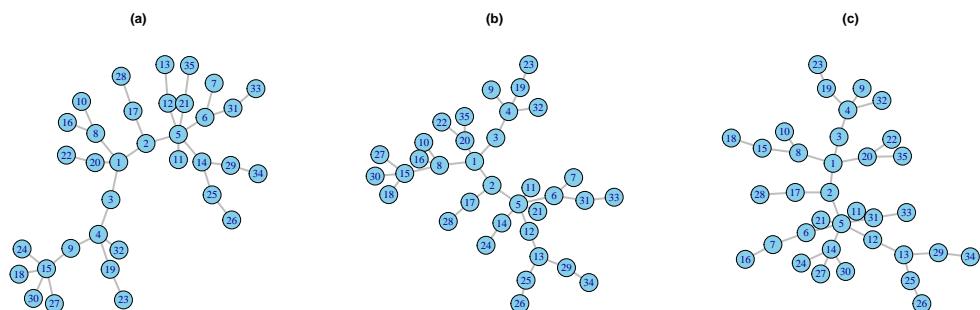


Figura 27: Estructura de la gráfica para la simulación de datos con 35 variables: (a): Clase 1, (b): Clase 2 y (c): Clase 3.

B

ANEXO 2

Para consultar el código que se utilizó para el desarrollo de este trabajo, se puede consultar el siguiente [repositorio](#) en GitHub.

Para visualizar los principales resultados del caso práctico, se puede consultar el siguiente [tablero](#) en shinyapps.

```
1 #####
2 # An lisis Exploratorio #
3 #####
4
5
6
7 rm(list = ls(all.names = TRUE))
8 gc()
9
10 library(readr)
11 library(ppcor)
12 library(dplyr)
13 library(tidyr)
14 library(forcats)
15 library(ggplot2)
16 library(ggcorrplot)
17 library(rstatix)
18 library(ggpubr)
19 library(GGally)
20 library(cowplot)
21 library(gridExtra)
22 library(ggbiplot)
23
24 # Datos -----
25
26 ## Datos en escala log2(norm_count+1)
27 Datos <- read_tsv("../Datos/denseDataOnlyDownload.tsv")
28
29 ## Preprocesamiento de datos originales.
30 Datos <- Datos %>%
31   dplyr::select(-c("sample","samples")) %>%
32   mutate(TCGA_GTEX_main_category = factor(TCGA_GTEX_main_category),
33         detailed_category = factor(detailed_category)) %>%
34   filter(TCGA_GTEX_main_category %in% c("GTEX Brain",
35                                         "TCGA Brain Lower Grade Glioma",
36                                         "TCGA Glioblastoma Multiforme")) %>%
```

```

37 filter(detailed_category %in% c("Brain - Cortex",
38                               "Brain - Anterior Cingulate Cortex (Ba24)",
39                               "Brain - Frontal Cortex (Ba9)",
40                               "Brain Lower Grade Glioma",
41                               "Glioblastoma Multiforme")) %>%
42 mutate(TCGA_GTEX_main_category = fct_relevel(droplevels(TCGA_GTEX_main_category),
43                                               c("GTEX Brain",
44                                                 "TCGA Brain Lower Grade Glioma",
45                                                 "TCGA Glioblastoma Multiforme"))) %>%
46 dplyr::select(-c("_gender", "detailed_category")) %>%
47 mutate(Clase = factor(case_when(TCGA_GTEX_main_category == "GTEX Brain" ~
48                           "GTEX_Brain",
49                           TCGA_GTEX_main_category == "TCGA Brain Lower Grade
50                           Glioma" ~ "TCGA_LGG",
51                           TCGA_GTEX_main_category == "TCGA Glioblastoma
52                           Multiforme" ~ "TCGA_GM")))) %>%
53 dplyr::select(-TCGA_GTEX_main_category)
54
55 ## write.csv(Datos, file = "Datos.csv", row.names = FALSE)
56
57 # Correlacion de Pearson -----
58
59 ## Clase: Brain_Cortex: * Brain - Cortex
60 ##                               * Brain - Anterior Cingulate Cortex (Ba24)
61 ##                               * Brain - Frontal Cortex (Ba9)
62 Pearson.GTEX_Brain <- Datos %>%
63   filter(Clase == "GTEX_Brain") %>%
64   dplyr::select(-Clase) %>%
65   pcor(method = "pearson")
66
67 Pearson.GTEX_Brain <- Pearson.GTEX_Brain$estimate %>%
68   round(digits = 2) %>%
69   data.frame()
70
71 ggcorrplot(Pearson.GTEX_Brain,
72             colors = c("#6D9EC1", "white", "#E46726"),
73             type = "lower",
74             lab = TRUE,
75             lab_size = 5,
76             lab_col = "black") +
77   labs(title = "Matriz de correlaciones parciales",
78        subtitle = "Grupo GTEX Brain",
79        x = "",
80        y = "") +
81   theme_bw() +
82   theme(plot.title = element_text(size = 18),
83         plot.subtitle = element_text(size = 14),
84         axis.title.x = element_text(size = 14),
85         axis.title.y = element_text(size = 14),
86         axis.text.x = element_text(size = 12),
87         axis.text.y = element_text(size = 12))
88
89 ## Clase: TCGA Lower Grade Glioma
90 Pearson.TCGA_LGG <- Datos %>%

```

```
89 filter(Clase == "TCGA_LGG") %>%
90 dplyr::select(-Clase) %>%
91 pcor(method = "pearson")
92
93 Pearson.TCGA_LGG <- Pearson.TCGA_LGG$estimate %>%
94 round(digits = 2) %>%
95 data.frame()
96
97 ggcorrplot(Pearson.TCGA_LGG,
98             colors = c("#6D9EC1", "white", "#E46726"),
99             type = "lower",
100            lab = TRUE,
101            lab_size = 5,
102            lab_col = "black") +
103 labs(title = "Matriz de correlaciones parciales",
104       subtitle = "Grupo TCGA LGG",
105       x = "",
106       y = "") +
107 theme_bw() +
108 theme(plot.title = element_text(size = 18),
109       plot.subtitle = element_text(size = 14),
110       axis.title.x = element_text(size = 14),
111       axis.title.y = element_text(size = 14),
112       axis.text.x = element_text(size = 12),
113       axis.text.y = element_text(size = 12))
114
115 ## Clase: TCGA Glioblastoma Multiforme
116 Pearson.TCGA_GM <- Datos %>%
117 filter(Clase == "TCGA_GM") %>%
118 dplyr::select(-Clase) %>%
119 pcor(method = "pearson")
120
121 Pearson.TCGA_GM <- Pearson.TCGA_GM$estimate %>%
122 round(digits = 2) %>%
123 data.frame()
124
125 ggcorrplot(Pearson.TCGA_GM,
126             colors = c("#6D9EC1", "white", "#E46726"),
127             type = "lower",
128             lab = TRUE,
129             lab_size = 5,
130             lab_col = "black") +
131 labs(title = "Matriz de correlaciones parciales",
132       subtitle = "Grupo TCGA GM",
133       x = "",
134       y = "") +
135 theme_bw() +
136 theme(plot.title = element_text(size = 18),
137       plot.subtitle = element_text(size = 14),
138       axis.title.x = element_text(size = 14),
139       axis.title.y = element_text(size = 14),
140       axis.text.x = element_text(size = 12),
141       axis.text.y = element_text(size = 12))
142
143
```

```

144 # Diagrama de dispersión -----
145
146 Datos <- Datos %>%
147   mutate(Clase = factor(recode(Clase,
148     "GTEX_Brain" = "GTEX Brain",
149     "TCGA_LGG" = "TCGA LGG",
150     "TCGA_GM" = "TCGA GM"),
151     levels = c("GTEX Brain", "TCGA LGG", "TCGA GM")))
152
153 Aux1 <- ggpairs(Datos,
154   mapping = aes(color = Clase),
155   columns = 1:11,
156   lower = list(continuous = "blank"),
157   diag = list(continuous = "blank"),
158   upper = list(continuous = wrap("points", alpha = 0.7, size = 0.5)),
159   legend = 1) +
160   scale_color_brewer(palette = "Set2", name = "Clase") +
161   labs(title = "Diagramas de dispersión",
162     subtitle = "Principales enzimas involucradas en la vía de las Kinureninas") +
163   theme_bw() +
164   theme(legend.position = "none",
165     plot.title = element_text(size = 18),
166     plot.subtitle = element_text(size = 14),
167     axis.title.x = element_text(size = 14),
168     axis.title.y = element_text(size = 14),
169     axis.text.x = element_text(size = 12),
170     axis.text.y = element_text(size = 12))
171
172 Aux2 <- ggplot(Datos, aes(x = 1, y = 1, color = Clase)) +
173   geom_point() +
174   scale_color_brewer(palette = "Set2", name = "Clase") +
175   theme_minimal() +
176   theme(legend.position = "right",
177     axis.title = element_blank(),
178     axis.text = element_blank(),
179     axis.ticks = element_blank(),
180     panel.grid = element_blank())
181
182 legenda <- get_legend(Aux2)
183 Diagrama <- ggmatrix_gtable(Aux1)
184
185 plot_grid(Diagrama, legenda, rel_widths = c(6, 1))
186
187
188 # Componentes principales -----
189
190 pca <- prcomp(Datos[, -which(names(Datos) == "Clase")],
191   center = TRUE,
192   scale. = FALSE)
193
194 var_exp <- summary(pca)$importance[2, 1:3] * 100
195
196 ## PC1 v.s PC2
197 ggbiplot(pca, group = Datos$Clase,
198   choices = c(1, 2),

```

```

199     ellipse = TRUE,
200     ellipse.alpha = 0.0) +
201 theme_bw() +
202 labs(title = "Componentes Principales",
203       subtitle = "An lisis de la varianza explicada por los componentes",
204       x = paste0("PC1 (", round(var_exp[1], 2), "%)"),
205       y = paste0("PC2 (", round(var_exp[2], 2), "%)")) +
206 scale_color_brewer(palette = "Set2", name = "Clase") +
207 guides(fill = "none",
208        color = guide_legend(title = "Clase")) +
209 theme(plot.title = element_text(size = 18),
210       plot.subtitle = element_text(size = 14),
211       axis.title.x = element_text(size = 14),
212       axis.title.y = element_text(size = 14),
213       axis.text.x = element_text(size = 12),
214       axis.text.y = element_text(size = 12)) +
215 xlim(-4, 4) +
216 ylim(-4, 4)
217
218
219 ## PC1 v.s PC3
220
221 ggbiplots(pca, group = Datos$Clase,
222             choices = c(1, 3),
223             ellipse = TRUE,
224             ellipse.alpha = 0.05) +
225 theme_bw() +
226 labs(title = "Componentes Principales",
227       subtitle = "An lisis de la varianza explicada por los componentes",
228       x = paste0("PC1 (", round(var_exp[1], 2), "%)"),
229       y = paste0("PC3 (", round(var_exp[3], 2), "%)")) +
230 scale_color_brewer(palette = "Set2", name = "Clase") +
231 guides(fill = "none",
232        color = guide_legend(title = "Clase")) +
233 theme(plot.title = element_text(size = 18),
234       plot.subtitle = element_text(size = 14),
235       axis.title.x = element_text(size = 14),
236       axis.title.y = element_text(size = 14),
237       axis.text.x = element_text(size = 12),
238       axis.text.y = element_text(size = 12)) +
239 xlim(-4, 4) +
240 ylim(-4, 4)
241
242
243 ## PC2 v.s PC3
244
245 ggbiplots(pca, group = Datos$Clase,
246             choices = c(2, 3),
247             ellipse = TRUE,
248             ellipse.alpha = 0.05) +
249 theme_bw() +
250 labs(title = "Componentes Principales",
251       subtitle = "An lisis de la varianza explicada por los componentes",
252       x = paste0("PC2 (", round(var_exp[2], 2), "%)"),
253       y = paste0("PC3 (", round(var_exp[3], 2), "%)")) +

```

```

254 scale_color_brewer(palette = "Set2", name = "Clase") +
255   guides(fill = "none",
256         color = guide_legend(title = "Clase")) +
257   theme_bw() +
258   theme(plot.title = element_text(size = 18),
259         plot.subtitle = element_text(size = 14),
260         axis.title.x = element_text(size = 14),
261         axis.title.y = element_text(size = 14),
262         axis.text.x = element_text(size = 12),
263         axis.text.y = element_text(size = 12)) +
264   xlim(-4, 4) +
265   ylim(-4, 4)
266
267
268 # Pruebas Kruskal-Wallis y Wilcoxon -----
269
270 ## Lista de resultados
271 dunn_results <- list()
272 kw <- data.frame()
273
274 ## Comparaciones KW y Dunn
275 for (var in names(Datos)[1:11]) {
276
277   formula <- as.formula(paste(var, "~ Clase"))
278
279   # Prueba de Kruskal-Wallis
280   kw_test <- kruskal_test(Datos, formula = formula)
281
282   kw <- rbind(kw, data.frame(variable = var,
283                               p.value = kw_test$p))
284
285   # Prueba de Dunn con corrección Bonferroni
286   dunn_test <- Datos %>%
287     dunn_test(formula = formula, p.adjust.method = "bonferroni")
288
289   dunn_test$variable <- var
290
291   # Ajustar posiciones para etiquetas de p-values
292   max_y <- max(Datos[[var]], na.rm = TRUE)
293   step <- 0.1 * max_y
294
295   dunn_test <- dunn_test %>%
296     mutate(y.position = seq(max_y + step, by = step, length.out = nrow(dunn_test)),
297           group1 = recode(group1,
298                           "GTEX_Brain" = "GTEX Brain",
299                           "TCGA_LGG" = "TCGA LGG",
300                           "TCGA_GM" = "TCGA GM"),
301           group2 = recode(group2,
302                           "GTEX_Brain" = "GTEX Brain",
303                           "TCGA_LGG" = "TCGA LGG",
304                           "TCGA_GM" = "TCGA GM"))
305
306   dunn_results[[var]] <- dunn_test
307 }
308

```

```

309 | dunn_df <- do.call(rbind, dunn_results)
310 |
311 | # Transformar los datos a formato largo
312 | Datos_Long <- Datos %>%
313 |   pivot_longer(cols = names(Datos)[1:11],
314 |                 names_to = "variable",
315 |                 values_to = "value")
316 |
317 | ggplot(Datos_Long,
318 |         aes(x = Clase, y = value)) +
319 |   geom_jitter(aes(color = Clase),
320 |               width = 0.2,
321 |               alpha = 0.5) +
322 |   geom_boxplot(aes(fill = Clase),
323 |                 alpha = 0.6,
324 |                 outlier.shape = NA) +
325 |   stat_pvalue_manual(dunn_df,
326 |                       label = "p.adj.signif",
327 |                       hide.ns = FALSE,
328 |                       size = 3) +
329 |   facet_wrap(~variable, scales = "free_y") +
330 |   scale_fill_brewer(palette = "Set2") +
331 |   scale_color_brewer(palette = "Set2") +
332 |   labs(x = "Clase",
333 |         y = "log2(norm count + 1)",
334 |         title = "Kruskal-Wallis Test",
335 |         subtitle = "Dunn's Test | Post Hoc") +
336 |   theme_bw() +
337 |   theme(legend.position = "none") +
338 |   geom_text(data = kw,
339 |             aes(x = 1.2, y = Inf, label = paste("K-W p =", p.value)),
340 |             vjust = 1.5, hjust = 0.5, size = 3, inherit.aes = FALSE, color =
"steelblue4")

```

```

1 #####
2 # Preprocesamiento | 3 Clases #
3 #####
4
5
6 rm(list = ls(all.names = TRUE))
7 gc()
8
9 library(readr)
10 library(dplyr)
11 library(forcats)
12 library(rsample)
13
14 # Datos -----
15
16 ## Lectura de datos originales.
17 Datos <- read_tsv("../Datos/denseDataOnlyDownload.tsv")
18
19 ## Pre-procesamiento de datos originales.
20 Datos <- Datos %>% select(-c("sample", "samples")) %>%
21   mutate(TCGA_GTEX_main_category = factor(TCGA_GTEX_main_category),

```

```

22     detailed_category = factor(detailed_category)) %>%
23   filter(TCGA_GTEX_main_category %in% c("GTEX Brain",
24         "TCGA Brain Lower Grade Glioma",
25         "TCGA Glioblastoma Multiforme")) %>%
26   filter(detailed_category %in% c("Brain - Cortex",
27         "Brain - Anterior Cingulate Cortex (Ba24)",
28         "Brain - Frontal Cortex (Ba9)",
29         "Brain Lower Grade Glioma",
30         "Glioblastoma Multiforme")) %>%
31   mutate(TCGA_GTEX_main_category = fct_relevel(droplevels(TCGA_GTEX_main_category),
32         c("GTEX Brain",
33             "TCGA Brain Lower Grade Glioma",
34             "TCGA Glioblastoma Multiforme"))) %>%
35   select(-c("_gender", "detailed_category")) %>%
36   rename(y = TCGA_GTEX_main_category) %>%
37   mutate(Clase = factor(case_when(y == "GTEX Brain" ~ "GTEX_B",
38         y == "TCGA Brain Lower Grade Glioma" ~ "TCGA_BLGG",
39         y == "TCGA Glioblastoma Multiforme" ~ "TCGA_GM")))
40   %>%
41   select(-y)
42
43 summary(Datos)
44
45 ## Remuestreo de datos en conjuntos de entrenamiento (Train) y evaluacion (Test)
46 ## V-K Cross-validation (V = 20, K = 5).
47 set.seed(1234)
48 Folds <- vfold_cv(data = Datos,
49   v = 5,
50   repeats = 100,
51   strata = Clase)
52
53 rm(Datos)
54 save.image("Folds.RData")

```

```

1 #####
2 ##### Undirected Gaussian Graphical Model - Quadratic Discriminant Analysis #
3 # UGGM_QDA
4 #####
5
6
7 UGGM_QDA <- function(formula, data, rho, prior = NULL) {
8
9   etiqueta <- as.character(formula[[2]])
10
11  if (!inherits(formula, "formula")) {
12    stop("El argumento 'formula' debe ser tipo formula")
13  }
14
15  if (!is.data.frame(data)) {
16    stop("El argumento 'datos' debe ser un data.frame.")
17  }
18
19  # Si el lado derecho de la formula es '.', usar todas las otras columnas como
20  # predictores

```

```

20 if (length(formula[[3]]) == 1 && as.character(formula[[3]]) == ".") {
21   predictores <- setdiff(names(data), etiqueta)
22   formula <- reformulate(predictores, response = etiqueta)
23 }
24
25 # Vector de etiquetas
26 y <- as.factor(data[[etiqueta]])
27
28 # Matriz de diseño sin intercepto
29 X <- model.matrix(formula, data)[,-1]
30
31 # Niveles nicos de la variable y
32 niveles.y <- unique(y)
33
34 # Medias de cada variable en X agrupadas por las clases de y
35 mu <- lapply(niveles.y, function(nivel) {
36   colMeans(X[y == nivel, , drop = FALSE])
37 })
38
39 # Probabilidades a priori por cada clase
40 if (is.null(prior)) {
41   pi <- as.list(table(y)/length(y))
42 } else {
43   if (length(prior) != length(niveles.y)) {
44     stop("El número de probabilidades a priori no coincide con el número de clases
45       en la variable de y")
46   }
47   pi <- as.list(prior/sum(prior))
48 }
49
50 # Matriz de covarianza por cada clase
51 Sigma <- lapply(niveles.y, function(nivel) {
52   as.matrix(var(X[y == nivel, , drop = FALSE]))
53 })
54
55 # Matriz de precisión por cada clase
56 Omega <- lapply(Sigma, function(sigma) {
57   glasso::glasso(sigma, rho = rho)$wi
58 })
59
60 # Asignar nombres de las clases
61 names(mu) <- niveles.y
62
63 names(pi) <- niveles.y
64
65 names(Sigma) <- niveles.y
66
67 names(Omega) <- niveles.y
68
69 # Estimaciones
70 return(list(mu = mu, pi = pi, sigma = Sigma, omega = Omega))
71 }
72 predict.UGGM_QDA <- function(object, newdata){
73

```

```

74 NewData <- as.matrix(newdata)
75 mu <- object$mu
76 pi <- object$pi
77 Sigma <- object$sigma
78 Omega <- object$omega
79
80 # Número de clases y observaciones
81 num.clases <- length(mu)
82 num.obs <- nrow(NewData)
83
84 # Probabilidades de pertenecer a cada clase para cada observación
85 probs <- matrix(NA, nrow = num.obs, ncol = num.clases)
86
87 for (j in 1:num.clases) {
88
89   # Función discriminante para la clase j
90   delta <- log(pi[[j]]) - 0.5 * log(det(solve(Omega[[j]]))) - 0.5 * rowSums((NewData
91     - rep(mu[[j]], each = num.obs)) %*% Omega[[j]] * (NewData - rep(mu[[j]], each
92     = num.obs)))
93
94   # Matriz de probabilidades
95   probs[, j] <- delta
96
97 }
98
99 # Probabilidades normalizadas
100 probs <- exp(probs)
101 probs <- data.frame(probs = probs/rowSums(probs))
102
103 # Nombre de las clases
104 nombres.clases <- names(pi)
105
106 # Clase predicha para cada observación con los nombres de las clases
107 clase.predicha <- data.frame(yhat = apply(probs, 1, function(x)
108   nombres.clases[which.max(x)]))
109
110 colnames(probs) <- nombres.clases
111
112 # Probabilidad estimada y clase predicha
113 return(list(prob = probs,
114             clase = clase.predicha))
115
116 }
117 tune.rho <- function(formula, data, rhos, prior = NULL, nfolds = 5) {
118
119   etiqueta <- as.character(formula[2])
120
121   # Resultados de la validación cruzada
122   cv.results <- data.frame(rho = rhos,
123                             accuracy = rep(NA, length(rhos)),
124                             std.dev = rep(NA, length(rhos)))
125
126   # Subconjuntos (folds) de entrenamiento y validación
127   set.seed(123)

```

```

126 folds <- caret::createFolds(data[[etiqueta]], k = nfolds)
127
128 for (i in 1:length(rhos)) {
129
130   accuracy.val <- numeric(nfolds)
131
132   # Realizar validaci n cruzada
133   for (fold_id in folds) {
134
135     MC <- NULL
136
137     # Conjunto de entrenamiento
138     train <- data[-fold_id, ]
139
140     # Conjunto de validaci n
141     val <- data[fold_id, ]
142
143     # Modelo
144     model <- UGGM_QDA(formula, prior = prior, train, rhos[i])
145
146     # Predicciones
147     predictions <- predict.UGGM_QDA(model, val[, -which(names(data) == etiqueta)])
148
149     MC <- table(val[[etiqueta]], predictions$clase$yhat)
150
151     # Precisi n
152     accuracy.val[fold_id] <- sum(diag(MC))/sum(MC)
153
154   }
155
156   # Precisi n promedio
157   avg.accuracy <- mean(accuracy.val)
158
159   # Desviaci n est ndar de la precisi n
160   std.dev <- sd(accuracy.val)
161
162   # Precisi n promedio
163   cv.results$accuracy[i] <- avg.accuracy
164   cv.results$std.dev[i] <- std.dev
165 }
166
167 # Valor de rho que maximiza la precisi n
168 best.rho <- cv.results$rho[which.max(cv.results$accuracy)]
169
170 # Resultados
171 return(list(cv.results = cv.results, best.rho = best.rho))
172 }
173
174 predict.theoricQDA <- function(object, newdata){
175
176   NewData <- as.matrix(newdata)
177   mu <- object$mu
178   pi <- object$pi
179   Sigma <- object$sigma
180

```

```

181 num.clases <- length(mu)
182 num.obs <- nrow(NewData)
183
184 probs <- matrix(NA, nrow = num.obs, ncol = num.clases)
185
186 for (j in 1:num.clases) {
187
188   delta <- log(pi[[j]]) - 0.5 * log(det(Sigma[[j]])) - 0.5 * rowSums((NewData -
189     rep(mu[[j]], each = num.obs)) %*% solve(Sigma[[j]]) * (NewData - rep(mu[[j]],
190     each = num.obs)))
191
192   probs[, j] <- delta
193
194 }
195
196 probs <- exp(probs)
197 probs <- data.frame(probs = probs/rowSums(probs))
198
199 nombres.clases <- names(pi)
200
201 clase.pred <- data.frame(yhat = apply(probs, 1, function(x)
202   nombres.clases[which.max(x)]))
203
204 colnames(probs) <- nombres.clases
205
206 return(list(prob = probs,
207             clase = clase.pred))
208
209 }
```

```

1 #####
2 # Multinomial Logistic Regression #
3 # MLR Elastic-Net
4 #####
5
6
7
8 library(tibble)
9 library(plyr)
10 library(dplyr)
11 library(tidyr)
12 library(forcats)
13
14 library(rsample)
15
16 library(purrr)
17 library(furrr)
18
19 library(VGAM)
20 library(glmnet)
21
22 library(ggplot2)
23
24 library(readr)
25
```

```
26 library(tictoc)
27
28
29 # Funciones Auxiliares -----
30
31 source("../FuncionesAuxiliares.R")
32
33
34 # Conjuntos Train y Test -----
35
36 load("../Folds.RData")
37
38
39 # Algoritmos de clasificacion -----
40
41 MLR.1 <- function(Train, Test) {
42
43   XTrain <- model.matrix(Clase~.,
44                         data = Train)[,-1]
45   YTrain <- Train$Clase
46
47   XTest <- model.matrix(Clase~.,
48                         data = Test)[,-1]
49
50   MLR <- glmnet(x = XTrain,
51                  y = YTrain,
52                  family = "multinomial",
53                  type.multinomial = "ungrouped",
54                  lambda = 0)
55
56   PredTrain <- predict(object = MLR,
57                         newx = XTrain,
58                         type = "class")
59
60   PredTest <- predict(object = MLR,
61                         newx = XTest,
62                         type = "class")
63
64   MC.Train <- table(Train$Clase, PredTrain)
65   MC.Test <- table(Test$Clase, PredTest)
66
67   return(list(MC.Train = MC.Train,
68               MC.Test = MC.Test))
69
70 }
71
72 MLR.2 <- function(Train, Test) {
73
74   XTrain <- model.matrix(Clase~.^2,
75                         data = Train)[,-1]
76   YTrain <- Train$Clase
77
78   XTest <- model.matrix(Clase~.^2,
79                         data = Test)[,-1]
```

```

81 alphas <- seq(0, 1, by = 0.5)
82
83 mejor_modelo <- NULL
84 mejor_alpha <- NULL
85 mejor_lambda <- NULL
86 mejor_error <- Inf
87
88 for (a in alphas) {
89   modelo_tuneo <- cv.glmnet(
90     x = XTrain,
91     y = YTrain,
92     nfolds = 3,
93     alpha = a,
94     type.measure = "class",
95     family = "multinomial",
96     type.multinomial = "ungrouped"
97   )
98
99   error_actual <- min(modelo_tuneo$cvm)
100  lambda_actual <- modelo_tuneo$lambda.min
101
102  if (error_actual < mejor_error) {
103    mejor_error <- error_actual
104    mejor_modelo <- modelo_tuneo
105    mejor_alpha <- a
106    mejor_lambda <- lambda_actual
107  }
108 }
109
110 PredTrain <- predict(object = mejor_modelo,
111                       newx = XTrain,
112                       type = "class",
113                       s = mejor_lambda)
114
115 PredTest <- predict(object = mejor_modelo,
116                       newx = XTest,
117                       type = "class",
118                       s = mejor_lambda)
119
120 MC.Train <- table(Train$Clase, PredTrain)
121 MC.Test <- table(Test$Clase, PredTest)
122
123 return(list(MC.Train = MC.Train,
124             MC.Test = MC.Test))
125 }
126
127
128 # Resultados -----
129
130 tic()
131 M1.MLR.1 <- Evaluacion1(Metodo = "MLR.1",
132                           workers = availableCores())
133 TE_MLR.1 <- toc(log = TRUE)
134
135 tic()

```

```

136 M1.MLR.2 <- Evaluacion1(Metodo = "MLR.2",
137                               workers = availableCores())
138 TE_MLR.2 <- toc(log = TRUE)
139
140
141 # Gráficas -----
142
143 G.MLR.1 <- M1.MLR.1[["Global"]] %>%
144   mutate(Modelo = "Multinomial Logistic Regression",
145         Nombre = "MLR 1")
146
147 G.MLR.2 <- M1.MLR.2[["Global"]] %>%
148   mutate(Modelo = "Multinomial Logistic Regression",
149         Nombre = "MLR 2")
150
151 M1 <- bind_rows(G.MLR.1, G.MLR.2) %>%
152   mutate(Modelo = as.factor(Modelo),
153         Nombre = as.factor(Nombre))
154
155 G.TE_MLR.1 <- data.frame(TE = TE_MLR.1[["callback_msg"]]) %>%
156   mutate(Modelo = "Multinomial Logistic Regression",
157         Nombre = "MLR 1")
158
159 G.TE_MLR.2 <- data.frame(TE = TE_MLR.2[["callback_msg"]]) %>%
160   mutate(Modelo = "Multinomial Logistic Regression",
161         Nombre = "MLR 2")
162
163 TE_M2 <- bind_rows(G.TE_MLR.1, G.TE_MLR.2) %>%
164   mutate(Modelo = as.factor(Modelo),
165         Nombre = as.factor(Nombre))
166
167 write.csv(x = M1,
168            file = "Modelo1.csv",
169            row.names = FALSE)
170
171 write.csv(x = TE_M2,
172            file = "TE_Modelo1.csv",
173            row.names = FALSE)
174
175
176 # Matriz de confusión promediada -----
177
178 MC.M1.MLR.1 <- M1.MLR.1[["MatricesConfusion"]] %>%
179   transpose()
180
181 MC.M1.MLR.2 <- M1.MLR.2[["MatricesConfusion"]] %>%
182   transpose()
183
184 MC.M1.MLR.1.PROM <- Reduce("+", MC.M1.MLR.1$MC.Test) / length(MC.M1.MLR.1$MC.Test)
185 MC.M1.MLR.1.PROM <- round(t(apply(MC.M1.MLR.1.PROM, 1, function(x) x / sum(x) *
186   100)),2)
187
188 MC.M1.MLR.2.PROM <- Reduce("+", MC.M1.MLR.2$MC.Test) / length(MC.M1.MLR.2$MC.Test)
189 MC.M1.MLR.2.PROM <- round(t(apply(MC.M1.MLR.2.PROM, 1, function(x) x / sum(x) *
190   100)),2)

```



```

244     Prediccion = fct_relevel(droplevels(Prediccion),
245                               c("GTEX Brain",
246                                 "TCGA LGG",
247                                 "TCGA GM")))
248
249 M1.MC <- bind_rows(MC.M1.MLR.1.DF, MC.M1.MLR.2.DF)
250
251 write.csv(x = M1.MC,
252            file = "MC1.csv",
253            row.names = FALSE)

```

```

1 #####
2 # Linear Discriminant Analysis #
3 # LDA                         #
4 #####
5
6
7 library(tibble)
8 library(plyr)
9 library(dplyr)
10 library(tidyr)
11 library(forcats)
12
13 library(rsample)
14
15 library(purrr)
16 library(furrr)
17
18 library(MASS)
19 library(NetDA)
20
21 library(ggplot2)
22
23 library(readr)
24
25 library(tictoc)
26
27
28
29 # Funciones Auxiliares -----
30
31 source("../FuncionesAuxiliares.R")
32
33
34 # Conjuntos Train y Test -----
35
36 load("../Folds.RData")
37
38
39 # Esquemas de clasificacion -----
40
41 LDA.1 <- function(Train, Test) {
42
43   LDA <- lda(formula = Clase~.,
44               data = Train)

```

```
45 PredTrain <- predict(object = LDA,
46                         newdata = Train)$class
47
48 PredTest <- predict(object = LDA,
49                         newdata = Test)$class
50
51 MC.Train <- table(Train$Clase, PredTrain)
52 MC.Test <- table(Test$Clase, PredTest)
53
54 return(list(MC.Train = MC.Train,
55             MC.Test = MC.Test))
56
57 }
58
59
60 LDA.2 <- function(Train, Test) {
61
62   X_Train <- Train %>%
63     dplyr::select(-Clase) %>%
64     as.matrix()
65
66   X_Test <- Test %>%
67     dplyr::select(-Clase) %>%
68     as.matrix()
69
70   Y_Train <- Train %>%
71     mutate(Clase = case_when(
72       Clase == "GTEX_B" ~ 1,
73       Clase == "TCGA_BLGG" ~ 2,
74       Clase == "TCGA_GM" ~ 3)) %>%
75     dplyr::select(Clase) %>%
76     as.matrix()
77
78   Y_Test <- Test %>%
79     mutate(Clase = case_when(
80       Clase == "GTEX_B" ~ 1,
81       Clase == "TCGA_BLGG" ~ 2,
82       Clase == "TCGA_GM" ~ 3)) %>%
83     dplyr::select(Clase) %>%
84     as.matrix()
85
86   NetLDA.Train <- NetDA(X = X_Train,
87                           Y = Y_Train,
88                           method = 1,
89                           X_test = X_Train)
90
91   NetLDA.Test <- NetDA(X = X_Train,
92                         Y = Y_Train,
93                         method = 1,
94                         X_test = X_Test)
95
96   MC.Train <- table(Y_Train, NetLDA.Train$yhat)
97   MC.Test <- table(Y_Test, NetLDA.Test$yhat)
98
99   return(list(MC.Train = MC.Train,
```

```
100         MC.Test = MC.Test))
101     }
102 
103 
104 
105 # Resultados -----
106 
107 tic()
108 M2.LDA.1 <- Evaluacion1(Metodo = "LDA.1",
109                         workers = availableCores())
110 TE_LDA.1 <- toc(log = TRUE)
111 
112 tic()
113 M2.LDA.2 <- Evaluacion1(Metodo = "LDA.2",
114                         workers = availableCores())
115 TE_LDA.2 <- toc(log = TRUE)
116 
117 
118 # Gráficas -----
119 
120 G.LDA.1 <- M2.LDA.1[["Global"]] %>%
121   mutate(Modelo = "Linear Discriminant Analysis",
122         Nombre = "LDA 1")
123 
124 G.LDA.2 <- M2.LDA.2[["Global"]] %>%
125   mutate(Modelo = "Linear Discriminant Analysis",
126         Nombre = "LDA 2")
127 
128 M2 <- bind_rows(G.LDA.1, G.LDA.2) %>%
129   mutate(Modelo = as.factor(Modelo),
130         Nombre = as.factor(Nombre))
131 
132 G.TE_LDA.1 <- data.frame(TE = TE_LDA.1[["callback_msg"]]) %>%
133   mutate(Modelo = "Linear Discriminant Analysis",
134         Nombre = "LDA 1")
135 
136 G.TE_LDA.2 <- data.frame(TE = TE_LDA.2[["callback_msg"]]) %>%
137   mutate(Modelo = "Linear Discriminant Analysis",
138         Nombre = "LDA 2")
139 
140 TE_M2 <- bind_rows(G.TE_LDA.1, G.TE_LDA.2) %>%
141   mutate(Modelo = as.factor(Modelo),
142         Nombre = as.factor(Nombre))
143 
144 write.csv(x = M2,
145            file = "Modelo2.csv",
146            row.names = FALSE)
147 
148 write.csv(x = TE_M2,
149            file = "TE_Modelo2.csv",
150            row.names = FALSE)
151 
152 
153 # Promedios matriz de confusión -----
```

```

155 MC.M2.LDA.1 <- M2.LDA.1[["MatricesConfusion"]] %>%
156   transpose()
157
158 MC.M2.LDA.2 <- M2.LDA.2[["MatricesConfusion"]] %>%
159   transpose()
160
161 MC.M2.LDA.1.PROM <- Reduce("+", MC.M2.LDA.1$MC.Test) / length(MC.M2.LDA.1$MC.Test)
162 MC.M2.LDA.1.PROM <- round(t(apply(MC.M2.LDA.1.PROM, 1, function(x) x / sum(x) *
163   100)),2)
164
165 MC.M2.LDA.2.PROM <- Reduce("+", MC.M2.LDA.2$MC.Test) / length(MC.M2.LDA.2$MC.Test)
166 MC.M2.LDA.2.PROM <- round(t(apply(MC.M2.LDA.2.PROM, 1, function(x) x / sum(x) *
167   100)),2)
168
169 # MC Mapa de Calor -----
170 MC.M2.LDA.1.DF <- tibble(
171   PredTest = c("GTEX_Brain", "TCGA_LGG", "TCGA_GM"),
172   GTEX_Brain = c(87.56, 11.70, 0.74),
173   TCGA_LGG = c(6.37, 85.52, 8.12),
174   TCGA_GM = c(2.05, 27.20, 70.75)) %>%
175   pivot_longer(-PredTest, names_to = "Referencia", values_to = "Porcentaje") %>%
176   rename(Prediccion = PredTest) %>%
177   mutate(Referencia = as.factor(Referencia),
178         Prediccion = as.factor(Prediccion)) %>%
179   mutate(Prediccion = recode(Prediccion,
180     "GTEX_Brain" = "GTEX Brain",
181     "TCGA_LGG" = "TCGA LGG",
182     "TCGA_GM" = "TCGA GM"),
183     Referencia = recode(Referencia,
184     "GTEX_Brain" = "GTEX Brain",
185     "TCGA_LGG" = "TCGA LGG",
186     "TCGA_GM" = "TCGA GM"),
187     Modelo = as.factor("Linear Discriminant Analysis"),
188     Nombre = as.factor("Modelo 1")) %>%
189   mutate(Referencia = fct_relevel(droplevels(Referencia),
190     c("TCGA GM",
191       "TCGA LGG",
192       "GTEX Brain")),
193     Prediccion = fct_relevel(droplevels(Prediccion),
194     c("GTEX Brain",
195       "TCGA LGG",
196       "TCGA GM")))
197
198 MC.M2.LDA.2.DF <- tibble(
199   PredTest = c("GTEX_Brain", "TCGA_LGG", "TCGA_GM"),
200   GTEX_Brain = c(81.77, 17.53, 0.71),
201   TCGA_LGG = c(3.93, 91.20, 4.88),
202   TCGA_GM = c(2.20, 40.66, 57.14)) %>%
203   pivot_longer(-PredTest, names_to = "Referencia", values_to = "Porcentaje") %>%
204   rename(Prediccion = PredTest) %>%
205   mutate(Referencia = as.factor(Referencia),
206         Prediccion = as.factor(Prediccion)) %>%
207   mutate(Prediccion = recode(Prediccion,

```

```

208 "GTEX_Brain" = "GTEX Brain",
209 "TCGA_LGG" = "TCGA LGG",
210 "TCGA_GM" = "TCGA GM"),
211 Referencia = recode(Referencia,
212 "GTEX_Brain" = "GTEX Brain",
213 "TCGA_LGG" = "TCGA LGG",
214 "TCGA_GM" = "TCGA GM"),
215 Modelo = as.factor("Linear Discriminant Analysis"),
216 Nombre = as.factor("Modelo 2")) %>%
217 mutate(Referencia = fct_relevel(droplevels(Referencia),
218 c("TCGA GM",
219 "TCGA LGG",
220 "GTEX Brain")),
221 Prediccion = fct_relevel(droplevels(Prediccion),
222 c("GTEX Brain",
223 "TCGA LGG",
224 "TCGA GM")))
225
226 M2.MC <- bind_rows(MC.M2.LDA.1.DF, MC.M2.LDA.2.DF)
227
228 write.csv(x = M2.MC,
229 file = "MC2.csv",
230 row.names = FALSE)

```

```

1 #####
2 # Quadratic Discriminant Analysis #
3 # QDA #
4 #####
5
6
7 library(tibble)
8 library(plyr)
9 library(dplyr)
10 library(tidyr)
11 library(forcats)
12
13 library(rsample)
14
15 library(purrr)
16 library(furrr)
17
18 library(MASS)
19
20 library(ggplot2)
21
22 library(readr)
23
24 library(tictoc)
25
26
27
28 # Funciones Auxiliares -----
29
30 source("../FuncionesAuxiliares.R")
31

```

```
32 | source("../UGGM/Modelo_UGGM-QDA.R")
33 |
34 |
35 | # Conjuntos Train y Test -----
36 |
37 | load("../Folds.RData")
38 |
39 |
40 | # Esquemas de clasificacion -----
41 |
42 | QDA.1 <- function(Train, Test) {
43 |
44 |   QDA <- qda(formula = Clase~.,
45 |                 data = Train)
46 |
47 |   PredTrain <- predict(object = QDA,
48 |                         newdata = Train)$class
49 |
50 |   PredTest <- predict(object = QDA,
51 |                         newdata = Test)$class
52 |
53 |   MC.Train <- table(Train$Clase, PredTrain)
54 |   MC.Test <- table(Test$Clase, PredTest)
55 |
56 |   return(list(MC.Train = MC.Train,
57 |               MC.Test = MC.Test))
58 |
59 }
60 |
61 | QDA.2 <- function(Train, Test) {
62 |
63 |   rho.tune <- tune.rho(formula = Clase~.,
64 |                         data = Train,
65 |                         rhos = seq(0.1, 1, by = 0.01),
66 |                         nfolds = 5)
67 |
68 |   NetQDA <- UGGM_QDA(formula = Clase~.,
69 |                         data = Train,
70 |                         rho = rho.tune$best.rho)
71 |
72 |   PredTrain <- predict.UGGM_QDA(object = NetQDA,
73 |                                   newdata = dplyr::select(Train, -Clase))
74 |
75 |   PredTest <- predict.UGGM_QDA(object = NetQDA,
76 |                                   newdata = dplyr::select(Test, -Clase))
77 |
78 |   MC.Train <- table(Train$Clase, PredTrain$clase$yhat)
79 |   MC.Test <- table(Test$Clase, PredTest$clase$yhat)
80 |
81 |   return(list(MC.Train = MC.Train,
82 |               MC.Test = MC.Test))
83 |
84 }
85 |
86 }
```

```

87 # Resultados -----
88
89 tic()
90 M3.QDA.1 <- Evaluacion1(Metodo = "QDA.1",
91                         workers = availableCores())
92 TE_QDA.1 <- toc(log = TRUE)
93
94 tic()
95 M3.QDA.2 <- Evaluacion1(Metodo = "QDA.2",
96                         workers = availableCores())
97 TE_QDA.2 <- toc(log = TRUE)
98
99
100 # Gráficas -----
101
102 G.QDA.1 <- M3.QDA.1[["Global"]] %>%
103   mutate(Modelo = "Quadratic Discriminant Analysis",
104         Nombre = "QDA 1")
105
106 G.QDA.2 <- M3.QDA.2[["Global"]] %>%
107   mutate(Modelo = "Quadratic Discriminant Analysis",
108         Nombre = "QDA 2")
109
110 M3 <- bind_rows(G.QDA.1, G.QDA.2) %>%
111   mutate(Modelo = as.factor(Modelo),
112         Nombre = as.factor(Nombre))
113
114 G.TE_QDA.1 <- data.frame(TE = TE_QDA.1[["callback_msg"]]) %>%
115   mutate(Modelo = "Quadratic Discriminant Analysis",
116         Nombre = "QDA 1")
117
118 G.TE_QDA.2 <- data.frame(TE = TE_QDA.2[["callback_msg"]]) %>%
119   mutate(Modelo = "Quadratic Discriminant Analysis",
120         Nombre = "QDA 2")
121
122 TE_M3 <- bind_rows(G.TE_QDA.1, G.TE_QDA.2) %>%
123   mutate(Modelo = as.factor(Modelo),
124         Nombre = as.factor(Nombre))
125
126 write.csv(x = M3,
127             file = "Modelo3.csv",
128             row.names = FALSE)
129
130 write.csv(x = TE_M3,
131             file = "TE_Modelo3.csv",
132             row.names = FALSE)
133
134
135 # Matriz de confusión promediada -----
136
137 MC.M3.QDA.1 <- M3.QDA.1[["MatricesConfusion"]] %>%
138   transpose()
139
140 MC.M3.QDA.2 <- M3.QDA.2[["MatricesConfusion"]] %>%
141   transpose()

```

```

142 MC.M3.QDA.1.PROM <- Reduce("+", MC.M3.QDA.1$MC.Test) / length(MC.M3.QDA.1$MC.Test)
143 MC.M3.QDA.1.PROM <- round(t(apply(MC.M3.QDA.1.PROM, 1, function(x) x / sum(x) *
144   100)),2)
145
146 MC.M3.QDA.2.PROM <- Reduce("+", MC.M3.QDA.2$MC.Test) / length(MC.M3.QDA.2$MC.Test)
147 MC.M3.QDA.2.PROM <- round(t(apply(MC.M3.QDA.2.PROM, 1, function(x) x / sum(x) *
148   100)),2)
149
150 # MC Mapa de Calor -----
151
152 MC.M3.QDA.1.DF <- tibble(
153   PredTest = c("GTEX_Brain", "TCGA_LGG", "TCGA_GM"),
154   GTEX_Brain = c(89.88, 8.57, 1.55),
155   TCGA_LGG = c(5.41, 87.34, 7.25),
156   TCGA_GM = c(2.14, 25.27, 72.59)) %>%
157   pivot_longer(-PredTest, names_to = "Referencia", values_to = "Porcentaje") %>%
158   rename(Prediccion = PredTest) %>%
159   mutate(Referencia = as.factor(Referencia),
160         Prediccion = as.factor(Prediccion)) %>%
161   mutate(Prediccion = recode(Prediccion,
162                             "GTEX_Brain" = "GTEX Brain",
163                             "TCGA_LGG" = "TCGA LGG",
164                             "TCGA_GM" = "TCGA GM"),
165     Referencia = recode(Referencia,
166                           "GTEX_Brain" = "GTEX Brain",
167                           "TCGA_LGG" = "TCGA LGG",
168                           "TCGA_GM" = "TCGA GM"),
169     Modelo = as.factor("Quadratic Discriminant Analysis"),
170     Nombre = as.factor("Modelo 1")) %>%
171   mutate(Referencia = fct_relevel(droplevels(Referencia),
172                                     c("TCGA GM",
173                                       "TCGA LGG",
174                                       "GTEX Brain")),
175     Prediccion = fct_relevel(droplevels(Prediccion),
176                               c("GTEX Brain",
177                                 "TCGA LGG",
178                                 "TCGA GM")))
179
180 MC.M3.QDA.2.DF <- tibble(
181   PredTest = c("GTEX_Brain", "TCGA_LGG", "TCGA_GM"),
182   GTEX_Brain = c(91.06, 7.23, 1.71),
183   TCGA_LGG = c(9.68, 77.55, 12.76),
184   TCGA_GM = c(3.49, 13.37, 83.13)) %>%
185   pivot_longer(-PredTest, names_to = "Referencia", values_to = "Porcentaje") %>%
186   rename(Prediccion = PredTest) %>%
187   mutate(Referencia = as.factor(Referencia),
188         Prediccion = as.factor(Prediccion)) %>%
189   mutate(Prediccion = recode(Prediccion,
190                             "GTEX_Brain" = "GTEX Brain",
191                             "TCGA_LGG" = "TCGA LGG",
192                             "TCGA_GM" = "TCGA GM"),
193     Referencia = recode(Referencia,
194                           "GTEX_Brain" = "GTEX Brain",
195                           "TCGA_LGG" = "TCGA LGG",
196                           "TCGA_GM" = "TCGA GM"))

```

```

195         "TCGA_LGG" = "TCGA LGG",
196         "TCGA_GM" = "TCGA GM"),
197     Modelo = as.factor("Quadratic Discriminant Analysis"),
198     Nombre = as.factor("Modelo 2")) %>%
199     mutate(Referencia = fct_relevel(droplevels(Referencia),
200                                         c("TCGA GM",
201                                         "TCGA LGG",
202                                         "GTEX Brain")),
203     Prediccion = fct_relevel(droplevels(Prediccion),
204                               c("GTEX Brain",
205                                 "TCGA LGG",
206                                 "TCGA GM")))
207
208 M3.MC <- bind_rows(MC.M3.QDA.1.DF, MC.M3.QDA.2.DF)
209
210 write.csv(x = M3.MC,
211             file = "MC3.csv",
212             row.names = FALSE)

```

```

1 #####
2 # Support Vector Machine #
3 # SVM                      #
4 #####
5
6
7
8 library(tibble)
9 library(plyr)
10 library(dplyr)
11 library(tidyr)
12 library(forcats)
13
14 library(rsample)
15
16 library(purrr)
17 library(furrr)
18
19 library(e1071)
20
21 library(ggplot2)
22
23 library(tictoc)
24
25
26 # Funciones Auxiliares -----
27
28 source("../FuncionesAuxiliares.R")
29
30
31 # Conjuntos Train y Test -----
32
33 load("../Folds.RData")
34
35
36 # Esquemas de clasificacion -----

```

```
37 SVM.1 <- function(Train, Test) {  
38  
39   svm <- svm(formula = Clase~,  
40                 kernel = "radial",  
41                 data = Train)  
42  
43   PredTrain <- predict(object = svm,  
44                         newdata = Train,  
45                         type = "response")  
46  
47   PredTest <- predict(object = svm,  
48                         newdata = Test,  
49                         type = "response")  
50  
51   MC.Train <- table(Train$Clase, PredTrain)  
52   MC.Test <- table(Test$Clase, PredTest)  
53  
54   return(list(MC.Train = MC.Train,  
55               MC.Test = MC.Test))  
56  
57 }  
58  
59 SVM.2 <- function(Train, Test) {  
60  
61   svm.tune <- tune(svm, Clase~,  
62                     data = Train,  
63                     tunecontrol = tune.control(cross = 5),  
64                     ranges = list(cost = 2^(2:6), gamma = 2^(-6:-2)))  
65  
66   svm <- svm(formula = Clase~,  
67                 cost = svm.tune$best.parameters[[1]],  
68                 gamma = svm.tune$best.parameters[[2]],  
69                 data = Train)  
70  
71   PredTrain <- predict(object = svm,  
72                         newdata = Train,  
73                         type = "response")  
74  
75   PredTest <- predict(object = svm,  
76                         newdata = Test,  
77                         type = "response")  
78  
79   MC.Train <- table(Train$Clase, PredTrain)  
80   MC.Test <- table(Test$Clase, PredTest)  
81  
82   return(list(MC.Train = MC.Train,  
83               MC.Test = MC.Test))  
84  
85 }  
86  
87  
88 # Resultados -----  
89  
90  
91 tic()
```

```

92 M4.SVM.1 <- Evaluacion1(Metodo = "SVM.1",
93                               workers = availableCores())
94 TE_SVM.1 <- toc(log = TRUE)
95
96 tic()
97 M4.SVM.2 <- Evaluacion1(Metodo = "SVM.2",
98                               workers = availableCores())
99 TE_SVM.2 <- toc(log = TRUE)
100
101
102 # Gr fica -----
103
104 G.SVM.1 <- M4.SVM.1[["Global"]] %>%
105   mutate(Modelo = "Support Vector Machine",
106         Nombre = "SVM 1")
107
108 G.SVM.2 <- M4.SVM.2[["Global"]] %>%
109   mutate(Modelo = "Support Vector Machine",
110         Nombre = "SVM 2")
111
112 M4 <- bind_rows(G.SVM.1, G.SVM.2) %>%
113   mutate(Modelo = as.factor(Modelo),
114         Nombre = as.factor(Nombre))
115
116 G.TE_SVM.1 <- data.frame(TE = TE_SVM.1[["callback_msg"]]) %>%
117   mutate(Modelo = "Support Vector Machine",
118         Nombre = "SVM 1")
119
120 G.TE_SVM.2 <- data.frame(TE = TE_SVM.2[["callback_msg"]]) %>%
121   mutate(Modelo = "Support Vector Machine",
122         Nombre = "SVM 2")
123
124 TE_M4 <- bind_rows(G.TE_SVM.1, G.TE_SVM.2) %>%
125   mutate(Modelo = as.factor(Modelo),
126         Nombre = as.factor(Nombre))
127
128 write.csv(x = M4,
129             file = "Modelo4.csv",
130             row.names = FALSE)
131
132 write.csv(x = TE_M4,
133             file = "TE_Modelo4.csv",
134             row.names = FALSE)
135
136
137 # Matriz de confusin promediada -----
138
139 MC.M4.SVM.1 <- M4.SVM.1[["MatricesConfusion"]] %>%
140   transpose()
141
142 MC.M4.SVM.2 <- M4.SVM.2[["MatricesConfusion"]] %>%
143   transpose()
144
145 MC.M4.SVM.1.PROM <- Reduce("+", MC.M4.SVM.1$MC.Test) / length(MC.M4.SVM.1$MC.Test)

```

```

146 MC.M4.SVM.1.PROM <- round(t(apply(MC.M4.SVM.1.PROM, 1, function(x) x / sum(x) *
147   100)),2)
148 MC.M4.SVM.2.PROM <- Reduce("+", MC.M4.SVM.2$MC.Test) / length(MC.M4.SVM.2$MC.Test)
149 MC.M4.SVM.2.PROM <- round(t(apply(MC.M4.SVM.2.PROM, 1, function(x) x / sum(x) *
150   100)),2)
151
152 # MC Mapa de Calor -----
153
154 MC.M4.SVM.1.DF <- tibble(
155   PredTest = c("GTEX_Brain", "TCGA_LGG", "TCGA_GM"),
156   GTEX_Brain = c(89.17, 9.95, 0.88),
157   TCGA_LGG = c(3.54, 90.90, 5.56),
158   TCGA_GM = c(2.41, 31.48, 66.11)) %>%
159   pivot_longer(-PredTest, names_to = "Referencia", values_to = "Porcentaje") %>%
160   rename(Prediccion = PredTest) %>%
161   mutate(Referencia = as.factor(Referencia),
162         Prediccion = as.factor(Prediccion)) %>%
163   mutate(Prediccion = recode(Prediccion,
164     "GTEX_Brain" = "GTEX Brain",
165     "TCGA_LGG" = "TCGA LGG",
166     "TCGA_GM" = "TCGA GM"),
167     Referencia = recode(Referencia,
168       "GTEX_Brain" = "GTEX Brain",
169       "TCGA_LGG" = "TCGA LGG",
170       "TCGA_GM" = "TCGA GM"),
171     Modelo = as.factor("Support Vector Machine"),
172     Nombre = as.factor("Modelo 1")) %>%
173   mutate(Referencia = fct_relevel(droplevels(Referencia),
174     c("TCGA GM",
175       "TCGA LGG",
176       "GTEX Brain")),
177     Prediccion = fct_relevel(droplevels(Prediccion),
178       c("GTEX Brain",
179         "TCGA LGG",
180         "TCGA GM")))
181
182 MC.M4.SVM.2.DF <- tibble(
183   PredTest = c("GTEX_Brain", "TCGA_LGG", "TCGA_GM"),
184   GTEX_Brain = c(88.76, 10.32, 0.92),
185   TCGA_LGG = c(3.39, 90.96, 5.65),
186   TCGA_GM = c(2.47, 32.29, 65.24)) %>%
187   pivot_longer(-PredTest, names_to = "Referencia", values_to = "Porcentaje") %>%
188   rename(Prediccion = PredTest) %>%
189   mutate(Referencia = as.factor(Referencia),
190         Prediccion = as.factor(Prediccion)) %>%
191   mutate(Prediccion = recode(Prediccion,
192     "GTEX_Brain" = "GTEX Brain",
193     "TCGA_LGG" = "TCGA LGG",
194     "TCGA_GM" = "TCGA GM"),
195     Referencia = recode(Referencia,
196       "GTEX_Brain" = "GTEX Brain",
197       "TCGA_LGG" = "TCGA LGG",
198       "TCGA_GM" = "TCGA GM"),

```

```

199     Modelo = as.factor("Support Vector Machine"),
200     Nombre = as.factor("Modelo 2")) %>%
201   mutate(Referencia = fct_relevel(droplevels(Referencia),
202                                     c("TCGA GM",
203                                       "TCGA LGG",
204                                       "GTEX Brain")),
205         Prediccion = fct_relevel(droplevels(Prediccion),
206                                     c("GTEX Brain",
207                                       "TCGA LGG",
208                                       "TCGA GM")))
209
210 M4.MC <- bind_rows(MC.M4.SVM.1.DF, MC.M4.SVM.2.DF)
211
212 write.csv(x = M4.MC,
213            file = "MC4.csv",
214            row.names = FALSE)

```

```

1 #####
2 # Random Forest #
3 # RF             #
4 #####
5
6
7 library(tibble)
8 library(plyr)
9 library(dplyr)
10 library(tidyr)
11 library(forcats)
12
13 library(rsample)
14
15 library(purrr)
16 library(furrr)
17
18 library(ranger)
19
20 library(ggplot2)
21
22 library(tictoc)
23
24
25
26 # Funciones Auxiliares -----
27
28 source("../FuncionesAuxiliares.R")
29
30
31 # Conjuntos Train y Test -----
32
33 load("../Folds.RData")
34
35
36 # Esquemas de clasificacion -----
37
38 RF.1 <- function(Train, Test) {

```

```

39 RF <- ranger(formula = Clase~.,
40                 data = Train,
41                 importance = "impurity",
42                 probability = FALSE)
43
44 PredTrain <- predict(object = RF,
45                         data = Train)
46
47 PredTest <- predict(object = RF,
48                         data = Test)
49
50
51 MC.Train <- table(Train$Clase, PredTrain$predictions)
52 MC.Test <- table(Test$Clase, PredTest$predictions)
53
54 return(list(MC.Train = MC.Train,
55             MC.Test = MC.Test))
56
57 }
58
59 RF.2 <- function(Train, Test) {
60
61   malla_hyper <- expand.grid(
62     mtry      = seq(from = 1, to = 11, by = 1),
63     node_size = c(1,10,15),
64     num.trees = c(50,100,500)
65   )
66
67   malla_hyper$00Berr <- NA
68
69   for(i in 1:nrow(malla_hyper)) {
70     rf <- ranger(
71       formula      = Clase~.,
72       data         = Train,
73       num.trees    = malla_hyper$num.trees[i],
74       mtry         = malla_hyper$mtry[i],
75       min.node.size = malla_hyper$node_size[i],
76       importance = 'impurity')
77     malla_hyper$00Berr[i] <- rf$prediction.error
78   }
79
80   position <- which.min(malla_hyper$00Berr)
81
82   RF.Tune <- ranger(Clase~.,
83                     data = Train,
84                     num.trees = malla_hyper$num.trees[position],
85                     min.node.size = malla_hyper$node_size[position],
86                     mtry = malla_hyper$mtry[position],
87                     importance = 'impurity',
88                     probability = FALSE)
89
90   PredTrain <- predict(object = RF.Tune,
91                         data = Train)
92
93   PredTest <- predict(object = RF.Tune,

```

```

94         data = Test)
95
96 MC.Train <- table(PredTrain$predictions, Train$Clase)
97 MC.Test <- table(PredTest$predictions, Test$Clase)
98
99 return(list(MC.Train = MC.Train,
100             MC.Test = MC.Test))
101 }
102
103
104 # Resultados -----
105
106 tic()
107 M5.RF.1 <- Evaluacion1(Metodo = "RF.1",
108                           workers = availableCores())
109 TE_RF.1 <- toc(log = TRUE)
110
111 tic()
112 M5.RF.2 <- Evaluacion1(Metodo = "RF.2",
113                           workers = availableCores())
114 TE_RF.2 <- toc(log = TRUE)
115
116
117 # Gráfica -----
118
119 G.RF.1 <- M5.RF.1[["Global"]] %>%
120   mutate(Modelo = "Random Forest",
121         Nombre = "RF 1")
122
123 G.RF.2 <- M5.RF.2[["Global"]] %>%
124   mutate(Modelo = "Random Forest",
125         Nombre = "RF 2")
126
127 M5 <- bind_rows(G.RF.1, G.RF.2) %>%
128   mutate(Modelo = as.factor(Modelo),
129         Nombre = as.factor(Nombre))
130
131 G.TE_RF.1 <- data.frame(TE = TE_RF.1[["callback_msg"]]) %>%
132   mutate(Modelo = "Random Forest",
133         Nombre = "RF 1")
134
135 G.TE_RF.2 <- data.frame(TE = TE_RF.2[["callback_msg"]]) %>%
136   mutate(Modelo = "Random Forest",
137         Nombre = "RF 2")
138
139 TE_M5 <- bind_rows(G.TE_RF.1, G.TE_RF.2) %>%
140   mutate(Modelo = as.factor(Modelo),
141         Nombre = as.factor(Nombre))
142
143 write.csv(x = M5,
144            file = "Modelo5.csv",
145            row.names = FALSE)
146
147 write.csv(x = TE_M5,
148            file = "TE_Modelo5.csv",

```

```

149     row.names = FALSE)
150
151
152 # Matriz de confusión promediada -----
153
154 MC.M5.RF.1 <- M5.RF.1[["MatricesConfusion"]] %>%
155   transpose()
156
157 MC.M5.RF.2 <- M5.RF.2[["MatricesConfusion"]] %>%
158   transpose()
159
160 MC.M5.RF.1.PROM <- Reduce("+", MC.M5.RF.1$MC.Test) / length(MC.M5.RF.1$MC.Test)
161 MC.M5.RF.1.PROM <- round(t(apply(MC.M5.RF.1.PROM, 1, function(x) x / sum(x) * 100)),2)
162
163 MC.M5.RF.2.PROM <- Reduce("+", MC.M5.RF.2$MC.Test) / length(MC.M5.RF.2$MC.Test)
164 MC.M5.RF.2.PROM <- round(t(apply(MC.M5.RF.2.PROM, 1, function(x) x / sum(x) * 100)),2)
165
166
167 # MC Mapa de Calor -----
168
169 MC.M5.RF.1.DF <- tibble(
170   PredTest = c("GTEX_Brain", "TCGA_LGG", "TCGA_GM"),
171   GTEX_Brain = c(87.26, 11.78, 0.95),
172   TCGA_LGG = c(4.30, 89.89, 5.80),
173   TCGA_GM = c(2.02, 31.20, 66.78)) %>%
174   pivot_longer(-PredTest, names_to = "Referencia", values_to = "Porcentaje") %>%
175   rename(Prediccion = PredTest) %>%
176   mutate(Referencia = as.factor(Referencia),
177         Prediccion = as.factor(Prediccion)) %>%
178   mutate(Prediccion = recode(Prediccion,
179                             "GTEX_Brain" = "GTEX Brain",
180                             "TCGA_LGG" = "TCGA LGG",
181                             "TCGA_GM" = "TCGA GM"),
182     Referencia = recode(Referencia,
183                             "GTEX_Brain" = "GTEX Brain",
184                             "TCGA_LGG" = "TCGA LGG",
185                             "TCGA_GM" = "TCGA GM"),
186     Modelo = as.factor("Random Forest"),
187     Nombre = as.factor("Modelo 1")) %>%
188   mutate(Referencia = fct_relevel(droplevels(Referencia),
189                                     c("TCGA GM",
190                                       "TCGA LGG",
191                                       "GTEX Brain")),
192     Prediccion = fct_relevel(droplevels(Prediccion),
193                               c("GTEX Brain",
194                                 "TCGA LGG",
195                                 "TCGA GM")))
196
197 MC.M5.RF.2.DF <- tibble(
198   PredTest = c("GTEX_Brain", "TCGA_LGG", "TCGA_GM"),
199   GTEX_Brain = c(90.77, 8.14, 1.09),
200   TCGA_LGG = c(6.22, 84.31, 9.47),
201   TCGA_GM = c(1.96, 20.96, 77.08)) %>%
202   pivot_longer(-PredTest, names_to = "Referencia", values_to = "Porcentaje") %>%
203   rename(Prediccion = PredTest) %>%

```

```
204 mutate(Referencia = as.factor(Referencia),  
205     Prediccion = as.factor(Prediccion)) %>%  
206 mutate(Prediccion = recode(Prediccion,  
207             "GTEX_Brain" = "GTEX Brain",  
208             "TCGA_LGG" = "TCGA LGG",  
209             "TCGA_GM" = "TCGA GM"),  
210     Referencia = recode(Referencia,  
211             "GTEX_Brain" = "GTEX Brain",  
212             "TCGA_LGG" = "TCGA LGG",  
213             "TCGA_GM" = "TCGA GM"),  
214     Modelo = as.factor("Random Forest"),  
215     Nombre = as.factor("Modelo 2")) %>%  
216 mutate(Referencia = fct_relevel(droplevels(Referencia),  
217             c("TCGA GM",  
218                 "TCGA LGG",  
219                 "GTEX Brain")),  
220     Prediccion = fct_relevel(droplevels(Prediccion),  
221             c("GTEX Brain",  
222                 "TCGA LGG",  
223                 "TCGA GM")))  
224  
225 M5.MC <- bind_rows(MC.M5.RF.1.DF, MC.M5.RF.2.DF)  
226  
227 write.csv(x = M5.MC,  
228             file = "MC5.csv",  
229             row.names = FALSE)
```