UNIVERSIDAD
DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Facultad de Ciencias

Doble Grado en Informática y Matemáticas

trabajo de fin de grado

Aprendizaje Automático con datos genéticos. Aplicación al autismo.

Presentado por:
Víctor Manuel Arroyo Martín

Tutor:
María del Mar Abad Grau
Lenguajes y Sistemas Informáticos

Ofelia Paula Retamero Pascual
Ciencias de la Computación e Inteligencia Artificial

Curso académico 2020-2021

Aprendizaje Automático con datos genéticos. Aplicación al autismo.

Víctor Manuel Arroyo Martín

Víctor Manuel Arroyo Martín *Aprendizaje Automático con datos genéticos. Aplicación al autismo.*

Trabajo de fin de Grado. Curso académico 2020-2021.

Responsable de tutorización

María del Mar Abad Grau
Lenguajes y Sistemas Informáticos

Ofelia Paula Retamero Pascual
Ciencias de la Computación e Inteligencia Artificial

Doble Grado en Informática y Matemáticas

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Facultad de Ciencias

Universidad de Granada

Declaración de originalidad

D./Dña. Víctor Manuel Arroyo Martín

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2020-2021, es original, entendida esta, en el sentido de que no ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 16 de diciembre de 2021

Fdo: Víctor Manuel Arroyo Martín

Índice general

Índice de figuras

Índice de tablas

Resumen

El trabajo está dirigido principalmente a la detección del sesgo de laboratorio en afecciones complicadas de detectar a través de la genética, como es el trastorno del espectro autista (ASD); razón por la que los múltiples intentos de predecir estas afecciones no han obtenido buenos resultados. Para ello, usaremos modelos predictivos encargados de detectar ese sesgo y diversos tests para comprobar que el sesgo es real y no debido a diferencias aleatorias. El trabajo se divide en dos problemas interconectados: el primero es la introducción a las matemáticas que nos permitirán la justificación y explicación general y matemática de los modelos predictivos que se usarán en el trabajo. Además, se hará antes una introducción a los conceptos del aprendizaje automático y algoritmos de aprendizaje, sus tipos, los errores del entrenamiento y conceptos como el sobreajuste y la regularización. También se aborda brevemente las métricas usadas para llevar a cabo la evaluación del rendimiento de estos modelos predictivos. Continuaremos desarrollando en la segunda parte las principales características del conjunto de datos que se ha utilizado para detectar ese sesgo de laboratorio así como el origen de los datos y los conceptos genéticos más relevantes para su entendimiento, y daremos paso a la explicación de los modelos. En esta parte se exponen las librerías y modelos predictivos utilizados así como sus principales parámetros y valores de los mismos empleados. Tras esto, se presentan los resultados y análisis de los mismos para cada modelo en los conjuntos de datos utilizados, así como de los tests empleados. Por último se expone la conclusión y vías de trabajo futuras pendientes de este trabajo.

PALABRAS CLAVE: trastorno del espectro autista ASD modelos predictivos sesgo de laboratorio aprendizaje automático algoritmos de aprendizaje sobreajuste regularización

Summary

The main goal of this project is to detect the laboratory bias from raw genotyping intensities. The search of genetic causes of complex diseases from GWAS (Genome-wide association study) hasn't shown the results expected from them due to different problem such as the laboratory bias, that is, the difference in genotyping that two different genetic samples can present because of the reactive used to get them or serveral customisations from their respective laboratories. The genotyping process usually leads to data with hundreds of variables that is really difficult to manage with classical mathematic tools. So, for these kind of data, machine learning techniques have proven to be very effective, because they are created to be able to learn from large and complex amount of data. In fact, they perform better when there is a lot of data. Machine learning is a discipline that builds computer science systems that improve with experience through statistics and several mathematics tools, and through computing.

For this detection, we use several machine learning predictive models and some hypothesis contrast tests to confirm the results obtained by these models. The project is divided into seven chapters, each one with a differentiated target to explain.

Firstly we start with a brief introduction to become familiar with the objetives of this project, motivation and its background as well as the techniques used to achieve this goal. After that, we begin with the first chapter dedicated to explain and justify the statistical inference behind the predictive models used later on. The main goal of this part is to understand the maximum likelihood estimators and the optimization related to them. That lead us to the second chapter in wich we introduce and explain the main multivariate statistics used in the predictive models, the definite matrixes and their main properties and the main point of this chapter: the multivariate normal distribution and some properties deduced from it. Then we start the main part of the mathematical justification for this project: the machine learning. We start this third chapter with an extended introduction to get used to the previous notions for machine learning, such as supervised, unsupervised and reinforcement learning types. After that, we introduce a basic learning schema to define the main traits of a machine learning problem like the target function, the labels, the training and test sets and the variables of the problem. We present an example of a credit bank decision problem to det familiar with these concepts even more and we finally connect them with our problem: detection of laboratori bias from raw genotyping intensitis for three different autism spectrum disorder data sets. And then we begin one of the most important parts of machine learning, the in-sample and out-of-sample errors. Firts, we present a theory of generalization for these error with the main goal of connecting them in what is known as the *VC generalization bound*, that is one of the most important results in the machine learning theory thanks to the relation that it establishes within the two types of error. In this part we also discuss the well-known bias-variance tradeoff and the importance of regularization in these predictive models to make a reliable prediction.

After that, we begin the mathmatical and general explanation of the predictive model that have been used in this project to detect that laboratory bias. We start with an easy one called the k-nearest neighbours, a model that classifies data based on the closest points around it,

then we explain the linear model, logistic regresion that establish a correlation in the data used to predict the labels of some new data. For the next model we have decision trees and random forest, the first one is a model based on set of decisions to make on a data point in the form of a tree and the second one uses a bunch of these trees and randomize them to make and ponderate the decisions on that data point. The next one is the gradient boosting, a model that uses decision trees too but with an added optimization in their decision. In the next section we have the last model, the support vector machine. This model is focused on finding a hyperplane to separate data and for that purpose, it solves an optimization problem to minimize the missclassified data. We finish this chapter with and explanation of the metrics used to evaluate the results of these models, such as the accuracy, precision, the F_1 metric or the confusion matrix.

In the next chapter we explain the data set used to train the predictive models and detect the laboratory bias with them. We start this chapter by introducing the most relevant genetic concepts: the nucleotides, the *Single Nucleotide Plymotphisms* (SNPs) that is the main concept in our data and the *Polymerase Chain Reaction* (PCR) used to obtain them. We also explain the microarrays and their intensity (the numerical values of our data) and finally the GWAS mentioned before. After that, we present the data structure and the relevant variables, concluding this part with the folder structure followed to make the experiment.

We now reach the experimentation part where we talk about the code structure, the libraries used to implement it and the code and its programming language. This chapter is divided as it follows: after we present the code structure, we start explaining the libraries used for the implementation of that code in which we reference all of them and make a short presentation of their functionalities. Then we explain the data exploration performed, that is, the structure of each dataset used to train the models and detect the laboratory bias and some graphical representation to see visually this bias. We also explain how we store the data across the code and the variables we finally kept to develop the program. Then we present the preprocessing applied to the data to try to improve the models results. We explain concepts like the variance threshold, the standard scaler and the polynomial features and in this section we also explain the hipotesys contrast tests used to determine if the detected bias is real or the differences between the data detected by the predictive models are due to random differences. We also present to models that have been used to variable selection and the hipotesys contrast tests used to make variable selection too. After this introduction to the code structure, the preprocessing and the variable selection, we start explaining the models we used, their implementation and their main parameters. We also explain the methods used to get the most influential variables in the training of these models and the method known as cross validation. In the section dedicated to this method we talk about the concept, the implementation used and the different predictive models employed for its implementation. Finally we discuss the hipotesys contrast tests to test the hipotesys that the data have the same expected value and conclude that the differences in data that the predictive models have detected are real bias and not random differences in the data.

The next chapter is about presenting and evaluating the results of the predictive models by using the metric presented in the third chapter. For each model we present its confusion matrix, precision, accuracy, f1-score and their average over both of the labels predicted for each dataset. We also present in some of the models the main variables that these models used for their prediction and we connect the results with our problem of detecting the laboratory bias. For the cross validation method, we show the best model and its results just like we do with the other models. Then, we start the variable selection section where we present the results and selected variables after testing the hipotesys contrast tests and the interpretation

for these results. We explain in this section the retrained models using this data after the variable selection. To conclude the chapter, the final tests results to test if the bias is real are given and interpreted to finally reach the last chapter where we discuss if the goals of this invetigation project are achieved or not and we suggest some future work to continue and complete this project.

KEY WORDS (PALABRAS CLAVE): genotyping intensities laboratory bias machine learning bias-variance tradeoff regularization statistical inference

Introducción

El estudio del trastorno del espectro autista (*ASD* por sus siglas en inglés) mediante *machine learning* se ha llevado a cabo sobre todo desde el punto de vista del comportamiento del paciente o sus síntomas en diversas áreas como la neurología [Hyde et al., 2019] pero los resultados de los estudios desde el punto de vista genético en general son mucho peores de lo que se podía esperar [Su et al., 2007]; de aquí surge la idea de intentar encontrar anomalías que puedan predecir los errores de **genotipados** [Su et al., 2007] y el **sesgo de laboratorio** y que puedan explicar los bajos resultados en la detección de las bases genéticas de las afecciones complejas de este tipo. En general, el estudio desde el punto de vista genético y de los GWAS (estudios con el genoma completo) de este tipo de enfermedades complejas es difícil de llevar a cabo, no sólo por el sesgo de laboratorio que puede haber en los datos sino también por la poca cantidad de datos disponibles para enfermedades raras.
Normalmente, los datos genéticos presentan un gran número de individuos y variables a tener en cuenta. Es por esto que el **aprendizaje automático** (*machine learning*) tiene mucho que aportar a este campo. [Libbrecht and Noble, 2015]
El **aprendizaje automático** (*machine learning*) es una disciplina que construye sistemas informáticos que mejoran con la experiencia a través de las leyes de la estadística y la computación [Jordan and Mitchell, 2015] [Abu-Mostafa et al., 2012].

En los primeros momentos de la inteligencia artificial, la disciplina abordó y resolvió rápidamente problemas que son intelectualmente difíciles para los seres humanos pero relativamente sencillos para la computadora, problemas que pueden ser descritos por una lista de reglas formales y matemáticas. Las dificultades a las que se enfrentan más adelante los sistemas que se basan en el conocimiento codificado formal sugieren que los sistemas de IA necesitan la capacidad de adquirir su propio conocimiento, extrayendo patrones de datos sin procesar. Esta capacidad se conoce como *machine learning*. [Goodfellow et al., 2016].

Así, a grandes rasgos, se puede dividir el desarrollo de la IA en cuatro momentos diferentes: primero, ente los años 1950 y 1960 se empezó con su exploración por ejemplo en el reconocimiento de escritura a mano. Entre 1960 y 1970 el avance fue rápido y empezaron a resolverse problema de la visión por computador así como biológicos pero no fue hasta 1970 que se abandonaron los problemas 'de juguete' y se entró a mundos tales como los de los sistemas expertos y el procesamiento del lenguaje natural. En 1980 hacia delante se presenta la cuarta etapa donde el trabajo técnico y matemático cobraron más importancia y el *machine learning* se desarrolló drásticamente desde la curiosidad científica a aplicaciones comerciales. [Nilsson, 2009] [Jordan and Mitchell, 2015].
Hoy en día el *machine learning* es una parte esencial de muchas aplicaciones comerciales y proyectos de investigación en áreas que van desde el diagnóstico y tratamiento médico hasta encontrar a tus amigos en las redes sociales [Müller et al., 2016] y ha mejorado mucho la tecnología moderna en muchas tareas duras de inteligencia artificial como la detección de objetos, el reconocimiento de voz y la traducción automática.

Para nuestro trabajo, tendremos que hacer uso de disciplinas matemáticas tales como el álgebra lineal, la teoría de probabilidad y estadística y el cálculo multivariante.

De esta forma, en este trabajo se han introducido los conceptos matemáticos necesarios para explicar de forma general el aprendizaje automático así como los **modelos predictivos** utilizados para su desarrollo. También se ha explicado en detalle la base de datos genética y los conceptos relevantes sobre la misma para entenderla. Por último se ha explicado la implementación de cada algoritmo, librerías usadas y tests de contraste de hipótesis empleados con sus conclusiones y análisis, así como una conclusión final y evaluación de objetivos del trabajo.

Objetivos

El trabajo presentado es un trabajo de investigación y, como tal, su objetivo es el de tomar unos datos que no sabemos con certeza qué resultados van a dar y experimentar con ellos para obtener unas conclusiones, sean resultados buenos o malos. En concreto, el objetivo de esta investigación ha sido la detección de sesgo de laboratorio en datos genéticos de afecciones complejas como la de este caso: el trastorno del espectro autista (ASD por sus siglas en inglés). Se busca, a través del aprendizaje automático y la estadística, encontrar diferencias entre tres conjuntos de datos genéticos relativos a esta afección tomados desde distintos laboratorios para comprobar que efectivamente existe ese sesgo. Además se comprueba que las diferencias no sean aleatorias sino diferencias significativas que lleven a la conclusión de que existe ese sesgo.

El objetivo de detección de sesgo de laboratorio finalmente se consiguió. El estudio se llevó a cabo de la siguiente manera: en el Capítulo 1 y el Capítulo 2 se hace una introducción a los conceptos matemáticos relevantes para luego, en el Capítulo 3, explicar de forma general y matemática los modelos predictivos utilizados para la detección del sesgo. Tras esto, en el Capítulo 4 se explican los conceptos genéticos necesarios para entender el cnojunto de datos utilizado y se estudia este conjunto destacando sus principales características y su origen. Después, se ha pasado a explicar la implementación de cada algoritmo de parendizaje y test de contraste de hipótesis utilizado así como de las diferentes librerías utilizadas para ello en el Capítulo 5 y en el Capítulo 6 los resultados de ejecurtar esta implementación sobre el conjunto de datos explicado. Por último se ha concluído el trabajo en el Capítulo 7 y se han presentado las vías de trabajo futuras.

1 Inferencia estadística y estadística empleada

En este primer capítulo se va a exponer, justificar y demostrar la inferencia estadística y la estadística empleada en el capítulo de explicación teórica de los modelos (Capítulo 3).

1.1. Espacios muestrales y probabilidad básica

Imaginemos que repetimos un mismo experimento aleatorio, por ejemplo lanzar una moneda numerosas veces: no siempre esperamos obtener el mismo resultado y algunos resultados pueden ser más probables que otros. Cada ejecución del experimento produce exactamente un conjunto de posibles resultados, nunca obtenemos dos o más conjuntos de resultados de un solo experimento y nunca obtenemos un experimento sin resultados. Así, podemos calcular la frecuencia de que cada resultado aparezca.

Definición 1.1. El **espacio muestral** Ω es el conjunto de todos los posibles resultados de un experimento aleatorio.

Supongamos ahora que ejecutamos un experimento y obtenemos un resultado. Podemos ver si está en un conjunto particular de posibles resultados, así que deberíamos ser capaces de predecir la probabilidad de un conjunto de resultados, por ejemplo, la probabilidad de obtener un número par al tirar un dado. Esto nos lleva a la definición de evento.

Definición 1.2. Un **evento** $\mathcal{A}$ es un conjunto de resultados, un subconjunto del espacio muestral $\mathcal{A} \in \Omega$.

Todo evento cumple los llamados **axiomas de probabilidad**: [Forsyth, 2018]

- La probabilidad de cualquier evento $\mathcal{A}$ es siempre positiva y nunca mayor que 1:

$$0 \leq P(\mathcal{A}) \leq 1.$$

- Todos los experimentos tienen un resultado. Esto quiere decir que

$$P(\Omega) = 1$$

- La probabilidad de eventos disjuntos es aditiva. Sea $\mathcal{A}_i$ una colección de eventos tales que $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ cuando $i \neq j$, lo que quiere decir que no hay ningún resultado que aparezca en más de un $\mathcal{A}_i$. Así, se tiene que

$$P(\cup_i \mathcal{A}_i) = \sum_i P(\mathcal{A}_i)$$

Denotamos como $\mathcal{A}^c$ al **evento complementario** de $\mathcal{A}$ y veamos algunas proposiciones básicas de los eventos.

Proposición 1.1. $P(\mathcal{A}^c) = 1 - P(\mathcal{A})$

Demostración. $\mathcal{A}^c$ y $\mathcal{A}$ son eventos disjuntos (si ocurre uno nunca ocurre el otro y viceversa) así que $P(\mathcal{A}^c \cup \mathcal{A}) = P(\mathcal{A}^c) + P(\mathcal{A}) = P(\Omega) = 1$. □

Proposición 1.2. $P(\emptyset) = 0$

Demostración. $P(\emptyset) = P(\Omega^c) = P(\Omega - \Omega) = 1 - P(\Omega) = 1 - 1 = 0$ □

Proposición 1.3. *Para cualquiera dos eventos $\mathcal{A}$ y $\mathcal{B} \in \Omega$, $P(\mathcal{A} - \mathcal{B}) = P(\mathcal{A}) - P(\mathcal{A} \cap \mathcal{B})$*

Demostración. $\mathcal{A} - \mathcal{B}$ es disjunto de $\mathcal{A} \cap \mathcal{B}$, y $(\mathcal{A} - \mathcal{B}) \cup (\mathcal{A} \cap \mathcal{B}) = \mathcal{A}$. Esto significa que $P(\mathcal{A} - \mathcal{B}) + P(\mathcal{A} \cap \mathcal{B}) = P(\mathcal{A})$ □

Proposición 1.4. *Para cualquiera dos eventos $\mathcal{A}$ y $\mathcal{B} \in \Omega$, $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) - P(\mathcal{A} \cap \mathcal{B})$*

Demostración. $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A} \cup (\mathcal{B} \cup \mathcal{A}^c)) = P(\mathcal{A}) + P((\mathcal{B} \cup \mathcal{A}^c))$. Ahora $\mathcal{B} = (\mathcal{B} \cup \mathcal{A}) \cup (\mathcal{B} \cap \mathcal{A}^c)$. Además, $(\mathcal{B} \cap \mathcal{A})$ es disjunto de $(\mathcal{B} \cap \mathcal{A}^c)$ así que tenemos que $P(\mathcal{B}) = P((\mathcal{B} \cup \mathcal{A})) + P((\mathcal{B} \cap \mathcal{A}^c))$. Esto significa que $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P((\mathcal{B} \cap \mathcal{A}^c)) = P(\mathcal{A}) + P(\mathcal{B}) - P((\mathcal{B} \cup \mathcal{A}))$. □

Algunos resultados experimentales no afectan a otros, por ejemplo, en dos tiradas de una moneda el resultado de la primera no afecta al de la segunda. Nos referimos a eventos con esta propiedad como **independientes**.

Definición 1.3. Dos eventos $\mathcal{A}$ y $\mathcal{B}$ son **independientes** si y solo si

$$P(\mathcal{A} \cap \mathcal{B}) = P(\mathcal{A})P(\mathcal{B})$$

En cambio, si no son independientes entonces sabiendo que ha ocurrido uno, puede tener un efecto significativo en la probabilidad de que ocurra el otro. Por ello, hacemos la siguiente definición. [Forsyth, 2018]

Definición 1.4. Definimos la **probabilidad condicional** de $\mathcal{B}$ condicionada a $\mathcal{A}$ como la probabilidad de que ocurra $\mathcal{B}$ cuando ha ocurrido $\mathcal{A}$. La escribimos como $P(\mathcal{B}|\mathcal{A}) = \frac{P(\mathcal{A} \cap \mathcal{B})}{P(\mathcal{A})}$.

Otra forma de escribir la probabilidad consicionada es:

$$P(\mathcal{B}|\mathcal{A})P(\mathcal{A}) = P(\mathcal{B} \cap \mathcal{A})$$

y ahora como $\mathcal{B} \cap \mathcal{A} = \mathcal{A} \cap \mathcal{B}$, tenemos que:

$$P(\mathcal{B}|\mathcal{A}) = \frac{P(\mathcal{A}|\mathcal{B})P(\mathcal{B})}{P(\mathcal{A})}$$

1.2. Variables aleatorias

Para definirlas, debemos conocer antes los conceptos de función de probabilidad y conjunto numerable.

Definición 1.5. Una **función de probabilidad** es una función que mide el grado de incertidumbre de un evento $\mathcal{A} \in \Omega$ de forma numérica entre 0 y 1, siendo 0 un evento imposible de ocurrir y 1 un evento que seguro ocurrirá.

Definición 1.6. Un **conjunto numerable** es aquel conjunto en el que se puede establecer una biyección con los números naturales $\mathbb{N}$.

Definición 1.7. (Variable Aleatoria Discreta) Dado un espacio muestral Ω, un conjunto de eventos $\mathcal{F}$, una función de probabilidad P y un conjunto numerable D de número reales, una variable aleatoria discreta es una función con dominio Ω y rango D.

Definición 1.8. (Variable Aleatoria Continua) Dado un espacio muestral Ω, un conjunto de eventos $\mathcal{F}$, una función de probabilidad P y un conjunto continuo D de número reales, una variable aleatoria discreta es una función con dominio Ω y rango D.

Para las variables aleatorias continuas, damos la siguiente interpretación de la función de densidad.
Sea $p(x)$ una función de densidad para una variable aleatoria continua X, asumimos que dx es un intervalo infinitesimal, entonces:

$$p(x)dx = P(\text{evento en el que X toma el valor del rango [x,x+dx])}\text{[Forsyth, 2018]}$$

Nos centraremos en las variables aleatorias que son las que nos interesan en este trabajo.
La probabilidad de que una variable aleatoria X tome el valor x viene dado por $P(X = x)$, definida como:

Definición 1.9. Distribución de probabilidad de una variable aleatoria discreta. Es un conjunto de números $P(X = x)$ para cada valor x que X puede tomar. La distribución toma el valor 0 en el resto de valores y es no negativa. Se conoce a veces como la **función masa de probabilidad**.

Ahora pasemos a definir la distribución de probabilidad conjunta para dos variables aleatorias.

Definición 1.10. Distribución de probabilidad conjunta de dos variables aleatorias discretas. Sean X e Y dos variables aleatorias discretas, la probabilidad de que X tome el valor x e Y tome el valor y es $P((X = x) \cap (Y = y))$ y se escribe como $P(x, y)$.

Definición 1.11. Sea $P(x, y)$ la distribución de probabilidad conjunta de dos variables aleatorias X e Y, entonces

$$P(x) = \sum_y P(x, y) = P(X = x)$$

se refiere a la **distribución de probabilidad marginal** de X.

El siguiente concepto tiene que ver con la independencia, en concreto de dos variables aleatorias.

Definición 1.12. Dos variables aleatorias X e Y son **independientes** si los eventos $X = x$ e $Y = y$ son independientes para todos los valores x e y. Esto significa que

$$P(x, y) = P(x)P(y)$$

Pasamos ahora a hablar sobre la esperanza.

Definición 1.13. Esperanza. Dada una variable aleatoria discreta X que toma valores en un conjunto D y con una distribución de probabilidad P, definimos la esperanza como

$$E[X] = \sum_{x \in D} xP(X = x)$$

Si tenemos una función f que lleva la variable aleatoria X a un conjunto de números D_f entonces $f(X)$ también es una variable aleatoria que escribimos como F. La esperanza entonces de esta variable se escribe como

$$E[f] = \sum_{x \in D_f} uP(F = u) = \sum_{x \in D} P(X = x)$$

que se llama a veces la 'esperanza de f'.

En su versión continua, la esperanza se define como sigue.

Definición 1.14. Dada una variable continua X que toma valores en el conjunto D y tiene una distribución P, definimos la **esperanza** como

$$E[X] = \int_{x \in D} xp(x)dx$$

Y si tenemos una función f que lleva X a un conjunto de números D_f, entonces $f(X)$ es una variable aleatoria continua también, que se escribe como F. La esperanza de esta variable aleatoria es

$$E[f] = \int_{x \in D} f(x)p(x)dx$$

que se llama a veces la 'esperanza de f'.

La media o esperanza de una variable aleatoria X es $E[X]$ y sus propiedades más importantes son:

- $E[0] = 0$
- para cualquier constante k, $E[kf] = kE[f]$
- $E[f + g] = E[f] + E[g]$

que son consecuencia de las propiedades de la integral y el sumatorio. Y una última propiedad importante, que necesita de demostración, se muestra a continuación:

Proposición 1.5. *Si X e Y son variables aleatorias independientes, entonces $E[XY] = E[X]E[Y]$*

Demostración. Tenemos que:

$$E[XY] = \sum_{(x,y) \in D_x \times D_y} xyP(X = x, Y = y) = \sum_{x \in D_x} \sum_{y \in D_y} (xyP(X = x, Y = y))$$

y por ser X e Y independientes, tenemos que

$$\sum_{x \in D_x} \sum_{y \in D_y} (xyP(X = x)P(Y = y)) = \left(\sum_{x \in D_x} xP(X = x) \right) \left(\sum_{y \in D_y} yP(Y = y) \right)$$

$$= E[X]E[Y]$$

□

se procede de forma análoga para el caso continuo.
Ahora definimos la varianza.

Definición 1.15. Varianza. La varianza de una variable aleatoria X es

$$var[X] = E[(X - E[X])^2]$$

Las propiedades principales de la varianza son:

- Para cualquier constante k, $var[k] = 0$;
- $var[X] \geq 0$;
- $var[kX] = k^2 var[X]$;
- Si X e Y son independientes, entonces $var[X + Y] = var[X] + var[Y]$;

donde las tres primeras vienen directamente de la definición y la cuarta vamos a probarla a continuación.

Demostración. (Prueba de la cuarta propiedad de la varianza). Si X e Y son independientes, entonces $(X - E[X])$ e $(Y - E[Y])$ también lo son. Tenemos así que:

$$var[X + Y] = E[(X + Y - E[X + Y])^2] = E[(X - E[X] + Y - E[Y])^2] =$$

$$= var[X] + var[Y] + 2E[(X - E[X])(Y - E[Y])] = var[X] + var[Y] + 2E[X - E[X]]E[Y - E[Y]]$$

$$= var[X] + var[Y]$$

□

Para terminar esta sección de conceptos básicos, vamos a definir lo que son las muestras aleatorias independientes e idénticamente distribuidas.
Observar un valor de una variable aleatoria se suele llamar **ensayo**. El valor resultante se suele llamar **muestra** de una variable aleatoria (o de su distribución de probabilidad) o a veces **realización**. Con esto, llegamos a la definición:

Definición 1.16. Supongamos que tenemos un conjunto de variables aleatorias X_i que son independientes y tienen la misma distribución $P(X)$. Entonces nos referimos a este conjunto como **muestras aleatorias independientes e idénticamente distribuidas (muestras iid)** o **muestra aleatoria simple (m.a.s.)** de una variable X con distribución $P(X)$. [Forsyth, 2018]

1.3. Estimadores de máxima verosimilitud

Sean $X_1, ..., X_n$ muestras independientes e idénticamente distribuidas de una población con función masa de probabilidad o función de densidad $f(x|\theta_1, ..., \theta_k)$, la función de verosimilitud se define como

$$L(\theta|x) = L(\theta_1, ..., \theta_k|x_1, ..., x_n) = \prod_{i=1}^{n} f(x_1|\theta_1, ..., \theta_k)$$

Definición 1.17. Para cada realización muestral x, sea $\hat{\theta}(x)$ un valor paramétrico donde $L(\theta|x)$ alcanza su máximo como función de θ, con x fija. Un **Estimador Máximo Verosímil (EMV)** del parámetro θ basado en la muestra X es $\hat{\theta}(x)$. [Garthwaite et al., 2002]

Por la forma en la que está construído, el EMV coincide en rango con el parámetro θ.
Una propiedad muy útil de los estimadores máximo verosímiles es lo que se conoce como *propiedad de invarianza de los estimadores máximo verosímiles*, que viene a decir, informalmente, que si tenemos una distribución dependiente de un parámetro θ, pero nos interesa encontrar una estimación de alguna función suya, $\tau(\theta)$ se puede encontrar el EMV de θ, $\hat{\theta}$, y el EMV de $\tau(\theta)$ será $\tau(\hat{\theta})$. Hay, sin embargo, algunos problemas técnicos antes de formalizar la noción de invarianza de los EMV. Si la asignación $\theta \to \tau(\theta)$ es uno a uno (para cada valor de θ hay un único valor de $\tau(\theta)$), no hay problema pues si tomamos $\eta = \tau(\theta)$, entonces la inversa de la función $\tau^{-1}(\eta) = \theta$ está bien definida y la función de verosimilitud de $\tau(\theta)$ se escribe en función de η dada por:

$$L^*(\eta|x) = \prod_{i=1}^{n} f(x_i|\tau^{-1}(\eta)) = L(\tau^{-1}(\eta)|x)$$

y

$$\sup_{\eta} L^*(\eta|x) = \sup_{\eta} L(\tau^{-1}(\eta)|x) = \sup_{\theta} L(\theta|x)$$

Así, el máximo de $L^*(\eta|x)$ es alcanzado en $\eta = \tau(\theta) = \tau(\hat{\theta})$, mostrando que el EMV de $\tau(\theta)$ es $\tau(\hat{\theta})$.
Ahora bien, si la asignación no es uno a uno, necesitamos una definición más general de la función de verosimilitud $\tau(\theta)$ y un teorema más general, pues para un valor dado η puede haber más de un valor θ que satisfaga $\tau(\theta) = \eta$.
Para ello, usamos para $\tau(\theta)$ la *función de verosimilitud inducida* dada por

$$L^*(\eta|x) = \sup_{\{\theta:\tau(\theta)=\eta\}} L(\theta|x)$$

Teorema 1.1. ***(Invarianza de los EMV)*** *Si $\hat{\theta}$ es el EMV de θ entonces para cualquier función $\tau(\theta)$ el EMV de $\tau(\theta)$ es $\tau(\hat{\theta})$*

Demostración. Sea $\hat{\eta}$ el valor que maximiza $L^*(\eta|x)$. Debemos probar que $L^*(\hat{\eta}|x) = L^*(\tau(\theta)|x)$. Ahora, como se puede deducir de la fórmula anterior, el máximo de L y L^* coinciden así que tenemos:

$$\begin{aligned} L^*(\hat{\eta}|x) = \sup_{\eta} \sup_{\{\theta:\tau(\theta)=\eta\}} L(\theta|x) = & \qquad \text{(definición de } L^*) \\ = \sup_{\theta} L(\theta|x) = & \\ = L(\hat{\theta}|x) & \qquad \text{(definición de } \hat{\theta}) \end{aligned}$$

donde la segunda igualdad se da ya que la maximización iterada es igual que la maximización sin condiciones sobre θ, que es alcanzada en $\hat{\theta}$. Además

$$\begin{aligned} L(\theta|x) = \sup_{\{\theta:\tau(\theta)=\tau(\hat{\theta})\}} L(\theta|x) = & \qquad (\hat{\theta} \text{ es el EMV}) \\ = L^*(\tau(\hat{\theta})|x) & \qquad \text{(definición de } L^*) \end{aligned}$$

Por lo que la cadena de igualdades muestra que $L^*(\hat{\eta}|x) = L^*(\tau(\theta)|x)$ y que $\tau(\hat{\theta})$ es el EMV de $\tau(\theta)$. □

Este teorema tampoco excluye la posibilidad de que θ sea un vector: si el EMV de $\theta = (\theta_1, ..., \theta_k)$ es $\hat{\theta} = (\hat{\theta_1}, ..., \hat{\theta_k})$ entonces para cualquier función τ del vector, el EMV de $\tau(\theta_1, ..., \theta_k)$ es $\tau(\hat{\theta_1}, ..., \hat{\theta_k})$. [Garthwaite et al., 2002]

1.4. Desigualdad de Chebyshev

Antes de probarla, necesitamos probar la siguiente proposición.

Proposición 1.6. ***Desigualdad de Markov****. Para una variable aleatoria X y un número real $a > 0$,*

$$P(|X| \geq a) \leq \frac{E\left[|X|\right]}{a}$$

Demostración. Para todo $a > 0$, $aI_{[|X|\geq a]}(X) \leq |X|$, así que se tiene que $E[aI_{[|X|\geq a]}] \leq E[|X|]$ y por las propedades de la esperanza, tenemos:

$$E[aI_{[|X|\geq a]}] = aE[I_{[|X|\geq a]}] = aP(|X| \geq a)$$

y tenemos así que

$$aP(|X| \geq a) \leq E\left[|X|\right]$$

y se puede despejar la a pues es mayor que o, concluyendo la demostración. □

Ahora sí estamos en condiciones de probar la desigualdad de Chebyshev.

Proposición 1.7. ***Desigualdad de Chebyshev****. Para una variable aleatoria X y $a > 0$,*

$$P(|X - E[X]| \geq a) \leq \frac{Var[X]}{a^2}$$

Demostración. Escribimos como U a la variable aleatoria $(X - E[X])^2$. La desigualdad de Markov nos dice que, para $w > 0$, $P(|U| \geq w) \leq \frac{E[|U|]}{w}$.
Si establecemos $w = a^2$, $P(|U| \geq w) = P(|X - E[X]| \geq a)$ y tenemos que:

$$P(|X - E[X]| \geq a) \leq \frac{E[|U|]}{w} = \frac{Var[X]}{a^2}$$

□

[Forsyth, 2018]

2 Estadística Multivariante empleada

2.1. Primeras nociones

Rescatamos el concepto de **distribución conjunta** del capítulo anterior que se define involucrando la función de distribución:

$$R(x,y) = P(X \leq x, Y \leq y)$$

para todo par de números reales (x,y). Nos interesan en concreto los casos donde F es absolutamente continua, lo que quiere decir que la derivada parcial siguiente existe casi en todas partes (c.t.p.) (es decir no existe para un conjunto de medida nula):

$$\frac{\partial^2 F(x,y)}{\partial x \partial y} = f(x,y)$$

y

$$F(x,y) = \int_{-\infty}^{y}\int_{-\infty}^{x} f(u,v)dudv$$

donde f indica la **función de densidad**.
Estas definiciones pueden extrapolarse al caso de p variables aleatorias $X_1, X_2, ..., X_p$. La función de distribución es

$$F(x_1, ..., x_p) = P(X_1 \leq x_1, ..., X_p \leq x_p)$$

para cualquier conjunto de números reales $x_1, ..., x_p$. La función de densidad absolutamente continua (c.t.p.) es

$$\frac{\partial^p F(x_1, ..., x_p}{\partial x_1, ..., \partial x_p}$$

y

$$F(x_1, ..., x_p) = \int_{-\infty}^{x_p} \cdots \int_{-\infty}^{x_1} f(u_1, ..., u_p)du_1 \cdots du_p$$

Los **momentos conjuntos** se definen como

$$\mathcal{E} X_1^{h_1} \cdots X_p^{h_p} = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} x_1^{h_1} \cdots x_p^{h_p} f(x_1, ..., x_p)dx_1 \cdots dx_p$$

La **distribución marginal** de r variables aleatorias con $r < p$ es:

$$P(X_1 \leq x_1, ..., X_r \leq x_r) = P(X_1 \leq x_1, ..., X_r \leq x_r, X_{r+1} \leq \infty, ..., X_p \leq \infty) =$$

$$= F(x_1, ..., x_r, \infty, ..., \infty)$$

y la **densidad marginal** de $X_1, ..., X_r$ es

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x_1, ..., x_r, u_{r+1}, ..., u_p)du_{r+1} \cdots du_p$$

Los momentos conjuntos pueden obtenerse de la distribución marginal, por ejemplo:

$$\mathcal{E}X_1^{h_1}\cdots X_r^{h_r} = \mathcal{E}X_1^{h_1}\cdots X_r^{h_r}X_{r+1}^0\cdots X_p^0 = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} x_1^{h_1}\cdots x_r^{h_r} f(x_1,...,x_p)dx_1\cdots dx_p =$$

$$= \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} x_1^{h_1}\cdots x_r^{h_r}\left[\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} f(x_1,...,x_p)dx_{r+1}\cdots dx_p\right]dx_1\cdots dx_r$$

Si la función de distribución de $X_1,...,X_p$ es $F(x_1,...,x_p)$, el conjunto de variables aleatorias se dice que es **mutuamente independientes** si

$$F(x_1,...,x_p) = F_1(x_1)\cdots F_p(x_p),$$

donde $F_i(x_i)$ es la función de distribución marginal de X_i, i=1,...,p. El conjunto $X_1,...,X_r$ se dice que es **independiente** de $X_{r+1},...,X_p$ si

$$F(x_1,...,x_p) = F(x_1,...,x_r,\infty,...,\infty)F(\infty,...,\infty,x_{r+1},...,x_p).$$

Un resultado de la independencia es que los momentos conjuntos se multiplican. Por ejemplo, si tenemos que $X_1,...,X_p$ son mutamente independientes, entonces

$$\mathcal{E}X_1^{h_1}\cdots X_p^{h_p} = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} x_1^{h_1}\cdots x_p^{h_p} f(x_1,...,x_p)dx_1\cdots dx_p =$$

$$= \prod_{i=1}^{p}\int_{-\infty}^{\infty} x_i^{h_i} f_i(x_i)dx_i = \prod_{i=1}^{p}\{\mathcal{E}X_i^{h_i}\}$$

Si la función de distribución de $X_1,...,X_p$ es $F(x_1,...,x_p)$, la densidad condicional de $X_1,...,X_r$ dados $X_{r+1}=x_{r+1},...,X_p=x_p$, es

$$\frac{f(x_1,...,x_p)}{\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} f(u_1,...,r_r,x_{r+1},...,x_p)du_1\cdots du_r}$$

Por último, vamos a hablar de la transformación de variables.
Sea $F(x_1,...,x_p)$ la función de densidad de $X_1,...,X_p$. Consideramos las p funciones reales $y_i = y_i(x_1,...,x_p)$ con $i = 1,...,p$.
Asumimos que la transformación del espacio de x al de y es una a una; la transformación inversa es $x_i = x_i(y_1,...,y_p)$ con $i = 1,...,p$. Sean las variables aleatorias $Y_1,...,Y_p$ definidas como $Y_i = y_i(X_1,...,X_p)$ con $i = 1,...,p$. La función de densidad de $Y_1,...,Y_p$ es

$$g(y_1,...,y_p) = f[x_1(y_1,...,y_p),...,x_p(y_1,...,y_p)]J(y_1,...,y_p),$$

donde $J(y_1,...,y_p)$ es el Jacobiano

$$J(y_1,...,y_p) = mod\begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} & \cdots & \frac{\partial x_1}{\partial y_p} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} & \cdots & \frac{\partial x_2}{\partial y_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_p}{\partial y_1} & \frac{\partial x_p}{\partial y_2} & \cdots & \frac{\partial x_p}{\partial y_p} \end{vmatrix}$$

Donde asumimos que la derivada existe y donde 'mod' se refiere al módulo del determinante. [Anderson et al., 1958]

2.2. Propiedades de matrices definidas positivas

La forma cuadrática de una matriz la definimos como

$$x^T Ax = \sum_{i,j=1}^{p} a_{ij} x_i x_j,$$

donde x^T es el vector $(x_1, ..., x_p)$ y $A = (a_{ij})$ es una matriz simétrica. Se dice que A y la forma cuadrática son **semidefinidas positivas** si $x^T Ax \geq 0$ para todo x. Si $x^T Ax > 0$ para todo $x \neq 0$ entonces A y la forma cuadrática se llaman **definidas positivas** que implica que la matriz es simétrica.

Teorema 2.1. *Si C es una matriz cuadrada de p filas y columnas definida positiva, y si B una matriz con p filas y q columnas, $q \leq p$, es de rango q, entonces $B^T CB$ es definida positiva*

Demostración. Dado un vector $y \neq 0$, sea $x = BY$. Como B es de rango q, $By = x \neq 0$. Entonces

$$y^T (B^T CB) y = (By)^T C(By) = x^T Cx > 0$$

y la prueba concluye observando que $B^T CB$ es simétrica. Notar que sólo se cumple cuando B es de rango q pues en otro caso, existiría $y \neq 0$ tal que $By = 0$. [Anderson et al., 1958] □

La última afirmación de la demostración anterior nos lleva al siguiente corolario:

Corolario 2.1. *Si C es definida positiva y B es no singular, entonces $B^T CB$ es definida positiva.*

Corolario 2.2. *Si C es definida positiva entonces C^{-1} es definida positiva.*

Demostración. C debe ser no singular. Si fuera singular, ocurriría que $Cx = 0$ para algún $x \neq 0$, entonces $x^T Cx = 0$ lo que contradice la hipótesis de que c es definida positiva. Sea B en Teorema 2.1 igual a C^{-1}. Entonces $B^T CB = (C^{-1})^T CC^{-1} = (C^{-1})^T$. Trasponiendo $CC^{-1} = I$, tenemos que $(C^{-1})^T C^T = (C^{-1})^T C = I$. Por lo que $C^{-1} = (C^{-1})^T$. □

Corolario 2.3. *La matriz $q \times q$ formada por la eliminación de $p - q$ filas de una matriz definida positiva C y las correspondientes $p - q$ columnas de C es definida positiva.*

Demostración. Se deduce del Teorema 2.1 formando B a través de coger la identidad $p \times p$ y borrando las columnas correspondientes a las borradas de C. □

Teorema 2.2. *Si A es no singular, existe una matriz triangular inferior no singular F tal que $FA = A^*$ es una matriz triangular superior no singular.*

Demostración. Sea $A = A_1$. Se define recursivamente $A_G = (a_{ij}^{(g)}) = F_{g-1} A_{g-1}, g = 2, ..., p$, donde $F_{g-1} = (f_{ij}^{(g-1)})$ tiene los elementos

$$f_{jj}^{(g-1)} = 1,\ j = 1, ..., p,$$

$$f_{i,g-1}^{(g-1)} = \frac{a_{i,g-1}^{(g-1)}}{a_{g-1,g-1}^{(g-1)}},\ i = g, ..., p$$

$$f_{ij}^{(g-1)} = 0, \text{en otro caso}$$

Entonces

$$a_{ij}^{(g)} = 0,\ i = j+1, ..., p, j = 1, ..., g-1,$$

$$a_{ij}^{(g)} = a_{ij}^{(g-1)},\ i = 1, ..., g-1, j = 1, ..., p,$$

$$a_{ij}^{(g)} = a_{ij}^{(g-1)} + f_{i,g-1}^{(g-1)} a_{g-1,j}^{(g-1)} = a_{ij}^{(g-1)} - \frac{a_{i,g-1}^{(g-1)} a_{g-1,j}^{(g-1)}}{a_{g-1,g-1}^{(g-1)}}, i, j = g, ..., p.$$

Y vemos que $F = F_{p-1}, ..., F_1$ es triangular inferior y los elementos de A_g en las primeras $g-1$ columnas bajo la diagonal son 0; en particular, $A^* = FA$ es triangular superior. De $|A| \neq 0$ y $|F_{g-1}| = 1$, tenemos que $|A_{g-1}| \neq 0$. Por tanto, $a_{11}^{(1)}, ..., a_{g-2,g-2}^{(g-2)}$ son diferentews de 0 y las últimas $p-g$ columnas de A_{g-1} pueden ser numeradas para que $a_{g-1,g-1}^{(g-1)} \neq 0$; entonces $f_{i,g-1}^{(g-1)}$ está bien definida. □

La ecuación $FA = A^*$ puede ser resuelta para obtener $A = LR$, donde $R = A^*$ es triangular superior y $L = F^{-1}$ es triangular inferior y tiene 1 en la diagonal principal (puesto que F también los tiene). Esto se conoce como la *descomposición LR*.

Corolario 2.4. *Si A es definida positiva, existe una matriz triangular inferior no singurar F tal que FAF^T es diagonal y positiva definida.*

Demostración. Del Teorema 2.2, esiste una triangular inferior no singular F tal que FA es tiangular superior y no singular, entonces, FAF^T es triangular superior y simétrica y por lo tanto, diagonal. □

Corolario 2.5. *El determinante de una matriz definida positiva A es positivo.*

Demostración. De la construcción de FAF^T,

$$FAF^T = \begin{pmatrix} a_{11}^{(1)} & 0 & \cdots & 0 \\ 0 & a_{22}^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{pp}^{(p)} \end{pmatrix}$$

□

es definida positiva y por tanto $a_{gg}^{(g)} > 0, g = 1, ..., p$, y $0 < |FAF^T| = |F||A||F| = |A|$.

Corolario 2.6. *Si A es definida positiva, existe una matriz triangular inferior G tal que $GAG^T = I$*

Demostración. Sea $FAF^T = D^2$, y sea D la matriz diagonal cuyos elementos de la diagonal son las raíces cuadradas positivas de los elementos de la diagonal de D^2. Entonces tomando $C = D^{-1}F$ se termina la prueba. □

Corolario 2.7. ***Descomposición de Cholsky.*** *Si A es definida positiva, existe una matriz triangular superior única T ($t_{ij} = 0, i < j$) con elementos positivos en la diagonal tal que $A = TT^T$*

Demostración. Del Corolario 2.6, $A = G^{-1}(G^T)^{-1}$, donde G es triangular inferior. Entonces $T = G^{-1}$ es triangular inferior. [Anderson et al., 1958] □

2.3. Distribución normal multivariante

La función de densidad univariante puede escribirse como

$$ke^{-\frac{1}{2}\alpha(x-\beta)^2} = ke^{-\frac{1}{2}(x-\beta)\alpha(x-\beta)}$$

donde α es positivo y k es elegida de forma que la función de distribución integrada el todo el eje x sea la unidad. La función de densidad de una distribución normal de $X_1, ..., X_p$ tiene una forma análoga. La variable escalar x es sutituida por un vector $x = (x_1, ..., x_p)^T$, el escalar β por un vector $b = (b_1, ..., b_p)^T$ y la constante positiva α por una matriz definida positiva (simétrica)

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pp} \end{pmatrix}$$

El cuadrado $\alpha(x-\beta)^2 = (x-\beta)\alpha(x-\beta)$ es reemplazado por la forma cuadrática

$$(x-b)^T A(x-b) = \sum_{i,j=1}^{p} a_{ij}(x_i - b_i)(x_j - b_j).$$

Por tanto, la función de densidad de una normal p-variante es

$$f(x_1, ..., x_p) = Ke^{-\frac{1}{2}(x-b)^T A(x-b)},$$

donde $K > 0$ es elegida de forma que la integral sobre al espacio euclídeo p-dimensional de $x_1, ..., x_p$ sea la unidad. Vamos a determinar esa K, evaluando

$$K^* = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} e^{-\frac{1}{2}(x-b)^T A(x-b)} dx_p \cdots dx_1$$

Para ello, vamos a usar el Corolario 2.6 que dice que si A es definida positiva, existe una matriz no singular C tal que $C^T AC = I$, donde I denota la identidad y C^T la traspuesta de C.
Sea $x - b = Cy$ donde $y = (y_1, ..., y_p)^T$, entonces $(x-b)^T A(x-b) = y^T C^T ACy = y^T y$. El jacobiano de la transformación es $J = mod|C|$. Por tanto, K^* se vuelve:

$$K^* = mod|C| \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} e^{-\frac{1}{2}y^T y} dy_p, ..., dy_1.$$

Tenemos que

$$e^{-\frac{1}{2}y^T y} = exp\left(-\frac{1}{2}\sum_{i=1}^{p} y_i^2\right) = \prod_{i=1}^{p} e^{-\frac{1}{2}y_i^2},$$

y así podemos escribir K^* como:

$$K^* = mod|C| \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} e^{-\frac{1}{2}y_1^2} \cdots e^{-\frac{1}{2}y_p^2} dy_p, ..., dy_1 = mod|C| \prod_{i=1}^{p} \{\int_{-\infty}^{\infty} e^{-\frac{1}{2}y_i^2} dy_i\} =$$

$$= mod|C| \prod_{i=1}^{p} \{\sqrt{2\pi}\} = mod|C|(2\pi)^{\frac{1}{2}p}$$

en virtud de que

$$\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{\frac{1}{2}t^2}dt = 1$$

Y tomamos el determinante de las matrices deducidas del Corolario 2.6 $|C^T||A||C| = |I|$ y como $|C| = |C|$ y $|I| = 1$, deducimos que $mod|C| = \frac{1}{\sqrt{|A|}}$.
Por lo tanto, $K = \frac{1}{K^*} = \sqrt{|A|}(2\pi)^{-\frac{1}{2}p}$.
La función de densidad normal es

$$\frac{\sqrt{|A|}}{(2\pi)^{\frac{1}{2}p}} e^{-\frac{1}{2}(x-b)^T A(x-b)}.$$

[Anderson et al., 1958]

Vamos a definir ahora de forma general el concepto de matriz aleatoria y el vector aleatorio será un caso especial de la matriz aleatoria con una sola columna.

Definición 2.1. Una **matriz aleatoria** Z es una matriz $Z = (Z_{gh})$, $g = 1, ..., m$, $h = 1, ..., n$, de variables aleatorias $Z_{11}, ..., Z_{mn}$.

Si las variables aleatorias $Z_{11}, ..., Z_{mn}$ pueden tomar solo un finito número de valores, la matriz aleatoria Z puede ser una de las finitas matrices, $Z(1), ..., Z(q)$. Si la probabilidad de $Z = Z(i)$ es p_i, entonces definiríamos $\mathcal{E}Z$ como $\sum_{i=1}^{q} Z(i)p_i$. Entonces $\mathcal{E}Z = (\mathcal{E}Z_{gh})$.Si las variables aleatorias $Z_{11}, ..., Z_{mn}$ tienen densidad conjunta, entonces operando con las sumas de Riemann podemos definir $\mathcal{E}Z$ como el límite (si existe) de aproximar las sumas del tipo que ocurren en el caso discreto, de nuevo, $\mathcal{E}Z = (\mathcal{E}Z_{gh})$. Por lo tanto, en general, usamos la siguiente definición:

Definición 2.2. La **esperanza de una matriz aleatoria** Z es $\mathcal{E}Z = (\mathcal{E}Z_{gh})$, $g = 1, ..., m$, $h = 1, ..., n$.

En particular, si Z es un vector aleatorio X, la esperanza $\mathcal{E}X = (\mathcal{E}X_1, ..., \mathcal{E}X_p)^T$ es la media o el vector media de X que denotamos como μ. Si Z es $(X - \mu)(X - \mu)^T$, la esperanza es

$$\mathcal{C}(X) = \mathcal{E}(X - \mu)(X - \mu)^T = [\mathcal{E}(X_i - \mu_i)(X_j - \mu_j)]$$

la **matriz de covarianzas** de X, donde el i-ésimo elemento de la diagonal, $\mathcal{E}(X_i - \mu_i)^2$, es la varianza de X_i y el elemento i,j-ésimo fuera de la diagonal, $\mathcal{E}(X_i - \mu_i)(X_j - \mu_j)$, es la covarianza de X_i y X_j con $i \neq j$. Denotamos esta matriz por Σ. Notar que:

$$\mathcal{C}(X) = \mathcal{E}(XX^T - \mu X^T - X\mu^T + \mu\mu^T) = \mathcal{E}XX^T - \mu\mu^T.$$

La esperanza de una matriz o vector aleatorio cumple ciertas propiedades que podemos resumir en los siguientes lemas:

Lema 2.1. *Si Z es una matriz aleatoria $m \times n$, D es una matriz real $l \times m$, E una matriz real $n \times q$ y F otra matriz real $l \times q$, entonces*

$$\mathcal{E}(DZE + F) = D(\mathcal{E}E) + F.$$

Demostración. El elemento de la i-ésima fila y la j-ésima columna de $\mathcal{E}(DZE+F)$ es

$$\mathcal{E}\left(\sum_{h,g} d_{ih} Z_{hg} e_{gj} + f_{ij}\right) = \sum_{h,g} d_{ih}(\mathcal{E}Z_{hg}) e_{gj} + f_{ij},$$

que es el elemento i,j-ésimo de $D(\mathcal{E}Z)E+F$. □

Lema 2.2. *Si $Y = DX + f$, donde X es un vector aleatorio, entonces:*

$$\mathcal{E}Y = D\mathcal{E}X + f,$$

$$\mathcal{C}(Y) = D\mathcal{C}(X)D^T.$$

Demostración. La primera propiedad se deduce de forma directa del Lema 2.1.
Para la segunda,

$$\begin{aligned}\mathcal{C}(Y) &= \mathcal{E}(Y-\mathcal{E}Y)(Y-\mathcal{E}Y)^T \\ &= \mathcal{E}[DX+f-(D\mathcal{E}X+f)][DX+f-(D\mathcal{E}X+f)]^T \\ &= \mathcal{E}[D(X-\mathcal{E}X)][D(X-\mathcal{E}X)]^T = \mathcal{E}[D(X-\mathcal{E}X)(X-\mathcal{E}X)^T D^T]\end{aligned}$$

que lleva a la segunda propiedad por el Lema 2.1. □

Cuando la transformación corresponde a una normal, $X = CY + b$, entonces $\mathcal{E}X = C\mathcal{E}Y + b$. Por las transformaciones vistas con anterioridad, la función de densidad de Y es proporcional a:

$$\frac{1}{(2\pi)^{\frac{1}{2}p}} e^{-\frac{1}{2}yy^T} = \prod_{j=1}^{p}\{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y_j^2}\}.$$

La esperanza de la componente i-ésima de Y es

$$\mathcal{E}Y_i = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} y_i \prod_{j=1}^{p}\{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y_j^2}\} dy_1\cdots dy_p$$

$$= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} y_i e^{-\frac{1}{2}y_i^2} dy_i \prod_{j=1, j\neq i}^{p}\{\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y_j^2} dy_j\}$$

$$= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} y_i e^{-\frac{1}{2}y_i^2} dy_i = 0$$

donde la última igualdad sigue de que $y_i e^{-\frac{1}{2}y_i^2}$ es una función impar de y_i. También puede deducirse de que es la esperanza de una normal con media 0. Por tanto, $\mathcal{E}Y = 0$ y la media de X, denotada por μ, es $\mu = \mathcal{E}X = b$.
De la segunda propiedad del Lema 2.2, podemos ver que $\mathcal{C}(X) = C(\mathcal{E}YY^T)C^T$. El elemento i,j-ésimo de $\mathcal{E}YY^T$ es

$$\mathcal{E}YY^T = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} y_i y_j \prod_{h=1}^{p}\{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y_h^2}\} dy_1\cdots dy_p.$$

Si $i = j$ tenemos que

$$\mathcal{E}Y^2 = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} y_i^2 e^{-\frac{1}{2}y_i^2} dy_i \prod_{h=1, h\neq i}^{p}\{\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y_h^2} y_h\}$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y_i^2 e^{-\frac{1}{2}y_i^2} dy_i = 1$$

La última igualdad se deduce de que la expresión antes de la final es la esperanza del cuadrado de una variable con ditribución normal de media 0 y varianza 1. Si $i \neq j$ la expresión se convierte en

$$\mathcal{E}Y_iY_j = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y_i e^{-\frac{1}{2}y_i^2} dy_i \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y_j e^{-\frac{1}{2}y_j^2} dy_j \prod_{h=1,h\neq i,j}^{p} \{\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}y_h^2} y_h\} = 0$$

ya que la primera integral da 0. Podemos resumir los dos casos como $\mathcal{E}YY^T = I$. Y por lo tanto

$$\mathcal{E}(X-\mu)(X-\mu)^T = CIC^T = CC^T.$$

Por el Corolario 2.6 obtenemos que $A = (C^T)^{-1}C^{-1}$ multiplicando $(C^T)^{-1}$ en la parte izquierda y C^{-1} en la derecha. Tomando inversas en ambos lados de la igualdad nos da $CC^T = A^{-1}$.
Por lo tanto, la matriz de covarianzas de X es

$$\Sigma = \mathcal{E}(X-\mu)(X-\mu)^T = A^{-1}.$$

y se deduce que es definida poritiva. Resumimos los resultados en el siguiente teorema.

Teorema 2.3. *Si la función de densidad de un vector aleatorio X es la de una normal, entonces la esperanza de X es b y la matriz de covarianzas es A^{-1}. De forma inversa, dados un vector μ y una matriz definida positiva Σ, hay una función de densidad norma multivariante*

$$(2\pi)^{-\frac{1}{2}p} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

tales que la esperanza del vector con esta función de densidad es μ y la matriz de covarianzas es Σ.

Denotamos a la función de densidad anterior como $n(x|\mu,\Sigma)$ y a la distribución, $N(\mu,\Sigma)$.

3 Aprendizaje automático

Definiremos los modelos utilizados en el trabajo y nociones previas necesarias para entender el aprendizaje automático. En la Sección 3.2 se plantea un escenario básico de aprendizaje y se dan las primeras definiciones, en la Sección 3.3 se definen los errores al ajustar un modelo así como el llamado *compromiso sesgo-varianza* y en Sección 3.4 se da la definición general y matemática de los modelos utilizados en el estudio.

3.1. Nociones previas

El aprendizaje automático es una aplicación de la inteligencia artificial que hace uso de los algoritmos de aprendizaje que estiman una dependencia desconocida entre los datos de entrada y de salida de un sistema, desde unas muestras conocidas. Una vez esta dependencia ha sido estimada, se puede usar para predecir las futuras salidas del sistema a partir de los valores de entrada conocidos.
Hay tres tipos de aprendizajes:

- Aprendizaje **supervisado**: Es el tipo de aprendizaje más utilizado. Se aplica a datos **etiquetados**. Tiene como objetivo aprender la función desconocida que asigna los datos con las etiquetas minimizando una función de error. lo usaremos para nuestro conjunto de datos, pues es etiquetado.
- Aprendizaje **no supervisado**: Se usa para los datos no etiquetados y se pretende buscar patrones o relaciones entre los datos. Un ejemplo típico de este tipo de aprendizaje es el de **agrupamiento** o *clustering* que busca agrupar los datos según algún patrón o relación.
- Aprendizaje **por refuerzo**: En vez de ser etiquetado, da solo una indicación de si la predicción es correcta o no mediante alguna recompensa o penalización en función de las acciones que haga.

[Jordan and Mitchell, 2015]
Y hay dos tipos de problemas: regresión, donde se predicen etiquetas con valores continuos, y clasificación donde los valores de las etiquetas son discretos y finitos.

3.2. Esquema básico de aprendizaje

Un primer ejemplo y esquema de aprendizaje automático sería el siguiente, supongamos que hay un banco que recive miles de solicitudes para una tarjeta de crédito y quiere automatizar el proceso. No se conoce ninguna fórmula para aceptar o denegar una tarjeta pero se dispone de gran cantidad de datos de los aplicantes: información personal, sueldo, prestamos pendientes y más información relacionada al crédito, así que se usan estos datos para encontrar una fórmula adecuada.
Así, se tiene la entrada x (información del cliente), la función desconocida ideal para llevar

a cabo la decisión $f : X \to Y$, donde X es el espacio de la entrada (conjunto de las posibles entradas de $\mathbf{x}$), e Y es el espacio de la salida (conjunto de todas las posibles salidas, en este caso una decisión de sí o no). Se denotará al set de datos por D, con ejemplos de entrada-salida $(x_1, y_1), ..., (x_N, y_N)$ donde $y_n = f(x_n)$ para $n = 1, ..., N$. Los ejemplos se suelen llamar muestras. Por último, hay un algoritmo de aprendizaje que aproxima $h : X \to Y$ a f. El algoritmo elige h de entre un conjunto de funciones candidatas o hipótesis, H, como por ejemplo el conjunto de todas las funciones lineales.
El banco usará la función h para hacer la decisión puesto que la f ideal no la tiene así que el algoritmo elige la h que mejor aproxima f en un conjunto de *entrenamiento* o *training* de anteriores clientes. [Abu-Mostafa et al., 2012] Este conjunto es un subconjunto de D o D al completo, y se usa para entrenar o ajustar el modelo a él, de ahí su nombre. Luego se miden las capacidades del modelo en un conjunto de *test* o *prueba* diferente al de entrenamiento mediante unas *métricas* que miden la bondad del mismo en dicho conjunto.
Los datos suelen presentarse en forma de tabla donde las filas son los individuos de la población a las columnas se las llama **variables** o **características**.
La siguiente figura ilusta esquemáticamente el problema de aprendizaje:

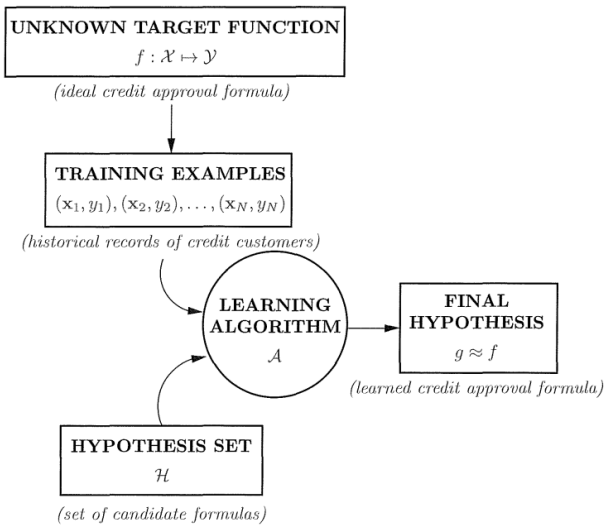

Figura 3.1: Esquema básico de aprendizaje. [Abu-Mostafa et al., 2012]

Para este trabajo, $X = \mathbb{R}^d$, siendo $\mathbb{R}^d$ un espacio euclídeo d-dimensional, con las coordenadas descritas en el Capítulo 4 siendo cada columna una coordenada, $Y = \{0, 2\}$ una decisión binaria de si el individuo es sano o afectado (o padre de afectado para el estudio con controles).

3.3. Errores

También hay que tener en cuenta dos conceptos importantes: los errores o funciones de pérdida. Hay de dos tipos, el primero es el error dentro de la muestra definido como la fracción de D donde f y h no concuerdan:

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N}[[h(x_n) \neq f(x_n)]]$$

donde el operador [[·]]=1 si lo de dentro se cumple y 0 en caso contrario. Análogamente también está el error fuera de la muestra:

$$E_{out}(h) = \mathbb{P}[h(x) \neq f(x)]$$

la probabilidad está basada en la distribución de X que se usa para muestrear.
Para cuantificar la relación que existe entre estos dos errores se usa la *desigualdad de Hoeffding*, que provee de una cota superior de la probabilidad de que la suma de variables aleatorias independientes y acotadas se desvíe de su valor esperado más de una cierta cantidad. Esta desigualdad indica que para cualquier tamaño de muestra N,

$$\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N} \quad \forall \epsilon > 0$$

El error dentro de la muestra E_{in} es una variable aleatoria que depende de la muestra y el error fuera de la muestra E_{out} es desconocido pero no aleatorio. Ahora bien, esto ocurre bajo el supuesto de que h está fija y es una en particular antes de generar el conjunto de datos. [Abu-Mostafa et al., 2012]

3.3.1. Generalización de errores I

Para solucionar que h tenga que estar **fija**, consideramos un conjunto de hipótesis H y asumimos primero que tien un número finito de hipótesis $H = \{h_1, ..., h_M\}$. El objetivo es poder poner el límite a una hipótesis g de H que no esté fija antes de generar los datos, porque la hipótesis seleccionada g depende de los datos. Así que vamos a intentar limitar $\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon]$ de forma que no dependa de la g que el algoritmo de aprendizaje elija. Como g es una de las hipótesis independientemente del algoritmo y de la muestra, siempre es verdad que: "$|E_{in}(g) - E_{out}(g)| > \epsilon$"$\Longrightarrow$ "$|E_{in}(h_1) - E_{out}(h_1)| > \epsilon$ **or** $|E_{in}(h_2) - E_{out}(h_2)| > \epsilon$ **or** ... **or** $|E_{in}(h_M) - E_{out}(h_M)| > \epsilon$".
Llamemos $\mathcal{B}_1$ a la parte izquierda de la implicación anterior y $\mathcal{B}_2$ a la parte derecha. $\mathcal{B}_2$ tiene la propiedad deseada: las hipótesis h_m están fijas, y aplicamos dos reglas básicas de probabilidad:

$$\text{si } \mathcal{B}_1 \Longrightarrow \mathcal{B}_2 \text{ entonces } \mathbb{P}[\mathcal{B}_1] \leq \mathbb{P}[\mathcal{B}_2]$$

y que si $\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_M$ son cualesquiera eventos, entonces:

$$\mathbb{P}[\mathcal{B}_1 \text{ or } ... \text{ or } \mathcal{B}_M] \leq \mathbb{P}[\mathcal{B}_1] + ... + \mathbb{P}[\mathcal{B}_M]$$

Tenemos con estas dos propiedades que:

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq \mathbb{P}[|E_{in}(h_1) - E_{out}(h_1)| > \epsilon \textbf{ or } |E_{in}(h_2) - E_{out}(h_2)| > \epsilon \textbf{ or } ...$$

$$\textbf{or } |E_{in}(h_M) - E_{out}(h_M)| > \epsilon] \leq \sum_{m=1}^{M}\mathbb{P}[|E_{in}(h_m) - E_{out}(h_m)| > \epsilon]$$

Aplicando la desigualdad de Hoeffding a los M términos de la sumatoria obtenemos el resultado deseado:

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

Como se puede ver, esto funciona cuando M es finita. [Abu-Mostafa et al., 2012]

3.3.2. Generalización de errores II

Para terminar esta sección se va a definir la *función de crecimiento* o, más conocida en inglés, ***growth function*** que formaliza el número efectivo de hipótesis.
Antes vamos a convertir la formula $\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$ en algo más conveniente. Tomamos un *nivel de tolerancia* δ y afirmamos con una probabilidad de al menos $1 - \delta$ que:

$$E_{out} \leq E_{in}(g) + \sqrt{\frac{1}{2N} ln \frac{2M}{\delta}}$$

A esta cota se la llama *cota de generalización* pues acota E_{out} en términos de E_{in}. Se ha conseguido la expresión de la siguiente forma: con al menos una probabilidad de $1 - 2Me^{-2N\epsilon^2}$, $|E_{out} - E_{in}| \leq \epsilon$ lo que implica que $E_{out} \leq E_{in} + \epsilon$. Escribimos $\delta = 2Me^{-2N\epsilon^2}$, de donde $\epsilon = \sqrt{\frac{1}{2N} ln \frac{2M}{\delta}}$ y se deduce la cota de generalización.

Definición 3.1. Sean $x_1, ..., x_N \in X$. La dicotomía generada por H en estos puntos está definida por

$$H(x_1, ..., x_N) = \{(h(x_1), ..., h(x_n))|h \in H\}$$

Es una dicotomía pues separa $x_1, ..., x_n$ en dos grupos: los que su h vale 1 y los que vale -1, así, cuanto mayor es la dicotomía, más 'diversa' es H.

Definición 3.2. La **función de crecimiento** está definida por un conjunto de hipótesis H por

$$m_H(N) = \max_{x_1,...,x_N \in X} |H(x_1, ..., x_N)|$$

donde $|\cdot|$ indica la cardinalidad del conjunto.

En otras palabras, $m_H(N)$ es el máximo número de dicotomías que pueden ser generadas por H en cualquiera N puntos.
Para cualquier H, como $H(x_1, ..., x_N) \subseteq \{-1, +1\}$, el valor de $m_H(N)$ es como mucho $|\{-1, +1\}|$, así que $m_H(N) \leq 2^N$. Por intuición se puede afirmar que $m_H(N)$ crece más rápido cuanto más compleja se vuelve H, que es lo que se espera ya que es lo que va a reemplazar a M en la cota de generalización. Si H es capaz de generar todas las posibles dicotomías en la muestra, se dice que *separa*, o en inglés ***shatter***, $x_1, ..., x_N$
Sin embargo, no es práctico computar m_H para cada conjunto de hipótesis que se use, pero esto no es necesario pues basta con usar una cota superior y la cota de generalización aún se sostendrá y así se hará más facil computar m_H. [Abu-Mostafa et al., 2012]

Definición 3.3. Si el conjunto nungún conjunto de datos de tamaño k puede ser *separado* por H, entonces k se dice que es un **punto de ruptura** para H.

Así, $m_H < 2^k$ y en general es más fácil de encontrar este punto que de computar la función de crecimiento completa para esa H. Lo más importante de este concepto es que si $m_H(N) = 2^N$ rompe en cualquier punto, podemos acotar m_H por un polinomio de forma

que si se sustituye por M en el *error de generalización* $\sqrt{\frac{1}{2N} ln \frac{2M}{\delta}}$ cuando $N \longrightarrow \infty$ irá a cero, cosa que no pasa sin esa cota polinómica.
Ahora se va a probar esa cota polinómica. Introducimos antes un concepto. No depende de H así que la cota será aplicada a cluaquier H.

Definición 3.4. Se denota $B(N,k)$ al número máximo de dicotomías en N puntos tales que ningún subconjunto de tamaño k del de N puntos puede ser *separado* por estas dicotomías.

Esta definición asume un punto de ruptura k y está definido como un máximo así que sirve como cota superior para cualquier $m_H(N)$ que tenga un punto de ruptura k:

$$m_H(N) \leq B(N,k) \text{ si } k \text{ es un punto de ruptura para } H$$

Lema 3.1. *(Lema de Sauer)*

$$B(N,k) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

Demostración. Cuando $k = 1$, $B(N,1) = 1$ para todo N pues ningún subconjunto de tamaño 1 puede ser separado así que sólo admite una dicotomía así que el lema se cumple. Cuando $N = 1$, $B(1,k) = 2$ para $k > 1$ puesto que no existen subconjuntos de tamaño k así que las dos únicas posibilidades son 1 y -1, con lo que el lema se cumple. Ahora se va a realizar la prueba por inducción, asumimos que es cierto para todo $N \leq N_0$ y todo k. Necesitamos probar que es verdad para $N = N_0 + 1$ y todo k. Como la desigualdad ya es cierta cuando $k = 1$ solo hace flata preocuparse por $k \geq 2$. Por definición de B, ocurre que

$$B(N_0 + 1, k) \leq B(N_0, k) + B(N_0, k-1)$$

Aplicando la hipótesis de inducción a cada término de la derecha de la desigualdad se tiene que:

$$B(N_0 + 1, k) \leq \sum_{i=0}^{k-1} \binom{N_0}{i} + \sum_{i=0}^{k-2} \binom{N_0}{i} =$$

$$= 1 + \sum_{i=0}^{k-1} \binom{N_0}{i} + \sum_{i=0}^{k-1} \binom{N_0}{i-1} = 1 + \sum_{i=0}^{k-1} \left[\binom{N_0}{i} + \binom{N_0}{i-1} \right] =$$

$$= 1 + \sum_{i=0}^{k-1} \binom{N_0+1}{i} = \sum_{i=0}^{k-1} \binom{N_0+1}{i}$$

donde se ha usado la identidad de combinatoria $\binom{N_0+1}{i} = \binom{N_0}{i} + \binom{N_0}{i-1}$. Probado esto, la inducción concluye y el lema es cierto para todo N y k. □

Como $B(N,k)$ es una cota superior para $M_H(N)$, el siguiente teorema está desmotrado:

Teorema 3.1. *Si $m_H(k) < 2^k$ para algún k, entonces*

$$m_H(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

para todo N.

Así, tenemoos una cota polinómica de grado $k-1$ para $m_H(N)$.[Abu-Mostafa et al., 2012]

Ahora presentamos una nueva definición qeu caracterizará a la función de crecimiento.

Definición 3.5. La **dimensión Vapnik-Chervonenkis** de un conjunto de hipótesis H, denotado por $d_{VC}(H)$ o simplemente d_{VC}, es el mayor valor de N para el que $m_H(N) = 2^N$. Si $m_H(N) = 2^N$ para todo N, entonces $d_{VC}(H) = \infty$.

Si d_{VC} es la dimensión VC de H, entonces $k = d_{VC} + 1$ es un punto de ruptura para m_H. Por tanto, podemos escribir el Teorema 3.1 como:

$$m_H(N) \leq \sum_{i=0}^{d_{VC}} \binom{N}{i}$$

Y la dimensión VC es el orden de la cota polinómica.
Para terminar la sección, vamos a introducir la cota donde se reemplaza M por $m_H(N)$ con ciertos ajustes, el resultado matemático más importante de la teoría de aprendizaje. [Abu-Mostafa et al., 2012]
Denotemos P_{xy} a la distribución que sigue cada (x, y) de D. Para la demostración se usarán las etiquetas 0 y 1 en vez de -1 y 1.

Teorema 3.2. ***Cota de generalización VC.*** *Para una clasificación binaria tenemos y una función de pérdida (o error):*

$$P\left(\sup_{h\in H} |E_{in}(h) - E_{out}(h)| > \epsilon\right) \leq 8 m_H(N) e^{-N\epsilon^2/32},$$

y

$$E\left[\sup_{h\in H} |E_{in}(h) - E_{out}(h)|\right] \leq 2\sqrt{\frac{log(m_H(N)) + log2}{N}}$$

Demostración. La segunda expresión es consecuencia directa de la primera, así que vamos a demostrar la primera.
La prueba consistirá en varios pasos. Se usará para la demostración una *"muestra fantasma"* para ayudar a desarrollarla pero no jugará ningún papel en el resultado final, es decir, no se necesita esa muestra fantasma para aplicar el resultado. La definimos como $D' = \{(x'_1, y'_1), ..., (x'_n, y'_n)\}$, un conjunto de variables aleatorias independientes de D_n tales que $(x'_i, y'_i) \sim^{i.i.d.} P_{xy}$. El error bajo esta muestra es:

$$E'_{in}(h) = \frac{1}{N}\sum_{i=0}^{N}[[h(x'_i) \neq y'_i]]$$

Para el resto de la demostración asumiremos que $N\epsilon^2 \geq 2$ sin pérdida de genereladidad pues de otra forma, la cota sería trivial.
Paso 1: Primera simetrización por una muestra fantasma:
Mostraremos que

$$P\left(\sup_{h\in H} |E_{in}(h) - E_{out}(h)| > \epsilon\right) \leq 2P\left(\sup_{h\in H} |E_{in}(h) - E'_{in}(h)| > \frac{\epsilon}{2}\right)$$

Empezamos por definir $\tilde{h}(D) \equiv \tilde{h}$ que es un elemento de H tal que $|E_{in}(h) - E_{out}(h)| > \epsilon$ si tal elemento existe, si no, $\tilde{h}$ es un elemento arbitrario de H. Se podría pensar en $\tilde{h}$ como

$$\tilde{h} \approx arg \max_{h \in H} |E_{in}(h) - E_{out}(h)|$$

aunque no está bien definido porque puede no haber tal elemento en H alcanzando el máximo, basta para nuestro propósito. Notar que $\tilde{h}$ es una función de D.
Ahora vamos a probar la desigualdad de este paso fijándonos en la parte derecha.

$$P\left(\sup_{h \in H} |E_{in}(h) - E'_{in}(h)| > \frac{\epsilon}{2}\right) \geq P\left(|E_{in}(\tilde{h}) - E'_{in}(\tilde{h})| > \frac{\epsilon}{2}\right)$$

$$P\left(|E_{in}(\tilde{h}) - E_{out}(\tilde{h})| > \epsilon \text{ y } |E'_{in}(\tilde{h}) - E_{out}(\tilde{h})| < \frac{\epsilon}{2}\right)$$

$$= E\left[[[|E_{in}(\tilde{h}) - E_{out}(\tilde{h})| > \epsilon]][[|E'_{in}(\tilde{h}) - E_{out}(\tilde{h})| < \frac{\epsilon}{2}]]\right]$$

$$= E\left[[[|E_{in}(\tilde{h}) - E_{out}(\tilde{h})| > \epsilon]] E\left[[[|E'_{in}(\tilde{h}) - E_{out}(\tilde{h})| < \frac{\epsilon}{2}]] \middle| D\right]\right]$$

$$= E\left[[[|E_{in}(\tilde{h}) - E_{out}(\tilde{h})| > \epsilon]] P\left(|E'_{in}(\tilde{h}) - E_{out}(\tilde{h})| < \frac{\epsilon}{2} \middle| D\right)\right]$$

donde la segunda desigualdad viene de que dados cualesquiera reales x, y, z

$$|x - z| < \epsilon \text{ y } |y - z| \leq \frac{\epsilon}{2} \rightarrow |x - y| \geq \frac{\epsilon}{2}$$

Ahora, condicionado a D vemos que

$$|E'_{in}(\tilde{h}) - E_{out}(\tilde{h})| = \frac{1}{N} \sum_{i=1}^{N} U_i$$

donde $Ui = [[\tilde{h}(x'_i) \neq y'_i]] - E\left[[[\tilde{h}(x'_i) \neq y'_i]]|D\right]$ son variables aleatorias i.i.d. de media cero. Usaremos ahora la desigualdad de Chebyshev (probada en la Sección 1.4).

$$P\left(|E'_{in}(\tilde{h}) - E_{out}(\tilde{h})| < \frac{\epsilon}{2} \middle| D\right) = P\left(|\frac{1}{N} \sum_{i=1}^{N} U_i| < \frac{\epsilon}{2} \middle| D\right)$$

$$= P\left(|\sum_{i=1}^{N} U_i| < \frac{N\epsilon}{2} \middle| D\right) \geq 1 - \frac{4}{N^2\epsilon^2} Var\left(|\sum_{i=1}^{N} U_i| \middle| D\right)$$

$$= 1 - \frac{4}{N^2\epsilon^2} NVar\left(U_i|D\right) \geq 1 - \frac{4}{N\epsilon^2}\frac{1}{4} = 1 - \frac{1}{N\epsilon^2} \geq \frac{1}{2}$$

ya que asumimos que $N\epsilon^2 \geq 2$. Finalmente,

$$P\left(\sup_{h \in H} |E_{in}(h) - E'_{in}(h)| > \frac{\epsilon}{2}\right) \geq E\left[[[|E_{in}(\tilde{h}) - E_{out}(\tilde{h})| > \epsilon]] P\left(|E'_{in}(\tilde{h}) - E_{out}(\tilde{h})| < \frac{\epsilon}{2} \middle| D\right)\right]$$

$$\geq \frac{1}{2} E\left[[[|E_{in}(\tilde{h}) - E_{out}(\tilde{h})| > \epsilon]]\right] = \frac{1}{2} P\left(|E_{in}(\tilde{h}) - E_{out}(\tilde{h})| > \epsilon\right)$$

$$\geq \frac{1}{2} P\left(\sup_{h\in H} |E_{in}(h) - E_{out}(h)| > \epsilon\right)$$

Concluyendo así la prueba del primer paso.
Paso 2:
Reescribimos la desigualdad de la parte derecha de la desigualdad del paso 1.

$$P\left(\sup_{h\in H} |E_{in}(h) - E'_{in}(h)| > \frac{\epsilon}{2}\right) = P\left(\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} [[h(x_i) \neq y_i]] - [[h(x'_i) \neq y'_i]]\right| > \frac{\epsilon}{2}\right)$$

Resaltar que $[[h(x_i) \neq y_i]]$ y $[[h(x'_i) \neq y'_i]]$ tienen la misma distribución y por lo tanto $[[h(x_i) \neq y_i]] - [[h(x'_i) \neq y'_i]]$ tiene media cero y dirtribución simétrica. Así que si permutamos de forma aleatoria los signos dentro del valor absoluto no cambiará la probabilidad. Vamos a introducir otra secuencia del tipo de la "muestra fantasma".
Sean $\sigma_1, ..., \sigma_N$ variables aleatorias i.i.d., independientes de D y D' tales que $P(\sigma_i = 1) = P(\sigma_i = -1) = 1/2$ para todo i. Estas son las llamadas variables aleatorias de Rademacher. A la luz de nuestras anteriores observaciones tenemos:

$$P\left(\sup_{h\in H} |E_{in}(h) - E'_{in}(h)| > \frac{\epsilon}{2}\right) = P\left(\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} [[h(x_i) \neq y_i]] - [[h(x'_i) \neq y'_i]]\right| > \frac{\epsilon}{2}\right)$$

$$= P\left(\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]] - [[h(x'_i) \neq y'_i]])\right| > \frac{\epsilon}{2}\right)$$

$$\leq P\left(\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4} \text{ o } \sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x'_i) \neq y'_i]])\right| > \frac{\epsilon}{4}\right)$$

$$\leq 2P\left(\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right)$$

donde la última desigualdad viene de la unión de las dos partes de la línea anterior. Así que en estos dos pasos hemos probado que

$$P\left(\sup_{h\in H} |E_{in}(h) - E_{out}(h)| > \epsilon\right) \leq 4P\left(\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right)$$

Dejando así la parte izquierda desigualdad del paso 1 en una cota de la suma de variables aleatorias i.i.d. con media cero. También se ha eliminado la dependencia de la *muestra frantasma* D'.
Paso3: Condicionando en D
Sean $x_1, ..., x_N \in X$ y $y_1, ..., y_N \in Y$ arbitrarios. Vamos a estudiar la expresión

$$\frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right|$$

donde la aleatoriedad depende únicamente del signo aleatorio σ_i. Como ya se ha visto, $(h(x_1), ..., h(x_N))$ puede tomar como mucho $m_H(N)$ dicotomías y por tanto, $[[h(x_1) \neq y_1]], ..., [[h(x_N) \neq$

$y_N]]$ puede tomar $m_H(N)$ valores diferentes. Sea $H_{x_1,...,x_N} \subseteq H$ el menor subconjunto de H tal que

$$H(x_1, ..., x_N) = H_{x_1,...,x_N}(x_1, ..., x_N)$$

donde, como antes, $H(x_1, ..., x_N) = \{(h(x_1), ..., h(x_N)|h \in H\}$. En otras palabras, $H_{x_1,...,x_N}$ es el menor subconjunto de H que genera todas las posibles dicotomías para D así que $|H(x_1, ..., x_N)| \leq m_H(N)$. Ahora se puede empezar a acotar:

$$P\left(\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right) = P\left(\max_{h\in H_{x_1,...,x_N}} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right)$$

$$P\left(\bigcup_{h\in H_{x_1,...,x_N}} \left\{\frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right\}\right) \leq \sum_{h\in H_{x_1,...,x_N}} P\left(\frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right)$$

$$\leq |H_{x_1,...,x_N}| \sup_{h\in H_{x_1,...,x_N}} P\left(\frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right)$$

$$\leq m_H(N) \sup_{h\in H_{x_1,...,x_N}} P\left(\frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right)$$

$$\leq m_H(N) \sup_{h\in H} P\left(\frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right)$$

Y ya solo queda el último paso para terminar la demostración.
Paso 4: Desigualdad de Hoeffding:
Primero, darse cuenta de que:

$$\frac{1}{N}\left|\sum_{i=1}^{N} \underbrace{\sigma_i([[h(x_i) \neq y_i]])}_{A_i}\right|$$

es el valor absoluto de la suma de N variables independientes A_i con media cero y acotadas entre -1 y 1 así que podemos aplicar la desigualdad de Hoeffding.

$$P\left(\frac{1}{N}\left|\sum_{i=1}^{N} A_i\right| > \frac{\epsilon}{4}\right) \leq P\left(\left|\sum_{i=1}^{N} A_i\right| > \frac{N\epsilon}{4}\right)$$

$$\leq 2e^{-\frac{2(N\frac{\epsilon}{4})^2}{\sum_{i=1}^{N}(\max_i A_i - \min_i A_i)^2}} \leq 2e^{-\frac{N^2\frac{\epsilon^2}{8}}{4N}} \leq 2e^{-\frac{n\epsilon^2}{32}}$$

Volvemos a la desigualdad demostrada en el paso 2

$$P\left(\sup_{h\in H} |E_{in}(h) - E_{out}(h)| > \epsilon\right) \leq 4P\left(\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right)$$

y acotamos la parte derecha:

$$P\left(\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}\right) = E\left[[\sup_{h\in H} \frac{1}{N}\left|\sum_{i=1}^{N} \sigma_i([[h(x_i) \neq y_i]])\right| > \frac{\epsilon}{4}]\right]$$

$$= E\left[E\left[[\sup_{h\in H}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i([[h(x_i)\neq y_i]])\right| > \frac{\epsilon}{4}]]\middle|D\right]\right]$$

$$= E\left[P\left(\sup_{h\in H}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i([[h(x_i)\neq y_i]])\right| > \frac{\epsilon}{4}\middle|D\right)\right]$$

$$\leq E\left[m_H(N)\sup_{h\in H}P\left(\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i([[h(x_i)\neq y_i]])\right| > \frac{\epsilon}{4}\middle|D\right)\right]$$

$$\leq m_H(N)E\left[\sup_{h\in H}P\left(\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i([[h(x_i)\neq y_i]])\right| > \frac{\epsilon}{4}\middle|D\right)\right]$$

$$\leq m_H(N)E\left[2e^{-\frac{n\epsilon^2}{32}}|D\right]$$

$$= 2m_H(N)e^{-\frac{n\epsilon^2}{32}}$$

Por último, esta última cota nos da el resultado deseado cocluyendo la prueba:

$$P\left(\sup_{h\in H}|E_{in}(h)-E_{out}(h)|>\epsilon\right)\leq 4P\left(\sup_{h\in H}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i([[h(x_i)\neq y_i]])\right| > \frac{\epsilon}{4}\right)\leq 8m_H(N)e^{-\frac{n\epsilon^2}{32}}$$

[Nowak, 2009] □

La cota de generalización VC es un resultado universal en el sentido de que se puede aplicar a cualquier conjunto de hipótesis, algoritmos de aprendizaje, espacios de entrada, distribuciones de probabilidad y funciones objetivo binarias. También se puede extender a otro tipo de funciones objetivo. [Abu-Mostafa et al., 2012]

3.3.3. Compromiso Sesgo-Varianza

El análisis anterior ha mostrado que la elección de H debe tener un balance entre aproximar f en los datos de entrenamiento y generalizar en nuevos datos: si H es muy simple, fallará a la hora de aproximar f bien y si H es demasiado compleja, la generalización será la que falle.
Para ello, descomponemos el error fuera de la muestra, E_{out}, donde g será la hipótesis final, E_x denota la esperanza respecto a x (basada en la distribución de probabilidad del espacio de entrada X) y hacemos explícita la dependencia de g con los datos D.

$$E_{out}(g^{(D)}) = E_x\left[(g^{(D)}(x)-f(x))^2\right]$$

Podemos quitar la dependencia en un conjunto de datos concreto tomando la esperanza respecto a todos los conjuntos de datos:

$$E_D[E_{out}(g^{(D)})] = E_D\left[E_x[(g^{(D)}(x)-f(x))^2]\right] = E_x\left[E_D[(g^{(D)}(x)-f(x))^2]\right]$$

$$= E_x\left[E_D[g^{(D)}(x)^2]-2E_D[g^{(D)}(x)]f(x)+f(x)^2\right]$$

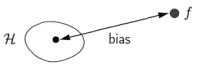

Figura 3.2: **Modelo demasiado pequeño**. Como solo hay una hipótesis, $\bar{g}$ y la hipótesis final g son las mismas y por tanto, la varianza o. El sesgo dependerá de la suerte que hayamos tenido aproximando la función objetivo f. [Abu-Mostafa et al., 2012]

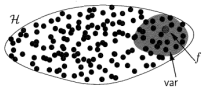

Figura 3.3: **Modelo demasiado grande**. La función objetivo está en H y muchos conjunto de datos llevarán a la hipótesis f consiguiendo sesgo=o. Sin embargo, la varianza será grande (región gris representada en la figura). [Abu-Mostafa et al., 2012]

El término $E_D[g^{(D)}(x)]$ da una 'función media' que denotamos por $\bar{g}(x)$. Podemos interpretar $\bar{g}(x)$ de la siguiente manera: generamos varios conjuntos de datos $D_1, ..., D_K$ y aplicamos el algoritmo a cada conjunto de datos para obtener $g_1, ..., g_K$. Podemos estimar la media para cualquier x como $\bar{g}(x) \approx \frac{1}{K}\sum_{k=1}^{K} g_k(x)$. Ahora reescribimos con ello la ecuación:

$$E_D[E_{out}(g^{(D)})] = E_x\left[E_D[g^{(D)}(x)^2] - 2\bar{g}(x)f(x) + f(x)^2\right]$$

$$E_x\left[E_D[g^{(D)}(x)^2] - \bar{g}(x)^2 + \bar{g}(x)^2 - 2\bar{g}(x)f(x) + f(x)^2\right]$$

sumando y restando $\bar{g}(x)^2$. De esta esperanza obtenemos los dos siguientes términos.
Se le llama **sesgo** (*bias* en inglés) a la agrupación del último cuadrado de la espezanda de la ecuación anterior:

$$sesgo(x) = (\bar{g}(x) - f(x))^2$$

y la **varianza** se obtiene de la primera resta, introduciendo $\bar{g}(x)^2$ en la esperanza:

$$var(x) = E_D[(g^{(D)}(x) - \bar{g}(x))^2]$$

que mide la variación que hay en la hipótesis final, dependiendo del conjunto de datos. Así la descomposición del error fuera de la muestra queda

$$E_D[E_{out}(g^{(D)})] = E_x[sesgo(x) + var(x)] = sesgo + var$$

donde $sesgo = E_x[sesgo(x)]$ y $var = E_x[var(x)]$.[Abu-Mostafa et al., 2012]
Tomemos dos ejemplos extremos como ilustración:

3.3.4. Sobreajuste y regularización

El **sobreajuste** (*overfitting* en inglés) ocurre cuando el modelo se ajusta muy bien a los datos de entrenamiento pero no a los de fuera de este conjunto.
El sobreajuste puede ocurrir incluso cuando el conjunto de hipótesis contiene sólo funciones que son 'mucho más simples' que la función objetivo. [Abu-Mostafa et al., 2012]
Para ilustrarlo, tomemos un problema sencillo de regresión unidimensional con cinco puntos

en el conjunto de datos. No sabemos la función objetivo así que, como 5 puntos pueden ajustarse por un polinomio de cuarto grado, seleccionamos los polinomios de cuarto grado como el conjunto de hipótesis. La función objetivo es un polinomio de segundo orden con ruido añadido en los puntos de los datos como muestra la siguiente figura:

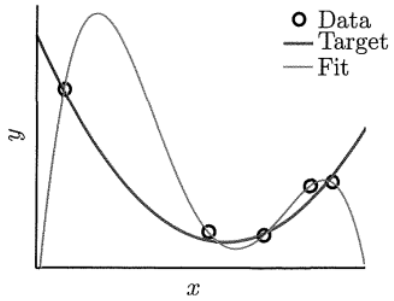

Figura 3.4: Ejemplo de *overfitting*. [Abu-Mostafa et al., 2012]

La linea más clara representa nuestro polinomio de cuarto grado y la oscura, el posinomio de segundo orden objetivo. Con este ajuste, hemos conseguido un error dentro de la muestra nulo pero un gran error fuera de la muestra aproximando la función objetivo.

La **regularización** (en inglés *regularization*) es una herramienta para combatir el sobreajuste. Limita el algoritmo de aprendizaje para mejorar el error fuera de la muestra.
El algoritmo de regularización utilizado en este trabajo es el conocido como *Lasso* (de sus siglas en inglés *Least Absolute Shrinkage and Selection Operator*) que controla los coeficientes y los limita haciendo que algunos puedan ir incluso a cero lo que da lugar a un procedimiento de estimación y selección de variables simultáneo. Es una forma de mínimos cuadrados penalizada que minimiza la suma de residuos mientras controla la norma L_1 del vector de coeficientes β:

$$argmin_{\beta}(y - X\beta)^T(y - X\beta) + \lambda \parallel \beta \parallel_1$$

donde $\lambda \geq 0$ indica el grado de limitación que tendrán los coeficientes, que estableciéndolo a o se consiguen los mínimos cuadrados ordinarios. [Hans, 2009]

3.4. Descripción de los modelos utilizados

El problema que se pretende resolver en este trabajo consiste en una predicción de etiquetas, es decir, al uso del aprendizaje supervisado para resolverlo. Dentro del aprendizaje supervisado encontramos varios modelos de gran potencia predictiva como los que describiremos en esta sección.

3.4.1. k-Nearest Neighbors

El modelo k-Nearest Neighbours (k-NN) es uno de los modelos más intuitivos y transparentes y se aleja un poco de la idea de aproximar una función ideal para tomar la decisión

de clasificación. La idea básica se muestra en la Figura 6.36. Se clasificará el vecino desconocido con la clase de los vecinos más cercanos a él, y si hay de varios tipos como en la figura, se puede resolver por simple mayoría o por votación ponderada por distancia, que veremos más adelante. Así que en k-NN se pueden diferenciar dos etapas: primero determinar los vecinos más cercanos y segundo, determinar la clase usando esos vecinos. [Cunningham and Delany, 2007]

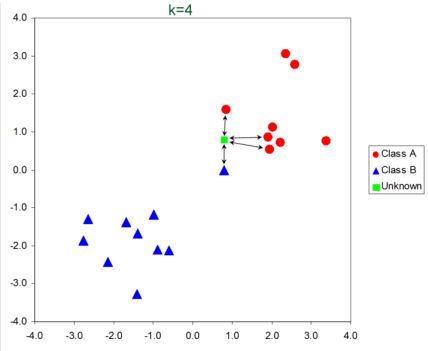

Figura 3.5: Ejemplo de 4-NN.

[Peterson, 2009]

3.4.1.1. Métrica

Sea $X=\{x_1,...,x_n\}$ donde $n =\mid X \mid$ un conjunto de entrenamiento, con x representando las instancias. Las instancias están descritas por un conjunto de variables F. Cada instancia de entrenamiento está etiquetada con una etiqueta de clase $y_i \in Y$, es decir, la función de etiquetado establece una aplicación biyectiva entre X e Y. El objetivo es clasificar un ejemplo desconocido q.[Cunningham and Delany, 2007]
Para cada $x_i \in X$ calculamos la distancia como la distancia de :

$$MD_p(\mathbf{q}, x_i) = (\sum_{f \in F} |q_f - x_{if}|^p)^{1/p} \quad \text{[Chomboon et al., 2015]}$$

Para p=1 la distancia de Minkowski es la llamada distancia Manhattan, para p=2, la Euclídea y para p=∞ la de Chebyshev. Sus expresiones son las que siguen [Chomboon et al., 2015]:

$$d^2(q, x_i) = (q - x_i)(q - x_i)' \quad Eucldea$$

$$d(q, x_i) = \sum_{f \in F} |q_f - x_{if}| \quad Manhattan$$

$$d(q, x_i) = max_{f \in F}\{|q_f - x_{if}|\} \quad Chebyshev$$

3.4.1.2. Votación

La votación consiste en establecer un criterio para asignar a q una de las etiquetas de Y. Esto puede hacerse asignando la etiqueta que tengan la mayoría de vecinos más cercanos a q o

también puede establecerse una función de votación. La más usual es establecer la votación en función de la inversa de la distancia entre q y sus vecinos.

$$Vote(y_i) = \sum_{c=1}^{k} \frac{1}{(d(q, x_c)^p} 1(y_j, y_c)$$

Donde $1(y_j, y_c)$ devuelve 1 si las etiquetas son iguales y 0 en caso contrario. En la función, p se suele establecer a 1 pero se puede aumentar o disminuir para cambiar la influencia que tiene la distancia en el voto. [Cunningham and Delany, 2007]

3.4.2. Regresión Logística

Para la regresión logística sí que se busca aproximar esa función que se hablaba al principio del capítulo. Para ello, se especifica el conjunto de hipótesis H mediante una forma funcional que todas las hipótesis $h \in H$ comparten. Esta fórmula funcional $h(x)$ se elige dando diferentes *pesos* a cada coordenada de x que reflejan la importancia que esa coordenada tiene en la decisión tomada. Este primer modelo de H se da en el *perceptron*, si llamamos w a los pesos, se diría que el individuo es afectado si

$$\sum_{i=1}^{d} w_i x_i > umbral$$

$$\text{y no afectado si } \sum_{i=1}^{d} w_i x_i < umbral$$

Pudiendo escribir la fórmula más compacta como:

$$h(x) = sign((\sum_{i=1}^{d} w_i x_i) + b)$$

donde $x_1, ..., x_d$ son los componentes del vector x. Si da 1 el individuo es afectado y si da -1 es sano. En el primer caso se le asigna la etiqueta 2 y en el segundo caso la 0, entrando así en el recorrido de nuestro problema. Los pesos y el umbral están determinados por el sesgo b de forma que el individuo está afectado si $\sum_{i=1}^{d} w_i x_i > -b$. Para simplificar la notación se va a escribir b como el peso $w_0 = b$ y juntarlo con los demás pesos en un solo vector columna $\mathbf{w}= [w_0, w_1, ..., w_d]^T$ y también modificamos el vector x como x=$[x_0, x_1, ..., x_n]$, donde se le añade la coordenada x_0 establecida a 1. El espacio de entrada, el dominio, será así

$$X = \{1\} \times \mathbb{R} = \{[x_0, ..., x_d]^T | x_0 = 1, x_i \in \mathbb{R}\ \forall i \in \{1, ..., d\}\}$$

y se puede escribir $w^T x$=$\sum_{i=1}^{d} w_i x_i$ y la ecuación de la hipótesis $h(x)$ quedaría:

$$h(x) = sign(w^T x)$$

[Abu-Mostafa et al., 2012]
La regresión logística se separa de la clasificación lineal en que la decisión que toma no es binaria sino que tiene como salida una **probabilidad**, un valor entre 0 y 1. La clasificación lineal usa un umbral basado en $w^T x$, $h(x) = sign(w^T x)$ mientras que la regresión lineal no

usa ningún umbral $h(x) = w^T x$. En la regresión logística se busca algo intermedio que se consigue con la función *logística* $\theta(s) = \frac{e^s}{1+e^s}$ cuyo valor está entre 0 y 1, quedando

$$h(x) = \theta(w^T x)$$

cuyo valor se interpreta como la probabilidad de que ocurra un suceso binario. Se está tratando de aprender la probabilidad de que el individuo esté afectado por el trastorno y eso depende de la entrada x. La función objetivo es por tanto $f(x) = \mathbb{P}[y = 1|x]$ pero los datos no dan el valor de f de forma explícita sino observaciones de ésta (individuos que ya se sabe que están afectados). Por tanto, los datos están generados por una distribución objetivo $P(y|x)$,

$$P(y|x) = \begin{cases} f(x) \text{ para } & y = 1 \\ 1 - f(x) \text{ para } & y = -1 \end{cases}$$

así que para aprenderlo se necesita una medida de **error** que indique cómo de parecida es la h a la función objetivo f en términos de las etiquetas de las observaciones.[Abu-Mostafa et al., 2012] Por tanto se tiene que el conjunto de muestras $X = \{x_1, ..., x_N\}$ es una muestra aleatoria simple pues los individuos son tomados de forma independiente y están idénticamente distribuidos.
El error en este modelo se basa en la idea de cómo es de "probable"que se consiga la salida y de la entrada x si la distribución objetivo $P(y|x)$ estuviera definida por nuestra hipótesis $h(x)$. Esto sería:

$$P(y|x) = \begin{cases} h(x) \text{ para } & y = 1 \\ 1 - h(x) \text{ para } & y = -1 \end{cases}$$

Sustituyendo $h(x)$ por su valor $\theta(yw^T x)$ y usando que $1 - \theta(s) = \theta(-s)$, probado a continuación, queda

$$P(y|x) = \theta(yw^T x)$$

Proposición 3.1. *La función logística* $\theta : \mathbb{R} \to [0, 1]$, $\theta(s) = \frac{e^s}{1+e^s}$ *tiene la propiedad simétrica:*

$$1 - \theta(s) = \theta(-s) \; \forall s \in \mathbb{R}$$

Demostración. Se toma $s \in \mathbb{R}$ y se tiene que $1 - \theta(s) = 1 - \frac{e^s}{1+e^s}$, multiplicando numerador y denominador por e^{-s} en la fracción queda $1 - \frac{1}{1+e^{-s}}$, realizando la resta se tiene $\frac{1+e^{-s}-1}{1+e^{-s}}$ y simplificando queda $\frac{e^{-s}}{1+e^{-s}} = \theta(-s)$, como se quería demostrar. □

Las observaciones $(x_1, y_1), ..., (x_N, y_N)$ son tomadas independientemente así que la probabilidad de obtener todas las salidas y_n en los datos desde la correspondiente muestra x_n sería el producto

$$\prod_{n=1}^{N} P(y_n|x_n)$$

El método de la máxima verosimilitud (Sección 1.3) selecciona la hipótesis h que maximiza esta probabilidad. Equivalentemente, se puede minimizar la siguiente expresión más conveniente para la medida del error:

$$\frac{-1}{N} ln \left(\prod_{n=1}^{N} P(y_n|x_n) \right) = \frac{1}{N} \sum_{n=1}^{N} ln \left(\frac{1}{P(y_n|x_n)} \right)$$

dado que la expresión '$\frac{-1}{N} ln(\cdot)$' es una función monótona decreciente y sustituyendo el valor de la probabilidad, estaríamos minimizando

$$\frac{1}{N} \sum_{n=1}^{N} ln \left(\frac{1}{\theta(y_n w^T x_n)} \right)$$

con respecto al vector de pesos w. Y como estamos minimizando, nos lleva al concepto de error. Sustituyendo la forma funcional de $\theta(y_n w^T x_n)$ produce el error dentro de la muestra para la regresión logística,

$$E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} ln \left(1 + e^{-y_n w^T x_n} \right).$$

Para minimizarlo, usamos el **gradiente descendente** en su versión **estocástica**. Repasaremos antes la versión no estocástica.
El gradiente descendiente es una técnica para minimizar funciones de clase C^2 como $E_{in}(w)$ en regresión logística. Como E_{in} es una función convexa, esta técnica siempre alcanzará el mínimo absoluto empiece donde empiece. Supongamos que empezamos en $w(0)$ y tomamos un paso hacia 'abajo' de tamaño η con la dirección de un vector $\hat{v}$. Como η es pequeña, usamos la serie de Taylor de primer orden, quedando el diferencial de E_{in} como

$$\Delta E_{in} = E_{in}(w(0) + \eta\hat{v}) - E_{in}(w(0)) = \eta \nabla E_{in}(w(0))^T \hat{v} + O(n^2) \geq -\eta \parallel \nabla E_{in}(w(0)) \parallel$$

donde hemos ignorado un pequeño término $O(\eta^2)$. La igualdad se cumple si, y solo si,

$$\hat{v} = -\frac{\nabla E_{in}(w(0))}{\parallel \nabla E_{in}(w(0)) \parallel},$$

que indica la dirección al mayor decrecimiento de E_{in} para un tamaño ∇ dado. [Abu-Mostafa et al., 2012]

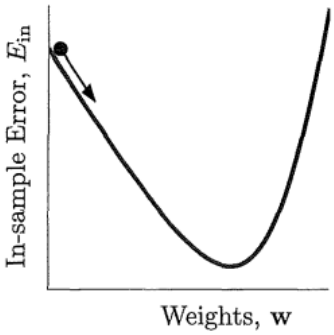

Figura 3.6: Un paso hacia abajo de tamaño η con dirección $\hat{v}$. [Abu-Mostafa et al., 2012]

La longitud de ese paso que tomamos es importante en esta técnica pues define el comportamiento que tiene como puede verse en los siguientes ejemplos visuales:

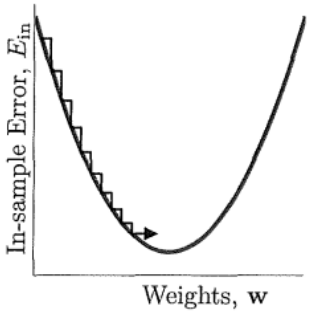

Figura 3.7: η muy pequeña.

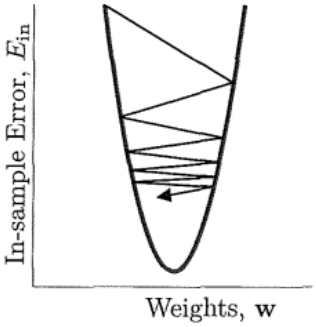

Figura 3.8: η muy grande.

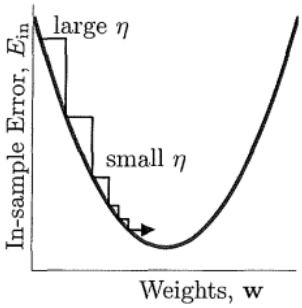

Figura 3.9: η correcta.

Cuando η es muy pequeño, se hace ineficiente llegar al mínimo local y cuando es grande es posible que incluso aumente E_{in}.
Con esto, podemos definir el algoritmo de regresión logística usando gradiente descendiente: Pasamos ahora sí al **gradiente descendiente estocástico**.

Algorithm 1 Regresión logística con gradiente descendiente

1: Inicializar los pesos en el paso t=0 a w(0)
2: **for** t=0,1,2... **do**
3: Computar el gradiente $g_t = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n x_n}{1+e^{y_n w^T(t) x_n}}$
4: Establecer la direccion del movimiento, $v_t = -g_t$
5: Actualizar los pesos $w(t+1) = w(t) + \eta V_t$
6: Iterar al siguiente paso hasta alcanzar la parada
7: Devolver los pesos finales w

Esta vez, en vez de considerar el gradiente completo en todos los N puntos del conjunto de entrenamiento, consideramos una versión estocástica del gradiente. Primero, tomamos un punto del entrenamiento aleatorio de forma uniforme (x_n, y_n) y consideramos solo el error en ese punto, en caso de la regresión logística

$$e_n(w) = ln(1 + e^{-y_n w^T x_n}).$$

Y usamos el gradiente en este error para actualizar los pesos:

$$\nabla e_n(w) = -\frac{y_n x_n}{1 + e^{y_n w^T x_n}}$$

actualizamos con ello los pesos $w \leftarrow w - \eta \nabla e_n(w)$. Así, obtenemos que la esperanza del cambio de los pesos, como n la tomamos de forma aleatoria de $1, ..., N$, es la misma que para el gradiente descendiente:

$$-\eta \frac{1}{N} \sum_{n=1}^{N} \nabla e_n(w).$$

La diferencia radica en una componente aleatoria que tiene en cada actualización de pesos, que a la larga se cancela y conseguimos un coste computacional más barato en un factor de N. [Abu-Mostafa et al., 2012]

3.4.3. Árboles de decisión

Son un modelo de aprendizaje no paramétrico usado para clasificación y regresión (para clasificación en este trabajo). Es un modelo fácil de entender y eficiente (eficiencia logarítmica a la hora de predecir) pero tiende al sobreajuste. [Scikit-Learn, 2021a]
Hay numerosos algoritmos para este modelo y el que se usa en la implementación de *scikit-learn* es una versión optimizada del llamado **CART** (por sus siglas en inglés *Classification and Regression Trees*). Su metodología consiste en tres partes:

- 1. Construcción del árbol *máximo*.
- 2. Elección del tamaño del árbol. Que se puede hacer con validación cruzada. [Timofeev, 2004]
- 3. Clasificación de nuevos datos usando el árbol construido.

[Timofeev, 2004]
Construir el árbol máximo es la parte que lleva más tiempo. Sea t_p un nodo padre y t_l, t_r sus nodos hijos de la izquierda y la derecha respectivamente. Consideremos la muestra de entrenamieno como una matriz variable X con M números de variables x_j y N observaciones. Sea el vector de etiquetas Y compuesto de N observaciones con K clases.
El árbol de clasificación se construye de acuerdo a la *regla de separación* donde los cada vez los datos se separan en dos partes con la máxima homogeneidad. Si llamamos x_j^R el mejor valor de separación de la variable x_j nos queda como regla de separación:

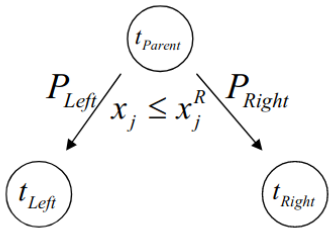

Figura 3.10: Algoritmo de separación de CART. [Timofeev, 2004]

La homogeneidad máxima de los nodos hijos se define por la llamada *función de impureza i(t)* que se definirá un poco más adelante. Como la impureza del nodo padre es constante para cada posible separación, la máxima homogeneidad de los hijos izquierdo y derecho será equivalente a la maximización de la diferencia en la función de impureza $\Delta i(t)$:

$$\Delta i(t) = i(t_p) - E[i(t_c)]$$

donde t_c indica ambos nodos hijos. Si P_l, P_r son las probabilidades de los nodos izquierdo y derecho, obtenemos:

$$\Delta i(t) = i(t_p) - P_l i(t_l) - P_r i(t_r)$$

Por tanto, en cada nodo CART soluciona el siguiente problema de optimización:

$$\underset{x_j \leq x_j^R, j=1,\ldots,M}{\arg\max} \quad [i(t_p) - P_l i(t_l) - P_r i(t_r)]$$

[Timofeev, 2004]
Ahora sí, definimos la función de impureza. La regla más usada es la de **Gini** que usa la siguiente función de impureza:

$$i(t) = \sum_{k \neq l} p(k|t)p(l|t)$$

donde $k, l \in 1, ..., K$ son índices de las clases y $p(k|t)$ la probabilidad condicional de la clase k condicionada a que estamos en el nodo t. Así, la diferencia $\Delta i(t)$ queda:

$$\Delta i(t) = -\sum_{k=1}^{K} p^2(k|t_p) + P_l \sum_{k=1}^{K} p^2(k|t_l) + P_r \sum_{k=1}^{K} p^2(k|t_r)$$

Por lo tano, Gini soluciona el siguiente problema de optimización:

$$\underset{x_j \leq x_j^R, j=1,...,M}{\arg\max} \left[-\sum_{k=1}^{K} p^2(k|t_p) + P_l \sum_{k=1}^{K} p^2(k|t_l) + P_r \sum_{k=1}^{K} p^2(k|t_r) \right]$$

buscará en la muestra de entrenamiento la mayor clase y la aislará del resto. Funciona bien para datos con ruido.
Para la parte de clasificación de nuevos datos, se usa el árbol ya construido y a un nuevo dato se le asigna la *clase dominante* del nodo terminal (el último nodo del árbol donde queda el nuevo dato) que es la clase con el mayor número de observaciones.[Timofeev, 2004]

3.4.4. Bosques aleatorios

Los bosques aleatorios, más conocidos en inglés como *Random Forests*, son un modelo predictor propuesto por Leo Breiman que se basa en un conjunto de árboles de decisión que crecen en subespacios de los datos seleccionados aleatoriamente. [Biau, 2012]
Llamaremos a los datos de entrada como siempre $D = \{(X_1, Y_1), ..., (X_n, Y_n)\}$ y con ellos, el modelo consiste en una colección de árboles de regresión de base aleatoria $\{r_n(x, \Theta, D), m \geq 1\}$, donde $\Theta_1, \Theta_2, ...$ son salidas de una variable aleatoria Θ. Se combinan estos árboles aleatorios y se crea una estimación de regresión agregada, esto es,

$$\bar{r}_n(X, D) = E_\Theta[r_n(X, \Theta, D)],$$

donde la variable aleatoria se usa para determinar cómo se hacen las sucesivas podas cuando se crean los árboles individuales.
Asumimos que los árboles individuales se crean de la siguiente forma: los nodos de las hojas del árbol forman una partición de $[0, 1]^d$ y la raíz es $[0, 1]^d$. Se sigue el siguiente esquema $\lceil log_2 k_n \rceil$ veces donde $\lceil \bullet \rceil$ es la función techo de parte entera:

1. En cada nodo, una coordenada de $X = (X^{(1)}, ..., X^{(d)})$ es seleccionada con una probabilidad $p_{nj} \in (0, 1)$.

2. Cuando se selecciona, en cada nodo se divide en el punto medio del lado elegido.

Con este tipo de árboles se crea $\bar{r}_n(X, D)$, tomando una media o esperanza de las decisiones de estos árboles para reducir los errores. [Biau, 2012]
Los bosques aleatorios consiguen una varianza reducida por la combinación de árboles, a veces al costo de aumentar el sesgo. [Scikit-Learn, 2021f]

3.4.5. Potenciación del gradiente

Más conocido como *Gradient boosting* en inglés es una forma de optimizar lo que se llaman 'aprendices débiles' (*weak learners*) como por ejemplo los árboles de regresión vistos en el anterior modelo [Scikit-Learn, 2021b]. Vamos a explicar el modelo de regresión y el clasificador será muy parecido.
De nuevo, llamando a los datos de entrada x_i con sus respectivas etiquetas y_i, el *Gradient boosting* es un modelo aditivo para aquellos modelos que predicen de la forma:

$$\hat{y}_i = F_M(x_i) = \sum_{m=1}^{M} hM(x_i)$$

donde h_m son los mencionados aprendices débiles y usa árboles de decisión para ello. Está construido de una forma parecida a los algoritmos voraces (algoritmos que siempre buscan el mejor valor sin volver atrás):

$$F_m(x) = F_{m-1}(x) + h_m(x),$$

donde el nuevo árbol añadido se entrena de forma que minimice la suma de pérdidas L_m, dado el anterior F_{m-1}:

$$h_m = \arg\min_h L_m = \arg\min_h \sum_{i=1}^{n} l(y_i, F_{m-1}(x_i) + h(x_i)),$$

donde l es una función de pérdida como los mínimos cuadrados, por ejemplo.
Usando una aproximación de Taylor de primer orden, l puede aproximarse:

$$l(y_i, F_{m-1}(x_i) + h(x_i)) \approx l(y_i, F_{m-1}(x_i)) + h_m(x_i) \left[\frac{\partial y_i, F(x_i)}{\partial F(x_i)} \right]_{F=F_{m-1}}$$

Denotamos al diferencial g_i y obtenemos tras quitar los términos constantes:

$$h_m \approx \arg\min_h \sum_{i=1} h(x_i) g_i$$

Esto se minimiza si $h(x_i)$ es ajustado para predecir el valor proporcional al gradiente negativo $-g_i$. Por tanto, el estimador h_m en cada iteración se ajusta para predecir los gradientes negativos de las muestras, que se actualizan en cada iteración.
Ahora, para la clasificación, la diferencia es que la suma de los árboles $F_M(x_i) = \sum_m h_m(x_i)$ es no homogénea a una predicción: no puede ser una clase ya que los árboles predicen valores continuos.
La aplicación que lleva $F_M(x_i)$ a una clase o probabilidad depende de la función de pérdida. [Scikit-Learn, 2021b]

3.4.6. Máquinas de soporte vectorial

Las máquinas de soporte vectorial (**SVM** por sus siglas en inglés) son un algoritmo de aprendizaje automático supervisado que es efectivo para conjuntos de datos de alta dimensionalidad y suele ser usado por ello para problemas de clasificación biológica [Cogill and Wang, 2016] [Cortes and Vapnik, 1995].

Hace más de 80 años, R.A. Fisher sugirió el primer algoritmo para reconocimiento de patrones. Consideró un modelo de dos poblaciones normales, $N(m_1, \Sigma_1)$ y $N(m_2, \Sigma_2)$ de vectores n-dimensionales x con vectores de medias m_1 y m_2 y matrices de covarianza Σ_1 y Σ_2, y demostró que la solución óptima (Bayesiana) es una función de decisión cuadrática, esto es, una función que divide el espacio vectorial en dos conjuntos, uno para cada etiqueta

$$F_{sq} = sign\left[\frac{1}{2}(x-m_1)^T\Sigma_1^{-1}(x-m_1) - \frac{1}{2}(x-m_2)^T\Sigma_2^{-1}(x-m_2) + ln\frac{|\Sigma_2|}{|\Sigma_1|}\right]$$

En el caso de que $\Sigma_1 = \Sigma_2 = \Sigma$ la función cuadrática de decisión degenera a una función lineal:

$$F_{lin} = sign\left[(m_1-m_2)^T\Sigma^{-1}x - \frac{1}{2}(m_1^T\Sigma^{-1}m_1 - m_2^T\Sigma^{-1}m_2)\right].$$

Para estimar la función de decisión cuadrática se necesita determinar $\frac{n(n+3)}{2}$ parámetros. En cambio, para estimar la función lineal, sólo n. Fisher por tanto, recomendó incluso en el caso $\Sigma_1 \neq \Sigma_2$ usar la función discriminante lineal con Σ de la forma:

$$\Sigma = \tau\Sigma_1 + (1-\tau)\Sigma_2$$

donde τ es una constante. También recomendó usar la lineal cuando las dos distribuciones fueran no normales. [Cortes and Vapnik, 1995]
Tras él, en 1962 Rosenblatt exploró el perceptrón al que más tarde se el añadiría el algoritmo de *back-propagation* (propagación hacia atrás, del inglés) y finalmente se construirían las máquinas de soporte vectorial siguiendo la idea de aplicar los vectores de entrada en algún espacio de alta dimensionalidad Z a través de alguna aplicación no lineal elegida a priori. En este espacio se construye una superficio de decisión lineal con propiedades especiales que asegure una alta generalización de la decisión. [Cortes and Vapnik, 1995]
Primero, vamos a hacer una revisión del método del ***hiperplano óptimo*** (Vapnik, 1982 [Cortes and Vapnik, 1995]) para separar datos de entrenamiento sin errores. Después se introducirá el concepto de ***margen blando*** (del inglés, *soft margin*) que permitirá un tratamiento analítico para aprender con errores en el conjunto de entrenamiento.
El conjunto de entrenamiento (etiquetado) $(y_1, x_1), ..., (y_l, x_l)$ con $y_i \in \{-1, 1\}$ se dice que es linealmente separable si existe un vector w y un escalar b tales que las desigualdades

$$wx_i + b \geq 1 \text{ si } y_i = 1,$$

$$wx_i + b \leq -1 \text{ si } y_i = -1,$$

son válidas para todos los elementos del conjunto, es decir, existe un hiperplano que separa los datos de etiqueta 1 y -1 por completo. Escribimos las desigualdades de la forma:

$$y_i(wx_i + b) \geq 1, \text{ con } i = 1, ..., l.$$

El **hiperplano óptimo**

$$w_0x + b_0 = 0$$

es el único que separa los datos de entrenamiento con un margen máximo: determina la dirección $\frac{w}{|w|}$ donde la distancia entre las proyecciones de los vectores de dos diferentes clases es máximo. Por ejemplo, en la Figura 6.44 siguiente los vectores de soporte, marcados con cajas grises, definen el mayor margen de separación entre las dos clases.

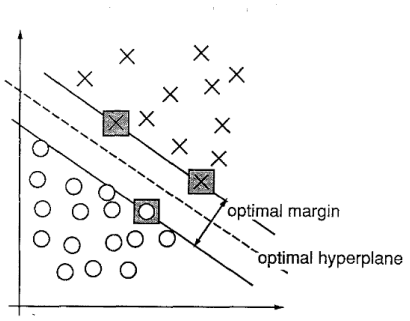

Figura 3.11: Problema separable en 2 dimensiones.[Cortes and Vapnik, 1995]

Esta distancia está dada por $\rho(w,b) = \min_{\{x;y=1\}} \frac{xw}{|w|} - \max_{\{x;y=-1\}} \frac{xw}{|w|}$.
El hiperplano óptimo (w_0, b_0) es el argumento que maximiza la distancia ρ:

$$\rho(w_0, b_0) = \frac{2}{|w_0|} = \frac{2}{\sqrt{w_0 w_0}}.$$

Esto significa que el hiperplano óptimo es aquel que minimiza ww bajo la condición $y_i(wx_i + b) \geq 1$, con $i = 1, ..., l$. Construir un hiperplano óptimo es por tanto un problema de programación cuadrático, esto es, un problema de optimización de una función cuadrática.
Los vectores x_i para los que $y_i(wx_i + b) = 1$ se llaman *vectores de soporte*. Vamos a demostrar ahora que el vector w_0 que dteermina el hiperplano óptimo puede ser escrito como una combinación lineal de vectores de entrenamiento:

$$w_0 = \sum_{i=1}^{l} y_i \alpha_i^0 x_i$$

donde $\alpha_i^0 \geq 0$.

Demostración. Llamemos $\Phi = ww$ al funcional que debemos minimizar. Para hacerlo, contruimos un Lagrangiano

$$L(w, b, \Lambda) = \frac{1}{2} ww - \sum_{i=1}^{l} \alpha_i [y_i(x_i w + b) - 1],$$

donde $\Lambda = (\alpha_1, ..., \alpha_l)$ es lo que se llama un vector de multiplicadores de Lagrange no negativos correspondientes a la condición de que $y_i(wx_i + b) \geq 1$, con $i = 1, ..., l$.
Se sabe que la solución al problema de optimización está determinado por el punto de silla (punto sobre una superficie en el que la pendiente es cero pero no es un extremo local. Es el punto donde la elevación es máxima en una dirección y mínima en la dirección perpendicular) de este Lagrangiano en el espacio $2l + 1$ dimensional de w, Λ y b, donde el mínimo se toma respecto a los parámetros w y b y el máximo respecto a los multiplicadores de Lagrange Λ.
En el punto del mínimo con respecto a w y b se obtiene:

$$\frac{\partial L(w, b, \Lambda)}{\partial w}\Big|_{w=w_0} = \left(w_0 - \sum_{i=1}^{l} \alpha_i y_i x_i \right) = 0,$$

$$\frac{\partial L(w,b,\Lambda)}{\partial b}\Big|_{b=b_0} = \sum_{\alpha_i} y_i \alpha_i = 0.$$

Y de la igualdad con respecto a w obtenemos

$$w_0 = \sum_{i=1}^{l} y_i \alpha_i^0 x_i$$

como queríamos. Notar que sólo los vectores x_i con $\alpha_i > 0$ aportan a la suma.

□

Vamos a seguir con la optimización, pues servirá para seguir explicando las SVM. Sustituyendo en el Lagrangiano las dos anteriores expresiones, obtenemos:

$$W(\Lambda) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} w_0 w_0 = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j.$$

En notación vectorial, puede ser reescrito como sigue:

$$W(\Lambda) = \Lambda^T \mathbf{1} - \frac{1}{2} \Lambda^T D \Lambda$$

dondo $\mathbf{1}$ es el vector unidad l dimensional, y D es una matriz simétrica con elementos $D_{ij} = y_i y_j x_i x_j$.
Para encontrar el punto de silla deseado, queda por localizar el máximo de $W(\Lambda)$ bajo la condición $\Lambda^T Y = 0$ donde $Y^T = (y_1, ..., y_l)$ y $\Lambda \geq 0$.
El teorema de Kuhn-Tucker juega un papel impotante en la teoría de optimización. De acuerdo cn este teorema, en nuestro punto de silla en w_0, b_0, Λ_0, cualquier multiplicador de Lagrange y su correspondiente condición se conectan por una igualdad

$$\alpha_i [y_i (x_i w_0 + b_0) - 1] = 0, \, i = 1, ..., l.$$

De esta igualdad viene que lso valores distintos de cero α_i son solo alcanzados en los casos en los que

$$y_i (x_i w_0 + b_0) - 1 = 0.$$

Los vectores x_i para los que $y_i (x_i w_0 + b_0) = 1$, son los vectores de soporte. Notar que el hiperplano óptimo w_0 se puede expandir sobre los vectores de soporte.
Juntando Kuhn-Tucker, la derivada parcial con respecto a b y la expresión del hiperplano óptimo como combinación lineal de los vectores de entrenamiento, obtenemos la relación entre el valor máximo de $W(\Lambda_0)$ y la distancia de separación ρ_0:

$$w_0 w_0 = \sum_{i=1}^{l} \alpha_i^0 y_i x_i w_0 = \sum_{i=1}^{l} \alpha_i^0 (1 - y_i b_0) = \sum_{i=0}^{l} \alpha_i^0.$$

Sustituyendo esta igualdad en la expresión de $W(\Lambda_0)$, obtenemos

$$W(\Lambda_0) = \sum_{i=1}^{l} \alpha_i^0 - \frac{1}{2} w_0 w_0 = \frac{w_0 w_0}{2}.$$

Teniendo en cuenta la expresión de la distancia, obtenemos

$$W(\Lambda_0) = \frac{2}{\rho_o^2},$$

donde ρ_0 es el margen para el hiperplano óptimo. [Cortes and Vapnik, 1995]
Como $\alpha > 0$, solo para los vectores de soporte la expresión $w_0 = \sum_{i=1}^{l} y_i \alpha_i^0 x_i$ representa una forma compacta de escribir w_0.
La desigualdad $\Lambda \geq 0$ describe el cuadrante no negativo y además tenemos que maximizar la forma cuadrática $W(\Lambda) = \Lambda^T \mathbf{1} - \frac{1}{2}\Lambda^T D \Lambda$ en este cuadrante, sujeto a las condiciones $\Lambda^T Y = 0$, donde $Y^T = (y_1, ..., y_l)$ es el vector l-dimensional de etiquetas.
Cuando los datos de entrenamiento pueden ser separados sin errores tenemos una relación entre el máximo del funcional $W(\Lambda)$, el par (Λ_0, b_0), y el margen máximo ρ_0: $W(\Lambda_0) = \frac{2}{\rho_0^2}$.
Si para algún Λ_* y una constante grande W_0, la desigualdad $W(\Lambda_*) > W_0$ es válida, se puede asumir que todos los hiperplanos que separan los datos de entrenamiento tienen un margen $\rho < \sqrt{\frac{2}{W_0}}$.[Cortes and Vapnik, 1995]
Si el conjuno de entrenamiento no puede ser separado por un hiperplano, los márgenes entre patrones de distintas clases se vuelven arbitrariamente pequeños y con ello, el valor resultante del funcional $W(\Lambda)$, arbitrariamente grande. Maximizar el funcional bajo sus ocndiciones puede llevar a encontrar el máximo buscado o puede que el máximo exceda un valor grande dado W_0 constante (en cuyo caso, una separación de los datos de entrenamiento con un margen mayor que $\sqrt{2/W_0}$ es imposible).
El problema de maximizar el funcional bajo sus condiciones se puede hacer eficientemente dividiendo el conjunto en un numero de partes con un número razonablemente pequeño de vectores de entrenamiento en cada parte. Se empieza resolviendo el problema de programación cuadrático determinado por esta primera porción de los datos. Se puede llegar a que se puede separar por un hiperplano o que es imposible hacerlo (con lo que sería imposible para todo el conjunto de datos, por tanto).
Sea el vector que maximiza el funcional en la primera porción Λ_1. Algunas coordenadas de Λ_1 son cero. Corresponden a los vectores de no-soporte de esta porción. Se hace un nuevo conjunto de entrenamiento lo vectores de soporte de la primera porción y los vectores de la segunda porción que no satisfacen que $y_i(wx_i + b) \geq 1$, donde Λ_1 determina w. Se construye un nuevo funcional $W_2(\Lambda)$ y se maximiza en Λ_2. Esto se continúa hasta obtener el vector solución $\Lambda_* = \Lambda_0$ al acabar con todas las porciones de datos. Durante el proceso, $W(\Lambda)$ crece de forma monótona pues al añadir vectores de entrenamiento, hay menor separación entre ellos.[Cortes and Vapnik, 1995]

Consideramos ahora el caso en el que los datos de entrenamiento no pueden ser separados sin error. En este caso, se quiere separar el conjunto con el mínimo número de errores. Para expresarlo formalmente, introducimos algunas variables no negativas $\xi \geq 0, i = 1, ..., l$. Y con ellas, podemos minimizar ahora el funcional

$$\Phi(\xi) = \sum_{i=1}^{l} \xi_i^{\sigma}$$

para un $\sigma > 0$ pequeño

$$y_i(wx_i + b) \geq 1 - \xi_i, \; i = 1, ..., l,$$
$$\xi_i \geq 0, \; i = 1, ..., l.$$

Para σ suficientemente pequeños, el funcional describe el número de errores de entrenamiento. Con errores nos referimos a un patrón donde la primera condición se mantiene con $\xi > 0$.
Hay cierto subconjunto de datos de entrenamientos $(y_{i_1}, x_{i_1}), ..., (y_{i_k}, x_{i_k})$, para el que si lo

quitáramos del conjunto completo, éste se podría separar sin errores.
La idea se puede expresar como sigue: minimizar el funcional

$$\frac{1}{2}w^2 + CF\left(\sum_{i=1}^{l}\xi_i^\sigma\right)$$

sujeto a las condiciones antes mencionadas, donde $F(u)$ es una función monótona convexa y C es una constante. Para C suficientement grande y σ suficientemente pequeña, el vector w_0 y la constante b_0, que minimizan el funcional bajo las condiciones, determinan el hiperplano que minimiza el número de errores en el conjunto de entrenamiento y separa el resto de elementos con un margen máximo.
Sin embargo, este problema es NP-completo (esto es, altamente costoso computacionalmente). Para evitarlo, vamos a considerar el caso de $\sigma = 1$ (el valor más pequeño para el que el problema de optimizar $W(\Lambda)$ tiene una solución única). Para un C suficientemente grande en est caso, el problema de de construir un hiperplano que minimice la *suma de desviaciones*, ξ, de los errores de entrenamiento y que maximice el margen para clasificar los vectores correctamente.
Para el caso $\sigma = 1$ existe un método eficiente llamado *el hiperplano de margen blando*.
[Cortes and Vapnik, 1995]
Se va a describir el algoritmo del hiperplano de margen blando empezando para el caso en el que $F(u) = u^k$ y generalizando luego a cualquier función monótona y convexa $F(u)$.
Para ello, se maximiza el funcional

$$\Phi = \frac{1}{2}ww + C\left(\sum_{i=1}^{l}\xi_i\right)^k, k > 1,$$

bajo las mismas condiciones anteriormente mencionadas:

$$y_i(wx_i + b) \geq 1 - \xi_i, i = 1, ..., l,$$

$$\xi_i \geq 0, i = 1, ..., l.$$

La funcion Lagragiana en este caso es

$$L(w, \xi, b, \Lambda, R) = \frac{1}{2}ww + C\left(\sum_{i=1}^{l}\xi_i\right)^k - \sum_{i=1}^{l}\alpha_i[y_i(x_iw + b) - 1 + \xi_i] - \sum_{i=1}^{l}r_i\xi_i,$$

donde los multiplicadores $\Lambda^T = (\alpha_1, \alpha_2, ..., \alpha_l)$ surgen de la primera restricción y los multiplicadores $R^T = (r_1, r_2, ..., r_l)$ hacen cumplir la segunda restricción.
Debemos encontrar el mínimo con respecto a w_i, b y ξ_i y el máximo respecto a las variables α_i y r_i, es decir, el punto de silla.
Para ello, vamos a calcular los extremos:

$$\frac{\partial L}{\partial w}\Big|_{w=w_0} = w_0 - \sum_{i=1}^{l}\alpha_1 y_i x_i = 0,$$

$$\frac{\partial L}{\partial b}\Big|_{b=b_0} = \sum_{i=1}^{l}\alpha_i y_i = 0,$$

$$\frac{\partial L}{\partial \xi_i}\Big|_{\xi_i=\xi_i^0} = kC\left(\sum_{i=1}^{l}\xi_i^0\right)^{k-1} - \alpha_i - r_i.$$

Y denotando

$$\sum_{i=1}^{l}\xi_i^0 = \left(\frac{\delta}{Ck}\right)^{\frac{1}{k-1}},$$

podemos reescribir la última derivada parcial como $\delta - \alpha_i - r_i = 0$. De la primera derivada parcial y con la última notación empleada, obtenemos que $w_0 = \sum_{i=1}^{l}\alpha_i y_i x_i$, $\sum_{i=1}^{l}\alpha_i y_i = 0$ y que $\delta = \alpha_i + r_i$.
Sustituyendo estas últimas expresiones en el funcional de Lagrange obtenemos la siguiente expresión que tendremos que maximizar bajo las condiciones ya descritas.

$$W(\Lambda,\delta) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j x_i x_j - \frac{\delta^{k/k-1}}{(kC)^{1/k-1}}\left(1-\frac{1}{k}\right).$$

Y en notación vectorial, esto queda

$$W(\Lambda,\delta) = \Lambda^T\mathbf{1} - \left[\frac{1}{2}\Lambda^T D\Lambda + \frac{\delta^{k/k-1}}{(kC)^{1/k-1}}\left(1-\frac{1}{k}\right)\right]$$

y las condiciones quedarían como: $\Lambda^T Y = 0$, $\Lambda + R = \delta\mathbf{1}$, $\Lambda \geq 0$ y $R \geq 0$. Además de las segunda y cuarta condiciones se obtiene que el vector Λ debe satisfacer que $0 \leq \Lambda \leq \delta\mathbf{1}$ y que para maximizar $W(\Lambda)$, $\delta = \alpha_{max} = \max(\alpha_1,...,\alpha_l)$. Sustituyendo este valor de δ se obtiene

$$W(\Lambda) = \Lambda^T\mathbf{1} - \left[\frac{1}{2}\Lambda^T D\Lambda + \frac{\alpha_{max}^{k/k-1}}{(kC)^{1/k-1}}\left(1-\frac{1}{k}\right)\right].$$

Vamos a usar $k = 2$ y resolver el problema de programación cuadrático $L(w,\xi,b,\Lambda,R)$. Empezamos como ya adelantamos antes, con el caso $F(u) = u$ donde la misma técnica nos lleva a minimizar esta vez el funcional $W(\Lambda) = \Lambda^T\mathbf{1} - \frac{1}{2}\Lambda^T D\Lambda$, bajo las condiciones $\Lambda^T Y = 0$ y $0 \leq \Lambda \leq C\mathbf{1}$.
La solución para el caso general de una función monótona convexa $F(u)$ también se puede obtener de esta técnica donde el hiperplano de margen blando tiene la forma $w = \sum_{i=1}^{l}\alpha_i y_i x_i$ y en el que $\Lambda_0^T = (\alpha_1^0,...,\alpha_l^0)$ maximiza el siguiente funcional, solucionando así el problema dual convexo de programacion:

$$W(\Lambda) = \Lambda^T\mathbf{1} - \left[\frac{1}{2}\Lambda^t D\Lambda + \left(\alpha_{max} f^{-1}\left(\frac{\alpha_{max}}{C}\right)\right) - CF\left(f^{-1}\left(\frac{\alpha_{max}}{C}\right)\right)\right]$$

bajo las condiciones $\Lambda^T Y = 0$ y $\Lambda \geq 0$ y donde denotamos $f(u)$ a la derivada de $F(u)$. [Cortes and Vapnik, 1995]

Para terminar, notar que los algoritmos descritos hasta ahora construyen hiperplanos en el espacio de entrada y para hacerlo en el espacio de etiquetas, hace falta que llevemos el espacio de entrada n-dimensional al espacio de etiquetas N-dimensional mediante una función

$$\phi : \mathbb{R}^n \to \mathbb{R}^N$$

Definiendo

$$K(u,v) = \sum_{i=1}^{\infty} \lambda_i \phi_i(u)\phi_i(v)$$

Podemos encontrar los vectores x_i y los pesos para los vectores de soporte en el espacio de etiqeutas de la función objetivo

$$f(x) = \sum_{i=1}^{l} y_i \alpha_i K(x, x_i)$$

siguiendo el mismo esquema de hiperplano óptimo y margen blando con la única diferencia de la matriz D:

$$D_{ij} = y_i y_j K(x_i, x_j),\ i,j = 1,...,l.$$

[Cortes and Vapnik, 1995]

3.5. Métricas

En este apartado vamos a hablar de las métricas utilizadas en el trabajo. Las métricas son funciones que sirven para medir la bondad del ajuste de los modelos entrenados en diversos problemas. No se debe confundir con la función de error que algunos modelos optimizan, pues la métrica se mide sobre los conjuntos de test y entrenamiento y se usa cuando el modelo ya se ha entrenado y la función de error se optimiza para el de entrenamiento solamente y se usa dentro del modelo.
Las métricas empleadas han sido las siguientes.

3.5.1. Accuracy

Exactitud en español, se define como el porcentaje de acierto del modelo, es decir, el porcnetaje de datos que ha etiquetado bien.

$$Accuracy = \frac{N\acute{u}mero\ de\ predicciones\ Correctas}{N\acute{u}mero\ total\ de\ predicciones}.$$

Podemos ver que está calculada con el número de predicciones y por tanto, sólo funcionará en datos balanceados, esto es, datos que tienen el mismo número de individuos en ambas clases.
El problema surge cuando es más importante acertar una etiqueta que otra. Tomemos de ejemplo el diagnóstico de un cáncer: en este caso, el coste que conlleva decir a un paciente que no tiene cáncer cuando sí lo tiene es mucho mayor que el caso contrario y por lo tanto, es más importante predecir bien la clase "tiene la enfermedad", cosa que esta métrica no tiene en cuenta.

3.5.2. Matriz de confusión

Es una matriz 2×2 (en caso de clasificación binaria como el de este trabajo) en la que las columnas indican las etiquetas predichas por el modelo y las filas, las reales, esto es, las etiquetadas en el conjunto D.

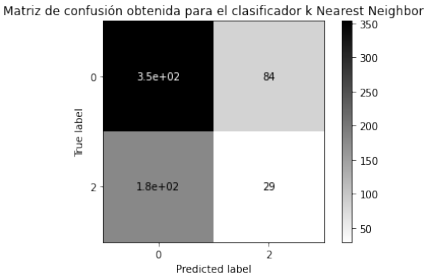

Figura 3.12: Ejemplo de matriz de confusión

La matriz de ejmplo anterior, generada para este trabajo, etiqueta al individuo a 0 cuando no tiene ASD y 2 cuando tiene, lo que se podría traducir como negativo y postivo respectivamente.
De esta forma, se generan cuatro términos importantes:

- Verdaderos positivos: Son los casos en los que se predice la etiqueta 'positivo' y el dato real está etiquetado de la misma forma.
- Falsos positivos: Los casos en los que se predice 'positivo' pero el dato real es 'negativo'.
- Verdaderos negativos: Casos en los que se predice 'negativo' y efectivamente, el dato real es 'negativo'.
- Falsos negativos: El caso que falta, donde se predice 'negativo' pero en realidad está etiquetado como 'positivo'.

También así, se genera otra forma de expresar el *accuracy*:

$$Accuracy = \frac{Verdaderos\ positivos + Verdaderos\ negativos}{Muestra\ total}.$$

Por tanto, según qué problema nos interesará que las cantidades de alguno de los cuatro términos se incremente o decrezca, indicando así la importancia de clasificar bien o mal ciertas etiquetas, como en el anterior ejemplo.

3.5.3. Precisión

Se define como el número de verdaderos positivos dividido entre el número total de positivos predicho por el modelo. Esto indica el porcentaje de acierto del modelo en predecir la etiqueta 'positivo', es decir, si es un porcentaje alto, indica que si el modelo predice positivo, casi seguro será así.

$$Precision = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + Falsos\ positivos}.$$

3.5.4. Sensibilidad

En inglés *recall*. Está definida como el número de verdaderos positivos entre la suma de verdaderos positivos y falsos negativos e indica si clasifica correctamente la etiqueta positiva.

$$Recall = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + Falsos\ negativos}.$$

3.5.5. Métrica F_1

Esta es la última métrica, llamada valor F_1 o más conocida en inglés F_1 *score*.
En su forma general, la métrica F_β se define como:

$$F_\beta = (1 + \beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall'},$$

donde usamos $\beta = 1$ para obetener la métrica.
Indica cómo de robusto es el modelo, es decir, si se equivoca en un número significativo de etiquetas, y cómo de preciso es (cuántas instancias clasifica correctamente).

4 Conjunto de datos

Definiremos los conceptos más relevantes para el estudio del conjunto de datos al que se han aplicado los modelos.

4.1. Conceptos genéticos relevantes

Las cuatro bases nitrogenadas, también llamadas **nucleótidos**, que se encuentran en el ADN son Adeninda (A), Guanina (G), Citosina (C) y Timina (T) y son las unidades estructurales básicas del genoma. A la mutación de una sola posición de un nucleótido se la llama ***Single Nucleotide Polymorphism*** (SNP) y a cada posible tipo nucleótido se le llama **alelo**. Normalmente los SNPs son bialélicos, es decir, sólo presentan dos posibles tipos de nucleótidos y de éstos, los SNPs se suelen referir sólo a los que tengan una frecuencia de alelo menos común (*MAF* en inglés) mayor al 1 %. Definimos esta frecuencia como la frecuencia del segundo alelo más común en un determinado **locus** (posición de un SNP o gen en un cromosoma), que para los SNPs bialélicos será el alelo menos común y para los trialélicos o tetralélicos se tomará el segundo más común. Un locus puede ser homocigótico cuando los dos alelos son el mismo y heterocigótico cuando son diferentes. [Su et al., 2007]
Los SNPs tienen un gran interés en el estudio de enfermedades complejas puesto que representan el 90 % de todas las variaciones genéticas humanas y los SNPs con una MAF de al menos el 1 % ocurre cada 100 o 300 bases del genoma humano que tiene unas 3000 millones de pares de bases. A pesar de que estas variaciones del ADN humano representan menos de un 1 % (el 99 % restante es igual en toda la población), pueden tener un mayor impacto en cómo los humanos responden a enfermedades, haciendo de los SNPs unos valores muy importantes para el desarrollo de medicinas o para el diagnóstico médico. [Su et al., 2007]
Los SNPs también han demostrado ser relevantes a la hora de encontrar genes relacionados con enfermedades complejas como cáncer, diabetes y afecciones como el trastorno del espectro autista, que es lo que trabajamos aquí. Sin embargo, es complicado establecer estas relaciones a partir de un sólo gen alterado así que hay que alejarse de los medios convencionales para encontrar estas relaciones, como por ejemplo el *machine learning*. [Su et al., 2007]

Los cromosomas humanos vienen en pares: uno de la madre y otro del padre, y al conjunto de alelos que una persona tiene en un par de cromosomas se le llama **genotipo** y el **genotipado** es el método para obtenerlo. El término genotipo puede incluir los alelos del SNP que una persona tiene en un determinado locus o muchos SNPs a lo largo del genoma. Al conjunto de los alelos asociados de los SNPs en el mismo cromosoma se le llama **haplotipo**.
Un método muy común y actual es el ***Polymerase Chain Reaction*** (PCR).

Cuando son muchos los individuos que hay que genotipar, se utilizan los llamados microarrays, una tecnología que se usó primero para estudiar la expresión de muchos genes a la vez y más adelante para genotipar. Los microarrays han hecho posible identificar, clasificar y asignar funciones a muchos genes no caracterizados, simplemente mediante la determina-

ción de cuándo los genes se expresan o están reprimidos [Estudiantes, 2012]. Son chips de tamaño pequeño con una superficie sólida a la que se le une una colección de fragmentos de ADN. El color y la **intensidad** de fluorescencia en cada punto permite identificar las diferencias genéticas clave. Esta intensidad es la que se nos da para estudiar en el conjunto de datos.

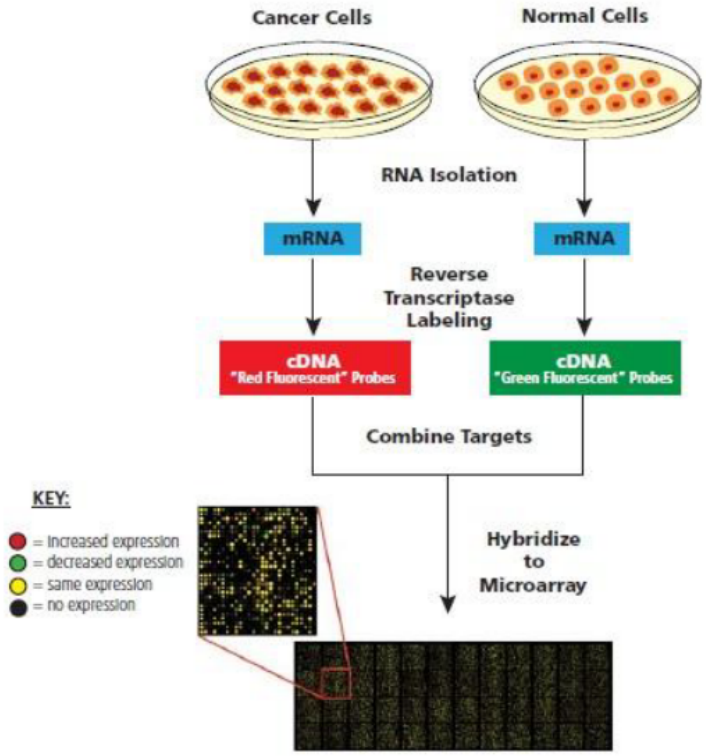

Figura 4.1: Análisis de microarrays. [Estudiantes, 2012]

Por último, cabe resaltar el concepto de **GWAS**. Son siglas en inglés que dan nombre al *estudio de asociación de genoma completo*, un enfoque de la investigación genética para asociar variaciones genéticas con enfermedades. Se trata de analizar el genoma de un gran número de personas y buscar algún rasgo genético que sirva para predecir la enfermedad o afección. En este caso, Trastornos del Espectro Autista (en inglés, *Autism Spectrum Disease*, ASD).

4.2. Materiales

Definimos ahora en esta sección el conjunto de datos utilizado y en concreto, el conjunto o *dataset* utilizado para el estudio.

4.2.1. Conjunto de datos completo

Los datos de autismo han sido proporcionados por el National Center for Biotechnology Information (NCBI) a través del dbGaP (por sus siglas, database of Genotypes and Phenotypes) y los datos de control por el European Genome-Phenome Archive (EGA). Los ficheros están en el formato original IDAT, habiendo dos ficheros para cada individuo. Investigadores del área de bioinformática del grupo UTAI de la UGR los han preprocesado para extraer las intensidades originales de cada SNP en un archivo único por cada conjunto de datos y cromosoma con extensión .iou, uno por cada cromosoma (sin contar el sexual). Cada archivo se compone de una cierta cantidad de filas (individuos, especificados en Tabla 6.23) y 62441 columnas donde las 7 primeras contienen información del individuo y el resto son valores numéricos de las intensidades de los SNPs en ambos alelos. Todos los datos han sido genotipados con el chip de Illumina 1.2 Duo aunque personalizados en algunos SNPs para los datos de ASD lo que hace que no se puedan comparar directamente y es el objeto de estudio de este trabajo.

Conjunto de datos				
Dataset	Otro nombre	Nº Individuos	Plataforma	Nº SNPs
AGP1M	Autism NoDuo	1194 (398 trios)	ILLUMINA 1M	1069796
AGP1MDuo	Autism Duo	2592 (864 trios)	ILLUMINA 1M	1069796
1958BC	BC	2930 control individuals	ILLUMINA 1M	1069796
NBS	NBS	2699 control individuals	ILLUMINA 1M	1069796

Tabla 4.1: Información de los datasets

Los datos y las intensidades han sido extraídas de los ficheros originales en formato IDAT (uno por individuo) con la siguiente descripción de columnas por investigadores de la línea de bioinformática del grupo de investigación UTAI:

- Columna 1: Código de familia. Es un identificador alfanumérico asignado de forma unívoca a cada trío o familia (hijo y dos padres).
- Columna 2: Código del individuo. Identificador único para cada individuo con el formato "n_m"donde n es el código de familia antes mecionado y m el ID.
- Columna 3: Código del padre. Representa el ID del padre del individuo. Su valor es 0 si el individuo no es un hijo.
- Columna 4: Código de la madre. El ID de la madre del individuo, que es 0 si no es un hijo.
- Columna 5: Sexo. Vale 1 si el individuo es de sexo masculino y 2 si es femenino.
- Columna 6: Estado de afección. Es nuestra variable a predecir, donde 0 indica paciente sano y 2 enfermo.
- Columna 7: Es el mismo código individual de la columna 2.
- Columna 8 hasta 7+2*$TotalSNPs$: Valores numéricos de los SNPs. Van por pares representando los pares de alelos.

4.2.2. Conjunto de datos estudiado

Para este trabajo se ha llevado a cabo el estudio del ADN mitocondrial. Los datos, tanto los de tríos como los sanos de control vienen dado en archivos .iou, cuyo nombre finaliza en 'Chrom26' ya que se ha trabajado con el cromosoma 26, que hace referencia al material genético mitocondrial. Los datos se presentan con la siguiente estructura:

- Carpeta 'Autism': Contiene el archivo .iou con el dataset y un archivo .col con los nombres de cada SNP en el mismo orden que aparecen en los pares de columnas.
- Carpeta 'NBS': Los primeros datos de control que he usado, también en un archivo.iou. Contiene 2699 individuos sanos con dos posiciones menos de SNPs que los tríos.
- Carpeta 'BC': Con 2920 individuos, contiene los segundos datos de control.
- Carpeta 'MS': El archivo .col de esta carpeta contiene en orden los nombre de los SNPs para los dos datos de control.

Las columnas se describen de la misma forma que en el apartado anterior.
Para llevar a cabo el estudio de los datos a través de los modelos predictivos, se deben eliminar las 7 primeras columnas a excepción de la sexta (el estado de afección) como etiqueta a predecir durante el aprendizaje supervisado.
Los conjuntos no presentan datos perdidos pero sí hay dos categorías SNPs en el conjunto de los tríos que no están presentes en los datos de control: *mitoc10874t* y *mitog10590a*, que los borramos del conjunto de los tríos para tener exactamente el mismo formato en los tres.
En el primer estudio llevado a cabo sólo con los datos de las familias, encontramos que hay 864 afectados (que son los hijos) y 1728 individuos presuntamente sanos (los padres). En el segundo y tercer estudio llevados a cabo con los primeros y segundos datos de control respectivamente, se han eliminado los individuos hijos y se han dejado a los padres marcando su estado de afección con un 2 (afectado) y se han juntado con cada control marcados con 0 (sanos) para predecir qué individuos podrían tener hijos con trastornos del espectro autista (ASD en inglés).

5 Implementación

El código se ha desarrollado usando el software *Jupyter Notebook* en el lenguaje *Python*, concretamente en su versión 3 [Van Rossum and Drake, 2009]. La ventaja de *Jupyter Notebook* para este trabajo es su estructura visual con celdas y texto, la fácil implementación y ejecución del código en *Python* y que el resultado de las ejecuciones se queda grabado justo debajo de la celda ejecutada. Esto último es útil para la entrega del software, pues el conjunto de datos es privado y no puede entregarse y por tanto, hay que solicitar permiso para obtener los datos y ejecutar el código. Sin embargo, con Jupyter pueden verse los resultados de cada celda sin la necesidad de ejecutarlo. Una celda es un trozo de código en el cuaderno que puede ejecutarse independientemente del resto, si se han cargado antes las librerías necesarias en el programa, y se diferencia visualmente de las demás celdas con una caja. En este caso, se ha tenido que ejecutar todas las celdas secuencialmente pues la mayoría dependen de la ejecución de celdas anteriores, pero se ha dejado la estructura en celdas para poder intercalar con celdas de texto y hacer la ejecución más visual e intuitiva.
El código se estructura en seis partes:

- Exploración con datos de tríos: En esta primera exploración se han utilizado sólo los datos de la familia (sin los datos de control) para la experimentación.
- Primeros datos de control: Se han utilizado los padres del conjunto de tríos y se han juntado con los primeros datos de control para repetir la experimentación.
- Segundos datos de control: Padres y segundos datos de control para verificar los resultados obtenidos con los primeros controles.
- Diferencias entre ambos controles: Ambos conjuntos de datos de control para detectar sus diferencias.
- Selección de variables: Apartado con selección de variables, reducción de dimensionalidad y reentrenamiento de modelos.
- Diferencias de sesgo: Apartado con dos tests para comprobar las diferencias entre los conjuntos de datos debidas al sesgo del laboratorio.

El objetivo del programa es la detección de sesgo de laboratorio en datos genéticos, concretamente en datos genéticos de intensidades de SNPs con una estructura de carpetas y columnas definida en la Sección 4.2. Este sesgo puede deberse a diferencias en la cantidad de reactivo usado en cada experimento o en los tiempos de reacción ya que el genotipado se basa en la PCR que genera una replicación exponencial de la información genética de cada individuo que va a ser extraída. Además el programa calcula tests para medir la significación de ese sesgo, es decir, si las diferencias detectadas son aleatorias o es sesgo real, es decir, existe un patrón común a los SNPs que puede estar indicando un sesgo de laboratorio o del experimento.
El programa detecta el sesgo entre tres conjuntos de datos (tríos y los dos de control) mediante modelos de aprendizaje automático que clasifican los datos.

El código, llamado 'TFG.ipynb', puede verse y descargarse del siguiente repositorio: código

5.1. Paquetes utilizados

En esta sección se repasarán los paquetes que han sido necesarios incluir para la realización del código en *Python*.
Los paquetes que se han utilizado en el trabajo son los siguientes:

- *scikit-learn* [Pedregosa et al., 2011]: Es un paquete desarrollado específicamente para el aprendizaje automático en *Python* y es más utilizado en el código desarrollado. Se han usado distintos clasificadores, métricas y preprocesado de *datasets*.
- *pandas* [Wes McKinney, 2010]: Librería utilizada para la representación de datos. Es un paquete desarrollado para tratar, manipular y analizar datos con facilidad.
- *matplotlib* [Hunter, 2007]: Paquete para la creación de diversos tipos de gráficas, animaciones y visualizaciones interactivas en *Python*. En este trabajo se ha utilizado para la creación de diferentes gráficos.
- *scipy* [Jones et al., 01]: Es un paquete de código abierto desarrollado para ingeniería, matemáticas y ciencia en general. En este trabajo lo usamos para los tests de Wilcoxon y el T-Test.

5.2. Exploración de los datos

En las cuatro primeras secciones del código, se ha realizado una exploración de los datos sin ningún tipo de selección de variables o reducción de dimensionalidad. En la representación de los datos se ha usado un *dataframe* del paquete *pandas* [Wes McKinney, 2010], dejando como nombre de las filas el ID del individuo correspondiente y como nombre de las columnas o variables, se ha establecido el nombre del SNP, tanto para el alelo A como para el B.
Para la representación gráfica de un SNP, cada eje representa la intensidad de un alelo y los individuos pueden ser homocigóticos para uno de los alelos (intensidad 0 en el eje de abcisas), homocigóticos para el otro alelo (intensidad 0 en el otro eje de ordenadas) o heterocigóticos (distinto de 0 en ambos alelos). En el genoma mitocondrial, dado que generalmente se transmite siempre por la madre, no se produce recombinación en la transmisión genética y los individuos suelen ser homocigóticos.

5.2.1. Conjunto de datos de tríos y primer control

Antes que nada se ha realizado un estudio de la base de datos de tríos en el que no esperamos encontrar diferencias significativas, dado que los estudios del GWAS realizados con estos datos no obtuvieron buenos resultados [Szatmari et al., 2007], pero era necesario intentar diferenciar padres de hijos afectados y nos dará una idea de cómo son los datos y cómo llegan a comportase los modelos. Para empezar, se han leído sólo los datos de tríos mediante el paquete *pandas* [Wes McKinney, 2010] para leer tanto los datos como los nombres de las columnas. Los nombres de las columnas (nombres de las variables) se han repetido para que las parejas de columnas de valores de SNPs tengan el mismo nombre y se ha añadido tanto el Género como el ID familiar para seguir el formato de los datos explicado en el Capítulo 4.

Para los datos, se han eliminado todas las columnas de IDs excepto el familiar debido a que si no se eliminaran, los modelos harían una predicción perfecta sólo basándose en si el sujeto tiene ID de padre y madre (es decir, es un hijo y es afectado) o no (es padre y es sano). Se ha dejado el género para que el modelo lo tenga en cuenta pues en la herencia genética el género es influyente, así como el ID familiar para que se relacionen los tres individuos de una familia.
Después se ha separado en conjuntos train (entrenamiento) y test usando un 25 % de los individuos para el test y un 75 % para el train manteniendo la proporción entre sanos y enfermos y se han contabilizado cuántos individuos hay en cada conjunto quedando 438 sanos y 210 afectados en el test y 1290 sanos y 654 afectados en el conjunto de entrenamiento, con un total de 1944 instancias en train y 648 en test.
También para realizar un acercamiento más visual a los datos se han representado las intensidades de dos SNPs. En el eje de abscisas está el valor de la primera columna A para el SNP y en el de ordenadas el valor de la segunda columna B. No se ha modificado el valor de las intensidades de ninguna forma, están directamente representadas en la gráfica. Podemos ver dos colas, dos líneas en la gráfica que representan los individuos homocigóticos para un alelo y homocigóticos para el otro alelo. Además, podemos apreciar que las intensidades no están en forma de punto sino de línea reflejando la naturaleza exponencial de la *Polymerase Chain Reaction* (PCR), mostrando inviduos con intensidad total ($intensidad_{AB} = intensidad_A + intensidad_B$) muy baja y otros mucho más alta. La gráfica se ha dibujado mediante la librería *matplotlib* [Hunter, 2007].

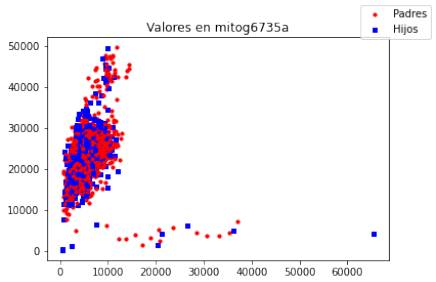

Figura 5.1: SNP mitog6735a en tríos.

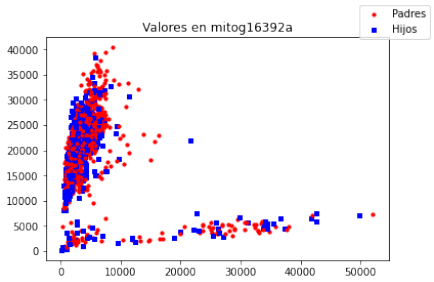

Figura 5.2: SNP mitog16392a en tríos.

Para los SNPs *mitog6735a* y *mitog16392a* tenemos marcados en azul los hijos y en rojo los padres para compararlos visualmente, mostrando una clara uniformidad y gran parecido en los valores. A través de estas gráficas, podemos intuir que va a ser difícil encontrar una manera de poder separar las dos clases, pues en ambas nubes de puntos están muy mezclados padres e hijos.

El siguiente paso fue intentar predecir si los padres iban a tener un hijo afectado quitando los individuos hijos de la base de datos de tríos, marcando los padres como afectados y añadiendo los individuos de control al conjunto marcados como sanos. También es necesario borrar dos SNPs, cuatro variables en el conjunto, que hay de más en los datos de tríos. En concreto son los SNPs mitoc10874t y mitog10590a.
El número de instancias total que queda en el conjunto es de 4427 y 273 variables. Se ha partido en conjuntos de entrenamiento (75 %) y test (25 %) quedando 3320 instancias en el entrenamiento con 2040 individuos sanos y 1280 padres de individuos afectados y en el test

1107 con 659 individuos sanos y 448 padres de afectados.
Visualmente se han representado dos SNPs con los valores del alelo A en el eje x y los del alelo B en el y para cada individuo.

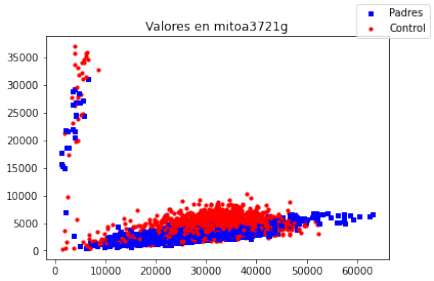

Figura 5.3: SNP mitoa3721g primer control.

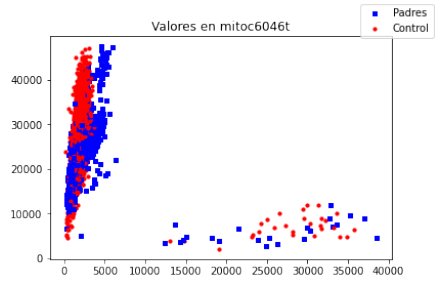

Figura 5.4: SNP mitoc6046t primer control.

En estas gráficas vemos representados los SNPs *mitoa3721g* y *mitoc6046t* con los padres en azul y el primer control en rojo para compararlos visualmente. En el *mitoa3721g* se puede comprobar como comentábamos antes, que los individuos son homocigóticos, en este caso hay más individuos con un valor 0 o cercano a 0 en el eje de ordenadas. Es en esta nube de puntos inferior en la que se puede apreciar a simple vista que los datos están diferenciados: los padres tienen en general un menor valor en el eje y. Para el SPN *mitoc6046t*, ocurre algo parecido pero en el eje de abscisas: los padres tienden a tener un valor mayor en el eje x, llevando así a poder generar una buena separación sólo visualmente.

5.2.2. Segundo conjunto de control

En el caso de los segundos datos de control se ha juntado con los datos de los tríos de la misma forma que con el primer control: se han eliminado los individuos afectados (hijos) de los datos de familias y se han marcado a los padres como afectados. Después se han eliminado los SNPs redundantes de los padres y se han juntado con estos segundos datos de control. Por último, se ha separado en conjuntos de entrenamiento y test en un 75 % y 25 % respectivamente quedando de las 4648 instancias totales, 3486 para el conjunto *train* (2211 sanos y 1275 padres de afectados) y 1162 para el test (con 709 sanos y 453 padres de afectados). Hay un desbalanceo de datos con 2920 individuos sanos y 1728 padres de afectados.
Se han representado dos SNPs con sus dos alelos A y B en el eje de abscisas y ordenadas respectivamente:

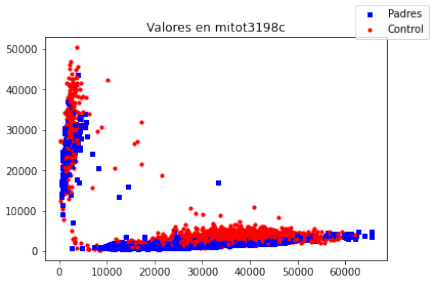

Figura 5.5: SNP mitot3198c segundo control.

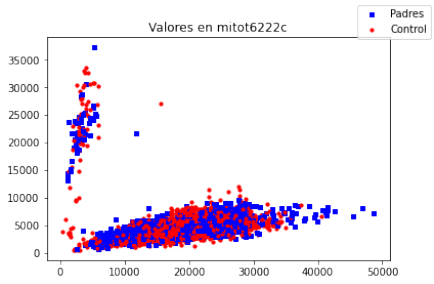

Figura 5.6: SNP mitot6222c segundo control.

En estas gráficas se presentan los SNPs *mitot3198c* y *mitot6222c* marcando en azul los padres y en rojo los datos del segundo control. A diferencia del primer control, para estos SNPs hay individuos homocigóticos para ambos alelos y encontramos también algunos heterocigóticos. A simple vista, parece que sólo para el primer SNP *mitot3198c* se diferencian claramente las dos clases y para el segundo no hay mucha separación entre ambas clases y los valores están repartidos de manera más uniforme.

5.2.3. Ambos datos de controles

Para comprobar las diferencias de sesgo que haya podido haber por la extracción de los datos en laboratorio, se ha hecho también un estudio sobre ambos datos de control. Se ha leído de la misma forma que los anteriores apartados pero esta vez no hay SNPs redundantes y se han separado en conjuntos de entrenamiento y test en un 75 % y 25 %. También se ha marcado con la etiqueta 0 los datos del primer control y con 1 los del segundo.
Esta vez los datos están más balanceados quedando de los 5619 individuos totales, 2920 son del segundo control y 2699 del primero. Para el test quedan 714 del primer control y 691 del segundo y para el entrenamiento, 1985 del primer control y 2229 del segundo control. Se han representado de nuevo dos de los SNPs:

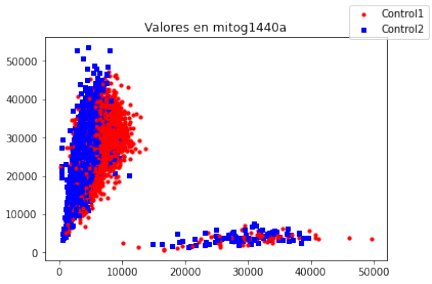

Figura 5.7: SNP mitog1440a dos controles.

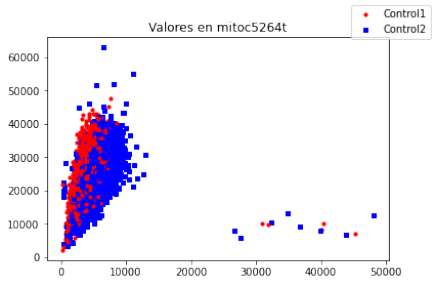

Figura 5.8: SNP mitoc5264t dos controles.

Las anteriores gráficas representan las intensidades de los alelos A y B de los SNPs *mitog1440a* y *mitoc5264t*, marcando en rojo el primer control y en azul el segundo. Se muestra una tendencia a ser homocigótico para un solo alelo y no hay ningún individuo heterocigótico. Ambas nubes de puntos muestran una separación que puede verse a simple vista, mostrando la diferencia para cada array a la hora de tomar los datos. En el caso del primer

SNP se ve una tendencia del priemer control a tener un mayor valor en el eje x y en el segundo SNP, esa misma tendencia es del segundo control, llevando así a poder diferenciar ambas clases de manera efectiva.

5.3. Preprocesado de datos y selección de variables

En general, en los modelos se ha trabajado con los datos *en crudo* (esto es, sin transformar ni seleccionar variables) excepto para el *GridSearch CV* que explicaremos más adelante y para la parte de selección de variables.
Las transformaciones de los datos que se han realizado para ejecutar un modelo KNN sobre el conjunto de datos de tríos han sido tres:

- **Variance Threshold**: De la librería *sklearn* [Pedregosa et al., 2011], elimina todos las características que no superan cierto umbral (*threshold* en inglés). Esto se hace para eliminar características muy parecidas y aporten más complejidad que información a los datos. Se ha usado el valor 0.01.

- **StandardScaler**: Estandariza los datos eliminando la media y escalando a varianza unidad. El valor del estandarizado en una observación x se calcula como $z = \frac{x-u}{s}$, donde u es la media de la muestra de entrenamiento y s es la desviación estándar de la muestra de entrenamiento. También usado de la librería *sklearn*.

- **PolynomialFeatures**: Añade una nueva matriz de calacterísticas de combinaciones de las características existentes con grado menor o igual que el indicado por parámetro, en este caso 2. Se usa para cambiar la distribución de probabilidad separando más valores pequeños y grandes. De nuevo, usando el implementado por *sklearn*.

También en el apartado de selección de variables para una reducción de dimensionalidad, se empieza por una matriz de correlación para ver visualmente las posibles correlaciones entre variables mediante el método `corr` de `LabelEncoder` de la librería *sklearn* y tras ello, lanzamos dos tests para la selección y ajustamos dos modelos para seleccionar las variables que nos indiquen.
Los dos test son el test χ^2 de independencia y el de *VarianceThreshold* explicado anteriormente. La hipótesis nula del test χ^2 es que las variables son idependientes de sus etiquetas, con lo que si nos devuelve un p-valor mayor que 0.05 aceptamos esta hipótesis y eliminamos esta variable del conjunto ya que no nos aporta al entrenamiento por ser idependiente del etiquetado que deseamos.
Los dos modelos entrenados para realizar la selección son *LinearSVC* y *ExtraTreesClassifier*, estos modelos se entrenan con el conjunto de entrenamiento y se seleccionan las variables que más hayan aportando al entrenamiento. En el primer modelo, se seleccionan las variables que tengan un aporte mayor a 10^{-5} por haber usado regularización L1 (especificado por parámetro al modelo) y en el segundo, los que estén por encima de la media (por defecto). Esta selección se lleva acabo con `SelectFromModel` de *sklearn*.
El modelo *LinearSVC* es un modelo de soporte vectorial como el que ya se ha explicado pero en su versión lineal, que permite la selección de variables. Se usan los parámetros $C = 0.01$, regresión L1 y $dual = False$ que es preferible cuando hay más muestras que variables como en este caso [Pedregosa et al., 2011]. Por otra parte el *ExtraTreesClassifier* es un meta-modelo que lo que hace es entrenar cierto número de árboles de decisión aleatorios

en varios subconjuntos de datos usando la media para mejorar el *accuracy* y controlar el sobreajuste [Pedregosa et al., 2011]. De parámetros se han escogido 50 árboles.

5.4. Modelos empleados y sus parámetros

Describimos en esta sección la implementación de los modelos utilizados y los parámetros de los mismos para la clasificación.
Para evaluar cada modelo se ha dibujado [Hunter, 2007] la matriz de confusión para cada clasificador. La matriz representa el número de falsos y verdaderos positivos así como el de falsos y verdaderos negativos, se ha realizado con la librería `metrics` de *scikit-learn* [Scikit-Learn, 2021e], contando en forma de matriz $2x2$ las etiquetas que el modelo ha predicho y su verdadera etiqueta, como se explicó en la Subsección 3.5.2. Además, mediante el método `classification_report` de `metrics` se han mostrado todas las métricas explicadas en la Sección 3.5 pasando por parámetro las etiquetas predichas por el modelo y las reales.

5.4.1. Modelo K-NN

Se ha utilizado la implementación del paquete *scikit-learn* [Scikit-Learn, 2021c]. En concreto, dentro del subpaquete `neighbours` el método `KNeighboursClassifier`. Se ha usado con los parámetros por defecto:

- Número de vecinos: 5, es un modelo 5-NN.
- Pesos: Uniforme, todos los vecinos tienen la misma importancia.
- Métrica: Se usa la de Minkowski.
- p: 2, se usa la distancia euclídea.

Este modelo se ha entrenado para las partes de tríos y del primer control, dando una primera idea del rendimiento del resto de modelos sobre estos conjuntos. Normalmente su rendimiento es relativamente pobre pero como comentábamos en secciones anteriores, lo compensa con su baja complejidad y su alta entendibilidad. También se ha probado a aplicar el preprocesamiento de datos de la Sección 5.3, lo que ha subido el tiempo de ejecución pero no mejoraba resultados, y por ello se descartó esta opción para el resto de modelos.

5.4.2. Regresión Logística

Para la regresión logística (RL) he establecido que los pesos estén balanceados para contrarrestar el gran desbalanceo de los datos (hay el doble de padres, no afectados, que de hijos, afectados y muchos más de controles que de padres de afectados). Los parámetros utilizados han sido los siguientes [Scikit-Learn, 2021d]:

- *Loss*: La función de pérdida, establecida a *log* (función logística) para que el modelo sea aplique una regresión logística como la explicada en el Capítulo 3.
- *penalty*: La regularización empleada. En este caso la L2 o de Ridge.
- *alpha*: Parámetro que multiplica a la regularización, cuanto más alto, mayor será la regularización aplicada al modelo y menor el sobreajuste. Establecida por defecto a 0.0001.
- *class_weight*: Pesos asociados con las variables. Se establecen como *balanced*.

Se ha usado en su versión estocástica del subpaquete `linear_model`, el método `SGDClassifier`. También se muestran los ceficientes resultantes del entrenamiento para cada variable mediante el método `coef_` y se han mostrado las más influyentes, es decir, las que tienen mayor coeficiente, para la predicción.

5.4.3. Árboles de decisión

Estableciendo de nuevo los datos balanceados para reducir el desbalanceo que presentan estos conjuntos de datos, los parámetros establecidos para el modelo han sido los siguientes [Scikit-Learn, 2021a]:

- Criterio para medir la calidad de una división: 'Gini'.
- Elegir la mejor división (parámetro *splitter=best*)
- Mínimo 2 muestras para dividir un nodo
- Mínimo una muestra para ser un nodo hoja

Se muestran también como en regresión logística las variables más importantes a la hora de predecir. esto se hace mediante el método `feature_importances_` del modelo de *DecisionTreeClassifier* de *sklearn*.

5.4.4. Máquinas de soporte vectorial

Se han entrenado dos versiones de este modelo, para el conjunto de datos de tríos se han establecido los pesos balanceados y se han usado los valores por defecto y para los primeros datos de control se han cambiado varios parámetros para mejorar los resultados. La implementación usada es la de la librería de *sklearn* [Scikit-Learn, 2021g]. Los parámetros modificados son:

- Parámetro 'C': Es el parámetro de regularización, cuanto mayor sea, menor es la regularización aplicada, es decir, son inversamente proporcionales. Por defecto es 1 y se ha establecido a 100.
- 'kernel': se usa *Radical Basis Function (RFB)*.
- Clases balanceadas como ya se venía usando con anterioridad.

Como en el resto de modelos, se muestra su matriz de confusión y las distintas métricas para medir el rendimiento del modelo.

5.4.5. Potenciación de gradiente

Se ha empleado la implementación de *sklearn* [Scikit-Learn, 2021b] para este modelo, más conocido como Gradient Boosting, que se ha usado en todas las secciones del código, incluso tras la selección de variables debido a sus buenos resultados, que veremos en la siguiente sección.
Se ha utilizado en su versión clasificadora con los siguientes parámetros:

- 'n_estimators': parámetro que representa el número de estimadores del modelo. Al ser muy robusto al sobreajuste [Scikit-Learn, 2021b], se ha establecido a un número alto para mejorar su rendimiento, 200.

- 'learning rate': disminuye la contribución de cada árbol. Establecido a 0.9, lo que compensa el alto número de estimadores.
- 'max_depth': La profundidad de cada estimador individual, es decir, limita el número de nodos del árbol. Establecido a 2.

5.4.6. *GridSearch CV*

Esto no es un modelo en sí, sino un método para entrenar distintos modelos predictivos mediante la llamada **validación cruzada**.
Empecemos definiendo la validación: es un proceso por el cual gaurdamos una parte de los datos de entrenamiento para evaluar el modelo una vez entrenado y darnos una idea de cómo se comporta el modelo a la hora de predecir los datos del entrenamiento y comprobar si puede haber sobreajuste. Ahora bien, este método puede extenderse a lo que se conoce como **validación cruzada**, que se trata de dividir en N subconjuntos los datos de entrenamiento, entrenar con validaciones y obtener la media aritmética de los resultados para obtener uno único. Como cabe esperar, el proceso es altamente costoso computacionalmente hablando, es decir, es lento.
Así, *GridSearch CV* usa validación cruzada para obtener los mejores parámetros de un conjunto de modelos predictivos y, de entre ellos, obtiene el modelo con mejores resultados. Esto hace que sea aún más lento pero obtenga muy buenos resultados. Se han utilizado los siguientes modelos con las siguientes posibilidades de parámetros:

- Regresión logística con regularización L1, un máximo de iteraciones igual a 500 como parámetros fijos y 'lbfgs' como 'solver'; como parámetro variable se ha elegido:
  - 'C': Cinco valores espaciados uniformemente en la escala logarítmica del -2 al 2.
- Bosque aleatorio, con los siguientes parámetros.
  - 'n_jobs': Indica la paralelización del proceso, como está establecido a -1, se usan todos los procesadores.
  - 'max_depth': La profundidad máxima del árbol, variable entre 100 y 250.
  - 'criterion': La función para medir la calidad de la división de un nodo. Se usa una alternativa a 'Gini' llamada 'entropy'.
  - 'n_estimators': Puede ser 100 o 250.
- Máquinas de soporte vectorial que usan clases balanceadas, un 'gamma' de tipo 'scale' fijo, grado 2 del 'kernel' 'poly', 'C' variando entre cinco posibles valores en la escala logarítmica y un 'kernel' posible entre 'rbf' y 'poly'.

De entre ellos, se ha obtenido el mejor modelo con los mejores parámetros de los posibles para evaluar con las distintas métricas explicadas.

5.5. Tests de medias

Para comprobar si las diferencias son aleatorias o debidas a un sesgo real, se ha llevado a cabo dos tests de medias para comprobar si efectivamente los datos tienen medias distintas y hay sesgo real o si son diferencias aleatorias y las medias son iguales. Para estos tests se

ha usado la implementación de *scipy* [Jones et al., 01].
El primer test utilizado es el conocido como ***T-test*** para las medias de dos muestras independientes. Este test tiene como hipótesis nula que dos muestras independientes tienen la misma media. Se asume que tienen la misma varianza. El test se ha obtenido para cada alelo de los dos conjuntos de datos que se comparan (datos y controles o ambos controles) y se obtiene el *p-value* que nos indicará si hay diferencias en la media o no. Este es un primer paso para ver si hay diferencias significativas en cada SNP entre los dos conjuntos de datos
Para el segundo test se ha usado el de ***los rangos con signo de Wilcoxon*** que tiene como hipótesis nula que dos pares de muestras tienen la misma distribución. Para la implementación se ha obtenido la media en cada columna de los datos a testear y se ha pasado este vector como parámetro para realizar el test. Con ello, se comprueba el *p-value* y se saca la conclusión oportuna que veremos en el siguiente apartado de resultados. En este paso se comprueba si las diferencias de intensidades tienen siempre el mismo sentido, es decir, si la mayoría de SNPs tienen un valor mayor en un conjunto de datos que en otro. Esto se hace para ver si hay sesgo en los datos debido al laboratorio, ya sea por distintas dosis del reactivo o distintos tiempos de reacción en la prueba para obtener las intensidades.

6 Resultados finales y conclusiones

En este capítulo se expondrán los resultados obtenidos y un análisis de los mismos por los distintos modelos y se comentará su rendimiento en las bases de datos a las que se han aplicado. La variable a clasificar es en el caso de los tríos, si el individuo es sano o enfermo, en el caso del primer y segundo control, si el individuo es sano o es padre de algún afectado y en el caso de ambos controles, si el individuo es del primer control o del segundo. También se presentarán los resultados de los tests de medias.

6.1. K Nearest Neighbours

Se alcanza una precisión de 0.591 y de hasta un 0.6641 usando validación cruzada (*cross validation* en inglés) con 5 particiones del conjunto. De media, la validación cruzada ha conseguido una precisión del 0.62347.

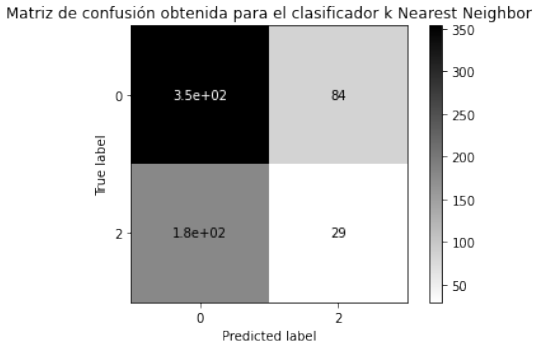

Figura 6.1: Matriz para KNN en tríos

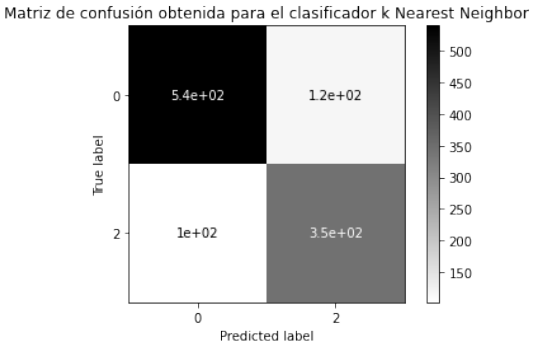

Figura 6.2: Matriz para KNN en el primer control

También se ha realizado un modelo con el preprocesado de datos explicado en la sección anterior. Sin embargo, este preprocesado sólo ha aumentado el tiempo de ejecución, pues no ha conseguido mejorar los resultados de la ejecución con los datos 'en crudo' por lo que se descarta para el resto de modelos más complejos que además aumenta exponencialmente el tiempo de ejecución.

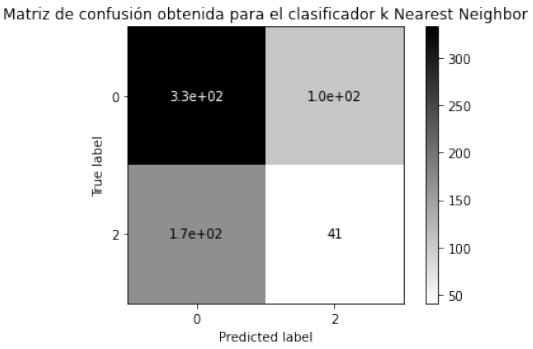

Figura 6.3: KNN en tríos con preprocesado.

También con la librería *metrics* de *scikit-learn* se ha obtenido el valor para cada métrica:

Conjunto de datos				
	precision	recall	f1-score	support
0	0.66	0.81	0.73	438
2	0.26	0.14	0.18	210
accuracy			0.59	648
macro avg	0.46	0.47	0.45	648
weighted avg	0.53	0.59	0.55	648

Tabla 6.1: Métricas obtenidas para K-NN en tríos

Conjunto de datos				
	precision	recall	f1-score	support
0	0.66	0.76	0.71	438
2	0.28	0.20	0.23	210
accuracy			0.58	648
macro avg	0.47	0.48	0.47	648
weighted avg	0.54	0.58	0.55	648

Tabla 6.2: Métricas obtenidas para K-NN en tríos con preprocesado

Los resultados para este modelo, como se podía intuir en la representación gráfica de los SNPs, no es muy buena con un recall de 0.47 o 0.48 con preprocesado y una medida F1 de 0.18 o 0.23 con preprocesado para la etiqueta de afectado. El modelo predice bien a los no afectados debido al desbalance de los datos pues es más probable que tenga cerca a más no afectados que a afectados y por ello, sólo ha predicho bien los individuos sanos.

Para los primeros datos de control se ha repetido el entrenamiento del mismo modelo dando una precisión de hasta 0.8136 en *cross validation*. La tabla de métricas ha sido la siguiente:

Conjunto de datos				
	precision	recall	f1-score	support
0	0.84	0.82	0.83	659
2	0.74	0.77	0.76	448
accuracy			0.80	1107
macro avg	0.79	0.80	0.79	1107
weighted avg	0.80	0.80	0.80	1107

Tabla 6.3: Métricas obtenidas para K-NN en el primer control

Esta vez los datos estaban más balanceados y a simple vista más diferenciados entre padres de afectados y individuos sanos, con lo que se han logrado mejores resultados.

6.2. Regresión Logística

Llegamos ahora a un modelo que, aunque lineal, es más complejo que KNN con lo que arrojará mejores resultados y además nos premitirá ver una gráfica con las variables más importantes en la predicción.
Los coeficientes de cada variable en la regresión logística indican el grado de importancia en el entrenamiento del modelo, así que se han ordenado los SNPs según su importancia quedando de la siguiente forma los diez más importantes:

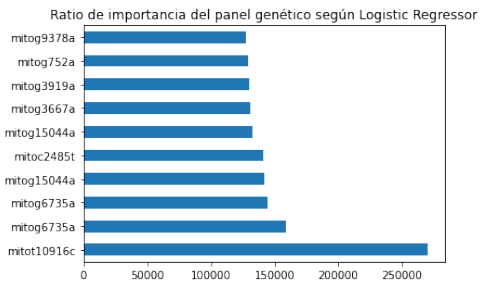

Figura 6.4: Ratio de importancia Regresión Logística en tríos

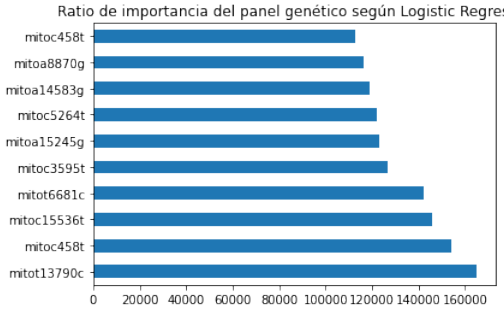

Figura 6.5: Variables importantes control 1

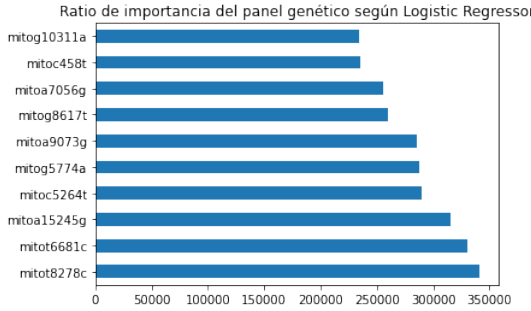

Figura 6.6: Variables importantes control 2

La importancia en estos diez primeros en los datos de tríos es muy parecida entre sí, con valores de coeficientes relativamente bajos, lo que lleva a pensar que para el modelo no ha habido ningún alelo especialmente significativo.
Sin embargo, las variables más influyentes cuando entrenamos padres de afectados y controles, son parecidas para ambos casos, esto puede deberse a que los datos de control se tomaron en el mismo laboratorio así que es lógico que las diferencias de sesgo se encuentren en variables parecidas. Entre ellas, tenemos a *mitoa15245g* que se encuentra en una posición alta de importancia para el primer y segundo control y a *mitox458t*.
Ahora pasamos a analizar el rendimiento de este modelo mediante las métricas presentadas en secciones anteriores. Se podrá observar unos mejores resultados que en el modelo de KNN aunque para los datos de tríos sigue sin conseguir resultados decentes, lo que será una constante ya que en un mismo array no existe sesgo de laboratorio.

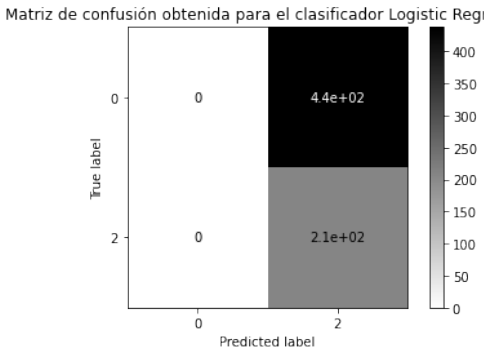

Figura 6.7: Matriz para RL en tríos

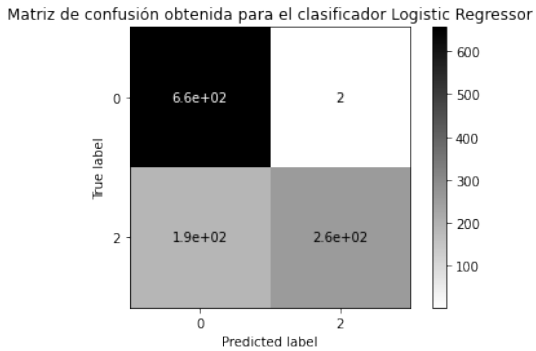

Figura 6.8: Matriz para RL en el primer control

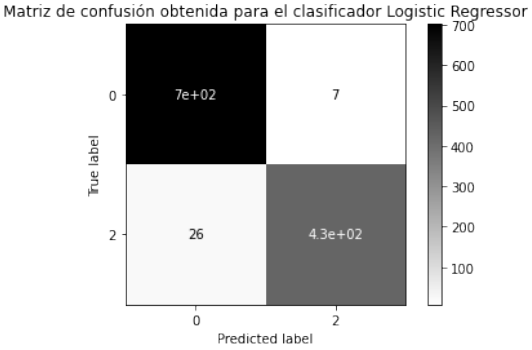

Figura 6.9: Matriz para RL en el segundo control

Efectivamente, con este modelo en los tríos no se ha conseguido tampoco buenos resultados, lo que empieza a indicar que a penas hay diferencias significativas entre padres e hijos en el material genético de estos SNPs. Siempre clasifica con la etiqueta 2. Sin embargo, en el caso del primer control, los resultados mejoran bastante y en el caso del segundo control podemos intuir puntuaciones muy altas pues casi no tiene datos mal clasificados: 29 para la etiqueta 2 y solamente 7 para la etiqueta 0. Presentamos ahora las métricas que respaldan estas afirmaciones:

Conjunto de datos				
	precision	recall	f1-score	support
0	0.00	0.00	0.00	438
2	0.32	1.00	0.49	210
accuracy			0.32	648
macro avg	0.16	0.50	0.24	648
weighted avg	0.11	0.32	0.16	648

Tabla 6.4: Métricas obtenidas para Regresión Logística en datos de tríos

Conjunto de datos				
	precision	recall	f1-score	support
0	0.78	1.00	0.87	659
2	0.99	0.58	0.73	448
accuracy			0.83	1107
macro avg	0.88	0.79	0.80	1107
weighted avg	0.86	0.83	0.81	1107

Tabla 6.5: Métricas obtenidas para Regresión Logística en el primer control

Conjunto de datos				
	precision	recall	f1-score	support
0	0.96	0.99	0.98	709
2	0.98	0.94	0.96	453
accuracy			0.97	1162
macro avg	0.97	0.97	0.97	1162
weighted avg	0.97	0.97	0.97	1162

Tabla 6.6: Métricas obtenidas para Regresión Logística en el segundo control

Sobre todo en este último caso, los resultados son muy buenos, con una precisión del 0.9628 con lo que se puede diferenciar entre ambos arrays de forma casi perfecta usando este modelo.
Para el caso de los tríos en este modelo, tiene unos resultados curiosos: clasifica siempre con la etiqueta 2. Por ello, cabe resaltar la importancia de usar distintas métricas y no sólo la *accuracy* o *recall*. Estas dos últimas métricas tienen un valor medio de 0.32 y 0.5, indicando que aunque el modelo predice mal, no es tan malo como podemos ver en la matriz de confusion. Sin embargo, en este caso la métrica *precision* nos da una mejor valoración con 0.11 de puntuación y el desglose de las métricas para cada etiqueta también nos indica el fallo del modelo: todos los valores de éstas en la etiqueta 0 son 0.00 indicando de nuevo que nunca predice bien para este caso. Si nos quedáramos solo con las métricas típicas no habríamos podido resaltar este caso y tendríamos un menor entendimiento de lo que ha pasado.

6.3. Árboles de decisión

Subimos un escalón en la complejidad de los modelos y llegamos a los árboles de decisión (AD para abreviar),‘ que tendrán un rendimiento parecido a la regresión logística, en algunos casos superior.
Para este modelo también se pueden obtener las principales variables que han aportado al entrenamiento del mismo:

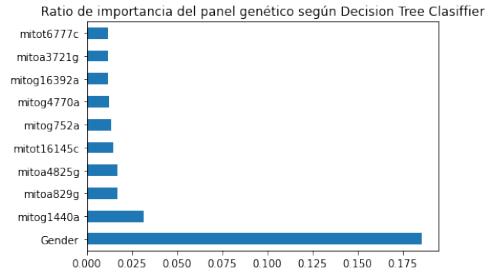

Figura 6.10: Ratio de importancia Árboles de Decisión en tríos

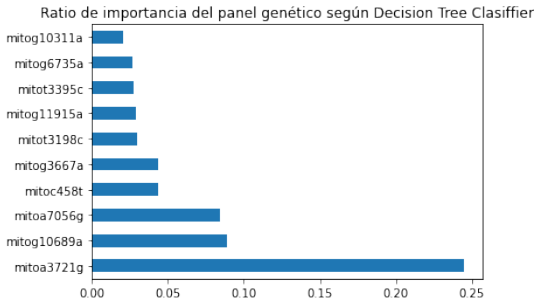

Figura 6.11: Importancia AD control 1

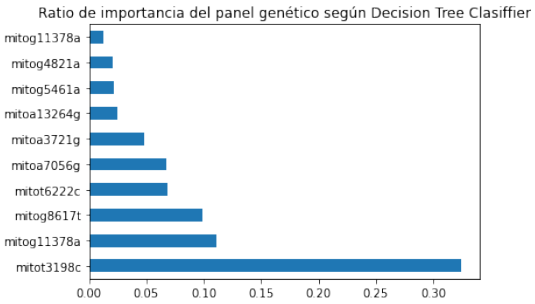

Figura 6.12: Importancia AD control 2

Vemos como algunas variables importantes coinciden con las vistas en la regresión logística para los casos de control: *mitoc498t* y *mitog10311a* son ejemplos de ello. Además, entre ambos conjuntos de datos de padres más control coinciden variables como *mitot3198c*, reforzando así que las diferencias son parecidas por venir ambos controles del mismo laboratorio. En el caso de los tríos lo más impotante ha sido el género, es decir, no ha usado casi el resto de variables en su predicción y las diferencias halladas no son de utilidad pues son un simple patrón en los datos debido al género y no al material genético.
Ahora se presentan las evaluaciones de las métricas sobre este modelo y sus matrices de confusión:

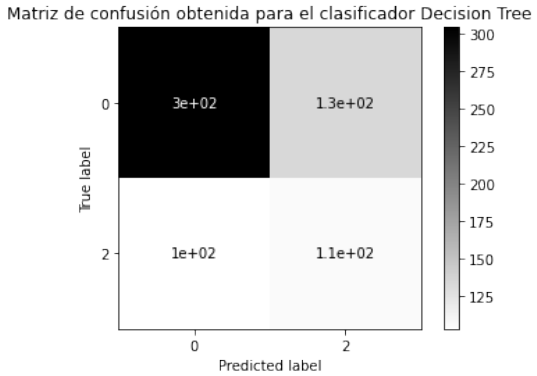

Figura 6.13: Matriz para AD en tríos

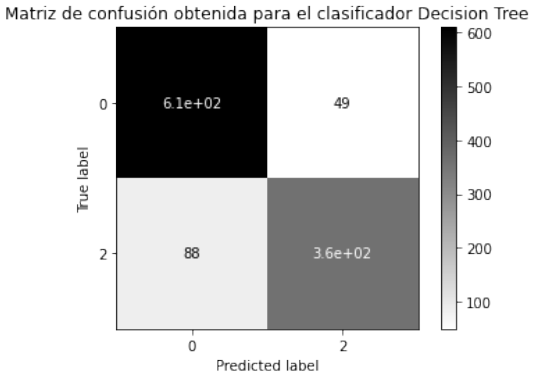

Figura 6.14: Matriz para AD en el primer control

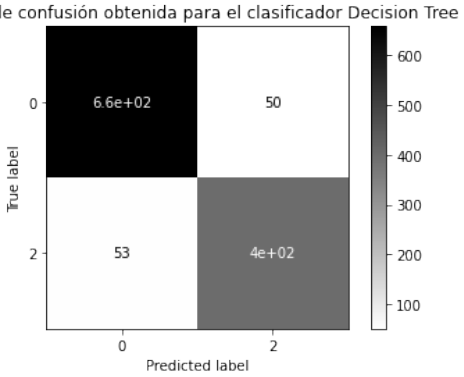

Figura 6.15: Matriz para AD en el segundo control

Conjunto de datos				
	precision	recall	f1-score	support
0	0.75	0.69	0.72	438
2	0.44	0.51	0.47	210
accuracy			0.63	648
macro avg	0.60	0.60	0.60	648
weighted avg	0.65	0.63	0.64	648

Tabla 6.7: Métricas obtenidas para AD en datos de tríos

Conjunto de datos				
	precision	recall	f1-score	support
0	0.87	0.93	0.90	659
2	0.88	0.80	0.84	448
accuracy			0.88	1107
macro avg	0.88	0.86	0.87	1107
weighted avg	0.88	0.88	0.88	1107

Tabla 6.8: Métricas obtenidas para AD en el primer control

Conjunto de datos				
	precision	recall	f1-score	support
0	0.93	0.93	0.93	709
2	0.89	0.88	0.89	453
accuracy			0.91	1162
macro avg	0.91	0.91	0.91	1162
weighted avg	0.91	0.91	0.91	1162

Tabla 6.9: Métricas obtenidas para AD en el segundo control

Vemos que en los datos de tríos aunque los resultados de las métricas en media sean parecidos entre el anterior modelo y este, los de los árboles de decisión es mucho más significativa pues no predice solo una etiqueta como sí pasa en regresión logística. En el primer control se llega a unos resultados también superiores a regresión logística pero en el segundo, árboles de decisión se queda algo atrás a pesar de ser también buenos resultados. Esto indica aún con más certeza que entre arrays distintos existe ese sesgo pero en un mismo array no.

De hecho, esta última idea cobra aún más sentido al comprobar que también puede diferenciarse entre datos de un control u otro. Para ello, se han entrenado tres modelos cuyos resultados se presentarán en su correspondiente subsección. Uno de estos modelos son los árboles de decisión que obtienen de nuevo muy buenos resultados para diferenciar entre ambos controles. Esta vez la etiqueta 0 es para datos del primer control y la 1 para datos pertenecientes al segundo.

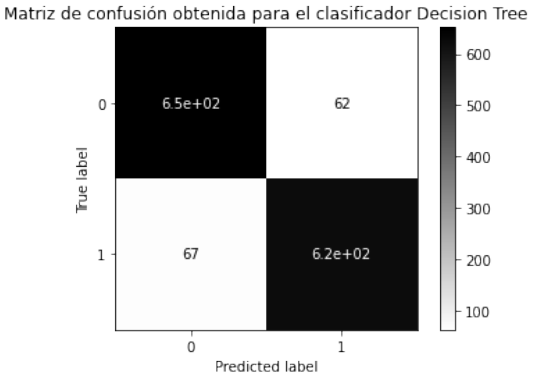

Figura 6.16: Matriz para AD en el ambos controles

Conjunto de datos				
	precision	recall	f1-score	support
0	0.91	0.91	0.91	714
1	0.91	0.90	0.91	691
accuracy			0.91	1405
macro avg	0.91	0.91	0.91	1405
weighted avg	0.91	0.91	0.91	1405

Tabla 6.10: Métricas obtenidas para árboles de decision en ambos controles

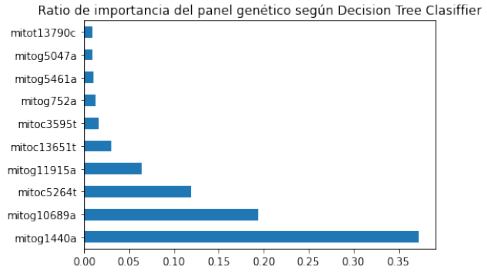

Figura 6.17: Ratio de importancia Árboles de Decisión en ambos controles

Tenemos de nuevo entre las variables más importantes, algunas que ya hemos visto en los datos de padres y control para este mismo modelo, por ejemplo, *mitog10689a*, lo que quiere decir que son las variables en las que más se diferencian entre arrays, seguramente debido al sesgo.

6.4. Máquinas de soporte vectorial

Este modelo (SVM para abreviar) se ha usado para los tríos, primeros datos de control y ambos controles, obteniendo de los mejores resultados para todos los modelos presentados. En tríos sin embargo, tampoco consigue clasificar de forma eficaz.
Los resultados son los que se muestran a continuación:

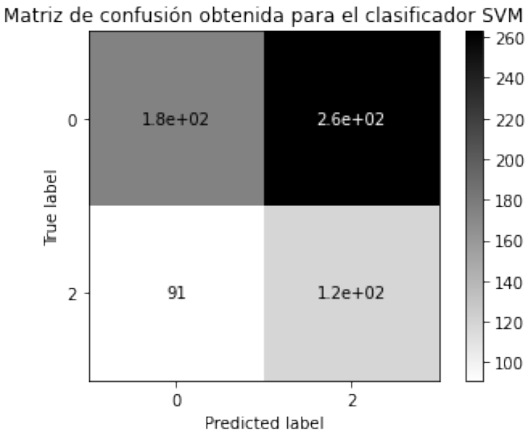

Figura 6.18: Matriz para SVM en tríos

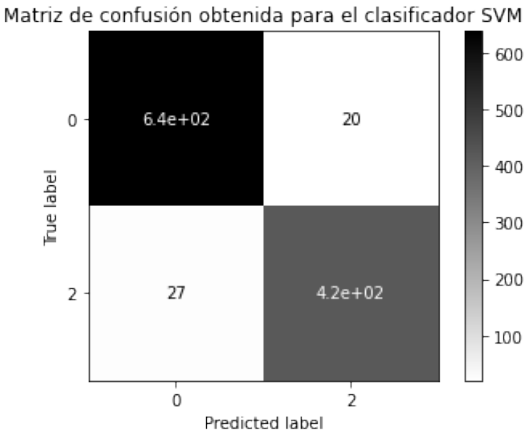

Figura 6.19: Matriz para SVM en el primer control

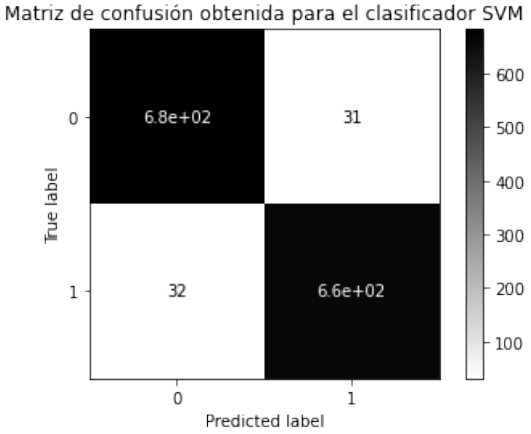

Figura 6.20: Matriz para SVM en ambos controles

Conjunto de datos				
	precision	recall	f1-score	support
0	0.66	0.40	0.50	438
2	0.31	0.57	0.40	210
accuracy			0.45	648
macro avg	0.48	0.48	0.45	648
weighted avg	0.55	0.45	0.47	648

Tabla 6.11: Métricas obtenidas para SVM en datos de tríos

Conjunto de datos				
	precision	recall	f1-score	support
0	0.96	0.97	0.96	659
2	0.95	0.94	0.95	448
accuracy			0.96	1107
macro avg	0.96	0.95	0.96	1107
weighted avg	0.96	0.96	0.96	1107

Tabla 6.12: Métricas obtenidas para SVM en el primer control

Conjunto de datos				
	precision	recall	f1-score	support
0	0.96	0.96	0.96	714
1	0.96	0.95	0.95	691
accuracy			0.96	1405
macro avg	0.96	0.96	0.96	1405
weighted avg	0.96	0.96	0.96	1405

Tabla 6.13: Métricas obtenidas para SVM en ambos controles

Es un modelo con unos resultados *balanceados* en el conjunto de tríos, es decir, que aunque no obtenga unos resultados buenos, se mantiene al rededor de 0.5 en cada métrica para cada etiqueta con lo que podemos afirmar que aproximadamente la mitad de las veces va a predecir bien, a diferencia de regresión logística que predice siempre bien una etiqueta y siempre mal la otra.
En los otros dos conjuntos podemos ver resultados extremadamente buenos: con este modelo se pueden distinguir arrays de SNPs distintos, indicando ese sesgo que tiene como objetivo detectar este trabajo. Las máquinas de soporte vectorial demuestran ser un buen predictor para este tipo de datos.

6.5. Potenciación de gradiente

El *Gradient Boosting* (GB para abreviar) obtiene unos resultados parecidos e incluso superiores a algunos casos a las máquinas de soporte vectorial y además pueden mostrarse las variables que más han aportado al entrenamiento. Es por ello que se ha usado para todos los conjuntos de datos. Mostramos a continuación los resultados obtenidos:

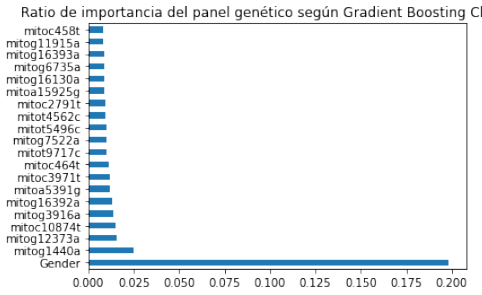

Figura 6.21: Importancia GB tríos

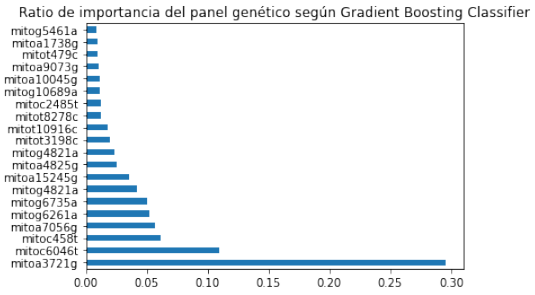

Figura 6.22: Variables importantes GB control 1

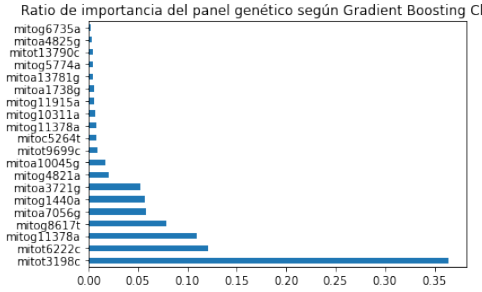

Figura 6.23: Importancia GB control 2

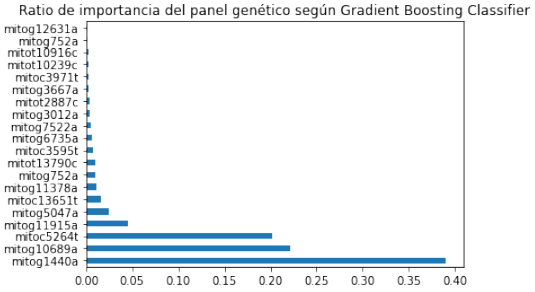

Figura 6.24: Importancia GB ambos controles

Como ya veníamos comentando, los modelos coinciden en cierta medida sobre las variables más importantes en su entrenamiento. En este caso, la potenciación de gradiente coincide con los árboles de decisión claramente en que el género es la variable más importante para los tríos lo que nos indica que las diferencias dentro del array de SNPs no son claras. También en los controles 1 y 2 coinciden en los SPNs más influyente, así como en el conjunto de ambos controles lo cual indica que hay posiciones SNP con más sesgo que otras con lo que se pueden usar estas para diferenciar entre distintos arrays.
Mostramos ahora el rendimiento de *gradient boosting* con los valores de las distintas métricas:

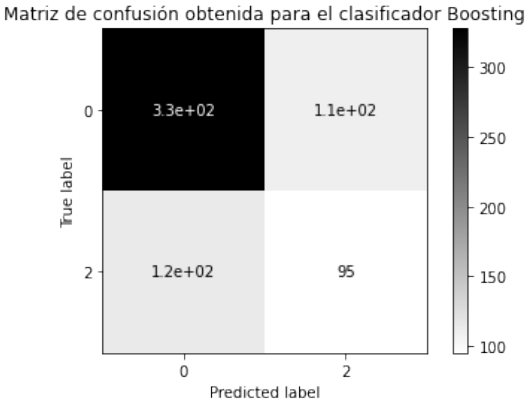

Figura 6.25: Matriz para GB en tríos

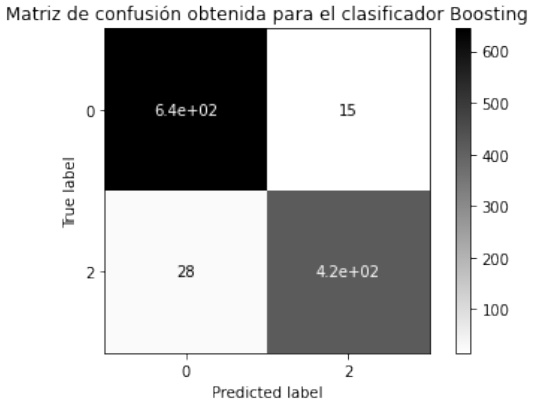

Figura 6.26: Matriz para GB en el primer control

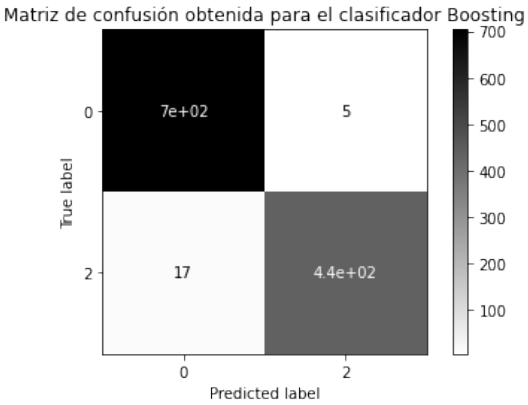

Figura 6.27: Matriz para GB en el segundo control

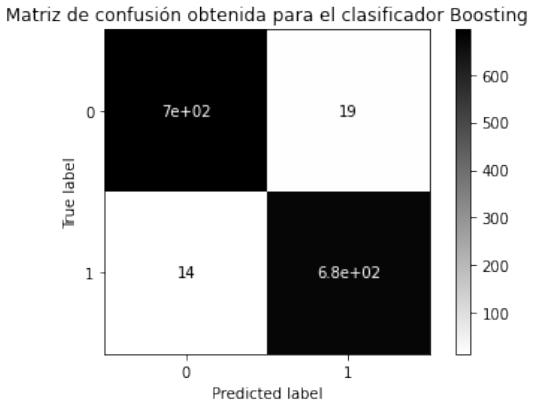

Figura 6.28: Matriz para GB ambos controles

Conjunto de datos				
	precision	recall	f1-score	support
0	0.74	0.75	0.74	438
2	0.46	0.45	0.46	210
accuracy			0.65	648
macro avg	0.60	0.60	0.60	648
weighted avg	0.65	0.65	0.65	648

Tabla 6.14: Métricas obtenidas para GB en datos de tríos

Conjunto de datos				
	precision	recall	f1-score	support
0	0.96	0.98	0.97	659
2	0.97	0.94	0.95	448
accuracy			0.96	1107
macro avg	0.96	0.96	0.96	1107
weighted avg	0.96	0.96	0.96	1107

Tabla 6.15: Métricas obtenidas para GB en el primer control

Conjunto de datos				
	precision	recall	f1-score	support
0	0.98	0.99	0.98	709
2	0.99	0.96	0.98	453
accuracy			0.98	1162
macro avg	0.98	0.98	0.98	1162
weighted avg	0.98	0.98	0.98	1162

Tabla 6.16: Métricas obtenidas para GB en el segundo control

Conjunto de datos				
	precision	recall	f1-score	support
0	0.98	0.97	0.98	714
1	0.97	0.98	0.98	691
accuracy			0.98	1405
macro avg	0.98	0.98	0.98	1405
weighted avg	0.98	0.98	0.98	1405

Tabla 6.17: Métricas obtenidas para GB en ambos controles

Como comentábamos, obtiene buenos resultados en conjuntos de datos provenientes de distintos arrays al igual que las máquinas de soporte vectorial, probando ser un modelo robusto y con resistencia al sobreajuste. En el caso de los tríos, ocurre lo mismo: el modelo se queda atrás y no consigue predecir como debería ya que el conjunto de datos proviene del mismo array de SNPs. Sin embargo, en este conjunto consigue los mejores resultados de todos los modelos entrenados.

6.6. Validación cruzada

Por último, se ha utilizado la validación cruzada mediante *GridSearch CV* (abreviado CV) donde se han entrenado los métodos explicados en el Capítulo 5. Se ha utilizado este método

para los conjuntos de datos de tríos y del primer control. En el primer caso, el modelo con mejor rendimiento ha sido el bosque aleatorio con 200 estimadores y una profundidad máxima de 9 y en el segundo han sido las máquinas de soporte vectorial con un parámetro $C = 100$ y un núcleo tipo 'rbf'. Se presentan a continuación los resultados:

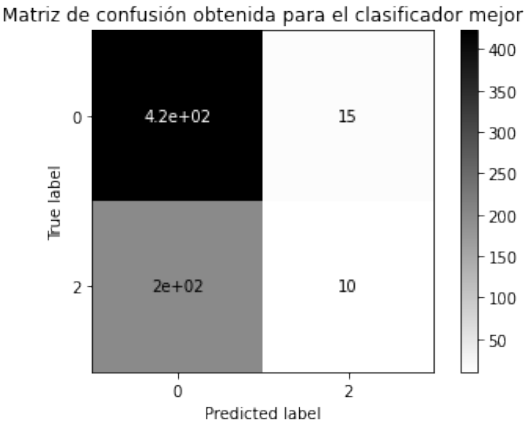

Figura 6.29: Matriz para CV en tríos

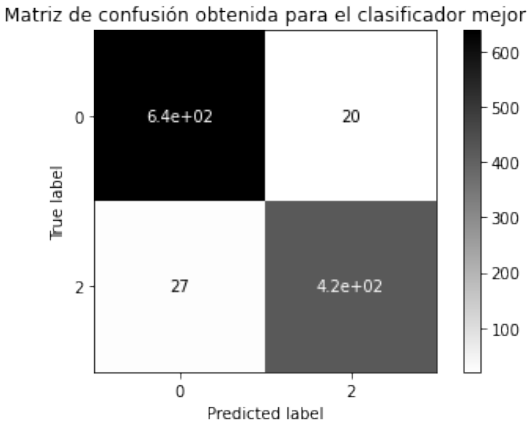

Figura 6.30: Matriz para CV en primer control

Conjunto de datos				
	precision	recall	f1-score	support
0	0.68	0.97	0.80	438
2	0.40	0.05	0.09	210
accuracy			0.67	648
macro avg	0.54	0.51	0.44	648
weighted avg	0.59	0.67	0.57	648

Tabla 6.18: Métricas obtenidas para CV en datos de tríos

Conjunto de datos				
	precision	recall	f1-score	support
0	0.96	0.97	0.96	659
2	0.95	0.94	0.95	448
accuracy			0.96	1107
macro avg	0.96	0.95	0.96	1107
weighted avg	0.96	0.96	0.96	1107

Tabla 6.19: Métricas obtenidas para CV en el primer control

En el conjunto de tríos ocurre algo parecido a la regresión logística: predice casi siempre una sola etiqueta además de producirse sobreajuste, un 0.87 de *accuracy* en el conjunto de

entrenamiento y tan solo 0.66 en el de test. Sin embargo, en el primer control consigue un muy buen modelo para diferenciar entre los padres de afectados y los individuos de control.

6.7. Selección de variables

Presentamos ahora los resultados de la selección de variables llevada a cabo. Se ha aplicado al conjunto de datos de padres de afectados y el primer control. Primero, se ha dibujado la matriz de correlación para, en un primer vistazo, intuir las variables que podrían aportar poco al entrenamiento por depender linealmente de otras.

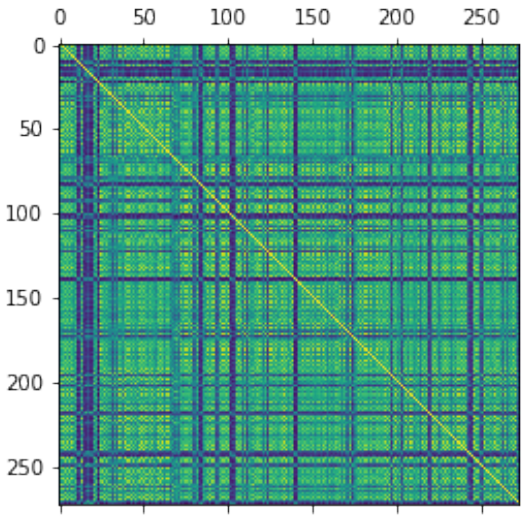

Figura 6.31: Matriz de correlación

Se representan en amarillo las correlaciones más fuertes. Podemos ver algunas fuera de la diagonal principal de la matriz que llevan a pensar que se podrían eliminar y reducir así la dimensionalidad para facilitar el entrenamiento y quitar ruido en los datos.
Primero, aplicamos el test χ^2 entre las 273 variables y el conjunto de etiquetas, que nos lleva a que sólo hay una variable independiente de las etiquetas y que por tanto no nos interesa pues no va a aportar al entrenamiento: el género del individuo. Se muestra a continuación la gráfica con los *p-values* de las distintas variables ordenadas de mayor a menor.

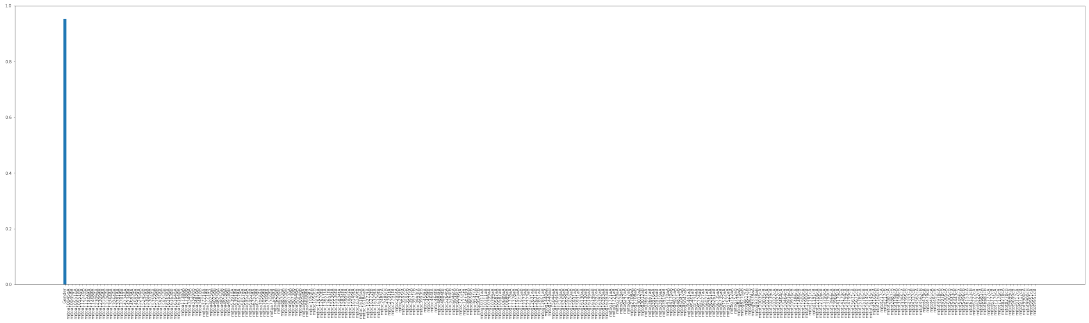

Figura 6.32: *p-values* del test χ^2

Ahora con las 272 variables restantes, aplicamos *variance threshold* con el valor del parámetro a $0.8(1-0.8)$. Sin embargo, esto no nos lleva a eliminar ninguna variable así que hay que probar con selecciones basadas en la importancia del entrenamiento en modelos predictivos, que ya explicamos en el Capítulo 5. Uno de ellos es el basado en la norma $L1$ mediante másquinas de soporte vectorial en su versión lineal. Esta selección nos deja con 258 que aún son una gran cantidad de variables. Por último aplicamos la selección basada en árboles que es la que más cantidad de variables elimina, dejando tan solo un total de 76 columnas.
Pasamos ahora al reentrenamiento de algunos modelos con esta selección de variables para comprobar su rendimiento en una menor dimensionalidad. Con las 76 variables se han reentrenado los árboles de decisión, la potenciación del gradiente y la validación cruzada. Los resultados han sido muy parecidos a los que veíamos con los datos en crudo, indicando que incluso con esta reducción de dimensionalidad, se puede diferenciar perfectamente entre arrays de SNPs distintos manteniendo las variables más influyentes del entrenamiento. Además las variables más importantes del entrenamiento para árboles de decisión y para potenciación del gradiente son de nuevo muy parecidas.

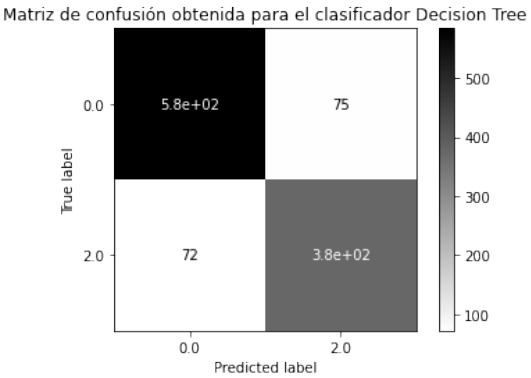

Figura 6.33: Matriz selección variables AD

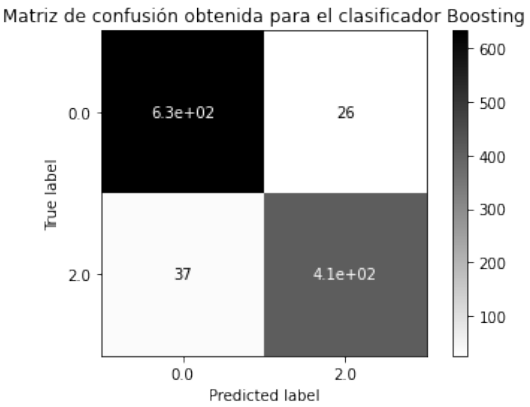

Figura 6.34: Matriz selección variables GB

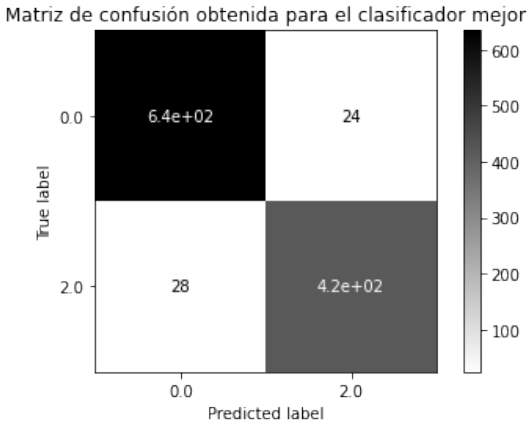

Figura 6.35: Matriz de confusión para CV en selección variables

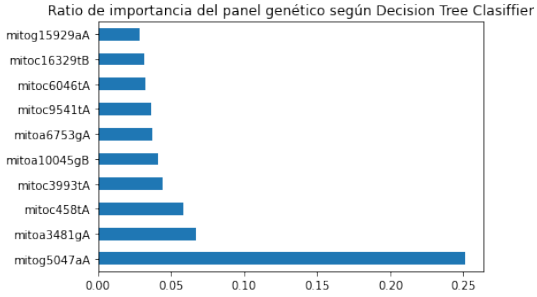

Figura 6.36: Importancia de variables AD

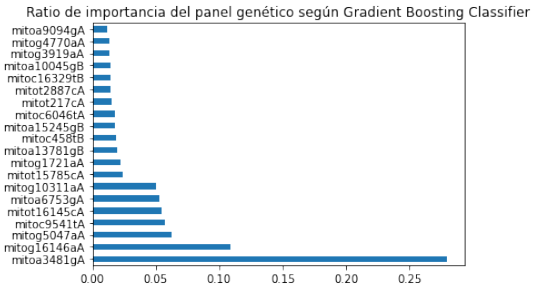

Figura 6.37: Importancia de variables GB

Conjunto de datos				
	precision	recall	f1-score	support
0	0.89	0.89	0.89	659
2	0.83	0.84	0.84	448
accuracy			0.87	1107
macro avg	0.86	0.86	0.86	1107
weighted avg	0.87	0.87	0.87	1107

Tabla 6.20: Métricas obtenidas para AD en la selección de variables

Conjunto de datos				
	precision	recall	f1-score	support
0	0.94	0.96	0.95	659
2	0.94	0.92	0.93	448
accuracy			0.96	1107
macro avg	0.94	0.94	0.94	1107
weighted avg	0.94	0.94	0.94	1107

Tabla 6.21: Métricas obtenidas para GB en la selección de variables

Conjunto de datos				
	precision	recall	f1-score	support
0	0.96	0.96	0.96	659
2	0.95	0.94	0.94	448
accuracy			0.95	1107
macro avg	0.95	0.95	0.95	1107
weighted avg	0.95	0.95	0.95	1107

Tabla 6.22: Métricas obtenidas para CV en la selección de variables

Por último, se han reentrenado con sólo la selección de variables llevada a cabo hasta la basada en la norma $L1$ los modelos de validación cruzada, obteniendo así los mejores resultados de todo el estudio con puntuación perfecta en el conjunto de entrenamiento y casi perfecta en el conjunto de test. El mejor modelo que ha seleccionado *GridSearch CV* han sido las máquinas de soporte vectorial con un valor de $C = 10$ y un núcleo tipo 'rbf'. Ha conseguido un *accuracy* del 100 % en entrenamiento y del 98.103 % en test. Se presentan a continuación los resultados de las métricas:

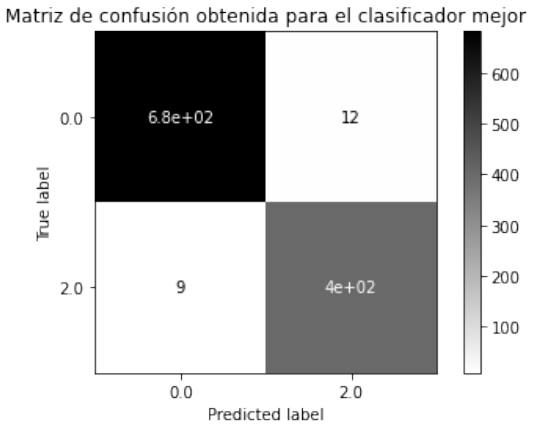

Figura 6.38: Matriz de confusión para CV en selección variables basada en L1

Conjunto de datos				
	precision	recall	f1-score	support
0	0.99	0.98	0.98	659
2	0.97	0.98	0.97	448
accuracy			0.98	1107
macro avg	0.98	0.98	0.98	1107
weighted avg	0.98	0.98	0.98	1107

Tabla 6.23: Métricas obtenidas para CV en la selección de variables basada en L1

Demostrando así la utilidad de reducir dimensionalidad en algunos conjuntos de datos con muchas variables, como es nuestro caso.

6.8. Tests de medias

En esta sección se presenta el estudio de las medias de los conjuntos y la comprobación de si las diferencias encontradas son aleatorias o podemos clasificarlas como sesgo real de laboratorio. Lo primero que se ha hecho ha sido una representación gráfica de las medias de los arrays que se están comparando (padres y controles 1 y 2, y entre ambos controles). Se presentan a continuación estas gráficas:

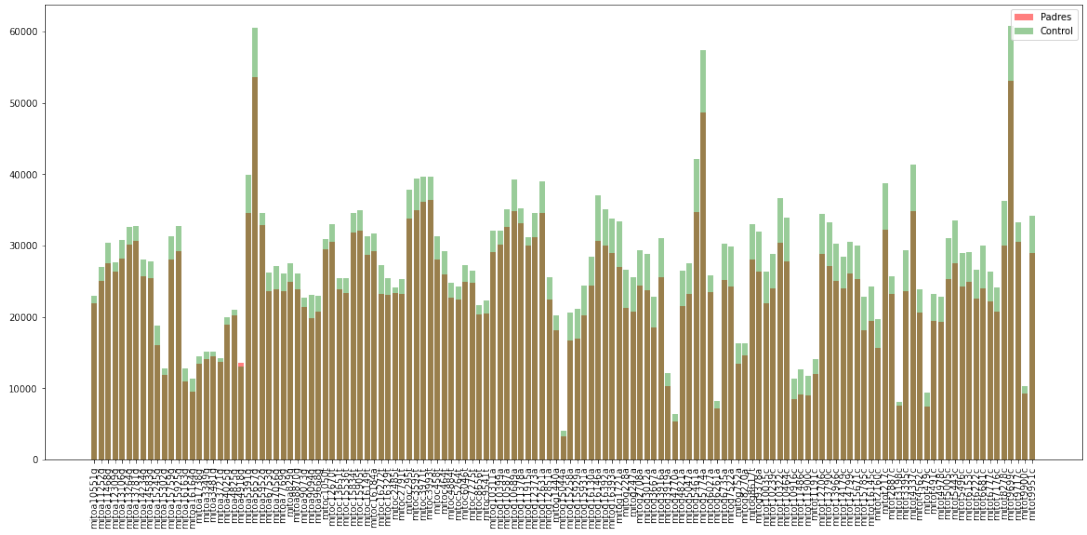

Figura 6.39: Gráfica comparativa de las medias de los padres de afectados y el primer control

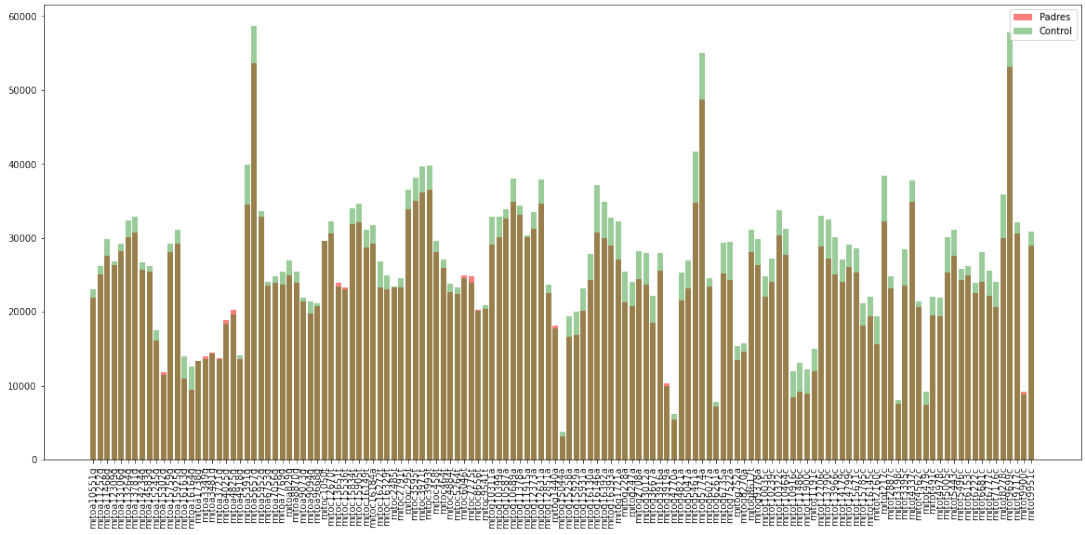

Figura 6.40: Gráfica comparativa de las medias de los padres de afectados y el segundo control

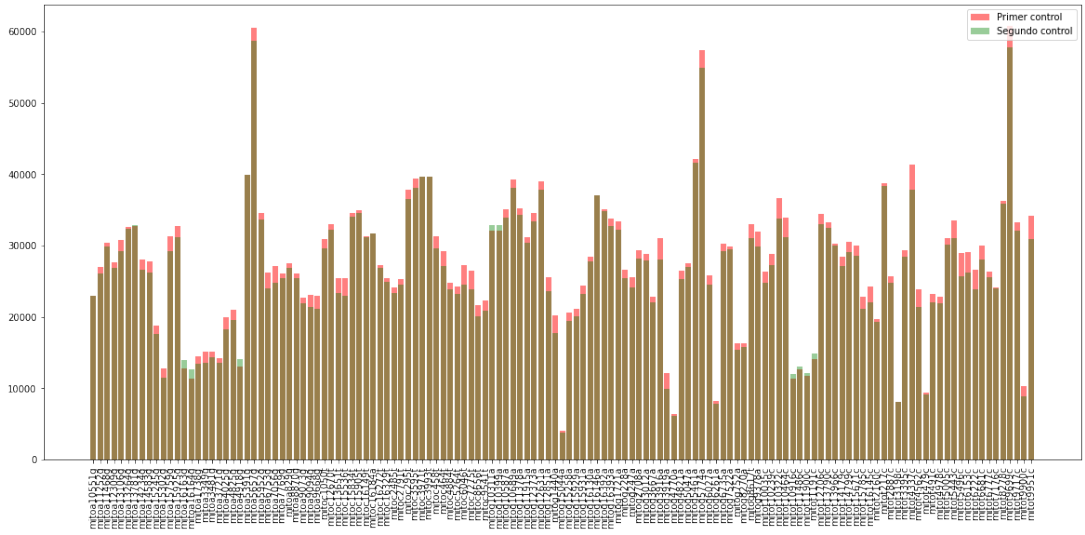

Figura 6.41: Gráfica comparativa de las medias de ambos controles

Para crear los gráficos y hacer los tests, se han leído los datos esta vez sumando las dos posiciones de cada SNPs y haciendo la media de entre todos los individuos.
En las gráficas anteriores podemos ver que hay cierta tendencia en las medias de los datos de control a ser superiores a las de los padres y más aún en el primer control. Esto indica

que es bastante posible que los test nos den el resultado esperado: las diferencias no son aleatorias.
Primero se ha realizado el t-test para cada SNP. Obteniendo que se dan diferencias aleatorias en muy pocas posiciones con un *p-value* alto, y en el resto las diferencias son debidas al sesgo de laboratorio.

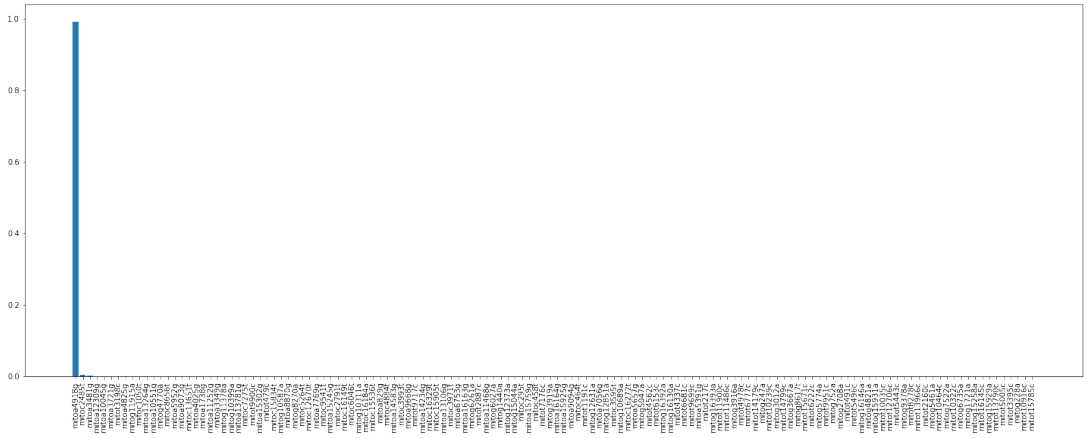

Figura 6.42: *p-values* del t-test para el primer control y padres de afectados

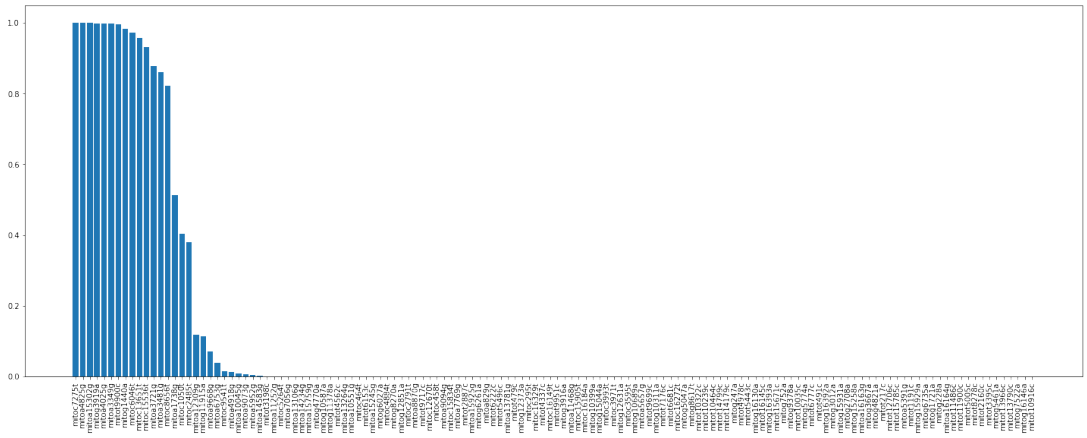

Figura 6.43: *p-values* del t-test para el segundo control y padres de afectados

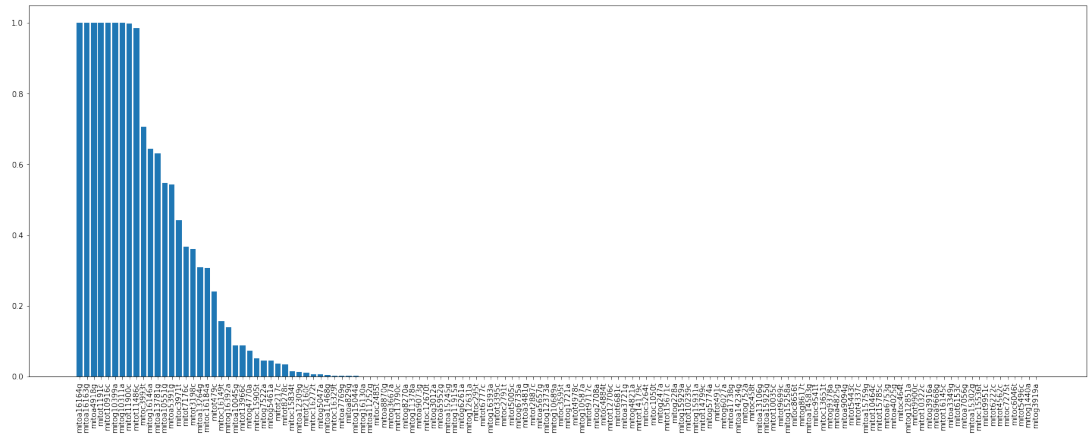

Figura 6.44: *p-values* del t-test para ambos control

Efectivamente, hay varias posiciones con un alto *p-value* indicando que las diferencias son debidas a la aleatoriedad, pero en general podemos afirmar que existe ese sego de laboratorio y las diferencias vienen de las medias de la distribución que siguen los datos.
Esto se reafirma cuando evaluamos el test de Wilcoxon. Se hace para las medias de cada conjunto de datos obteniendo en el primer control un *p-value* de 3.43×10^{-24}, en el segundo 2.55×10^{-22} y entre ambos controles de 4.37×10^{-20}, lo que indica que en los tres casos se rechaza la hipótesis nula de que las medias de las poblaciones siguen la misma distribución, es decir, las diferencias que se hayan en ellas no son debidas a la aleatoriedad sino diferencias de sesgo reales.

7 Conclusiones y trabajo futuro

En este trabajo se ha estudiado, no sólo la aplicación de técnicas de aprendizaje automático a datos genéticos, sino el sesgo de laboratorio que hay entre datos que vienen de distintos laboratorios o datos obtenidos de distintas fechas ya sea por diferencias en la cantidad de ractivo, tiempo de reacción, distinta hibridación cruzada como consecuencia de personalizaciones aplicadas al array, personalizaciones aplicadas *a priori* por los laboratorios a los datos... sobre todo en afecciones complicadas de detectar genéticamente como el trastorno del espectro autista.

Como punto de partida, se han introducido conceptos necesarios de estadística y aprendizaje automático. También se ha estudiado cómo se relacionan los errores sobre los conjuntos de entrenamiento y test de los modelos predictivos mediante la cota VC. A continuación, se ha presentado de forma general la idea intuitiva detrás de cada algoritmo de aprendizaje básico así como de las métricas usadas.

Tras ello, se ha explicado la implementación seguida en el programa desarrollado para aplicar estos modelos, métricas y tests de contraste de hipótesis utilizados, las librerías de donde se obtienen y sus principales parámetros para por último presentar los resultados finales del trabajo.

El trabajo ha sido realmente un trabajo de investigación. En un primer momento no se tenía certeza de que el sesgo de laboratorio que se ha ido buscando realmente estuviera ahí sino que se intuía. Como conclusión del trabajo, el objetivo se considera cumplido pues se ha conseguido diferenciar, con diferencias que no son aleatorias, los distintos arrays y no se ha conseguido establecer una diferencia clara en un mismo array, como se esperaba. A partir de esto, se dejan abiertas varias vías de trabajo futuras:

- Deshacerse del sesgo de laboratorio detectado para poder identificar correctamente SNPs asociados con este tipo de afecciones.
- La aplicación de ésta y otro tipo de técnicas de aprendizaje para detectar el sesgo o predecir SNPs asociados con enfermedades distintas a las empleadas aquí.
- Utilizar técnicas de aprendizaje profundo en vez de aprendizaje clásico como el utilizado en este trabajo.

Bibliografía

Las referencias se listan por orden alfabético. Aquellas referencias con más de un autor están ordenadas de acuerdo con el primer autor.

[Abu-Mostafa et al., 2012] Abu-Mostafa, Y. S., Magdon-Ismail, M., and Lin, H.-T. (2012). *Learning from data*, volume 4. AMLBook New York, NY, USA:. [Citado en págs. ix, ix, ix, ix, ix, xvii, 17, 18, 19, 21, 25, 26, 27, 29, 30, 31, and 32]

[Anderson et al., 1958] Anderson, T. W., Anderson, T. W., Anderson, T. W., and Anderson, T. W. (1958). *An introduction to multivariate statistical analysis*, volume 2. Wiley New York. [Citado en págs. 9, 10, 11, and 13]

[Biau, 2012] Biau, G. (2012). Analysis of a random forests model. *The Journal of Machine Learning Research*, 13(1):1063–1095. [Citado en pág. 34]

[Chomboon et al., 2015] Chomboon, K., Chujai, P., Teerarassammee, P., Kerdprasop, K., and Kerdprasop, N. (2015). An empirical study of distance metrics for k-nearest neighbor algorithm. pages 280–285. [Citado en pág. 28]

[Cogill and Wang, 2016] Cogill, S. and Wang, L. (2016). Support vector machine model of developmental brain gene expression data for prioritization of Autism risk gene candidates. *Bioinformatics*, 32(23):3611–3618. [Citado en pág. 35]

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297. [Citado en págs. ix, 35, 36, 37, 39, 40, 41, and 42]

[Cunningham and Delany, 2007] Cunningham, P. and Delany, S. (2007). k-nearest neighbour classifiers. *Mult Classif Syst*. [Citado en págs. 28 and 29]

[Estudiantes, 2012] Estudiantes (2012). ADN/ARN Microarrays. [Citado en págs. ix and 46]

[Forsyth, 2018] Forsyth, D. (2018). *Probability and statistics for computer science*. Springer. [Citado en págs. 1, 2, 3, 5, and 7]

[Garthwaite et al., 2002] Garthwaite, P. H., Jolliffe, I. T., Jolliffe, I., and Jones, B. (2002). *Statistical inference*. Oxford University Press on Demand. [Citado en págs. 5 and 6]

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org. [Citado en pág. xvii]

[Hans, 2009] Hans, C. (2009). Bayesian lasso regression. *Biometrika*, 96(4):835–845. [Citado en pág. 27]

[Hunter, 2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95. [Citado en págs. 50, 51, and 55]

[Hyde et al., 2019] Hyde, K. K., Novack, M. N., LaHaye, N., Parlett-Pelleriti, C., Anden, R., Dixon, D. R., and Linstead, E. (2019). Applications of supervised machine learning in autism spectrum disorder research: a review. *Review Journal of Autism and Developmental Disorders*, 6(2):128–146. [Citado en pág. xvii]

[Jones et al., 01] Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. [Citado en págs. 50 and 58]

[Jordan and Mitchell, 2015] Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260. [Citado en págs. xvii, xvii, and 16]

[Libbrecht and Noble, 2015] Libbrecht, M. W. and Noble, W. S. (2015). Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321–332. [Citado en pág. xvii]

[Müller et al., 2016] Müller, A. C., Guido, S., et al. (2016). *Introduction to machine learning with Python: a guide for data scientists*. .ºReilly Media, Inc.". [Citado en pág. xvii]

[Nilsson, 2009] Nilsson, N. J. (2009). *The quest for artificial intelligence*. Cambridge University Press. [Citado en pág. xvii]

[Nowak, 2009] Nowak, R. (2009). Lecture 19: The Proof of the Vapnik-Chervonenkis (VC) Inequality. [Citado en pág. 25]

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. [Citado en págs. 50, 54, and 55]

[Peterson, 2009] Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, 4(2):1883. [Citado en pág. 28]

[Scikit-Learn, 2021a] Scikit-Learn (2021a). Decision Tree Classifier. [Citado en págs. 33 and 56]

[Scikit-Learn, 2021b] Scikit-Learn (2021b). Gradient Boosting Classifier. [Citado en págs. 35 and 56]

[Scikit-Learn, 2021c] Scikit-Learn (2021c). K Nearest Neighbors Classifier Python Implementation. [Citado en pág. 55]

[Scikit-Learn, 2021d] Scikit-Learn (2021d). Logistic Regression Classifier. [Citado en pág. 55]

[Scikit-Learn, 2021e] Scikit-Learn (2021e). Metrics. [Citado en pág. 55]

[Scikit-Learn, 2021f] Scikit-Learn (2021f). Random Forest Classifier. [Citado en pág. 34]

[Scikit-Learn, 2021g] Scikit-Learn (2021g). SVC. [Citado en pág. 56]

[Su et al., 2007] Su, S.-C., Kuo, C.-C. J., and Chen, T. (2007). Single nucleotide polymorphism data analysis-state-of-the-art review on this emerging field from a signal processing viewpoint. *IEEE Signal Processing Magazine*, 24(1):75–82. [Citado en págs. xvii, xvii, and 45]

[Szatmari et al., 2007] Szatmari, P., Paterson, A., Zwaigenbaum, L., Roberts, W., Brian, J., Liu, X.-Q., Vincent, J., Skaug, J., Thompson, A., Senman, L., et al. (2007). Mapping autism risk loci using genetic linkage and chromosomal rearrangements. *Nature genetics*, 39(3):319. [Citado en pág. 50]

[Timofeev, 2004] Timofeev, R. (2004). Classification and regression trees (cart) theory and applications. *Humboldt University, Berlin*, pages 1–40. [Citado en págs. IX, 33, and 34]

[Van Rossum and Drake, 2009] Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA. [Citado en pág. 49]

[Wes McKinney, 2010] Wes McKinney (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61. [Citado en pág. 50]

Agradecimientos

Agradezco a mis tutoras, Maria del Mar y Ofelia Retamero, por ofrecerme este proyecto y ayudarme con sus consejos y explicaciones, que han sido esenciales durante todo el desarrollo del mismo.
También a mi familia, sobre todo a mis padres Mercedes Martín y Manuel Arroyo y en especial a mi hermana María Arroyo, que han sido un apoyo indispensable durante toda mi carrera universitaria.
A todos mis amigos y compañeros de clase, especialmente a Iñaki Melguizo, compañero de clase y amigo desde bachillerato.