_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSIDAD POLITECNICA DE VALENCIA

Reducción de dimensionalidad en Machine Learning.

Diagnóstico de cáncer de mama basado en datos genómicos y de imagen.

Javier Galarza Hernández

Tutora
Alicia Roca Martínez
Departamento de Matemática Aplicada

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2016-17

Valencia, 21 de Marzo de 2017

Agradecimientos

A mis padres, que siempre me han apoyado en cada decisión y han confiado siempre en mis posibilidades a pesar de que el viento soplase en contra.

A mi tutora, Alicia Roca, por haberse comprometido tanto conmigo y con este proyecto. Todo un ejemplo de profesional por vocación que se ha ganado mi total admiración.

Y por último, a mis amigos Álvaro, Patricia, María, Noelia, Carolina, Cristina, Lidia y Stephany por dar significado al concepto de amistad incondicional.

Resumen

El objetivo del proyecto es analizar algunas de las técnicas de aprendizaje automático (Machine Learning) que se emplean en la actualidad para extracción de información de grandes cantidades de datos, estudiar las herramientas estadísticas y algebraicas que estas técnicas emplean en los cálculos, y aplicarlas al diagnóstico y clasificación de tipos de cáncer de mama.

El manejo de grandes cantidades de datos requiere de un pre-procesamiento de los mismos para poder ser empleados. En este proyecto se presentan y analizan distintas herramientas utilizadas en el pre-procesado de datos y su impacto en los resultados finales. Se pretende con ello reducir la dimensionalidad de los datos, manteniendo la máxima cantidad posible de la información que los datos encierran.

Las técnicas introducidas se aplican a la detección y clasificación de cáncer de mama. Se dispone de dos bases de datos. Una de ellas se usa en la detección de la enfermedad y contiene indicadores extraídos de mamografías. La otra se emplea en la clasificación de tipos de cáncer de mama y consta de datos genómicos. Se efectúa un estudio de la información contenida en cada base de datos, se realiza un pre-procesado de dichos datos y se procede a la elaboración de los respectivos modelos.

En el caso de la detección, se plantean tres modelos iniciales para la detección de cáncer: uno basado en un regresor bayesiano, otro basado en una regresión logística y por último, otro utilizando *Support Vector Machines*. Posteriormente, se les aplica el análisis de componentes principales para reducir dimensionalidad y se comparan los resultados.

En la clasificación de tipos de cáncer de mama, se exploran los datos y se utilizan técnicas de *clustering* para observar cómo se agrupan las muestras. Esto permite la identificación de patrones y grupos de características similares.

Para concluir, se contrastan los resultados con otros obtenidos con otras técnicas utilizadas con el mismo fin para comprobar su robustez.

Resum

L'objectiu del projecte és analitzar algunes de les tècniques d'aprenentatge automàtic (Machine Learning) que s'empren en l'actualitat per a extracció d'informació de grans quantitats de dades, estudiar les eines estadístiques i algebraiques que aquestes tècniques empren en els càlculs, i aplicar-les al diagnòstic i classificació de tipus de càncer de mama.

El maneig de grans quantitats de dades requereix d'un pre-processament dels mateixos per poder ser emprats. En aquest projecte es presenten i analitzen diferents eines utilitzades en el pre-processat de dades i el seu impacte en els resultats finals. Es pretén amb això reduir la dimensionalitat de les dades, mantenint la màxima quantitat possible de la informació que les dades tanquen.

Les tècniques introduïdes s'apliquen a la detecció i classificació de càncer de mama. Es disposa de dues bases de dades. Una d'elles s'usa en la detecció de la malaltia i es constitueix d'indicadors extrets de mamografies. L'altra base de dades, s'empra en la classificació de tipus de càncer de mama i consta de dades genómiques. S'efectua un estudi de la informació continguda en cada base de dades, es realitza un pre-processat d'aquestes dades i es procedeix a l'elaboració dels respectius models.

En el cas de la detecció, es plantegen tres models inicials per a la detecció de càncer: un basat en un regresor bayesiano, un altre basat en una regressió logística i finalment, un altre utilitzant *Support Vector Machines*. Posteriorment, se'ls aplica l'anàlisi de components principals per reduir dimensionalitat i es comparen els resultats.

En la classificació de tipus de càncer de mama, s'exploren les dades i s'utilitzen tècniques de *Clustering* per observar com s'agrupen les mostres. Això permet la identificació de patrons i grups de característiques similars.

Per concloure, es contrasten els resultats amb altres obtinguts fent ús d'altres tècniques utilitzades amb la mateixa fi per comprovar la seva robustesa.

Abstract

The aim of the project is to analyze some of the machine learning techniques that are currently being used to extract information from large amounts of data, to study the statistical and algebraic tools that these techniques use in calculations, and to apply them to Diagnosis and Classification of breast cancer.

The handling of large amounts of data requires pre-processing of the same to be employed. This project presents and analyzes different tools used in the pre-processing of data and their impact on the final results. It is intended to reduce the dimensionality of the data, keeping the maximum possible amount of information that is contained within the data.

The techniques introduced are applied to the detection and classification of breast cancer. Two databases are used. One of them is used in the detection of the disease and contains indicators taken from mammograms. The other database is used in the classification of breast cancer types and consists of genomic data. A study of the information contained in each database is carried out, as well as a pre-processing of said Data and the elaboration of the respective models.

In the case of detection, there are three initial models for the detection of cancer: one based on a Bayesian regressor, another based in a logistic regression and finally, another using *Support Vector Machines*. Subsequently, the Analysis of main components to reduce dimensionality is applied and the results are compared.

In the classification of breast cancer, the data is explored and Clustering techniques are used to observe how clusters behave. This enables the identification of patterns and groups of similar characteristics within the samples.

To conclude, the results are contrasted against other techniques used for the same purpose to verify their robustness.

Contenidos

1. Introducción y conceptos básicos

Se diagnosticaron 14,1 millones de casos de cáncer en 2012, de los cuales 7,4 millones fueron pacientes hombres y 6,7 millones, mujeres. Se predice que esta cifra llegará a aumentar hasta 24 millones de casos diagnosticados para 2035 [1].

El cáncer de mama es de entre todos, el cáncer con mayor ratio de diagnóstico en etapa no terminal y la primera causa de mortalidad entre mujeres de todo el mundo [2]. En la actualidad, no existen métodos específicos de prevención del mismo, puesto que su origen aún permanece desconocido. Sin embargo, el diagnóstico en un estadio temprano de la enfermedad y la correcta clasificación del tipo de cáncer de mama otorga mayores probabilidades de recuperación, por lo que resulta vital para reducir la tasa de mortalidad actual y aplicar el mejor tratamiento posible. Con los últimos avances tecnológicos y la cantidad de información que se procesa por paciente, es posible hacer uso de técnicas de aprendizaje automático no sólo para agilizar y optimizar el diagnóstico de la enfermedad, sino también para descubrir patrones aún imperceptibles y poder así alcanzar una mayor comprensión de la enfermedad y de sus posibles causas. De esta manera, es posible conseguir una alta eficacia en el proceso y la mínima latencia a la hora de aplicar el tratamiento que proceda, a la vez que discernir posibles patrones que influyen en la evolución de la enfermedad. A continuación, en las figuras 1 y 2, se puede apreciar la incidencia del cáncer en general así como del cáncer de mama en particular.

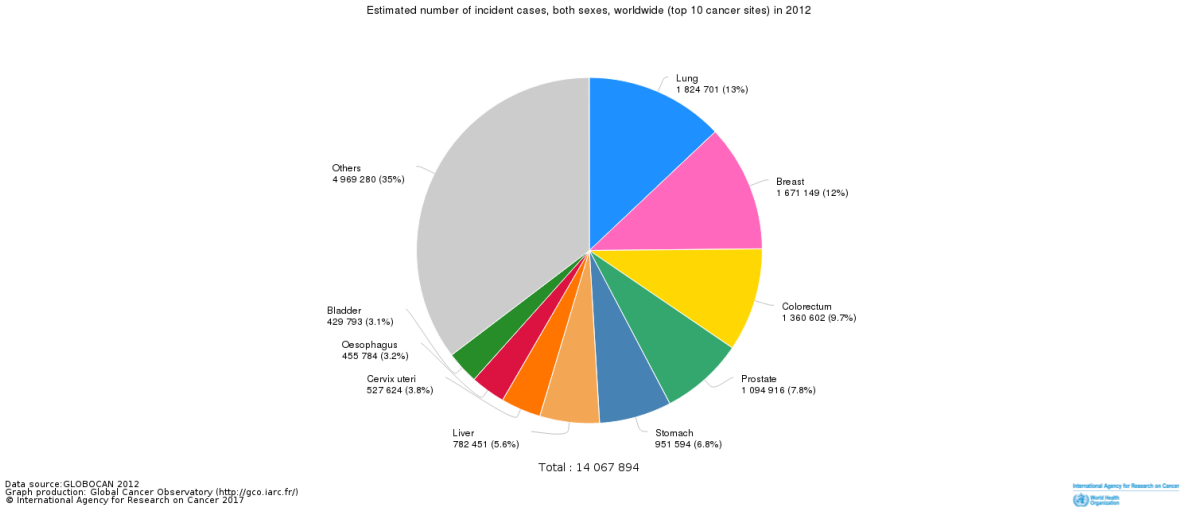

Figura 1: Porcentajes de incidencia por tipo de cáncer en ambos sexos.

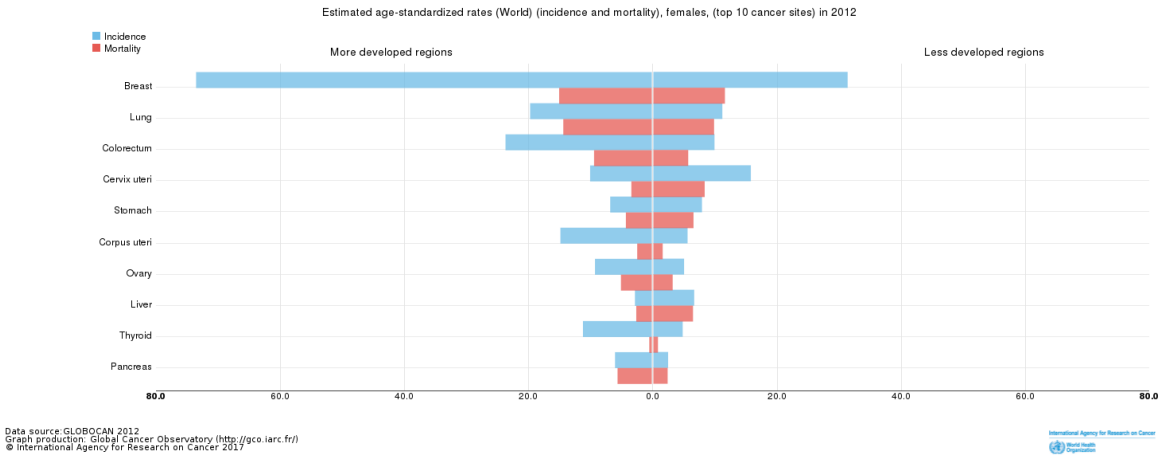

Figura 2: Tasas de incidencia y mortalidad por tipo de cáncer en mujeres.

El objetivo de este proyecto es presentar algunas de las técnicas de aprendizaje automático aplicado al diagnóstico y clasificación de tipos de cáncer de mama de una manera concisa y clara, profundizando en las herramientas estadísticas de reducción de dimensionalidad utilizadas en el preprocesado de datos y su impacto ante el modelo de predicción. Para ello, se crearán dos modelos de aprendizaje automático: Uno focalizado en la detección de la enfermedad utilizando indicadores de imagen y otro focalizado en la clasificación de subtipos y descubrimiento de patrones utilizando datos genómicos y proteómicos. Han sido elegidas específicamente estas dos bases de datos debido a la adecuación del proyecto, su complementariedad entre ellas, su disponibilidad y por motivación personal.

El presente documento se estructura en 6 capítulos. El primero de ellos, el presente capítulo, recorre brevemente los conceptos básicos y el contexto necesario para facilitar la comprensión de los conceptos en el ámbito del Machine Learning y del sistema de diagnóstico que se implementará, así como la justificación y explicación del entorno de trabajo escogido. En el segundo capítulo se aborda la metodología que se emplea de manera genérica en el campo del Machine Learning así como la explicación de algunas de las técnicas más utilizadas. En el tercer capítulo se trata de una manera matemática y más profunda los algoritmos de reducción de dimensionalidad más comunes y con mayor repercusión en los sistemas Machine Learning. En el cuarto capítulo se expone la implementación paso a paso y los resultados específicos de cada modelo propuesto y el impacto de la reducción de dimensionalidad sobre los mismos. Finalmente se recogen las conclusiones de todo el proyecto en el quinto capítulo y se comentan las posibilidades de mejora del sistema en el capítulo de Líneas futuras.

1.1. Introducción al Machine Learning

Se conoce como Aprendizaje Automático, o adoptando el anglicismo de ahora en adelante *Machine Learning*, como una sub-rama de la Inteligencia Artificial que permite a los computadores aprender de manera autónoma a partir de datos.

Esto es posible gracias a la implementación de diversos algoritmos, que con un conjunto de datos inicial, son capaces de determinar relaciones matemáticas entre los atributos y las instancias. En Machine Learning se hace referencia a atributos a aquellas características contempladas para cada instancia, y se define instancia como una muestra que contiene los atributos especificados. Con las relaciones matemáticas que Machine Learning es capaz de computar y almacenar, es posible establecer relaciones y patrones entre los datos y el atributo sobre el que se desea realizar la predicción. De esta manera, el algoritmo una vez procesa la información inicial, es capaz de realizar predicciones sobre otros datos aún no contemplados siguiendo dichas relaciones y patrones. Todo ello, concebible debido a que prácticamente todo conjunto de datos contiene patrones, y son estos patrones lo que permite a la máquina generalizar entrenando un modelo que determina los aspectos más relevantes encontrados entre los datos que posee. Es decir, se espera que a partir de los datos iniciales introducidos al modelo, se reflejen suficientemente bien las características de todo el espacio muestral del problema. Así, Machine Learning es capaz de realizar esa generalización sobre instancias no contempladas aún, y proveer un resultado para el atributo resultado [3].

Hay una gran variedad de algoritmos utilizados en Machine Learning, algunos con gran popularidad como Redes Neuronales, Árboles de Decisión, Regresiones Lineales o algoritmos de K-vecinos próximos. No obstante, no hay ningún modelo predefinido y validado para el funcionamiento eficaz sobre cualquier conjunto de datos. Dependiendo de la naturaleza propia de los datos y de la variable resultado a predecir, deberá de escogerse uno o varios algoritmos para la creación de un modelo y entonces proceder con su posterior validación para asegurar un funcionamiento óptimo.

El propio término Machine Learning se empezó a utilizar entre 1950 y 1952, cuando Arthur L. Samuel, trabajador informático en IBM por aquel entonces, desarrolló el primer programa de Machine Learning capaz de jugar a las damas utilizando movimientos propios de un maestro jugador experimentado. Fue en 1952 cuando su modelo Machine Learning llegó a vencer al entonces cuarto mejor jugador de Damas de la nación y por ende, cuando Arthur se ganó el título de pionero en el campo de la Inteligencia Artificial y Padre del Machine Learning. Fue entonces, cuando se abrió toda una nueva infinidad de posibilidades de aplicación más allá del propio juego de mesa, que por aquel entonces aún no visionaban. Desde entonces el Machine Learning ha permanecido humildemente en un segundo plano hasta llegar a la actualidad, donde el término no deja de escucharse en casi cualquier ámbito: Desde Google con su modelo de procesado inteligente de fotos de *Google Photos* basado en redes neuronales convolucionales [4], Apple con el procesado de Lenguaje Natural de *Siri* o Netflix y su sistema de recomendación [5] hasta descubrimiento de nuevos fármacos [6], diagnóstico de enfermedades [7] o Predicción de valores en Bolsa. Este auge exponencial actual del Machine Learning está directamente relacionado con la evolución tecnológica que ha visto el mundo en este último quincenio y de nuestro comportamiento con esta tecnología. Se estima que se envían 2.9 millones de correos electrónicos por segundo, Google procesa alrededor de 24 Petabytes de datos al día, Facebook registra un total de 700 billones de minutos gastados en su red social al mes y así prosigue un largo etcétera de hechos que respaldan la masiva generación de datos que está tomando lugar actualmente y que hasta ahora no han sido de interés ni explotados. Todos estos hechos son indicios de una nueva revolución industrial que podría ser considerada como la cuarta revolución industrial [8][9], donde el modelo de negocio e investigación orbitará entorno a sistemas inteligentes, Internet of Things y Cloud Computing, siendo estos conceptos o bien herramientas o sistemas que podrían beneficiarse del Machine Learning. Así pues, se presagia un camino fácil para desatar el potencial que Machine Learning es capaz de proveer en un sinfín de aplicaciones en los años venideros.

En las próximas secciones se recorrerá de manera introductoria los diferentes tipos de Machine Learning y su correspondiente ámbito de aplicación.

1.1.1. Supervised Learning

Se denomina Aprendizaje supervisado, o adoptando de ahora en adelante el anglicismo *Supervised Learning*, a aquellas aplicaciones designadas al Machine Learning que hacen uso de una base de datos "supervisada"[10]. Esto quiere decir que se conoce previamente el resultado de los datos introducidos al modelo de manera inicial (denominado *training set*). De manera generalista se podría expresar

$$y = f(x), \tag{1}$$

donde y es nuestra variable resultado y x el conjunto de información que introducimos al modelo. En el *training set* tanto x como y son conocidas y deseamos que el modelo implementado de Machine Learning aprenda la función aplicada para que cuando se introduzca un nuevo conjunto de datos, se obtenga el resultado que corresponda.

Supervised learning se aplica a dos tareas:

1. **Clasificación:** En ella se pretende que el atributo estudiado de cada muestra sea clasificado en un grupo de entre todos los posibles. Un ejemplo de clasificación podría ser la distinción entre dos razas de perros diferentes donde por ciertos atributos fuese posible distinguirlas. Una representación gráfica de ello podría representarse con un modelo como el indicado en la figura 3, donde se pueden apreciar dos planos correspondientes a un modelo utilizado con distancia euclídea (azul) y otro siguiendo un modelo de Fisher (rojo) que distinguen entre ambas razas naranja y azul. El plano que sigue el modelo de Fisher es calculado por distancia entre medias, mientras que el plano euclídeo tal y como indica su nombre, con distancia euclídea.

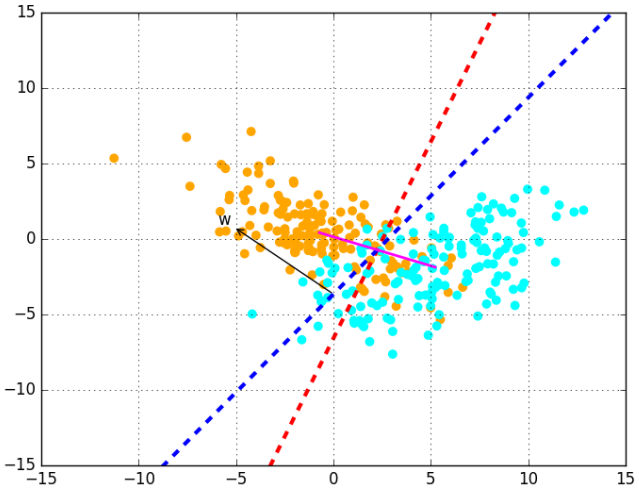

Figura 3: Ejemplo de discriminación lineal.

2. **Regresión:** En la regresión la variable resultado buscada es un valor específico. Un ejemplo de regresión podría ser la predicción de la función $y = sin(2\pi x)$ dados 8 puntos discretos como puede apreciarse en la figura 4. Para este ejemplo, se ha utilizado una regresión polinomial que se puede apreciar en morado representada en las figuras. Es fácilmente observable el error cometido para cada grado del polinomio por el área resultante entre la función predicha y la originial.

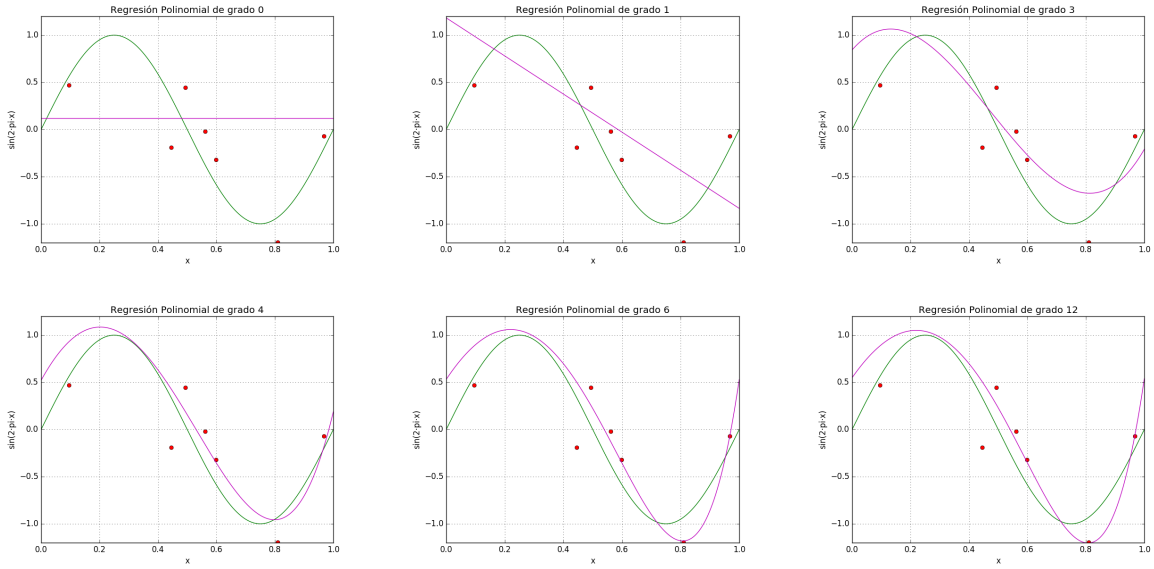

Figura 4: Ejemplo de regresión polinomial.

En este caso se puede apreciar que la mejor aproximación a la función es el caso de la regresión polinomial de cuarto grado, puesto que a partir de éste, se va aumentando el error. Este problema es conocido como *overfitting* o sobreescalado. El *overfitting* impacta negativamente en el resultado debido a que toma como norma general resultados anómalos del conjunto de datos específico con el que se está trabajando, lo cual a la hora de generalizar desestabiliza la norma del modelo e induce un error adicional.

1.1.2. Unsupervised Learning

Se denomina Aprendizaje no supervisado, o adoptando de ahora en adelante el anglicismo *Unsupervised Learning*, a aquellas técnicas de Machine Learning que hacen uso de una base de datos sobre la que no hay variable resultado específica, sino que persiguen explorar y descubrir características o subgrupos dentro de la misma base de datos.

Unsupervised learning es aplicado en una amplia gama de tareas. No obstante las más comunes son dos:

1. **Clustering**: Se desea buscar grupos entre los datos introducidos, como por ejemplo para el descubrimiento de cepas de virus similares. En la figura 5, a la izquierda se muestra un conjunto de virus de la misma clase pero con diversidad de mutaciones. Se desea agruparlos para poder

identificar las diferentes cepas y poder estudiar la naturaleza de las mutaciones. A la derecha de la figura, se observa el resultado de clustering donde se han agrupado en 12 clústers mediante búsqueda de centroides. Se aprecian los diferentes clústers diferenciados por los colores y sus respectivos centros marcados con un punto.

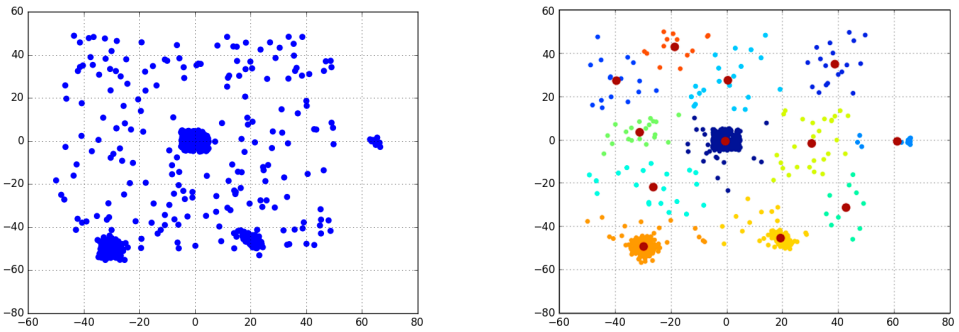

Figura 5: Ejemplo de clustering.

2. **Asociaciones**: Se pretende buscar relaciones significativas entre los datos introducidos, como por ejemplo descubrir relaciones entre patologías para el diagnóstico de diabetes como se muestra en la figura 6. Resulta muy útil para extraer conocimiento y entendimiento de patrones. En dicha figura se puede observar que tenemos dos grandes grupos correspondientes al diagnóstico positivo y negativo de diabetes, y de los indicadores que más influyen en dicho diagnóstico. Hay un tercer grupo más pequeño no relacionado con ningún otro grupo debido a que son valores anómalos o erróneos.

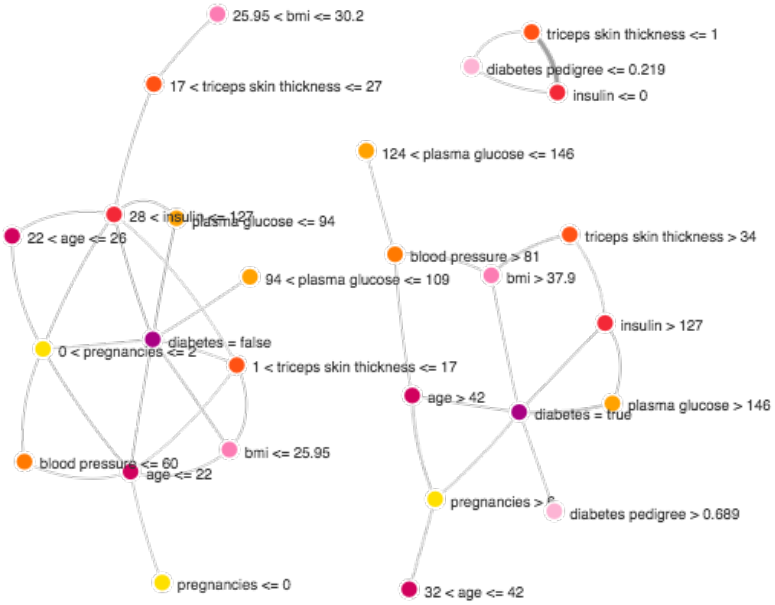

Figura 6: Ejemplo de asociación de indicadores en el diagnóstico de diabetes.

1.1.3. Reinforcement Learning

En algunas aplicaciones, la salida del sistema es una secuencia de acciones. En dichos casos, una sola acción no es relevante sino el conjunto de las acciones tomadas de manera certera para conseguir el objetivo establecido. De esta manera, los sistemas basados en Reinforcement Learning son capaces de valorar qué acciones son certeras y cuales no basándose en experiencias previas [11].

Un ejemplo claro de estos sistemas se da en los videojuegos multiplataforma, donde una acción no es importante, sino el conjunto de ellas para llegar al final de la pantalla. Así pues, el sistema inteligente de estos videojuegos, a base de prueba y error, puede registrar la combinación tiempo-acción óptima para llegar a superar el nivel, contemplando todas las posibles combinaciones y almacenando aquella que mejor ha resultado.

1.2. Introducción al diagnóstico de cáncer de mama

Machine Learning se está empleando en un amplio abanico de campos. No obstante, está tomando mucha fuerza en el ámbito biomédico debido al potencial de *Knowledge Discovery* o descubrimiento de conocimiento y los sistemas de ayuda al diagnóstico o *Computer-aided diagnosis systems*. Existen muchas patologías y patrones de procesos biológicos que a pesar de llevar décadas de estudio e investigación [12], siguen resistiéndose al entendimiento humano y por ende, son situaciones de diagnóstico y/o tratamiento meramente subjetivo a criterio del facultativo. Tal es el caso, por ejemplo, del diagnóstico de Esclerosis múltiple o de la Fibromialgia, o el descubrimiento de tratamientos efectivos de éstas y más enfermedades.

En el caso del cáncer de mama, el diagnóstico resulta costoso en términos temporales no por la subjetividad del diagnóstico en sí, sino por el proceso necesario para determinar con certeza el diagnóstico. Para determinar un diagnóstico de cáncer de mama se tardan entre dos y tres semanas, dependiendo de las políticas burocráticas hospitalarias que apliquen. Y además, en muchas ocasiones el cáncer de mama es diagnosticado después de la manifestación de ciertas patologías características, pero otras tantas veces, no. Es por esto, por lo que la única medida preventiva es la revisión médica periódica que en ocasiones puede dilatarse en el tiempo retrasando el comienzo y la elección del tratamiento. Y es en este aspecto en el que el Machine Learning podría contribuir a mejorar el proceso.

El diagnóstico de cáncer de mama [13] comienza con una revisión de la historia clínica y una exploración física. A continuación, se procede con las pruebas diagnósticas entre las que se encuentran:

- Análisis de sangre y orina

  Además de los indicadores más comunes, se incide en la realización de un hemograma para ver el resultado de las células de la sangre, bioquímica renal y hepática para conocer la función de los riñones e hígado y determinación de iones, como el calcio. También puede determinarse la búsqueda de marcadores tumorales, que para el cáncer de mama son el antígeno cárcinoembrionario (CEA) y el CA 15-3.

- Mamografías

  La mamografía es la técnica de exploración más eficaz hasta la fecha para la detección en estadios tempranos de la enfermedad. Consiste en la realización de una radiografía a muy baja

dosis de radiación para evitar efectos nocivos. En caso de detectarse una imagen sospechosa, el facultativo solicitará más pruebas de diagnóstico de imagen para conocer la naturaleza del cuerpo extraño. Dichas técnicas adicionales pueden ser tales como una ecografía o una resonancia magnética.

En una mamografía se pueden detectar una serie de signos que indican si el cuerpo extraño es benigno o maligno. Tales indicadores son:

- Calcificaciones: Son pequeños depósitos minerales que son detectadas como pequeñas manchas blancas en las mamografías. Las calcificaciones pueden ser indicativo de ser un tumor benigno, y a menudo, maligno.
- Masas: Pueden tratarse de lesiones benignas como es el caso del fibroma, o malignas. Pueden contener calcificaciones o no.
- Quistes: Se trata de cuerpos huecos rellenos de un líquido. En ocasiones, se requiere extraer una muestra de líquido para que sea analizado histológicamente. Con baja frecuencia se diagnostican quistes como malignos.

En el caso de la aparición de masas en la mamografía, las pruebas complementarias más comunes son la ecografía, la punción aspiración con aguja fina (PAAF), la biopsia con aguja gruesa (BAG) (puede ser por esterotaxia) o la bioposia quirúrgica (menos frecuente en la actualidad). Las microcalcificaciones se valoran mediante biopsia asistida por vacío (BAV). En el caso de mamas densas también se realiza una resonancia nuclear magnética (RNM).

Una vez el diagnóstico ha sido ratificado por el facultativo como maligno, comienzan los tratamientos. El tratamiento del cáncer de mama, como en muchas otras áreas de la medicina, es una labor multidisciplinar en la que contribuyen facultativos de diversos ámbitos. El primer paso es establecer un plan terapéutico en función de:

- Edad del paciente
- Estado general
- Estado hormonal
- Localización del tumor
- Estadio de la enfermedad
- Receptores hormonales
- Grado de células
- Positividad de indicadores biológicos

Los tratamientos más frecuentes en el cáncer de mama son: Cirugía, radioterapia, quimioterapia y hormonoterapia. No obstante, no existe método efectivo con alta probabilidad de éxito de manera general, por lo que en función de los factores comentados anteriormente y de la experiencia de los facultativos, se recomendará un tratamiento u otro [14].

En la actualidad se comienzan a utilizar indicadores genéticos y su correlación con la fuerte presencia o ausencia de ciertas proteínas, para no sólo el diagnóstico, sino también la clasificación

de los diferentes tipos de cáncer de mama. Hay diversas metodologías y estudios genéticos que proveen de esta información. No obstante, uno de los más innovadores por su dimensionalidad y aparentemente efectivos es el denominado PAM50. Este método hace uso de la expresión genética de 50 genes para clasificar la enfermedad en cuatro posibles tipos: *Basal-like*, *HER2-enriched*, *Luminal A* y *Luminal B*. Conocer el subtipo de cáncer facilita la labor de encontrar el tratamiento más adecuado para la paciente y así poder proveer una mayor probabilidad de no reincidencia del tumor [15].

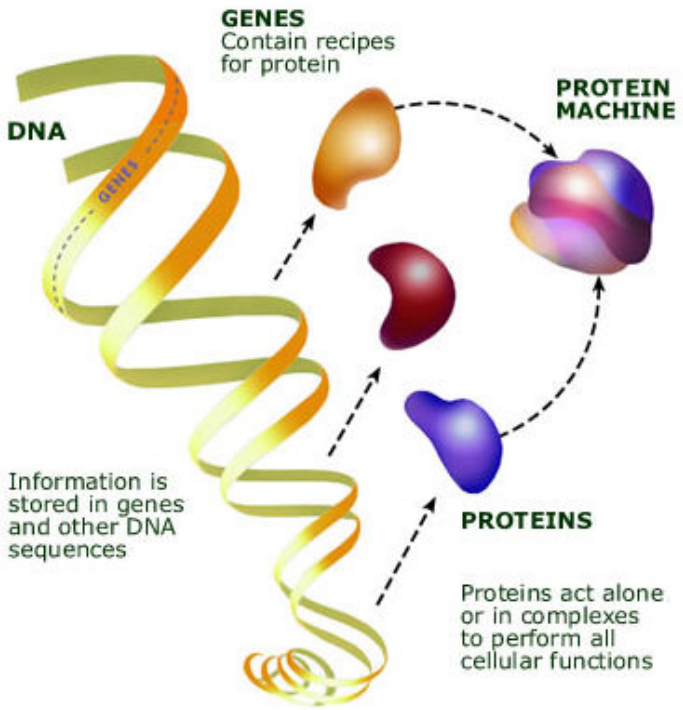

Figura 7: Relación entre ADN, gen y proteína.

El método PAM50, hace un estudio sobre la presencia de multitud de proteínas que provienen de los 50 genes que conforman el PAM50 (véase figura 7 para apreciar la relación entre gen y proteína), y en función de éstas, es posible diferenciar entre los cuatro tipos de cáncer mencionados anteriormente.

1.3. Entorno de trabajo

En este apartado se recorrerán las herramientas y software que han sido utilizados en el proyecto para llevarlo acabo.

El proyecto es esencialmente Software que se nutre de un conjunto de bases de datos. Se ha utilizado PyCharm como IDE de programación y adicionalmente Jupyter Notebooks para depuración de código y visualización de gráficos.

1.3.1. Python

El lenguaje de programación utilizado para la realización del software es Python.

Python es un lenguaje de programación orientado a objetos, interactivo e interpretado con sintaxis clara, sencilla e intuitiva [16]. Las características más remarcables de Python podrían resumirse a continuación

- **Es conciso**

  La simplicidad y sencillez de sintaxis permite desarrollar código de alta complejidad en pocas líneas. Consecuentemente, el código escrito en Python tiende a ser más corto que en otros lenguajes de programación.

- **Es legible**

  La sintaxis que utiliza es una sintaxis intuitiva que facilita tanto a los programadores experimentados como a los recién iniciados, el leer y entender código Python sin demasiadas complicaciones.

- **Es fácilmente extensible**

  Python contiene por defecto un conjunto básico aunque bastante amplio de librerías que cubre un gran rango de aplicaciones: Funciones Matemáticas, XML, Descargas de páginas Web etc. No obstante, también cuenta con un gran número de librerías creadas por la propia comunidad Python de las que cualquier usuario se puede nutrir.

- **Es interactivo**

  Mientras se trabaja en cualquier aplicación, resulta muy útil poder ir probando el código a medida que se va avanzando en el mismo. Python permite ejecutar código directamente mediante una terminal interactiva que permite definir funciones, llamar objetos o probar librerías de manera interactiva.

- **Es multiparadigma**

  Python es orientado a objetos con distintos estilos funcionales de programación. Los algoritmos de Machine Learning varían ampliamente y por ende el paradigma de implementación. En ocasiones es mejor introducir funciones como parámetros de entrada y en otras, capturas de estado de una variable. Python soporta ambas implementaciones.

- **Es multiplataforma y open-Source**

  Python es susceptible de ser implementado en la mayoría de plataformas, y a coste cero en todas ellas. Quizás el mayor punto fuerte de Python sea la comunidad que hay detrás del lenguaje. Al ser Open-Source y teniendo una comunidad tan activa, hay miles de librerías especializadas para casi cualquier tipo de aplicación imaginable.

Por todo ello y en concreto por las librerías tan desarrolladas en el ámbito del Machine Learning que posee, se ha escogido Python como lenguaje de programación para este proyecto.

La librería de Python de Machine Learning utilizada es la libreria *Scikit-Learn*. Esta librería ha sido desarrollada por Google, actualmente *Open-Source* y una de las más potentes en el ámbito del

Machine Learning. Provee funciones de Regresión, Clasificación, Clustering, Reducción de dimensionalidad, Selecciones de modelo y Pre-procesado. Para visualización se han utilizado las librerías de Bokeh y Seaborn.

1.3.2. Fuente de Datos

Para el desarrollo del motor de predicción que se detalla más adelante en este documento, se han utilizado dos Bases de Datos diferentes sobre Cáncer de Mama: *Breast Cancer Wisconsin Dataset* y *Breast Cancer Proteomes Dataset*.

- **Breast Cancer Wisconsin Dataset**

  Se trata de una base de datos cedida por los Departamentos de informática y cirugía general de la Universidad de Wisconsin-Madison, EEUU. Esta base de datos contiene un total de 569 muestras o filas y 32 atributos o columnas de pacientes reales. Los datos aquí recogidos proporcionan información sobre las características del núcleo celular mamario analizado, obtenido gracias al análisis previo computacional de una imagen tomada de la masa mamaria.

- **Breast Cancer Proteomes Dataset**

  Se trata de una base de datos cedida por el *Clinical Proteomic Tumor Analysis Consortium (CPTAC)*, perteneciente al Instituto Nacional contra el Cáncer de EEUU. El CPTAC es un consorcio estadounidense establecido para acelerar la comprensión de la naturaleza molecular del cáncer mediante la aplicación de tecnologías proteómicas y procesos robustos y cuantitativos.

  Esta base de datos contiene un total de 77 muestras o filas y 12553 atributos o columnas de pacientes reales. Los datos aquí recogidos proporcionan información sobre la cantidad presente de cada una de las proteínas en las muestras recogidas mediante una espectrometría de masa. Adicionalmente, existe una segunda base de datos denominada PAM50. Esta base de datos contiene un listado de las proteínas que conforman los 58 genes en los que se basa este estudio genético denominado PAM50. Estudios recientes indican que gracias a este método de estudio genético se puede extraer mayor información sobre la posible recurrencia y evolución del cáncer que con otros métodos.

Se utilizan estas dos bases de datos para poder cubrir un mayor espectro de Machine Learning, pudiendo así realizar dos motores predictivos: Uno enfocado al diagnóstico basado en *Supervised Machine Learning* para el que utilizaremos el *Breast Cancer Wisconsin Dataset* y el otro, enfocado al descubrimiento de patrones y clustering basado en *Unsupervised Machine Learning* para el que utilizaremos el *Breast Cancer Proteomes Dataset*.

De esta manera tendremos la información suficiente para poder dar un diagnóstico fiable basado en indicadores de imagen de células mamarias a la vez que poder realizar descubrimiento de patrones genéticos.

2. Tratamiento de datos y elaboración del modelo

La elaboración de un modelo Machine Learning consta de las fases que se resumen gráficamente en la figura 8, donde se puede observar que toda la fase de implementación se apoya en una base de datos ya conocida para entrenar el modelo y conseguir una robustez suficiente como para poder realizar predicciones de nuevas instancias de datos de manera fiable.

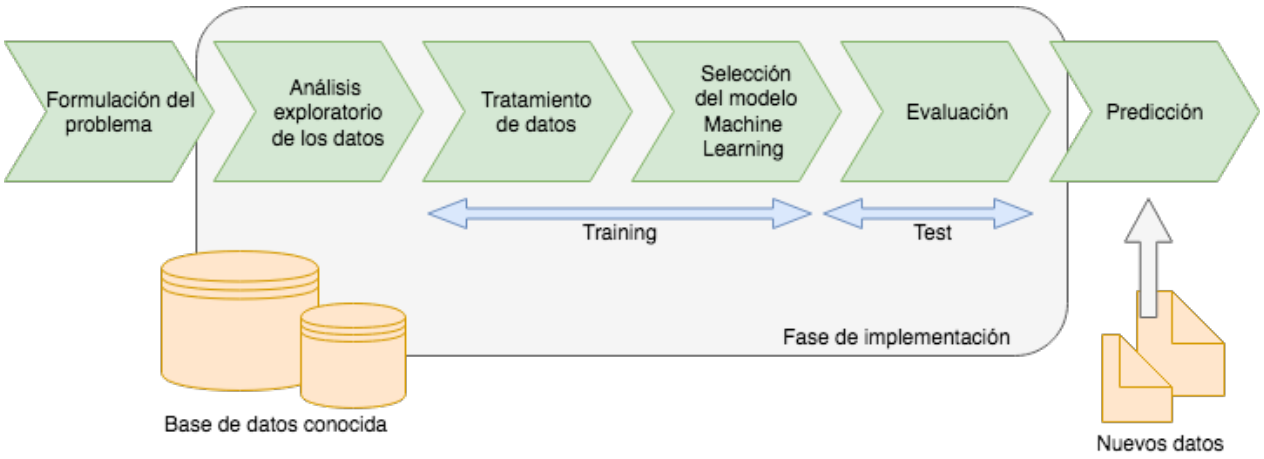

Figura 8: Decálogo para la elaboración de un modelo Machine Learning.

La mayor parte de la complejidad e impacto en el resultado final existente, reside en el tratamiento de datos previo al entrenamiento del modelo y la elección del algoritmo en el que se basará este mismo modelo. No obstante, la elección del algoritmo y la implementación práctica del software Machine Learning está optimizada gracias a los avances en eficiencia computacional y a la comunidad Machine Learning que crea y mantiene las librerías de desarrollo. Consecuentemente, donde realmente reside el mayor impacto y donde hay menor automatización, es en el tratamiento previo de los datos a introducir [17]. Por el momento, no existe ningún estándar ni ningún método de tratamiento que sea altamente eficiente para cualquier aplicación. Queda por tanto esta tarea para la interpretación del analista, de su experiencia y de su conocimiento de las bases de datos en las que esté trabajando.

A continuación, se recorrerán los pasos que conforman el decálogo para la elaboración óptima de un modelo Machine Learning:

1. Análisis exploratorio.
2. Tratamiento de datos.
3. Selección del modelo Machine Learning
4. Evaluación.

2.1. Análisis Exploratorio

El primer paso hacia la elaboración del modelo consiste en un análisis exploratorio de los datos disponibles. El objetivo de este análisis es obtener una visión general de los datos con los que se va

a trabajar y obtener una idea de lo que va a poder explotarse a la vez que identificar posibles datos inadecuados, faltantes, erróneos, irrelevantes, dispersos, etc. En ocasiones, la propia base de datos no está organizada de manera funcional para trabajar en Machine Learning: los casos o instancias situados por filas y los atributos o indicadores, en columnas. Por lo que en este primer estadio, debería ajustarse como tal si aplicase.

En este análisis inicial, es muy útil hacer uso de técnicas estadísticas y utilizar herramientas de visualización sobre la base de datos para ser conocedores de las características que se van a manejar. Así, se podrá establecer a posteriori un modelo que funcione eficazmente con la naturaleza de los datos de los que se disponen.

Python dispone de un método denominado *describe()* que ofrece un resumen estadístico sobre las variables contempladas en la base de datos, dando como resultado la media, desviación típica, cuartiles, valor mínimo, valor máximo y el total de instancias por cada atributo de la base de datos como se puede apreciar en la figura 9. Adicionalmente, si se desease algún otro tipo de indicador estadístico es fácilmente operable con la librería *Numpy*.

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400

Figura 9: Ejemplo del método *describe()* de Python.

Las librerías de visualización de datos ayudan mucho en esta tarea, ofrecen gráficos llamativos, sencillos de implementar y que facilitan el entendimiento completo de los datos de una manera intuitiva. Hay muchas librerías de Python para esto, sin embargo cabe destacar *Bokeh* y *Seaborn* por disponer de una amplia gama de gráficos y por su sencillez.

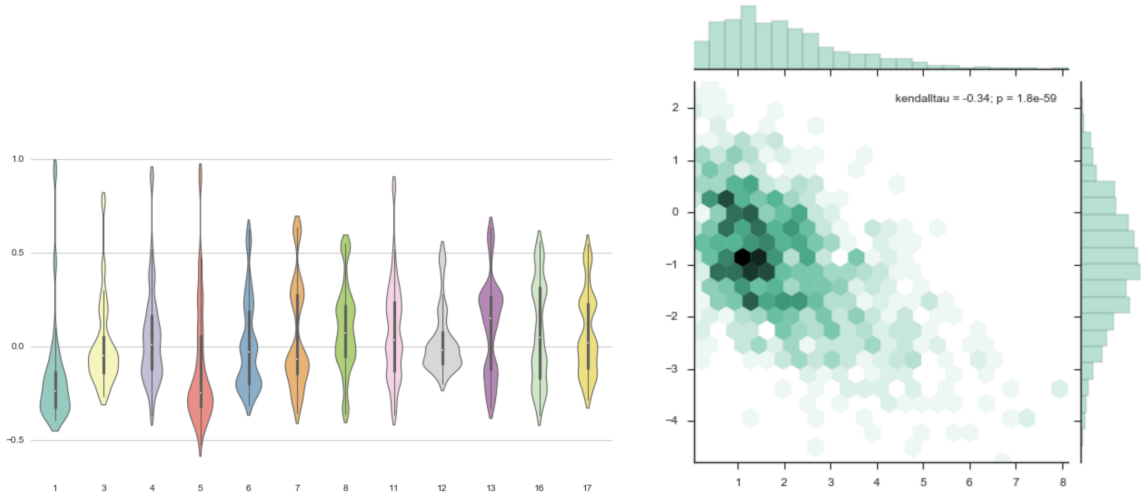

Figura 10: Ejemplo de gráficos Seaborn.

En la figura 10, se puede observar un par de ejemplos de gráficos *Seaborn*. En la figura de la izquierda aparece un gráfico en el que se representan la densidad de los diferentes atributos de una base de datos aleatoria. En el gráfico de la derecha, se representa la frecuencia de uno de los atributos con histogramas marginales.

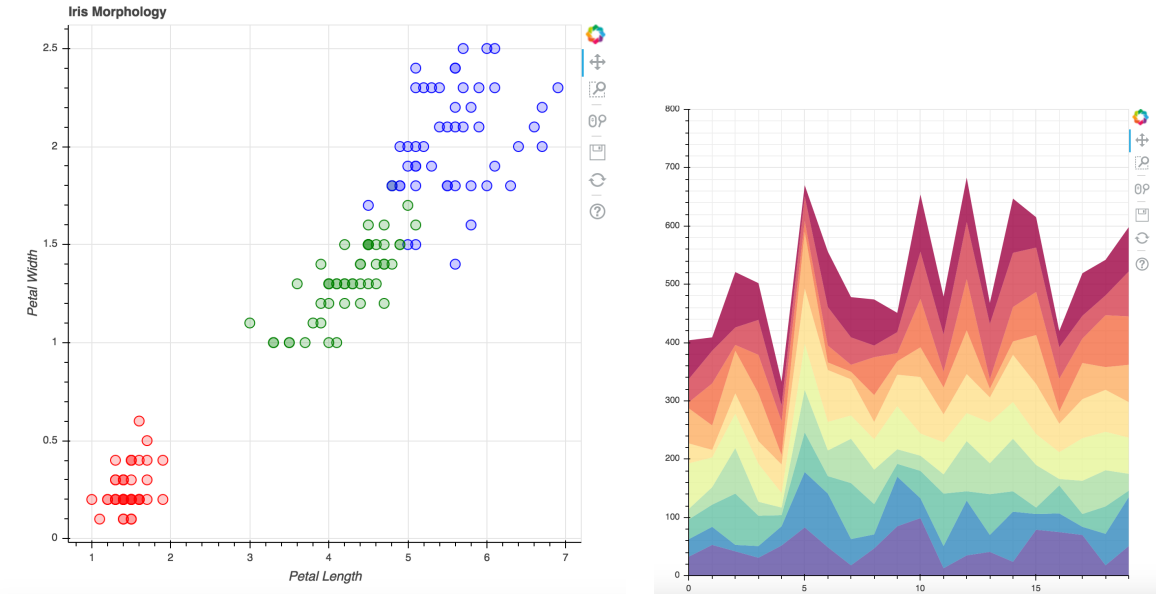

Figura 11: Ejemplo de gráficos Bokeh.

En la figura 11, se puede observar un par de ejemplos gráficos *Bokeh*. En el gráfico de la izquierda aparece representado un *scatterplot* (gráfico de dispersión) mientras que en el gráfico de la derecha, se representa el espectro de diferentes atributos.

2.2. Pre-procesamiento

Todo análisis estadístico consta de tres partes fundamentales que son modelado, predicción e inferencia, asumiendo que los datos son correctos, completos y en perfecto estado para el análisis. En la práctica, especialmente en el ámbito del Big Data, los datos crudos o *raw data* necesitan un proceso previo hasta que pueden considerarse técnicamente correctos. Mediante el pre-procesamiento se obtienen datos sin errores, perfectamente codificados y consistentes. Los resultados del pre-procesamiento afectan significativamente a los resultados de la predicción, y como se ha mencionado anteriormente, es la parte menos automatizada de la implementación [18].

Las cuatro fases del pre-procesamiento son:

1. Limpieza
2. Integración
3. Transformación
4. Reducción

2.2.1. Limpieza

El objetivo es obtener datos consistentes en los que los valores perdidos, especiales, anómalos y las inconsistencias (obvias) se han detectado, localizado y eliminado, corregido o imputado.

Básicamente comprenden los procedimientos de detección y tratamiento de:

- **Valores especiales**
  - Dato ausente: Las operaciones básicas funcionan bien con éstos. No se debe confundir nunca dato ausente con "0.º con "No aplica".
  - Inf: Infinito. Técnicamente es un número válido, resultado de operaciones como la división por "0".
  - NaN (Not a Number): No numérico. Dato que existe pero cuyo valor no es conocido, no por omisión, sino porque es el resultado de algunas operaciones que dan como resultado una indeterminación.

  Existen funciones en Python para detectar estos valores y poder realizar coerciones explícitas como por ejemplo: *isnumeric(), isnan(), fillna(), astype()*.

- **Valores perdidos**

  Son valores que no constan debido a causas tales como errores en la transcripción de los datos o falta de respuesta en algunos items de una encuesta. También pueden ser valores especiales. Pueden faltar de manera aleatoria o no aleatoria:

  - Los **datos perdidos aleatorios** pueden perturbar el análisis de datos dado que disminuyen el tamaño de las muestras y en consecuencia la potencia de las pruebas de contraste de hipótesis.

- Los **datos perdidos no aleatorios** ocasionan, además, disminución de la representatividad de la muestra.

En estos casos, debe ser el analista quien decida qué hacer con estos valores: o bien eliminarlos o bien asignarlos de algún modo.

Los métodos de imputación consisten en estimar los valores ausentes en base a los valores válidos de otras variables y/o casos de la muestra. La estimación se puede hacer a partir de la información del conjunto completo de variables o bien de algunas variables especialmente seleccionadas. Los principales procedimientos son:

- **Sustitución por la media u otro parámetro**
  Consiste en sustituir el valor faltante por la media de los valores válidos,

$$\hat{x_i} = \overline{x_i} \tag{2}$$

  aunque este procedimiento plantea inconvenientes: dificulta la estimación de la varianza, distorsiona la verdadera distribución de la variable y la correlación entre variables ya que añade valores constantes.
  Una variante en datos muy asimétricos es utilizar la mediana en lugar de la media,

$$\hat{x_i} = Me_i \tag{3}$$

- **Sustitución por constante**
  Consiste en sustituir los valores ausentes por constantes cuyo valor viene determinado por razones teóricas o relacionadas con la investigación previa. Presenta los mismos inconvenientes que la sustitución por la media, y solo debe ser utilizado si hay razones para suponer que es más adecuado que el método de la media. No obstante, puede ser utilizado en la imputación sobre variables cualitativas.

- **Valores anómalos o atípicos**

Son observaciones cuyos valores son muy diferentes a otras observaciones del mismo grupo de datos. Los datos atípicos son ocasionados por:

- Errores de procedimiento en la recogida de datos
- Acontecimientos extraordinarios
- Valores extremos
- Causas no conocidas

Los datos anómalos distorsionan los resultados de los análisis y afectan así pues a las predicciones realizadas por Machine Learning. Los valores anómalos no son necesariamente errores, por lo que han de ser detectados pero no eliminados sistemáticamente. La inclusión o no en el modelo es una decisión del analista. Son fácilmente detectables con el diagrama de caja y bigotes.

2.2.2. Integración

Se trabaja con los datos disponibles seleccionando subconjuntos o filtrando, ordenándolos o haciendo combinación o agregación de datos.

En muchas ocasiones no se dispone de una única base de datos, por lo que la integración entre ambas resulta fundamental para proveer de resultados coherentes en Machine Learning. Se trata de realizar operaciones básicas de selección y ajustes para poder crear un entorno de datos consistente y coherente en función de la variable resultado buscada.

2.2.3. Transformación

Para aplicar la técnica adecuada de Machine Learning, se necesita que las variables tengan el tipo y la distribución de frecuencias correctas según el caso. Esto nos puede llevar a necesitar transformar la variable para que sea manejable.

Las transformaciones más comunes son:

- **Normalización de variables cuantitativas**

  La normalización, estandarización o tipificación de una variable consiste en la transformación lineal que se declara a continuación en (4),

  $$Z = \frac{x - \mu}{\sigma} = \frac{1}{\sigma}x - \frac{\mu}{\sigma}. \tag{4}$$

  La variable tipificada expresa el número de desviaciones típicas que cada observación dista de la media. Por ello, se puede comparar la posición relativa de los datos de diferentes distribuciones, siendo posible comparar variables medidas sobre diferentes escalas o magnitudes.

- **Transformaciones exponencial y logarítmica**

  Son transformaciones no lineales que se usan generalmente para corregir la asimetría y curtosis de la distribución de frecuencias de una variable. Si se tienen distribuciones de frecuencias con asimetría negativa, es conveniente aplicar una transformación que comprima la escala para valores pequeños y la expanda para valores altos, como

  $$y = x^2. \tag{5}$$

  Para distribuciones con asimetría positiva, se usan las transformaciones que compriman los valores altos y expandan los pequeños como

  $$y = \sqrt{x} \tag{6}$$

  $$y = ln(x) \tag{7}$$

  $$y = \frac{1}{x}, \tag{8}$$

  ecuaciones ordenadas por efecto de expansión creciente. La transformación más utilizada es la del logaritmo. Muchas distribuciones de datos económicos o de consumos se convierten en simétricas al tomar la transformación del logaritmo.

- **Recodificación de variables cualitativas en cuantitavas**

  En ocasiones, algunos atributo contemplan valores categóricos traducidos a una codificación de carácteres como por ejemplo "S"para determinar que sí tiene dicho atributo o "N"para determinar que no tiene dicho atributo. Para este tipo de variables es necesario que sean

recodificados a un ámbito numérico para que pueda ser procesado por Machine Learning. El criterio de la recodificación de estas variables queda bajo criterio del analista, aunque en ocasiones está bastante estandarizado, como puede ser el caso Sí/No que se traduce en 0/1 (conocido como variables binarias).

2.2.4. Reducción

Cuando se dispone de una base de datos con gran cantidad de atributos e instancias, resulta necesario reducir dichos datos a un número menor de variables o de casos perdiendo la menor cantidad de información posible. De lo contrario, el tiempo de computación puede ser demasiado elevado además de poder estar induciendo rudio al sistema.

La técnica estadística en estas situaciones más habitual es el análisis de componentes principales.

El objetivo del análisis de componentes principales consiste en condensar la información contenida en x variables originales en un número reducido de nuevas variables incorrelacionadas llamadas componentes siendo éstas menos que el número original de variables x. Se verá más en profundidad en el apartado 3.2.

2.3. Elaboración y aplicación del modelo

Una vez ya se ha realizado todo el pre-procesamiento necesario de los datos, se ha de comenzar a construir el modelo Machine Learning. Como ya se ha mencionado en apartados anteriores, el ámbito del Machine Learning resulta ser un ámbito bastante abierto donde un modelo puede funcionar muy bien con un algoritmo y a la vez, funcionar igual o mejor con otro diferente. No hay ninguna norma generalista aplicable a cualquier conjunto de datos, y esto ocurre tanto en pre-procesamiento de datos como en la elección del algoritmo. Consecuentemente, en este estadio de la implementación ha de considerarse fundamentalmente la naturaleza de los datos que se disponen y cuál es la variable resultado que se está buscando. En función de estas dos variables y de una manera más subjetiva, de la opinión y experiencia del desarrollador, se ha de escoger el o los algoritmos que van ser el núcleo del modelo.

Los pasos a seguir para la arquitectura del núcleo del modelo son:

1. Identificación de la variable resultado
2. Selección del algoritmo

1. Identificación de la variable resultado

   Llegados a esta fase, es necesario tener claro cuál va a ser nuestra variable resultado y entonces clasificar la aplicación entre *Supervised, Unsupervised o Reinforcement Learning*. Parece una cuestión trivial, pero en ocasiones esta pregunta puede desencadenar cuestiones más trascendentales sobre la predicción.

   Incluso una aparentemente sencilla clasificación de sujetos puede dar lugar a cuestiones más profundas sobre la arquitectura del modelo. Por ejemplo, qué objetivo se desea conseguir bajo

esta aplicación: ¿Se desea clasificar únicamente los nuevos sujetos introducidos en el sistema o todos los sujetos bajo cierta política? En el caso de la primera casuística, ¿En cuántas categorías se pueden clasificar?¿se conocen?. En la segunda casuística, ¿cada cuánto se debería volver a ejecutar la clasificación?

Sin pensar demasiado, el primer impulso puede ser el clasificar cada uno de los sujetos cuando se introducen en el sistema. Sin embargo, con dicho modelo, no se es capaz de realizar un seguimiento periódico de los sujetos para guardar una clasificación fideligna en el tiempo. Además, se perdería la opción de descubrir la evolución del sujeto bajo ciertos atributos. Por otra parte, se podría elaborar un modelo en el que se actualizase la clasificación cada vez que se accede al gráfico o al resultado. Esto permitiría estudiar la clasificación en el tiempo y otorgaría la posibilidad de identificar situaciones anómalas. No obstante, puede no haber actualización ninguna entre las veces en las que se consulta el resultado, por lo que no sería óptimo.

2. Selección del algoritmo

   Una vez identificada y especificada la variable resultado y teniendo en mente la naturaleza matemática de los datos disponibles, se procede a la elección del algoritmo Machine Learning.

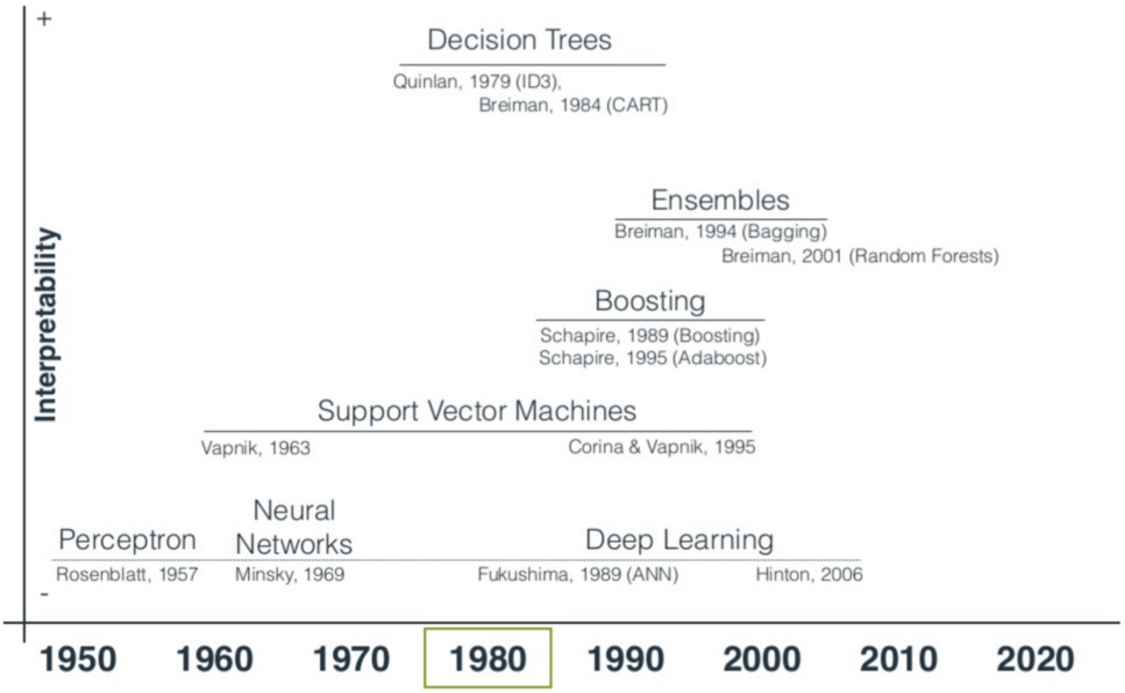

Figura 12: Evolución de la algoritmia central de Machine Learning.

La algoritmia en el campo del Machine Learning no es un campo que se haya descubierto u optimizado en la actualidad reciente. Los algoritmos más veteranos en Inteligencia Artificial fechan de 1957, concretamente el primer sistema de redes neuronales denominado Perceptrón multicapa mientras que los últimos algoritmos que entraron en el ámbito de Machine Learning son los relativos a Deep Learning. Estos llevan siendo la novedad del ámbito desde 2006 y aún considerados como los algoritmos novel de Machine Learning. En la figura 12 se puede observar la evolución de la algoritmia más remarcable en Machine Learning ordenados en el eje vertical

por interpretabilidad humana [19]. De esta misma imagen también podemos extraer que el año más relevante en el desarrollo y optimización de algoritmos fue aproximadamente 1980, hace ya 37 años. Este hecho refuerza lo ya comentado en la introducción de Machine Learning de este mismo proyecto: La parte algorítmica lleva ya bastante tiempo siendo estudiada y optimizada, ha sido el cambio de la tendencia de uso para con la tecnología lo que ha hecho posible y ha reforzado a que ahora Machine Learning esté en auge.

No obstante, aunque en la figura 12 se muestren los algoritmos más comunes, existen muchas categorías de algoritmos utilizadas en Machine Learning y a su vez, diferentes algoritmos dentro de una misma categoría. En la figura 13 se puede observar algunos de ellos que abarcan un gran espectro aunque existen muchas más variantes de los mismos. A continuación, se describirán por encima los ocho algoritmos más utilizados y sus características principales de una manera breve, ya que este tema es tan profundo que es motivo exclusivo de muchos libros.

Figura 13: Algoritmos más utilizados en Machine Learning.

Árboles de decisión

Se trata de un sistema de ayuda a la decisión basado en un conjunto de condiciones organizadas en una estructura jerárquica. Está formado por nodos de decisión que realizan una pregunta sobre el valor de un atributo. Las diferentes respuestas que se pueden dar generan otros nodos de decisión, de tal manera que el árbol se va construyendo creando un modelo que clasifica casos en grupos o pronostica valores de una variable dependiente.

Desde un punto de vista de negocio, un árbol de decisión es el número mínimo de preguntas Sí/No que se necesita plantear para establecer una probabilidad alta de tomar la decisión

correcta la mayor parte del tiempo. Como método, proporciona un enfoque al problema estructurado y sistemático para llegar a una conclusión lógica.

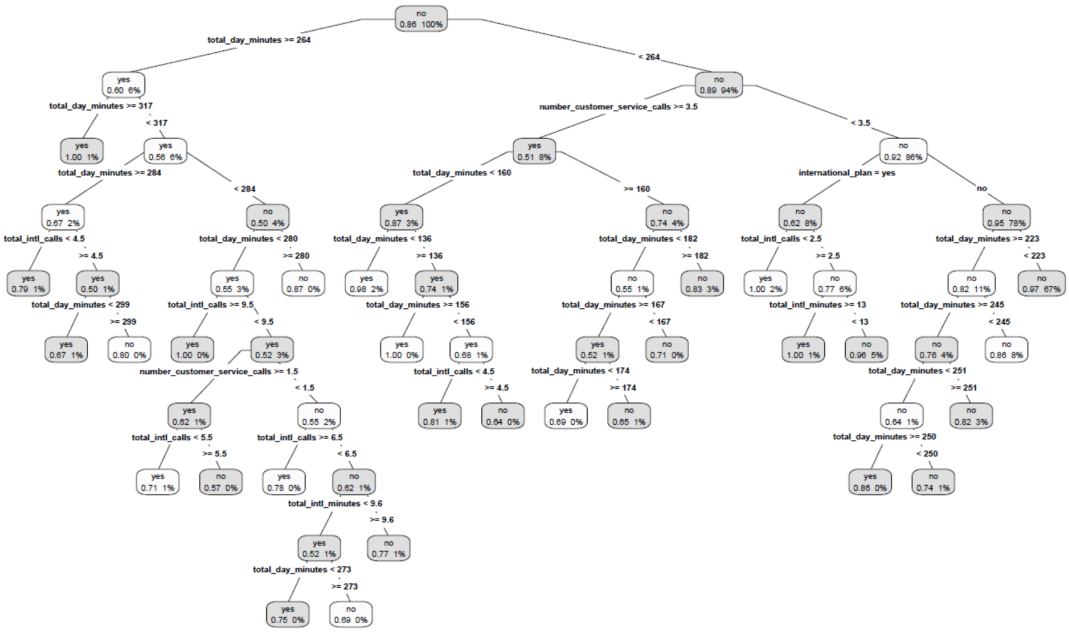

Figura 14: Ejemplo de Árbol de decisión.

Clasificación basada en Naïve-Bayes

Los clasificadores basados en Naïve-Bayes son una familia de clasificadores probabilísticos que se basan en el teorema de Bayes con la hipótesis de independencia entre variables del conjunto de datos. Si se desea predecir la clase o categoría t dado un vector como muestra de entrada $x = x_1, x_1, ..., x_d$ tal que $x \in \mathbb{R}$, se define

$$P(t|x) = P(t|x_1, x_2, ..., x_d) = \frac{P(x_1, x_2, ..., x_d|t)P(t)}{P(x_1, x_2, ..., x_d)}. \tag{9}$$

Obviamente, el cálculo de la probabilidad $P(x_i|t)$ se calcula en función del comportamiento estadístico de la variable aleatoria x_i dando lugar a distintas expresiones (9) o diferentes clasificadores del tipo Naïve-Bayes. Este tipo de clasificadores funciona bastante bien en muchos problemas reales. La principal ventaja de este tipo de clasificadores es que el tiempo de entrenamiento es mucho más rápido. Debido a la premisa de independencia, es decir, que las variables de entrada no están correlacionadas, la estimación de cada distribución de probabilidad $P(x_i|t)$ es más simple y por tanto más rápida. En muchos casos, también es más estable numéricamente. Algunos ejemplos que hacen uso de este algoritmo son, por ejemplo:

- Filtración de correos de Spam
- Clasificación de artículos en diversas categorías
- Software de reconocimiento facial

Regresión por mínimos cuadrados

La regresión por mínimos cuadrados es uno de los métodos utilizados para calcular una regresión lineal. Dado un conjunto de puntos $(x_i, y_i) \in \mathbb{R}^2,\ i = 1, \ldots, n$, la regresión lineeal por

mínimos cuadrados proporciona la ecuación de la recta $f(x) = ax + b$ que aproxima los puntos de manera óptima según el siguiente criterio:

$$\text{mín} \sum_{i=1}^{n} (ax_i + b - y_i)^2. \tag{10}$$

La expresión anterior recibe el nombre de varianza residual. Se trata por tanto de hallar la ecuación de la recta que minimiza la varianza residual. Una ventaja de la regresión por mínimos cuadrados es que podemos conocer la calidad de la aproximación de $f(x)$ a los datos calculando el coeficiente de correlación lineal $\rho = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$, donde σ_{xy} es la covarianza de las variables x e y y σ_x, σ_y son las desviaciones típicas de x e y, respectivamente (si $|\rho| \sim 1$, la aproximación es buena y por tanto $f(x)$ describe bien los datos. Por el contrario, si $|\rho| \sim 0$ la función $f(x)$ no sirve para describir los datos).

El problema (10) puede resolverse para funciones $f(x)$ no lineales, lo cual permite obtener distintas aproximaciones a los datos iniciales.

Regresión Logística

La regresión logística es un método estadístico que modela un resultado binomial ,es decir, acentúa el valor de una variable para discernir entre dos posibles resultados. Un ejemplo de función logística aplicada en este ámbito es el arco tangente. Mide la relación entre la variable categórica dependiente (binaria) y una o más variables independientes y haciendo uso de una función logística. Este algoritmo es utilizado en algunas aplicaciones reales tales como por ejemplo:

- Sistemas de scoring de banca
- Sistemas de análisis de beneficios
- Medición de éxito en campañas de Marketing

Support Vector Machines (SVM)

El Support Vector Machine es un algoritmo de clasificación binaria. Dado un conjunto de puntos de dos tipos en un espacio n dimensional, el SVM genera un hiperplano $(n-1)$-dimensional que permite separar de la manera más discriminatoria posible los puntos introducidos en los dos grupos buscados. Computacionalmente puede suponer un coste muy alto para conjuntos de datos de tamaño intermedio o grande tanto en la fase de training como en la de predicción. Algunas de las aplicaciones que hacen uso de este algoritmo son:

- Clasificación del género basado en imagen
- Clasificaciones de imagen a gran escala

Métodos Ensemble

Los métodos Ensemble, son algoritmos de aprendizaje que construyen un conjunto de clasificadores y toman los resultados de cada uno de los clasificadores que los componen. Lo que diferencia un método Ensemble de otro, es la manera en la que se unifican los resultados de los clasificadores. El método más clásico es la del promedio. Este tipo de algoritmos funciona extremadamente bien cuando se utilizan clasificadores muy diferentes entre sí, es decir, de diferentes familias de algoritmos aunque el coste computacional es muy alto.

K-medias

El algoritmo de K-medias es un algoritmo de clustering donde se trata de identificar grupos en espacios d-dimensionales. Dado un conjunto de training $x_1, x_2, ..., x_N$, $x_N \in \mathbb{R}^d$, el objetivo es particionar el conjunto de datos en K grupos, con k fijado previamente bajo criterio del desarrollador. Es decir, el objetivo es encontrar K prototipos representativos de cada grupo, tal que cada muestra será asignada al cluster correspondiente. Una posibilidad se basa en identificar cada grupo con un vector de medias μ_k y asignar una nueva muestra al grupo cuyo vector media sea más cercano.

Redes Neuronales

Son interesantes porque permiten estudiar aplicaciones no lineales utilizando funciones de activación no lineales. Las función discriminante lineal es una combinación lineal de las variables de entrada x_j y coeficientes w_j (pesos) más una constante w_0 de la forma,

$$y(x) = \sum_{j=1}^{d} w_j x_j + w_0, \tag{11}$$

donde w_j son los parámetros a aprender/estimar.

Desafortunadamente, las soluciones que aportan las funciones discriminantes lineales se limitan a clasificadores basados en hiperplanos que actúan como fronteras de decisión entre las distintas clases. Aunque son adecuadas para muchos problemas reales, obtienen tasas de error muy altas cuando las muestras no son linealmente separables. Las redes neuronales aportan la posibilidad

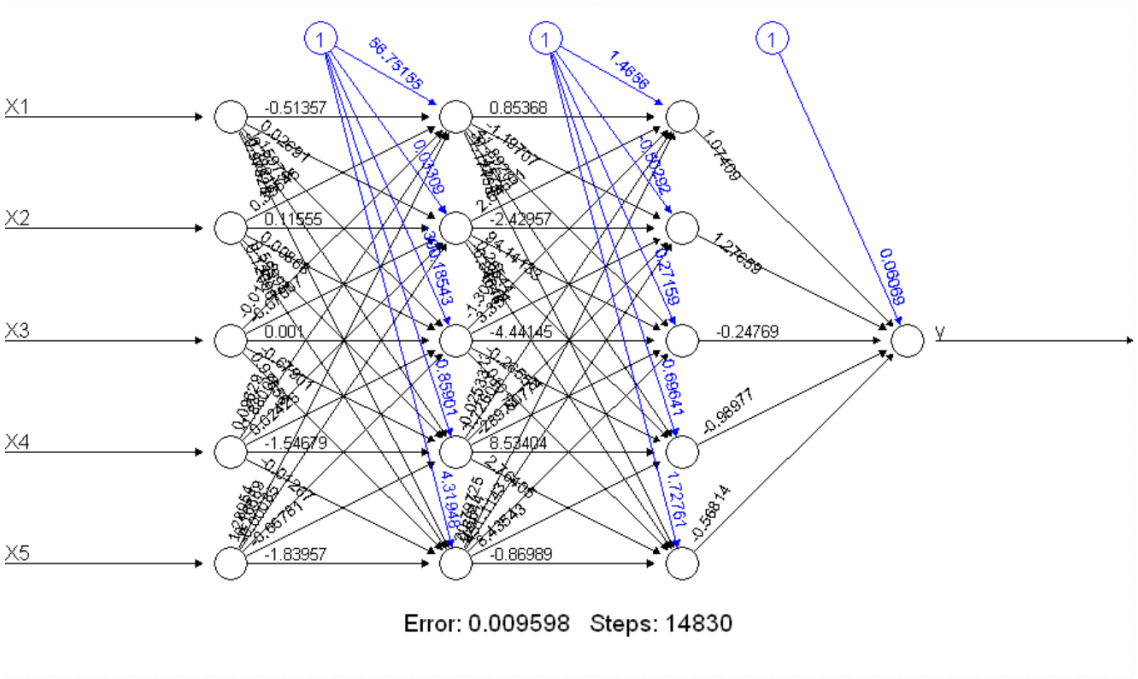

Figura 15: Ejemplo de Red neuronal.

de combinar de manera múltiple, funciones discriminantes lineales con funciones de activación no lineales. De tal manera que una vez se realiza la activación de la red neuronal, ésta ajusta y calcula los pesos w_j en función del error propagado a través de las capas de neuronas. Son eficaces a la hora de aplicarlas a problemas complejos, a costa de su coste computacional y su complejidad de ajuste. En la figura 15 se representa una red neuronal de ejemplo, donde se

pueden apreciar cinco inputs $x_1, x_2, ..., x_5$ y sus correspondientes pesos adjudicados por cada neurona. Los pesos marcados en azul para cada neurona son las constantes [20].

Clustering jerárquico

El *Clustering* jerárquico es una técnica utilizada en *Unsupervised Machine Learning* cuyo objetivo es formar agrupaciones para formar nuevos clusters o para separar uno ya creado dando lugar a otros dos [21]. Con esta técnica, si se realiza sucesivamente tanto por aglomeración como por división, se miniminizará alguna distancia o se maximizará alguna medida de similitud. Generalmente se pueden dividir en dos variantes:

- En métodos aglomerativos, o también conocidos como ascendentes, se comienza realizando el análisis con tantos grupos como muestras haya. A partir de estos grupos iniciales que representan la unidad, se van formando grupos de manera ascendente, hasta que finalmente estén todos los grupos bajo un mismo grupo genérico.
- Los métodos disociativos, o también conocidos como descendentes, constituyen la técnica inversa a los aglomerativos. Comienzan con un gran conglomerado, englobando todas las muestras, y se van formando grupos más pequeños mediante sucesivas divisiones. Al final del proceso, se obtienen tantos grupos como casos han sido tratados.

Para aclarar el concepto, sigamos el proceso de un *Clustering* jerárquico aglomerativo. Sea n el conjunto de muestras, de donde resulta el nivel $K = 0$, con n grupos. En el siguiente nivel, se unirán aquellas dos muestras con mayor similitud o menor distancia, resultando así $n - 1$ grupos. Se procederá sucesivamente de tal manera que en el nivel L, dispondremos de $n - L$ grupos formados hasta que se llegue a $L = n - 1$. En tal punto sólo habrá una agrupación que englobe a todas las muestras.

Estos métodos permiten la creación de un árbol de clasificación, denominado dendrograma, el cual facilita la comprensión del procedimiento gráficamente. En el dendrograma se aprecia la unión de los diferentes grupos y el nivel concreto al que lo hacen. En la figura 16 se muestra un ejemplo de dendrograma donde se pueden observar las relaciones entre un conjunto de países por términos de felicidad.

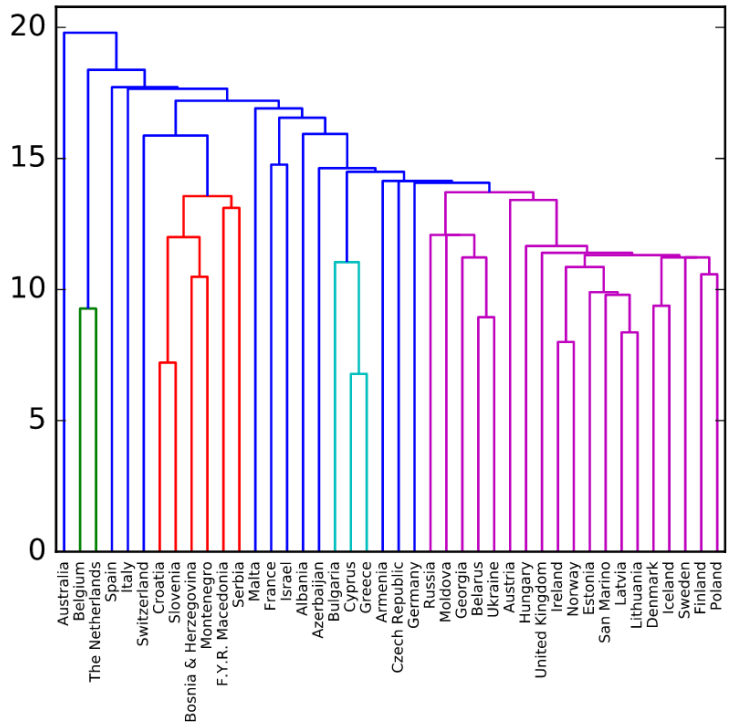

Figura 16: Ejemplo de dendrograma.

2.4. Evaluación del modelo

La base de datos sobre la que se trabaja de manera inicial, se divide en tres partes: *Training, Test* y *Validation*. Esto es debido a que si se introdujese toda la cantidad de información de la que se dispone, no habría manera de evaluar el error que está cometiendo nuestro modelo pues no se tendrían datos fiables contra los que contrastar. De esta manera, los datos introducidos inicialmente al modelo para que .ªprenda"se denominan datos de Training y los datos con los que se realiza una prueba para calcular el error existente, se denominan datos de Test. Adicionalmente, en ocasiones se suele dejar una tercera porción de datos denominados datos de validación por si deseamos o intuimos que se va a necesitar cambiar algún parámetro de nuestro modelo y así poderlo evaluar de nuevo una vez cambiados los parámetros.

En los modelos de clasificación se trata de minimizar los errores de clasificación. Por otra parte, en los modelos de regresión se busca aquel modelo que miniminice la distancia entre los valores observados y los predichos. En otras ocasiones conviene revaluar el modelo en función del coste asociado a los errores. Finalmente, la evaluación de los modelos de clustering depende del nivel de compactación al que se someten los propios datos.

Técnicas de evaluación

- Evaluación basada en la precisión

  La técnica más común se denomina validación cruzada. Consiste en dividir los datos disponibles aleatoriamente en n grupos, donde un grupo se reserva para el conjunto de prueba y con los

otro $n-1$ restantes se realiza el *training*, repitiendo este proceso n veces. Otra técnica común sería el *Bootstrapping*, método de remuestreo que se basa en generar muestras a partir del conjunto inicial de datos.

- Evaluación basada en coste

  Para este tipo de técnica es necesario conocer el tipo de errores y su coste asociado. A través de matrices que muestran el recuento de las clases predichas y sus valores verdaderos, denominadas matrices de confusión, y matrices de costes, el modelo puede ser evaluado y validado según los parámetros de exigencia de la misma aplicación.

Los pasos para la evaluación de grandes conjuntos de datos se podrían resumir en:

1. Dividir aleatoriamente los datos en conjunto de entrenamiento y conjunto de validación. Habitualmente una buena aproximación es dividir 2/3 para entrenamiento y 1/3 para el test.
2. Construir el clasificador usando el conjunto de training.
3. Evaluar dicho clasificador utilizando el conjunto de *testing*.

En el caso específico de no disponer de un gran conjunto de datos, la regla general de la división 2/3 para el *training* y 1/3 para el *test* puede no ser un método adecuado. En estos casos la validación cruzada suele ser una herramienta fundamental donde finalmente las tasas de error se promedian para proporcionar una tasa de error global.

La validación cruzada evita que los conjuntos de test se solapen proporcionando así un método efectivo de medición del error. Los datos se dividen en k subconjuntos de igual tamaño, entonces cada subconjunto es utilizado sucesivamente para test y el resto para el training. Finalmente, los errores estimados se promedian. Este proceso recibe el nombre de validación cruzada de k pliegues [22]. En la figura 17 se observa una representación gráfica de la validación cruzada, donde se seleccionan diferentes *training set* y *test set*.

Figura 17: Ejemplo de validación cruzada de cinco pliegues.

Existe una variante de la validación cruzada, denominada validación cruzada *Leave-One-Out*, que consiste en fijar k igual al número de instancias de *training*, es decir, para n instancias se construye un modelo n veces. Esto aumenta la fiabilidad del sistema ya que hace un mejor uso de los datos y no implica ningún muestreo aleatorio. No obstante, es altamente costoso en términos computacionales.

3. Herramientas estadísticas y algebraicas

En la siguiente sección se profundizará en las herramientas Estadísticas empleadas en el preprocesado de los datos en Machine Learning, desde un punto de vista matemático. Se trata fundamentalmente de aplicar a los datos el Análisis de Componentes Principales para reducir la dimensionalidad de los mismos, conservando la mayor parte de la información que contienen. El cálculo de las componentes principales requiere hallar los valores y vectores propios de una matriz (de la matriz de covarianzas). Ello puede llevarse a cabo mediante distintos métodos algebraicos. Un método excelente se basa en la factorización QR de una matriz [23]. No obstante, por la naturaleza de los datos, suele ser más eficiente en este caso utilizar la Descomposición en Valores Singulares de la matriz de datos.

Presentamos a continuación la descomposición en valores singulares de una matriz, sus principales propiedades y algunas de sus múltiples aplicaciones. En especial, pretendemos mostrar por qué es adecuada en Técnicas de Matching Learning. Asímismo, presentamos la técnica estadística de análisis de componentes principales, sus principales características y cómo puede obtenerse.

3.1. Descomposición por valores Singulares

La descomposición en valores singulares de una matriz rectangular A es una factorización de la matriz, muy útil por sus propiedades. Por una parte permite obtener importantes características de la propia matriz, y por otra es muy interesante por sus aplicaciones. La descomposición en valores singulares puede usarse en el cálculo del rango de una matriz, de su núcleo y de su imagen, de la matriz pesudoinversa, en la resolución del problema de aproximación por mínimos cuadrados, o en problemas de aproximación de matrices por otras de rango más pequeño. Describimos a continuación la descomposición, sus propiedades más importantes y algunas de sus aplicaciones [24] [23].

Teorema 3.1 (Descomposición en valores singulares)**.** *Sea $A \in \mathbb{R}^{m\times n}$ una matriz. Entonces, existen matrices ortogonales $U_{m\times m}$ y $V_{n\times n}$ y números reales no negativos $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, $r \leq p = min\{m, n\}$, tales que*

$$A = UDV^T, \tag{12}$$

donde $D_{m\times n} \in \mathbb{R}^{m\times n}$ es la matriz $diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$ y

$$A = U \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \sigma_r & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}_{m\times n} V^T. \tag{13}$$

Esta factorización de la matriz A se denomina *Descomposición en Valores Singulares* (SVD por sus siglas en inglés) de $A_{m\times n}$. Los elementos diagonales de D, $\sigma_1, \sigma_2, \ldots, \sigma_r$, reciben el nombre de *valores singulares* de $A_{m\times n}$. Las columnas de las matrices $U_{m\times m}$ y $V_{n\times n}$ son los vectores singulares a izquierda y derecha de $A_{m\times n}$, respectivamente. En el caso en el que $r < p = min\{m, n\}$, $A_{m\times n}$ resulta tener uno o más valores singulares adicionales nulos.

Demostración: Para demostrar la existencia de esta descomposición para cualquier matriz, tomamos la dirección de la mayor deformación que produce la transformación A (luego veremos la interpretación geométrica de A) y después procedemos por inducción.

Así, sea $\sigma_1 = \|A\|_2$. Como $\{x \in \mathbb{R}^n : \|x\|_2 = 1\}$ es cerrado y acotado, existen vectores $v_1, u_1 \in \mathbb{R}^n$ tales que $\|v_1\|_2 = \|u_1\| = 1$ y $\|Av_1\| = \sigma_1 u_1$. Extendamos v_1 a una base ortonormal $\{v_1, v_2, \ldots, v_n\}$ de $\mathbb{R}^n$ y u_1 a una base ortonormal $\{u_1, u_2, \ldots, u_n\}$ de $\mathbb{R}^m$. Sean U_1 y V_1 matrices unitarias cuyas columnas son u_j y v_j respectivamente. Se verifica:

$$U_1^* A V_1 = S = \begin{pmatrix} \sigma_1 & w^* \\ 0 & B \end{pmatrix} \tag{14}$$

Donde 0 es un vector columna de dimensión $m-1$, w^* es un vector fila de dimensión $n-1$ y B es una matriz de dimensiones $(m-1)\times(n-1)$. Además:

$$\left\| \begin{pmatrix} \sigma_1 & w^* \\ 0 & B \end{pmatrix} \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|_2 \geq \sigma_1^2 + w^* w = (\sigma_1^2 + w^* w)^{\frac{1}{2}} \left\| \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|_2$$

de donde $\|S\|_2 \geq (\sigma_1^2 + w^* w)^{\frac{1}{2}}$. Como U_1 y V_1 son unitarias, sabemos que $\|S\|_2 = \|A\|_2 = \sigma_1$, lo que implica que $w = 0$.

En caso de que $m = 1$ o $n = 1$, hemos finalizado el proceso. En otro caso, la submatriz B describe la acción de A sobre el subespacio ortogonal a v_1. Entonces, por hipótesis de inducción, B admite una descomposición en valores singulares $B = U_2 D_2 V_2^*$. Es fácil comprobar que

$$A = U_1 \begin{pmatrix} 1 & 0 \\ 0 & U_2 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & V_2 \end{pmatrix}^* V_1^*$$

es la descomposición de A buscada.

Para demostrar la unicidad de los valores de la matriz D, observemos que σ_1 está únicamente determinada por la condición de que es igual a $\|A\|_2$ como se indica en (3,1). A continuación supongamos que además de v_1, hay otro vector linealmente independiente w tal que $\|w\|_2 = 1$ y $\|Aw\|_2 = \sigma_1$. Definimos un vector v_2 de norma 1, ortogonal a v_1, como combinación lineal de v_1 y w como sigue:

$$v_2 = \frac{w - (v_1^* w)v_1}{\|w - (v_1^* w)v_1\|_2} \tag{15}$$

Como $\|A\|_2 = \sigma_1$, $\|Av_2\|_2 \leq \sigma_1$. No obstante, la condición anterior debería cumplir la igualdad ya que de otra manera, como $w = v_1 c + v_2 n$ para unas constantes c y n tal que $\|c\|^2 + \|n\|^2 = 1$ tendríamos que tener $\|Aw\|_2 < \sigma_1$. Este vector v_2 es un segundo vector singular a la derecha de A correspondiente al valor singular σ_1, lo que conlleva la aparición de un vector y igual a las últimas $n - 1$ componentes de $V_1^* v_2$ con $\|y\|_2 = 1$ y $\|By\|_2 = \sigma_1$. Entonces podemos concluir que, si el vector singular v_1 no es único entonces el valor singular σ_1 no es simple. Una vez que σ_1, v_1 y u_1 están determinados, el restante de la descomposición está definido por la acción de A en el espacio ortogonal de v_1. Por ende, como v_1 es único, este espacio ortogonal está únicamente definido y por tanto los valores y vectores singulares son únicos por inducción. Con ello queda probado que si los valroes singulares son distintos, los vectores singulares son distintos, salvo signos (veremos posteriormente la interpretación geométrica de la descomposición en valores singulares).

La descomposición SVD es válida también para matrices complejas reemplazando $(\star)^T$ por $(\star)^*$.

Se verifican las siguientes propiedades. Los resultados se presentan para matrices reales.

Proposición 3.2.

- *Los valores singulares de A son únicos.*
- *El rango de la matriz A es r: $r = rang\ (A)$.*
- *Los valores singulares no nulos de A son las raíces cuadradas positivas de los valores propios no nulos de la matriz $A^T A$ (o de AA^T).*
- *Los vectores singulares a izquierda y derecha de A son vectores propios ortonormales de $A^T A$ y de AA^T, respectivamente.*
- *Un conjunto completo de vectores propios ortonormales v_i de $A^T A$ puede servir como conjunto completo de vectores singulares a la derecha de A, y el correspondiente conjunto completo de vectores singulares a la izquierda viene dado por $u_i = \frac{Av_i}{\|Av_i\|} : i = 1, 2, \ldots, r$ junto con cualquier complemento hasta una base ortonormal del espacio $\{u_{r+1}, \ldots, u_m\}$. Análogamente, cada conjunto completo ortonormal de vectores propios de AA^T puede ser usado como conjunto de vectores singulares a la izquierda de A, y los correspondientes vectores singulares a la derecha se obtienen en forma similar.*

Nota: Es fácil comprobar que los vectores singulares no son únicos.

3.1.1. Interpretación geométrica de la SVD

La descomposición en valores singulares revela información geométrica sobre las transformaciones lineales: indica el grado de distorsión que una transformación origina sobre un vector.

Para analizar esta propiedad, vamos a estudiar cómo una transformación $A \in \mathbb{R}^{n\times n}$ no singular distorsiona la esfera unidad $S_2 = \{x : \|x\|_2 = 1\}$ de $\mathbb{R}^n$. Obtendremos que los valores singulares muestran de forma explícita esta distorsión.

Sea $A \in \mathbb{R}^{n\times n}$ una matriz no singular asociada a una transformación lineal respecto de ciertas bases. Sea

$$A = U\Sigma V^T,$$

su descomposición en valores singulares, y sean $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n > 0$ sus valores singulares. Dado $x \in S_2$, queremos estudiar la norma del vector imagen $y = Ax$.

Recordemos que las transformaciones ortogonales son isometrías, $\|y\|_2 = \left\|U^T y\right\|_2$. Denotemos por $w = U^t y$. Entonces,

$$1 = \|x\|_2^2 = \left\|A^{-1}Ax\right\|_2^2 = \left\|A^{-1}y\right\|_2^2 = \left\|VD^{-1}U^T y\right\|_2^2 = \left\|D^{-1}U^T y\right\|_2^2 = \left\|D^{-1}w\right\|_2^2 =$$

$$= \frac{w_1^2}{\sigma_1^2} + \frac{w_2^2}{\sigma_2^2} + \frac{w_3^2}{\sigma_3^2} + ... + \frac{w_r^2}{\sigma_n^2} \tag{16}$$

Obtenemos que las componentes del vetor w satisfacen la ecuación de un elipsoide en $\mathbb{R}^n$ de semiejes $\sigma_1, \ldots, \sigma_n$. La transformación ortogonal U^T únicamente puede afectar a la orientación de $A(S_2)$, así que $A(S_2)$ es también un elipsoide cuyo k-ésimo semi-eje es σ_k. De otro modo, la transformación A lleva la esfera unidad a un elipsoide en $\mathbb{R}^n$ cuyo k-ésimo semieje es el valor singular σ_k. Una representación gráfica de este efecto en $\mathbb{R}^3$ puede verse en la siguiente figura

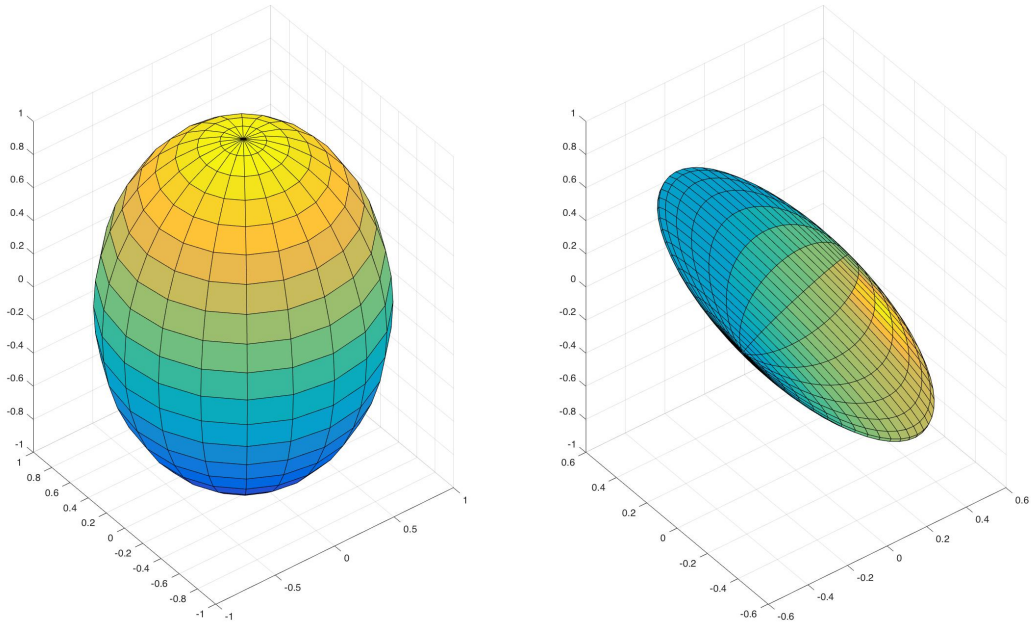

Figura 18: Distorsión de la esfera unidad.

El grado de distorsión que sufre la esfera unidad bajo la transformación no singular A puede medirse por el cociente,

$$\max_{\|x\|_2=1} \|Ax\|_2 / \min_{\|x\|_2=1} \|Ax\|_2.$$

Tenemos que,

$$\max_{\|x\|_2=1} \|Ax\|_2 = \|A\|_2 = \left\|UDV^T\right\|_2 = \|D\|_2 = \max_{\|x\|_2=1} \|Dx\|_2 = \sigma_1. \tag{17}$$

Análogamente, como $A^{-1} = VD^{-1}U^T$,

$$\left\|A^{-1}\right\|_2 = \max_{\|x\|_2=1} \left\|D^{-1}x\right\|_2 = 1/\sigma_n.$$

Observemos que $\sigma_n = \min_{\|x\|_2=1} \|Dx\|_2$. Como V es una matriz ortogonal, transforma biyectivamente la esfera unidad S_2 en la esfera unidad S_2. Así, $x \in S_2$ si y sólo si $y = V^T x \in S_2$. Entonces,

$$\sigma_n = \min_{\|x\|_2=1} \|Dx\|_2 = \min_{\|y\|_2=1} \left\|DV^T y\right\|_2 = \min_{\|y\|_2=1} \left\|UDV^T y\right\|_2 = \min_{\|y\|_2=1} \|Ax\|_2 .$$

Por tanto,

$$\frac{\max_{\|x\|_2=1} \|Ax\|_2}{\min_{\|x\|_2=1} \|Ax\|_2} = \frac{\sigma_1}{\sigma_n}, \tag{18}$$

y este cociente entre el mayor valor singular y el más pequeño se denota por $k_2 = \sigma_1/\sigma_n$. Hemos visto que el vector más grande y el más pequeño de $A(S_2)$ miden $\sigma_1 = \|A\|_2$ y $\sigma_n = 1/\left\|A^{-1}\right\|_2$, respectivamente. De esta manera podemos redefinir el coeficiente como

$$k_2 = \|A\|_2 \left\|A^{-1}\right\|_2 . \tag{19}$$

El coeficiente k_2 es conocido como el *número de condición* de A e indica el grado de distorsión que la transformación A origina sobre la esfera unidad. Se verifica que $k_2 \geq 1$. Si $k_2 = 1$, la transformación A no produce distorsión y ello sucede si y sólo si la matriz es ortogonal. Observemos que $k_2 = 1$ si y sólo si $\sigma_1 = \sigma_2 = \ldots = \sigma_n = 1$ si y sólo si A es ortogonal. Cuanto mayor sea k_2, mayor es la distorsión ocasionada.

El coeficiente $k = \|A\| \left\|A^{-1}\right\|$ siempre proporciona la misma información cualitativa de la distorsión independientemente de la norma que se utilice.

3.1.2. Estabilidad de sistemas de ecuaciones

El grado de distorsión que la esfera unidad puede sufrir bajo una transformación lineal es muy útil para determinar el grado de exactitud de las soluciones de Sistemas Lineales. En sistemas de ecuaciones lineales, las ecuaciones van frecuentemente acompañadas de incertidumbres debidas a errores de modelado por la realización de suposiciones, a errores cometidos en la recogida de datos y a errores de introducción de datos por el hecho de trabajar en matemática finita. A ello hay que sumar los errores de redondeo cometidos en el propio cálculo. El número de condición de la matriz del sistema nos proporciona una idea aproximada sobre la calidad de la solución obtenida.

Consideremos el sistema $Ax = b$. Supongamos que la matriz A es conocida de manera exacta y es no singular, y que b está sometido a cierta incertidumbre e. Denotemos $\tilde{b} = b - e$ y supongamos que $A\hat{x} = \tilde{b}$. Podemos estimar el error relativo cometido en la solución $\|x - \hat{x}\| / \|x\|$ en términos del error relativo $\left\|b - \widehat{b}\right\| / \|b\| = \|e\| / \|b\|$ en b. Tenemos que,

$$\|b\| = \|Ax\| \leq \|A\| \|x\|,$$

$$x - \widehat{x} = A^{-1}e,$$

Así,

$$\frac{\|x - \widehat{x}\|}{\|x\|} = \frac{\left\|A^{-1}e\right\|}{\|x\|} \leq \frac{\|A\| \left\|A^{-1}\right\| \|e\|}{\|b\|} = k\frac{\|e\|}{\|b\|}, \tag{20}$$

donde $k = \|A\| \left\|A^{-1}\right\|$ es el número de condición ($k_2 = \sigma_1/\sigma_n$ en caso de utilizar norma dos). Además,

$$\|e\| = \|A(x - \widehat{x})\| \leq \|A\| \|(x - \widehat{x})\|,$$

$$\|x\| \leq \left\|A^{-1}\right\| \|b\|,$$

lo cual implica

$$\frac{\|x - \widehat{x}\|}{\|x\|} \geq \frac{\|e\|}{\|A\| \|x\|} \geq \frac{\|e\|}{\|A\| \|A^{-1}\| \|b\|} = \frac{1}{k}\frac{\|e\|}{\|b\|}, \tag{21}$$

De (20) y (21) deducimos que

$$\frac{1}{k}\frac{\|e\|}{\|b\|} \leq \frac{\|x - \widehat{x}\|}{\|x\|} \leq k\frac{\|e\|}{\|b\|}. \tag{22}$$

donde $k = \|A\| \left\|A^{-1}\right\|$. Esto significa que cuando k es pequeño, las incertidumbres existentes sobre b son ínfimamente influyentes sobre la solución del sistema. Se dice entonces que la matriz A está *bien condicionada*. Sin embargo, si k es muy grande, una pequeña incertidumbre en b podría afectar a la solución de forma considerable. Se dice entonces que la matriz A está *mal condicionada*.

El número de condición nos indica también si podemos fiarnos del valor del residuo $r = b - A\widehat{x}$ para saber si una solución calculada $\widehat{x}$ es aproximada a la verdadera solución.

Sea $\widehat{x}$ la solución calculada para el anterior sistema $Ax = b$, con A no singular, y $r = b - A\widehat{x}$ el residuo obtenido para dicha solución. Como $A\widehat{x} = b - r$, tomando $e = r$ en (24) resulta

$$\frac{1}{k}\frac{\|r\|}{\|b\|} \leq \frac{\|x - \widehat{x}\|}{\|x\|} \leq k\frac{\|r\|}{\|b\|}. \tag{23}$$

Por tanto, si A está bien condicionada pequeños valores de $\|r\|$ indican que el error relativo cometido en la solución es pequeño, ya que k es pequeño. En cambio, si A está mal condicionada, un valor pequeño de $\|r\|$ no garantiza un pequeño error relativo en el cálculo de la solución.

3.1.3. Distancia a la matriz más próxima

Además de medir la distorsión a la que una transformación somete a la esfera unidad y determinar el grado de consistencia de sistemas lineales, los valores singulares indican la distancia de A a matrices de rango inferior. Ésta es una importante propiedad por las aplicaciones que presenta.

Teorema 3.3. *Sea $A \in \mathbb{R}^{m\times n}$ una matriz y $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r$ los valores singulares de A. Entonces, para cada $k < r$, la distancia de A a la matriz más próxima de rango k es*

$$\sigma_{k+1} = \min_{rango(B)=k} \|A - B\|_2 . \tag{24}$$

Demostración: Consideremos la descomposición en valores singulares de A,

$$A = U \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} V^T,$$

con $D = diag(\sigma_1, \sigma_2, ..., \sigma_r)$ y U, V matrices ortogonales de tamaños adecuados. Sea $k < r$, y sea $B \in \mathbb{R}^{m\times n}$ tal que rang $(B) = k$. Definimos $S = diag(\sigma_1, ..., \sigma_{k+1})$ y partimos V en la forma $V = [F_{n\times(k+1)}\ G]$. Como el rang $(BF) \leq$ rang $(B) = k$, la dimensión del núcleo de BF satisface

$$\dim N(BF) = (k+1) - \text{rang } (BF) \geq 1,$$

por tanto $N(BF) \neq \{0\}$. Elijamos $x \in N(BF)$ tal que $\|x\|_2 = 1$. Se cumple que $BFx = 0$, y

$$AFx = U \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} V^T Fx = U \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} F^T \\ G^T \end{pmatrix} Fx = U \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} F^TF \\ G^TF \end{pmatrix} x =$$

$$= U \begin{pmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 & 0 \\ & \ddots & & & & & \\ 0 & \dots & \sigma_{k+1} & 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & \sigma_{k+2} & \dots & 0 & 0 \\ & & & & \ddots & & \\ 0 & \dots & 0 & 0 & \dots & \sigma_r & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} I \\ 0 \end{pmatrix} x = U \begin{pmatrix} \sigma_1 & \dots & 0 \\ & \ddots & \\ 0 & \dots & \sigma_{k+1} \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \\ 0 & \dots & 0 \end{pmatrix} x$$

Como $\|A-B\|_2 = \text{máx}_{\|y\|_2=1} \|(A-B)y\|_2$ y $\|Fx\|_2 = \|x\|_2 = 1$,

$$\|A-B\|_2^2 \geq \|(A-B)Fx\|_2^2 = \sum_{i=1}^{k+1} \sigma_i^2 x_i^2 \geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} x_i^2 = \sigma_{k+1}^2 \tag{25}$$

Si tomamos $B_k = U \begin{pmatrix} D_k & 0 \\ 0 & 0 \end{pmatrix} V^T$ con $D = diag(\sigma_1, ..., \sigma_k)$ se alcanza la igualdad en la expresión (25), con lo cual queda probado (24).

Debido a esta última propiedad el uso de la descomposición en valores singulares está ampliamente extendido en una gran variedad de campos en los que la reducción de dimensionalidad puede optimizar procesos. A continuación veremos alguno de estos ejemplos:

Filtrado de Ruido

La descomposición en valores singulares puede ser muy útil a la hora de quitar ruido a una señal. Supongamos que $A \in \mathbb{R}^{m\times n}$ es una matriz cuyas componentes son datos de una señal contaminada con ruido, por ejemplo muestras digitales de una señal de vídeo o audio con cierto grado de ruido. Aplicando la descomposición en valores singulares podemos descomponer A como suma de componentes mutuamente ortogonales:

$$A = U \begin{pmatrix} D_{r\times r} & 0 \\ 0 & 0 \end{pmatrix} V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T = \sum_{i=1}^{r} \sigma_i Z_i, \tag{26}$$

donde $Z_i = u_i v_i^T$ y $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. Las matrices $Z_1, \ldots, Z_r$ constituyen un sistema ortonormal si tomamos la *traza* como producto escalar, puesto que

$$\langle Z_i, Z_j \rangle = traza(Z_i Z_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases}$$

Por tanto, la descomposición (26) puede entenderse como una Serie de Fourier, por lo que $\sigma_i =< Z_i|A >$ puede ser interpretado como la proporción de A representada en la dirección de Z_i. En muchas aplicaciones el ruido es aleatorio, es decir, está distribuido uniformemente por todas las Z_i de tal manera que tenemos prácticamente la misma cantidad de ruido en cualquier dirección. Así, si denotamos or SNR la relación señal/ruido (en inglés Signal to noise ratio SNR o S/N; se define como la proporción existente entre la potencia de la señal que se transmite y la potencia del ruido que la corrompe), podríamos suponer

$$SNR(\sigma_1 Z_1) > SNR(\sigma_2 Z_2) > \ldots > SNR(\sigma_r Z_r), \tag{27}$$

Así, en el caso en que algún o algunos valores singulares (pongamos $\sigma_{k+1}, \ldots, \sigma_r$) sean pequeños en comparación con la cantidad media de ruido, los términos $\sigma_{k+1} Z_{k+1}, \ldots, \sigma_r Z_r$ tienen pequeñas ratio señal/ruido. Si truncamos la expresión (26) eliminando los últimos $r-k$ términos, se eliminará una pequeña cantidad de señal, pero una gran cantidad de ruido. Por ello, la descomposición en valores singulares truncada de A

$$A_k = \sum_{i=1}^{k} \sigma_i Z_i,$$

permite en muchos casos filtrar la señal de manera que a pesar de estar eliminando una cierta cantidad de señal, quitamos una cantidad importante de ruido ganando en el balance general.

Determinar el mejor valor para establecer el umbral entre componente poco o muy ruidosa requiere de otras técnicas empíricas que varían en función de la aplicación en la que se esté trabajando, pero como norma por defecto, un buen método es establecer dicho umbral en aquellos valores singulares que presenten mayor separación entre ellos.

Motores de Búsqueda

Otra de las aplicaciones en la que la descomposición por valores singulares juega un papel importante es en Minería de Datos en la Recuperación de la Información o en términos más conocidos, en los Motores de Búsqueda.

Imaginemos que tenemos un conjunto de palabras o frases clave que representan nuestra idea a buscar $P_1, P_2, ..., P_n$, mientras que por otra parte tenemos en nuestra base de datos un conjunto de páginas web $W_1, W_2, ..., W_j$ sobre las que contrastaremos si contienen la información que buscamos. Cada una de estas páginas W_j será escaneada y analizada para detectar si contiene nuestros términos clave; es lo que se conoce en el área como indexación del documento. De esta manera, se crea un vector que contiene la frecuencia de cada uno de los términos por cada página contrastada denominado *vector documento* $d_j = (f_{1j}, f_{2j}, ..., f_{nj})^T$

$$f_{mj} = \text{Número de veces que el término } P_m \text{ aparece en } W_j.$$

Una vez realizado este proceso con cada término, podemos construir una matriz, la matriz general de la búsqueda, cuyas columnas son los vectores de frecuencias d_j:

$$A_{m\times n} = \begin{pmatrix} d_1 & d_2 & \dots & d_j \end{pmatrix} = \begin{array}{c} \\ P_1 \\ P_2 \\ \vdots \\ P_m \end{array} \begin{array}{c} \begin{array}{cccc} W_1 & W_2 & \cdots & W_j \end{array} \\ \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1j} \\ f_{21} & f_{22} & \cdots & f_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m_1} & f_{m_2} & \cdots & f_{mj} \end{pmatrix} \end{array} \tag{28}$$

Como bien podemos intuir, muchos términos del diccionario no aparecen en numerosas páginas, con los cual las frecuencias correspondientes serán cero, por lo que la matriz A suele ser una matriz dispersa de grandes dimensiones.

Cuando el usuario realiza una petición de búsqueda, se genera un vector de consulta (en inglés "query") $q^T = (q_1, q_2, ..., q_n)$ que irá rellenando el sistema tal que

$$q_i = \begin{cases} 1 & \text{Si el término } T_i \text{ está en la petición,} \\ 0 & \text{Si el término } T_i \text{ no está en la petición.} \end{cases}$$

Así pues, para determinar cuan bien se ajusta una query q a una página web D_j, calculamos cuan cerca está q de d_j como se indica a continuación

$$cos(\theta_j) = \frac{q^T d_j}{\|q\|_2 \|d_j\|_2} = \frac{q^T A e_j}{\|q\|_2 \|Ae_j\|_2} \tag{29}$$

En el caso en que $|cos(\theta_j)| \geq \gamma$ siendo γ un umbral de tolerancia, entonces el documento D_j sería considerado relevante para la búsqueda. Si tomamos las columnas de A y el vector q normalizados,

entonces $q^T A = (|cos(\theta_1)|, |cos(\theta_2)|, \ldots, |cos(\theta_n)|)$ proporciona la información adecuada para que el motor de búsqueda ordene la relevancia de cada documento en relación a cada consulta. No obstante, debido a la ambiguedad y variación del uso humano del vocabulario y a la propia naturaleza del problema, existe bastante "ruido" en la matriz A, lo que dificulta una búsqueda eficaz.

De nuevo, la Descomposición por Valores Singulares puede desempeñar un papel de filtro si la truncamos adecuadamente. Así en lugar de utilizar la descomposición (25), podemos usar

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T. \tag{30}$$

Recalculamos

$$cos(\theta_j) = \frac{q^T A_k e_j}{\|q\|_2 \|A_k e_j\|_2} \tag{31}$$

Si definimos $S_k = D_k V_k^T = \begin{pmatrix} s_1 & s_2 & \ldots & s_k \end{pmatrix}$, tenemos

$$\|A_k e_j\|_2 = \left\|U_k D_k V_k^T e_j\right\|_2 = \|U_k s_j\|_2 = \|s_j\|_2 , \tag{32}$$

entonces

$$cos(\theta_j) = \frac{q^T U_k s_j}{\|q\|_2 \|s_j\|_2}. \tag{33}$$

Los vectores U_k y S_k sólo han de calcularse una única vez y no hace falta que se calcule la descomposición en valores singulares completa, lo que agiliza mucho el cómputo de este cálculo ganando en eficiencia. Además, podemos deshacernos de muchos componentes debido a que la variación del vocabulario y la ambigüidad introducen mucho ruido en A. Si añadimos a ésto el hecho de que no estamos interesados tanto en el resultado exacto matemático de la operación sino en su ordenación, puede reemplazarse A por una aproximación A_k donde k es mucho menor que el rango de la matriz inicial. Ello determina que procesar una consulta requiera muy poca computación.

3.2. Análisis de Componentes Principales

En Big Data, como su propio nombre indica, se manejan grandes Bases de Datos con tal cantidad de variables que en general el cómputo de cualquier operación con los mismos supone un enorme consumo de recursos y de tiempo. Es por esto por lo que resulta fundamental, en la medida de lo posible, reducir al máximo el número de variables sin apenas perder información relevante para el conjunto de datos. Es aquí donde el Análisis de Componentes Principales juega un papel fundamental [25] [26].

En general disponemos de datos correspondientes a un cierta cantidad de variables que pueden estar correlacionadas. La idea principal del Análisis de Componentes Principales es reducir la dimensionalidad del conjunto de datos, manteniendo la máxima cantidad de información posible, pero eliminando información redundante. Esto se consigue mediante una transformación de las p variables iniciales observadas en q nuevas variables incorreladas, $q < p$, de manera que contengan la mayor cantidad de información posible de las variables originales.

Denotamos por $X_1, X_2, \ldots, X_p$ las variables originales. Suponemos que de cada una de ellas conocemos n datos, por tanto disponemos de la matriz de datos

$$A = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix}.$$

La cantidad de información se cifra, en términos estadísticos, en términos de las varianzas:

$$I = \sum_{j=1}^{p} \sum_{i=1}^{n} \frac{1}{n} (x_{ij} - \bar{x}_j)^2.$$

Así, los nuevos componentes deben retener la mayor cantidad de variación presente en las variables originales.

Supondremos que las variables están tipificadas, por tanto su matriz de correlación R coincide con la matriz de covarianzas Σ

$$R = \frac{1}{n} A^T A = \Sigma. \tag{34}$$

Para determinar las componentes principales buscamos en primer lugar una combinación lineal de las variables $X_1, X_2, \ldots, X_p$ que presente la mayor varianza posible. Sea $\alpha_1^T X$ con $X^T = \begin{pmatrix} X_1 & X_2 & \ldots & X_p \end{pmatrix}$ una función lineal de la forma

$$\alpha_1^T X = \alpha_{11} X_1 + \alpha_{12} X_2 + \ldots + \alpha_{1p} X_p = \sum_{j=1}^{p} \alpha_{1j} X_j. \tag{35}$$

Análogamente, se pueden expresar las restantes componentes $\alpha_2^T X, \ldots, \alpha_m^T X$ en la forma

$$\alpha_i^T X = \alpha_{i1} X_1 + \alpha_{i2} X_2 + ... + \alpha_{ip} X_p, \ i = 2, \ldots, m.$$

Obtención de las Componentes Principales

La información que presenten las nuevas variables viene dada por la suma de sus varianzas:

$$I = \sum Var(\text{nuevas variables}) = \sum_{i=1}^{m} Var(\alpha_i^T X).$$

Se trata, por tanto, de obtener nuevas variables con la mayor varianza posible.

Matemáticamente, la búsqueda de las componentes principales consiste en efectuar una transformación ortogonal que permita describir los datos en un nuevo sistema de coordenadas, de manera que la mayor varianza bajo cierta proyección de los datos corresponda a la primera coordenada, que se denominará *Primera Componente Principal*, la segunda mayor varianzad sobre la segunda coordenada, que se denominará *Segunda Componente Principal*, y así sucesivamente.

Para ello, hemos de calcular el vector α_1 de norma 1 que maximiza $Var(\alpha_1^T X) = \alpha_1^T \Sigma \alpha_1$, donde Σ representa la matriz de covarianzas. Es decir, hemos de resolver el problema:

$$\begin{gathered} \text{máx}\, \alpha_1^T \Sigma \alpha_1, \\ \alpha_1^T \alpha_1 = 1. \end{gathered}$$

Se trata de un problema de extremos condicionados; se puede resolver por tanto utilizando Multiplicadores de Lagrange. El conjunto de puntos tales que $\alpha_1^T \alpha_1 = 1$ es un conjunto cerrado y acotado, por tanto la función a maximizar presenta extremos sobre él. Consideremos la función,

$$\alpha_1^T \Sigma \alpha_1 - \lambda(\alpha_1^T \alpha_1 - 1),$$

donde λ es un multiplicador de Lagrange. En los extremos la diferencial de esta función con respecto a α_1 es 0:

$$\Sigma \alpha_1 - \lambda \alpha_1 = 0.$$

Es decir, λ es un valor propio de Σ y α_1 un vector propio asociado a λ. Para escoger qué vector propio maximiza la varianza $\alpha_1^T \Sigma \alpha_1$, observemos que

$$\alpha_1^T \Sigma \alpha_1 = \alpha_1^T \lambda \alpha_1 = \lambda \alpha_1^T \alpha_1 = \lambda,$$

con lo cual, lo que tenemos que maximizar es λ. Así pues, α_1 es un vector propio de norma 1 correspondiente al mayor valor propio de Σ, y $\max Var(\alpha_1^T x) = \lambda_1$.

La segunda Componente Principal se obtiene maximizando

$$\alpha_1^T \Sigma \alpha_1,$$

sujeta a la restricción $\alpha_2^T \alpha_2 = 1$ y a que $\alpha_2^T X$ debe presentar correlación nula con $\alpha_1^T X$ o lo que es lo mismo, su covarianza debe ser 0:

$$Cov(\alpha_2^T X, \alpha_1^T X) = \alpha_2^T \Sigma \alpha_1 = \lambda_1 \alpha_2^T \alpha_1 = \lambda_1 \alpha_1^T \alpha_2 = 0.$$

Es decir, hemos de resolver el problema de extremos condicionados

$$\begin{gathered} \max \alpha_2^T \Sigma \alpha_2, \\ \alpha_1^T \alpha_1 = 1, \\ \alpha_1^T \alpha_2 = 0. \end{gathered}$$

De nuevo por multiplizadores de Lagrange, la diferencial de la función

$$\alpha_2^T \Sigma \alpha_2 - \lambda(\alpha_2^T \alpha_2 - 1) - \mu \alpha_1^T \alpha_2,$$

debe ser 0. Calculando la diferencial con respecto a α_2 obtenemos,

$$\Sigma \alpha_2 - \lambda \alpha_2 - \mu \alpha_1 = 0.$$

Multiplicando esta ecuación por α_1^T resulta

$$\alpha_1^T \Sigma \alpha_2 - \lambda \alpha_1^T \alpha_2 - \mu \alpha_1^T \alpha_1 = 0.$$

Vemos que los dos primeros términos son 0, por tanto $\mu = 0$. Así, λ es un valor propio de Σ y α_2 es un vector propio asociado de norma 1. De nuevo, $\alpha_2^T \Sigma \alpha_2 = \lambda_2$ y debe ser maximizado. Suponiendo que Σ presenta valores propios diferentes (no analizamos aquí el caso en que Σ presente valores propios repetidos), y α_2 es incorrelado con α_1, λ_2 debe ser el segundo mayor valor propio y α_2 su vector propio asociado normalizado.

Siguiendo este proceso se obtiene que la k-ésima Componente Principal de X es $\alpha_k^T x$ cuya varianza $Var(\alpha_k^T x) = \lambda_k$, donde λ_k es el k-ésimo valor propio más grande de Σ, y α_k un vector propio asociado a λ_k de norma 1.

Obtenemos así la transformación ortogonal $\begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_p \end{pmatrix}$ que transforma las variables X en variables incorreladas. En general queremos seleccionar únicamente m de las p variables incorreladas con $m << p$.

A continuación se muestra un ejemplo donde se seleccionarán m Componentes Principales, con $m \ll p$, manteniendo la máxima cantidad de información. Para representarlo gráficamente, se muestra en la figura 19 una base de datos sobre flores donde se puede observar la distribución que siguen cada una de las cuatro variables con respecto las demás.

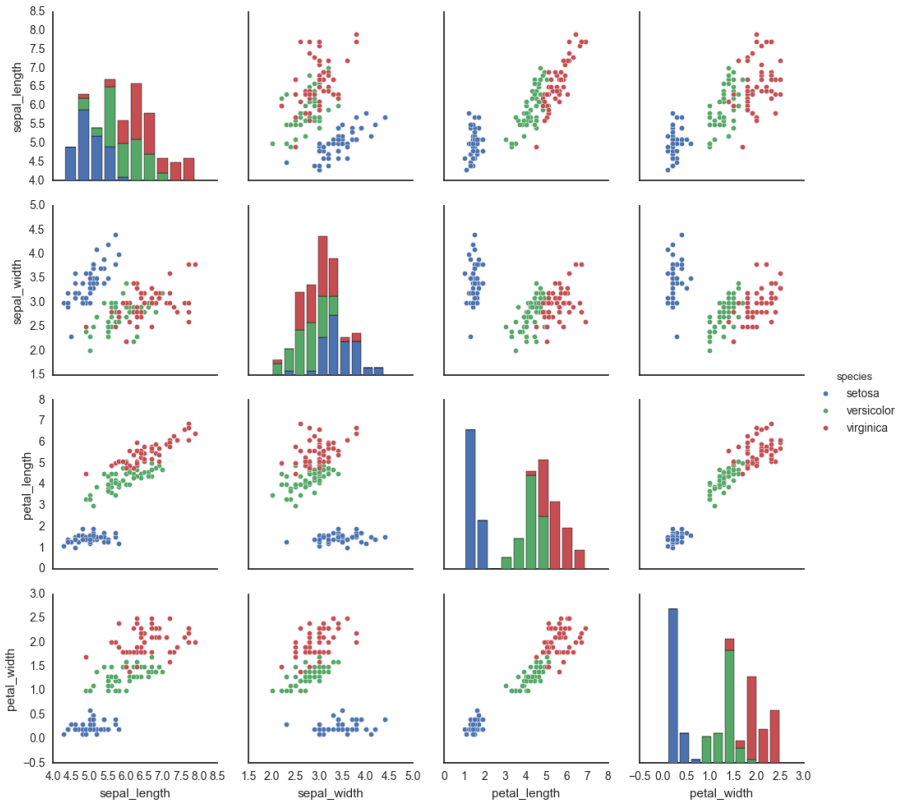

Figura 19: Distribuciones de las variables.

Seguidamente, se aplica componentes principales a la base de datos para reducir la dimensionalidad de la misma.

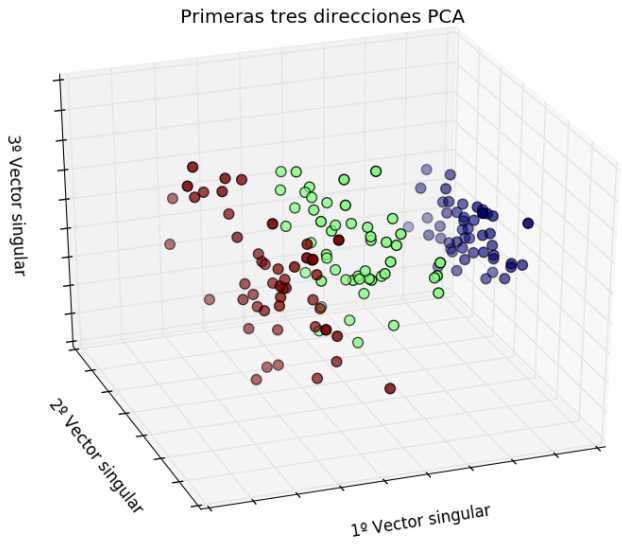

Figura 20: Conjunto de datos representado según las tres primeras componentes principales.

En la figura 20 se puede observar el nuevo subespacio de trabajo representado en función de las tres componentes principales más representativas, las cuales facilitan la discriminación de las clases de flores. Asimismo, se aprecia que este nuevo sub-espacio, ha sido resultado de una combinación lineal de las componentes principales en función de las cuatro variables originales.

Existen diversos métodos de selección del número de Componentes Principales para lograr máxima eficacia, aunque siempre queda bajo disposición de los requerimientos de la aplicación. No obstante, los más comunes son:

- **Porcentaje mínimo de inercia**

  Se establece un porcentaje umbral de varianza P que se desea conservar de tal manera que se conservarán todas las componentes principales que satisfagan

  $$\frac{\sum_{j=1}^{q} \lambda_j}{p} \geq P.$$

- **Gráfico de Sedimentación o Método del Codo**

  En la representación gráfica de las respectivas varianzas de las componentes principales, se puede en general identificar de manera visual una variación brusca en el valor de las mismas originando un codo. Así pues, se consideran de baja relevancia aquellas componentes cuya varianza queda por debajo del codo. Por ejemplo, en la figura 32, marcando el codo como indican las líneas rojas nos quedaríamos únicamente con las dos componentes que preceden al codo. No obstante, la ubicación del codo es algo subjetivo que queda bajo decisión del analista.

- **Criterio de la Media Aritmética** Se conservarán aquellas componentes principales cuya varianza esté por encima de la media aritmética,

  $$Var(\alpha_k^T x) > \frac{\sum_{j=1}^{p} Var(\alpha_j)}{p}.$$

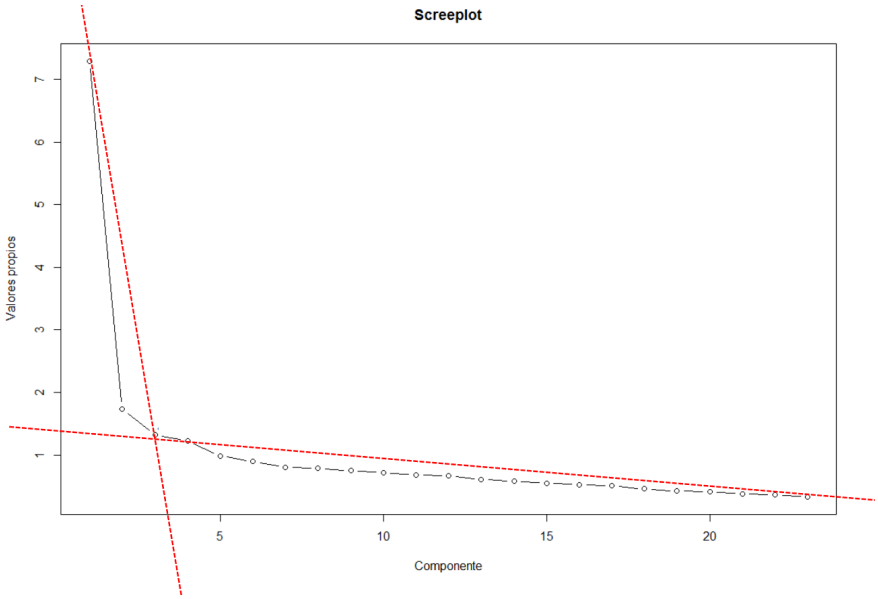

Figura 21: Ejemplo de método del codo.

Relación con la Descomposición por Valores Singulares

Existe una relación matemática directa entre la Descomposición en Valores Singulares visto anteriormente en el apartado 3.1.1 y el Análisis de Componentes Principales (Ver Proposición 3.2). Esta relación no sólo optimiza el cálculo de componentes principales sino que aporta robustez al cálculo.

Acabamos de ver que las componentes principales son obtenidas calculando los valores y vectores propios de la matriz de covarianzas Σ, la cual, como hemos visto en (34), si consideramos que las variables iniciales están tipificadas, es

$$\Sigma = A^T A,$$

donde A es la matriz de datos sobre los que se desea calcular los componentes principales. Sabemos que se trata de una matriz simétrica y por tanto posee valores propios reales y base de vectores propios ortonormales. Hemos supuesto que Σ posee todos los valores propios distintos, y hemos denominado $\alpha_1, \alpha_2, \ldots, \alpha_p$ a la base ortonormal de vectores propios obtenida. Denotemos por W la matriz ortogonal $W = \begin{pmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_p \end{pmatrix}$. Tenemos que,

$$A^T A W = W K,$$

con $K = diag(\lambda_1, \lambda_2, \ldots, \lambda_p)$. Es decir,

$$A^T A = W K W^T. \tag{36}$$

Por otra parte, si recordamos la expresión general de la descomposición por valores singulares (12), se podría reescribir (36) como sigue

$$A^T A = (UDV^T)^T (UDV^T) = VDU^T UDV^T,$$

y teniendo en cuenta que $U^T U = I$, resulta

$$A^T A = VD^2 V^T.$$

De esta última expresión es fácilmente observable la relación matemática entre Componentes Principales y Descomposición por Valores Singulares anunciada en la Proposición 3.2: Los valores propiosno nulos de la matriz de covarianzas Σ son el cuadrado de los valores singulares no nulos de la matriz A.

El cálculo de componentes principales utilizando la descomposición por valores singulares resulta ser mucho más eficiente computacionalmente, pero además también aporta consistencia por el hecho de que al generar la matriz de covarianzas A^TA se introduce pérdida de precisión numérica que puede ser crucial para algunas matrices, como por ejemplo la matriz de Läuchli:

$$L = \begin{pmatrix} 1 & 1 & 1 \\ \rho & 0 & 0 \\ 0 & \rho & 0 \\ 0 & 0 & \rho \end{pmatrix} \tag{37}$$

donde ρ es un número pequeño. Los valores singulares cuadrados para este tipo de matriz serían $3+\rho^2, \rho^2$ y ρ^2. Tomemos a continuación $\rho_1 = 1e^{-10}$ y $\rho_2 = 1e^{-25}$ y calculemos la descomposición en valores singulares al cuadrado de la matriz y el cálculo de valores propios de la matriz de covarianzas L^TL. Operando en *MatLab* resulta:

	ρ_1			ρ_2		
Val. singulares cuadrados de L	3	$1e^{-10}$	$1e^{-10}$	3	$1e^{-50}$	$1e^{-50}$
Val. propios de L^TL	3	$1e^{-10}$	$1e^{-10}$	3	$-2{,}3721e^{-17}$	$-3{,}3307e^{-16}$

por lo que queda comprobado que se introduce un error al sistema al calcular la matriz de covarianzas en lugar de apoyarse en el cálculo de descomposición por valores singulares.

4. Aplicación a la detección y clasificación del cáncer de mama

Como hemos mencionado, en este trabajo aplicamos técnicas de Machine Learning a la detección y clasificación del cáncer de mama. Se hará uso de los conocimientos teóricos recopilados previamente para la implementación y la evaluación de la misma. Adicionalmente, en la presentación, nos apoyaremos en herramientas de visualización diversas para facilitar la comprensión del lector.

Esta aplicación consta de dos modelos Machine Learning implementados por separado por la diferente naturaleza de las dos casuísticas contempladas: detección utilizando técnicas supervisadas y clasificación de tipos utilizando técnicas no supervisadas. En cada una de ellas, se recorrerán los pasos que se han seguido en la implementación y se comentarán los resultados obtenidos, haciendo mención en el papel de la reducción de dimensionalidad de los datos. Las dos bases de datos de cada respectivo modelo, no están relacionadas. Es decir, contienen información de diferentes pacientes. No obstante, las dos técnicas pueden reforzarse mutuamente si son utilizadas con datos procedentes del mismo conjunto de pacientes. Ello sería apropiado en una futura aplicación real.

4.1. Detección

El modelo de Machine Learning para la detección hará uso de la base de datos *Breast Cancer Wisconsin Dataset* comentado anteriormente en el apartado 1.3.2 Fuente de datos. Este conjunto de datos está formado por información procedente de 569 pacientes; para cada paciente se dispone de 33 atributos relacionados con indicadores de imagen de la masa tumoral, extraídos computacionalmente a partir de mamografías. Entre los indicadores se encuentran parámetros tales como la concavidad, densidad de la masa tumoral, tamaño etc. y también se encuentra el indicador que se va a convertir en nuestra variable a predecir: Si es una masa tumoral benigna o maligna. Partimos de la premisa de que todas las pacientes sometidas a este sistema de diagnóstico presentan una masa tumoral identificable en las mamografías. Por ello, sobre las pacientes analizadas los indicadores siempre van a presentar datos. Nuestro sistema de detección predecirá si se trata de una masa tumoral benigna o maligna haciendo uso de técnicas de *Supervised Learning*.

4.1.1. Tratamiento de datos

El primer paso supone únicamente una aproximación a los datos con el objetivo de ser conocedores de la información de la que se dispone. Así, en la primera toma de contacto con el conjunto de datos se exploran los datos para localizar atributos, eventualmente depurar algunos datos, observar ausencias y/o relaciones entre ellos. Esto se puede llevar a cabo fácilmente importando los datos en una variable *data* y aplicando los métodos de Python *data.head(),data.info(),data.describe()*. En las figuras 22 y 38, se muestran las salidas de los métodos *data.head()* y *data.describe()* respectivamente, donde podemos ver la naturaleza de los datos que conforman el dataset y donde podemos apreciar qué atributos se van a manejar. Ambas figuras se muestran cortadas mostrando únicamente las primeras instancias con el objeto de mantener un formato sencillo para el lector. Como puede observarse, los datos mostrados son suficientemente representativos para extraer la información que se pretende en esta fase.

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
id								
842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010
842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690
84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740
84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140
84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800
843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780
844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270
84458202	M	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366
844981	M	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590
84501001	M	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730

10 rows × 32 columns

Figura 22: Salida del método *data.head()* de python

Observando más en detalle el dataset que se muestra en la figura 22, se puede identificar fácilmente que de entre los 32 atributos que lo conforman, al menos ya existe uno categórico: *Diagnosis*, la primera columna. También se identifica que el dataset está indexado por número de paciente, lo que indica que formalmente está bien construido para trabajar con Machine Learning.

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200

8 rows × 31 columns

Figura 23: Salida del método *data.describe()* de python

Con el método *data.describe()* de la figura 38 se puede observar cuáles son los indicadores estadísticos que devuelve el método para cada atributo: Número de valores, media, desviación típica, valor mínimo, valor máximo y percentiles. Gracias a este sencillo método es muy fácil identificar la cantidad de valores ausentes en cada atributo (indicador *count*), descartar atributos no numéricos y verificar algunas características estadísticas de los mismos. Con este método confirmamos que únicamente se dispone de un atributo categórico, basta con observar que el tamaño de la tabla estadística resultante es 8×31, es decir, un atributo ha sido ignorado por ser no numérico.

Esto se puede asegurar de manera más contundende con el método *data.info()* como se representa en la figura 24. En dicho método de Python se especifica cada atributo presente en el dataset con su tipo de variable y el conteo de valores por atributo.

Efectivamente se ratifica que disponemos únicamente de un atributo categórico, *diagnosis*, y además que existe un atributo denominado *Unnamed: 32* con ningún valor y que por lo tanto, puede ser considerado como un atributo no significativo o erróneo. De esta manera se procede a la eliminación del atributo *Unnamed: 32* y se recodifica el atributo *diagnosis* para convertirlo de categórico a cuantitativo. En este caso, los posibles valores son *M* y *B* correspondientes a *Malign* y *Benign* en inglés, por lo que se escoge para la recodificación 1 y 0, siendo 1 maligno y 0 benigno.

```
Data columns (total 32 columns):
diagnosis                  569 non-null object
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
Unnamed: 32                0 non-null float64
dtypes: float64(31), object(1)
memory usage: 146.7+ KB
```

Figura 24: Salida del método *data.info()* de python

Así pues, el dataset después de esta primera toma de contacto quedaría tal y como se muestra en la figura 25.

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
id								
842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010
842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690
84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740
84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140
84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800
843786	1	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780
844359	1	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270
84458202	1	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366
844981	1	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590
84501001	1	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730

10 rows × 31 columns

Figura 25: Dataset después del primer análisis exploratorio

A continuación se realizará una exploración más profunda sobre los datos para poder extraer información no redundante. Para ello, en primer lugar, conviene saber de cuántos casos de tumores malignos y benignos se dispone, lo cual ser representado tal y como se muestra en la figura 26. Se aprecia que existen más casos con tumores benignos que malignos. Ahora podría ser interesante

visualizar la distribución de los atributos en relación con el diagnóstico, y tratar de comprobar si pudiera existir algún patrón visible. En la figura 27 se muestra la categoría de atributos *worst*, representada en función de los restantes atributos que conforman esta misma categoría, y con las gráficas de densidad dibujadas en la diagonal principal.

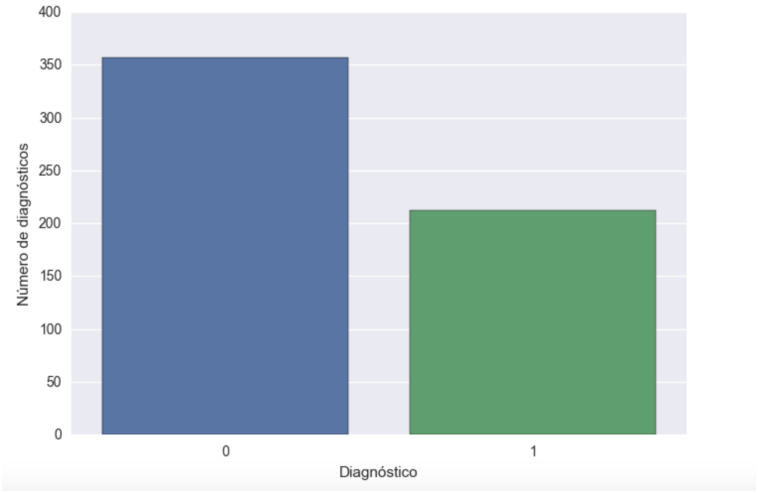

Figura 26: Diagnóstico de tumor maligno/benigno.

De la figura 27 se puede deducir que el radio, el perímetro y el área son atributos bastante discriminatorios para el diagnóstico como se podía intuir de una manera inicial (basta observar la diferencia entre las gráficas de densidad). Parece haber también alguna relación en un menor nivel con los atributos de dimensión fractal y concavidad. Por otra parte, la textura y la suavidad de la forma tumoral no parecen ser relevantes.

Otra manera de comprobar cómo se relacionan las variables entre sí, es analizando las correlaciones. En el gráfico de la figura 28 se representan las correlaciones de los atributos. Se puede comprobar que existe una fuerte relación entre los atributos de radio, perímetro y área así como una correlación negativa entre los atributos de dimensión fractal y suavidad de la forma tumoral. Se corrobora así la información extraída de la figura anterior.

4.1.2. Evaluación de modelos

Una vez se dispone de los datos tratados y preparados para ser introducidos al motor de Machine Learning, se debe separar todos los atributos de la variable resultado, como bien se podría representar matemáticamente según

$$y = f(x), \tag{38}$$

siendo y la variable resultado que se pretende predecir, y x la información en la que se va a basar el sistema para conseguir un buen resultado. Dependiendo de la naturaleza de nuestro conjunto de datos, se deberá realizar una partición de los datos o no. En el dataset utilizado, no ha habido ninguna preparación previa para el uso de Machine Learning, es decir, únicamente se dispone de un conjunto de datos homógeneo. Asímismo, se deberá realizar una partición de los datos disponibles para poder utilizar como training para entrenar el modelo y test para poder evaluar la actuación

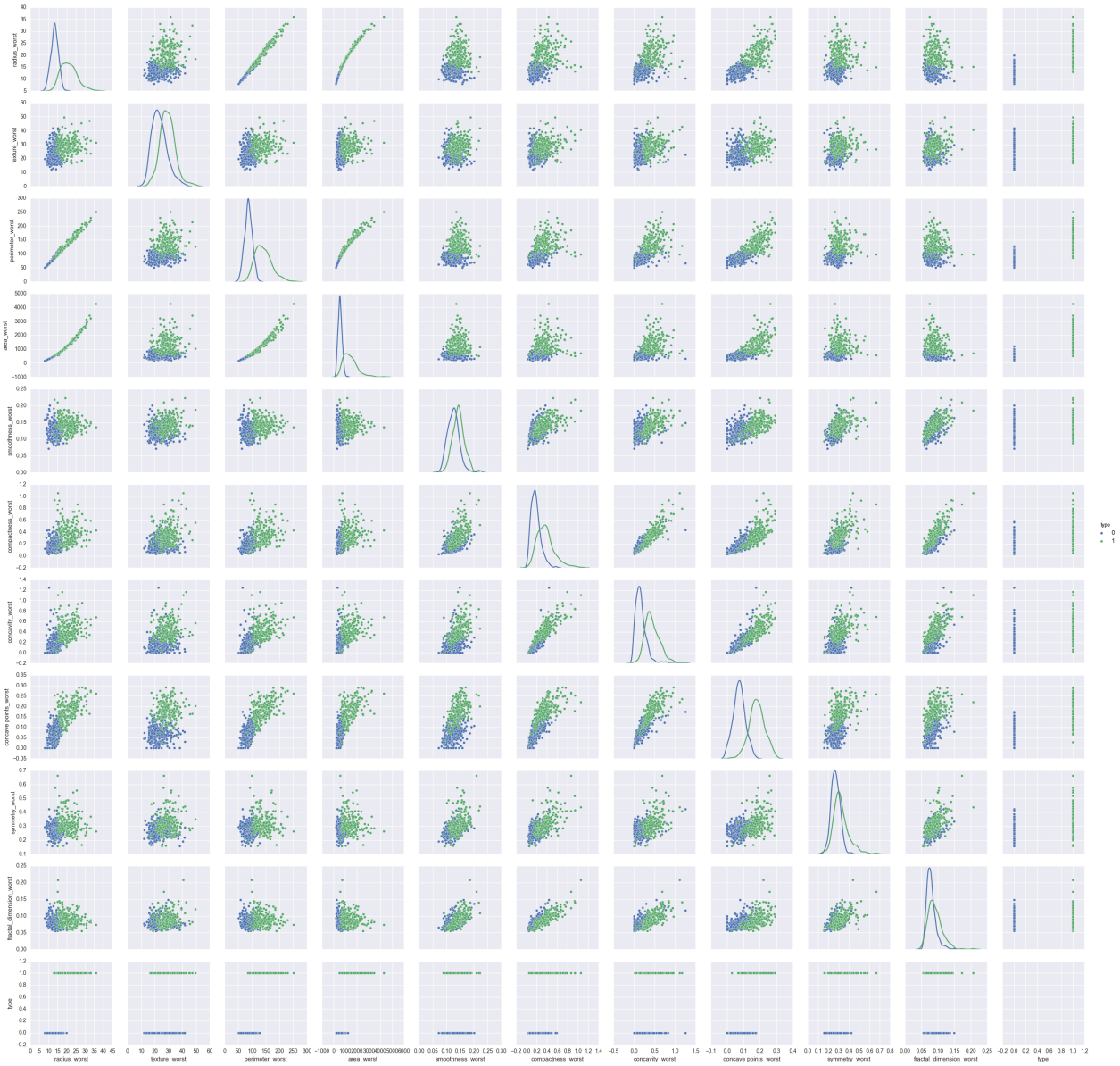

Figura 27: Cada atributo *worst* representado en función del resto de atributos distinguidos por diagnóstico.

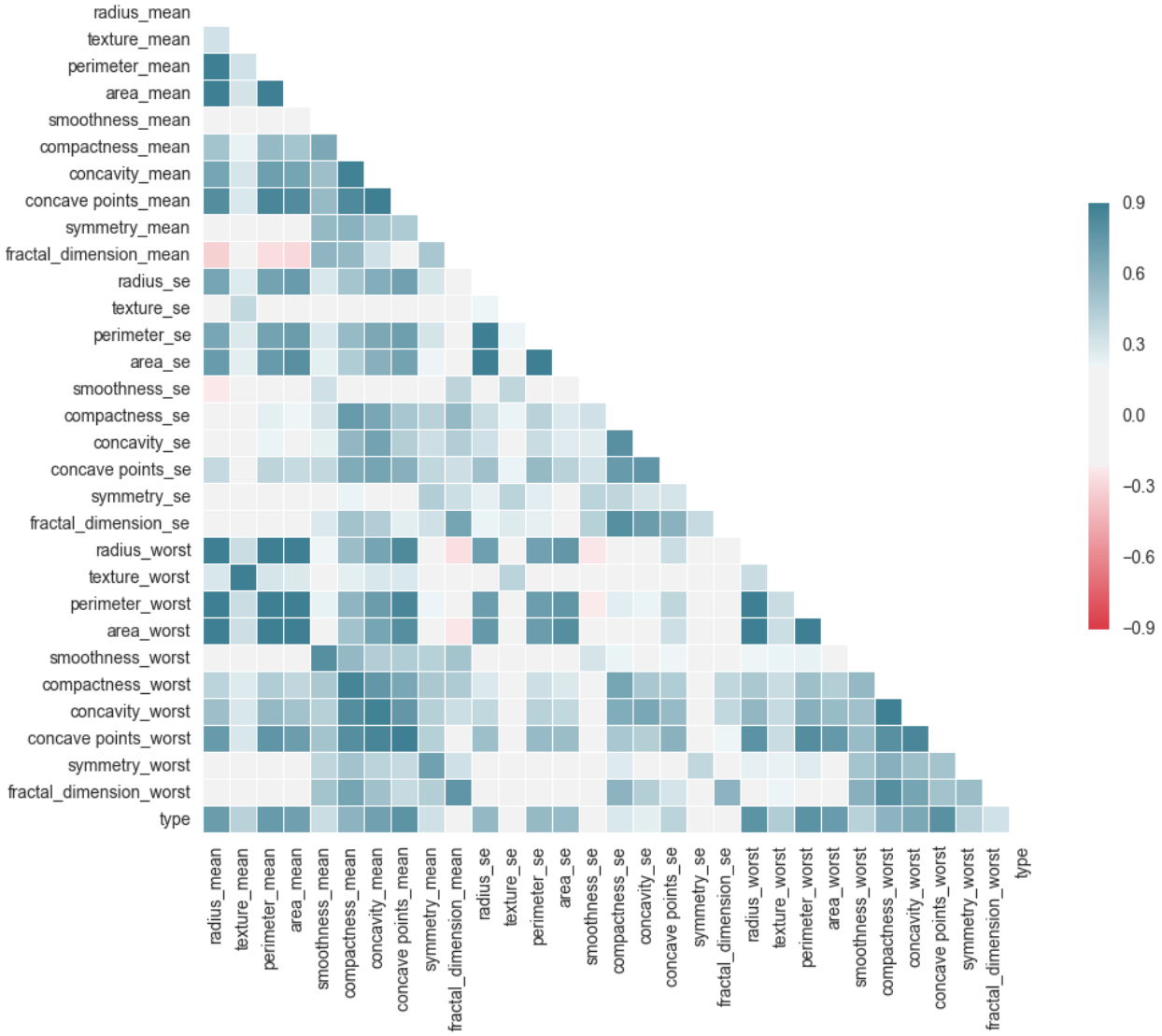

Figura 28: Gráfico de correlaciones.

del mismo sobre la predicción. Entonces, se podría decir que se ha dividido el conjunto de datos en cuatro porciones:

1. x de training.
2. x de test.
3. y de training.
4. y de test.

Consecuentemente, el proceso que se sigue para realizar el training se podría expresar como

$$y_{training} = f(x_{training}), \tag{39}$$

donde el modelo escogido aproximará la función $f()$ según la pareja de datos que se le han introducido $(x_{training}, y_{training})$. Para conocer el nivel de error que se comete y así evaluar la calidad del modelo, se le introduce x_{test} y se obtiene una predicción $y_{predicha}$ del mismo tamaño que x_{test},

$$y_{predicha} = f_{entrenada}(x_{test}). \tag{40}$$

Una vez obtenida $y_{predicha}$, se puede evaluar dicho resultado con los resultados verídicos y_{test}.

Todo este proceso resumido a grandes rasgos, se realiza mediante los comandos en Python *model.fit(x_{train}, y_{train})* y *model.predict(x_{test})*, donde *model* es el algoritmo escogido de la librería de Python de *Scikit-learn*.

Se han escogido tres algoritmos de entre los más comunes en Machine Learning representados en la figura 13 del apartado 2.3. Así, podremos contrastar resultados y seleccionar el óptimo de ellos. Los algoritmos son: *Gaussian Naïve-Bayes*, regresión logística y *Support Vector Machines* o SVM. En una primera instancia, para la elaboración de un modelo base que pueda ser usado como referencia, se ha utilizado un regresor lineal basado en *Gaussian Naïve-Bayes*. Este algoritmo es uno de los más simples del ámbito y proporciona predicciones suficientemente robustas para datos que siguen distribuciones gaussianas [27]. Dado que, como se ha visto anteriormente, disponemos de datos con dicha distribución, cabe esperar un resultado con bastante consistencia en la predicción. Basándose a continuación más específicamente en la variable resultado que se busca, se ha decidido utilizar la regresión logística debido a su adecuación a la variable resultado de naturaleza binaria. La regresión logística provee resultados muy certeros en el caso en el que sólo exista un único límite discriminatorio, hecho que se cumple en la detección. Además, tiene un bajo coste computacional. Por último, también podría realizar una buena actuación sobre la base de datos un modelo basado en SVM, algoritmo bastante potente para discriminar entre grupos difusos aunque muy costoso computacionalmente. No obstante, no debería haber gran impacto en la latencia debido al tamaño más bien pequeño de este conjunto de datos. Estos tres algoritmos se encuentran explicados de una manera más extensa en el apartado 2.3 de este mismo documento.

Por eficiencia de código y comodidad del programador, se ha declarado una función en Python para poder aplicar dichos algoritmos. Dados los parámetros de entrada: modelo escogido, x_{train} e y_{train}, se entrenará al modelo, se realizará una predicción sobre los datos y finalmente se calculará la precisión media y el tiempo de cómputo. Además, para evitar un sobreentrenamiento del modelo y no obtener resultados distorsionados, es decir, para evitar el problema del *overfitting* [28] [29] y

disponer a su vez de una precisión más robusta, también se ha decidido añadir a la precisión media una validación cruzada de 6 pliegues. Esto quiere decir que de la muestra disponible de datos de la que se dispone, se generarán 6 réplicas. En cada una de estas réplicas, se mezclarán aleatoriamente las instancias y se irán alternando las muestras que se toman como entrenamiento y test, promediando cada resultado con el anterior. Para cada uno de los pliegues, se verá por pantalla el promedio de precisión que se lleva hasta dicho pliegue, siendo de esta manera el último pliegue el que otorgue más representatividad al ser el promediado de él mismo y de todos los anteriores.

En las siguientes figuras se puede observar la llamada a la función así como los respectivos resultados obtenidos.

```
applyingmodel(GaussianNB(),X,Y)
```

```
Precisión media : 94.200%
Tiempo de cómputo (s): 0.003
Precisión por validación cruzada : 88.421%
Precisión por validación cruzada : 88.947%
Precisión por validación cruzada : 91.228%
Precisión por validación cruzada : 92.895%
Precisión por validación cruzada : 93.474%
Precisión por validación cruzada : 93.852%
```

Figura 29: Resultados del modelo *Gaussian Naïve-Bayes*.

```
applyingmodel(LogisticRegression(),X,Y)
```

```
Precisión media : 95.958%
Tiempo de cómputo (s): 0.008
Precisión por validación cruzada : 89.474%
Precisión por validación cruzada : 93.158%
Precisión por validación cruzada : 94.035%
Precisión por validación cruzada : 94.737%
Precisión por validación cruzada : 94.947%
Precisión por validación cruzada : 95.080%
```

Figura 30: Resultados del modelo de regresión logística.

```
applyingmodel(svc,X,Y)
```

```
Precisión media : 95.958%
Tiempo de cómputo (s): 1.321
Precisión por validación cruzada : 89.474%
Precisión por validación cruzada : 92.632%
Precisión por validación cruzada : 93.684%
Precisión por validación cruzada : 95.000%
Precisión por validación cruzada : 94.947%
Precisión por validación cruzada : 94.903%
```

Figura 31: Resultados del modelo de SVM.

En primer lugar, en la figura 29, se contemplan los resultados correspondientes al modelo bayesiano. Como esperábamos, ha proporcionado un resultado bastante consistente para ser uno de los modelos más simples con un $94,200\,\%$ de precisión media. El primer pliegue proporciona un $89,421\,\%$ de precisión, que sería la precisión más restrictiva al contar únicamente con $1/6$ de la información disponible, hasta llegar finalmente a un $93,852\,\%$. Prosiguiendo con la siguiente figura, la 30, analizamos los resultados de la regresión logística. Se observa que hay una mejoría de $1,758\,\%$ respecto al modelo bayesiano. Puede parecer una cantidad pequeña, pero manejando porcentajes tan altos, dicha cantidad es significativa por la dificultad de mejora. Este modelo parte aproxima-

damente del mismo porcentaje de aciertos en el primer pliegue, un $89,474\,\%$, aunque se ve mayor mejoría con la evolución de la validación cruzada, hasta llegar a un $95,080\,\%$ de precisión. Respecto al tiempo de cómputo, se ejecuta en aproximadamente el triple de tiempo que el modelo *Gaussian Naïve-Bayes*. Por último, en la figura 31 se representan los resultados del modelo basado en SVM (SVC correspondiente a *Support Vector Classifier*, nombre otorgado por la propia librería utilizada). El resultado de precisión media es idéntico a la regresión logística, es decir, han acertado y fallado la misma cantidad de casos respectivamente aunque si nos fijamos en la validación cruzada, podemos observar una pequeña diferencia. Se parte en el primer pliegue de un $89,474\,\%$ pero llega sólo hasta un $94,904\,\%$, valor por debajo del valor final de la regresión logística aunque sigue siendo mejor que el modelo de Bayes. Sin embargo, donde se nota realmente la diferencia en este caso, es en el tiempo de cómputo. Consume una gran cantidad de recursos de hardware en comparación con los otros dos algoritmos, introduciendo una latencia de $1,321s$, valor que sí que es perceptible para el analista en comparación a los $0,008s$ y $0,003s$ de la regresión logística y Bayes respectivamente.

A pesar de lo aparentemente buenos resultados que se han obtenido, pueden no ser lo suficientemente buenos para una aplicación de estas características en la que el coste de un falso positivo o un falso negativo es muy alto. Además, debemos considerar que en caso de aplicación real muy probablemente se disponga de una base de datos con miles de pacientes, y no siempre más datos equivalen a mejor resultado. Por ello, se ha decidido realizar como téncica de reducción de dimensionalidad un análisis de componentes principales para comprobar si podemos mejorar el sistema y a su vez, mitigar problemas de dimensionalidad.

4.1.3. Evaluación de modelos con PCA

Como bien se ha comentado la sección 3.2 de este mismo documento, el análisis de componentes principales es usado por una parte para reducir la dimensionalidad del conjunto de datos intentando perder la menor cantidad de información posible, y a la par, buscar las componentes que maximizan las varianzas en un subespacio de atributos diferente. Buscamos maximizar varianzas porque de esta manera nos aseguramos de que el sistema se queda con la mayor cantidad de información significativa e ignora la menos relevante. Esto resulta de gran utilidad cuando se manejan bases de datos del orden de Terabytes y Petabytes, puesto que aunque el tiempo de cómputo del algoritmo fuese ínfimo, se vería impactado en latencia por dicha cantidad de datos a procesar y mucho peor aún, también podría introducir más ruido al sistema generando peores predicciones [30].

Para evitar todos estos factores negativos aplicando análisis de componentes princpales, es importante volver a las cuatro particiones de las que disponíamos:

1. x de training.
2. x de test.
3. y de training.
4. y de test.

Como conclusión de (38), (39) y (40), podría decirse que siempre se abarca el problema empleando x_{train} y x_{test} como variables de trabajo mientras que y_{train} e y_{test} actúan como resultado.

En consecuencia, se aplicará únicamente PCA a las x y nunca a las variables resultado, ya que de lo contrario, se perderían. Seguidamente, se buscará un límite para establecer qué número de componentes es el óptimo y poder reducir dimensionalidad.

De nuevo, por inteligibilidad de código, de formato y por comodidad del propio programador, se ha elaborado una función que dados los datos de entrada y el número de componentes que se desea, genera una copia del conjunto de datos en el nuevo subespacio de PCA. Por tanto, es necesario conocer el número de componentes principales óptimo mediante algunos de los métodos que se pueden encontrar en el apartado 3.2 de este documento. Por simplicidad y por el atractivo gráfico, se ha escogido realizar el método del gráfico de sedimentación o gráfico del codo. Para ello, ha de representarse las varianzas de cada una de las componentes principales que forman el conjunto de datos y comprobar a partir de cuál de ellas puede establecerse el umbral adecuado para reducir la dimensionalidad. Seguidamente, se representará la varianza de las componentes principales para identificar una variación significativa para establecer un umbral.

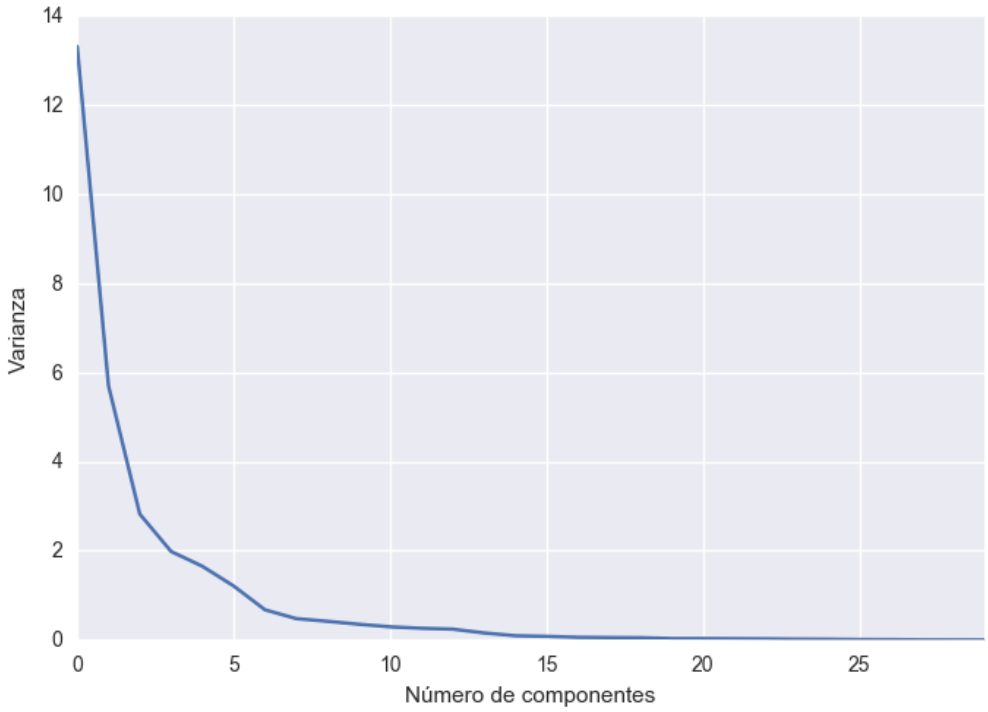

Figura 32: Gráfico de sedimentación o método del codo.

Se aprecia en la figura 32, que a partir de la sexta componente prácticamente no hay gran cantidad de información comparada con las primeras. Así pues, se establece 6 como el número de componentes seleccionadas en este caso. Por tanto, se ignoran las 24 componentes restantes. Según el marco teórico, quedándonos únicamente con esas 6 componentes deberíamos obtener al menos unos resultados igual de robustos que los modelos sin PCA.

```
DF_PCA(X,6).head(10)
```

	0	1	2	3	4	5
0	9.192837	1.948583	-1.123166	3.633731	-1.195110	1.411424
1	2.387802	-3.768172	-0.529293	1.118264	0.621775	0.028656
2	5.733896	-1.075174	-0.551748	0.912083	-0.177086	0.541452
3	7.122953	10.275589	-3.232790	0.152547	-2.960878	3.053422
4	3.935302	-1.948072	1.389767	2.940639	0.546747	-1.226495
5	2.380247	3.949929	-2.934877	0.941037	-1.056042	-0.451039
6	2.238883	-2.690031	-1.639913	0.149340	0.040360	-0.128948
7	2.143299	2.340244	-0.871947	-0.127043	-1.427437	-1.257039
8	3.174924	3.391813	-3.119986	-0.601297	-1.522290	0.559545
9	6.351747	7.727174	-4.341916	-3.375202	1.710263	-0.723909

Figura 33: Muestra de las 10 primeras instancias del conjunto de datos con PCA.

En la figura 33, se observa cómo ha quedado nuesto nuevo conjunto de datos una vez aplicado PCA y seleccionando únicamente las 6 primeras componentes. Uno de los puntos débiles de esta técnica es la interpretabilidad de los datos con los que se trabaja, puesto que las nuevas variables son una combinación lineal de las anteriores pero no identificables en este punto. No obstante, estas seis componentes resultan ser más significativas que el conjunto de datos anterior, aunque no facilitan el entendimiento objetivo de los atributos.

Análogamente al apartado anterior donde no se ha aplicado PCA, se procede a probar de nuevo los tres mismos algoritmos pero esta vez introduciendo el nuevo conjunto de datos con 6 componentes principales. A continuación, se muestran las llamadas a las funciones y los resultados de los respectivos modelos.

```
applyingmodel(GaussianNB(),ndf,Y)
```

```
Precisión media : 95.402%
Tiempo de cómputo (s): 0.002
Precisión por validación cruzada : 93.756%
Precisión por validación cruzada : 93.877%
Precisión por validación cruzada : 94.051%
Precisión por validación cruzada : 94.364%
Precisión por validación cruzada : 94.625%
Precisión por validación cruzada : 95.102%
```

Figura 34: Resultados del modelo *Gaussian Naïve-Bayes* con PCA.

```
applyingmodel(LogisticRegression(),ndf,Y)
```

```
Precisión media : 98.594%
Tiempo de cómputo (s): 0.004
Precisión por validación cruzada : 97.895%
Precisión por validación cruzada : 97.895%
Precisión por validación cruzada : 97.544%
Precisión por validación cruzada : 97.632%
Precisión por validación cruzada : 97.895%
Precisión por validación cruzada : 97.891%
```

Figura 35: Resultados del modelo de regresión logística con PCA.

```
applyingmodel(svc,ndf,Y)
```

```
Precisión media : 97.188%
Tiempo de cómputo (s): 0.014
Precisión por validación cruzada : 94.737%
Precisión por validación cruzada : 95.263%
Precisión por validación cruzada : 95.088%
Precisión por validación cruzada : 96.053%
Precisión por validación cruzada : 96.421%
Precisión por validación cruzada : 96.486%
```

Figura 36: Resultados del modelo de SVM con PCA.

Comenzando con por la figura 34 relativa al modelo *Gaussian Naïve-Bayes*, se observa una precisión media de un $95,402\,\%$. Dicha precisión representa una mejoría de un $1,202\,\%$ respecto al modelo sin PCA. El tiempo de cómputo se rebaja pero mínimamente, debido a que el algoritmo ya es de por sí bastante sencillo de operar. La validación cruzada otorga un $95,102\,\%$ de precisión, como se ha estado viendo hasta ahora, siempre más restrictiva. Continuando con la siguiente figura 35, se obtiene una precisión media de un $98,594\,\%$ para la regresión logística, precisión que mejora un $2,636\,\%$ respecto al modelo sin reducción de dimensionalidad, cifra bastante alta teniendo en cuenta los valores de precisión entre los que nos estamos moviendo.En términos de validación cruzada, se obtiene un $97,891\,\%$, cifra que dista de los resultados hasta ahora provistos por el modelo bayesiano. Por último, en la figura 36, se puede observar una precisión media de $97,188\,\%$ para el SVM. Porcentaje por detrás del correspondiente a la regresión logística. Este último modelo dispone de una precisión por validación cruzada de $96,486\,\%$.

Se puede afirmar que para los tres modelos, el *Gaussian Naïve-Bayes*, la regresión logística y el SVM, las seis componentes principales han resultado ser más significativas como se esperaba desde un primer momento. No obstante, dicha mejoría ha sido mayor para la regresión logística, posicionándose como algoritmo estrella para la implementación de esta aplicación por precisión y tiempo de cómputo. A pesar de escoger la regresión logística en este caso, hay que hacer especial mención en la mejora también significativa en el modelo SVM, por encima de todo, en la reducción de tiempo de cómputo donde se ha llegado a reducir un $98,9\,\%$.

4.1.4. Resultados

Tal y como se ha ido comentando en los apartados anteriores para la casuística de detección, se puede afirmar que con un tratamiento de datos bastante sencillo y sin aplicar reducción de dimensionalidad, el algoritmo que mejor actúa sobre esta base de datos es la regresión logística. Por poca diferencia nada más que por eficiencia computacional, le seguiría el modelo con SVM. Sin embargo, el modelo bayesiano dista más de estos dos quedando pues en la última posición de nuestras opciones. Contemplando a continuación los modelos con PCA, se nota una mejoría en la regresión logística y en el SVM. Las mejorías son notables en ambos algoritmos, aunque sigue quedando por delante la regresión logística. Por otra parte, el modelo bayesiano se ve impactado negativamente por la transformación de variables que aplica PCA.

		Gaussian Naïve-Bayes		Logistic Regression		Support vector machines	
Sin PCA	Precisión media	94,200 %		95,958 %		95,958 %	
	Validación cruzada	88,421 %	93,852 %	89,474 %	95,080 %	89,474 %	94,903 %
	Tiempo de cómputo	0,003s		0,011s		1,321s	
Con PCA	Precisión media	95,402 %		98,594 %		97,188 %	
	Validación cruzada	93,756 %	95,102 %	97,895 %	97,891 %	94,737 %	96,486 %
	Tiempo de cómputo	0,002s		0,004s		0,014s	

Cuadro 1: Tabla de resultados de diagnóstico.

En la tabla 1 se recopila de manera comparativa todos los resultados obtenidos teniendo en cuenta: Precisión media, los valores mínimo y máximo de la validación cruzada y el tiempo exclusivo de cómputo de la predicción para cada caso visto. Con los datos presentados se puede afirmar que el mejor de los modelos contemplados ha resultado ser la regresión logística con PCA, contando con un tiempo de ejecución de $0,010s$ y una precisión por validación cruzada final de $97,189\,\%$. También se confirma que a pesar de no ser el mejor modelo, PCA impacta de manera muy positiva a algoritmos más complejos como SVM, disminuyendo el tiempo de cómputo considerablemente y aún mejorando predicción con respecto al modelo sin PCA.

4.2. Clasificación

El modelo de Machine Learning para la clasificación de subtipos de cáncer de mama, hará uso de la base de datos *Breast Cancer Proteomes Dataset* comentado anteriormente en el apartado 1.3.2 Fuente de datos. Este conjunto de datos está formado por 77 pacientes diagnosticadas de cáncer de mama, cada paciente con información de la presencia de 12553 proteínas presentes en una muestra de tejido tumoral. En este tipo de problema, el enfoque es diferente al presentado anterioremnte en el diagnóstico: en este caso no se dispone de una variable concreta que se desee predecir, sino un conjunto enorme de datos de los que se desea extraer información o patrones para poder emplear de cara a nuevas muestras. Esta casuística corresponde a un problema de *Unsupervised Learning*.

La metodología más común en *Unsupervised Learning* para el descubrimiento de patrones se denomina *Clustering*. Incluye un conjunto heterogéneo de técnicas cuyo objetivo es identificar varios sub-grupos entre la muestra de datos facilitada. Dichos sub-grupos son denominados clústers y son utilizados para estudiar similitudes comunes entre muestras. Son de gran utilidad para Machine Learning no supervisado y minería de datos. Se empleará una técnica de clústering jerárquico en este apartado para poder comprobar a diversos niveles el conjunto de clústers (todos relacionados entre sí). Esta técnica específica también es conocida como dendrograma.

Con esta aplicación podría descubrirse qué proteínas tienen mayor representatividad entre las pacientes con tumores malignos y cómo se relacionan entre sí y entonces, sería posible discernir patrones genéticos para prever la predisposición de una paciente a poder desarrollar un tipo específico de cáncer de mama y poder ofrecer un tratamiento más efectivo en etapas más tempranas. Se usarán tres tablas disponibles dentro de la base de datos *Breast Cancer Proteomes Dataset* correpondientes a datos clínicos de las pacientes, datos proteómicos de las pacientes y una base de datos que contiene la relación proteína-gen del método PAM50. Se realizará un pre-procesado sobre el conjunto de datos proteómicos y se realizarán las operaciones necesarias para disponer de un conjunto de datos listo

para ser estudiado. Por último, se contrastarán las deducciones obtenidas con el mismo modelo aplicando el mismo modelo únicamente con los datos contemplados en PAM50.

4.2.1. Implementación

El primer paso, de nuevo, vuelve a ser una incursión en la base de datos *Breast Cancer Proteomes dataset* para contemplar con qué variables se va a trabajar, así como su naturaleza.

	AO-A12D	C8-A131	AO-A12B	BH-A18Q	C8-A130	C8-A138	E2-A154	C8-A12L	A2-A0EX	AO-A12D	...
RefSeqProteinID											
NP_958782	1.096131	2.609943	-0.659828	0.195341	-0.494060	2.765081	0.862659	1.407570	1.185108	1.100688	...
NP_958785	1.111370	2.650422	-0.648742	0.215413	-0.503899	2.779709	0.870186	1.407570	1.192612	1.100688	...
NP_958786	1.111370	2.650422	-0.654285	0.215413	-0.500619	2.779709	0.870186	1.410312	1.188860	1.100688	...
NP_000436	1.107561	2.646374	-0.632113	0.205377	-0.510459	2.797995	0.866423	1.407570	1.185108	1.100688	...
NP_958781	1.115180	2.646374	-0.640428	0.215413	-0.503899	2.787023	0.870186	1.413053	1.200116	1.093358	...
NP_958780	1.107561	2.646374	-0.654285	0.215413	-0.503899	2.779709	0.870186	1.407570	1.188860	1.097023	...
NP_958783	1.111370	2.650422	-0.648742	0.215413	-0.500619	2.783366	0.870186	1.410312	1.188860	1.097023	...
NP_958784	1.111370	2.650422	-0.648742	0.215413	-0.500619	2.783366	0.870186	1.413053	1.192612	1.097023	...
NP_112598	-1.517390	3.909313	-0.618256	-1.035760	-1.845366	2.205538	1.920171	3.195070	1.046289	-2.413909	...
NP_001611	0.482754	-1.045294	1.222003	-0.517226	-0.405503	0.749997	2.349197	-0.007077	2.138081	0.543630	...

10 rows × 83 columns

Figura 37: Base de datos *Breast Cancer Proteomes Dataset.*

En la figura 37 se observa el conjunto de datos. Se dispone del código de paciente en las columnas mientras que las proteínas están indexadas en las filas. Esta casuística formal deberá de cambiarse debido a que los atributos siempre deben ir en las columnas mientras que los sujetos o instancias en las filas. Por tanto, ya se debe realizar una operación formal sobre la base de datos. A continuación, convendría comprobar si tenemos datos ausentes o no, tarea realizable observando la salida del comando *.describe()* de Python como se representa en la figura 38.

	AO-A12D	C8-A131	AO-A12B	BH-A18Q	C8-A130	C8-A138	E2-A154	C8-A12L	A2-A0EX
count	11334.000000	11335.000000	11334.000000	12024.000000	12025.000000	11833.000000	11833.000000	11732.000000	11732.000000
mean	0.133079	0.128822	-0.440764	-0.729233	-0.039436	0.115353	-0.618354	-0.305320	-0.021228
std	1.734231	1.579845	1.634930	2.422452	1.353505	1.652918	2.281818	1.567687	1.651917
min	-12.466750	-13.156553	-9.911008	-24.553794	-15.004199	-12.956961	-16.640848	-18.710032	-14.711526
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	17.623036	12.677023	8.289261	11.790434	6.925004	10.602330	10.079191	8.804312	9.548006

8 rows × 83 columns

Figura 38: Salida del comando *.describe()* de Python sobre el conjunto de datos proteómico.

Tal y como se aprecia en esta última figura, tenemos diferentes valores *count* para cada paciente así como valores NaN para los percentiles, hecho que indica que sí que existen datos ausentes. Entonces, se debe escoger una técnica de asignación de datos para poder continuar con la aplicación. En esta aplicación, por la documentación proveída por el distribuidor de los datos, los NaN existentes son valores tan ínfimos que son considerados como tal. Entonces, pueden considerarse, y se asignan como "0".

Por otra parte, este conjunto de datos dispone de otra tabla en la que se dispone información clínica de cada una de las pacientes, tabla que contiene el atributo sobre el que se intenta buscar un patrón: Tipo de cáncer de mama. Se puede apreciar dicha tabla en la figura 39.

	Complete TCGA ID	Gender	Age at Initial Pathologic Diagnosis	ER Status	PR Status	HER2 Final Status	Tumor	Tumor--T1 Coded	Node	Node-Coded	...	SigClust Unsupervised mRNA	SigClust Intrinsic mRNA	miRNA Clusters	methylation Clusters
0	TCGA-A2-A0T2	FEMALE	66	Negative	Negative	Negative	T3	T_Other	N3	Positive	...	0	-13	3	5
1	TCGA-A2-A0CM	FEMALE	40	Negative	Negative	Negative	T2	T_Other	N0	Negative	...	-12	-13	4	4
2	TCGA-BH-A18V	FEMALE	48	Negative	Negative	Negative	T2	T_Other	N1	Positive	...	-12	-13	5	5
3	TCGA-BH-A18Q	FEMALE	56	Negative	Negative	Negative	T2	T_Other	N1	Positive	...	-12	-13	5	5
4	TCGA-BH-A0E0	FEMALE	38	Negative	Negative	Negative	T3	T_Other	N3	Positive	...	0	-13	5	5

5 rows × 31 columns

Figura 39: Tabla de datos clínicos de *Breast Cancer Proteomes Dataset*.

Las columnas que conforman la tabla de datos clínicos se puede averigüar ejecutando el comando *.info()* y se obtienen los siguientes parámetros representados en la figura 40

```
Data columns (total 31 columns):
Complete TCGA ID                          105 non-null object
Gender                                    105 non-null object
Age at Initial Pathologic Diagnosis       105 non-null int64
ER Status                                 105 non-null object
PR Status                                 105 non-null object
HER2 Final Status                         105 non-null object
Tumor                                     105 non-null object
Tumor--T1 Coded                           105 non-null object
Node                                      105 non-null object
Node-Coded                                105 non-null object
Metastasis                                105 non-null object
Metastasis-Coded                          105 non-null object
AJCC Stage                                105 non-null object
Converted Stage                           105 non-null object
Survival Data Form                        105 non-null object
Vital Status                              105 non-null object
Days to Date of Last Contact              105 non-null int64
Days to date of Death                     11 non-null float64
OS event                                  105 non-null int64
OS Time                                   105 non-null int64
PAM50 mRNA                                105 non-null object
SigClust Unsupervised mRNA                105 non-null int64
SigClust Intrinsic mRNA                   105 non-null int64
miRNA Clusters                            105 non-null int64
methylation Clusters                      105 non-null int64
RPPA Clusters                             105 non-null object
CN Clusters                               105 non-null int64
Integrated Clusters (with PAM50)          105 non-null int64
Integrated Clusters (no exp)              105 non-null int64
Integrated Clusters (unsup exp)           105 non-null int64
Patient                                   105 non-null object
```

Figura 40: Atributos de la tabla de datos clínicos.

En este caso, como se desea encontrar patrones para el diagnóstico certero de los diferentes tipos de cáncer, conviene juntar la tabla de datos proteómicos con el indicativo de tipo de cáncer que ha sido diagnosticado a cada paciente: "*PAM50 mRNA*". Se puede realizar un gráfico para contemplar cuántos casos de cada tipo existen entre los datos como se representa en la figura 41.

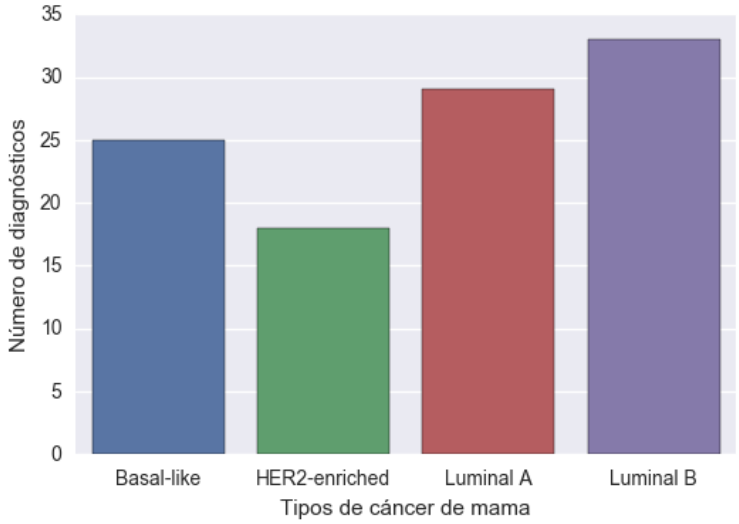

Figura 41: Conteo de los diferentes diagnósticos entre las pacientes.

Existen entonces cuatro tipos de cáncer contemplados en el conjunto de datos en el que se está trabajando: *Basal-like*, *HER2-enriched*, *Luminal A* y *Luminal B*. Además, se observa que los más numerosos son los de tipo *Luminal*.

Para extraer patrones basándose en el diagnóstico del tipo de cáncer, se necesitará el indicador *Diagnosis* en conjunto con todos los datos de las proteínas. Para añadirlo al conjunto de datos, se debe hacer una unión entre las tablas de datos clínicos y la de datos proteómicos, indexando

por paciente. Para ello, se hace uso del comando de Python *merge()* donde se cruzará el atributo *PAM50 mRNA* (atributo que corresponde al diagnóstico) de los datos clínicos con la tabla de datos proteómicos por paciente, obteniendo como resultado la figura 42 donde la columna *Diagnosis* es el atributo *PAM50 mRNA* con el nombre cambiado.

	NP_958782	NP_958785	NP_958786	NP_000436	NP_958781	NP_958780	NP_958783
Patient							
AO-A12D	1.096131	1.111370	1.111370	1.107561	1.115180	1.107561	1.111370
AO-A12D	1.100688	1.100688	1.100688	1.100688	1.093358	1.097023	1.097023
C8-A131	2.609943	2.650422	2.650422	2.646374	2.646374	2.646374	2.650422
C8-A131	2.707250	2.733832	2.737629	2.733832	2.752819	2.737629	2.737629
AO-A12B	-0.659828	-0.648742	-0.654285	-0.632113	-0.640428	-0.654285	-0.648742

NP_003593	NP_997203	NP_001191293	NP_775791	NP_004065	NP_068752	NP_219494	Diagnosis
-0.340163	0.0	0.000000	0.0	0.000000	-0.633517	12.666488	1
0.129501	0.0	-2.578828	0.0	1.294925	-0.189341	13.066445	1
3.451902	0.0	0.000000	0.0	0.000000	4.840325	0.140736	0
3.352807	0.0	-2.130632	0.0	0.000000	2.027516	0.000000	0
-1.718531	0.0	0.000000	0.0	0.000000	-1.965192	-2.854835	3

Figura 42: Base de datos cruzada.

Una vez llegados a este punto, ya tenemos el conjunto de datos listo para poder ser explotado e intentar distinguir algún patrón.

4.2.2. Resultados

En el área específica de clustering y *Knowledge Discovery* de Machine Learning, la metodología y técnicas de trabajo cambian. Buscamos relaciones significativas entre los atributos para poder comprender mejor la naturaleza de la enfermedad y poder diagnosticar qué tipo de cáncer desarrolla en función de la presencia o ausencia de un conjunto de proteínas. Entonces no se dispone de un atributo específico que sea la variable resultado, sino que se busca un entendimiento sobre el problema.

Esta nueva casuística ha de seguir una metodología diferente a la de un problema de Machine Learning supervisado, en este caso ya no es necesario partir el conjunto de datos en x_{train}, x_{test} e y_{train}, y_{test} como se hacía en el caso anterior. Aquí, se utilizará todos los datos íntegros para intentar identificar patrones.

La mejor manera de poder mostrar la representatividad de cada proteína con su correspondiente diagnóstico es realizar un *heatmap*. Este término se utiliza para dar nombre a la representación gráfica de datos donde los valores de una matriz son representados como colores. Sobre dicho gráfico, adicionalmente, se va a realizar un clustering jerárquico de variables para comprobar de cuántos grupos mayoritarios se disponen. A pesar de saber que existen cuatro tipos de cáncer contemplados en los datos, vamos a comprobarlo e intentar ver si efectivamente tienen un patrón genético similar.

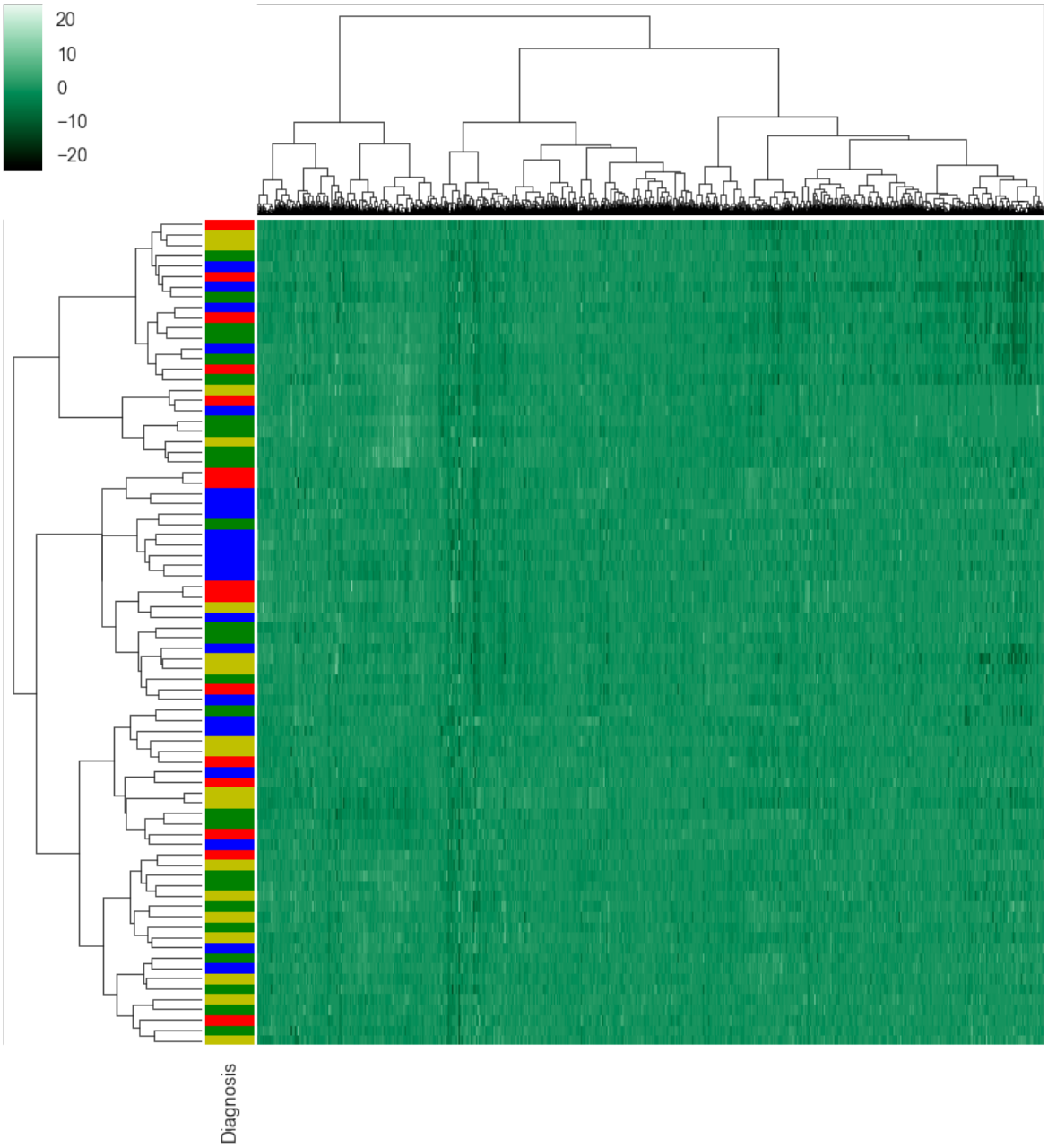

Figura 43: *Heatmap* con clustering jerárquico.

En la figura 43 se puede observar el *heatmap* de la base de datos, donde el eje horizontal son las aproximadamente 12000 proteínas y el eje vertical, las pacientes con el diagnóstico de tipo de cáncer de cada una de ellas. Los colores azul, rojo, verde y amarillo corresponden respectivamente a los tipos de cáncer *Basal-like*, *HER2-enriched*, *Luminal A* y *Luminal B*. Las diferentes intensidades de verde representan la mayor o menor presencia de dicha proteína en cada paciente, siendo los verdes más oscuros los valores más grandes negativos y los verdes más blanquecinos, los valores más grandes positivos. De esa manera, visualizando el gráfico podemos obtener una idea global

de qué proteínas tienen en común qué pacientes. Estableciendo como referencia base la primera ramificación, podemos observar que después de las dos ramificaciones superiores, donde se parten en tres clusters todas las proteínas, en el cluster central parte izquierda, se ubican unas proteínas con alta representatividad negativa en prácticamente todas las pacientes dibujando un corte horizontal más oscuro. En este mismo nivel de ramificaciones, se puede observar también que para las pacientes ubicadas en la parte superior del eje y, las proteínas de la esquina superior derecha tienen también una alta representatividad negativa. Después de estos dos grupos de proteínas, existen zonas de alta representatividad positiva más claras que también podrían ser indicativo de alguno de los clústers como por ejemplo el claro ubicado en la esquina superior izquierda que parece ser indicativo de la clase *Luminal A*. Fijándonos ahora en el clúster jerárquico del eje vertical por paciente, se aprecian tres clústers principales: El primero de ellos donde predominan los diagnósticos *Luminal A* (verde), el segundo de ellos donde predomina el diagnóstico *Basal-like* (azul) y el último, más grande en tamaño, donde predomina el *Luminal B* (amarillo) aunque la mayoría de casos *HER2-enriched* (rojo) también están contemplados en este.

No obstante, estudios demuestran que existe un conjunto de 50 genes óptimos (PAM50) para la clasificación de tipos de cáncer de mama con los que se puede obtener la misma precisión de diagnóstico, siendo este conjunto de datos mucho inferior en número [31]. Los genes contienen la información necesaria para poder crear cada tipo de proteína, es por esto por lo que se puede referenciar cada proteína al gen que lo secuencia. Por ello, vamos a realizar también un *heatmap* sobre nuestros datos quedándonos únicamente con aquellas proteínas que provengan de estos 50 genes para contrastar resultados con el resultado anterior y expresaremos las conclusiones en función de los genes por facilidad de nomenclatura, comprensión y relevancia.

Para disponer de única y exclusivamente de la información relacionada con los genes PAM50, se deben cruzar los datos de los que ya disponíamos, con las proteínas de la base de datos PAM50. A continuación, el siguiente *heatmap* estará representado en el eje x según cada proteina en función de los genes de los que provienen y de las pacientes en el eje y, con el respectivo diagnóstico utilizando la misma leyenda de colores que anteriormente. Además, se representará también un gráfico con las correlaciones entre atributos como se vió anteriormente en la casuística previa.

GeneSymbol	CENPF	MKI67	KRT5	KRT17	KRT14	ERBB2	EGFR	MAPT	MAPT	MAPT	...
Patient											
AO-A12D	0.490373	-0.625898	-3.056549	-3.113696	-2.629851	9.668177	0.402748	-3.026070	-2.938445	-3.094647	...
AO-A12D	0.327403	-0.471535	-2.391920	-2.018105	-1.182518	9.438237	0.232117	-2.300299	-2.135380	-2.553174	...
C8-A131	0.764109	1.743696	0.658864	2.496602	1.172945	0.197406	1.245807	-5.372478	-5.522250	-5.461532	...
C8-A131	0.922474	1.545247	0.478179	1.511071	1.173102	-3.383772	1.359175	-7.143193	-7.416606	-7.443188	...
AO-A12B	-1.108807	-1.507899	-4.121399	-3.916310	-4.528806	-1.491270	-2.386456	2.025730	2.017416	2.050673	...

5 rows × 45 columns

Figura 44: Base de datos cruzada con PAM50.

En la figura 44 se representan 5 pacientes de la base de datos cruzada con los genes que componen la PAM50. Cada columna sigue siendo una proteína pero referenciada por el gen del que provienen. Se puede observar que de los 50 genes que conforman el PAM50, entre nuestros datos solo están contemplados los 40 que se muestran en la figura 45 sin contar la de *Diagnosis*.

```
PPAM50.columns.unique()
```

```
array(['CENPF', 'MKI67', 'KRT5', 'KRT17', 'KRT14', 'ERBB2', 'EGFR', 'MAPT',
       'ANLN', 'MLPH', 'PGR', 'ACTR3B', 'KIF2C', 'PHGDH', 'CDH3', 'NDC80',
       'NAT1', 'RRM2', 'ESR1', 'BAG1', 'CEP55', 'MMP11', 'SFRP1', 'NUF2',
       'FGFR4', 'FOXA1', 'UBE2C', 'CCNB1', 'TYMS', 'FOXC1', 'CXXC5',
       'UBE2T', 'CDC20', 'MELK', 'SLC39A6', 'BCL2', 'EXO1', 'BIRC5',
       'CCNE1', 'MYC', 'Diagnosis'], dtype=object)
```

Figura 45: Genes presentes en la base de datos proteómica.

En la figura 47, están representadas las correlaciones entre proteínas indexadas por el gen que las genera y ellas mismas. En el gráfico se observa a primera vista que las proteínas procedentes del gen MAPT y MLPH están fuertemente correlacionadas con el conjunto de manera inversamente proporcional. Esto explicaría el comportamiento visto en la figura 43, donde se apreciaba el comportamiento inverso con la mayoría de atributos restantes. Por otra parte parece haber un conjunto de genes tales como ERBB2, CCNE1 o MYC que no presentan correlación con ningún otro gen.

Observando el *heatmap* de la figura 46, salta a primera vista que existe un conjunto de 9 proteínas provinientes del gen MAPT que son altamente representativas tanto positiva como negativamente para todas las pacientes contempladas, hecho que era de esperar por la figura de correlaciones 47. Esto confirma que este gen es un gen significativo para la detección y la clasificación del cáncer de mama en general. Se puede apreciar además, que para el primer cluster de diagnósticos en los que el cáncer de mama *Luminal B* (color amarillo) este gen MAPT tiene alta representatividad positiva pero alta negativa en el resto. Por lo tanto, se puede afirmar que el gen MAPT es indicador discriminatorio del cáncer de mama *Luminal B*. Este mismo gen esta altamente presente y de manera negativa en el cluster central donde predomina el tipo *Basal-like*. En una menor medida, por último, también es representativo del último y tercer clúster donde predomina el tipo *Luminal A*. El segundo grupo más numeroso de proteínas que provienen del mismo gen, son aquellas relativas al gen MLPH que tiene un comportamiento similar de la representatividad para cada clúster. Estos dos genes, son los más significativos del conjunto del *heatmap* como puede apreciarse en la jerarquía superior, puesto que forman parte del mismo clúster y presentan patrones similares. Por otra parte, cabe destacar también el papel de las proteínas representadas a la izquierda del eje, que aunque presentan unos patrones algo más difusos, tienen representatividad positiva y negativa bastante alta como se ve por la intensidad de las tonalidades. Aunque estas últimas puedan parecer a primera vista algo más aleatorias, puede que con más datos pueda haber algún patrón más marcado por lo que sería conveniente ir verificando estas afirmaciones con más datos.

Si nos fijamos a continuación en los clústers jerárquicos del eje vertical, podemos apreciar que se han clasificado todas las pacientes en tres grandes clústers. El primero de ellos, el de la parte superior, contiene más diagnósticos del tipo *Luminal B* (Amarillo). El segundo, es predominantemente de tipo *Basal-like* (azul) y por último, un clúster algo más extenso en tamaño y heterogéneo pero donde predomina el tipo *Luminal A* (verde). Este hecho deja entrever que el tipo más poco común (figura 40) y el más complejo de identificar un patrón no difuso es el tipo *HER2-enriched* (rojo). Se aprecia que la mayoría de *HER2-enriched* están en el clúster junto con *Luminal A*, pero no hay patrón identificable entre estos dos tipos. Se puede observar que la agrupación de clústers es

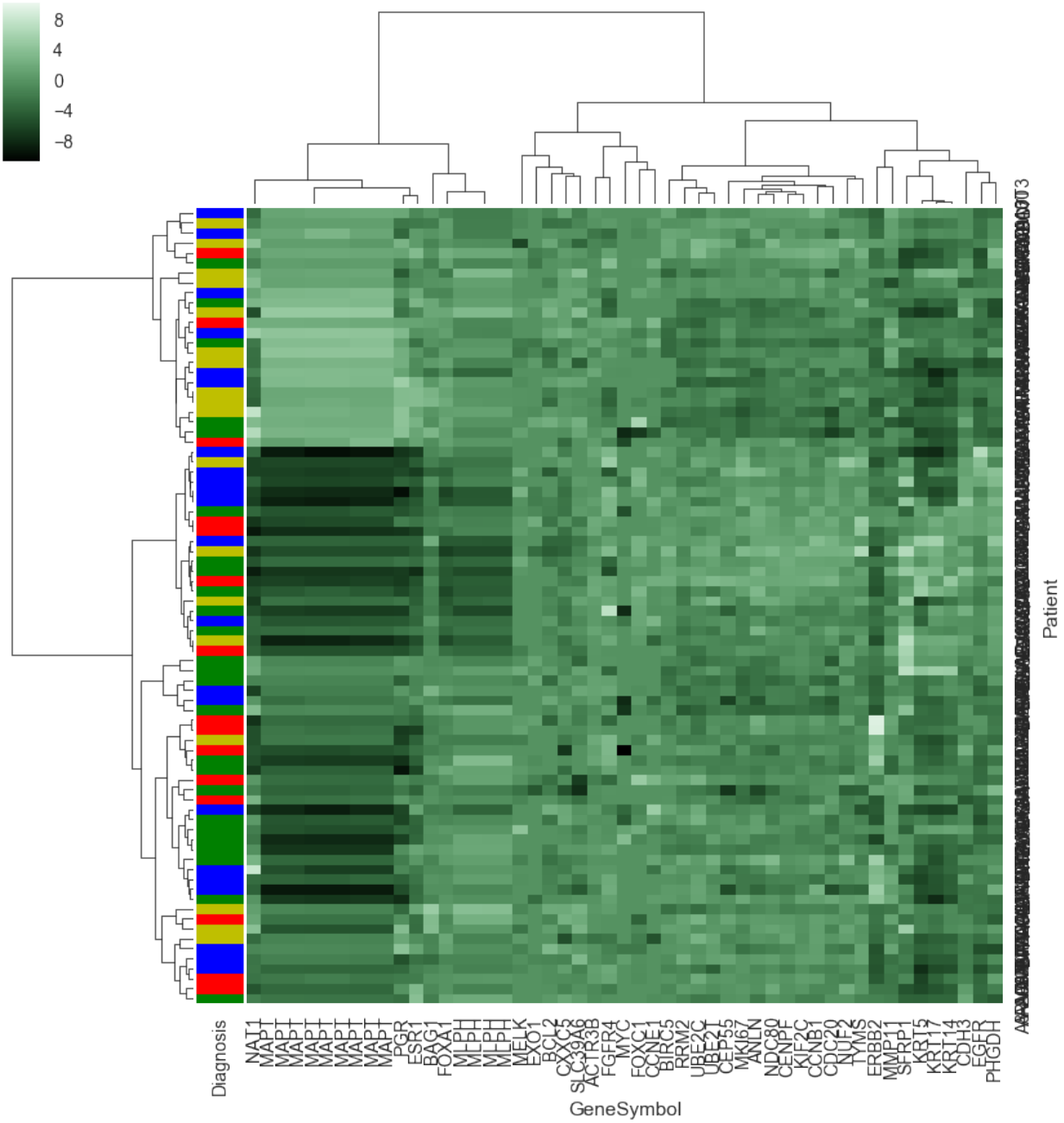

Figura 46: *Heatmap* con clustering jerárquico con PAM50.

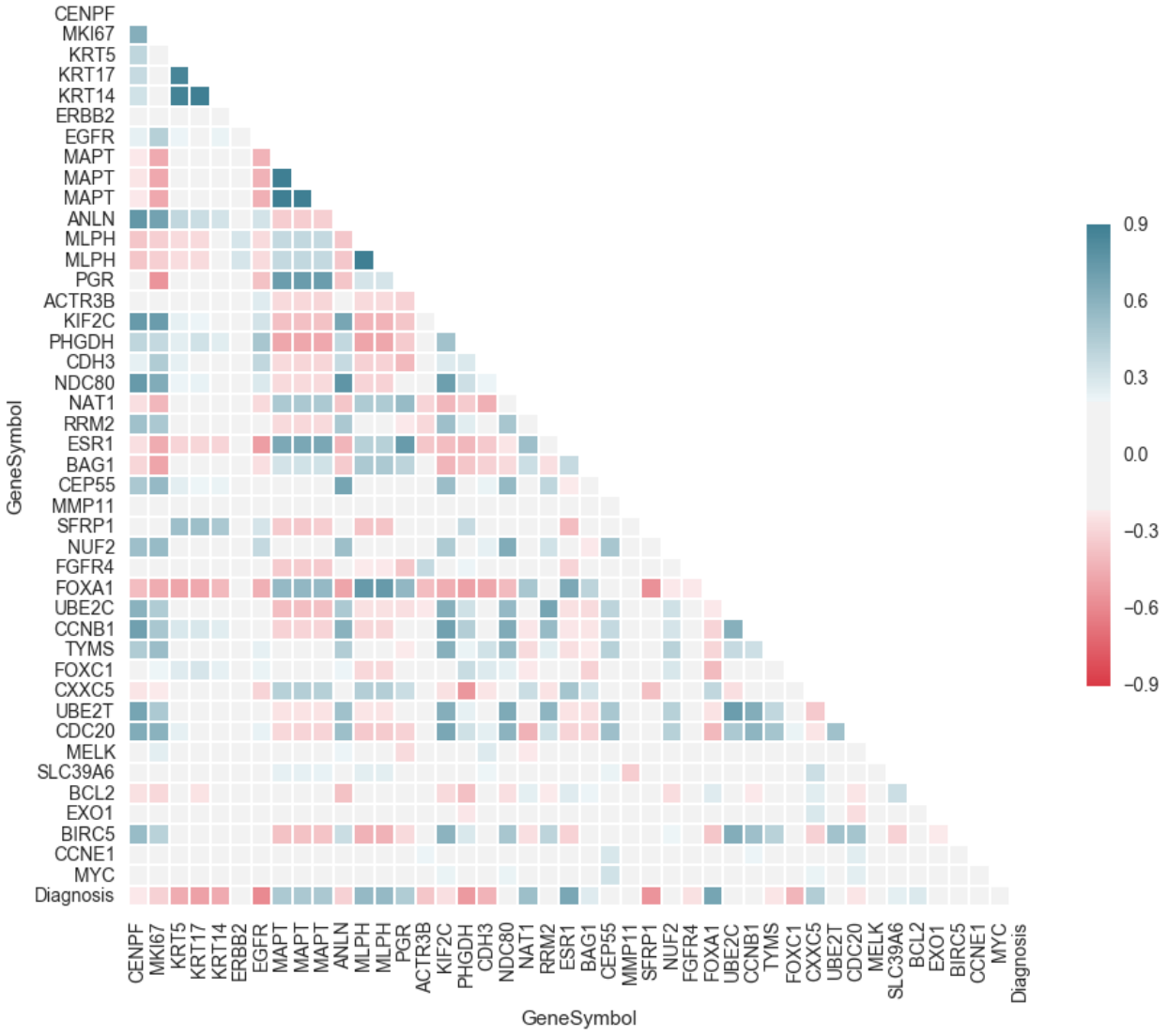

Figura 47: Correlaciones entre proteínas y sus respectivos genes.

exactamente igual a la que proveía el *heatmap* de la figura 43 aunque en este caso, con muchos menos datos. Esto prueba que efectivamente, el conjunto de genes que conforman el PAM50 es un conjunto óptimo para la clasificación, siendo éste un conjunto de datos mucho menor, aunque con los pocos datos introducidos al sistema no somos capaces de discriminar bien los cuatro tipos existentes de diagnóstico de cáncer de mama.

4.3. Contraste de resultados con otros métodos

Los resultados previamente expuestos pueden parecer muy buenos a primera instancia. A continuación, en este apartado, conviene determinar cuán buenos son los resultados en comparación con otros estudios ya realizados utilizando otras técnicas diferentes. De esta manera, nuestro modelo podrá ser contrastado con otras técnicas de sistemas de diagnóstico automático para extraer unas conclusiones fidelignas entre el ámbito científico-técnico.

El caso de detección de cáncer de mama va a ser contrastado con otras dos técnicas utilizadas para el diagnóstico: Por una parte, el diagnóstico automático de cáncer de mama basado en segmentación de mamografías, y por otra, el diagnóstico realizado a través de transformadas *Wavelet* y SVM también sobre mamografías. Ambos métodos son técnicas utilizadas en otros estudios experimentales con el fin de mejorar el diagnóstico de la enfermedad. El primer artículo relativo a la segmentación de mamografías [32], utiliza sistemas *fuzzy* para producir clasificaciones binarias muy discriminadas a partir de segmentación de mamografías. Obtienen un resultado de un 95 % de precisión media, un $3,594\,\%$ por debajo del resultado obtenido con nuestro modelo de una precisión de $97,594\,\%$ utilizando Machine Learning. El artículo [33] introduce la transformada *Wavelet* (DWT) y la transformada esférica *wavelet* (SWT) en el sistema de ayuda al diagnóstico, ya que son técnicas que rara vez han sido utilizadas y pueden operar bien sobre las mamografías para conseguir discriminar mejor los diagnósticos. Adicionalmente al tratamiento previo que realiza con las transformadas *wavelet*, implementa 5 modelos Machine Learning como núcleo: un discriminante lineal (LDC), un discriminante cuadrático (QDC), algoritmo de la media más próxima (NMC), *Support Vector Machines* y un clasificador Parzen (ParzenC). Los mejores resultados que se obtienen son gracias al uso del modelo SVM con la transformada esférica alcanzando un máximo de un $88,8\,\%$ de precisión media. En la siguiente figura 48, se expone los resultados presentados en el artículo.

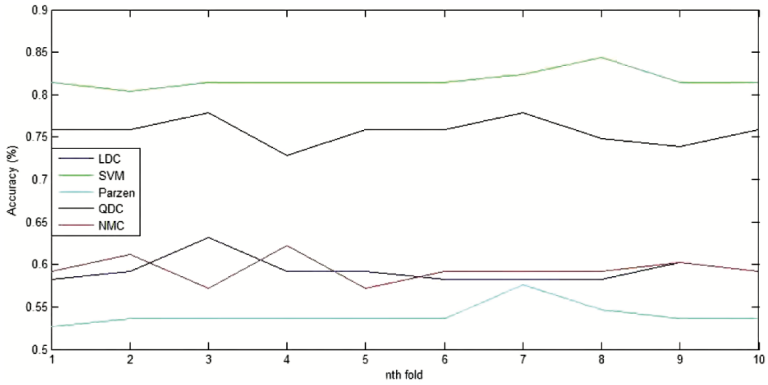

Figure 5: Classification accuracy for DWT using a ten-fold cross validation scheme for various classifiers.

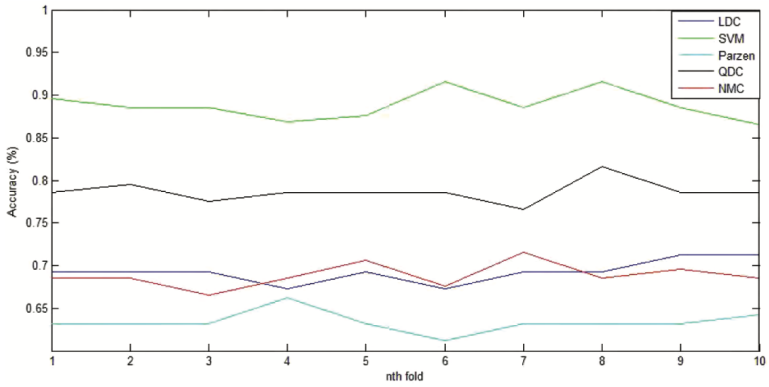

Figure 6: Classification accuracy for SWT using a ten-fold cross validation scheme for various classifiers.

Figura 48: Resultados expuestos en el estudio.

Con los resultados de este estudio [33], los presentados en este proyecto representan una mejora del $9,794\,\%$, alcanzando un $98,594\,\%$ con una regresión logística con respecto al $88,8\,\%$ presentado en dicho estudio con SVM.

En el caso de la clasificación de tipos de cáncer, conviene comparar los patrones y conclusiones que se han extraído con otras técnicas de clasificación genética para contrastar la validez o la contraposición de dichas conclusiones.

Como se ha expuesto en los resultados de la casuística de clasificación, se identificaban dos genes muy significativos entre los diversos tipos de cáncer de mama: MAPT y MLPH. Según el artículo [34], el gen MAPT es un gen que se ve afectado de tal manera que puede ser considerado como un atributo predictivo significativo en el cáncer de mama. Por otra parte, también se han identificado tres genes con una presencia más marcada en pacientes con cáncer de mama entre las que se encuentra el gen MLPH [35]. De esta manera los resultados obtenidos sobre estos dos genes tienen coherencia. Fijándonos en los patrones obtenidos, se han obtenido unos resultados similares en el artículo [15] donde han utilizado un algoritmo de clúster jerárquico.

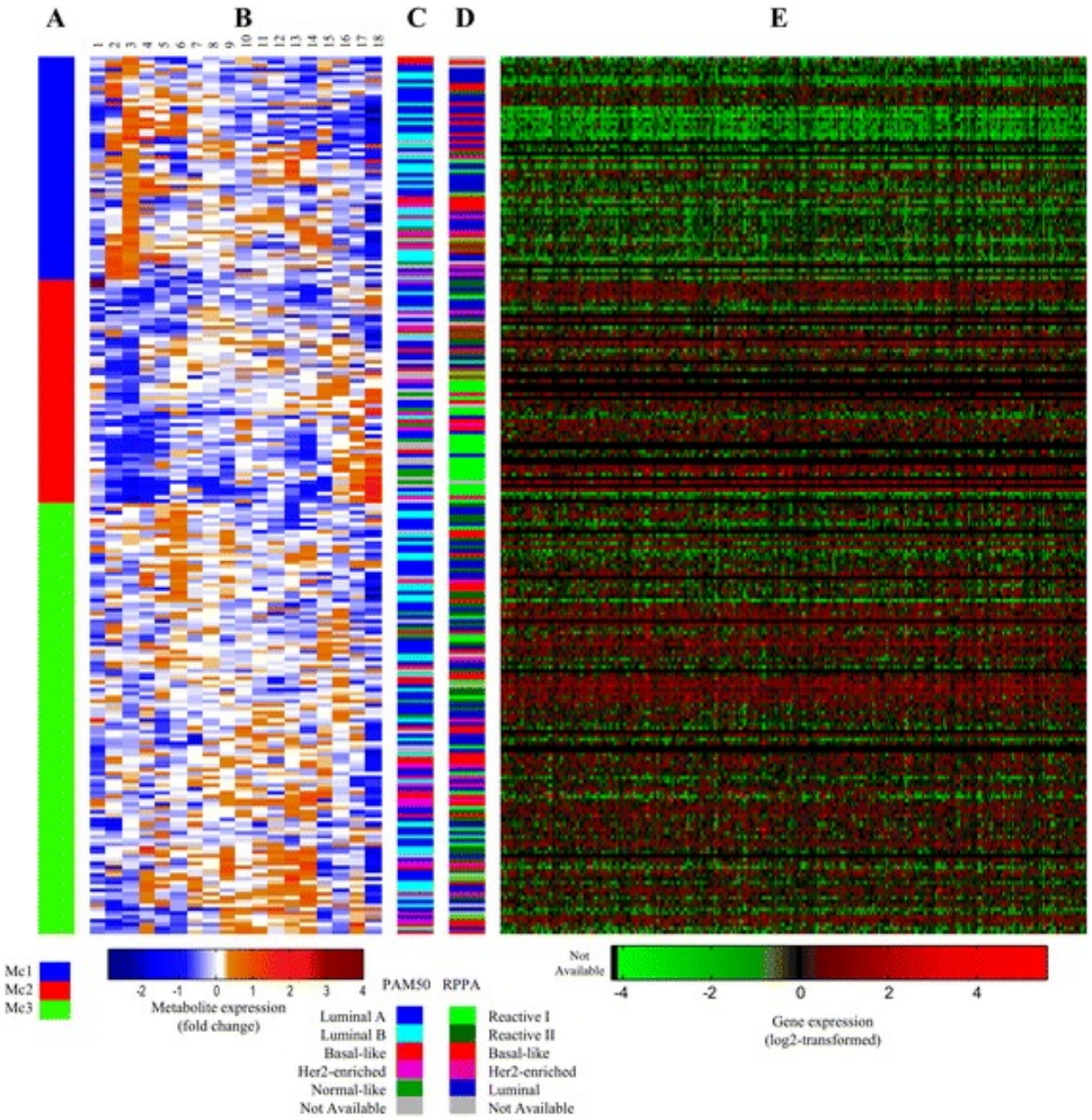

Figura 49: Resultados expuestos en clasificación de tipos de cáncer de mama.

Como se puede apreciar en la figura 49, la agrupación de clústers A también se ha obtenido tres grandes clústers, donde si se observan los diagnósticos por PAM50 en C, encontramos algunas similitudes con nuestras conclusiones. Por una parte, el primer clúster se ve claramente que es predominantemente diagnósticos de *Luminal A*, aunque los otros dos clústers siguientes poseen diagnósticos mucho más heterogéneos. Así pues, coinciden los resultados en número de clústers pero no podemos afirmar que contemplan los mismos tipos, a excepción del *Luminal A* donde coinciden ambos modelos.

5. Conclusiones

En este proyecto, se ha realizado una introducción al Machine Learning haciendo hincapié en el papel del tratamiento de datos y la reducción de dimensionalidad que éstos tienen sobre los resultados. Se ha comprobado todo el marco teórico tratado a lo largo del documento, en la realización de una aplicación con datos reales para el diagnóstico predictivo y clasificación de tipos de cáncer de mama.

Se ha comenzado con una introducción al ámbito del Machine Learning y sus tipos, así como a los conceptos clave biomédicos necesarios para poder comprender la aplicación. Seguidamente, se ha recorrido el decálogo del pre-procesado de datos que se ha de aplicar a un modelo Machine Learning, explicando cada paso y contemplando los métodos más utilizados. Debido a la optimización de los algoritmos utilizados como núcleos de Machine Learning y al ámbito Big Data hacia el que se está evolucionando, se ha enfatizado en la importancia del pre-procesado de los datos y la reducción de dimensionalidad. Parte en la que se ha profundizado más a nivel matemático viendo la descomposición por valores singulares y el análisis de componentes principales.

Todo ello, se ha aplicado por una parte al diagnóstico de cáncer de mama, donde de entre un modelo bayesiano, un modelo basado en regresión logística y otro basado en *Support vector machines*, el mejor diagnóstico conseguido ha alcanzado una precisión de $97,891\,\%$ siendo éste el modelo basado en regresión logística con análisis de componentes principales. Por otra parte, también se ha visto un modelo de clasificación de tipos de cáncer de mama. Se han extraído unos patrones que resultan significativos para la discrimincación de los cuatro tipos contemplados, donde cabe mencionar la importancia de los genes MAPT y MLPH. A su vez, se ha corroborado la presencia de tres grandes clústers entre las pacientes: Uno predominantemente del tipo *Luminal B*, otro en el que predomina el tipo *Basal-like* y por último, otro más grande y heterogéneo pero en el que predomina el tipo *Luminal A*, siendo pues el *HER2-enriched* el caso más subjetivo de diagnosticar. Seguidamente, los resultados han sido contrastados con otros estudios publicados y han resultado ser coherentes con éstos. Se puede afirmar entonces que grandes conjuntos de datos no equivale siempre a mejores resultados, osbervando el impacto en los resultados de este proyecto con reducción de dimensionalidad.

6. Líneas Futuras

En este proyecto se ha abarcado la detección y la clasificación de tipos bajo el objetivo de mejorar el proceso de diagnóstico y, a su vez, intentar detectar nuevos patrones con los que poder discernir entre los diferentes tipos. El modelo final presentado de detección, provee un $97,891\,\%$ de precisión media de acierto en el diagnóstico. A pesar del consistente resultado obtenido, se podría intentar mejorar aún más si cabe, utilizando algún modelo *ensemble* formado por dos modelos o más además de la regresión logística, como por ejemplo un *Random Forest* o un *K-means*. Por otra parte, para el modelo de clasificación, podría entrenarse otro modelo no supervisado utilizando también un *ensemble* con un *Gaussian Mixture Model (GMM)* que podría proveer resultados robustos. Ambos modelos funcionarían con mucha más precisión con más datos, por lo que sería beneficioso para el sistema disponer de más cantidad de datos sobre los que trabajar y poder corroborar conclusiones y mejorar la actuación total del mismo sistema.

Además de intentar mejorar todo el sistema implementando otros modelos o introduciendo más cantidad de datos, también se podría realizar una tercera aplicación para el tratamiento de la enfermedad una vez haya sido diagnosticada. El campo de la medicina personalizada será en un corto período de tiempo una de los pilares fundamentales de la medicina moderna. Podría construirse un tercer modelo con datos genómicos para poder encontrar el tratamiento más eficaz y menos nocivo para la paciente en función de su diagnóstico y de su genoma. Hasta ahora, el tratamiento también es subjetivo y queda bajo la propia opinión y experiencia del facultativo. Realizando este sistema, se podría lograr un tratamiento personalizado por paciente para asegurar que el tratamiento escogido tiene mayores probabilidades de éxito e intentar reducir al máximo la metodología prueba-error.

Referencias

[1] "Global cancer organization, cancer today." `https://gco.iarc.fr/`. consultado: 02-03-2017.

[2] F. Kamangar, G. M. Dores, and W. F. Anderson, *Patterns of Cancer Incidence, Mortality, and Prevalence Across Five Continents: Defining Priorities to Reduce Cancer Disparities in Different Geographic Regions of the World.* Journal of Clinical Oncology, 2006.

[3] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python: A Guide for Data Scientists.* O'Reilly Ed., 2016.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks.* University of Toronto.

[5] Y. Koren, *The BellKor Solution to the Netflix Grand Prize.* Yehuda Koren, 2009.

[6] W. M. Czarnecki, *Weighted Tanimoto Extreme Learning Machine with Case Study in Drug Discovery.* IEEE Computational Intelligence Magazine, 2015.

[7] S. Fiorini, A. Verri, A. Tacchino, M. Ponzio, G. Brichetto, and A. Barla, *A machine learning pipeline for multiple sclerosis course detection from clinical scales and patient reported outcomes.* IEEE Medical Engineering Conference, 2015.

[8] *Learning from User Interactions in Personal Search via Attribute Parameterization.* Proceedings of the 10th ACM international Conference on Web Search and Data Mining (WSDM), 2017.

[9] J. M. Tien, *Towards the next industrial revolution.* Logistics and Industrial Informatics (LINDI), 4th IEEE International Symposium on, 2012.

[10] E. Alpaydin, *Introduction to Machine Learning, Chapter 2.* The MIT Press, 2014.

[11] E. Alpaydin, *Introduction to Machine Learning, Chapter 18.* The MIT Press, 2014.

[12] C. Li, *Breast Cancer Epidemiology.* Springer Ed., 2010.

[13] "Cancer .net." `http://www.cancer.net/es/tipos-de-c%C3%A1ncer/c%C3%A1ncer-de-mama/diagn%C3%B3stico`. consultado: 10-03-2017.

[14] "American cancer society." `https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection.html`. consultado: 10-03-2017.

[15] T. H. Haukaas, L. R. Euceda, and G. F. Giskeodegard, *Metabolic clusters of breast cancer in relation to gene- and protein expression subtypes.* Haukaas et al. Cancer and Metabolism, 2016.

[16] M. Lutz, *Learning Python.* O'Reilly Ed., 2013.

[17] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data.* AMLBook Ed., 2012.

[18] T. Segaran, *Programming Collective Intelligence.* O'Reilly, 2011.

[19] D. W. Aha, D. Kibler, and M. K. Albert, *Instance based algorithms.* Springer ed., 1991.

[20] S. Haykin, *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 2009.

[21] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[22] S. Yadav and S. Shukla, *Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification.* Advanced Computing (IACC), IEEE 6th International Conference on, 2016.

[23] L. N. Trefethen and D. Bau, *Numerical Linear Algebra.* Siam Ed., 1997.

[24] C. D. Meyer, *Matrix Analysis and Linear Algebra.* Siam Ed., 2000.

[25] I. Jolliffe, *Principal Component Analysis, 2nd Ed.* Wiley-Interscience, 2002.

[26] J. E. Jackson, *User's Guide To Principal Components.* Wiley-Interscience, 1991.

[27] D. Lowd and P. Domingos, *Naive Bayes Models for Probability Estimation.* Department of Computer Science and Engineering, University of Washington.

[28] A. Y. Ng, *Preventing Overfitting of Cross-Validation Data.* Carnegie Mellon University.

[29] G. C. Cawley, *Over-Fitting in Model Selection and Its Avoidance.* International Symposium on Intelligent Data Analysis, 2012.

[30] T. Howley, M. G. Madden, M.-L. O'Connell, and A. G. Ryder, *The Effect of Principal Component Analysis on Machine Learning Accuracy with High Dimensional Spectral Data.* National University of Ireland, 2006.

[31] B. Wallden, J. Storhoff, T. Nielsen, N. Dowidar, C. Schaper, S. Ferree, S. Liu, S. Leung, G. Geiss, J. Snider, T. Vickery, S. R. Davies, E. R. Mardis, M. Gnant, I. Sestak, M. J. Ellis, C. M. Perou, P. S. Bernard, and J. S. Park, *Development and verification of the PAM50-based Prosigna breast cancer gene signature assay.* BMC Medical Genomics, 2015.

[32] J. Malek, A. Sebri, S. Mabrouk, K. Torki, and R. Tourki, *Automated Breast Cancer Diagnosis Based on GVF-Snake Segmentation, Wavelet Features Extraction and Fuzzy Classification.* Springer ed., 2008.

[33] K. Ganesan, U. R. Acharya, C. K. Chua, L. C. Min, and T. K. Abraham, *Automated Diagnosis of Mammogram Images of Breast Cancer Using Discrete Wavelet Transform and Spherical Wavelet Transform Features: A Comparative Study.* Technology in Cancer Research and Treatment, Vol. 13, 6, 2014.

[34] M.-L. Caillet-Boudin, L. Buee, N. Sergeant, and B. Lefebvre, *Regulation of human MAPT gene expression.* Caillet-Boudin et al. Molecular Neurodegeneration, 2015.

[35] arvind thakkar, hemanth raj, ravishankar, B. muthuvelan, arun Balakrishnan, and muralidhara Padigaru, *High Expression of Three-Gene Signature Improves Prediction of Relapse-Free Survival in Estrogen Receptor-Positive and Node-Positive Breast Tumors.* Biomarker Insights, 2015.