UNIVERSIDAD
DE GRANADA

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

Grado en Ingeniería Informática y Matemáticas

TRABAJO DE FIN DE GRADO

Técnicas de minería en bases de conocimiento

Presentado por:
Mónica Calzado Granados

Curso académico 2023-2024

Técnicas de minería en bases de conocimiento

Mónica Calzado Granados

Mónica Calzado Granados *Técnicas de minería en bases de conocimiento*.
Trabajo de fin de Grado. Curso académico 2023-2024.

Responsable de tutorización

Úrsula Torres Parejo
Departamento de Estadística e Investigación Operativa

Daniel Sánchez Fernández
Departamento de Ciencias de la Computación e Inteligencia Artificial

Grado en Ingeniería Informática y Matemáticas

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

Universidad de Granada

Declaración de originalidad

D./Dña. Mónica Calzado Granados

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2023-2024, es original, entendido esto en el sentido de que no he utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 4 de junio de 2024

Fdo: Mónica Calzado Granados

Índice general

Agradecimientos

Agradecimientos (opcional, ver archivo `preliminares/agradecimiento.tex`).

Summary

An english summary of the project (around 800 and 1500 words are recommended).
File: `preliminares/summary.tex`

Introducción

De acuerdo con la comisión de grado, el TFG debe incluir una introducción en la que se describan claramente los objetivos previstos inicialmente en la propuesta de TFG, indicando si han sido o no alcanzados, los antecedentes importantes para el desarrollo, los resultados obtenidos, en su caso y las principales fuentes consultadas.

Ver archivo `preliminares/introduccion.tex`

1. Minería de textos

La Minería de textos es una rama de estudio crucial en la Ciencia de Datos. Se ha convertido en una herramienta invaluable para extraer conocimiento a partir de grandes cantidades de texto no estructurado.

En los últimos años, la Minería de textos ha ganado una gran importancia debido al explosivo crecimiento de datos textuales que se pueden encontrar en diversas fuentes: documentos, correos electrónicos, redes sociales, páginas web y otros tipos de contenido textual.

Esta disciplina se ha vuelto esencial en medida que se encarga de analizar y descubrir nuevo patrones, tendencias y relaciones significativas dentro de conjuntos de información no estructurada. Mediante el uso de técnicas avanzadas de procesamiento del lenguaje natural (PLN) y aprendizaje automático, la minería de textos permite a las empresas, científicos y organizaciones obtener información valiosa que les ayude a impulsar una toma de decisiones estratégica.

1.1. Knowledge Discovery from Databases (KDD)

La información contenida en las bases de datos puede ser analizada de forma manual por expertos para obtener conclusiones. Sin embargo, en estos tiempos de avance tecnológico, debido al gran volumen de datos, necesitamos encontrar una forma de automatizar el proceso.

Surge así el Descubrimiento de Conocimiento en Bases de Datos (en inglés KDD, Knowledge Discovery from Databases), que tiene sus bases en disciplinas como la Estadística, Sistemas de Información y Aprendizaje Automático. El término KDD se puede definir como:

> *El proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos* [1].

1.1.1. Fases del KDD

El proceso de KDD resulta bastante interactivo, en el sentido de que intervienen muchas decisiones tomadas por el usuario según este considere oportuno. Estos pasos se pueden resumir de la siguiente manera [1]:

1. **Aprendizaje sobre el dominio de la aplicación:** es necesario determinar los objetivos y reunir el conocimiento previo que tenemos sobre el problema.

2. **Creación de un corpus de datos:** seleccionamos un dataset del cual queremos extraer conocimiento.

3. **Limpieza de datos y preprocesamiento:** partiendo de los datos recopilados, esta es la fase en la que se eliminan los datos ruidosos y anómalos. También se llevan a cabo estrategias para manejar valores perdidos.

4. **Reducción de datos:** consiste en seleccionar las características más útiles para representar los datos, y reducir la dimensionalidad del problema descartando atributos poco relevantes.

5. **Elegir una técnica de minería de datos:** se trata de decidir el propósito del modelo (qué tipo de información se quiere descubrir) y qué técnica de minería de datos podemos aplicar para ello (sumarización, clasificación, regresión, clustering, etc).

6. **Elegir un algoritmo de minería de datos:** incluye decidir qué modelos y parámetros pueden ser apropiados (por ejemplo, existen modelos para datos categóricos que son distintos de los modelos para datos numéricos).

7. **Minería de datos:** consiste en el descubrimiento de patrones, relaciones y conocimiento de interés, mediante el uso de algoritmos de clustering, clasificación, regresión, asociación, y otras técnicas de aprendizaje automático.

8. **Interpretación:** evaluar los patrones descubiertos y regresar a cualquiera de los pasos anteriores si es necesario. También trata la representación y visualización de los patrones extraídos, para facilitar la comprensión por los usuarios.

9. **Utilización del conocimiento descubierto:** consiste en acciones desde incorporar el nuevo conocimiento en un sistema inteligente, llevar a cabo acciones respaldadas en dicho conocimiento, o simplemente documentarlo y contrastarlo con conocimiento anteriormente extraído.

La Figura 1.1 muestra un resumen de las fases del proceso KDD, donde podemos apreciar que aunque el proceso es lineal, se puede regresar a cualquiera de las fases anteriores si se considera necesario.

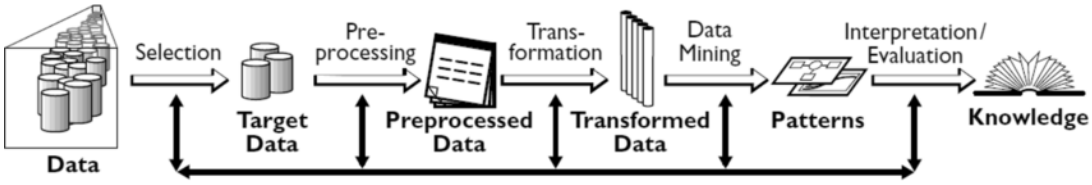


Figura 1.1.: Fases del proceso KDD [1]

La mayoría de los estudios sobre KDD se enfocan principalmente en la fase de minería de datos. Sin embargo, los otros pasos son igualmente importantes para la aplicación del proceso de KDD.

1.2. Knowledge Discovery from Text (KDT)

La mayoría de información generada por las organizaciones se encuentra en forma de texto. Procesar dicha información puede resultar difícil debido al gran volumen de documentos y la falta de estructura de estos con respecto a una base de datos convencional. Es por ello que necesitaremos tratar algunas de las fases del KDD de forma distinta al tratar con textos.

1.2.1. KDD vs KDT

En general, la diferencia principal entre KDD y KDT reside en que el texto carece de una estructura procesable de forma automática. En tal caso, necesitamos de un proceso previo de transformación del texto en datos, utilizando para ello diversas estructuras de datos, a las que llamaremos formas intermedias. Por tanto, en el KDT las etapas de selección, preprocesamiento y transformación de los datos textuales se convierten en fundamentales e imprescindibles [2]. A partir de ahora nos referiremos a esta etapa como Preprocesamiento. En la tabla 1.1 se muestra una comparación entre las fases de KDD y su equivalencia en el proceso KDT. Podemos observar que algunas fases se mantienen igual en ambos casos y otras difieren en el enfoque y las herramientas utilizadas:

Fase	KDD (Knowledge Discovery in Databases)	KDT (Knowledge Discovery in Texts)
1	Comprender el dominio de la aplicación.	Definición de los conceptos de interés y determinar un objetivo.
2	Seleccionar un conjunto de datos objetivo. Generar una base de datos o tomar una.	Los textos se obtienen con herramientas de Recuperación de Información o de forma manual.
3	Limpieza (eliminar ruido) y manejar valores perdidos.	Preparar texto a un formato aceptado por las técnicas de minería (formas intermedias)
4	Transformación de datos (reducción de dimensionalidad, tareas de balanceo de clases...)	Transformación de datos: se eliminan stop words y palabras frecuentes poco relevantes para reducir la dimensionalidad del problema, tokenización, etc. Utilización de redes semánticas y otras herramientas para agrupar conceptos.
5	Elegir de una técnica adecuada de minería de datos.	Elegir de una técnica adecuada de minería de textos.
6	Elección de algoritmos de minería de datos. Elegir parámetros apropiados.	Elección de algoritmos de minería de datos que funcionen bien con texto. Elegir parámetros apropiados.
7	Aplicación del algoritmo	Aplicación del algoritmo
8	Interpretar patrones descubiertos. Cálculo de medidas de validación. Visualización.	Visualización (nubes de palabras, ocurrencia de términos...) para interpretar los resultados y encontrar nuevos patrones en el texto.
9	Utilización del nuevo conocimiento.	Utilización del nuevo conocimiento.

Tabla 1.1.: Comparativa entre los procesos KDD y KDT

1.2.2. Fases del KDT

Teniendo en cuenta las comparaciones de la tabla 1.1, observamos que el proceso de KDT utiliza sus propias tareas de descubrimiento de conocimiento, que a veces coinciden con las del KDD y otras no. De forma más general, podemos dividir las fases del KDT en tres [2]:

- **Preprocesamiento:** antes de realizar la minería, necesitamos preparar el texto a un formato que sea aceptado por las técnicas que queremos aplicar. Esto abarca desde la homegenización de distintos documentos, hasta la representación en Formas Intermedias. Entre las distintas técnicas de transformación de texto encontramos las siguientes:
  1. Normalización de texto: convertir todo el texto a minúsculas o mayúsculas para que las palabras sean tratadas de manera uniforme.
  2. Tokenización: dividir el texto en unidades más pequeñas, como palabras o frases, llamadas tokens.
  3. Etiquetado: asignar a cada token una etiqueta que indica su función gramatical: sustantivo, verbo, adjetivo, etc.
  4. Eliminación de *stop words*: eliminar palabras comunes pero no informativas que no aportan significado al texto, como determinantes, preposiciones, etc.
  5. *Stemming*: Reducir las palabras a su raíz, eliminando sufijos y prefijos, para reducir las palabras a su forma base.
  6. Corrección ortográfica: Corregir errores ortográficos comunes para mejorar la precisión del análisis.
  7. Eliminación de palabras raras o infrecuentes: Eliminar palabras que aparecen con poca frecuencia en el corpus, ya que pueden no ser útiles para el análisis.
  8. Eliminación de URLs, referencias y etiquetas HTML, que no aportan información semántica.
  9. Uso de *synsets* [3] que identifican conceptos a partir de palabras sinónimas, haciendo uso de diccionarios ontológicos.

- **Minería de textos**: En la literatura, algunos autores identifican el término *Minería de Textos* con todo el proceso de Descubrimiento de Conocimiento en Texto (KDT) [2]; sin embargo, en este contexto, nos referiremos específicamente a la fase encargada de localizar patrones, relaciones y estructuras interesantes, útiles y desconocidos en el texto.

- **Visualización**: una vez obtenidos los resultados de la fase anterior, es conveniente una fase de visualización para representar de manera efectiva los patrones, términos, tendencias y los resultados descubiertos. La visualización juega un papel crucial en la comprensión de la información extraída, permitiendo a los usuarios explorar y analizar los datos de manera intuitiva y amigable.
  Se pueden emplear numerosas herramientas dependiendo de la técnica de minería empleada, como nubes de palabras, diagramas de ocurrencia de términos, gráficos de dispersión de términos, clusterización de documentos, etc. Estas herramientas no sólo facilitan la interpretación de los resultados, sino que también pueden ayudar a descubrir más patrones, relaciones semánticas entre términos, etc.

1.2.3. Aplicaciones del KDT

Debido a la dificultad inherente de estructurar y procesar el texto de manera adecuada, las aplicaciones de la Minería de Texto enfrentan desafíos bastante complejos. Al igual que en la Minería de Datos, los expertos humanos tienen un papel crucial a la hora de interpretar

los resultados de manera efectiva. Entre las distintas aplicaciones del KDT destacamos las siguientes [2]:

1. **Segmentación de clientes:** se trata de clasificar a los clientes en grupos según patrones de compra, preferencias o características demográficas. Esta segmentación permite a las empresas satisfacer mejor las necesidades del cliente y realizar estrategias de Marketing más eficaces.

2. **Detección de fraude financiero:** este es un desafío constante en el sector financiero debido a la evolución de las técnicas empleadas por los delincuentes. Existen algoritmos de detección de anomalías, que se utilizan para detectar transacciones y patrones de compras inusuales para un cliente específico.

3. **Filtrado de correos electrónicos:** este problema surge debido al volumen considerable de mensajes que un usuario puede recibir diariamente, lo que puede resultar abrumador sin herramientas para gestionarlos. La implementación de filtros de correo electrónico puede mejorar significativamente la productividad en el lugar de trabajo, al ayudar a procesar los correos de forma más eficiente.

4. **Análisis de sentimientos:** se trata de una técnica fundamental para comprender la opinión y el comportamiento de los usuarios en plataformas como Twitter, Facebook o Instagram. Esto implica la extracción de tendencias y patrones a partir de las publicaciones y comentarios hechos por los usuarios. En el ámbito político, el análisis de sentimientos en las redes sociales ha demostrado ser útil para evaluar la popularidad de los candidatos y comprender las preocupaciones y necesidades de los votantes. Esta información puede resultar crucial para diseñar con éxito campañas políticas.

5. **Medicina personalizada:** es un área revolucionaria en la atención médica cuyo objetivo es adaptar los tratamientos y terapias a las características individuales de cada paciente. Para ello se nutre de datos genéticos y clínicos para identificar patrones que ayuden a predecir la reacción del paciente ante un tratamiento específico.

1.3. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural (PLN) es un campo de estudio centrado en el desarrollo de sistemas que puedan analizar y comprender el lenguaje humano a partir de un texto libre.

El objetivo del PLN es extraer el significado y el contexto de los textos. Usando PLN podemos hacer tareas como resumen automático de textos, generación de textos, traducción de idiomas, análisis de sentimientos, reconocimiento del habla y clasificación de artículos por temáticas. Para ello, utiliza conceptos lingüísticos como nombres, verbos, adjetivos, etc. El PLN se enfrenta a retos como las numerosas ambigüedades que contiene el lenguaje natural, tanto de palabras con doble significado como de estructuras gramaticales o frases hechas.

Para llevar a cabo estas tareas, el PLN hace uso de *representaciones de conocimiento*, como diccionarios de palabras con su significado, conjuntos de reglas gramaticales, etc. También se suele hacer uso de ontologías, diccionarios de sinónimos, abreviaciones, etc.

El procesamiento del lenguaje natural es un paso clave en el proceso de extracción de conocimiento a partir de texto en el KDT. En este contexto, el PLN se utiliza para transformar el texto en una representación estructurada y procesable, permitiendo así aplicar técnicas de

minería de texto. Algunas tareas típicas del PLN en el KDT incluyen la tokenización (división del texto en unidades más pequeñas como palabras o frases), el etiquetado de partes del discurso, la eliminación de stopwords, la detección de entidades nombradas, el análisis de sentimientos y la extracción de información relevante.

1.4. Formas intermedias

Debido a que el texto se trata de información no estructurada, dentro de la etapa de Prepocesamiento del proceso KDT, es crucial convertir el texto a una representación o forma intermedia que se pueda manejar de forma computacional. Esto va desde dividir el texto en unidades mínimas hasta identificar relaciones semánticas entre palabras. Al mismo tiempo se debe procurar no perder la integridad del texto inicial, es decir, no perder información sobre las relaciones entre términos. A continuación veremos algunas de las formas intermedias comúnmente empleadas [2].

1.4.1. Palabra

La palabra es el elemento mínimo de estudio sobre un texto. Por sí misma, una palabra no aporta suficiente información, pero algunas consideraciones como el conteo de palabras y la riqueza léxica (proporción de palabras diferentes con respecto al total) puede proporcionar información sobre el texto.

1.4.2. Bolsa de palabras

Consiste en recopilar todas las palabras que aparecen en un documento, ignorando el orden, la estructura gramatical y las relaciones semánticas. En esta representación, cada palabra se considera un *token* y se crea un vector que indica cuántas veces aparece cada palabra en el documento. Dentro de una bolsa podemos encontrar distintos tipos de tokens:

- **Palabra vacía** *(stop word)*: preposiciones, artículos y conjunciones, en definitiva son palabras que sirven para relacionar términos pero por sí mismas no aportan valor.
- **Palabra clave** *(keyword)*: se trata de palabras que identifican la temática del texto. Se les suele asociar un peso numérico para destacar su importancia.

A modo de ejemplo, podemos considerar el siguiente texto:

> *El perro persigue al gato. El gato huye del perro.* (1.4.2)

La representación del texto mediante una bolsa de palabras podría verse como en la tabla 1.2.

La misma bolsa de palabras pero eliminado las *stop words*, se vería como en la tabla 1.3.

Observamos que al representar el texto como una bolsa de palabras, se pierde completamente el orden. La bolsa de palabras nos dice cuántas veces aparece cada palabra, pero ya no sabemos nada sobre la relación entre el perro y el gato, y entre quién persigue y quién huye. Esta pérdida de contexto puede ser crítica en algunas tareas, ya que al perder la relación entre palabras, podemos perder información crucial.

Palabra	Frecuencia
el	2
perro	2
persigue	1
al	1
gato	2
huye	1
del	1

Tabla 1.2.: Bolsa de palabras del ejemplo 1.4.2

Palabra	Frecuencia
perro	2
persigue	1
gato	2
huye	1

Tabla 1.3.: Bolsa de palabras sin *stop words* del ejemplo 1.4.2

1.4.3. Concepto

Un concepto es la representación mental de un objeto o una idea, expresada a través de un término. Los conceptos organizan palabras relacionadas bajo una idea común. Por ejemplo, el concepto de *animales domésticos* podría incluir palabras como perro, gato, pájaro, etc. Uno de los inconvenientes al tratar con conceptos es que a diferencia de las palabras, estos suelen depender del contexto.

1.4.4. Taxonomía

Una taxonomía o jerarquía de conceptos se trata de una clasificación que organiza de forma ordenada conceptos o términos que se encuentran en niveles jerárquicos distintos, basándose en las relaciones entre ellos. Las taxonomías pueden representarse mediante jerarquías de términos, donde los conceptos más generales aparecen en los niveles superiores, y los conceptos más particulares aparecen en niveles inferiores. Un ejemplo de taxonomía del reino animal es el siguiente:

1. *Animalia*
    a) *Mammalia*
        I. *Felidae (Gatos)*
        II. *Canidae (Perros)*
    b) *Aves*
        I. *Accipitridae (Águilas)*
        II. *Anatidae (Patos)*
    c) *Reptilia*
        I. *Pythonidae (Pitones)*

II. *Crocodylidae (Cocodrilos)*

1.4.5. Grafos

Un grafo o red semántica es una representación visual de conocimiento que emplea conceptos y las relaciones semánticas entre ellos. Estos conceptos son representados mediante nodos, y las relaciones mediante aristas o enlaces.

Una característica distintiva de las redes semánticas es el uso de la herencia, donde los nodos hijos heredan las características de los nodos padres. Estas redes son útiles para representar y comprender la semántica y las relaciones entre conceptos en un dominio específico. Además, destacan por ser capaces de preservar gran carga semántica, mientras que otras formas intermedias la pierden. Sin embargo, en ocasiones esto se traduce en un aumento excesivo de la complejidad del tratamiento de datos.

Entre los distintos tipos de red semántica destacan las redes IS-A y los grafos conceptuales.

1.4.6. Ontología

La literatura científica ofrece una gran variedad de definiciones para las ontologías. En términos generales, se trata de un mapa conceptual detallado y estructurado del conocimiento sobre un dominio específico. Una ontología está formada por los siguientes elementos básicos:

1. *Conceptos*: representan las entidades o clases dentro del dominio de interés. Por ejemplo, en una ontología sobre automóviles, podría haber conceptos como *Automóvil*, *Motor*, *Rueda*, etc.

2. *Roles*: describen relaciones binarias entre conceptos y propiedades. Por ejemplo, las propiedades *color*, *velocidad máxima*, *número de puertas* y las relaciones *es un tipo de*, *tiene*, etc.

3. *Individuos*: instancias o ejemplos.

4. *Axiomas*: especifican las reglas o restricciones que deben cumplirse en el dominio. Por ejemplo, *un automóvil tiene cuatro ruedas*.

Las ontologías proporcionan un marco formal para representar el conocimiento, lo que facilita la interoperabilidad entre distintos sistemas y permite a las máquinas realizar inferencias y razonamientos sobre el dominio de conocimiento especificado.

1.5. Minería de bases de conocimiento

En la minería de datos, el conocimiento es obtenido mediante razonamiento de tipo *inductivo*. Consiste en encontrar patrones o modelos generales que expliquen los datos. Se trata de un proceso que va de lo particular a lo general, es decir, se parte de los datos para llegar a conclusiones sobre estos. Este es el tipo de razonamiento que se aplica en las técnicas de clustering, reglas de asociación, árboles de decisión, etc. [2]

La minería de textos puede verse como un caso particular de la minería de datos, donde es necesario un proceso previo de transformación del texto en datos (formas intermedias), dado que el lenguaje natural de un texto es incomprensible e improcesable por una máquina.

Este enfoque convencional ha demostrado ser muy útil en una gran cantidad de aplicaciones; sin embargo, no podemos olvidar que el texto contiene una capacidad expresiva mucho mayor que las estructuras de datos. Por tanto, al reducir el texto a forma intermedia, se pierde una gran cantidad de información valiosa. De aquí surge la necesidad de definir los términos *conocimiento* y *base de conocimiento* para hacer referencia a una representación del texto más rica que los datos y las bases de datos.

1.5.1. Bases de conocimiento

En el ámbito de Ingeniería del Conocimiento, se distinguen tres niveles de contenido: *datos*, *información* y *conocimiento* [4]:

- Los **datos** son la mínima unidad semántica, son hechos o elementos discretos que pueden ser registrados o almacenados. Por sí solos son irrelevantes y no aportan información.
- La **información** es el resultado del procesamiento de esos datos, lo que les otorga significado y contexto.
- El **conocimiento** va un paso más allá al incorporar la comprensión y la capacidad de aplicar la información en situaciones reales, por ejemplo para la toma de decisiones.

Las *bases de conocimiento* son estructuras que se utilizan para la representación del conocimiento de un dominio específico. En el contexto de la inteligencia artificial, una base de conocimiento se utiliza para capturar información relevante, reglas, hechos, conceptos y relaciones dentro de un área de interés. Una base de conocimiento puede estar formada por distintos tipos de datos estructurados, como ontologías, reglas de producción, redes semánticas, grafos, entre otros.

Teniendo esto en cuenta, la Ingeniería de conocimiento se enfoca en convertir datos en conocimiento accionable, mediante un proceso que usa métodos y técnicas para estructurar y procesar datos de manera que puedan ser analizados para tomar decisiones.

La Ingeniería de conocimiento y la Ciencia de datos tienen una relación estrecha en el ámbito de la inteligencia artificial. Mientras que la Ciencia de datos se centra en el estudio y análisis de grandes volúmenes de datos con el fin de obtener patrones y relaciones, la Ingeniería de conocimiento se enfoca en representar y utilizar ese conocimiento de manera efectiva.

1.5.2. Sistemas de representación de conocimiento

Hemos visto que una base de conocimiento es un sistema que almacena y organiza el conocimiento de un dominio específico para que pueda ser accesible por otros sistemas de software. Dentro de una base de conocimiento podemos encontrar información, hechos, reglas, y más elementos que ayudan a describir y comprender el dominio estudiado. En este contexto, una base de conocimiento puede estar representada mediante estructuras de datos tales como redes semánticas, marcos, reglas de producción, ontologías, entre otros. Es a estos elementos a lo que denominamos sistemas de representación de conocimiento.

En breves palabras, una estructura de representación de conocimiento es la estructura interna que define de qué forma está modelado y organizado el conocimiento dentro de una base de conocimiento. Como es de esperar, esta estructura debe ser fácil de manejar computacionalmente y útil de comprender tanto por humanos como por máquinas.

Existen varios tipos de sistemas de representación de conocimiento, algunos de los más comunes son [4]:

- **Redes semánticas**: representan el conocimiento de forma simple y visual, mediante un grafo dirigido etiquetado. Está formada por nodos (conceptos) y arcos (relaciones binarias entre los conceptos). Entre las relaciones que se suelen usar entre las redes semánticas, encontramos *Subclase-de* e *Instancia-de*, que sirven para representar la herencia entre conceptos. La herencia se corresponde con el razonamiento de que *las subclases y las instancias heredan las propiedades de las clases más generales*. Esta estructura se suele utilizar por ejemplo para representar relaciones de hiponimia e hiperonimia entre conceptos.

- **Marcos o *Frames***: Cada frame representa un objeto o un concepto relevante, e incluye las propiedades del mismo. Un conjunto de frames trata de representar el conocimiento de un dominio de interés, y está organizado de forma jerárquica en una taxonomía. Estos frames se dividen en frames *clase* (frames genéricas, representan conocimiento de clases de objetos) y frames *instancia* (representan conocimiento de objetos individuales. Esta estructura también sirve para representar herencia entre conceptos. Cada frame hereda las propiedades del frame padre. Por tanto los frames están relacionados con las redes semánticas en tanto que podemos representar una red semántica con un conjunto de frames.

- **Reglas de producción**: Se utilizan en sistemas expertos para representar el conocimiento de forma condicional. Consisten en conjuntos de reglas del tipo *Si-Entonces*, donde se especifica qué hacer si se cumple cierta condición. La estructura general de una regla de producción es la de *Antecedente* $\implies$ *Consecuente*, donde el antecedente contiene las cláusulas que deben cumplirse para aplicar la regla y el consecuente indica las conclusiones o acciones a realizar por el sistema.
  Como ejemplo de aplicación de las reglas de producción, podemos considerar un sistema diagnóstico médico, donde se diagnostique una patología en base a los síntomas del paciente. Una regla de este sistema podría ser *Si el paciente tiene fiebre y dolor de garganta, entonces diagnosticar una infección de garganta*.

- **Ontologías**: Como ya vimos en 1.4.6, son modelos conceptuales que representan un conjunto de conceptos y las relaciones entre ellos en un dominio específico. Gracias a las ontologías podemos definir términos y sus relaciones de manera formal, para facilitar el intercambio de datos entre varios sistemas informáticos.

- **Lógica de Predicados**: Es un formalismo matemático para representar el conocimiento mediante predicados y cuantificadores. Permite expresar proposiciones y relaciones de manera precisa, lo cual facilita el razonamiento lógico y la inferencia en bases de conocimiento.

1.5.3. Minería de textos y Minería de bases de conocimiento

De este modo, la minería de bases de conocimiento se podría situar en un marco más general que el de la minería de textos, ya que abarca la extracción de información y conocimiento de conjuntos de datos que van más allá del texto no estructurado. Mientras que la minería de textos está principalmente enfocada en el análisis de grandes cantidades de textos sin formato, la minería de bases de conocimiento se extiende para incluir cualquier sistema informático

que contenga información valiosa. Esto puede incluir desde bases de datos relacionales hasta repositorios de información y sistemas de gestión de documentos, entre otros.

1.5.4. Relación de las bases de conocimiento con el proyecto

En este proyecto, se llevará a cabo la implementación de los pasos previamente estudiados del proceso de KDT, con el objetivo de transformar una base de datos textual inicial en un formato que sea procesable por una máquina. Esto se hará en primer lugar mediante técnicas de preprocesamiento típicas del KDT. Además de esto, se podrá hacer uso de estructuras de conocimiento tales como ontologías y redes semánticas, que ayuden a estructurar el texto de forma más efectiva, teniendo ahora en cuenta los conceptos y las relaciones semánticas entre ellos.

Por ejemplo, mediante la detección de sinónimos dentro de un corpus, podríamos agrupar palabras que en un principio no tendrían relación entre sí si sólo aplicáramos las técnicas estándar de preprocesamiento. Además, mediante redes semánticas podemos agrupar distintas palabras si están vinculadas jerárquicamente mediante relaciones de hiponimia e hiperonimia.

Es por esto que hemos elegido el título *Minería de bases de conocimiento* en lugar de *Minería de bases de datos textuales*. Hemos considerado que el proceso de KDT abarca un marco más extenso que la simple aplicación de algoritmos de detección de patrones, y que con el respaldo de herramientas de Ingeniería de Conocimiento, podemos enriquecer el proceso de Descubrimiento de Conocimiento para lograr resultados más reveladores.

2. *Spam*

En este capítulo vamos a abarcar el concepto de *spam*, los tipos de *spam*, así como las distintas técnicas existentes en la literatura para detectarlo.

2.1. Definición e historia

El *spam* es el envío masivo y no solicitado de mensajes electrónicos, ya sea por correo electrónico, mensajes de texto, redes sociales y otros medios digitales. Generalmente los mensajes *spam* tienen como objetivo promocionar productos o servicios. A veces, estos mensajes también son usados para engañar a los usuarios para que revelen información personal, como contraseñas o información financiera. En contraste con el spam, el término *ham* se refiere a los mensajes de correo electrónico legítimos y deseados, que son enviados por remitentes conocidos o esperados. Estos mensajes suelen ser correos electrónicos personales, comerciales o informativos que se envían a destinatarios que han optado por recibirlos, por lo que no presentan ningún tipo de amenaza o molestia.

El término *spam* en el contexto de los mensajes de correo electrónico, se popularizó en la década de los años 90. El primer mensaje de *spam* conocido tiene su origen 1994, en un post de Usenet, donde una firma de abogados publicó un mensaje de anuncio de su firma legal. Desde entonces, la publicidad y el marketing mediante correo electrónico ha crecido a niveles impensables [5].

Según cifras del informe Kaspersky [6], se estima que en 2022 el 48.63 % de correos electrónicos en todo el mundo fueron *spam*.

2.2. Tipos de *spam*

El *spam* no es solo molesto para los usuarios, sino que también puede presentar una amenaza y un riesgo para la preservar la seguridad y la privacidad online. Los *spammers*, quienes envían estos mensajes no deseados, utilizan diversas técnicas para difundir dicho contenido no solicitado. A continuación vamos a exponer los distintos tipos de *spam* conocidos, según su naturaleza y el riesgo que representan [7]:

- **Correo electrónico no deseado:** es el tipo de *spam* más común y todos estamos habituados a encontrarlo. Con frecuencia inundan nuestra bandeja de entrada, resultando muy molesto al usuario. Tiene fines publicitarios o promocionales, por lo que no suelen representar un riesgo directo para la seguridad del usuario, aunque consumen recursos de almacenamiento.

- ***Spam* SEO:** también conocido como *spamdexing*, se refiere a la manipulación de los métodos de optimización de los motores de búsqueda para mejorar la clasificación de un sitio web y lograr que más usuarios accedan a él. Se puede clasificar en dos categorías:

    - ***Spam* de contenido:** los *spammers* llenan la página de palabras clave populares que el sitio web aparezca más arriba en las búsquedas.
    - ***Spam* de enlaces:** se trata de comentarios que podemos encontrar en foros o redes sociales, y que contiene un enlace externo que conduce a otro sitio web. Así se consigue atraer tráfico a la página web.

- ***Spam* en redes sociales:** se trata de perfiles y cuentas falsas creadas de forma masiva con el fin de propagar un mensaje *spam*, por ejemplo, propaganda de ideas de índole política o enlaces a sitios web externos.
- ***Spam* de mensajería:** es similar al *spam* por correo electrónico pero en SMS y en plataformas de mensajería instantánea tales como WhatsApp o Telegram.
- **Estafas de soporte técnico:** suelen comenzar con la llamada telefónica de alguien haciéndose pasar por empleado de una empresa. El estafador trata de convencer al usuario de que hay un problema con su cuenta o el producto contratado, pidiendole información sensible.
- ***Spam* de malware:** este *spam* contiene enlaces o archivos que pueden comprometer la seguridad del usuario simplemente con un click o una descarga. Suele llegar a través de un mensaje de texto o un correo electrónico no deseado.

2.3. *Spam* convencional y *Phishing*

A pesar de la molestia que representa el *spam* convencional, este no suele resultar dañino, pues su objetivo es anunciar productos o servicios. Sin embargo, el *phishing* es mucho más peligroso, pues supone el envío de correos fraudulentos que se hacen pasar por instituciones legítimas (red social, banco, institución pública, etc.) [8]. Estos mensajes engañosos buscan obtener información confidencial, como contraseñas o números de tarjetas de crédito, con el fin de cometer robo de identidad o fraude financiero. Es de suma importancia que los usuarios estén capacitados para reconocer los patrones de estos intentos de *phishing* y que duden antes de enviar información sensible a terceros.

2.4. Técnicas de filtrado de *spam*

Tal y como hemos mencionado en la anterior sección, detectar el *spam* es fundamental para evitar caer en fraudes y timos online. El *spam*, especialmente en forma de *phishing*, es hoy en día la herramienta principal usada por los estafadores para engañar a los usuarios y acceder a su información confidencial.

Con el fin de proteger a los usuarios, los proveedores de servicios de electrónico han desarrollado durante estas décadas numerosas técnicas de detección de *spam*. En esta sección vamos a revisar el conjunto de técnicas de detección de *spam* que podemos encontrar en la literatura [9] [10]:

- **Listas negras y blancas:** también conocidas como *blacklists* y *whitelists*, este fue uno de los primeros métodos utilizados para detectar correo *spam*. Están basadas en la exclusión de mensajes que provienen de ciertos dominios, redes o servidores de Internet. Mediante este método se pueden identificar grandes cantidades de correo no deseado,

aunque bien es cierto que es fácilmente falsificable y manipulable.
Algunas de estas listas se encuentran en forma de ficheros de texto que contienen directamente remitentes o expresiones regulares de direcciones de correo electrónico.
Por el contrario, las listas blancas contienen direcciones de correo *verificadas* en las que se puede confiar. Este mecanismo, aunque simple, es muy difícil de burlar. Sin embargo, existe el inconveniente de que las empresas verificadas y exentas de estas listas pueden seguir enviando boletines de suscripción y publicidad en forma de *spam*.

- **Resúmenes:** son aplicaciones cuyo funcionamiento se basa en la creación de una representación compacta y única (resumen) de un correo electrónico. Se utilizan algoritmos de resumen, como *Nilsimsa* [11]; estos algoritmos generan una firma única para cada mensaje basándose en su contenido, de manera que incluso pequeñas modificaciones en el mensaje producen un cambio significativo en el resumen. Esto nos permite detectar variantes triviales de mensajes, como números aleatorios en el asunto o cambios mínimos en el contenido.

- **Modelos basados en contenido:** estas técnicas hacen uso del Machine Learning para determinar patrones y relaciones entre características de los mensajes clasificados como Spam y Ham. Normalmente la forma de representación de los datos es mediante un vector de características por cada correo. Las características del vector contienen una lista de palabras representativas de la legitimidad de los mensajes.
  La elección de los términos más representativos de cada mensaje se realiza mediante técnicas de selección de características. La técnica más habitual es el cálculo de la ganancia de información (IG, *Information Gain*) de cada término con respecto a los posibles valores del atributo a predecir (Spam o Ham). Se acaba tomando los términos dentro del corpus con mayor ganancia de información.
  Otra de las técnicas comúnmente empleadas en el cálculo de representatividad de un término es la frecuencia de documentos que contienen un término dado (DF, *Document Frecuency*).

- **Técnicas basadas en análisis heurístico o en reglas:** A diferencia de los enfoques basados en aprendizaje automático, el análisis heurístico no requiere un conjunto de datos de entrenamiento, sino que se basa en el conocimiento previo de los patrones típicos de *spam*. Este enfoque usa reglas o heurísticas ya creadas para evaluar una gran cantidad de patrones, que suelen ser expresiones regulares. Si el email estudiado coincide con varios de los patrones evaluados, esto va aumentando su puntuación. A su vez, si alguno de los patrones no coincide, se resta puntuación. Se establece un umbral de *spam*, y cualquier mensaje cuya puntuación lo supere se filtra como Spam; de lo contrario se considera válido.
  Estas reglas están sujetas a actualizaciones frente a la amenaza de los *spammers*, que continuamente presentan nuevos tipos de mensajes spam que podrían pasar inadvertidos ante los filtros antiguos. Un ejemplo de filtro basado en reglas es *SpamAssassin* [12].

- **Métodos basados en casos:** el filtrado basado en ejemplos es una de las técnicas de filtrado de *spam* más populares. En este enfoque, se toman todos los correos de ambas clases (Spam y Ham) y se crea una colección. Luego se llevan a cabo los pasos de preprocesamiento para transformar el email, como selección de características, agrupación de datos, etc. Los datos son clasificados en dos conjuntos de vectores. Se utiliza un algoritmo de Aprendizaje Automático (basado en instancias) para entrenar datasets y testearlos para decidir si son Spam o Ham.

Podemos encontrar en la literatura existente numerosos métodos de filtrado de *spam* en emails. De entre las técnicas anteriomente mencionadas, algunas de ellas aplican distintos algoritmos de *Machine Learning*. A continuación vamos a exponer varias de las distintas técnicas de filtrado de *spam* que han sido usadas con éxito en los últimos años [9].

- **Naive Bayes**: es el algoritmo de Aprendizaje Automático más conocido en clasificación de textos. Este modelo ha adquirido gran popularidad por su capacidad de representar de forma simple y eficiente distribuciones complejas de probabilidad, además de su fácil implementación y rápida convergencia. El algoritmo se basa en el Teorema de Bayes y asume la independencia de los atributos, lo cual es una simplificación poco realista; no obstante, ha demostrado gran efectividad en numerosos estudios [13]. Se puede usar para resolver problemas de clasificación de dos o más clases, y maneja tanto datos continuos como discretos (siempre y cuando sean numéricos).

- **Máquinas de soporte vectorial** (SVM, *Support Vector Machines*): este tipo de modelos destaca por su sólida base teórica. El algoritmo se basa en la transformación de los datos existentes para encontrar un hiperplano de mayor dimensión donde maximinzar la separación existente entre las clases. Cabe destacar que con SVM no es necesario realizar selección de características en la fase de Preprocesamiento, pues su capacidad de aprendizaje no se ve afectada por haber demasiados atributos.

- **Boosting de Árboles de Decisión:** se basa en tomar varios algoritmos de aprendizaje débiles y combinar las hipótesis generadas en una única hipótesis de gran precisión. Para ello, el algoritmo se ejecuta varias veces sobre distintos subconjuntos de entrenamiento. Normalmente en filtrado de *spam* se combinan algoritmos como AdaBoost y C4.5.

- **Random Forest**: es un modelo basado en una colección de árboles de decisión, entrenado cada uno a partir de un subconjunto del corpus, de forma aleatoria. Para clasificar un correo nuevo, se pasan las características del correo a cada árbol, y estos emiten cada uno un voto unitario; entonces se le asigna al correo la clase con mayor número de votos. En este algoritmo es crucial tomar un número adecuado de atributos, pues afecta de forma directamente proporcional a la correlación y a la fortaleza. Para su ajuste se suele emplear la tasa de error. Los estudios de filtrado de *spam* realizados con este clasificador han demostrado obtener un alto grado de precisión [14].

- **k-NN (*K-Nearest Neighbors*)**: es uno de los algoritmos más empleados en clasificación basada en casos o instancias. En este método, cada ejemplo de correo electrónico se representa como un punto en un espacio multidimensional, donde las dimensiones son las características del correo electrónico (por ejemplo, palabras clave, frecuencia de palabras, etc.). Luego, cuando llega un nuevo correo electrónico, se compara con los ejemplos de entrenamiento y se clasifica según la mayoría de los k correos electrónicos más cercanos en términos de similitud. Si la mayoría de los k vecinos son correos electrónicos de *spam*, entonces el nuevo correo electrónico también se clasificará como Spam. Este método es eficaz para detectar patrones sutiles en los datos y es relativamente simple de implementar.

Como hemos visto, el filtrado de emails *automático* parece ser actualmente el enfoque más exitoso para el filtrado de *spam*. Hace años, la mayor parte del correo no deseado podía detectarse simplemente analizando las direcciones de remitente y los asuntos de los emails,

mediante listas negras. Sin embargo, los *spammers* adoptaron técnicas más sofisticadas como el uso de direcciones arbitrarias y la inserción de caracteres al principio o final de la línea de asunto del mensaje.

Actualmente existen un gran número de filtros que hacen uso de una combinación de *Machine Learning* y conjuntos reglas generadas a partir de la Ingeniería de Conocimiento. El problema principal de utilizar reglas, es que este método no garantiza un resultado eficiente, ya que en primer lugar se debe generar un conjunto reglas (esto requerirá la ayuda de expertos en el tema) y en segundo lugar es preciso actualizar continuamente dicho conjunto. Esto puede llevar a una pérdida de tiempo y no es adecuado especialmente para usuarios inexpertos. En cambio, aplicando *Machine Learning*, no se requiere especificar ninguna regla, sino que se proporciona un conjunto de muestras de entrenamiento que son emails previamente clasificados. Por tanto el enfoque de Aprendizaje Automático ha demostrado ser más eficiente que el enfoque de Ingeniería de Conocimiento, al menos con respecto al esfuerzo que implica construir el sistema de filtrado.

3. Fundamentos matemáticos

En el contexto de la detección de correos *spam*, podemos explorar principalmente dos posibles tipos de aprendizaje: supervisado y no supervisado [15].

El **aprendizaje supervisado** consiste en entrenar un modelo a partir un conjunto de datos etiquetados, donde cada ejemplo de entrada está asociado con una etiqueta correcta (en este caso, Spam o Ham). Este enfoque es el más adecuado para el problema en cuestión, ya que permite al modelo aprender las características que distinguen los correos Spam de los Ham a partir de ejemplos previamente etiquetados. Ejemplos de técnicas de aprendizaje no supervisado serían Árboles de Decisión, Regresión lineal, Redes neuronales, k-NN, SVM y métodos probabilísticos.

Por otro lado, en el **aprendizaje no supervisado** no disponemos de datos etiquetados. En su lugar, intenta identificar patrones o estructuras subyacentes en los datos de entrada. Aunque puede ser útil para descubrir grupos de datos similares o para reducción de dimensionalidad, no es el enfoque preferido para la detección de correos electrónicos debido a la falta de orientación explícita sobre lo que constituye Spam frente a Ham. No obstante estos métodos pueden ayudar a revelar patrones ocultos, agrupaciones o anomalías en los datos que no serían evidentes de otra manera. Ejemplos de técnicas de aprendizaje no supervisado serían Clustering (K-means, DBSCAN...), detección basada en reglas de asociación y métodos de detección de anomalías (Isolation Forest, One-Class SVM).

Existen otros dos tipos de aprendizaje: el **aprendizaje semi-supervisado** y el **aprendizaje por refuerzo**. Estos enfoques se utilizan en contextos específicos donde se dispone de una cantidad limitada de datos etiquetados (semi-supervisado) o donde un agente aprende a tomar decisiones a través de recompensas (aprendizaje por refuerzo). Como hemos visto en el punto 2.4, las técnicas comúnmente utilizadas en la literatura para el filtrado de *spam* no parecen usar este tipo de aprendizaje.

En este trabajo nos vamos a centrar en el aprendizaje supervisado. En particular, dada la naturaleza binaria del problema de filtrado de correos (puede ser *spam* o no serlo), vamos a aplicar algoritmos de clasificación. La detección de correos spam es un claro ejemplo de un problema de clasificación binaria, donde cada correo electrónico se clasifica en una de dos categorías (Spam o Ham). Este problema se presta idealmente al uso de técnicas de clasificación, ya que podemos entrenar modelos utilizando conjuntos de datos previamente etiquetados para aprender a distinguir entre estas dos categorías.

La tarea de **clasificación** consiste en predecir la clase de una instancia no etiquetada. Formalmente, un clasificador es un modelo o función M que predice la clase $\hat{y}$ para un ejemplo de entrada x. Es decir, $\hat{y} = M(x)$, donde $\hat{y} \in \{c_1, c_2, ..., c_k\}$ es la clase predicha [16]. Para construir este modelo se requiere un conjunto de instancias etiquetadas correctamente,

llamado *conjunto de entrenamiento*. Una vez que el modelo M ha sido entrenado, podemos predecir automáticamente la clase para cualquier nueva instancia; el conjunto de nuevas instancias es denominado *conjunto de prueba o test*.

A lo largo de la literatura se han propuesto diversos modelos de clasificación, como árboles de decisión, clasificadores probabilísticos, máquinas de soporte vectorial y un largo etcétera, como hemos visto con más detalle en el punto 2.4. En este capítulo vamos a estudiar algunos de estos tipos de clasificadores desde sus cimientos matemáticos.

Entre los algoritmos seleccionados en este proyecto se encuentran el de Naive Bayes y las Máquinas de Soporte Vectorial (SVM, *Support Vector Machines*). Ambos algoritmos tienen sus fortalezas en el contexto de la clasificación de texto y cada uno posee características distintivas que ofrecen ventajas en términos de precisión, eficiencia y capacidad para manejar los datos de los que dispondremos, sobre otros algoritmos [16].

- Los **clasificadores probabilísticos o Bayesianos**, como Naïve Bayes, calculan la probabilidad de que una instancia pertenezca a una clase, basándose en la frecuencia de las palabras. Este clasificador resulta especialmente efectivo debido a su simplicidad (al suponer que los atributos son independientes), permitiendo que el algoritmo sea extremadamente eficiente tanto en tiempo de entrenamiento como de predicción. A pesar de su suposición simplista de independencia, en la práctica, Naive Bayes funciona sorprendentemente bien para la clasificación de textos. Además, Naive Bayes es robusto ante características irrelevantes (que suelen reducir la precisión), lo cual es una ventaja en conjuntos de atributos de alta dimensión donde no todas las características son igualmente informativas.

- Las **Máquinas de Soporte Vectorial (SVM)**, buscan el hiperplano que mejor separa las clases en el espacio de características. Es particularmente útil en espacios de alta dimensión, donde el número de características supera el número de muestras, lo cual es común en clasificación de textos. Además, es conocido por su robustez y precisión, pues tiene gran capacidad para manejar la alta dimensionalidad sin caer en el sobreajuste (gracias a la selección de hiperplanos óptimos). A diferencia de Naive Bayes, SVM no asume independencia entre las características, lo que permite capturar interacciones más complejas entre ellas.

La combinación de Naive Bayes y SVM permite abordar el problema de clasificación de correos *spam* desde dos perspectivas complementarias: una probabilística y otra geométrica, maximizando así la capacidad del sistema para identificar correctamente los correos *spam*.

3.1. Clasificadores Bayesianos

Los clasificadores bayesianos son un ejemplo del enfoque probabilístico en la tarea de clasificación. Este enfoque tiene su base en el *Teorema de Bayes*, para predecir la clase que maximice la probabilidad resultante dada una serie de eventos previos, $P(c_i|x)$. Una de las limitaciones del enfoque bayesiano es que el número de parámetros a estimar asciende a $O(d^2)$. Para mejorar la eficiencia computacional, los distintos clasificadores existentes en la literatura han adoptado algunas simplificaciones al problema. Por ejemplo, el clasificador *Naïve Bayes* asume que todos los atributos son independientes entre sí, lo que reduce drásticamente la

cantidad de parámetros a estimar a $O(d)$ [16]. A pesar de que en la práctica, la idea de que los atributos son independientes no se suele cumplir, sorprendentemente estos clasificadores consiguen resultados muy efectivos.

Para una mejor comprensión del funcionamiento de un clasificador bayesiano, es necesario definir algunos conceptos básicos de probabilidad, que abordaremos a continuación.

3.1.1. Conceptos básicos de probabilidad

La probabilidad es una herramienta esencial en el estudio de los fenómenos aleatorios, aquellos cuyo resultado no puede predecirse con certeza. A continuación, exploraremos los conceptos clave que nos permitirán entender mejor qué es la probabilidad y cómo se aplica en el contexto de la clasificación bayesiana [17].

3.1.1.1. Fenómenos y experimentos aleatorios

La Teoría de la probabilidad estudia el comportamiento de los fenómenos o experimentos aleatorios. En primer lugar es crucial distinguir entre experimentos determinísticos y experimentos aleatorios.

Definición 3.1. Un *experimento determinístico* es aquel que siempre da lugar al mismo resultado bajo las mismas condiciones. Un *experimento aleatorio* es aquel cuyo resultado puede variar, a pesar de realizarse bajo idénticas condiciones.

Un ejemplo de experimento aleatorio es el de tirar un dado. Siempre estamos tirando el dado bajo las mismas condiciones; sin embargo, cada vez sale un número distinto, el cual no podemos predecir.

Una característica clave de los experimentos aleatorios es que podemos estudiar el conjunto de posibles resultados del experimento y su frecuencia, pero *nunca* podemos predecir un resultado particular. En el ejemplo del dado conocemos la probabilidad de que salga un número ($\frac{1}{6}$), pero no podemos predecir qué número saldrá a continuación.

3.1.1.2. Espacio muestral y Álgebra de sucesos

Definición 3.2. Denominamos *espacio muestral* al conjunto de todos los posibles resultados de un experimento aleatorio, y se denota por Ω.

En el experimento aleatorio de tirar un dado, el espacio muestral es $\Omega = \{1, 2, 3, 4, 5, 6\}$

Definición 3.3. Llamamos *suceso aleatorio* (o simplemente *suceso*) a un subconjunto del espacio muestral Ω. Decimos que se da un suceso si ocurre alguno de sus elementos como resultado del experimento aleatorio.

Por ejemplo, en el caso del dado, el suceso *obtener un número par* sería el conjunto $\{2, 4, 6\}$, que contiene los resultados 2, 4 y 6. Destacamos cuatro tipos de suceso según el número de elementos que contenga:

- **Suceso elemental**: representa cada uno de los posibles resultados del experimento aleatorio. Por tanto está formado por un único elemento del espacio muestral.

- **Suceso compuesto**: consta de dos o más elementos del espacio muestral.
- **Suceso seguro** (Ω): es aquel suceso que siempre va a ocurrir. Está formado por todos los elementos del espacio muestral.
- **Suceso imposible** ($\emptyset$): es aquel suceso que nunca va a ocurrir. No contiene ningún elemento del espacio muestral.

Esta definición de suceso como un subconjunto de Ω nos permitirá aplicar conceptos de Teoría de Conjuntos a partir de ahora. Podemos definir varias **operaciones y relaciones entre sucesos** [17]:

- **Suceso contenido en otro:** dados dos sucesos A y B de un experimento aleatorio, diremos que A está contenido en B ($A \subset B$) si siempre que ocurre el suceso A, también ocurre el suceso B.
- **Igualdad de sucesos:** dados dos sucesos A y B de un experimento aleatorio, diremos que son iguales si cada vez que ocurre A también ocurre B y viceversa.

$$A = B \iff A \subset B \text{ y } B \subset A$$

- **Suceso complementario o contrario:** dado un suceso A, se define su suceso contrario como aquel suceso que ocurre si o solo si no ocurre A. Se denota por $\overline{A}$.
- **Unión de sucesos:** dados dos sucesos A y B de un experimento aleatorio, la unión de ambos es el suceso que ocurre siempre que ocurra el A, el B o ambos a la vez. Se denota por $A \cup B$.
- **Intersección de sucesos:** dados dos sucesos A y B de un experimento aleatorio, la intersección de ambos es el suceso que ocurre cuando ocurren A y B simultáneamente. Se denota por $A \cap B$.
- **Diferencia de sucesos:** dados dos sucesos A y B de un experimento aleatorio, la diferencia de ambos es el suceso que ocurre siempre que ocurra A pero no ocurra B. Se denota por $A - B$.
- **Sucesos disjuntos o incompatibles:** dos sucesos A y B son disjuntos si no pueden ocurrir simultáneamente, es decir, si siempre que ocurre uno no se verifica el otro. Esto se cumple cuando la intersección $A \cap B = \emptyset$

Ahora, considerando estos sucesos, podemos avanzar hacia el concepto de *Álgebra de sucesos*. Este concepto nos permite realizar operaciones y establecer relaciones entre los sucesos de manera similar a como lo hacemos en el álgebra convencional.

Veremos que un *álgebra de sucesos* sobre un espacio muestral Ω es una colección de sucesos que cumple ciertas condiciones.

Definición 3.4. Sea Ω un conjunto arbitrario y $\mathcal{A} \subseteq \mathcal{P}(\Omega)$ una clase no vacía de subconjuntos de Ω. La clase $\mathcal{A}$ tiene estructura de *Álgebra de sucesos* si cumple lo siguiente:

1. *Es cerrada para la operación complementario*: $\forall A \in \mathcal{A}$, se verifica que $\overline{A} \in \mathcal{A}$.
2. *Es cerrada para uniones finitas*: $\forall A, B \in \mathcal{A}$, se verifica $A \cup B \in \mathcal{A}$.

A partir de esta definición, podemos deducir de forma inmediata las siguientes propiedades:

- El espacio muestral $\Omega \in \mathcal{A}$.
- El suceso imposible $\emptyset \in \mathcal{A}$.

Hemos observado cómo los sucesos de un espacio muestral pueden ser tratados como conjuntos, permitiéndonos aplicar operaciones y relaciones entre ellos, como uniones, intersecciones y complementos sobre ellos.

Si consideramos un experimento aleatorio con espacio muestral no finito, la clase de sucesos de interés podría no ser finita, de tal forma que la estructura de álgebra dada sería insuficiente para describir la clase de sucesos $\mathcal{A} \subseteq \mathcal{P}(\Omega)$. A continuación, estudiaremos el concepto de σ-álgebra, que va un paso más allá al considerar también uniones *numerables* de sucesos.

Definición 3.5. Sea $\mathcal{A} \subseteq \mathcal{P}(\Omega)$ una clase no vacía de sucesos de Ω. La clase $\mathcal{A}$ tiene estructura de *σ-álgebra de sucesos* si cumple lo siguiente:

1. *Es cerrada para la operación complementario*: $\forall A \in \mathcal{A}$, se verifica que $\overline{A} \in \mathcal{A}$.
2. *Es cerrada para uniones numerables*: $\forall A_1, A_2, A_3, ... \in \mathcal{A}$, se verifica que

$$A_1 \cup A_2 \cup A_3 \cup ... = \bigcup_{i=1}^{\infty} A_i \in \mathcal{A}.$$

Al igual que en el álgebra de sucesos, el total Ω y el vacío $\emptyset$ pertenecen a la σ-álgebra. Observamos que toda σ-álgebra es un álgebra.

3.1.1.3. Probabilidad

Teniendo presentes las definiciones de los apartados anteriores, nos vemos en las condiciones de definir el concepto de probabilidad. De forma intuitiva, la probabilidad de un suceso indica la certeza con la que puede ocurrir dicho suceso. Dicho de otra forma, la probabilidad se podría ver como la frecuencia con la que ocurre un suceso con respecto a la frecuencia con la que ocurren los demás dentro de un espacio muestral. En esta sección vamos a estudiar la definición axiomática, aprovechando que hemos definido anteriormente la estructura de σ-álgebra.

Definición 3.6 (Definición axiomática de Kolmogorov). Sea $(\Omega, \mathcal{A})$ el espacio medible asociado a un experimento aleatorio. Se define una probabilidad como una función de conjunto $P : \mathcal{A} \longrightarrow [0, 1]$ que verifica los siguientes axiomas:

A1 *Axioma de no negatividad*: $P(A) \geq 0, \forall A \in \mathcal{A}$.

A2 *Axioma del suceso seguro*: $P(\Omega) = 1$.

A3 *Axioma de σ-aditividad*: Dada $\{A_i\}_{i \in \mathbb{N}} \subseteq \mathcal{A}$ una familia numerable de sucesos con $A_i \cap A_j = \emptyset, \forall i \neq j$, entonces

$$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$$

Al tomar el par $(\Omega, \mathcal{A})$, obtenemos un espacio medible, lo que significa que tenemos un marco matemático sobre el cual podemos definir y calcular probabilidades de eventos. Cada conjunto en $\mathcal{A}$ representa un posible evento en nuestro espacio muestral, y podemos asignar probabilidades a estos eventos de manera coherente y consistente.

Definición 3.7. Dados un espacio muestral Ω sobre un experimento aleatorio, su σ-álgebra de sucesos $\mathcal{A}$ y P la función de probabilidad sobre $(\Omega, \mathcal{A})$, definimos como $(\Omega, \mathcal{A}, P)$ el *espacio de probabilidades* o *espacio probabilístico* asociado al experimento.

Como consecuencia de la definición 3.6, podemos deducir varias propiedades asociadas a la probabilidad, dentro de un espacio probabilístico:

1. La probabilidad del suceso imposible es nula: $P(\emptyset) = 0$.
2. La probabilidad del complementario de un suceso $A \in \mathcal{A}$ es $P(\overline{A}) = 1 - P(A)$
3. La probabilidad P es monótona no decreciente, es decir,

   $$\forall A, B \in \mathcal{A}, \text{ con } A \subset B \implies P(A) \leq P(B)$$

   Además, $P(B - A) = P(B) - P(A)$.
4. Todo suceso $A \in \mathcal{A}$ verifica que $P(A) \leq 1$.
5. Dados dos sucesos cualesquiera $A, B \in \mathcal{A}$, se verifica que

   $$P(B - A) = P(B) - P(A \cap B)$$

6. Dados dos sucesos cualesquiera $A, B \in \mathcal{A}$, se verifica que

   $$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

7. Subaditividad finita: Dados $A, B \in \mathcal{A}$, se cumple que

   $$P(A \cup B) \leq P(A) + P(B)$$

   En general, dados $A_1, A_2, ..., A_n \in \mathcal{A}$, entonces

   $$P(\bigcup_{i=1}^{n} A_i) \leq \sum_{i=1}^{n} P(A_i)$$

8. Subaditividad numerable: Dada $\{A_i\}_{i=1}^{\infty} \subset \mathcal{A}$ una familia numerable de sucesos, entonces

   $$P(\bigcup_{i=1}^{\infty} A_i) \leq \sum_{i=1}^{\infty} P(A_i)$$

9. Desigualdad de Boole: Dados $A, B \in \mathcal{A}$, entonces

   $$P(A \cap B) \geq 1 - P(\overline{A}) - P(\overline{B})$$

Retomando el ejemplo del dado, supongamos que ya sabemos que ha ocurrido el evento "Ha salido un número mayor que 3". A partir de esta información, podríamos estar interesados en calcular la probabilidad de que el resultado además sea un número par. Este escenario nos hace ver la necesidad de una herramienta matemática específica que nos permita calcular la probabilidad de un suceso, a sabiendas de que otro suceso ya ha ocurrido. Este cálculo se realiza a través de la *probabilidad condicionada*.

3.1.1.4. Probabilidad condicionada

En este apartado exploraremos el concepto de *probabilidad condicionada*, una herramienta esencial para evaluar la probabilidad de un evento bajo la condición de que otro evento ha ocurrido. Profundizaremos en cómo este principio se extiende al *Teorema de Bayes*, permitiéndonos actualizar probabilidades a partir de nueva evidencia. Finalmente discutiremos el concepto de *independencia*, que se usa para simplificar el cálculo de probabilidades conjuntas. Estos fundamentos nos permitirán hacer inferencias más precisas y tomar decisiones en situaciones inciertas.

Definición 3.8. Sea $(\Omega, \mathcal{A}, P)$ un espacio probabilístico arbitrario y sea $A \in \mathcal{A}$ un suceso tal que $P(A) > 0$. Dado otro suceso $B \in \mathcal{A}$, definimos la *probabilidad de B condicionada a A* como

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

Observación 3.1. Observamos que de la propia definición se tiene:

$$P(A \cap B) = P(A)P(B|A), \text{ si } P(A) > 0$$

o bien

$$P(A \cap B) = P(B)P(A|B), \text{ si } P(B) > 0$$

Teorema 3.1 (de la probabilidad compuesta). *Sea $(\Omega, \mathcal{A}, P)$ un espacio probabilístico y sean $A_1, A_2, ..., A_n \in \mathcal{A}$ sucesos tal que $P\left[\bigcap_{i=1}^{n-1} A_i\right] > 0$. Entonces,*

$$P\left[\bigcap_{i=1}^{n} A_i\right] = P(A_1) \cdot P(A_2|A_1) \cdot P(A_3|A_1 \cap A_2) \cdot ... \cdot P\left[A_n| \bigcap_{i=1}^{n-1} A_i\right]$$

Teorema 3.2 (de la probabilidad total). *Sea $(\Omega, \mathcal{A}, P)$ un espacio probabilístico y sea $\{A_n\}_{n\in\mathbb{N}} \subset \mathcal{A}$ una familia de sucesos tal que $P(A_n) > 0$, $\forall n \in \mathbb{N}$. Sea $B \in \mathcal{A}$ un suceso cualquiera. Entonces,*

$$P(B) = \sum_{n=1}^{\infty} P(A_n)P(B|A_n)$$

Tras esto, nos encontramos en las condiciones para presentar el *Teorema de Bayes*. Formulado en el s.XVIII, el Teorema de Bayes ofrece un marco matemático para actualizar la probabilidad de una hipótesis a medida que se dispone de más evidencia. En esencia, el Teorema de Bayes permite calcular la probabilidad de que ocurra un evento, basándose en el conocimiento previo de condiciones que podrían o no estar relacionadas con dicho evento.

Teorema 3.3 (Regla de Bayes o Teorema de la probabilidad inversa). *Sea $(\Omega, \mathcal{A}, P)$ un espacio probabilístico y sea $\{A_n\}_{n\in\mathbb{N}} \subset \mathcal{A}$ una familia de sucesos tal que $P(A_n) > 0$, $\forall n \in \mathbb{N}$. Sea $B \in \mathcal{A}$*

un suceso cualquiera con $P(B) \neq 0$. *Entonces,*

$$P(A_n|B) = \frac{P(B|A_n)P(A_n)}{\sum_{n\in\mathbb{N}} P(B|A_n)P(A_n)}$$

Detrás de este cálculo de probabilidades se esconde el siguiente razonamiento: supongamos que el suceso B es el resultado de aplicar un experimento, mientras que los sucesos A_n son todas las posibles causas de que ocurra el suceso B. Supongamos además que para cada causa A_n conocemos su *probabilidad a priori* $P(A_n)$ y la *verosimilitud* $P(B|A_n)$ de que el suceso B haya sido causado por A_n. Entonces la aplicación del Teorema de Bayes nos permite entender $P(A_n|B)$ como una *propiedad a posteriori* de que la verdadera causa de B haya sido A_n.

A partir de ahora, todo este apartado va a girar en torno al Teorema de Bayes, por lo que es importante comprender las implicaciones de este teorema antes de pasar a lo siguiente. Vamos a ilustrar dichas implicaciones con un ejemplo.

Ejemplo 3.1. Supongamos que tenemos un filtro de correo electrónico cuyo objetivo es clasificar los correos entrantes en Spam o Ham. Sabemos que el 20 % de los correos recibidos están clasificados como Spam. Dicho filtro utiliza palabras clave específicas para identificar el *spam*. Una de estas palabras clave es *ganar*, y se ha observado que aparece en el 30 % de los correos clasificados como Spam y sólo en el 2 % de los correos Ham. Nos planteamos la siguiente pregunta: si recibimos un correo electrónico que contiene la palabra *ganar*, ¿cuál es la probabilidad de que sea realmente *spam*?

Conocemos los siguientes datos:

- Probabilidad de recibir un correo spam: $P(Spam) = 0.2$.
- Probabilidad de recibir un correo ham: $P(Ham) = 1 - P(Spam) = 0.8$.
- Probabilidad de que la palabra *ganar* aparezca en un correo spam: $P(Ganar|Spam) = 0.3$.
- Probabilidad de que la palabra *ganar* aparezca en un correo no spam: $P(Ganar|Ham) = 0.02$.

Aplicando el Teorema de Bayes, podemos encontrar la probabilidad de que un correo sea spam sabiendo que contiene la palabra *ganar* de la siguiente forma:

$$P(Spam|Ganar) = \frac{P(Ganar|Spam)P(Spam)}{P(Ganar)}$$

Primero hay que calcular la probabilidad total de que salga la palabra *ganar*, que sabemos por el Teorema 3.2 que se calcula como

$$P(Ganar) = P(Ganar|Spam)P(Spam) + P(Ganar|Ham)P(Ham)$$

Por tanto $P(Ganar) = 0.3 \cdot 0.2 + 0.02 \cdot 0.8 = 0.076$. Entonces $P(Spam|Ganar) = \frac{0.3 \cdot 0.2}{0.076} = 0.789$. En vista de los resultados, encontramos que la probabilidad de que en correo sea *spam*, dado que contiene la palabra *ganar* es del 78.9 %.

Este ejemplo ilustra cómo el Teorema de Bayes permite a un filtro de correo calcular sus probabilidades basándose en la evidencia (la presencia de ciertas palabras clave), consiguiendo una clasificación más precisa. A pesar de que inicialmente solo el 20 % de los correos estaban clasificados como Spam, la aparición de la palabra *ganar* aumenta significativamente la probabilidad de que dicho correo sea Spam.

Esta idea es fundamental pues así los filtros de correo pueden aprender a partir de correos previamente etiquetados como Spam o Ham, y mejorar su precisión en el tiempo gracias a este conocimiento previo.

Ahora que hemos entendido que hay sucesos cuya probabilidad depende de que ocurran otros sucesos, vamos a estudiar la independencia de sucesos.

Definición 3.9. Sea $(\Omega, \mathcal{A}, P)$ un espacio probabilístico arbitrario y sea $A \in \mathcal{A}$ un suceso tal que $P(A) > 0$. La ocurrencia del suceso A puede alterar la probabilidad de ocurrencia de cualquier otro suceso $B \in \mathcal{A}$. Se pueden dar los casos siguientes:

1. $P(B|A) \neq P(B)$, es decir, la ocurrencia del suceso A modifica la probabilidad de ocurrencia de B. Diremos entonces que **B depende de A**.
   - Si $P(B|A) > P(B)$ se dice que el suceso A *favorece* a B.
   - Si $P(B|A) < P(B)$ se dice que el suceso A *desfavorece* a B.
2. $P(B|A) = P(B)$, es decir, la ocurrencia del suceso A no modifica la probabilidad de ocurrencia de B. Diremos entonces que **B es independiente de A**.

Observación 3.2 (**Caracterización de independencia**).

$$A \text{ y } B \text{ son independientes} \iff P(A \cap B) = P(A) \cdot P(B)$$

3.1.2. Redes probabilísticas

Las *redes probabilísticas* son un tipo de modelos probabilísticos que se caracterizan porque podemos representar de manera natural, a través de grafos, sus distribuciones de probabilidad conjunta. En estos grafos, los nodos representan las variables sobre las que definimos una distribución de probabilidad conjunta. La presencia o ausencia de enlace entre estos nodos representan la dependencia o independencia entre las variables.

Las redes probabilísticas pueden verse como representaciones de *reglas causa-efecto*, mediante las cuales podemos realizar razonamientos deductivos (conclusiones o efecto), abductivos (explicaciones o diagnóstico) e intercausales.

En la imagen 3.1 podemos observar un ejemplo. El razonamiento deductivo sigue la dirección de los enlaces causales entre las variables de un modelo; por ejemplo, si una persona ha contraído un resfriado, podemos concluir (con alta probabilidad) que la persona tiene fiebre y secreción nasal. Por otro lado, el razonamiento abductivo va en contra de la dirección de los enlaces causales; por ejemplo, si observamos que una persona tiene secreción nasal, tendremos evidencia para diagnosticar un resfriado o una alergia.

La propiedad más característica de las redes probabilísticas es la capacidad para hacer razonamiento intercausal. Esto consiste en que si obtenemos evidencia que apoya una hipótesis, la creencia en las demás hipótesis competidoras disminuye. Por ejemplo, en la figura 3.1,

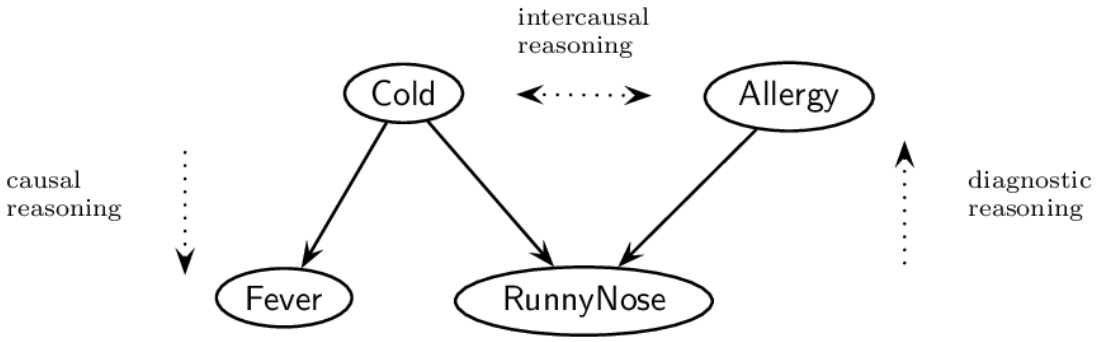


Figura 3.1.: Ejemplo de red probabilística [18]

hay dos causas competidoras de la secreción nasal. Sin embargo, observar fiebre proporciona una evidencia muy fuerte de que el resfriado es la causa del problema, mientras que creencia de que la alergia sea la causa disminuye sustancialmente (se descarta por la observación de la fiebre). La capacidad de las redes probabilísticas para realizar automáticamente esta inferencia intercausal es una característica clave de su poder de razonamiento.

Entre los distintos modelos basados en redes probabilísticas, dos de los ejemplos más relevantes son las redes bayesianas y las redes de Markov [cita o quitar tanto texto]:

- Las **redes bayesianas** utilizan grafos acíclicos dirigidos (*DAG*) para representar y para manejar la dependencia condicional entre variables. Cada nodo representa una variable aleatoria, y cada arista dirigida que conecta dos nodos indica una relación de dependencia directa. Lo que más distingue a las redes bayesianas es su capacidad para actualizar probabilidades de forma dinámica a medida que llega nueva información (inferencia bayesiana). Esto las hace muy útiles en campos como el diagnóstico médico, donde la inferencia sobre presencia de enfermedades puede mejorar con la inclusión de resultados provenientes de pruebas nuevas o síntomas adicionales observados.
- Las **redes de Markov** son estructuras que permiten modelar la transición de estados en un sistema donde el estado siguiente depende únicamente del estado actual (propiedad de Markov). Estos modelos son muy útiles en secuencias temporales donde se quiere predecir una secuencia de eventos o estados. Se usan en distintos campos como en procesamiento de señales, análisis de secuencias genéticas y PLN (Procesamiento del Lenguaje Natural).

Ambos tipos de redes ofrecen herramientas poderosas en la toma de decisiones y el análisis en un contexto de incertidumbre.

De entre estas dos redes vamos a estudiar las redes bayesianas. Mientras que las redes de Markov sirven para modelar secuencias y transiciones de estado, las redes bayesianas tienen capacidad para modelar directamente la incertidumbre y actualizar las creencias de manera incremental, lo que las convierte en una herramienta valiosa para un problema de clasificación de instancias.

Consideremos $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ un conjunto finito de variables aleatorias discretas, donde cada X_i toma valores dentro de un conjunto finito Ω_{X_i}. Una *red bayesiana* es una representación gráfica de la distribución de probabilidad conjunta. Formalmente, se trata

de un par (G, Θ), donde G es un grafo dirigido acíclico (DAG) y Θ son los parámetros que indican las distribuciones de probabilidad. Los nodos del grafo G se corresponden con las variables aleatorias de $\mathbf{X}$. Entonces podemos afirmar que una red bayesiana viene dada por una componente cualitativa y otra cuantitativa [19].

- La **componente cualitativa** es el grafo dirigido acíclico $G = (\mathbf{X}, E_G)$, donde $\mathbf{X}$ es el conjunto de nodos del grafo (representando las variables del sistema), y E_G es el conjunto de arcos (representando las relaciones de dependencia directas entre dichas variables). Si dos variables están conectadas por un arco diremos que están relacionadas. En cambio, si no lo están, diremos que hay una relación de *independencia*.

  Las relaciones entre los nodos de un arco en una red bayesiana se pueden ver como una relación causa-efecto. La variable del nodo destino es dependiente del origen.

- La **componente cuantitativa** es la colección de parámetros numéricos para cada variable en $\mathbf{X}$. Estos parámetros suelen presentarse en forma de tablas de probabilidad condicional, las cuales expresan nuestras creencias sobre las relaciones entre las variables. Para cada variable X_i en el conjunto $\mathbf{X}$, existe un grupo de distribuciones condicionales, cada una correspondiente a una configuración particular de los padres de X_i en el grafo. Representamos mediante $pa_G(X) = \{Y \in \mathbf{X} | Y \longrightarrow X \in E_G\}$ al conjunto de los padres o predecesores de X. Utilizando esto, podemos calcular la distribución conjunta de todas las variables en $\mathbf{X}$. Esto se hace multiplicando las distribuciones condicionales de cada variable dada la configuración de sus padres:

$$P(x_1, ..., x_N) = \prod_{X_i \in \mathbf{X}} P(x_i | pa_G(x_i))$$

3.1.2.1. Grafos

A continuación presentamos algunas nociones básicas sobre teoría de grafos que pueden sernos de utilidad para comprender mejor las redes bayesianas.

Definición 3.10. Un *grafo* G es un par (V, E), donde V es llamado *conjunto de vértices o nodos* y E *conjunto de aristas*, que representa las relaciones entre dichos vértices.

Para poder explicar cómo son las relaciones entre vértices, necesitamos definir el concepto de *grafo dirigido*.

Definición 3.11. Un *grafo dirigido* es un grafo (V, E) donde cada arista en E es un *arco dirigido*, es decir, tiene un nodo origen y un nodo destino.

En las figuras 3.2 y 3.3 podemos distinguir las diferencias entre un grafo simple y un grafo dirigido.

Para referirnos a estos nodos origen y destino que están relacionados entre sí en cada arco, vamos a hablar de nodos *padre* e *hijo*.

Definición 3.12. Un nodo X es *padre* de un nodo Y si existe un arco dirigido que va de X a Y $(X \to Y)$. Análogamente, se dice que Y es *hijo* de X. El conjunto de todos los padres de Y se denota como $pa(Y)$ y el conjunto de los hijos como $de(Y)$.

Definición 3.13. Un *camino* dentro de un grafo (V, E) es una sucesión de nodos $\{X_1, ..., X_N\}$, de modo que $X_i \neq X_{i+1}$, donde cada nodo está conectado con el anterior y con el siguiente, es decir, se cumple $\overrightarrow{(X_i, X_{i+1})} \in E$ o $\overrightarrow{(X_{i+1}, X_i)} \in E$, $\forall i = 1, ..., N-1$.

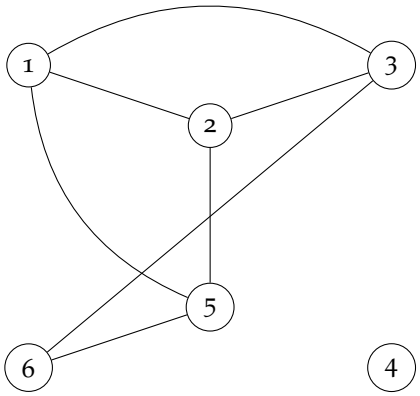


Figura 3.2.: Grafo simple

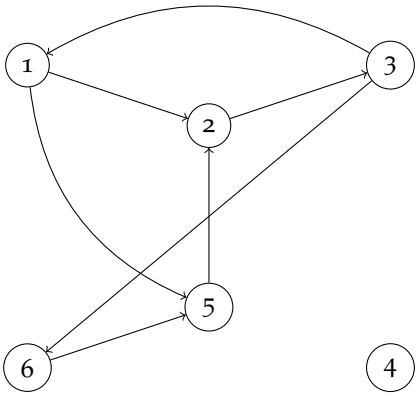


Figura 3.3.: Grafo dirigido

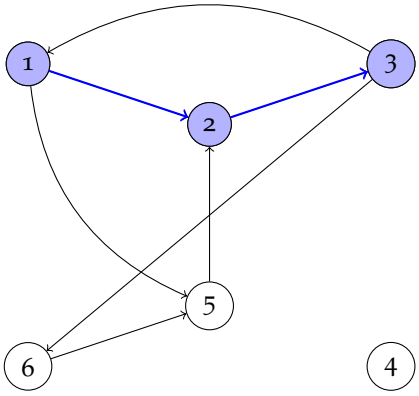


Figura 3.4.: Camino simple abierto

Definición 3.14. Un *camino dirigido* es aquel donde todas las aristas son consistentes en dirección, es decir, todas cumplen $\overrightarrow{(X_i, X_{i+1})} \in E$ o todas cumplen $\overrightarrow{(X_{i+1}, X_i)} \in E$, $\forall i = 1, ..., N-1$.

Definición 3.15. Un camino es *simple* si no pasamos más de una vez por el mismo nodo, es decir, si $X_i \neq X_j$, para $1 < i < j < N$. Además, el camino será *cerrado* si el primer y último nodo coinciden $(X_1 = X_N)$.

En la figura 3.4 podemos ver un camino entre los nodos 1 y 3, señalado en azul. Este camino es simple, pues no pasa más de una vez por el mismo nodo, y abierto, pues el primer y último nodo no coinciden.

A partir de estos conceptos, podemos definir lo que es un *ciclo* dentro de un grafo.

Definición 3.16. Un *ciclo* es un camino cerrado simple, es decir, un camino donde $X_i \neq X_j$, para $1 < i < j < N$ y donde el arco $\overrightarrow{(X_N, X_1)}$ cierra el ciclo. Si el camino además es dirigido, estaremos hablando de un *ciclo dirigido*. Si un grafo no tiene ciclos se trata de un grafo *acíclico*.

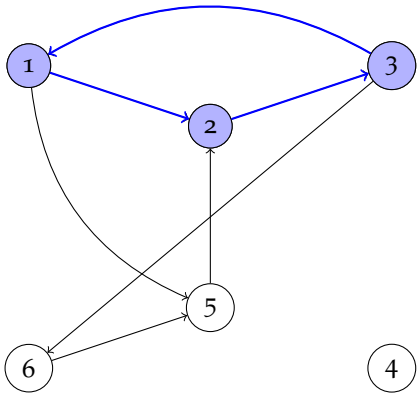


Figura 3.5.: Ciclo dirigido

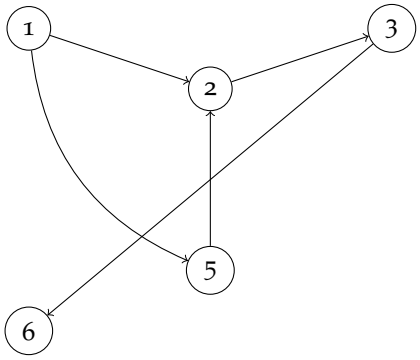


Figura 3.6.: Grafo dirigido acíclico

En la figura 3.5 podemos ver un ciclo dirigido.

Tras estas nociones básicas sobre Teoría de Grafos, estamos en condiciones de definir lo que es un *DAG*.

Definición 3.17. Un *grafo dirigido acíclico* (*DAG*, *Directed Acyclic Graph*), es un grafo dirigido que no contiene ningún ciclo.

El ejemplo anterior de la figura 3.5 representa un grafo que no es acíclico. El hecho de que un *DAG* no tenga ciclos, nos permitirá representar relaciones que no se repiten ni retroceden. En la figura 3.6 podemos ver un grafo dirigido sin ciclos, es decir, un *DAG*. Para ello hemos eliminado los ciclos del ejemplo anterior y hemos omitido el nodo 4 el cual no se relacionaba con ningún otro nodo.

Una ***red bayesiana*** es un modelo gráfico probabilístico que representa un conjunto de variables y sus dependencias condicionales mediante un grafo acíclico dirigido ó *DAG*. Al garantizar que no haya ciclos, este grafo ayuda a prevenir paradojas o dependencias recursivas infinitas en las representaciones de probabilidad.

En una red bayesiana, la distribución de probabilidad conjunta de un conjunto de variables se factoriza según la estructura del DAG. Cada nodo en el grafo representa una variable aleatoria y cada arista representa una dependencia condicional. La ausencia de ciclos asegura que se puede establecer un *orden* entre las variables, es decir, cada variable se puede expresar como condicionalmente dependiente solo de sus predecesores (sus padres) en el grafo. Esto permite que la distribución conjunta de todas las variables se escriba como el producto de las distribuciones condicionales de cada variable dado sus padres. Veremos todo esto de manera formal en el siguiente apartado.

3.1.2.2. Redes bayesianas

Definición 3.18. [20] Una *red bayesiana* es una estructura formada por:

- Un conjunto de variables proposicionales $V = \{X_1, ..., X_N\}$.
- Un conjunto E de relaciones binarias sobre las variables de V.
- Una distribución de probabilidad conjunta P definida sobre las variables de V.

Estos elementos cumplen lo siguiente:

- $\mathcal{G} = (V, E)$ es un grafo dirigido acíclico y conexo.
- $(\mathcal{G}, P)$ cumple las *hipótesis de independencia condicional*.

Decimos que un DAG (V, E) conexo junto con una distribución de probabilidad conjunta P cumple la **hipótesis de independencia condicional** si para toda variable $X_i \in V$ se tiene que el conjunto de los padres de X_i separa condicionalmente a X_i de todo subconjunto $Y \subset V$ que no contenga a X_i ni a sus hijos. En términos formales:

$$\forall X_i \in V \text{ y } \forall Y \subset V \backslash \{X_i \cup de(X_i)\} \text{ se tiene que } P(X_i|pa(X_i), Y) = P(X_i|pa(X_i))$$

Observación 3.3. Las variables en V son proposicionales en el sentido de que pueden tomar un número de estados finito. La variable $X \in V$ puede tomar los valores $\{x_1, x_2, ..., x_N\}$ Esto nos sirve para estudiar situaciones donde los estados están claramente definidos y son mutuamente excluyentes. Por ejemplo, en un diagnóstico médico, una enfermedad puede estar presente o no, y es natural representar esta situación con una variable proposicional.

En esta definición de red bayesiana, hemos partido de una distribución de probabilidad conjunta para las variables, lo cual normalmente es difícil de obtener. El siguiente resultado nos permitirá expresar esta distribución de probabilidad conjunta como producto de las distribuciones condicionadas de cada nodo dados sus padres.

Teorema 3.4 (Factorización de la probabilidad). *[20] Dada una red bayesiana, la distribución de probabilidad conjunta puede expresarse como:*

$$P(x_1, ..., x_N) = \prod_{x_i \in V} P(x_i|pa_G(x_i))$$

Demostración: *Supongamos una ordenación de las variables $\{X_1, ..., X_N\}$ donde los padres de cada nodo aparezcan siempre después de este. Aplicando el Teorema de la Probabilidad Total 3.2:*

$$P(x_1, ..., x_N) = \prod_{x_i \in V} P(x_i|x_{i+1}, ..., x_N)$$

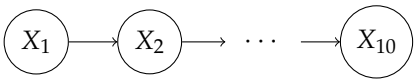
Por la forma en que hemos cogido la ordenadación, el conjunto $\{X_{i+1}, ..., X_N\}$ *incluye a todos los padres de* X_i*. En consecuencia, la hipótesis de independencia condicional nos indica que:*

$$P(x_i|x_{i+1}, ..., x_N) = P(x_i|pa_G(x_i))$$

□

La ausencia de ciclos del *DAG* nos asegura que podemos establecer un *orden* entre las variables, es decir, cada variable puede ser expresada como condicionalmente dependiente solo de sus predecesores (sus padres) en el grafo. Gracias a esto, el teorema anterior nos permite escribir la distribución de probabilidad conjunta como el producto de las distribuciones condicionales de cada nodo, lo cual requiere de menos cálculos.

Ejemplo 3.2. [20] Supongamos que tenemos la siguiente red bayesiana:



Si todas las variables son binarias, necesitaríamos $2^{10} - 1$ parámetros para calcular la distribución de probabilidad conjunta. Sin embargo, si aplicamos el Teorema 3.4, las distribuciones condicionales a calcular son:

- **Distribución de** X_1**:** Como X_1 es la primera variable en la cadena y no tiene padres, $P(X_1|pa_G(X_1)) = P(X_1)$. Como las variables son binarias, supongamos que X_1 toma los valores 1 o 0. Entonces bastaría con calcular $P(X_1 = 1) = p$. Esto suma 1 parámetro.
- **Distribuciones condicionales de** X_2 **a** X_N**:** Cada una de estas variables tiene un padre; entonces la probabilidad de que X_i tome un valor depende de su padre X_{i-1}. Para cada variable necesitamos especificar dos probabilidades: una cuando el padre toma el valor 1 y otra cuando toma el valor 0. Por ejemplo, para X_2 necesitamos calcular $P(X_2 = 1|X_1 = 1)$ y $P(X_2 = 1|X_1 = 0)$. Esto nos añade $2 \cdot 9 = 18$ parámetros.

Obtenemos un total de 19 parámetros necesarios para describir la distribución conjunta. Este ejemplo demuestra la eficiencia de las redes bayesianas para representar distribuciones conjuntas, pues hemos podido reducir muy significativamente el cálculo de probabilidades de $2^{10} - 1 = 1023$ a 19, gracias a las propiedades de independencia condicional.

3.1.3. Clasificadores basados en redes bayesianas

En el ámbito del aprendizaje automático, las redes bayesianas ofrecen un marco sólido para realizar clasificación, debido a su capacidad de modelar dependencias condicionales entre variables. Debemos tener en cuenta que dentro de una red bayesiana, cualquier variable tan solo se encuentra influenciada por su ***Markov Blanket*** (manto de Markov), que es el conjunto de sus variables padre, variables hijas y las variables que son también padre de las hijas. Resulta entonces intuitivo tener en cuenta modelos clasificatorios que tengan solo información del manto de Markov de la variable a clasificar. Por tanto se deben buscar estructuras donde todos los elementos formen parte del manto de Markov de la variable a clasificar [21].

En términos generales, un clasificador bayesiano operan bajo el principio del Teorema de Bayes, para predecir la clase que maximiza la probabilidad posterior, basadandose en las

evidencias observadas.

La *hipótesis de Máximo A Posteriori (hipótesis MAP)* juega un papel clave en el contexto de los clasificadores bayesianos. La hipótesis MAP busca maximizar la probabilidad posterior $P(c_i \mid \mathbf{x})$ de la clase, dado el vector de atributos $\mathbf{x}$ de una instancia.

Definición 3.19. Dada una instancia con atributos $\mathbf{x} = (x_1, \ldots, x_n)$ y sea la clase C con k posibles categorías $\Omega = \{c_1, \ldots, c_k\}$, la hipótesis MAP se define como:

$$C_{\text{MAP}} = \arg\max_{c_i \in \Omega} P(c_i \mid x_1, \ldots, x_n)$$

Por tanto se trata de la categoría que maximiza la probabilidad posterior.

Observación 3.4. Aunque puedan parecer similares, la probabilidad condicional y la probabilidad posterior son conceptos distintos. La **probabilidad condicional** $P(A \mid B)$ describe la probabilidad de un evento A sabiendo que otro evento B ha ocurrido. Por otro lado, la **probabilidad posterior** se calcula después de observar nueva evidencia E y se utiliza para ajustar la probabilidad de una hipótesis H. Se calcula usando el Teorema de Bayes:

$$P(H \mid E) = \frac{P(E \mid H) \cdot P(H)}{P(E)}$$

Podemos ilustrarlo mejor mediante un ejemplo. Consideremos un caso médico en el que un médico evalúa la probabilidad de que un paciente tenga una enfermedad basada en la presencia de un síntoma. La probabilidad condicional sería $P(\text{Enfermedad} \mid \text{Síntoma})$. Suponiendo que se realiza una prueba médica adicional, podemos actualizar nuestra creencia sobre la enfermedad mediante la probabilidad posterior, usando el Teorema de Bayes:

$$\begin{aligned} &P(\text{Enfermedad} \mid \text{Síntoma, Resultado de prueba}) = \\ &\quad = \frac{P(\text{Resultado de prueba} \mid \text{Enfermedad}) \times P(\text{Enfermedad} \mid \text{Síntoma})}{P(\text{Resultado de prueba})} \end{aligned}$$

Esta expresión modifica la probabilidad inicial de que el paciente tenga la enfermedad, basándose en la nueva evidencia proporcionada por el resultado de la prueba.

Aplicando el Teorema de Bayes, la expresión de la hipótesis MAP queda así:

$$\begin{aligned} C_{\text{MAP}} &= \arg\max_{c_i \in \Omega} P(c_i \mid x_1, \ldots, x_n) \\ &= \arg\max_{c_i \in \Omega} \frac{P(x_1, \ldots, x_n \mid c_i) P(c_i)}{P(x_1, \ldots, x_n)} \\ &= \arg\max_{c_i \in \Omega} P(x_1, \ldots, x_n \mid c_i) P(c_i) \end{aligned}$$

donde $P(x_1, \ldots, x_n)$ es constante dada una instancia.

El problema principal de este enfoque es que hay que trabajar con la distribución conjunta y eso normalmente es inmanejable computacionalmente, sobretodo en espacios de alta

dimensión y grandes conjuntos de datos.

3.1.4. Naive Bayes y tipos

El clasificador Naive Bayes es una simplificación del clasificador bayesiano general. Mientras que un clasificador bayesiano completo considera todas las posibles interacciones y dependencias entre características, Naive Bayes asume que todas las características son independientes entre sí dado el resultado de la clase. Esto simplifica notablemente los cálculos y la implementación a costa de ignorar posibles dependencias entre características. Esta simplificación hace que Naive Bayes sea menos preciso en teoría, aunque se demuestra que estos clasificadores son efectivos en la práctica, ofreciendo soluciones competentes incluso en comparación con métodos más complejos.

A continuación vamos a estudiar los distintos clasificadores pertenecientes a la familia Naive Bayes [22] [19] [23].

3.1.4.1. Naive Bayes simple o binomial

Este clasificador considera de forma binaria la probabilidad de aparación de cada término dada la clase de documento, es decir, el término aparece o no en la clase [22].

Este clasificador se basa en dos supuestos [19]:

- Cada atributo $\{x_1, x_2, ..., x_n\}$ es condicionalmente independiente de los otros atributos, dada la clase C.
- Todos los atributos tienen influencia sobre la clase C.

El Teorema de Bayes 3.3 establece la siguiente relación, dada la variable de clase c_i y el vector de características $\{x_1, ..., x_n\}$:

$$P(c_i \mid x_1, \ldots, x_n) = \frac{P(c_i)P(x_1, \ldots, x_n \mid c_i)}{P(x_1, \ldots, x_n)}$$

Utilizando la suposición de independencia condicional, la cual indica que

$$P(x_i \mid c_i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i \mid c_i),$$

para todo $i = 1, ..., n$, entonces

$$P(x_1, \ldots, x_n \mid c_i) = \prod_{i=1}^{n} P(x_i \mid c_i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid c_i),$$

La relación anterior se simplifica a

$$P(c_i \mid x_1, \ldots, x_n) = \frac{P(c_i) \prod_{i=1}^{n} P(x_i \mid c_i)}{P(x_1, \ldots, x_n)}$$

Sabemos que el valor de la probabilidad conjunta de todas las características $P(x_1, \ldots, x_n)$ es constante, pues no cambia con respecto a diferentes clases. Por tanto podemos omitir este término en la fórmula:

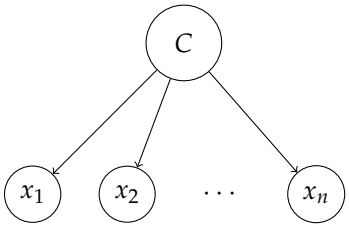


Figura 3.7.: Estructura del clasificador Naive Bayes

$$P(c_i \mid x_1, \ldots, x_n) \propto P(c_i) \prod_{i=1}^{n} P(x_i \mid c_i)$$

Para determinar la clase más probable $\hat{c}$, elegimos la clase que maximiza este producto, y la hipótesis MAP quedaría como sigue:

$$\begin{aligned} \hat{c} = C_{\text{MAP}} &= \arg\max_{c_i \in \Omega} P(c_i \mid x_1, \ldots, x_n) \\ &= \arg\max_{c_i \in \Omega} P(c_i) \prod_{i=1}^{n} P(x_i \mid c_i) \end{aligned}$$

y entonces solo tenemos que estimar $P(c_i)$ y $P(x_i \mid c_i)$. $P(c_i)$ es la probabilidad a priori de la clase (frecuencia relativa de la clase c_i en el conjunto de entrenamiento). $P(x_i \mid c_i)$ es cada una de las probabilidades condicionales de los atributos x_i dada la clase c_i. De este modo, la tarea que realiza el clasificador es buscar la clase que maximiza la función C_{MAP}.

Los distintos clasificadores de Naive Bayes se diferencian principalmente por las suposiciones que se hacen con respecto a la distribución probabilística de $P(x_i \mid y)$. Dicha distribución puede variar dependiendo del tipo de datos y de las características. Por ejemplo, dichas distribuciones pueden ser Gaussianas, multinomiales o de Bernoulli, cada una adecuada para distintos tipos de variables de entrada (continuas, multivariantes o binarias, respectivamente).

A pesar de la simplicidad de Naive Bayes y de las restricción de que los atributos sean independientes entre sí, este clasificador consigue un alto grado de acierto, especialmente en dominios relacionados con la medicina. A la hora de realizar un diagnóstico, los médicos recogen los atributos igual que en Naive Bayes, es decir, independientes de cada clase [19].

En la figura 3.7 podemos observar dicha estructura, donde tenemos un nodo C para la clase y un nodo x_i para cada atributo. Por ejemplo, el nodo C indica si un mensaje es *spam* o *ham*, y cada x_i es un indicador de si el mensaje contiene una palabra clave (por ejemplo, *millonario*). El hecho de que el mensaje sea *spam* o no, altera la probabilidad de ocurrencia de dichas palabras, es decir, x_i depende de la clase y por tanto existe $P(x_i \mid y)$. Observamos que no hay arcos entre los x_i, pues se supone la independencia entre variables.

3.1.4.2. Naive Bayes multinomial

Mientras que el clasificador Naive Bayes simple toma la aparición o no aparición de cada término en el conjunto de datos, este clasificador considera el número de apariciones de

cada término para evaluar la contribución de su probabilidad condicional dada la clase del documento [22]. Este clasificador suele mejorar el desempeño del algoritmo Naive Bayes, pues se está proporcionando información adicional al clasificador para que la asignación de la clase mejore.

Este clasificador [24] aplica Naive Bayes a datos distribuidos de forma multinomial. Esta implementación suele trabajar con vectores de conteo de palabras, y es una de las variantes de Naive Bayes más utilizadas en problemas de clasificación de textos.

La distribución del modelo es parametrizada por vectores de la forma $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ para cada clase y, donde n es el número de características y θ_{yi} es la probabilidad de que la característica i aparezca en una muestra o instancia perteneciente a la clase y, es decir, $P(x_i \mid y)$. Si tenemos dos clases, obtendremos dos vectores θ_y, donde cada uno refleja las probabilidades de que las distintas n características sean observadas en esa clase particular.

Los parámetros θ_{yi} son estimados mediante una versión suavizada de la estimación por máxima verosimilitud (EMV), conocida como conteo de frecuencia relativa:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

En este contexto:

- $N_{yi} = \sum_{x \in T} x_i$ representa el número de veces que la característica i aparece en las muestras pertenecientes a la clase y, en el conjunto de entrenamiento T.
- $N_y = \sum_{i=1}^{n} N_{yi}$ es el conteo total de todas las características para la clase y.
- α es el parámetro de suavizado, normalmente mayor o igual a cero, el cual ayuda a manejar características que no están presentes en el entrenamiento y evita asignar probabilidades nulas. Un valor de $\alpha = 1$ es conocido como *suavizado de Laplace*, mientras que un $\alpha < 1$ como *suavizado de Lidstone*.

Si tomamos los vectores θ_y para cada clase y, obtenemos una matriz de dimensión $m \times n$, donde m es el número de clases y n el número total de características. Entonces cada fila de la matriz corresponde a una clase, y cada columna a una característica específica, donde el elemento en la fila y y columna i representa la probabilidad θ_{yi} de que la característica i aparezca en una muestra de la clase y.

En consecuencia, la matriz de parámetros Θ para un clasificador Naive Bayes Multinomial con m clases y n características se define como:

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \cdots & \theta_{1n} \\ \theta_{21} & \theta_{22} & \cdots & \theta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{m1} & \theta_{m2} & \cdots & \theta_{mn} \end{bmatrix}$$

Ejemplo 3.3. Vamos a ver un ejemplo de aplicación del clasificador Naive Bayes Multinomial en el contexto de la filtración de correos electrónicos para identificar *spam*. En este caso, cada correo electrónico (cada instancia) se representaría como un vector de conteos de palabras.

El modelo calcularía la probabilidad de que un correo pertenezca a la categoría *spam* o *ham*, basándose en la frecuencia de ciertas palabras que tienden a aparecer más en una u otra clase.

Suponemos que contamos las palabras *oferta*, *gratis* y *click* en nuestro conjunto de entrenamiento. Consideramos los siguientes conteos:

Palabra	**Spam**	**Ham**
oferta	40	5
gratis	25	3
click	30	2
Total	150	100

Utilizando un suavizado de Laplace con $\alpha = 1$, la matriz de probabilidades Θ se calcularía como sigue:

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} = \begin{bmatrix} \theta_{\text{spam, oferta}} & \theta_{\text{spam, gratis}} & \theta_{\text{spam, click}} \\ \theta_{\text{no spam, oferta}} & \theta_{\text{no spam, gratis}} & \theta_{\text{no spam, click}} \end{bmatrix}$$

$$\Theta = \begin{bmatrix} \frac{40+1}{150+3} & \frac{25+1}{150+3} & \frac{30+1}{150+3} \\ \frac{5+1}{100+3} & \frac{3+1}{100+3} & \frac{2+1}{100+3} \end{bmatrix} = \begin{bmatrix} 0.273 & 0.173 & 0.207 \\ 0.053 & 0.035 & 0.026 \end{bmatrix}$$

donde cada fila de Θ representa una clase (primera fila para *spam*, segunda fila para *ham*), y cada columna representa una palabra (*oferta*, *gratis* y *click*).

3.1.4.3. Naive Bayes gaussiano

El clasificador Naive Bayes gaussiano es una extensión de Naive Bayes para datos que siguen una distribución normal. Se utiliza cuando las características de los datos son continuas y se asume que cada característica sigue una distribución gaussiana (normal).

El modelo calcula la probabilidad de que un dato pertenezca a cierta clase, basándose en la función de densidad de probabilidad gaussiana. Se considera la media μ_y y la desviación típica σ_y de cada característica para cada clase y, para realizar la clasificación:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Los parámetros σ_y y μ_y se estiman utilizando el método de máxima verosimilitud.

Este clasificador es útil cuando las características pueden expresarse como medidas que varían de manera continua y se distribuyen normalmente, como en el caso de mediciones físicas o ciertas características financieras.

3.1.4.4. Naive Bayes de Bernoulli

El modelo Naive Bayes de Bernoulli está pensado para datos que se distribuyen según distribuciones de Bernoulli multivariadas. Esto implica que puede haber múltiples características,

pero cada una es binaria. Por tanto las muestras se representan como vectores de características binarias.

La regla de decisión para este clasificador se basa en la siguiente probabilidad:

$$P(x_i \mid y) = P(x_i = 1 \mid y)x_i + (1 - P(x_i = 1 \mid y))(1 - x_i)$$

Esta fórmula penaliza la no ocurrencia de una característica i que es indicativa de la clase y, a diferencia de Naives Bayes Multinomial, que ignora las características que no aparecen.

En el contexto de la clasificación de textos, para entrenar y usar este clasificador se usan vectores de ocurrencia de palabras (en lugar de conteo de palabras), que indican si una palabra se encuentra (1) o no (0) en el texto. Este clasificador resulta adecuado en conjuntos de datos con documentos más cortos, donde el conteo de frecuencias de los modelos multinomiales pueden no ser efectivos debido a la baja frecuencia de palabras, mientras que la presencia o ausencia de términos suele ser más informativa.

3.1.4.5. Clasificadores que admiten dependencias

Alternativamente, si consideramos que nuestro modelo no debería omitir las dependencias entre características, una solución sería considerar modelos TAN (*Tree Augmented Naive Bayes*) o BAN (*Bayesian Augmented Naive-Bayes*). La idea de estos modelos es relajar la restricción de que todos los atributos son condicionalmente independientes entre sí.

- **TAN** (*Tree Augmented Naive Bayes*): Se trata de una red bayesiana donde los atributos o variables pueden otro padre además de C, es decir, se pueden encontrar relaciones entre atributos. Por tanto la estructura de la red ya no es trivial, sino que se tienen que obtener las relaciones entre atributos. La red pasa a tener estructura de árbol. Estas estructuras obtienen mejores resultados que Naive Bayes, al mismo tiempo que no aumentan demasiado la complejidad computacional, y conservan la robustez del predecesor. Un ejemplo de la estructura de una red TAN es la mostrada en la figura 3.8.

- **BAN** (*Bayesian Augmented Naive-Bayes*): Partiendo de la estructura del TAN, ahora relajamos completamente la restricción de que los atributos sean independientes entre sí. Cada atributo podrá tener varios padres además de la clase C. De esta forma, la estructura ya no tiene por qué ser un árbol. En la figura 3.9 podemos ver un ejemplo de red BAN.

- **Semi Naive Bayes**: Este modelo consiste en tomar los atributos correlacionados entre sí y reunirlos en un atributo compuesto. Así se consigue un clasificador Naive Bayes usual, pues los nuevos atributos compuestos se consideran independientes entre ellos dada la clase.

  En la figura 3.10 podemos ver un ejemplo. Supongamos que las variables x_1, x_2 y x_3 predicen la clase C. Supongamos que las variables x_1 y x_3 son condicionalmente dependientes dada C. Entonces podríamos agruparlas en una variable compuesta, donde las variables $\{x_1, x_3\}$ y x_2 son independientes.

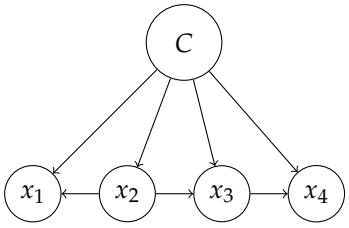


Figura 3.8.: Estructura de una red TAN

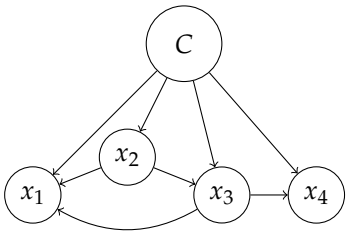


Figura 3.9.: Estructura de una red BAN

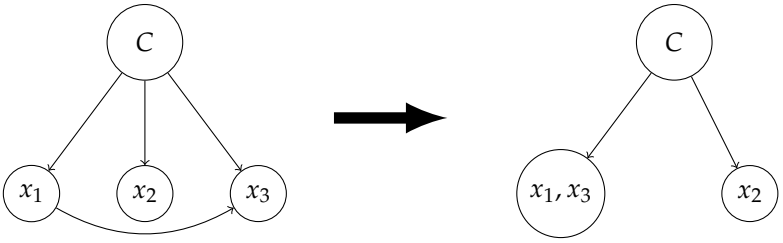


Figura 3.10.: Ejemplo de clasificador Semi Naive Bayes

3.1.4.6. Dependencias y su impacto en Naive Bayes

La suposición fundamental de Naive Bayes es que los atributos son condicionalmente independientes dados los valores de clase. Matemáticamente, esto se expresa en el Teorema 3.3.

A pesar de esta suposición, la realidad es que los atributos a menudo están correlacionados. Sin embargo, el impacto de estas dependencias puede ser mitigado o incluso cancelado bajo ciertas condiciones. Según [25], esto se debe a que las interacciones entre dependencias pueden trabajar en direcciones opuestas, neutralizando el efecto neto sobre la probabilidad condicional de clase.

Consideremos dos atributos x_1 y x_2 que son dependientes entre sí. Aunque esta dependencia viola la suposición de independencia de Naive Bayes, su efecto en la decisión final de clasificación puede ser insignificante si estas dependencias se distribuyen de manera uniforme a través de las clases o si se cancelan mutuamente.

A partir de ahora vamos a tratar con una red bayesiana aumentada o BAN, donde además de compartir todos los nodos un mismo nodo padre, pueden también estar relacionados entre sí de forma libre. Consideraremos dos clases, denotadas como {+, -}.

La dependencia local de un nodo se refiere a cómo los nodos padres de un nodo específico influyen en este último dentro de cada clase. Para cuantificar esta influencia, se utiliza la razón entre la probabilidad condicional del nodo dado sus padres y la probabilidad condicional del nodo sin considerar a sus padres. Matemáticamente, se define de la siguiente manera:

Definición 3.20. Dado un nodo X en una BAN G, la *derivada de dependencia local* de X en las clases {+, -} se define como sigue:

$$dd_G^+(x|pa(x)) = \frac{p(x|pa(x),+)}{p(x|+)}$$

$$dd_G^-(x|pa(x)) = \frac{p(x|pa(x),-)}{p(x|-)}$$

La medida $dd_G^+(x|pa(x))$ refleja la fuerza de la dependencia local del nodo X para la clase positiva (+), indicando cuánto la presencia de los padres afecta la probabilidad de X en dicha clase. Análogamente, $dd_G^-(x|pa(x))$ actúa de la misma forma para la clase negativa (-).

Observación 3.5. De la anterior definición se extraen los siguientes resultados:

- Cuando X no tiene padres, entonces:

$$dd_G^+(x|pa(x)) = dd_G^-(x|pa(x)) = 1.$$

- Cuando $dd_G^+(x|pa(x)) \geq 1$, la dependencia local de X apoya la clasificación de la clase positiva (pues el numerador es mayor que el denominador). De lo contrario, apoya la clasificación de la clase negativa. Oucrre de forma análoga cuando $dd_G^-(x|pa(x)) \geq 1$.

Definición 3.21. Para un nodo X en una BAN G, la *razón de la derivada de dependencia local* en

el nodo X se define como sigue:

$$ddr_G(x) = \frac{dd_G^+(x|pa(x))}{dd_G^-(x|pa(x))}$$

Observación 3.6. De acuerdo con la definición anterior, $ddr_G(x)$ cuantifica la influencia de la dependencia local de X en la clasificación. Además, extraemos los siguientes resultados:

1. Si X no tiene padres, entonces $ddr_G(x) = 1$.
2. Si $dd_G^+(x|pa(x)) = dd_G^-(x|pa(x))$, entonces $ddr_G(x) = 1$. Esto significa que la dependencia local de x se distribuye uniformemente en la clase + y en la clase -. Por lo tanto, la dependencia no afecta la clasificación, sin importar cuán fuerte sea.
3. Si $ddr_G(x) > 1$, la dependencia local de X en la clase + es más fuerte que en la clase -. Si $ddr_G(x) < 1$, significa lo contrario.

El resultado más relevante que extraemos es que cuando el valor de $ddr_G(x)$ es cercano a 1, entonces la dependencia entre x_1 y x_2 no afecta significativamente a la clasificación. Por ejemplo, si en una red bayesiana x_1 es padre de x_2, entonces $ddr_G(x) = 1$ indica que la dependencia de x_1 con sus padres es similar para ambas clases {+,-}, sugiriendo que esta dependencia no juega un papel crucial a la hora de clasificar en la clase positiva o la negativa.

Definición 3.22. Dada una red bayesiana aumentada (BAN), definimos el *factor de distribución de dependencia global* para un ejemplo E como:

$$DFG(E) = \prod_{i=1}^{n} ddrG(x_i)$$

donde E es un ejemplo con atributos $x_1, x_2, \ldots, x_n$.

Este factor mide cómo las dependencias locales entre atributos afectan la clasificación del ejemplo E. Si $DFG(E) \approx 1$, entonces las dependencias se consideran balanceadas o que se cancelan entre sí, permitiendo que el clasificador mantenga un buen rendimiento.

Cuando $DFG(E) = 1$, las dependencias modeladas en la red bayesiana G no tienen ninguna influencia en la clasificación. En otras palabras, la clasificación de G es exactamente igual a la de su correspondiente clasificador Naive Bayes. Existen tres casos en los que $DFG(E)$ puede ser igual a uno:

1. **No existe dependencia entre atributos:** Esto implica que cada atributo es independiente de los demás, lo que valida la suposición de independencia de Naive Bayes.
2. **Distribución local equitativa entre clases:** Para cada atributo X en G, $ddr_G(x) = 1$. Esto significa que la distribución de la dependencia local de cada nodo es uniforme entre las clases, neutralizando cualquier influencia en la clasificación debido a dependencias.
3. **Cancelación de influencias en la clasificación:** La influencia de algunas dependencias locales que apoyan la clasificación de E en la clase positiva es contrarrestada por la influencia de otras dependencias locales que apoyan la clasificación de E en la clase negativa. Esto da lugar a un balance que neutraliza el efecto neto de las dependencias en la clasificación final.

Hemos probado que bajo ciertas condiciones, un modelo BAN puede comportarse de manera equivalente que un modelo de Naive Bayes simple, a pesar de las dependencias entre atributos.

Esto demuestra que a pesar de que Naive Bayes opera teóricamente bajo suposiciones de independencia, su robustez se puede mantener aunque estas suposiciones no se cumplan. Las dependencias entre atributos pueden no afectar de forma negativa al rendimiento de Naive Bayes siempre y cuando estas dependencias se cancelen entre sí, o dicho de otra forma, mientras se distribuyan uniformemente a través de las clases. Esta propiedad convierte a Naive Bayes en un clasificador resistente a la presencia de dependencias entre atributos.

3.2. SVM

not yet :(

4. Experimentación

Tras haber estudiado las bases matemáticas de diversos algoritmos de aprendizaje supervisado, vamos a aplicar dichas técnicas a un caso real.

4.1. Dataset

Para la fase de experimentación, se ha elegido como corpus una base de datos de Kaggle [26]. En esta se combinan mensajes de correo electrónico de tres fuentes distintas:

- **`LingSpam.csv`**: este dataset es uno de los más antiguos conocidos a nuestra disposición y está constituido por 2591 mensajes de *spam* y *ham* recuperados de Linguist List [27]. Estos mensajes se centran en ofertas de trabajo, oportunidades de investigación y discusiones sobre software.
- **`EnronSpam.csv`**: el dataset está formado por 9687 correos de varios trabajadores de la empresa Enron Corporation. El dataset original contiene aproximadamente 500000 emails y se puede encontrar en [28]
- **`SpamAssassin.csv`**: este dataset recoge 6045 mensajes donados por varios usuarios de un foro público. Al tratarse de usuarios variados, estos mensajes son de temas menos específicos que aquellos mensajes que podrían provenir de un mismo usuario. El dataset original se puede encontrar en [29]

Los datasets que vamos a utilizar son simplificaciones de los originales, proporcionados por el dataset de Kaggle. Esta decisión de coger mensajes de distintas fuentes se ha tomado para fomentar la diversidad de contenido y que el modelo esté entrenado sobre una mayor variedad de escenarios. Por ejemplo, si sólo empleáramos la base de datos de EnronSpam, estaríamos limitando el espacio de detección del modelo al del ámbito empresarial.

Los tres archivos .csv constan de los siguientes atributos:

- **`id:`** identificador de cada instancia o mensaje.
- **`Body:`** cuerpo o contenido del mensaje.
- **`Label:`** clase a la que pertenece la instancia (toma el valor 1 si es Spam y 0 si es Ham).

4.2. Definición del problema

Nos encontramos pues ante un problema de clasificación binaria, al cual vamos a aplicar varios algoritmos de aprendizaje supervisado para discernir qué técnica es capaz de identificar mejor las relaciones entre los contenidos de los mensajes.
Vamos a dividir el Dataset en un conjunto de entrenamiento para entrenar el modelo y un conjunto de prueba para constatar la eficacia del modelo. El objetivo de esta fase va a ser

entrenar varios modelos que puedan discernir con gran exactitud entre correo Spam y Ham, tanto para el conjunto de prueba como para un nuevo conjunto creado por nosotros y ajeno a las tres fuentes expuestas en 4.1.
La división del Dataset la vamos a hacer mediante la función `train_test_split` de Skicit-learn (ver Apéndice A.2). El conjunto de entrenamiento contendrá el 80 % de los datos y el conjunto de test el 20 % de los datos.

4.3. Clasificación de textos

La tarea de clasificación de textos consiste en asignar una etiqueta a cada documento perteneciente a una colección, la cual indica a qué clase pertenece dicho documento. La decisión sobre qué etiqueta debe asignarse se toma a partir de un modelo construido mediante un conjunto de entrenamiento, que no es más que una parte de la colección donde los documentos tienen ya asignados una etiqueta. El objetivo es la construicción de un modelo que prediga correctamiente la clase de nuevos documentos, llamado conjunto de prueba, los cuales no deben intervenir en la construcción del modelo para evitar cometer un sesgo.

Cuando deseamos mejorar la calidad o eficacia de la clasificación para un dominio concreto, podemos aplicar dos enfoques [30]:

- **Modificar el modelo predictivo:** ya sea cambiándo el clasificador elegido por otro, o alterando los hiperparámetros, estudiando qué combinación provee de mejores resultados.
- **Modificar la representación de los datos:** la eficacia de los clasificadores puede variar significativamente dependiendo de cómo se representen los datos. Elegir la representación adecuada para los datos es un desafío en sí mismo. Las formas más comunes de representar un documento incluyen el modelo vectorial, donde cada documento se representa como un vector cuyas dimensiones corresponden al tamaño del vocabulario y cuyos atributos reflejan la frecuencia de cada término en el documento. También se utilizan representaciones binarias, donde se indica simplemente la presencia o ausencia de términos, y otros esquemas de ponderación.

  —Decir tb que es importante el preprocesado (transformación de los datos, distinto a la representación de estos...)

4.4. Visualización y análisis exploratorio

4.4.1. Limpieza y representación de los datos

Dentro de este preprocesamiento inicial, previo a la visualización de los datos, vamos a aplicar las siguientes técnicas para limpiar y normalizar los datos textuales en nuestro dataset:

1. **Eliminar valores duplicados**: Para asegurar que nuestro análisis sea sobre datos únicos, utilizamos la función `drop_duplicates()` (Apéndice A.1). Esta función detecta y elimina cualquier instancia repetida en el dataset, necesario para mantener la calidad de los datos y evitar sesgos debidos a repetición de información.
2. **Eliminar valores nulos**: Empleamos una expresión regular que reemplaza cadenas vacías que consisten únicamente en espacios en blanco por `NaN`. Esta operación es seguida

por la eliminación de todas las filas que contengan valores `NaN` en la columna `Body` mediante `dropna()` (Apéndice A.1). Esta limpieza es fundamental para no distorsionar los análisis con textos que no contienen información útil.

3. **Eliminar mayúsculas**: Convertimos todo el texto a minúscula para uniformizar el formato y facilitar el procesamiento de texto. Esto es importante porque en el análisis textual, las palabras en mayúsculas y minúsculas se consideran distintas por defecto. Esto reduce la dispersión de variantes que pueda tener una misma palabra, simplificando el análisis.

4. **Eliminar signos de puntuación**: Eliminamos todos los signos de puntuación mediante una expresión regular que los identifica y reemplaza por un espacio vacío. Esto ayuda a reducir el ruido en el texto y permite que los modelos se enfoquen en las palabras y su significado sin distracciones causadas por caracteres especiales.

5. **Eliminar palabras vacías**: Las palabras vacías o *stop words* (pronombres, preposiciones, conjunciones, etc.) se deben eliminar del texto antes de la tokenización, para concentrar el análisis en palabras que aportan más significado al contenido. Para ello, scikit-learn proporciona una lista predeterminada de palabras vacías en inglés, denominada ENGLISH_STOP_WORDS. Esta lista se puede actualizar con palabras adicionales según sea necesario. Eliminar estas palabras vacías también reduce la dimensionalidad del dataset.

6. **Tokenización**: utilizamos la función `word_tokenize` de la librería de PLN (Apéndice A.3), para convertir nuestro dataset en una lista de listas de términos, donde cada instancia es representada por una lista de términos. Así conseguimos una estructura de datos en forma de términos separados, gracias al cual podremos hacer un análisis sobre la distribución del dataset, como estudiar las palabras más frecuentes.

Estas técnicas de preprocesamiento preparan el dataset de manera efectiva, optimizando su estructura y contenido para un análisis más robusto. De esta forma podemos pasar de una estructura repleta de símbolos, caracteres especiales, enlaces, etc., a otra después de limpiar los datos, como mostramos en la tabla 4.1 para tres ejemplos extraidos de nuestro dataset.

Tras conseguir un texto limpio, el siguiente paso es tokenizar los datos, tal y como vemos en la tabla 4.2 para poder aplicar algunas técnicas de visualizado de forma más práctica.

4.4.2. Análisis estadístico descriptivo

4.4.3. Distribución de las clases

4.4.3.1. Desbalanceo de clases

4.4.3.2. Distribucion de la longitud

4.4.3.3. Distribución de palabras

4.5. Preprocesado

Antes de entrenar el clasificador, es crucial realizar el preprocesado de los datos para garantizar la máxima eficacia del algoritmo de aprendizaje automático seleccionado. Hemos dividido el preprocesado en dos etapas principales: *data cleaning* y *data preprocessing*.

Label	Texto original	Texto limpio
Spam	1) Fight The Risk of Cancer! http://www.adclick.ws/p.cfm?o=315&s=pk007 2) Slim Down - Guaranteed to lose 10-12 lbs in 30 days http://www.adclick.ws/p.cfm?o=249&s=pk007	1 fight risk cancer URL slim guaranteed lose 10 12 lbs 30 days URL
Spam	Subject: make $3500 per week using your home computer! put my free software in your computer... start making huge amounts of cash... without working!!!	make 3500 week using home computer free software computer start making huge amounts cash working
Ham	Subject: intern compensation sally : summer analysts will benefit from our recent compensation changes. their monthly salary this summer will be $3333.00 please let me know if you have any questions	intern compensation sally summer analysts benefit recent compensation changes monthly salary summer 3333 00 let know questions

Tabla 4.1.: Comparación de correos antes y después del *Data cleaning*

Label	Texto limpio	Texto tokenizado
Spam	1 fight risk cancer URL slim guaranteed lose 10 12 lbs 30 days URL	['1', 'fight', 'risk', 'cancer', 'URL', 'slim', 'guaranteed', 'lose', '10', '12', 'lbs', '30', 'days', 'URL']
Spam	make 3500 week using home computer free software computer start making huge amounts cash working	['make', '3500', 'week', 'using', 'home', 'computer', 'put', 'free', 'software', 'computer', 'start', 'making', 'huge', 'amounts', 'cash', 'working']
Ham	intern compensation sally summer analysts benefit recent compensation changes monthly salary summer 3333 00 let know questions	['intern', 'compensation', 'sally', 'summer', 'analysts', 'benefit', 'recent', 'compensation', 'changes', 'monthly', 'salary', 'summer', '3333', '00', 'let', 'know', 'questions']

Tabla 4.2.: Comparación de textos limpios y tokenizados

Primero, el preprocesado inicial (*data cleaning*), realizado en 4.4.1 que ha sido necesario para poder realizar una visualización en condiciones de los datos. Este paso se centró limpiar los datos de cualquier inconsistencia o imperfección que pudiera distorsionar el análisis visual y estadístico.

Tras aplicar diversas técnicas de visualización a nuestros datos, nos hemos percatado de varias tendencias, patrones y características presentes en nuestro dataset. Esta comprensión nos permite realizar un segundo nivel de preprocesado (*data preprocessing*), donde vamos a sacarle el máximo provecho al conocimiento obtenido. En esta sección, ajustamos y transformamos los datos de manera que se alineen mejor con los requisitos y la naturaleza del modelo de aprendizaje automático que vamos a emplear.

1. **Selección de características. Filtrado de palabras poco frecuentes:** eliminamos aquellas palabras que aparecen solo una vez en todo el corpus. Para ello vamos a usar el parámetro `min_df` en la configuración de `CountVectorizer()`. Este parámetro nos permite

establecer un umbral mínimo del número de documentos en los que tiene que aparecer una palabra para poder formar parte del vocabulario. Al establecer `min_df=2`, estamos eliminando todas las palabras que aparezcan sólo una vez en todo el corpus. Con ello conseguimos reducir notablemente la matriz dispersa y quitar palabras irrelevantes, a la vez que mejoramos la eficiencia.

2. **Balanceo de clases**: Las distintas técnicas que hemos considerado son:
   - **Sobremuestreo de la clase minoritaria:** Consiste en aumentar el número de instancias en la clase minoritaria replicándolas hasta alcanzar un balance más equitativo.
   - **Submuestreo de la clase mayoritaria:** Esta técnica reduce el número de instancias en la clase mayoritaria para igualar la cantidad de la clase minoritaria.
   - **Generación sintética de muestras (SMOTE):** SMOTE crea instancias sintéticas de la clase minoritaria en lugar de crear copias exactas, ayudando a proporcionar más variabilidad. Sin embargo, también introduce el riesgo de generar datos que no representan adecuadamente la realidad.

   Finalmente hemos optado por aplicar sobremuestreo, pues nos permite mantener toda la información de la clase mayoritaria mientras aumentamos la representación de la clase minoritaria, evitando la pérdida de datos potencialmente valiosos que ocurre en el submuestreo.

   Para ello hacemos uso de `RandomOverSampler` de la biblioteca `imbalanced-learn` **??**. `RandomOverSampler` va replicando aleatoriamente las instancias de la clase minoritaria hasta alcanzar un equilibrio deseado.

3. **Creación de nuevas características**: como conteo de errores, pues es común que un mensaje con muchos errores gramaticales sea spam (demostrar esto en el visualizado, por ejemplo contando los errores gramaticales en spam/ham y representar estos conteos en un histograma)

4. **Vectorización:** utilizamos la función `CountVectorizer()` de `sklearn` (Apéndice A.2) para convertir el cuerpo de cada correo electrónico en una representación numérica, donde cada palabra se convierte en una característica y se cuenta su frecuencia en cada correo electrónico.

   Realizamos este proceso porque el clasificador de Naive Bayes multinomial está pensado para aplicarlo en clasificación con características discretas, como recuentos de palabras [24]. Los datos de entrenamiento y de prueba se deben pasar al clasificador bien como una matriz densa (array-like) o una matriz dispersa (sparse matrix).

   La función de tokenización `CountVectorizer()` de `sklearn` precisamente devuelve una matriz dispersa, que ocupa menos espacio en memoria que una matriz densa o array-like. Hemos elegido esta función porque el conjunto de datos, al ser texto, es muy disperso, lo que significa que la mayoría de entradas en la matriz serán ceros.

   Nota: Incluir también TFIDFVectorizer si al final lo uso.

4.6. Algoritmos elegidos (entrenamiento)

-Naive Bayes Multinomial
-SVM

4.6.1. Selección de hiperparámetros

4.7. Validación y evaluación

4.7.1. Evaluación y validación de clasificadores

A la hora de construir clasificadores, es crucial cuantificar y determinar su eficacia. Por ejemplo, en aplicaciones críticas como la detección de enfermedades, no podemos permitir un alto índice de fallos. En la evaluación de un clasificador se deben tener presentes varios criterios: costes computacionales de tiempo, simplicidad, interpretabilidad del modelo obtenido y su precisión (*accuracy*). Esta última es la medida que más atención suele recibir, pues refleja la proporción de predicciones correctas, o el número de aciertos sobre el total de casos (Ver apartado 4.7.2).

Además de usarse para evaluar clasificadores, la precisión también puede ser utilizada durante el propio proceso de construcción del clasificador [19]:

- **Estimación por resustitución (*resubstitution estimate*):** este método utiliza los mismos datos para entrenar y para evaluar el clasificador.
- ***Holdout*:** consiste en dividir los datos en dos conjuntos: uno de entrenamiento, para construir el clasificador, y otro de prueba, para evaluar la precisión del clasificador de forma independiente y así evitar sesgos. Normalmente el conjunto de entrenamiento contiene la mayoría del conjunto de datos (entre un 60 % y un 80 %).
- **Remuestreo (*random subsampling*)**: es similar al método anterior pero se realizan varias particiones de los conjuntos de entrenamiento y prueba. La precisión del clasificador se obtiene como la media para los distintos conjuntos de prueba.
- **Validación cruzada (*cross-validation*):** se realizan k particiones de igual tamaño del conjunto de datos, y a lo largo de k iteraciones se utilizan k-1 particiones para entrenar el clasificador, para luego validar con la partición sobrante. La precisión del clasificador se obtiene como la media de las k precisiones calculadas en cada iteración.
- **Dejar uno fuera (*leave-one-out*)**: se trata de un caso particular de *cross-validation*, donde el número de particiones k es igual al tamaño del conjunto de datos. Este método sólo es óptimo para conjuntos de datos muy pequeños pues tener que construir k clasificadores resulta muy costoso computacionalmente.
- ***Bootstrapping*:** este método se basa en tomar múltiples muestras del conjunto de datos original con reemplazo, construyendo así nuevos conjuntos de datos con el mismo tamaño que el original, donde puede haber datos repetidos. Cada conjunto de datos se usa para entrenar un modelo, y las predicciones de estos modelos se utilizan para obtener una medida más robusta de la precisión o de otras métricas de rendimiento.

Además de los métodos de evaluación anteriores, podemos aplicar combinaciones de estos para un rendimiento del modelo aún mejor. Lo más común es combinar las técnicas de *holdout* y de validación cruzada. Inicialmente se divide el conjunto de datos en entrenamiento y prueba mediante *holdout*. Así se garantiza que el modelo se evalúe con datos que no conoce; de esta forma se evita el sesgo en la evaluación final. Por otro lado, se aplica validación cruzada dentro del conjunto de entrenamiento, lo cual asegura que se maximice el aprendizaje y la

generalización, al entrenar el modelo múltiples veces rotando los conjuntos de entrenamiento y prueba. En la figura 4.1 podemos observar el esquema del método de evaluación propuesto, para $k = 5$.

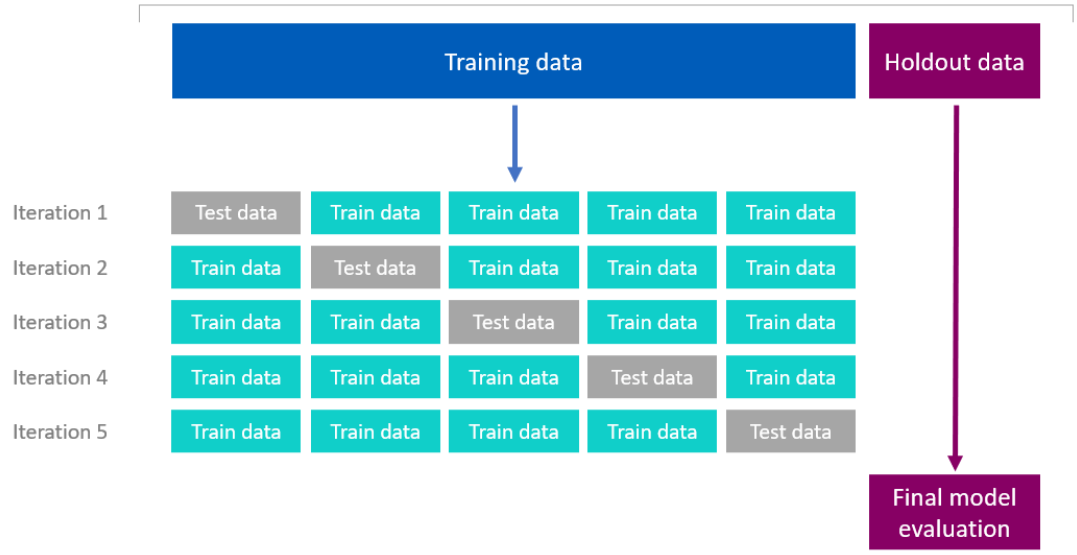


Figura 4.1.: Combinación de holdout y cross-validation [31]

4.7.2. Métricas de evaluación

Tomando como positiva la clase **Spam** y como negativa la clase **Ham**, vamos a definir varias de las métricas que nos servirán para evaluar el rendimiento y la eficacia del modelo construido [16].

En primer lugar representamos la matriz de confusión, que nos proporcionará una visión general del rendimiento del modelo, al mostrar cuántas instancias se han clasificado correctamente e incorrectamente en cada clase. La matriz se compone de las siguientes cuatro medidas:

- **TP (verdadero positivo):** el número de instancias *Spam* que han sido correctamente etiquetadas como *Spam*.
- **FP (falso positivo):** el número de instancias *Ham* que han sido incorrectamente etiquetadas como *Spam*.
- **TN (verdadero negativo):** el número de instancias *Ham* que han sido correctamente etiquetadas como *Ham*.
- **FN (falso negativo):** el número de instancias *Spam* que han sido incorrectamente etiquetadas como *Ham*.

A partir de esta matriz, podemos calcular diversas métricas de evaluación del modelo [16]:

		Clase real	
		Positiva	Negativa
Clase predicha	Positiva	Verdaderos Positivos (TP)	Falsos Positivos (FP)
	Negativa	Falsos Negativos (FN)	Verdaderos Negativos (TN)

Tabla 4.3.: Matriz de confusión para un problema con dos clases

- **Sensibilidad (*recall*):** También conocida como *Tasa de verdaderos positivos (TPR)*, es utilizada para conocer la proporción de instancias Spam que son etiquetadas correctamente.

$$Sensibilidad = \frac{TP}{TP + FN}$$

- **Especificidad (*specificity*):** También conocida como *Tasa de verdaderos negativos (TNR)*, es utilizada para conocer la proporción de instancias Ham que son etiquetadas correctamente.

$$Especificidad = \frac{TN}{TN + FP}$$

- **Precisión (*accuracy*):** Es la proporción de instancias (tanto Spam como Ham) que han sido etiquetadas correctamente.

$$Precisión = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Exactitud (*precision*):** Es la proporción de instancias Spam bien clasificadas entre todos los casos clasificados como positivos.

$$Exactitud = \frac{TP}{TP + FP}$$

- **Tasa de error (*error rate*):** Es la proporción de instancias (tanto Spam como Ham) que han sido etiquetadas incorrectamente.

$$ERR = \frac{FP + FN}{TP + TN + FP + FN}$$

- **F1-Score:** es una métrica muy utilizadas en problemas desbalanceados. Se construye a

partir de la media armónica de la exactitud y la sensibilidad (recall).

$$F1\text{-}score = 2 \cdot \frac{exactitud \cdot recall}{exactitud + recall}$$

...

4.7.2.1. Curva ROC

Ver en libro [16], punto 22.1.3

5. Interpretación de los resultados

Mejor lo incluyo todo en el punto 4

A. Bibliotecas usadas

A.1. Pandas

Pandas [32] es una biblioteca de Python utilizada principalmente para manipulación y análisis de datos. Ofrece estructuras de datos flexibles y herramientas para trabajar con datos tabulares y de series temporales. Estas funciones son fundamentales para realizar tareas comunes de limpieza y manipulación de datos. Algunas de las funciones que hemos empleado son:

- `read_csv`: Una función utilizada para leer datos desde archivos CSV y cargarlos en un DataFrame de Pandas. Esto permite la manipulación y análisis de datos de manera conveniente en Python.
- `drop_duplicates`: Una función utilizada para eliminar filas duplicadas de un DataFrame. Esto puede ser útil cuando se trabaja con conjuntos de datos que pueden contener duplicados y se desea mantener solo una instancia de cada fila única.
- `dropna`: Una función utilizada para eliminar filas o columnas que contienen valores nulos (NaN) en un DataFrame. Esto es útil para limpiar los datos y prepararlos para su análisis, eliminando observaciones incompletas o no válidas.
- `describe`:

A.2. Scikit-learn

Scikit-learn [33] es una biblioteca de aprendizaje automático de Python que ofrece una amplia gama de herramientas para tareas como clasificación, regresión, clustering y más. A continuación, enumeramos algunas de las funciones que hemos empleado en este trabajo:

- `train_test_split`: Una función utilizada para dividir los datos en conjuntos de entrenamiento y prueba.
- `CountVectorizer`: Una función utilizada para convertir una colección de documentos de texto en una matriz de recuentos de términos/palabras.
- `MultinomialNB`: Una implementación del clasificador Naive Bayes multinomial en scikit-learn, adecuado para clasificación de textos con características discretas como recuentos de palabras.
- `ENGLISH_STOP_WORDS`: Una lista predeterminada de palabras vacías en inglés, como artículos, pronombres y preposiciones, utilizada para la eliminación de palabras vacías en el procesamiento de texto.

A.3. NLTK

NLTK (Natural Language Toolkit) [34] es una biblioteca líder para el procesamiento de lenguaje natural (NLP) en Python. Ofrece múltiples de herramientas de procesamiento de texto para la clasificación, tokenización, derivación, etiquetado, análisis sintáctico y razonamiento semántico. Algunas de las funciones que hemos empleado en este trabajo son:

- `word_tokenize`: Una función utilizada para dividir textos en palabras o tokens. Es fundamental en el preprocesamiento de texto para la mayoría de las tareas de NLP, ya que convierte cadenas de texto largas en piezas manejables y analizables.
- `bigrams`: Una función utilizada para generar pares de palabras consecutivas (bigramas) de un listado de tokens. Esta herramienta es esencial para analizar la estructura lingüística y el contexto en el texto, permitiendo estudiar cómo las palabras coexisten de manera adyacente en los documentos.

A.4. wordcloud?

como solo he usado una función y no es tan relevante no se si ponerlo

Glosario

La inclusión de un glosario es opcional.
Archivo: `glosario.tex`

$\mathbb{R}$ Conjunto de números reales.

$\mathbb{C}$ Conjunto de números complejos.

$\mathbb{Z}$ Conjunto de números enteros.

Bibliografía

[1] Usama Fayyad, Gregory Piatetsky-Shapiro, y Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, nov 1996.

[2] María del Consuelo Justicia de la Torre. *Nuevas técnicas de minería de textos: Aplicaciones*. PhD thesis, Universidad de Granada, 2017.

[3] María Novo-Lourés, Reyes Pavón, Rosalía Laza, David Ruano-Ordas, y Jose R Méndez. Using natural language preprocessing architecture (NLPA) for big data text sources. *Sci. Program.*, 2020:1–13, 2020.

[4] Juan Luis Castro. Apuntes de la asignatura Ingeniería del Conocimiento, grado en Ingeniería Informática, Universidad de Granada, Curso 22-23.

[5] Wikipedia contributors. Correo basura. https://es.wikipedia.org/w/index.php?title=Correo_basura&oldid=157238043. Accessed: 2024-2-26.

[6] Tatyana Kulikova. El spam y el phishing en 2022. https://securelist.lat/spam-phishing-scam-report-2022/97582/, February 2023. Accessed: 2024-3-4.

[7] Ivan Belcic. Qué es el spam: guía esencial para detectar y prevenir el spam. https://www.avast.com/es-es/c-spam, January 2020. Accessed: 2024-3-4.

[8] Phishing. https://www.incibe.es/aprendeciberseguridad/phishing. Accessed: 2024-3-22.

[9] José R Méndez, Florentino Fdez Riverola, Fernando Díaz, y Juan M Corchado. Sistemas inteligentes para la detección y filtrado de correo spam: una revisión. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 11(34):63–81, 2007.

[10] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, y Opeyemi Emmanuel Ajibuwa. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6):e01802, 2019.

[11] Wikipedia contributors. Nilsimsa hash. https://en.wikipedia.org/w/index.php?title=Nilsimsa_Hash&oldid=1211187431. Accessed: 2024-3-16.

[12] Apache SpamAssassin: Welcome. https://spamassassin.apache.org/. Accessed: 2024-2-26.

[13] Ion Androutsopoulos. *Learning to filter unsolicited commercial E-mail*. "DEMOKRITOS", National Center for Scientific Research.

[14] Gordon Parker Rios y Hongyuan Zha. Exploring support vector machines and random forests for spam detection. En *CEAS 2004 - First Conference on Email and Anti-Spam*, páginas 30–31, Mountain View, California, USA, 2004.

[15] Francisco Herrera Triguero. Apuntes de la asignatura Aprendizaje Automático, grado en Ingeniería Informática, Universidad de Granada, Curso 23-24.

[16] Mohammed J. Zaki y Wagner Meira. *Data mining and analysis: Fundamental concepts and algorithms*. Cambridge University Press, Cambridge, England, 2014.

[17] Juan Antonio Maldonado Jurado. Apuntes de la asignatura Estadística Descriptiva e Introducción a la Probabilidad, doble grado en Ingeniería Informática y Matemáticas, Universidad de Granada, Curso 19-20.

[18] Uffe Kjærulff y Anders L. Madsen. Probabilistic networks - an introduction to bayesian networks and influence diagrams. Aalborg, Denmark, 2005. Aalborg University.

[19] Francisco Javier García Castellano. *Modelos bayesianos para la clasificación supervisada. Aplicaciones al análisis de datos de expresión genética*. PhD thesis, Universidad de Granada, 2009.

[20] Eva Millán Valldeperas. *Sistema bayesiano para modelado del alumno*. PhD thesis, Universidad de Málaga, 2000.

[21] Víctor Robles Forcada. *Clasificación supervisada basada en redes bayesianas. Aplicación en biología computacional*. PhD thesis, Universidad Politécnica de Madrid, 2003.

[22] Juan Manuel Cabrera Jiménez y Fabricio O. Pérez Pérez. Clasificación de documentos usando naive bayes multinomial y representaciones distribucionales. páginas 1–13, 2011.

[23] 1.9. Naive Bayes. https://scikit-learn.org/stable/modules/naive_bayes.html. Accessed: 2024-5-14.

[24] Sklearn.Naive_bayes.MultinomialNB. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html. Accessed: 2024-2-9.

[25] Harry Zhang. The optimality of naive bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*, 2, 2004.

[26] Nitisha Bharathi. Email Spam Dataset. https://www.kaggle.com/datasets/nitishabharathi/email-spam-dataset/version/1, 2020. Kaggle Dataset. Accessed: 2024-1-30.

[27] Ion Androutsopoulos, John Koutsias, Konstantinos Chandrinos, Georgios Paliouras, y Costantine Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *CoRR*, cs.CL/0006013, 2000.

[28] Leslie Kaelbling y Melinda Gervasio. Enron email dataset. https://www.cs.cmu.edu/~./enron/. Accessed: 2024-1-30.

[29] Spam assassin dataset. https://spamassassin.apache.org/old/publiccorpus/. Accessed: 2024-1-30.

[30] Emmanuel Anguiano-Hernández. Naive bayes multinomial para clasificación de texto usando un esquema de pesado por clases. páginas 1–8, 2009.

[31] Holdout data and cross-validation. https://help.qlik.com/en-US/cloud-services/Subsystems/Hub/Content/Sense_Hub/AutoML/holdout-crossvalidation.htm. Accessed: 2024-5-20.

[32] DataFrame - pandas 2.2.0 documentation. https://pandas.pydata.org/docs/reference/frame.html. Accessed: 2024-2-11.

[33] Scikit-learn. https://scikit-learn.org/stable/index.html. Accessed: 2024-2-9.

[34] NLTK :: Natural language toolkit. https://www.nltk.org/index.html. Accessed: 2024-5-31.