UNIVERSIDAD DE SALAMANCA

FACULTAD DE CIENCIAS

DEPARTAMENTO DE ESTADÍSTICA

TRABAJO DE FIN DE GRADO

USO DE LA CLASIFICACIÓN BAYESIANA EN LA MINERÍA DE DATOS. CREACIÓN DE UNA APLICACIÓN WEB CON R PARA CLASIFICAR O PREDECIR DATOS REALES

Tutor: MIGUEL RODRÍGUEZ ROSA

Autor: MARCO LEDO VALLEJO

Grado en Estadística

Curso académico 2022-23

UNIVERSIDAD DE SALAMANCA

FACULTAD DE CIENCIAS

DEPARTAMENTO DE ESTADÍSTICA

TRABAJO DE FIN DE GRADO

USO DE LA CLASIFICACIÓN BAYESIANA EN LA MINERÍA DE DATOS. CREACIÓN DE UNA APLICACIÓN WEB CON R PARA CLASIFICAR O PREDECIR DATOS REALES

Firmado por:

Autor: Marco Ledo Vallejo

Tutor: Miguel Rodríguez Rosa

Grado en Estadística

Curso académico 2022-23

Certificado del tutor TFG Grado en Estadística

D. Miguel Rodríguez Rosa, profesor del Departamento de Estadística de la Universidad de Salamanca,

HACE CONSTAR:

Que el trabajo titulado "*Uso de la clasificación bayesiana en la minería de datos. Creación de una aplicación web con R para clasificar o predecir datos reales*", que se presenta, ha sido realizado por D. Marco Ledo Vallejo, con DNI 12346771A y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Estadística en esta Universidad.

Salamanca, a 3 de julio de 2023.

RODRIGUEZ ROSA MIGUEL - 70821913Z

Firmado digitalmente por RODRIGUEZ ROSA MIGUEL - 70821913Z
Fecha: 2023.07.05 19:31:34 +02'00'

Fdo.: Miguel Rodríguez Rosa

Índice general

Summary

This work addresses the functioning of Bayesian classification in data mining, in addition to the creation of a web application with R that executes the Naïve Bayes classification method. It begins by introducing the topic at hand and stating a series of objectives sought through the creation of this work. Afterwards, the contents of two databases that will be subsequently used are explained. One of them contains categorical attributes related to lung cancer, while the other contains numerical attributes and provides information about breast cancer. Next, the statistical development of Bayesian classification is discussed to provide a more theoretical understanding of the method. Subsequently, the explanation of the created web application is presented, including a user manual and a description of the necessary codes for its implementation. Following that, a series of results demonstrating the effectiveness of the web application are presented, obtained from two studies conducted for each of the databases. Finally, a set of conclusions drawn from the completion of the work will be observed.

Introducción

Minería de datos

La minería de datos hoy en día toma gran importancia en las empresas, ya que tiene gran culpa de que estas sean lo más competitivas posible en el mercado. Esta rama de la estadística consiste en un proceso que implica descubrir patrones, relaciones y conocimientos útiles a partir de grandes conjuntos de datos, es decir, se creó para poder trabajar con grandes conjuntos de datos de manera que nos ayudara a comprenderlos y que estos nos pudieran servir para sacar información y posteriormente unas conclusiones y soluciones a problemas que nos estuviéramos planteando. La aparición de la minería de datos se podría explicar con la siguiente reflexión de Hernández-Orallo et al. (2004): los datos pasan de ser un "producto" a ser una "materia prima" que hay que explotar para obtener el verdadero "producto elaborado".

Podríamos considerar que su nacimiento aparece en torno a los años 50 y años 60 cuando las empresas utilizan por primera vez los ordenadores para guardar datos y trabajar con ellos, pero de manera muy primaria. Ya en la década de los 80 aparecen las bases relacionales, lo que supone un gran avance en cuanto a la flexibilidad para realizar consultas en las bases de datos. Podemos considerar que la minería de datos ha ido avanzando a lo largo de estos mencionados años, sin embargo, no es hasta los años 90 cuando se utiliza por primera vez esta terminología ("minería de datos"), teniendo gran culpa de ello el trabajo realizado por Agrawal y Ramakrishnan (1994).

En estos años se descubrió el proceso de descubrir patrones y relaciones en grandes conjuntos de datos, además se popularizaron los data warehouses (almacenes de datos). En la actualidad se siguen desarrollando métodos que hagan óptima la búsqueda de información en las bases de datos, pero lo más destacable es el desarrollo de técnicas que permiten el análisis de datos en tiempo real en grandes flujos de información, como son los datos de redes sociales.

Métodos de Minería de Datos

A lo largo de la historia se han ido desarrollando numerosas técnicas y algoritmos relacionados con la Minería de Datos. Para organizar mejor estos algoritmos encontramos principalmente dos tipos de aprendizaje: el supervisado y el no supervisado.

Los algoritmos de aprendizaje supervisado son implementados para predecir o clasificar una variable de salida en función de un conjunto de variables de entrada conocidas. En este enfoque, se utiliza un conjunto de datos de entrenamiento etiquetados previamente con las salidas correspondientes para entrenar un modelo de predicción o clasificación. Luego, se utiliza este modelo para hacer predicciones en nuevos conjuntos de datos. Dentro de este campo encontramos dos grandes

grupos:

- Algoritmos de clasificación. En ellos se pretende predecir un atributo categórico. Dentro de este grupo encontramos técnicas como la clasificación bayesiana, los árboles de decisión o los $k-$vecinos más cercanos, entre otros.
- Algoritmos de regresión. A diferencia de los algoritmos de clasificación, estos predicen valores numéricos continuos. Algunos de los algoritmos de regresión más comunes utilizados en la minería de datos son la regresión lineal, regresión polinómica, redes neuronales ...

Cabe destacar que algunas de estas técnicas mencionadas pueden ser utilizadas tanto para algoritmos de clasificación como para algoritmos de regresión como pueden ser las redes neuronales o los árboles de decisión entre otros.

Los algoritmos de aprendizaje no supervisado tratan de describir unos datos, es decir, a priori no se conocen los valores de la variable respuesta. Lo que realizan estos algoritmos son agrupaciones de los valores de las variables con el objetivo de encontrar grupos similares (clústeres) que puedan ser definidos a posteriori. Las dos grandes familias de algoritmos no supervisados son:

1. Métodos jerárquicos. Estos van generando nuevos grupos automáticamente cada vez más pequeños buscando la agrupación óptima. Encontramos métodos como el método de Ward, método del centroide ...
2. Métodos no jerárquicos. En este caso es necesario alimentar al algoritmo con el valor del número de clústeres que queremos. Algunos ejemplos son $k-$medias, $k-$medioides ...

Ambos métodos pueden ser empleados en un mismo trabajo en el cual un método no jerárquico se apoye en uno jerárquico, de tal manera que utilicemos el método jerárquico para obtener un número de clústeres (apoyándonos en un dendrograma) el cual utilizaremos para implementar el no jerárquico.

Teorema de Bayes

En este trabajo se va a tratar la clasificación bayesiana en minería de datos, la cual se apoya en la estadística bayesiana y en el Teorema de Bayes. Este teorema fue creado por el matemático Thomas Bayes, sin embargo, no es hasta después de su muerte cuando se publica por primera vez en el ensayo "An Essay towards solving a Problem in the Doctrine of Chances" (Un ensayo para resolver un problema de la doctrina de las probabilidades) (Bayes, 1763).

El Teorema de Bayes, en resumidas palabras, trata de obtener una probabilidad condicionada, es decir, nos permite calcular la probabilidad de que suceda un evento A dada la ocurrencia de otro relacionado B. Como dice Peñalva-Martínez y Posadas (2009) buscamos una probabilidad "a posteriori" a partir del conocimiento de unas probabilidades "a priori". Esta probabilidad a priori podemos decir que es la de que ocurra el suceso ($P(A)$) y la probabilidad a posteriori es la que obtendríamos tras conocer cierta información, es decir, la probabilidad del suceso dadas unas observaciones $P(A|B)$

Este teorema se utiliza en una amplia variedad de aplicaciones, incluyendo la estadística, la inteligencia artificial, el aprendizaje automático, la medicina y muchas otras áreas. Resultando especialmente útil para la inferencia y la toma de decisiones en situaciones donde se tienen datos limitados o inciertos, y donde se necesita actualizar la probabilidad de una hipótesis a medida que se recopila más información.

Clasificación Bayesiana

En este trabajo vamos a tratar de obtener predicciones sobre personas con ciertas características relacionadas con el cáncer de pulmón por un lado y por otro lado con el cáncer de mama, de esta manera, gracias a estos atributos y a las relaciones entre ellos obtendremos la probabilidad de que los pacientes padezcan la enfermedad. Para poder hacer este estudio vamos a utilizar la Clasificaión Bayesiana, más concretamente el clasificador Naïve Bayes, ya que este permite trabajar con variables independiendes tanto continuas como de tipo categórico. Además, en las dos bases de datos que vamos a estudiar nuestra variable dependiente es categórica, lo cual es necesario para poder aplicar esta clasificación.

Como ya hemos hablado previamente, este método de clasificación estadística se apoya en el Teorema de Bayes, aplicándolo para obtener la probabilidad a posteriori de que cada individuo pertenezca a cada clase dadas unas características observadas. Pero para poder realizar esta operación, antes es necesario obtener unas probabilidades a priori de manera que primero haya que estimar la probabilidad previa de cada clase al igual que la probabilidad de cada característica dada una clase u otra, es decir, la probabilidad de observar un valor específico de cada atributo para cada una de las clases posibles. Posteriormente, como ya hemos dicho, aplicamos el Teorema de Bayes y, gracias a los cálculos ya realizados, los introducimos en el teorema y obtenemos la probabilidad de cada individuo de pertenecer a cada clase. Finalmente se asigna a cada individuo la clase con la probabilidad posterior más alta.

Si hablamos de ventajas y desventajas de este método frente a otros métodos estadísticos de clasificación nos encontramos con que una de sus principales ventajas es su simplicidad ya que es un método fácil de entender y explicar, además la interpretación de sus resultados es muy sencilla debido a que simplemente hay que fijarse en los valores de las probabilidades obtenidas y quedarnos con el más alto. Otra de sus ventajas es que es flexible en cuanto a la elección del tipo de variables independientes ya que puede tratar tanto con variables continuas como categóricas. Una ventaja más es que es eficaz con conjuntos de datos pequeños debido a que puede trabajar con información previa. Alguna más, es que es efectiva en cuanto a los datos con ruido o incompletos gracias a la probabilidad condicional, la cual hace que estos se clasifiquen con mayor precisión. Sin embargo, en esta clasificación, al igual que en otros métodos, no sólo encontramos ventajas. Una de sus principales desventajas es la suposición de independencia, esto quiere decir que el método siempre asume independencia en los atributos cuando no siempre es así. Otra desventaja es que puede ser menos precisa que otros métodos cuando el conjunto de datos es grande y complejo. También requiere de una buena elección de las distribuciónes de probabilidad a priori, ya que si estas no son precisas, el modelo puede verse afectado negativamente y su rendimiento será menor.

Ahora una vez definidos los términos principales de este trabajo veremos ahora su estructura:

- **Objetivos:** en este apartado inicial se definirán los objetivos del trabajo.
- **Capítulo 1:** nuestro primer capítulo tratará de la explicación de los datos, en nuestro caso serán dos explicaciones ya que trabajaremos dos bases de datos, una para variables categóricas y otra para variables continuas. Se explicará la indormación que contienen estas bases de datos y se detallarán las variables que aparecen en ellas. También se hablará de la limpieza y codificación llevada a cabo para un correcto uso de las mismas. Después, se realizará un breve análisis descriptivo para profundizar y conocer un poco más la información. Por último, se explicará la división necesaria de las bases de datos para aplicar de manera correcta la clasificación bayesiana.

- **Capítulo 2:** en este capítulo, se explicará el desarrollo estadístico de la clasificación bayesiana, comenzando por el teorema de Bayes, donde se habla también de la hipótesis *MAP*. Después, aparecerán las redes bayesianas y seguidamente se explican los clasificadores bayesianos de manera general y el clasificador Naïve Bayes de forma más específica, donde también se enseñará la estimación de sus parámetros.
- **Capítulo 3:** el tercer capítulo habla sobre el software, es decir, sobre la aplicación web creada. Aquí, aparece primero un manual de uso de la app donde se detallan sus *inputs* y sus *outputs*. Luego, se describe el código de ejecución del clasificador Naïve Bayes tanto para atributos categóricos como para los numéricos. Finalmente, se explica también el código de ejecución de la aplicación web creada.
- **Capítulo 4:** el último capítulo corresponde con los resultados obtenidos sobre las dos bases de datos gracias a la aplicación web. Primero se realiza el estudio para las variables categóricas y después para las variables continuas.
- **Conclusiones:** aquí aparecerán las conclusiones obtenidas tras la realización del trabajo.
- **Bibliografía:** se ofrecerá la bibliografía utilizada para la realización de este trabajo. En esta encontraremos diversos libros, artículos, y páginas web.
- **Anexo:** en él encontraremos los códigos de *R* utilizados para la implementación tanto de la página web como del método de clasificación.

Objetivos

En este trabajo tendremos los siguientes objetivos, ordenados de más relevante a menos:

- Objetivo 1: utilizar la clasifiación bayesiana para realizar minería de datos y obtener información útil sobre datos reales.
- Objetivo 2: crear una aplicación web con el programa R para llevar a cabo el procedimiento de clasificación bayesiana, y usarla para clasificar datos reales de manera efectiva y eficiente.
- Objetivo 3: demostrar, gracias a la aplicación web, que la clasificación bayesiana es efectiva tanto para atributos categóricos como para atributos numéricos.
- Objetivo 4: explicar de forma clara y concisa el funcionamiento de la clasificación bayesiana.

Capítulo 1

Base de Datos

Para poder aplicar la clasificación bayesiana a una base de datos es necesario que la variable dependiente sea de tipo categórico, sin embargo, las variables independientes pueden ser tanto categóricas como continuas, por ello, para demostrar este supuesto y ver que este método es fiable con los distintos tipos de variables se va a trabajar con dos bases de datos obtenidas de la plataforma en línea Kaggle. En una aparecerán variables independientes de tipo categórico y en la otra serán continuas. En ambos casos se observará información relacionada con la medicina y el cáncer. La primera contendrá datos sobre el cáncer de pulmón (Kaggle, 2023b) y después se trabajará con ella para tratar de predecir si los pacientes padecen este cáncer. En la segunda base de datos aparecerá contenido sobre el cáncer de mama (Kaggle, 2023a) y posteriormente se intentará predecir si los tumores encontrados son malignos o benignos.

1.1. Base de datos con variables categóricas

La primera base de datos que se va a trabajar contiene información sobre el cáncer de pulmón y características relacionadas con este. En esta base de datos aparecen las variables categóricas. El número de individuos que se puede observar es de 309, los cuales están caracterizados por 16 variables, dos de ellas son demográficas, estas son el género y la edad, la primera toma los valores 'M' si el individuo es un hombre y 'F' en caso de que sea una mujer. Otra es la variable dependiente la cual explica simplemente si los pacientes padecen cáncer de pulmón o no, es decir, puede tomar dos valores: 'YES' si la persona tiene el cáncer o 'NO' si no lo tiene. El resto de variables son las independientes, las cuales dan información sobre las características de las personas y además ayudan a predecir si estos tienen cáncer de pulmón. Estas variables, al igual que la dependiente toman los valores 'YES' en caso de que la persona tenga cierta característica o 'NO' en caso contrario. Sin embargo, en la base de datos aparecen codificadas previamente donde el valor 'YES' se representa con un 2 y el 'NO' con un 1. Estas variables son las siguientes:

- Fumador: indica si la persona fuma o no.
- Dedos amarillos: indica si tiene los dedos amarillos.
- Ansiedad: representa la ansiedad del paciente.
- Presión grupal: dice si el paciente sufre presión de grupo a la hora de fumar.
- Enfermedad crónica: dice si el individuo tienen alguna enfermedad crónica.
- Fatiga: indica la fatiga del paciente.
- Alergia: indica si la persona padece alguna alergia.

- Sibilancias: representa si el paciente padece de sibilancias, las cuales son un sonido silbante y chillón durante la respiración, que ocurre cuando el aire se desplaza a través de los conductos respiratorios estrechos en los pulmones.
- Alcohol: dice si toma alcohol.
- Tos: indica si la persona tose frecuentemente.
- Dificultad para respirar: representa la dificultad para respirar.
- Dificultad para tragar: indica la dificultad del paciente para tragar.
- Dolor torácico: dice si la persona padece de dolor torácico.

1.1.1. Limpieza y codificación

Esta base de datos no se va a utilizar tal y como aparece de inicio, sino que se va a realizar una serie de modificaciones para adaptarla al estudio y que quede con la forma más óptima. Para ello se utilizará la aplicación Excel.

Lo primero que se hizo fue colocar los valores en cada respectiva variable, ya que estos aparecían en la primera columna separados únicamente por comas. Una vez colocados los valores en sus correspondientes variables lo siguiente que se hizo fue eliminar la variable 'Age' debido a que se consideró una variable irrelevante para el estudio, por lo que quedan un total de 15 atributos, todos de tipo categórico. A continuación, se codificaron las variables de género y la variable dependiente 'Lung cancer'. En la primera, las mujeres toman el valor 1 y los hombres el 2, es decir, la categoría 'F' pasa a ser un 1 y la 'M' un 2 . En cuanto a la segunda, el valor 'YES' adopta un 2 y el 'NO' un 1. Otro apartado importante es el de la limpieza de datos faltantes, sin embargo, en este caso no es necesario realizar esta operación ya que no aparece ningún dato de este tipo.

1.1.2. Análisis descriptivo

Para profundizar y conocer un poco más la base de datos procederemos a realizar un sencillo análisis decriptivo de los datos. Para ello, utilizaremos el programa *SPSS*.
Primero observaremos la variable respuesta.

LUNG_CANCER

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	1	39	12,6	12,6	12,6
	2	270	87,4	87,4	100,0
	Total	309	100,0	100,0	

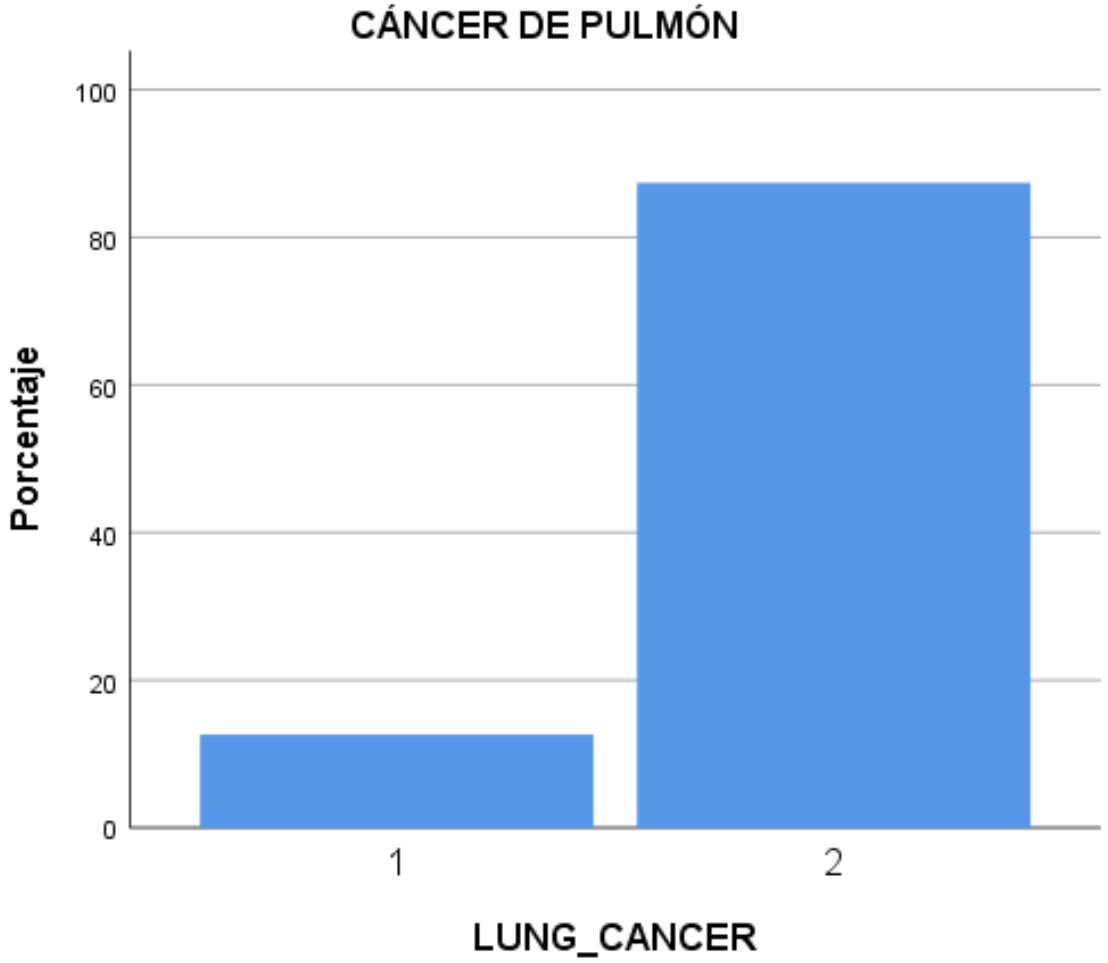

Figura 1.1: Variable *LUNG_CANCER*

Tanto en la tabla como en la gráfica de la figura 1.1 se observa cómo el porcentaje de pacientes que padecen cáncer de pulmón es muy superior al de los individuos sanos, alcanzando los valores de 87.4 % y 12.6 % respectivamente.

A continuación, aparecerán análisis de algunos de los distintos atributos, el primero que veremos es el de la variable *SMOKING*.

SMOKING

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	1	135	43,7	43,7	43,7
	2	174	56,3	56,3	100,0
	Total	309	100,0	100,0	

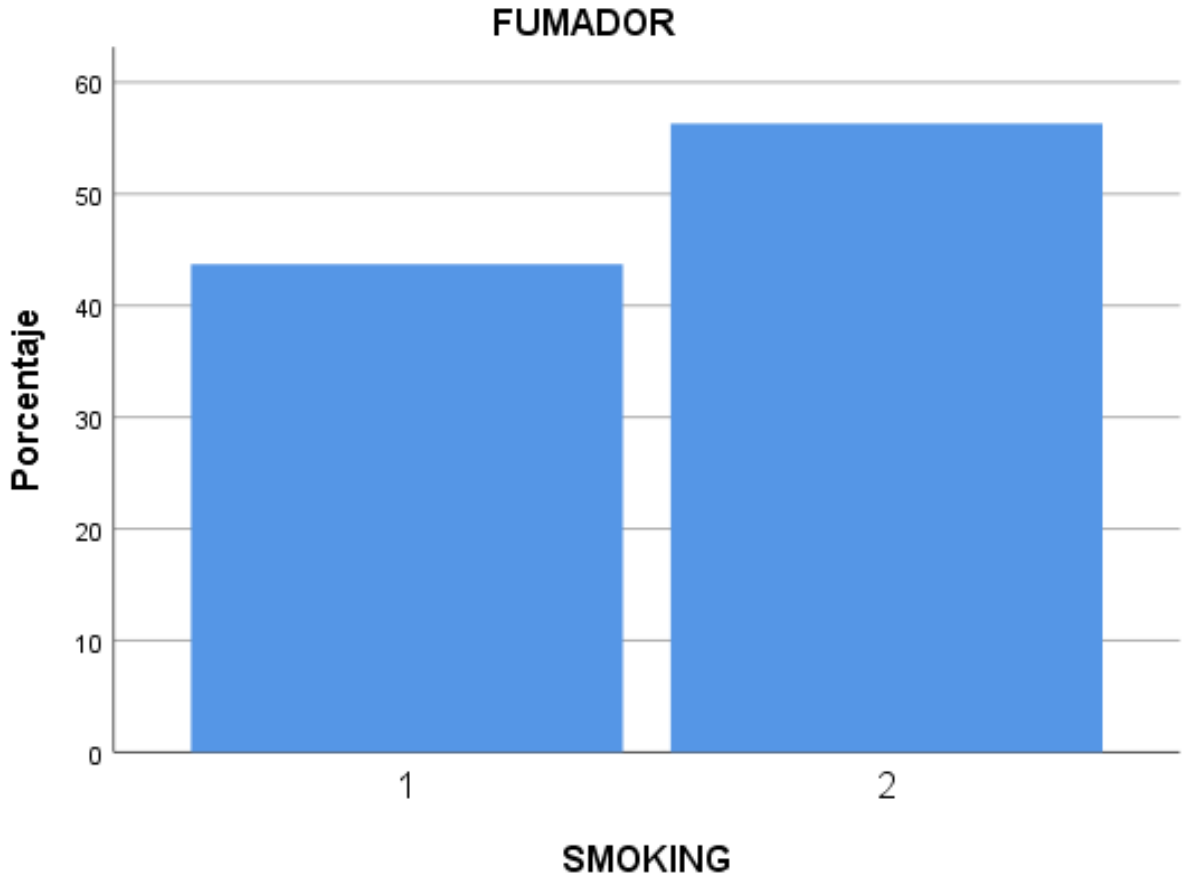

Figura 1.2: Variable *SMOKING*

Vemos que el porcentaje está igualado, aunque se decanta ligeramente por los fumadores, llegando este valor al 56.3 %. Si comparamos el número de los no fumadores con el de personas que no padecen el cáncer, vemos que es mayor el primero, por lo que para padecer cáncer de pulmón no es necesario ser fumador obligatoriamente. Veámoslo con una gráfica de barras agrupadas:

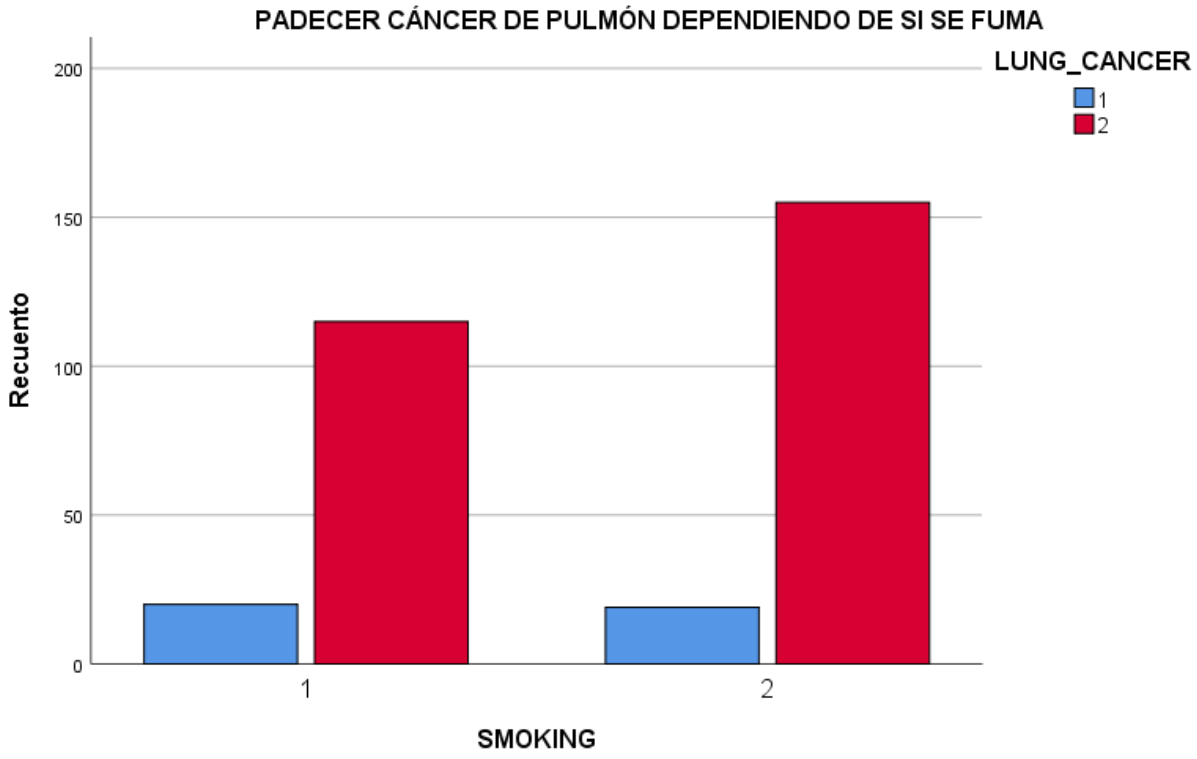

Figura 1.3: Padecer cáncer de pulmón dependiendo de si se fuma

En la figura 1.3 se ve lo dicho previamente, es decir, tanto para fumadores como para no fumadores, el número de personas que padecen cáncer es mucho mayor comparado al de que no padezcan la enfermedad. A continuación, pasaremos a observar la variable *WHEEZING*, correspondiente a las sibilancias de los pacientes.

WHEEZING

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	1	137	44,3	44,3	44,3
	2	172	55,7	55,7	100,0
	Total	309	100,0	100,0	

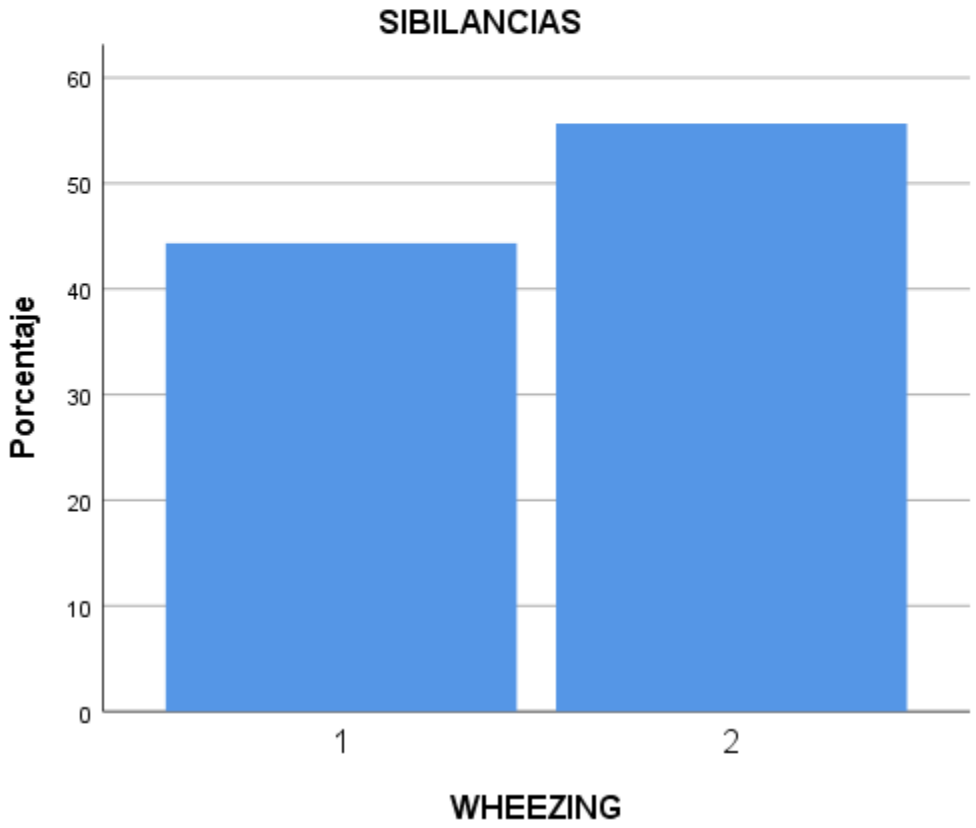

Figura 1.4: Variable *WHEEZING*

Al igual que pasaba con la variable anterior, aparecen porcentajes muy parecidos, sin embargo, el correspondiente a pacientes que sí padecen sibilancias es ligeramente superior alcanzando el 55.77 %. Ahora, compararemos esta variable con la variable respuesta al igual que hicimos anteriormente.

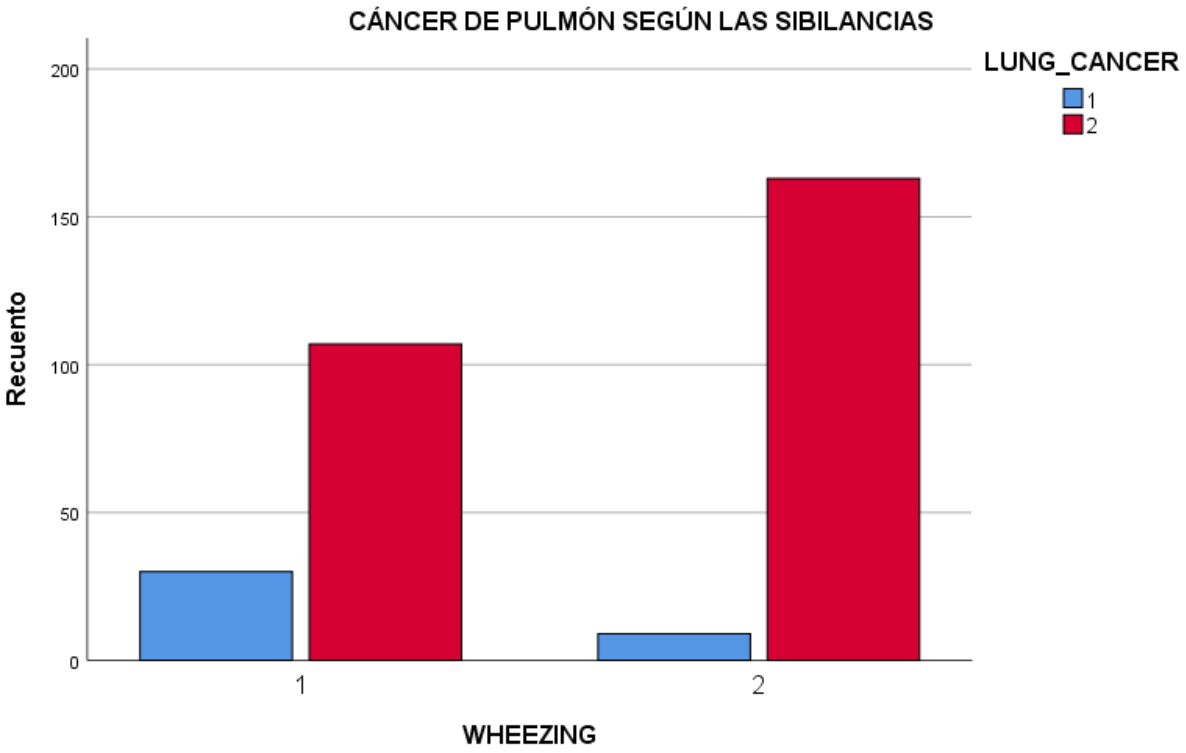

Figura 1.5: Padecer cáncer de pulmón dependiendo las sibilancias

En la figura 1.5 se observa como hay más personas que tienen la enfermedad independientemente de si padecen sibilancias o no, sin embargo, si nos fijamos en los individuos que sí padecen sibilancias, prácticamente la totalidad están enfermas.

SHORTNESS_OF_BREATH

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	1	111	35,9	35,9	35,9
	2	198	64,1	64,1	100,0
	Total	309	100,0	100,0	

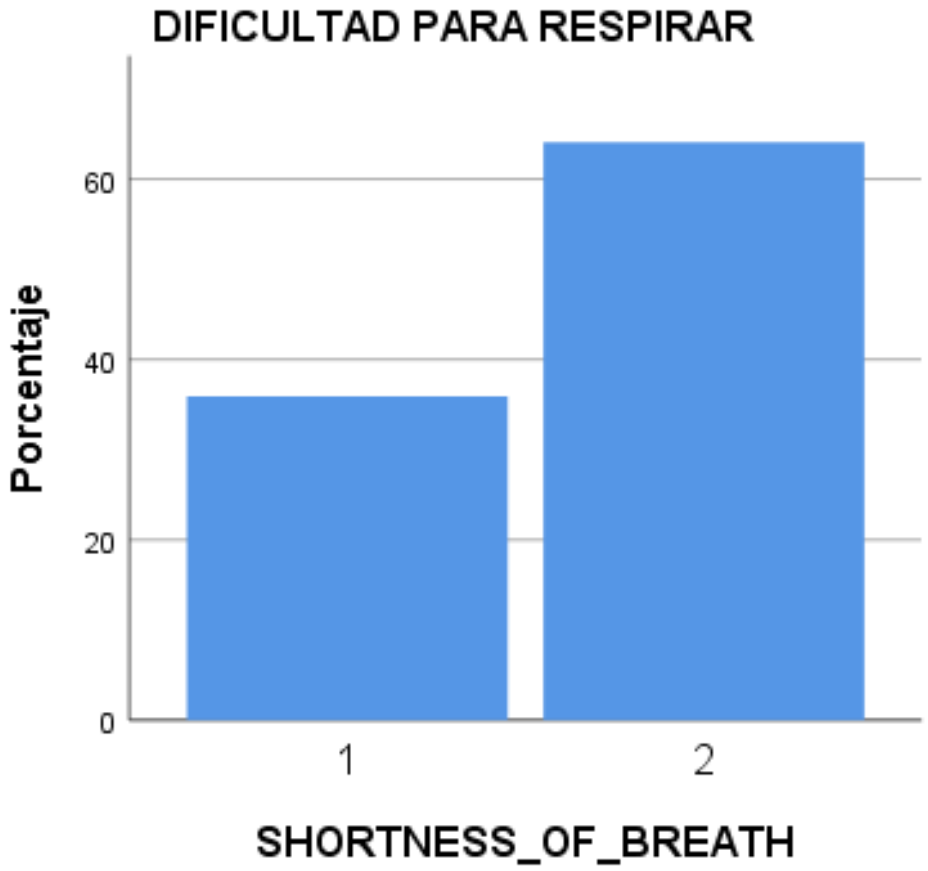

Figura 1.6: Variable *SHORTNESS_OF_BREATH*

En la figura 1.6 aparece la variable dificultad para respirar. En ella se ve como el porcentaje de personas con dificultades para respirar es mayor que el de personas que no padecen este problema. A continuación, haremos un gráfico de barras agrupadas como se ha realizado con las variables anteriores.

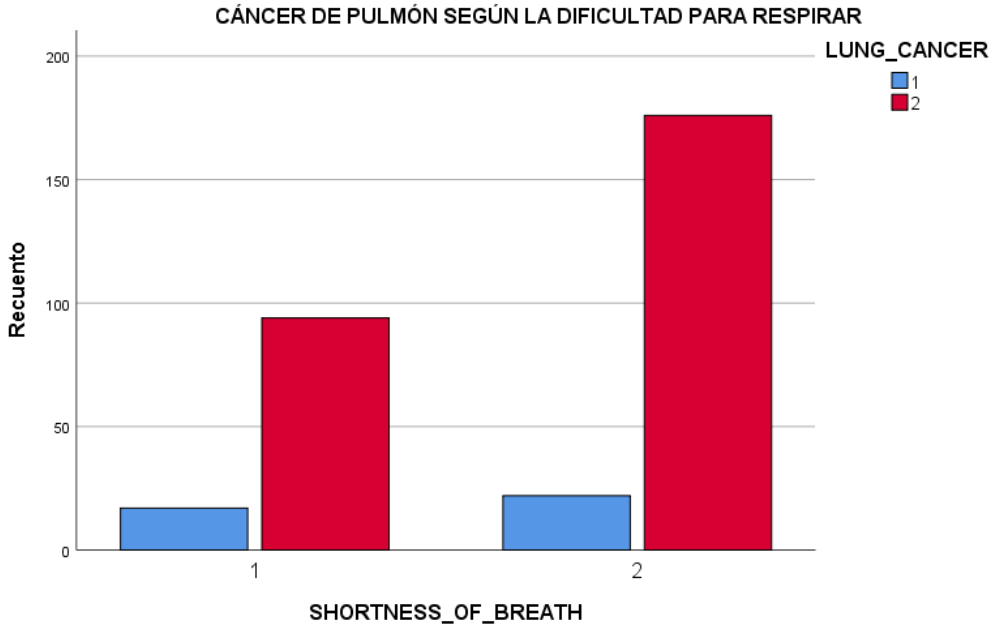

Figura 1.7: Padecer cáncer de pulmón dependiendo de la dificultad para respirar

En este gráfico vemos cómo también predominan de una forma clara las personas con cáncer pulmonar, independientemente de si los individuos tienen dificultad para respirar o no.

1.2. Base de datos con variables continuas

La segunda base de datos sobre la que vamos a trabajar contiene un total de 569 instancias y contiene información sobre mujeres de Wisconsin con cáncer de mama. Esta información observada trata con valores obtenidos a partir de una imagen digitalizada de una aspiración con aguja fina (FNA) de una masa mamaria y describen características de los núcleos celulares presentes en la imagen. Aquí, a diferencia de la base de datos sobre el cáncer de pulmón, encontraremos variables independientes continuas, las cuales van a caracterizar a dichas mujeres con atributos relacionados con el cáncer de mama. Se obtuvo la información de 569 mujeres y nuestra variable dependiente, al igual que en la primera base de datos, es categórica y nos indica simplemente si las pacientes padecen de cáncer de mama maligno o benigno. Esta variable toma los valores 'M' en caso de que el tumor sea maligno y 'B' en caso de que sea benigno. También encontramos una variable llamada 'id', la cual se trata de un número identificativo y único para cada paciente. En total observamos 32 variables, pero si restamos las ya explicadas encontramos que el número de variables independientes que aparecen es 30. Sin embargo, no es el número real, ya que digamos que cada atributo está caracterizado por la media, el error estándar y el valor más alto de cada variable para cada paciente (el valor más alto es considerado el peor). Realmente nos interesa únicamente el valor de la media de cada variable por lo que el número de atributos independientes verdaderamente sería 10. A continuación, observaremos cuáles son estas variables:

- Radio: representa una media de las distancias desde el centro del núcleo al perímetro.
- Textura.
- Perímetro.
- Área.

- Uniformidad: variación local en las longitudes del radio.
- Compacidad: medida que describe qué tan compacto o redondeado es un objeto en relación a su área y perímetro. Cuanto más cercano a cero sea el valor de la compacidad, más compacto y redondeado será el objeto. Su fórmula es la siguiente: $\frac{\text{perímetro}^2}{\text{área}} - 1$.
- Concavidad: severidad de las porciones cóncavas del contorno.
- Puntos cóncavos: número de porciones cóncavas del contorno.
- Simetría.
- Dimensión fractal: "aproximación de la línea de costa" − 1.

1.2.1. Limpieza y codificación

Al igual que en la primera base de datos, en esta también se van a realizar cambios sobre la base inicial en cuanto a la limpieza y codificación de variables. De la misma manera se utilizará la aplicación Excel.

Al abrir la base de datos aparece el mismo problema que sucedía con la anterior base y es que todos los valores de todas las variables se encuentran en la primera columna, separados únicamente por comas, así que lo primero es colocar los valores en sus correspondientes atributos. Posteriormente, se procede a la eliminación de las variables irrelevantes para el estudio. Como ya se ha mencionado antes, para el estudio serán necesarias únicamente las variables que representen la media de las características así como la variable dependiente que indica el tipo de tumor. Entonces, se elimina la variable 'id', al igual que todos los atributos que indican el error estándar y el valor más alto de cada variable para cada paciente. En total, el número de variables que queda es de 11, una de ellas de tipo categórico y 10 de tipo continuo. A continuación, se procede a codificar la variable dependiente, de tal manera que el valor 'B' será un 1 y el valor 'M' será representado con un 2. En cuanto al apartado de eliminación de datos faltantes aparece la misma situación que en la anterior base de datos y es que no hay ninguno de estos, por lo que no se aplica ningún procedimiento para su eliminación.

1.2.2. Análisis descriptivo

Al igual que se hizo con la base de datos para variables categóricas, se realizará un análisis descriptivo para profundizar un poco más en estos datos. Para este análisis utilizaremos el programa *SPSS* como se hizo con la base de datos del cáncer de pulmón.
Lo primero que se realizará será una visualización de la variable respuesta, la cual nos indica si las mujeres estudiadas padecen de un tumor benigno o maligno.

diagnosis

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	1	357	62,7	62,7	62,7
	2	212	37,3	37,3	100,0
	Total	569	100,0	100,0	

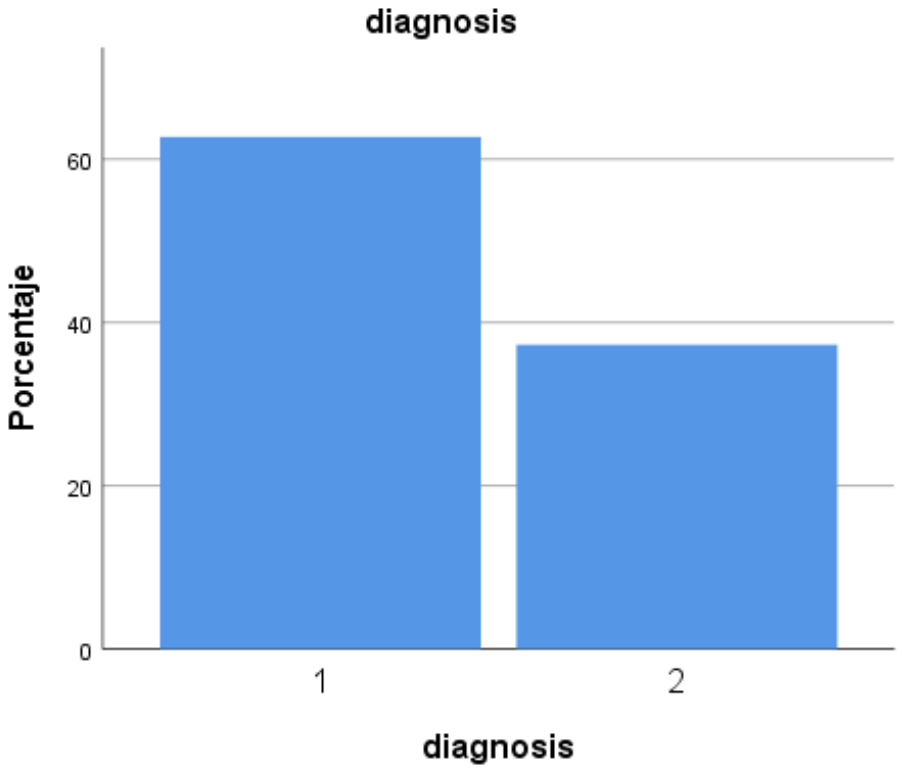

Figura 1.8: Variable *DIAGNOSIS*

En la figura 1.8 se observa como la mayoría de las personas de la base de datos tiene un tumor benigno, llegando al 62.7 %, sin embargo, las mujeres que padecen un tumor maligno alcanzan el 37.3 % del total. Ahora, se procederá a la visualización de una serie de valores estadísticos correspondientes a cada variable independiente de la base de datos.

Estadísticos

	N				
	Válido	Perdidos	Media	Desv. Desviación	Rango
radius	569	0	14,12729	3,524049	21,129
texture	569	0	19,2896	4,30104	29,57
perimeter	569	0	91,9690	24,29898	144,71
area	569	0	654,889	351,9141	2357,5
smoothness	569	0	,0963603	,01406413	,11077
compactness	569	0	,1043410	,05281276	,32602
concavity	569	0	,08879932	,079719809	,426800
concave points	569	0	,04891915	,038802845	,201200
symmetry	569	0	,181162	,0274143	,1980
fractal_dimension	569	0	,0627976	,00706036	,04748

Figura 1.9: Estadísticos de los atributos numéricos

En la figura 1.9, se encuentra un resumen general de todos los atributos numéricos de la base de datos. De ella, se pueden destacar varias cosas: la primera es que la media más alta que aparece corresponde a la variable *area*, con un valor de aproximadamente 655 y la menor media pertenece a *concave_points* con 0.0489 aproximadamente. La variable con mayor desviación típica también es *area*, alcanzando un valor de 351.9141 y el atributo con menor desviación típica es *fractal_dimension* con 0.007. Finalmente, si nos fijamos en el rango obtenemos las mismas conclusiones que con la desviación típica.

1.3. Entrenamiento, testeo y predicción

Para la realización del estudio va a ser necesario dividir ambas bases de datos en tres partes: una de entrenamiento otra de testeo y otra de predicción. Esto es algo muy importante para el desarrollo de modelos de aprendizaje automático, ya que nos permite entre otras comprobar el rendimiento del modelo con el que se está trabajando.

Los datos de entrenamiento sirven para alimentar el algoritmo y de esta manera poder definir el modelo. Durante el proceso de entrenamiento, el modelo ajusta sus parámetros y ponderaciones internas en función de los datos de entrenamiento para lograr un rendimiento óptimo. Después de la fase de entrenamiento viene la parte de testeo, donde con otros individuos distintos a los de entrenamiento, evaluaremos y mediremos la precisión del modelo que hemos entrenado, es decir, permite validar el modelo. Esto se realizará gracias al valor que nos devolverá el testeo, el cual oscilará entre 0 % y 100 %, si es un buen modelo estará cercano al 100 % y en caso contrario se aproximará a 0 %. Esto proporciona una estimación objetiva de cómo se comportará el modelo en situaciones reales y cuán confiable es para realizar predicciones o clasificaciones precisas. Otra de las ventajas que tiene el testeo es que permite comparar modelos, por ejemplo, si estamos dudando

entre dos a elegir nos puede ayudar a seleccionar el más preciso. En estas dos primeras partes serán necesarias todas las variables, sin embargo, en la última parte, la predicción, eliminaremos la variable respuesta, ya que el objetivo de este apartado es predecir los valores de esta simulando una situación real, es decir, clasificaremos a los individuos de estudio asignándoles un 1 (“no padece de cáncer” en la base de datos del cáncer de pulmón, o en el caso de la base de datos del cáncer de mama indicará que el cáncer es “benigno”) o un 2 (si “padece de cáncer de pulmón” en la primera base de datos o si el cáncer es “maligno” si hablamos de la segunda) según sus características gracias al modelo ya entrenado.

Capítulo 2

Desarrollo estadístico

2.1. Teorema de Bayes

Como ya se ha mencionado previamente, el teorema de Bayes es utilizado en la clasificación bayesiana con el objetivo de obtener conclusiones sobre problemas propuestos gracias a probabilidades calculadas para unas instancias con unas ciertas características. Dicho teorema expresa la probabilidad condicional de un evento aleatorio A dado el evento B, en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de A. Explicado de una forma más coloquial sería: premite pasar de una probabilidad a priori (probabilidad del suceso de estudio) a una probabilidad a posteriori, en la que se obtiene una información más detallada de los resultados gracias a ciertas observaciones, es decir, premite un refinamiento de la información. Su fórmula se representa de la siguiente manera:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.1}$$

Donde $P(A|B)$ es la probabilidad condicional de que ocurra el evento A dado que el evento B ha ocurrido. $P(B|A)$ es la probabilidad condicional de que ocurra el evento B dado que el evento A ha ocurrido, y se conoce con el nombre de verosimilitud. $P(A)$ es la probabilidad marginal de que ocurra el evento A y $P(B)$ es la probabilidad marginal de que ocurra el evento B.

Esta ecuación se obtiene gracias a las probabilidades conjuntas. Por definición, la probabilidad condicional es la siguiente:

$$P(A|B) = P(A \cap B) \cdot P(B) \tag{2.2}$$

siendo $P(A \cap B)$ la probabilidad conjunta de los eventos A y B. Despejando se obtiene:

$$P(A \cap B) = P(A|B) \cdot P(B).$$

Esta expresión es conmutativa, es decir:

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

y

$$P(A \cap B) = P(B|A) \cdot P(A)$$

entonces, despejando de la ecuación de probabilidad condicional (2.2) se llega a

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

Una forma mejor de entender esta fórmula es con un ejemplo. Supongamos que se está realizando un estudio sobre el covid sabiendo que el uso de las mascarillas tiene inluencia en la tenencia del mismo. Se quiere ver la probabilidad de padecer covid en personas que usan frecuentemente la mascarilla, es decir, se quiere calcular $P(\text{Covid} = \text{Sí} \mid \text{Mascarilla} = \text{Sí})$. En este caso, la investigación de si la persona tiene covid es el suceso de estudio y el uso de mascarilla es una observación que se realiza para pulir el resultado. La fórmula quedaría de la siguiente forma:

$$P(\text{Covid} = \text{Sí} \mid \text{Mascarilla} = \text{Sí}) = \frac{P(\text{Mascarilla} = \text{Sí} \mid \text{Covid} = \text{Sí}) \cdot P(\text{Covid} = \text{Sí})}{P(\text{Mascarilla} = \text{Sí})}$$

es decir, depende de: cómo de verosímil es que una persona haya usado mascarilla si se sabe que padece covid, cuál es la prevalencia de covid a priori, y cuál es la probabilidad de que una persona use mascarilla.

2.1.1. Hipótesis MAP

Centrándonos en el ámbito de la clasificación, sabemos que B representa un conjunto de n variables predictoras $B_1, \ldots, B_n$ y que nuestra hipótesis A tiene k posibles valores, entonces para cada uno de estos valores la fórmula se representaría:

$$P(A_i|B_1, \ldots, B_n) = \frac{P(B_1, \ldots, B_n|A_i) \cdot P(A_i)}{P(B_1, \ldots, B_n)} \tag{2.3}$$

donde i representa cualquiera de los k valores que toma nuestra hipótesis.

Trasladando estos conocimientos a la vida real, donde la clase objetivo necesita determinarse, el objetivo sería buscar el valor que mejor se adapte a cada individuo respecto a sus características, entonces uno de los k valores de la variable respuesta resultará ser el más óptimo para las características estudiadas $B_1, \ldots, B_n$. En el caso del clasificador Naïve Bayes (clasificador bayesiano utilizado en este trabajo), el resultado más plausible será aquel que tenga máxima probabilidad a posteriori dados unos atributos seleccionados previamente. Esta hipótesis, con la que se obtiene el mejor resultado, es conocida con las siglas MAP (del inglés, maximum a posteriori) y su representación es la siguiente:

$$A_{MAP} = arg\left(\max_{A \in \Omega_A} \{P(A|B_1, \ldots, B_n)\}\right) = arg\left(\max_{A \in \Omega_A} \left\{\frac{P(B_1, \ldots, B_n|A) \cdot P(A)}{P(B_1, \ldots, B_n)}\right\}\right) = \tag{2.4}$$

$$= arg\left(\max_{A \in \Omega_A} \{P(B_1, \ldots, B_n|A) \cdot P(A)\}\right) \tag{2.5}$$

donde la instrucción $arg\left(\max_{A \in \Omega_A} \{\cdot\}\right)$ compara todos los resultados obtenidos y muestra el valor del argumento para el cual se alcanza el máximo entre ellos, y Ω_A representa el conjunto de valores que puede tomar la variable A. Nótese que en el paso de la expresión (2.4) a (2.5) desaparece el denominador ($P(B_1, \ldots, B_n)$), esto es debido a que es el mismo para todos los valores de A y por lo tanto no tiene influencia al comparar $P(A|B_1, \ldots, B_n) \cdot P(A)$ para todos los valores de A.

Como ya se ha mencionado previamente, la verosimilitud se representa con $P(B|A)$ siendo A la clase objetivo y B los atributos utilizados en el estudio. La máxima verosimilitud se alcanza seleccionando el valor más alto de esta probabilidad y se representaría:

$$A_{ML} = arg\left(\underset{A\in\Omega_A}{\text{máx}}\{P(B_1,\ldots,B_n|A)\}\right), \tag{2.6}$$

y como se menciona en Ye (2013), si $P(A) = P(A')$ para cualquier $A \neq A'$ donde $A \in \Omega_A$ y $A' \in \Omega_A$, entonces en ese caso:

$$A_{MAP} = arg\left(\underset{A\in\Omega_A}{\text{máx}}\{P(B_1,\ldots,B_n|A)\cdot P(A)\}\right) = arg\left(\underset{A\in\Omega_A}{\text{máx}}\{P(B_1,\ldots,B_n|A)\}\right),$$

y, por lo tanto,

$$A_{MAP} = A_{ML}.$$

2.2. Redes bayesianas

Según se afirma en Hernández-Orallo et al. (2004), una definición formal de red bayesiana sería:

"Una tupla $B = (G, \Theta)$, donde G es un grafo y Θ es el conjunto de distribuciones de probabilidad $P(X_i \mid Pa(X_i))$ para cada variable desde $i = 1$ hasta n, y $Pa(X_i)$ representa los padres de la variable X_i en el grafo G".

Explicándolo de una manera menos formal diríamos que las redes bayesianas son modelos estadísticos que representan relaciones probabilísticas entre variables. Estas redes se reflejan mediante grafos dirigidos acíclicos, los cuales reproducen las dependencias condicionales entre variables. Cada una de estas variables está representada por un nodo y las dependencias probabilísticas entre variables se reflejan mediante aristas dirigidas. Que el grafo sea dirigido quiere decir que los enlaces entre los vértices del grafo están orientados y que sea acíclico significa que no pueden existir ciclos en el grafo, esto es, que si comenzamos a recorrer un camino dirigido desde un nodo, nunca podríamos regresar al punto de partida.

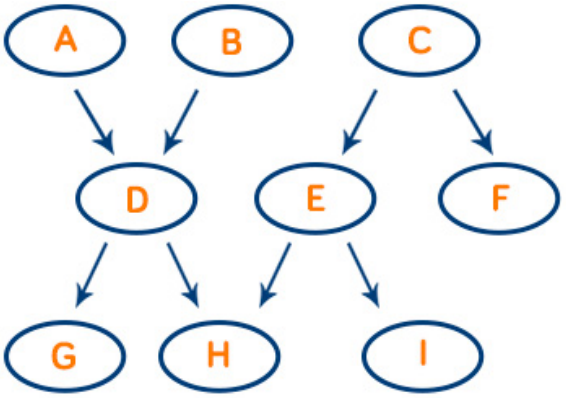

Figura 2.1: Red bayesiana

En la figura 2.1 aparece un ejemplo de red bayesiana donde se pueden observar las características

mencionadas. Las letras rodeadas representan cada una de las variables, las dependencias probabilísticas se reflejan gracias a las "flechitas"que, como se ve, nacen en un nodo y finalizan en otro al que están apuntando, y también se puede demostrar que es acíclico ya que si, por ejemplo, iniciamos en la variable A, no habría ningún camino posible para volver a ella. Ahora, vamos a fijarnos en el grupo de los nodos $\{A, B, D, G, H\}$, entonces decimos que A y B son nodos padre de D y, a su vez, D es nodo padre de G y H.

Cada variable tiene asociada una distribución de probabilidad, esto es, por ejemplo, una variable que es padecer una enfermedad y sus probabilidades son 70 % para el sí y 30 % para el no. En el caso de que esta sea nodo hijo, la distribución que tendrá será una distribución de probabilidad condicional, en la que sus valores dependerán de los valores de sus nodos padres. Una vez implementada la red bayesiana y gracias al Teorema de Bayes y las probabilidades condicionales se podría realizar inferencia sobre nuevos eventos desconocidos a priori.

Al estar en un entorno probabilístico, las redes bayesianas vienen expresadas por una distribución de probabilidad conjunta definida sobre las variables del modelo. Esta distribución es la siguiente:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa(X_i)) \tag{2.7}$$

Donde $P(X_i \mid Pa(X_i))$ representa la probabilidad de obtener X_i dados unos valores previos de sus nodos padres.

2.3. Clasificadores bayesianos

Como afirman Sucar y Tonantzintla (2006), un clasificador bayesiano se puede ver como un caso especial de una red bayesiana en la cual hay una variable especial que es la clase y las demás variables son los atributos. El objetivo de estos clasificadores es, como su propio nombre indica, clasificar instancias de datos en diferentes categorías o clases predefinidas según características concretas, gracias a una información previa y a estimaciones de probabilidades condicionales. La asignación de una clase a una instancia se hará seleccionando la clase con mayor probabilidad. Con un ejemplo se verá más claro: estamos realizando un estudio en el que queremos clasificar si los nuevos estudiantes de la Universidad de Salamanca compran un ordenador (siendo la categoría *Sí* si lo compran y *No* si no lo hacen) según el nivel económico de cada uno, donde las categorías de la variable nivel económico son A para un nivel económico alto, M para un nivel medio y B para bajo. Tras la realización de los cálculos pertinentes hechos previamente y gracias al Teorema de Bayes se obtuvo que los nuevos estudiantes con un nivel económico A compran un ordenador con probabilidad del 85 % y no lo compran con un 15 %, entonces, automáticamente, cuando aparezca un nuevo estudiante con un nivel económico A, el clasificador asignará el valor *Sí* a la variable comprar ordenador.

2.4. Clasificador Naïve Bayes

Resulta el modelo más simple de clasificación con redes bayesianas. Este modelo, además de basarse en el Teorema de Bayes, tiene como fundamento principal el supuesto de independencia de todas las variables que representan las características del modelo, algo que puede llegar a impresionar ya que no siempre es realista. Sin embargo, a pesar de esta suposición, se ha demostrado en varios trabajos que resulta un método muy eficaz y que además es muy fácil de implementar.

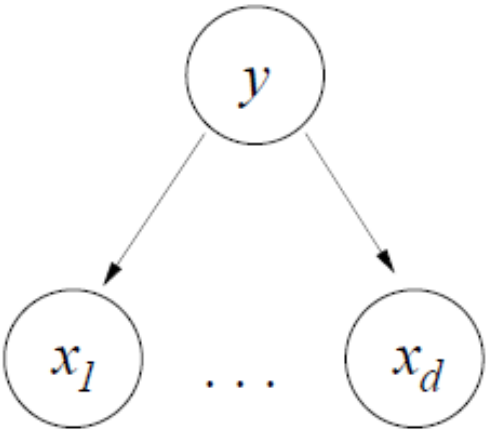

Figura 2.2: Red Naïve Bayes

En la figura 2.2 se puede observar la estructura de red del clasificador Naïve Bayes, donde Y es la variable respuesta y nodo padre de todas las variables $X_1, \ldots, X_d$, las cuales son atributos y nodos hijo. Además, se puede ver la hipótesis de independencia entre los atributos, ya que no salen aristas dirigidas desde las características entre sí.

Se considera el clasificador bayesiano más simple debido a que gracias al ya mencionado supuesto de independencia se simplifica significativamente el cálculo de las probabilidades condicionales. Entonces, el cálculo de la hipótesis MAP para el clasificador Naïve Bayes se puede resumir de la siguiente manera:

Con la ecuación ya explicada de la hipótesis MAP,

$$y_{MAP} = arg\left(\max_{y\in\Omega_y}\{P(X_1,\ldots,X_d|y)\cdot P(y)\}\right),$$

donde en este caso la variable respuesta es la Y, los atributos son $X_1, \ldots, X_d$, y, por lo tanto, $P(y)$ es la probabilidad a priori de obtener cierta clase de la variable respuesta, y $P(X_1, \ldots, X_d|y)$ es la verosimilitud. Entonces, gracias al supuesto de independencia, podemos obtener la verosimilitud como

$$P(X_1,\ldots,X_d|y) = \prod_{i=1}^{d} P(X_i|y), \tag{2.8}$$

y, por lo tanto,

$$y_{MAP} = arg\left(\max_{y\in\Omega_y}\left\{P(y)\cdot\prod_{i=1}^{d}P(X_i|y)\right\}\right). \tag{2.9}$$

2.4.1. Estimación de los parámetros

En la ecuación (2.9) se puede observar que los parámetros a estimar para el clasificador Naïve Bayes son la probabilidad a priori de la variable clase $P(y)$ y las verosimilitudes $P(X_i|y)$ para cada atributo. Como ya se ha mencionado previamente, sabemos que este clasificador puede trabajar de manera efectiva tanto con atribuos categóricos como continuos (refiriéndonos a las variables independientes), sin embargo, la estimación de los parámetros se realiza de manera distinta para cada tipo de atributo. En cuanto a la estimación de $P(y)$, esta siempre será igual, debido a que la variable respuesta siempre tiene que ser categórica para poder aplicar el método. Entonces, primero observaremos cómo realizar la estimación de $P(y)$, la cual se realiza obteniendo la proporción de registros que tienen cierta clase de la variable respuesta entre el número total de registros, esto es:

$$P(y) = \frac{n_y}{n}. \tag{2.10}$$

donde n representa el número total de registros de la base de datos y n_y, el número de registros que tienen como clase objetivo el valor y.

A continuación, se procederá al cálculo de la estimación de los parámetros de las verosimilitudes tanto para atributos categóricos como continuos:

- Atributos categóricos: para las variables categóricas la estimación de la probabilidad condicional se basa en las frecuencias de aparición de cada valor de las variables independientes según su variable respuesta, esto es,

$$P(X_i = x_i|Y = y) = \frac{n_{y\&x_i}}{n_y} \tag{2.11}$$

  donde $n_{y\&x_i}$ representa el número de registros de la base de datos en que la variable X_i toma el valor x_i y además su variable respuesta es y. Esta técnica se conoce como estimación por máxima verosimilitud.

- Atributos continuos: en este caso, lo que hace el clasificador Naïve Bayes para estimar las probabilidades condicionales es asumir que el atributo en cuestión sigue una distribución normal con media μ y desviación típica σ. Por lo que habría que calcular la media y la desviación típica de los atributos con clase y en la variable respuesta. Esto sería:

$$P(X_i = x_i|Y = y) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{-\frac{1}{2} \cdot \frac{(X-\mu)^2}{\sigma^2}}, \tag{2.12}$$

  pues se supone que $X_i \equiv N(\mu, \sigma)$

Capítulo 3

Software

Para poder aplicar el clasificador Naïve Bayes a situaciones reales se creará una aplicación web con la interfaz *RShiny* (Chang, 2020) del programa *R*, en la cual, gracias a la introducción de una base de datos y de otros parámetros, el programa deloverá la clasificación de ciertos registros según pertenezcan a una clase u otra. Esta app está formada internamente por dos códigos principales, uno para atributos categóricos y otro para atributos contínuos.

3.1. Manual del Software

A continuación, se explicará el funcionamiento de la aplicación.

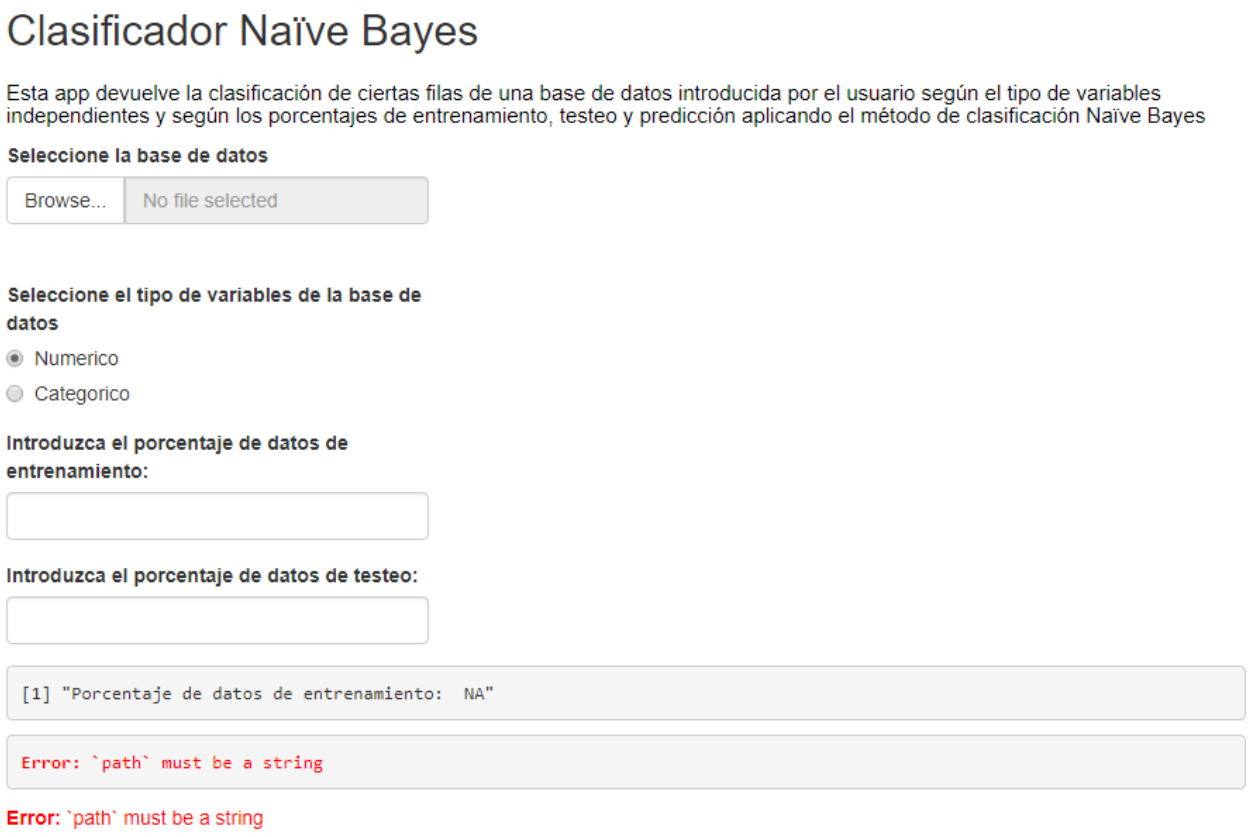

Figura 3.1: Interfaz app web

En la figura 3.1 se observa la interfaz de la aplicación, la cual recibe el nombre de *Clasificador Naïve Bayes* y justo debajo aparece una breve explicación que indica en qué consiste. Se puede observar como al final aparecen dos errores, no hay que alarmarse, ya que esto es debido a que aún no se ha introducido ninguna base de datos, en cuanto se mete este parámetro los errores desaparecen. Esta aplicación web está formada por cuatro parámetros de entrada, llamados *inputs*, y tres parámetros de salida o *outputs*. Se comenzará explicando los *inputs* necesarios para la ejecución:

- **Selección de la base de datos:** este *input* es el primero que hay que introducir al ejecutar la aplicación, consiste en la elección de la base de datos con la que va a trabajar la app. Esta tiene que contener una variable respuesta de tipo categórico que aparezca en la última columna de la base de datos y una serie de variables independientes que caracterizen a cada registro, estas pueden ser bien de tipo numérico o de tipo categórico. Además, la base de datos tiene que ser de tipo '.xlsx' para que el programa lo lea de forma correcta.

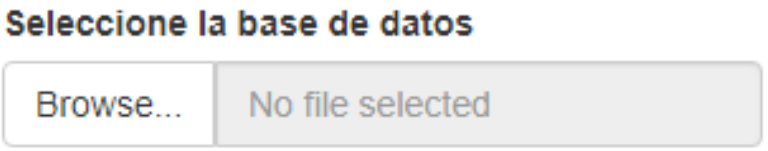

Figura 3.2: Primer *input*

Para la introducción de este *input* hay que hacer clic sobre *Browse*, al hacer eso, aparecerá en pantalla el explorador de archivos del ordenador, una vez ahí, el usuario debe buscar y seleccionar la base de datos a la que se le quiere aplicar el método.

- **Selección del tipo de variables de la base de datos:** el segundo *input* trata de la elección del tipo de variables independientes que tiene la base de datos introducida, ya sean o bien de tipo numérico o de tipo categórico.

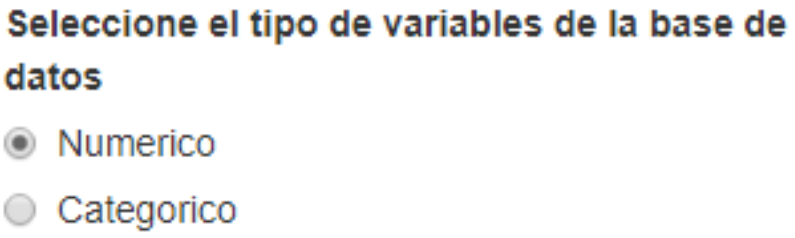

Figura 3.3: Segundo *input*

Como se ve en la figura 3.3 aparecen dos posibilidades de selección y estas opciones son *Numerico* y *Categorico*. La aplicación solo nos dejará seleccionar una de las dos y por defecto al iniciar la app aparece selecionada la opción *Numerico*, entonces, si las variables son de tipo numérico no es necesario cambiarla, sin embargo, si estas son de tipo categórico habría que hacer clic sobre la opción *Categorico*. Cada una de estas dos posibilidades hace que el programa ejecute un código u otro, ya que hay dos distintos dependiendo del tipo de variable.

- **Introducción del porcentaje de datos de entrenamiento:** este *input* consiste en la introducción desde teclado del porcentaje de base de datos que el usuario quiere para la parte del

entrenamiento del modelo. Este porcentaje afecta únicamente a las filas, ya que en las columnas aparecen las variables y son todas necesarias para la correcta ejecución del método.

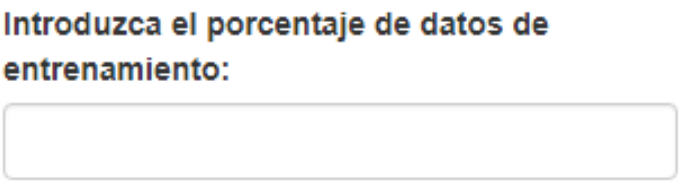

Figura 3.4: Tercer *input*

En el recuadro justo debajo del texto de la figura 3.4 es donde hay que introducir por teclado el porcentaje. Este tiene que oscilar entre 0 y 1, ya que así se ha definido en el código.

- **Introducción del porcentaje de datos de testeo:** el último *input* consiste en la introducción, también por teclado, del porcentaje de base de datos que se desea utilizar para la parte de testeo y al igual que el anterior *input* afecta únicamente a las filas.

Figura 3.5: Cuarto *input*

Se ejecuta de la misma forma que el anterior parámetro de entrada, es decir, hay que introducir un número entre 0 y 1 en la casilla correspondiente. Es importante que la suma de ambos porcentajes no supere el valor 1 ya que como se explicará más adelante, la aplicación calcula el porcentaje para la parte de predicción restando la suma de los porcentajes ya introducidos al total, el cual es 1.

Una vez detallados los *inputs*, es momento de pasar a la explicación de los *outputs* los cuales devuelven una serie de cálculos implementados internamente mediante código. Como ya hemos dicho la aplicación cuenta con tres *outputs*:

- **Porcentaje de datos de entrenamiento:** este *output* delvuelve el porcentaje faltante correspondiente a los datos de predicción, el cuál es obtenido sumando los anteriores porcentajes y restándoselo al total. Al igual que los anteriores, este afecta únicamente a las filas de la base de datos.

Figura 3.6: Primer *output*

Como se observa en 3.6 el resultado aparece como 'NA' debido a que aún no se ha introducido ninguna base de datos, en cuanto se realiza esta acción, el resultado en forma de porcentaje aparece y sustituye a 'NA'.

- **Bondad de ajuste:** el segundo *output* se encarga de reflejar el valor de la bondad de ajuste en forma de porcentaje. Es obtenido gracias a la comparación de la parte de entrenamiento con la de testeo. Cuanto mayor sea este valor mejor será el modelo definido.

Figura 3.7: Segundo *output*

En la figura 3.7 se observa el apartado donde aparecerá el resultado de la bodad de ajuste del modelo, como aún no se ha introducido la base de datos aparece un error, sin embargo, en cuanto se introducen los parámetros de entrada este error desaparece.

- **Resultado de la clasificación:** el tercer y último *output* refleja una base de datos que corresponde al resultado del método Naïve Bayes, es decir, en él aparece la clasificación de cada fila según la variable respuesta, la cual se encuentra en la última columna, por lo que para ver el resultado hay que fijarse en dicha columna.

Figura 3.8: Tercer *output*

En la figura 3.8 se observa como aparece el mismo error que en el anterior *output*, esto es debido al mismo motivo, en el momento que se introduce la base de datos el resultado aparece.

3.2. Descripción del código de ejecución del clasificador Naïve Bayes

Lo primero de todo es introducir la base de datos con la que se va a trabajar, para ello, utilizamos la instrucción 'read_excel' de la librería 'readxl'. A continuación, realizamos la división de la base de datos en las partes de entrenamiento , testeo y predicción. Para ello implementaremos una función que automatice esta acción, y le daremos el nombre 'dividir_datset'. A esta función habrá que alimentarla con cuatro cosas: la base de datos, el porcentaje de filas que queremos utilizar para la fase de entrenamiento, el porcentaje que queremos para el testeo y el porcentaje para la predicción. La función 'dividir_dataset' se ejecuta automáticamente de la siguiente manera: primero, con la instrucción 'nrow()' e introduciendo la base de datos entre los paréntesis guarda el número de filas en una variable llamada 'total_filas'. A continuación, calcula la cantidad de registros para cada conjunto multiplicando cada porcentaje introducido al inicio por la variable 'total_filas' y utilizando 'round()' para redondear el resultado. De esta manera se obtienen tres variables: 'num_entrenamiento', 'num_testeo' y 'num_prediccion'. Ahora, genera indices aleatorios con la instrucción 'sample()', estos irán desde 1 hasta el valor de 'total_filas', esta variable llamada 'indices' servirá para que se elijan filas aleatorias de la base de datos y no que sean todas consecutivas. Ya casi para finalizar esta función, se realiza la división según los porcentajes introducidos de manera que se obtengan tres variables que serán 'entrenamiento', 'testeo' y predicción'. Para obtenerlas, la función hace lo siguiente: primero, obtiene la división para la variable

entrenamiento seleccionando la base de datos introducida, abriendo corchetes para seleccionar filas y/o columnas y escribiendo 'indices' que vayan desde 1 hasta el valor de 'num_entrenamiento' y para que seleccione filas y no columnas escribimos esto último antes de una coma y cerramos corchetes. La siguiente variable se obtiene de la misma manera, únicamente cambiando la escritura al seleccionar los índices, esto es, al abrir corchetes después de 'indices' hacemos que estos vayan desde 'num_entrenamiento' más 1 hasta 'num_entrenamiento' más 'num_testeo'. La variable 'predicción' se obtiene con la misma operación pero haciendo que 'indices' vaya desde 'num_entrenamiento' más 'num_testeo' más 1 hasta 'num_entrenamiento' + 'num_testeo' más 'num_prediccion'. Para acabar, se utiliza 'return()' para que devuleva una lista gracias a la instrucción 'list()' con las últimas variables calculadas.

Después de ejecutar la función 'dividir_dataset' es necesario dar valor a la parte de entrenamiento, testeo y predicción por lo que renombramos las tres variables. Para realizar esta acción, a cada una de ellas les damos valor con la función ya creada 'dividir_dataset' introcuciendo los datos y los porcentajes correspondientes a cada parte, además al final de cada línea hay que añadir un 1 para que seleccione el primer apartado de la lista que devuelve la función para el caso de entrenamiento, un 2 para el testeo y un 3 para la predicción. Estos valores deben ir entre dos corchetes. También, para la línea de predicción, es necesario añadir al final '-ncol()' con la base de datos entre los paréntesis y además todo ello entre corchetes y después de coma. Esto se hace para eliminar la variable respuesta, ya que el objetivo de la predicción es obtener los valores de esta variable.

Después de haber ejecutado estas primeras líneas de código ya podemos pasar a la explicación de las partes de entrenamiento, testeo y predicción, las cuales no son las mismas para las bases de datos con variables categóricas que con variables numéricas.

3.2.1. Para atributos categóricos

Como ya se ha mencionado anteriormente, los parámetros a estimar para el clasificador Naïve Bayes son distintos dependientdo si los atributos son categóricos o contínuos, por lo que se requieren dos códigos distintos a pesar de que siguen una estructura parecida. Se comenzará explicando las instrucciones que ejecutan el método para las variables categóricas.

Para la ejecución del método de clasificación Naïve Bayes se utilizarán tres funciones, una para la fase de entrenamiento del modelo, otra para la parte de testeo donde obtendremos la bondad de ajuste y otra para la predicción de los valores de la variable respuesta:

- **Entrenamiento:** esta primera función recibirá el nombre de 'training' y solo tendrá un parámetro de entrada, en el cual introduciremos la base de datos correspondiente al entrenamiento. Las dos primeras líneas del código se utilizan para obtener el número de filas y el número de columnas de la base de datos utilizando las insrtucciones 'nrow()' y 'ncol()' respectivamente. A continuación, se introduce en una variable llamada 'Y' la variable respuesta del modelo y posteriormente se obtienen las categorías de esta gracias a 'unique()', también se utiliza 'sort()' para ordenar de menor a mayor estos valores. Después, se calculan las probabilidades de cada clase gracias a la instrucción 'lapply()', la cual se utiliza para iterar sobre 'categorias'. Para cada categoría 'i', se aplica una función que calcula la frecuencia relativa de esta en el vector 'Y' dividiendo el número de ocurrencias de 'i' entre el número total de filas. El resultado de cada iteración se almacena en una lista. Luego, se utiliza 'unlist()' para convertir la lista resultante en un vector. Ahora, se aplicará una estructura anidada para

obtener las probabilidades a priori de las variables independientes según su categoría en la variable clase. Para ello, se utiliza lapply() en tres ocasiones de forma anidada para iterar sobre cada categoría 'k', cada variable 'i' (excepto la última columna que corresponde a la variable respuesta) y cada valor único 'j' en esa variable. La función anónima 'function(j)' calcula la probabilidad condicional de que una fila pertenezca a la categoría 'k', dado que su valor en la variable 'i' es igual a 'j'. Finalmente, se utiliza 'return()' para que 'training' devuelva una lista con las probabilidades de las categorías de la variable respuesta, las probabilidades a priori de los atributos según el valor de la variable clase y la base de datos de entrenamiento.

- **Testeo:** la segunda función implementada es la que se corresponde a la parte de testeo y recibe el nombre de 'testing'. Esta función depende de dos parámetros, uno en el cual introduciremos la base de datos correspondiente al testeo y el otro será el modelo calculado previamente gracias a 'training'. Las cuatro primeras líneas son las mismas que las de la anterior función, es decir, sirven para dar valor al número de filas y al número de columnas, también se introduce en 'Y' la variable respuesta y se obtienen las categorías de dicha variable. Posteriormente, se escriben dos nuevos objetos, 'p_categorias' y 'p_apriori' donde se introducen respectivamente las probabilidades de aparición de cada categoría y las probabilidades a priori de cada variable independiente según su categoría, ambas probabilidades se seleccionan del modelo calculado en la fase de entrenamiento. La siguiente línea de código servirá para obtener las probabilidades a posteriori, es decir, las probabilidades de pertenecer a cada categoría para cada fila de la matriz de datos. Para la implementación de esta línea se utilizó una estructura anidada que se guardará en 'p_aposteriori', en esta, aparece un 'apply()' para recorrer las 'v' filas de la base de datos de testeo, un 'lapply()' para cada 'k' categoría de la variable respuesta y un 'mapply()' para iterar simultáneamente sobre los índices 'i' y 'j' (correspondientes a las variables y a los valores de estas respectivamente) y las columnas de la fila 'v' (excepto la última columna). La función anónima 'function(i, j)' se utiliza para obtener las probabilidades condicionales correspondientes a los índices 'i' y 'j' en el modelo almacenado 'p_apriori'. Luego, se calcula el producto de todas las probabilidades condicionales utilizando 'prod()'. Después, se multiplica el producto de las probabilidades condicionales por la probabilidad a priori correspondiente a la categoría 'k'. Ya casi para finalizar esta función se asigna a cada fila el valor de la variable respuesta que le corresponde, esto se realiza fijándonos en el mayor valor de las probabilidades a posteriori obtenidas y la línea de código consiste en utilizar un 'apply()' para las filas de la matriz de datos en el que se aplique la función 'which.max' con el objetivo de asignar el mayor valor a cada registro. Esto se guardará en 'predicciones'. Finalmente, lo que nos devuelve la función 'testing' es un porcentaje que corresponde a la bondad de ajuste del modelo, esto se calcula utilizando la instrucción 'sum()' para que sume todos los valores de 'Y' que coinciden con los de 'predicciones' y se divida entre el total de filas. Todo ello multiplicado por 100 para obtenerlo en forma de porcentaje.

- **Predicción:** la tercera y última función del código llamada 'prediction' corresponde a la parte de predicción y en ella introduciremos dos parámetros, uno para la matriz de datos de la parte perteneciente a la predicción y otro que será el modelo definido en el entrenamiento. Las primeras líneas del código sirven para dar valor a 'p_categorias' con las probabilidades de las categorías, a 'p_apriori' con las probabilidades a priori y a 'entrenamiento' con la base de datos correspondiente al entrenamiento. Estas tres cosas se obtienen del modelo calculado en la fase de entrenamiento. Después, se calcula el número de filas, el número de columnas de 'entrenamiento', la variable respuesta (última columna de 'entrenamiento') y

las categorías de dicha variable respuesta. A continuación, se obtienen las probabilidades a posteriori de la misma manera que en 'testing' pero esta vez se aplicará a la base de datos de predicción. Luego, en 'predicciones' uniremos dicha base de datos con las predicciones de la variable respuesta obtenidas. Para ello, se utilizará la instrucción 'cbind()', la cual, al final de la base de datos añadirá en modo columna la variable 'p_aposteriori', a la cual se aplica un 'apply()' para las filas con la función 'which.max' para que asigne la mayor probabilidad a cada fila. Finalmente, se añade a la última columna el nombre que tenía en la base de entrenamiento y se pide a la función, gracias a 'return()', que nos devuelva la matriz de datos 'predicciones'.

Una vez implementadas las tres funciones 'training', 'testing' y 'prediction' solo queda escribir unas últimas líneas para poder ejecutar dichas funciones. La primera de estas líneas consiste en dar valor a 'modelo' con la función 'training' y la base de datos de entrenamiento, ya que 'modelo' será utilizado en la parte tanto de testeo como de predicción. La siguiente realiza la ejecución de 'testing' con la base de datos de testeo y el modelo calculado y la última línea ejecuta 'prediction' con la base de datos correspondiente a la predicción, y al igual que en el testeo se introduce también el modelo.

3.2.2. Para atributos numéricos

Una vez descrito el código utilizado para las variables independientes categóricas pasamos ahora a la explicación de otro código distinto aunque con una estructura similar, este desarrollará el método de clasificación Naïve Bayes para atributos numéricos.

Al igual que para las variables categóricas este código esta compuesto por tres funciones correspondientes al entrenamiento, testeo y predicción y como ya hemos dicho su estructura será parecida. Veamos como se ejecutan estas funciones.

- **Entrenamiento:** para esta parte del método se ha creado una función también llamada 'training' y con el mismo parámetro de entrada que la utilizada para variables categóricas. En las cinco primeras líneas se obtiene el número de filas, el número de columnas, la variable respuesta, las categorías de dicha variable respuesta y las probabilidades de aparición de estas categorías. A continuación, se procede al calculo de la media y la desviación estándar de cada columna (excepto la última) para cada categoría de la variable clase, esto se guarda en 'mean_sd' y para llevar a cabo este proceso se ha utilizado una estructura anidada que comienza con un 'lapply()' que recorre las 'k' categorías al que se une otro 'lapply()' para las 'i' variables (menos la última). Para guardar los dos valores (media y desviación estándar) se utiliza 'c()', donde primero aparece la media obtenida gracias a 'mean()' y después, se encuentra separada por una coma la desviación estándar corregida calculada con la raíz cuadrada (instrucción 'sqrt()') de la suma del número de valores para la categoría 'k' menos 1 entre la suma total de estos valores, todo ello multiplicado por la desviación estándar de las filas con clase 'k' para la columna correspondiente. Finalmente, esta función devuelve una lista en la que aparece primero las probabilidades de cada categoría, segundo 'mean_sd' y por último la matriz de datos de entrenamiento.

- **Testeo:** esta función es casi idéntica a la utilizada para el testeo de las variables categóricas, recibe el mismo nombre y tiene los mismos parámetros de entrada. Únicamente cambian dos cosas, la primera es que no se seleccionan las probabilidades a priori si no que se utiliza la media y la desviación estándar de cada variable independiente, para ello se renombra

'mean_sd' y se obtienen estos valores del modelo calculado en el entrenamiendo. La segunda cosa que cambia es la obtención de las probabilidades a posteriori, en este caso también se utiliza una estructuta anidada que comienza con un 'apply()' para recorrer las 'v' filas, le sige un 'lapply()' para las 'k' categorías y continúa otro 'lapply()' que itera sobre todas las 'i' columnas menos la última. La función anónima aplicada consiste en la instrucción 'dnorm()' para obtener la densidad de probabilidad normal para el valor en la fila 'v' y columa 'i' utilizando la media y desvianción estándar correspondientes a la categoría 'k' y columna 'i'. El último 'lapply()' está dentro de 'prod()' para obtener el producto de todas las densidades de probabilidad normales obtenidas para cada columna 'i'. Finalmente, se multiplica el producto de las densidades de probabilidad normales por la probabilidad a priori de la categoría 'k'. De esta manera se calculan las probabilidades de cada fila de pertenecer a una clase u otra. Después, se asigna a cada registro el mayor valor obtenido, es decir, a cada fila se le asigna la clase con mayor probabilidad, esto se realiza aplicando un 'apply()' por filas e incluyendo la intrucción 'which.max'. Para da por finalizada la función 'testing' se pide con 'return' que devuelva el porcentaje correspondiente a la bondad de ajuste, al igual que se hace para las variables de tipo categórico.

- **Predicción:** esta función al igual que las anteriores tiene el mismo nombre y los mismos parámetros de entrada que la función asignada a la predicción de variables categóricas, estos parámetros son los datos correspondientes a la predicción y el modelo calculado en la parte de entrenamiento. La función comienza dando valor a los tres cálculos obtenidos en 'modelo', estos son: la probabilidad de las categorías, la media y desviación estándar de las variables independientes y la matriz de datos de entrenamiento. A continuación, al igual que las otras funciones se obtiene el número de filas, el número de columnas, la variable respuesta y las categorías de dicha variable. Ahora, se asigna a 'p_aposteriori' las probabilidades a posteriori de exactamente la misma forma que se hizo en el testeo. Luego, se calculan las predicciones uniendo, gracias a 'cbind()', los datos correspondientes a este apartado con 'p_aposteriori', a la cual se le aplica un 'apply()' para recorrer las filas y asignar el mayor valor gracias a 'which.max()'. Finalmente, se cambia el nombre de la última fila para asignarle el de la variable respuesta y se pide a la función que devuelva 'predicciones' gracias a 'return()'.

Para acabar, se realiza la misma operación que con el código de variables categóricas, es decir, se da valor a 'modelo' ejecutando 'training', se ejecuta 'testing' y se asigna a 'prediccion' la ejecución de 'prediction'.

3.3. Descripción del código de ejecución de la aplicación web

El usuario cuando utilice la aplicación web únicamente verá la interfaz con los *inputs* y los *outputs*, sin embargo, detrás hay un código para el correcto funcionamiento de la app y también para organizar la estructura de la interfaz.

Para implementar la app se utiliza el paquete 'shiny', por lo que lo primero que hay que hacer es usar 'library()' introduciendo dicho paquete. Los códigos para la creación de la app web en el programa *R* están formados por dos partes, 'ui' y 'server', en la primera se describe la estructura visible por el usuario: el título, la descripción, los *inputs* y los *outputs*. En la parte de *server* se implementa el funcionamiento de los *outputs*, es decir, gracias a los parámetros de entrada y a las funciones del método de clasificación se detallan los cálculos necesarios para que los resultados sean visibles para el usuario.

- **Ui:** en 'ui' se utilizará 'fluidPage()' ya que es una de las funciones clave en la librería 'shiny', se utiliza para construir la interfaz de usuario de la aplicación. Proporciona una estructura básica para organizar y colocar los elementos visuales. En esta función incluiremos todas las instrucciones necesarias para hacer dicha estructura. La primera instrucción es 'titlePanel()', la cual nos permite escribir un título para la app. Después, se escribe una breve descripción de lo que realiza la aplicación, para ello utilizamos dos instrucciones, la primera es 'tags$style()' y sirve para dar nombre a este apartado además de dar forma al texto y la segunda es 'h2()' donde hay que añadir el nombre del apartado escrito antes y después el texto de la descripción. A continuación, se introducen los *inputs*, comenzando con 'fileInput()' para introducir la base de datos, en él hay que dar nombre a este parámetro, luego escribir un texto que verá el usuario y por último el tipo de archivos que aceptará la aplicación. El siguiente *input* es 'radioButtons()', el cual permite seleccionar una entre varias opciones, al igual que en el anterior parámetro hay que escribir su nombre y el texto para el usuario y además, las opciones entre las que tiene que elegir. Luego, aparecen dos 'numericInput()' para introducir los porcentajes de entrenamiento y testeo, en estos también hay que dar un nombre y redactar un texto, incluyendo al final que el valor que queremos que aparezca al iniciar la aplicación sea nulo y que tenga que ser el usuario el que introduzca dicho valor. Pasamos ahora a la descripción de los *outputs* en 'ui', lo cual es algo sencillo. Los dos primeros son muy similares, se introducen con 'verbatimTextOutput()' y lo único que hay que escribir dentro de esta instrucción es el nombre que se les da, estos parámetros devuelven tanto el porcentaje para la parte de predicción como la bondad de ajuste. El último parámetro de salida es 'tableOutput()' y también, únicamente hay que introducir su nombre, este *output* devolverá la base de datos con la clasificación de las filas de la base de predicción según la variable respuesta.

- **Server:** este apartado se utiliza para implementar la ejecución de los *outputs* y se corresponde con una función en la que aparecen dos parámetros de entrada, dichos *outputs* y los *inputs*. Para implementarlos es necesario guardarlos en una variable que sea *output*, seguido del símbolo del dólar y el nombre que se le dió en el apratado 'ui'. Entonces, el primero sería 'output$porcentaje_prediccion' y su ejecución se introduce dentro de 'renderPrint()' que es la función correspondiente a 'verbatimTextOutput()'. Dentro de esta función se utiliza un 'paste()' para unir texto con la operación que resuelve uno (representa el porcentaje total) menos el porcentaje de entrenamiento y menos el porcentaje de testeo. El siguiente *output* es la bondad de ajuste del modelo y al igual que con el anterior se utiliza 'renderPrint()'. Ahora, introducimos la parte común de las funciones del método Naïve Bayes y después utilizamos un 'if()' para diferenciar si los atributos son numéricos o categóricos. Para que el programa sepa si son de un tipo o de otro se utiliza el *input* donde se selecciona el tipo de datos, por lo que dentro de los paréntesis es necesario escribir *input* seguido de '$' y el nombre de dicho parámetro de entrada, dándole el valor que se desea. En el 'if()' donde son numéricos, se introduce la función para estos atributos y lo mismo se hace para los categóricos. Para finalizar la implementación de este *output*, se introduce dentro de 'modelo' la ejecución de 'training' y se utiliza 'paste()' para juntar texto de lectura para el usuario con la ejecución de 'testing'. El último *output* representa la clasificación de los datos de predicción según la variable respuesta, por lo que tiene forma de tabla. En este caso se usa 'renderTable()' que es la función correspondiente a 'tableOutput()' y su implementación es prácticamente idéntica a la del anterior parámetro de salida, lo único que cambia es la línea final donde se sustituye 'paste()' por la ejecución de 'prediction'.

Para dar por finalizado este código, es necesario utilizar 'shinyApp()', en la cual se le introduce 'ui' y 'server' para que la aplicación se ejecute correctamente.

Capítulo 4

Resultados

Para observar el uso y la efectividad de la aplicación creada se va a trabajar a continuación con las bases de datos explicadas previamente. Se realizarán dos análisis, uno sobre el cáncer de pulmón, en el que se encuentran atributos categóricos y otro sobre el cáncer de mama, en cuya base de datos aparecen variables numéricas.

4.1. Uso de la aplicación para atributos categóricos

En este estudio se va a predecir si individuos con ciertas características padecen cáncer de pulmón o no. En la base de datos aparecen 309 individuos, sin embargo, no se va a predecir si todos ellos tienen cáncer, si no que como ya se ha explicado, un porcentaje de ellos será destinado al entrenamiento, otro al testeo y las personas de estudio restantes serán las que sí se predigan si padecen cáncer de pulmón o no, estos porcentajes serán 33 %, 33 % y 34 % respectivamente. Para ello utilizaremos la aplicación web creada previamente en la que introduciremos los parámetros necesarios.

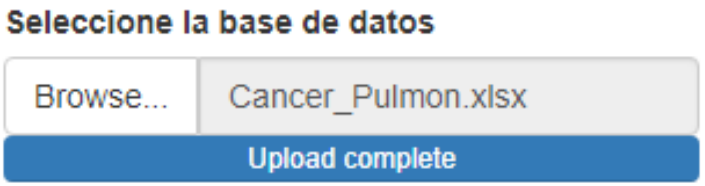

Figura 4.1: Introducción de la base de datos

En la figura 4.1 se observa el primer *input* introducido, el cual se corresponde con la base de datos que ya ha sido introducida, esta es de tipo .xlsx, lo cual era un requisito necesario. En la parte de abajo aparece en color azul *Upload complete*, esto quiere decir que se ha completado la entrada de los datos con los que se va a trabajar.

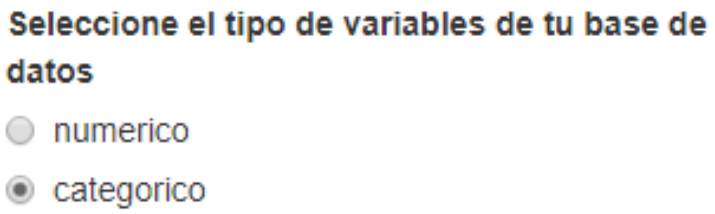

Figura 4.2: Selección del tipo de variable

El siguiente *input* que introducimos es la selección *categorico*, refiriéndonos al tipo de variables independientes que tenemos en la base de datos. De entrada, la aplicación tiene marcado *numerico*, por lo que es necesario cambiar la selección. Ahora, introduciremos los porcentajes corresponidnetes al entrenamiento y testeo.

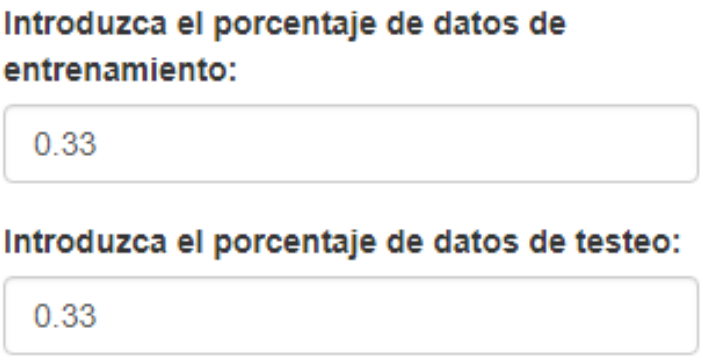

Figura 4.3: Introducción de los porcentajes

Como se ve en la figura 4.3 los porcentajes hay que introducirlos con valores de 0 a 1 ya que si no la app no sabe interpretarlos. A continuación, la aplicación nos devolverá el porcentaje faltante, este es el de la parte de predicción.

```
"porcentaje de datos de prediccion:  0.34"
```

Figura 4.4: Porcentaje de predicción

Como aparece en la figra 4.4, el porcentaje correspondiente a la predicción es del 34 %, esto es obtenido de la resta de uno menos los porcentajes ya introducidos. Entonces, el 34 % de los individuos de la base de datos será destinado a la predicción del cáncer. Ahora, veremos el pocentaje de bondad de ajuste del modelo, lo ideal es que sea un valor alto.

```
"Bondad del ajuste:  87.2549019607843 %"
```

Figura 4.5: Bondad de ajuste

En la figura 4.5 aparece el valor buscado, este alcanza aproximadamente un 87 %, lo cual nos dice que el modelo es bueno ya que es una bondad de ajuste bastante alta. Por último, pasamos

a la clasificación de los pacientes a los que se les asignará un 1 o un 2 en la variable respuesta 'lung_cancer'. El 1 representa que el paciente no padece el cáncer y el 2 indica que sí lo padece.

LUNG_CANCER
2
1
1
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
1

Figura 4.6: Variable respuesta

En la figura 4.6 se observan unas pocas de las predicciones hechas en el estudio, para encontrar la variable respuesta ha sido necesario ir hasta la última columna de la tabla de resultados.

4.2. Uso de la aplicación para atributos numéricos

En este caso se realizará la predicción para la base de datos con atributos numéricos, en esta aparecen mujeres con características sobre tumores mamarios y su variable respuesta consiste en determinar si el tumor es benigno o maligno. Esta base está compuesta por 569 individuos y los porcentajes que corresponderán a cada parte serán del 33 % para entrenamiento, 33 % para testeo y 34 % para la predicción.

Como se ha hecho en el anterior apartado se comenzará explicando los parámetros de entrada necesarios para el estudio.

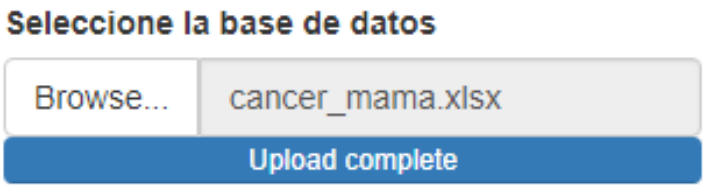

Figura 4.7: Introducción de la base de datos

En la figura 4.7 se puede observar como la introducción de los datos es igual que en el apartado anterior. También se ve, como es necesario, que el archivo introducido es de tipo .xlsx. Ahora, pasamos a la selección del tipo de variable.

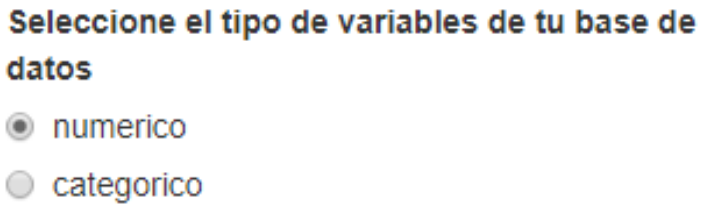

Figura 4.8: Selección del tipo de variable

En este caso no ha sido necesario cambiar nada ya que por defecto, la aplicación marca la categoría *numerico*. A continuación, pasamos a la introducción de los porcentajes correspondientes al entrenamiento y al testeo, esto se hace exactamente igual que en el anterior apartado.

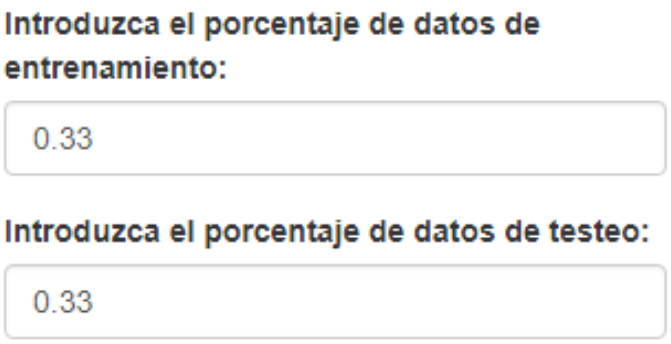

Figura 4.9: Introducción de los porcentajes

Una vez introducidos todos los parámetros de entrada necesarios, veamos los resultados obtenidos.

```
"porcentaje de datos de prediccion:  0.34"
```

Figura 4.10: Porcentaje de predicción

Lo primero que nos devuelve el programa es el porcentaje de datos correspondiente a la parte de predicción. Este resultado es del 34 %. Ahora, veamos la bondad de ajuste.

```
"Bondad del ajuste:  93.0851063829787 %"
```

Figura 4.11: Bondad de ajuste

Como se ve en la figura 4.11 el porcentaje alcanza un 93 % por lo que se considera un muy buen modelo,incluso algo mejor que el modelo para las variables categóricas. Por último, observaremos los resultados de la clasificación de las mujeres según padezcan de tumor maligno o benigno. Si dicho tumor es maligno se asigna un 2 y en caso de que sea benigno un 1.

diagnosis
1
1
1
2
1
1
2
1
1
2
2
2
2
2
1
1
2
1
2
1

Figura 4.12: Variable respuesta

Al igual que en la figura 4.6, en la 4.12 solo aparecen unas pocas predicciones de todas las visibles.

Conclusiones

Tras la explicación y el uso de la clasificación bayesiana se han llegado a las siguientes conclusiones:

- La clasificación bayesiana resulta ser un método efectivo a pesar del supuesto de independencia comentado anteriormente. Además, su implementación es sencilla de realizar.
- Como se ha demostrado en el apartado de resultados, la aplicación web funciona y clasifica los datos reales de forma correcta, de manera que asigna a cada fila una categoría dependiendo de ciertas características. También, tiene un uso muy sencillo e intuitivo a la hora de introducir los parámetros de entrada necesarios en el modelo.
- Otra de las conclusiones importantes es que la aplicación es efectiva tanto para atributos categóricos como para atributos numéricos. Esto se vio en el apartado de resultados, donde aparecía en ambos estudios una bondad de ajuste muy alta.
- En cuanto al uso de la clasificación bayesiana en el ámbito de la medicina vimos que se puede utilizar para predecir posibles casos de cáncer o de tumores malignos. Esto es algo que puede llegar a ser determinante y ayudar a muchas personas. Además, no solo se tiene que limitar a enfermedades cancerígenas, sino que puede ayudar en muchos otros ámbitos médicos.

Bibliografía

Agrawal, R., & Ramakrishnan, R. (1994). Fast algorithms for mining association rules. *Proc. 20th int. conf. very large data bases, VLDB*, *1215*, 487-499.

Bayes, T. (1763). LII. An essay towards solving a problem in the doctrine of chances. *Philosophical transactions of the Royal Society of London*, (53), 370-418. The Royal Society London. https://doi.org/https://doi.org/10.1098/rstl.1763.0053

Chang, W. (2020). *shiny: Web Application Framework for R* [Accessed: 05-07-2023]. https://cran.r-project.org/web/packages/shiny/index.html

Hernández-Orallo, J., Ramíres, M. J., & Ferri, C. (2004). *Introducción a la Minería de Datos*. Pearson.

Kaggle. (2023a). *Breast cancer prediction* [Accessed: 05-07-2023]. https://www.kaggle.com/code/buddhiniw/breast-cancer-prediction

Kaggle. (2023b). *Lung Cancer* [Accessed: 05-07-2023]. https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer

Peñalva-Martínez, M. C., & Posadas, J. A. (2009). El planteamiento de problemas y la construcción del teorema de Bayes. *Enseñanza de las Ciencias. Revista de investigación y experiencias didácticas*, *27(3)*, 331-342. https://doi.org/https://doi.org/10.5565/rev/ensciencias.3645

Sucar, L. E., & Tonantzintla, M. (2006). Redes bayesianas. *Aprendizaje Automático: conceptos básicos y avanzados*, *77*, 100.

Ye, N. (2013). *Data mining: theories, algorithms, and examples*. CRC press.