UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE CIENCIA
Departamento de Matemática y Ciencia de la Computación

Clasificación de Streaming Data usando Naïve Bayes.

Carolina Alejandra Olmos Sáez

Profesor guía: Felipe Elorrieta López

Trabajo de titulación presentado a la Facultad de Ciencia en cumplimiento parcial de los requisitos exigidos para optar al título de Ingeniera Estadística.

Santiago - Chile
2019

© 2019, Carolina Alejandra Olmos Sáez
Se autoriza la reproducción parcial o total de esta obra, con fines académicos, por cualquier forma, medio o procedimiento, siempre y cuando se incluya la cita bibliográfica del documento.

Resumen

Frente a la rápida y continua evolución que ha tenido la tecnología en los últimos años, se están recopilando datos en volúmenes cada vez más grandes. Las bases de datos que recopilan información constantemente se denominan data streams y requieren de técnicas apropiadas de minería de datos y aprendizaje automático para su análisis. En ese sentido, Naïve Bayes es un método de clasificación bayesiano que asume que las variables explicativas son independientes dentro de cada clase. Este método se adapta bastante bien al contexto de un data stream, pues actualizar el modelo no es complejo y existen resultados experimentales que demuestran altos niveles de rendimiento (P. E. Lutu, 2013).

A raíz de lo anterior, el objetivo principal de este estudio es implementar el algoritmo de clasificación Naïve Bayes en línea. Para esto, se calcularon las reglas de clasificación, considerando variable respuesta binaria y atributos con distribución normal o bernoulli. Posteriormente, se calularon las reglas para actualizar los parámetros cuando llega un nuevo stream de datos. Finalmente, se crearon funciones en RStudio para procesamiento en batch y stream del algoritmo de Naive Bayes.

Los clasificadores fueron implementados en conjuntos de datos simulados con variables normales y variables bernoulli. Además, se aplicó el algoritmo en un problema de clasificación astronómico, donde se pretende detectar si las observaciones corresponden a un tipo de estrella denominado RR Lyrae; y datos extraídos de la red social Twitter, con el que se llevó a cabo un análisis de sentimientos, mediante la clasificación de los tweets (como positivo o negativo) a partir de las palabras que incluyen.

De acuerdo a los resultados obtenidos, se observó que los parámetros del método Naïve Bayes en un data stream convergen a los parámetros del método en batch, pero en un menor tiempo de ejecución. Las funciones creadas fueron comparadas con el comando *naiveBayes* de la librería e1071 del software RStudio, observándose que las funciones presentaron en general un mejor rendimiento de clasificación y un menor tiempo de ejecución.

Palabras clave: *clasificación, naïve bayes, online learning, data stream, bernoulli, normal, text mining, análisis de sentimientos, astronomía*

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Motivación

Desde las últimas décadas se ha estado viviendo un momento de explosión en la forma que se recopila la información, a causa de la rápida y continua evolución de la tecnología, así como de los dispositivos y herramientas computacionales, que han permitido el almacenamiento de datos en grandes cantidades.

Las bases de datos que recopilan constantemente estos gigantescos volúmenes de información, se denominan *flujos de datos* o *data streams*. Estas almacenan información de manera continua, pues los datos llegan incesantemente, sin seguir ningún orden ni patrón particular (Y. Xu, K. Wang, A. W.-C. Fu, R. Shee & J. Pei, 2006).

Los data streams o flujos de datos poseen ciertas características o particularidades: tienen un gran volumen, son de tamaño ilimitado y son dinámicos, es decir, cambian constantemente. De hecho, los datos pueden estar sujetos a cambios distribucionales en el tiempo (Frías Blanco, 2012).

Un ejemplo de la gran cantidad de datos que son recopilados a diario podrían ser fuentes de datos científicos, de marketing, financieros, demográficos, entre otros; los cuales provienen de servidores web, internet, telefonía móvil, etcétera (Y. Xu et al., 2006).

Bajo este contexto, la extracción de conocimiento útil para colaborar con la toma de decisiones en diversas áreas de la vida real se ha vuelto un proceso cada vez más desafiante, pues realizarlo de manera manual es muy costoso en términos de tiempo. Por lo tanto, se requiere el uso de técnicas apropiadas de aprendizaje automático (machine learning) y minería de datos (data mining).

Dentro de estas técnicas, la clasificación es una de las que más éxito ha tenido al ser aplicada en diversos contextos. H. Borchani (2012) plantea que, en general, la clasificación se compone de dos pasos importantes: en primer lugar, aprender un modelo de clasificación o un clasificador a partir

de un conjunto de datos de entrenamiento; y en segundo lugar, clasificar las nuevas instancias de datos manipulando el clasificador aprendido.

Las redes bayesianas, en conjunto con los árboles de decisión y las redes neuronales artificiales, están entre los métodos de clasificación más utilizados. Algunos contextos cotidianos en que suelen ser utilizadas estas técnicas son la clasificación de documentos o filtros de mensajes de correos electrónicos, que muchas veces no son deseados, ofrecen dudosas oportunidades financieras o están relacionados a marketing de algún tipo. Para esto, se han desarrollado reglas de clasificación llamadas *filtros de spam*, que examinan los múltiples correos electrónicos que reciben los individuos a diario, y los asignan a clase "spam" o "no spam", según corresponda. Los modelos Naïve Bayes son muy populares para clasificar y filtrar correos spam, remontándose a los primeros trabajos aportados por Sahami en 1998 (X. Wu et al., 2007).

Al implementar técnicas de minería de datos, como Naïve Bayes, en contextos que incorporan información continuamente (data streaming), se forma una retroalimentación del mismo clasificador (K. M. Adam Chai, H. T. Ng & H. L. Chieu, 2002). Por consiguiente, si en cierto momento se recoge una clasificación con un determinado conjunto de datos de entrenamiento, al incorporar más datos, se debe obtener una nueva clasificación. Entonces, la información obtenida puede ser utilizada para los datos que se adquieren posteriormente a la base de datos inicial, generando lo que se conoce como "aprendizaje continuo" (K. M. Adam Chai et al., 2002).

En definitiva, la extracción de conocimiento a partir de grandes flujos de datos es una de las tareas que proponen un mayor desafío, sin embargo, minería de datos y aprendizaje automático ayudan a resolver este problema.

Tomando en cuenta el contexto en el que está inmersa esta tesis, I. Frías Blanco (2014) plantea restricciones que se deben tener en consideración: la cantidad de datos a procesar es potencialmente infinita; la velocidad de llegada de los datos es alta, por lo tanto, deben ser procesados en tiempo real o en línea; y las características estadísticas de los datos podría variar en el tiempo (por ejemplo, si se presenta algún cambio debido a algún evento meteorológico, los modelos de clasificación deben ser capaces de adaptarse a los cambios).

1.2. Planteamiento del problema

Hoy en día, se está viviendo una época de revolución de los datos. Cada minuto se generan más y más, recopilando grandes volúmenes de información. Transformar los datos en información útil para la toma de decisiones es una necesidad en diversos contextos de la vida real, por ejemplo: salud, marketing, ciencias, redes sociales, entre otras (R. Rivera et al., 2017).

Lo anterior tiene una gran dificultad si se considera la alta velocidad con la que se generan los datos. En relación a esto, R. Rivera et al. (2017) señalan que ya no es suficiente con ser capaces de procesar grandes cantidades de datos, sino que además deben ser procesados rápidamente o en "tiempo real". En ese sentido, surge la necesidad de transformar y analizar los datos de manera óptima considerando su gran volumen y velocidad.

Existen dos principales perspectivas para analizar la información: procesamiento en Batch (Batch processing) y procesamiento en Stream (Streaming processing). El primero, también llamado procesamiento en lotes, consiste en analizar grandes volúmenes de datos en un mismo instante de tiempo. Por otra parte, el procesamiento en Stream, consiste en analizar los datos en tiempo real o en línea. En lugar de procesar grandes cantidades de datos, el procesamiento en stream analiza pequeñas cantidades de manera continua, es decir, en todo momento (A. Rayón, 2016).

Tomando en cuenta lo anterior, la idea de este proyecto es aplicar el algoritmo de clasificación Naïve Bayes para datos en línea o streaming data.

X. Wu et al. (2007) señalan que Naïve Bayes es un método muy importante, puesto que, al no necesitar esquemas complicados para la estimación de parámetros, su construcción no presenta mayores dificultades. Además, permite que se pueda aplicar a grandes conjuntos de datos.

La idea principal del método Naïve Bayes consiste en utilizar un conjunto inicial de objetos (los que ya pertenecen a una clasificación conocida). Este set inicial se denomina "datos de entrenamiento" y con ellos se debe construir una regla de clasificación que permita asignar valores futuros a cierta clase, dado el vector de variables explicativas (X. Wu et al., 2007). Este método asume que los componentes de las variables explicativas son independientes dentro de cada clase, de modo que transforma un problema multivariante p-dimensional a p problemas de estimación univariados. Esto simplifica bastante el problema y hace que el conjunto de datos de entrenamiento que se requiere para obtener las estimaciones sea más pequeño.

En el software estadístico R, existe una librería que incluye el algoritmo de Naïve Bayes (paquete e1071). El comando naiveBayes calcula las probabilidades a posterioris condicionales de la variable respuesta (categórica), en función de las variables predictoras independientes, usando la regla de Bayes (D. Meyer et al., 2017). No obstante, esta función tiene dos limitaciones: asume normalidad en las variables explicativas y procesa los datos en lotes (procesamiento en batch).

Dado que este proyecto tiene por finalidad aplicar al método Naïve Bayes usando procesamiento en streaming, las reglas de clasificación deberán ser calculadas y posteriormente programadas en RStudio.

Tomando en cuenta lo expuesto anteriormente, el método de clasificación de Naïve Bayes debe ser implementado considerando que la variable respuesta será del tipo Bernoulli y las variables explicativas podrán asumir un comportamiento Normal o Bernoulli.

1.3. Objetivos

1.3.1. Objetivo general

Implementar el método Naïve Bayes para clasificar un data stream.

1.3.2. Objetivos Específicos

1. Realizar revisión bibliográfica respecto al método Naïve Bayes y su implementación en Streaming data.
2. Calcular las reglas de clasificación de Naïve Bayes binario para variables explicativas con distribución Normal y Bernoulli.
3. Creas funciones de las reglas de clasificación Naïve Bayes mediante el software RStudio.
4. Implementar funciones en datos simulados, datos astronómicos y datos extraídos de Twitter.
5. Evaluar el rendimiento del clasificador, comparando el método en línea frente al método en lote.

1.4. Metodología

En primera instancia, se llevó a cabo la revisión de la literatura pertinente para la investigación. Con el conocimiento respecto al tema, se realizó una búsqueda de documentos (libros, revistas, publicaciones, entre otros) que estuviesen relacionados con el tema de investigación. Este paso fue de vital importancia para el desarrollo del proyecto de manera óptima.

Por tanto, con mayor conocimiento respecto del tema, se calcularon las reglas de clasificación del método Naïve Bayes. En este caso, el problema de clasificación es binario, por lo tanto, la variable respuesta es del tipo Bernoulli y las variables explicativas pueden seguir distribución Normal o Bernoulli. Luego, a partir de las reglas de clasificación se crearon funciones en RStudio para procesamiento en batch y procesamiento en stream. Estas funciones fueron implementadas en distintos conjuntos de datos.

Para el caso de variables explicativas normales, en primera instancia se simuló una base de datos y se aplicaron los clasificadores. Luego, se utilizó una base de datos astronómica en un problema de clasificación binario, donde se pretende detectar cierta clase de estrella denominada "RR Lyrae". Las variables independientes de esta base de datos fueron previamente extraídas del comportamiento temporal de las estrellas, entre las cuales se encuentra el periodo, la amplitud y la fase.

En lo que respecta al caso de variables explicativas bernoulli, se comenzó por simular una base de datos a la que posteriormente se le aplicaron los clasificadores. Asimismo, se utilizó un conjunto de datos extraído de la red social Twitter, esto con la finalidad de realizar un análisis de sentimientos mediante la clasificación de los tweets (como sentimiento positivo o sentimiento negativo). Previo a la clasificación se llevó a cabo una transformación de la base de datos, donde se realizó una selección de las palabras más frecuentes en los tweets y cada término seleccionado corresponde a una variable explicativa bernoulli.

Finalmente, para cada conjunto de datos se evaluó el rendimiento de los clasificadores. Además, en esta etapa del proyecto se compararon las funciones creadas para procesamiento en lotes y procesamiento en línea, frente al algoritmo Naïve Bayes que incorpora RStudio (el cual está basado en procesamiento en lotes). Para esto, se consideró que los parámetros con el método en línea deberían converger a los del método en batch, pero en menor tiempo de ejecución de las funciones.

En la Figura 1.1 se presenta un diagrama que resume la metodología recién explicada.

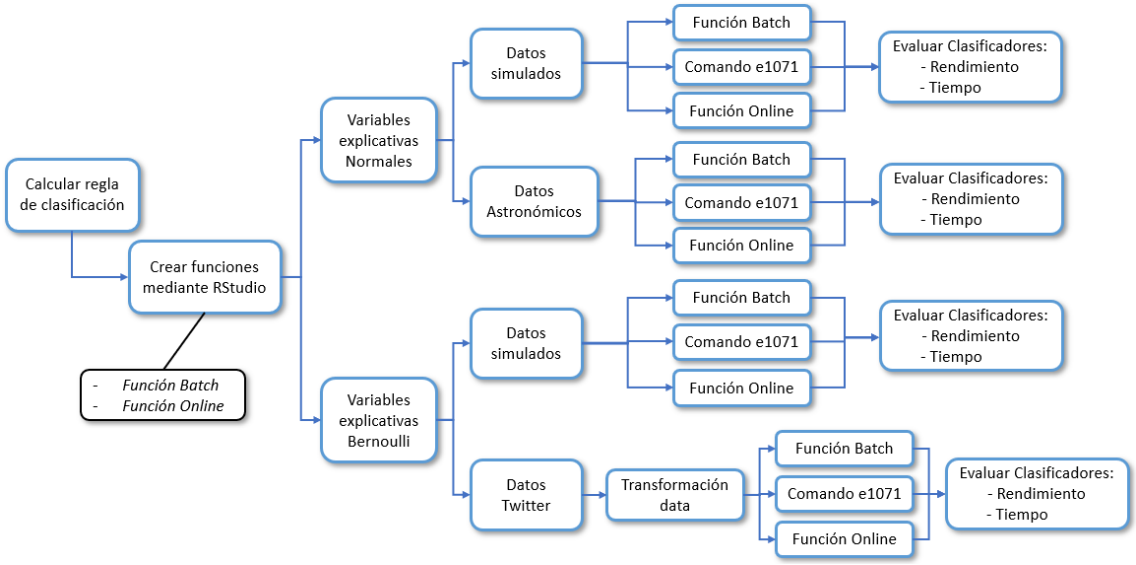


Figura 1.1: Diagrama de flujo de la Metodología.

Capítulo 2

Marco Teórico

2.1. Clasificación

Clasificación es una técnica utilizada en minería de datos y aprendizaje automático, la cual consiste en describir clases o categorías a las que pertenecen los datos. Se generan modelos que predicen etiquetas de clase comenzando con un conjunto de observaciones inicial, llamada base de datos de entrenamiento. De esta forma, con la base de entrenamiento el método de clasificación genera un conjunto de reglas que establecen a qué clase pertenecen las observaciones futuras (C. Versellis, 2009).

La clasificación es un proceso de dos pasos. El primero corresponde al paso de aprendizaje, donde se construye el modelo de clasificación; y la segunda fase es de clasificación, donde el modelo creado anteriormente se utiliza para predecir las etiquetas de clase de nuevos datos u observaciones (J. Han, M. Kamber & J. Pei, 2012).

Las técnicas de clasificación son muy útiles en numerosos campos de la vida real y son muy potentes en la teoría del aprendizaje debido a sus implicancias teóricas. Así, C. Versellis (2009) plantea que desde el punto de vista teórico, el desarrollo de algoritmos de clasificación que sean capaces de aprender de experiencias anteriores es fundamental para emular la capacidad de inducción del cerebro humano; y desde el punto de vista práctico, son numerosas las áreas o aplicaciones que abarca la clasificación: modelos de clasificación de préstamos bancarios, clasificadores de mercado objetivo, modelos de detección de fraudes, modelos de detección de desempeño, clasificadores diseñados para reconocimiento de imágenes, catalogación de textos, reconocimiento de correos electrónicos como no deseados o spam, diagnósticos médicos, entre muchos otros.

Se pueden distinguir dos tipos de clasificación: Clasificación supervisada y Clasificación no supervisada. En la primera, existe conocimiento a priori de los datos, es decir, se conoce la etiqueta de clase o categoría a la que pertenecen los objetos en estudio; mientras que en la clasificación no

supervisada no existe conocimiento a priori, esto significa que se cuenta con objetos u observaciones cuya etiqueta de clase o categoría es desconocida. En estos casos, los objetos se agrupan mediante sus similitudes, razón por la cual la clasificación no supervisada suele conocerse como Clustering (J. Han et al., 2012).

En particular, dentro de las técnicas de clasificación supervisada se encuentran los Clasificadores Bayesianos, que predicen la probabilidad de que una determinada tupla pertenezca a una categoría en particular (C. Vercellis, 2009).

2.2. Teorema de Bayes

El Teorema de Bayes, propuesto por Thomas Bayes (1702-1761), representa la probabilidad de un evento A, dado cierto evento B que condiciona a A y que se conoce como "evidencia". Este teorema plantea un vínculo entre la probabilidad de A dado B y la probabilidad de B dado A.

Para definirlo, suponga que $A = \{A_1, ..., A_n\}$ es un conjunto de sucesos mutuamente excluyentes y sea B otro evento, donde se conoce $P(B|A_i)$ para cada $i = 1, \ldots, n$.

El Teorema de Bayes viene dado por:

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

donde $P(A_i)$ se conoce como la probabilidad *a priori* de A_i, es decir, sin tomar en cuenta ningún tipo de información adicional; $P(B|A_i)$ es la probabilidad de B condicionada en A_i, $P(B)$ es la probabilidad a priori de que suceda el evento B, sin tomar en cuenta información adicional y $P(A_i|B)$ es la probabilidad *a posteriori* de A_i dado el evento B.

$P(B|A_i)$, $P(A_i)$ y $P(B)$ se pueden estimar a partir de los datos muestrales, lo que permite realizar el cálculo de la probabilidad a posteriori $P(A_i|B)$ para cada valor de $i = 1, \ldots, n$.

2.3. Naïve Bayes

Naïve Bayes, también conocido como Idiot's Bayes o Independence Bayes, es un modelo de clasificación supervisada probabilístico que está basado en el Teorema de Bayes. Los clasificadores Naïve Bayes asumen "ingenuamente" que las variables explicativas son condicionalmente independientes en relación con la clase objetivo (C. Vercellis, 2009). El modelo ha alcanzado gran popularidad en la clasificación de textos y en el último tiempo se ha convertido en una solución tradicional a los problemas de detección de spam.

Este método utiliza un conjunto de datos inicial o de entrenamiento con categoría ya conocida y el objetivo principal es construir una *regla de clasificación* que permita que futuros objetos sean asignados a alguna clase, dado un vector de variables explicativas que pueden ser de diferente tipo, categóricas y numéricas discretas o continuas (X. Wu et al., 2007). En ese sentido D. J. Hand & K. Yu (2001) explican que existen dos paradigmas a través de los cuales se pueden diseñar las reglas de clasificación, denominados *Paradigma de Muestreo* y *Paradigma de Diagnóstico.* El paradigma de diagnóstico enfoca su atención en las diferencias entre clases (discriminando entre clases), mientras que el paradigma de muestreo enfoca su atención en las distribuciones individuales de las clases. El método Naïve Bayes puede ser abordado desde ambas perspectivas.

X. Wu et al. (2007) explican que el método Naïve Bayes ha tomado bastante importancia en los últimos años debido a que es fácil de construir, no necesita tediosos esquemas iterativos para la estimación de parámetros, se puede aplicar a grandes conjuntos de datos y es fácil de interpretar. De hecho, Domingos y Pazzani (1997) ejemplifican un conjunto de investigaciones, entre las cuales se presenta el trabajo de Kononenko (1990), donde reportó que la representación del clasificador bayesiano Naïve Bayes es bastante intuitiva y fácil de entender por al menos una clase de usuarios (médicos), algo que a menudo es una preocupación significativa en el aprendizaje automático.

La suposición de independencia de los atributos dentro de cada clase para el método Naïve Bayes es un aspecto relevante, puesto que bajo cualquier lógica podría considerarse restrictivo tomando en cuenta que en la mayoría de los problemas que se presentan en la vida real no hay independencia. En ese sentido, se podría esperar que el método no funcione bien y esto ha llevado a un rechazo general del modelo por parte de algunos investigadores teóricos, quiénes toman una postura a favor de alternativas más complicadas, según explican Hand & Yu (2001). No obstante, de manera contraria a lo que se podría pensar, el método lo hace realmente bien. Incluso cuando su desempeño no es el óptimo, el clasificador Naïve Bayes puede funcionar mejor que otros clasificadores con mayor poder representativo (P. Domingos & N. Pazzani, 1997).

Hand & Yu (2001) explican que existen varios factores que también entran en juego y hacen que la suposición de independencia no sea tan perjudicial como podría parecer.

Una primera razón importante es que el modelo naïve Bayes requiere que se estimen menos

parámetros en comparación con otros métodos que buscan modelar interacciones entre las distribuciones de las variables condicionadas a cada clase. Esto es relevante puesto que modelos basados en menor cantidad de parámetros tienden a presentar una menor varianza para las estimaciones de $P(I_i|X)$. En ese sentido, la muestra disponible conducirá a un estimador con una varianza menor si se asume la independencia de las variables dentro de las clases.

Ahora bien, si la suposición no es cierta, existe un riesgo de sesgo. Por ejemplo, en el caso de que todas las variables estuviesen perfectamente correlacionadas, entonces el modelo de independencia podría sobreestimar o subestimar el valor de $P(I_i|X)$[1], sin embargo, algunas veces la reducción en la varianza resultante de los pocos parámetros en el modelo de independencia compensa con creces cualquier aumento en el sesgo y se sospecha que este hecho ocurre frecuentemente en la práctica, siendo una factor importante para explicar el buen desempeño del método (Hand & Yu, 2001).

Una segunda razón por la cual el supuesto de independencia no es tan irrazonable podría ser que a menudo los datos son sometidos a algún procedimiento de selección de variables antes de realizar la clasificación. En estos procedimientos se eliminan variables altamente correlacionadas llevando a que las relaciones entre las variables restantes se aproximen a la independencia.

Asimismo, diversos estudios empíricos han demostrado lo bien que funciona el método al compararlo con árboles de decisión y redes neuronales. Por ejemplo, X. Wu et al. (2007) señala un primer estudio clásico que compara métodos de clasificación supervisada, donde Titterington (1981) descubrió que el modelo Naïve Bayes produjo el mejor resultado general, mientras que Mani (1997), en otro estudio, llegó a la conclusión de que el método Naïve Bayes arrojó la predicción más efectiva para la concurrencia de cáncer de mama. Hand & Yu (2001) presentan otras investigaciones sobre la sorprendente eficacia de Naïve Bayes entre los cuales se pueden destacar estudios de enfermedades cardiacas (Russek, Kronmal & Fisher, 1983), enfermedades de la tiroides (Nordyke, Kulikowski & Kulikowski, 1971), enfermedades del hígado (Croft & Machol, 1987) y dispepsia (Fox, Barber & Bardhan, 1980), donde el método Naïve Bayes fue una muy buena elección. Domingos y Pazzani (1997), presentan más comparaciones empíricas donde se refleja el buen desempeño del clasificador Naïve Bayes; por ejemplo, los trabajos de Muramatsu & Billsus (1996), que compararon métodos para filtrado de información y encontraron que el clasificador bayesiano era el más preciso; y Langley, Iba & Thompson (1992) que compararon el clasificador naïve Bayes con un árbol de decisión y notaron que era más preciso en cuatro de los cinco conjuntos de datos utilizados.

Como era de esperarse, también existen estudios que muestran un menor desempeño por parte del método Naïve Bayes, pero al parecer son una minoría. Un ejemplo es el proyecto STATLOG (King, Feng & Sutherland, 1995), donde el modelo de independencia Naïve Bayes apareció cerca del final de la tabla de desempeño para casi todos los conjuntos de datos, aunque para el conjuntos de datos cardiacos, el método bayesiano fue el que presentó el mejor desempeño (Hand & Yu, 2001).

[1] Razón por la cual, algunos investigadores han creado extensiones del modele Naïve Bayes, como por ejemplo la "Extensión de Laplace" o la "Extensión de Langley" (1993).

En las secciones siguientes se explica el modelo Naïve Bayes, basándose en ambos paradigmas mencionados.

2.3.1. Caso general

Sea D un conjunto de tuplas u observaciones iniciales, las cuales se consideran como el conjunto de datos de entrenamiento. Cada tupla, X, perteneciente a D, está representada por un vector de medición, $X = (x_1, x_2, \ldots, x_p)$, donde cada valor x_k representa un atributo que fue medido en la observación X.

Suponga también que existen m etiquetas de la clase objetivo I_i con $i = 1, \ldots, m$.

2.3.1.1. Paradigma de Muestreo

El objetivo del clasificador es determinar la probabilidad de pertenencia a la clase I_i dado la tupla X y seleccionar la clase con probabilidad más alta. Por consiguiente, para cada categoría se desea calcular $P(I_i|x_1, \ldots, x_p)$.

Bajo el contexto del clasificador Naïve Bayes, los eventos A_i y B utilizados en la ecuación 2.1 pueden sustituirse por una clase I_i y por el conjunto de atributos $X = \{x_1, \ldots, x_p\}$, respectivamente. De este modo, $P(I_i|X)$ puede escribirse como se muestra a continuación:

$$P(I_i|X) = \frac{P(X|I_i)P(I_i)}{P(X)} \propto P(X|I_i)P(I_i) \tag{2.2}$$

donde $P(I_i|X)$ es lo que se pretende predecir y corresponde a la probabilidad de que un objeto con vector de medición $X = (x_1, x_2, ..., x_p)$ pertenezca a la clase i, $P(X|I_i)$ es la distribución condicional de X dado I_i y $P(I_i)$ es la probabilidad a priori de que un objeto sea de clase i. $P(X)$ es la distribución global de la mezcla de ambas clases y se define de la siguiente forma:

$$P(X) = \sum_{i=1}^{m} P(X|I_i)P(I_i)$$

Sin embargo, $P(X)$ es constante para todas las clases i, por lo tanto para calcular $P(I_i|X)$ solo es necesario maximizar $P(X|I_i)P(I_i)$.

Como generalmente el cálculo de $P(I_i)$ no se hace muy complicado, el método se centra mayoritariamente en estimar $P(X|I_i)$; y para simplificar su cálculo, el método supone la existencia de independencia condicionada entre los atributos de X en relación con las clases, por lo tanto la probabilidad $P(X|I_i)$ se puede reescribir de la siguiente forma:

$$P(X|I_i) = P(x_1|I_i) \cdot P(x_2|I_i) \ldots P(x_p|I_i) = \prod_{k=1}^{p} P(x_k|I_i)$$

y el cálculo de los términos individuales de $P(x_k|I_i)$ depende de la distribución que siga cada variable x_k condicionada a I_i.

La ecuación 2.2, bajo el supuesto de independencia, queda definida de la siguiente forma:

$$P(I_i|X) = \frac{P(I_i)\prod_{k=1}^{p} P(x_k|I_i)}{P(X)} \propto P(I_i)\prod_{k=1}^{p} P(x_k|I_i) \tag{2.3}$$

De este modo, el problema que en un comienzo era p-dimensional multivariado, se transforma en p problemas univariados, pues cada distribución $P(x_k|I_i)$ se calcula de manera individual. La estimación de distribuciones univariadas es mucho más simple y familiar, además requiere de menor cantidad de observaciones para obtener estimaciones precisas que cuando se utiliza una estimación de distribución multivariada (X. Wu et al., 2007).

Luego de calcular $P(I_i|X)$ para cada $i = 1, \ldots, m$ se debe escoger la categoría que presente la mayor probabilidad dada la observación X.

$$\underset{i\in I}{argmax}\ P(I_i|X) \propto \underset{i\in I}{argmax} P(I_i)\prod_{k=1}^{p} P(x_k|I_i) \tag{2.4}$$

Esto se conoce como *Regla de decisión de Máxima probabilidad a Posteriori.*

2.3.1.2. Paradigma de Diagnóstico

El método ha sido abordado hasta ahora según el Paradigma de muestreo, pues ha sido tratado basándose en la estimación de distribuciones condicionales de clase de manera separada. Ahora, desde el Paradigma de diagnóstico, X. Wu et al. (2007) señalan que el método refleja su verdadera elegancia cuando se toma en cuenta que si se utiliza cualquier transformación, T, estrictamente monótona de $P(I_i|X)$, se pueden obtener clasificaciones equivalentes a los mencionados anteriormente:

$$P(I_i|X) > P(I_i|Y) \Leftrightarrow T(P(I_i|X)) > T(P(I_i|Y))$$

Lo mismo sucede al realizar una transformación, T, estrictamente monótona del Umbral de clasificación t:

$$P(I_i|X) > t \Leftrightarrow T(P(I_i|X)) > T(t)$$

Esto significa que si t es el umbral de clasificación con el que cada $P(I_i|X)$ es comparado, entonces al comparar $T(t)$ con cada $T(P(I_i|X))$ se obtiene la misma clasificación que sin aplicar la transformación.

Ahora bien, para un problema de clasificación binaria, se debe utilizar un conjunto inicial de objetos que ya pertenece a una clase y con estos datos de entrenamiento se construye un *puntaje* o

score con el que nuevas tuplas se pueden asignar a una categoría por medio de la regla de decisión de máxima probabilidad a posteriori de la ecuación 2.4, o bien, comparando su puntaje con un *Umbral de Clasificación* que se debe escoger previamente (es muy común utilizar como umbral el valor 0.5 y así cada nueva observación se asigna a la clase que se estime es más probable que provenga).

Para el problema binario donde I puede tomar los valores 0 o 1, una transformación monótona creciente es la razón

$$\frac{P(I=1|X)}{1-P(I=1|X)} = \frac{P(1|X)}{P(0|X)} \tag{2.5}$$

Usando el método Naïve Bayes planteado en la ecuación 2.3, se tiene:

$$P(1|X) = \frac{P(1)\prod_{k=1}^{p} P(x_k|1)}{P(X)} \quad \text{y} \quad 1-P(1|X) = \frac{P(0)\prod_{k=1}^{p} P(x_k|0)}{P(X)}$$

y la razón 2.5 puede ser reescrita de la siguiente forma:

$$\frac{P(1|X)}{1-P(1|X)} = \frac{P(1)}{P(0)} \prod_{k=1}^{p} \frac{P(x_k|1)}{P(x_k|0)} \tag{2.6}$$

Otra manera de obtener el puntaje o score es mediante la "log transformación", que también es una transformación monótona creciente.

$$ln\frac{P(1|X)}{1-P(1|X)} = ln\frac{P(1)}{P(0)} + \sum_{k=1}^{p} ln\frac{P(x_k|1)}{P(x_k|0)} \tag{2.7}$$

Si se define $g_k(x_k) = ln\frac{P(x_k|1)}{P(x_k|0)}$ y la constante $k = ln\frac{P(1)}{P(0)}$, la ecuación 2.7 queda expresada como:

$$W = k + \sum_{k=1}^{p} g_k(x_k) \tag{2.8}$$

De este modo, la ecuación 2.8 es simplemente una suma de las contribuciones de cada atributo por separado, es decir, el modelo Naïve Bayes es una suma de valores transformados del vector de medición X (lo que deja en evidencia la facilidad de interpretación del método).

2.4. Métricas de evaluación

Para evaluar el rendimiento de los clasificadores se utilizaron 4 medidas que cuantifican su desempeño. Para ello, se requiere la utilización de una matriz de confusión que, en términos simples, es un conteo de aciertos y errores de cada una de las categorías que fueron clasificadas respecto a su verdadero valor.

En el Cuadro 2.1 se presenta la estructura de una matriz de confusión.

		Predicción		
		0	1	Total
Valor	0	VN	FP	N’
real	1	FN	VP	P
	Total	N’	P’	T

Cuadro 2.1: Matriz de Confusión.

donde,
Verdadero Positivo (VP) : es el número de predicciones correctas de la clase objetivo 1.
Verdadero Negativo (VN) : es el número de predicciones correctas de la clase objetivo 0.
Falso Positivo (FP) : es el número de predicciones incorrectas de que un caso pertenece a la clase 1, cuando en realidad pertenece a la clase 0.
Falso Negativo (FN) : es el número de predicciones incorrectas de que un caso pertenece a la clase 0, cuando en realidad pertenece a la clase 1.

A continuación se explican las medidas de evaluación utilizadas parar comparar el rendimiento de los clasificadores.

- **Accuracy**: corresponde a la tasa de aciertos, es decir, el número total de predicciones que fueron correctas.
$$Acc = \frac{VP + VN}{T}$$
- **Precisión**: es la proporción de predichos positivos que fueron correctos.
$$P = \frac{VP}{VP + FP} = \frac{VP}{P'}$$
- **Sensibilidad**: es el número de casos clasificados correctamente como positivos respecto del total de positivos observados.
$$R = \frac{VP}{VP + FN} = \frac{VP}{P}$$
- **Medida F1**: corresponde a la media aritmética entre la Precisión y la Sensibilidad.
$$F1 = 2 \cdot \frac{PR}{P + R}$$

Capítulo 3

Implementación Naïve Bayes

Este capítulo aborda el método Naïve Bayes desde el paradigma de diagnóstico, considerando que los atributos pueden seguir distribución Normal o Bernoulli. Además, para el caso de Naïve Bayes en streaming data, se explica cómo actualizar los parámetros ante la llegada de nuevas observaciones a la base de datos.

Por último, en esa misma línea, trata la implementación computacional del método mediante funciones, así como la simulación de un data stream.

3.1. Procesamiento en Batch

3.1.1. Variables independientes Normales

Suponga que los atributos siguen una distribución Normal condicionado a la clase I, es decir, $x_k|i \sim Normal(\mu_{ik}, \sigma^2_{ik})$, donde μ_{ik} y σ^2_{ik} son respectivamente la media y varianza de la k-ésima variable en la i-ésima clase.

La función de densidad está dada por:

$$f(x_k|i) = \frac{1}{\sqrt{2\pi\sigma^2_{ik}}} e^{-\frac{(x_k-\mu_{ik})^2}{2\sigma^2_{ik}}} \tag{3.1}$$

Para el caso $i = 1$ y bajo el supuesto de independencia, la ecuación 2.3 queda definida como:

$$P(1|X) = \frac{P(1)\prod_{k=1}^{p} f(x_k|1)}{f(X)} \propto P(1)\prod_{k=1}^{p} f(x_k|1) \tag{3.2}$$

y como la clase I puede tomar los valores 0 o 1, se está en presencia de una variable del tipo Bernoulli de parámetro θ.

De esta manera, la probabilidad $P(I = 1)$ es igual a θ[1] y como los atributos siguen una distribución Normal, $P(1|X)$ se puede reescribir como:

$$P(1|X) \propto \theta \prod_{k=1}^{p} \frac{1}{\sqrt{2\pi\sigma_{1k}^2}} e^{-\frac{(x_k-\mu_{1k})^2}{2\sigma_{1k}^2}}$$

Por lo tanto, la probabilidad $P(1|X)$ queda definida como se muestra en la ecuación 3.3.

$$\boxed{P(1|X) = \frac{\theta \frac{1}{(\sqrt{2\pi})^p} \left(\prod_{k=1}^{p} \frac{1}{\sqrt{\sigma_{1k}^2}} \right) e^{-\frac{1}{2}\sum_{k=1}^{p} \frac{(x_k-\mu_{1k})^2}{\sigma_{1k}^2}}}{f(X)}} \tag{3.3}$$

Ahora, para el caso $I = 0$, el cálculo de $P(0|X)$ se puede realizar de manera análoga, resultando:

$$P(0|X) = \frac{P(0)\prod_{k=1}^{p} f(x_k|0)}{f(X)} \propto P(0) \prod_{k=1}^{p} f(x_k|0) \tag{3.4}$$

Además, como se explicó anteriormente, I sigue una distribución Bernoulli de parámetro θ y la probabilidad $P(I = 0)$ es igual a $(1 - \theta)$[2]. Por lo tanto,

$$P(0|X) \propto (1-\theta) \frac{1}{(\sqrt{2\pi})^p} \left(\prod_{k=1}^{p} \frac{1}{\sqrt{\sigma_{0k}^2}} \right) e^{-\frac{1}{2}\sum_{k=1}^{p} \frac{(x_k-\mu_{0k})^2}{\sigma_{0k}^2}} \tag{3.5}$$

y la probabilidad $P(0|X)$ queda definida como se muestra en la ecuación 3.6.

$$\boxed{P(0|X) = \frac{(1-\theta) \frac{1}{(\sqrt{2\pi})^p} \left(\prod_{k=1}^{p} \frac{1}{\sqrt{\sigma_{0k}^2}} \right) e^{-\frac{1}{2}\sum_{k=1}^{p} \frac{(x_k-\mu_{0k})^2}{\sigma_{0k}^2}}}{f(X)}} \tag{3.6}$$

Ahora, basándose en el Paradigma de diagnóstico se llega a la siguiente expresión:

$$\boxed{\frac{P(1|X)}{1-P(1|X)} = \frac{\theta}{(1-\theta)} \left(\prod_{k=1}^{p} \sqrt{\frac{\sigma_{0k}^2}{\sigma_{1k}^2}} \right) exp\left\{ \sum_{k=1}^{p} \frac{(x_k-\mu_{0k})^2}{2\sigma_{0k}^2} - \frac{(x_k-\mu_{1k})^2}{2\sigma_{1k}^2} \right\}} \tag{3.7}$$

[1] Puesto que $P(I = 1) = \theta^1(1-\theta)^{1-1} = \theta$.
[2] Puesto que $P(I = 0) = \theta^0(1-\theta)^{1-0} = (1-\theta)$.

Si se aplica la "log transformación" a la relación planteada en la ecuación 3.7 se tiene lo siguiente:

$$ln\frac{P(1|X)}{1-P(1|X)} = ln\left(\frac{\theta}{1-\theta}\right) + \sum_{k=1}^{p} ln\left(\sqrt{\frac{\sigma_{0k}^2}{\sigma_{1k}^2}}\right) + \sum_{k=1}^{p}\left(\frac{(x_k-\mu_{0k})^2}{2\sigma_{0k}^2} - \frac{(x_k-\mu_{1k})^2}{2\sigma_{1k}^2}\right) \tag{3.8}$$

$$\underbrace{ln\frac{P(1|X)}{1-P(1|X)}}_{W} = \underbrace{ln\left(\frac{\theta}{1-\theta}\right)}_{k} + \sum_{k=1}^{p}\underbrace{\left(ln\left(\sqrt{\frac{\sigma_{0k}^2}{\sigma_{1k}^2}}\right) + \frac{(x_k-\mu_{0k})^2}{2\sigma_{0k}^2} - \frac{(x_k-\mu_{1k})^2}{2\sigma_{1k}^2}\right)}_{g_k(x_k)} \tag{3.9}$$

Sea $k = ln\left(\frac{\theta}{1-\theta}\right)$ y $g_k(x_k) = ln\left(\sqrt{\frac{\sigma_{0k}^2}{\sigma_{1k}^2}}\right) + \frac{(x_k-\mu_{0k})^2}{2\sigma_{0k}^2} - \frac{(x_k-\mu_{1k})^2}{2\sigma_{1k}^2}$, entonces:

$$W = k + \sum_{k=1}^{p} g_k(x_k) \tag{3.10}$$

De esta forma, el puntaje W queda definido como una suma de las contribuciones de cada variable x_{ik}.

3.1.1.1. Estimación por Máxima verosimilitud

Para cada clase I_i el puntaje $P(I_i|X)$ debe ser calculado utilizando tres parámetros, que se estiman por medio del método de Máxima Verosimilitud[3]. Así, para el caso $I = 1$, se deben estimar los parámetros θ, μ_{1k} y σ_{1k}^2, mientras que para el caso en que $I = 0$, los parámetros son θ, μ_{0k} y σ_{0k}^2.

Parámetro θ. Como se comentó anteriormente, la clase I puede tomar los valores 0 y 1, por lo tanto se está en presencia de una distribución de probabilidad Bernoulli con parámetro θ. De esta forma, si $I = 1$ se considera "éxito", la función de distribución de I estaría dada por:

$$f(x;\theta) = \begin{cases} \theta & si\ i{=}1 \\ (1-\theta) & si\ i{=}0 \end{cases}$$

El estimador de máxima verosimilitud de una distribución Bernoulli es la proporción o frecuencia empírica de la etiqueta de clase que se considera como éxito.

[3]Por esta razón el método también es conocido como "Empirical Naïve Bayes" o "Naïve Bayes Empírico", pues en estos casos la probabilidad a priori es estimada a partir de los datos.

Así, para una muestra $D = \{X_1, \ldots, X_j, \ldots, X_n\}$ el estimador de máxima verosimilitud se define como:

$$\hat{\theta}_n = \frac{\sum_{j=1}^{n} i_j}{n}$$

donde $\sum_{j=1}^{n} i_j$ es el número de observaciones cuyo valor de I es 1.

Parámetro μ_{ik}. El estimador de máxima verosimilitud de μ_{ik} para una muestra D es el promedio muestral y está dado por:

$$\hat{\mu}_{ik,n_i} = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ik,j} = \overline{X}_{ik,n_i}$$

donde $x_{ik,j}$ es el valor que toma la k-ésima variable en la j-ésima tupla perteneciente a la clase i y n_i es la cantidad de observaciones de D que pertenecen a la clase $I = i$.

Parámetro σ^2_{ik}. El estimador de máxima verosimilitud de σ^2_{ik} para una muestra D corresponde a la varianza muestral y está dado por:

$$\hat{\sigma}^2_{ik,n_i} = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} (x_{ik,j} - \overline{X}_{ik,n_i})^2$$

donde $\overline{X}_{ik,n_i}$ es el promedio muestral de la k-ésima variable para observaciones pertenecientes a la clase $I = i$ (es decir, es el estimador de máxima verosimilitud de μ_{ik}).

3.1.2. Variables independientes Bernoulli

Suponga que las variables explicativas x_k condicionadas a la clase I_i tienen distribución Bernoulli, es decir, $x_k|I = i \sim Bernoulli(\theta_{ik})$, con la siguiente función de probabilidad:

$$P(x_k|I = i) = \theta_{ik}^{x_k}(1 - \theta_{ik})^{1-x_k} \text{ donde } i = 0, 1 \ , \ k = 1, \ldots, p \tag{3.11}$$

Si se consideran las distribuciones de I y $x_k|I$, la probabilidad $P(1|X)$ puede ser escrita como se muestra a continuación.

$$P(1|X) \propto \theta \cdot \theta_{11}^{x_1}(1 - \theta_{11})^{1-x_1} \cdot \theta_{12}^{x_2}(1 - \theta_{12})^{1-x_2} \ldots \theta_{1p}^{x_p}(1 - \theta_{1p})^{1-x_p}$$

$$\iff P(1|X) \propto \theta \prod_{k=1}^{p} \theta_{1k}^{x_k}(1 - \theta_{1k})^{1-x_k}$$

Llegando a la siguiente expresión:

$$\boxed{P(1|X) = \frac{\theta \prod_{k=1}^{p} \theta_{1k}^{x_k}(1 - \theta_{1k})^{1-x_k}}{P(X)}} \tag{3.12}$$

Para el caso en que la clase objetivo toma el valor $I = 0$, el cálculo se realiza de manera análoga, considerando las distribuciones de I y $x_k|I$.

$$P(0|X) \propto (1 - \theta) \prod_{k=1}^{p} \theta_{0k}^{x_k}(1 - \theta_{0k})^{1-x_k}$$

Resultando la siguiente expresión:

$$\boxed{P(0|X) = \frac{(1 - \theta) \prod_{k=1}^{p} \theta_{0k}^{x_k}(1 - \theta_{0k})^{1-x_k}}{P(X)}} \tag{3.13}$$

Ahora, enfocándose en el paradigma de diagnóstico, se plantea la siguiente razón:

$$\boxed{\frac{P(1|X)}{1 - P(1|X)} = \frac{\theta}{(1 - \theta)} \prod_{k=1}^{p} \left(\frac{\theta_{1k}}{\theta_{0k}}\right)^{x_k} \prod_{k=1}^{p} \left(\frac{1 - \theta_{1k}}{1 - \theta_{0k}}\right)^{1-x_k}} \tag{3.14}$$

Si se aplica logaritmo a la transformación, también se obtiene una transformación monótona creciente, del siguiente modo:

$$ln\frac{P(1|X)}{1-P(1|X)} = ln\left(\frac{\theta}{1-\theta}\right) + \sum_{k=1}^{p} x_k\ ln\left(\frac{\theta_{1k}}{\theta_{0k}}\right) + \sum_{k=1}^{p}(1-x_k)ln\left(\frac{1-\theta_{1k}}{1-\theta_{0k}}\right)$$

$$\iff \underbrace{ln\frac{P(1|X)}{1-P(1|X)}}_{W} = \underbrace{ln\left(\frac{\theta}{1-\theta}\right)}_{k} + \sum_{k=1}^{p}\underbrace{\left[x_k\ ln\left(\frac{\theta_{1k}}{\theta_{0k}}\right) + (1-x_k)ln\left(\frac{1-\theta_{1k}}{1-\theta_{0k}}\right)\right]}_{g_x(x_k)}$$

Sea $k = ln\left(\frac{\theta}{1-\theta}\right)$ y $g_x(x_k) = x_k\ ln\left(\frac{\theta_{1k}}{\theta_{0k}}\right) + (1-x_k)ln\left(\frac{1-\theta_{1k}}{1-\theta_{0k}}\right)$, entonces:

$$W = k + \sum_{k=1}^{p} g_x(x_k)$$

Por lo tanto, para el caso de variables independientes bernoulli, el puntaje W también queda representado como una suma de valores transformados.

3.1.2.1. Estimación por Máxima verosimilitud

Para calcular el puntaje $P(I_i|X)$ se deben estimar los parámetros por medio del método de Máxima Veromilitud. En este caso las variables siguen distribución Bernoulli, entonces los parámetros a estimar para la clase $I = i$ serían θ y θ_{ik}.

Parámetro θ. Idéntico al caso de variables normales.

Parámetro θ_{ik}. El estimador de máxima verosimilitud de una distribución de probabilidad Bernoulli es la proporción. Por lo tanto, como la variable x_k puede tomar los valores 0 y 1, se supondrá que el "éxito" es cuando x_k toma el valor 1. De esta forma, para una muestra $D = \{X_1, \ldots, X_n\}$, el estimador de máxima verosimilitud está definido por:

$$\hat{\theta}_{ik,n_i} = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ik,j}$$

donde n_i es la cantidad de observaciones de D que pertenecen a la clase i y $x_{ik,j}$ es el valor que toma la k-ésima variable en la observación j-ésima de la clase i.

3.2. Procesamiento en Stream

Desde hace algunos años se está viviendo una época de revolución de la información, donde los avances en la tecnología han facilitado la capacidad de recopilar datos de manera continua. Esto ha generado que los datos no solo se almacenen en grandes cantidades, sino que también sigan creciendo rápidamente a través del tiempo (Y. Xu et al., 2006). Transacciones simples de la vida diaria, como el uso de tarjeta de crédito, transferencias bancarias, llamadas telefónicas y correos electrónicos se añaden a bases de datos ya existentes gracias al almacenamiento automatizado de datos.

Los data streams poseen ciertas características o particularidades: tienen gran volumen, son de tamaño ilimitado y son dinámicos, es decir, cambian constantemente (I. Frías Blanco, 2012). Por lo tanto, muchas operaciones de minería de datos, como la clasificación o agrupación, se vuelven más difíciles de abordar según explica C. C. Aggarwal (2006). De hecho, uno de los mayores desafíos para el análisis de streaming data se debe al hecho de que no es factible almacenar el flujo de datos en su totalidad y esto hace que las soluciones de los problemas sean aún más difíciles desde un punto de vista algorítmico y computacional.

Además, como se almacena la información de manera continua, las propiedades estadísticas de los datos no son estables y podrían cambiar con el tiempo. Este fenómeno se conoce como "*Concept drift*" (P. E. Lutu, 2013).

Existen dos principales perspectivas para analizar la información, estas son procesamiento en batch (batch processing) y procesamiento en stream (streaming processing). La primera consiste en analizar grandes volúmenes de datos en un mismo instante de tiempo y es una forma extremadamente eficiente de procesar datos que se recopilan a lo largo de un periodo de tiempo.

Ahora bien, las mismas tareas que se realizan en batch también se pueden realizar en un data stream. De esta forma, el procesamiento en stream consiste en analizar los datos en tiempo real o de manera casi instantánea, donde la principal dificultad radica en la velocidad con que los datos llegan. En lugar de procesar grandes cantidades de datos una sola vez, el procesamiento en stream analiza pequeñas cantidades de manera continua, es decir, en todo momento (A. Rayón, 2016), actualizando continuamente los patrones o modelos.

H. Borchani (2012) refiere que en el problema tradicional de clasificación en batch, todo el conjunto de datos de entrenamiento se utiliza para inducir un clasificador o un modelo que será válido para clasificar todas las nuevas instancias a lo largo del tiempo. Sin embargo, la clasificación de un data stream presenta más desafíos, esto debido a su longitud infinita y naturaleza evolutiva. Así, es muy probable es que el clasificador quede obsoleto o sea menos relevante después de una ventana de tiempo, y el uso de un modelo desactualizado podría dar lugar a una precisión muy baja. Por lo tanto, capturar de estos cambios mediante la actualización de los modelos permite

desarrollar el modelo de manera efectiva (C. C. Aggarwal, 2006).

Durante el último tiempo, la minería de datos en stream ha recibido una atención cada vez mayor, siendo el principal objetivo hacer frente al "concept drift" para mantener actualizados los modelos. En particular, las técnicas de clasificación en data streams han llamado la atención de muchos investigadores, debido a la importancia de sus aplicaciones que pueden variar desde la astronomía y geofísica hasta el soporte de decisiones en tiempo real de empresas e industrias.

Generalmente estas técnicas han sido diseñadas para construir modelos de clasificación a partir de conjuntos de datos finitos, lo que no es posible en el caso de un data stream. Por ende, ha sido necesario rediseñar los clasificadores en el contexto de un flujo de datos. Este proceso de adaptación de los modelos de clasificación no es a menudo trivial. Sin embargo, P. E. Lutu (2013) señala que el método Naïve Bayes ha tomado un rol importante en lo que se refiere a clasificación de streaming data, pues actualizar el modelo a medida que llegan nuevas observaciones no es tan complejo y presenta resultados experimentales que demuestran que el método proporciona altos niveles de rendimiento al ser aplicado en un flujo de datos.

3.2.1. Regla de actualización

Para aplicar el método en el contexto de un data stream se debe plantear una regla de actualización, que permita actualizar los parámetros cada vez que una nueva observación se incorpore a la base de datos.

Con el método en batch se obtienen por primera vez las estimaciones de cada parámetro considerando n observaciones iniciales (que corresponden a la base de datos de entrenamiento). Así, cada vez que lleguen nuevas observaciones, el clasificador debe asignar las nuevas tuplas a una clase objetivo y posteriormente actualizar los parámetros. En ese sentido, con el procesamiento en stream, el modelo se ahorra clasificar la base de datos completa cada vez que llegan nuevas observaciones.

La regla para actualizar el clasificador Naïve Bayes consiste en obtener constantes de actualización con las que se puede obtener la estimación de cada parámetro cada vez que llegan m observaciones nuevas.

3.2.1.1. Caso variables independientes Normales

Cuando llega una nueva observación X_{n+1}, los parámetros se deben actualizar bajo las siguientes **condiciones**:

(1) Si la observación X_{n+1} pertenece a la clase $I = 0$, μ_{1k} y σ^2_{1k} se mantienen y los parámetros θ, μ_{0k} y σ^2_{0k} se deben actualizar.

(2) Si la observación X_{n+1} pertenece a la clase $I = 1$, μ_{0k} y σ^2_{0k} se mantienen y los parámetros θ, μ_{1k} y σ^2_{1k} se deben actualizar.

A continuación se presentan las fórmulas de actualización de cada parámetro.

i) Parámetro θ. El estimador de máxima verosimilitud de θ, para el conjunto de datos de entrenamiento $D = \{X_1, \ldots, X_n\}$ está definido por:

$$\hat{\theta}_n = \frac{\sum_{j=1}^{n} y_j}{n}$$

y dado m tuplas nuevas $X_{n+1}, X_{n+2}, ..., X_{n+m}$, el parámetro θ debe actualizarse de la siguiente forma:

$$\hat{\theta}_{n+m} = \frac{\sum_{j=1}^{n+m} y_j}{n+m} = \frac{\sum_{j=1}^{n} y_j + \sum_{j=n+1}^{n+m} y_j}{n+m} = \left(\frac{n}{n+m}\right) \frac{\sum_{j=1}^{n} y_j}{n} + \frac{\sum_{j=n+1}^{n+m} y_j}{n+m}$$

La fórmula de actualización del parámetro θ queda definida como:

$$\hat{\theta}_{n+m} = D \cdot \hat{\theta}_n + C$$

donde $D = \frac{n}{n+m}$ y $C = \frac{\sum_{j=n+1}^{n+m} y_j}{n+m}$.

ii) Parámetro μ_{ik}. El estimador de máxima verosimilitud de μ_{ik}, para el conjunto de datos de entrenamiento D está definido por la siguiente expresión:

$$\hat{\mu}_{ik,n_i} = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ik,j} \quad ; \; i = 0, 1 \text{ y } k = 1, \ldots, p$$

donde $x_{ik,j}$ es el valor que toma la j-ésima observación en la k-ésima variable dentro de la clase $I = i$.

De esta forma, dado m observaciones nuevas $X_{n+1}, ..., X_{n+m}$, el parámetro μ_{ik} se debe actualizar del siguiente modo:

$$\hat{\mu}_{ik,n_i+m_i} = \frac{1}{n_i + m_i} \sum_{j=1}^{n_i+m_i} x_{ik,j} = \frac{1}{n_i + m_i} \left(\sum_{j=1}^{n_i} x_{ik,j} + \sum_{j=n_i+1}^{n_i+m_i} x_{ik,j} \right)$$

$$= \left(\frac{n_i}{n_i + m_i}\right) \cdot \frac{\sum_{j=1}^{n_i} x_{ik,j}}{n_i} + \frac{\sum_{j=n_i+1}^{n_i+m_i} x_{ik,j}}{n_i + m_i}$$

Por lo tanto, la fórmula de actualización del parámetro μ_{ik} queda definida por:

$$\hat{\mu}_{ik,n_i+m_i} = D \cdot \hat{\mu}_{ik,n_i} + C$$

donde $D = \frac{n_i}{n_i+m_i}$ y $C = \frac{\sum_{j=n_i+1}^{n_i+m_i} x_{ik,j}}{n_i+m_i}$.

iii) Parámetro σ^2_{ik}. El estimador de máxima verosimilitud de σ_{ik} para el conjunto de datos de entrenamiento D se define por la siguiente expresión:

$$\hat{\sigma}^2_{ik,n_i} = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}(x_{ik,j} - \overline{X}_{ik,n_i})^2$$

donde $x_{ik,j}$ es el valor que toma la j-ésima observación en la k-ésima variable dentro de la clase $I = i$ y $\overline{X}_{ik,n_i}$ es el promedio de la k-ésima variable con n_i observaciones que pertenecen a la clase $I = i$.

La fórmula para actualizar el parámetro σ^2_{ik} cuando llegan observaciones nuevas X_{n+m} requiere un poco más de cálculo y se realiza del siguiente modo:

$$\hat{\sigma}^2_{ik,n_i+m_i} = \frac{1}{(n_i + m_i) - 1}\sum_{j=1}^{n_i+m_i}(x_{ik,j} - \overline{X}_{ik,n_i+m_i})^2$$

$$= \frac{1}{n_i + m_i - 1}\left[\sum_{j=1}^{n_i+m_i} x^2_{ik,j} - (n_i + m_i)\overline{X}^2_{ik,n_i+m_i}\right]$$

$$= \frac{1}{n_i + m_i - 1}\left[\sum_{j=1}^{n_i} x^2_{ik,j} + \sum_{j=n_i+1}^{n_i+m_i} x^2_{ik,j} - (n_i + m_i)\overline{X}^2_{ik,n_i+m_i} + n_i\overline{X}^2_{ik,n_i} - n_i\overline{X}^2_{ik,n_i}\right]$$

$$= \frac{1}{n_i + m_i - 1}\left[\left(\sum_{j=1}^{n_i} x^2_{ik,j} - n_i\overline{X}^2_{ik,n_i}\right) + \sum_{j=n_i+1}^{n_i+m_i} x^2_{ik,j} - (n_i + m_i)\overline{X}^2_{ik,n_i+m_i} + n_i\overline{X}^2_{ik,n_i}\right]\cdot\frac{n_i - 1}{n_i - 1}$$

$$= \left(\frac{n_i - 1}{n_i + m_i - 1}\right)\cdot\frac{\sum_{j=1}^{n_i} x^2_{ik,j} - n_i\overline{X}^2_{ik,n_i}}{n_i - 1} + \frac{\sum_{j=n_i+1}^{n_i+m_i} x^2_{ik,j} - (n_i + m_i)\overline{X}^2_{ik,n_i+m_i} + n_i\overline{X}^2_{ik,n_i}}{n_i + m_i - 1}$$

Por lo tanto, la fórmula de actualización del parámetro σ^2_{ik} queda definida de la siguiente forma:

$$\hat{\sigma}^2_{ik,n_i+m_i} = D\cdot\hat{\sigma}^2_{ik,n_i} + C$$

donde $D = \frac{n_i - 1}{n_i + m_i - 1}$ y $C = \frac{\sum_{j=n_i+1}^{n_i+m_i} x^2_{ik,j} - (n_i + m_i)\overline{X}^2_{ik,n_i+m_i} + n_i\overline{X}^2_{ik,n_i}}{n_i + m_i - 1}$.

3.2.1.2. Caso variables independientes Bernoulli

Cuando llega una nueva observación X_{n+1}, los parámetros se deben actualizar bajo las siguientes **condiciones**:

(1) Si la observación X_{n+1} pertenece a la clase $I = 0$, μ_{1k} se mantiene y los parámetros θ y θ_{0k} se deben actualizar para las $n + 1$ tuplas.

(2) Si la observación X_{n+1} pertenece a la clase $I = 1$, μ_{0k} se mantiene y los parámetros θ y θ_{1k} se deben actualizar para las $n + 1$ tuplas.

A continuación se expresan las fórmulas de actualización de cada parámetro cuando los atributos siguen distribución de probabilidad Bernoulli.

i) Parámetro θ. Idéntico al caso con variables normales.

ii) Parámetro θ_{ik}. El estimador de máxima verosimilitud de θ_{ik} que se obtiene a través de los datos de entrenamiento D, se define de la siguiente forma:

$$\hat{\theta}_{ik,n_i} = \frac{\sum_{j=1}^{n_i} x_{ik,n_i}}{n_i}$$

Dado m observaciones nuevas $X_{n+1}, ..., X_{n+m}$, el parámetro θ_{ik} se debe actualizar del siguiente modo:

$$\hat{\theta}_{ik,n_i+m_i} = \frac{\sum_{j=1}^{n_i+m_i} x_{ik,j}}{n_i + m_i} = \left[\frac{\sum_{j=1}^{n_i} x_{ik,j} + \sum_{j=n_i+1}^{n_i+m_i} x_{ik,j}}{n_i + m_i} \right] \cdot \frac{n_i}{n_i}$$

$$= \left(\frac{n_i}{n_i + m_i} \right) \cdot \frac{\sum_{j=1}^{n_i} x_{ik,n_i}}{n_i} + \frac{\sum_{j=n_i+1}^{n_i+m_i} x_{ik,j}}{n_i + m_i}$$

Por lo tanto, la fórmula de actualización de parámetro θ_{ik} está dada por:

$$\hat{\theta}_{ik,n_i+m_i} = D \cdot \hat{\theta}_{ik,n_i} + C$$

donde $D = \frac{n_i}{n_i+m_i}$ y $C = \frac{\sum_{j=n_i+1}^{n_i+m_i} x_{ik,j}}{n_i+m_i}$.

3.3. Implementación en Software

En esta sección se presentan las funciones creadas para implementar el algoritmo Naïve Bayes considerando variables independientes Normales y Bernoulli.

El desarrollo de las funciones es mediante el Software estadístico RStudio en su versión 1.1.456 y todas las simulaciones, así como la implementación en datos astronómicos y de Twitter, se realizaron en un equipo con procesador Intel(R) Core (TM) i5-7200U CPU 2.50GHz 2.71GHz, memoria RAM de 4 GB y sistema operativo Windows 10 de 64 bits.

3.3.1. Variables Normales

- ***base_normal*** genera un conjunto de datos con variables normales condicionadas a la clase I. El conjunto de datos que se genera contiene n observaciones de p variables explicativas normales, donde $prop1$ es la proporción de observaciones que toman el valor $I = 1$ del total de tuplas. Está definida por los suguientes argumentos:

  ```
  base_normal(b,p,n,prop1,medias1,medias0,sd2_1,sd2_0)
  ```

  Argumentos:
  1. b : conjunto de datos donde se guardarán las n observaciones y p variables simuladas por la función.
  2. p : cantidad de variables explicativas.
  3. n : cantidad total de observaciones.
  4. *prop1* : proporción de observaciones de la categoría $I = 1$ en b, es decir, θ.
  5. *medias1/medias0* : vector de medias de las categorías $I = 1$ e $I = 0$, respectivamente.
  6. *sd2_1/sd2_0* : vector de desviaciones estándar de las categorías $I = 1$ y $I = 0$, respectivamente.

- ***NB_normal*** corresponde al algoritmo Naïve Bayes para el caso en batch. Esta función estima los parámetros y luego calcula el score de la ecuación 3.7, así como las probabilidades de pertenencia a cada clase $I = 1$ e $I = 0$. Está definida por los siguientes argumentos:

  ```
  NB_normal(b,threshold)
  ```

  Argumento:
  1. b : base de datos con variable respuesta I y variables explicativas $X_1, ..., X_p$ normales.
  2. *threshold* : umbral de clasificación con el cual se debe comparar el score.

- ***NB_actualizado*** corresponde al algoritmo Naïve Bayes para el caso en Stream. Esta función calcula el score de la ecuación 3.7 y las probabilidades de pertenencia a cada clase para las observaciones nuevas. Está definida por los siguientes argumentos:

```
NB_actualizado(mu0,mu1,sigma0,sigma1,theta,threshold,nuevo)
```

Argumentos:
1. *mu0/mu1* : vector de medias para la clase $I = 0$ e $I = 1$, respectivamente.
2. *sigma0/sigma1* : vector de varianzas para la clase $I = 0$ e $I = 1$, respectivamente.
3. *theta* : parámetro θ.
4. *threshold* : umbral de clasificación con el cual se debe comparar el score.
5. *nuevo* : conjunto de m observaciones nuevas.

- ***act_parametros*** actualiza los parámetros θ, μ_{ik} y σ^2_{ik} cada vez que se agregan nuevas tuplas a la base de datos. Está definida por los siguientes argumentos:

```
act_parametros<-function(mu0,mu1,sigma0,sigma1,theta,nuevo,Nb)
```

Argumentos:
1. *mu0/mu1* : vector de medias de la clase $I = 0$ e $I = 1$, respectivamente, considerando n tuplas.
2. *sigma0/sigma1* : vector de varianzas de la clase $I = 0$ e $I = 1$, respectivamente, considerando n tuplas.
3. *theta* : estimación de θ considerando n observaciones.
4. *nuevo* : conjunto de m observaciones nuevas que se incorporan a la base de datos.
5. *Nb* : vector que contiene el tamaño muestral n, n_0 y n_1.

3.3.2. Variables Bernoulli

- ***base_bernoulli*** genera un conjunto de datos con variables Bernoulli condicionadas a la clase. El conjunto de datos que se genera contiene n observaciones de p variables explicativas Bernoulli, donde $prop1$ es la proporción de observaciones que toman el valor $I = 1$ del total de tuplas. Está definida por los suguientes argumentos:

```
base_bernoulli<-function(b,p,n,prop1,thet1,thet0)
```

Argumentos:
1. b : conjunto de datos donde se guardarán las n observaciones y p variables simuladas por la función.
2. p : cantidad de variables explicativas Bernoulli.
3. n : cantidad total de observaciones de b.
4. *prop1* : proporción de observaciones de la categoría $I = 1$ en b, es decir, θ.
5. *thet1/thet0* : vector de θ_{1k} y θ_{0k}, respectivamente, para las p variables.

- ***NB_ber*** corresponde al algoritmo Naïve Bayes para el caso en batch. Esta función estima los parámetros y luego calcula el score de la ecuación 3.14, así como las probabilidades de pertenencia a cada clase $I = 1$ e $I = 0$. Está definida por los siguientes argumentos:

```
NB_ber(b,threshold)
```

Argumento:
1. b : base de datos con variable respuesta I y variables explicativas $X_1, ..., X_p$ Bernoulli.
2. *threshold* : umbral de clasificación con el cual se debe comparar el score.

- ***NB_ber_actualizado*** corresponde al algoritmo Naïve Bayes para el caso en Stream. Esta función calcula el score de la ecuación 3.14 y las probabilidades de pertenencia a cada clase de las nuevas observaciones de la base de datos. Está definida por los siguientes argumentos:

```
NB_ber_actualizado<-function(theta,theta1,theta0,nuevo,threshold,n,p)
```

Argumentos:
1. *theta* : parámetro θ.
2. *theta1/theta0* : estimación del vector θ_{1k} y θ_{0k}, respectivamente.
3. *nuevo* : conjunto de m tuplas nuevas que se incorporan a la base de datos.
4. *threshold* : umbral de clasificación con el cual se debe comparar el score.
5. n : cantidad de observaciones que tiene la base de datos sin las m observaciones nuevas.
6. p : cantidad de variables explicativas.

- ***act_parametros_ber*** tiene como objetivo actualizar las estimaciones de los parámetros θ y θ_{ik} cada vez que se agregan nuevas tuplas a la base de datos. Está definida por los siguientes argumentos:

```
act_parametros_ber<-function(theta,theta1,theta0,nuevo,Nb)
```

Argumentos:
1. *theta* : estimación de θ considerando n observaciones.
2. *theta1/theta0* : estimación del vector θ_{1k} y θ_{0k}, respectivamente.
3. *nuevo* : conjunto de observaciones nuevas que se incorporan a la base de datos b.
4. *Nb* : vector que contiene el tamaño muestral n, n_0 y n_1.

3.3.3. Simulación de un data stream

Para realizar la implementación, en primer lugar se debe dividir la base de datos en dos subconjuntos: 70 % correspondiente a la base de datos de entrenamiento, que se utiliza para la fase de aprendizaje del modelo; y el 30 % restante corresponde a la base de datos de prueba, utilizada en la etapa de clasificación.

En ese sentido, con las funciones creadas se abordó la clasificación mediante los dos tipos de procesamientos de datos mencionados:

- **Procesamiento en Batch**. En este caso, se utilizan las funciones *NB_normal* y *NB_ber*, dependiendo de cuál sea el caso. Con la base de datos del 70 % se entrena el modelo y luego, se aplica el método Naïve Bayes para la base de datos de prueba. Esto se realiza solo una vez y si llegan nuevas observaciones a la base de datos, se debe realizar nuevamente con todo el conjunto de datos.

- **Procesamiento en Stream**. A continuación se explica cómo se aplica el algoritmo Naïve Bayes para el caso en línea.

  1. En primer lugar, se debe entrenar el modelo para la base de datos del 70 % mediante la función *NB_normal* o *NB_ber*, dependiendo del caso. Aquí, se estiman los parámetros del modelo por primera vez.
  2. Luego, con la base de datos de prueba se realiza el método en línea de la siguiente forma:
     i) Seleccionar una cantidad m de tuplas tal que m debe ser menor a la cantidad de observaciones de la base de prueba.
     ii) Con los parámetros obtenidos del entrenamiento del modelo, clasificar las m observaciones nuevas.
     iii) Agregar las m observaciones a la base de datos y actualizar los parámetros del modelo.

iv) Seleccionar las siguiente m observaciones de la base de datos de prueba y clasificarlas considerando la actualización de parámetros proveniente de la iteración anterior.

v) Volver al paso iii).

El proceso termina cuando ya no quedan observaciones en la base de datos de prueba del 30 %.

Por lo tanto, cada vez que se incorporan m observaciones nuevas a la base de datos, estas se clasifican y posteriormente se actualizan los parámetros del modelo con el fin de ser utilizados la próxima vez que lleguen nuevas observaciones a la base de datos (lo que se conoce como aprendizaje continuo).

Capítulo 4

Resultados variables Normales

En este capítulo se exponen los resultados para el caso de variables independientes normales, comparándose las funciones creadas para los casos en lotes y en línea, frente al comando *naiveBayes* que incorpora el paquete e1071 de RStudio.

4.1. Conjunto de datos simulado

Mediante la función *base_normal* se generó una base de datos de un total de 2 millones de observaciones (véase anexo Figura 6.1), con 4 variables explicativas normales. Los datos utilizados fueron los siguientes:

- θ =0.6
- μ_{0k} =(10.2 9.7 4.3 7.1)
- σ^2_{0k} =(16 36 16 9)
- μ_{1k} =(4.3 5.2 10.3 15.2)
- σ^2_{1k} =(4 4 1 25)

Además, como se mencionó en la sección 3.3.3, para realizar la clasificación el conjunto de datos fue dividido en dos subconjuntos: 70 % correspondiente a la base de datos de entrenamiento y el 30 % restante corresponde a la base de datos de prueba.

La división de la base de datos en dos submuestras se presenta en la Figura 4.1.

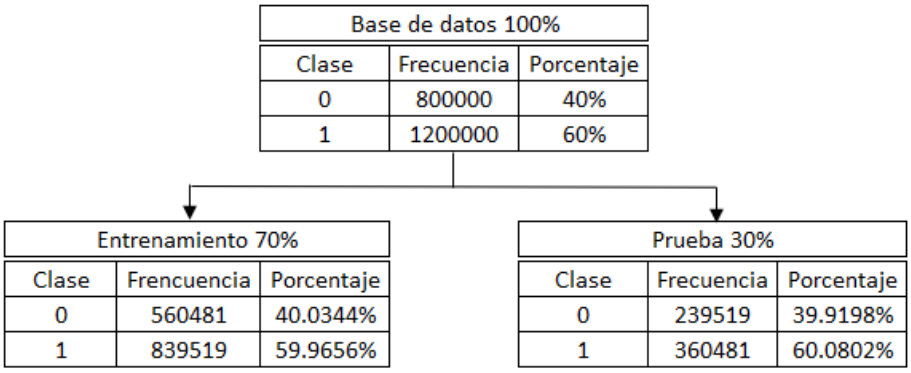


Figura 4.1: División base de datos en entrenamiento y prueba.

Se observa que la variable respuesta binaria presenta proporciones similares tanto para la base de datos de entrenamiento como para la base de datos de prueba, respecto del conjunto de datos inicial (en un ambos subconjuntos hay aproximadamente un 40 % de observaciones pertenecientes a la clase 0 y un 60 % de observaciones pertenecientes a la clase 1).

4.1.1. Batch Naïve Bayes

Se realizó la clasificación considerando procesamiento en lotes por medio de la función creada previamente en RStudio. En este caso, el clasificador se aplica sobre la base de datos del 70 % para la fase de entrenamiento del modelo, y luego, se clasificaron todas las observaciones de la base de datos de prueba del 30 % en un mismo instante de tiempo.

A continuación se presenta la matriz de confusión resultante del clasificador.

		Predicción		
		0	1	Total
Valor real	0	38.9635 %	0.9563 %	39.9198 %
	1	0.3957 %	59.6845 %	60.0802 %
	Total	39.3592 %	60.6408 %	100 %

Cuadro 4.1: Matriz de Confusión Batch Naïve Bayes, en términos porcentuales.

De acuerdo a la matriz de confusión, se puede notar que la cantidad de clasificaciones erróneas no supera el orden del 2 %, observándose alrededor de un 99 % de predicciones correctas.

4.1.2. Naïve Bayes e1071

Se implementó el clasificador Naïve Bayes que incorpora RStudio. Este comando está basado en procesamiento en lotes, de modo que la base de datos de entrenamiento se utilizó en la etapa de aprendizaje del modelo, mientras que la base de datos de prueba fue utilizada para la fase de predicción o clasificación del mismo.

En el Cuadro 4.2 se presenta la matriz de confusión resultante del clasificador.

		Predicción		
		0	1	Total
Valor real	0	38.9635 %	0.9563 %	39.9198 %
	1	0.3957 %	59.6845 %	60.0802 %
	Total	39.3592 %	60.6408 %	100 %

Cuadro 4.2: Matriz de Confusión Naïve Bayes e1071, en términos porcentuales.

Note que el clasificador Naïve Bayes e1071 presentó la misma matriz de confusión que el algoritmo Batch Naïve Bayes.

4.1.3. Online Naïve Bayes

El algoritmo Naïve Bayes para procesamiento en stream se implementó siguiendo los pasos explicados en la sección 3.3.3. Esto significa que con la base de datos de entrenamiento se obtienen por primera vez los parámetros del modelo (fase de aprendizaje). Luego, a medida que se van incorporando nuevas observaciones a la base de datos inicial (provenientes del conjunto de datos de prueba), estas se clasifican y posteriormente se actualizan los parámetros.

Así, cada vez que se agregan observaciones a la base de datos, se clasifican utilizando los parámetros actualizados de la iteración anterior. Para ello, se utilizó un valor $m = 750$, generando un total de 800 iteraciones (es decir, 800 veces se incorporaron nuevas observaciones a la base de datos).

A continuación se presenta la matriz de confusión resultante del clasificador en línea.

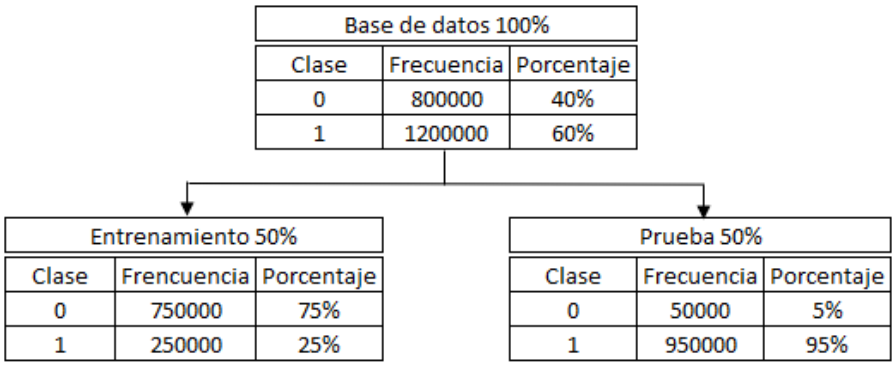
		Predicción		
		0	1	Total
Valor real	0	39.9627 %	0.9572 %	39.9198 %
	1	0.3957 %	59.6845 %	60.0802 %
	Total	39.3583 %	60.6417 %	100 %

Cuadro 4.3: Matriz de Confusión Online Naïve Bayes, en términos porcentuales.

De acuerdo a la matriz de confusión del Cuadro 4.3, se puede notar que el método en línea presentó resultados muy similares a los del método en batch.

Ahora, suponga que la base de datos simulada en un comienzo, fue dividida en 50 % para la base de datos de entrenamiento y 50 % para la base de datos de prueba.

Además, de manera particular, se planteó que dentro de la base de datos de entrenamiento, un 75 % de las observaciones deben pertenecer a la clase 0, mientras que el 25 % restante debe pertenecer a la clase 1. Las observaciones restantes se asignaron a la base de datos de prueba por muestreo aleatorio simple y en la Figura 4.2 se presenta dicha división.



Base de datos 100%

Clase	Frecuencia	Porcentaje
0	800000	40%
1	1200000	60%

Entrenamiento 50%

Clase	Frencuencia	Porcentaje
0	750000	75%
1	250000	25%

Prueba 50%

Clase	Frecuencia	Porcentaje
0	50000	5%
1	950000	95%

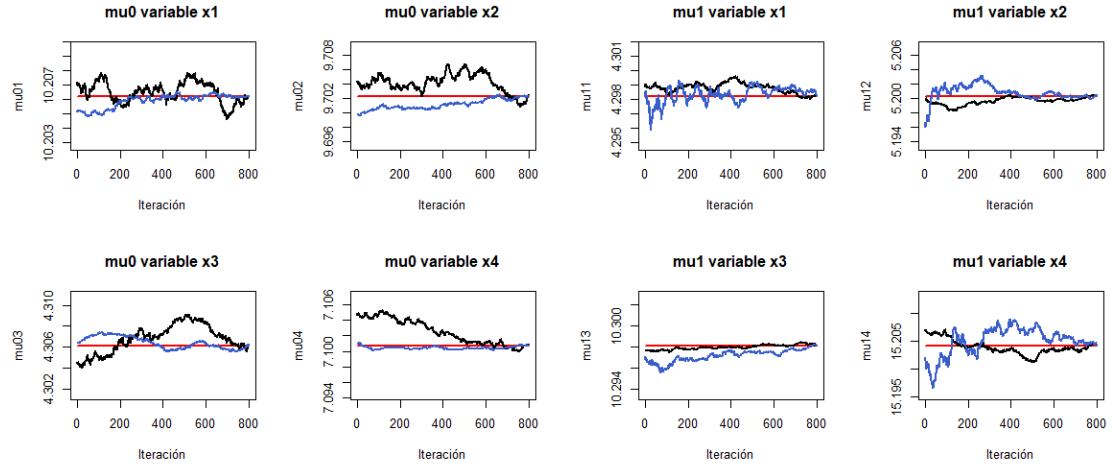
Figura 4.2: División de la base de datos en entrenamiento y prueba.

Bajo estas condiciones, se implementó nuevamente el método Naïve Bayes considerando procesamiento en stream.

En este caso, la fase de aprendizaje del modelo se llevó a cabo con la base de datos de entrenamiento (que ahora es del 50 %), obteniéndose la estimación de parámetros por primera vez. Por otra parte, la etapa de clasificación se realizó considerando un valor $m = 1250$, lo que generó un total de 800 iteraciones; esto significa que 800 veces se incorporaron nuevas observaciones a la base de datos inicial, en grupos de 1250 tuplas.

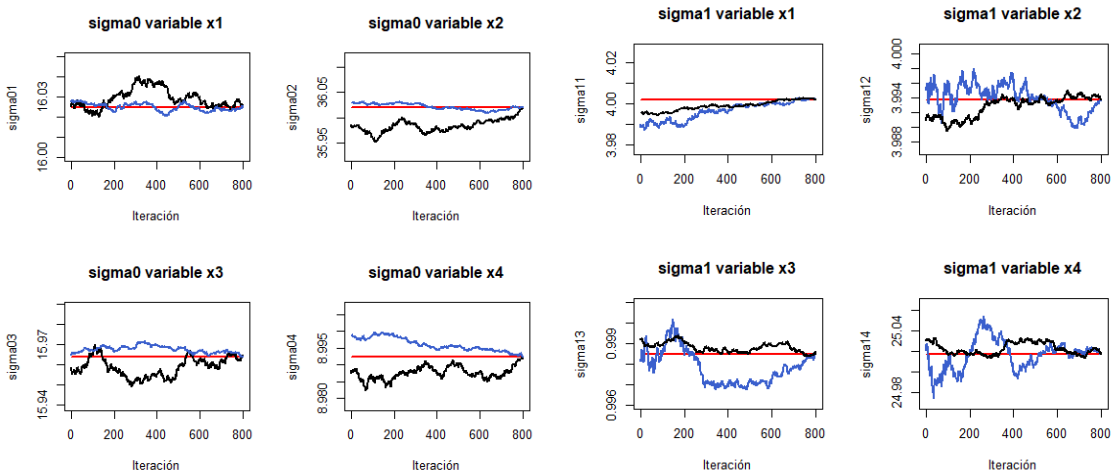
Así, cada vez que se agregaron nuevas tuplas a la base de datos inicial, estas se clasificaron y posteriormente los parámetros fueron actualizados. De esta forma, por cada iteración del data stream, se tiene una estimación de cada parámetro.

En la Figura 4.3 se presentan las estimaciones de cada parámetro en función de la iteración para las 4 variables explicativas. En la gráfica, las curvas de color negro representan el valor del parámetro en stream considerando una división de la data en 70 % y 30 % para la base de datos de entrenamiento y prueba, respectivamente. Por otra parte, las curvas de color azul representan las estimaciones de cada parámetro en función de la iteración para el método en línea, pero ahora considerando la división de la data en 50 % y 50 %. Por último, la recta de color rojo es la estimación de cada parámetro para el caso en batch.



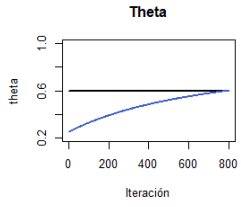
(a) Parámetro μ_{0k}

(b) Parámetro μ_{1k}



(c) Parámetro σ^2_{0k}

(d) Parámetro $\sigma 1k^2$



(e) Parámetro θ

Figura 4.3: Estimación parámetros en función de la iteración para cada variable.

De acuerdo a la gráfica, es de interés señalar que en ambos casos los parámetros del método en línea convergieron a la estimación de parámetros del caso en lotes.

En particular, se pudo notar que en el caso de los parámetros de la clase 0 (μ_{0k} y σ^2_{0k}), la curva azul comienza a converger antes que la curva de color negro. Por otra parte, para aquellos parámetros pertenecientes a la clase 1 (μ_{1k} y σ^2_{1k}), la curva de color negro comienza a converger antes que la curva de color azul. Esto se puede deber a que en el segundo caso (color azul), dentro de la base de datos de entrenamiento un 90 % de las observaciones corresponden a la clase 0, por ende se tiene una mejor aproximación a parámetros pertenecientes a esa clase.

Respecto del parámetro θ, se observó que ambos métodos convergieron al valor del caso en lotes, sin embargo, el primer caso (curva negra) converge mucho antes que el segundo caso (curva azul). Esto también se puede explicar por el hecho de que en el segundo caso la mayoría de las observaciones de la clase 0 se encuentran en la base de datos de entrenamiento, provocando que en las primeras iteraciones el parámetro esté más alejado.

4.1.4. Comparación clasificadores

Los tres clasificadores fueron comparados en términos de rendimiento y tiempo de ejecución.

Rendimiento

De acuerdo a los resultados arrojados por los tres clasificadores, en el Cuadro 4.4 se presentan las medidas de evaluación.

Métrica	Batch NB	e1071 NB	Online NB
Accuracy	0.9865	0.9865	0.9865
Precisión	0.9842	0.9842	0.9842
Sensibilidad	0.9934	0.9934	0.9934
F1	0.9888	0.9888	0.9888

Cuadro 4.4: Métricas de evaluación considerando n=2000000.

En ese sentido, se pudo observar que los tres clasificadores presentaron el mismo rendimiento. También se analizó el rendimiento considerando distintos tamaños muestrales, donde se pudo notar que los tres clasificadores presentan valores muy similares de las cuatro métricas evaluadas, independiente del tamaño muestral (véase Anexo Cuadro 6.1).

Tiempo

Para analizar el tiempo de ejecución de los clasificadores se realizó un total de 10 simulaciones por cada uno. En el Cuadro 4.5 se presenta el tiempo promedio en segundos y su desviación estándar.

	Batch NB		e1071 NB		Online NB	
N	Media	D. Estándar	Media	D. Estándar	Media	D. Estándar
500	0.1200	0.0249	0.0630	0.0164	0.0440	0.0272
1000	0.1220	0.0123	0.1240	0.0802	0.0750	0.0629
3000	0.1360	0.0357	0.3490	0.0500	0.0780	0.0394
5000	0.1510	0.0493	0.5430	0.0371	0.0850	0.0460
10000	0.1470	0.0206	1.1150	0.0538	0.0900	0.0365
50000	0.2050	0.0626	5.6580	0.1281	0.1920	0.0469
100000	0.2670	0.0903	11.7250	0.2653	0.2530	0.0488
500000	0.9620	0.2279	61.3170	2.0557	0.8720	0.0900
1000000	1.8500	0.5752	123.1080	4.5225	1.7360	0.2421
2000000	4.3480	0.5033	249.3170	7.3623	3.1560	0.1251

Cuadro 4.5: Tiempo (segundos) en función del tamaño muestral.

En ese sentido, se puede apreciar que para todos los tamaños muestrales, el clasificador que presenta un menor tiempo de ejecución es el que utiliza procesamiento en stream (Online Naïve Bayes), seguido de la función creada para procesamiento en batch; y finalmente, el algoritmo Naïve Bayes e1071 es el que presenta un mayor tiempo de ejecución.

Lo mismo se puede ver en el gráfico de la Figura 4.4, donde además se puede notar que mientras mayor es el tamaño de la muestra utilizada, mayor es la diferencia entre los tiempos de cada clasificador.

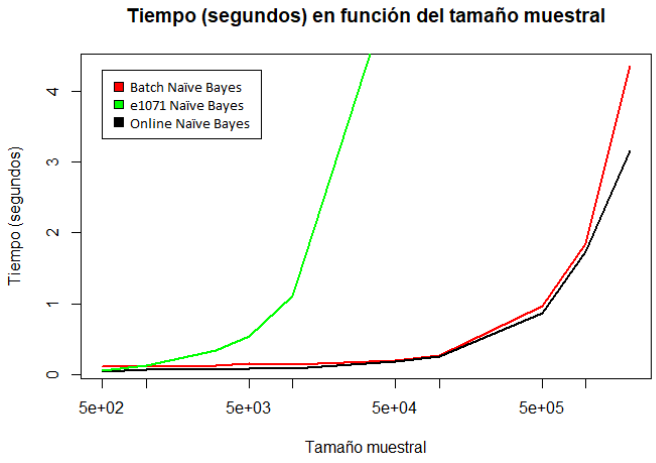


Figura 4.4: Tiempo (segundos) en función de tamaño muestral.

Además, note que el algoritmo Naïve Bayes e1071 supera el segundo de ejecución con un tamaño muestral de aproximadamente 10000 observaciones, mientras que en el caso de los métodos creados para este proyecto, el segundo de ejecución se supera con un tamaño muestral de aproximadamente 700000 observaciones.

4.2. Conjunto de datos astronómicos

Los datos corresponden a un subconjunto de 18.809 observaciones y 13 variables que fueron extraídas de un modelo armónico. En particular, está compuesta por la variable respuesta "RR Lyrae (RRL)" y 12 variables independientes que fueron seleccionadas por Elorrieta et al. (2016) como las más importantes, de un total de 68 variables, mediante análisis del rendimiento por validación cruzada en el que se utilizaron las medidas de calidad *Precisión*, *Recall*, *F1* y *Área bajo la curva (AUC)*.

4.2.1. Descripción de variables y subconjuntos

A continuación se presenta el listado de variables que componen la base de datos:

N	Variable*	Descripción
1	rrl	1: pertenece a la clase, 0: no pertenece a la clase
2	f_1	Frecuencia
3	ϕ_{12}	Fase de la 1ra frecuencia y 2da armónica
4	ϕ_{13}	Fase de la 1ra frecuencia y 3ra armónica
5	A_{11}	Amplitud de la 1ra frecuencia y 1ra armónica
6	p2p_scatter_2praw	$\sum_{i=2}^{N}(y_{2P,i+1} - y_{2P,i})^2 / \sum_{i=2}^{N}(y_{i+1} - y_i)^2$
7	$P1$	Cima en el periodograma de la 1ra frecuencia
8	$R1$	$(\phi_{max,1} - \phi_{min,1})/(\phi_{min,1} - \phi_{max,2})$ (for $y_{s,2P}$)
9	skew	Asimetría de y
10	A_{12}	Amplitud de la 1ra frecuencia y 2da armónica
11	A_{13}	Amplitud de la 1ra frecuencia y 3ra armónica
12	ϕ_{14}	Fase de la 1ra frecuencia y 4ta armónica
13	p2p_scatter_pfold_over_mad	$\sum_{i=2}^{N} \lvert y_{P,i+1} - y_{P,i}\rvert/(N-1)MAD(y)$

Cuadro 4.6: Listado de variables.
(*) Para mayor información acerca de las variables, ver Elorrieta et al. (2016).

La base de datos fue dividida en dos subconjuntos: 70 % correspondiente a la base de datos de entrenamiento, los cuales se utilizaron en la fase de aprendizaje del modelo; y el 30 % restante corresponde a la base de datos de prueba, que se utilizó en la etapa de clasificación del mismo.

En la Figura 4.5 se presenta la división de la base de datos.

Base de datos 100%		
Clase	Frecuencia	Porcentaje
0	16530	87.8835%
1	2279	12.1165%

Entrenamiento 70%		
Clase	Frecuencia	Porcentaje
0	11595	88.0678%
1	1571	11.9323%

Prueba 30%		
Clase	Frecuencia	Porcentaje
0	4935	87.4535%
1	708	12.5465%

Figura 4.5: División de la base de datos en entrenamiento y prueba.

Note que para ambas submuestras generadas, la proporción de observaciones que no son del tipo RR Lyrae (clase 0) se aproxima al 87.9 % de la base de datos inicial y lo mismo sucede con observaciones que sí son del tipo RRL (clase 1), donde ambas submuestras se aproximan al 12.1 %.

4.2.2. Umbral de Clasificación

La variable respuesta de este conjunto de datos está considerablemente desbalanceada (pues solo un 12.12 % de las observaciones pertenece a la clase $rrl = 1$), por lo tanto, es necesario escoger el umbral adecuado que permita clasificar las observaciones de manera óptima[1]. Para esto, se obtuvieron las medidas accuracy y F1 con distintos umbrales de clasificación.

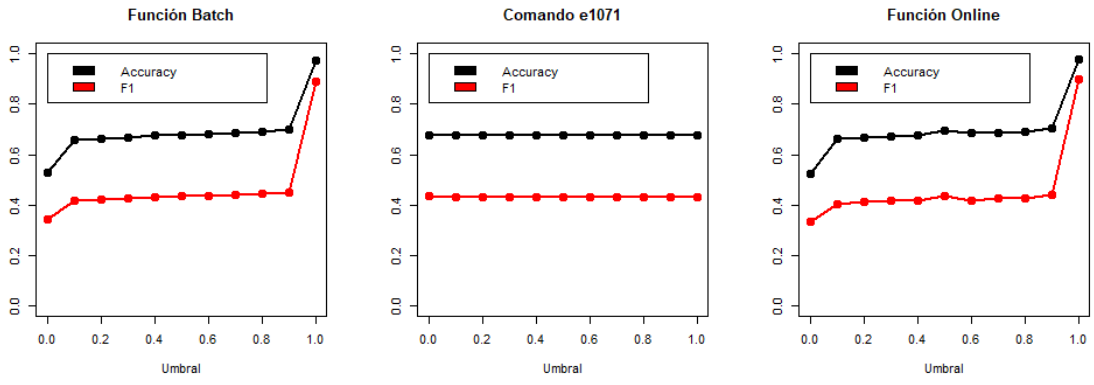


Figura 4.6: Accuracy y F1 en función del Umbral de Clasificación*.
* *Véase tabla de valores en Anexo Cuadro 6.2.*

[1]Generalmente se utiliza como umbral el valor 0.5, pero en algunas ocasiones este valor no es el óptimo y deben emplearse métodos más sofisticados para determinar el umbral de clasificación correcto (X. Wu et. al, 2007).

De acuerdo a la Figura 4.6, el umbral de clasificación óptimo estaría entre los valores 0.95 y 1 (≈ 0.99). El hecho de que sea un valor tan cercano a 1 se puede explicar mediante el gráfico de la figura 4.7, donde no solo las observaciones de la clase $rrl = 1$ (color rojo) presentan valores de $P(I = 1|X)$ muy cercanos a 1, sino que también se observaron muchas tuplas pertenecientes a la clase $rrl = 0$ (color negro) con valores de $P(I = 1|X)$ muy altos. Por tanto, se consideró como umbral de clasificación el valor 0.99.

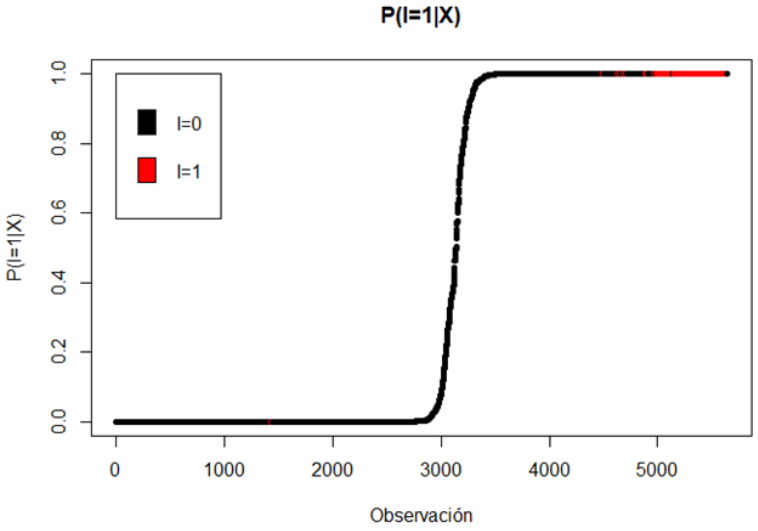


Figura 4.7: Probabilidad de pertenencia a la clase 1 ordenada de menor a mayor.

4.2.3. Batch Naïve Bayes

Se aplicó el clasificador creado para procesamiento en lotes con el conjunto de datos astronómico, de tal forma que la base de datos de entrenamiento se utilizó para la fase de aprendizaje del modelo, y luego, se clasificaron las observaciones de la base de datos de prueba.

En la Figura 4.7 se presenta la matriz de confusión resultante de la base de datos de prueba.

		Predicción		
		0	1	Total
Valor real	0	85.6637 %	1.7898 %	87.4535 %
	1	1.0810 %	11.4655 %	12.5465 %
	Total	86.7446 %	13.2554 %	100 %

Cuadro 4.7: Matriz de Confusión Batch Naïve Bayes, en términos porcentuales.

De acuerdo a la matriz de confusión resultante del clasificador en batch, se puede señalar que las

observaciones que fueron clasificadas erróneamente (falsos positivos y falsos negativos) no superan el orden del 3 %.

4.2.4. e1071 Naïve Bayes

El clasificador Naïve Bayes de la librería e1071 se implementó en el conjunto de datos astronómico. A continuación se presenta la matriz de confusión resultante.

		Predicción		
		0	1	Total
Valor real	0	55.2366 %	32.2169 %	87.4535 %
	1	0.1772 %	12.3693 %	12.5465 %
	Total	55.4138 %	44.5862 %	100 %

Cuadro 4.8: Matriz de Confusión e1071 Naïve Bayes, en términos porcentuales.

Con este clasificador, la cantidad de falsos positivos asciende a un 32 % aproximadamente y las observaciones verdaderas negativas disminuyen a un 55 % aproximadamente. Por lo tanto, con este clasificador se espera que las métricas de evaluación reflejen un desempeño inferior, respecto del método en batch.

4.2.5. Online Naïve Bayes

El clasificador Naïve Bayes para procesamiento en stream se implementó siguiendo los pasos señalados en la sección 3.3.3. De esta forma, el modelo se entrenó con la base de datos del 70 % y se obtuvo la estimación de parámetros por primera vez. Luego, a medida que se incorporaron nuevas observaciones a la base de datos, estas fueron clasificadas y los parámetros actualizados.

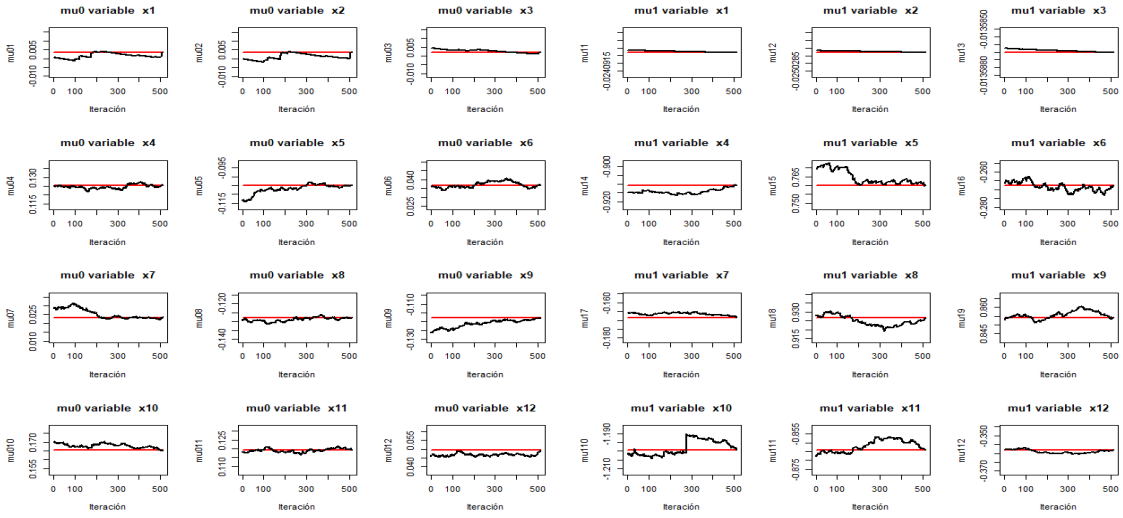
Por lo tanto, cada vez que se agregaron nuevas observaciones a la base de datos, se obtuvo una nueva estimación de cada parámetro. Para esto, se utilizó un valor $m = 11$, lo que generó un total de 513 iteraciones en el data stream (es decir, 513 veces se incorporaron nuevas observaciones a la base de datos, en grupos de 11 tuplas).

En el Cuadro 4.9 se presenta la matriz de confusión resultante.

		Predicción		
		0	1	Total
Valor real	0	86.3902 %	1.8607 %	88.2509 %
	1	0.8506 %	10.8985 %	11.7491 %
	Total	87.2408 %	12.7592 %	100 %

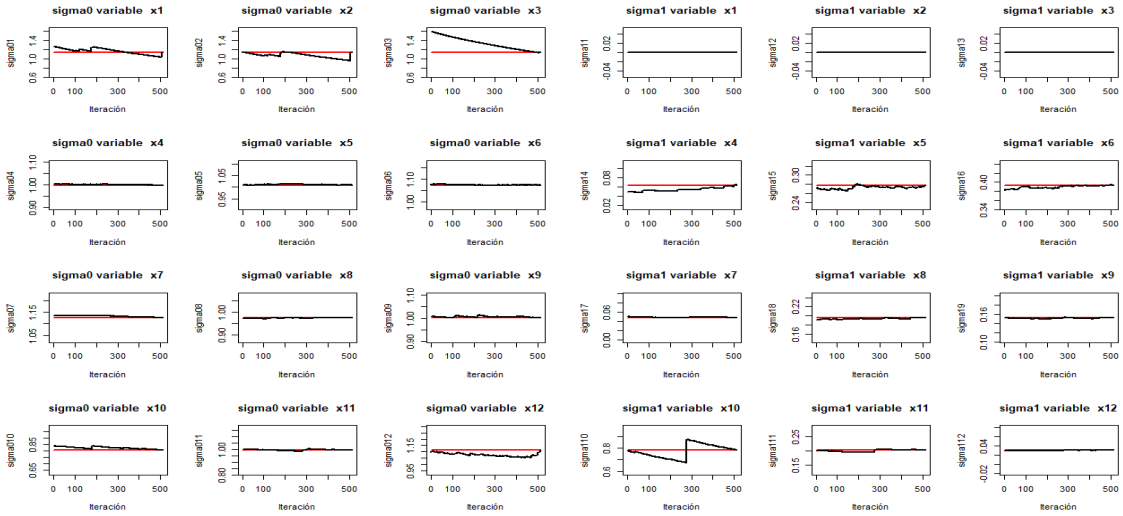
Cuadro 4.9: Matriz de Confusión Online Naïve Bayes, en términos porcentuales.

Asimismo, las estimaciones de cada parámetro para el caso en línea convergen al caso en batch. Esto se puede apreciar la siguiente gráfica (Figura 4.8), donde las curvas de color negro representan las estimaciones de cada parámetro en función de la iteración para un data stream, mientras que la recta de color rojo es la estimación de cada parámetro en el caso en batch.



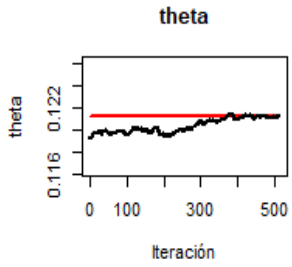
(a) Parámetro μ_{0k}

(b) Parámetro μ_{1k}



(c) Parámetro σ^2_{0k}

(d) Parámetro $\sigma 1k^2$



(e) Parámetro θ

Figura 4.8: Estimación parámetros en función de la iteración para cada variable.

4.2.6. Comparación clasificadores

Los tres clasificadores fueron comparados en términos de rendimiento y tiempo de ejecución.

Rendimiento

De acuerdo a los resultados arrojados por cada clasificador al ser implementado en los datos astronómicos, en el Cuadro 4.10 se presentan las métricas de evaluación.

Métrica	Batch NB	e1071 NB	Online NB
Accuracy	0.9713	0.6761	0.9729
Precisión	0.8650	0.2774	0.8542
Sensibilidad	0.9138	0.9859	0.9276
F1	0.8887	0.4330	0.8894

Cuadro 4.10: Métricas de evaluación para conjunto de datos astronómicos.

En primer lugar, tres de las cuatro medidas de evaluación indicaron que el rendimiento del clasificador Naïve Bayes e1071 es inferior al de los dos clasificadores creados para este proyecto. También, en términos generales, el rendimiento del clasificador online presentó el mejor rendimiento entre los tres clasificadores evaluados (a pesar de que su diferencia con la función en batch es mínima).

Tiempo

Cada método se implementó 10 veces y se obtuvo el tiempo de ejecución con los datos astronómicos. En el Cuadro 4.11 se presenta el tiempo promedio de ejecución y la desviación estándar de cada clasificador.

	Batch NB	e1071 NB	Online NB
Promedio	0.2170	3.6820	0.1730
D. Estándar	0.0460	0.1762	0.0333

Cuadro 4.11: Tiempo promedio (segundos) y desviación estándar considerando 10 simulaciones*.
**Tabla con los tiempos de cada simulación se encuentran en Anexo Cuadro 6.3.*

En promedio, el clasificador que presentó un menor tiempo de ejecución es Online Naïve Bayes, seguido del algoritmo Batch Naïve Bayes y, finalmente, el método Naïve Bayes e1071 presentó un mayor tiempo de ejecución con el conjunto de datos astronómicos.

Además, note que en el caso de ambas funciones creadas, el tiempo promedio de ejecución no superó medio segundo, mientras que el método Naïve Bayes e1071 tardó 3.7 segundos aproximadamente.

Capítulo 5

Resultados variables Bernoulli

A lo largo de este capítulo se presentan los resultados de la implementación del método Naïve Bayes con variables independientes del tipo Bernoulli, comparándose las funciones creadas para el caso con procesamiento en batch y procesamiento en stream, frente al comando *naiveBayes* perteneciente a la librería e1071 de RStudio.

5.1. Conjunto de datos simulado

Se generó una base de datos de 1 millón de observaciones y 7 variables del tipo Bernoulli (véase anexo Figura 6.2), usando los siguientes valores:

- θ =0.45
- θ_{0k} =(0.90 0.23 0.05 0.88 0.83 0.45 0.25)
- θ_{1k} =(0.23 0.82 0.73 0.22 0.18 0.85 0.74)

Además, como se explicó en la sección 3.3.3, el conjunto de datos simulado se dividió en dos subconjuntos: base de datos de entrenamiento, correspondiente al 70 % de las observaciones (utilizado en la fase de aprendizaje del modelo) y base de datos de prueba, que corresponde al 30 % restante y se utilizó en la etapa de clasificación del modelo.

En la figura 5.1 se presenta la división del conjunto de datos simulado.

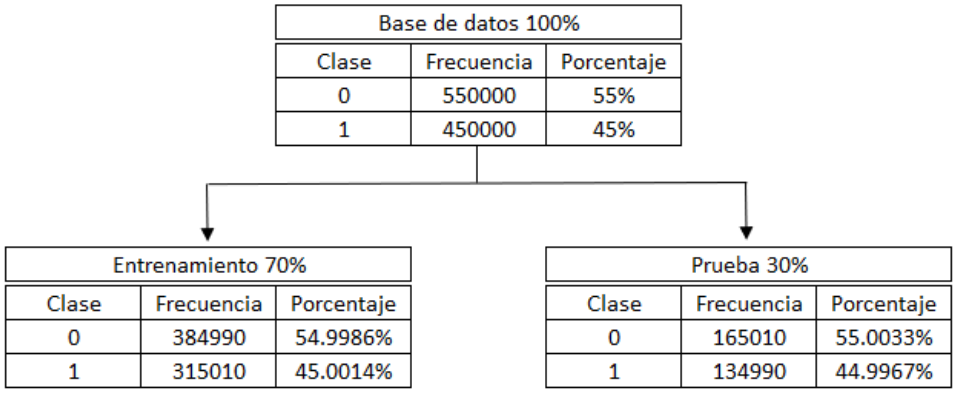


Figura 5.1: División de la base de datos en entrenamiento y prueba.

Note que en ambas submuestras, la proporción de observaciones pertenecientes a la clase 0 se aproxima al 55 % de la base de datos inicial; y para la clase 1 sucede lo mismo, pues ambos subconjuntos se aproximan a un 45 % de observaciones.

5.1.1. Batch Naïve Bayes

Se implementó la fución en batch en el conjunto de datos simulado con variables explicativas Bernoulli. De modo que la base de datos del 70 % se utilizó para la fase de entrenamiento, y luego, con la base de datos de prueba se implementó la etapa de clasificación.

En el Cuadro 5.1 se presenta la matriz de confusión resultante.

		Predicción		
		0	1	Total
Valor real	0	53.4463 %	1.5120 %	54.9583 %
	1	1.5167 %	43.5250 %	45.0417 %
	Total	54.9630 %	45.0370 %	100 %

Cuadro 5.1: Matriz de Confusión Batch Naïve Bayes, en términos porcentuales.

Se puede notar que la cantidad de observaciones mal clasificadas no superan el 4 %, observándose alrededor de un 97 % de observaciones correctamente clasificadas.

5.1.2. e1071 Naïve Bayes

Se implementó el algoritmo Naïve Bayes de la librería e1071, que utiliza procesamiento en batch. Con la base de datos del 70 % se entrenó el modelo y con la base de datos de prueba, se clasificaron las observaciones.

Enseguida se presenta la matriz de confusión resultante.

		Predicción		
		0	1	Total
Valor real	0	52.3150 %	2.6433 %	54.9583 %
	1	1.3140 %	43.7277 %	45.0417 %
	Total	53.6290 %	46.3710 %	100 %

Cuadro 5.2: Matriz de Confusión Naïve Bayes e1071, en términos porcentuales.

En este caso, la cantidad de observaciones falsas positivas asciende a 2.6 % y las observaciones falsas negativas disminuyeron a un 1.31 % aproximadamente. De esta forma, se observó que un 96 % de las observaciones fueron clasificadas correctamente.

5.1.3. Online Naïve Bayes

Se implementó la función creada del algoritmo Naïve Bayes para procesamiento en stream como se explicó en la sección 3.3.3. Por lo tanto, en la fase de entrenamiento del modelo, se obtuvo la estimación de los parámetros por primera vez. Luego, para simular un data stream, se incorporaron observaciones a la base de datos inicial en pequeños grupos, de modo que cada vez que se agregaron tuplas, estas fueron clasificadas y posteriormente los parámetros se actualizaron.

Así, la próxima vez que lleguen nuevas observaciones a la base de datos, estas se deben clasificar utilizando los parámetros actualizados en la iteración anterior. En este caso, se consideró un valor $m = 750$, lo que generó un total de 400 iteraciones en el procesamiento en stream (es decir, 400 veces se incorporaron nuevas tuplas a la base de datos, en grupos de 750).

En el Cuadro 5.3 continuación se presenta la matriz de confusión resultante.

		Predicción		
		0	1	Total
Valor real	0	53.4463 %	1.5120 %	54.9583 %
	1	1.5167 %	43.5250 %	45.0417 %
	Total	54.9630 %	45.0370 %	100 %

Cuadro 5.3: Matriz de Confusión Online Naïve Bayes, en términos porcentuales.

De acuerdo a la matriz de confusión resultante, es de interés señalar que se obtuvieron exactamente los mismos resultados que en el caso del método Batch Naïve Bayes.

Ahora, suponga que la base de datos que se simuló en un comienzo es dividida en un 50 % para la base de datos de entrenamiento y el 50 % restante corresponde a la base de datos de prueba.

En particular, se planteó que dentro de la base de datos de entrenamiento, un 90 % de las observaciones deben pertenecer a la clase 0 y el 10 % a la clase 1. El resto de observaciones se asignó a la base de datos de prueba por medio de muestreo aleatorio simple, donde se observó que un 20 % de las observaciones pertenece a la clase 1 y un 80 % a la clase 0.

En la Figura 5.2 se presenta la división de la base de datos.

Base de datos 100%		
Clase	Frecuencia	Porcentaje
0	550000	55%
1	450000	45%

Entrenamiento 50%		
Clase	Frencuencia	Porcentaje
0	450000	90%
1	50000	10%

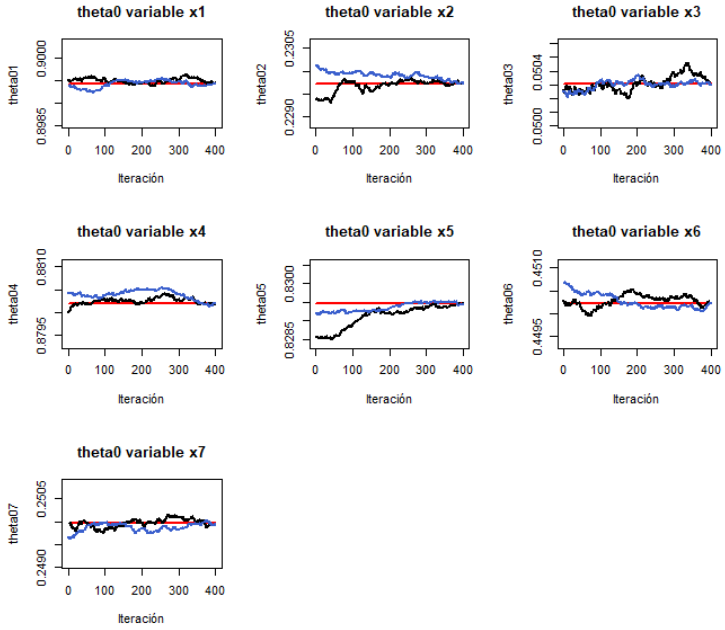
Prueba 50%		
Clase	Frecuencia	Porcentaje
0	100000	20%
1	400000	80%

Figura 5.2: División de la base de datos en entrenamiento y prueba.

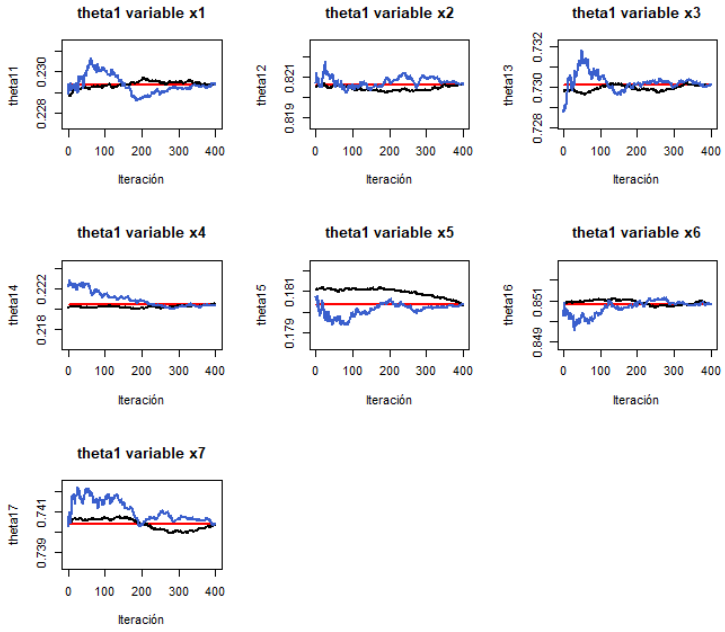
Bajo esas condiciones se implementó el algoritmo Naïve Bayes para procesamiento en stream. De esta forma, con la base de datos de entrenamiento (que ahora es del 50 %) se llevó a cabo la etapa de aprendizaje del modelo, donde se obtuvieron las estimaciones de cada parámetro por primera vez. Posteriormente, con la base de datos de prueba se simuló un data stream considerando un valor $m = 1250$, lo que generó un total de 400 iteraciones (esto significa que, en total, 400 veces se incorporaron nuevas observaciones a la base de datos, en grupos de 1250).

Así, cada vez que se agregaron nuevas observaciones a la base de datos, estas se clasificaron utilizando los parámetros actualizados de la iteración anterior.

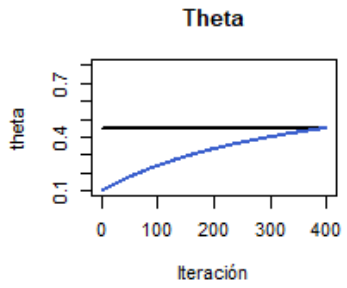
En la Figura 5.3 se pueden observar las estimaciones de cada parámetro en función de la iteración, para las 7 variables explicativas Benoulli. En la gráfica, las curvas de color negro representan la estimación del parámetro considerando una división de la base de datos en 70 % para entrenamiento y 30 % para prueba. Por otra parte, la curva de color azul representa la estimación de cada parámetro en el método en línea, pero considerando una división de la data en 50 % para la base de datos de entrenamiento y el 50 % restante para la base de datos de prueba. Por último, la recta de color rojo representa la cada parámetro estimado bajo procesamiento en batch.



(a) Parámetro θ_{0k}



(b) Parámetro θ_{1k}



(c) Parámetro θ

Figura 5.3: Estimación de parámetros en función de la iteración en base de datos con variables Bernoulli.

De acuerdo a los gráficos de cada parámetro, en primer lugar se puede señalar que en ambos casos los parámetros del método en línea convergen a las estimaciones del método en lotes.

En particular, en el caso del parámetro θ_{0k} la curva de color azul (división 50 %-50 %) comienza a converger antes que la curva de color negro (división 70 %-30 %), mientras que en el caso del parámetro θ_{1k} sucede lo contrario, es decir, la curva que converge antes es la de color negro. Esto se puede explicar porque en el segundo caso (división 50 %-50 %) la base de datos de entrenamiento contempla un 90 % de observaciones de la clase 0, lo que hace que en parámetros de dicha clase las estimaciones sean más precisas desde las primeras iteraciones (y por ende, en parámetros de la clase 1, las estimaciones sean menos exactas en las primeras iteraciones).

Asimismo, se observó que las estimaciones del parámetro θ comienzan a converger desde las primeras iteraciones en la curva de color negro, que corresponde a la división 70 %-30 % (de hecho, las variaciones respecto a la estimación del caso batch es tan mínima, que la curva de color negro está superpuesta en la línea roja). No sucede así con la curva de color azul, que en las primeras iteraciones presenta estimaciones bastantes alejadas, pero finalmente converge al valor en lotes.

5.1.4. Comparación clasificadores

Los clasificadores fueron comparados en términos de rendimiento y tiempo de ejecución.

Rendimiento

En el cuadro 5.4 se presentan las medidas de evaluación obtenidas mediante la función creada para procesamiento en batch, el comando *naiveBayes* de la librería e1071 y la función creada para procesamiento en stream.

Métrica	Batch NB	e1071 NB	Online NB
Accuracy	0.9697	0.9604	0.9697
Precisión	0.9664	0.9430	0.9664
Sensibilidad	0.9663	0.9708	0.9663
F1	0.9664	0.9567	0.9664

Cuadro 5.4: Métricas de evaluación considerando n=1000000.

De acuerdo a los resultados, los tres métodos presentaron buen rendimiento (con valores superiores al 95 %) y note que los algoritmos creados para este proyecto presentaron los mismos valores en las cuatro métricas evaluadas. En particular, el clasificador Naïve Bayes e1071 fue el que mostró un peor rendimiento respecto a los dos restantes en 3 de las 4 métricas utilizadas.

Además, se evaluó el rendimiento considerando distintos tamaños muestrales, obteniéndose resultados muy similares a los presentados en el Cuadro 5.4 (tabla de valores para cada tamaño muestral se encuentra en Anexo Cuadro 6.4).

Tiempo

Se evaluó el tiempo de ejecución en función del tamaño muestral en los tres clasificadores considerando un total de 10 simulaciones para cada caso. Así, en el Cuadro 5.5 se muestra el tiempo promedio para cada caso y su desviación estándar. En promedio, el método Online Naïve Bayes presentó un menor tiempo de ejecución, seguido del del método Batch Naïve Bayes, y por último, el algoritmo Naïve Bayes e1071 presentó el mayor tiempo de ejecución medio para todos los tamaños muestrales evaluados.

	Batch NB		e1071 NB		Online NB	
N	Media	D. Estándar	Media	D. Estándar	Media	D. Estándar
500	0.0320	0.0103	0.0980	0.0199	0.0200	0.0216
1000	0.0380	0.0092	0.1510	0.0160	0.0230	0.0170
3000	0.0420	0.0199	0.4890	0.0197	0.0260	0.0255
5000	0.0340	0.0070	0.8460	0.0378	0.0310	0.0242
10000	0.0510	0.0228	1.7350	0.1110	0.0450	0.0369
50000	0.1190	0.0145	8.6950	0.2515	0.1060	0.0196
100000	0.2410	0.1088	18.142	0.4267	0.2310	0.0223
500000	1.5680	0.0874	74.0910	2.1626	1.2070	0.0727
1000000	3.3170	0.1958	150.7730	2.2500	2.4190	0.0936
2000000	7.3670	0.3536	314.5560	1.4110	5.0860	0.6837

Cuadro 5.5: Tiempo (segundos) en función del tamaño muestral.

Lo mismo se observa en la Figura 5.4, donde a partir de las 100000 observaciones aproximadamente, la diferencia de tiempos entre las funciones en lotes y en línea se hace más visible; y por consiguiente, mientras mayor es el tamaño muestral, más evidente es la diferencia de tiempos de ejecución.

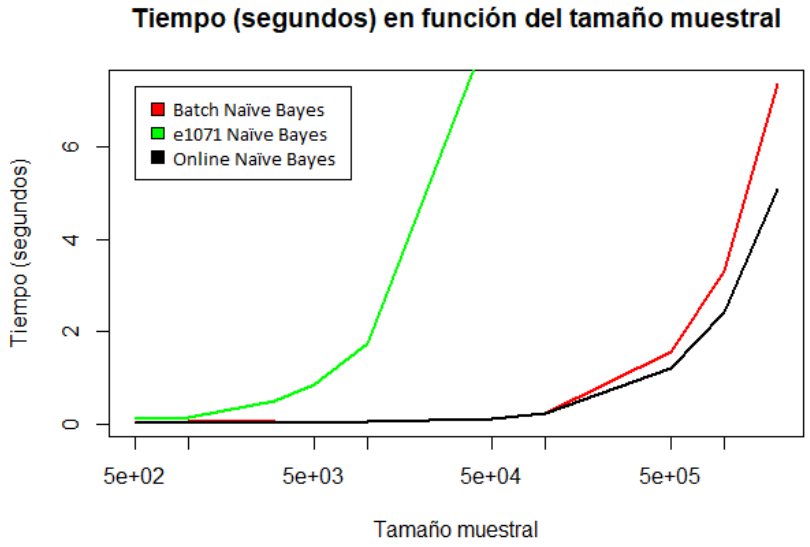


Figura 5.4: Tiempo (segundos) en función de tamaño muestral con variables explicativas Bernoulli.

Además, resulta interesante destacar que en el caso del método Naïve Bayes e1071, el tiempo de ejecución supera 1 segundo aproximadamente con un tamaño muestral de 7000 observaciones. Por otra parte, los métodos Naïve Bayes Batch y Online, se supera 1 segundo de ejecución con un tamaño muestral de alrededor de 250000 observaciones.

5.2. Conjunto de datos Twitter

La base de datos corresponde a un conjunto de tweets extraídos de la red social Twitter (véase anexo Figura 6.5), con el objeto de ser clasificados de acuerdo a su sentimiento, que puede ser positivo (clase 1) o negativo (clase 0).

5.2.1. Transformación base de datos a variables Bernoulli

Antes de realizar la clasificación, se realizó una transformación de la base de datos de tal forma que tuviera variables explicativas del tipo Bernoulli.

Para ello, en primer lugar, se realizó una limpieza de la base de datos, quitando signos de puntuación, números, etcétera.

Luego, se realizó una selección de las palabras más utilizadas en los tweets dependiendo del sentimiento (0 o 1), sin embargo, se observó que los términos más frecuentes corresponden a artículos, pronombres, preposiciones, etcétera; en general, palabras que no aportan información relevante sobre el texto en sí. Estas palabras son conocidas como "stopwords" y se generó una lista de 197 términos[1] que fueron descartados.

Por tanto, en la siguiente imagen se muestra una nube de palabras con aquellos términos que son más frecuentes por cada clase.

(a) Sentimiento Negativo (clase 0) (b) Sentimiento Positivo (clase 1)

Figura 5.5: Nube de palabras más frecuentes por cada sentimiento*.
**Frecuencias se encuentra en Anexo Figura 6.6.*

[1] Entre los cuales se encuentran las palabras: i, me, my, myself, we, our, you, your, he, him, she, their, who, did, do, be, have, they, of, but, etcétera.

Es de interés hacer énfasis en lo descriptiva que es la Figura 5.5 en relación con las palabras más frecuentes de cada tipo de sentimiento. De tal modo, que en la clase 0 (sentimiento negativo) se observan palabras como sorry, work, miss, sad, bad, entre otras; y en la clase 1 (sentimiento positivo) se presentan palabras como thanks, love, well, fun, happy, etc.

En esa línea, se seleccionaron 38 palabras que fueron utilizadas como variables explicativas del tipo Bernoulli. El listado de palabras se presenta en el Cuadro 5.6.

sorry	work	miss	sad	still	want	wish	bad	need	hate
sucks	feel	ok	didn	lol	going	well	yeah	hope	im
back	thanks	love	haha	great	thank	happy	awesome	fun	much
nice	new	friends	night	hey	sure	best	better		

Cuadro 5.6: Listado de palabras más frecuentes.

5.2.2. Descripción de variables y subconjuntos

La base de datos final contempla 52735 observaciones y 39 variables: la variable respuesta, que indica el sentimiento del tweet; y 38 variables explicativas del tipo bernoulli, que indican si cierta palabra está o no presente en el tweet (véase Anexo Figura 6.7).

A continuación se presenta la descripción de variables que componen la base de datos.

N	Variable	Descripción
1	y	1: sentimiento positivo, 0: sentimiento negativo
2	sorry	1: está en el tweet, 0: no está en el tweet
3	work	1: está en el tweet, 0: no está en el tweet
4	miss	1: está en el tweet, 0: no está en el tweet
5	sad	1: está en el tweet, 0: no está en el tweet
...		
39	better	1: está en el tweet, 0: no está en el tweet

Cuadro 5.7: Listado de variables.

La base de datos fue dividida en dos subconjuntos: 70 % de las observaciones se asignaron a la base de datos de entrenamiento (que se utiliza en la fase de aprendizaje del modelo); y el 30 % restante se asigno a la base de datos de prueba, con la que se desarrolló la etapa de clasificación de modelo.

En la Figura 5.6 se presenta la división de la base de datos en entrenamiento y prueba.

Base de datos 100%		
Clase	Frecuencia	Porcentaje
0	23303	44.1889%
1	29432	55.8111%

Entrenamiento 70%		
Clase	Frencuencia	Porcentaje
0	16299	44.1540%
1	20615	55.8460%

Prueba 30%		
Clase	Frecuencia	Porcentaje
0	7004	44.2703%
1	8817	55.7297%

Figura 5.6: División de la base de datos en entrenamiento y prueba.

Note que en ambos subconjuntos la proporción de observaciones pertenecientes a la clase 0 (sentimiento negativo) se aproxima a un 44.1 % y la proporción de tuplas pertenecientes a la clase 1 (sentimiento positivo) se aproxima al 55.8 % de la base de datos inicial.

5.2.3. Umbral de clasificación

Como se explicó para el conjunto de datos astronómico, es necesario determinar un umbral de clasificación con el cual comparar el puntaje o score de cada observación. Generalmente se utiliza como umbral de clasificación el valor 0.5, no obstante, en algunas ocasiones este valor no es el óptimo y se deben utilizar métodos para detectar aquel umbral que permita obtener la clasificación óptima. Por lo tanto, de acuerdo a distintos umbrales de clasificación se obtuvieron las medidas accuracy y F1.

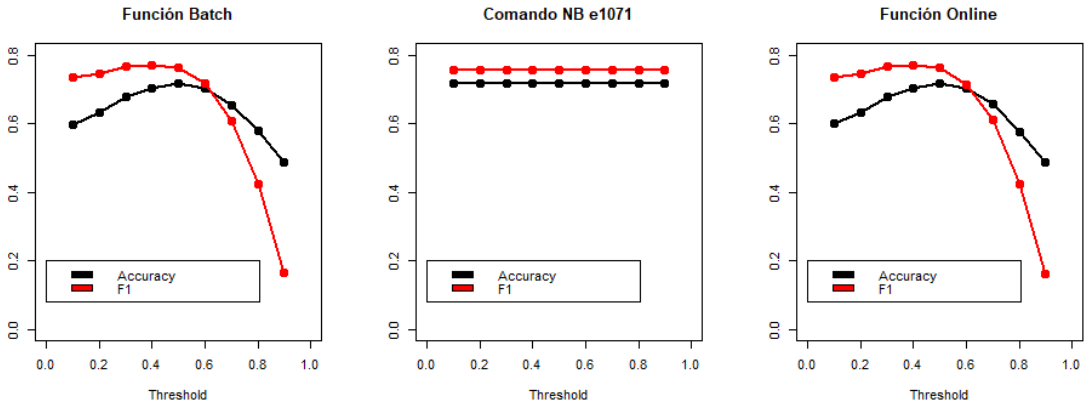


Figura 5.7: Accuracy y F1 en función del Umbral de Clasificación*.
* *Véase tabla de valores en Anexo Cuadro 6.5.*

En la figura 5.7 se muestran los resultados para cada clasificador. De acuerdo a la gráfica,

se puede notar que el umbral de clasificación óptimo se encuentra entre los valores 0.4 y 0.5, aproximadamente en el valor 0.45.

Asimismo, es de interés recalcar dos aspectos importantes: en el caso del método Naïve Bayes e1071, la clasificación resultante siempre fue la misma a pesar de que se utilizaron distintos umbrales de clasificación; y en segundo lugar, se pudo observar que las clasificaciones resultantes de los algoritmos creados para este proyecto fueron casi idénticas.

5.2.4. Batch Naïve Bayes

Se implementó el algoritmo Batch Naïve Bayes en el 100 % de los datos de tal forma que con la base de entrenamiento se realizó la etapa de aprendizaje del modelo y con la base de datos de prueba se aplicó la fase de clasificación del mismo.

En el Cuadro 5.8 se presenta la matriz de confusión resultante.

		Predicción		
		0	1	Total
Valor real	0	24.8104 %	19.4627 %	44.2731 %
	1	8.7484 %	46.9785 %	55.7269 %
	Total	33.5588 %	66.4412 %	100 %

Cuadro 5.8: Matriz de Confusión Batch Naïve Bayes, en términos porcentuales.

De acuerdo a la matriz de confusión, se pudo notar que en este ejemplo hay aproximadamente un 28 % de las observaciones fueron mal clasificadas, donde un 19.46 % corresponden a falsos positivos, es decir, el tweet fue predicho como sentimiento positivo (clase 1), pero en realidad corresponde a sentimiento negativo (clase 0).

5.2.5. e1071 Naïve Bayes

Se implementó el algoritmo Naïve Bayes e1071 considerando procesamiento en batch para el 100 % de la base de datos.

La matriz de confusión resultante se presenta en el Cuadro 5.9.

		Predicción		
		0	1	Total
Valor real	0	27.6043 %	16.6688 %	44.2731 %
	1	11.6372 %	44.0898 %	55.7269 %
	Total	39.2415 %	60.7585 %	100 %

Cuadro 5.9: Matriz de Confusión Naïve Bayes e1071, en términos porcentuales.

En este caso, también se observó que aproximadamente un 28 % de las observaciones fueron mal clasificadas, sin embargo, la distribución errores ha cambiado: los falsos positivos disminuyeron a un 16.7 % y los falsos negativos aumentaron a un 11.6 %.

5.2.6. Online Naïve Bayes

Para aplicar el algoritmo Naïve Bayes en un data stream, se siguieron los pasos del punto 3.3.3, es decir, se aplicó la función en batch para el 70 % correspondiente a la data de entrenamiento, y luego, se incorporaron observaciones a la base de datos inicial en grupos de $m = 140$ tuplas. Esto generó un total de 113 iteraciones (es decir, 113 veces se incorporaron observaciones nuevas a la base de datos, en grupos de 140).

Así, la próxima vez que lleguen nuevas observaciones a la base de datos, estas se deben clasificar utilizando los parámetros actualizados de la iteración anterior.

En el Cuadro 5.10 se presenta la matriz de confusión resultate del clasificador Online Naïve Bayes.

		Predicción		
		0	1	Total
Valor real	0	24.8420 %	19.4311 %	44.2731 %
	1	8.7800 %	46.9469 %	55.7269 %
	Total	33.6220 %	66.3780 %	100 %

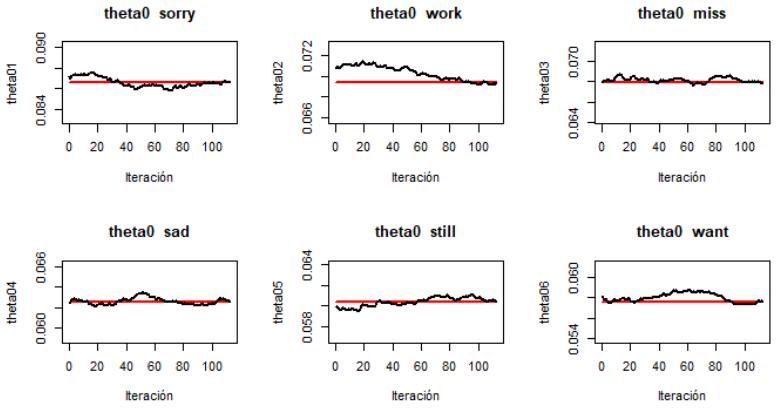
Cuadro 5.10: Matriz de Confusión Online Naïve Bayes, en términos porcentuales.

Note que con el clasificador en línea (Online Naïve Bayes) se obtuvieron resultados muy similares al clasificador en batch.

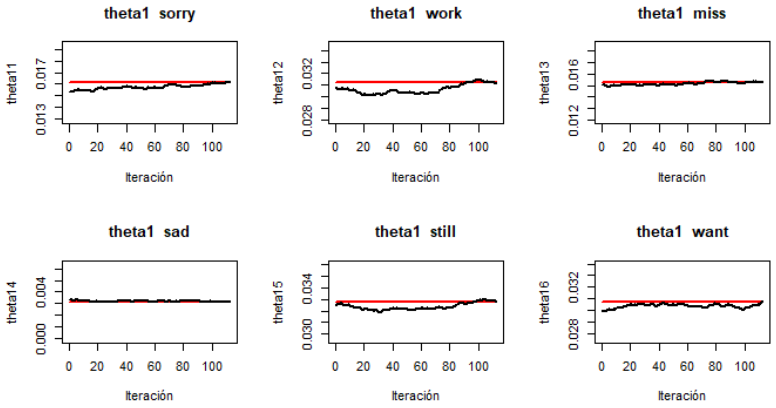
En la Figura 5.8 se presenta la estimación de cada parámetro en función de la iteración para la función online (en este caso solo se muestran las primeras 6 variables, sin embargo, en los Anexos Figura 6.3 y Figura 6.4 se presenta la gráfica de los parámetros θ_{0k} y θ_{1k}, respectivamente, con las 38 variables explicativas de la base de datos).

En la gráfica, las curvas de color negro representan la estimación de cada parámetro en función de la iteración para el método en línea; mientras que la línea de color rojo representa la estimación de cada parámetro obtenida de la estimación en lotes.

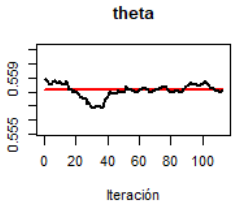
De esta forma, se pudo observar que en todos los casos los parámetros del método en línea convergieron a las estimaciones del método en batch.



(a) Parámetro θ_{0k}*



(b) Parámetro θ_{1k}**



(c) Parámetro θ

Figura 5.8: Estimación de parámetros en función de la iteración.

* *La estimación del parámetro θ_{0k} para las 38 variables se encuentra en el Anexo Figura 6.3.*

* *La estimación del parámetro θ_{1k} para las 38 variables se encuentra en el Anexo Figura 6.4.*

5.2.7. Comparación clasificadores

Los tres clasificadores fueron comparados en términos de rendimiento y tiempo de ejecución.

Rendimiento

De acuerdo a los resultados entregados por cada clasificador, se obtuvieron las medidas de calidad que se presentan en el Cuadro 5.11.

Métrica	Batch NB	e1071 NB	Online NB
Accuracy	0.7179	0.7169	0.7177
Precisión	0.7071	0.7257	0.7071
Sensibilidad	0.8430	0.7912	0.8424
F1	0.7691	0.7570	0.7688

Cuadro 5.11: Métricas de evaluación para conjunto de datos de Twitter.

Se pudo notar que los tres clasificadores presentaron un rendimiento muy similar. En particular, el método Naïve Bayes e1071 mostró un peor rendimiento respecto de los dos restantes en todas las medidas de evaluación utilizadas. Además, los algoritmos Naïve Bayes Batch y Online presentaron un rendimiento casi idéntico según las métricas de evaluación, siendo la función para procesamiento en batch la que presentó el mejor rendimiento (sin embargo, la diferencia con la función Online es tan pequeña, que de manera global se puede concluir que ambas presentan igual rendimiento).

Tiempo

Se realizaron 10 simulaciones por cada clasificador y se obtuvo el tiempo de ejecución de cada uno de ellos. En el cuadro 5.12 se presentan el tiempo promedio de ejecución de cada función (en segundos) y su desviación estándar.

	Batch NB	e1071 NB	Online NB
Promedio	0.4960	29.5310	0.4260
D. Estándar	0.0327	1.7729	0.0685

Cuadro 5.12: Tiempo promedio de ejecución y desviación estándar considerando 10 simulaciones*.
**Tabla con los tiempos de cada simulación se encuentran en Anexo Cuadro 6.6.*

De acuerdo a los resultados observados en el Cuadro 5.12, el método Online Naïve Bayes fue el que presentó un menor tiempo de ejecución, con un promedio de 0.43 segundos aproximadamente. Le sigue el algoritmo Batch Naïve Bayes, con un tiempo promedio de ejecución de 0.5 segundos. Por último, el algoritmo Naïve Bayes e1071, cuyo tiempo promedio de ejecución es de medio minuto aproximadamente (cantidad de tiempo que se considera sustancialmente mayor al de las funciones

creadas para procesamiento en batch y en stream, pues en aquellos casos el tiempo promedio de ejecución no supera 1 segundo).

Capítulo 6

Conclusión

En primer lugar, se considera importante señalar que los parámetros estimados con el método Naïve Bayes en streaming data convergen a los parámetros del método en lotes, tanto para variables independientes Normales como para variables independientes del tipo Bernoulli. Además, cabe destacar que las proporciones en las que se divide la base de datos para las fases de entrenamiento y prueba, influyen en la convergencia de parámetros, de modo que algunas particiones pueden provocar que los parámetros comiencen a converger desde las primeras iteraciones o desde las últimas (pero todos los casos finalmente llegan a la convergencia).

En el caso de variables independientes Normales, los tres clasificadores presentaron igual rendimiento en términos de las métricas de evaluación utilizadas (accuracy, precisión, sensibilidad y F1). Esto se puede entender, debido a que el comando *naiveBayes* de la librería e1071 de RStudio asume normalidad en las variables explicativas. Además, las funciones creadas del método, tanto para procesamiento en batch como para procesamiento en stream, están basadas en los mismos cálculos matemáticos (expuestos en el capítulo 3).

Por otra parte, en el caso de variables independientes Bernoulli, el rendimiento general de los clasificadores implementados en este trabajo es mejor respecto del algoritmo Naïve Bayes e1071. Se piensa que esto se podría deber, principalmente, a que el método Naïve Bayes e1071 asume distribución normal en las variables explicativas. Aún así, cabe destacar que pese a no cumplirse la normalidad de variables en el método Naïve Bayes e1071, este no presentó un rendimiendo evidentenemente inferior respecto de los otros dos.

Otro aspecto interesante a destacar acerca del rendimiento de los tres clasificadores, es que si se cambia el tamaño muestral, este no se ve afectado.

Desde la perspectiva del tiempo de ejecución de los clasificadores Naïve Bayes, el que presentó el menor tiempo promedio de ejecución para todos los casos evaluados es el método Online, seguido del método en Batch, y finalmente, el algoritmo Naïve Bayes e1071, cuyo tiempo de ejecución es

evidentemente mayor al de las dos primeras.

Además, se observó que mientras mayor es el tamaño muestral de la base de datos a clasificar, la diferencia de tiempos entre los clasificadores se hace mucho más evidente. En particular, para un tamaño muestral de 2 millones de observaciones y 4 variables explicativas Normales, el tiempo promedio de ejecución del comando de RStudio es de 4 minutos, mientras que el método Online Naïve Bayes presentó un tiempo promedio de 3 segundos bajo las mismas condiciones. En esa línea, para el mismo tamaño muestral, pero considerando 7 variables explicativas Bernoulli, el tiempo promedio de ejecución del algoritmo de la librería e1071 es de 5 minutos aproximadamente, en cambio, el método Online presentó un tiempo promedio de 5 segundos.

Otro aspecto importante, es que para el caso de muestras cuya clase objetivo está desbalanceada (por ejemplo, la base de datos astronómica), siempre es una buena opción determinar un umbral de clasificación que permita obtener el mejor resultado. De acuerdo a esto, los algoritmos implementados en este proyecto, se adaptan al umbral de clasificación planteado de tal manera que el score calculado se compara con dicho umbral para determinar a qué clase pertenecen las observaciones. Por el contrario, en el caso del método Naïve Bayes e1071, pese a incluir como argumento el umbral de clasificación, siempre entrega las mismas predicciones, provocando que no siempre se obtenga la mejor clasificación.

A raíz de la revisión de la literatura, los cálculos matemáticos realizados del método Naïve Bayes y los resultados obtenidos, surgen un par de reflexiones que pueden conducir a nuevos estudios que permitan profundizar en este método de clasificación Bayesiana en el contexto de un data stream.

Dado que en este proyecto se implementó el método de clasificación Naïve Bayes considerando que la variable respuesta es del tipo binario, resulta interesante plantear la posibilidad de poner en práctica el método suponiendo que la variable objetivo puede tener más de dos clases. Asimismo, una idea sugerente podría ser el considerar que las variables explicativas pueden seguir otra distribución de probabilidad; o que en un mismo ejemplo de clasificación, existan tanto atributos con distribución Normal como atributos con distribución Bernoulli.

También, con el fin de mejorar la exactitud predictiva del método Naïve Bayes, algunos investigadores han desarrollado extensiones del modelo. Por ejemplo, la corrección de Laplace, que puede ser muy útil cuando existe alguna clase con muy pocas observaciones; o la extensión de Langley (1993) que combina el método Naïve Bayes con árboles de decisión. En ese sentido, se propone poner en funcionamiento alguna de estas extensiones del método para evaluar su desempeño en data streams.

Por último, es importante señalar que el método Naïve Bayes es muy llamativo por la sencillez que implica el supuesto de independencia y su sorprendente efectividad al momento de clasificar. Por lo tanto, tales extensiones implican necesariamente añadir complejidad al método, lo que haría

que desapareciera una de las características que hace de Naïve Bayes un método tan popular: su simplicidad.

Referencias

[1] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand and D. Steinberg (2007): Top 10 algorithms in data mining, *Naïve Bayes*.

[2] K. M. Adam Chai, H. T. Ng and H. L. Chieu (2002): Bayesian Online Classifiers for Text Classification and Filtering, *SIGIR'02*. Tampere, Finland.

[3] Y. Xu, K. Wang, A. W.-C. Fu, R. Shee and J. Pei (2006): Classification Spanning Correlated Data Streams, *CIKM'06*. Arlington, Virginia, USA.

[4] C. Malagón Luque (2003): Clasicadores bayesianos. El algoritmo Naïve Bayes.

[5] J. Friedman, T. Hastie and R. Tibshirani (2008): Elements of Statistical Learning, The Naïve Bayes Classifier.

[6] H. Borchani (2012): Multi-Dimensional classification using Bayesian Networks for stationary and evolving streaming data, Ph. D. THESIS, Universidad Politécnica de Madrid, España.

[7] I. I. Frías Blanco (2012): Nuevos métodos para el aprendizaje en flujos de datos no estacionarios, Tesis Doctoral, Universidad Politécnica de Madrid, España.

[8] R. Rivera, L. Bracco, V, Costa, F. Coto, P. Cristaldo, L. Ramos, N. Rapesta, J. P. Núñez, S. Retamar, A. de Battista and N. E. Herrera (2017): Tecnologías de Procesamiento de Datos Masivos, Universidad Tecnológica Nacional, Entre Ríos; Universidad Nacional de San Luis; San Luis, Argentina.

[9] A. Rayón (2016): Procesando BIG DATA: paradigmas Batch, Tiempo real y Lambda.
URL `https://blogs.deusto.es/bigdata`

[10] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, C.-C. Chang and C.-C. Lin (2018): Package e1071, Version 1.7-0.

[11] J. Han, M. Kamber and J. Pei (2012): Data Mining. Concepts and Techniques, Third Edition, USA, pp. 327-363.

[12] C. Vercellis (2009): Business Intelligence: Data mining and Optimization for decision making, Politecnico di Milano, Italy.

[13] D. J. Hand and K. Yu (2001): Idiot's Bayes—Not so stupid after all? *International Statistical Review*, 69, pp. 385-398.

[14] P. Domingos and M. Pazzani (1997): On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29, pp. 103-130.

[15] C. C. Aggarwal (2006): Data Streams: Models and Algorithms (Advances in Database Systems). IBM T. J. Watson Research Center, Yorktown Heights, New York, EEUU.

[16] P. E. Lutu (2013): Fast Feature Selection for Naïve Bayes Classification in Data Stream Mining, *Proceedings of the World Congress on Engineering 2013, Vol. III.* London, UK.

[17] I. Kononenko (1990): Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga (Ed.), *Current Trends in Knowledge Acquisition.* Amsterdam, The Netherlands: IOS Press.

[18] D. M. Titterington, G. D. Murray, L. S. Murray, D. J. Spiegelhalter, A. M. Skene, J. D. F. Habbema and G. J. Gelpke (1981): Comparison of discrimination techniques applied to a complex data set of head injured patients. *Journal of the Royal Statistical Society, Series A*, 144, 145–175.

[19] S. Mani, M. J. Pazzani and J. West (1997): Knowledge discovery from a breast cancer database. *Lecture Notes in Artificial Intelligence*, 1211, 130–133.

[20] Russet E., Kronmal R.A. and Fisher L.D. (1983): The effect of assuming independence in applying Bayes' theorem to risk estimation and classification in diagnosis. *Computers and Biomedical Research*, 16,537-552.

[21] Nordyke R., Kulikowski C.A. and Kulikowski C.W. (1971): A comparison of methods for the automated diagnosis of thyroid dysfunction. *Computers and Biomedical Research*, 4, 374-389.

[22] Croft D.J. and Mitchol R.E. (1987): Mathematical models io medical diagnosis. *Ann Biomed. Engng.*, 2,69439.

[23] Fox. J., Bark. D. and Bardhan K.D. (1980): A quaotitative compyison with rule-based diagoostic inference. *Methods of Informarion in Medicine*, 19. 210-215.

[24] Pazzani M., Muramatsu J. and Billsus D. (1996): Syskill & Webert: Identifying interesting web sites. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 54–61. Portland, OR: AAAI Press.

[25] Langley P., Iba W. and Thompson K. (1992): An analysis of Bayesian classifiers. *Proceedings of the Tenth National Conference on Artificial Intelligence,* pp. 223–228. San Jose, CA: AAAI Press.

[26] Langley P. (1993): Induction of recursive Bayesian classifiers. *Proceedings of the Eighth European Conference on Machine Learning*, Vienna, Austria: Springer-Verlag, 153–164.

[27] Clark P. and Niblett T. (1989): The CN2 induction algorithm. *Machine Learning*, 3, 261–283.

[28] King R.D., Feng C. and Sutherland A. (1995): STATLOG-comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9,289-333.

[29] F. Elorrieta, S. Eyheramendy, A. Jordán, I. Dékány, M. Catelan, R. Angeloni, J. Alonso-García, R, Contreras-Ramos, F. Gran, G. Hajdu, N. Espinoza, R. K. Saito and D. Minniti (2016): A Machine Learned Classifier for RR Lyrae in the VVV Survey. *Catalogs and data*, A&A Volume 595, A82.

[30] RStudio Team (2016). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA. URL `http://www.rstudio.com/`

Anexos

Funciones creadas

Variables Normales

```
base_normal<-function(b,b0,b1,p,n,prop1,medias1,medias0,sd2_1,sd2_0)
{
  n1=n*prop1
  n0=n-n1
  b0[1:n0,]=0
  b1[1:n1,]=1
  nombres="y"
  for (i in 1:p) {
    b0[,i+1]=rnorm(n0,medias0[i],sd2_0[i])
    b1[,i+1]=rnorm(n1,medias1[i],sd2_1[i])
    nombres=c(nombres,paste("x",i,sep=""))}
  names(b0)=nombres
  names(b1)=nombres
  b=rbind(b0,b1)
  b$id=1:n
  return(list(base_n=b,base0_n=b0,base1_n=b1))
}

NB_normal<-function(b,threshold)
{
  b0=subset(b,b$y==0);b1=subset(b,b$y==1)
  n=length(b$y);n1=length(b1$y);n0=length(b0$y)
  p=dim(b)[2]-2
  theta=sum(b$y)/n
  mu1=apply(b1[,2:(p+1)], 2,sum)/n1
  mu0=apply(b0[,2:(p+1)], 2,sum)/n0
```

```
  dif=0
  dif=matrix(dif,nrow=n0,ncol=p)
  for (i in 1:p) {
    dif[,i]=b0[,i+1]-mu0[i]
  }
  dif2=dif^2;vecsum=apply(dif2, 2,sum)
  sigma0=vecsum/(n0-1)
  dif=0
  dif=matrix(dif,nrow=n1,ncol=p)
  for (i in 1:p) {
    dif[,i]=b1[,i+1]-mu1[i]
  }
  dif2=dif^2;vecsum=apply(dif2, 2,sum)
  sigma1=vecsum/(n1-1)
  uno=theta/(1-theta)
  dos=sqrt( prod(sigma0)/prod(sigma1) )
  est0=matrix(0,nrow = n,ncol = p)
  est1=matrix(0,nrow = n,ncol = p)
  for (i in 1:p) {
    est0[,i]=scale(b[,i+1],center = mu0[i],scale = sqrt(sigma0[i]))
    est1[,i]=scale(b[,i+1],center = mu1[i],scale = sqrt(sigma1[i])) }
  est0=est0^2;est1=est1^2
  sum_prob=0
  for (j in 1:p) {
    rest=(est0[,j]-est1[,j])
    sum_prob=sum_prob+rest }
  tres=exp(sum_prob/2)
  score=uno*dos*tres
  prob_1=score/(1+score)
  prob_0=1-prob_1
  b$score=score
  b$Prob_0=prob_0
  b$Prob_1=prob_1
  b$y_est=ifelse(b$Prob_1>threshold,1,0)
  b$flag=ifelse(b$y==b$y_est , "Bien","Mal")
  return(list(base_n=b,theta_y=theta,mu0=mu0,mu1=mu1,sigma0=sigma0,sigma1=sigma1))
}
```

```
act_parametros<-function(mu0,mu1,sigma0,sigma1,theta,nuevo,Nb)
{
  nuevo<-nuevo[,-(length(nuevo)-5):-length(nuevo)]
  p<-dim(nuevo)[2]-1;  n<-Nb[1];n0=Nb[2];n1=Nb[3]
  n_nuevo<-dim(nuevo)[1]
  nuevo0=subset(nuevo, nuevo$y==0);nuevo1=subset(nuevo, nuevo$y==1)
  n0_nuev<-dim(nuevo0)[1];n1_nuev<-dim(nuevo1)[1]
  D<-n/(n+n_nuevo);  C<-sum(nuevo[1])/(n+n_nuevo);  theta_nuevo<-(D*theta)+C
  D<-n0/(n0+n0_nuev); C<-(apply(nuevo0[-1], 2, sum))/(n0+n0_nuev)
  mu0_nuevo<-(D*mu0)+C
  D<-n1/(n1+n1_nuev); C<-(apply(nuevo1[-1], 2, sum))/(n1+n1_nuev)
  mu1_nuevo<-(D*mu1)+C
  D<-(n0-1)/(n0+n0_nuev-1);C<-((apply(nuevo0[-1]^2,2,sum))-((n0+n0_nuev)*
  (mu0_nuevo^2))+(n0*(mu0^2)))/(n0+n0_nuev-1)
  sigma0_nuevo<-(D*sigma0)+C
  D<-(n1-1)/(n1+n1_nuev-1);C<-((apply(nuevo1[-1]^2,2,sum))-((n1+n1_nuev)*
  (mu1_nuevo^2))+(n1*(mu1^2)))/(n1+n1_nuev-1)
  sigma1_nuevo<-(D*sigma1)+C
  return(list(theta_act=theta_nuevo, mu0_act=mu0_nuevo, mu1_act=mu1_nuevo,
  sigma0_act=sigma0_nuevo,  sigma1_act=sigma1_nuevo))
}


NB_actualizado<-function(mu0,mu1,sigma0,sigma1,theta,threshold,nuevo)
{
  p=dim(nuevo)[2]-2;n_nuevo=dim(nuevo)[1]
  uno=theta/(1-theta)
  dos=sqrt( prod(sigma0)/prod(sigma1) )
  est0=(scale(nuevo[,2:(p+1)],center=mu0,scale = sqrt(sigma0)))^2
  est1=(scale(nuevo[,2:(p+1)],center=mu1,scale = sqrt(sigma1)))^2
  tres=exp((apply(est0-est1, 1, sum))/2)
  score=uno*dos*tres
  prob_1=score/(1+score);  prob_0=1-prob_1
  nuevo$score=score
  nuevo$Prob_0=as.numeric(prob_0);  nuevo$Prob_1=as.numeric(prob_1)
  nuevo$y_est=ifelse(nuevo$Prob_1>threshold,1,0)
  nuevo$flag=ifelse(nuevo$y==nuevo$y_est, "Bien","Mal")
  return(list(nuevo=nuevo))
}
```

Variables Bernoulli

```
base_bernoulli<-function(b,b0,b1,p,n,prop1,thet1,thet0)
{
  n1=n*prop1;n0=n-n1
  b0[1:n0,]=0;b1[1:n1,]=1
  nombres="y"
  for (i in 1:p) {
    b0[,i+1]=sample(c(1,0),n0,prob = c(thet0[i],1-thet0[i]),replace = T)
    b1[,i+1]=sample(c(1,0),n1,prob = c(thet1[i],1-thet1[i]),replace = T)
    nombres=c(nombres,paste("x",i,sep=""))
  }
  names(b0)=nombres;names(b1)=nombres
  b=rbind(b0,b1);b$id=1:n
  return(list(base_n=b,base0_n=b0,base1_n=b1))
}


NB_ber<-function(b,threshold)
{
  b0=subset(b,b$y==0);  b1=subset(b,b$y==1)
  n=length(b$y);n1=length(b1$y);n0=length(b0$y)
  p=dim(b)[2]-2
  theta=sum(b$y)/n
  vecsum=apply(b1[,2:(p+1)], 2,sum)
  theta1k=vecsum/n1
  vecsum=apply(b0[,2:(p+1)], 2,sum)
  theta0k=vecsum/n0
  uno=theta/(1-theta)
  dos=1
  for (i in 1:p) {
    produc=((theta1k[i]/theta0k[i])^(b[,i+1]))
    dos=dos*produc  }
  tres=1
  for (i in 1:p) {
    produc=(( (1-theta1k[i])/(1-theta0k[i]) )^(1-b[,i+1]))
    tres=tres*produc  }
  score=uno*dos*tres
  prob_1=score/(1+score)
```

```
  prob_0=1-prob_1
  b$score=score
  b$Prob_0=prob_0
  b$Prob_1=prob_1
  b$y_est=ifelse(b$Prob_1>threshold,1,0)
  b$flag=ifelse(b$y==b$y_est , "Bien","Mal")
  return(list(base=b,theta0k=theta0k,theta1k=theta1k,theta=theta))
}


act_parametros_ber<-function(theta,theta1,theta0,nuevo,Nb)
{
  nuevo=nuevo[,-(length(nuevo)-5):-length(nuevo)]
  n=N[1];n0=N[2];n1=N[3];n_nuevo=dim(nuevo)[1]
  nuevo0=subset(nuevo,nuevo$y==0);nuevo1=subset(nuevo,nuevo$y==1)
  n0_nuev=dim(nuevo0)[1];n1_nuev=dim(nuevo1)[1]
  D=n/(n+n_nuevo);C=as.numeric(sum(nuevo[,1])/(n+n_nuevo));theta_nuevo=(D*theta)+C
  D=n0/(n0+n0_nuev);C=as.numeric(apply(nuevo0[-1],2,sum)/(n0+n0_nuev))
  theta0k_nuevo=(D*theta0)+C
  D=n1/(n1+n1_nuev);C=as.numeric(apply(nuevo1[-1],2,sum)/(n1+n1_nuev))
  theta1k_nuevo=(D*theta1)+C
  return(list(theta_act=theta_nuevo, theta0k_act=theta0k_nuevo,
              theta1k_act=theta1k_nuevo))
}


NB_ber_actualizado<-function(theta,theta1,theta0,nuevo,threshold,n,p)
{
  h=as.data.frame(t(theta1/theta0));h[1:n,]=h;h=h^nuevo[,2:(p+1)]
  hh=as.data.frame(t((1-theta1)/(1-theta0)));hh[1:n,]=hh;hh=hh^(1-nuevo[,2:(p+1)])
  score=(theta/(1-theta))*(apply(h, 1, prod))*(apply(hh, 1, prod))
  nuevo$score=score
  nuevo$Prob_0=as.numeric(1-(score/(1+score)))
  nuevo$Prob_1=as.numeric(score/(1+score))
  nuevo$y_est=ifelse(nuevo$Prob_1>threshold,1,0)
  nuevo$flag=ifelse(nuevo$y==nuevo$y_est, "Bien","Mal")
  return(list(nuevo=nuevo))
}
```

Naïve Bayes con variables independientes Normales

Datos simulados con variables explicativas Normales

	y	x1	x2	x3	x4	id
1	0	10.458242	12.7250990	0.3271121	4.405606	1
2	0	9.954575	13.1412750	6.2350664	5.721688	2
3	0	9.485903	3.7829259	7.2304281	11.224778	3
4	0	8.116793	29.7404267	-0.6920307	6.819420	4
5	0	7.295759	17.7833510	2.8849688	12.403287	5
6	0	15.244778	5.3592434	5.3684985	10.038041	6
7	0	10.760648	9.9243813	2.5281108	6.831332	7

Showing 1 to 7 of 2,000,000 entries

Figura 6.1: Subconjunto base de datos generada para simulación de variables Normales.

Métricas de evaluación conjunto de datos simulados Normal

	Batch Naïve Bayes			
N	Accuracy	Precisión	Sensibilidad	F1
500	0.9867	0.9882	0.9882	0.9882
1000	0.9867	0.9821	0.9940	0.9880
3000	0.9878	0.9800	1.0000	0.9899
5000	0.9887	0.9868	0.9945	0.9906
10000	0.9843	0.9800	0.9938	0.9869
50000	0.9886	0.9867	0.9943	0.9905
100000	0.9868	0.9848	0.9934	0.9891
500000	0.9864	0.9841	0.9933	0.9887
1000000	0.9866	0.9843	0.9935	0.9889
2000000	0.9865	0.9842	0.9934	0.9888

	Naïve Bayes e1071			
N	Accuracy	Precisión	Sensibilidad	F1
500	0.9867	0.9882	0.9882	0.9882
1000	0.9867	0.9821	0.9940	0.9880
3000	0.9878	0.9800	1.0000	0.9899
5000	0.9887	0.9868	0.9945	0.9906
10000	0.9843	0.9800	0.9938	0.9869
50000	0.9886	0.9867	0.9943	0.9905
100000	0.9868	0.9848	0.9934	0.9891
500000	0.9864	0.9841	0.9933	0.9887
1000000	0.9866	0.9843	0.9935	0.9889
2000000	0.9865	0.9842	0.9934	0.9888

	Online Naïve Bayes			
N	Accuracy	Precisión	Sensibilidad	F1
500	0.9867	0.9882	0.9882	0.9882
1000	0.9867	0.9821	0.9940	0.9880
3000	0.9878	0.9800	1.0000	0.9899
5000	0.9852	0.9835	0.9920	0.9877
10000	0.9843	0.9800	0.9938	0.9869
50000	0.9886	0.9867	0.9943	0.9905
100000	0.9862	0.9837	0.9935	0.9886
500000	0.9864	0.9841	0.9933	0.9887
1000000	0.9866	0.9842	0.9935	0.9888
2000000	0.9865	0.9842	0.9934	0.9888

Cuadro 6.1: Métricas de evaluación en función del tamaño muestral.

Umbral de Clasificación datos astronómicos

Umbral	Batch NB		NB e1071		Online NB	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
0.0	0.5270	0.3441	0.6771	0.4338	0.5233	0.3292
0.1	0.6559	0.4182	0.6762	0.4331	0.6603	0.4052
0.2	0.6614	0.4221	0.6762	0.4331	0.6684	0.4188
0.3	0.6672	0.4264	0.6762	0.4331	0.6727	0.4123
0.4	0.6743	0.4317	0.6762	0.4331	0.6817	0.4262
0.5	0.6780	0.4345	0.6762	0.4331	0.6842	0.4270
0.6	0.6803	0.4363	0.6762	0.4331	0.6913	0.4311
0.7	0.6837	0.4389	0.6762	0.4331	0.6941	0.4326
0.8	0.6901	0.4439	0.6761	0.4330	0.6961	0.4397
0.9	0.6980	0.4503	0.6761	0.4330	0.7032	0.4392
1.0	0.9713	0.8887	0.6761	0.4330	0.9759	0.8994

Cuadro 6.2: Accuracy y F1 en función del Umbral de clasificación..

Tiempo de ejecución en datos astronómicos

	Clasificador		
Ejecución	Batch NB	e1071 NB	Online NB
1	0.28	3.73	0.16
2	0.28	3.61	0.15
3	0.19	3.59	0.14
4	0.18	3.59	0.15
5	0.19	3.74	0.17
6	0.19	3.59	0.24
7	0.19	4.15	0.19
8	0.19	3.55	0.22
9	0.19	3.67	0.15
10	0.29	3.60	0.16
Promedio	0.2170	3.6820	0.1730
D. Estándar	0.0460	0.1762	0.0333

Cuadro 6.3: Tiempo (en segundos) de los clasificadores considerando 10 simulaciones.

Naïve Bayes con variables independientes Bernoulli

Conjunto de datos simulado

	y	x1	x2	x3	x4	x5	x6	x7	id
1	0	0	0	0	1	1	1	1	1
2	0	1	0	0	1	1	0	1	2
3	0	1	0	0	1	0	0	0	3
4	0	0	0	0	1	1	1	0	4
5	0	1	0	0	1	0	0	0	5
6	0	1	0	0	1	1	1	1	6
7	0	1	0	0	1	1	1	0	7

Showing 1 to 7 of 1,000,000 entries

Figura 6.2: Subconjunto base de datos generada para simulación de variables Bernoulli.

Estimación de parámetros Datos Twitter

(Próxima página)

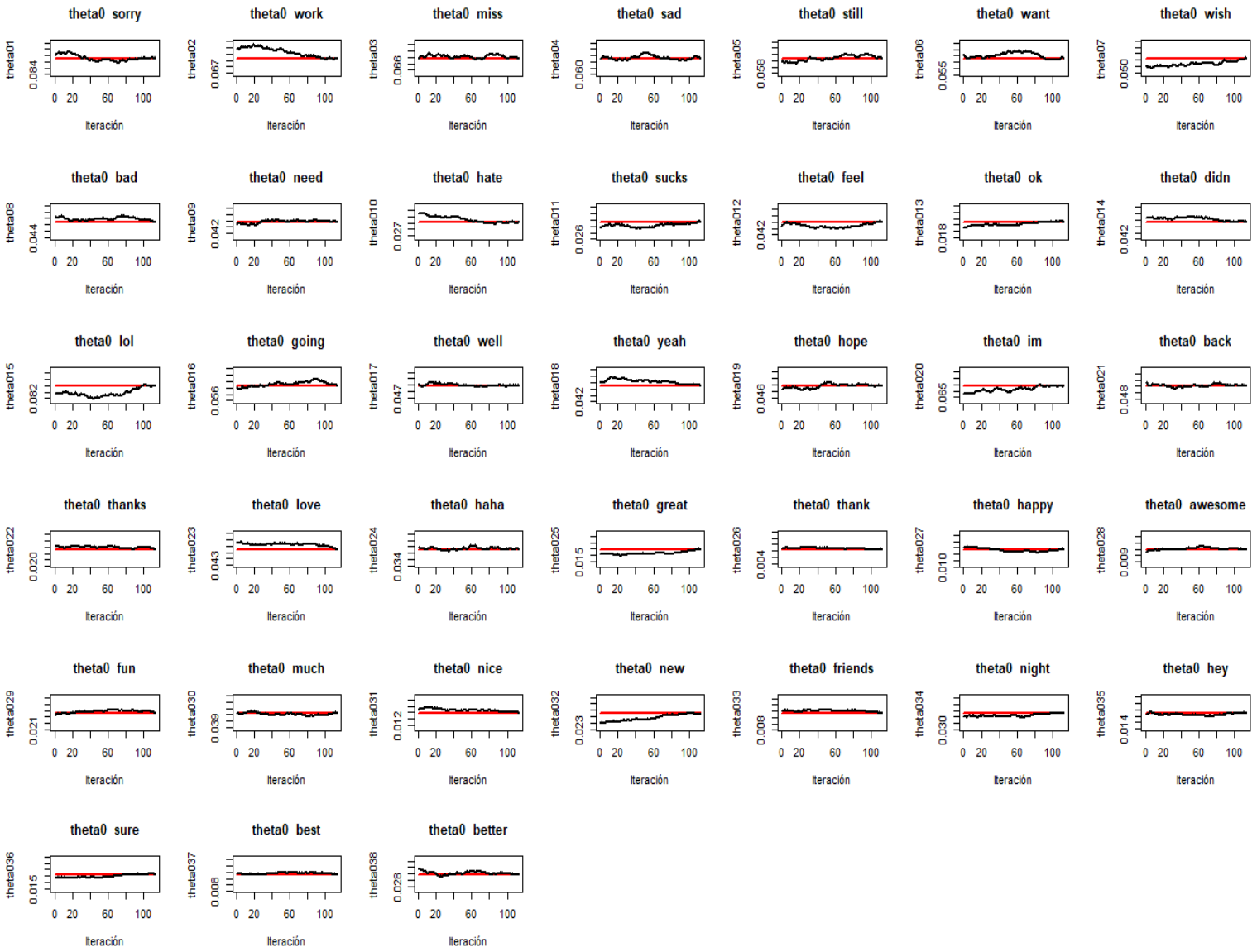


Figura 6.3: Estimación de parámetro θ_{0k} en función de la iteración.

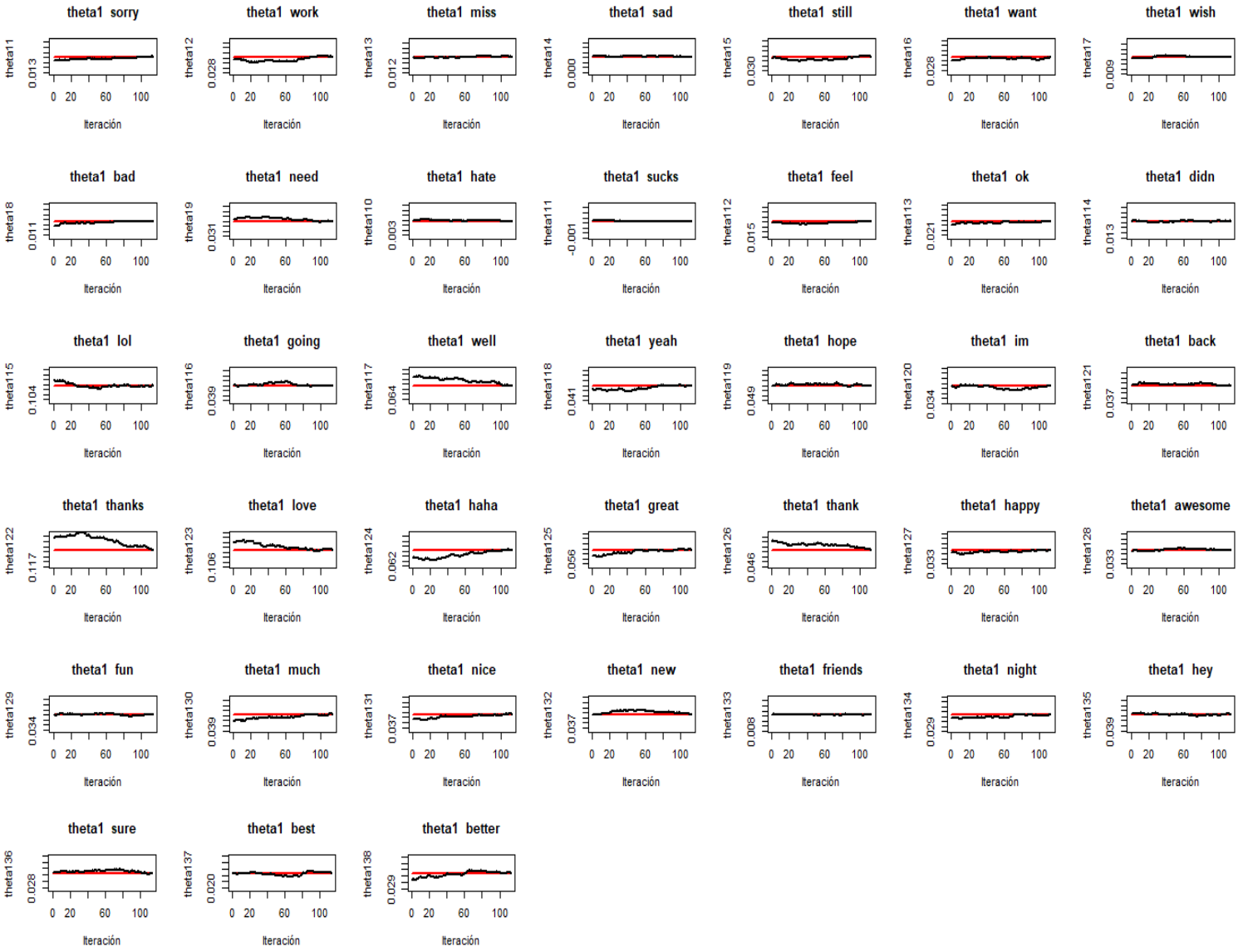


Figura 6.4: Estimación de parámetro θ_{1k} en función de la iteración.

Métricas de evaluación de base de datos simulados Bernoulli

	Batch Naïve Bayes			
N	Accuracy	Precisión	Sensibilidad	F1
500	0.9600	0.9403	0.9692	0.9545
1000	0.9700	0.9664	0.9730	0.9697
3000	0.9689	0.9620	0.9712	0.9666
5000	0.9700	0.9776	0.9561	0.9667
10000	0.9680	0.9682	0.9626	0.9654
50000	0.9702	0.9653	0.9686	0.9669
100000	0.9702	0.9663	0.9678	0.9670
500000	0.9698	0.9661	0.9665	0.9663
1000000	0.9697	0.9664	0.9663	0.9664
2000000	0.9697	0.9662	0.9664	0.9663

	Naïve Bayes e1071			
N	Accuracy	Precisión	Sensibilidad	F1
500	0.9400	0.9118	0.9538	0.9323
1000	0.9500	0.9346	0.9662	0.9502
3000	0.9522	0.9289	0.9712	0.9496
5000	0.9533	0.9411	0.9576	0.9493
10000	0.9527	0.9304	0.9705	0.9501
50000	0.9617	0.9377	0.980	0.9584
100000	0.9554	0.9273	0.9777	0.9518
500000	0.9599	0.9422	0.9703	0.9560
1000000	0.9604	0.9430	0.9708	0.9567
2000000	0.9601	0.9421	0.9711	0.9564

	Online Naïve Bayes			
N	Accuracy	Precisión	Sensibilidad	F1
500	0.9600	0.9403	0.9692	0.9545
1000	0.9700	0.9664	0.9730	0.9697
3000	0.9689	0.9620	0.9712	0.9666
5000	0.9700	0.9776	0.9561	0.9667
10000	0.9680	0.9682	0.9626	0.9654
50000	0.9702	0.9653	0.9686	0.9669
100000	0.9702	0.9663	0.9678	0.9670
500000	0.9698	0.9661	0.9665	0.9663
1000000	0.9697	0.9664	0.9663	0.9664
2000000	0.9697	0.9662	0.9664	0.9663

Cuadro 6.4: Métricas de evaluación en función del tamaño muestral.

Base de datos Twitter

	ItemID	Sentiment	SentimentText
1	1	0	is so sad for my APL friend.....
2	2	0	I missed the New Moon trailer...
3	3	1	omg its already 7 30 O
4	4	0	.. Omgaga. Im sooo im gunna CRy. I ve been at this denti...
5	5	0	i think mi bf is cheating on me T T
6	6	0	or i just worry too much
7	7	1	Juuuuuuuuuuuuuuuussssst Chillin

Figura 6.5: Subconjunto base de datos Twitter.

Palabras más frecuentes por cada sentimiento.

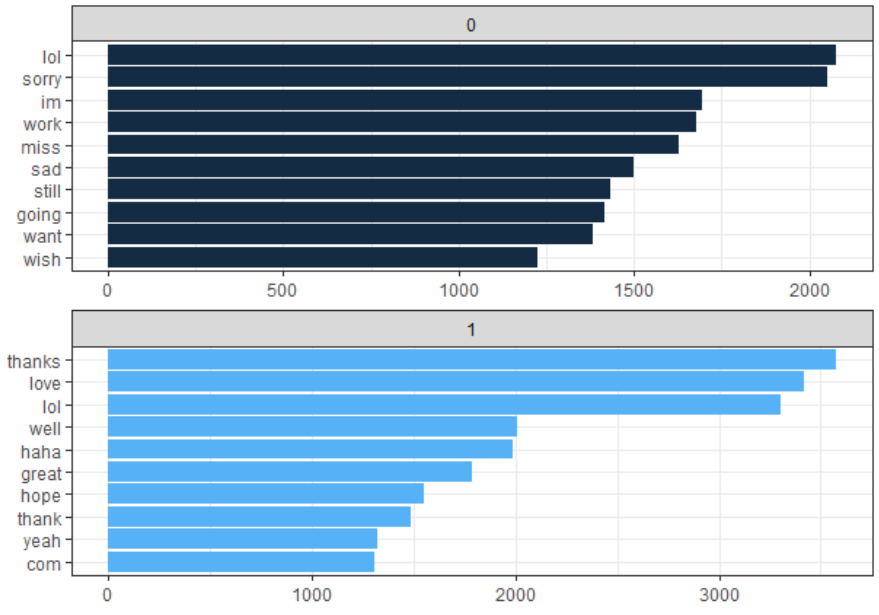


Figura 6.6: Frecuencia de palabras más utilizadas por cada clase.

Base de datos Twitter Final

	y	sorry	work	miss	sad	still	want	wish	bad	need	hate	sucks	feel	ok
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	1	0	0	1	0	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 6.7: Subconjunto base de datos final Twitter (mostrándose 13 de 38 variables Bernoulli).

Selección Umbral de clasificación Datos Twitter

Umbral	Batch Naïve Bayes		Naïve Bayes e1071		Online Naïve Bayes	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
0.1	0.5984	0.7337	0.7169	0.7570	0.5988	0.7339
0.2	0.6315	0.7469	0.7169	0.7570	0.6313	0.7468
0.3	0.6780	0.7657	0.7169	0.7570	0.6786	0.7662
0.4	0.7014	0.7701	0.7169	0.7570	0.7015	0.7703
0.5	0.7180	0.7624	0.7169	0.7570	0.7185	0.7623
0.6	0.7041	0.7168	0.7169	0.7570	0.7029	0.7130
0.7	0.6549	0.6084	0.7169	0.7570	0.6559	0.6106
0.8	0.5780	0.4245	0.7169	0.7570	0.5771	0.4220
0.9	0.4889	0.1642	0.7169	0.7570	0.4886	0.1629

Cuadro 6.5: Accuracy y F1 en función del Umbral de clasificación.

Tiempo de ejecución clasificadores en base de datos Twitter

	Clasificador		
Ejecución	Batch NB	e1071 NB	Online NB
1	0.56	31.62	0.38
2	0.50	29.70	0.41
3	0.55	33.64	0.37
4	0.47	28.64	0.46
5	0.49	28.22	0.53
6	0.47	28.42	0.46
7	0.48	28.50	0.52
8	0.49	29.44	0.39
9	0.47	28.97	0.31
10	0.48	28.16	0.43
Promedio	0.4960	29.5310	0.4260
D. Estándar	0.0327	1.7729	0.0685

Cuadro 6.6: Tiempo (en segundos) de los clasificadores considerando 10 simulaciones.