La deserción en cursos universitarios.

Construcción de modelos sobre datos de la Universidad Nacional de Río Negro
usando técnicas de Extracción de Conocimiento.

AUTORA
Ing. Sonia Alejandra Formia

DIRECTORA
Prof. Lic. Laura C. Lanzarini

ASESOR ACADEMICO
Dr. Waldo Hasperué

Tesis presentada para obtener el grado de
Magister en Tecnología Informática aplicada en Educación

FACULTAD DE INFORMATICA
UNIVERSIDAD NACIONAL DE LA PLATA

9 de octubre de 2013

Objetivo

En el ámbito de la Universidad Nacional de Río Negro (UNRN), y en particular desde la Licenciatura en Sistemas, es una creciente preocupación del cuerpo docente el fenómeno de deserción y desgranamiento que se ha podido apreciar en los primeros años de vida de la carrera.

Esta problemática ha sido expresada en el Informe de Autoevaluación a la CONEAU para el proceso de acreditación (2010):

> *"Existe un fenómeno de desgranamiento y deserción que se ha visto en los cursos que comenzaron a la actualidad. Creemos que es importante realizar un análisis con al menos tres años de datos estadísticos para determinar causas y orígenes de los problemas..."*

Es también un hecho que el fenómeno descripto no se circunscribe a la Licenciatura en Sistemas, sino que se da en general en las carreras que dicta la Universidad, siendo esta situación el motivo principal de este trabajo.

El objetivo general de esta tesis es abordar el estudio del fenómeno de deserción estudiantil universitaria mediante un proceso de extracción de conocimiento a partir de datos.

En el camino hacia la concreción del objetivo de máxima: predecir la deserción, se pueden encontrar otras metas que aporten información no trivial y de utilidad para la toma de decisiones, por ejemplo, describir o caracterizar a los estudiantes de la UNRN a través de perfiles que ayuden a orientar la implementación de medidas a los estratos en los que las mismas puedan ejercer más influencia positiva.

El objetivo específico de esta tesis es caracterizar a los estudiantes de la UNRN que abandonan la carrera en los primeros años. Se busca establecer perfiles que permitan realizar recomendaciones tendientes a revertir esta situación.

Publicaciones relacionadas con esta tesis

- Formia, S.; Lanzarini, L. *"Evaluación de técnicas de Extracción de Conocimiento en Bases de Datos y su aplicación a la deserción de alumnos universitarios"*. VIII Congreso Tecnología en Educación y Educación en Tecnología (TE&ET) Santiago del Estero. Argentina. Junio 2013.
- Formia, S.; Lanzarini, L.; Hasperué W. *"Characterization of University Drop-Out at UNRN Using Data Mining"*. XIX Congreso Argentino de Ciencias de la Computación (CACIC 2013) Mar del Plata. Argentina. Octubre 2013.

Prefacio

Esta tesis resume mis investigaciones realizadas durante los últimos dos años en lo que se refiere a la aplicación de técnicas de Minería de Datos para obtener un modelo que ayude a comprender el problema de la deserción universitaria en la UNRN en general y en la Licenciatura en Sistemas de la Sede Atlántica en particular. Se trata de un problema que lejos de resolverse, crece año a año.

La Minería de Datos enfrenta la resolución de problemas desde un ángulo muy distinto al que habitualmente se utiliza en Informática. Busca generalizar la información disponible con el objetivo de detectar patrones útiles, novedosos, relevantes sin recurrir a ninguna hipótesis previa. Esto ha llamado mi atención motivándome a utilizarla para obtener respuestas objetivas en esta problemática concreta.

La ciudad de Viedma vió marcharse a sus adolescentes a estudiar a otras provincias durante muchos años. Por motivos emocionales o económicos, también vió a muchos de ellos resignar su vocación. Luego de más de veinte años de ejercer mi profesión en la zona y esperar una oportunidad para desarrollar mi vocación docente, junto a un grupo de colegas sentimos el orgullo de dictar las primeras clases de una carrera universitaria en Sistemas en la recientemente creada Universidad Nacional de Río Negro, en el año 2009. Algunos de los alumnos que asistían a esas clases habían dejado sus carreras en otras ciudades por diferentes motivos, hacía ya varios años, otros veían posible, por primera vez, estudiar lo que querían sin irse de sus hogares.

La deserción y el desgranamiento que es ampliamente conocido en todas las universidades del país, también se vio reflejado muy rápidamente en la UNRN. Es allí donde surge en el grupo docente que integro, el interés por encontrar rápidos paliativos al problema. Siendo profesora de las cátedras de Bases de Datos, que se inician el segundo año de la carrera, recibí y sigo recibiendo muchos menos alumnos de los que ingresan en primer año. Los esfuerzos académicos realizados por la Institución como cursos nivelatorios, tutorías, etc., no parecen dar los rápidos resultados que la problemática requiere.

Por estas razones y dado mi interés personal de profundizar en el estudio de las técnicas de Minería de Datos y el acercamiento al tema que obtuve de las materias de la Maestría, surge la motivación para el desarrollo del presente trabajo, que viene a aportar un camino alternativo de solución al problema detectado.

La investigación que se presenta en este documento puede dividirse en dos partes. En la primera de ellas, descripta en los 3 primeros capítulos del informe, se llevan a cabo las siguientes tareas: se exponen las distintas etapas del proceso de extracción de conocimiento que deben llevarse a cabo para modelizar la información presente en las bases de datos operativas de la Universidad Nacional de Río Negro, se preparan los datos de la UNRN para poder utilizar algoritmos de minería de datos y se investigan y analizan las técnicas de $\mathcal{DM}$ factibles de aplicar al caso de estudio. En la segunda parte se describe cronológicamente el proceso seguido para la obtención del modelo final. Este proceso se inicia con la construcción de modelos descriptivos, cuyos resultados reafirman la necesidad de abocarse a la selección de atributos, tarea en la que se pone el mayor esfuerzo, dando lugar así a la construcción de un modelo predictivo de deserción sencillo y entendible.

Índice general

Capítulo 1

Extracción de Conocimiento

1.1. Introducción

El presente trabajo se enmarca en lo que se conoce como proceso de Extracción de Conocimiento o $\mathcal{KDD}$ (del inglés *Knowledge Discovery in Databases*) el cual tiene como objetivo la detección automática de patrones existentes en la información disponible sin requerir una hipótesis especificada a priori. Su aplicación requiere identificar, en base al problema a resolver, cuál es la información sobre la que se va a trabajar y cuál es el tipo de modelo que se desea obtener. Esto último tiene una fuerte incidencia en la técnica a utilizar.

La información disponible proviene del sistema SIU-Guaraní de la UNRN. Es decir que para cada alumno se relevan un número importante de características relacionadas con su situación personal, académica y laboral. Seleccionar de este conjunto las adecuadas para construir un modelo es un verdadero desafío. Existe una relación proporcional entre la cantidad de atributos a utilizar y la complejidad del modelo a obtener. Por lo antes expuesto, resulta de interés la detección temprana de los atributos más relevantes a fin de simplificar el modelo y reducir el tiempo de obtención del mismo.

El proceso de Extracción de Conocimiento a partir de Bases de Datos ($\mathcal{KDD}$), según Fayyad et al. 1996 [Usama et al., 1996] es el

> *"proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y en última instancia comprensibles a partir de los datos"*

Se trata de un proceso que, a diferencia de los sistemas tradicionales de explotación de datos basados en la existencia de hipótesis o modelos previos, busca el descubrimiento del conocimiento sin una hipótesis preconcebida.

Es importante remarcar que bajo el enfoque tradicional, generalmente estadístico, es preciso definir las hipótesis que se desean verificar, en otras palabras, habitualmente alguien debe sugerir la respuesta

esperada para luego contrastarla con la información de la base de datos. Este no es el enfoque deseado en este trabajo.

De más está decir que contar con herramientas que permitan detectar automáticamente nuevas asociaciones entre patrones será de suma utilidad en cualquier proceso de toma de decisiones.

El proceso de $\mathcal{KDD}$ consta de una serie de fases que definen la metodología a utilizar. Esta metodología provee una representación completa del ciclo de vida de un proyecto de data mining.

La secuencia de estas fases no es estricta y frecuentemente hay movimiento entre ellas, dependiendo del resultado de cada fase. La figura 1.1 muestra las interrelaciones entre las fases y los resultados de las mismas, dejando en claro la naturaleza cíclica del proceso.

1.2. Fases del proceso de $\mathcal{KDD}$

La etapa más relevante de este proceso es la Minería de Datos ($\mathcal{DM}$, del inglés *Data Mining*), que involucra las técnicas necesarias para la construcción de modelos a partir de la información disponible. Dichas técnicas tienen la probada capacidad de descubrir reglas y/o patrones significativos de información que puedan ayudar tanto en el diagnóstico correcto del problema como en la formulación de estrategias de solución.

> *"La minería de datos es un término relativamente moderno que integra numerosas técnicas de análisis de datos y extracción de modelos ... Es capaz de extraer patrones, de describir tendencias y regularidades, de predecir comportamientos y, en general, de sacar partido de la información computarizada que nos rodea hoy en día, generalmente heterogénea y en grandes cantidades ... permite a los individuos y a las organizaciones comprender y modelar de una manera más eficiente y precisa el contexto en que deben actuar y tomar decisiones."* [Hernández Orallo et al., 2004]

Las áreas en las que se emplean métodos de $\mathcal{DM}$ son cada día más variadas. Se encuentran ejemplos de casos de éxito en aplicaciones bancarias, financieras, científicas, empresariales, económicas, industriales, etc. [Westphal and Teresa, 1998] [Neukart et al., 2012]. En el campo específico de la educación, se han utilizado tanto en la captación de estudiantes como en el análisis y detección de abandonos y para la estimación de la duración de la carrera [La Red Martínez et al., 2009] [Luo, 2008] [Alcover et al., 2007] [Valero and Salvador, 2009] [Rodallegas et al., 2010] [Valero et al., 2010] [Wang et al., 2012]. Recientemente se han desarrollado entornos que facilitan la aplicación de técnicas de $\mathcal{DM}$ en contextos educativos [Ngo et al., 2012]. En todas estas disciplinas el modelado de los datos puede ayudar a entender el entorno donde se desenvuelve la organización y así colaborar en una mejor toma de decisiones.

La Minería de Datos ha evolucionado en los últimos años hacia una disciplina que se encarga de la modelización predictiva, *forecasting* (uso de datos históricos para determinar tendencias a futuro) y optimización de todo tipo de fenómenos y problemas. Se trata de construir modelos a partir de grandes

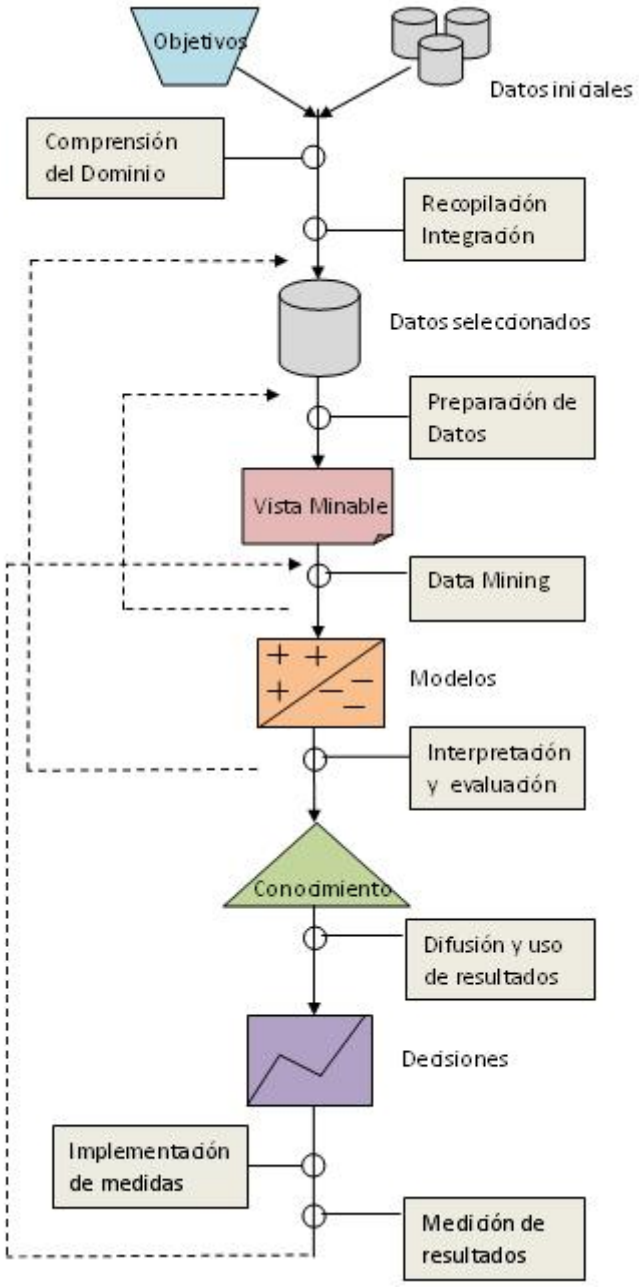

Figura 1.1: Fases que componen el proceso de $\mathcal{KDD}$

cantidades de datos, análisis inteligente, aprendizaje automático y métodos estadísticos multivariados, que permiten analizar bases de datos con muchas variables (alta dimensionalidad y complejidad de los datos).

Las técnicas de $\mathcal{DM}$ apuntan a transformar datos en información para la toma de decisiones. Dependen en gran medida de los datos de que se disponga y de la preparación adecuada que se les dé a los mismos, de manera de poder utilizar diferentes algoritmos y metodologías de descubrimiento.

El objetivo de las técnicas de $\mathcal{DM}$ es extraer conocimiento desde los datos, y ese conocimiento constituye el modelo de los datos analizados. Los patrones pueden ser utilizados para predecir observaciones futuras o explicar observaciones pasadas, capacidades fundamentales para mejorar el comportamiento en relación a un fenómeno, como el caso de la deserción universitaria.

Si bien las técnicas de $\mathcal{DM}$ ocupan un lugar relevante en la empresa y la toma de decisiones del sector privado, también son aplicables a la educación superior, considerando las grandes cantidades de datos que conforman el expediente de los estudiantes. Con dicha información las Universidades podrían conocer los perfiles de sus alumnos, así como las características de los estudiantes desertores.

Este trabajo realiza un estudio de las diferentes técnicas de Minería de Datos para evaluar su posibilidad de aplicación en el análisis del fenómeno de deserción de alumnos universitarios, con el objetivo de obtener modelos que ayuden a predecir el abandono, utilizando las bases de datos que registran información de los alumnos de las carreras de grado de la UNRN.

1.2.1. Comprensión del Dominio

Una fase importante de cualquier proyecto de $\mathcal{DM}$ consiste en entender los objetivos del proyecto desde una perspectiva de la organización para poder desarrollar un plan preliminar en pos de los objetivos.

Para entender qué datos deben ser analizados y cómo, es vital poseer un completo entendimiento del problema para el que se está buscando una solución. Esta fase involucra pasos clave como definir los objetivos, comprender la situación, determinar el papel del $\mathcal{DM}$ en el proyecto y visualizar un plan de trabajo.

La importancia de esta fase se ha incrementado últimamente, debido a la tendencia a acercar las técnicas de Minería de Datos a la realidad de los negocios, aprovechando el conocimiento del dominio de los expertos. Esta metodología, conocida como *Domain Driven Data Mining* [Cao, 2010], permite reconocer subjetivamente los modelos de interés para los usuarios.

1.2.2. Recopilación e integración de datos

Esta fase se inicia con la obtención de los datos. Una vez conseguida la información se procede a familiarizarse con ella e identificar su procedencia. En esta etapa se trabaja en recolectar los datos, describirlos, explorarlos y verificar su calidad.

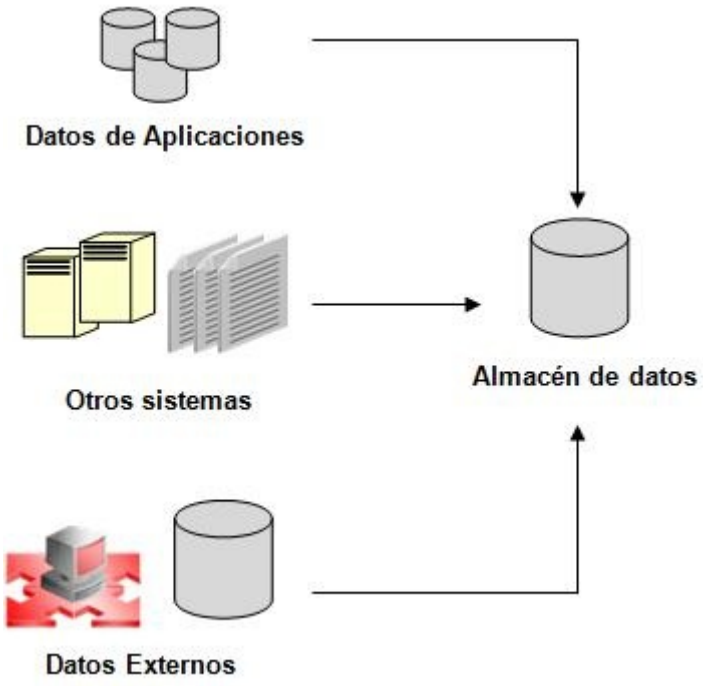

Figura 1.2: Almacenes de Datos (*Data warehouses*)

Comenzando con la tarea de recopilación puede decirse que, las bases de datos y las aplicaciones basadas en el procesamiento tradicional de datos en línea (OLTP, del inglés *On Line Transaction Processing*) cubren las necesidades diarias de información de una organización, pero no siempre son suficientes para funciones como el análisis, planificación y predicción. Por ello, en algunos casos se requiere obtener datos de otras áreas de la organización. Incluso puede ocurrir que algunos datos indispensables para el análisis no hayan sido recolectados hasta el momento, en estos casos puede ser necesario obtener datos desde bases de datos públicas (datos censales, datos demográficos, etc.) o privadas (de bancos, compañías de servicios, etc.). Cuando se utilizan fuentes de datos de diferentes orígenes, se enfrentan nuevos problemas, por ejemplo diferentes formatos de registro, diferentes grados de agregación en los datos, claves primarias no coincidentes, etc.

De estos hechos se desprende, entonces, la necesidad de integrar todos los datos de los que se dispone, y esto da lugar a la tecnología de almacenes de datos (*Data Warehouses*). Un almacén de datos (Ver figura 1.2) recopila información de diferentes fuentes y las unifica en un único repositorio con un diseño que se adapta perfectamente a las consultas estadísticas y al análisis de datos. Los almacenes de datos se modelan con una estructura de base de datos multidimensional, donde cada dimensión corresponde a un atributo o conjunto de atributos que caracterizan unos hechos o medidas agregadas, como por ejemplo la cantidad de alumnos en una determinada carrera en un determinado año académico. Esta representación multidimensional es la adecuada para el procesamiento analítico en línea (OLAP, del inglés *On-Line Analytical Processing*).

Los almacenes de datos permiten al usuario obtener informes agregados por diferentes dimensiones

en tiempo real, a partir de la información detallada almacenada en los mismos. Las herramientas OLAP pueden utilizarse también para comprobar patrones mediante un proceso deductivo, en cambio la minería de datos es un proceso inductivo que permite encontrar dichos patrones. Un almacén de datos es una herramienta muy útil para su uso en las primeras etapas del proceso de $\mathcal{KDD}$, para explorar y comprender los datos, favoreciendo el descubrimiento de conocimiento de las etapas posteriores. Se puede decir que un almacén de datos es muy aconsejable para la minería de datos, pero no imprescindible, en algunos casos, en particular cuando el volumen de datos no es muy grande, se puede trabajar con los datos originales.

También en esta etapa es donde el analista debe sopesar el nivel de agregación e identificación de los datos que puede obtener y la legalidad del uso de los mismos, y tomar decisiones al respecto.

1.2.3. Preparación de los datos

Hernández Orallo en [Hernández Orallo et al., 2004] escribió:

> *"La calidad del conocimiento descubierto no sólo depende del algoritmo de minería utilizado, sino también de la calidad de los datos minados. Por ello, después de la recopilación, el siguiente paso en el proceso de $\mathcal{KDD}$ es seleccionar y preparar el subconjunto de datos que se van a minar, los cuales constituyen lo que se conoce como vista minable".*

La necesidad de construir una vista minable surge principalmente del hecho que la mayoría de los métodos de $\mathcal{DM}$ sólo tratan con una única tabla. También se debe considerar que dada una base de datos relacional con muchas tablas vinculadas por claves foráneas, existen muchas maneras de relacionarlas. La vista minable deja en claro las relaciones que se quieren definir para trabajar sobre ellas.

Esta fase cubre todas las actividades para construir el conjunto final de los datos que serán utilizados en las herramientas de modelado, incluye la selección de las tablas (o archivos), registros y atributos, así como la transformación y limpieza de los datos. Las operaciones del lenguaje relacional SQL (del inglés *Structured Query Language*) son un estándar que se adapta perfectamente para esta tarea.

En esta etapa se utilizan técnicas de limpieza, transformación y reducción de dimensionalidad que aseguren la calidad de los datos y su adecuación para ser utilizados por las herramientas de modelado.

Algunos de los problemas que se atacan en esta fase son:

- *Presencia de valores que no se ajustan al comportamiento general de los datos (outliers):*
  Pueden deberse a errores o a valores correctos que son muy diferentes al resto, algunos algoritmos de Minería de Datos los ignoran, otros los descartan y otros son muy sensibles y su resultado se ve perjudicado. De todas formas, es necesario un análisis de los valores extremos antes de tomar la decisión de eliminarlos, ya que en algunos casos son justamente los valores anómalos los que se quiere detectar. Por ejemplo, puede encontrarse un fraude en las compras realizadas con tarjeta de

crédito analizando observaciones de compras por valores extremadamente mayores a la media de la tarjeta utilizada por un cliente.

- *Presencia de datos faltantes o perdidos (missing values):*
  La ausencia de información puede llevar a resultados poco precisos. Antes de tomar una decisión sobre como tratarlos es necesario entender el significado de los atributos con valores faltantes. Los motivos para el faltante de datos puede tener orígenes muy dispares, pueden deberse a errores en la aplicación de carga de datos o bien provenir de la integración de diferentes fuentes.

- *Transformación y selección de atributos:*
  Es importante que los atributos seleccionados sean relevantes para la tarea de minería de datos. Por ejemplo, en el caso de estudio, si se incluye en el proceso de minería el atributo que corresponde al número de inscripción del alumno, un algoritmo de generación de reglas podría obtener un modelo correcto pero falto de generalidad. Por ejemplo, obtener la regla

  ```
  SI (nro_inscripcion = 2565) ENTONCES (Abandona)
  ```

  que al hacer referencia a un alumno específico no es relevante para la tarea que se desea llevar a cabo.

  En la práctica, si bien existe la posibilidad de recurrir al conocimiento del dominio para realizar el proceso de selección en forma manual, suele tratarse de un problema complejo y por lo tanto es necesario recurrir nuevamente a las técnicas de $\mathcal{DM}$ las cuales proveen algoritmos de selección de características relevantes que operan utilizando diferentes criterios.

- *Construcción de atributos:*
  Se pueden construir atributos que faciliten el proceso de minería aplicando operaciones o funciones a atributos originales. En los casos en que se puede detectar que los atributos originales no poseen un alto poder predictivo por si solos, es deseable buscar expresiones que calculen nuevos valores con más potencia de predicción o descripción. Por ejemplo, el promedio de notas de un alumno puede ser un atributo más rico que las notas individuales que haya obtenido en cada materia.

- *Modificación de tipos de datos:*
  Para facilitar el uso de técnicas que requieren tipos de datos específicos se pueden numerizar datos nominales o discretizar datos numéricos según sea necesario. Por ejemplo, los valores de los atributos referidos a nivel de estudios cursados por los padres del alumno pueden convertirse a números enteros que representen el orden de los títulos obtenidos. El caso contrario consiste en tomar valores numéricos continuos, por ejemplo para el atributo `Nota`, determinar rangos: 0 a 3, 4 a 7 y 8 a 10, y luego dar un valor discreto a cada rango: *"desaprobado"*, *"bueno"*, *"muy bueno"*, como se muestra en la figura 1.3.

- *Selección de la muestra de datos:*
  La característica fundamental que debe reunir la muestra de datos a utilizar es ser representativa del proceso que se quiere modelizar. Generalmente se dispone de un número de ejemplos mucho más grande que los estrictamente necesarios. Como en el caso de los atributos, podría construirse

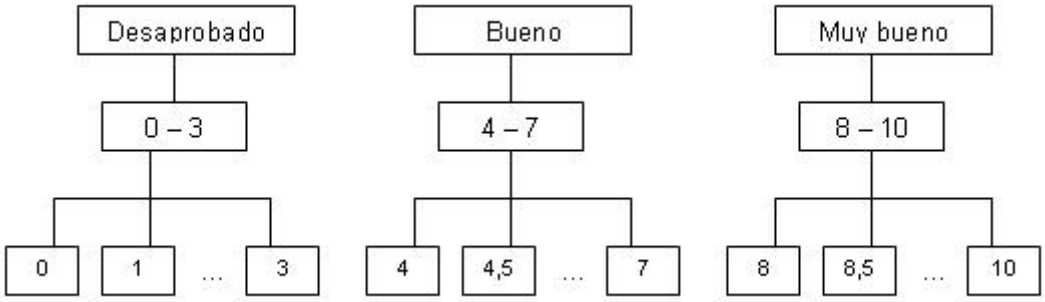

Figura 1.3: Ejemplo de discretización del atributo `Nota`

el modelo a partir de todos los datos disponibles o bien puede utilizarse un subconjunto de ellos. Esto último puede realizarse a través de distintas técnicas y permitiendo, en una primera instancia, disminuir los tiempos de procesamiento.

1.2.4. Modelado

También denominada Minería de Datos, por ser la más característica del $\mathcal{KDD}$, es la fase en la que se seleccionan y aplican diferentes técnicas de modelado, configurando sus parámetros para la obtención de resultados. Usualmente existen varias técnicas para los mismos problemas de $\mathcal{DM}$. Algunas de ellas tienen requerimientos específicos en el formato de los datos, por lo que puede ser necesario un paso atrás hacia la preparación de datos.

Aquí es donde se produce conocimiento nuevo, construyendo modelos basados en los datos recopilados. El modelo describe los patrones y relaciones existentes en los datos. Estos patrones y relaciones son los que se pueden utilizar para entender mejor los datos, predecir comportamientos o explicar situaciones observadas. En esta etapa se deben tomar las siguientes decisiones:

- Elegir la tarea de $\mathcal{DM}$ apropiada para el objetivo del proyecto y para los datos involucrados. Por ejemplo, se puede decidir usar una tarea descriptiva que ayude a conocer con más precisión las características de los alumnos desertores de la Universidad.
- Elegir el tipo de modelo. Por ejemplo, se puede elegir el agrupamiento para obtener grupos de alumnos con características semejantes que puedan ser descriptos apropiadamente.
- Elegir el algoritmo de $\mathcal{DM}$ que resuelva la tarea y ofrezca un modelo resultante. Para tomar esta determinación, en capítulos subsiguientes se describen en detalle los algoritmos plausibles de ser seleccionados para el objetivo planteado.

Las tareas pueden ser predictivas o descriptivas. Dentro de las tareas predictivas se encuentran la clasificación y la regresión. Son tareas descriptivas el agrupamiento, las reglas de asociación y las correlaciones. Se investigan en este trabajo algunas de ellas para aplicarlas al problema objeto de estudio.

Las técnicas de $\mathcal{DM}$ utilizadas para esta fase son de carácter interdisciplinar. Existen técnicas de inferencia estadística, árboles de decisión, redes neuronales, inducción de reglas, aprendizaje basado en instancias, algoritmos genéticos y varias más. Cada uno de estos paradigmas incluye a su vez diferentes algoritmos y variaciones de los mismos, con restricciones que hacen que la efectividad del algoritmo elegido dependa del dominio de aplicación.

La génesis de este trabajo se basa en esta premisa: no existe un método de $\mathcal{DM}$ universal aplicable a cualquier tipo de problema, es por eso que surge la necesidad de evaluar los métodos conocidos en función de la problemática abordada.

El carácter iterativo del proceso de $\mathcal{KDD}$ se ve reflejado con más claridad en la construcción del modelo, ya que será necesario explorar diferentes alternativas hasta dar con aquel que resulte de utilidad para la resolución del problema. Esa búsqueda es la que hace muchas veces necesario retroceder a fases anteriores y hacer cambios en los datos que se están utilizando, o incluso modificar el planteo del problema.

1.2.5. Interpretación y evaluación

Los modelos obtenidos en la fase anterior son interpretados y evaluados. Se revisa la construcción de los mismos a fin de comprobar que se cumplen los objetivos planteados en las fases preliminares. Es indispensable en esta etapa encontrar una manera de medir la calidad de los modelos obtenidos. De la definición de minería de datos se puede ver que los modelos descubiertos deben ser precisos, comprensibles, útiles y novedosos, de estas características deseables, se priorizan unas u otras dependiendo de la aplicación y el uso de los mismos.

Una forma de abordar esta etapa es dividir los datos en dos subconjuntos: el conjunto de entrenamiento, que se utiliza para construir y entrenar un modelo, y el conjunto de prueba, que es usado para validar su efectividad. La separación permite garantizar que la validación de la precisión del modelo es una medida independiente. Existen diferentes técnicas para efectuar la división en subconjuntos, dependiendo de la cantidad total de datos de que se dispone.

Una técnica básica de evaluación es la validación simple, que reserva un porcentaje (normalmente entre un 5 y un 50 %) de observaciones de la vista minable como conjunto de prueba, mediante una selección aleatoria. Estos datos no intervienen en la generación del modelo.

En el caso que los datos sean escasos, se puede utilizar el método de validación cruzada que divide los datos aleatoriamente en dos conjuntos equitativos, luego utiliza el primer conjunto para la construcción del modelo y el segundo conjunto para validarlo, y posteriormente realiza la misma tarea con los subconjuntos cambiados de rol. Un método muy utilizado es la validación cruzada con n pliegues, que divide los datos en n grupos, reserva uno de los grupos para la prueba y con los otros $n-1$ grupos restantes (todos juntos) construye el modelo y lo usa para predecir los datos del grupo reservado. Este proceso se repite n veces, dejando cada vez un grupo diferente para la prueba. Ambos métodos de validación cruzada calculan la tasa de error en cada pasada y finalmente construyen un modelo con todos los datos cuya tasa

de error se estima promediando las obtenidas en cada pasada.

Existe otra técnica de evaluación conocida como *bootstrapping* que construye numerosos conjuntos de datos por muestreo con reemplazo, es decir que se van seleccionando observaciones del conjunto original pudiendo repetirse. Luego construye un modelo con cada conjunto y lo evalúa contra el conjunto de prueba, que son los datos sobrantes de cada muestreo, el error final se calcula promediando los errores para cada muestreo.

Una vez aplicada la técnica de evaluación, existen diferentes medidas para evaluar los modelos, dependiendo de la tarea de $\mathcal{DM}$. Para tareas de tipo predictivo se puede medir la precisión predictiva, que se calcula como el número de instancias del conjunto de prueba que el modelo predice correctamente dividido por el número de instancias totales del conjunto de prueba. Para tareas descriptivas como el agrupamiento, las medidas de evaluación suelen depender del método utilizado y se formalizan en función de las medidas de distancia entre instancias como se verá al describir estos algoritmos más adelante.

En esta etapa es crítico determinar si partes importantes de la realidad han sido lo suficientemente consideradas, y se debe decidir sobre la utilización de los resultados del proceso de $\mathcal{DM}$. Las tareas involucran evaluar los resultados, revisar los procesos y determinar los pasos a seguir basados en el modelo obtenido.

La naturaleza iterativa del $\mathcal{DM}$ puede llevar en esta etapa a la revisión de etapas anteriores y pueden surgir nuevas preguntas a responder que hagan que el proyecto retorne a la fase de conocimiento del dominio a fin de poder responderlas.

1.2.6. Difusión y uso de los resultados

La creación del modelo no implica la finalización del proyecto. El conocimiento obtenido debe ser organizado y presentado de manera que pueda ser comprendido y utilizado por el usuario final.

Los modelos construidos pueden utilizarse para decidir acciones basándose en sus resultados. También pueden utilizarse para aplicarlos a nuevos conjuntos de datos e incluso incorporarlos a aplicaciones que utilice la organización.

Esta fase puede ser tan simple como la generación de un informe o tan compleja como la aplicación del modelo a diferentes juegos de datos, de manera que dé lugar a fases complementarias que implementen un proceso iterativo de $\mathcal{DM}$ repetible tantas veces como sea necesario para concretar los objetivos.

La tarea importante de esta fase consiste en que el usuario entienda los resultados y pueda utilizar los modelos creados. También es relevante medir la evolución del modelo, aún cuando éste funcione bien, se debe comprobar continuamente su efectividad, principalmente debido a que la realidad puede cambiar con el tiempo.

1.2.7. Implementación de medidas basadas en el conocimiento obtenido

Cuando la fase de uso de resultados genera una clase de conocimiento que habilita al usuario a ejecutar acciones en pos de resolver el problema planteado originalmente, se produce una etapa de implementación de medidas que debe llevar a cabo la organización. Estas medidas tendrán como objetivo mejorar o corregir la realidad descubierta a través del modelado, actuando directamente sobre la organización.

Si bien estas tareas no son parte de la metodología interna del $\mathcal{DM}$, sino más bien son la implementación de decisiones generadas por el resultado de los modelos, deben ser tenidas en cuenta para retroalimentar el ciclo completo de resolución del problema.

1.2.8. Medición de resultados

Luego de la implementación de las medidas de la fase anterior, es posible la utilización del $\mathcal{DM}$ para medir los resultados alcanzados por esas acciones.

En esta fase se pueden volver a ejecutar los modelos para compararlos con los obtenidos en la primera iteración y de esa manera conseguir mediciones concretas del éxito o fracaso de las medidas tomadas.

Capítulo 2

Preparación de datos y Técnicas de Selección de Atributos

En este capítulo se hará hincapié en las tareas correspondientes a las primeras fases del proceso de $\mathcal{KDD}$. Son precisamente las que insumen la mayor cantidad de tiempo ya que de ellas depende el resultado a obtener. Antes de aplicar una técnica de $\mathcal{DM}$ es necesario contar con la *vista minable* adecuada. Esto implica no sólo acondicionar la información disponible sino seleccionar los aspectos que se desean considerar de los datos de entrada.

Las tareas involucradas en estas fases han sido descriptas en el capítulo anterior. En esta oportunidad se detallará su aplicación al problema de la deserción de alumnos de la UNRN. El objetivo principal es obtener una vista minable con información de interés de los alumnos, la que se utilizará en el resto del trabajo.

2.1. Comprensión del dominio

Como se expresó inicialmente, la organización objeto de estudio es la Universidad Nacional de Río Negro, que habiendo sido creada en el año 2008, comenzó a dictar sus carreras de grado en el año 2009. En la actualidad consta de cuatro sedes (Andina, Alto Valle, Valle Medio y Atlántica) en las que se dictan un total de 60 carreras de grado.

Desde sus inicios ha sido preocupación de las autoridades y de los docentes de las diferentes carreras, el alto índice de deserción y desgranamiento que se observa, a pesar de los pocos años de vida de la Institución. La comunidad educativa coincide en la necesidad de hacer esfuerzos para revertir esta situación y cualquier tipo de medidas que se adopten deben estar basadas en información útil para la rápida toma de decisiones.

El objetivo principal es poder determinar a priori situaciones potenciales de fracaso académico con el fin de tomar medidas tendientes a minimizar el problema.

2.2. Recopilación e integración de datos

La UNRN utiliza el sistema SIU-Guaraní para la gestión académica. Este sistema almacena los datos en una base de datos relacional.

> *"El SIU-Guaraní registra y administra todas las actividades académicas de la universidad, desde que los alumnos ingresan como aspirantes hasta que obtienen el diploma. Fue concebido para administrar la gestión de alumnos en forma segura, con la finalidad de obtener información consistente para los niveles operativos y directivos."* (Consorcio de Universidades SIU).

El relevamiento realizado con los responsables de la administración de los datos del Rectorado de UNRN dejó en claro que las tablas del modelo de datos del SIU-Guaraní son la fuente principal de información de la Universidad en lo que a alumnos se refiere y que por el momento no se utilizan fuentes externas que provean otra información relevante.

La recopilación de datos se reduce entonces a la obtención de la definición de todas las tablas del modelo relacional y los datos de cada una de ellas. Cabe aclarar que es necesario proteger la identidad de los alumnos cuyos datos se van a analizar. Por este motivo se decidió trabajar con la identificación que provee el sistema para las tablas que lo componen, es decir, el número de inscripción. De esta manera, no es indispensable contar con los datos personales (apellido, nombre, tipo y número de documento, nombre y apellido de los padres), por lo que fueron exceptuados de la copia de la base de datos utilizada. Esta medida mantiene la confidencialidad de los datos sensibles del alumnado, permitiendo de todas formas utilizar el resto de la información para continuar los procesos de Minería de Datos.

Una vez definida y cargada la base de datos que será fuente de información para todo el proyecto, se procede a estudiar el modelo de datos y entender las relaciones entre las tablas. Como primera aproximación en integración, se obtiene una tabla maestra con todos los datos relacionados a los alumnos.

En $\mathcal{DM}$ los datos generalmente están en forma de tabla, en donde cada fila representa el objeto de interés, en este caso un alumno inscripto en una carrera de la UNRN y cada columna contiene información acerca de algún atributo del alumno. Por ejemplo en el caso de estudio, algunos atributos son la edad, el lugar de procedencia, el colegio secundario al que asistió, etc. Como los datos del SIU-Guaraní están almacenados para su uso transaccional es necesario un trabajo de ensamblado previo a fin de obtener la tabla mencionada, utilizando sentencias SQL. A partir de allí comienza el trabajo de preparación de datos para $\mathcal{DM}$.

Es importante destacar en este punto que los alumnos pueden estar inscriptos en más de una carrera a la vez; es por eso que la tabla con la que se inicia el trabajo tiene una fila para cada par alumno-carrera, de manera que un alumno inscripto en dos carreras aparecerá dos veces en la tabla, con sus datos personales repetidos y sus datos académicos pertenecientes a cada carrera en la fila correspondiente.

La investigación realizada para la presente tesis se comenzó en el año 2012, de manera tal que la implementación de todas las tareas de comprensión del dominio, recopilación e integración de datos y

su posterior preparación para los algoritmos de $\mathcal{DM}$ se realizó inicialmente con los datos disponibles en la base de datos del SIU-guaraní hasta ese momento. En el transcurso del trabajo, cuando se concluyó un nuevo año académico (a principios de 2013), se decidió agregar los datos correspondientes a este período, de manera de contar con más información para alimentar el desarrollo de los modelos. Esta nueva integración de datos consistió en incorporar registros correspondientes a alumnos inscriptos en el nuevo año académico y modificar datos de los alumnos ya existentes, por cambios en los mismos, por ejemplo, materias aprobadas en el año.

La decisión de incorporar los nuevos datos a la investigación produce que algunas de las implementaciones descriptas en el presente documento se refieran al primer conjunto de datos y otras al conjunto de datos completo con el que se cuenta en la actualidad, por lo que a lo largo del texto será necesario aclarar con qué grupo de ejemplos se realizó cada tarea.

Como se explicó anteriormente, fue necesario un trabajo de ensamblado y transformación de datos para obtener la tabla a utilizar en $\mathcal{DM}$. Esta tarea se vio simplificada en la segunda etapa dado que fue posible utilizar las mismas sentencias SQL de transformación escritas para el primer ensamble. Esto fue posible dado que la estructura de la base de datos no cambió entre el primer vuelco de datos y el segundo.

2.3. Preparación de los datos

La preparación de datos en general tiene como objetivo principal organizar y representar las vistas minables a las que se les pueda aplicar las herramientas concretas de Minería de Datos. Esta organización de los datos debe ir acompañada de una limpieza e integración de los mismos para que estén en condiciones para su análisis. Además, debido a las características propias de las técnicas de Minería de Datos, es necesario generalmente hacer una transformación de los datos antes de utilizarlos.

Lo primero que se puede hacer es un resumen de las características o informe de estado de los atributos. Esa misma tabla será de utilidad para registrar los cambios que se vayan realizando en los datos a medida que pasan por el proceso de preparación.

A modo de ejemplo, la tabla 2.1 muestra una porción de la tabla original para algunos de los atributos de los alumnos. En el Apéndice A del presente trabajo se puede encontrar la lista completa de características analizadas, luego de todas las transformaciones efectuadas.

En la mayoría de las bases de datos existen problemas de calidad de datos que deben ser detectados antes de utilizarlos para hacer $\mathcal{DM}$, como los valores anómalos o faltantes.

2.3.1. Detección de valores anómalos

Algunos problemas saltan a la vista al analizar la tabla. Por ejemplo, el valor máximo del atributo `anio_nacim`: 2091 es claramente un error en los datos. Una herramienta básica pero muy útil para detectar anomalías en la distribución de frecuencias de un atributo es el histograma. Por ejemplo,

Atributo	Tipo	Cardinalidad	Nulos	%nulos	Mínimo	Máximo
unidad_academica	nominal	1	0	0,00 %		
carrera	nominal	60	0	0,00 %		
nro_inscripcion		clave	0	0,00 %		
legajo		clave	0	0,00 %		
Plan	nominal	7	0	0,00 %		
sede	nominal	7	0	0,00 %		
sexo	nominal	2	0	0,00 %		
anio_nacim	numérico	59	18	0,14 %	1925	2091
Nacionalidad	nominal	4	0	0,00 %		
loc_nacimiento	nominal	828	18	0,14 %		
Colegio_secundario	nominal	1673	2259	17,20 %		
Anio_egreso_sec	numérico	75	363	2,76 %	0	2088
periodo_inscripcion	numerico	4	0	0,00 %	2009	2012
tipo_residencia	nominal	1	10103	76,95 %		
cnt_readmisiones	numerico	1	0	0,00 %	0	0

Tabla 2.1: Listado parcial de atributos originales correspondiente a la situación personal de los alumnos de la UNRN

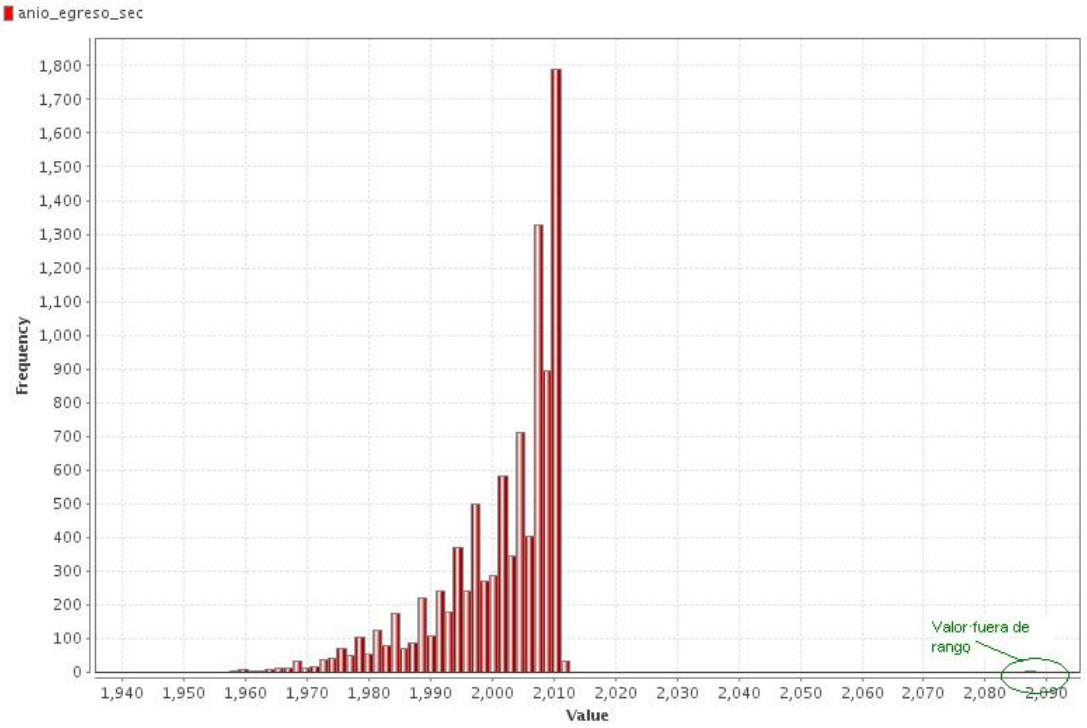

Figura 2.1: Histograma correspondiente al atributo `anio_egreso_sec`

la figura 2.1 permite visualizar fácilmente los valores fuera del rango razonable para el atributo `anio_egreso_sec`.

En este caso el conocimiento del dominio permite además decidir cuales valores de los extremos del gráfico son incorrectos (teniendo en cuenta la edad mínima de los estudiantes universitarios y la condición de haber terminado el secundario para ser inscripto).

Cuando se trata de atributos numéricos, también pueden utilizarse los diagramas de caja. En ellos no sólo se representan los cuartiles correspondientes sino que también se establece un criterio para determinar si existen valores fuera de rango. Generalmente se considera una proporción de la distancia intercuartil (la diferencia entre el tercer y primer cuartil) para identificar dichos valores anómalos. La figura 2.2 ilustra el diagrama de caja correspondiente al atributo `anio_egreso_sec`.

2.3.2. Tratamiento de valores faltantes

El problema de los datos faltantes debe ser resuelto antes de la aplicación de los métodos de $\mathcal{DM}$, ya que algunos algoritmos son sensibles a este tipo de datos.

El primer paso es la detección de los mismos, dado que no necesariamente son valores nulos. Por ejemplo, el caso del atributo `loc_nacimiento`, que puede tomar el valor `Indeterminada`.

Una vez realizada la detección, las posibles acciones son las siguientes:

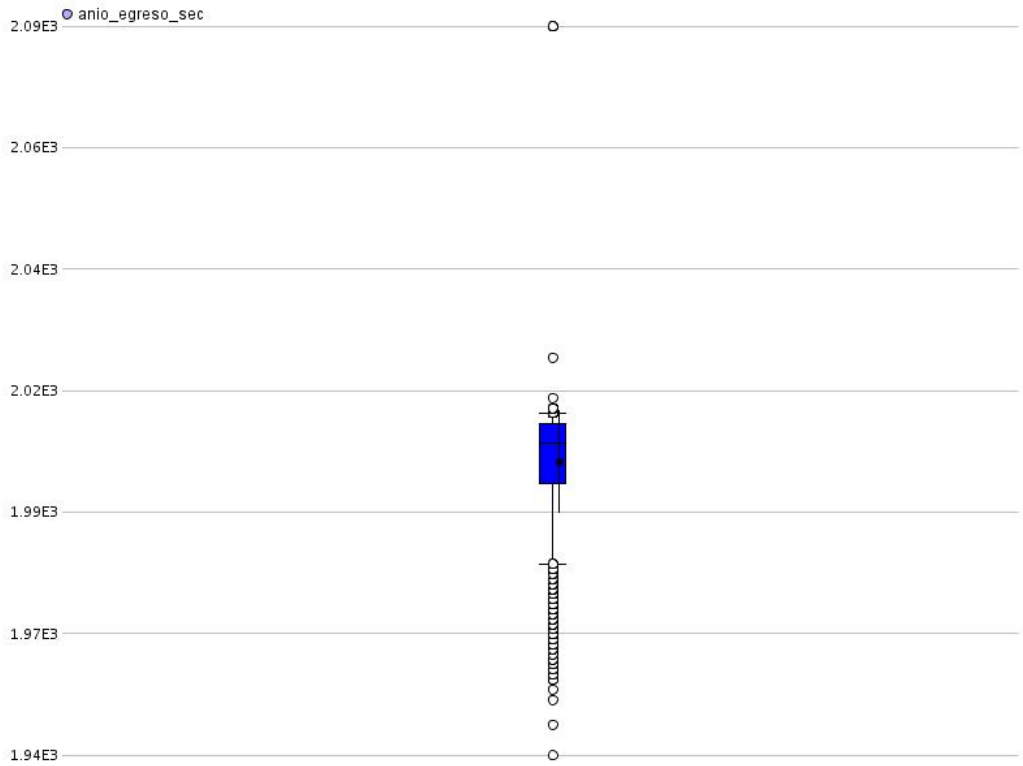

Figura 2.2: Diagrama de caja correspondiente al atributo `anio_egreso_sec`

- *Ignorar:* Para el caso de algoritmos robustos a datos faltantes, como árboles de decisión, esta puede ser una acción factible ya que no implica ignorar la tupla completa. Otros métodos, como por ejemplo los basados en redes neuronales, no permiten trabajar con datos faltantes y por lo tanto, esta acción no puede aplicarse. La opción adoptada en el caso de estudio es mantener una vista minable paralela con los datos faltantes, de manera de poder decidir según el algoritmo si utilizarla con o sin datos faltantes.

- *Eliminar la columna:* Esta acción implica eliminar por completo el atributo. Es el caso de `tipo_residencia` que con 10103 valores nulos, no parece aportar información útil.

- *Filtrar la fila:* Esta opción puede producir sesgo en los datos, en los casos en que un dato faltante esté indicando un caso particular que puede ser de interés. Por ejemplo, en algunos atributos referidos a la condición laboral de los alumnos, la falta de los mismos puede indicar que el alumno no trabaja.

- *Reemplazar el valor:* Para poder utilizar los algoritmos que no son robustos a la presencia de nulos, se pueden reemplazar los valores faltantes automáticamente por algún valor que preserve la media o la varianza, en caso de valores numéricos, o por la moda en el caso de valores nominales. Otra opción es predecir los valores con algún algoritmo de $\mathcal{DM}$ a partir de otros ejemplos.

2.3.3. Transformación y Selección de atributos

A partir del conocimiento del dominio se trabaja en esta etapa en disminuir la dimensionalidad del problema.

> *"En principio, si disponemos de un problema con muchos atributos puede parecer que siempre será mejor que cuando tenemos pocos atributos ... El problema es que la gran mayoría de métodos de $\mathcal{DM}$ pueden perderse entre tantas características en un espacio que al tener alta dimensionalidad resulta estar más desierto (especialmente si las hay irrelevantes, redundantes o con valores erróneos) y obtener modelos que se ajustan a particularidades de los datos de entrenamiento y no de los datos en general. Esto ocurre especialmente cuando existen muchas dimensiones pero no tenemos un número suficiente de ejemplos para reducir los grados de libertad."* [Hernández Orallo et al., 2004]

Se describen aquí algunas de las transformaciones realizadas:

- *Eliminación de atributos constantes:* El atributo `cnt_readmisiones` es siempre 0, dado que en el corto tiempo de vida de la UNRN nunca se ha ejecutado el proceso que determina cuales alumnos han perdido la regularidad, caso en el cual deberían ser readmitidos. Claramente este atributo puede eliminarse. Otro caso es el atributo `unidad_académica`, que toma el valor `UNRN` para todas las instancias.

- *Eliminación de claves candidatas:* Es regla general eliminar atributos que puedan ser clave primaria de la tabla. Es por eso que no se utilizan en los modelos los atributos `legajo` y `nro_inscripcion`.

- *Generalización de atributos:* en el caso en que un atributo presente muchos valores, se pueden considerar generalizaciones que provean un número menor de valores distintos incrementando su capacidad predictiva. En los datos originales, se registra `colegio_secundario` con 1673 valores diferentes que representan los nombres de los colegios en los que los alumnos cursaron el nivel medio. Se realiza una transformación de los valores de este atributo de manera que representen el tipo de colegio (*"Público"* o *"Privado"*). Otra transformación similar realizada sobre el `título_secundario` convierte los excesivos valores diferentes en categorías: *"Bachiller"*, *"Técnico"*, *"Perito Mercantil"* y *"Otros"*.

- *Eliminación de atributos no generalizables:* Cuando un atributo tiene muchos valores y no se puede generalizar, es conveniente eliminarlo de la vista minable. Este es el caso la información personal del alumno como `domicilio`, `telefono`, etc.

- *Sumarización:* La fuente de datos utilizada registra en tablas separadas los datos de las actas de examen y de cursado, de las que puede obtenerse la historia académica de los alumnos. En un proceso constructivo se trabaja sobre estos datos para generar atributos que describan la evolución del alumno en términos de cantidades. Parte de este proceso incluye la sumarización o agregación

de datos que permita mostrar los mismos de manera más resumida. Este aspecto favorece la detección de patrones. Los atributos obtenidos de esta manera se explican en detalle en el apartado de construcción de atributos.

- *Eliminación de atributos dependientes:* Al desnormalizar la base de datos relacional del SIU-Guaraní, aparecen atributos unidos por dependencias funcionales, como el caso del código postal y la localidad de nacimiento, lo que obliga a eliminar uno de ellos, en este caso `cod_postal`, por ser el menos descriptivo.
- *Reducción de cardinalidad por jerarquías:* Como otra aplicación de la generalización, en este caso por jerarquías, se transforma el atributo `loc_nacimiento` (localidad de nacimiento) en `lugar_nacimiento`, regionalizando las localidades por departamentos de la Provincia de Río Negro y agregando un valor *"Fuera de Río Negro"*.
- *Métodos de filtro:* Se filtran los atributos irrelevantes mediante técnicas estadísticas (medidas de información, distancia, dependencia, etc.). En la sección 5.1 se analiza e implementa un método de selección de filtro para los datos de la UNRN.
- *Métodos basados en modelo:* También denominados métodos de envolvente (*wrapper*), se evalúa la selección de atributos respecto a la calidad de un modelo de $\mathcal{DM}$ extraído a partir de los datos. En la sección 5.1 se realiza una selección de atributos por medio de un wrapper.

Las transformaciones y selecciones posibles para los datos no se agotan con la enumeración anterior. La naturaleza iterativa del proceso de $\mathcal{DM}$ puede llevar a realizar nuevas transformaciones o selecciones que ayuden a la expresividad de los datos y a la reducción de dimensionalidad necesaria para obtener modelos que aporten a la descripción del problema.

2.3.4. Selección de la muestra de datos

El problema de la selección de datos puede tratarse desde dos dimensiones: hay que decidir que atributos (columnas) se necesitan y cuantas instancias (filas) se van a utilizar.

Si se define la vista minable con todos los atributos existentes y todos los ejemplos que se obtuvieron, en algunos casos el tamaño de la tabla resultante puede ser excesivo para algunas técnicas de $\mathcal{DM}$. Por otro lado, una muy alta dimensionalidad puede llevar a resultados poco expresivos, justamente por la cantidad de variables involucradas.

La reducción de la dimensionalidad o selección vertical, donde se eliminan algunas características de los individuos ya se abordó en el apartado anterior y debe tenerse en cuenta cada vez que se advierta la posibilidad de expresar el problema de manera precisa con menor cantidad de variables. La relevancia de la correcta selección de atributos para la obtención de mejores resultados hace que se vuelva a considerar en sucesivas depuraciones, utilizando técnicas más sofisticadas, que se describen posteriormente.

Cuando se trata de reducir el número de ejemplos a evaluar, también denominado selección horizontal o muestreo, aparece el problema de elegir las filas a utilizar. Al respecto pueden darse dos situaciones:

- Se dispone de toda la población: En este caso se debe determinar qué cantidad de datos son necesarios y como hacer la muestra. En muchos casos una muestra aleatoria no es la mejor elección, sobre todo si se quiere escoger un mínimo de individuos de cada tipo. Por ejemplo en el caso de estudio: edades, procedencias, evolución académica, etc. Esto podría dar lugar a la utilización de un muestreo estratificado o balanceado, que garantiza suficientes elementos en todos los estratos o grupos de interés.

  De todas formas, ésta es la situación ideal, ya que se dispone de la población total.

- Los datos son ya una muestra de la realidad: en este caso disminuye la libertad para la elección de la muestra (por falta de disponibilidad de más datos).

2.3.5. Construcción de atributos

La creación de características consiste en crear nuevos atributos con el objetivo de mejorar la calidad y comprensión del conocimiento extraído. Una forma de abordar las combinaciones posibles es el modelo *"knowledge-driven"* (guiado por el conocimiento).

Algunos ejemplos de construcción de atributos utilizados en la resolución del problema:

- Con los datos de las actas de examen y cursada, que proveen numerosos atributos que amplían mucho la dimensionalidad del problema se procede a construir atributos mediante operaciones matemáticas de conteo y sumarización. Se obtienen atributos promediados para los años académicos analizados en los que el alumno ya estaba inscripto (2009 a 2011 en la primer carga y posteriormente 2012 para los datos agregados en la segunda carga de ejemplos).

  Esta situación surge del hecho que la UNRN inició el dictado de carreras en el año 2009 y los datos sobre los que se trabajó en primera instancia fueron de principios de 2012, con lo cual no había registro de exámenes ni cursadas del año académico 2012, si bien existían alumnos inscriptos en dicho año, situación que se traslada al año 2013 en la segunda carga de datos.

  Los atributos generados son:

  - Cantidad promedio de cursadas a las que se inscribió el alumno y luego las abandonó, desaprobó, aprobó o promocionó. Estos atributos fueron denominados `prom_cnt_abandono`, `prom_cnt_desaprobo`, `prom_cnt_aprobo` y `prom_cnt_promociono`, respectivamente.
  - Cantidad promedio de finales que se inscribió y luego desaprobó, aprobó o no se presentó; denominados `prom_cnt_fin_desaprobo`, `prom_cnt_fin_aprobo` y `prom_cnt_fin_ausente`, respectivamente.
  - Promedio de notas en examenes finales (aprobados y desaprobados); `prom_notas_finales`.

- Se crea un campo que resume el estado académico del alumno teniendo en cuenta las condiciones de pérdida de regularidad de los alumnos de la UNRN:

  PÉRDIDA DE LA CONDICIÓN DE ALUMNO.

> *ARTÍCULO 12°. CASOS. ... Se perderá la condición de alumno por las siguientes causas:*
>
> *a. Haber dejado transcurrir un (1) año lectivo, entendiéndose por tal el lapso comprendido entre el 1 de marzo y el 30 de diciembre, sin aprobar por lo menos dos (2) asignaturas correspondientes a la carrera en la que se ha inscripto.*
>
> *b. Haber dejado más del triple de los años previstos por el plan de estudios para la respectiva carrera, sin haber aprobado la totalidad de las asignaturas comprendidas en dicho plan ...*
>
> *c. Haber sido aplazado, en los exámenes de las asignaturas, un número de veces que supere a la mitad más una* $(1/2 + 1)$ *de las materias que integran el plan de estudios respectivo, computándose a tal fin, en su caso, las calificaciones obtenidas en otras Universidades o Carreras*. Universidad nacional de Río Negro. Proyecto Institucional. Anexo VI: Reglamento de alumnos.

De esta manera se registra como *"Pérdida de Regularidad"* si se cumplen las condiciones explicadas en el Reglamento. El estado se registra como *"Abandono"* cuando el alumno no realizó ninguna actividad académica en todo el transcurso de un año. Según el Plan de Retención de alumnos LISIS, se considera *"Deserción de la carrera"* al abandono total o definitivo de la carrera durante el ciclo lectivo por falta de cumplimiento en todas las asignaturas o factores externos.

Los alumnos que no caen en ninguna de estas categorías aún pueden tener dos estados diferentes: *"Ingresantes"* cuando no tienen historia académica por ser el corriente su año de ingreso, y *"Cursa normalmente"* para el resto de los casos.

- Otro caso de un campo construido por condición de igualdad es `Loc_perlect_distinta_loc_proc`, que indica con valor *"S"* si la localidad de residencia del alumno es diferente de la localidad de procedencia, distinguiendo así a aquellos estudiantes que residen lejos de su familia para cursar su carrera.

2.3.6. Modificación de tipos de datos

El tipo de los atributos es una característica que determina en gran medida la forma en que van a ser tratados por las herramientas de minería, por lo tanto, modificaciones en este sentido pueden ser útiles en algunos casos. Las transformaciones de tipo son:

- *Discretización:* También llamada *"binning"*, se trata de convertir un valor numérico en un valor nominal ordenado, representando rangos. Algunas variables nominales presentes en los datos de alumnos ya están discretizadas en el sistema de origen. Por ejemplo `alu_trab_remmon` representa la remuneración recibida por el alumno en su trabajo y está representada por rangos de $. En este caso, el proceso de transformación redujo esos rangos a un número menor de ellos, teniendo en cuenta la frecuencia de los valores. Hay diversas motivaciones para discretizar variables, una de ellas es la utilización de técnicas como por ejemplo, algunas que generan reglas de asociación que sólo operan sobre atributos nominales.

- *Numerización:* Es el proceso inverso a la discretización. Se utiliza cuando el método de $\mathcal{DM}$ no acepta valores nominales. La forma de realizarlo es crear variables indicadores (o *dummy*): si una variable nominal tiene x valores, se crean x variables *dummy* donde cada una tomará el valor 1 si la variable nominal toma ese valor, y 0 en caso contrario. En algunos casos, si las variables 1 hasta $x - 1$ tienen valor 0, se deduce que la variable nominal toma el valor x y por lo tanto sólo son necesarias $x - 1$ variables. Otra forma de numerización, en este caso con orden, se puede utilizar cuando los valores nominales implican un ordenamiento. Por ejemplo, la variable `ult_est_cur_madre`, que se refiere al nivel de estudios alcanzados por la madre del alumno (primario incompleto, primario completo, secundario incompleto, secundario completo, universitario incompleto, etc.), puede numerizarse manteniendo el significado del ordenamiento.
- *Normalización de rango:* Algunos algoritmos requieren que los atributos se normalicen al mismo rango. En particular en los algoritmos basados en distancias es importante la normalización, ya que las distancias debidas a diferencias de un atributo que va entre 0 y 100 serán mucho mayores que las distancias debidas a diferencias de un atributo que va entre 0 y 10. La normalización más común es la lineal uniforme, que normaliza a una escala entre 0 y 1. La transformación z normaliza a un conjunto de datos nuevo que tiene como media 0 y como desviación estándar 1.

2.4. Selección de atributos o características

Uno de los problemas centrales en la Minería de Datos es identificar un conjunto representativo de características adecuadas para construir un modelo para una tarea en particular. Hay muchos factores que afectan el éxito de una tarea de $\mathcal{DM}$, la calidad de los datos de ejemplo es uno muy importante. En teoría, tener más características debería resultar en una mayor potencia descriptiva o predictiva, sin embargo, la experiencia práctica ha demostrado que no siempre es éste el caso. Los problemas con una alta dimensionalidad, cantidad limitada de ejemplos disponibles y mucha información redundante o irrelevante son difíciles de tratar.

La selección de características es el proceso de identificar y remover la información redundante e irrelevante en la mayor medida posible. Esto reduce la dimensionalidad de los datos y permite a los algoritmos trabajar más rápido y con mayor efectividad. La selección tiene incidencia positiva en la precisión de clasificaciones y genera una representación más compacta y fácil de interpretar de los resultados obtenidos. En resumen, el proceso de selección de atributos trata de elegir el subconjunto más pequeño de atributos a fin de mejorar el entendimiento, el desempeño predictivo y la eficiencia del modelo.

Una característica se considera relevante si no es irrelevante o redundante. Una característica o atributo es irrelevante si no afecta de ninguna forma al objetivo final y es redundante si no añade nada nuevo al objetivo final.

El proceso de selección de atributos involucra cuatro pasos:

- *Generación de candidatos (subconjuntos)*: Lo que requiere una estrategia de búsqueda que mejore

el desempeño de la simple selección aleatoria de subconjuntos de atributos. En la sección siguiente se analizan algunas opciones.

- *Evaluación de los subconjuntos de características seleccionados*: Esta etapa implica definir un criterio para realizar esta tarea.
- *Criterio de finalización:* la búsqueda del conjunto de características óptimo puede finalizar por la propia estrategia de búsqueda (en caso de no poder proveer mejores soluciones) o porque ha transcurrido la cantidad máxima de intentos o porque se ha alcanzado una cota de error predeterminada, entre otras.
- *Validación de resultados:* Dado que generalmente se desconoce cual es el subconjunto mínimo de atributos relevantes, una opción para validar los resultados obtenidos es analizar el modelo con y sin la selección de atributos.

2.4.1. Tipos de algoritmos de selección

En general, los algoritmos de selección de atributos se distinguen por su forma de evaluar atributos y pueden clasificarse en:

- *Wrappers:* utilizan el algoritmo de $\mathcal{DM}$ que se va a aplicar finalmente a los datos para evaluar la relevancia de los atributos. Parten del razonamiento que el método que se usará finalmente para la predicción debería proveer la mejor estimación de características. Estos algoritmos tienden a ser lentos dado que deben llamar repetidamente al algoritmo elegido.

  La idea detrás de los modelos wrapper es sencilla: el algoritmo inductivo (aquél que se va a utilizar para la tarea de clasificación) es considerado como una caja negra. Es decir que no es necesario conocer el algoritmo, solamente se necesita conocer su interface [Kohavi and John, 1997]. Se ejecuta el algoritmo sobre el conjunto de datos con diferentes conjuntos de características y luego se elige el subconjunto con la mejor evaluación. Luego se evalúa el subconjunto obtenido para un subconjunto de datos no utilizado durante la búsqueda. La figura 2.3 ilustra este proceso.
- *Filtros (filters):* seleccionan/evalúan los atributos en forma independiente del algoritmo de aprendizaje (ver figura 2.4).
- *Híbridos:* usan una combinación de los dos criterios de evaluación en diferentes etapas del proceso de búsqueda.

El algoritmo 1 presenta un pseudo-código para seleccionar atributos. Si el método para evaluar un subconjunto de atributos, M, es independiente del algoritmo de aprendizaje entonces es un filtro y si M es un algoritmo de aprendizaje, es un wrapper.

Nótese que el conjunto de atributos seleccionados dependerá de la estrategia de búsqueda y el criterio de parada utilizados. Diferentes criterios de evaluación (independientes del algoritmo de aprendizaje)

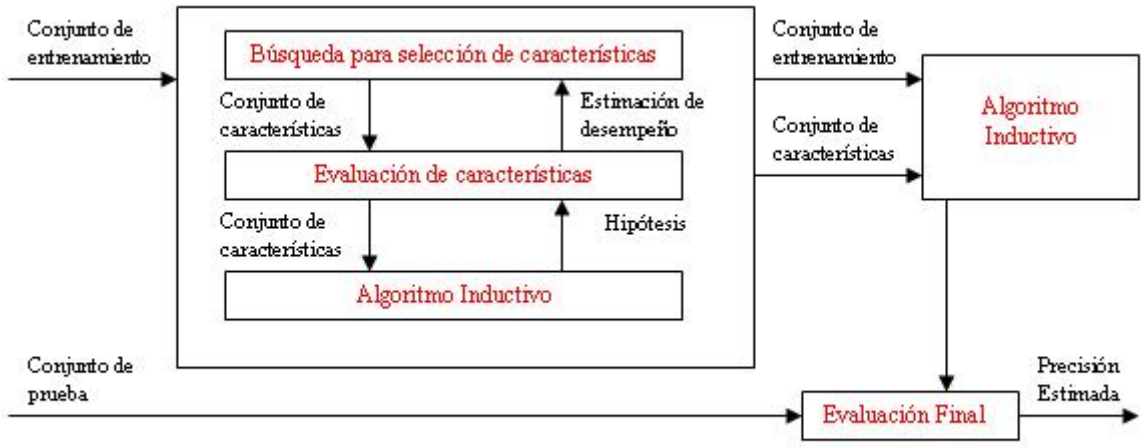

Figura 2.3: Esquema genérico de algoritmo tipo *wrapper*

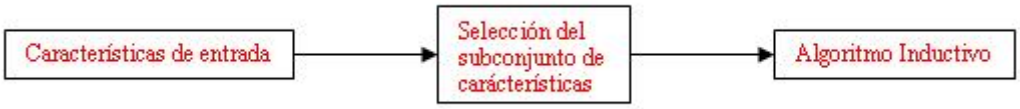

Figura 2.4: Esquema genérico de algoritmo tipo filtro

Algoritmo 1: Pseudo-código para seleccionar atributos. Si el método para evaluar un subconjunto de atributos, M, es independiente del algoritmo de aprendizaje entonces es un filtro y si M es un algoritmo de aprendizaje, es un wrapper.

$D \leftarrow$ conj.de datos de entrenamiento ;
$S_{mejor} \leftarrow$ conj.inicial de atributos seleccionados ;
$Aptitud_{mejor} = eval(S_{mejor}, D, M)$ { *evalúa* S_{mejor} *usando* M }
repeat
  $S_{nuevo} = genera(S_{mejor}, D)$
  $Aptitud_{nuevo} = eval(S_{nuevo}, D, M)$
  if $Aptitud_{nuevo}$ *es mejor que* $Aptitud_{mejor}$ **then**
    $Aptitud_{mejor} = Aptitud_{nuevo}$
    $S_{mejor} = S_{nuevo}$
until *se cumpla la condición de parada*;
Output : S_{mejor}

darán lugar a diferentes algoritmos filtros. Diferentes algoritmos de aprendizaje generarán diferentes algoritmos wrapper.

Como se dijo anteriormente, el otro tipo de algoritmo es el híbrido que combina el criterio de evaluación de ambos. Este método comienza con un conjunto de atributos inicial que según el criterio de selección empleado puede estar vacío o formado por todos los atributos. Luego modifica la cardinalidad incrementando o decrementado en 1 atributo según el método de construcción y genera todos los subconjuntos posibles para dicha cardinalidad. Cada uno de estos subconjuntos se evalúa utilizando un mismo filtro y se selecciona el mejor. El subconjunto seleccionado se evalúa nuevamente utilizando un wrapper y si su desempeño es mejor que los anteriores, se guarda. Este proceso se repite hasta encontrar el criterio de parada que en general suele ser la no mejora del desempeño del subconjunto de atributos. Finalmente retorna el mejor subconjunto de atributos encontrado. El algoritmo 2 resume lo antes dicho. La sección 2.4.2 describe distintas maneras de construir los subconjuntos de atributos de una cardinalidad dada.

Algoritmo 2: Pseudo-código de un algoritmo híbrido para seleccionar atributos

$D \leftarrow$ conj.de datos de entrenamiento
$S_{mejor} \leftarrow$1 conjunto inicial de atributos
$Aptitud_{mejor} = 0$
repeat
  // Modificar la cardinalidad C del subconjunto de atributos
  $S_{mejorC} = \emptyset$
  $Aptitud_{mejorC} = 0$
  for *cada subconj.* S_{nuevo} *de atributos de cardinalidad C* **do**
    $Aptitud_{nuevo} = eval(S_{nuevo}, D, M_1)$ *// M_1 es un filtro*
    if $Aptitud_{nuevo} > Aptitud_{mejorC}$ **then**
      $Aptitud_{mejorC} = Aptitud_{nuevo}$
      $S_{mejorC} = S_{nuevo}$
  $Aptitud_{mejorC} = eval(S_{mejorC}, D, M_2)$ *// M_2 es un wrapper*
  if $Aptitud_{mejorC} > Aptitud_{mejor}$ **then**
    $Aptitud_{mejor} = Aptitud_{mejorC}$
    $S_{mejor} = S_{mejorC}$
until *se cumpla la condición de parada*
Output : S_{mejor}

2.4.2. Algoritmos de Búsqueda

La búsqueda de subconjuntos de características sobre datos con un gran número de atributos exige que el algoritmo pueda producir resultados dentro de tiempos razonables. La generación de subconjuntos involucra una dirección y una estrategia de búsqueda. Para N atributos existen 2^N subconjuntos, por lo

que se requiere de una buena estrategia de búsqueda.

Para encontrar una estrategia de búsqueda se debe especificar el punto inicial, lo que afecta la dirección de la búsqueda. Existen distintas estrategias para determinar el subconjunto de características adecuado. A continuación se describen brevemente algunas de ellas:

Técnicas Secuenciales

Los algoritmos de búsqueda secuenciales (o deterministas) mas comunes son la búsqueda hacia delante (*forward selection*), la eliminación hacia atrás (*backward elimination*) y la selección paso a paso (*stepwise selection*). Estos métodos se detallan a continuación:

- *Búsqueda hacia adelante (Forward Selection):* Se comienza con un conjunto vacío al que se le van agregando atributos hasta que el criterio de selección haya alcanzado un mínimo o se hayan añadido todas las características. El proceso comienza considerando individualmente cada atributo y seleccionando el que mejor comportamiento obtiene cuando se emplea solo como entrada del algoritmo. El procedimiento se repite considerando individualmente el resto de las características. En cada paso se elige aquella cuya inclusión en el subconjunto disminuya en mayor medida el error global del sistema. Se finaliza cuando la inclusión de nuevos atributos no produzca una reducción de dicho error, o se hayan agregado todos los disponibles.

  El principal inconveniente de hacer un recorrido siguiendo esta dirección, es la posibilidad de no detectar interacciones básicas entre atributos que sean muy interesantes para la clasificación. Es decir, puede ocurrir que atributos que por separado son irrelevantes, el estar juntos en un determinado subconjunto les haga ser muy importantes. Se puede modificar para añadir los dos mejores atributos, o los tres, etc., pero relentizaría el tiempo de computación. [Ruiz Sanchez, 2006].

- *Eliminación hacia atrás (Backward Elimination):* Funciona de forma inversa al forward selection. Inicialmente se consideran todas las características y, paso a paso, se van eliminando aquellas cuya exclusión degrada en menor medida el resultado del algoritmo.

- *Selección paso a paso (Stepwise Selection):* Consiste en encadenar pasos de los dos métodos anteriores. Se comienza con un conjunto vacío de características, agregando en cada paso una nueva característica significativa. La diferencia es que, tras la inclusión de un nuevo atributo, se comprueba si alguno de los ya presentes puede ser eliminado sin afectar el rendimiento global, para lo que se emplea la eliminación hacia atrás. El proceso termina cuando ninguna característica aún no seleccionada tiene la relevancia suficiente para se incluida en el subconjunto.

Algoritmos genéticos

Los algoritmos genéticos son técnicas de búsqueda adaptativas basadas en los principios de la selección natural en biología [Goldberg, 1989]. Emplean soluciones que compiten entre sí para converger en el

Figura 2.5: Ejemplo de representación binaria de características

tiempo a una solución óptima.

Una solución de este estilo para selección de características consiste en representar a cada "individuo", que corresponde a un posible subconjunto de características, como una tira binaria de largo N, donde N corresponde al número de características existentes para la descripción del problema (Ver Figura 2.5), la existencia de un 1 en la posición i indica que la característica i de la muestra debe ser considerada en el subconjunto seleccionado. El algoritmo es un proceso iterativo donde cada generación sucesiva se produce aplicando operadores genéticos como cruce (*crossover*) y mutación a los miembros de la generación actual. La mutación cambia alguno de los valores (agregando o borrando características, cambiando 0 por 1 en una posición i o viceversa) aleatoriamente en un subconjunto. El cruce combina diferentes atributos de un par de subconjuntos en un nuevo subconjunto. Los subconjuntos resultantes se van seleccionando según una estrategia de evaluación, los mejores subconjuntos tienen más probabilidad de ser seleccionados y de esa forma evolucionan en el tiempo hacia el conjunto resultado [Pei et al., 1997] [Shafti and Pérez, 2004]. El algoritmo 3 resume este proceso.

Algoritmo 3: Pseudo-código de un algoritmo genético básico

Pob ← Crear la población inicial
Evaluar los individuos de *Pob*
while *la condición de terminación no se satisfaga* **do**
  Seleccionar un conjunto de individuos de *Pob* para reproducir
  Generar nuevos individuos a partir de la recombinación y mutación los seleccionados
  Evaluar los nuevos individuos
  Seleccionar los individuos de *Pob* que van a ser reemplazados
  Reemplazar los individuos por los recién generados
Output : el mejor individuo de la población

Selección de características basada en correlación

La técnica de selección por correlación parte de la hipótesis de que un buen conjunto de atributos contiene características que están altamente correlacionadas con la clase y no correlacionadas entre sí.

Una característica V_i se dice que es relevante si y solo sí existe algún v_i y c para los cuales

$$p(V_i = v_i) > 0 \text{ tal que } p(C = c|V_i = v_i) \neq p(C = c)$$

Una característica se dice que es redundante si una o más de las otras características están altamente

correlacionadas con ella.

El problema reside en desarrollar formas de medir la correlación característica-clase y la intercorrelación característica-característica. Las medidas de correlación se calculan con la fórmula del coeficiente de correlación de Pearson.

CFS (*Correlation-based Features Selection*) es un algoritmo simple que genera subconjuntos de características de acuerdo a una función de evaluación basada en correlación [Hall, 1999]. La función de evaluación tiende a generar subconjuntos de atributos altamente correlacionados con la clase y no correlacionados entre sí. La eliminación de las características irrelevantes se deberá a su baja correlación con la clase y la eliminación de las redundantes debido a su alta correlación con otras características. La aceptación de una característica dependerá de su capacidad de predecir la clase en áreas del espacio de instancias que no son predichas por otra característica.

La ecuación (2.1) define la correlación entre todas las características y la clase r_{zc} .

$$r_{zc} = \frac{k * r_{zi}}{\sqrt{k + k(k-1)r_{ii}}} \tag{2.1}$$

donde k es el número de características, r_{zi} es el promedio de las correlaciones entre las características y la clase y r_{ii} es el promedio de las intercorrelaciones entre las características.

La ecuación sirve como estrategia de evaluación en los algoritmos de selección, el numerador indica cuan predictiva de la clase es un conjunto de características y el denominador cuanta redundancia existe entre las características. Las diferentes implementaciones de CFS utilizan alguna de las técnicas de búsqueda (selection forward, backward elimination, etc.). El subconjunto con mayor valor de evaluación encontrado durante la búsqueda será el resultado del algoritmo.

Actualmente se han desarrollado nuevas técnicas de correlación basadas en medidas de correlación [Hsu and Hsieh, 2010].

2.4.3. Salida del Algoritmo

Otra forma de clasificar los algoritmos de selección considera el tipo de salida que el algoritmo produce. De esta forma se pueden dividir en los que producen un subconjunto de atributos y los que generan un ranking.

Los métodos de ranking de atributos, también denominados *feature weighting* asignan pesos a los atributos individualmente y los ordenan basándose en su relevancia con respecto al atributo clase, mientras que los algoritmos de selección de subconjunto de atributos evalúan la bondad de cada uno de los subconjuntos candidatos. (Ocasionalmente, algunas estrategias en combinación con la evaluación de subconjuntos pueden proporcionar una lista ordenada de atributos).

El Algoritmo 4 es un ejemplo de elaboración de un ranking. La idea es evaluar cada atributo individualmente según un criterio e insertarlo en orden en una lista según el valor obtenido. Existen

Algoritmo 4: Pseudo-código de un algoritmo para elaborar una lista de atributos

$E \leftarrow$ conj.de datos de entrenamiento
$U \leftarrow$ medida de evaluación
$L \leftarrow \{\}$ lista inicial de atributos vacía
for *cada atributo* X_i *en* L **do**
    Se evalúa el atributo segun U
    $v_i = evaluar(X_i, U)$
    $Posicionar(X_i, L)$ *con respecto a* v_i
Output : $L \leftarrow$ lista de atributos con el más relevante en primer lugar

muchas variantes de este algoritmo, teniendo en común la salida de una lista de atributos ordenada.

El Algoritmo 1 ya presentado es un ejemplo de como obtener un subconjunto de atributos. En general, devuelve un subconjunto mínimo de atributos, entre los cuales no se hace diferencia en cuanto a la relevancia. Se genera un subconjunto a evaluar. Este proceso se repite hasta que se cumpla uno de los criterios de parada (un número de atributos determinado, un número de iteraciones o que no se mejore al subconjunto anterior).

2.5. SOAP. Selección de atributos por proyecciones

En esta sección se describe con detalle la manera de seleccionar atributos utilizando el método SOAP (Selection of attributes by projections) definido por Sánchez en [Ruiz Sanchez, 2006]. El método presentado será utilizado para identificar las características más relevantes al momento de resolver el problema planteado en esta tesis.

Este método incorpora un criterio alternativo para medir la importancia de un atributo dentro de un marco de aprendizaje supervisado, utiliza el número de cambios de etiqueta o NLC (*Number of Label Changes*) como medida para la evaluación individual de atributos, calculada analizando las proyecciones de los elementos del conjunto de datos sobre cada dimensión o atributo. En las secciones siguientes se explica el método detalladamente.

El método SOAP evalúa los atributos de una base de datos analizando las proyecciones de los elementos del conjunto de datos sobre cada atributo individual, de esta manera se pueden ordenar los atributos por orden de importancia en la determinación de la clase.

Para entender la idea del algoritmo, se reducirá el problema a seleccionar atributos para una tabla con solo dos columnas de atributos y una columna de clase. La simplificación permite graficar en un espacio bidimensional los diferentes valores que toma cada atributo junto con la clase a la que pertenece cada ejemplo.

Para el ejemplo de la tabla 2.2, el gráfico obtenido es el de la figura 2.6. Se puede observar el atributo X_1 en el eje de las abscisas, el atributo X_2 en el de las ordenadas y el valor de la clase representado

ejemplo	sexo	anio_nacim	etiqueta
1	2	1973	Abandono
2	1	1972	Abandono
3	2	1989	Abandono
4	1	1992	Abandono
5	1	1963	Abandono
6	2	1986	Cursa
7	1	1988	Cursa
8	1	1992	Cursa
9	2	1981	Abandono
10	2	1993	Cursa
11	2	1972	Abandono
12	1	1973	Abandono
13	1	1987	Abandono
14	1	1986	Abandono
15	2	1991	Abandono
16	2	1991	Cursa
17	1	1990	Abandono
18	2	1975	Cursa
19	1	1975	Abandono
20	1	1975	Abandono

Tabla 2.2: Conjunto de datos de ejemplo para proyecciones

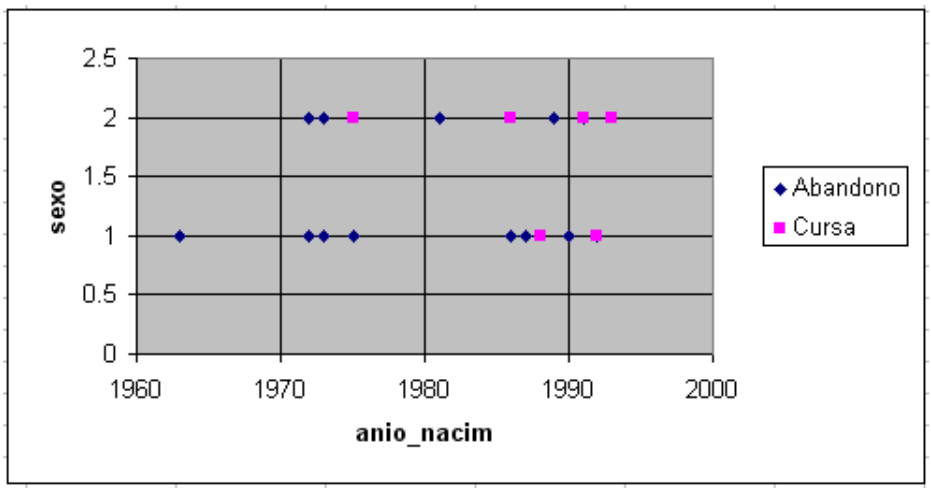

Figura 2.6: Proyección de atributos `sexo` y `anio_nacim`

por diferentes colores. Recorriendo el eje de abscisas desde el origen hasta el mayor valor del atributo, se puede distinguir aproximadamente una serie de intervalos donde prevalece una clase. Esto no ocurre al recorrer el eje de las ordenadas. Se puede intuir que será más fácil clasificar por los atributos que presenten un menor número de cambios de etiquetas (NLC, *Number of Label Changes*). Ordenando los atributos ascendentemente según su NLC, se logra un ranking con los mejores atributos para clasificar los ejemplos.

Otro ejemplo de un gráfico de proyecciones, esta vez con un número mayor de datos, se puede ver en la figura 2.7. Aquí se graficaron 600 ejemplos tomados del caso de estudio en relación a los atributos `anio_nacim` y `prom_notas_finales`.

Para describir el algoritmo que se va a implementar es necesario comenzar con algunas definiciones:

Definición 1. Un dominio, representado con $Dom()$, es un conjunto de valores del mismo tipo.

Definición 2. Universo de discurso es el entorno donde se define un determinado problema. Se representa como el producto cartesiano de un conjunto finito de dominios.

Definición 3. Un atributo o característica toma valores de un determinado dominio para describir una medida del universo de discurso. Se denota:

X_i: i-ésimo atributo
x_i: valor del i-ésimo atributo
$Dom(X_i)$: el dominio del i-ésimo atributo

Vector de atributos $\overrightarrow{x} = (x_1, .., x_n)$: conjunto de valores correspondiente a cada uno de los atributos

$\mathcal{X} = Dom(X_1) \times .. \times Dom(X_n)$: espacio formado por el conjunto de los atributos, siendo n el total de atributos.

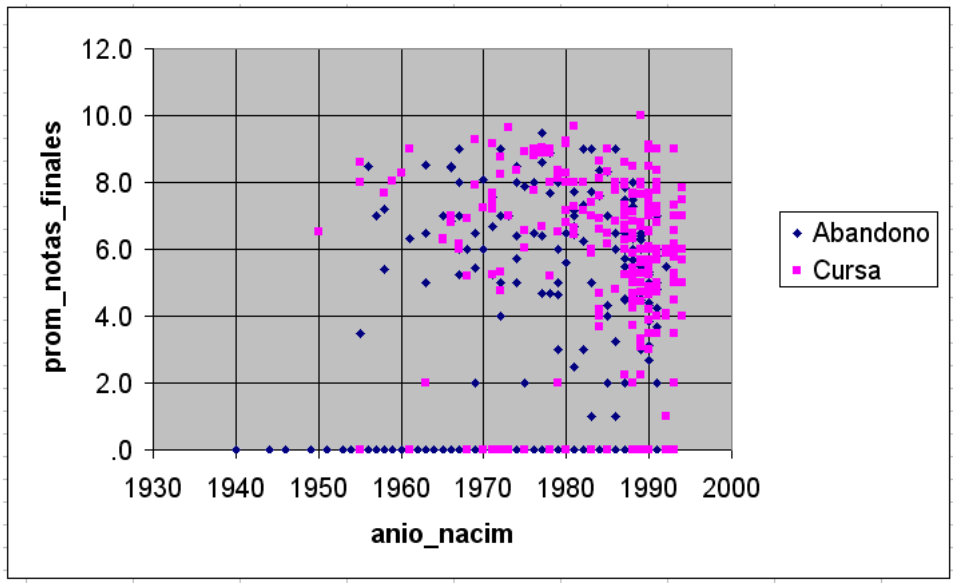

Figura 2.7: Proyección de atributos `anio_nacim` y `prom_notas_finales` para un conjunto de 600 ejemplos tomados del caso de estudio

Definición 4. La clase es un atributo especial de salida que indica la pertenencia a un grupo determinado de casos. Se denomina etiquetas de clase al conjunto o dominio de valores que la clase puede tomar. Se representa con Y y su dominio $Dom(Y)$, teniendo k valores posibles $y_1, .., y_k$.

Definición 5. Un ejemplo o instancia es una tupla del universo de discurso representada por un conjunto de valores de atributos, cada uno de un dominio respectivamente, y una etiqueta de clase que lo clasifica. El ejemplo se representa e.

Definición 6. Un conjunto de datos etiquetados es un subconjunto finito de ejemplos E de m instancias $(\overrightarrow{x_j}, y_j)$, donde $j = 1..m$, cada una compuesta por n valores de entrada $x_{j,i}$ con $(i = 1..n)$ y uno de salida y_j.

Definición 7. Se definen las funciones π_i y lab :

$$\pi_i : E \rightarrow Dom(X_i) \quad \pi_i(e_j) = x_{j,i} \quad \forall j : 1..m \tag{2.2}$$

$$lab : E \rightarrow Dom(Y) \quad lab(e_j) = y_l \quad \forall l : 1..k \tag{2.3}$$

π_i representa el conjunto de los valores que toma el atributo X_i para todos los ejemplos de un conjunto de ejemplos.

Se toma un subconjunto limitado de ejemplos del caso de estudio para ejemplificar las definiciones.

El subconjunto está formado por 20 ejemplos, para los atributos `sexo` y `anio_nacim`, y la etiqueta de Cursa/Abandono (ver tabla 2.2).

Para el atributo `sexo`, dado que el subconjunto de ejemplos actual contiene 20 instancias, π_{sexo} tomará 20 valores 1 o 2 según cada ejemplo de la base de datos:

$$\pi_{sexo}(e_j) = \{1, 2, 1, 2, 2, 1, 2, 2, 1, 1, 1, 2, 2, 2, 1, 1, 2, 1, 2, 2\} \quad \forall j : 1..\ 20$$

porque :

$$\begin{gathered}\pi_{sexo}(e1) = 1\\ \pi_{sexo}(e2) = 2\\ \pi_{sexo}(e3) = 1\\ \pi_{sexo}(e4) = 2\\ \pi_{sexo}(e5) = 2\\ ....\end{gathered}$$

Para el atributo `anio_nacim`:

$$\begin{gathered}\pi_{anio\_nacim}(e_j) = \{1973, 1972, 1989, 1992, 1963, 1986, 1988, 1992, 1981, 1993, 1972, 1973, 1987, 1986,\\ 1991, 1991, 1990, 1975, 1975, 1975\}\\ \forall j : 1..\ 20\end{gathered}$$

De la misma forma, si se considera la etiqueta del conjunto de datos:

$$\begin{gathered}lab(e_j) = \{Abandono, Abandono, Abandono, Abandono, Abandono, Cursa, Cursa,\\ Cursa, Abandono, Cursa, Abandono, Abandono, Abandono, Abandono,\\ Abandono, Cursa, Abandono, Cursa, Abandono, Abandono\}\\ \forall j : 1..\ 20\end{gathered}$$

Definición 8. Para cada atributo X_i se definen dos relaciones de orden en $Dom(X_i)$ que se Denotan <= y < que son las relaciones de orden establecidas en el dominio del atributo (por ejemplo en el dominio del atributo `sexo`, $1 < 2$, en el dominio del atributo `anio_nacim`, $1986 < 1994$).

En el conjunto E se define para cada atributo X_i dos relaciones de orden $<=_i$ y $<_i$ de forma que dados los ejemplos e y f se dice que:

$$e <=_i f \quad si \quad \pi_i(e) <= \pi_i(f) \tag{2.4}$$

$$e <_i f \quad si \quad \pi_i(e) < \pi_i(f) \tag{2.5}$$

Y una relación de equivalencia donde

$$e =_i f \quad si \quad \pi_i(e) = \pi_i(f) \tag{2.6}$$

Por ejemplo, para `anio_nacim`:

$$\pi_{anio\_nacim}(e_1) = 1973$$
$$\pi_{anio\_nacim}(e_4) = 1992$$
$$\pi_{anio\_nacim}(e_8) = 1992$$

entonces :

$$e_1 <=_{anio\_nacim} e_4$$
$$e_1 <_{anio\_nacim} e_4$$
$$e_4 =_{anio\_nacim} e_8$$

porque :

$$\pi_{anio\_nacim}(e_3) <= \pi_{anio\_nacim}(e_4)$$
$$\pi_{anio\_nacim}(e_3) < \pi_{anio\_nacim}(e_4)$$
$$\pi_{anio\_nacim}(e_2) = \pi_{anio\_nacim}(e_4)$$

Definición 9. Dado el conjunto E se denomina $\sigma_i(E)$ a la secuencia ordenada por el atributo X_i de los ejemplos de E. Es decir

$$\sigma_i(E) =< e_1, ..., e_n > \quad / \quad e_j \, \varepsilon \, E \quad \forall j : 1..n \quad \wedge j : 1..(n-1) \quad e_j <=_i e_{j+1} \tag{2.7}$$

En el conjunto de datos de ejemplo se pueden reconocer las siguientes secuencias ordenadas:

$$\sigma_{sexo}(E) =< e_2, e_4, e_5, e_7, e_8, e_{12}, e_{13}, e_{14}, e_{17}, e_{19}, e_{20}, e_1, e_3, e_6, e_9, e_{10}, e_{11}, e_{15}, e_{16}, e_{18} >$$
$$\sigma_{anio\_nacim}(E) =< e_5, e_2, e_{11}, e_{12}, e_1, e_{19}, e_{20}, e_{18}, e_9, e_{14}, e_6, e_{13}, e_7, e_3, e_{17}, e_{15}, e_{16}, e_4, e_8, e_{10} >$$

Como puede observarse, se ordenan los ejemplos por el valor del atributo.

Definición 10. Sea $\sigma_i(E) =< e_1, ..., e_n >$ una secuencia ordenada de E por el atributo X_i, entonces se dice que existe una subsecuencia ordenada múltiple (SOM) y se denomina S si

$$\exists j,\ k > 0\ /\ e_{j-1} <_i e_j =_i ... =_i e_{j+k} <_i e_{j+k+1} \tag{2.8}$$

siendo la SOM $S =< e_j, ..., e_{j+k} >$ (si $j = 1$ la primera condición se obvia).

Se define etiqueta mayoritaria de una SOM, $ml(S)$, a la moda del conjunto:

$$\{lab(e_l)\} \quad l = j...j + k$$

En el ejemplo podemos encontrar varias SOM, algunas de ellas son:

$$S =< e_5, e_2, e_{12}, e_{19}, e_{20}, e_{14}, e_{13}, e_7, e_{17}, e_4, e_8 >$$

es una SOM por `sexo` dado que todos estos ejemplos tienen el mismo valor (2) para dicho atributo y $ml(S) = Abandono$, al ser esa la etiqueta de 9 de los 10 ejemplos incluidos.

$$S =< e_2, e_{11} >$$

es una SOM por `anio_nacim`, ya que los ejemplos e_2 y e_{11} tienen para `anio_nacim` el mismo valor (1972) y $ml(S) = Abandono$ dado que todos los ejemplos pertenecen a esa clase.

Definición 11. Sea $\sigma_i(E) =< e_1, ..., e_n >$ una secuencia ordenada de E por el atributo X_i, entonces se dice que existe una subsecuencia ordenada simple (SOS) y se denomina S si

$$\exists j;\ k > 0 \quad e_{j-1} =_i e_j <_i e_{j+1} <_i ... <_i e_{j+k-1} <_i e_{j+k} =_i e_{j+k+1} \tag{2.9}$$

siendo la SOS $S =< e_{j+1}, ..., e_{j+k-1} >$ (si $j = 1$ la primera condición se obvia).

El ejemplo e_{j+1} es denominado primer ejemplo, $fe(S)$, y e_{j+k-1} último ejemplo $le(S)$.

En el ejemplo es una SOS:

$$S =< e_{13}, e_7, e_3, e_{17} >$$

en `anio_nacim`.

Es posible afirmar que dada una secuencia ordenada de ejemplos $\sigma_i(E)$ por algún atributo X_i, siempre se podrá dividir en subsecuencias donde o bien hay una SOS, o una SOM o una sucesión de secuencias de unas y otras, teniendo en cuenta que tras una SOS siempre vendrá una SOM, pero detrás de una SOM puede aparecer una SOS o SOM.

En el ejemplo:

$$\sigma_{sexo}(E) =$$
$$< e_5, e_2, e_{12}, e_{19}, e_{20}, e_{14}, e_{13}, e_7, e_{17}, e_4, e_8 >, \quad \rightarrow SOM$$
$$< e_{11}, e_1, e_{18}, e_9, e_6, e_3, e_{15}, e_{16}, e_{10} > \quad \rightarrow SOM$$

$$\sigma_{anio\_nacim}(E) =$$
$$< e_5 >, \quad \rightarrow SOS$$
$$< e_2, e_{11} >, \quad \rightarrow SOM$$
$$< e_{12}, e_1 >, \quad \rightarrow SOM$$
$$< e_{19}, e_{20}, e_{18} >, \quad \rightarrow SOM$$
$$< e_9 >, \quad \rightarrow SOS$$
$$< e_{14}, e_6 >, \quad \rightarrow SOM$$
$$< e_{13}, e_7, e_3, e_{17} >, \quad \rightarrow SOS$$
$$< e_{15}, e_{16} >, \quad \rightarrow SOM$$
$$< e_4, e_8 >, \quad \rightarrow SOM$$
$$< e_{10} > \quad \rightarrow SOS$$

Definición 12. Dada una secuencia ordenada S de k ejemplos $S =< e_1, e_2, ..., e_k >$ se define la función:

$$ch : S - \{e_k\} \rightarrow 0, 1 \tag{2.10}$$

$$ch(e_j) = 1 \text{ si } lab(e_j) <> lab(e_{j+1}) \quad \wedge$$
$$ch(e_j) = 0 \text{ si } lab(e_j) = lab(e_{j+1}) \quad j : 1..(k-1)$$

En el ejemplo, para `anio_nacim` para la SOS $S =< e_{13}, e_7, e_3, e_{17} >$

$$ch(e_{13}) = 1 \quad lab(e_{13}) <> lab(e_7)$$
$$ch(e_7) = 1 \quad lab(e_7) <> lab(e_3)$$
$$ch(e_3) = 0 \quad lab(e_3) = lab(e_{17})$$

Esta función determina si hay o no cambio de etiqueta entre los ejemplos sucesivos.

Definición 13. Dada una secuencia ordenada S de k ejemplos $S =< e_1, e_2, ..., e_k >$ se define la función *ChangeCounter*:

$$ChangeCounter(S) = \sum_{j=1}^{k-1} ch(e_j) \tag{2.11}$$

Esta función cuenta la cantidad de cambios de etiqueta detectados en la secuencia.

Para la secuencia del ejemplo anterior $S =< e_{13}, e_7, e_3, e_{17} >, \quad ChangeCounter = 2$.

Definición 14. Sea S una SOS de $\sigma_i(E)$ para algún $i = 1..n$ de un conjunto E. Entonces S o es la última subsecuencia o está seguida de una SOM que se denota S'. Entonces se define:

$$nch(S) = ChangeCounter(S) + ChangeAdd(S; S') \tag{2.12}$$

donde $ChangeAdd(S; S')$ vale 0 ó 1 en función de S', según lo siguiente:

- Si S' no existe (si S es la última subsecuencia) $\rightarrow ChangeAdd(S; S') = 0$
- Si $lab(le(S)) = ml(S') \rightarrow ChangeAdd(S; S') = 0$
- Si $lab(le(S)) <> ml(S') \rightarrow ChangeAdd(S; S') = 1$

Es decir, cuando se tiene una SOS seguida de una SOM, el cálculo del número de cambios tiene en cuenta si al pasar de una subsecuencia a otra se ha producido algún cambio de etiqueta. Para ello se compara la etiqueta del último ejemplo de la SOS con la etiqueta mayoritaria de la SOM.

En el ejemplo, para el atributo `anio_nacim`, en la SOS $S =< e_5 >$, $ChangeCounter(S) = 0$ siendo $lab(le(S)) = Abandono$ y como le sigue una SOM $S' =< e_2, e_{11} >$ que tiene como $lab(ml(S')) = Abandono$, $ChangeAdd(S; S') = 0$, totalizando, $nch(S) = 0$.

Definición 15. Sea S una SOM de $\sigma_i(E)$ para algún $i = 1..n$ de un conjunto E. Entonces S o es la última subsecuencia o está seguida por otra SOM o por una SOS. Si S es una SOM, el valor de la función *ChangeCounter* viene influido por el orden de los ejemplos con igual valor en el parámetro por el que se ha ordenado S. Esto hace que el valor de *ChangeCounter* no sea unívoco y por ello se define *nch* como el peor caso, es decir, el valor del mayor número de cambios de etiqueta para todas las ordenaciones posibles de ejemplos de S. Entonces, dada $S =< e_1, ..., e_k >$ una SOM , se define $\tau(S)$ el conjunto de todas las permutaciones posibles de los ejemplos de S , y se define

$$nch(S) = max_{t \epsilon \tau(S)} ChangeCounter(t) + n(S; S') \tag{2.13}$$

donde $n(S; S')$ vale 0 ó 1 en función de S', según lo siguiente:

- Si S' es SOM:
  - Si S' no existe $\rightarrow n(S; S') = 0$

- Si $ml(S) <> ml(S') \rightarrow n(S;S') = 1$
- Si $ml(S) = ml(S') \rightarrow n(S;S') = 0$

- Si S' es SOS:
  - Si S' no existe $\rightarrow n(S;S') = 0$
  - Si $ml(S) <> lab(fe(S')) \rightarrow n(S;S') = 1$
  - Si $ml(S) = lab(fe(S')) \rightarrow n(S;S') = 0$

En el ejemplo, para `anio_nacim`, se tiene la SOM $S =< e_{19}, e_{20}, e_{18} >$ cuyas etiquetas son estas: $< Abandono, Abandono, Cursa >$

Posibilidades ChangeCounter:

$$< Abandono, Abandono, Cursa > \rightarrow 1$$
$$< Abandono, Cursa, Abandono > \rightarrow 2$$
$$< Cursa, Abandono, Abandono > \rightarrow 1$$

Con $ml(S) = Abandono$

Y está seguida de una SOS $S =< e_9 >$, donde $lab(fe(S')) = Abandono$, por lo tanto $nch(S) = 2$.

2.5.1. NLC: Número de cambios de etiqueta

Definición 16. Dado un conjunto E de ejemplos, $\sigma_i(E)$ la secuencia ordenada por el atributo X_i y $S_1 ... S_r$ la división de $\sigma_i(E)$ en SOS y SOM correspondiente, entonces se define la función

$$NLC(i) = \sum_{j=1}^{r} nch(S_j) \tag{2.14}$$

Para el ejemplo:

$$\sigma_{sexo}(E) =< e_5, e_2, e_{12}, e_{19}, e_{20}, e_{14}, e_{13}, e_7, e_{17}, e_4, e_8 >, \rightarrow SOM \rightarrow nch = 4$$
$$< e_{11}, e_1, e_{18}, e_9, e_6, e_3, e_{15}, e_{16}, e_{10} > \rightarrow SOM \rightarrow nch = 8$$

Por lo tanto $NLC(sexo) = 12$.

$$\begin{aligned}\sigma_{anio\_nacim}(E) = \\ &< e_5 >, \rightarrow SOS \rightarrow nch = 0 \\ &< e_2, e_{11} >, \rightarrow SOM \rightarrow nch = 0 \\ &< e_{12}, e_1 >, \rightarrow SOM \rightarrow nch = 0 \\ &< e_{19}, e_{20}, e_{18} >, \rightarrow SOM \rightarrow nch = 2 \\ &< e_9 >, \rightarrow SOS \rightarrow nch = 0 \\ &< e_{14}, e_6 >, \rightarrow SOM \rightarrow nch = 2 \\ &< e_{13}, e_7, e_3, e_{17} >, \rightarrow SOS \rightarrow nch = 2 \\ &< e_{15}, e_{16} >, \rightarrow SOM \rightarrow nch = 2 \\ &< e_4, e_8 >, \rightarrow SOM \rightarrow nch = 2 \\ &< e_{10} > \rightarrow SOS \rightarrow nch = 0\end{aligned}$$

Por lo tanto $NLC(anio\_nacim) = 10$.

Existe un caso particular no analizado en [Ruiz Sanchez, 2006] y en este punto se toma una decisión al respecto. Cuando los datos carecen de moda, porque existe más de un valor con la máxima frecuencia, se debe decidir que acción tomar respecto al valor asignado a $ml(S)$. En primera instancia la existencia de dos valores del atributo con exactamente la misma frecuencia podría inducir a elegir al azar entre ellos. Pero siguiendo el criterio presentado en la Definición 15 (2.13), se decide elegir el valor que produzca el peor caso respecto a la secuencia siguiente. Es decir, se elige de entre los valores posibles (aquellos que coinciden en la máxima frecuencia) uno que suponga un cambio de etiqueta respecto a la secuencia siguiente S'.

A partir de estas definiciones el trabajo [Ruiz Sanchez, 2006] presenta una medida para la selección de atributos basada en un único valor: NLC (*Number of Label Changes*) que relaciona cada atributo con la etiqueta que sirve de clasificación. El NLC se calcula proyectando los ejemplos sobre el eje correspondiente a ese atributo (es decir, ordenando los ejemplos por el atributo), para a continuación recorrer el eje desde el origen hasta el mayor valor del atributo contabilizando el número de cambios de etiqueta que se producen. El algoritmo se adapta a cualquier número de clases (ver algoritmos 5 y 6).

Algoritmo 5: Pseudo-código del algoritmo SOAP para la generación de un ranking de atributos por NLC

```
E ← conj.de datos (m ejemplos, n atributos)
R ← {} ranking inicial de atributos vacío
for i = 1 a n do
    Ordenar E por el atributo X_i
    NLC_i ← ContarCambios(E; X_i)
Output : R ← ranking de atributos según su NLC
```

Algoritmo 6: Pseudo-código del algoritmo SOAP que cuenta los cambios de etiqueta para un atributo

```
E ← conj.de datos de datos (m ejemplos, n atributos, X_i atributo a procesar
Cambios ← 0
for j = 1 a m do
    if x_{j,i} ∈ SOM then
        Cambios ← Cambios + VerCambiosParaMismoValor()
    else
        if labe_j <> labe_{j+1} then
            Cambios ← Cambios + 1
Output : Cambios ← Cambios de etiqueta
```

La función *VerCambiosParaMismoValor*() resuelve los cambios de etiqueta en el caso de las SOM sin realizar todas las combinaciones posibles, para realizarlo se basa en las siguientes situaciones posibles:

- SOS → cuenta un cambio por cada cambio de etiqueta.
- SOM → número máximo de cambios posibles con *VerCambiosParaMismoValor*:
  - Todos los ejemplos con la misma etiqueta → ningún cambio de clase.
  - Etiquetas distintas:
    - Mayoría de una de ellas → (número elementos SOM - nro. elementos de la clase mayoritaria)* 2
    - No existe mayoría → número elementos SOM - 1

Cuando se detecta una SOM, en la función *VerCambiosParaMismoValor*() se acumulan los ejemplos según su etiqueta, y el algoritmo comprueba si existe una clase cuya frecuencia sea superior a la suma del resto de frecuencias (que supere la mitad del número de ejemplos), si es así, el número de cambios de etiquetas para el peor de los casos será la suma del resto de las frecuencias multiplicado por dos, ya que se podrá intercalar cada uno de los ejemplos de las clases no mayoritarias entre los de la mayoritaria. En el caso de que no exista ninguna frecuencia mayor que la suma del resto de frecuencias, el número máximo de cambios posibles será equivalente al número de ejemplos menos uno.

Si se contaran los cambios de etiqueta uno a uno, en varias ejecuciones del algoritmo sobre un mismo conjunto de ejemplos, los valores iguales de un mismo atributo podrían quedar ordenados de forma diferente en cada corrida. Eso produciría valores de NLC diferentes y por lo tanto un atributo podría quedar ubicado en distintas posiciones del ranking para diferentes corridas sobre los mismos datos. La heurística aplicada en el algoritmo consiste en encontrar el peor de los casos, es decir obtener el máximo número de cambios de etiqueta posible dentro del intervalo que contiene valores iguales del atributo.

El motivo de contar el número máximo de cambios de etiquetas es doble, por un lado penalizar al atributo que presente este tipo de situaciones inconsistentes, y por otro, evitar ambigüedades que se producirían según la posición en la que aparecieran los elementos tras aplicar el algoritmo de ordenación.

El valor de NLC obtenido para cada atributo permite ordenar los mismos en un ranking, de manera que el primer atributo del ranking sea aquél que menor valor de NLC muestre. El resultado del algoritmo SOAP para selección de atributos genera un ranking de atributos ordenados por su relevancia para la obtención de la clase.

Capítulo 3

Técnicas de extracción de conocimiento

3.1. Extracción de Patrones

Una vez que los datos han sido preprocesados utilizando los métodos descriptos en el capítulo anterior, la información es considerada una vista minable y está preparada para ser sometida a la técnica que permita establecer el modelo buscado.

La Minería de Datos presenta un amplio espectro de técnicas. Los conjuntos de datos a los que se aplica $\mathcal{DM}$ pueden exhibir estructuras diferentes y en tal sentido las técnicas seleccionadas para tratarlos pueden ser elegidas de una amplia variedad. La claridad de los resultados va a depender en gran medida de la técnica elegida, es por eso que este análisis previo resulta relevante.

La simple aplicación de una técnica de $\mathcal{DM}$ a una vista minable y el conocimiento previo del problema, no garantizan patrones expresivos, novedosos y útiles. Los algoritmos muchas veces ofrecen malos resultados debido a causas ajenas a su efectividad, ya sea porque no existe patrón en los datos o porque no se está usando la herramienta adecuada o porque el patrón es realmente difícil de encontrar.

En este capítulo se presentan brevemente las tareas y métodos de $\mathcal{DM}$ con el objetivo de elegir un subconjunto de ellas en función de la utilidad que puedan prestar al tratamiento de los datos con que se cuenta. Una vez determinado ese subconjunto, en las secciones subsiguientes se exponen en detalle las técnicas seleccionadas, para proporcionar el marco teórico que exige su aplicación al análisis de la deserción de alumnos universitarios de la UNRN.

3.2. Tareas

Como se dijo con anterioridad en la presentación de las fases del $\mathcal{KDD}$, existen dos tipos de tareas, las predictivas y las descriptivas. A continuación se mencionan las más importantes.

3.2.1. Tareas Predictivas

Se consideran predictivas aquellas tareas que requieren de la obtención de un modelo capaz de dar una respuesta, en una etapa posterior, ante la presencia de información nueva. Según si la respuesta esperada es discreta o continua, se considera que la tarea predictiva es una clasificación o una regresión, respectivamente. Un ejemplo de tarea de clasificación es obtener un modelo que dado un nuevo producto pueda clasificarlo como "básico", "estandar" o "de lujo". Un ejemplo de tarea de regresión es obtener un modelo que dado un paciente nuevo determine la probabilidad de que tenga cierta enfermedad.

La *clasificación* es una de las tareas más utilizadas. En ella, cada ejemplo o registro de la vista minable, pertenece a una clase la cual se indica mediante el valor de un atributo nominal que se denomina "etiqueta". Esta característica permite obtener el modelo a través de una estrategia supervisada que, operando sobre el resto de los atributos de cada instancia, buscará maximizar la tasa de acierto sobre el conjunto de ejemplos de entrada. Al finalizar el proceso, el clasificador obtenido será capaz de determinar la clase para cada nuevo ejemplo sin etiquetar. Entre las tareas de clasificación hay distintas variantes:

- *Clasificación suave:* Según la técnica utilizada para la construcción del modelo puede incorporarse a la clasificación una función que determine el grado de certeza de la predicción. De esta forma, podría permitirse que un clasificador etiquetara un mismo ejemplo con más de una clase asignando a cada una de ellas un valor de certeza diferente y sería la persona encargada de tomar las decisiones quien debería decidir entre las opciones presentadas.
- *Estimación de la probabilidad de clasificación:* Es una generalización de la clasificación suave que provee para cada valor de la clase la probabilidad de que un ejemplo sea de la clase. A diferencia del clasificador suave, en este caso las opciones serían excluyentes ya que se basan en la teoría de la probabilidad y la cantidad de ejemplos requeridos para su construcción debe ser grande.
- *Categorización:* A diferencia de la clasificación, se trata de aprender una correspondencia pudiendo asignar más de una categoría a cada ejemplo.

Las tareas de *regresión* también utilizan conjuntos de ejemplos etiquetados y tienen como objetivo aprender una función que represente la correspondencia existente entre los atributos considerados en cada ejemplo y la clase o etiqueta indicada. Su diferencia con respecto a la clasificación es que la salida es numérica mientras que en la clasificación es nominal.

3.2.2. Tareas Descriptivas

Este tipo de tareas buscan mostrar nuevas relaciones entre las variables y generalmente son utilizadas para mejorar el modelo. Su objetivo es describir los datos existentes. Entre las tareas descriptivas más frecuentes, pueden mencionarse las siguientes:

- *Agrupamiento (clustering):* El objetivo de esta tarea es obtener grupos o conjuntos entre los ejemplos, de manera que los elementos asignados al mismo grupo sean similares. A priori no

se sabe ni cómo son los grupos ni cuantos hay, eso se determina con el proceso de aprendizaje. Una utilidad del agrupamiento reside en que utilizando la función obtenida con nuevos ejemplos se puede determinar a qué grupo pertenece el nuevo elemento y con eso indicar su comportamiento. La tarea de agrupamiento también suele utilizarse con el objetivo de reducir un gran número de ejemplos a sólo algunos grupos que sirvan como resumen de los datos originales.

- *Correlaciones y factorizaciones:* Se centran en atributos numéricos. Su objetivo es detectar si dos atributos numéricos están correlacionados linealmente o relacionados de algún otro modo. Su utilidad es la detección de atributos redundantes o dependientes y analizar la relevancia de atributos para hacer una selección entre ellos.

- *Reglas de asociación:* Es un estudio similar al de correlaciones pero para atributos nominales. Dados dos ejemplos del conjunto de entrada una regla de asociación se define generalmente de la forma

$$SI\ (atrib_1 = valor_1)\ y\ (atrib_2 = valor_2)\ y\ (atrib_k = valor_k)\ ENTONCES$$
$$(atrib_r = valor_r)\ y\ (atrib_s = valor_s)\ y\ (atrib_z = valor_z)$$

  donde todos los atributos son nominales y las igualdades se definen utilizando algún valor de los posibles para cada atributo. Este tipo de reglas se utilizan en el conocido análisis de la cesta de mercado.

3.3. Métodos

Cada una de las tareas presentadas requiere métodos, técnicas o algoritmos para resolverlas.

Una tarea puede tener muchos métodos para resolverla y el mismo método (o al menos el mismo tipo de técnica) puede resolver un gran abanico de tareas, dado que la mayoría de las tareas son caras del aprendizaje inductivo [Hernández Orallo et al., 2004].

Algunos de los tipos de técnicas más utilizadas se reseñan en la lista siguiente:

- *Técnicas algebraicas y estadísticas:* También denominadas paramétricas, expresan modelos mediante fórmulas, funciones, distribuciones o valores estadísticos como medias, varianzas, etc. Obtienen un patrón a partir de un modelo predeterminado del cual se estiman los coeficientes o parámetros. Son ejemplos de estas técnicas la regresión lineal, regresión logarítmica y logística [Freedman, 2009].

  Los modelos en los que el comportamiento de una variable Y se puede expresar como una función de una variable X se pueden representar mediante $Y = f(X)$, si se considera que la relación f es una función lineal, que las variables explicativas pueden ser N en lugar de una única X y que las relaciones no son exactas, sino mas bien aproximaciones, por lo que se debe agregar un término de perturbación aleatoria u que refleje esos factores, la fórmula anterior puede escribirse

$$Y_i = B_0 + B_1 X_{i1} + \ldots + B_N X_{iN} + u_i$$

y el modelo se denomina regresión lineal.

Cuando la regresión lineal no logra determinar los coeficientes, o el fenómeno en estudio tiene un comportamiento que puede considerarse potencial o logarítmico, se utiliza la regresión logarítmica, que transforma la ecuación anterior aplicando logaritmo a ambos lados de la igualdad.

Un caso de regresión lineal generalizado es la regresión logística que permite modelizar una probabilidad. La variable de respuesta tiene dos o mas posibilidades, cada una con su respectiva probabilidad.

- *Técnicas bayesianas:* Utilizan el teorema de Bayes para estimar la probabilidad de pertenencia a una clase o grupo. Un ejemplo clásico es el clasificador bayesiano ingenuo o *naive* Bayes [Winkler, 1972].

  Una red bayesiana es un grafo acíclico dirigido en el que cada nodo representa un atributo y cada arco una dependencia probabilística que expresa la probabilidad condicional de cada atributo dados sus padres. El arco apunta a un atributo dependiente del que está en el origen del arco. La estructura de la red provee información sobre dependencias y también sobre las independencias de un atributo (o conjunto de ellos) de otro u otros. La construcción de una red bayesiana a partir de los datos consta de un proceso de aprendizaje estructural, donde se obtiene la estructura de la red, y un aprendizaje paramétrico en que se obtienen las probabilidades y condicionales de la estructura.

- *Técnicas basadas en conteos de frecuencias y tablas de contingencia:* Cuentan la frecuencia en que dos o más sucesos se dan conjuntamente, el algoritmo comienza por pares de sucesos y va incrementando los conjuntos para los casos en el que las frecuencias conjuntas superen un umbral. El algoritmo "a priori" es un ejemplo de estas técnicas [Agrawal and Srikant, 1994].

- *Técnicas basadas en árboles de decisión y sistemas de aprendizaje de reglas:* Se basan el algoritmos del tipo "divide y vencerás" como el ID3/C4.5 [Quinlan, 1993] o el CART y los denominados "separa y vencerás" como el CN2 [Clark and Niblett, 1989]. Más adelante se profundiza la descripción de estas técnicas.

- *Técnicas relacionales y estructurales:* Representan los modelos mediante lenguajes declarativos como los lenguajes lógicos y funcionales. La mayoría de las técnicas de $\mathcal{DM}$ trabajan sobre datos en formato atributo-valor (vista minable), para extender el aprendizaje a una representación del conocimiento de forma estructural o relacional, se debe cambiar el lenguaje de representación y usar lógica de primer orden, esta es la idea de las técnicas relacionales (RDM) [Dzeroski and Lavrač, 2001]. La programación lógica inductiva (ILP) es una rama del aprendizaje automático en la que la programación lógica se emplea como técnica de representación uniforme de ejemplos, conocimientos de base e hipótesis.

- *Técnicas basadas en redes neuronales artificiales:* Son un método de aprendizaje que parte de la presunción de que la capacidad humana de procesar información se debe a la naturaleza biológica del cerebro, y para imitar esta característica se basan en el uso de soportes artificiales similares a los del cerebro . Algunas variantes son las redes multicapas [Laboratories et al., 1960], perceptrón simple [Rosenblatt, 1962], redes de Kohonen [Kohonen, 1988], que permiten realizar clasificación no supervisada, etc.

- *Técnicas basadas en núcleo y máquinas de soporte vectorial:* Representan vectorialmente los ejemplos, con un componente real para cada atributo, el vector se suele denominar vector de pesos. El modelo de maquinas de soporte vectorial (SVM) fue presentado en 1992 por Vapnik, Boser y Guyon [Boser et al., 1992] y descripto en [Cortes and Vapnik, 1995] y [Vapnik, 1998]. Intentan maximizar el margen entre los grupos o clases formadas mediante transformaciones llamadas funciones núcleo, que calculan el producto escalar de dos vectores en el espacio de características, es importante la elección de la función núcleo a utilizar, que debe reflejar el conocimiento a priori del problema.

- *Técnicas estocásticas y difusas:* Junto con las redes neuronales estas técnicas forman lo que se llama computación flexible. Son ejemplo los métodos evolutivos [Tettamanzi et al., 2001] y las funciones de lógica difusa, tales como el algoritmo de reglas difusas para generación de reglas de Wang y Mendel [I.X. and J.M., 1992].

- *Técnicas basadas en casos, en vecindad o distancia:* Se basan en las distancias al resto de los elementos, como vecinos más próximos o los algoritmos jerárquicos como Two-step o COBWeb [Fisher, 1987] y los no jerárquicos como k-medias [Moody and Darken, 1989] [MacQueen, 1967]. Dado que son algoritmos que se adaptan a las tareas descriptivas, se analizan en secciones posteriores con el objeto de ser utilizados como primera aproximación al problema planteado.

Todas las tareas (exceptuando quizá las reglas de asociación y correlaciones) y los métodos descriptos se centran en la idea del aprendizaje inductivo. El aprendizaje inductivo es un tipo especial de aprendizaje capaz de obtener reglas o modelos que generalizan o abstraen la evidencia determinada por un conjunto de ejemplos particulares.

El aprendizaje puede ser incremental o no, dependiendo de la forma en que se presentan los datos. Si se considera que los datos pueden ir cambiando por períodos de tiempo (año académico por ejemplo) podría ser interesante utilizar métodos específicos para el aprendizaje incremental, que permite revisar los modelos aprendidos y no tener que realizarlos de nuevo con todos los datos.

3.4. Elección de técnicas aplicables

En este punto, cuando ya se han enumerado y caracterizado las técnicas de $\mathcal{DM}$ disponibles, es cuando se hace necesario enfocar el análisis al dominio de estudio para profundizar en aquellos métodos que permitan obtener los resultados esperados. En los próximos apartados del trabajo se analizan algunas de las técnicas enunciadas en detalle, poniendo énfasis en la posibilidad de aplicación a las tareas que se relacionan con el estudio de la deserción de alumnos universitarios. Luego se utilizan algunas de ellas con los datos preparados en secciones anteriores a fin de demostrar su aplicabilidad al caso de estudio y elegir las que provean un modelo que se adapte a los objetivos planteados.

La información de la que se dispone incluye algunos datos demográficos, económicos, sociales, familiares y académicos de los alumnos inscriptos en todas las carreras de grado de la UNRN desde

su creación. Esta información podría ayudar a conocer los perfiles de los estudiantes que alberga la institución, en particular los perfiles de los alumnos desertores.

El mayor conocimiento del estudiante universitario es un tema que despierta el interés de las autoridades de la Universidad, ya que definir sus características aportaría a la organización la información mínima necesaria para implementar medidas paliativas de los fenómenos de abandono y desgranamiento. La definición de los factores sociales, económicos, escolares y familiares que describen a los estudiantes y su influencia en los índices de deserción parecen ser el primer paso en el camino hacia la disminución del riesgo de fracaso académico.

Estas consideraciones llevan a iniciar la investigación del caso de estudio mediante las técnicas descriptivas, que permitan resumir las características generales del conjunto de datos respecto a la información socio-económica y/o al rendimiento académico de los estudiantes de las diferentes cohortes en los años de vida de la UNRN y que determinen próximos caminos a seguir hacia la obtención del modelo.

En la sección siguiente se describen los métodos de agrupamiento, con el objetivo de utilizarlos para organizar los datos de los estudiantes en grupos similares. A partir de la caracterización de dichos grupos se espera poder describir sus perfiles y ayudar a la comprensión inicial del problema. Posteriormente se analizan métodos básicos de $\mathcal{DM}$, como los árboles de decisión y las reglas de clasificación, métodos de aprendizaje supervisado que podrán ser utilizados para la construcción de soluciones de carácter predictivo que permitan clasificar a nuevos estudiantes en desertores y no desertores.

3.5. Técnicas aplicables al problema de deserción universitaria

El agrupamiento (clustering) es el proceso de organizar ejemplos en grupos cuyos miembros son similares en algún sentido. Un grupo o *cluster* es una colección de ejemplos que son similares entre sí y diferentes a los ejemplos en otro grupo [Witten and Frank, 2011].

El primer paso, entonces, es determinar qué se entiende por similitud, o qué da lugar al concepto matemático de distancia, función inversa de la similitud.

3.5.1. Medidas de distancia

El conjunto de datos que se va a agrupar puede considerarse como una colección de vectores n-dimensionales. En particular, los vectores en un espacio Euclídeo están formados por números reales, que en el caso de ejemplos del conjunto de datos, corresponden a los valores que toman los atributos para cada ejemplo. Calculando la distancia entre dos ejemplos (individuos del conjunto) se puede determinar la similitud entre los mismos. Hay diferentes funciones para calcular la distancia entre dos vectores de números reales, algunas de las mas utilizadas son [Rajaraman and Ullman, 2011]:

- *Distancia de Manhattan*, o distancia por cuadras, que recorre un camino zigzagueando, es el

promedio de las diferencias entre dimensiones:

$$d(x,y) = \sum_{i=1}^{n} \mid x_i - y_i \mid \tag{3.1}$$

- *Distancia de Chebychev*, que calcula la discrepancia más grande en alguna de las dimensiones, se usa cuando se quiere considerar que los individuos son diferentes si lo son en una de las dimensiones:

$$d(x,y) = \max_{i=1..n} \mid x_i - y_i \mid \tag{3.2}$$

- *Distancia coseno:* la distancia es el coseno del ángulo que forman los vectores:

$$d(x,y) = arccos\left(\frac{x^t\, y}{||x||\ .\ ||y||}\right) \tag{3.3}$$

- *Distancia de Mahalanobis*, que generaliza la distancia euclídea admitiendo escalas lineales arbitrarias y rotaciones del espacio de características [Davis et al., 2007], tiende a eliminar información redundante:

$$d(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)} \tag{3.4}$$

- *Distancia Euclídea:* se define como la longitud de la recta que une dos puntos en el espacio euclídeo, es una de las mas utilizadas:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.5}$$

Se hace evidente en este punto que es muy conveniente normalizar todos los atributos, como se explicó en la sección de preparación de datos. Si no se normaliza, alguna dimensión (atributo) puede tener una magnitud media superior al resto y pesará mucho más a la hora de calcular las distancias.

La detección de valores anómalos también es importante, ya que la normalización puede verse muy afectada por estos valores. Es además recomendado numerizar los atributos nominales, para poder utilizar la medida de distancia definida, caso contrario debe utilizarse alguna función que se adapte a atributos nominales.

A continuación se describen brevemente algunos métodos que permiten modelizar la información disponible utilizando alguna medida de distancia, es decir, que se trata de un proceso no supervisado.

3.5.2. Agrupamiento por centroides

El algoritmo K-medias o *K-means*, es uno de los algoritmos de agrupamiento basados en centroides más conocido [MacQueen, 1967]. Es un método de agrupamiento adaptativo que requiere conocer de antemano el número de grupos a formar, k.

El algoritmo está basado en la minimización de la distancia interna (la suma de las distancias de los patrones asignados a un agrupamiento con respecto al centroide de dicho agrupamiento). De hecho, este algoritmo minimiza la suma de las distancias al cuadrado de cada patrón al centroide de su agrupamiento.

El algoritmo parte de una cantidad de ejemplos a agrupar y de k prototipos. La idea es situar a los prototipos (o centros) en el espacio, de forma que los datos pertenecientes al mismo prototipo tengan características similares.

El algoritmo comienza calculando para cada ejemplo x_n el prototipo más próximo C_j e incluye el ejemplo en la lista de dicho prototipo. Después de haber introducido todos los ejemplos, cada prototipo C_j tendrá un conjunto de ejemplos a los que representa. Se desplaza el prototipo hacia el centro de masa de su conjunto de ejemplos y se repite el procedimiento hasta que los prototipos ya no se desplacen. En ese momento los ejemplos de entrada quedan divididos en k grupos y el prototipo correspondiente se encuentra en el centro del mismo, por lo que también es denominado *centroide*. Estos centros minimizan las distancias cuadráticas euclídeas entre los ejemplos de entrada y el centro más cercano, es decir, minimizan el valor de J indicado en la ecuación 3.6

$$J = \sum_{j=1}^{k} \sum_{n=1}^{m} M_{x_n,C_j} D(x_n - C_j)^2 \tag{3.6}$$

$$M(x_n, C_j) = \begin{cases} 1 & si\ D(x_n - C_j) < D(x_p - C_j)\ \ \forall p\ ;\ n \neq p \\ 0 & sino \end{cases} \tag{3.7}$$

donde m es el tamaño del conjunto de ejemplos, D es una medida de distancia, x_n es el n-ésimo ejemplo de entrada, C_j es el prototipo de la clase j y $M(j, n)$ es la función de pertenencia del ejemplo n a la clase j indicada en 3.7

El algoritmo 7 representa el proceso descripto previamente.

Algoritmo 7: Algoritmo de construcción de grupos utilizando K-Medias

```
Function KMedias(E: conjunto de ejemplos, k:cant. de grupos a formar)
begin
    C ← Tomar al azar k ejemplos como centros iniciales
    Asignaciones ← Asignar cada ejemplo a su centro más cercano
    repeat
        C ← Recalcular los centros promediando los ejemplos asignados a cada uno
        AsigAnteriores ← Asignaciones
        Asignaciones ← Asignar cada ejemplo a su centro más cercano
    until Asignaciones = AsigAnteriores
    return C
```

En la Figura 3.1 se grafica el desplazamiento de los prototipos y se puede apreciar como se van definiendo los grupos o clusters.

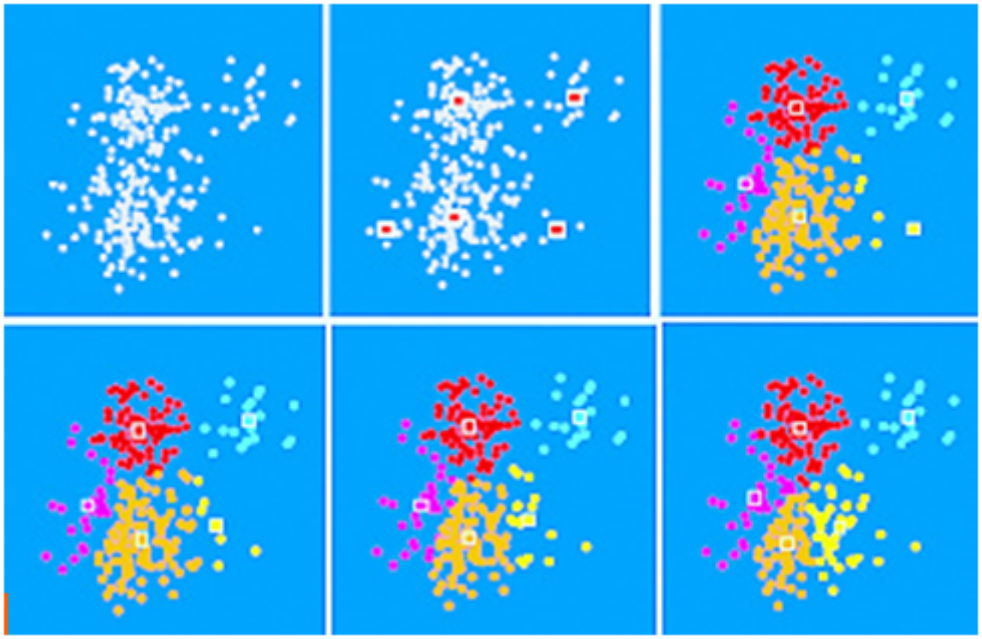

Figura 3.1: Las distintas figuras (comenzando por el extremo superior izquierdo hasta el inferior derecho) ejemplifican el desplazamiento de los prototipos durante el proceso de entrenamiento del método k-medias. Los colores permiten apreciar como se van definiendo los clusters

La ventaja principal del algoritmo k-medias es su simplicidad y eficiencia; es fácil de entender y de implementar. El tiempo de ejecución es del orden de t, k y m, donde m es el número de ejemplos, k el número de clusters y t el número de iteraciones. Como k y t son generalmente mucho menores que m, se considera que k-medias es un algoritmo lineal con respecto a la cantidad de ejemplos a agrupar.

Una de sus desventajas es que los datos a agrupar deben tener una media definida, lo que complica su aplicación a variables nominales. Algunas variaciones del algoritmo (por ejemplo k-modes) utilizan la moda en lugar de la media para el centroide, la otra opción es numerizar los datos de entrada, como ya se ha visto.

Otra desventaja es la necesidad de determinar el valor de k previamente. Existen otros métodos que utilizan distintas métricas para determinar la cantidad adecuada de clusters a formar, por ejemplo el algoritmo ISODATA [Ball and Hall, 1965].

Esta variante divide los grupos si la desviación estándar del grupo excede un cierto parámetro y el número de ejemplos excede el doble del parámetro que determina el valor mínimo de miembros para un grupo. Por otro lado, mezcla dos grupos si el número de ejemplos en ellos es menor que un parámetro dado o si los centros de ambos están suficientemente cercanos según una parámetro de distancia.

Dada la cantidad de información a manejar en el caso de estudio, un método como el ISODATA puede requerir un tiempo de ejecución elevado, por tal motivo, se consideró ejecutar el algoritmo k-medias con varios valores de k y elegir el que genere el resultado más adecuado. A pesar de sus desventajas, k-medias es el algoritmo más popular, debido a su simplicidad y eficiencia.

Comparar algoritmos de agrupamiento no es una tarea fácil, dado que, a diferencia del aprendizaje supervisado, no se conoce con anterioridad a la aplicación de las técnicas, cuáles debieran ser los agrupamientos correctos. Se exponen aquí algunos métodos de evaluación:

- *Inspección del usuario:* Se consulta un panel de usuarios que conocen el dominio para que inspeccionen los clusters resultantes. Este es un proceso subjetivo y una labor manual que consume tiempo y esfuerzo. Sin embargo, en la mayoría de las aplicaciones es necesario algún nivel de inspección manual, que puede ser acompañado de otros métodos de aprendizaje supervisado, como árboles o reglas que caractericen los clusters y ayuden al usuario a interpretar los resultados.
- *Ground Truth:* Este método utiliza conjuntos de datos clasificados (con una variable de clase) para evaluar algoritmos de agrupamiento. Se puede asumir que cada clase debe corresponder a un grupo y luego aplicar el algoritmo de clustering, comparar la pertenencia a los grupos con la pertenencia a las clases para determinar la calidad del agrupamiento. Para ello se pueden usar medidas como la entropía o la pureza, entre otras.
- *Información interna:* Evalúan los grupos basados en la información interna de los mismos. Miden la cohesión intra-cluster y la separación inter-cluster. La cohesión mide cuan cerca están los ejemplos de su centroide, por ejemplo por medio de la suma de errores cuadrados. La separación mide cuan alejados están los centroides de diferentes grupos, utilizando cualquier medida de distancia.
- *Evaluación indirecta:* En algunas aplicaciones, el agrupamiento no es la tarea primaria, sino que es usada para ayudar a otra tarea más importante. En este caso se puede usar la evaluación de la tarea primaria para determinar cual algoritmo de agrupamiento es mejor para dicha tarea.

Una vez que se encuentra el conjunto de clusters, la próxima tarea es encontrar una manera de representarlos, si bien para algunas aplicaciones el solo hecho de decir a que grupo pertenece un elemento es suficiente, en otras, como la que trata este trabajo, en la que se involucra la toma de decisiones, los grupos resultantes deben ser representados de una manera compacta y entendible, de manera de facilitar su uso.

Se pueden enumerar tres formas de representar grupos [Liu, 2011]:

1. Utilizar el centroide de cada grupo para representarlo, dado que el centroide indica los valores del centro del grupo. La representación por centroides funciona bien para grupos con forma hiper-esférica.
2. Utilizar modelos de clasificación para representar los grupos. Este método trata los grupos como clases y luego se ejecuta un algoritmo de aprendizaje supervisado sobre los datos para encontrar un modelo de clasificación (por ejemplo un árbol de decisión o conjunto de reglas que distinga entre las clases).
3. Utilizar valores frecuentes en cada grupo para representarlo. Este método funciona bien para variables nominales.

3.5.3. Agrupamiento jerárquico

Los algoritmos de agrupamiento jerárquico se inician considerando a cada ejemplo un grupo distinto y a medida que el procesamiento avanza se van construyendo nuevos grupos combinando dos grupos más pequeños.

Si los ejemplos pertenecen a un espacio euclídeo, los grupos están representados por sus centroides. Al inicio del algoritmo cada punto es el centroide de su grupo. El criterio para unir grupos entonces será la mínima distancia entre sus centroides. Lo que resta definir es el criterio de corte del proceso de unión de grupos, hay diferentes modelos que pueden seguirse en este sentido, algunos son:

- Determinar el número de grupos deseado.

- Detener la combinación de grupos cuando alguna combinación produce un cluster inadecuado por un criterio definido, por ejemplo, cuando la distancia entre el centroide y alguno de sus elementos supera un límite pre establecido.

La forma descripta de construcción de agrupamiento jerárquico es también denominada agrupamiento aglomerativo y se resume en el algoritmo 8.

Algoritmo 8: Construcción de grupos utilizando un algoritmo jerárquico aglomerativo

```
Function(Aglomerativo(E: conjunto de ejemplos))
begin
    G ← Considerar inicialmente a cada ejemplo como un grupo diferente
    while no se alcance el criterio de terminación do
        aux = 0
        for cada par de grupos distintos G_i y G_j do
            if SIMILAR(G_i, G_j) > aux then
                aux = SIMILAR(G_i, G_j)
                p = i
                q = j
        Unir los grupos G_p y G_q
    return G
```

En contraposición existe el agrupamiento desaglomerativo o divisivo, que parte de un único grupo con todos los elementos y se van haciendo divisiones paulatinas en subgrupos. Un ejemplo de agrupamiento divisivo que escala bien para un número de ejemplos mayor es el COBWEB [Fisher, 1987] [Gennari et al., 1990].

3.5.4. Mapas auto-organizativos

Las redes neuronales competitivas con entrenamiento no supervisado son una de las herramientas más utilizadas para resolver problemas de clustering ya que no requieren del conocimiento de soluciones aisladas del problema para realizar su aprendizaje. En esta categoría, los mapas auto-organizativos (SOM, del inglés *Self-organizing Map*) [Kohonen, 1982] han demostrado ser capaces de aprender la organización de los datos de entrada permitiendo obtener una estructura que respeta su topología.

Puede ser representada como una estructura de dos capas: la capa de entrada cuya función es sólo permitir el ingreso de la información a la red y la capa competitiva que es la encargada de realizar el agrupamiento. Las neuronas que forman esta segunda capa se encuentran conectadas y poseen la capacidad de identificar la cantidad de "saltos" o conexiones que la separan de cada una de las restantes dentro de este nivel. Cada neurona competitiva lleva asociado un vector de pesos o centroide representado por los valores de los arcos que llegan a ella desde la capa de entrada.

De esta forma, la red SOM maneja dos estructuras de información: una referida a los centroides asociados a las neuronas competitivas y otra encargada de determinar la proximidad entre neuronas. Esto, a diferencia de un método del estilo *"winner-take-all"* como el método k-medias, brinda información adicional con respecto a los agrupamientos ya que neuronas cercanas dentro de la arquitectura representarán agrupamientos similares en el espacio de los datos de entrada.

Inicialmente los pesos de la red, almacenados en una matriz que se denominará W, son aleatorios y se adaptan con las sucesivas presentaciones de los vectores de entrada. Por tratarse de una estructura competitiva, cada vector de entrada se considera representado por (o asociado con) la neurona competitiva que posea el vector de pesos más parecido según una medida de similitud dada. El valor final de W se obtiene mediante un proceso iterativo que se repite hasta que los vectores de pesos no presenten modificaciones significativas o lo que es lo mismo, hasta que cada vector de entrada sea representado por la misma neurona competitiva que en la iteración anterior. En cada iteración, para cada vector de entrada se determina la neurona que lo representa. A esta neurona se le llama neurona ganadora ya que es la que gana la competencia por la representación del vector (es la más parecida hasta el momento). Luego se actualiza el vector de peso de dicha neurona y de su vecindad según la ecuación (3.8)

$$w_{ij} = w_{ij} + \alpha(x_{ij} - w_{ij}) \qquad i = 1..n \tag{3.8}$$

siendo n la dimensión del espacio de entrada, j la neurona competitiva cuyo vector se desea actualizar y α un valor entre 0 y 1 que representa la velocidad de aprendizaje. La ecuación (3.8) tiene variantes que pueden consultarse en [Kohonen et al., 2001]. El concepto de vecindad es utilizado para permitir que la red se adapte adecuadamente. Esto implica que neuronas competitivas vecinas representan patrones de entrada similares. Por tal motivo, durante el proceso de entrenamiento (obtención de los valores de W) se comienza con una vecindad amplia para luego ir reduciéndola a lo largo de las iteraciones.

Sin embargo, el SOM y otras redes similares tienen dos grandes limitaciones. En primer lugar, la dimensión y estructura de la red deben ser definidas a priori, antes de comenzar con el

entrenamiento, condicionando de esta forma los resultados y la eficiencia de la respuesta obtenida. En segundo lugar, la capacidad de la red está definida por el número de nodos que contiene así como los parámetros de aprendizaje. Los mapas auto-organizativos dinámicos buscan resolver estos problemas. Entre los distintos métodos existentes propuestos para definir la arquitectura puede observarse que la incorporación de elementos es variada encontrando redes neuronales que los agregan de manera aislada hasta otras que adicionan capas completas [Alahakoon et al., 2000] [Fritzke, 1994][Fritzke, 1995]. También se han definido algoritmos de entrenamiento supervisados para este tipo de redes [Jirayusakul and Auwatanamongkol, 2007].

3.5.5. Arboles de decisión

Los árboles de decisión son una de las técnicas más usadas para clasificación. La precisión de su clasificación es del nivel de otros métodos, son muy eficientes y fáciles de utilizar y entender.

El modelo "divide y vencerás" y la característica de los problemas de clasificación que asumen que las clases son disjuntas, lleva naturalmente al estilo de representación de un árbol [Liu, 2011]. Los nodos en un árbol de decisión involucran una condición sobre un atributo en particular (usualmente la comparación con una constante), el resultado, que determina opciones excluyentes entre sí, define la rama del árbol por la que se avanza. Los nodos hojas dan la clasificación que se aplica a todas las instancias que alcanzan esa hoja. Para clasificar un nuevo ejemplo, sólo hay que conducirlo por el árbol de acuerdo a los valores de sus atributos de cada nodo y cuando se alcanza una hoja, el ejemplo queda clasificado de acuerdo a la clase de la hoja.

Si el atributo que se compara en un nodo es nominal, el número de hijos es igual a la cardinalidad del atributo. En este caso el atributo no será comparado nuevamente más abajo en el árbol, dado que se agotaron todos sus posibles valores en esta comparación. En cambio, los valores de un atributo numérico pueden dividirse en subconjuntos, en ese caso el atributo podría ser utilizado en otro nodo con una nueva comparación más específica. Esto se puede apreciar en la figura 3.2 donde el atributo `anio_nacim` que contiene un número entero correspondiente al año de nacimiento del alumno aparece varias veces. Puede verse en dicha figura que el nodo raíz separa los alumnos según si su año de nacimiento es menor o igual y mayor a 1986. Luego la rama derecha correspondiente a los que tienen año de nacimiento mayor a 1986 vuelve a dividirse según si lo hicieron antes o después de 1990, utilizando una división más específica. Como ya se dijo anteriormente, la aparición de un mismo atributo más de una vez sobre la misma rama del árbol sólo es válido para los atributos numéricos.

Los algoritmos básicos de aprendizaje de árboles de decisión se basan en la disyunción de las clases, de manera que las particiones en el árbol también deben ser disjuntas. De esta manera el espacio de instancias se va partiendo de arriba hacia abajo mediante condiciones excluyentes y exhaustivas. Por tal motivo, es importante la selección de "buenas" particiones, es decir que, deben seleccionarse en primer lugar los atributos que mejor separen los ejemplos entre todos sus hijos.

Los atributos que proveen buenas particiones son aquellos relevantes para el problema, de manera que los árboles proveen también una solución a las tareas descriptivas, mostrando jerárquicamente la

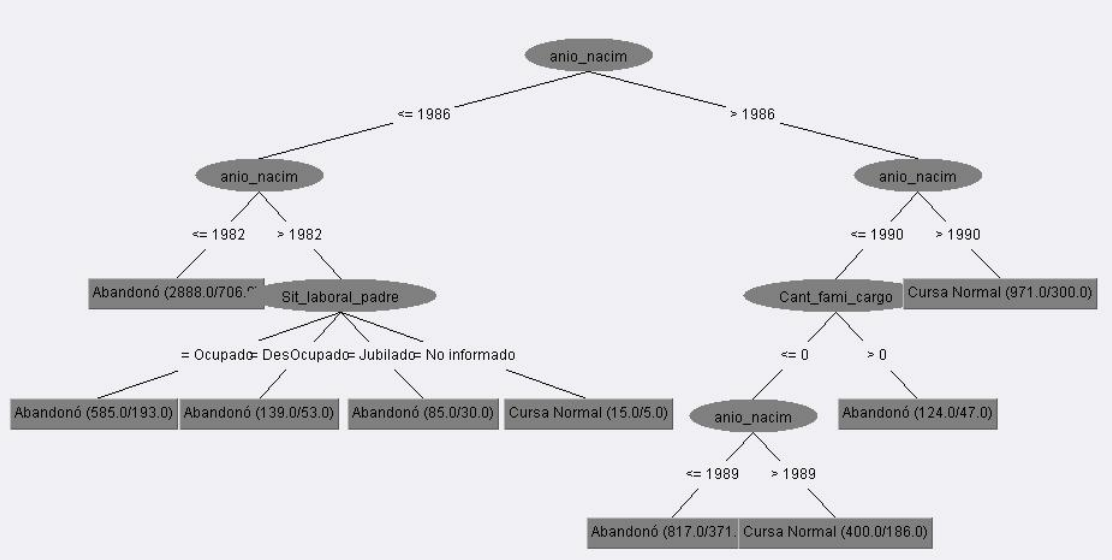

Figura 3.2: Ejemplo de Arbol de decisión

Algoritmo 9: Algoritmo genérico de construcción de un árbol

Function Particion(E: conjunto de ejemplos)
begin
  if *(todos los ejemplos de E son de la misma clase c)* ***or*** *(son muy pocos como para dividirlos)*
  then
    $N \leftarrow$ Generar un nodo hoja
    $N.clase \leftarrow$ la clase mayoritaria de E
  else
    Seleccionar el atributo con menor desorden
    $N \leftarrow$ Generar un nodo con tantas ramas como valores tenga el atributo seleccionado
    for *i=1* **to** *Cantidad de valores posibles para el atributo seleccionado* **do**
      // E_i es el conjunto de ejemplos que corresponden a la rama i del nodo N
      $N.rama(i) \leftarrow Particion(E_i)$
  return N

organización de la información.

Algunos algoritmos para árboles de decisión necesitan que todos los atributos sean discretos, si existen atributos con valores continuos deben ser discretizados. Tal es el caso del método ID3 definido por J. Ross Quinlan de la Universidad de Sydney en Australia [Quinlan, 1986].

El árbol se construye en forma recursiva, de arriba hacia abajo. Al comienzo, todos los ejemplos del conjunto de datos están en el nodo raíz y se particionan recursivamente basado en los atributos seleccionados. El particionamiento se detiene cuando todas las muestras para un nodo dado corresponden a la misma clase, cuando no hay más atributos para particionar, o cuando no quedan más ejemplos. El algoritmo 9 genera un árbol de decisión para un conjunto de ejemplos E siguiendo este proceso.

Lo que falta determinar es la forma de seleccionar los atributos, esta tarea puede ser heurística o mediante una medida estadística, por ejemplo la ganancia de información. La idea es elegir el atributo que tenga la mayor cantidad de elementos en subconjuntos homogéneos respecto a la clase. Se explica ahora el cálculo de la medida de ganancia de información usado como base para la selección de atributos.

La ganancia de información se relaciona con la cantidad de información obtenida al tomar una decisión [Witten and Frank, 2011]. Para ello se calcula el desorden promedio producido por la selección de un atributo:

$$Desorden\ promedio = \sum_{b} \left(\frac{N_b}{N_t} \right) * \left(\sum_{c} -\frac{N_{bc}}{N_b}\ log_2\ \frac{N_{bc}}{N_b} \right) \tag{3.9}$$

donde N_b es el número de ejemplos en la rama b, N_t es el número total de ejemplos en todas las ramas y N_{bc} es el total de ejemplos en la rama b de la clase c. Este cálculo se aplica a cada uno de los atributos que pueden ser seleccionados para el árbol y da como resultado un número real entre 0 y 1 que será más chico cuanto más homogéneos sean los subconjuntos que este atributo genere. Una vez calculados los desórdenes promedio de todos los atributos, se elige el de menor valor.

El algoritmo ID3 fue mejorado para convertirse en el algoritmo C4.5 incorporando la capacidad de operar con atributos numéricos [Quinlan, 1993].

Los algoritmos de aprendizaje de árboles de decisión obtienen un modelo que cubre todos los ejemplos del conjunto utilizado. Esta situación que puede parecer ideal, es un ajuste demasiado estricto a la evidencia y suele provocar que el modelo trabaje mal para nuevos ejemplos. La solución a este problema se resuelve con la denominada “poda” del árbol obtenido. La poda elimina nodos inferiores de un árbol que se consideran demasiado específicos.

La poda puede realizarse durante el proceso de construcción (prepoda) o luego de éste (pospoda). En el primer caso se trata de determinar el criterio de parada al momento de seguir especializando una rama, y se basa en el número de ejemplos en un nodo, en el número de excepciones respecto a la clase mayoritaria (error esperado) u otras técnicas más sofisticadas. La pospoda trata de eliminar nodos de abajo hacia arriba hasta un límite, basado en las mismas medidas que la prepoda. La diferencia es que la pospoda se realiza con una visión completa del modelo, pudiendo por eso obtener mejores resultados.

Cuando se poda se consiguen nodos impuros (con elementos de diferentes clases), normalmente se elige la clase mayoritaria para etiquetar el nodo hoja.

Es una difícil tarea determinar el nivel de poda con exactitud, dado que depende en gran medida de los datos específicos de cada problema, es por esto que suele utilizarse el conjunto de datos de validación para esclarecer este punto, si se dispone de ellos.

3.5.6. Reglas de Clasificación.

Los sistemas de reglas son una generalización de los árboles de decisión en la que no se exige exclusión ni exhaustividad en las condiciones de las reglas (es decir, podría aplicarse más de una regla o ninguna).

Se puede expresar un árbol de decisión en forma de reglas del tipo:

```
SI <condición>ENTONCES clase = <valor>
...
EN OTRO CASO clase = <valor>
```

También se pueden obtener reglas mediante un mecanismo de cobertura. En este caso, el objetivo es tomar cada clase y buscar las condiciones de reglas (par atributo-valor) que cubran la mayor cantidad de ejemplos de una clase y la menor cantidad del resto de las clases. Se intenta maximizar la cobertura minimizando errores [Witten and Frank, 2011].

La cobertura secuencial aprende una lista de reglas secuencialmente, una a la vez, para cubrir los datos de entrenamiento. Después de aprender cada regla, los ejemplos de entrenamiento cubiertos por la regla son removidos del conjunto. Solamente se utilizan los elementos restantes para encontrar las reglas subsiguientes. Una regla cubre un ejemplo si éste satisface las condiciones de la regla.

Las reglas suelen ser más compactas que los árboles, sobre todo si se puede establecer una regla por defecto. En el aprendizaje de árboles de decisión, en cada paso se evalúan todos los atributos y se elige uno para dividir los datos en m subconjuntos disjuntos, donde m es el número de valores del atributo. La inducción de reglas evalúa todos los pares atributo-valor (condiciones) y selecciona sólo uno. La cantidad de pares atributo-valor es mucho mayor que el número de atributos. De esta manera, cada paso en la construcción de un árbol genera m reglas, mientras que cada paso de la construcción de reglas genera solo una regla. Estos efectos producen que la construcción de reglas sea mucho más lenta que los árboles.

En otro sentido, las reglas del tipo si-entonces son fáciles de entender por un usuario, siempre que se tenga en cuenta que las reglas generadas por cobertura secuencial deben respetar su orden (por lo que también son llamadas “lista de decisión”). Como los datos cubiertos por una regla se remueven luego de generarla, las reglas se convierten en dependientes unas de otras [Liu, 2011].

Algunos ejemplos basados en el algoritmo de cobertura son AQ [Michalski and Larson, 1983] y CN2 [Clark and Niblett, 1989].

Capítulo 4

Enfoque del Trabajo

4.1. Motivación: El estudio de la deserción universitaria

Como se ha expresado a lo largo del documento, existe un problema tangible en la UNRN dado por el alto índice de abandono que presenta en sus pocos años de vida. La cantidad de inscriptos en los años 2009, 2010 y 2011 fue de 7381 alumnos, de los cuales cursaban normalmente al inicio del año 2012, 3652 estudiantes. Al incluir las inscripciones 2012 el número total de alumnos asciende a 11023, de los cuales cursan normalmente 4461 estudiantes a inicios de 2013.

La minería de datos viene a colaborar en la interpretación del fenómeno y en particular desde esta investigación, se pretende estudiar la metodología de extracción de conocimiento en grandes bases de datos con el objetivo concreto de aplicarla al caso de estudio. Se espera utilizar la minería de datos para obtener modelos basados en la comprensión de las variables que rigen el problema e iniciar a partir de allí la construcción de soluciones.

La deserción de estudiantes es un tema instalado en la educación superior y ha sido objeto de innumerables investigaciones y abordado desde diferentes perspectivas [Pal, 2012] [La Red Martínez et al., 2009] [Alcover et al., 2007] [Rodallegas et al., 2010]. Temas como la problemática de la educación superior y el abandono temprano de los estudios universitarios han sido investigados para tratar de indagar las variables que llevan al "fracaso" y abandono de la universidad. La mera consulta de los Anuarios de Estadísticas Universitarias que lleva adelante el Ministerio de Educación arroja porcentajes que determinan la urgencia en el tratamiento del tema. En particular, en la UNRN, el fenómeno se pudo detectar desde sus orígenes y hace necesario implementar políticas correctivas en el corto plazo.

4.2. Cronología de tareas realizadas

El presente documento resume los trabajos realizados de manera cronológica, mostrando así la naturaleza cíclica del proceso de extracción del conocimiento descripto en la sección 1.2.

Hasta el momento, en el capítulo 2 se describieron las tareas de preparación de datos llevadas a cabo y se analizaron varios métodos diferentes de selección de características. En el capítulo 3 se resumió la investigación bibliográfica que dió lugar a la elección de los métodos a utilizar sobre el conjunto de datos disponible.

En el presente capítulo se comienza a realizar pruebas de aplicación de técnicas descriptivas para el primer acercamiento, haciendo foco en la obtención de modelos que colaboren en el entendimiento del dominio del problema enfrentado. En las secciones subsiguientes se explican estas tareas en detalle y se pone en evidencia la necesidad de profundizar en la selección de características relevantes, lo que significa volver hacia atrás en el ciclo de vida del proyecto de $\mathcal{KDD}$, hacia la fase de preparación de datos.

El capítulo siguiente estará completamente abocado a la descripción de los esfuerzos realizados para la obtención de un subconjunto de atributos relevante, se presentarán las técnicas utilizadas y las implementaciones en el caso de estudio.

Sobre el final del documento, una vez resuelta la tarea central de selección de características, será posible retomar el ciclo del $\mathcal{KDD}$, centrándose nuevamente en las técnicas de $\mathcal{DM}$, esta vez con el objetivo de hallar un modelo predictivo para la deserción en la unidad académica estudiada.

4.3. Aplicación de técnicas al caso de estudio

La investigación bibliográfica que ya ha sido presentada muestra un amplio abanico de técnicas potencialmente aplicables al entorno de trabajo. Es menester en este momento decidir el punto de inicio de la aplicación de dichas técnicas, con el objetivo final de predecir la deserción estudiantil.

Los algoritmos de clustering dividen los datos en grupos significativos, de manera que los grupos capturan la natural estructura de los datos. En algunos casos el análisis de clusters es útil como punto de comienzo para otros propósitos. Hay muchas aplicaciones del análisis de clusters para problemas prácticos, tanto para propósitos de entendimiento del problema como de utilidad por sí mismos.

- *Clustering con propósito de entendimiento de problemas*: Las clases, o grupos significativos de objetos que comparten características comunes, juegan un rol importante en la forma en que la gente analiza y describe el mundo. Los seres humanos tienen la habilidad de dividir objetos en grupos (clustering) y asignar objetos particulares a esos grupos (clasificación). En el contexto de entendimiento de datos, los clusters son clases potenciales y el análisis de los mismos es el estudio de técnicas para encontrar automáticamente las clases.
- *Clustering de utilidad por sí mismo*: El análisis de clusters provee una abstracción de los objetos de datos individuales a los grupos en los cuales esos objetos residen. Además, las técnicas de clustering caracterizan cada grupo en términos de un prototipo, es decir, un objeto de datos que es representativo del resto de los objetos en el cluster. Estos prototipos pueden ser utilizados como la base para un gran número de análisis de datos y técnicas de procesamiento. Por lo tanto, en el

contexto de la utilidad, el análisis de clusters es el estudio de técnicas para encontrar los prototipos más significativos.

Por todo lo expuesto, para la primera aproximación al tratamiento del problema se seleccionan las técnicas de agrupamiento, con la meta de caracterizar a los alumnos para ofrecer elementos de análisis. Se decide utilizar la técnica de k-medias descripta anteriormente, siendo necesario determinar el conjunto de ejemplos a usar como entrada.

4.4. Selección del subconjunto de datos

Para la selección del primer subconjunto de datos a utilizar en las pruebas iniciales se toman en cuenta algunas consideraciones surgidas del conocimiento del dominio y del análisis realizado en la etapa de preparación de datos.

El conjunto de datos de que se dispone en esta primera etapa tiene a simple vista una composición heterogénea determinada por la existencia de instancias pertenecientes a alumnos ingresantes en el año 2012 de los cuales no se registra ninguna historia académica, lo que produce un vacío de información en varios atributos de la vista minable. Esta situación responde al hecho que la base de datos con la que se inicia este trabajo fue extraída desde la base del SIU-guaraní a principios de 2012, de manera que la ausencia de información es temporal. Como primera medida se exceptúan esos datos, los registros ignorados en esta etapa del análisis serán completados en años subsiguientes con los valores académicos generados y utilizados como fuente de datos para control y validación de resultados y para el desarrollo de nuevos modelos.

A partir de los datos resultantes, se decide agrupar en primer lugar el conjunto de alumnos objeto de estudio, es decir, los registros de alumnos que abandonaron. Motiva esta elección la intención de seleccionar las características relevantes para los alumnos desertores.

La implementación de muchos de los métodos utilizados a lo largo del trabajo se resuelve mediante la utilización de RapidMiner 5.2, herramienta open-source de minería de datos [RapidMiner, 2012]. El Apéndice B incluye una breve descripción de esta herramienta.

4.5. Agrupamiento para la obtención de perfiles del alumno desertor

Para la realización de las pruebas de agrupamiento de los alumnos desertores se utiliza el algoritmo k-medias. El operador que implementa k-medias en RapidMiner es `k-Means`. Como conjunto de datos de entrada se usa la vista minable generada en la sección de preparación de datos, previa selección de los registros con `estado = Abandono`.

Para la correcta utilización del algoritmo es necesario asegurar que los datos de entrada sean numéricos, para lo cual se aplica a los valores nominales de la vista minable alguno de los métodos descriptos en

2.3.6, realizando las siguientes transformaciones:

- Los atributos nominales `hora_sem_trab_alum`, `rel_trab_carrera`, `ult_est_cur_padre`, `ult_est_cur_madre`, `alu_trab_renmon`, `alu_trab_futhor`, `alu_ingre_grupfa` que poseen valores que respetan un orden, fueron transformados a valores numéricos ordenados que mantienen el orden original. Por ejemplo, `hora_sem_trab_alum` tenía valores: *"No trabaja"*, *"hasta 20 hs."*, *"de 21 a 35 hs."*, *"de 36 o más horas"* que fueron transformados a 0, 1, 2 y 3 respectivamente (operador `Map` de RapidMiner).

- El resto de los atributos nominales que no representan valores ordenados se transformaron en n atributos *dummy* representando cada uno un valor nominal original. Esta tarea fue llevada a cabo mediante el operador `Nominal to Numerical` de RapidMiner.

El resultado final de las transformaciones descriptas es la vista minable obtenida a partir de los atributos originales y puede encontrarse en el Apéndice A.

Una vez numerizadas las variables de entrada, se aplica una normalización de rango sobre todos los atributos utilizando el método de transformación z (ver Normalización de Rango en 2.3.6), mediante el operador `Normalize` de RapidMiner.

Se realizan varios intentos de aplicación de k-medias al conjunto de datos con valores diferentes de k, obteniendo finalmente 5 grupos.

Luego se enfrenta la tarea de describir los grupos obtenidos a través de sus centroides y calcular los valores frecuentes en los grupos. En todos los casos se utiliza el conocimiento del dominio para guiar la descripción, pero los resultados no son satisfactorios dada la cantidad de atributos intervinientes.

Las pruebas realizadas aplicando k-medias ponen en evidencia la gran dimensionalidad del problema, que oscurece la interpretación de los agrupamientos obtenidos. Dada la cantidad de atributos involucrados (todos los de la vista minable inicial), no es posible encontrar un conjunto de clusters descriptivo de los datos de entrada.

Como se describió en los capítulos introductorios referidos a las fases del $\mathcal{KDD}$, la selección de características puede ser guiada por el conocimiento del dominio o por técnicas específicas de $\mathcal{DM}$. En este punto surge la necesidad de utilizar las herramientas de la minería de datos para guiar la selección de un subconjunto de características (atributos) que sean relevantes para el problema.

Por esta razón se deja en suspenso la aplicación de los métodos de agrupamiento y se enfoca la tarea en la utilización de técnicas que permitan visualizar el conjunto de atributos adecuado para la aplicación de dichas técnicas.

4.6. El problema de la selección de atributos

En la sección anterior quedó determinado que es necesario un retorno a las fases de preparación de datos para obtener una vista minable que contenga solo los atributos relevantes para la descripción del problema. La técnica de agrupamiento utilizada puso en evidencia que se requiere de un tratamiento de los datos previo a la aplicación de cualquier algoritmo de $\mathcal{DM}$ para lograr una efectiva implementación y resultados útiles, significativos y de fácil comprensión.

Algunos problemas comunes a la mayoría de los algoritmos de aprendizaje automático cuando trabajan con muchos atributos se refieren a tiempos de ejecución muy elevados, pobre generalización o incomprensión de los resultados obtenidos. Por ello, desde la década de los sesenta, las investigaciones relacionadas con la selección de atributos intentan reducir el espacio de hipótesis de las bases de datos en tareas concretas, en un intento de encontrar subconjuntos de atributos que proporcionen un mejor rendimiento de los algoritmos de aprendizaje. Existen numerosos algoritmos de selección de características en la bibliografía, algunos de ellos descriptos en el capítulo 2 de este trabajo, donde se combinan técnicas de búsqueda y de evaluación de subconjuntos de atributos, llegando incluso a analizar todas las combinaciones posibles de atributos.

En este contexto, el objetivo es encontrar un subconjunto de atributos del conjunto total inicial, que incluya aquellos que sean relevantes para el objetivo perseguido de predicción de la deserción en la UNRN. En la selección de características se intenta escoger el subconjunto mínimo de atributos de acuerdo con dos criterios: que la tasa de aciertos no descienda significativamente; y que la distribución de clase resultante sea lo más semejante posible a la distribución de clase original, dados todos los atributos. En general, la aplicación de la selección de características ayuda en todas las fases del proceso de minería de datos para el descubrimiento de conocimiento.

Es un hecho que el comportamiento de los clasificadores mejora cuando se eliminan los atributos no relevantes y redundantes. La selección de los atributos relevantes se debe a la preferencia por los modelos más sencillos frente a los más complejos.

Capítulo 5

Selección de características

En este capítulo se aborda uno de los principales problemas en el campo del Aprendizaje Automático, como es el identificar un conjunto representativo de atributos para construir un modelo.

La tarea de encontrar las características relevantes de un conjunto de datos, permite que los algoritmos utilizados posteriormente enfoquen su aprendizaje en aquellos aspectos de los datos más útiles para el análisis y la predicción.

El proceso de selección de atributos puede beneficiar a los algoritmos aplicables a un determinado problema, eliminando datos irrelevantes o redundantes para la tarea y, en algunos casos, mejorando la performance del algoritmo.

5.1. Selección de características relevantes

Este nuevo marco de trabajo se enfrenta con un problema: las técnicas conocidas de selección de características parten de un conjunto de datos clasificado, es decir, un grupo de registros en el cual cada uno de ellos está asociado a una clase, y los procesos selectivos apuntan a obtener las características más correlacionadas con la clase. No es posible calcular correlación o relevancia si no existe un atributo clase en los datos.

La solución a este problema se halla en el resultado del agrupamiento inicial obtenido con k-medias. Si bien no fue posible describir los grupos de manera aceptable, sí se pudo conseguir un atributo de clase (dado por el grupo al cual cada ejemplo pertenece), de manera que la entrada a los algoritmos de selección de características es el resultado del paso anterior, el agrupamiento en cinco grupos de los alumnos desertores.

Una vez determinado el conjunto de datos de entrada, como primera aproximación a la selección de atributos, se procede a la transformación del espacio de características mediante dos de los esquemas encontrados en la bibliografía y descriptos en el capítulo 2. Las siguientes secciones describen estas transformaciones en detalle.

5.2. Aplicación de método wrapper

El primero de los métodos hace uso de un proceso de selección del tipo *selection forward* que toma en cuenta la performance de un determinado modelo de aprendizaje para realizar la validación de los conjuntos de características. Como se explicó en 2.4, el uso de un algoritmo inductivo posiciona al método dentro de los procesos wrapper. Como algoritmo de validación se utiliza un agrupamiento del tipo k-medias, con k = 5. Para la implementación del procedimiento se utilizó el operador `Optimize Selection` de RapidMiner, con la parametrización adecuada [Tito and Mullicundo, 2010].

Selection Forward o selección hacia adelante es un algoritmo de búsqueda de características de tipo determinístico, lo que en terminos informales significa que el algoritmo se comporta de forma predecible. Dada una entrada en particular, siempre producirá la misma salida, y el proceso que lo implementa siempre pasará por los mismos estados.

También se puede clasificar a *selection forward* como un algoritmo de tipo greedy lo que significa que realiza elecciones óptimas localmente en cada estado, esperando encontrar un óptimo global.

El operador de RapidMiner que implementa este método tiene algunas mejoras respecto al algoritmo standard: crea un conjunto inicial con n individuos, donde n es el número de atributos en el conjunto de datos de entrada. Cada individuo usará exactamente una de las características. Evalúa los subconjuntos de atributos y selecciona solo los k mejores. Para cada uno de los k subconjuntos hace : Si hay j atributos no usados, hace j copias del subconjunto y agrega exactamente uno de los atributos no usados previamente al subconjunto, luego itera hasta que la performance no mejora en las últimas p iteraciones.

El proceso da como resultado el subconjunto de la Tabla 5.1.

La Tabla 5.2 muestra la precisión en la predicción de la clase con el subconjunto de características seleccionado.

5.3. Aplicación de método genético

El segundo método de selección implementado está enfocado a la selección genética de características (a través de mutación y cruce), que no solo intenta maximizar la performance del conjunto de características sino también minimizar el número de ellas. Dado que no utiliza un algoritmo inductivo determinado para la evaluación, también es conocido como selección multiobjetivo, es de propósito general y se adapta a los casos de poco conocimiento del dominio del problema. Para la evaluación utiliza el método CFS [Hall, 1999]. La implementación se realiza con los operadores `Optimize Selection (Evolutionary)` y `Performance (CFS)` de RapidMiner [Tito and Mullicundo, 2010].

Como se explicó en la sección 2.4.2, un algoritmo genético es una heurística para la búsqueda que imita el proceso de evolución natural. Esta heurística se usa para generar soluciones útiles a problemas de búsqueda y optimización. Los algoritmos genéticos pertenecen a una clase mayor de algoritmos evolutivos, que generan soluciones para problemas de optimización utilizando técnicas inspiradas en

estado_civil = Soltero
Padre_vive = S
madre_vive = S
Alu_tec_int = tiene internet en la casa
Rel_trab_carrera
Alu_trab_remmon
sede = Valle Medio y Río Colorado
Lugar_nacimiento = NEUQUEN
Colegio_secundario = N
Titulo_secundario = PER.MERC.
Sit_laboral_padre = DesOcupado
Sit_laboral_madre = No informado
Alu_trab_sitimp = Relación de dependencia
Alu_trab_sitimp = no trabaja
Alu_trab_sitimp = Monotributista
Alu_trab_futtip = No trabajaré
Alu_trab_futtip = Desconoce
Alu_trab_futtip = Cuenta propia
anio_nacim
Cant_fami_cargo
Cant_hijos_alum

Tabla 5.1: Lista de atributos seleccionados por método wrapper

	true cluster_3	true cluster_4	true cluster_2	true cluster_1	true cluster_0	class precision
pred. cluster_3	1124	12	11	3	4	97.40 %
pred. cluster_4	10	1024	93	23	1	88.97 %
pred. cluster_2	9	94	879	5	4	88.70 %
pred. cluster_1	2	12	8	304	0	93.25 %
pred. cluster_0	1	3	2	0	101	94.39 %
class recall	98.08 %	89.43 %	88.52 %	90.75 %	91.82 %	

Tabla 5.2: Matriz de confusión correspondiente a los atributos seleccionados

estado_civil = Soltero
Padre_vive = S
Alu_beca = necesita beca
Rel_trab_carrera
Alu_trab_remmon
Alu_trab_futhor
sede = Rectorado
Lugar_nacimiento = ADOLFO ALSINA
Titulo_secundario = BACHILLER
Sit_laboral_padre = DesOcupado
Sit_laboral_madre = No informado
Alu_trab_sitimp = Relación de dependencia
Alu_trab_sitimp = no trabaja
Alu_trab_sitimp = Monotributista
Alu_trab_futtip = Obrero o empleado (asalariado)
Alu_trab_futtip = Cuenta propia
Anio_egreso_sec
Cant_fami_cargo
Cant_hijos_alum

Tabla 5.3: Lista de atributos seleccionados por método genético

la evolución natural, tales como la herencia, mutación, selección y cruce. La implantación utilizada parte de una población inicial, cuyo tamaño es ajustable por un parámetro y realiza la evolución a través de los operadores de mutación y cruce, según parámetros que ajustan la probabilidad de cada uno de ellos.

El operador Performance (CFS) crea una medida de performance basada en filtro para un subconjunto de características. Evalúa el valor de un subconjunto de atributos considerando la habilidad predictiva individual de cada característica junto con el grado de redundancia entre ellas. Se prefieren los subconjuntos de atributos mas altamente correlacionados con la clase con baja intercorrelación.

El método genético utilizado da como resultado el conjunto de atributos de la Tabla 5.3.

Se puede apreciar que los subconjuntos de características seleccionados por ambos métodos son muy similares. Los atributos `estado_civil`, `padre_vive`, `rel_trab_carrera`, `alu_trab_remmon`, `cant_fami_cargo` y `cant_hijos_alum`, `Sit_laboral_padre` = *"DesOcupado"*, `Sit_laboral_madre` = *"No informado"* y todas las variables *dummy* del atributo original `alu_trab_sitimp` coinciden en ambos subconjuntos. Los atributos originales `sede`, `lugar_nacimiento`, `titulo_secundario` y

Descripción		Nombre del Atributo	Seleccionado por	
			Wrapper	Genético
Alumno Soltero		estado_civil = soltero	SI	SI
Padre alumno vive		padre_vive = SI	SI	SI
		situacion_laboral_padre	SI	SI
Internet en casa		alu_tec_int	SI	NO
Necesita beca		alu_beca = necesita beca	NO	SI
Trabajo actual	Situación laboral	alu_trab_sitimp	SI	SI
	Relacionado con carrera	rel_trab_carrera	SI	SI
	Sueldo mensual	alu_trab_remmon	SI	SI
Sede en la que estudia		Sede	SI	SI
Lugar nacimiento		lugar_nacimiento	SI	SI
Año egreso del secundario		anio_egreso_sec	NO	SI
Piensa trabajar futuro	Tipo trabajo	alu_trab_futtip	SI	SI
	Horario	alu_trab_futhor	SI	NO
Año de nacimiento		anio_nacim	SI	NO
Familiares a cargo		cant_fami_cargo	SI	SI
Hijos		cant_hijos_alum	SI	SI

Tabla 5.4: Lista de atributos seleccionados por métodos wrapper y genético

`alu_trab_futtip` aparecen (con diferentes valores) en ambos subconjuntos. Esta comparación se puede apreciar en la tabla 5.4.

5.4. Aplicación del modelo agrupamiento para las características seleccionadas

Luego de la utilización de dos métodos diferentes de selección de características, y en correspondencia con la decisión ya tomada de investigar el problema, en sus etapas preliminares, a través de métodos de agrupamiento, se opta por el subconjunto de características obtenido por el método wrapper, mostrado en la tabla 5.1 y avalado por los resultados mostrados en la tabla comparativa 5.4.

Con ese subconjunto de características, y aún con los datos originales del problema (años académicos 2009 a 2011) se volverán a ejecutar los algoritmos de agrupamiento con el fin de determinar la utilidad del proceso de selección.

El siguiente paso es la ejecución de k-medias para los atributos seleccionados, nuevamente con $k = 5$.

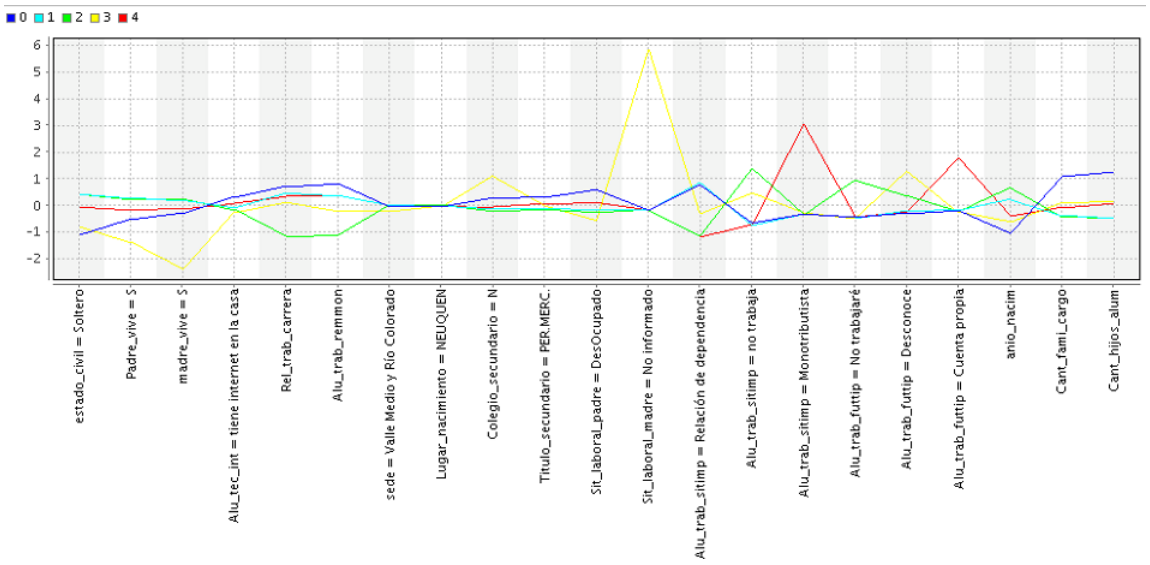

Figura 5.1: Centroides Clusters Abandonos

Los datos de entrada son esta vez los registros pertenecientes a alumnos desertores, con las mismas transformaciones descriptas en la primera corrida del método, pero solo aplicadas a los atributos del subconjunto seleccionado. El universo de ejemplos es el mismo que la primer aplicación: los alumnos con `estado` = *"Abandono"*, la diferencia radica en los atributos que describen a cada ejemplo.

El proceso de reducción de dimensionalidad determina que los datos relevantes para agrupar a los alumnos desertores son variables de tipo socio-económicas, como la edad, el estado civil, las cargas familiares, la situación laboral actual y futura del alumno y la de sus padres.

Una vez aplicado el método, el resultado de la asignación a grupos de esta ejecución se compara con el resultado de la ejecución anterior, determinando que menos de un 10 % de los ejemplos se movieron de grupo, lo que indica que el criterio de agrupamiento se conserva a pesar de la reducción de características.

Los 5 grupos resultantes pueden ser descriptos y representados de manera que aporten valor al tratamiento del problema. Esta es la tarea que se vio obstaculizada por la alta dimensionalidad de los ejemplos, es de esperar que en esta instancia la descripción de los grupos se clarifique al estar determinada por un número notablemente menor de variables.

La figura 5.1 muestra la representación gráfica provista por RapidMiner de los centroides resultantes.

5.5. Descripción de perfiles obtenidos

En esta instancia del proceso de selección de características con el uso de dos métodos diferentes y comúnmente utilizados, descriptos en la bibliografía, se pudo hallar un grupo de atributos en primera instancia relevantes. Con este subconjunto de atributos fue posible repetir el clustering de los datos comprobando que se mantuvo en un 90 % el agrupamiento conseguido sin dicha selección. Sería

interesante comprobar que los nuevos clusters, ya definidos como comparables con los preliminares, se puedan describir de manera más clara y entendible, ya que este fue el objetivo planteado al inicio de las tareas de selección de características.

En este apartado se presentan las caracterizaciones de los grupos obtenidos en la sección anterior para los alumnos desertores.

Se utilizan los centroides y los valores frecuentes en cada grupo para representarlo, considerando las frecuencias de valores a $+/-$ una desviación estándar del valor del centroide para los atributos. Una vez realizado este trabajo se le asigna un nombre descriptivo a cada grupo:

- **Mayores Relación de Dependencia** (Cluster 0): 848 alumnos. Tienen una edad promedio de 44 años, trabajan en relación de dependencia. Su trabajo está relacionado parcial o totalmente con la carrera que cursan, poseen cargas familiares, no son solteros. Ganan más de 2000$.
- **Mayores Monotributistas** (Cluster 4): 354 alumnos. La edad promedio de este grupo es de 40 años, son en su mayoría solteros, sus padres viven, trabajan como monotributistas y más de la mitad tiene cargas familiares.
- **Mayores sin Información** (Cluster 3): 105 alumnos. Este grupo de relativamente pequeña cardinalidad se caracteriza por tener muchos atributos no informados, lo que solo permite afirmar que son de edad promedio 40 años y no tienen padres.
- **Menores Trabajan** (Cluster 1): 1259 alumnos. El promedio de edad del grupo es de 30 años, trabajan en relación de dependencia con un sueldo menor a 2000$ en tareas no relacionadas o solo relacionadas parcialmente con sus carreras, no tienen cargas familiares, son solteros, sus padres trabajan.
- **Menores no Trabajan** (Cluster 2): 1163 alumnos. Son los más jóvenes, con promedio de 26 años, no trabajan, sus padres viven, no tienen cargas familiares y son solteros.

La selección de características permitió encontrar descripciones concretas de los grupos que caracterizan a los alumnos que contienen y los diferencian claramente de los pertenecientes a otros grupos.

5.6. Agrupamiento para la obtención de perfiles del alumno no desertor.

La mera segmentación en grupos de los alumnos que abandonan permite tener un mayor conocimiento de los subgrupos que componen la clase de interés (alumnos desertores), pero no permite compararla con la clase de alumnos que continúan con sus carreras. Una opción que puede colaborar a la integración de conceptos es la realización de un agrupamiento sobre los alumnos que continúan cursando, utilizando una vista minable con los mismos atributos seleccionados.

Si se piensa en el objetivo de máxima: obtener un modelo para la deserción estudiantil, cabe esperar que el subconjunto de atributos utilizado en el mismo sirva para describir tanto los alumnos desertores

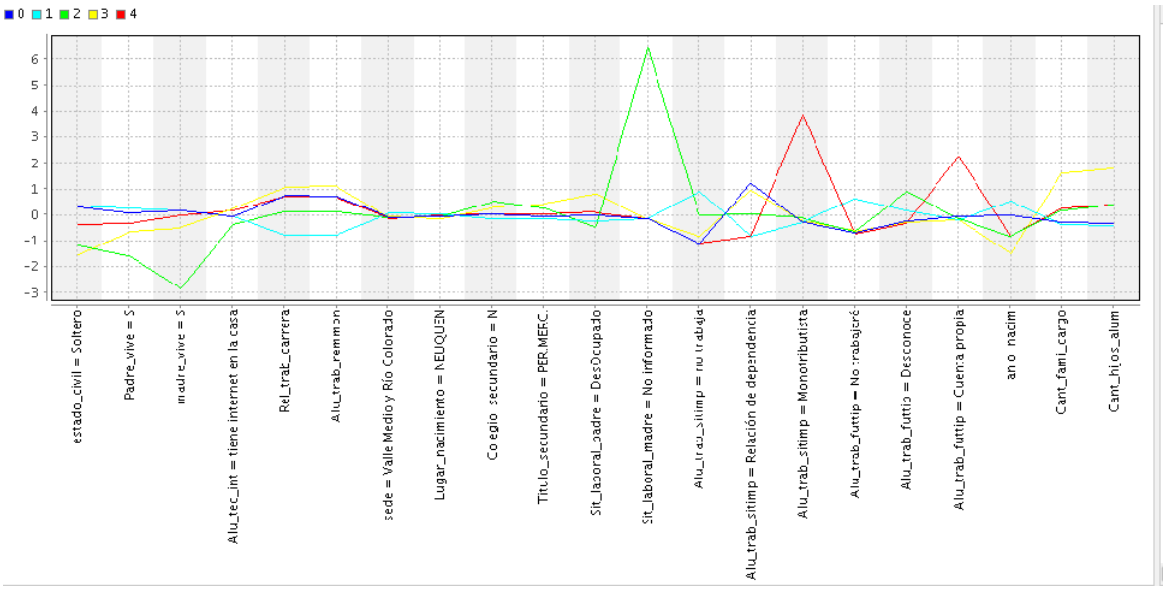

Figura 5.2: Centroides clusters cursan

como aquellos que continúan cursando, dado que el modelo predictivo deberá tender a clasificar alumnos nuevos en una de esas dos categorías (abandona / cursa).

Con la idea de evaluar la posibilidad de una comparación entre los alumnos desertores y los que no lo son (al menos hasta el momento), se preparan los datos de estos alumnos con los mismos criterios de numerización y normalización utilizados anteriormente y se ejecuta sobre ellos un método de agrupamiento k-medias con $k = 5$. La representación gráfica de los centroides obtenidos se muestra en la Figura 5.2.

Al describir los grupos obtenidos, claramente se encuentran grupos “comparables” con los anteriores, los que se caracterizan a continuación:

- **Mayores Relación de Dependencia** (Cluster 3): 511 alumnos. Tienen una edad promedio de 44 años, trabajan en relación de dependencia. Su trabajo está relacionado parcial o totalmente con la carrera que cursan, poseen cargas familiares, no son solteros. Ganan más de 2000$.

- **Mayores Monotributistas** (Cluster 4): 220 alumnos.: La edad promedio de este grupo es de 37 años, son en su mayoría solteros, sus padres viven, trabajan como monotributistas y algo menos de la mitad tiene cargas familiares.

- **Mayores sin Información** (Cluster 2): 85 alumnos. Este grupo de relativamente pequeña cardinalidad se caracteriza por tener muchos atributos no informados, lo que solo permite afirmar que son de edad promedio 37 años y no tienen padres.

- **Menores Trabajan** (Cluster 0): 966 alumnos. El promedio de edad del grupo es de 29 años, trabajan en relación de dependencia con un sueldo menor a 2000$ en tareas no relacionadas o solo relacionadas parcialmente con sus carreras, no tienen cargas familiares, son solteros, sus padres

trabajan.

- **Menores no Trabajan** (Cluster 1): 1870 alumnos. Son los más jóvenes, con promedio de 24 años, no trabajan, sus padres viven, no tienen cargas familiares y son solteros.

5.7. Interpretación de resultados del agrupamiento

La descripción y caracterización de los grupos emergentes del trabajo previo logra describir a los alumnos que han abandonado y a los que no lo han hecho, y expone una correspondencia entre los grupos análogos dentro de cada clase (donde la clase, en esta instancia evaluativa, es binaria e indica el estado o no de abandono).

El hecho de haber encontrado una segmentación comparable en atributos predominantes permite pensar en la utilización de los atributos seleccionados en algoritmos predictivos de abandono.

Por otro lado, el momento de la finalización de esta etapa y la obtención de este grupo de atributos, prácticamente coincide con la finalización del año académico 2012. Por ese motivo a partir de esta instancia se cuenta con los datos académicos de dicho año, faltantes en el conjunto de datos que dio inicio a la investigación.

Estas dos situaciones, sumadas a la necesidad de ahondar en el proceso de selección de atributos, con el objetivo de proveer el subconjunto mas apto de características para encarar modelos predictivos, hacen que se decida realizar un nuevo intento de utilización de algoritmo de selección, esta vez con todos los datos disponibles.

La siguiente sección describe la implementación del algoritmo de selección de características basado en proyecciones que se describió en 2.5.

5.8. Aplicación del Algoritmo SOAP

Una vez tomada la decisión de profundizar la evaluación de los atributos relevantes, y estudiado el criterio de selección propuesto en [Ruiz Sanchez, 2006], se debe iniciar la implementación para los datos del caso de estudio. Es importante en este punto determinar con qué vista minable se debe trabajar para que los resultados sean comparables o mejoren las selecciones de atributos realizadas en la sección 5.1.

La aplicación de los algoritmos de selección en la etapa anterior del análisis se realizó sobre el conjunto de datos de los alumnos desertores de la UNRN inscriptos hasta el año 2011 inclusive, con la categorización obtenida del primer agrupamiento realizado, que determinaba 5 grupos o clusters.

En esta instancia de la investigación se cuenta con los datos completos de los alumnos inscriptos hasta el año 2012 inclusive, actualizados a abril/2013, de manera que se espera poder utilizar estos datos para las pruebas con el algoritmo SOAP. Para mantener la coherencia, se utiliza el mismo agrupamiento en 5 clusters.

De esta manera, se implementa SOAP sobre un conjunto de datos de 6562 registros con los mismos atributos del Apéndice A.

A modo de ejemplo se realiza el seguimiento del algoritmo implementado en forma detallada para uno de los atributos: `anio_nacim`. El primer paso es ordenar el conjunto de registros por `anio_nacim`, luego se identifican las subsecuencias encontradas $S_1, ..S_m$, clasificándolas en SOS o SOM según corresponda. Para las SOS se muestra cada valor del atributo involucrado $v_S =< v_1, ..., v_n >$ (donde S es el número de secuencia). En el caso de las SOM, y dado la gran cardinalidad de las mismas, solo se muestra el valor del atributo y el número total de ejemplos (te) que la componen. De esta manera, para el conjunto de ejemplos analizado, para la secuencia $\sigma_{anio\_nacim}(E)$, las subsecuencias ordenadas que lo componen son las siguientes:

$$S_1 =< e_1, e_2, e_3 > \text{es una } SOS$$
$$v_1 =< 1925, 1932, 1934 >$$
$$lab(S_1) =< cluster4, cluster4, cluster4 >$$
$$nch(S_1) = 1$$
(difiere la última etiqueta de $ml(S)$ de la SOM siguiente)

$$S_2 =< e_4, ...e_6 > \text{es una } SOM$$
$$te = 3$$
$$v_2 =< 1940 >$$
$$ml(S_2) =< cluster1 >$$
$$nch(S_2) = 2$$
(no hay una clase mayoritaria)

$$S_3 =< e_7 > \text{es una } SOS$$
$$v_3 =< 1941 >$$
$$lab(S_3) =< cluster4 >$$
$$nch(S_3) = 0$$
(no hay cambios internos ni con $ml(S)$ de la próxima SOM)

$$S_4 =< e_8, .., e_{11} > \text{es una } SOM$$
$$te = 4$$
$$v_4 =< 1942 >$$
$$ml(S_4) =< cluster4 >$$
$$nch(S4) = 0$$
(todos los ejemplos la misma etiqueta)

$$S_5 =< e12, e13 > \text{es una } SOM$$
$$te = 2$$
$$v_5 =< 1943 >$$
$$ml(S_5) =< cluster3 >$$
$$nch(S_5) = 2$$
(no hay una clase mayoritaria)

$$S_6 =< e_{14}, .., e_{21} > \text{es una } SOM$$
$$te = 8$$
$$v_6 =< 1944 >$$
$$ml(S_6) =< cluster4 >$$
$$nch(S4) = 2$$
(hay una clase mayoritaria con 7 elementos)

Los casos analizados cubren todas las situaciones posibles del algoritmo. De allí en adelante se encuentra una sucesión de SOM, en total 50, que reflejan casos comprendidos en los analizados, los resultados de cada una de ellas pueden verse en las tablas C.1 y C.2 del Apéndice C. Luego se cierra la secuencia con un par de SOS de un solo elemento, ambas con clase idéntica de manera que sus $nch = 0$. La suma de los *nch* para `anio_nacim` arroja un NLC(`anio_nacim`) = 4063.

En contraposición a la gran cantidad de subsecuencias generadas para el atributo `anio_nacim`, se pueden encontrar otros atributos con pocas subsecuencias, como el caso de `cant_fami_cargo` (ver tabla C.9 en el Apéndice C). En este caso solo se encuentran 4 subsecuencias, todas SOM, generando un NLC(`cant_fami_cargo`) = 5851.

Las tablas del Apéndice C muestran algunos de los atributos analizados y sus valores de NLC. Las columnas de las tablas reflejan los cálculos intermedios que realiza el algoritmo SOAP, de la siguiente manera:

Sec. Es el número de subsecuencia dentro de $\sigma_i(E)$ para cada atributo X_i.

Valor. Es el valor del atributo $\pi_i(e)$.

Tipo Sec. Indica si la subsecuencia es una SOM o una SOS. En el caso de las SOM, la subsecuencia ocupa en la tabla una sola fila. En el caso de las SOS, los diferentes $\pi_i(e)$ que conforman la subsecuencia aparecen en las filas sucesivas.

Cant. Para una SOM es el número de ejemplos que la componen. Para una SOS, es 1 para cada valor de atributo involucrado.

Ch. Solo se aplica a los valores de subsecuencias SOS, indica el resultado de la función $ch(E)$ para el ejemplo de la fila.

Nch. Es el resultado de la función *nch* para la subsecuencia.

Ml(S)/clase. Es el valor de la función $ml(S)$ para las SOM (moda), y el valor que toma la clase para el ejemplo de la fila correspondiente en las SOS.

Los resultados de la aplicación del algoritmo SOAP a los datos de los alumnos que abandonaron, segmentados en cinco grupos, arrojan un ranking de atributos que se presenta en la tabla 5.5, ordenado por el valor de NLC.

5.8.1. Interpretación de los atributos rankeados con SOAP

Si se observan los atributos que aparecen en el tope del ranking generado por SOAP, se puede ver que predominan los atributos relacionados al trabajo del alumno, junto con los atributos que determinan la edad y las cargas familiares. Estos atributos en general han sido parte integrante de las listas de atributos que se obtuvieron con los primeros algoritmos de selección de características utilizados en este trabajo. Basta con revisar las tablas 5.1 y 5.3 que listan los atributos seleccionados por un método wrapper y uno genético respectivamente.

Por otro lado se puede apreciar que las caracterizaciones de los perfiles de los alumnos desertores y los no desertores obtenidas en la sección 5.5, también utilizan como descriptores de los grupos obtenidos a los atributos de edad, carga laboral y familiar de los alumnos.

Resulta evidente en la interpretación de los resultados obtenidos, que aquellos atributos tienen una fuerte influencia en el porcentaje de deserción de los alumnos de las cohortes de la UNRN.

Dada la cantidad de ejemplos con los que se cuenta para la investigación (hecho inapelable por la corta vida de la UNRN) y el gran número de atributos con los que se iniciaron las tareas, se puede inferir en este punto que, por el momento, y hasta tanto se pueda contar con mas ejemplos para alimentar otros algoritmos, se debe aceptar este grupo de atributos personales y laborales como los que describen a los alumnos desertores y no desertores. Con esa idea en mente, es posible hacer una aplicación de un algoritmo de clasificación a la totalidad de los ejemplos disponibles, utilizando un grupo de atributos de los más altos en el ranking del SOAP para intentar clasificar a los alumnos en desertores y no desertores. La pròxima sección abordará esta implementación.

5.9. Arbol de decisión para determinar deserción

Como ya se ha expresado en la sección 3.5.5, los árboles de decisión son muy utilizados para tareas predictivas debido a su precisión, eficiencia y claridad. Estos simples hechos los posicionan como los primeros candidatos a utilizar para el caso de estudio.

El objetivo es, claramente, encontrar un modelo de clasificación que funcione dentro de parámetros aceptables para los datos disponibles a la fecha y que pueda ser utilizado en el futuro para clasificar nuevos alumnos de la UNRN. La utilización de árboles de decisión permitirá cumplir con el objetivo de predicción del abandono y además, por las características ya analizadas, ofrecerá también patrones útiles para tipificar ambas clases, la de los alumnos desertores y aquella conformada por los alumnos regulares.

NLC	atributo
2186	alu_trab_remmon
2186	Alu_trab_sitimp
2252	rel_trab_carrera
3044	alu_trab_futtip
3571	alu_trab_futhor
3987	anio_egreso_sec
4004	hora_sem_trab_alum
4063	anio_nacim
5070	alu_otestsup_uni
5394	Sit_laboral_madre
5734	cant_hijos_alum
5747	Sit_laboral_padre
5786	alu_beca
5790	sede
5851	cant_fami_cargo
5935	ult_est_cur_madre
6008	alu_ingre_grupfa
6013	estado_civil
6019	lugar_nacimiento
6178	ult_est_cur_padre
6207	padres_prop_viv
6300	prom_cnt_desaprobo
6304	tipo_res_per_lect
6315	prom_notas_finales
6348	alu_tec_pc
6352	madre_vive
6356	alu_tec_int
6380	prom_cnt_abandono
6401	titulo_secundario
6411	prom_cnt_aprobo
6454	loc_perlect_distinta_loc_proc
6484	colegio_secundario
6487	prom_cnt_fin_aprobo
6502	prom_cnt_fin_ausente
6503	alu_idioma_ingl
6504	prom_cnt_promociono
6517	prom_cnt_fin_desaprobo
6523	sexo
6530	padre_vive
6544	nacionalidad

Tabla 5.5: Ranking de atributos generados con algoritmo SOAP

NLC	atributo
2186	alu_trab_remmon
2186	Alu_trab_sitimp
2252	rel_trab_carrera
3044	alu_trab_futtip
3571	alu_trab_futhor
4004	hora_sem_trab_alum
4063	anio_nacim
5070	alu_otestsup_uni

Tabla 5.6: Atributos ubicados en el tope del ranking obtenido a través del método SOAP. La información referida al año de egreso del colegio secundario no fue considerada ya que se encuentra fuertemente correlacionada con el año de nacimiento.

Precisión: 68.84 %			
	true Abandono	true Cursa	class precision
pred.Abandono	5174	2047	71.65 %
pred.Cursa	1388	2414	63.49 %
class recall	78.85 %	54.11 %	

Tabla 5.7: Performance del árbol de decisión

5.10. Un Arbol de decisión para el caso de estudio

Tomada la decisión de utilizar la técnica de los árboles de decisión, se realizan pruebas con el algoritmo C4.5 [Quinlan, 1993] implementado por el operador `W-J48` de la extensión Weka de RapidMiner.

Se utilizan los nueve primeros atributos del ranking SOAP (tabla 5.5), exceptuando `anio_egreso_sec`, dado que el atributo `anio_nacim` ya está incluido y que ambos atributos están estrechamente correlacionados. De esta forma los atributos elegidos se muestran en la tabla 5.6.

Se realizan varias pruebas con diferentes parámetros, hasta quedarse con la corrida con C (umbral de confianza o *confidence threshold*) del 35 %, M (mínimo número de instancias por hoja) = 25, que produce un árbol podado como el que se puede apreciar en la figura 5.3. La tabla 5.7 muestra la performance del modelo obtenido.

Puede observarse que la lista de atributos utilizada tiene una longitud equivalente al 56.25 % de la lista de atributos de la tabla 5.4 (9/16 = .5625). Los atributos seleccionados permiten construir un modelo capaz de predecir correctamente el 71.65 % de los casos de abandono. La tasa de acierto es menor en el caso de tener que predecir si el alumno sigue cursando.

```
anio_nacim <= 1989: Abandono (7614.0/2538.0)
anio_nacim > 1989
|   anio_nacim <= 1992
|   |   alu_otestsup_uni = S
|   |   |   Alu_trab_sitimp = Relación de dependencia: Cursa (65.0/21.0)
|   |   |   Alu_trab_sitimp = no trabaja: Cursa (257.0/90.0)
|   |   |   Alu_trab_sitimp = Monotributista: Abandono (11.0/4.0)
|   |   alu_otestsup_uni = N
|   |   |   Alu_trab_futtip = Obrero o empleado (asalariado)
|   |   |   |   Alu_trab_futhor = Más de 10 y hasta 20 horas: Cursa (93.0/42.0)
|   |   |   |   Alu_trab_futhor = No trabajaré: Cursa (7.0/2.0)
|   |   |   |   Alu_trab_futhor = 35 o más horas
|   |   |   |   |   Hora_sem_trab_alum = No informado: Abandono (33.0/13.0)
|   |   |   |   |   Hora_sem_trab_alum = de 21 a 35 hs.: Abandono (0.0)
|   |   |   |   |   Hora_sem_trab_alum = hasta 20 hs.: Cursa (4.0/1.0)
|   |   |   |   |   Hora_sem_trab_alum = de 36 o más hs.: Cursa (1.0)
|   |   |   |   |   Hora_sem_trab_alum = no trabaja: Abandono (48.0/17.0)
|   |   |   |   Alu_trab_futhor = Hasta 10 horas: Abandono (128.0/51.0)
|   |   |   |   Alu_trab_futhor = Más de 20 y menos de 35 horas
|   |   |   |   |   Hora_sem_trab_alum = No informado: Abandono (25.0/10.0)
|   |   |   |   |   Hora_sem_trab_alum = de 21 a 35 hs.: Cursa (3.0/1.0)
|   |   |   |   |   Hora_sem_trab_alum = hasta 20 hs.: Abandono (2.0/1.0)
|   |   |   |   |   Hora_sem_trab_alum = de 36 o más hs.: Abandono (36.0/15.0)
|   |   |   |   |   Hora_sem_trab_alum = no trabaja: Cursa (11.0/3.0)
|   |   |   Alu_trab_futtip = No trabajaré: Cursa (1358.0/613.0)
|   |   |   Alu_trab_futtip = Cuenta propia
|   |   |   |   Alu_trab_remmon = de 1200 a 2000$: Abandono (9.0/4.0)
|   |   |   |   Alu_trab_remmon = más de 3000$: Cursa (2.0)
|   |   |   |   Alu_trab_remmon = no trabaja: Abandono (35.0/16.0)
|   |   |   |   Alu_trab_remmon = hasta 1200$: Cursa (33.0/14.0)
|   |   |   |   Alu_trab_remmon = de 2000 a 3000$: Abandono (2.0/1.0)
|   |   |   Alu_trab_futtip = Desconoce
|   |   |   |   Alu_trab_futhor = Más de 10 y hasta 20 horas: Abandono (81.0/28.0)
|   |   |   |   Alu_trab_futhor = No trabajaré: Abandono (107.0/52.0)
|   |   |   |   Alu_trab_futhor = 35 o más horas: Abandono (6.0/3.0)
|   |   |   |   Alu_trab_futhor = Hasta 10 horas: Cursa (138.0/66.0)
|   |   |   |   Alu_trab_futhor = Más de 20 y menos de 35 horas: Abandono (29.0/9.0)
|   anio_nacim > 1992
|   |   Hora_sem_trab_alum = No informado: Cursa (827.0/278.0)
|   |   Hora_sem_trab_alum = de 21 a 35 hs.: Cursa (13.0/5.0)
|   |   Hora_sem_trab_alum = hasta 20 hs.: Cursa (26.0/9.0)
|   |   Hora_sem_trab_alum = de 36 o más hs.: Abandono (10.0/4.0)
|   |   Hora_sem_trab_alum = no trabaja: Abandono (9.0/2.0)
Number of Leaves  : 33
Size of the tree : 44
```

Figura 5.3: Arbol de decisión para predecir Abandono

Capítulo 6

Conclusiones

Se inició este trabajo con el objetivo de estudiar las técnicas de $\mathcal{DM}$ y su aplicabilidad al análisis de la deserción de alumnos universitarios de la UNRN. Para ello se realizó una investigación bibliográfica sobre la metodología del proceso de extracción de conocimiento en grandes bases de datos, seleccionando algunas técnicas y algoritmos para abordar el problema. Luego se prepararon los datos disponibles para la aplicación de los algoritmos seleccionados.

Las pruebas preliminares dejaron en claro la naturaleza multidimensional del problema, orientando la investigación hacia los métodos de selección de características. De esta forma se probaron métodos reconocidos de selección de atributos y se avanzó en esa línea de trabajo presentando la aplicación del método de selección de características, SOAP, basado en proyecciones. Este método es capaz de operar sobre atributos nominales y numéricos de manera supervisada.

A partir del ranking establecido entre los atributos fue posible determinar un punto de corte para identificar los más representativos. En este caso, su aplicación permitió reducir la lista original obtenida con los métodos tradicionales de selección en más de un 40 %.

A lo largo del proceso se pudo apreciar que la selección de atributos permite mejorar la precisión e interpretabilidad de los métodos de aprendizaje automático, además de reducir el tamaño del conjunto de datos de entrada y el tiempo de los algoritmos de aprendizaje. También se pudo constatar que la selección de características no es un problema que se pueda enfocar desde un único marco de trabajo, y que diferentes algoritmos pueden ser aplicables y aportar aproximaciones a la solución buscada.

Para el caso analizado se hizo evidente que los atributos más relevantes de los estudiantes de la UNRN son los relacionados con la situación laboral del alumno tanto en lo que se refiere a su trabajo actual como a sus intenciones de trabajar en el futuro.

A partir de esta investigación se pueden obtener orientaciones útiles para guiar las acciones a tomar a favor de la disminución de la deserción en la UNRN: es claro que las variables laborales de los alumnos tienen marcada influencia en su posibilidad de permanencia en los claustros, de manera que acciones directas sobre esta realidad, como el aumento de las becas otorgadas, podría brindar un camino a seguir.

Finalmente, es importante remarcar que se ha dejado planteado un modelo predictivo que puede ser mejorado a lo largo del tiempo con la incorporación de más ejemplos al conjunto de datos.

Apéndice A

Atributos Vista Minable

Nro.	Atributo	Tipo	Estadísticas	Min	Max
1	Nacionalidad = Argentino	integer	avg = 0.952 +/- 0.305	-1	1
2	estado_civil = Soltero	integer	avg = 0.408 +/- 0.913	-1	1
3	Padres_prop_viv = Propia	integer	avg = 0.397 +/- 0.918	-1	1
4	sexo = M	integer	avg = -0.247 +/- 0.969	-1	1
5	Loc_perlect_distinta_loc_proc = S	integer	avg = -0.781 +/- 0.625	-1	1
6	Padre_vive = S	integer	avg = 0.467 +/- 0.884	-1	1
7	madre_vive = S	integer	avg = 0.725 +/- 0.689	-1	1
8	alu_otestsup_uni = S	integer	avg = 0.197 +/- 0.981	-1	1
9	Alu_tec_pc = tiene pc en la casa	integer	avg = 0.591 +/- 0.807	-1	1
10	Alu_tec_int = tiene internet en la casa	integer	avg = 0.235 +/- 0.972	-1	1
11	Alu_beca = necesita beca	integer	avg = -0.059 +/- 0.998	-1	1
12	Hora_sem_trab_alum	integer	avg = 0.485 +/- 1.011	0	3
13	Rel_trab_carrera	integer	avg = 1.473 +/- 1.244	0	3
14	Ult_est_cur_padre	integer	avg = 3.747 +/- 2.029	0	7
15	Ult_est_cur_madre	integer	avg = 4.119 +/- 1.947	0	7
16	Alu_trab_remmon	integer	avg = 1.561 +/- 1.408	0	4
17	Alu_trab_futhor	integer	avg = 2.210 +/- 1.524	0	4
18	Alu_ingre_grupfa	integer	avg = 0.794 +/- 1.121	0	3
19	sede = Andina	integer	avg = 0.421 +/- 0.494	0	1
20	sede = Atlántica	integer	avg = 0.269 +/- 0.444	0	1
21	sede = Valle Medio y Río Colorado	integer	avg = 0.037 +/- 0.188	0	1
22	sede = Alto Valle y Valle Medio	integer	avg = 0.223 +/- 0.416	0	1
23	sede = Rectorado	integer	avg = 0.040 +/- 0.196	0	1

Tabla A.1: Lista completa de atributos - Parte I

Nro.	Atributo	Tipo	Estadísticas	Min	Max
24	sede = Sede Unica	integer	avg = 0.001 +/- 0.033	0	1
25	sede = San Antonio Oeste	integer	avg = 0.009 +/- 0.096	0	1
26	Lugar_nacimiento = OTROS FUERA RIO NEGRO	integer	avg = 0.255 +/- 0.436	0	1
27	Lugar_nacimiento = BARILOCHE	integer	avg = 0.164 +/- 0.370	0	1
28	Lugar_nacimiento = INDETERMINADA	integer	avg = 0.075 +/- 0.264	0	1
29	Lugar_nacimiento = CARMEN DE PATAGONES	integer	avg = 0.018 +/- 0.134	0	1
30	Lugar_nacimiento = CIUDAD AUTONOMA BS. AS.	integer	avg = 0.084 +/- 0.277	0	1
31	Lugar_nacimiento = ADOLFO ALSINA	integer	avg = 0.087 +/- 0.282	0	1
32	Lugar_nacimiento = GENERAL ROCA	integer	avg = 0.163 +/- 0.369	0	1
33	Lugar_nacimiento = OTROS DPTOS RIO NEGRO	integer	avg = 0.096 +/- 0.294	0	1
34	Lugar_nacimiento = BAHIA BLANCA	integer	avg = 0.029 +/- 0.167	0	1
35	Lugar_nacimiento = NEUQUEN	integer	avg = 0.029 +/- 0.169	0	1
36	Colegio_secundario = E	integer	avg = 0.589 +/- 0.492	0	1
37	Colegio_secundario = P	integer	avg = 0.218 +/- 0.413	0	1
38	Colegio_secundario = N	integer	avg = 0.193 +/- 0.395	0	1
39	Titulo_secundario = BACHILLER	integer	avg = 0.700 +/- 0.458	0	1
40	Titulo_secundario = TECNICO	integer	avg = 0.067 +/- 0.250	0	1
41	Titulo_secundario = PER.MERC.	integer	avg = 0.142 +/- 0.350	0	1
42	Titulo_secundario = OTROS	integer	avg = 0.090 +/- 0.286	0	1
43	Tipo_res_per_lect = con familiares	integer	avg = 0.767 +/- 0.423	0	1
44	Tipo_res_per_lect = forma independiente	integer	avg = 0.182 +/- 0.386	0	1
45	Tipo_res_per_lect = residencia universitaria	integer	avg = 0.014 +/- 0.117	0	1
46	Tipo_res_per_lect = En otra situación	integer	avg = 0.037 +/- 0.189	0	1
47	Sit_laboral_padre = Ocupado	integer	avg = 0.524 +/- 0.499	0	1
48	Sit_laboral_padre = DesOcupado	integer	avg = 0.302 +/- 0.459	0	1
49	Sit_laboral_padre = Jubilado	integer	avg = 0.141 +/- 0.348	0	1
50	Sit_laboral_padre = No informado	integer	avg = 0.033 +/- 0.179	0	1
51	Sit_laboral_madre = Ocupado	integer	avg = 0.433 +/- 0.495	0	1
52	Sit_laboral_madre = DesOcupado	integer	avg = 0.458 +/- 0.498	0	1
53	Sit_laboral_madre = Jubilado	integer	avg = 0.065 +/- 0.247	0	1
54	Sit_laboral_madre = SubOcupado	integer	avg = 0.016 +/- 0.124	0	1
55	Sit_laboral_madre = No informado	integer	avg = 0.028 +/- 0.165	0	1
56	Alu_trab_sitimp = Relación de dependencia	integer	avg = 0.566 +/- 0.496	0	1
57	Alu_trab_sitimp = no trabaja	integer	avg = 0.338 +/- 0.473	0	1
58	Alu_trab_sitimp = Monotributista	integer	avg = 0.096 +/- 0.294	0	1

Tabla A.2: Lista completa de atributos - Parte II

Nro.	Atributo	Tipo	Estadísticas	Min	Max
59	Alu_trab_futtip = No trabajaré	integer	avg = 0.183 +/- 0.386	0	1
60	Alu_trab_futtip = Obrero o empleado (asalariado)	integer	avg = 0.585 +/- 0.493	0	1
61	Alu_trab_futtip = Desconoce	integer	avg = 0.124 +/- 0.330	0	1
62	Alu_trab_futtip = Cuenta propia	integer	avg = 0.108 +/- 0.311	0	1
63	anio_nacim	integer	avg = 1978.502 +/- 10.321	1925	1994
64	Anio_egreso_sec	integer	avg = 1825.690 +/- 560.471	0	2011
65	Cant_fami_cargo	integer	avg = 0.666 +/- 1.011	0	3
66	Cant_hijos_alum	integer	avg = 0.787 +/- 1.058	0	3
67	Alu_otestsup_egre	integer	avg = 628.629 +/- 928.666	0	2011
68	Alu_idioma_ingl	integer	avg = 2.786 +/- 0.771	1	4
69	prom_cnt_abandono	real	avg = 0.857 +/- 1.675	0	12
70	prom_cnt_desaprobo	real	avg = 0.318 +/- 0.727	0	5
71	prom_cnt_aprobo	real	avg = 0.580 +/- 1.224	0	10
72	prom_cnt_promociono	real	avg = 0.182 +/- 0.728	0	6
73	prom_cnt_fin_ausente	real	avg = 0.212 +/- 0.761	0	10
74	prom_cnt_fin_desaprobo	real	avg = 0.062 +/- 0.326	0	6
75	prom_cnt_fin_aprobo	real	avg = 0.190 +/- 0.637	0	9
76	prom_notas_finales	real	avg = 0.853 +/- 2.261	0	10

Tabla A.3: Lista completa de atributos - Parte III

Apéndice B

RapidMiner

RapidMiner es una solución completa de Inteligencia Empresarial que hace foco en la minería de datos y análisis predictivo. Utiliza una amplia variedad de técnicas descriptivas y predictivas para ayudar en la toma de decisiones. Se distribuye bajo licencia de código abierto: AGPL (Affero General Public License o AGPL) que es una licencia derivada de la Licencia Pública General de GNU. La primera versión creada por la Universidad de Dortmund (Alemania) apareció en 2001, está desarrollado en Java.

El producto está disponible en la versión libre RapidMiner Community Edition (utilizada en este trabajo), que puede ser descargada desde el sitio web de Rapid-I (http://www.rapid-i.com) de forma gratuita, y la RapidMiner Enterprise Edition, que combina las ventajas de la Community Edition con el soporte profesional con garantía de tiempo de respuesta ofrecido por la empresa.

Para utilizar la versión libre sólo es necesario descargar el paquete de instalación apropiado para el sistema operativo del que se dispone, e instalar de acuerdo a las instrucciones provistas en el sitio web. Se soportan todas las versiones de Windows, Macintosh, Linux y Unix. También es necesaria una maquina virtual java actualizada.

La interfaz gráfica de usuario de RapidMiner permite el diseño de procesos analíticos auto-documentados que pueden actualizarse y re-utilizarse fácilmente para nuevos problemas. Provee gran cantidad de métodos de integración y transformación de datos, selección de atributos, análisis y modelado, con herramientas para visualización de los resultados. La Figura B.1 presenta la interfaz gráfica de RapidMiner con el proyecto de clustering de alumnos que abandonaron, desarrollado en el presente trabajo.

La herramienta dispone de un amplio número de extensiones que pueden instalarse, entre ellas se destaca Weka, código abierto con licencia GNU, que ofrece una colección de algoritmos de aprendizaje para tareas de minería de datos. Otras extensiones incluyen Text (para análisis estadístico de textos), Web Mining (para análisis de páginas web), R-connector (integración con el lenguaje R de programación de análisis estadístico).

RapidMiner provee acceso a buena cantidad de tipos de archivos y bases de datos (Microsoft

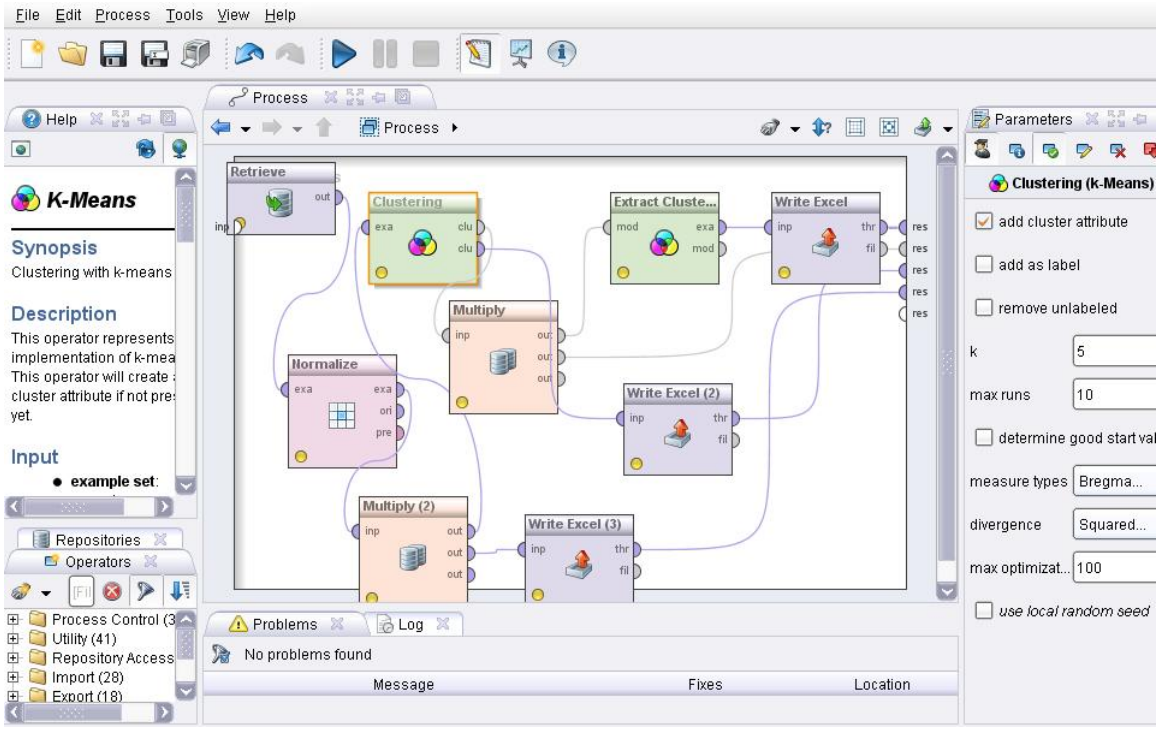

Figura B.1: Interfaz gráfica de RapidMiner

Excel, Microsoft Access, Oracle, IBM DB2, Microsoft SQL Server, MySQL, Postgres, Teradata, Ingres,VectorWise, SAP, paginas web, pdf, html, xml, etc.).

También ofrece más de 500 operadores para una amplia cantidad de tareas de minería de datos y otras características relacionadas como entrada, salida y procesamiento de datos. Entre ellas:

- Muestreo de datos
- Particionamiento de conjuntos de datos
- Transformaciones de datos
- Selección de atributos
- Generación de atributos
- Estadísticas descriptivas
- Gráficos y visualización
- Agrupamiento
- Reglas de asociación
- Arboles de decisión

- Reglas de inducción
- Modelos Bayesianos
- Regresión
- Redes Neuronales
- Máquinas de soporte vectorial
- Combinación de modelos
- Evaluación de modelos

Apéndice C

Ejemplos de Implementación del Algoritmo SOAP al caso de estudio

En este Apéndice se muestran los cálculos detallados para la obtención del NLC para algunos atributos del caso de estudio (anio_nacim, prom_cnt_aprobo, prom_notas_finales, cant_fami_cargo). A continuación se pueden ver las tablas para cada uno de los atributos enumerados.

Sec	valor	Tipo Sec	cant	ch	nch	ml(S) /clase
1	1925	SOS		0		cluster_4
	1932			0		cluster_4
	1934		3	0	1	cluster_4
2	1940	SOM	3		2	cluster_1
3	1941	SOS	1	0	0	cluster_4
4	1942	SOM	4		0	cluster_4
5	1943	SOM	2		1	cluster_3
6	1944	SOM	8		2	cluster_4
7	1945	SOM	5		0	cluster_4
8	1946	SOM	3		0	cluster_4
9	1947	SOM	6		5	cluster_1
10	1948	SOM	10		9	cluster_1
11	1949	SOM	4		3	cluster_4
12	1950	SOM	7		6	cluster_1
13	1951	SOM	14		13	cluster_4
14	1952	SOM	19		18	cluster_4
15	1953	SOM	17		14	cluster_1
16	1954	SOM	14		12	cluster_1
17	1955	SOM	25		22	cluster_1
18	1956	SOM	30		20	cluster_1
19	1957	SOM	38		26	cluster_1
20	1958	SOM	43		34	cluster_1
21	1959	SOM	48		36	cluster_1
22	1960	SOM	55		46	cluster_1
23	1961	SOM	56		28	cluster_1
24	1962	SOM	49		22	cluster_1
25	1963	SOM	70		44	cluster_1
26	1964	SOM	73		42	cluster_1
27	1965	SOM	70		36	cluster_1
28	1966	SOM	79		52	cluster_1
29	1967	SOM	87		46	cluster_1
30	1968	SOM	67		32	cluster_1
31	1969	SOM	81		46	cluster_1
32	1970	SOM	121		70	cluster_1
33	1971	SOM	96		46	cluster_1
34	1972	SOM	109		52	cluster_1
35	1973	SOM	119		68	cluster_1

Tabla C.1: Cálculo NLC atributo anio_nacim - Parte I

Sec	valor	Tipo Sec	cant	ch	nch	ml(S) /clase
36	1974	SOM	143		90	cluster_1
37	1975	SOM	145		76	cluster_1
38	1976	SOM	195		104	cluster_1
39	1977	SOM	215		110	cluster_1
40	1978	SOM	197		112	cluster_1
41	1979	SOM	205		142	cluster_1
42	1980	SOM	229		158	cluster_1
43	1981	SOM	213		124	cluster_1
44	1982	SOM	214		114	cluster_1
45	1983	SOM	201		144	cluster_1
46	1984	SOM	220		176	cluster_1
47	1985	SOM	238		200	cluster_1
48	1986	SOM	270		269	cluster_1
49	1987	SOM	301		300	cluster_3
50	1988	SOM	315		256	cluster_3
51	1989	SOM	339		228	cluster_3
52	1990	SOM	389		208	cluster_3
53	1991	SOM	433		168	cluster_3
54	1992	SOM	359		138	cluster_3
55	1993	SOM	226		64	cluster_3
56	1994	SOM	77		28	cluster_3
57	1995	SOS		0		cluster_4
	1996		2	0	0	cluster_4
NLC					**4063**	

Tabla C.2: Cálculo NLC atributo anio_nacim - Parte II

Sec	valor	Tipo Sec	cant	ch	nch	ml(S) /clase
1	0	SOM	4446		4358	cluster_1
2	0.3333	SOM	2		1	cluster_1
3	0.5	SOM	44		43	cluster_3
4	0.6667	SOM	11		10	cluster_1
5	1	SOM	591		590	cluster_1
6	1.3333	SOM	20		18	cluster_1
7	1.5	SOM	93		92	cluster_1
8	1.6667	SOM	16		15	cluster_1
9	2	SOM	571		570	cluster_1
10	2.3333	SOM	18		16	cluster_1
11	2.5	SOM	58		57	cluster_1
12	2.6667	SOM	13		12	cluster_1
13	3	SOM	279		278	cluster_3
14	3.3333	SOM	10		9	cluster_3
15	3.5	SOM	33		32	cluster_1
16	3.6667	SOM	15		10	cluster_1
17	4	SOM	136		122	cluster_1
18	4.3333	SOM	5		4	cluster_1
19	4.5	SOM	26		25	cluster_1
20	4.6667	SOM	3		2	cluster_1
21	5	SOM	84		83	cluster_1
22	5.3333	SOS	1	0	1	cluster_1
23	5.5	SOM	5		2	cluster_3
24	5.6667	SOM	2		1	cluster_1
25	6	SOM	42		41	cluster_1
26	6.3333	SOS	1	0	0	cluster_1
27	6.5	SOM	2		1	cluster_1
28	6.6667	SOM	2		0	cluster_1
29	7	SOM	18		10	cluster_3
30	7.6667	SOS	1	0	0	cluster_3
31	8	SOM	10		6	cluster_3
32	8.5	SOS	1	0	1	cluster_2
33	10	SOM	2		1	cluster_0
34	11	SOS	1	0	0	cluster_3
NLC					**6411**	

Tabla C.3: Cálculo NLC atributo `prom_cnt_aprobo`

Sec	valor	Tipo Sec	cant	ch	nch	ml(S) /clase
1	0	SOM	5261		5254	cluster_1
2	1	SOM	14		12	cluster_3
3	1.5	SOM	5		4	cluster_3
4	1.6	SOS	1	0	1	cluster_3
5	1.66666667	SOM	2		1	cluster_0
6	2	SOM	80		70	cluster_3
7	2.33333333	SOS	1	0	0	cluster_3
8	2.5	SOM	16		15	cluster_3
9	2.53846154	SOS	1	0	0	cluster_3
10	2.66666667	SOM	5		2	cluster_3
11	2.76923077	SOS		1		cluster_3
	2.83333333		2	0	1	cluster_1
12	3	SOM	29		28	cluster_1
13	3.11111111	SOS		0		cluster_3
	3.125			1		cluster_3
	3.16666667		3	0	2	cluster_2
14	3.25	SOM	3		0	cluster_3
15	3.2631579	SOS	1	0	1	cluster_1
16	3.33333333	SOM	5		2	cluster_3
17	3.44444444	SOS	1	0	0	cluster_3
18	3.5	SOM	15		14	cluster_3
19	3.6	SOM	3		0	cluster_3
20	3.66666667	SOM	8		7	cluster_1
21	3.72727273	SOS	1	0	1	cluster_3
22	3.75	SOM	3		2	cluster_1
23	3.77777778	SOS		1		cluster_0
	3.83333333			1		cluster_3
	3.84615385		3	0	2	cluster_2
24	3.85714286	SOM	2		1	cluster_2
25	3.875	SOS	1	0	1	cluster_1
26	4	SOM	86		70	cluster_3
27	4.1	SOS	1	0	1	cluster_3
28	4.125	SOM	2		0	cluster_1
29	4.2	SOM	2		1	cluster_1

Tabla C.4: Cálculo NLC atributo `prom_finales_aprobados` - Parte I

Sec	valor	Tipo Sec	cant	ch	nch	ml(S) /clase
30	4.25	SOM	4		2	cluster_3
31	4.3	SOS	1	0	1	cluster_3
32	4.33333333	SOM	13		12	cluster_1
33	4.4	SOM	3		2	cluster_3
34	4.5	SOM	49		48	cluster_3
35	4.52941177	SOS		0		cluster_1
	4.57142857			0		cluster_1
	4.58333333		3	0	0	cluster_1
36	4.6	SOM	3		2	cluster_1
37	4.625	SOS	1	0	0	cluster_1
38	4.66666667	SOM	16		14	cluster_1
39	4.71428571	SOS	1	0	1	cluster_3
40	4.75	SOM	4		3	cluster_1
41	4.8	SOS	1	0	1	cluster_3
42	4.83333333	SOM	3		2	cluster_4
43	4.85714286	SOS		0		cluster_3
	4.91666667			0		cluster_3
	4.93333333		3	0	0	cluster_3
44	5	SOM	91		90	cluster_3
45	5.1	SOS	1	1		cluster_2
46	5.16666667	SOS	1	0	1	cluster_3
47	5.2	SOM	3		2	cluster_3
48	5.25	SOM	3		2	cluster_1
49	5.28571429	SOS	1	0	0	cluster_3
50	5.33333333	SOM	6		4	cluster_3
51	5.4	SOS		0		cluster_1
	5.42857143			1		cluster_1
	5.44444444		3	0	1	cluster_3
52	5.5	SOM	54		50	cluster_3
53	5.6	SOM	2		1	cluster_1
54	5.625	SOS	1	0	1	cluster_3
55	5.66666667	SOM	9		8	cluster_1
56	5.71428571	SOM	2		1	cluster_1
57	5.75	SOM	4		3	cluster_1

Tabla C.5: Cálculo NLC atributo `prom_finales_aprobados` - Parte II

Sec	valor	Tipo Sec	cant	ch	nch	ml(S) /clase
58	5.8	SOM	2		1	cluster_1
59	5.8125	SOS		1		cluster_1
	5.86666667		2	0	2	cluster_2
60	6	SOM	100		98	cluster_1
61	6.1	SOM	2		1	cluster_1
62	6.14285714	SOS	1	0	0	cluster_1
63	6.16666667	SOM	2		0	cluster_1
64	6.2	SOS	1	0	0	cluster_3
65	6.25	SOM	8		7	cluster_3
66	6.27777778	SOS		0		cluster_1
	6.28571429			1		cluster_1
	6.3		3	0	1	cluster_3
67	6.33333333	SOM	14		13	cluster_3
68	6.4	SOM	5		4	cluster_1
69	6.41666667	SOS		0		cluster_1
	6.42857143			1		cluster_1
	6.44444444		3	0	2	cluster_3
70	6.5	SOM	40		39	cluster_1
71	6.6	SOM	3		0	cluster_1
72	6.625	SOS		1		cluster_1
	6.63636364		2	0	2	cluster_3
73	6.66666667	SOM	15		10	cluster_1
74	6.7	SOS		0		cluster_1
	6.72727273		2	0	0	cluster_1
75	6.75	SOM	9		4	cluster_1
76	6.8	SOM	2		0	cluster_3
77	6.83333333	SOM	3		2	cluster_1
78	6.85714286	SOM	2		1	cluster_0
79	6.875	SOS	1	0	0	cluster_1
80	7	SOM	116		112	cluster_1
81	7.08333333	SOS		0		cluster_1
	7.09090909			0		cluster_1
	7.125		3	0	0	cluster_1
82	7.14285714	SOM	2		1	cluster_1

Tabla C.6: Cálculo NLC atributo `prom_finales_aprobados` - Parte III

Sec	valor	Tipo Sec	cant	ch	nch	ml(S) /clase
83	7.16666667	SOM	2		0	cluster_1
84	7.2	SOM	7		6	cluster_3
85	7.25	SOM	5		2	cluster_1
86	7.27272727	SOS	1	0	0	cluster_1
87	7.33333333	SOM	14		12	cluster_1
88	7.375	SOS	1	0	0	cluster_1
89	7.4	SOM	2		1	cluster_1
90	7.5	SOM	30		29	cluster_3
91	7.54545455	SOS		1		cluster_4
	7.57142857			0		cluster_1
	7.58333333		3	0	2	cluster_1
92	7.6	SOM	3		2	cluster_3
93	7.66666667	SOM	6		5	cluster_1
94	7.7	SOM	2		0	cluster_1
95	7.71428571	SOM	2		1	cluster_1
96	7.75	SOM	2		1	cluster_1
97	7.8	SOM	4		2	cluster_1
98	7.83333333	SOM	3		0	cluster_1
99	7.86956522	SOS	1	0	0	cluster_1
100	8	SOM	112		94	cluster_1
101	8.08333333	SOS		1		cluster_1
	8.125		2	0	2	cluster_4
102	8.14285714	SOM	2		1	cluster_1
103	8.16666667	SOS		1		cluster_3
	8.18181818			0		cluster_1
	8.2		3	0	1	cluster_1
104	8.25	SOM	3		0	cluster_1
105	8.28571429	SOM	2		1	cluster_2
106	8.33333333	SOM	5		2	cluster_1
107	8.36363636	SOS		1		cluster_1
	8.4			1		cluster_2
	8.41666667			0		cluster_1
	8.42105263			1		cluster_1
	8.44444444			1		cluster_3
	8.45454546		6	0	4	cluster_1

Tabla C.7: Cálculo NLC atributo `prom_finales_aprobados` - Parte IV

Sec	valor	Tipo Sec	cant	ch	nch	ml(S) /clase
108	8.5	SOM	26		24	cluster_1
109	8.53846154	SOS		0		cluster_1
	8.55555556		2	0	1	cluster_1
110	8.6	SOM	3		2	cluster_3
111	8.625	SOS	1	0	0	cluster_1
112	8.66666667	SOM	4		2	cluster_1
113	8.7	SOS	1	0	0	cluster_1
114	8.71428571	SOM	2		1	cluster_1
115	8.75	SOM	3		2	cluster_1
116	8.8	SOS	1	0	0	cluster_1
117	8.875	SOM	2		0	cluster_1
118	8.9	SOM	3		2	cluster_1
119	8.95652174	SOS	1	0	0	cluster_1
120	9	SOM	57		38	cluster_1
121	9.07692308	SOS	1	0	0	cluster_1
122	9.25	SOM	2		0	cluster_1
123	9.33333333	SOM	4		3	cluster_3
124	9.375	SOS	1	0	0	cluster_1
125	9.5	SOM	6		4	cluster_1
126	9.625	SOS	1	0	1	cluster_2
127	9.66666667	SOM	2		1	cluster_1
128	9.71428571	SOM	2		1	cluster_1
129	10	SOM	28		20	cluster_1
NLC					**6315**	

Tabla C.8: Cálculo NLC atributo `prom_finales_aprobados` - Parte V

Sec	valor	Tipo Sec	cant	ch	nch	ml(S) /clase
1	0	SOM	4149		4148	cluster_3
2	1	SOM	1006		682	cluster_1
3	2	SOM	792		486	cluster_1
4	3	SOM	615		418	cluster_1
NLC					**5851**	

Tabla C.9: Cálculo NLC atributo `cant_fami_cargo`

Indice de figuras

Lista de Algoritmos

Indice de tablas

Bibliografía

[Agrawal and Srikant, 1994] Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Alahakoon et al., 2000] Alahakoon, D., Halgamuge, S. K., and Srinivasan, B. (2000). Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, 11(3):601–614.

[Alcover et al., 2007] Alcover, R., Benlloch, J., Blesa, P., Calduch, M. A., Celma, M., Ferri, C., Hernández Orallo, J., Iniesta, L., Más, J., Ramírez Quintana, M. J., Robles, A., Valiente, J. M., Vicent, M. J., and Zúnica, L. R. (2007). Análisis del rendimiento académico en los estudios de informática de la universidad politécnica de valencia aplicando técnicas de minería de datos. Technical report, Universidad Politécnica de Valencia.

[Ball and Hall, 1965] Ball, G. and Hall, D. (1965). *Isodata: A Method of Data Analysis and Pattern Classification*. Stanford Research Institute.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

[Cao, 2010] Cao, L. (2010). Domain-driven data mining: Challenges and prospects. *Knowledge and Data Engineering, IEEE Transactions on*, 22(6):755 –769.

[Clark and Niblett, 1989] Clark, P. and Niblett, T. (1989). The cn2 induction algorithm. *Machine Learning*, 3:261–283.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

[Davis et al., 2007] Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 209–216, New York, NY, USA. ACM.

[Dzeroski and Lavrač, 2001] Dzeroski, S. and Lavrač, N. (2001). *Relational Data Mining*. Relational Data Mining. Springer.

[Fisher, 1987] Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.*, 2(2):139–172.

[Freedman, 2009] Freedman, D. A. (2009). *Statistical Models: Theory and Practice*. Cambridge University Press.

[Fritzke, 1994] Fritzke, B. (1994). Growing cell structures: A self organizing network for supervised and un-supervised learning. *Neural networks*, 7(9):1441–1460.

[Fritzke, 1995] Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems*, volume 7, pages 625–632. MIT Press.

[Gennari et al., 1990] Gennari, J. H., Langley, P., and Fisher, D. (1990). Models of incremental concept formation. *Artificial Intelligence*, 40:11–61.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[Hall, 1999] Hall, M. A. (1999). *Correlation-based Feature Selection for Machine Learning*. PhD thesis, University of Waikato, Hamilton, New Zealand.

[Hernández Orallo et al., 2004] Hernández Orallo, J., Ramírez Quintana, M., and Ferri Ramírez, C. (2004). *Introducción a la Minería de Datos*. Editorial Pearson.

[Hsu and Hsieh, 2010] Hsu, H.-H. and Hsieh, C.-W. (2010). Feature selection via correlation coefficient clustering. *JSW*, 5(12):1371–1377.

[I.X. and J.M., 1992] I.X., W. and J.M., M. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on System, Man and Cybernetics*, 22(6):1414–1427.

[Jirayusakul and Auwatanamongkol, 2007] Jirayusakul, A. and Auwatanamongkol, S. (2007). A supervised growing neural gas algorithm for cluster analysis. *Int. J. Hybrid Intell. Syst.*, 4(2):129–141.

[Kohavi and John, 1997] Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324.

[Kohonen, 1982] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

[Kohonen, 1988] Kohonen, T. (1988). *Self-organization and associative memory*. Springer series in information sciences. Springer-Verlag.

[Kohonen et al., 2001] Kohonen, T., Schroeder, M. R., and Huang, T. S., editors (2001). *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition.

[La Red Martínez et al., 2009] La Red Martínez, D. L., Acosta, J. C., Cutro, L. A., Uribe, V. E., and Rambo, A. R. (2009). Data warehouse y data mining aplicados al estudio del rendimiento académico y de perfiles de alumnos. In *XII Workshop de Investigadores en Ciencias de la Computación - CACIC 2010*, pages 162–166.

[Laboratories et al., 1960] Laboratories, S. U. S. E., Widrow, B., Hoff, E., of Naval Research, U. S. O., Corps, U. S. A. S., Force, U. S. A., and Navy, U. S. (1960). *Adaptive switching circuits*.

[Liu, 2011] Liu, B. (2011). *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Data-Centric Systems and Applications. Springer.

[Luo, 2008] Luo, Q. (2008). Advancing knowledge discovery and data mining. In *Knowledge Discovery and Data Mining, 2008. WKDD 2008. First International Workshop on*.

[MacQueen, 1967] MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

[Michalski and Larson, 1983] Michalski, R. S. and Larson, J. (1983). Incremental generation of vl1 hypotheses: the underlying methodology and the de-scription of program aq11. Technical report, Department of Computer Science, University of Illinois.

[Moody and Darken, 1989] Moody, J. and Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Comput.*, 1(2):281–294.

[Neukart et al., 2012] Neukart, F., Moraru, S.-A., Grigorescu, C.-M., and Szakacs-Simon, P. (2012). Transgenetic neuroevolution. In *Optimization of Electrical and Electronic Equipment (OPTIM), 2012 13th International Conference on*, pages 1120 –1125.

[Ngo et al., 2012] Ngo, L., Dantuluri, V., Stealey, M., Ahalt, S., and Apon, A. (2012). An architecture for mining and visualization of u.s. higher educational data. In *Proceedings of the 2012 Ninth International Conference on Information Technology - New Generations*, ITNG '12, pages 783–789, Washington, DC, USA. IEEE Computer Society.

[Pal, 2012] Pal, S. (2012). Mining educational data using classification to decrease dropout rate of students. *International Journal of multidisciplinary Sciences and Engineering*, 3(5):35–39.

[Pei et al., 1997] Pei, M., Goodman, E. D., and Punch, W. F. (1997). Feature extraction using genetic algorithms. In *Proceeding of International Symposium on Intelligent Data Engineering and Learning'98 (IDEAL?98), Hong Kong*, page 98.

[Quinlan, 1986] Quinlan, R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[Quinlan, 1993] Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

[Rajaraman and Ullman, 2011] Rajaraman, A. and Ullman, J. (2011). *Mining of Massive Datasets*. Mining of Massive Datasets. Cambridge University Press.

[RapidMiner, 2012] RapidMiner (2012). Rapid miner. `http://http://rapid-i.com/content/view/181/190`. [Ultimo acceso : 18-Nov-2012].

[Rodallegas et al., 2010] Rodallegas, E., Torres, A., Gaona, B., Gastelloú, E., Lezama, R., and Valero, S. (2010). Modelo predictivo para la determinación de causas de reprobación mediante minería de datos. In *II Conferencia Conjunta Iberoamericana sobre Tecnologías para el aprendizaje - CcITA 2010*, pages 48–55.

[Rosenblatt, 1962] Rosenblatt, F. (1962). *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books.

[Ruiz Sanchez, 2006] Ruiz Sanchez, R. (2006). *Heurísticas de selección de atributos para datos de gran dimensionalidad*. PhD thesis, Universidad de Sevilla, Sevilla, España.

[Shafti and Pérez, 2004] Shafti, L. S. and Pérez, E. (2004). Machine learning by multi-feature extraction using genetic algorithms. In *Advances in Artificial Intelligence - IBERAMIA 2004*, volume 3315 of *Lecture Notes in Computer Science*, pages 246–255. Springer Berlin Heidelberg.

[Tettamanzi et al., 2001] Tettamanzi, A., Tomassini, M., and Janßen, J. (2001). *Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems*. Springer.

[Tito and Mullicundo, 2010] Tito, L. and Mullicundo, F. (2010). Rapidminer. tutorial on-line + operadores. `http://es.scribd.com/doc/78886734/Rapid-Miner-Tutorial-Online-Ope-Rad-Ores`.

[Usama et al., 1996] Usama, F., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34.

[Valero and Salvador, 2009] Valero, S. and Salvador, A. (2009). Predicción de la deserción escolar usando técnicas de minería de datos. In *Simposio Internacional en Sistemas Telemáticos y Organizaciones Inteligentes SITOI 2009*, pages 332–340.

[Valero et al., 2010] Valero, S., Salvador, A., and García, M. (2010). Minería de datos: predicción de la deserción escolar mediante el algoritmo de árboles de decisión y el algoritmo de los k vecinos más cercanos. In *II Conferencia Conjunta Iberoamericana sobre Tecnologías para el aprendizaje - CcITA 2010*, pages 33–39.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical learning theory*. Adaptive and learning systems for signal processing, communications, and control. Wiley.

[Wang et al., 2012] Wang, J., Lu, Z., Wu, W., and Li, Y. (2012). The application of data mining technology based on teaching information. In *Computer Science Education (ICCSE), 2012 7th International Conference on*, pages 652 –657.

[Westphal and Teresa, 1998] Westphal, C. and Teresa, B. (1998). *Data Mining Solutions. Methods and Tools for Solving Real-World Problems*. John Wiley & Sons Inc.

[Winkler, 1972] Winkler, R. L. (1972). *An introduction to Bayesian inference and decision*. Holt, Rinehart and Winston, New York.

[Witten and Frank, 2011] Witten, I. H. and Frank, E. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, CA, 3th edition.